The Design of a Real Time, Fault-Tolerant, Multiprocessor System.

Volume 2 of 2.

Timothy Edwin Sharp.

Submitted for Ph.D. Degree.
The Computer Centre,
The University of Aston in Birmingham.
June, 1983.

List of Contents

Volume 2 - Appendices

List of Diagrams

Volume 2 - Appendices

List of Tables

Volume 2 - Appendices

Appendix 1

The Software and Firmware Development System

1.1 Introduction

It is required to write software in the form of high-level language programs, machine code and microcode. Before it can be written and tested there must be a satisfactory means of producing this code so that it can be executed on the target processor. It was anticipated that over 1K X 96 bit words of microcode alone would be produced. It was therefore clearly impossible to perform this process by hand and quite a sophisticated development system was needed. Such systems are commercially available but their average cost is between ten and twenty thousand pounds. Therefore, it was decided that the development system would need to be constructed using only the facilities already available within the University. The first part of this section describes a typical commercially-available development system. The second part of this section describes the actual facilities available, which became the building blocks of the development system used. The rest of this appendix then goes on to describe exactly how this was implemented.

1.1.1 A Conventional Microcode Development System - The AMD System 29

One of the companies specialising in bit-slice hardware is Advanced Micro Devices (AMD). Their microprogram development system, System 29, is one of the best and few complete microcode development aids available. It includes facilities for assembling and debugging microcode, debugging hardware and assembling machine code for the target processor.

An optimum System 29 configuration, including certain optional features, would consist of the central processor, a pair of floppy disc drives, a VDU, a printer, a PROM programmer, Writeable Control Store (WCS) for simulating microcode ROM, hardware for monitoring up to 64 test points on the target processor and also facilities for mapping the target processor's RAM onto System 29's RAM space.

A complete design scheme would be implemented in the following fashion (Fig. 1.1). The target processor would first be designed and built. Some extra components would be added to allow it to interface to System 29. This extra hardware would provide facilities for allowing the System 29 to disable the microcode memory of the target processor and allow all control signals to come from its own WCS. This would allow microcode to be quickly re-assembled and run without the need to program PROMs. There would also be a memory interface allowing the target processor's memory to be directly accessed as though it were part of System 29's RAM.

The next stage in the design would be to write and assemble the microcode. Because no two bit-slice machines are the same (the main

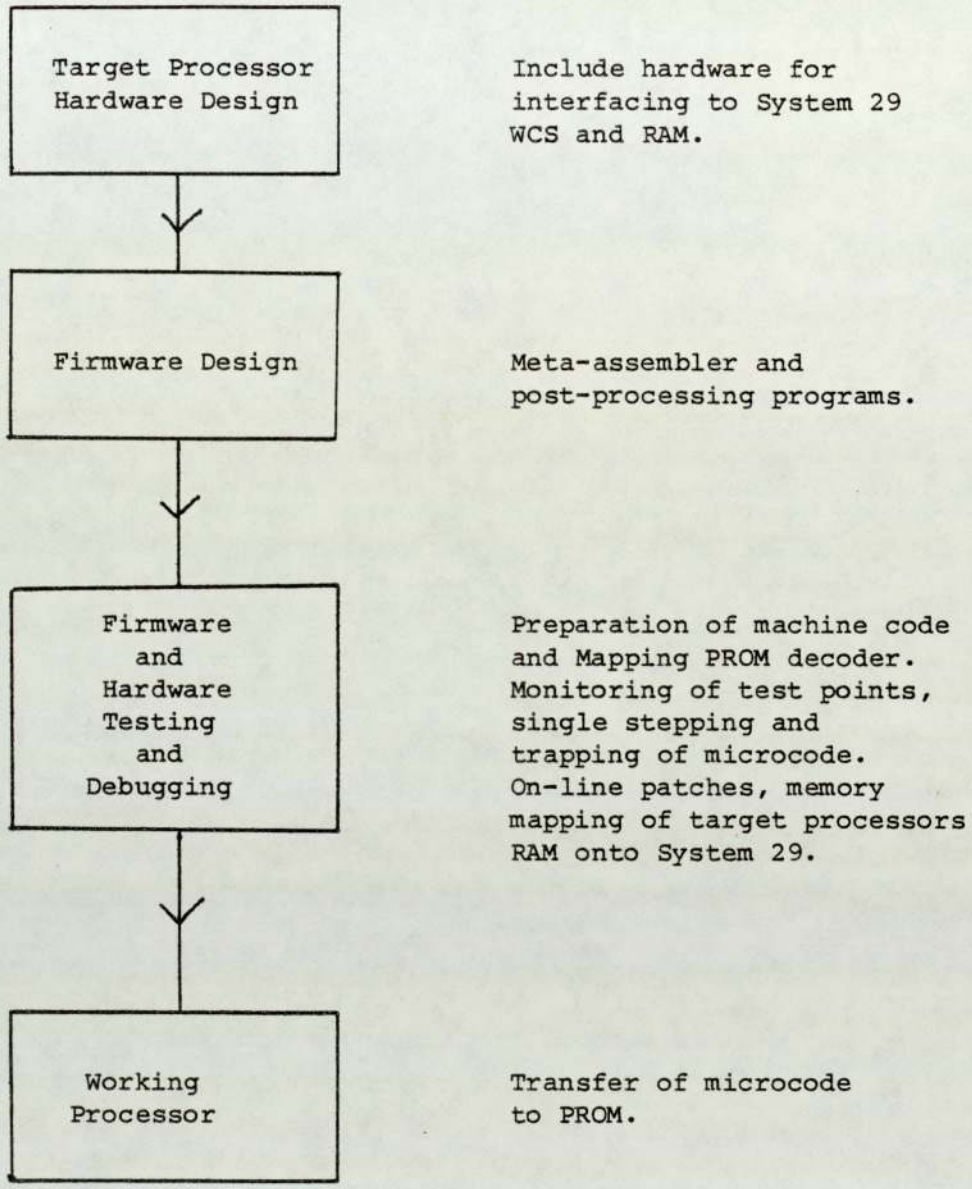


Fig. 1.1 A Conventional Firmware Development Scheme Using the AMD System 29.

purpose of their design is to obtain a processor with a different architecture) it is not possible to have a specific assembler for each target processor. Instead a meta-assembler is used. This is known as AMDASM on System 29. A meta-assembly consists of two stages. The first stage would be to assemble a definition file containing a description of the processor architecture. This file would contain a complete bit by bit specification of the microcode memory word format. That is, a description of each field within the microcode memory word together with symbolic equates for all its possible values. This file could then be used to assemble a microcode file which called these definitions. A sophisticated system like System 29 would also have what is known as a post-processing program. This would convert the logical bit patterns produced by the meta assembler into their physical form. This is to allow the design engineer to write microcode before the processor design is complete and to change the actual bit allocation in the microcode memory without altering the definition file.

The next stage would be to debug the microcode. Essentially, microcode treats machine code as data. In other words, it reads a machine-code instruction from memory ,decodes it and the appropriate piece of microcode is executed. Therefore, it is important to be able to produce machine code for the target processor. System 29 has such a capability. Machine code is assembled by using the System 29 machine-code assembler. This program has facilities for disabling its own op-codes to define new ones using macros. It is also necessary to define a Mapping PROM for converting machine instruction op-codes into microcode addresses. System 29 has a program for producing the contents of this PROM. There also exists a PROM programmer which is used for putting microcode onto PROMs

after it has been tested and a final, independent, working, target processor is required.

After successfully assembling the source code, the debugging of the microcode and the target processor would take place. For testing the hardware, System 29 has the ability to monitor 20 test points (i.e. "chip" pins) and display their bit values at each microcycle. A logic analyser option is available which can expand this capability to 64 test points. The microcode is stored in System 29's WCS rather than the target processor's ROM. The WCS can be up to 128 bits wide. The ability to manipulate the microcode memory means that code can be single stepped, traps can be set, online patches can be made and address jumps can be forced in order to execute microcode test routines at any point. Once a bug has been located the microcode can be quickly re-assembled and run. In addition, the RAM of the target processor can be accessed by System 29 using a memory-mapping technique. In other words the RAM of the target processor appears as the RAM of System 29. For example, memory location 0000 of the target processor could be accessed as memory location 8000 of System 29, loacation 0001 would be accessed as 8001 and so on. Finally, once the microcode had been completely tested it could be put onto PROM and the target processor would operate independently of System 29. This would complete the development of the firmware. One could then go on to write machine code programs for the target processor.

Therefore, it can be seen that the design, production and testing of both firmware and hardware requires a considerable amount of time and effort. Therefore, it is important, if no standard firmware development system is available, that the maximum use of

the computing facilities available must be made.

1.1.2 Facilities Available

The only readily available computing facilities were those within the department. These consisted of the following.

A PR1ME 250 minicomputer was in use within the department. This machine ran under the PRIMOS multi-user operating system and had 64 megabytes of disc storage. Although this was shared with other users there was no difficulty in storing large source files such as compilers and microcode. The PR1ME also had several high level languages available, including Algol 68C, which could be used for writing development software. However, more importantly, there was an assembler with good macro facilities which could be used to assemble microcode. This could be done using a technique pioneered by PR1ME to assemble their microcode.

The department also had several Motorola M6809 microcomputer systems available. These were used in two different ways. There existed a Motorola development system, the Exorciser, which had 56K of RAM and a dual floppy disc drive with 240 KB of storage. This development system had an assembler for producing M6809 code and also a PROM programmer which could be used to put microcode onto EPROMs. There also existed single board M6809 based systems which contained 8K of RAM and 16K of EPROM. These boards also contained several Peripheral Interface Adapters (PIA) which could be used to interface the boards with external devices or other processors.

There was also available a Tektronix Logic Analyser comprising of a Tektronix 7603 Oscilliscope, a Tektronix 7D01 Logic Analyser and a Tektronix DF1 Display Formatter. This device was capable of monitoring 16 test points and displaying 256 sequences of these values in a numerical format.

It was decided that the PR1ME 250 minicomputer could be used to compile Concurrent Pascal programs and also to develop the microcode. The code produced could then be transferred either to the Motorola Exorciser or to a single board system. From there it could either be transferred to EPROM or the RAM of the target processor (Fig. 1.2). The single board systems could be used either to monitor the Super Sixteen's memory or to actually emulate it. This facility of being able to examine the processor's memory, together with the logic analysers ability to monitor the processor's control and data flow would allow the microcode to be tested and debugged. Therefore, these facilities were to form the building blocks of the microcode development system. This will be described in more detail in the following sections.

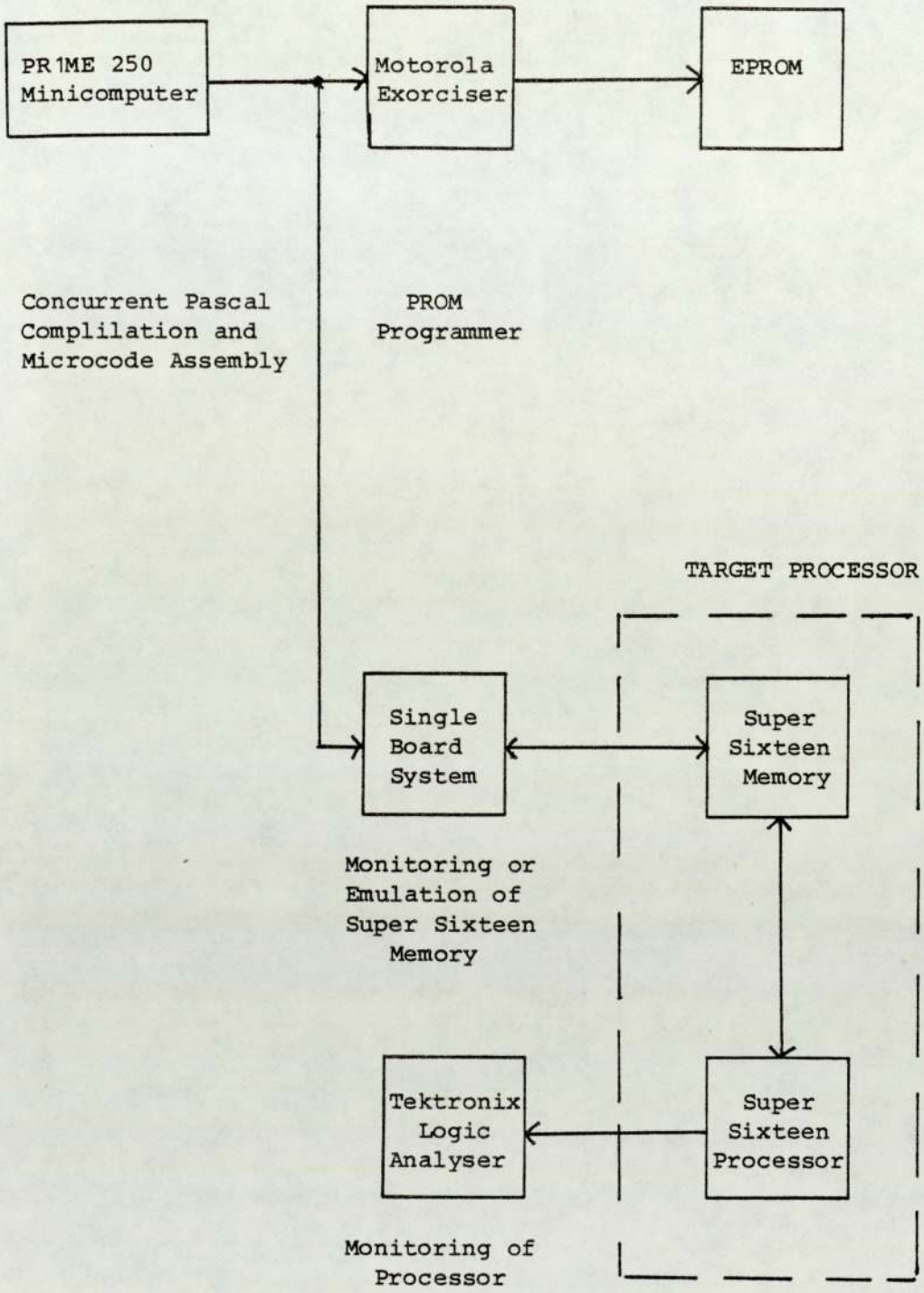


Fig. 1.2. A Microcode Development System Using The Facilities Available.

1.2 The Concurrent Pascal Development System

A Concurrent Pascal compiler had already been written at UCLA and a copy of it was obtained via UMIST and loaded up onto the PR1ME. This compiler had been designed to run under the SOLO operating system which was itself written in Concurrent Pascal and was designed to be portable. To achieve this the compiler produces Concurrent P-code, a hypothetical machine code. To transfer the SOLO operating system to another machine it is necessary to do two things. First, an interpreter for the P-code must be written. Second, a Kernel must be written to multiplex the processes in a Concurrent Pascal program between the single physical processor available.

The SOLO operating system itself was not actually required. Only the Concurrent and Sequential Pascal compilers, both of which had been written in Sequential Pascal, were needed. However, these programs did use a set of library routines (known as "prefix procedures") which were part of the SOLO operating system. Therefore, in order to run the compilers on the PR1ME, it was necessary to write a program which interpreted the P-code. This program also simulated the environment of the compiler by providing the library routines required. The P-code interpreter was written in Algol 68C and was designed to accept commands in a format similar to the SOLO operating system. The result of this was to have a Concurrent Pascal compiler running on the PR1ME in a similar manner to its original implementation. Since P-code is relocatable there is no need for a link loader and this is all that is required to produce executable programs. These could then be transferred to the Super Sixteen. The P-code interpreter is documented in Appendix 2

and source listings for it are given in Appendix 3.

As well as several utility programs which had to be written to transfer the compiler to the PR1ME and some interpreter test programs it was also necessary to write a set of utility programs to manipulate Concurrent Pascal source files. This was because the PR1ME source file format was slightly different to that necessary for the Concurrent Pascal compiler. Therefore, software had to be written to convert a PR1ME source file to a Concurrent Pascal source file and vice versa. A P-code disassembler was also written. This was necessary for debugging microcode on the Super Sixteen.

1.3 The Microcode Development System

1.3.1 Microcode Assembly

1.3.1.1 Using an Ordinary Machine Code Assembler to Assemble Microcode

The PR1ME Macro Assembler (PMA) was used to assemble microcode. The way in which this was done requires some explanation. As an example take the 'CONTROL' definition which enables certain control bits. The System 29 definition file defines it as :-

```
CONTROL    DEF 48X,1VB#0,1VB#1,1VB#1,1VB#1,44X
```

Which means that there are 4 variable 1 bit fields occupying the 48th, 49th, 50th and 51st bit positions and are 0, 1, 1 and 1 respectively by default. This could then be called in the

microprogram source file, for example :-

```
CONTROL ROM,INTDIS
```

Which would set the first bit field to the value of the 'ROM' equate (1), the second bit field to its default value of 1, the third bit field would be set to the value of the 'INTDIS' equate (0) and the last bit field would be set to its default value of 1.

To obtain a similar type of format of assembly a method was used which was similar to that used by PR1ME to assemble their microcode. PMA has a macro facility which allows text substitution, equates for giving values symbolic names and a message output facility which allows the user to output his own messages in the assembly listing.

Six equates were used to represent a microcode memory word. They were named W1 to W6. W1 holds the first 16 bits, W2 holds the next 16 bits and so on. At the start of each microinstruction a macro called 'INST' is called which sets these six equates to zero. To implement the 'CONTROL' macro it is necessary to set the first 4 bits of W4, but first the operand equates must be defined. The operand names are prefixed with 'O$' if their default value is 1 and 'A$' if their default value is zero. O$ and A$ are equates which have been set to 1 and 0 respectively. The macro definition file would be as follows :-

```
CONTROL    MAC
W4         XSET W4 .AND. $0FFF .OR. $7000
W4         XSET W4 .OR. (A$<1> .LS. 15)
W4         XSET W4 .AND. (O$<2> .LS. 14 .OR. $BFFF)
```

```
W4          XSET W4 .AND. (O$<3> .LS. 13 .OR. $DFFF)
W4          XSET W4 .AND. (O$<4> .LS. 12 .OR. $EFFF)
            ENDM
```

The first 'XSET' sets the first 4 bits of W4 to zero by ANDing it with $0FFF and also the default values by ORing it with $7000 which leaves the last 12 bits unaltered. The next 'XSET' sets the first bit of W4 by ORing it with the macro's first parameter shifted to the relevant bit position. Note that if the first parameter is null then the value 'A$' is used, i.e. the bit is set to zero by default. The next 'XSET' sets the second bit of W4. This is achieved by shifting the second parameter of the macro call to the current bit position and forcing all the other bits to 1 by ORing with $BFFF. This is in case an illegal value other than 0 or 1 has been used. The value obtained is then ANDed with W4 to set the bit appropriately and leave all the other bits unaffected. The third and fourth bits are set in a similar manner.

Finally, at the end of each microinstruction, the 'ENDINS' macro is called. This writes the current microword to the binary disc output file using the 'DATA' directive :-

```
DATA W1,W2,W3,W4,W5,W6
```

This then creates 6 new words in the output file containing the values W1, W2, W3, W4,W5 and W6. At this point the current address and contents of the micromemory word are output to the assembly listing file. The 'OUTPUT' macro is used for this. To output the contents of ADDRESS, say, the OUTPUT macro would contain the following code :-

```
SAY <(ADDRESS .RS. 12) + 1><((ADDRESS .RS. 8) .AND. $F) + 1>;
    <((ADDRESS .RS. 4) .AND. $F) + 1><(ADDRESS .AND. $F) + 1>
```

Where (ADDRESS .RS. 12) + 1

delivers the m.s. nibble of ADDRESS plus one.

and ((ADDRESS .RS. 8) .AND. $F) + 1

delivers the second m.s. nibble of ADDRESS plus one.

etc.

These values are calculated and since they are in angled brackets (<>) the result delivers the parameter number of the macro. The macro is called with a lookup table of hex characters as parameters :-

```
OUTPUT 0 1 2 3 4 5 6 7 8 9 A B C D E F
```

So, the text substituted for that parameter is the actual hex character representation of the relevant nibble. For example, suppose 'ADDRESS' contained the value $F034, the values inside the angled brackets would be :-

```
SAY <16><1><4><5>
```

Which, after parameter substitution, would become :-

```
SAY F034
```

Which would output "F034" to the assembly listing file. The contents of each byte of microcode is displayed in a similar manner.

This method of macro substitution allowed some useful modifications to the microcode structure to be made as the project proceeded. One of these proved extremely useful from a fault-tolerance point of view. Under certain conditions, for example if the Mapping PROM Decoder was faulty, the Am2910 sequencer would generate $00FF as the next micromemory address. The reasons for this are explained in Chapter 5. Therefore, it was necessary to have a jump at address $00FF to a section of microcode dealing with the error. However, every time that new microcode was inserted (or deleted) before address $00FF it moved the "jump to error code" instruction away from $00FF. This problem was overcome by altering the macro definition file so that the ENDINS macro automatically checked the current address to see if it was $00FE. At this address it inserted the microcode to jump to address $0100, at $00FF it inserted a jump to the error code and at $0100 it placed the code which should have been at address $00FE. If the code at $00FE was already a jump instruction then it was not necessary for the code to be moved.

In other words the macro assembler could be directed to implicitly insert an instruction at a certain address. This allowed code in previous microprogram memory locations to be inserted or deleted without affecting that location.

1.3.1.2 Producing a Loadable File

As more and more microcode was written it was found that the time taken to assemble it was becoming unacceptable. It was therefore decided to use the existing link loader on the PR1ME to allow the source microcode to be spread across more than one file. However, there was a slight problem here. PMA allowed the use of external references to addresses but the Super Sixteen microcode memory was 96 bits wide and a word of PR1ME memory was only 16 bits wide. This meant that a Super Sixteen micromemory word was stored in 6 PR1ME words. Therefore, any address references that PMA encountered had to be divided by 6 (Fig. 1.3). Unfortunately, an external value cannot be divided by 6. Since it has not yet been defined, arithmetic operations on it cannot take place. This problem was overcome by putting all addresses into the object file as they occured without dividing them. After they were link loaded another program (written in Algol 68C) was run. This divided the address fields of the loaded file by 6 to form the true addresses. The operation of this program was complicated by the fact that the address field of a microinstruction was also used for other purposes (microcode formatting). It might be used as the Immediate Field or the Load Interrupt Counter value or as a value for loading into the Am2910 counter. This meant that the program had to check for this before modifying the address value.

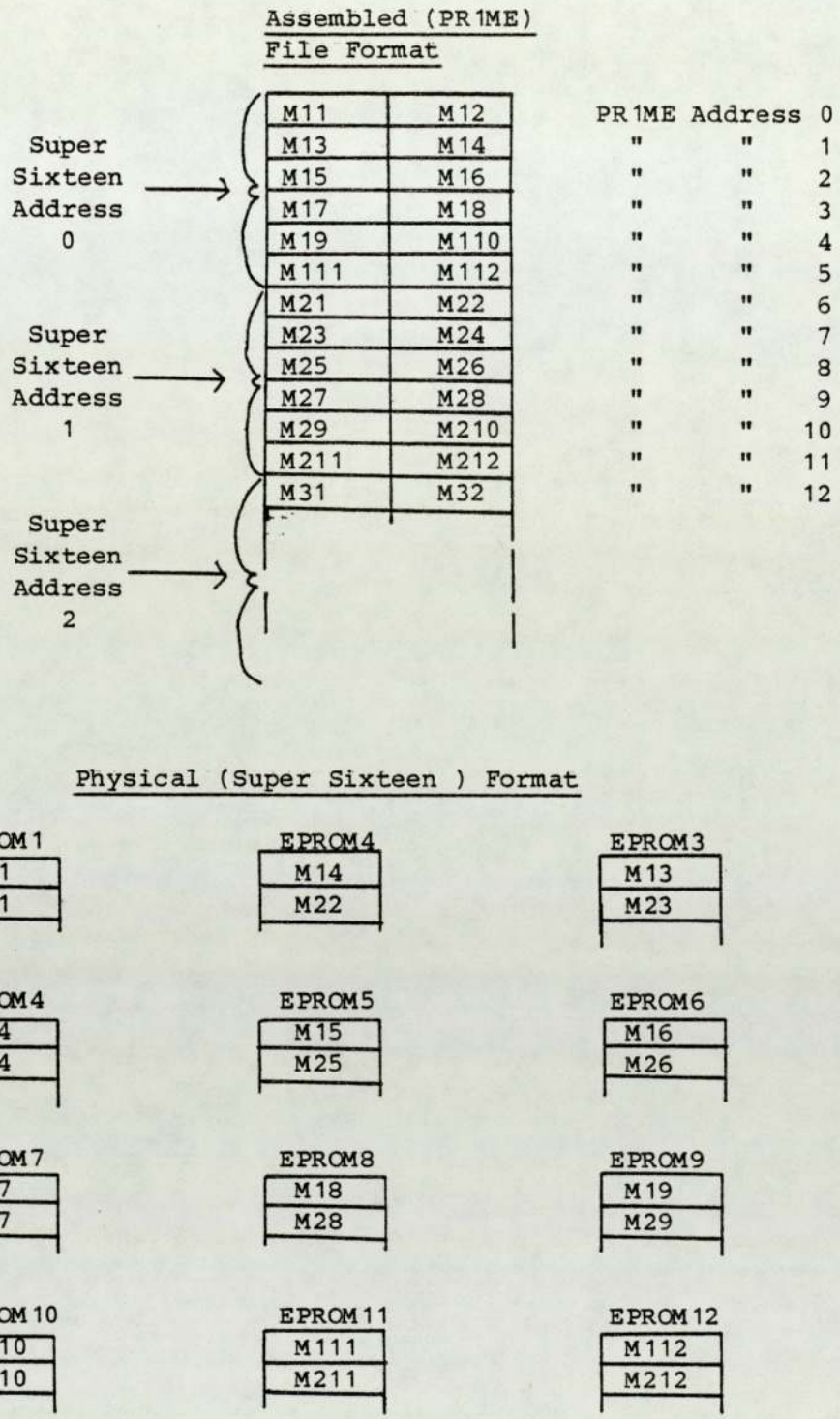
Therefore, it was necessary to modify the PR1ME link loading system by writing an Algol 68C program in order to split the source microcode into manageable parts.

The second problem that needed to be overcome in order to

produce microcode in an executable form was caused because the structure of the binary microcode on the PR1ME was different to its structure on the Super Sixteen. The PR1ME contains the assembled and loaded microcode in its logical format. That is, each microinstruction, consisting of 96 bits, appears one after the other. However, the actual physical format that the microcode will take is in contiguous 8 bit bytes on an EPROM as shown in Fig. 1.3. The easiest place to divide up the microcode was on the PR1ME, rather than the Motorola, where software development is the most convenient. The code is actually divided up and transferred at the same time. In other words when the PR1ME transfers the code to the Motorola it first sends across the contents of EPROM1, then the contents of EPROM2 and so on.

1.3.2 Producing a Mapping PROM

The Am2910 microcode sequencer on the Super Sixteen uses a mapping PROM (or Decoder) to convert a machine code instruction into the address of the microcode that executes it. On the Super Sixteen the Decoder consists of a 2K X 8-bit EPROM. The address lines are connected to the op-code of the machine code instruction and the data lines produce the microcode address. During the firmware development stage the microcode addresses are constantly altering as new microcode is being added and deleted. This means that the contents of the Mapping PROM must change accordingly. The addresses could be calculated manually but as there are 112 P-code instructions this becomes quite a laborious process. It would also mean obtaining a new assembly listing each time the microcode is altered which can also be time consuming and wasteful.



MXY = Microinstruction X, byte Y.

Fig. 1.3. Logical and Physical Microcode Format.

It was therefore decided to write a program which took the microcode source file and produced a binary file containing the Mapping PROM addresses. Apart from the time factor for the manual method, it was reasoned that the time taken in writing the program (about one and a half days) would be well spent as it would ensure a much higher probability of the contents of the Mapping PROM being correct.

The program contains a lookup table of P-code labels. It scans the source microcode maintaining an address count each time a new microinstruction is encountered. Whenever a valid P-code marker label is found its address is noted and put in the output file at the address corresponding to its pre-defined op-code. It is also necessary that all the microcode start addresses must appear within the first 256 words of microcode memory. This is because the Mapping PROM is only 8 bits wide. The program checks this and outputs an error message if any of the addresses overflow.

1.4 The PR1ME-AMD Interface

1.4.1 The Motorola M6809 Microprocessor Board

The Motorola M6809 based single board systems mentioned earlier were used as a "staging post" between the PR1ME and the Super Sixteen. These boards included 8K of RAM which could be used for storing any code transferred from the PR1ME before it was moved either to EPROM or the RAM of the Super Sixteen. However, in the case of Intelligent Memory, the RAM of the Motorola board is also the RAM of the Super Sixteen. This concept is discussed more fully

in the next section.

The boards could also contain up to 16K of EPROM. During the testing of the microcode it was quite common to alter the microcode, but the Concurrent P-code changed less frequently. It was therefore useful to be able to store code semi-permanently on the Motorola rather than the PR1ME. Since the Motorola software only took up 2K of EPROM there was 14K left for this purpose.

The boards also contained several memory-mapped I/O devices. The first of these was an Asynchronous Communications Interface Adapter (ACIA). This device allowed the Motorola board to communicate with a VDU. This meant that commands typed in at a terminal could be used to manipulate the code between the PR1ME, the Motorola and the Super Sixteen.

The boards contained 5 PIAs. These devices could be used for two purposes. They could drive a PROM programmer and thus, with the appropriate software, code could be transferred from the Motorola RAM to EPROMs. The PIAs could also be used for interfacing the Motorola board to the Super Sixteen. By connecting the PIA outputs to the Address and Data Buses and Control Signals of the Super Sixteen a level of control over it could be obtained. Also, the Super Sixteen memory could be accessed.

1.4.2 Transfer from the Prime to the Super Sixteen

The technique for transferring any type of code or Mapping PROM data to the Super Sixteen was to first transfer it to the Motorola. The method of achieving this was crude but simple. The PR1ME and the Motorola were connected by a simple switch box and the PR1ME sent hexadecimal characters to the Motorola as if it were a terminal. The Motorola receives these characters as if they were coming from a terminal.

On the PR1ME two pieces of software were required, one to send a file exactly as it appears in the filestore and another to divide the file up before sending it as mentioned in section 1.3.2. The former program was used for transferring P-code and Mapping PROM files and the latter was used for transferring microcode.

The Motorola required software for receiving this data and storing it in RAM. Software for driving a PROM programmer was also required but this was already in existence. The memory board in the original Super Sixteen design included hardware for interfacing it to System 29. It was found that with minimal modification this interface could be directly connected to the Motorola via PIAs. Software was written to transfer data between the Super Sixteen memory and the Motorola in both directions. However, hardware faults in the Super Sixteen memory board soon made this system unworkable and the concept of Intelligent Memory was evolved.

1.5 Testing the Microcode

1.5.1 Intelligent Memory

When hardware problems developed with the Super Sixteen memory it was decided that it would be desirable if some hardware could be used which was already known to be reliable. It was decided to use the Motorola M6809 single board systems. The idea was to connect the PIAs to the address, data and memory control buses of the Super Sixteen. The Motorola would scan the PIAs and wait for a memory request signal. The memory address required would be read from the PIAs and the M6809 processor would perform a read or write as requested to its own memory. In other words, the Super Sixteen memory and the Motorola memory are one and the same. In order to access its memory the Super Sixteen must send a memory request via the PIAs and the M6809 processor. This is known as Intelligent Memory. Although the Motorola memory is only 8 bits wide and the Super Sixteen memory is 16 bits wide the M6809 processor disguises this fact.

The Motorola single board systems were reliable and no additional hardware, only software, needed to be developed. Also, the Motorola boards gave the Super Sixteen access to RAM, EPROM, an ACIA for linking up to a terminal, and PIAs for communicating with another Super Sixteen processor.

There are several Motorola clock cycles required for processing each Super Sixteen memory request. This means that the Intelligent Memory requires the Super Sixteen to operate at a much slower speed than its maximum, in fact about 800 times slower. However, this was

partially offset by the fact that the programmer had complete access to the Super Sixteen memory via the Motorola. Also, speed was not paricularly important during the development stage. It would also have been possible with the appropriate software to monitor all Super Sixteen memory accesses. This would have been done if debugging of the microcode proved to be extremely difficult. However, in the event, this option was not required.

When the microcode development was completed the Intelligent Memory could be replaced with a plug-compatible hardware-memory. This would require almost no modification to the firmware or the Super Sixteen hardware. This hardware board could be based on two 8-bit Motorola boards cascaded together to form a 16-bit memory. These boards would have no processor and would be directly under the control of the Super Sixteen processor. This is mentioned in slightly more detail in Chapter 3. The software listings for the Intelligent Memory appear in Appendix 5 and the design for the hardware memory board appears in Appendix 6.

1.5.2 The Logic Analyser

The only other firmware debugging aid used was the Tektronix Logic Analyser memtioned in section 1.1.2. The method used to debug the microcode was simple but tedious. It was found that by monitoring four sets of outputs on the Super Sixteen that any section of microcode could be tested and any errors located. These four outputs were the Memory Address Register (MAR), the Z-Register (ZREG) through which data read from memory passes, the D-Register (DREG) through which data written to the memory passes and the YBUS on which the ALU outputs appear.

The Logic Analyser had only 16 lines for monitoring test points. It was necessary to use 8 bits for monitoring the Am2910 sequencer (i.e. the microcode memory address). The remaining 8 bits were used to monitor other outputs. Since each of of the four sets of test outputs were 16 bits wide it took at least 8 runs to check any particular section of microcode, hence the tedium.

1.6 Disadvantages of this system vs. a conventional system

The major disadvantage of the microcode development system which was constructed was that it was extremely slow. The Concurrent Pascal compiler was driven by an interpreter which was inevitably found to be slow. The fact that it was written in Algol 68C made this worse. As far as the microcode is concerned it was necssary to assemble the macro definition file every time that a source file was assembled. A large amount of text substitution was involved which slowed down the assembly process considerably. In practice it proved impractical to split the microcode into more than about three files. Each of these were at least 5000 lines long (about 400 microinstructions) and took 30 minutes to assemble. After the microcode had been assembled it had to be transferred to the Motorola and then put on an EPROM. It normally took over an hour for the complete process. However, it must be said that minor patches could quite often be made which took considerably less time. Since there were so many stages involved it was easy to make a mistake which could prove to be very costly in terms of time and effort.

Apart from the speed factor there were some other disadvantages of this system compared with, say, System 29. It was not really possible to single step or set traps and breakpoints in the microcode. It was only possible to monitor the microcode execution on the Logic Analyser and examine the contents of memory after the execution was complete. Furthermore, it was not possible to perform on-line patches since the microcode was stored on EPROM. This meant that whenever an error was discovered it had to be painstakingly corrected before further debugging could take place.

Another minor disadvantage was the fact that System 29 could be adapted for any target processor. However, this development system was written with one particular application in mind. Certain pieces of software such as the Mapping PROM generator and the program for splitting up the microcode from its logical to its physical format were written in the shortest possible time and were applicable only to the Super Sixteen. However, it would take little effort to re-write this software to make it general purpose.

One advantage to this system was that the Intelligent Memory board required no extra hardware to interface it with the Super Sixteen. Also, further memory diagnostics could have been programmed into the Intelligent Memory should it have been required. Another minor advantage was the ability to implicitly insert instructions at pre-defined locations as described in Section 1.3.1.1.

However, the system constructed served to develop all the microcode necessary for this project. When viewed from that aspect it could be deemed to have been successful, although slow and tedious.

Appendix 2

The P-code Interpreter

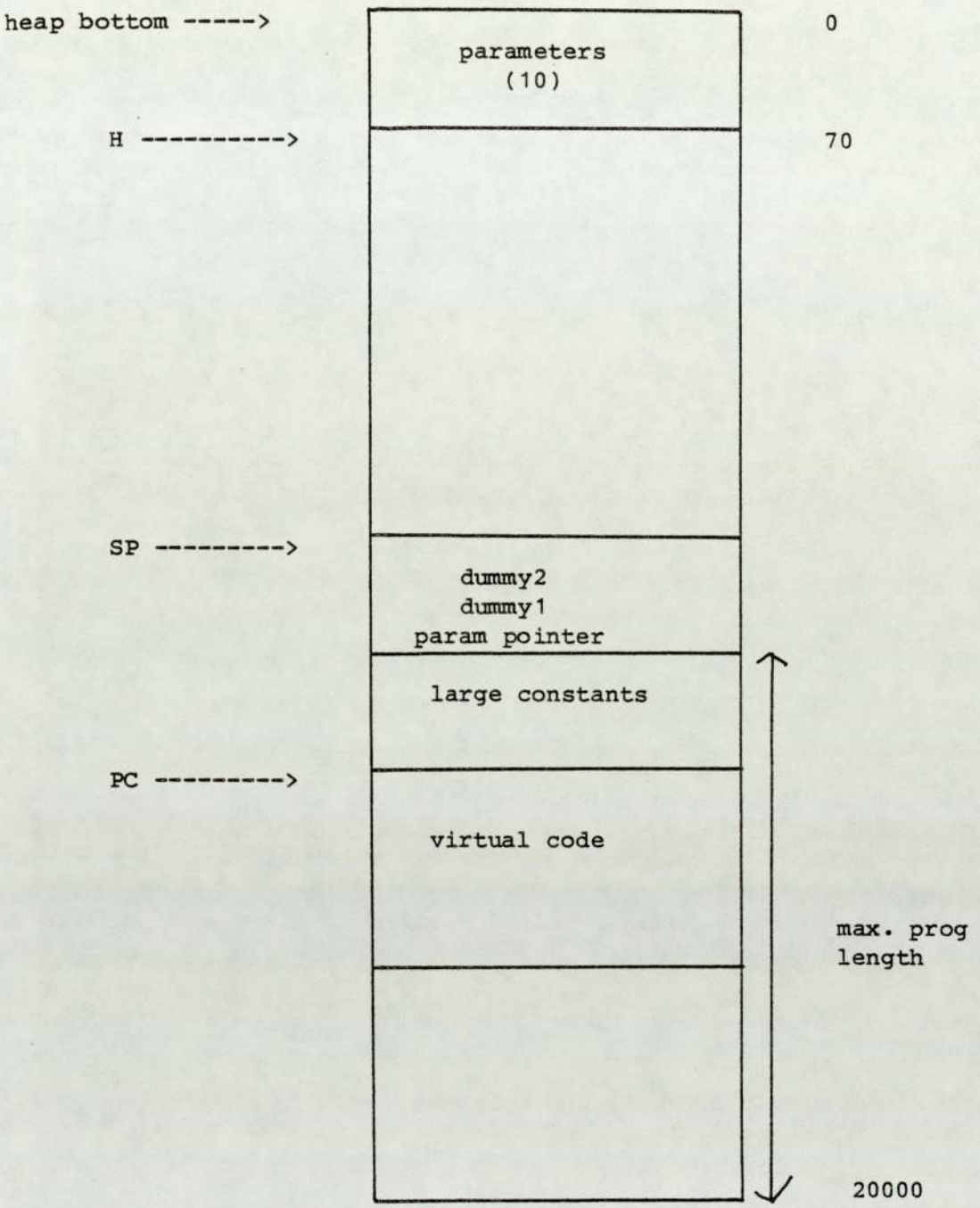
2.1) Introduction

This Appendix describes the P-code interpreter which runs on the PR1ME 250. The interpreter was written in order to run the Concurrent and Sequential Pascal compilers which are themselves written in Sequential Pascal and had originally been implemented on a PDP 11/45. There are three main sections to this interpreter; initialisation, environment simulation and the actual code interpretation. The interpreter is written in Algol 68C.

2.2) Initialisation

This part of the interpreter consists of inputting the program and its parameters and setting up the program registers and the stack. The largest program that will be run is Pass 3 of the compiler. This requires 20,000 words of core store which is simulated by declaring an array called "store". The initial memory structure is best explained diagramatically, as in Fig. 2.1.

The virtual code is stored at the bottom end of the array and the heap and stack are stored at the top end. This is because of the addressing mechanism of the P-code which is by bytes, this means that consecutive words are addressed 0,2,4....and so on. The maximum address of a word that can be held in Algol 68C is 16683 ((32767 - 1)/2). However, all registers store the word address i.e.



max prog length - The length of the largest program that can be run.
dummy1 - The Prefix routines base address,not required.
dummy2 - The Program return Adress,not required, since this is the root program and there is nowhere to return to.
param pointer - The address of the parameter list, stored in the heap.

Fig. 2.1. Initial Memory Structure.

0,1,2....etc. This means that the virtual code, which is accessed using the program counter register, can be addressed anywhere in "store". The data areas (i.e. The heap and the stack) can only be accessed if they lie in the first 16K of "store".

The user types in a command line which is read into the string "command line". The first argument, i.e.the program name, is extracted and put into the CHAR array "argument" by the procedure "read argument" which has a []CHAR parameter called "endchar". This contains a list of all valid termination characters for the argument. In the case of the program name they are "(" and newline.

i.e. SPASCL(param1,param2,.....)

or SPASCL

What "readchar" does is to read in all alphanumeric characters, ignore any spaces, and stop when any other character is encountered. If this is not a valid termination character then it returns a value of FALSE, otherwise it returns a value of TRUE.

If the program name is valid then an attempt is made to read in the program parameters, if there are any (all Pascal programs can have parameters,just like a procedure). This is done, again, using "readchar", valid termination characters are either "," or ")", i.e. a parameter is either followed by another parameter or a closing bracket. If the terminating character is ")" then an exit is made from the loop. A count of the number of parameters processed is kept in "params". A program can have up to a maximum of 10 parameters. If less than 10 have been input then all the other

parameters are set to NILTYPE. The format of the parameter list (on the heap) is shown in Fig. 2.2 · for an example program with an IDENTIFIER parameter "ABCD", a BOOLEAN parameter TRUE and an INTEGER parameter 29.

i.e. EXAMPL(ABCD,TRUE,29)

Note that the first parameter, which is always BOOLEAN, is inacessible to the user.

If all the parameters are syntactically correct then an attempt is made to read in the program. This is done using the procedure "input program" which has a STRING parameter, "pname", which holds the program name. The first 4 words of the program file are read in first. The first word contains the total program length and the second word contains the code length. After this information has been obtained the virtual code and the large constants can be read in. The program counter "pc" is made to point to the start of the virtual code, "pcdisplacement" is also set to this value. This means that if an error or a program exception occurs during the interpretation of the virtual code then the value "pc - pcdisplacement" can be printed out. This will give the displacement from the start of the virtual code rather than the actual program counter value which would be meaningless to the Pascal programmer.

Note also that if the user types in "QUIT" as the command line then execution of the interpreter ceases. This is done by checking the program name (if there are no parameters) to see if it is "QUIT". If it is then the Algol 68C procedure "stop" is called.

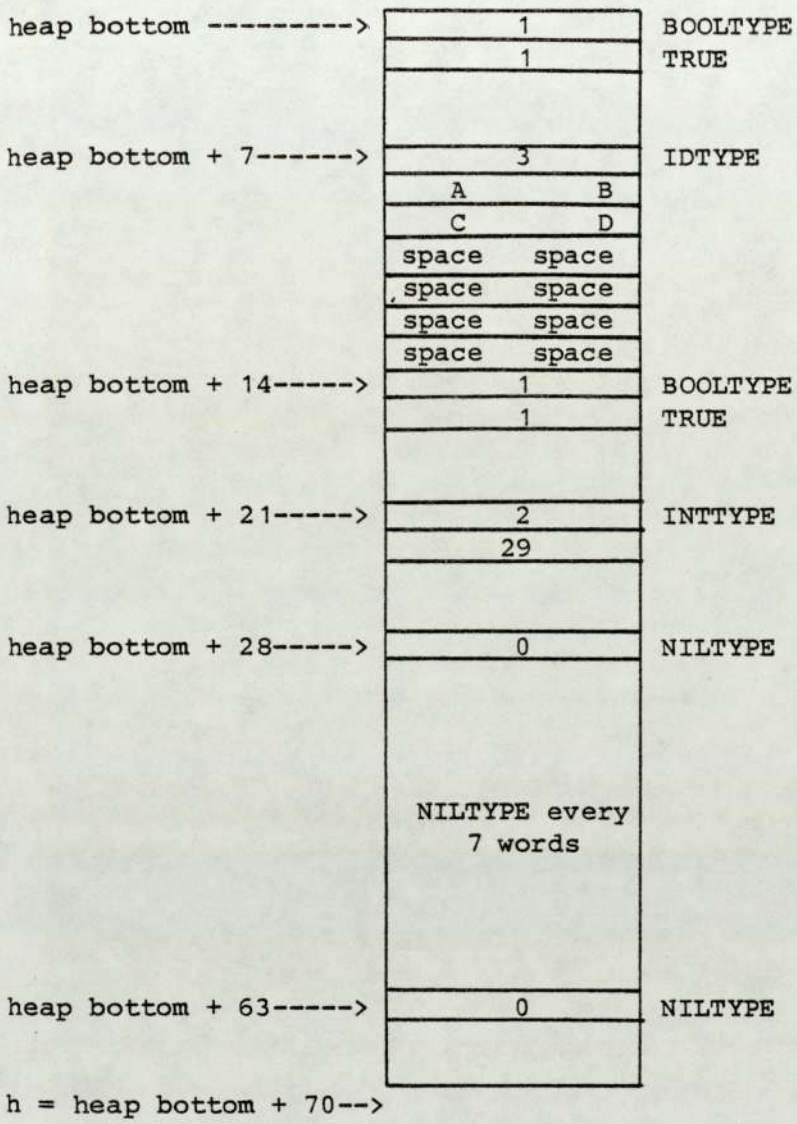


Fig 2.2. Initial Parameter Structure.

2.3 Environment Simulation

This part of the interpreter is needed because certain routines used in Sequential Pascal programs are implemented within the operating system. Since the operating system is not used they must be implemented within the interpreter. These routines include such facilities as opening, reading, writing to and closing disc files, calling other Pascal programs and input and output to and from the teletype.

These routines are called using a "system call" P-CODE instruction, SCL. it has a non-negative even index (0,2,4 etc.) which indicates which routine is to be called. This is implemented as a CASE statement in Algol 68C.

A Pascal program can have up to 4 files open at any one time, an input source file, an output source file and two random access binary files which can be written to or read from. The individual procedures are as follows :-

READ

Reads a character from the current source input file. A buffer of 256 words at a time is read in from the disc and a pointer to this buffer is maintained. When the buffer is empty another disc access is made.

WRITE

Writes a character to the current source output file. Its implementation is similar to that of READ.

OPEN

Opens a random access binary file. If the Pascal FILE parameter is 1 it is opened on unit 13, if the FILE parameter is 2 it is opened on unit 12. If a file is already opened on that unit then it is closed first. If the opening is not succesful then the Pascal FOUND parameter is set to FALSE.

CLOSE

Closes a random access binary file. If no file is opened then an error occurs and control is returned to the console.

GET

Reads a block of 256 words from a random access binary file. This is done by rewinding the file, skipping a number of blocks defined by the Pascal parameter P and reading in a block. Note that use is made of the Fortran "prwfil" procedure, rather than the standard Algol 68C "prwfil" procedure. This is because the Fortran procedure has a fifth parameter defining the number of words to be skipped before reading from the disc. It is renamed "prwfile" to avoid confusing it with the standard Algol 68C procedure.

PUT

Writes a block of 256 words to a random access binary file. Its implementation is similar to that of GET.

LENGTH

Returns the number of 256 word blocks in a random access binary file. This is done by reading in the entire file, 256 words at a time, and counting the number of disc accesses made. Since this method is very slow and messy a check is made to see if the filename

is either "TEMP1", "TEMP2" or "NEXT". This is done because these are the only three files on which the Pascal compilers use the LENGTH function. If the filename is one of these then a value of 255 is automatically returned without making any disc accesses.

MARK,RELEASE

Respectively gives the value of or sets the heap top register "h".

IDENTIFY

Sets up a string for the current program which is output to the teletype the first time that any output is attempted. It is also printed out if the program has been interrupted and subsequently attempts to output to the teletype. A program could be interrupted by calling another program, for example.

ACCEPT

Inputs a character from the terminal. At the start of each new line a "P" is output as a prompt. The procedure "tty read ch" is used to read in a character since this will detect a newline character, whereas the standard procedure "read" will not. The standard "tty erase char" is still used as the erase character though.

DISPLAY

Outputs a character to the terminal. If it is the first time that any output has been made then the identification string set up by IDENTIFY is output as well.

READPAGE

Reads a block of 256 words from the current source file, which will be located on logical file unit 15.

WRITEPAGE

Writes a block of 256 words to the current source output file, which will be located on logical file unit 14.

READLINE

This procedure is similar to READ. However, instead of reading one character from the current source file, it reads a line. A line consists of up to 132 characters, terminated by a newline character.

WRITELINE

Writes a line to the current source output file. Its implementation is similar to that of READLINE.

READARG

Closes the current input or output source file. If the TAG field of the Pascal ARG parameter is INT and the source file is an output file then ARG is set to an integer value defining the size of the output file (in 256 words blocks). Otherwise, it is set to a BOOLEAN value which is TRUE if there have been no transmission errors and FALSE otherwise. Also, if the source file being closed is an output file then the contents of the output buffer is written to disc.

WRITEARG

Opens a source file for input or output. The Pascal ARG parameter contains the file name, the Pascal ARGSEQ parameter is set to 0 for input and 1 for output. The buffer pointers used by READ and READLINE for input and WRITE and WRITELINE for output are initialsed also.

LOOKUP

Checks for the existence of a file. The Pascal parameter FOUND is set to TRUE if the file exists and to FALSE otherwise. The file attributes are also set to ascii and unprotected, thus missing out a lot of the SOLO file management system.

IOTRANSFER,IOMOVE

These two procedures correspond to the very lowest level I/O operations of the P-CODE machine, hence they are not implemented.

RUN

This procedure calls, and runs, a Pascal program. Its operation is essentially that of performing an overlay. Some Pascal programs have been classed as special cases since they can't run without the complete SOLO environment to support them. At present there is only one of these special case programs needed by the compiler, its name is FILE.

The FILE program is called by the compiler to copy LENGTH words from the file NEXT to another file. In effect NEXT is used as a temporary file and this is the method of transferring it to a permanent file. This special case program is implemented entirely by the interpreter without calling any other programs.

If the Pascal program to be run is not a special case then the following procedure is adopted. First, the stack is rearranged. The addresses of the LINE and RESULT variables are already on the stack since they are parameters of the Pascal RUN procedure. They are rearranged in the order ARGLIST address, LINE address and RESULT address. Then, a dummy word is placed on the stack. This corresponds to the prefix routine base address which is not used. The return address of the current program is also put on the stack. Finally, the new program is input and the program counter is made to point to word 0 of the virtual code, this is done by calling the procedure "input program".

2.4 Code Interpretation

The basic format of this section of the program is as follows.

```
DO
   initialise variables;
   WHILE input parameters; parameters not ok
   DO SKIP OD;
   input program;
   WHILE execution
   DO
      CASE instruction
      IN
         list of case clauses, each corresponding to a P-CODE
         instruction
      OUT error("unknown instruction")
      ESAC
   OD
OD;
```

Notice that there is an infinite loop. This means that as soon as one Pascal program has run another can be run without having to call the interpreter again. Within this loop the following

procedure is adopted. First, any variables which need to be initialised are set to the appropriate values. Then the parameters and the Pascal program are input, this is done inside a loop and will be repeated until there are no errors. Next, the program is input from disc, if there have been any errors then the boolean variable "execution" will be set to FALSE. Since the main interpretation loop is controlled by this variable it will not be entered if errors have occured during the execution of the "input program" procedure.

The actual CASE statement is not quite as shown above since there are too many P-CODE instructions to fit in one Algol 68C segment. To overcome this difficulty the first 64 P-CODE instructions are interpreted in one segment and the OUT clause of the CASE statement is an ENVIRON statement. If the instruction is not one of the first 64 then the segment specified by the ENVIRON statement is entered, this segment contains another CASE clause of the same format dealing with the remaining P-code instructions.

Most of the instructions need no further description, other than that in Appendix 7. However, there now follows a few comments on some of the instructions whose implementation differs significantly from that of the original PDP 11/45 interpreter.

1) Note that all the real number instructions assume a double length floating point format, using three words for the mantissa and one for the exponent. It is also assumed that the exponent is stored in Excess 128 form. This corresponds exactly to PR1ME double-length floating-point format, but not to the PDP 11/45 format. This is no problem since there are no real constants used in any of the

compiler passes. The "LARGEST_REAL" procedure in Pass 1 of the compiler was easily altered by making a binary patch. The statement:-

```
MAX (.4.) := MAX (.3.)
```

was changed to :-

```
MAX (.4.) := MAX (.1.)
```

It is necessary to use assembler language routines to implement these instructions. The Algol 68 program will contain, for each real number P-code instruction, a procedure whose body consists of a single CODE statement jumping to the assembler language routine. Using an Algol 68 procedure is a convenient (and necessary) method of putting the Pascal stack addresses on the Algol 68 stack when the procedure is called. Most of the assembly language routines consist of no more than 5 instructions. As an example, consider the ADD REAL P-code instruction, which adds the top two numbers on the stack.

The Algol 68 code would appear as follows :-

```
PROC add real = (REF INT addr1,addr2) VOID:
CODE "MADREAL" EDOC;
addreal (store [sp],store [sp + 4]);
```

Calling the procedure "addreal" will enter the assembly language routine "ADREAL". Furthermore, the third word after the external variable R5 will contain the address of the top stack word and the fourth word after R5 will contain the address of the top but fourth stack word. The assembler code looks like this :-

```
SUBR ADREAL
```

```
ADREAL DAC **        RETURN ADDRESS
    LDX R5
    DFLD 4,1*        REGISTER = REAL2
    DFAD 3,1*        REGISTER = REAL2 + REAL1
    DFST 4,1*        REAL2 = REAL2 + REAL1
    JMP* ADREAL      RETURN TO A68 CODE
```

Finally, the Algol 68 code pops 4 words off the top of the stack:-

```
sp +:= 4;
```

2) All registers (other than scratch registers) hold the array index of "store", i.e. half the actual address. When an address is put on the stack it is stored as its full value, i.e. twice the array index value. Hence, when using push instructions which use addresses relative to a register, such as PUSH GLOBAL ADDRESS, the value in the "g" register has to be multiplied by 2.

3) The COMPARE STRUCTURE instructions all share a common procedure called "check struct". This procedure compares the two structures, word by word, to see if they are equal. For the COMPARE STRUCTURE EQUAL and COMPARE STRUCTURE NOT EQUAL instructions nothing remains except to set the stack top. For the other 4 COMPARE STRUCTURE instructions the last word tested by "check struct" must also be examined. The stack top is then set according to the result of this comparison.

4) The INITIALISE POINTER VARIABLE and INITIALISE POINTER VARIABLE AND CLEAR instructions also share a common procedure called "ipv" which initialises the pointer variable. For the first instruction above, nothing else remains to be done. For the second instruction above, the new heap space generated must be set to zero after calling "ipv".

5) The EXIT PASCAL PROGRAM instruction performs the reverse of the RUN procedure in the environment simulation segment. In other words, it returns to the calling program. The calling program is first input and also any program variables, such as the program name are reset. If the program is input from disc without any errors occuring then the LINE and RESULT variables initially specified by the call of RUN are set (the result will always be TERMINATED since if an exception occurs then interpretation of the program ceases). The stack and the registers are reset using the procedure "ext" which is shared with the EXIT PROCEDURE P-code instruction.

6) Note that the TRUNCATE REAL and CONVERT WORD TO REAL instructions are the only real P-code instructions that are implemented entirely in Algol 68. The TRUNCATE REAL instruction is the only one which can possibly generate an overflow exception.

2.5 Operational Details

2.5.1 P-code Utility Programs

Because of the awkward format of the SOLO system files it was necessary to write several utility programs before the interpreter could be used. This was because all Pascal programs expect the characters in source files to be stored in ascii code. However, PR1ME source files have the most significant bit of each character set to 1.

The P-code utility programs are listed below :-

1) PDASSM

Produces a source file called 'PCODE' containing a dissassembled P-code listing of a binary file. The run file is called PDASSM.

2) SPLIST

Lists out, at the terminal, a source file. The source file must be in SOLO format, i.e. it must have no parity bits set. The run file is called PLIST and is stored in CMDNC0.

3) PSTRIP

Converts a PR1ME source file to a SOLO source file by stripping off the parity bits. The run file is called *PSTRIP.

4) CONVSC

Converts a SOLO source file to a PR1ME source file by adding on parity bits. The run file is called *CONVS.

5) BPRINT

Produces a source file, called 'PCODE', containing a numerical listing of a binary file.

6) BCOMP

Compares two binary files to see if they are equal. This was used to compare the compiler output, when run by the interpreter, with a compiled file from the distribution tape. This helps to check whether the interpreter was working correctly. The run file is called *BCOMP.

7) PATCH

Performs patches on words of a file. This was used to alter the LARGEST_REAL procedure used in PASS1 of the compiler.

2.5.2 Running the Interpreter

To run the interpreter first attach the directory CPCOMP. Run the interpreter by typing :-

```
R PINT 2/31
```

The "2/31" part is necessary to avoid overflowing the Algol 68 stack. The interpreter then responds with :-

```
P-CODE INTERPRETER - 6/3/81
$
```

The dollar sign is a prompt for input. Note that a Pascal program uses the character "P" as a prompt. All that is needed now is to type in the program name, followed by a list of parameters (if any) in brackets.

```
e.g. SPASCL(FILE1,FILE2,FILE3)
```

The program will now be interpreted. When the Pascal program terminates the interpreter will output a prompt ("$"). The same process can then be repeated or an exit can be made from the interpreter by typing "QUIT" when the prompt appears.

Some useful Pascal utility programs are described below.

EDIT

This is the editor from the SOLO operating system. It has two parameters, the first being the name of the input file to be edited and the second, which is optional, is the name of a new output file. This program can be used to create a new Sequential Pascal source program by editing the prefix file.

e.g.

```
 $ EDIT(PREFIX,NPROG)
P DEL(101)
P insert the new program text
P .
  .
  .
  .
P .
P #
```

A full description of the EDIT program can be found in the file CP49 in the PASCAL directory. After this files can either be edited using EDIT or by using the PR1ME editor. To use the PR1ME editor the files must first be converted to PR1ME format using CONVSC, edited and then converted back to SOLO format using PSTRIP.

e.g.

```
OK, R *CONVS
    INPUT FILE -
    NPROG
    OUTPUT FILE -
    TEMP
    TRANSFER COMPLETE
OK, AED TEMP
  .
  .
  .
  .
 FILE
OK, R *PSTRIP
   PARITY STRIP -
   SOURCE FILE
```

```
   TEMP
   DESTINATION FILE
   NPROG
   TRANSFER COMPLETE
OK,
```

The machine's responses are underlined.

LIST

This is a Pascal program which lists out a source file. It has 3 parameters, the file name and two optional parameters which are integer and specify the first line to be printed out and the last line to be printed out.

e.g. LIST(NPROG)

or LIST(NPROG,1,100)

Note that a SOLO format source file can also be listed out without entering the interpreter by typing:-

PLIST filename

PLIST is the run file of SPLIST described in section 5.1. It is stored in CMDNC0 and hence can be called from any directory.

SPASCL, CPASCL

These are the Sequential and Concurrent Pascal compiler driver programs. They have three parameters, the first is the file name of the program to be compiled. The second two parameters are, respectively, the name of a listing file and the name of an object

code file. If they are ommitted then the listing file name will be the program source file name prefixed with "L_" and truncated to six characters. The object file name will be the program source file name prefixed by "B_" and truncated to six characters.

e.g. SPASCL (NPROG,LSTNG,OBJECT)
Would compile a Sequential Pascal program called "NPROG", produce a listing in the file "LSTNG" and, if the program compiled successfully, put the object code in a file called "OBJECT".

CPASCL(CPROG)
Would compile a Concurrent Pascal program stored in a file called "CPROG", produce a listing in "L_CPRO" and, if the compilation was successful , put the object code in a file called "B_CPRO".

2.5.3 Interpreter Performance

The interpreter is extremely slow. A Pascal program, of about 600 lines say, can take up to half an hour to compile.

Some benchmarks were tested on the PR1ME and also on the Motorola 6809 compiler (the P-code for which is translated and not interpreted). In some cases the PR1ME interpreter was up to 10 times slower, although the real arithmetic benchmark, for example, was only twice as slow.

Appendix 3

P-code Interpreter Source Listings

```
TITLE parser segment
£ SEGMENT 1£
£
 PCODE INTERPRETER - PART 1 - INITIALISATION
 T.E. SHARP - FEBRUARY 1981
£

USING USER FROM "A68SYS)U_ENVR)REAL"

INT max proq length = 35*256;
INT max store = 20000;
INT spstart = max store - max proq length - 1;
INT pcstart = max store - max proq length;
INT heap bottom = 0;
INT param length = 11;     £ including space for return address£
INT null space = ABS 16r20;
INT word null = ABS 16r2020;
INT upcase = ABS ("A") - ABS ("a");
FILE tty = standout;

STRING proqname;
[0:max store] INT store;
[0:2] INT errvec;
[0:6] INT proq;

INT pc, q := 0, b := 0, h, sp; £registers£
INT const start, file size, hold;
INT pcdisplacement;
INT startpos, argument type, arqpointer;
BOOL syntax error;
BOOL parseok := TRUE;
BOOL interpret mode;
STRING syntax ="SYNTAX ERROR ";
STRING command line;
CHAR c;
[1:12] CHAR argument;
[] CHAR quit = ("Q", "U", "I", "T");

PROC error = (STRING message, name) VOID:
IF parseok
THEN
     FILE save := standout;
     standout := tty;
     print ("ERROR :", message, name, newline);
     IF interpret mode
     THEN
          print ("program counter", pc - pcdisplacement, " PROGLINE", store [b], newline)
     FI;
     parseok := FALSE;
     standout := save
FI;

[] CHAR lookup = ("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F");
PROC print hex = (REF INT param) VOID :
```

```
£prints out param as a 4 digit hex number ,uses "lookup" CHAR array£
BEGIN
     BITS number;
     [1:4] BITS digit;
     number := BIN param;
     FOR i FROM 4 DOWNTO 1
     DO digit[i] := number & (BIN 15);
        number := number SHR 4
     OD;
     FOR i FROM 1 TO 4
     DO printchar( lookup[ABS (digit[i]) + 1]) OD
END;

PROC read argument = ([] CHAR endchar) BOOL:
£ Reads in all characters until a non alpha numeric character is read in.
 Argument should end with one of the characters in "endchar","startpos" points to
the string element of "command line" where the argument begins.
"argument type "will be set to 1 for all alphabetic characters
                               2 for all numeric characters
                               3 for a combination of alphabetic and numeric characters.
returns TRUE if syntactically correct.
£
BEGIN
      BOOL alpha := FALSE, numeric:= FALSE, syntax error;
       c := REPR 0;
      argpointer := 0;
      £ process the argument £
      FOR char FROM startpos TO UPB command line
      WHILE
            BOOL valid1,valid2;
            c := char ELEM command line;
            startpos := char;
            IF c >= "a" ANDF c<="z"
            £ Convert to Upper Case £
            THEN c := REPR (ABS c + upcase) FI;
            valid1 := c >= "A" AND c <= "Z" OR c = " " OR c = "**";
            valid2 := c >= "0" AND c <= "9";
            alpha := alpha OR valid1;
            numeric := numeric OR valid2;
            (valid1 OR valid2 OR c = " ") AND parseok
       DO
          IF argpointer >= UPB argument
          THEN error ("IDENTIFIER LENGTH"," IS MAXIMUM 12 CHARACTERS");c := REPR 0
          ELIF c /= " "
          THEN argument [argpointer +:= 1] := c
          FI
       OD;

       £ Check for valid termination character £
       IF startpos = UPB command line
       THEN c := REPR 10 FI;      £ Newline £
       FOR  char FROM 1 TO UPB endchar
       WHILE syntax error := FALSE;
             c /= endchar [char]
```

```
        DO syntax error := TRUE OD;

        argument type := ABS (alpha) + 2*ABS (numeric);
        NOT syntax error
END;
PROC input parameters = VOID:
BEGIN BITS word;
      INT params;
      STRING param type, param name;
      BOOL param not ok;
      INT integer;
      interpret mode := FALSE;
      pc := pcstart;
      h := heap bottom;
      sp := spstart;
      read (newline, command line );
      startpos := 1;
      IF NOT read argument ((","(",REPR 10))   £ progname £
      THEN error (syntax,"PROGRAM NAME")
      ELSE
           £
           ************************************************************
           * THIS IS WHERE THE PROGRAM EXITS WHEN THE USER TYPES "QUIT"*
           ************************************************************
           £
           BOOL terminate := TRUE;
           FOR char TO UPB quit
           WHILE terminate
           DO
              IF argument [char] ~= quit [char]
              THEN terminate := FALSE
              FI
            OD;
            IF terminate AND argpointer = 4 THEN stop FI;
            prog [0] := argpointer;      £ Char Count £
            FOR i TO argpointer OVER 2
            DO prog [i] := ABS ( BIN ABS argument [i*2] ! (BIN ABS argument [i*2 - 1] SHL 8))
            OD;
            IF argpointer/2 ~= argpointer OVER 2
            THEN
                 prog [argpointer OVER 2 + 1] := ABS (BIN ABS argument [argpointer] SHL 8)
            FI;
            progname := as string (prog);
           £ First parameter is not accessible to the user £
           store [sp] := h;
           store [h] := 1;       £ OK = BOOLEAN £
           store [h + 1] := 1;
           h +:= 7;
           £ Input the other parameters £
           params := 1;
           FOR count FROM 2 TO paramlength - 1
           WHILE startpos ELEM command line ~= ")"
            ANDF startpos ELEM command line ~= ";"
            ANDF parseok
```

```
 ANDF c /= REPR 10
DO
   params := count;     £ No. of paramaters so far £
   FOR i TO UPB argument DO argument [i] := REPR 0 OD;    £ Clear Argument £
   startpos +:= 1;
   IF NOT read argument ((",",")",REPR 10))
   THEN error (syntax,"PARAMATER LIST")
   ELSE  IF argument type = 1 ORF argument type = 3
         THEN
               £ Alpha numeric chars £
               IF    argument [1] = "I"
               ANDF  argument [2] = "R"
               ANDF  argument [3] = "U"
               ANDF  argument [4] = "E"
               ANDF  argument [5] = REPR 0
               THEN
                     store [h] := 1;
                     store [h + 1] := 1
               ELIF  argument [1] = "F"
               ANDF  argument [2] = "A"
               ANDF  argument [3] = "L"
               ANDF  argument [4] = "S"
               ANDF  argument [5] = "E"
               ANDF  argument [6] = REPR 0
               THEN
                      store [h] := 1;
                      store [h + 1] := 0
                ELSE  £id type £
                      store [h] := 3; £ ID TYPE £
                      integer := h + 1;
                      hold := 0;
                      FOR i TO 6 WHILE i*2 <= argpointer
                      DO
                         word := BIN ABS  argument [i*2] ;
                         word := (BIN ABS argument [i*2 - 1] SHL 8 ) ! word;
                         store [integer] := ABS word;
                         integer +:= 1;
                         hold := i
                      OD;
                      hold  +:= 1;
                      IF argpointer/2 /= argpointer OVER 2
                      THEN £ 1 more character to go £
                           word := BIN null space;
                           word := ( BIN ABS argument [argpointer]  SHL 8) ! word;
                           store [integer] := ABS word;
                           integer +:= 1
                      FI;
                      FOR i FROM integer TO h + 6
                      DO store [i] := wordnull OD
               FI
         ELIF  argument type = 2
         THEN
               INT number := 0;
               store [h] := 2;      £ INT type £
```

```
                    FOR i TO argpointer
                    DO number := number*10 + (ABS argument [i] - ABS ("0")) OD;
                    store [h + 1] := number
                ELSE error (syntax,"")
                FI;
                h +:= 7
            FI
        OD
    FI;
    FOR count FROM params + 1 TO 10
    £ All other paramaters are 'niltype' £
    DO store [h] := 0; h +:= 7 OD;
    TO 2 DO store [sp -:= 1] := 0 OD  £ return address and prefix routines address - not use
d £
END £ PROC input paramaters £;


PROC input program = (STRING pname) VOID:
BEGIN [0:2] INT errvec;REF INT page;[0:3] INT buffer;
    search (1,dos string (pname),5); £Open prog file for reading £
    get err (errvec,2);
    IF errvec [1] ~= 0
    THEN error (" CAN'T OPEN ",pname);GOTO exit
    FI;

    £ Read in the first 4 words £
    IF prwfil (BIN 101,5,buffer,4) ~= 0
    THEN error ("FILE ERROR IN ",pname);GOTO exit
    FI;
    file size :=  buffer [0] OVER 2;
    const start :=  buffer [1] OVER 2;
    IF filesize > max prog length
    THEN error (pname," PROGRAM IS TOO LARGE");GOTO exit
    FI;

    £ Read in virtual code £
    page := store [pcstart + 1 + file size - const start];
    IF prwfil (BIN 1,5,page,const start) ~= 0
    THEN error ("CAN'T READ ",pname);GOTO exit
    FI;

    £ Read in the constants £
    page := store [spstart + 2];
    IF prwfil (BIN 1,5,page,file size - const start) ~= 0
    THEN error ("CAN'T READ ",pname);GOTO exit
    FI;

    pc +:= ( pcdisplacement := file size - const start);
    const start := spstart + 1;
    pcdisplacement +:= spstart+1;

    exit: search (4,dos string (pname),5); £close file £
    interpret mode := TRUE
END £ PROC input program £;
```

```
£ Remove when testing of segment complete £
£*************************************************************************£
CO
IF parseok
THEN
     FILE result;
     input parameters;
     input program (progname);
     open (result, "RESULT", standout channel);
     standout := result;
     print("P-CODE INTERPRETER : CORE DUMP", newline);
     print (pc, h, sp, newline);
     FOR i FROM pcstart TO pc + filesize
     DO print (i, store [i] );print("  ");print hex (store [i]); print (newline) OD;
     print(newline, "*************************************************************", newline);
     print("HEAP :", newline);
     FOR i FROM heap bottom TO h
     DO print (i, store [i] );print ("  "); print hex (store [i]); print (newline) OD;
     print("*************************************************************************", newline, ne
wline);
     print("STACK :", newline);
     FOR i FROM sp TO spstart
     DO print (i, store [i]);print ("  ");print hex (store [i]); print (newline) OD
FI
CO
£*************************************************************************£
£END OF SEGMENT£
ENVIRON SEGONE
```

```
TITLE trace segment
USING SEGONE FROM "PLEASE)U ENVR)PINIT"
£ Compile between segment 1 and segment 2 and insert "trace" calls in segment 2£
£
P-CODE INTERPRETER - OPTIONAL TRACER SEGMENT
T.E. SHARP - FEBRUARY 1981
Produces a printout of stack pointer, program counter and top "sp" words whenever
a SCL instruction occurs
£

FILE tracer,save;
INT stackprint = 6;
INT start = 1,end = 2;
INT trace trigger,trace length,trace left := 0,num traces := 0,trace start := 1;
BOOL trace go := FALSE;

[] STRUCT (STRING name,INT paramlength) prlist
       = (("READ",1) , ("WRITE",1) ,
          ("OPEN",3) , ("CLOSE",1) ,
          ("GET",3)  , ("PUT",3)   , ("LENGTH",1) ,
          ("MARK",1) , ("RELEASE",1) ,
          ("IDENTIFY",1) , ("ACCEPT",1) , ("DISPLAY",1) ,
          ("READPAGE",3) , ("WRITEPAGE",2) ,
          ("READLINE",1) , ("WRITELINE",1) ,
          ("READARG",2)  , ("WRITEARG" ,2) ,
          ("LOOKUP" ,3)  , ("IOTRANSFER",3) , ("IOMOVE",2) ,
          ("TASK",0) , ("RUN",4) , ("NULL",0));

PROC trace = ( INT routine,where) VOID:
BEGIN
      IF pc - pcdisplacement = trace trigger
      THEN numtraces +:= 1;
           trace start -:= 1;
           IF trace start = 0
           THEN trace go := TRUE
           FI;
           trace left := trace length
      FI;
      IF trace left > 0 ANDF trace go
      THEN
           trace left -:= 1;
           save := standout;
           standout := tracer;
           print (IF where = start THEN "START" ELSE "END" FI );
           print (pc - pcdisplacement,sp,g,b,h,name OF prlist [routine],newline);
           FOR i FROM sp TO sp + paramlength OF prlist [routine] + stackprint
           DO print (store [i]) OD;
           print (newline,newline);
           standout := save
      FI
END £ PROC trace £ ;

open (tracer,"TRACE",standout channel);
print ("TRACE TRIGGER -",newline);
```

```
read (newline, trace trigger);
print ("TRACE LENGTH -", newline);
read (newline, trace length);
print ("TRACE START -", newline);
read (newline, trace start);
ENVIRON SEGTRACE
```

```
TITLE enviroment simulation segment1
USING SEGTRACE FROM "PLEASE)U ENVR)PTRACE"
£
P-CODE INTERPRETER
PART 2 (A) - SIMULATION OF SOLO PREFIX ROUTINES
- UTILITY PROCEDURES USED BY "PROC sc1" IN NEXT SEGMENT
T.E.SHARP - FEBRUARY 1981
£
INT address;
INT maxarg = 10, arglength = 7;
[1:4] STRUCT (STRING name, dos name) filename;
[] STRUCT (STRING name, INT dummy) special prog = (("FILE        ",0),("PRINT       ",0));
INT scratchfile = 1, ascii = 2, seqcode = 3, concode = 4;
[1:2] FILE file channel;
INT nesting; £ Level of nesting in program calls £
INT max call = 10;
[1:maxcall] STRUCT (STRING name, idname, BOOL new) program;
REF [] INT page;
[1:4] BOOL trans err := (FALSE, FALSE, FALSE, FALSE);
[1:2] BOOL endfile;
INT iptr, optr, iwords read;
INT filenum;
BOOL nline := TRUE;
BOOL execution := TRUE;
[-1:255] INT inbuff, obuff;
REF [] INT ibsptr = inbuff [0];   £ "ibsptr [0]" points to "inbuff [1]" £
INT idlength = 7, line length = 32;
INT prog type = -1, id type = max call - 1, file type = 2*max call - 1;
HEAP [0:6] INT scratch;
HEAP [0:(2 * max call + 4) * idlength - 1] INT name store;
INT racall := 0;
INT blocks written := 0;
[1:132] CHAR accept buffer;
INT buffptr;

£
 get/put ____ param procedures follow
 - "index" specifies displacement from stack top
 - "get" procedures assume value on stack
 - "put" procedures assume address on stack
£

PROC prwfile = (INT a1, INT a2, REF INT a3, INT a4, INT a5) VOID: CODE "MABCD0" EDOC;

PROC get char param = (INT index) CHAR :
(REPR store [sp + index - 1]);

PROC put char param =(INT index, CHAR char) VOID:
(store [ ABS (BIN store [sp + index - 1] SHR 1)] := ABS char);

PROC get int param = (INT index) INT :
(store [sp + index - 1]);

PROC put int param = (INT index, int) VOID:
```

```
(store [ ABS (BIN store [sp + index - 1] SHR 1)] := int);

PROC get bool param = (INT index) BOOL:
(store [sp + index - 1] = 1);

PROC put bool param = (INT index,BOOL b) VOID:
(store [ ABS (BIN store [sp + index - 1] SHR 1)] := ABS b);

PROC get id param = (INT index,length,heap index,REF STRING id,BOOL argtype,permanent) VOID:
BEGIN
     REF [] INT hptr;
       INT displacement;
     page := store [IF argtype THEN ABS (BIN store [sp + index - 1] SHR 1) + 1 ELSE ABS (BIN s
tore [sp + index - 1] SHR 1) FI ];
     hptr := IF permanent
             THEN displacement := IF heap index = file type
                                  THEN filenum
                                  ELSE nesting
                                  FI;
                  name store [(heap index + displacement) * idlength + 1]
              ELSE scratch [1]
              FI;
     hptr [0] := length * 2;        £ Character Count £
     FOR i TO 6 DO hptr [i] := page [i] OD;
     id := as string (hptr)
END £ PROC get id param £;

PROC strip params = (INT params) VOID:
(sp +:= params );

PROC read buffer = VOID:
BEGIN
      transerr [1] := transerr [1] OR prwfil (BIN 401,15,inbuff,256) ~= 0;
      geterr (errvec,2);
      iwords read := errvec [1] * 2;
      iptr := 0;
      inbuff [-1] := iwords read          £ Fiddling The Caharacter Count £
END;

PROC write buffer = VOID:
BEGIN
      transerr [2] := transerr [2] OR prwfil (BIN 2,14,obuff,256) ~= 0;
      optr := -1
END;

PROC run = VOID:
BEGIN
      BOOL special case := FALSE;
      INT prog num;
      INT arg address; BOOL ok;
      nesting +:= 1;
      get id param (4,6,prog type,name OF program [nesting],FALSE,TRUE);
      FOR i TO UPB special prog WHILE NOT special case
      DO special case := name OF program [nesting] = name OF special prog [i];
```

```
   prognum := i
OD;
IF special case
THEN
     CASE prognum
     IN
     £ FILE £
       (STRING command;
         arg address := ABS (BIN get int param (3) SHR 1) + 7;
        INT save ;
        REF [] INT param;
        INT charcount = 2*idlength - 2;
        STRING srf = "SCL RUN (FILE) ", ipt = "INVALID PARAMATER TYPE";
        param := store [argaddress + 1];
        IF param [0] <> 3     £ ID TYPE £
        THEN error (srf, ipt)
        FI;
        save := param [0];
        param [0] := charcount;
        command := as string (param);
        IF command = "CREATE       " ORF command = "REPLACE      "
        THEN
             [0:255] INT buffer;
             param [0] := save;
             param :=  store [arg address + idlength + 1];    £ pointer to new file namef
             IF param [0] <> 3
             THEN error (srf, ipt)
             FI;
             save := param [0];
             param [0] := char count;
             search (4, dos string("NEXT"), 12);
              £ Open File £
                  search (3, dos string ("NEXT"), 10);
                  geterr (errvec, 2);
              IF errvec [1] <> 0
              THEN error (srf, "CREATE - CAN'T OPEN 'NEXT'")
              FI;

             £Copy The File Across £
             search (2, dos string(as string(param)), 11);
             get err (errvec, 2);
             IF errvec [1] <> 0 THEN
             error("SCL RUN (FILE) - CREATE - CAN'T OPEN ", as string(param))
             FI;
             IF store [arg address + 2*idlength] <> 2  £ INT type £
             THEN error (srf, ipt)
             FI;
             TO store [arg address + 2*idlength + 1]
             WHILE prwfil (BIN 1, 10, buffer, 256 ) = 0
             DO ok := prwfil (BIN 2, 11, buffer, 256) = 0
             OD;
             search (4, dos string ("NEXT"), 10);
             search (4, dos string (as string (param)), 11);
             param [0] := save
```

```
                ELSE error(srf,"UNIMPLEMENTED COMMAND")
                FI)
                ,SKIP
            ESAC;
            name OF program [nesting] := "";
            store [arg address - 7] := 1;       £ BOOL type £
            store [arg address - 6] := ABS ok;
            put int param (1,0);        £ RESULT = TERMINATED £
            nesting -:= 1;
            strip params (4)
        ELSE
            FILE save := standout;
            INT line addr,result addr;
            idname OF program [nesting] := "";
            new OF program [nesting] := TRUE;

            £Put the parameters on the stack£
            result addr := get int param (1);
            line addr := get int param (2);
            address := get int param (3); £start address "ARGLIST" £
            store [sp + 1] := address;      £ Leave dummy word and ARGLIST,LINE and RESULT addre
sses on top of stack £
            store [sp + 2] := line addr;
            store [sp + 3] := result addr;
            store [sp -:= 1] := pc + 1; £return address£
            parsok := TRUE;
            input program (name OF program [nesting]);
            IF execution := parsok
            THEN
                pc := pcdisplacement - 1;
                standout := tty;
                print (clock," ",name OF program [nesting]," ENTERED",newline)
            FI;
            standout := save
        FI
END;

FOR i TO 4 DO name OF filename [i] := ""OD;
accept buffer [132] := REPR 10;
ENVIRON SEGATWO
```

```
TITLE enviroment simulation segment2

£
P-CODE INTERPRETER
PART 2 (B) - SIMULATION OF SOLO PREFIX ROUTINES
T.E.SHARP FEBRUARY 1981
£
USING SEGATWO FROM "PLEASE>U_ENVR>PPROC"

PROC scl = VOID: £ CALL SYSTEM instruction - 1 Argument ENTRY £
BEGIN
      CHAR c;
      BOOL test;
      CASE ROUND (store [pc]/2) + 1
           IN
           (
           £INDEX = 0
            PROCEDURE READ (VAR C:CHAR);
           £
            trace(1,start);
           IF iptr = iwords read
           THEN £ buffer is exhausted £
                read buffer
           FI;
           put char param (1,(iptr +:= 1) ELEM as string (ibsptr));
           strip params (1);
            trace (1,end)
           )
           ,
           (
           £
           INDEX = 2
           PROCEDURE WRITE (C:CHAR) ;
           £
           BITS word;
            trace(2,start);
           word := BIN ABS get char param (1);
           obuff [ABS (BIN optr SHR 1)] := IF optr/2 = ABS (BIN optr SHR 1)
                                           THEN £(m.s.) byte £ ABS (word SHL 8)
                                           ELSE £(l.s.) byte £ ABS (BIN ( obuff [ ABS (BIN
optr SHR 1)]) ! word)
                                           FI;
           IF optr = 256*2-1
           THEN £buffer is full£
                write buffer
           FI;
           optr +:= 1;
           strip params(1);
            trace(2,end)
           )
           ,
           (
           £
           INDEX = 4
```

```
PROCEDURE OPEN (F:FILE;ID:IDENTIFIER;VAR FOUND:BOOLEAN);
£
 trace(3,start);
filenum := get int param (3) + 2;
IF name OF filename [filenum] <> ""
THEN search (4,dos string(name OF filename [filenum]),16 - filenum)      £ Close Cu
rrent File First £
FI;
get id param (2,6,file type,name OF filename [filenum],FALSE,TRUE);
dos name OF filename [filenum] := dos string (name OF filename [filenum]);
search(3,dos name OF filename [filenum],16 - filenum);
get err (errvec,2);
put bool param (1, errvec [1] = 0);
transerr [filenum] := FALSE;
strip params (3);
 trace(3,end)
)
,
(
£
INDEX = 6
PROCEDURE CLOSE (F:FILE);
£
 trace(4,start);
filenum := get int param (1) + 2;
IF name OF filename [filenum] = ""
THEN error("SCL CLOSE","NO FILE OPEN")
ELSE
     search(4,dos name OF file name [filenum],16 - filenum);
     name OF filename[filenum] := ""
FI;
strip params (1);
 trace (4,end)
)
,
(
£
INDEX = 8
PROCEDURE GET (F:FILE;P:INTEGER;VAR BLOCK:UNIV PAGE);
£
INT blocknum;
 trace (5,start);
filenum := get int param (3) + 2;
blocknum := get int param (2) - 1;
page := store [ ABS (BIN get int param (1) SHR 1) + 1];
search ( 7,dos name OF file name [filenum],16 - filenum); £rewind£
prwfile (1,16 - filenum,page [0],256,blocknum*256);
get err(errvec,2);
transerr [filenum] := transerr [filenum] OR  errvec [1] <> 0;
strip params (3);
 trace(5,end)
)
,
(
```

```
£
INDEX = 10
PROCEDURE PUT (F:FILE;P:INTEGER;VAR BLOCK:UNIV PAGE);
£
INT blocknum;
 trace (6,start);
filenum := get int param (3) + 2;
blocknum := get int param (2) - 1;
page := store [ ABS (BIN get int param (1) SHR 1) + 1];
search( 7,dos name OF file name [filenum],16 - filenum);  £rewind£
prwfile (2,16 - filenum,page [0],256,blocknum*256);
geterr(errvec,2);
transerr [filenum] := transerr [filenum] OR errvec [1] ~= 0;
strip params (3);
 trace (6,end)
)
,
(
£
INDEX = 12
FUNCTION LENGTH (F:FILE):INTEGER;
£
INT lehgth;
 trace (7,start);
filenum := get int param (1) + 2;
IF name OF filename [filenum] = "TEMP1        "
ORFname OF filename [filenum] = "TEMP2        "
ORFname OF filename [filenum] = "NEXT         "
THEN length := 255
ELSE [0:255] INT buffer;
     search ( 7,dos name OF filename [filenum],16 - filenum); £rewind£
     FOR i WHILE prwfil (BIN 1,16 - filenum,buffer,256) = 0
     DO length := i OD
 FI;
strip params (1);
store [sp] := length;          £ FUNCTION RESULT £
 trace (7,end)
)
,
(
£
INDEX = 14
PROCEDURE MARK (VAR TOP:INTEGER);
£
 trace (8,start);
put int param (1,h*2); £return heap top address£
strip params (1);
 trace (8,end)
)
,
(
£
INDEX = 16
PROCEDURE RELEASE(TOP:INTEGER);
```

```
£
 trace (9,start);
h := ABS (BIN get int param (1) SHR 1); £set heaptop£
strip params (1);
 trace (9,end)
)
,
(
£
INDEX = 18
PROCEDURE IDENTIFY (HEADER:LINE);
£
INT faddr;
 trace(10,start);
faddr := address :=  ABS (BIN get int param (1) SHR 1);
FOR i TO 6
WHILE (BIN store [address]) SHR 8 ¬= BIN 10
  AND ((BIN store [address]) & 16rff) ¬= BIN 10
DO address +:= 1 OD;
get id param (1,address - faddr + 1,idtype,idname OF program [nesting],FALSE,TRUE);
new OF program [nesting] := TRUE;
strip params (1);
 trace (10,end)
)
,
(
£
INDEX = 20
PROCEDURE ACCEPT (VAR C:CHAR);
£
 trace (11,start);
IF  nline THEN tty write ch ("P");tty write ch (" ");nline := FALSE ;
              buffptr := 0;
              WHILE
                    buffptr +:= 1;
                    tty read ch (c);
                    IF c = tty erase char ANDF buffptr > 1
                    THEN buffptr -:= 2
                    ELSE accept buffer [buffptr] := c
                    FI;
                    c ¬= REPR 10 AND buffptr < 132
              DO SKIP OD;
              buffptr := 0
         FI;
c := accept buffer [buffptr +:= 1];
IF ABS c = 10 THEN nline := TRUE FI;
put char param (1,c);
strip params (1);
 trace (11,end)
)
,
(
£
INDEX = 22
```

```
PROCEDURE DISPLAY (C:CHAR);
£
FILE save;
 trace (12,start);
save := standout;
standout := tty;
IF new OF program [nesting]
THEN print (idname OF program [nesting],newline);
     new OF program [nesting] := FALSE
FI;
c := get char param (1);
IF ABS c = 10 THEN print (newline) ELSE print (c) FI;
standout := save;
strip params (1);
 trace (12,end)
)
,
(
£
INDEX = 24
PROCEDURE READPAGE (VAR BLOCK:UNIV PAGE;VAR EOF:BOOLEAN);
£
BOOL eof;
 trace (13,start);
page := store [ ABS (BIN get int param (2) SHR 1) + 11;
put bool param (1,prwfil (BIN 1,15,page,256) ¬= 0);
strip params (2);
 trace (13,end)
)
,
(
£
INDEX = 26
PROCEDURE WRITEPAGE (BLOCK:UNIV PAGE;EOF:BOOLEAN);
£
 trace (14,start);
page := store [ABS (BIN get int param (2) SHR 1) + 11;
IF NOT get bool param (1)
THEN trans err [2] := trans err [2] OR prwfil (BIN 2,14,page,256) ¬= 0;blocks writt
en :=: 1
ELSE search (4,dos string(name OF filename [2]),14) fclose file£
FI;
strip params (2);
 trace (14,end)
)
,
(
£
INDEX = 28
PROCEDURE READLINE (VAR TEXT:UNIV LINE);
£
REF [] INT ptr;
REF [] INT line;
STRING instring;
```

```
            trace (15,start);
            line := store [ABS (BIN get int param (1) SHR 1) + 1];
            ptr := inbuff [0];
            instring := as string (ptr);
            FOR line count TO line length
            WHILE INT lo2 := ABS (BIN line count SHR 1);
                  BITS word;
                  IF iptr = iwords read
                  THEN read buffer FI;
                  word := IF line count /2 = lo2
                              THEN £ (l.s.) byte £
                                   BIN line [lo2] ! BIN ABS ((iptr +:= 1) ELEM instring)
                              ELSE £ (m.s.) byte £
                                   BIN ABS ((iptr +:= 1) ELEM instring) SHL 8
                              FI;
                              line [lo2] := ABS word;
                              word ~= 16ra
                              DO SKIP OD;
            strip params (1);
             trace (15,end)
            )
            ,
            (
            £
            INDEX = 30
            PROCEDURE WRITELINE (TEXT:UNIV LINE);
            £
             trace (16,start);
            BITS word;
            STRING line;
            REF [] INT ptr;
            INT save;
            ptr := store [ABS (BIN get int param (1) SHR 1)];          £ Start Address Line £
            save := ptr [0];
            ptr [0] := line length;                              £ Fiddle character count £
            line := as string (ptr);
            FOR i TO 132
            WHILE
                  word := BIN ABS (i ELEM line);
                  obuff [ABS (BIN optr SHR 1)] := IF optr/2 = ABS (BIN optr SHR 1)
                                                  THEN £ (m.s.) byte £ ABS (word SHL 8)
                                                  ELSE £ (l.s.) byte £ ABS (BIN (obuff [ABS (BIN optr SH
R 1)]) ! word)
                                                  FI;
                  IF optr ~= 256*2 - 1 THEN write buffer FI;
                  optr +:= 1;
                  word ~= 16ra
             DO SKIP OD;
             ptr [0] := save;
            strip params (1);
             trace (16,end)
            )
            ,
            (
```

```
£
INDEX = 32
PROCEDURE READARG (S:ARGSEQ;VAR ARG:ARGTYPE);
£
INT argaddress;
 trace (17,start);
 argaddress := ABS (BIN get int param (1) SHR 1);
filenum := get int param (2) + 1;
 IF store [argaddress] = 2   £ INT type £
 THEN £ return length of file to ARGTYPE £
      store [argaddress + 1] := blocks written
 ELSE
      store [argaddress] := 1;     £BOOL TYPE£
      store [argaddress + 1] := ABS NOT transerr [filenum];
      IF filenum = 2 THEN write buffer FI;
      search (4,dos string (name OF filename [filenum]),16 - filenum);
      name OF filename [filenum] := ""
  FI;
strip params (2);
 trace (17,end)
)
,
(
£
INDEX = 34
PROCEDURE WRITEARG (S:ARGSEQ;ARG:ARGTYPE);
£
 trace (18,start);
filenum := get int param (2) + 1;
get id param (1,6,file type,name OF filename [filenum] ,TRUE,TRUE);
search (filenum,dos string (name OF filename [filenum]),16 - filenum);
get err (errvec,2);
IF errvec [1] <> 0
THEN error ("SCL WRITEARG ,CAN'T OPEN ",name OF filename [filenum])
FI;
transerr [filenum] := FALSE;
IF filenum = 1  £ Input £
THEN iptr := iwords read := 0
ELSE optr := 0  £ Output £
FI;
blocks written := 0;
strip params (2);
 trace(18,end)
)
,
(
£
INDEX = 36
PROCEDURE LOOKUP (ID:IDENTIFIER;VAR ATTR:FILE ATTR;VAR FOUND:BOOLEAN);
£
STRING filename;INT file attr;
INT index;
 trace (19,start);
get id param (3,6,file type,filename,FALSE,FALSE);
```

```
            address := ABS (BIN get int param (1) SHR 1);       £ FOUND address £
            fileattr:= ABS (BIN get int param (2) SHR 1);       £ FILEATTR address £
            search (6, dos string (filename), 1); £existf
            geterr (errvec, 2);
            store [address] := ABS (errvec [1] = 0);
            store [fileattr] :=  ascii;
            store [fileattr + 2] := 0;      £ Unprotected £
            strip params (3);
             trace (19, end)
            )
            ,
            (
            £
            INDEX = 38
            IOTRANSFER - NOT IMPLEMENTED
            £
             trace (20, start);
            error ("SCL IOTRANSFER", " NOT IMPLEMENTED");
             trace (20, end)
            )
            ,
            (
            £
            INDEX = 40
            IOMOVE - NOT IMPLEMENTED
            £
             trace (21, start);
            error ("SCL IOMOVE", " NOT IMPLEMENTED");
             trace (21, end)
            )
            ,
            (
            £
            INDEX = 42
            FUNCTION TASK :TASKKIND;
            £
             trace (22, start);
            store [sp] := 1;
             trace (22, end)
            )
            ,
            (
            £
            INDEX = 44
            PROCEDURE RUN (ID:IDENTIFIER;VAR PARAM:ARGLIST;VAR LINE:INTEGER;VAR RESULT:PROGRESU
LT);
            £
             trace (23, start);
            run;
            trace (23, end)
            )
        ESAC;
        pc ::= 1
END £ PROC scl £;
```

```
£end of segment£
ENVIRON SEGTWO
```

```
TITLE pcode interpreter segment1
£
  P-CODE INTERPRETER - PART 3(A) INSTRUCTION INTERPRETATION CODE
  T.E.SHARP - FEBRUARY 1981
£

USING SEGTWO FROM "PLEASE)U ENVR)PENVR"

INT instruction;
INT terminated = 0, overflow error = 1, pointer error = 2, range error = 3, variant error =4, heap
limit = 5, stack limit = 6;
BITS first sig digit = 16r4000;
[] BITS mask word = (16rfffe, 16rfffc, 16rfff8, 16rfff0,
                     16rffe0, 16rffc0, 16rff80, 16rff00,
                     16rfe00, 16rfc00, 16rf800, 16rf000,
                     16re000, 16rc000, 16r8000, 16r0000);
INT result := terminated, line, continue := 1;
INT w, x, y; £Scratch Registers£
BOOL overflow := FALSE;        £ Overflow Exception Not Implemented £
STRING uii = " UNIMPLEMENTED INSTRUCTION";
BOOL equal;
OP LESSUNSIGNED = (INT a, b) BOOL: (a<b);
PRIO LESSUNSIGNED = 5;

PROC ipv = VOID :
£ INITIALISE POINTER VARIABLE £
BEGIN
      IF (b - h) LESSUNSIGNED ABS (BIN store [pc] SHR 1) THEN exception (heap limit) FI;
      pc +:= 1;
      store [ ABS (BIN store [sp] SHR 1)] := h;              £ Set Pointer £
      sp +:= 1;
      h +:= store [pc];                                    £ Allocate Heap Space £
      pc +:= 1
END;

PROC check struct = VOID:
£Procedure to check two structures and see if they are equal
 or not. Used by "csl", "cse", "csg", "csge", "csne", "csle". Result
 is put in the Boolean "equal".£
BEGIN
     x := ABS (BIN store [sp] SHR 1);          £ Source Address £
     y := ABS (BIN store [sp +:= 1] SHR 1);          £ Destination Address £
     TO store [pc] WHILE equal := store [x] = store [y] DO x +:= 1; y +:= 1 OD;
     pc +:= 1
END;

£
Each of the following procedures corresponds to a P-code
instuction (except "exception"). For description see "Concurrent
Pascal P-code Description".
£

PROC exception = (INT reason) VOID:
BEGIN
```

```
      INT save := pc - pcdisplacement;
     result := reason;
     line := store [b];
     sp := g + 1;
     w := store [sp]; sp +:= 1;
     b := store [sp]; sp +:= 1;
     g := store [sp]; sp +:= 1;
     pc := store [sp];
     sp := w;
     standout := tty;
     print(CASE reason + 1
           IN "TERMINATED","OVERFLOW","POINTER","RANGE","VARIANT","HEAP LIMIT",
              "STACK LIMIT"
           OUT "UNKNOWN"
           ESAC);
     print(" EXCEPTION GENERATED IN ",name OF program [nesting],result,line,newline);
     print ("program counter ",save,newline);
     execution := FALSE
END;

£
*****************************************************************************
*                                                                           *
*            EXECUTABLE CODE STARTS HERE                                    *
*                                                                           *
***************************************************************************** £

print ("P-CODE INTERPRETER - 6/3/81",newline);
DO
   parseok := TRUE;
   nesting := 1;
   idname OF program [1] := "";
   WHILE input paramaters; NOT parseok DO parseok := TRUE OD;
   input program (name OF program [nesting] := progname);
   execution := parseok;     £ Program Read In Correctly £

   WHILE execution
   DO
       trace (24,start);
      instruction := ROUND (store[pc]/2);
      pc +:= 1;
      CASE instruction
      IN
          £ PUSH CONSTANT ADDRESS
          - 1 Argument DISPLACEMENT
          - INDEX = 2£
          BEGIN
                store [sp -:= 1] := const start * 2 + store [pc];
                pc +:= 1
          END,

          £ PUSH LOCAL ADDRESS
          - 1 Argument DISPLACEMENT
          - INDEX = 4 £
```

```
BEGIN
      store [sp -:= 1] := b * 2 + store [pc];
      pc +:= 1
END,

£ PUSH GLOBAL ADDRESS
- 1 Argument DISPLACEMENT
- INDEX = 6 £
BEGIN
      store [sp -:= 1] := g * 2 + store [pc];
      pc +:= 1
END,

£ PUSH CONSTANT
- 1 Argument VALUE
- INDEX = 8 £
BEGIN
      store [sp -:= 1] := store [pc];
      pc +:= 1
END,

£ PUSH LOCAL
- 1 Argument DISPLACEMENT
- INDEX = 10 £
BEGIN
      store [sp -:= 1] := store [  (b*2 + store [pc]) OVER 2];
      pc +:= 1
END,

£ PUSH GLOBAL
- 1 Argument DISPLACEMENT
- INDEX = 12 £
BEGIN
      store [sp -:= 1] := store [  ( g*2 + store [pc]) OVER 2];
      pc +:= 1
END,

£ PUSH WORD
- No Arguments
- INDEX = 14 £
( store [sp] := store [ ABS (BIN store [sp] SHR 1)] ),

£ PUSH BYTE
- No Arguments
- INDEX = 16 £
BEGIN
      w := ABS (BIN store [sp] SHR 1);
      store [sp] := IF store [sp]/2 = w
                    THEN £ (m.s.) byte £ ABS (BIN store [w] SHR 8)
                    ELSE £ (l.s.) byte £ ABS (BIN store [w] % 16rff)
                    FI
END,

£ PUSH REAL
```

```
- No Arguments
- INDEX = 18 £
BEGIN
      w := ABS (BIN store [sp] SHR 1) + 4;        £ Adress First Word £
      sp +:= 1;
      TO 4 DO w -:= 1;store [sp -:= 1] := store [w] OD
END,

£PUSH SET
- No Arguments
- INDEX = 20 £
BEGIN
      w := ABS (BIN store [sp] SHR 1) + 8;     £ Address First Word £
      sp +:= 1;
      TO 8 DO w -:= 1; store [sp -:= 1] := store [w] OD
END,

£ FIELD
- No Arguments
- INDEX = 22 £
( store [sp] +:= store [pc]; pc +:= 1 ),

£ PUSH ARRAY COMPONENT
- 3 Arguments MIN, MAX-MIN, LENGTH
- INDEX = 24 £
BEGIN
      x := store [sp] - store [pc];     £ Array Index - MIN £
      sp +:= 1;
      pc +:= 1;
      IF x < 0 ORF x > store [pc]
      THEN exception (range error)
      FI;
      store [sp] +:= x*store [pc +:= 1];       £ Add To Base Address £
      pc +:= 1
END,

£ TEST POINTER
- No Arguments
- INDEX = 26 £
IF store [sp] = 0 THEN exception (pointer error) FI,

£ CHECK TAG FIELD
- 2 Arguments DISPLACEMENT, TAGSET
- INDEX = 28 £
BEGIN
      BITS w;
      x := store [ (store [sp] + store [pc]) OVER 2];      £ Tag Address £
      pc +:= 1;
      w := BIN 1 SHL x;
      w := w & BIN store [pc];         £ Mask Out All the Other Bits £
      pc +:= 1;
      IF w = BIN 0 THEN exception (variant error) FI
END,
```

```
        £ CHECK RANGE
        - 2 Arguments MIN,MAX
        - INDEX = 30 £
        ( IF store [sp] ( store [pc] ORF store [sp] ) store [pc + 1] THEN exception (range e
rror) FI;pc :+= 2),

        £ COPY BYTE
        - No Arguments
        - INDEX = 32 £
        BEGIN
              BITS clear;
              sp +:= 1;
              w := ABS (BIN store [sp] SHR 1);      £ Destination Address £
              IF store [sp]/2 = w
              THEN £ (m.s) byte £
                   clear := BIN store [w] & 16r00ff;     £ Clear Byte £
                   store [w] := ABS ( clear ! (BIN store [sp - 1] SHL 8))
              ELSE £ (l.s.) byte £
                   clear := BIN store [w] & 16rff00;     £ Clear Byte £
                   store [w] := ABS ( clear ! (BIN store [sp - 1] & 16rff))
              FI;
              sp +:= 1
        END,

        £ COPY WORD
        - No Arguments
        - INDEX = 34 £
        ( store [ ABS (BIN store [sp + 1] SHR 1)] := store [sp];sp +:= 2 ),

        £ COPY REAL
        - No Arguments
        - INDEX = 36 £
        BEGIN
              w := ABS (BIN store [sp + 4] SHR 1);       £ First Address To Be Copied To £
              TO 4 DO store [w] := store [sp];w +:= 1;sp +:= 1 OD;
              sp +:= 1
        END,

        £ COPY SET
        - No Arguments
        - INDEX = 38 £
        BEGIN
              w := ABS (BIN store [sp + 8] SHR 1);      £ First Address To Be Copied To £
              TO 8 DO store [w] := store [sp];w +:= 1;sp +:= 1 OD;
              sp +:= 1
        END,

        £ COPY TAG
          1 Argument LENGTH (in words)
        - INDEX = 40 £
        BEGIN
              store [ ABS (BIN store [sp + 1] SHR 1)] := store [sp];      £ Copy Tag £
              sp +:= 1;
              x :=  ABS (BIN store [sp] SHR 1) + 1;                      £ Address First Wors
```

```
To Be Cleared £
            sp +:= 1;
            TO store [pc] DO store [x] := 0;x +:= 1 OD;
            pc +:= 1
      END,

      £ COPY STRUCTURE
      - 1 Argument LENGTH (in words)
      - INDEX = 42 £
      BEGIN
            x := ABS (BIN store [sp] SHR 1);              £ Source Address £
            y := ABS (BIN store [sp +:= 1] SHR 1);        £ Destination Address £
            sp +:= 1;
            TO store [pc] DO store [y] := store [x];y +:= 1;x +:= 1 OD;
            pc +:= 1
      END,

      £ INITIALISE POINTER VARIABLE
      - 2 Arguments STACKLENGTH:LENGTH,LENGTH
      - INDEX = 44 £
      BEGIN
            ipv
      END,

      £ INITIALISE POINTER VARIABLE AND CLEAR
      - 2 Arguments STACKLENGTH:LENGTH,LENGTH
      - INDEX = 46 £
      BEGIN
            ipv;
            x :=  h;
            TO  ABS (BIN store [pc - 1] SHR 1) DO x -:= 1; store [x] := 0 OD
      END,

      £ NOT
      - No Arguments
      - INDEX = 48 £
      ( store [sp] :=  1 - store [sp]),

      £ AND WORD
      - No Arguments
      - INDEX = 50 £
      ( store [sp + 1] := ABS ( BIN store [sp] & BIN store [sp + 1]);sp +:= 1 ),

      £ AND SET
      - No Arguments
      - INDEX = 52 £
      BEGIN
            w := sp + 8;
            x := sp : 15;
            FOR i FROM w TO x
            DO store [i] := ABS ( BIN store [sp] & BIN store [i]);sp +:= 1 OD
      END,

      £ OR WORD
```

```
- No Arguments
- INDEX = 54 £
( store [sp + 1] := ABS ( BIN store [sp] ! BIN store [sp + 1]); sp +:= 1 ),

£ OR SET
- No Arguments
- INDEX = 56 £
BEGIN
      w := sp + 8;
      x := sp + 15;
      FOR i FROM w TO x
      DO store [i] := ABS ( BIN store [sp] ! BIN store [i]);sp +:= 1 OD
END,

£ NEGATE WORD
- No Arguments
- INDEX = 58 £
BEGIN
      store [sp] := -store [sp];
      IF overflow THEN exception (overflow error) FI
END,

£ NEGATE REAL
- No Arguments
- INDEX = 60 £
BEGIN
      PROC negate real = (REF INT addr1) VOID: CODE "MNGREAL" EDOC;
      negate real (store [sp])       £REAL: store [sp] := -store [sp] £
END,

£ ADD WORD
- No Arguments
- INDEX = 62 £
BEGIN
      store [sp + 1] +:= store [sp];
      IF overflow THEN exception (overflow error) FI;
      sp +:= 1
END,

£ ADD REAL
- No Arguments
- INDEX = 64 £
BEGIN
     PROC add real = (REF INT addr1, addr2) VOID:CODE "MADREAL"EDOC;
     add real (store [sp], store [sp + 4]);      £ REAL: store [sp + 4] +:= store [sp]
£
     sp +:= 4
END,

£ SUBTRACT WORD
- No Arguments
- INDEX = 66 £
BEGIN
      store [sp + 1] -:= store [sp];
```

```
            IF overflow THEN exception (overflow error) FI;
            sp +:= 1
        END,

        £ SUBTRACT REAL
        - No Arguments
        - INDEX = 68 £
        BEGIN
            PROC subtract real = (REF INT addr1, addr2) VOID: CODE "MSBREAL" EDOC;
            subtract real (store [sp], store [sp + 4]);      £REAL: store [sp + 4] -:= store
[sp] £
            sp +:= 4
        END,

        £ SUBTRACT SET
        - No Arguments
        - INDEX = 70 £
        ( TO 8 DO store [sp + 8] := ABS ( BIN store [sp + 8] & ( BIN store [sp] NEQ 16rffff)
);sp +:= 1 OD ),

        £ MULTIPLY WORD
        - No Arguments
        - INDEX = 72 £
        BEGIN
            store [sp + 1] *:= store [sp];
            IF overflow THEN exception (overflow error) FI;
            sp +:= 1
        END,

        £ MULTIPLY REAL
        - No Arguments
        - INDEX = 74 £
        BEGIN
            PROC multiply real = (REF INT addr1, addr2) VOID:CODE"MMLREAL"EDOC;
            multiply real (store[sp], store[sp + 4]);     £REAL: store [sp + 4] *:= store [s
p] £
            sp +:= 4
        END,

        £ DIVIDE WORD
        - No Arguments
        - INDEX = 76 £
        BEGIN
            store [sp + 1] := store [sp + 1] OVER store [sp];
            IF overflow THEN exception (overflow error) FI;
            sp +:= 1
        END,

        £ DIVIDE REAL
        - No Arguments
        - INDEX = 78 £
        BEGIN
            PROC divide real = (REF INT addr1, addr2) VOID :CODE "MDVREAL" EDOC;
            divide real( store [sp], store [sp + 4]);     £ REAL :store [sp + 4] := store [sp
```

```
+ 1] / store [sp] £
            sp +:= 4
      END,

      £ MODULO WORD
      - No Arguments
      - INDEX = 80 £
      BEGIN
            store [sp + 1] := store [sp + 1] MOD store [sp];
            IF overflow THEN exception (overflow error) FI;
            sp +:= 1
      END,

      £ BUILD SET
      - No Arguments
      - INDEX = 82 £
      BEGIN
            w := store [sp];                    £ Set Member £
            sp +:= 1;
            IF w < 0 ORF w > 127 THEN exception (range error) FI;
            x := sp + ABS (BIN w SHR 4);              £ Address Set Member £
            w := w MOD 16;                      £ Bit Field £
            store [x] := ABS ( BIN store [x] ! ( BIN 1 SHL w))
      END,

      £ TEST IN SET
      - No Arguments
      - INDEX = 84 £
      BEGIN
            w := store [sp + 8];                £ Set Member £
            IF w < 0 ORF w > 127 THEN exception (range error) FI;
            y := store [sp + ABS (BIN w SHR 4)];      £ Word Containing Set Member £
            w := w MOD 16;                      £ Bit Field Index £
            sp +:= 8;
            store [sp] := ABS ( BIN y SHR w & BIN 1)
      END,

      £ COMPARE WORD LESS THAN
      - No Arguments
      - INDEX = 86 £
      ( sp +:= 1;store [sp] := ABS ( store [sp] < store [sp - 1])),

      £ COMPARE WORD EQUAL TO
      - No Arguments
      - INDEX = 88 £
      ( sp +:= 1;store [sp] := ABS ( store [sp] = store [sp - 1])),

      £ COMPARE WORD GREATER THAN
      - No Arguments
      - INDEX = 90 £
      ( sp +:= 1;store [sp] := ABS (store [sp] > store [sp - 1])),

      £ COMPARE WORD GREATER THAN OR EQUAL TO
      - No Arguments
```

```
- INDEX = 92 £
( sp +:= 1;store [sp] := ABS (store [sp] >= store [sp - 1])),

£ COMPARE WORD NOT EQUAL TO
- No Arguments
- INDEX = 94 £
( sp +:= 1;store [sp] := ABS (store [sp] ~= store [sp - 1])),

£ COMPARE WORD LESS THAN OR EQUAL TO
- No Arguments
- INDEX = 96 £
( sp +:= 1;store [sp] := ABS (store [sp] <= store [sp - 1])),

£ COMPARE REAL LESS THAN
- No Arguments
- INDEX = 98 £
BEGIN
      PROC less than = (REF INT addr1, addr2, result) VOID :CODE "MLTREAL" EDOC;
      £ IF bottom real number < top real number THEN stacktop := 1 ELSE stacktop :=
0 £
      less than (store [sp], store [sp + 4], store [sp + 7]);
      sp +:= 7
END,

£ COMPARE REAL EQUAL TO
- No Arguments
- INDEX = 100 £
BEGIN
      PROC equal to = (REF INT addr1, addr2, result) VOID: CODE "MEQREAL" EDOC;
      £IF real numbers equal THEN stacktop := 1 ELSE stacktop := 0 £
      equal to (store [sp], store [sp + 4], store [sp + 7]);
      sp +:= 7
END,

£ COMPARE REAL GREATER THAN
- No Arguments
- INDEX = 102 £
BEGIN
      PROC greater than = (REF INT addr1, addr2, result) VOID:CODE "MGTREAL" EDOC;
      £IF bottom real number > top real number THEN stacktop := 1 ELSE stacktop := 0
£
      greater than (store [sp], store [sp + 4], store [sp + 7]);
      sp +:= 7
END,

£ COMPARE REAL GREATER THAN OR EQUAL TO
- No Arguments
- INDEX = 104 £
BEGIN
     PROC greater than eq = (REF INT addr1, addr2, result) VOID: CODE "MGEREAL" EDOC;
     £ IF bottom real number >= top real number THEN stacktop := 1 ELSE stacktop :=
0£
     greater than eq (store [sp], store [sp + 4], store [sp + 7]);
     sp +:= 7
```

```
END,

£ COMPARE REAL NOT EQUAL TO
  No Arguments
  INDEX = 106 £
BEGIN
      PROC not equal = (REF INT addr1, addr2, result) VOID:CODE "MNEREAL" EDOC;
      £IF real numbers not equal THEN stacktop := 1 ELSE stacktop := 0£
      not equal (store [sp], store [sp + 4], store [sp + 7]);
      sp +:= 7
END,

£ COMPARE REAL LESS THAN OR EQUAL TO
  No Arguments
  INDEX = 108 £
BEGIN
      PROC less than eq = (REF INT addr1, addr2, result) VOID: CODE "MLEREAL" EDOC;
      £IF bottom real number (= top real number THEN stacktop := 1 ELSE stacktop :=
0£
      less than eq (store [sp], store [sp + 4], store [sp + 7]);
      sp +:= 7
END,

£ COMPARE SET EQUAL TO
  No Arguments
  INDEX = 110 £
BEGIN
      x := sp;
      TO 8 WHILE equal := store [x + 8] = store [x] DO x +:= 1 OD;
      sp +:= 15;
      store [sp] := ABS equal
END,

£ COMPARE SET GREATER THAN OR EQUAL TO
  No Arguments
  INDEX = 112 £
BEGIN BOOL zero;
      x := sp;
      TO 8 WHILE zero := ABS ( BIN store [x] & (BIN store [x + 8] NEQ 16rffff)) = 0
DO x +:= 1 OD;
      sp +:= 15;
      store [sp] := ABS zero
END,

£ COMPARE SET NOT EQUAL TO
  No Arguments
  INDEX = 114 £
BEGIN
      x := sp;
      TO 8 WHILE equal := store [x + 8] = store [x] DO x +:= 1 OD;
      sp +:= 15;
      store [sp] := ABS NOT equal
END,
```

```
            £ COMPARE SET LESS THAN OR EQUAL TO
              No Arguments
              INDEX = 116 £
            BEGIN BOOL zero;
                  x := sp;
                  TO 0 WHILE zero := ABS ( BIN store [x + 8] & (BIN store [x] NEQ 16rffff)) = 0
DO x +:= 1 OD;
                  sp +:= 15;
                  store [sp] := ABS zero
            END,

            £ COMPARE STRUCTURE LESS THAN
              1 Argument LENGTH (in words)
              INDEX = 118 £
            BEGIN
                  check struct;
                  store [sp] := ABS (store [y] < store [x] AND NOT equal)
            END,

            £ COMPARE STRUCTURE EQUAL TO
              1 Argument LENGTH (in words)
              INDEX = 120 £
            BEGIN
                  check struct;
                  store [sp] := ABS equal
            END,

            £ COMPARE STRUCTURE GREATER THAN
              1 Argument LENGTH (in words)
              INDEX = 122 £
            BEGIN
                  check struct;
                  store [sp] := ABS (store [y] > store [x] AND NOT equal )
            END,

            £ COMPARE STRUCTURE GREATER THAN OR EQUAL TO
              1 Argument LENGTH (in words)
              INDEX = 124 £
            BEGIN
                  check struct;
                  store [sp] := ABS (store [y] >= store [x] AND NOT equal)
            END,

            £ COMPARE STRUCTURE NOT EQUAL TO
              1 Argument LENGTH (in words)
              INDEX = 126 £
            BEGIN
                  check struct;
                  store [sp] := ABS NOT equal
            END,

            £ COMPARE STRUCTURE LESS THAN OR EQUAL TO
              1 Argument LENGTH (in words)
              INDEX = 128 £
```

```
            BEGIN
                  check struct;
                  store [sp] := ABS (store [y] <= store [x] AND NOT equal )
            END

            OUT ENVIRON SEGATHREE
            ESAC
      OD
OD
£ END OF SEGMENT AND PROGRAM £
```

```
TITLE pcode interpreter segment2
£
 P-CODE INTERPRETER - PART 3 (B) INSTRUCTION INTERPRETATION CODE
 T.E.SHARP - FEBRUARY 1981
£

USING SEGATHREE FROM "PLEASE)U_ENVR)PINTA"

PROC ext = VOID:
£ EXIT (PROCEDURE OR PROGRAM) £
BEGIN
      sp := b + 1;
      b := store [sp + 1];
      q := store [sp + 2];
      pc := store [sp + 3];
      sp := store [sp]
END;


CASE instruction - 64
IN

   £ FUNCTION VALUE
   - 1 Argument KIND
   - INDEX = 130 £
   BEGIN
         CASE ROUND (store [pc]/8) + 1
              IN
                (store [sp -:= 1] := 0),          £ WORD £
                (sp -:= 4)                        £ REAL £
              OUT error ("FNV",uii)
          ESAC;
          pc +:= 1
   END,

   £ JUMP
   - 1 Argument DISTANCE
   - INDEX = 132 £
   ( pc +:=  store [pc] OVER 2),

   £ FALSE JUMP
   - 1 Argument DISTANCE
   - INDEX = 134 £
   BEGIN
         IF continue = 0 THEN exception (result)
         ELSE
              pc +:= IF store [sp] = 0
                     THEN  store[pc] OVER 2          £ FALSE,make the jump £
                     ELSE 1
                     FI;
              sp +:= 1
         FI
   END,
```

```
£ CASE JUMP
- 3 Arguments MIN,MAX-MIN,DISTANCES ("MAX-MIN" words)
- INDEX = 136 £
BEGIN
      w := store [sp] - store [pc];                    £ index - MIN £
      sp +:= 1;
      pc +:= 1;
      IF w < 0 ORF w > store [pc] THEN exception (range error) FI;
      pc +:= w + 1;
      pc +:=  store [pc] OVER 2
END,

£ INITIALISE VARIABLES
- 1 Argument LENGTH (in words)
- INDEX = 138 £
BEGIN
      x := sp;
      TO store [pc] DO store [x] := 0;x +:= 1 OD;
      pc +:= 1
END,

£ CALL PROCEDURE
- 1 Argument DISTANCE
- INDEX = 140 £
BEGIN
      store [sp -:= 1] := pc + 1;              £ Push Return Address £
      pc +:=  store [pc] OVER 2              £ Jump £
END,

£ SYSTEM CALL
- 1 Argument ENTRY
- INDEX = 142 £
£ Declared in Environment Segment £
scl,

£ ENTER PROCEDURE
- 4 Arguments STACKLENGTH,POPLENGTH,LINE,VARLENGTH
- INDEX = 144 £
BEGIN
      IF sp - h LESSUNSIGNED ABS (BIN store [pc] SHR 1) THEN exception (stack limit) FI;
      pc +:= 1;
      store [sp - 1] := g;
      store [sp - 2] := b;
      sp -:= 3;
      store [sp] := sp + ABS (BIN store [pc] SHR 1);          £ POPLENGTH £
      store [sp -:= 1] := store [pc +:= 1];                 £ LINE £
      b := sp;
      sp -:=  ABS (BIN store [pc +:= 1] SHR 1);                              £ VARLENGTH £
      pc +:= 1
END,

£ EXIT PROCEDURE
- No Arguments
- INDEX = 146 £
```

```
   BEGIN
         ext
   END,

   £ ENTER PASCAL PROGRAM
   - 4 Arguments POPLENGTH, LINE, STACKLENGTH, VARLENGTH
   - INDEX = 148 £
   BEGIN
         FILE save := standout;
         store [sp - 1] := g;
         store [sp - 2] := b;
         sp -:= 3;
          store [sp] := sp + ABS (BIN store [pc] SHR 1);          £ POPLENGTH £
         store [sp -:= 1] := store [pc +:= 1];              £ LINE £
         g := b := sp;
         IF sp - h LESSUNSIGNED ABS (BIN store [pc +:= 1] SHR 1) THEN exception (stack limit)
FI;
         sp -:= ABS (BIN store [pc +:= 1] SHR 1);
         pc +:= 1
   END,

   £ EXIT PASCAL PROGRAM
   - No Arguments
   - INDEX = 150 £
   BEGIN
         IF continue = 0 THEN exception (result)
         ELSE
               IF nesting = 1 £ Root Program £
               THEN execution := FALSE;
                     name OF program [nesting ] :=""
               FI;
               idname OF program [nesting] := "";
               new OF program [nesting] := TRUE;
               IF nesting <> 1
               THEN
                     £ Return To Previous Program £
                     FILE save := standout;
                     standout := tty;
                     print (clock," ",name OF program [nesting]," EXIT",newline);
                     standout := save;
                     name OF program [nesting] := "";
                     parseok := TRUE;
                     input program (name OF program [nesting -:= 1]);
                     IF execution := parseok
                     THEN
                           new OF program [nesting] := TRUE;
                           line := store [g];
                           b := g;          £ "ext" assumes b register points to stack bottom £
                           ext;
                           put int param (1, line);      £ LINE address from SCL RUN call still on
top of stack £
                           put int param (2, result);
                           strip params (2)
                     FI
```

```
            FI
        FI
END,

£ 152 £ error ("B6C",uii),
£ 154 £ error ("ENC",uii),
£ 156 £ error ("ETC",uii),
£ 158 £ error ("EXC",uii),
£ 160 £ error ("B6M",uii),
£ 162 £ error ("ENM",uii),
£ 164 £ error ("ETM",uii),
£ 166 £ error ("EXM",uii),
£ 168 £ error ("B6P",uii),
£ 170 £ error ("ENP",uii),
£ 172 £ error ("ETP",uii),
£ 174 £ error ("EXP",uii),

£ POP STACK
- 1 Argument LENGTH
- INDEX = 176 £
( sp +:= ABS (BIN store [pc] SHR 1);pc +:= 1),

£ NEW LINE
- 1 Argument NUMBER
- INDEX = 178 £
( store [b] := store [pc]; pc +:= 1),

£ INCREMENT WORD
- No Arguments
- INDEX = 180 £
( store [ ABS (BIN store [sp] SHR 1) ] +:= 1;sp +:= 1),

£ DECREMENT WORD
- No Arguments
- INDEX = 182 £
( store [ ABS (BIN store [sp] SHR 1) ] -:= 1;sp +:= 1),

£ 184 £ error ("ICL",uii),
£ 186 £ error ("IMN",uii),
£ 188 £ error ("IPC",uii),
£ 190 £ error ("PLB",uii),
£ 192 £ error ("CLP",uii),
£ TRUNCATE REAL
- No Arguments
- INDEX = 194 £
BEGIN
      INT exponent;
      BITS mantissa;
      IF (exponent := store [sp +:= 3] - 128) < 0
      THEN store [sp] := 0                          £ Number < 1 £
      ELIF exponent <= 15
      THEN mantissa := BIN store [sp - 3] SHR (15 - exponent);
           IF store [sp - 3] < 0
           THEN mantissa := mantissa ! maskword [exponent + 1]    £ Propogate the sign bit
```

```
£
            FI;
            store [sp] := ABS mantissa
        ELSE exception (overflow error)
        FI
END,

£ ABSOLUTE VALUE OF WORD
- No Arguments
- INDEX = 196 £
BEGIN
        store [sp] := ABS store [sp];
        IF overflow THEN exception (overflow error) FI
END,

£ ABSOLUTE VALUE OF REAL
- No Arguments
- INDEX = 198 £
BEGIN
        £ IF mantissa is negative THEN negate mantissa FI £
        IF store [sp] < 0
        THEN
            PROC negate real = (REF INT addr1) VOID: CODE "MNGREAL" EDOC;
            negate real (store [sp])        £REAL: store [sp] := -store [sp] £
         FI
END,

£ SUCCESSOR WORD
- No Arguments
- INDEX = 200 £
( store [sp] :== 1),

£ PREDECESSOR WORD
- No Arguments
- INDEX = 202 £
(store [sp] :== 1),

£ CONVERT WORD TO REAL
- No Arguments
- INDEX = 204 £
BEGIN
        BITS mantissa, found;
        INT exponent := 0;
        found := IF ABS (mantissa := BIN store [sp]) <= 0
                 THEN 16r0
                 ELSE first sig digit
                 FI;
        WHILE (mantissa & first sig digit) /= found
        DO
           exponent +:= 1;
           mantissa := mantissa SHL 1
        OD;
        store [sp] := IF store [sp] /= 0 THEN 15 - exponent ELSE 0 FI;
        store [sp] +:= 128;        £ EXCESS 128 £
```

```
            store [sp - 1] := store [sp - 2] := 0;
            store [sp - 3] := ABS mantissa
    END,

    £ TEST EMPTY
    - No Arguments
    - INDEX = 206 £
    ( store [sp] := ABS (store [sp] = 0)),

    £ 208 £ error ("ATR",uii),
    £ 210 £ error ("RTM",uii),
    £ 212 £ error ("DLY",uii),
    £ 214 £ error ("CNT",uii),
    £ 216 £ error ("IOP",uii),
    £ 218 £ error ("STRT",uii),
    £ 220 £ error ("STP",uii),
    £ 222 £ error ("SHP",uii),
    £ 224 £ error ("WAIT",uii)
OUT error ("UNKNOWN",uii)
ESAC

£ END OF SEGMENT £
```

```
*
* P-CODE INTERPRETER - PART 4 - ASSEMBLY LANGUAGE ROUTINES
* T.E. SHARP - FEBRUARY 1981
*
* This file contains subroutines called from an Algol68 program.
* The first Subroutine calls the Fortran library subroutine "prwfile"
* All the other subroutines are double precision floating point routines.
* All parameters for dp floating point routines are addresses.
* R5+3 contains first paramater
*
 REL
 C64R
 D64R
 EXT R5
 EXT R10
 EXT R15
 SUBR ABCD0
ABCD0 DAC **
 LDA R5
 ADD =3
 STA AR6+1
 ADD =1
 STA AR6+2
 LDX R5
 LDA 5,1
 STA BPTR
 LDA R5
 ADD =6
 STA AR6+4
 ADD =1
 STA AR6+5
AR6 CALL PRWFIL
 DAC **
 DAC **
 DAC BPTR
 DAC **
 DAC **
 OCT 0
 JMP* ABCD0
 SUBR NGREAL
* NEGATES A REAL NUMBER
NGREAL DAC **
 LDX R5
 DFLD 3,1*                REG = REAL1
 DFCM                  REG = -REAL1
 DFST 3,1*             REAL1 = - REAL1
 JMP* NGREAL
 SUBR ADREAL
ADREAL DAC **
*ADDS TWO NUMBERS STORES THE RESULT IN SECOND LOCATION
 LDX R5
 DFLD 4,1*               REG = REAL2
 DFAD 3,1*               REG = REAL2 + REAL1
 DFST 4,1*               REAL2 = REAL2 + REAL1
```

```
 JMP* ADREAL
 SUBR SBREAL
SBREAL DAC **
* SUBTRACTS TWO NUMBERS ,STORES RESULT IN SECOND LOCATION
 LDX R5
 DFLD 4,1*                    REG = REAL2
 DFSB 3,1*                    REG = REAL2 - REAL1
 DFST 4,1*                    REAL2 = REAL2 - REAL1
 JMP* SBREAL
 SUBR MLREAL
MLREAL DAC **
* MULTIPLIES TWO NUMBERS ,STORES RESULT IN SECOND LOCATION
 LDX R5
 DFLD 4,1*                    REG = REAL2
 DFMP 3,1*                    REG = REAL2 * REAL1
 DFST 4,1*                    REAL2 = REAL2 * REAL1
 JMP* MLREAL
 SUBR DVREAL
DVREAL DAC **
*DIVIDES TWO NUMBERS ON TOP OF STACK
*STORE ANSWER IN SECOND WORD
 LDX R5
 DFLD 4,1*                REG = REAL2
 DFDV 3,1*                REG = REAL2/REAL1
 DFST 4,1*                REAL2 = REAL2/REAL1
 JMP* DVREAL
 SUBR GTREAL
GTREAL DAC **
 LDX R5
 LDA = 1
 DFLD 4,1*           REG = REAL2
 DFCS 3,1*
 JMP *+3             REAL2 > REAL1
 NOP                 REAL2 = REAL1
 LDA =0              REAL2 < REAL1
 STA 5,1*            RESULT = IF REAL2 > REAL1 THEN 1 ELSE 0
 JMP* GTREAL
 SUBR GEREAL
GEREAL DAC **
 LDX R5
 LDA =1
 DFLD 4,1*           REG = REAL2
 DFCS 3,1*
 JMP *+3             REAL2 > REAL1
 JMP *+2             REAL2 = REAL1
 LDA =0              REAL2 < REAL1
 STA 5,1*            RESULT = 1 IF REAL2 >= REAL1 ELSE 0
 JMP* GEREAL
 SUBR LTREAL
LTREAL DAC **
 LDX R5
 LDA =0
 DFLD 4,1*                REG = REAL2
 DFCS 3,1*
```

```
 JMP *+3                    REAL2 > REAL1
 JMP *+2                    REAL2 = REAL1
 LDA =1                     REAL2 < REAL1
 STA 5,1*                   RESULT = 1 IF REAL2 < REAL1 ELSE 0
 JMP* LTREAL
 SUBR LEREAL
*COMPARES TWO REAL NUMBERS TO SEE IF REAL2 <= REAL1
LEREAL DAC **
 LDX R5
 LDA =0
 DFLD 4,1*          REG = REAL2
 DFCS 3,1*
 JMP *+3            REAL2 > REAL1
 NOP                REAL2 = REAL1
 LDA =1             REAL2 < REAL1
 STA 5,1*           RESULT = 1 IF <= ELSE 0
 JMP* LEREAL
 SUBR EQREAL
* COMPARES TWO REAL NUMBERS TO SEE IF THEY ARE EQUAL
EQREAL DAC **
 LDX R5
 LDA =1
 DFLD 4,1*              REG = REAL2
 DFCS 3,1*
 LDA =0                 REAL2 > REAL1
 JMP *+2                REAL2 = REAL1
 LDA =0                 REAL2 < REAL1
 STA 5,1*               RESULT = 1 IF = ELSE 0
 JMP* EQREAL
 SUBR NEREAL
* COMPARES TWO REAL NUMBERS TO SEE IF THEY ARE EQUAL
NEREAL DAC **
 LDX R5
 LDA =0
 DFLD 4,1*              REG = REAL2
 DFCS 3,1*
 LDA =1                 REAL2 > REAL1
 JMP *+2                REAL2 = REAL1
 LDA =1                 REAL2 < REAL1
 STA 5,1*               RESULT = 1 IF NOT = ELSE 0
 JMP* NEREAL

BPTR DAC **
 END
```

Appendix 4

Microcode Development System User Guide

4.1 Microcode Assembly

Any filenames referred to on this Appendix on the PR1ME will be in the directory 'PLEASE' unless otherwise stated.

When preparing a microcode source file to be assembled it should conform to the following requirements:-

1) The first line should insert the file MAC1 into the main source file:-

$INSERT MAC1

This contains all of the macro definitions required to assemble the microcode.

2) Each microinstruction should begin with the macro INST and end with the macro ENDINS.

3) Each macro should be on a seperate line.

4) Except for these requirements the format of the file is identical to that of [3.2]. However, it should be noted that the listings in Appendix 9 have been post-processed to compact the microcode.

For an example of a microcode source listing, refer to the file

MCOD on the PR1ME.

The file is assembled in the normal manner by typing:-

```
PMA filename
```

This will produce a binary output file prefixed with 'B_' and an assembly listing 'L_'. Assembly takes rather a long time due to the amount of text substitution (about 200 lines a minute). After the file has been successfully assembled the output file should be link loaded:-

```
LOAD
$LO B_filename
LOAD COMPLETE
$SA *filename
$QU
```

If it is required to link load more than one file then the following steps must be taken:-

1) All external labels are specified using the standard PMA 'EXT' and 'ENT' directives.

2) All microcode source files, except for the first, must have an ADDR macro immediately after the INSERT line. The parameter of this macro specifies the starting (Super Sixteen) address of this block of microcode.

e.g. Suppose there were three files F1, F2 and F3. First assemble F1. Examine the L_F1 file and find the address of the

last microinstruction. Call this value x. After this, insert the line 'ADDR x+1' at the start of F2. Assemble F2 and similarly find its last microinstruction address, y say. Insert the line 'ADDR y+1' at the start of F3 and assemble it.

3) When link-loading, type in all the filenames in the order in which they should appear in the microcode memory. After the last filename has been typed in the system should respond with:-

```
LOAD COMPLETE
```

e.g.

```
     LOAD
     $ LO B_F1
     $ LO B_F2
     $ LO B_F3
     LOAD COMPLETE
     $ SA *filename
     $ QU
```

Once a file or a set of files have been assembled and link loaded, they must have their address fields modified. This is done by typing:-

```
    R *ADDRCO
```

The system responds with:-

```
    FILE LINK
    INPUT FILENAME -
```

and the user should type in the previously loaded '*filename' name.

The program then asks for an output file:-

OUTPUT FILENAME -

and the user should type in the name of another file. When the program terminates the output file will be ready to transfer to the Motorola.

4.2 Generation of the Mapping PROM File

This is done by running the MAPGEN program on the PR1ME. Since the Mapping PROM is only 8 bits wide it can only point to addresses in the first 256 microcode-memory locations. Therefore, when preparing the source file containing these microinstructions, the following points should be born in mind:-

1) The microcode for each P-code instruction must begin with a label corresponding to the mnemonic for that particular instruction. These are defined in Appendix 7.

2) The program expects an end of text marker. The best place to put this is at the end of the last P-code instruction (rather than the end of the file). This will speed up the program execution. The marker consists of the line:-

* ENDP *

The program is run by typing:-

R *MAPGEN

The program responds with:-

INPUT FILE NAME -

The user should type in the name of the microcode source file containing the first 256 microinstructions. The program will then produce a file called MAP which can be transferred to the Motorola. If a P-code label is encountered at a microinstruction whose address is greater than 255 then an error message will be output:-

ERROR, PROG ADDRESS TOO LARGE

4.3 Transfer to the Motorola and Programming EPROMs

There are two types of Motorola M6809 based systems which the user may wish to transfer code to. The Exorciser is a standard Motorola Development System and can be used to program 16K or 32K EPROMs. There are also some Motorola based single board systems as described in Appendix 1. These are used as Intelligent Memory but they can also be used to program 16K EPROMs. Whichever machine is to be used, the following steps must be taken:-

1) A Switching Mechanism is required to transfer code, See Fig. 4.1.

2) This has three connections. Connect the PR1ME to the TX (Transmit) line. Connect the Motorola to the RX (Receive) line. Connect the VDU to the remaining line.

3) Switch the TX switch OFF, the RX switch ON and the SEND/RECEIVE switch to SEND+RECEIVE. This puts the Motorola online.

4) The procedure for the Intelligent Memory will be described later. However, for the Exorciser, type:-

LOAD PTRAN;G

This loads and runs the Motorola transfer program which then waits for the PR1ME to start sending. Switch the S/R switch to SEND only and the TX switch to ON. This puts the PR1ME online, type:-

R *MTRAN

The program will ask for the name of the file to be transferred:-

INPUT FILE NAME -

Type in the name of the loaded microcode file. The program will respond with:-

NUMBER OF TRANSFERS -

Type the value '1'. The transfer then starts. A series of hex characters will be output to the VDU.

5) Wait until the transfer stops and then reconnect the Motorola (S/R to SEND+RECEIVE and TX OFF). The data will then be sent and stored in the Exorciser memory as shown in Table 4.1.

This data can then be put onto EPROM using the standard departmental PROM programmer. For each PROM, type:-

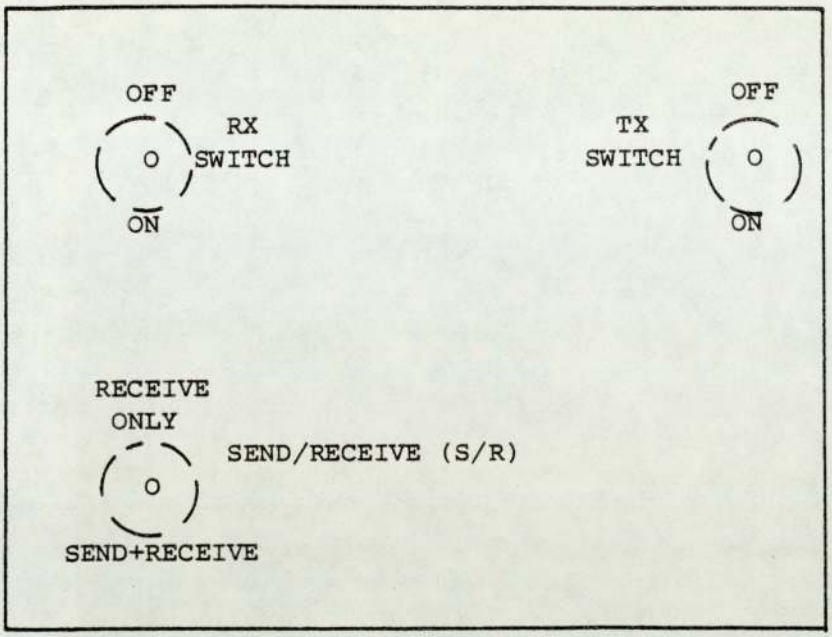


Fig. 4.1 The Transfer Switch Box.

SEQUENCE/PROM NO.	MICROWORD BITS	MEMORY ADDRESS
1	95-88	$3000 - $37FF
2	87-80	$3800 - $3FFF
3	79-72	$4000 - $47FF
4	71-64	$4800 - $4FFF
5	63-56	$5000 - $57FF
6	55-48	$5800 - $5FFF
7	47-40	$6000 - $67FF
8	39-32	$6800 - $6FFF
9	31-24	$7000 - $77FF
10	23-16	$7800 - $7FFF
11	15-8	$8000 - $87FF
12	7-0	$8800 - $8FFF
Error Flag (0 = No error, 1 = Transfer error)		$9000

Table 4.1. Exorciser Memory Assignment after Transfer.

PROM16

on the Exorciser. This will respond with:-

START ADDRESS:

Type in the start address of the PROM data as defined in Table 4.1. The program will then, in a similar manner, request the end address:-

END ADDRESS:

Again, type in the value defined in Table 4.1.

To transfer Mapping PROM data or P-code, the process is similar except for:-

1) The program '*NFPRNT' rather than '*MTRAN' should be run on the PR1ME. This does not request the number of transfers since there is only one.

2) Load the program 'PTRAN' on the Motorola instead of 'PTRAN1'.

3) The code or data will be stored in one contiguous block starting at memory location $3000.

4) To program 32K EPROMs (P-code) the program PROM32 rather than PROM16 should be loaded on the Motorola. Note that this asks for the number of bytes rather than the end address, otherwise it is identical to PROM16.

4.4 Use of The Intelligent Memory

The Intelligent Memory has four commands. They are as follows.

Transfer (T) Command

This is used to transfer data from the PR1ME. It has the format:-

T <<start address>> <<number of sequences>>

<<start address>> is a four digit hex number which specifies the start address to which the data transferred from the PR1ME should be stored.

<<number of sequences>> is also a four digit hex number and is optional. It specifies the number of sets of PROM data to be sent by MTRAN. On the Exorciser this value is automatically twelve. However, the Intelligent Memory only has enough memory for three of these sequences. Therefore, this value should be '0003'. Accordingly, since there are twelve EPROMs, the MTRAN program should be given the value 4 when it requests the number of transfers. MTRAN will pause at the end of each transfer. It can be re-started by typing CTRL and B at the terminal. The sequences will be stored in successive 2K blocks. When the current transfer is completed, the Intelligent Memory will pause and wait to be re-connected. It can be restarted by typing CTRL and T. It will then respond with:-

TRANSFER COMPLETE

Unless there is an error in which case it will output:-

TRANSFER ERROR, INVALID CHARACTER

If the transfer has failed or:-

TRANSFER ERROR, MEMORY OVERFLOW

If the user has attempted to input code beyond memory location $1F80 which the Intelligent Memory uses itself. Other than this, the transfer from the PR1ME is identical to the Exorciser transfer. If the number of sequences parameter is ommitted it is assumed to be one. This allows the transfer of Mapping PROM data and P-code.

e.g. T 0400

Would transfer one sequence of data to memory locations $400 onwards.

T 0000 0003

Would transfer three sequences of data to locations $0000, $0800 and $1000.

PROM Programmer (P) Command

This command is used to program 16K EPROMS. The user types 'P' at the terminal. A series of prompts are then issued. These have the same format as the PROM16 program on the Exorciser.

Read Memory (R) Command

This command is used to display the contents of memory. It has the format:-

R <<start address>> <<end address>>

<<start address>> is a four digit hex number specifying the first location to be output.

<<end address>> is a four digit hex number which specifies the last location to be displayed.

Go (G) Command

This command is used to start the Intelligent Memory. It is invoked by typing 'G'. It will then output the message:-

INTELLIGENT MEMORY MODE
PRESS RESET TO EXIT

Two occurences will then take place:-

1) The Super Sixteen processor will be reset.

2) The Motorola processor will then respond to all memory requests from the Super Sixteen. It should be noted that the Motorola is now dedicated to the Super Sixteen. The only way that an exit can be made is by pressing the reset button on the

M6809 board. This does not alter any memory data which can subsequently be examined with the 'R' command.

Appendix 5

Development System Source Listings

```
*
* PRIME AMD MICRO CODE MACRO ASSEMBLER
* VERSION 3, 2/10/82
   NLST
   REL
    ORG $0000
*
*
*    MACRO DEFINITION FILE FOR 16-BIT COMPUTER
*    USING AM2901A, AM2903, AM2904 & AM2910
WORD       XSET 6            WORD LENGTH = 96 (6*16)
DM1        XSET 0
A$         XSET 0            DEFAULT SETTING FOR A$ EQUATES
A$0        XSET 0
A$1        XSET 1
00$        XSET 3            FOR 00$ EQUATES
0$         XSET 1            FOR 0$ EQUATES
0$0        XSET 0
0$1        XSET 1
ADDRESS    XSET -1
BASE       XSET 0
*
*
INST MAC
(0) BSS 0
* INSERT THE DEFAULT VALUES
ADDRESS    XSET ADDRESS + 1
W1 XSET $4000
W2 XSET 0
W3 XSET $0001
W4 XSET 0
W5 XSET 0
W6 XSET $3000
EXTFLAG XSET 0
CNTFLG  XSET 0
JMPFLG  XSET 0
    ENDM
*
*
OUTPUT     MAC
           IFZ CNTFLG
W7         XSET (W6 .AND. $7FFF)/WORD
           ELSE                                    IMMEDIATE, IREG OR LOAD COUNTER
W7         XSET W6
           ENDC
           SAY ((ADDRESS.RS.12)+1)(((ADDRESS.RS.8).AND.$F)+1);
           (((ADDRESS.RS.4).AND.$F)+1)((ADDRESS.AND.$F)+1)!: ;
           ((W1.RS.12)+1)(((W1.RS.8).AND.$F)+1) ;
           (((W1.RS.4).AND.$F)+1)((W1.AND.$F)+1) ;
           ((W2.RS.12)+1)(((W2.RS.8).AND.$F)+1) ;
           (((W2.RS.4).AND.$F)+1)((W2.AND.$F)+1) ;
           ((W3.RS.12)+1)(((W3.RS.8).AND.$F)+1) ;
           (((W3.RS.4).AND.$F)+1)((W3.AND.$F)+1) ;
           ((W4.RS.12)+1)(((W4.RS.8).AND.$F)+1) ;
```

```
          ((((W4.RS.4).AND.$F)+1)((W4.AND.$F)+1) ;
          ((W5.RS.12)+1)(((W5.RS.8).AND.$F)+1) ;
          (((W5.RS.4).AND.$F)+1)((W5.AND.$F)+1) ;
          ((W7.RS.12)+1)(((W7.RS.8).AND.$F)+1) ;
          (((W7.RS.4).AND.$F)+1)((W7.AND.$F)+1)
          ENDM
*
*
EXTPUT    MAC
          SAY ((ADDRESS.RS.12)+1)(((ADDRESS.RS.8).AND.$F)+1);
          (((ADDRESS.RS.4).AND.$F)+1)((ADDRESS.AND.$F)+1)!: ;
          ((W1.RS.12)+1)(((W1.RS.8).AND.$F)+1) ;
          (((W1.RS.4).AND.$F)+1)((W1.AND.$F)+1) ;
          ((W2.RS.12)+1)(((W2.RS.8).AND.$F)+1) ;
          (((W2.RS.4).AND.$F)+1)((W2.AND.$F)+1) ;
          ((W3.RS.12)+1)(((W3.RS.8).AND.$F)+1) ;
          (((W3.RS.4).AND.$F)+1)((W3.AND.$F)+1) ;
          ((W4.RS.12)+1)(((W4.RS.8).AND.$F)+1) ;
          (((W4.RS.4).AND.$F)+1)((W4.AND.$F)+1) ;
          ((W5.RS.12)+1)(((W5.RS.8).AND.$F)+1) ;
          (((W5.RS.4).AND.$F)+1)((W5.AND.$F)+1) ;
          XX XX
          ENDM
*
*
*
*
* SET ADDRESS MACRO
ADDR      MAC
ADDRESS XSET (1)-1
BASE      XSET (1)*6
          ENDM
*
*
ENDINS    MAC
* IF CURRENT ADDRESS IS $00FE THEN MAKE
* $00FE = JUMP TO $0100 (UNLESS JMAP OR JUMP ALREADY)
* $00FF = MAP ERROR CODE
* $0100 = CURRENT MICROINSTRUCTION MOVED DOWN FROM $00FE
          IF (ADDRESS.NE.$00FE)  GO TO CALLPD
FFLAG     XSET 0
* TEST FOR JMAP
          IF ((W5.AND.$000F).EQ.2)  GO TO GENMAPF
* TEST FOR UNCONDITIONAL JUMP
     IF (((W5.AND.$000F).OR.(W1.AND.$0800)).EQ.$0803)  GO TO GENMAPF
* SAVE W1-W6 ETC.
FFLAG     XSET 1
X1        XSET W1
X2        XSET W2
X3        XSET W3
X4        XSET W4
X5        XSET W5
X6        XSET W6
C1        XSET CNTFLG
```

```
E1        XSET EXTFLAG
J1        XSET JMPFLG
* INSERT THE CODE FOR A JUMP
ADDRESS   XSET ADDRESS-1
          INST
            ALU YBUS PASS
            AB
            PCUNOP
            AM2904
            DATAPATH
            MEMCONT
            CONTROL
            JUMP *+12
          SAY 00FE!: JUMP TO $0100
GENMAPF   XSET DM1 = 0                                  DUMMY STATEMENT
          PUTDATA
* INSERT THE CODE FOR LOCATION $00FF
* JUMP TO MAP FAULTY CODE
          INST
            ALU YBUS PASS
            AB
            PCUNOP
            AM2904
            DATAPATH
            MEMCONT
            CONTROL
            XJUMP MAPFAULTY
            SAY * ARRIVED HERE IF MAP FAULTY, JUMP TO FAULT-TOLERANT CODE.
            PUTDATA
            IF (FFLAG .EQ. 0)  GO TO STP1
* INSERT CURRENT MICROINSTRUCTION AT $0100
* RETREIVE FLAGS
ADDRESS   XSET ADDRESS+1
W1        XSET X1
W2        XSET X2
W3        XSET X3
W4        XSET X4
W5        XSET X5
W6        XSET X6
CNTFLG    XSET C1
JMPFLG    XSET J1
EXTFLAG   XSET E1
CALLPD    PUTDATA
STP1   XSET DM1 = 0                    DUMMY STATEMEMT
       ENDM
PUTDATA MAC
* WRITE 6 NEW WORDS TO OUTPUT FILE CONTAINING CURRENT MICROINSTRUCTION
       IFZ EXTFLAG
       DATA W1,W2,W3,W4,W5
       ENDC
* OUTPUT THE CURRENT ADDRESS AND IT'S CONTENTS BY CALLING "OUTPUT" MACRO.
* THE PARAMETERS ARE A LOOKUP TABLE.
       IFZ EXTFLAG
          IFZ JMPFLG
```

```
            DATA W6
          ELSE
W6          XSET (W6 .OR. $8000) + BASE
            DATA W6
          ENDC
        OUTPUT 0 1 2 3 4 5 6 7 8 9 A B C D E F
      ELSE
        EXTPUT 0 1 2 3 4 5 6 7 8 9 A B C D E F
      ENDC
      ENDM
*
*
*           ************************************************
*           *   MACROS FOR AM2910 SEQUENCER                *
*           ************************************************
*
*
*
* NOTE THAT ALL THE MACROS PRECEDED BY 'X' ARE THE SAME AS
* THE OTHER MACROS EXCEPT THAT THEY REFER TO AN EXTERNAL ADDRESS
* ALSO NOTE THAT THEY MUST ALWAYS BE THE LAST MACRO IN ANY MICROINSTRUCTION
* DEFINITION. THIS IS BECAUSE THEY FORM THE DATA STATEMENTS, RATHER THAN THE
* 'ENDINS' MACRO.
*
*
JUMP    MAC
* UNCONDITIONAL JUMP PL
W1      XSET W1 .OR. $0800
W5      XSET W5 .OR. $3
W6      XSET (1)
W6      XSET W6/1
JMPFL6  XSET 1
        ENDM
XJUMP   MAC
W1      XSET W1 .OR. $0800
W5      XSET W5 .OR. $3
        EXTLAB (1)
        ENDM
JUMPX   MAC
W1      XSET W1 .OR. $0800
W5      XSET W5 .OR. $3
        LABEXT (1)
        ENDM
JZ      MAC
* JUMP ZERO
W1      XSET W1 .AND. $F7FF
W5      XSET W5 .AND. $FFF0
        ENDM
*
CJS     MAC
* COND JSB PL
W1      XSET W1 .AND. $F7FF
W5      XSET W5 .AND. $FFF0 .OR. $1
W6      XSET (1)
```

```
W6    XSET W6/1
JMPFLG  XSET 1
      ENDM
XCJS  MAC
W1    XSET W1 .AND. $F7FF
W5    XSET W5 .AND. $FFF0 .OR. $1
      EXTLAB (1)
      ENDM
*
* UNCONDITIONAL JSB PL
JSB       MAC
W1        XSET W1 .OR. $0800
W5        XSET W5 .AND. $FFF0 .OR. $1
W6        XSET (1)
W6    XSET W6/1
JMPFLG     XSET 1
          ENDM
XJSB      MAC
W1        XSET W1 .OR. $0800
W5        XSET W5 .AND. $FFF0 .OR. $1
          EXTLAB (1)
          ENDM
*
* JUMP MAP
JMAP      MAC
W1        XSET W1 .OR. $0800
W5        XSET W5 .AND. $FFF0 .OR. $2
          ENDM
*
* CONDITIONAL JUMP PL
CJP       MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $3
W6        XSET (1)
W6    XSET W6/1
JMPFLG     XSET 1
          ENDM
XCJP      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $3
          EXTLAB (1)
          ENDM
CJPX      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR.$3
          LABEXT (1)
          ENDM
*
* PUSH/COND LOAD CNTR
PUSH      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $4
W6        XSET (1)
W6        XSET W6/1
```

```
CNTFL6    XSET 1
          ENDM
*
* PUSH AND LOAD CNTR
PHLC      MAC
W1        XSET W1 .OR. $0800
W5        XSET W5 .AND. $FFF0 .OR. $4
W6        XSET (1)
W6        XSET W6/1
CNTFL6    XSET 1
          ENDM
*
* COND JSB R/PL
JSRP      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $5
W6        XSET (1)
W6    XSET W6/1
JMPFL6     XSET 1
          ENDM
XJSRP     MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $5
          EXTLAB (1)
          ENDM
*
* COND JUMP VECTOR
CJV       MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $6
          ENDM
*
* UNCONDITIONAL JUMP VECTOR
JMPV      MAC
W1        XSET W1 .OR. $0800
W5        XSET W5 .AND. $FFF0 .OR.$6
          ENDM
*
* COND JUMP R/PL
JRP       MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $7
W6        XSET (1)
W6    XSET W6/1
JMPFL6     XSET 1
          ENDM
XJRP      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $7
          EXTLAB (1)
          ENDM
JRPX     MAC
W1       XSET W1 .AND. $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $7
```

```
         LABEXT (1)
         ENDM
*
* REPEAT LOOP, CNTR ()0
RPCT     MAC
W1       XSET W1 .AND. $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $8
         ENDM
*
* REPEAT PL, CNTR ()0
RPCT     MAC
W1       XSET W1 .AND. $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $9
W6       XSET (1)
W6    XSET W6/1
JMPFLG     XSET 1
         ENDM
XRPCT    MAC
W1       XSET W1 .AND $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $9
         EXTLAB (1)
         ENDM
*
* COND RTN
CRTN     MAC
W1       XSET W1 .AND. $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $A
         ENDM
*
* UNCONDITIONAL RETURN
RTN      MAC
W1       XSET W1 .OR. $0800
W5       XSET W5 .AND. $FFF0 .OR. $A
         ENDM
*
* COND JUMP PL & POP
CJPP     MAC
W1       XSET W1 .AND. $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $B
W6       XSET (1)
W6      XSET W6/1
JMPFLG    XSET 1
         ENDM
XCJPP    MAC
W1       XSET W1 .AND. $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $B
         EXTLAB (1)
         ENDM
CJPPX    MAC
W1       XSET W1 .AND. $F7FF
W5       XSET W5 .AND. $FFF0 .OR. $B
         LABEXT (1)
         ENDM
* UNCONDITIONAL JUMP PL & POP
```

```
JPOP      MAC
W1        XSET W1 .OR. $0800
W5        XSET W5 .AND. $FFF0 .OR. $B
W6        XSET (1)
W6        XSET W6/1
JMPFLG    XSET 1
          ENDM
*
* LOAD CNTR & CONTINUE
LDCT      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $C
W6        XSET (1)
W6        XSET W6/1
CNTFLG    XSET 1
          ENDM
* LOAD COUNTER WITH ADDRESS (I.E. DIVIDE ADDRESS BY 6)
LDCTA     MAC
          LDCT (1)
W6        XSET (W6+BASE)/WORD
          ENDM
*
* TEST END LOOP
LOOP      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $D
          ENDM
*
* CONTINUE
CONT      MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $E
          ENDM
*
* THREE-WAY BRANCH
TWB       MAC
W1        XSET W1 .AND. $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $F
W6        XSET (1)
W6        XSET W6/1
JMPFLG     XSET 1
          ENDM
XTWB      MAC
W1        XSET W1 .AND $F7FF
W5        XSET W5 .AND. $FFF0 .OR. $F
          EXTLAB (1)
          ENDM
*
* EXTERNAL LABEL MACRO.
EXTLAB    MAC
          EXT (1)
EXTFLAG   XSET 1
          DATA W1,W2,W3,W4,W5
          XAC (1)
```

```
          ENDM
LABEXT    MAC
EXTFLAG   XSET 1
          DATA W1,W2,W3,W4,W5
          XAC (1)
          ENDM
*
*
*                  ********************************************************
*                  *         LOAD INTERRUPT COUNTER MACRO.                *
*                  ********************************************************
*
INTLOAD   MAC
CNTFLG    XSET 1
W1        XSET W1.AND.$BFFF
W6        XSET (1)
          ENDM
*
*
*
*
*            ************************************************
*            *  MACRO DEFINITIONS FOR AM2903 ALU            *
*            ************************************************
*
*            THE ALU MACRO IS OF THE FOLLOWING FORMAT :-
*            ALU DESTINATION CONTROL, FUNCTION
*
*
*            ****************************************************
*            *   EQUATES FOR ALU DESTINATION CONTROL            *
*            ****************************************************
*
A$ADR     XSET $0                  ARITHMETIC SHIFT DOWN, RESULTS INTO RAM
A$LDR     XSET $1                  LOGICAL SHIFT DOWN, RESULTS INTO RAM
A$ADRQ    XSET $2                  ARITHMETIC SHIFT DOWN, REULTS INTO RAM AND Q
A$LDRQ    XSET $3                  LOGICAL SHIFT DOWN, RESULTS INTO RAM AND Q
A$RPT     XSET $4                  RESULTS INTO RAM, GENERATE PARITY
A$LDQP    XSET $5                  LOGICAL SHIFT DOWN Q,GENERATE PARITY
A$QPT     XSET $6                  RESULTS INTO Q, GENERATE PARITY
A$RQPT    XSET $7                  RESULTS INTO RAM AND Q, GENERATE PARITY
A$AUR     XSET $8                  ARITHMETIC SHIFT UP, RESULTS INTO RAM
A$LUR     XSET $9                  LOGICAL SHIFT UP, RESULTS INTO RAM
A$AURQ    XSET $A                  ARITHMETIC SHIFT UP, RESULTS INTO RAM AND Q
A$LURQ    XSET $B                  LOGICAL SHIFT UP, RESULTS INTO RAM AND Q
A$YBUS    XSET $C                  RESULTS TOP YBUS ONLY
A$LUQ     XSET $D                  LOGICAL SHIFT UP Q
A$SINEX   XSET $E                  SIGN EXTEND
A$REG     XSET $F                  RESULTS TO RAM, SIGN EXTEND
*
*
*            ************************************************
*            *  EQUATES FOR ALU FUNCTIONS                   *
*            ************************************************
```

```
*
A$HIGH    XSET $0                   FI = 1
A$SUBR    XSET $1                   SUBTRACT R FROM S
A$SUBS    XSET $2                   SUBTRACT S FROM R
A$ADD     XSET $3                   ADD R AND S
A$PASS    XSET $4                   PASS S
A$COMPLS  XSET $5                   2'S COMPLEMENT OF S
A$PASSR   XSET $6                   PASS R
A$COMPLR  XSET $7                   2'S COMPLEMENT OF R
A$LOW     XSET $8                   FI = 0
A$NOTRS   XSET $9                   COMPLEMENT R AND WITH S
A$EXNOR   XSET $A                   EXCLUSIVE NOR R WITH S
A$EXOR    XSET $B                   EXCLUSIVE OR R WITH S
A$AND     XSET $C                   AND R WITH S
A$NOR     XSET $D                   NOR R WITH S
A$NAND    XSET $E                   NAND R WITH S
A$OR      XSET $F                   OR R WITH S
*
*
*           ***************************************************
*           *    ALU MACRO                                    *
*           ***************************************************
*
ALU MAC
W1      XSET W1 .AND. $FF87 .OR. (A$(1).LS. 3)          ALU DESTINATION CONTROL
W1      XSET W1 .AND. $FFF8 .OR. (A$(2).RS. 1)          ALU FUNCTION WORD1
W2      XSET W2 .AND. $7FFF .OR. (A$(2).LS. 15)  ALU FUNCTION WORD2
        ENDM
*
*
*            ***************************************************
*            *  ALU OPERAND SOURCES                            *
*            ***************************************************
*
* R = RAM A, S = RAM B
AB      MAC
W1      XSET W1 .AND. $FD7F
W2      XSET W2 .AND. $8FFF
        ENDM
*
* R = RAM A, S = RAM B
ADB     MAC
W1      XSET W1 .AND. $FDFF .OR. $0080
W2      XSET W2 .AND. $8FFF
        ENDM
*
* R = RAM A, S = Q
AQ      MAC
W1      XSET W1 .AND. $FDFF
W2      XSET W2 .OR. $4000
        ENDM
*
* R = DA, S = RAM B
DAB     MAC
```

```
W1      XSET W1 .OR. $0200 .AND. $FF7F
W2      XSET W2 .AND. $BFFF
        ENDM
*
* R = DA, S = DB
DADB    MAC
W1      XSET W1 .OR. $0280
W2      XSET W2 .AND. $BFFF
        ENDM
*
* R = DA, S = Q
DAQ     MAC
W1      XSET W1 .OR. $0200
W2      XSET W2 .OR. $4000
        ENDM
*
*
*           ************************************************
*           *   OUTPUT Y ENABLE MACRO                      *
*           ************************************************
*
*
OEY     MAC
W1      XSET W1 .AND. $FEFF
        ENDM
YOFF    MAC
W1     XSET W1 .OR. $0100
        ENDM
*
*
*           ************************************************
*           *    SPECIAL FUNCTIONS FOR AM2903              *
*           ************************************************
*
*           TO USE THE SPECIAL FUNCTIONS, THE DESTINATION
*           CONTROL MUST NOT BE AQ OR DAQ
*
*           ************************************************
*           *   SPECIAL FUNCTION EQUATES                   *
*           ************************************************
*
*
A$USMUL    XSET $00                  UNSIGNED MULTIPLY
A$TCMUL    XSET $20                  TWO'S COMPLEMENT MULTIPLY
A$INCTWO   XSET $40                  INCREMENT BY ONE OR TWO
A$SMTC     XSET $50                  SIGN-MAGNITUDE/TWO'S COMPLEMENT
A$TCMLS    XSET $60                  TWO'S COMPLEMENT MULT. LAST STEP
A$SLN      XSET $80                  SINGLE LENGTH NORMALISE
A$DLN      XSET $A0                  DOUBLE LENGTH NORMALISE AND 1'ST DIVIDE OP.
A$TCDIV    XSET $C0                  TWO'S COMPLEMENT DIVIDE
A$TCDC     XSET $E0                  TWO'S COMPLEMENT DIVISION CORRECTION
*
*
*           ************************************************
```

```
*            *  SPECIAL FUNCTION MACRO DEFINITION                *
*            ***************************************************
*
*
SPF14    MAC
W1       XSET W1 .AND. $FF80 .OR. (A$(1).RS. 1)
W2       XSET W2 .AND. $7FFF .OR. (A$(2).LS. 15)
         ENDM
*
*
*            ***************************************************
*            *        DEFINITIONS FOR AM2901 PROGRAM           *
*            *            CONTROL UNIT (PCU)                   *
*            ***************************************************
*            PCU REGISTER DEFINITIONS :-
*
*            R0 = PC        PROGRAM COUNTER
*            R1 = SP        STACK POINTER
*            R2 = SPLL      STACK POINTER LOWER LIMIT
*            R3 = SPUL      STACK POINTER UPPER LIMIT
*            R4 = 2         CONSTANT 2
*            R5 = 4         CONSTANT 4
*
*
*            ***************************************************
*            *   EQUATES FOR PCU DEFINITIONS                   *
*            ***************************************************
*
*
O$QEU      XSET 0              Q REG = ZERO, B RAM = ONE DEFAULT
*
*
*            ***************************************************
*            *  EQUATES FOR PCU FUNCTIONS                      *
*            ***************************************************
*
*
A$SUB      XSET 1
*
*
*            ***************************************************
*            *   EQUATES FOR SOURCE CONTROL                    *
*            ***************************************************
*
*
O$PCUAQ    XSET '0
O$PCUAB    XSET '1
O$PCUZQ    XSET '2
O$PCUZB    XSET '3
O$PCUZA    XSET '4
O$PCUDA    XSET '5
O$PCUDQ    XSET '6
O$PCUDZ    XSET '7
*
```

```
*
*            ***********************************************
*            *    EQUATES FOR PCU A-RAM                    *
*            ***********************************************
*
*
A$A0       XSET '0
A$A1       XSET '1
A$A2       XSET '2
A$A3       XSET '3
A$A4       XSET '4
A$A5       XSET '5
A$A6       XSET '6
A$A7       XSET '7
*
*
*            ***********************************************
*            *      EQUATES FOR PCU B-RAM                  *
*            ***********************************************
*
*
A$B0       XSET '0
A$B1       XSET '1
A$B2       XSET '2
A$B3       XSET '3
A$B4       XSET '4
A$B5       XSET '5
A$B6       XSET '6
A$B7       XSET '7
A$PHREG    XSET A$B3
*
*
*            ***********************************************
*            *  PCU MACRO DEFINITION                       *
*            ***********************************************
*
*
PCU        MAC
W2         XSET W2 .AND. $FFFE .OR. (0$(1).AND. $0001)              PCUI7
W3         XSET W3 .AND. $7FFF .OR. (A$(2).LS. 15)                  PCUI3
W3         XSET W3 .AND. $8FFF .OR. ((0$(3).AND. $0007) .LS. 12)     PCUI2 - PCUI0
W3         XSET W3 .AND. $F1FF .OR. ((A$(4).AND. $0007) .LS. 9)      RAM-A
W3         XSET W3 .AND. $FE3F .OR. ((A$(5).AND. $0007) .LS. 6)      RAM-B
           ENDM
*
PCUNEXT    MAC
           PCU 1 0 PCUAB A4 B0              PC = PC + 2
           ENDM
*
PCUPUSH    MAC
           PCU 1 1 PCUAB A4 B1              SP = SP - 2
           ENDM
*
PCUPOP     MAC
```

```
        PCU 1 0 PCUAB A4 B1             SP = SP + 2
        ENDM
*
PCUJUMP MAC
        PCU 1 0 PCUDZ A0 B0             PC = D
        ENDM
*
PCUTR2  MAC
        PCU 1 0 PCUDA A4 B0             PC = TRE6 + 2
        ENDM
*
PCUNOP  MAC
        PCU 1 0 PCUZB A0 B0             PC TO OUTPUT
        ENDM
*
PCUSP   MAC
        PCU 1 0 PCUZB A1 B1             SP TO OUTPUT
        ENDM
*
PCUDEC4 MAC
        PCU 1 1 PCUAB A5 B0             PC = PC - 4
        ENDM
*
*
*           ************************************************
*           *    MACRO DEFINITIONS FOR AM2904 RELATED      *
*           *           CONTROL BITS                       *
*           ************************************************
*
*
*            AM2904 BIT DEFINITIONS ARE AS FOLLOWS :-
*
*            BITS 95-44 = DON'T CARES
*            BIT 43 = SHIFT ENABLE
*            BITS 42-35 = GENERAL PURPOSE CONTROL BITS
*            BIT 34 = OUT EN CONDITIONAL TEST
*            BIT 33 = ENABLE ZERO
*            BIT 32 = ENABLE CARRY
*            BIT 31 = ENABLE SIGN
*            BIT 30 = ENABLE OVERFLOW
*            BIT 29 = ENABLE MACHINE STATUS
*            BIT 28 = ENABLE MICRO STATUS
*            BITS 27-26 = CARRY OUT CONTROL
*            BITS 25-20 = CONDITIONAL BRANCH TEST
*
*
O$SHIFTEN  XSET 0               SHIFT ENABLE
O$OECT     XSET 0               OUT EN CONDITIONAL TEST
O$EZ       XSET 0               ENABLE ZERO
O$EC       XSET 0               ENABLE CARRY
O$ES       XSET 0               ENABLE SIGN
O$EOVR     XSET 0               ENABLE OVERFLOW
O$CEM      XSET 0                ENABLE MACHINE STATUS
O$CEU      XSET 0               ENABLE MICRO STATUS
```

```
*
*
AM2704   MAC
* SET THE DEFAULT VALUES FIRST
W4       XSET W4 .OR. $0307
W5       XSET W5 .OR. $F000
W4       XSET W4 .AND. (0$(1).LS. 11 .OR. $F7FF)
W4       XSET W4 .AND. (0$(2).LS. 2 .OR. $FFFB)
W4       XSET W4 .AND. (0$(3).LS. 1 .OR. $FFFD)
W4       XSET W4 .AND. (0$(4).OR. $FFFE)
W5       XSET W5 .AND. (0$(5).LS. 15 .OR. $7FFF)
W5       XSET W5 .AND. (0$(6).LS. 14 .OR. $BFFF)
W5       XSET W5 .AND. (0$(7).LS. 13 .OR. $DFFF)
W5       XSET W5 .AND. (0$(8).LS. 12 .OR. $EFFF)
         ENDM
*
*
*            ****************************************
*            * EQUATES FOR TEST MACRO SOURCE         *
*            ****************************************
*
*
MICRO      XSET 0                MICRO STATUS REGISTER
MICRO2     XSET 1                MICRO STATUS REGISTER
MACHINE    XSET 2                MACHINE STATUS REGISTER
DIRIN      XSET 3                DIRECT INPUTS
*
*
*              ************************************************
*              *  EQUATES FOR TEST MACRO FUNCTION             *
*              ************************************************
*
*
UAEQB      XSET 5                UNSIGNED A = B
UANEB      XSET 4                UNSIGNED A NOT EQUAL TO B
UAGEB      XSET $8               UNSIGNED A >= B
UALSB      XSET $A               UNSIGNED A < B
UAGRB      XSET $C               UNSIGNED A > B
UALEB      XSET $D               UNSIGNED A <= B
SAEQB      XSET 5                SIGNED A = B
SANEB      XSET 4                SIGNED A NOT EQUAL TO B
SAGEB      XSET 2                SIGNED A >= B
SALSB      XSET 3                SIGNED A < B
SAGRB      XSET 0                SIGNED A > B
SALEB      XSET 1                SIGNED A <= B
DLOAD      XSET 7                DIRECT INPUTS -> MACHINE STATUS REGISTER
POSITIVE   XSET $E               SIGN BIT = 0
NEGATIVE   XSET $F               SIGN BIT = 1
*
*
*            ************************************************
*            * TEST BITS MACRO DEFINITION                   *
*            ************************************************
*
```

```
*
TEST     MAC
W5       XSET W5 .AND. $FC0F .OR. (((1) .AND. $3) .LS. 8)
W5       XSET W5 .OR. (((2) .AND. $F) .LS. 4)
         ENDM
*
*
*              ***********************************************
*              *      EQUATES FOR TESTF MACRO                *
*              ***********************************************
*
*
A=B      XSET 0           A-PORT EQUALS B-PORT FROM AM25LS251
DA4      XSET 1           DA BUS BIT 4
DA5      XSET 2           DA BUS BIT 5
DA6      XSET 3           DA BUS BIT 6
DA7      XSET 4           DA BUS BIT 7
PCUZERO  XSET 5           PCU RESULT EQUALS ZERO
PCUNZ    XSET 6           PCU RESULT NOT EQUAL TO ZERO
INTERRUPT  XSET 7         TIMER INTERRUPT
*
*
*               *****************************************************
*               *    TESTF MACRO                                    *
*               *****************************************************
*
* ALLOWS DIRECT ENTRY OF CONSTANT PARAMETERS
TESTF    MAC
W5       XSET W5 .AND. $FC0F .OR. (((1) .AND. $003F) .LS. 4)
         ENDM
*
*
*            ************************************************
*            *   EQUATES FOR CARRY OUT CONTROL              *
*            ************************************************
*
*
A$COEQ0     XSET 0              CARRY-OUT = 0
A$COEQ1     XSET 1              CARRY-OUT = 1
A$COEQCI    XSET 2              CARRY-OUT = CARRY-IN
A$COEQST    XSET 3              CARRY-OUT = CARRY OF STATUS REGISTER
*
*
*            ************************************************
*            *    CARRY-OUT CONTROL MACRO DEFINITION        *
*            ************************************************
*
*
CARRYCTL  MAC
W5       XSET W5 .AND. $F3FF .OR. ((A$(1).AND. $3) .LS. 10)
         ENDM
*
*
*              ***********************************************
```

```
*             *     REGISTER MUX SELECT                          *
*             ************************************************
*
*
RTB       MAC
W1        XSET W1 .OR. $8000
          ENDM
*
*
*              ********************************************
*              *    EQUATES FOR DATAPATH DEFINITION       *
*              ********************************************
*
*
A$ZZI       XSET 1          Z REG TO ZI REG
O$ENTREG    XSET 0          ENABLE TRANSFER REGISTER
A$LDTREG    XSET 1          LOAD TRANSFER REGISTER
O$ENCTR     XSET 0          I-REG EN CTR
A$INC       XSET 1          I-REG INC/DEC
O$PCUY     XSET 0          PCU TRANSCEIVER TO Y-BUS
O$YMAR     XSET 3          PCU TRANSCEIVER TO MAR BUS
O$PCUMAR   XSET 1          PCU TRANSCEIVER CHIP DISABLE
A$LDMAR     XSET 1          LOAD MAR
A$LDD       XSET 1          LOAD D REG
A$LDI       XSET 1          LOAD PL REG INTO I-REG
O$ENZ0      XSET 0          ENABLE Z0 TO DA
O$PSW       XSET 0          ENABLE PSW
O$SHTCNTEN XSET 0          SHIFT CNT 2910 ADDR
O$BRIEN     XSET 0          BRANCH INSTRUCTION ENABLE
*
*
*             ************************************************
*             *  DATAPATH MACRO DEFINITION                   *
*             ************************************************
*
*
DATAPATH MAC
W2        XSET W2 .AND. $C001 .OR. $291E              SET THE DEFAULT VALUES
W1        XSET W1 .AND. $EFFF .OR. ((A$(1).AND. $1) .LS. 12)
W2        XSET W2 .AND. (O$(2).LS.13 .OR. $DFFF)
W2        XSET W2 .OR. (A$(3).LS. 12 .AND. $1000)
W2        XSET W2 .AND. (O$(4).LS. 11 .OR. $F7FF)
W2        XSET W2 .OR. (A$(5).LS. 10 .AND. $0400)
W2        XSET W2 .AND. $FCFF .OR. (O$(6).LS. 8 .AND. $0300)
W2        XSET W2 .OR. (A$(7).LS. 7 .AND. $0030)
W2        XSET W2 .OR. (A$(8).LS. 6 .AND. $0040)
W2        XSET W2 .OR. (A$(9).LS. 5 .AND. $0020)
W2        XSET W2 .AND. (O$(10).LS. 4 .OR. $FFEF)
W2        XSET W2 .AND. (O$(11).LS. 3 .OR. $FFF7)
W2        XSET W2 .AND. (O$(12).LS. 2 .OR. $FFFB)
W2        XSET W2 .AND. (O$(13).LS. 1 .OR. $FFFD)
          ENDM
*
*
```

```
*           ************************************************
*           *   EQUATES FOR MEMORY CONTROL                 *
*           ************************************************
*
*
A$REQB      XSET 1                  BUS REQUEST
A$MREQ      XSET 1                  MEMORY REQUEST
O$HREQ      XSET 0                  HOLD REQUEST
A$WRITE     XSET 1                  MEMORY WRITE
O$MBYTE     XSET 0                  MEMORY WORD/BYTE
*
*
*           ************************************************
*           *   MEMORY CONTROL MACRO DEFINITION            *
*           ************************************************
*
*
MEMCONT     MAC
W3          XSET W3 .AND. $FFC1 .OR. $000A          DEFAULT VALUES
W3          XSET W3 .OR. ((A$(1).AND. $1) .LS. 5)
W3          XSET W3 .OR. ((A$(2).AND. $1) .LS. 4)
W3          XSET W3 .AND. (O$(3).LS. 3 .OR. $FFF7)
W3          XSET W3 .OR. ((A$(4).AND. $1) .LS. 2)
W3          XSET W3 .AND. ((O$(5).LS. 1).OR. $FFFD)
* IF NO MREQ THEN MUST HAVE MBYTE MODE.
            IF (A$(2).EQ. A$) XSET W3=W3.AND.$FFFD
            ENDM
*
*
*           ************************************************
*           *     EQUATES FOR CONTROL STROBES              *
*           ************************************************
*
*
A$ROM       XSET 1                  ROM/IREGEN
O$IOEN      XSET 0                  I/O CONTROL
O$INTDIS    XSET 0                  INTERRUPT DISABLE
O$INTRIEN   XSET 0                  ENABLE I0-I3 ON AM2914
*
*
*           ************************************************
*           *    CONTROL STROBE MACRO DEFINITION         *
*           ************************************************
*
*
CONTROL     MAC
W4          XSET W4 .AND. $0FFF .OR. $7000          DEFAULT VALUES
W4          XSET W4 .OR. (A$(1).LS. 15 )
W4          XSET W4 .AND. (O$(2).LS. 14 .OR. $BFFF)
W4          XSET W4 .AND. (O$(3).LS. 13 .OR. $DFFF)
W4          XSET W4 .AND. (O$(4).LS. 12 .OR. $EFFF)
            ENDM
*
*
```

```
*            ************************************************
*            *    REGISTER EQUATES FOR AM2903 RAM PORT      *
*            ************************************************
*
* WARNING : DO NOT ALTER ANY REGISTER EQUATES
* OR ELSE 'MCOD' WILL NOT WORK.
*
BREG       XSET $0                 LOCAL BASE REGISTER
GREG       XSET $1                 GLOBAL BASE REGISTER
HREG       XSET $2                 HEAP REGISTER
WREG       XSET $3                 SCRATCH REGISTER
XREG       XSET $4                 SCRATCH REGISTER
YREG       XSET $5                 SCRATCH REGISTER
UREG       XSET $6                 SCRATCH REGISTER
VREG       XSET $7                 SCRATCH REGISTER
REAL1M     XSET $8                 MANTISSA M.S. REAL NO. REGISTER
REAL1I     XSET $9                 MANTISSA I.S. REAL NO. REGISTER
REAL1L     XSET $A                 MANTISSA L.S. REAL NO. REGISTER
EXP1       XSET $B                 EXPONENT REAL NO. REGISTER
REAL2M     XSET $C                 MANTISSA M.S. REAL NO. REGISTER
REAL2I     XSET $D                 MANTISSA I.S. REAL NO. REGISTER
REAL2L     XSET $E                 MANTISSA L.S. REAL NO. REGISTER
EXP2       XSET $F                 EXPONENT REAL NO. REGISTER
MULT1      XSET 5
MULT2      XSET 6
MULT3      XSET 7
A$BREG     XSET BREG
A$GREG     XSET GREG
A$HREG     XSET HREG
A$WREG     XSET WREG
A$XREG     XSET XREG
A$YREG     XSET YREG
A$UREG     XSET UREG
A$VREG     XSET VREG
A$REAL1M   XSET REAL1M
A$REAL1I   XSET REAL1I
A$REAL1L   XSET REAL1L
A$EXP1     XSET EXP1
A$REAL2M   XSET REAL2M
A$REAL2I   XSET REAL2I
A$REAL2L   XSET REAL2L
A$EXP2     XSET EXP2
A$MULT1    XSET MULT1
A$MULT2    XSET MULT2
A$MULT3    XSET MULT3
*
*
*            ***********************************************
*            *   CONTROL BITS DEFINITION                   *
*            ***********************************************
*
*
CNTLB      MAC
W4         XSET W4 .AND. $FF87 .OR. (((1).AND. $FF) .LS. 3)
```

```
          ENDM
*
*
BRAM      MAC
W4        XSET W4 .AND. $F87F .OR. (((1).AND. $F) .LS. 7)
          ENDM
*
*
ARAM      MAC
W4        XSET W4 .AND. $FF87 .OR. (((1).AND.$F) .LS. 3)
          ENDM
*
*
*              ******************************************
*              *    IMMEDIATE ROM DEFINITION             *
*              ******************************************
*
*
IMMD      MAC
W3        XSET W3 .AND. $FFFE            ENABLE IMMEDIATE OPERAND
W6        XSET (1)
CNTFLG    XSET 1
          ENDM
*
*
*              ******************************************
*              *  LOAD I REGISTER MACRO                 *
*              ******************************************
*
*
IREG      MAC
W6        XSET A$(1).AND. $F
W6        XSET W6 .OR. (A$(2).LS.4)
CNTFLG    XSET 1
          ENDM
*
*
*
* SWITCH LISTING BACK ON
          NLSM
          LIST
*
*
```

```
TITLE binout
£ OUTPUTS THE CONTENTS OF A BINARY FILE TO THE VDU WITH NO FORMAT
THIS PROGRAM IS USED TO TRANSFER TO THE MOTOROLA £
[] CHAR lookup = ("0","1","2","3","4","5","6","7","8","9","A","B","C
","D","E","F");
CHAR ctrlq = REPR(6);
CHAR termchar = REPR 20;  £ CTRL T £
[1:4] BITS digit;

PROC print hex = (INT param) VOID:
BEGIN
     BITS number;number := BIN param;
     FOR i FROM 4 DOWNTO 1
     DO digit[i] := number & (BIN 15);
        number := number SHR 4
     OD;
     FOR i FROM 1 TO 4
     DO ttywritech(lookup[ABS (digit[i]) + 1]) OD
END;

INT count := 0;
INT words per line = 6;
[0:10] INT errvec;
[0:1024] INT buffer;
FILE outf;
STRING infile;

PROC print buffer = VOID:
BEGIN
     geterr(errvec,2);
     FOR i FROM 0 TO errvec [1] - 1
     DO print hex (buffer [i])
     OD
END;

£start of program£
print ("NFPRNT",newline);
print("INPUT FILE NAME - ",newline);
read(newline,infile);
print ("READY FOR TRANSFER",newline);
tty write ch (ctrlq);
search(1,dos string(infile),5);  £open input file for reading£
WHILE prwfil(BIN 401,5,buffer,1024) = 0
DO print buffer OD;
search(4,dos string (infile) ,5);£close input file£
ttywritech(termchar)
```

```
TITLE split and transfer
£
Program to split a file into several parts and print it out, one par
t at a time.
First Part consists of byte 1,1 + n,1 + 2n.
Second Part consists of byte 2,2 + n,2 + 2n
Where n is the number of parts to be output.
The First 9 words of the file are ignored.
Program waits for a prompt before the output of each part.
£

INT file parts = 12;
INT ignore = 9;
INT wordinc = ROUND(file parts/2);
INT block size = 120;
INT delay = 500;
CHAR prompt = REPR 7;    £ CTRL G £
CHAR termchar = REPR 20;      £ CTRL T £
CHAR ctrlg = REPR 6, ctrlb = REPR 2;          £ CTRL T £
[] CHAR hex = ("0","1","2","3","4","5","6","7","8","9","A","B","C","
D","E","F");

INT wordnum, shift, words read, seqstart, seqno, numinseq;
BOOL notfinished;
BITS mask;
STRING filename, ds;

[] BITS nibble = (16r00f0, 16r000f);
[1:block size] INT buffer;
[1:2] INT errvec;
STRING dly;

PROC new buffer = VOID:
£ Read another "blocksize" words from the buffer £
BEGIN
      not finished := prwfil (BIN 401, 5, buffer, 120) = 0;
      geterr (errvec, 2);
      words read := errvec [1] - 1
END;

print ("FILENAME -", newline);
read (newline, filename);
WHILE
     print ("HOW MANY SEQUENCES - ", newline);
     read (newline, seqno);
     file parts/seqno ~= file parts OVER seqno
DO print ("MUST BE AN EXACT DIVISOR", newline) OD;
numinseq := ROUND (file parts/seqno);
ds := dos string (filename);
search (1, ds, 5);      £ Open File £
geterr (errvec, 2);
If errvec [1] ~= 0
THEN print ("CAN'T OPEN ", filename, newline)
ELSE
```

```
FOR seqstart TO seqno
DO
   print("READY FOR TRANSFER",newline);
   tty write ch (ctrlg);   £ Start of Transfer £
   FOR bytenum FROM (seqstart-1)*numinseq+1 TO seqstart*numinseq
   DO
     CHAR char;
     BITS this word;
     search(7,ds,5);      £ Rewind File £

     wordnum := ignore + (bytenum - 1) OVER 2;
     shift := IF bytenum OVER 2 = ROUND (bytenum/2)
              THEN £ l.s. byte, no shift needed £
                   mask := 16r00ff;
                   0
              ELSE £ m.s. byte , 8 shifts needed £
                   mask := 16rffff;
                   8
              FI;
     new buffer;
     WHILE not finished
     DO
        this word :=  (BIN buffer [wordnum] & mask) SHR shift;
        FOR i TO 2
        DO
          BITS temp;
          temp := this word & nibble [i];
          temp := temp SHR (4*(1 OVER i));       £ i = 1, 4 shifts;
i = 2, no shifts £
          tty write ch (hex [ABS temp + 1])
        OD;

        £ update counters £
        wordnum := wordnum + wordinc;
        IF wordnum ) words read
        THEN wordnum -:= (words read + 1);
             new buffer
        FI
     OD;
     FOR j TO delay DO dly := dly OD;
     tty write ch (termchar)
   OD;
   IF seqstart ~= seqno
   THEN £ wait for control b£
        CHAR ch;
        WHILE tty read ch (ch); ch ~= ctrlb DO SKIP OD
   FI
 OD £ seqstart loop £
FI;
search (4,ds,5)    £ Close file £
£ End of Program £
```

```
TITLE pcode compacter program
£ This program takes compiled Pascal programs and merges them,
together with a table pointing to their start addresses.

   Format of the file is :-
                                    Displacement from file start table
                                    Prog 1
                                    Prog 2
                                     .
                                     .
                                    Prog n

This file can then be transferred to Prom £
INT table length = 10;
INT table index := table length*2;
STRING ds, infilename, outfilename := dos string ("PROM");
INT words read, entry;
INT buffsize = 1024;
[0:buffsize-1] INT buffer;
[0:10] INT errvec;
[0:tablelength-1] INT table;
BOOL continue;

PROC get buffer = VOID:
BEGIN
     continue := prwfil (BIN 401, 1, buffer, buffsize) = 0;
     geterr (errvec, 2);
     words read := errvec [1]
END;

PROC write prog = VOID:
£ Transfers the input file to the output file £
BEGIN INT prog index := words read - 1, prog end := (buffer [0] OVER
2) - 1;
      WHILE continue
      DO prwfil (BIN 2, 2, buffer, IF progindex > progend
                                   THEN prog end - (prog index - words
read)
                                   ELSE words read FI);
         get buffer;
         IF progindex > progend
         THEN continue := FALSE
         FI;
         prog index +:= words read
      OD
END;

search (2, outfilename, 2);      £ Open output file £
FOR i FROM LWB table TO UPB table
DO table [i] := 0 OD;
prwfil (BIN 2, 2, table, tablelength); £ write dummy table £
print ("PROM PREPARATION LOADER", newline);
print ("PLEASE INPUT FILENAMES TO BE LOADED", newline);
read (newline, infilename);
```

```
entry := 0;
WHILE infilename /= "Q" AND entry <= tablelength-1
£ EXIT FROM LOOP WHEN FULL TABLE OR "Q" TYPED £
DO
  search (1, dos string (infilename), 1);    £ Open output file £
  geterr (errvec, 2);
  IF errvec [1] /= 0 THEN print ("CAN'T OPEN ", infilename, newline)
  ELSE
       get buffer;
       table [entry] := table index;
       table index +:= buffer [0];
       write prog;
       entry +:= 1;
       search (4, dos string (infilename), 1)      £ Close input file £
  FI;
  IF entry /= tablelength-1 THEN read (newline, infilename) FI
OD;

search (7, dos string (outfilename), 2);        £ Rewind output file £
prwfil (BIN 2, 2, table, table length);           £ Write table to disc
£
search (4, dos string (outfilename), 2);        £ Close output file £
print ("LOAD COMPLETE", newline)
```

```
TITLE mapping prom generator
£ This program takes a source file which contains microcode and generates a binary file
containing the mapping prom addresses for that microcode£
[0:127] INT prom;
STRING infile,outfile := dos string ("MAP");
BOOL more text := TRUE;
INT label length = 3, unidentified = ABS (16r0000);
BITS uni = BIN unidentified & 16r00ff;
INT invalid;
[0:label length] CHAR label;
INT inst count := -1;
INT maxline = 128;
INT lineptr;
[0:maxline] CHAR line;
FILE inf;
MODE INSTRUCTION = STRUCT ( STRING name, INT code);
[]INSTRUCTION lookup = (("PCA ",1) , ("PLA ",2) , ("PGA ",3) , ("PSC ",4) ,
                        ("PSL ",5) , ("PSG ",6) , ("PSW ",7) , ("PSB ",8) ,
                        ("PSR ",9) , ("PSS ",10) , ("FLD ",11) , ("PAC ",12) ,
                        ("PTR ",13) , ("CTF ",14) , ("RNG ",15) , ("CPB ",16) ,
                        ("CPW ",17) , ("CPR ",18) , ("CPS ",19) , ("CPT ",20) ,
                        ("CST ",21) , ("IPV ",22) , ("IPVC",23) , ("NOT ",24) ,
                        ("ANW ",25) , ("ANS ",26) , ("ORW ",27) , ("ORS ",28) ,
                        ("NGW ",29) , ("NGR ",30) , ("ADD ",31) , ("ADR ",32) ,
                        ("SUB ",33) , ("SBR ",34) , ("SBS ",35) , ("MLW ",36) ,
                        ("MLR ",37) , ("DVW ",38) , ("DVR ",39) , ("MOD ",40) ,
                        ("BLS ",41) , ("TIS ",42) , ("CLT ",43) , ("CEQ ",44) ,
                        ("CGT ",45) , ("CGE ",46) , ("CNE ",47) , ("CLE ",48) ,
                        ("CRLT",49) , ("CRE ",50) , ("CRG ",51) , ("CRGE",52) ,
                        ("CRNE",53) , ("CRLE",54) , ("EQS ",55) , ("SGE ",56) ,
                        ("SNE ",57) , ("SLE ",58) , ("CSLT",59) , ("CSE ",60) ,
                        ("CSG ",61) , ("CSGE",62) , ("CSNE",63) , ("CSLE",64) ,
                        ("FNV ",65) , ("JMP ",66) , ("FSP ",67) , ("CSP ",68) ,
                        ("IVR ",69) , ("CALL",70) , ("SCL ",71) , ("ENT ",72) ,
                        ("EXT ",73) , ("EPP ",74) , ("EXP ",75) , ("BGC ",76) ,
                        ("ENC ",77) , ("ETC ",78) , ("EXC ",79) , ("BGM ",80) ,
                        ("ENM ",81) , ("ETM ",82) , ("EXM ",83) , ("BGP ",84) ,
                        ("ENP ",85) , ("ETP ",86) , ("EXPC",87) , ("POP ",88) ,
                        ("NLN ",89) , ("INC ",90) , ("DEC ",91) , ("ICL ",92) ,
                        ("IMN ",93) , ("IPC ",94) , ("PLB ",95) , ("CLP ",96) ,
                        ("TNR ",97) , ("ABS ",98) , ("ABR ",99) , ("SUC ",100) ,
                        ("PRE ",101) , ("CVW ",102) , ("TEM ",103) , ("ATR ",104) ,
                        ("RTM ",105) , ("DLY ",106) , ("CNT ",107) , ("IOP ",108) ,
                        ("STRT",109) , ("STP ",110) , ("SHP ",111) , ("WAIT",112) ,
                        ("TST1",113) , ("TST2",114) , ("TST3",115) , ("TST4",116) , ("INVA",uniden
tified));

PROC input line = VOID:
BEGIN
     lineptr := 0;
     on line end (standin, (REF FILE f) BOOL: (GOTO line in;SKIP));
     read( newline);
     WHILE read (line [lineptr ]);lineptr (=max line DO lineptr +:= 1 OD;
     line in : lineptr -:= 1
```

```
END;

PROC check for end = VOID:
£ Scans the current line to see if it is the end of text marker (as defined by
the array "end marker"). Sets "more text" appropriately £
BEGIN
     [] CHAR end marker = ("**", " ", "P", "E", "N", "D", " ", "**");
     more text := FALSE;
     IF line ptr ¬= UPB end marker - 1
     THEN more text := TRUE
     ELSE FOR character FROM 2 TO UPB end marker £ first character is known to be "*" £
          WHILE NOT (more text := line [character - 1] ¬= end marker [character])
          DO SKIP OD
     FI
END;

PROC inst present = BOOL:
£ Searches the current line for the string defined by the array "inst".
Returns TRUE if found. £
BEGIN
     [] CHAR inst = (" ", "I", "N", "S", "T");
     BOOL found := FALSE;
     INT character, pos;
     FOR character TO line ptr - UPB inst + 1 £ Position 0 cannot contain a macro £
     WHILE NOT found
     DO
        IF line [character] = inst [1]
        THEN found := TRUE; pos := character;
             FOR lookahead FROM 2 TO UPB inst
             WHILE found := line [character + lookahead - 1] = inst [lookahead]
             DO SKIP OD
        FI
     OD;
     IF found
     THEN £ Rule out the possibility of another string (e.g. "INSTRUCTION") £
          found := (pos + UPB inst - 1) = line ptr;
          IF NOT found
          THEN found := line [pos + UPB inst ]  = " " FI
     FI;
     found
END;

PROC label present = BOOL:
£ Looks at the first character of the present line to see if a label is present.
If it is "label" is set to the first "label length + 1" characters of the line
and TRUE is returned £
BEGIN
     IF line [0] >= "A" AND line [0] <= "Z"
     THEN
          FOR character FROM 0 TO label length
          DO label [character] := line [character] OD;
          TRUE
     ELSE FALSE
     FI
```

```
END;

£ Main program starts here, set up the I/O first £
print ("FILENAME -",newline);
read (newline,infile);
open (inf,infile,standin channel);
standin := inf;
search(2,outfile,2);
FOR count FROM LWB prom TO UPB prom
DO prom [count] := unidentified OD;

WHILE more text
DO
   input line;
   IF line [0] = "**" £ Comment £
   THEN check for end
   ELSE
        IF inst present
        THEN inst count +:= 1
        FI;
        IF label present
        THEN £Search the lookup table to see if it is a P-code label £
             BOOL found := FALSE; INT code;
             FOR entry FROM LWB lookup TO UPB lookup
             WHILE NOT found
             DO
               found := TRUE;
               FOR character FROM LWB label TO UPB label
               WHILE found := label [character] = (character + 1) ELEM (name OF lookup [entry]
)
               DO SKIP OD;
               IF found THEN code := code OF lookup [entry] FI
               OD;
               IF found
               THEN £ update the output array "prom" £
                    INT promentry;
                    IF inst count > ABS (16rff)
                    THEN tty mess ("error,prog address too large");
                         stop
                    FI;
                    prom entry := code OVER 2;
                    IF code = unidentified
                    THEN £ Invalid instruction label £
                         invalid := inst count
                    ELSE
                         IF NOT ODD code
                         THEN £ even number, m.s. byte £
                              prom [prom entry] := ABS ((BIN prom [prom entry] & 16r00ff) ! (B
IN inst count SHL 8))
                         ELSE £ odd number, l.s. byte £
                              prom [prom entry] := ABS ((BIN prom [prom entry] & 16rff00) ! (B
IN inst count))
                         FI
                    FI
```

```
                FI £ found £
            FI £ label present £
        FI £ comment £
OD;
£ Set unidentified bytes to invalid instruction address £
FOR count FROM LWB prom TO UPB prom
DO
   REF INT promc = prom [count];
   IF (BIN promc SHR 8) = uni
   THEN promc := promc ! (invalid SHL 8)
   FI;
   IF (BIN promc & 16r00ff) = uni
   THEN promc := promc ! invalid
   FI
OD;
prwfil (BIN2, 2, prom, 128); £ Write to disc £
search (4, outfile, 2) £ Close the file £
```

```
00001                          *
00002                          * INTELLIGENT MEMORY/AMD INTERFACE
00003                          *
00004                          * 3/2/82
00005                          *
00006  0000    A PROM   EQU    0        0 FOR PROM, 1 FOR RAM
00007                          * INTELLIGENT MEMORY PROGRAM
00008                          * PIA EQUATES
00009  2020    A LADDRP EQU    $2020    L.S ADDRESS PIA DATA REGISTER
00010  2021    A LADDRC EQU    $2021    L.S ADDRESS PIA DATA CONTROL REGISTER
00011  2022    A MADDRP EQU    $2022    M.S.  ADDRESS PIA DATA REGISTER
00012  2023    A MADDRC EQU    $2023    M.S.  ADDRESS PIA CONTROL REGISTER
00013  2024    A LDATAP EQU    $2024    L.S.  DATA PIA DATA REG
00014  2025    A LDATAC EQU    $2025    L.S.  DATA PIA CONTROL REG
00015  2026    A MDATAP EQU    $2026    M.S.  DATA PIA DATA REG
00016  2027    A MDATAC EQU    $2027    L.S.  DATA PIA CONTROL REG
00017  202C    A HSHAKP EQU    $202C    HANDSHAKE PIA DATA REG
00018  202D    A HSHAKC EQU    $202D    KANDSHAKE PIA CONTROL REG
00019  2030    A P1DRA  EQU    $2030
00020  2031    A P1CRA  EQU    $2031
00021  2032    A P1DRB  EQU    $2032
00022  2033    A P1CRB  EQU    $2033
00023  2038    A P2DRA  EQU    $2038
00024  2039    A P2CRA  EQU    $2039
00025  203A    A P2DRB  EQU    $203A
00026  203B    A P2CRB  EQU    $203B
00027                          * HANDSHAKE PIA STRUCTURE
00028                          *       CRA1 = MREQ
00029                          *       PIA 0 = WORD/BYTE
00030                          *       PIA 1 = RD/WRT
00031                          *       PIA 2 = SYNC
00032                          *       PIA 3 = IRST
00033                          *       PIA 4 = DREQ
00034                          *       PIA 5 = PHI2
00035                          *       PIA 6 = DSTOP
00036                          *        PIA 7 = PHI 1
00037  0080    A MREQMK EQU
00038  0040    A DSTPMK EQU
00039  0020    A PHI2MK EQU
00040  0010    A DREQMK EQU
00041  0008    A IRSTMK EQU
00042  0004    A SYNCMK EQU
00043  0002    A RDWTMK EQU
00044  0001    A WDBTMK EQU
00045  0800    A BSIZE  EQU    2048     SIZE OF A MEMORY BLOCK
00046  001B    A BREAK  EQU    $1B
00047  2034    A ACIACS EQU    $2034    ACIA CONTROL/STATUS
00048  2035    A ACIADA EQU    ACIACS+1 ACIA DATA REGISTER
00049
00050  0000    A        IFEQ   PROM
0051A F800              ORG    $F800
00052                   ENDC
00053  0000    A        IFNE   PROM
00055                    ENDC
```

```
00056A F800 10CE 1F9A     A PSTRT  LDS    £STACK
00057                       * INITIALISE THE PIA'S
00058A F804 4F                     CLRA
00059                       * ACCESS DDR'S
00060A F805 B7   2023     A        STA    MADDRC
00061A F808 B7   2021     A        STA    LADDRC
00062A F80B B7   2025     A        STA    LDATAC
00063A F80E B7   2027     A        STA    MDATAC
00064A F811 B7   202D     A        STA    HSHAKC
00065A F814 B7   2022     A        STA    MADDRP   ADDRESS = INPUT
00066A F817 B7   2020     A        STA    LADDRP
00067A F81A B7   2026     A        STA    MDATAP   DATA = INPUT (INITIALLY)
00068A F81D B7   2024     A        STA    LDATAP
00069A F820 86   4C       A        LDA    £
00070A F822 B7   202C     A        STA    HSHAKP
00071A F825 86   04       A        LDA    £
00072A F827 B7   2023     A        STA    MADDRC
00073A F82A B7   2021     A        STA    LADDRC
00074A F82D B7   2027     A        STA    MDATAC
00075A F830 B7   2025     A        STA    LDATAC
00076A F833 86   06       A        LDA    £
00077A F835 B7   202D     A        STA    HSHAKC
00078A F838 86   4C       A        LDA    £
00079A F83A B7   202C     A        STA    HSHAKP
00080                       * INITIALISE OTHER VARIABLES
00081A F83D 7F   1F9B     A        CLR    MREQ
00082A F840 7F   1F9C     A        CLR    RDFLG
00083A F843 7F   1F9D     A        CLR    BYTE
00084A F846 7F   1F9E     A        CLR    SYNC
00085                       * SET UP THE BAUD RATE TO 9600 (SAME AS PRIME)
00086A F849 86   03       A        LDA    £3       MASTER RESET
00087A F84B B7   2034     A        STA    ACIACS
00088A F84E 86   02       A        LDA    £2
00089A F850 B7   2034     A        STA    ACIACS
00090A F853 F6   2034     A LOOPX  LDB    ACIACS   CHECK TX EMPTY
00091A F856 57                     ASRB
00092A F857 57                     ASRB
00093A F858 24   F9   F853         BCC    LOOPX
00094A F85A BD   FCFF     A        JSR    NLCR
00095A F85D 8E   FD86     A        LDX    £STRTMS
00096A F860 BD   FC59     A        JSR    OUTMES
00097                       *
00098                       * PROGRAM LOOP STARTS HERE
00099                       *
00100A F863 10CE 1F9A     A START  LDS    £STACK   IN CASE SUBROUTINE RETURN WAS NOT MADE
00101A F867 BD   FCFF     A        JSR    NLCR
00102A F86A B6   FD6A     A        LDA    PROMPT
00103A F86D BD   FD4E     A        JSR    OUTCH    GIVE THE USER A PROMPT
00104A F870 B6   FD6D     A        LDA    SPACE
00105A F873 BD   FD4E     A        JSR    OUTCH
00106A F876 BD   FD21     A        JSR    INCH     NEXT COMMAND
00107A F879 B1   FD67     A        CMPA   R
00108A F87C 27   1F   F89D         BEQ    READ
00109A F87E B1   FD68     A        CMPA   T
```

```
00110A F8B1 1027 0097 F91C        LBEQ   TRANSF
00111A F8B5 B1   FD69    A        CMPA   G
00112A F8B8 1027 015B F9E7        LBEQ   GO
00113A F8BC 81   50      A        CMPA   £'P
00114A F8BE 1027 024D FADF        LBEQ   PPROG
00115                          * ARRIVED HERE IF AN INVALID COMMAND HAS BEEN TYPED IN
00116A F8C2 BD   FCFF    A        JSR    NLCR
00117A F8C5 8E   FD9A    A        LDX    £IVCMND
00118A F8C8 BD   FC59    A        JSR    OUTMES   TRY AGAIN
00119A F8CB 20   C6   F863        BRA    START
00120                          *
00121                          * READ COMMAND
00122                          * R (Start Address) (End Address)
00123                          *
00124A F8CD BD   FC73    A READ   JSR    INPARA   INPUT (SA)
00125A F8A0 FD   1F80    A        STD    ADDRES
00126A F8A3 BD   FC73    A        JSR    INPARA   INPUT (EA)
00127A F8A6 FD   1F82    A        STD    ENDADR
00128A F8A9 7F   1F84    A        CLR    MSBYTE
00129                          * LOOP STARTS HERE
00130                          * EACH ITERATION READS ONE BYTE OF MOTOROLA MEMORY AND
00131                          * OUTPUTS IT TO THE VDU.
00132                          * EXITS WHEN "ADDRESS" > "ENDADR"
00133A F8AC FC   1F80    A RLOOP  LDD    ADDRES
00134A F8AF 10B3 1F82    A        CMPD   ENDADR   ANY MORE
00135A F8B3 2E   5B   F910        BGT    REND     NO
00136A F8B5 1F   01      A        TFR    D,X      SET UP THE ADDRESS IN X
00137A F8B7 A6   84      A        LDA    0,X
00138A F8B9 34   02      A        PSHS   A
00139                          * "MSBYTE" = 0 IF THE CURRENT DATA IS THE MOST SIGNIFICANT BYTE
00140A F8BB 7D   1F84    A        TST    MSBYTE
00141A F8BE 26   1F   F8DF        BNE    LSBYTE
00142                          * ARRIVED HERE IF ADDRESS (AS WELL AS DATA) IS TO BE OUTPUT.
00143A F8C0 7C   1F84    A        INC    MSBYTE
00144A F8C3 BD   FCFF    A        JSR    NLCR
00145A F8C6 FC   1F80    A        LDD    ADDRES
00146A F8C9 BD   FCCC    A        JSR    OBYTE    M.S. BYTE OF ADDRESS
00147A F8CC 1F   98      A        TFR    B,A      A := B
00148A F8CE BD   FCCC    A        JSR    OBYTE    L.S. BYTE OF ADDRESS
00149A F8D1 B6   FD6B    A        LDA    COLON
00150A F8D4 BD   FD4E    A        JSR    OUTCH
00151A F8D7 B6   FD6D    A        LDA    SPACE
00152A F8DA BD   FD4E    A        JSR    OUTCH
00153A F8DD 20   03   F8E2        BRA    OUTDAT
00154A F8DF 7F   1F84    A LSBYTE CLR    MSBYTE
00155A F8E2 4F                 OUTDAT CLRA
00156A F8E3 BD   FD42    A        JSR    NOWAIT
00157A F8E6 81   1B      A        CMPA   £BREAK
00158A F8E8 1027 FF77 F863        LBEQ   START    ABORT DISPLAY
00159A F8EC B1   FD6D    A        CMPA   SPACE    PAUSE DISPLAY
00160A F8EF 26   0F   F900        BNE    ODAT     NO
00161                          * ARRIVED HERE IF DISPLAY IS TO BE HALTED
00162                          * WAIT FOR SPACE TO PROCEED
00163A F8F1 4F                    CLRA
```

```
00164A F8F2 BD   FD32    A WAITSP JSR    NOECHO
00165A F8F5 81   1B      A        CMPA   £BREAK
00166A F8F7 1027 FF68 F863        LBEQ   START    ABORT DISPLAY
00167A F8FB B1   FD6D    A        CMPA   SPACE
00168A F8FE 26   F2   F8F2        BNE    WAITSP   NOT PRESSED YET
00169                       * OUTPUT THE DATA BYTE
00170A F900 A6   E0      A ODAT   LDA    ,S+      PULL DATA BYTE
00171A F902 BD   FCCC    A        JSR    OBYTE
00172A F905 FC   1F80    A        LDD    ADDRES
00173A F908 C3   0001    A        ADDD   £1
00174A F90B FD   1F80    A        STD    ADDRES   INCREMENT BY 1
00175A F90E 20   9C   F8AC        BRA    RLOOP
00176A F910 BD   FCFF    A REND   JSR    NLCR
00177A F913 8E   FDD2    A        LDX    £RMES
00178A F916 BD   FC59    A        JSR    OUTMES
00179A F919 7E   F863    A        JMP    START    FINISHED THIS COMMAND
00180                       **
00181                       * TRANSFER COMMAND
00182                       * T (Start Address) [(No. of Transfers)]
00183                       *
00184A F91C BD   F922    A TRANSF JSR    TSUBR
00185A F91F 7E   F863    A        JMP    START
00186                       * WRITTEN AS A SUBROUTINE TO ALLOW BOOT COMMAND TO CALL THE CODE
00187A F922 BD   FC73    A TSUBR  JSR    INPARA   INPUT (SA)
00188A F925 FD   1F80    A        STD    ADDRES
00189A F928 83   0800    A        SUBD   £BSIZE
00190A F92B FD   1F9F    A        STD    MEMSRT
00191                       * IS NEXT NON-SPACE CHARACTER A CARRIAGE RETURN?
00192A F92E BD   FD21    A MSP    JSR    INCH
00193A F931 B1   FD6D    A        CMPA   SPACE
00194A F934 27   F8   F92E        BEQ    MSP      IGNORE SPACES
00195A F936 B1   FD66    A        CMPA   CR
00196A F939 27   0B   F946        BEQ    TNOOPT   NO OPTIONAL PARAMETER
00197                       * ARRIVED HERE IF OPTIONAL (No. Transfers) PARAMETER HAS BEEN GIVEN
00198A F93B BD   FC7B    A        JSR    FSTCHI   INPUT (No. Transfers) PARAMETER
00199A F93E F7   1FA3    A        STB    NUMT
00200A F941 BD   FCFF    A        JSR    NLCR
00201A F944 20   0B   F951        BRA    TCLPNT
00202A F946 C6   01      A TNOOPT LDB    £1
00203A F948 F7   1FA3    A        STB    NUMT
00204A F94B B6   FD65    A        LDA    NL
00205A F94E BD   FD4E    A        JSR    OUTCH
00206A F951 FC   FD81    A TCLPNT LDD    MAXMEM
00207A F954 FD   1F82    A        STD    ENDADR
00208A F957 7F   1FA4    A        CLR    CHIPNO
00209A F95A CC   0800    A        LDD    £BSIZE   1K
00210A F95D FD   1FA1    A        STD    SIZE
00211A F960 7F   1F85    A        CLR    ERFLAG   CLEAR THE ERROR FLAG
00212A F963 8E   FDE7    A        LDX    £RDYMES
00213A F966 BD   FC59    A        JSR    OUTMES   TELL THE USER TO COMMENCE
00214A F969 BE   1F80    A        LDX    ADDRES
00215                       *WAIT FOR CONTROL AND G
00216A F96C BD   FD32    A WAITG6 JSR    NOECHO
00217A F96F B1   FD6E    A        CMPA   CTRLG
```

```
00218A F972 26   F8   F96C          BNE    WAITCG
00219A F974 7C   1FA4    A TSTART   INC    CHIPNO
00220A F977 B6   1FA3    A          LDA    NUMT
00221A F97A B1   1FA4    A          CMPA   CHIPNO    END OF TRANSFER?
00222A F97D 2D   3A   F9B9          BLT    TEND      YES
00223A F97F FC   1FA1    A          LDD    SIZE
00224A F982 F3   1F9F    A          ADDD   MEMSRT    UPDATE THE MEMORY POINTER
00225A F985 FD   1F9F    A          STD    MEMSRT
00226A F988 1F   01      A          TFR    D,X
00227A F98A BD   FD32    A FSTDIG   JSR    NOECHO    INPUT M.S. DIGIT
00228A F98D B1   FD6F    A          CMPA   CTRLT     IS THIS THE TERMINATION CHARACTER
00229A F990 27   E2   F974          BEQ    TSTART    YES
00230A F992 BC   1F82    A          CMPX   ENDADR
00231A F995 2E   15   F9AC          BGT    TERR1     MEMORY FULL
00232A F997 BD   FCE4    A          JSR    CNVERT
00233A F99A 48                      LSLA             M.S. NIBBLE*2
00234A F99B 48                      LSLA             *4
00235A F99C 48                      LSLA             *8
00236A F99D 48                      LSLA             *16
00237A F99E A7   84      A          STA    ,X
00238A F9A0 BD   FD32    A          JSR    NOECHO    INPUT L.S. NIBBLE
00239A F9A3 BD   FCE4    A          JSR    CNVERT
00240A F9A6 AB   84      A          ADDA   ,X        M.S. NIBBLE + L.S. NIBBLE
00241A F9A8 A7   80      A          STA    ,X+
00242A F9AA 20   DE   F98A          BRA    FSTDIG
00243A F9AC 86   01      A TERR1    LDA    #1
00244A F9AE B7   1F85    A          STA    ERFLAG
00245A F9B1 BD   FD32    A WAITCT   JSR    NOECHO
00246A F9B4 B1   FD6F    A          CMPA   CTRLT
00247A F9B7 26   F8   F9B1          BNE    WAITCT    WAIT FOR PRIME TO STOP SENDING
00248                       * END OF TRANSFER, WAIT FOR CTRLT TO BE PRESSED
00249                       * THIS GIVES THE USER TIME TO RECONNECT THE LINE.
00250A F9B9 BD   FD21    A TEND     JSR    INCH
00251A F9BC B1   FD6F    A          CMPA   CTRLT
00252A F9BF 26   F8   F9B9          BNE    TEND
00253A F9C1 BF   1F82    A          STX    ENDADR
00254                       * OUTPUT AN ERROR (OR SUCCESS) MESSAGE
00255A F9C4 BD   FCFF    A          JSR    NLCR
00256A F9C7 B6   1F85    A          LDA    ERFLAG
00257A F9CA 81   00      A          CMPA   #0
00258A F9CC 2E   07   F9D5          BGT    ERROR1
00259A F9CE 8E   FE45    A          LDX    #TMES     TRANSFER SUCCESSFUL
00260A F9D1 BD   FC59    A          JSR    OUTMES
00261A F9D4 39                      RTS
00262A F9D5 81   01      A ERROR1   CMPA   #1
00263A F9D7 2E   07   F9E0          BGT    ERROR2
00264A F9D9 8E   FE03    A          LDX    #TEMS1    MEMORY FULL
00265A F9DC BD   FC59    A          JSR    OUTMES
00266A F9DF 39                      RTS
00267A F9E0 8E   FE23    A ERROR2   LDX    #TEMS2    ILLEGAL CHARACTER ENCOUNTERED
00268A F9E3 BD   FC59    A          JSR    OUTMES
00269A F9E6 39                      RTS
00270                       *
00271                       * GO COMMAND
```

```
00272                          * 6
00273                          * CAUSE A SYSTEM RESET AND THEN RESPOND TO ALL MEMORY REQUESTS
00274                          * AMD ADDRESS MAPS DIRECTLY ONTO MOTOROLA ADDRESS
00275A F9E7 8E   FE57    A GO      LDX    #INSMES
00276A F9EA 17   026C FC59         LBSR   OUTMES
00277A F9ED 17   030F FCFF         LBSR   NLCR
00278A F9F0 86   44       A        LDA    #
00279A F9F2 B7   202C     A        STA    HSHAKP   SYSTEM RESET
00280A F9F5 17   0365 FD5D         LBSR   SRDLY
00281A F9F8 86   4C       A        LDA    #
00282A F9FA B7   202C     A        STA    HSHAKP   END OF RESET
00283A F9FD B6   202C     A CYCLE1 LDA    HSHAKP
00284A FA00 84   80       A        ANDA   #MREQMK  TEST FOR MEMORY REQUEST
00285A FA02 4D                     TSTA
00286A FA03 26   1A   FA1F         BNE    NOMREQ
00287A FA05 7C   1F9B     A        INC    MREQ
00288A FA08 B6   202C     A        LDA    HSHAKP
00289A FA0B 1F   89       A        TFR    A,B
00290A FA0D 84   02       A        ANDA   #RDWTMK
00291A FA0F B7   1F9C     A        STA    RDFLG
00292A FA12 C4   01       A        ANDB   #WDBTMK
00293A FA14 F7   1F9D     A        STB    BYTE     0 FOR WORD, NON-ZERO FOR BYTE
00294A FA17 B6   2022     A        LDA    MADDRP
00295A FA1A F6   2020     A        LDB    LADDRP
00296A FA1D 1F   01       A        TFR    D,X      ADDRESS NOW IN X REG
00297            FA1F     A NOMREQ EQU    *
00298A FA1F B6   202C     A WAITP2 LDA    HSHAKP
00299A FA22 84   20       A        ANDA   #PHI2MK
00300A FA24 27   F9   FA1F         BEQ    WAITP2   WAIT FOR PHI 2 = 1
00301A FA26 4F                CYCLE2 CLRA
00302A FA27 B7   2025     A        STA    LDATAC   ACCESS DDR
00303A FA2A B7   2027     A        STA    MDATAC
00304A FA2D B7   2024     A        STA    LDATAP   DDR SET FOR INPUT
00305A FA30 B7   2026     A        STA    MDATAP
00306A FA33 86   04       A        LDA    #
00307A FA35 B7   2025     A        STA    LDATAC
00308A FA38 B7   2027     A        STA    MDATAC
00309A FA3B 7D   1F9E     A        TST    SYNC
00310A FA3E 27   0B   FA4B         BEQ    NOSYNC
00311A FA40 7F   1F9E     A        CLR    SYNC
00312A FA43 B6   202C     A        LDA    HSHAKP
00313A FA46 8A   04       A        ORA    #SYNCMK  SYNC = 1
00314A FA48 B7   202C     A        STA    HSHAKP
00315A FA4B 7D   1F9B     A NOSYNC TST    MREQ
00316A FA4E 27   2A   FA7A         BEQ    NOWRT
00317A FA50 7D   1F9C     A        TST    RDFLG
00318A FA53 26   25   FA7A         BNE    NOWRT
00319A FA55 B6   202C     A        LDA    HSHAKP
00320A FA58 8A   40       A        ORA    #DSTPMK  DSTOP = 1
00321A FA5A B7   202C     A        STA    HSHAKP
00322A FA5D B6   2026     A        LDA    MDATAP
00323A FA60 F6   2024     A        LDB    LDATAP
00324A FA63 7D   1F9D     A        TST    BYTE
00325A FA66 27   10   FA78         BEQ    WDMODE   BYTE = 0 MEANS WORD MODE
```

```
00326                          * DON'T LET THE PIA INTERRUPT BIT BE SET.
00327A FA6B 8C   2031     A            CMPX    #P1CRA
00328A FA6B 27   05   FA72             BEQ     INTMSK
00329A FA6D 8C   2033     A            CMPX    #P1CRB
00330A FA70 26   02   FA74             BNE     BWRT
00331A FA72 C4   FE       A INTMSK    ANDB    #$FE        MASK OUT L.S. BIT
00332A FA74 E7   84       A BWRT      STB     0,X
00333A FA76 20   02   FA7A             BRA     NOWRT
00334A FA78 ED   84       A WDMODE    STD     0,X
00335            FA7A     A NOWRT     EQU     *
00336A FA7A B6   202C     A WAITC4    LDA     HSHAKP
00337A FA7D 84   20       A           ANDA    #PHI2MK
00338A FA7F 26   F9   FA7A            BNE     WAITC4      WAIT FOR PHI 2 = 0
00339A FA81 B6   202C     A CYCLE4    LDA     HSHAKP
00340A FA84 84   BF       A           ANDA    #
00341A FA86 B7   202C     A           STA     HSHAKP
00342A FA89 7D   1F9B     A           TST     MREQ
00343A FA8C 27   3E   FACC            BEQ     CLRFLG      NO MEMORY REQUEST
00344A FA8E 7C   1F9E     A           INC     SYNC
00345A FA91 B6   202C     A           LDA     HSHAKP
00346A FA94 84   FB       A           ANDA    #
00347A FA96 B7   202C     A           STA     HSHAKP
00348A FA99 7D   1F9C     A           TST     RDFLG
00349A FA9C 27   2E   FACC            BEQ     CLRFLG      NO READ REQUEST
00350A FA9E 4F                       CLRA
00351A FA9F B7   2027     A           STA     MDATAC      ACCESS DDR
00352A FAA2 B7   2025     A           STA     LDATAC
00353A FAA5 43                       COMA                A = $FF
00354A FAA6 B7   2026     A           STA     MDATAP
00355A FAA9 B7   2024     A           STA     LDATAP
00356A FAAC 86   04       A           LDA     #
00357A FAAE B7   2027     A           STA     MDATAC
00358A FAB1 B7   2025     A           STA     LDATAC      DATA PIA'S NOW SET FOR OUTPUT
00359                          * BYTE MODE : AREG = 0, BREG = DATA
00360                          * WORD MODE AREG = M.S. DATA, BREG = L.S. DATA
00361A FAB4 7D   1F9D     A           TST     BYTE
00362A FAB7 27   0B   FAC4            BEQ     NOTBYT
00363A FAB9 4F                       CLRA
00364A FABA E6   84       A           LDB     0,X
00365A FABC B7   2026     A           STA     MDATAP
00366A FABF F7   2024     A           STB     LDATAP
00367A FAC2 20   08   FACC            BRA     CLRFLG
00368A FAC4 EC   84       A NOTBYT    LDD     0,X
00369A FAC6 B7   2026     A           STA     MDATAP
00370A FAC9 F7   2024     A           STB     LDATAP
00371A FACC 7F   1F9B     A CLRFLG    CLR     MREQ
00372A FACF 7F   1F9C     A           CLR     RDFLG
00373A FAD2 7F   1F9D     A           CLR     BYTE
00374                          * WAIT FOR PHI 1 = 1
00375A FAD5 B6   202D     A WAITP1    LDA     HSHAKC
00376A FAD8 84   80       A           ANDA    #
00377A FADA 27   F9   FAD5            BEQ     WAITP1
00378A FADC 16   FF1E F9FD            LBRA    CYCLE1
00379                          *16K PROM CODE
```

```
00380                          *
00381                          *INITIALIZE PIAS AND VARIABLES
00382A FADF 4F                 PPROG  CLRA
00383A FAE0 B7   1FAD     A           STA    FERR
00384A FAE3 B7   1FAE     A           STA    VERR      CLEAR ERROR FLAGS
00385A FAE6 B7   2031     A           STA    P1CRA
00386A FAE9 B7   2033     A           STA    P1CRB
00387A FAEC B7   203B     A           STA    P2CRB
00388A FAEF 43                       COMA
00389A FAF0 B7   2030     A           STA    P1DRA     SET UP FOR OUTPUT
00390A FAF3 B7   2032     A           STA    P1DRB
00391A FAF6 B7   203A     A           STA    P2DRB
00392A FAF9 86   04       A           LDA    £4
00393A FAFB B7   2031     A           STA    P1CRA     RESTORE CRS
00394A FAFE B7   2033     A           STA    P1CRB
00395A FB01 B7   203B     A           STA    P2CRB
00396A FB04 86   16       A           LDA    £$16
00397A FB06 B7   203A     A           STA    P2DRB     ENSURE  5V ONLY ON VPP
00398A FB09 4F                       CLRA
00399A FB0A B7   2030     A           STA    P1DRA
00400A FB0D B7   2032     A           STA    P1DRB     PROM ADDR TO ZERO
00401                          *
00402                          *
00403                          *START PROM SEQUENCE
00404                          *
00405A FB10 8E   FE99     A SADDR  LDX    £MESG1    OUTPUT INITIAL COMMANDS
00406A FB13 BD   FC59     A        JSR    XPDATA
00407A FB16 8E   1FA5     A        LDX    £PSTART   GET START ADDR
00408A FB19 BD   FC6D     A        JSR    XINADD
00409A FB1C 8E   FEDA     A        LDX    £MESG2
00410A FB1F BD   FC59     A        JSR    XPDATA
00411A FB22 8E   1FA7     A        LDX    £END      GET END ADDRESS
00412A FB25 BD   FC6D     A        JSR    XINADD
00413A FB28 FC   1FA7     A        LDD    END
00414A FB2B B3   1FA5     A        SUBD   PSTART
00415A FB2E C3   0001     A        ADDD   £1
00416A FB31 FD   1FAB     A        STD    BYTES
00417A FB34 8E   FEEF     A        LDX    £MESG3
2000t$$
00418A FB37 BD   FC59     A        JSR    XPDATA    CHECK PROM
00419A FB3A BD   FB72     A        JSR    CHCK      CHECK ERASED
00420A FB3D B6   1FAD     A        LDA    FERR
00421A FB40 27   05   FB47        BEQ    NEXT
00422A FB42 7F   1FAD     A        CLR    FERR      NOT ERASED
00423A FB45 20   C9   FB10        BRA    SADDR     GO TO START
00424A FB47 8E   FEFD     A NEXT   LDX    £MESG4    PROGRAMMNING MESSAGE
00425A FB4A BD   FC59     A        JSR    XPDATA
00426A FB4D BD   FBB7     A        JSR    PROG
00427A FB50 BD   FC17     A        JSR    VERFY
00428A FB53 B6   1FAE     A        LDA    VERR      CHECK VERIFIED
00429A FB56 27   08   FB63        BEQ    NEXT1
00430A FB58 7F   1FAE     A        CLR    VERR
00431A FB5B 8E   FF41     A        LDX    £MESG9
00432A FB5E BD   FC59     A        JSR    XPDATA
```

```
00433A FB61 20   09   FB6C        BRA    OUT
00434A FB63 8E   FF0E    A NEXT1  LDX    #MES65   PROGRAMMED AND VERIFIED
00435A FB66 BD   FC59    A        JSR    XPDATA
00436A FB69 BD   FCFF    A        JSR    XPCRLF
00437A FB6C BD   FCFF    A OUT    JSR    XPCRLF
00438A FB6F 16   FCF1 F863        LBRA   START
00439                             *
00440                             *
00441A FB72 CC   0000    A CHCK   LDD    #0
00442A FB75 FD   1FA9    A        STD    ADDR
00443A FB78 86   0B      A        LDA    #$0B     LED ON CE LOW 5V READ
00444A FB7A B7   203A    A        STA    P2DRB    SET CONTROL TO READ PROM
00445A FB7D 4F                    CLRA
00446A FB7E B7   2039    A        STA    P2CRA
00447A FB81 B7   2038    A        STA    P2DRA    SET DATA REG FOR INPUT
00448A FB84 86   04      A        LDA    #4
00449A FB86 B7   2039    A        STA    P2CRA
00450A FB89 FC   1FA9    A        LDD    ADDR
00451A FB8C F7   2030    A CONT   STB    P1DRA
00452A FB8F B7   2032    A        STA    P1DRB    OUTPUT FROM ADDR
00453A FB92 B6   2038    A        LDA    P2DRA    READ DATA
00454A FB95 81   FF      A        CMPA   #$FF
00455A FB97 26   11   FBAA        BNE    ERR1
00456A FB99 FC   1FA9    A        LDD    ADDR
00457A FB9C 1083 0FFF    A        CMPD   #$0FFF
00458A FBA0 27   14   FBB6        BEQ    EXIT
00459A FBA2 C3   0001    A        ADDD   #1
00460A FBA5 FD   1FA9    A        STD    ADDR
00461A FBA8 20   E2   FB8C        BRA    CONT
00462A FBAA 86   01      A ERR1   LDA    #1
00463A FBAC B7   1FAD    A        STA    FERR
00464A FBAF 30   8D 0373          LEAX   MES67,PCR
00465A FBB3 BD   FC59    A        JSR    XPDATA
00466A FBB6 39                    EXIT   RTS
00467                             *
00468                             *
00469A FBB7 4F                    PROG   CLRA
00470A FBB8 B7   2039    A        STA    P2CRA
00471A FBBB 86   FF      A        LDA    #$FF     SET FOR OUTPUT
00472A FBBD B7   2038    A        STA    P2DRA
00473A FBC0 86   04      A        LDA    #4
00474A FBC2 B7   2039    A        STA    P2CRA
00475A FBC5 86   0A      A        LDA    #$0A
00476A FBC7 B7   203A    A        STA    P2DRB    CE LOW VPP=5V WRITE SELECT
00477A FBCA 7F   1FA9    A        CLR    ADDR
00478A FBCD 7F   1FAA    A        CLR    ADDR+1
00479A FBD0 BE   1FA5    A        LDX    PSTART
00480A FBD3 FC   1FA9    A RPT    LDD    ADDR
00481A FBD6 F7   2030    A        STB    P1DRA
00482A FBD9 8A   80      A        ORA    #$80     SET OE HIGH
00483A FBDB B7   2032    A        STA    P1DRB    OUTPUT ADDR
00484A FBDE E6   84      A        LDB    ,X
00485A FBE0 F7   2038    A        STB    P2DRA    OUTPUT DATA
00486A FBE3 86   08      A        LDA    #$08     ENABLE VPP = 25V
```

```
00487A FBE5 B7   203A    A          STA    P2DRB
00488A FBE8 86   0C      A          LDA    £$0C      VPP=25V  CE HIGH
00489A FBEA B7   203A    A          STA    P2DRB
00490A FBED 108E 0000    A          LDY    £0
00491A FBF1 31   21      A LOOP     LEAY   1,Y
00492A FBF3 108C 0F80    A          CMPY   £$F80     COUNT FOR 50ms
00493A FBF7 26   F8   FBF1          BNE    LOOP
00494A FBF9 86   08      A          LDA    £$08
00495A FBFB B7   203A    A          STA    P2DRB
00496A FBFE FC   1FA9    A          LDD    ADDR
00497A FC01 C3   0001    A          ADDD   £1
00498A FC04 10B3 1FAB    A          CMPD   BYTES
00499A FC08 27   07   FC11          BEQ    EXIT1
00500A FC0A FD   1FA9    A          STD    ADDR
00501A FC0D 30   01      A          LEAX   1,X
00502A FC0F 20   C2   FBD3          BRA    RPT
00503A FC11 86   0A      A EXIT1    LDA    £$0A      CE LOW  7805 =5V
00504A FC13 B7   203A    A          STA    P2DRB
00505A FC16 39                      RTS
00506                               *
00507                               *
00508A FC17 4F                      VERFY  CLRA
00509A FC18 B7   2039    A          STA    P2CRA
00510A FC1B B7   2038    A          STA    P2DRA     SET FOR INPUT
00511A FC1E 86   04      A          LDA    £4
00512A FC20 B7   2039    A          STA    P2CRA
00513A FC23 86   0B      A          LDA    £$0B      LED ON CE LOW 5V READ
00514A FC25 B7   203A    A          STA    P2DRB
00515A FC28 BE   1FA5    A          LDX    PSTART
00516A FC2B 7F   1FA9    A          CLR    ADDR
00517A FC2E 7F   1FAA    A          CLR    ADDR+1    RESET ADDR COUNTER
00518A FC31 FC   1FA9    A          LDD    ADDR
00519A FC34 F7   2030    A CVFY     STB    P1DRA
00520A FC37 B7   2032    A          STA    P1DRB
00521A FC3A B6   2030    A          LDA    P2DRA     OUTPUT ADDR & READ DATA
00522A FC3D A1   84      A          CMPA   ,X
00523A FC3F 26   12   FC53          BNE    ERR2      CHECK AGAINST MEMORY
00524A FC41 BC   1FA7    A          CMPX   END
00525A FC44 27   12   FC58          BEQ    EXIT2
00526A FC46 30   01      A          LEAX   1,X
00527A FC48 FC   1FA9    A          LDD    ADDR
00528A FC4B C3   0001    A          ADDD   £1
00529A FC4E FD   1FA9    A          STD    ADDR
00530A FC51 20   E1   FC34          BRA    CVFY
00531A FC53 86   01      A ERR2     LDA    £1
00532A FC55 B7   1FAE    A          STA    VERR
00533A FC58 39                      EXIT2  RTS
00534                               *
00535                               *
00536                               * SUBROUTINES
00537                               *
00538                               *
00539                               * OUTPUT A MESSAGE TO THE VDU
00540                               * MESSAGE ADDRESS IS IN THE INDEX REGISTER
```

```
00541              FC59     A OUTMES EQU    *
00542A FC59 1F     89       A XPDATA TFR    A,B       PRESERVE A REG
00543A FC5B A6     80       A        LDA    0,X+      NEXT CHAR
00544A FC5D B1     FD6C     A        CMPA   SEMICO    END OF MESSAGE?
00545A FC60 27     05   FC67        BEQ    OMRTS     YES
00546A FC62 BD     FD4E     A        JSR    OUTCH
00547A FC65 20     F2   FC59        BRA    XPDATA
00548A FC67 17     0095 FCFF OMRTS  LBSR   NLCR
00549A FC6A 1F     98       A        TFR    B,A       RESTORE A REGISTER
00550A FC6C 39                      RTS
00551                          *
00552                          * INPUTS A PARAMETER, LOOKS FOR THE FIRST NON-SPACE CHARACTER AND
00553                          * INPUTS THE NEXT 4 CHARACTERS.  IF AN INVALID DIGIT IS TYPED IN THE
N
00554                          * THE REST OF THE LINE IS SKIPPED AND CONTROL IS RETURNED TO THE
00555                          * MAIN PROGRAM, RATHER THAN THE CALLING CODE.
00556A FC6D 8D     01   FC73 XINADD BSR    INPARA
00557A FC6F 17     008D FCFF        LBSR   NLCR
00558A FC72 39                      RTS
00559A FC73 BD     FD21     A INPARA JSR    INCH
00560A FC76 B1     FD6D     A        CMPA   SPACE
00561A FC79 27     F8   FC73        BEQ    INPARA    IGNORE SPACES
00562A FC7B 7F     1F85     A FSTCHI CLR    ERFLAG
00563
00564A FC7E BD     FCE4     A        JSR    CNVERT
00565A FC81 BD     FCB8     A        JSR    CHOK
00566A FC84 48                      LSLA              *2
00567A FC85 48                      LSLA              *4
00568A FC86 48                      LSLA              *8
00569A FC87 48                      LSLA              *16
00570A FC88 34     02       A        PSHS   A
00571A FC8A BD     FD21     A        JSR    INCH
00572A FC8D BD     FCE4     A        JSR    CNVERT
00573A FC90 BD     FCB8     A        JSR    CHOK
00574A FC93 AB     E4       A        ADDA   ,S
00575A FC95 A7     E4       A        STA    ,S        M.S. BYTE ON STACK
00576A FC97 BD     FD21     A        JSR    INCH
00577A FC9A BD     FCE4     A        JSR    CNVERT
00578A FC9D BD     FCB8     A        JSR    CHOK
00579A FCA0 48                      LSLA              *2
00580A FCA1 48                      LSLA              *4
00581A FCA2 48                      LSLA              *8
00582A FCA3 48                      LSLA              *16
00583A FCA4 34     02       A        PSHS   A
00584A FCA6 BD     FD21     A        JSR    INCH
00585A FCA9 BD     FCE4     A        JSR    CNVERT
00586A FCAC BD     FCB8     A        JSR    CHOK
00587A FCAF AB     E0       A        ADDA   ,S+
00588A FCB1 1F     89       A        TFR    A,B       L.S. BYTE IN B
00589A FCB3 A6     E0       A        LDA    ,S+       M.S. BYTE IN A
00590A FCB5 ED     84       A        STD    ,X
00591A FCB7 39                      RTS
00592                          *
00593                          * EXAMINE THE ERROR FLAG AND OUTPUT A MESSAGE IF NECESSARY
```

```
00594                                  *
00595A FCB8 F6   1F85     A CHOK   LDB     ERFLAG
00596A FCBB C1   00       A        CMPB    £0
00597A FCBD 27   0C    FCCB        BEQ     POK
00598A FCBF BD   FCFF     A        JSR     NLCR
00599A FCC2 8E   FDB5     A        LDX     £PRMES
00600A FCC5 BD   FC59     A        JSR     OUTMES
00601A FCC8 7E   F863     A        JMP     START
00602A FCCB 39                     POK     RTS
00603                              *
00604                              * OUTPUT BYTE FROM A REG
00605                              *
00606A FCCC 34   02       A OBYTE  PSHS    A
00607A FCCE 44                     LSRA
00608A FCCF 44                     LSRA
00609A FCD0 44                     LSRA
00610A FCD1 44                     LSRA             ONLY M.S. NIBBLE LEFT
00611A FCD2 8E   FD72     A        LDX     £LOOKUP
00612A FCD5 A6   86       A        LDA     A,X      LOOKUP HEX DIGIT
00613A FCD7 BD   FD4E     A        JSR     OUTCH
00614A FCDA A6   E0       A        LDA     ,S+
00615A FCDC 84   0F       A        ANDA    £$F      ONLY L.S. NIBBLE LEFT
00616A FCDE A6   86       A        LDA     A,X      HEX DIGIT
00617A FCE0 BD   FD4E     A        JSR     OUTCH
00618
00619A FCE3 39                     RTS
00620                              *
00621                              * CONVERT THE CONTENTS OF THE A REGISTER FROM A CHARACTER TO THE
00622                              * EQUIVALENT HEX DIGIT
00623A FCE4 81   30       A CNVERT CMPA    £$30
00624A FCE6 2D   11    FCF9        BLT     CERR     CHAR < "0" , NOT VALID
00625A FCE8 81   39       A        CMPA    £$39
250T$$
00626A FCEA 2F   0A    FCF6        BLE     DECIMA   VALID "0" - "9"
00627A FCEC 81   41       A        CMPA    £$41
00628A FCEE 2D   09    FCF9        BLT     CERR     "9" < CHAR < "A", NOT VALID
00629A FCF0 81   46       A        CMPA    £$46
00630A FCF2 2E   05    FCF9        BGT     CERR     CHAR > "F", NOT VALID
00631A FCF4 80   07       A        SUBA    £7       CHAR'S NOW NUMBERED CONSECUTIVELY
00632A FCF6 80   30       A DECIMA SUBA    £$30     CONVERT TO A NUMBER
00633A FCF8 39                     RTS
00634A FCF9 C6   02       A CERR   LDB     £2
00635A FCFB F7   1F85     A        STB     ERFLAG   ERROR FLAG SET
00636A FCFE 39                     RTS
00637                              *
00638                              * OUTPUT NEWLINE
00639                              *
00640            FCFF     A XPCRLF EQU     *
00641A FCFF B6   FD66     A NLCR   LDA     CR
00642A FD02 BD   FD4E     A        JSR     OUTCH
00643A FD05 86   FD65     A        LDA     NL
00644A FD08 BD   FD4E     A        JSR     OUTCH
00645A FD0B 39                     RTS
00646                              *
```

```
00647                          * CHECK FOR BREAK KEY
00648                          *
00649A FD0C 8D   34    FD42 CHKBRK BSR    NOWAIT
00650A FD0E 81   1B       A         CMPA   £BREAK
00651A FD10 26   0E    FD20         BNE    CHRTS
00652A FD12 86   4C       A         LDA    £
00653A FD14 B7   202C     A         STA    HSHAKP
00654A FD17 8E   FE87     A         LDX    £ESCMES
00655A FD1A 17   FF3C FC59          LBSR   OUTMES
00656A FD1D 16   FB43 F863          LBRA   START
00657A FD20 39                CHRTS  RTS
00658                          *
00659                          * INPUT A CHARACTER
00660                          *
00661          FD21     A XINCH   EQU    *
00662A FD21 B6   2034     A INCH    LDA    ACIACS
00663A FD24 47                      ASRA
00664A FD25 24   FA    FD21         BCC    INCH      WAIT FOR CHAR RECIEVED
00665A FD27 B6   2035     A         LDA    ACIADA    READ CHAR
00666A FD2A 84   7F       A         ANDA   £$7F      RESET PARITY
00667A FD2C 81   7F       A         CMPA   £$7F
00668A FD2E 27   F1    FD21         BEQ    INCH      RUBOUT, IGNORE
00669A FD30 20   1C    FD4E         BRA    OUTCH     ECHO THE CHARACTER
00670                          *
00671                          * INPUT A CHARACTER BUT DON'T ECHO IT
00672                          *
00673A FD32 B6   2034     A NOECHO  LDA    ACIACS
00674A FD35 47                      ASRA
00675A FD36 24   FA    FD32         BCC    NOECHO    WAIT FOR CHARACTER
00676A FD38 B6   2035     A         LDA    ACIADA    READ CHAR
00677A FD3B 84   7F       A         ANDA   £$7F      RESET PARITY
00678A FD3D 81   7F       A         CMPA   £$7F
00679A FD3F 27   F1    FD32         BEQ    NOECHO    RUBOUT IGNORE
00680A FD41 39                      RTS
00681                          *
00682                          * INPUT A CHARACTER, BUT DON'T WAIT
00683                          *
00684A FD42 B6   2034     A NOWAIT  LDA    ACIACS
00685A FD45 47                      ASRA
00686A FD46 24   05    FD4D         BCC    NWRTS     NO KEY PRESSED
00687A FD48 B6   2035     A         LDA    ACIADA
00688A FD4B 84   7F       A         ANDA   £$7F      RESET PARITY
00689A FD4D 39                NWRTS  RTS
00690                          *
00691                          * OUTPUT A CHARACTER
00692                          *
00693A FD4E 34   04       A OUTCH   PSHS   B         SAVE B
00694A FD50 F6   2034     A OUTC1   LDB    ACIACS
00695A FD53 57                      ASRB
00696A FD54 57                      ASRB
00697A FD55 24   F9    FD50         BCC    OUTC1     XMIT NOT READY
00698A FD57 B7   2035     A         STA    ACIADA    OUTPUT CHAR
00699A FD5A 35   04       A         PULS   B
00700A FD5C 39                      RTS
```

```
00701                          *
00702                          * DELAY THE MOTOROLA.  THIS IS NEEDED BECAUSE THE AMD HAS
00703                          * A SLOWER CLOCK
00704                          *
00705A FD5D F6   FD82     A SRDLY  LDB     DELAY
00706A FD60 5A                SRDLP  DECB
00707A FD61 5D                       TSTB
00708A FD62 26   FC   FD60           BNE     SRDLP
00709A FD64 39                       RTS
00710                          *
00711                          *
00712                          * CONSTANT DATA
00713                          *
00714                          *
00715A FD65      0A       A NL      FCB     $A        NEW LINE
00716A FD66      0D       A CR      FCB     $D        CARRIAGE RETURN
00717A FD67      52       A R       FCC     "R"       READ COMMAND
00718A FD68      54       A T       FCC     "T"       TRANSFER COMMAND
00719A FD69      47       A G       FCC     "G"       GO COMMAND
00720A FD6A      2A       A PROMPT  FCC     "*"
00721A FD6B      3A       A COLON   FCC     ":"
00722A FD6C      3B       A SEMICO  FCC     ";"
00723A FD6D      20       A SPACE   FCC     " "
00724A FD6E      06       A CTRLG   FCB     6
00725A FD6F      14       A CTRLT   FCB     20
00726A FD70      0000     A FSA     FDB     0         (FSA) DEFAULT FOR 'B' AND 'W' COMMANDS
00727A FD72      30       A LOOKUP  FCC     "0123456789ABCDEF"HEX DIGITS
00728A FD82      FF       A DELAY   FCB     255
00729A FD83      27       A NOACT   FCB     $27       NO ACTION ON CONTROL PIA
00730A FD84      1F7F     A MAXMEM  FDB     $1F7F
00731A FD86      50       A STRTMS  FCC     "PRIME-AMD INTERFACE;"
00732A FD9A      49       A IVCMND  FCC     "INVALID COMMAND,  TRY AGAIN;"
00733A FDB5      49       A PRMES   FCC     "INVALID PARAMETER,  TRY AGAIN;"
00734A FDD2      4D       A RMES    FCC     "MEMORY READ COMPLETE;"
00735A FDE7      4D       A RDYMES  FCC     "MOTOROLA READY FOR TRANSFER;"
00736A FE03      54       A TEMS1   FCC     "TRANSFER ERROR,  MEMORY OVERFLOW;"
00737A FE23      54       A TEMS2   FCC     "TRANSFER ERROR,  INVALID CHARACTER;"
00738A FE45      54       A TMES    FCC     "TRANSFER COMPLETE;"
00739A FE57      0D       A INSMES  FCB     $D,$A
00740A FE59      49       A         FCC     "INTELLIGENT MEMORY MODE "
00741A FE71      0D       A         FCB     $D,$A
00742A FE73      50       A         FCC     "PRESS RESET TO EXIT;"
00743A FE87      45       A ESCMES  FCC     "EXIT FROM IM MODE;"
00744A FE99      0D       A MES61   FCB     $D,$A
00745A FE9B      49       A         FCC     "INSERT 16K PROM IN SOCKET WITH LED ON "
00746A FEC1      0D       A         FCB     $D,$A
00747A FEC3      53       A         FCC     "SPECIFY START ADDRESS ;"
00748A FEDA      53       A MES62   FCC     "SPECIFY END ADDRESS ;"
00749A FEEF      43       A MES63   FCC     "CHECKING PROM;"
00750A FEFD      50       A MES64   FCC     "PROGRAMMING PROM;"
00751A FF0E      50       A MES65   FCC     "PROGRAMMED AND VERIFIED;"
00752A FF26      50       A MES67   FCC     "PROM NOT COMPLETELY ERASED;"
00753A FF41      45       A MES69   FCC     "ERROR DETECTED IN VERIFY;"
00754            0000     A         IFEQ    PROM
```

```
00755A FFFE                    ORG     $FFFE
00756A FFFE      F800      A   FDB     PSTRT     RESTART ADDRESS
00757                          ENDC
00758                    *
00759                    * VARIABLE RAM DATA
00760                    *
00761            0000      A   IFEQ    PROM
00762A 1F80                    ORG     $1F80
00763                          ENDC
00764            0000      A   IFNE    PROM
00766                            ENDC
00767A 1F80      0002      A ADDRES RMB     2
00768A 1F82      0002      A ENDADR RMB     2
00769A 1F84      0001      A MSBYTE RMB     1
00770A 1F85      0001      A ERFLAG RMB     1
00771A 1F86      0014      A        RMB     20
00772A 1F9A      0001      A STACK  RMB     1
00773A 1F9B      0001      A MREQ   RMB     1
00774A 1F9C      0001      A RDFLG  RMB     1
00775A 1F9D      0001      A BYTE   RMB     1
00776A 1F9E      0001      A SYNC   RMB     1
00777A 1F9F      0002      A MEMSRT RMB     2
00778A 1FA1      0002      A SIZE   RMB     2
00779A 1FA3      0001      A NUMT   RMB     1
00780A 1FA4      0001      A CHIPNO RMB     1
00781A 1FA5      0002      A PSTART RMB     2
00782A 1FA7      0002      A END    RMB     2         PROM END ADDR
00783A 1FA9      0002      A ADDR   RMB     2         CURRENT PROM ADDR
00784A 1FAB      0002      A BYTES  RMB     2
00785A 1FAD      0001      A FERR   RMB     1         ERROR FLAG
00786A 1FAE      0001      A VERR   RMB     1         VERIFY ERROR FLAG
00787                          END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

```
00001                                * PROGRAM TO READ A SEQUENCE OF HEX NUMBERS FROM THE PRIME AND
00002                                *WRITE THEM TO MEMORY . THE NUMBERS ARRIVE IN SEQUENCES
00003                                * EACH SEQUENCE STARTS WITH CTRL G AND ENDS WITH CTRL T. EACH
00004                                * SEQUENCE IS PLACED IN A DIFFERENT BLOCK OF MEMORY.THERE ARE 12
00005                                * SEQUENCES.
00006                                *INCH      EQU NOECHO           CHANGE TO $F015 FOR OLD EXORCISER
00007              F018     A OUTCH  EQU    $F018
00008              FCF4     A ACIACS EQU    $FCF4
00009              FCF5     A ACIADA EQU    ACIACS+1
00010A 2000                          ORG    $2000
00011A 2000 4F                       CLRA
00012A 2001 B7   9000     A         STAA   ERFLAG   ERROR FLAG CLEAR
00013A 2004 10CE 208A     A         LDS    #STACK
00014                                * WAIT FOR CONTROL AND G BEFORE STARTING
00015A 2008 BD   2043     A WAITCG JSR    INCH
00016A 200B B1   2074     A         CMPA   CTRLG
00017A 200E 26   F8   2008          BNE    WAITCG
00018A 2010 7C   2073     A START  INC    CHIPNO   UPDATE THE SEQUENCE COUNT
00019A 2013 86   0C       A         LDA    #12      CHANGE TO '1' FOR PTRAN1
00020A 2015 B1   2073     A         CMPA   CHIPNO   HAVE ALL THE NO'S BEEN TRANSFERRED?
00021A 2018 2D   53   206D          BLT    FINISH   YES
00022A 201A FC   206F     A         LDD    SIZE
00023A 201D F3   2071     A         ADDD   MEMSRT   UPDATE THE MEMORY POINTER
00024A 2020 FD   2071     A         STD    MEMSRT   KEEP THE VALUE FOR NEXT TIME
00025A 2023 BE   2071     A         LDX    MEMSRT
00026A 2026 BD   2043     A FSTDIG JSR    INCH     INPUT M.S. DIGIT
00027A 2029 B1   2075     A         CMPA   CTRLT    IS THIS THE TERMINATION CHARACTER
00028A 202C 27   E2   2010          BEQ    START    YES IT IS
00029A 202E BD   2053     A         JSR    CNVERT   NO IT ISN'T
00030A 2031 48                       LSLA            M.S. NIBBLE, MULTIPLY BY 2
00031A 2032 48                       LSLA            "       "      4
00032A 2033 48                       LSLA            "       "        8
00033A 2034 48                       LSLA            "       "        16
00034A 2035 A7   84       A         STA    ,X
00035A 2037 BD   2043     A         JSR    INCH     INPUT L.S. DIGIT
00036A 203A BD   2053     A         JSR    CNVERT
00037A 203D AB   84       A         ADDA   ,X       M.S. DIGIT + L.S. DIGIT
00038A 203F A7   80       A         STA    ,X+
00039A 2041 20   E3   2026          BRA    FSTDIG
00040              2043     A INCH   EQU    *
00041A 2043 B6   FCF4     A NOECHO LDA    ACIACS
00042A 2046 47                       ASRA
00043A 2047 24   FA   2043          BCC    NOECHO   WAIT FOR CHARACTER
00044A 2049 B6   FCF5     A         LDA    ACIADA
00045A 204C 84   7F       A         ANDA   #$7F     RESET PARITY
00046A 204E 81   7F       A         CMPA   #$7F
00047A 2050 27   F1   2043          BEQ    NOECHO   IGNORE RUBOUT
00048A 2052 39                       RTS
00049                                * SUBROTINE TO CONVERT THE CONTENTS OF THE ACCUMULATOR FROM A CHARA
CTER
00050                                * CODE TO THE EQUIVALENT NUMBER.
00051A 2053 81   30       A CNVERT CMPA   #$30
00052A 2055 2D   11   2068          BLT    ERROR    CHAR < "0", NOT VALID
00053A 2057 81   39       A         CMPA   #$39
```

```
00054A 2059 2F   0A   2065        BLE    DECIMA   VALID CHARACTER "0"-"9"
00055A 205B 81   41     A        CMPA   £$41
00056A 205D 2D   09   2068        BLT    ERROR    "9"<CHAR<"A", NOT VALID
00057A 205F 81   46     A        CMPA   £$46
00058A 2061 2E   05   2068        BGT    ERROR    CHAR > "F", NOT VALID

00059A 2063 80   07     A        SUBA   £7       CHARACTERS NOW NUMBERED CONSECUTIVELY
00060A 2065 80   30     A DECIMA SUBA   £$30     CONVERT TO A NUMBER
00061A 2067 39                   RTS
00062A 2068 C6   01     A ERROR  LDB    £1
00063A 206A F7   9000   A        STB    ERFLAG   ERROR FLAG SET
00064A 206D 3F                FINISH SWI
00065A 206E      1A     A        FCB    $1A      MDENT
00066A 206F      0800   A SIZE   FDB    2*1024   CHIP SIZE
00067A 2071      2800   A MEMSRT FDB    $3000-(2*1024)
00068A 2073      00     A CHIPNO FCB    0        SEQUENCE COUNT
00069A 2074      06     A CTRLG  FCB    6
00070A 2075      14     A CTRLT  FCB    20
00071A 2076      0014   A        RMB    20
00072A 208A      00     A STACK  FCB    0
00073A 3000                      ORG    $3000
00074A 3000      6000   A        RMB    12*2*1024 BUFFER AREA
00075A 9000      0001   A ERFLAG RMB    1
00076                            END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

Appendix 6

Processor Logic Diagrams

CPU
Sheet 1 16-Bit Computer ALU

FROM SHEET 5

EN IMMED (M48)

U19 74S244

U20 74S244

IMMEDIATE GATE

DA Bus

DB Bus

ALU

U21 74S240

PSW

U22 74S244

FROM SH 4

TEST 1

TEST 2

TEST 3

TEST 4

TEST (M30)

TEST (M31)

TEST (M32)

TEST (M33)

CARRY IN

1k

+5V

U5 Am2904

TEST STATUS MULTI-PLEXER

Sheet 5

Sheet 4

U1 Am2903

U2 Am2903

U3 Am2903

U4 Am2903

Vcc = 36
GND = 13

CARRY IN

U18 25LS2519

EN PSW

CPC

CEN (M29)

OEY (M34)

SHIFTEN (M43)

CEM (M28)

U30 74S253

WORD/BYTE MUX

OEY (M

EA (M

OEB (M

YBUS

74S32 U31

74S32 U31

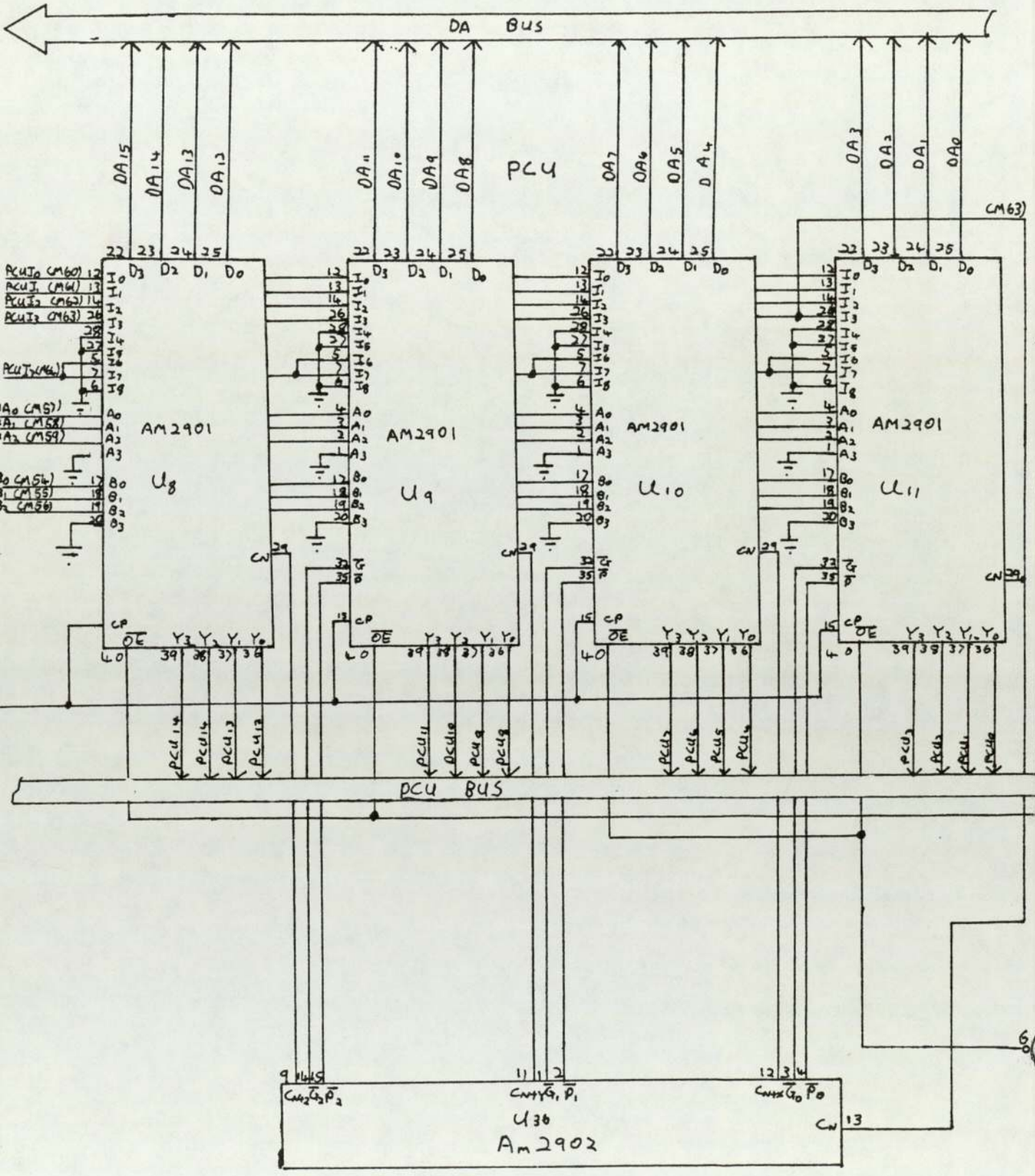
U29 Am2902

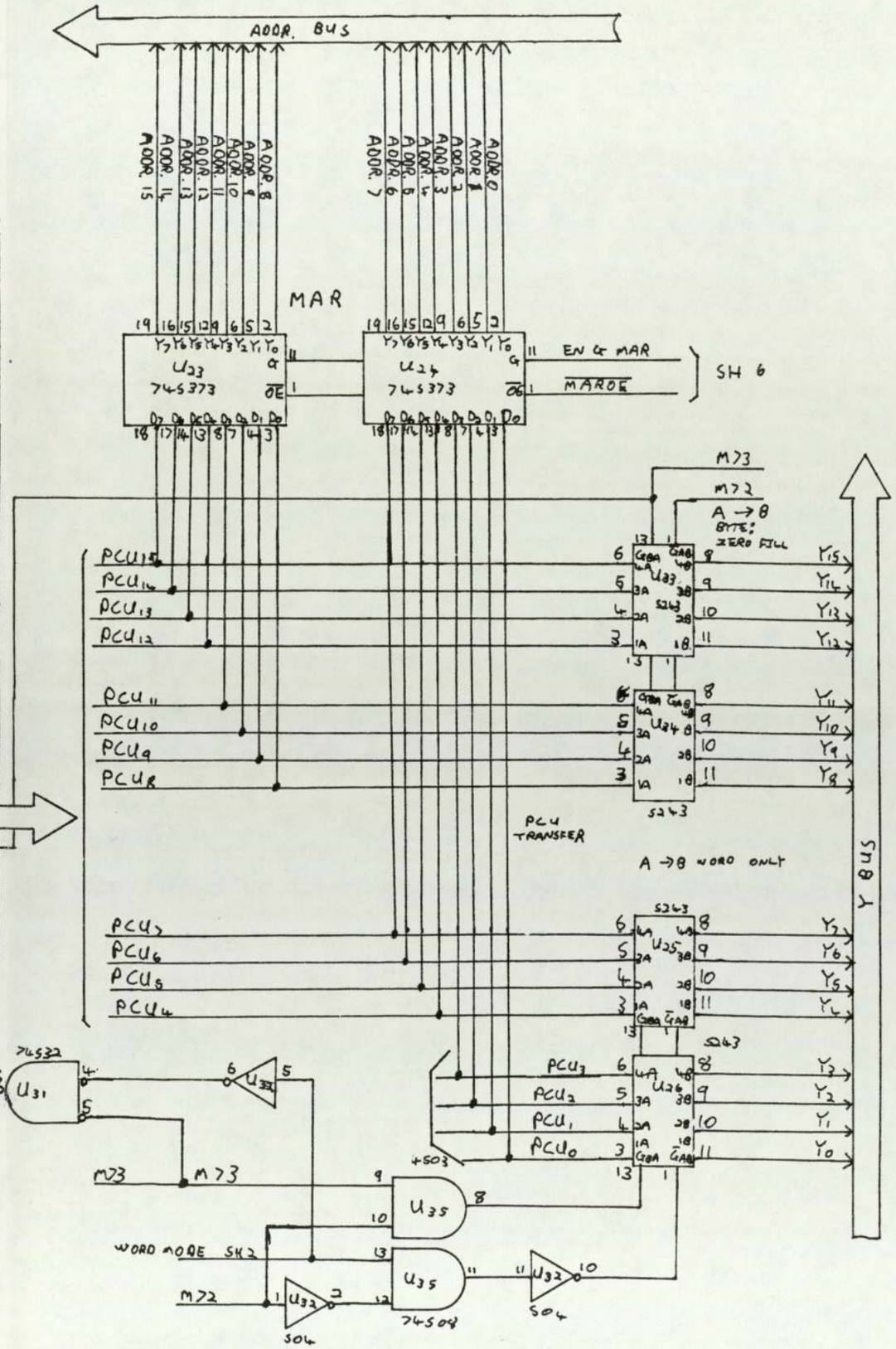
U32 74S04

WORD (M40)

WORD MODE TO SH3

16-Bit Computer PCU Memory Address Register





16-Bit Computer Data Path

CPU Sheet 3

DATA BUS

D REG.

Z REG.

TO SH5

Y BUS

TRANSFER REG.

ZI REG.

20 REG.

NOT USED

TO DECODER SH3

NOT USED

DA BUS

DA BUS

I REG.

TO AM2903 SH 2

A=B TO SH5

TO AM2903 SH 2

TO AM2904 SH 2

CPU
Sheet 4

FROM SH4

M66 SHIFT EN

U97 74608

U78 74LS244

U88 LS244

U52 2716

DECODER (MAPPING PROM)

RESET

74S00 U70

INTERRUPT REQ

500 U70

+5V

OBEZT (M20)

TEST TREE

TEST I0 (M30)
TEST I1 (M31)
TEST I2 (M32)

74S374 U87

25S261 U76

SH2 TEST

NAC0 (M19)
NAC1 (M18)
NAC2 (M17)
NAC3 (M16)

M11 M10 M9 M8 M7 M6 M5 M4 M3 M2 M1 M0

U51 Am2910

Vcc=10
GND=30

MAPEN

VECTEN

74S10 U48

PL

U60 27S19

VECTOR DECODER

U69 74LS374

SH7 INTREQ
VEC2
VEC1
VEC0

CCEN (M41)

NOT USED

FULL

CI

U62 2716 U80 2716 U90 2716 U54 2716 U63 2716 U81 2716 U91 2716 U55 2716 U64 2716 U82 2716 U92 2716

U68 U75 U70 U71 U74 U89 U99 U100 U101 U102 U103 LS374

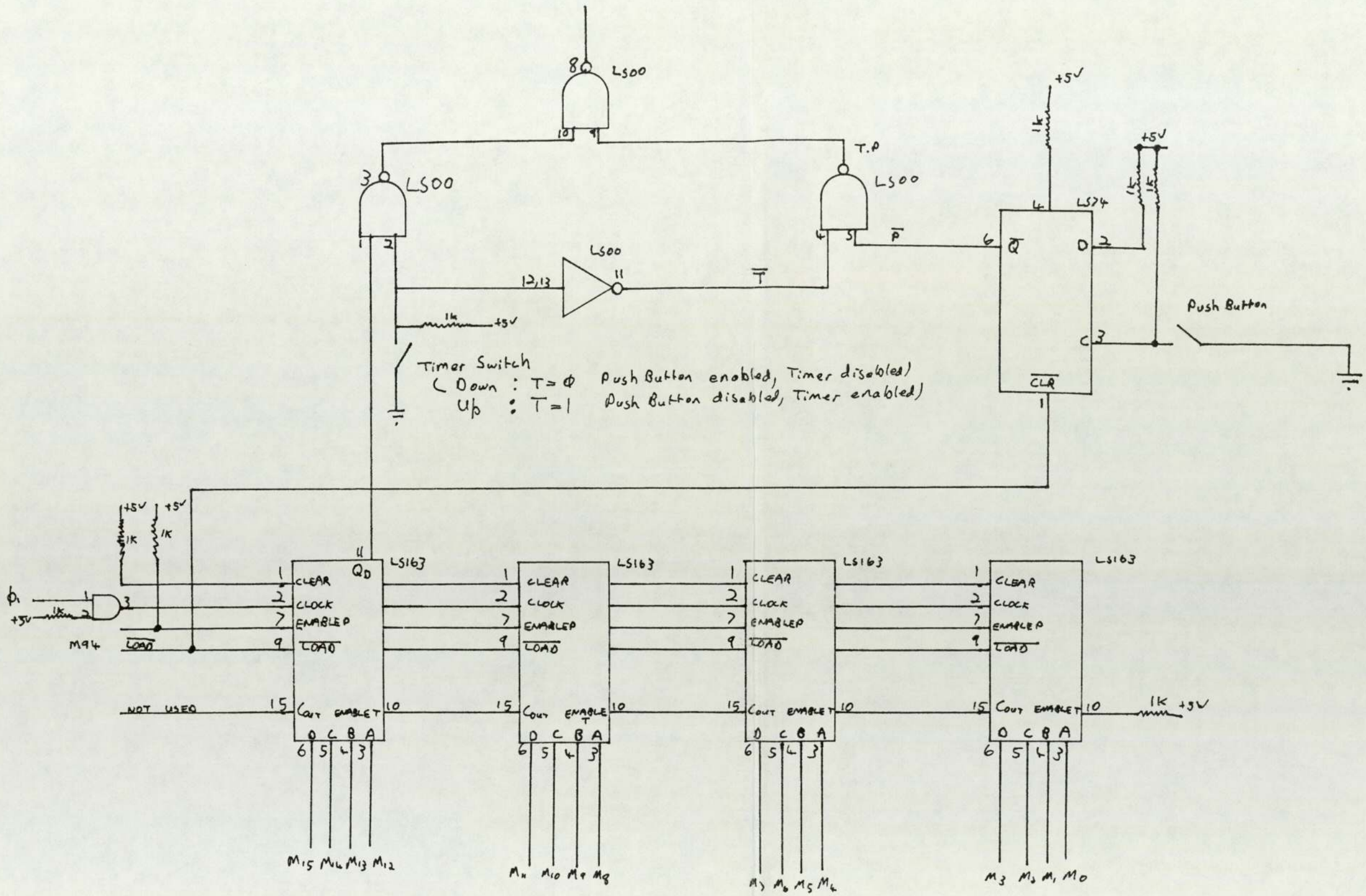
U50 7407

16-Bit Computer Microprogram Memory

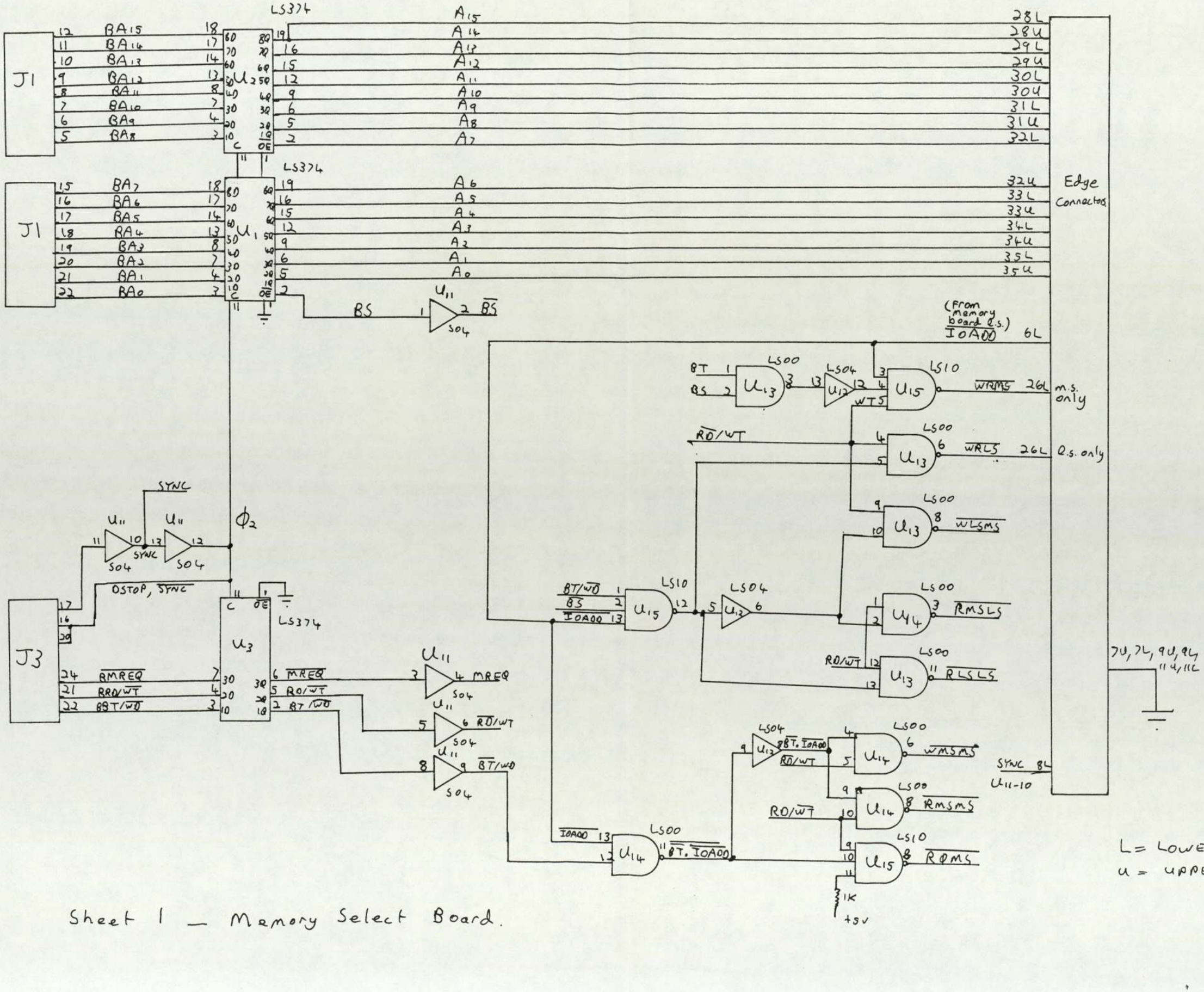
Memory and Clock Control.

CPU
Sheet 5

20MHz
ADDR ACCP
ENΦ1
RESET
WRTFF
MAREQ
DREQFF
REQFF
RESET SWITCH
SYSTEM RESET
IRST
MREQ FF
LO2FF
LO2O FF
ENO
ENO MAR TO SH 3
LO2I TO SH 4
LO2O TO SH L
DATA REQ
WD/BYTE
DSTOP
20 MHz CLOCK
MWRITE
REQB (MSI)
ENΦ1
MAER (MSI) BP
RED BUS
B65
B66
B67
B68
B69
B76
B77
B64
TO SH 4
LO0
LO1
LOT
(CM70) LO0
(CM69) ZI → I
(CM62) LOTREG
(CM92) BP → ZI
Pin 14, U53
U42 74S112
U45 74S112
U46 74S112
U47 74S112
U48 74S112
U49 74S112
U67 74S112
U68 74S112
U74 74S112
U85 74S74
U45 74S74
U63-11
U63-10
Φ1
Φ2
+5V



Sheet 6 Timer Interrupt Unit.



L = LOWER PIN
U = UPPER PIN

Sheet 1 — Memory Select Board.

J2

BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7

SYNC
U11-12

WRLS

LS373 U4
LS373 U5
S373 U6
LS373 U7
LS373 U8
S373 U9
LS374 U10

WLSMS
WMSMS
ENU7
ENU8
ENU9
ENU10

SYNC
U11-10

M.S. edge connector

D8 17L
D9 17U
D10 18L
D11 18U
D12 19L
D13 19U
D14 20L
D15 20U

L.S. edge connector

D0 1L
D1 1U
D2 18L
D3 18U
D4 19L
D5 19U
D6 20L
D7 20U

L = LOWER PIN
U = UPPER PIN

LS32 U16

RLSLS
ENU7

RMSLS
ENU8

RMSMS
ENU9

ROMS
SYNC
ENU10

U11-12

Sheet 2 - Memory Select Board.

TAL C1/C2
MHz 18
MHz 20
MHz 24

XTAL 1

D1 IN4001

R5 10K

470 R19

C32 100µ

R9 R11 R12

R2Φ R2 R3

3K

Processor Socket Empty.

5C

5B

IRQ

NMI

FIRQ

HALT

MRDY

DMA/ BREQ

470

R13 R8 R1Φ

R14 R4 R1

WR- Φ1

BS

BA

Q

E

R/W

D0 D1 D2 D3 D4 D5 D6 D7

A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15

XTAL

EXTAL

XTAL

RESET

C2 C1

R26

7D

2B

7Φ

6E

3B

4B

9B

9C

8B

1B

7B, 7C are all empty sockets

0·01 4·7µ

1,2,3,4 0V

11,12 +12V

7,8 -12V

C33 C30 C31

4·7µ 0·01 0·01

C27 0·01 C28 10µ

R6 4k7 0·47 C29

COM. IN OUT 79Φ5

Pullup resistors

DΦ	RMB 2
D1	RMB 8
A2	RMB 1
A3	RMA 8
A4	RMA 7
A5	RMA 6
A6	RMA 5
A7	RMA 4
A8	RMA 3
A9	RMA 2
A10	RMA 1
A11	RMB 7
A12	RMB 6
A13	RMB 3
A14	RMB 4
A15	RMB 5

4k7 RMA RMB

From Sheet 4

To processor Socket

To processor Socket

IOAOD-

83 WR-

47 Φ1

— — — represents wire link modifications.

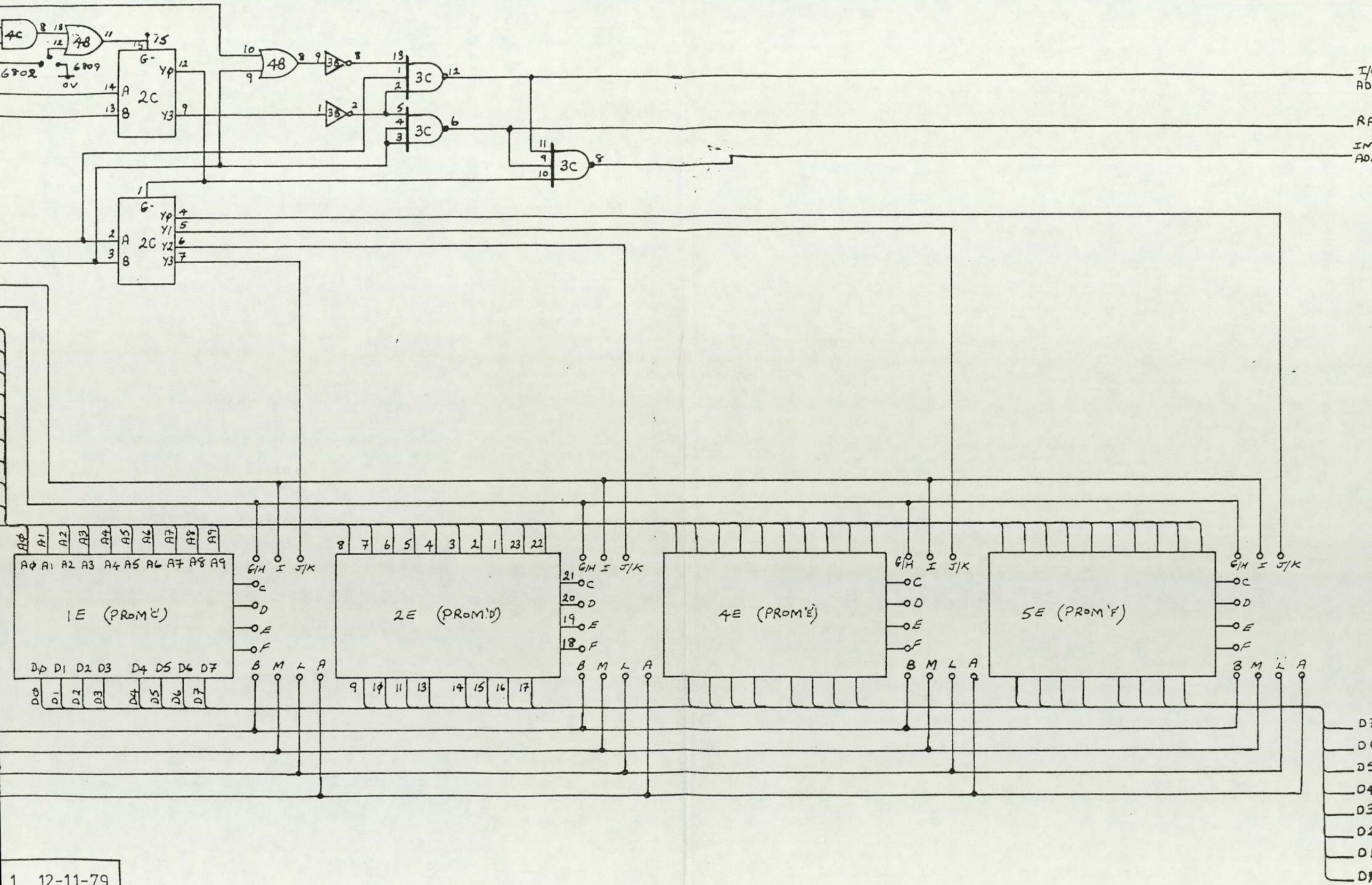
1	12-11-79
2	10-08-80

CPU BOARD BUFFERS

STUART JAMES SYSTEMS LIMITED

SJS790801

SHEET 1 OF



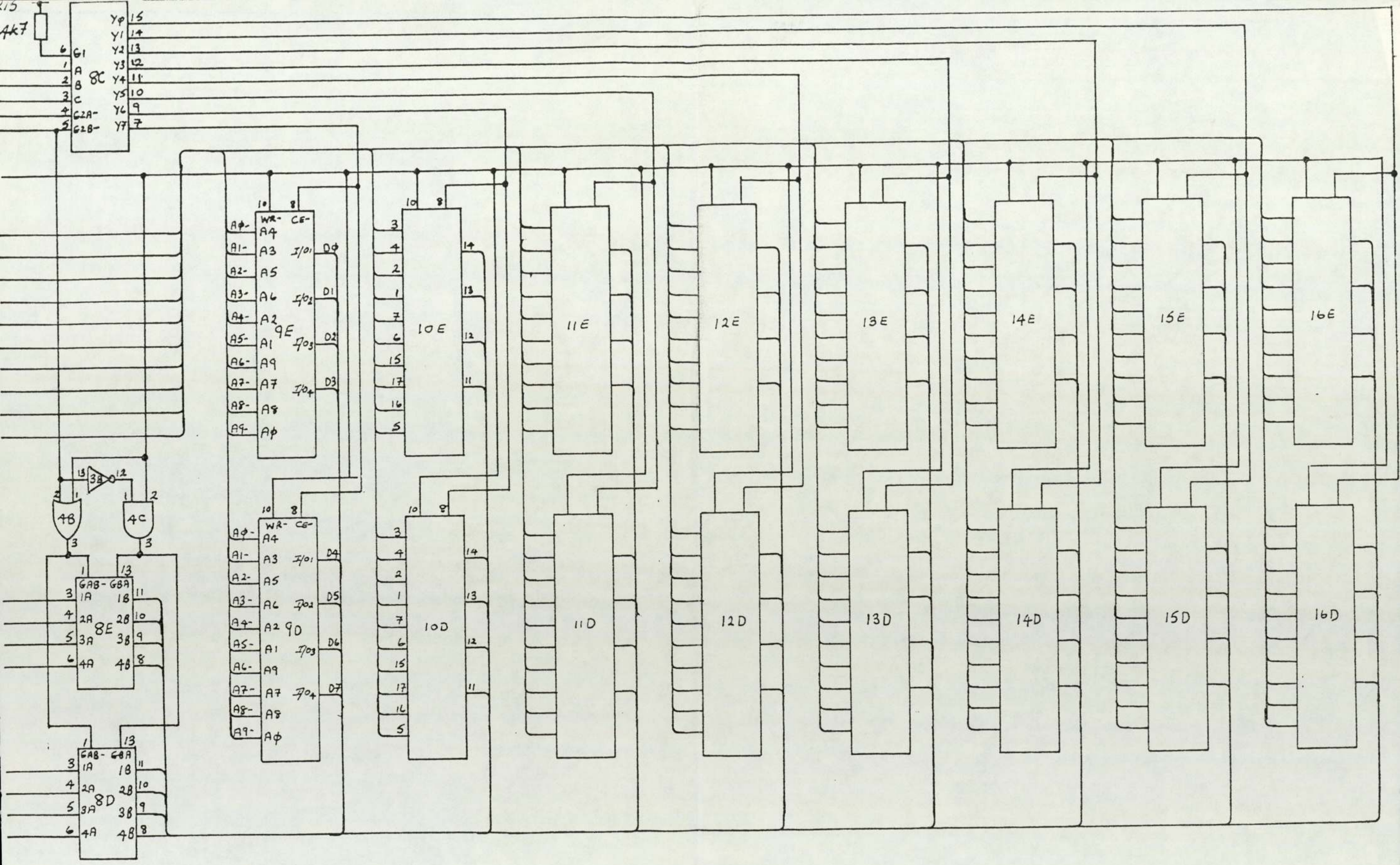
1	12-11-79
2	10-08-80

CPU BOARD

STUART JAMES SYSTEMS LIMITED

SJS790801

SHEET 2 OF



1	12-11-79
2	10-08-80

PU BOARD
RAM

 STUART JAMES SYSTEMS LIMITED

SJS790801

SHEET 3
OF

+5v

R22 1K7

R21 1K7

7E

6 G1 — Y1 14
1 A — Y2 13
2 B — Y3 12
3 C — Y4 11
— Y5 10
4 G2A — Y6 9
5 G2B — Y7 7

AC

SP

A0 A1

D0 D1 D2 D3 D4 D5 D6 D7

D0 D1 D2 D3 D4 D5 D6 D7 RS0 RS1 CS0 CS1 R/W E RES.

2F (PIA 1) >2φ20 >2φ21 >2φ22 >2φ23 IRQA IRQB

PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 CA1 CA2 PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7 CB1 CB2

21 19 17 15 23 6 8 10 12 3
22 20 18 16 24 5 7 9 11 4

13,14 +5v
1,2 25,26 0v

J1

33 32 31 30 29 28 27 26 25 24 23 22 21 20 34

5F (PIA2) >2φ24 >2φ25 >2φ26 >2φ27

38
37

2 3 4 5 6 7 8 9 40 39 10 11 12 13 14 15 16 17 18 19

J2

8F (PIA3) >2φ2C >2φ2D >2φ2E >2φ2F

J3

11F (PIA4) >2φ30 >2φ31 >2φ32 >2φ33

J4

14F (PIA5) >2φ38 >2φ39 >2φ3A >2φ3B

J5

1	12-11-79
2	10-08-80

Pin numbers shown are for option A mounting of flat flex plug. To obtain option B pin numbers subtract above numbers from 27.

CPU BOARD | STUART JAMES SYSTEMS LIMITED | SJS790801 | SHEET OF

CTS AS232 23,24 | DCD AS232 21,22 | RX DATA AS232 19,20 | J6 | RX LOOP- 15,16 | RX LOOP+ 17,18

ACIA >2034 >2035

13C

D0 22 | D1 21 | D2 20 | D3 19 | D4 18 | D5 17 | D6 16 | D7 15

E 14 | R/W 13 | RS 11 | CS2- 9 | CS1 10 | CS0 8 | IRQ- 7

CTS- 24 | DCD- 23 | RX DATA 2 | RX CLK 3 | TX CLK 4 | TX DATA 6 | RTS- 5

16C | 16B | 14B | R17 +5V 4K7 | C4 100 | H | I | J

R18 470 +12V | Z2

XTAL 2. 6·144 MHz | R25 680 | 14B | 330 C3 0V

11B | 12B | 11C

SPEED SELECT >2028 | 12C

D0 1 | D1 19 | D2 18 | D3 17 | D4 14 | D5 13 | D6 12 | D7 11

A | B | C | CP 7 | RE- 3 | E 15 | Y | CLR 8 | OE- ME- PDL 16 2 9 0V

14C | 16A | R23 470 +5V | R24 470 +12V | Z1

M | L | K | 0V

3,4 RTS AS232 | 7,8 TX DATA AS232 | 5,6 MODEM CONTROL AS232 | J6 | 11,12 TX LOOP- | 9,10 TX LOOP+

IRQ-

14B | R16 4K7 +5V

1	12-11-79
2	10-08-80

CPU BOARD
L I/O

 STUART JAMES SYSTEMS LIMITED

SJS790801

SHEET

Appendix 7

P-code Description

The Concurrent and Sequential Pascal compilers produce P-code. This is a zero-addressing, stack-machine instruction set. Accordingly, the Super Sixteen processor is microcoded to execute this code. The best way to document it is to define the P-code itself and the machine architecture that it expects in order to run. Accordingly, this appendix is presented in a "programmers manual" format. The appendix, therefore, describes the P-code machine and then goes on to detail each individual instruction.

7.1 The P-code Machine

P-code Registers

Five address registers are required, these are as follows:-

H register - Points to the current processes heap.

G register - Points to the base of the current processes global variables.

B register - Points to the base of the current processes local variables.

SP - The Stack Pointer.

PC - The Program Counter.

In addition several scratch registers are used and a PH (Process Head) register which points to the current process head (see

Appendix 8). The HREG, GREG and BREG are contained within the ALU. The SP, PC and PH registers are contained within the PCU.

P-code Addressing

The addressing mechanism is in units of bytes, despite a 16 bit word length. All words are accessed with even addresses:-

Word 0	0	1
Word 1	2	3
Word 2	4	5
Word 3	6	7

Data Types

Real numbers are represented by four words. The first three are the mantissa and the last one is the exponent.

Sets are represented by 8 words (128 bits). For each member of a set which is present its corresponding bit position is set to 1.

Strings are stored in packed form, i.e. two characters to a word.

Instruction Formats

Each instruction op-code requires one word. Also, each operand occupies one word. Thus, for example, an instruction with two operands would occupy three contiguous words. Therefore, when the instruction is executed, the Program Counter points to the first operand, the program counter plus two points to the second operand and so on.

Attributes

Each process has a set of attributes associated with it. For a description of these see Appendix 8.

7.2 The P-code Instruction Set

This section contains a complete list of all P-code instructions. Some of them are only called in Concurrent and Sequential Pascal programs. However, some op-codes refer to Concurrent operations, these are suffixed with '(*CP)' to indicate this. A key to the notation used is given below.

ST (x) - refers to the store (memory) location whose address is x.

e.g. ST (SP) refers to the top stack word.

ST (SP + 2) refers to the top but one stack word.

ST (ST (SP)) refers to the memory location pointed to by the top stack word.

x <- y means that the contents of y is put into x.

e.g. ST (SP) <- PC indicates that the Program Counter value has been put into the top stack word.

SP <- SP - 2 means that the stack pointer has been decremented by 2.

Exceptions

Many P-code programs generate exceptions if invalid data is encountered. There are seven different exceptions, their codes

are:-

TERMINATED exception - 0

OVERFLOW exception - 1

POINTERERROR exception - 2

RANGEERROR exception - 3

VARIANTEROOR exception - 4

HEAPLIMIT exception - 5

STACKLIMIT exception - 6

When an exception occurs, the following actions take place:-

Operation:

LINE attribute <- ST(B)

B <- G

G <- ST (B+6)

PC <- ST (B+8)

SP <- ST (B+2)

B <- ST (B+4)

Set mode to system

Description:

Sets the process attribute LINE to the contents of the address pointed to by the B register. The stack is then reset as follows. The word following the word pointed to by the Global Base Register G contains the new stack pointer value. The word after that contains the new B register value. The new G register value is in the next word and the word after this contains the new program counter value, i.e. a return address. The mode is set to system. If the process

is already in system mode (i.e. it is executing Concurrent P-code and not Sequential code) then the error is catastrophic and the fault diagnosis microcode is called. See Fig. 7.1.

ABR ABSOLUTE REAL

(No operands)

Description:

Converts the real number stored on the top four words of the stack into its absolute value.

ABS ABSOLUTE VALUE OF A WORD

(No operands)

Operation:

```
IF ST (SP) < 0
THEN ST (SP) <- -ST (SP)
```

Description:

The top word of the stack is converted into its absolute value. If the number is negative it is negated to give a positive value. If this produces an overflow then an OVERFLOW exception is generated.

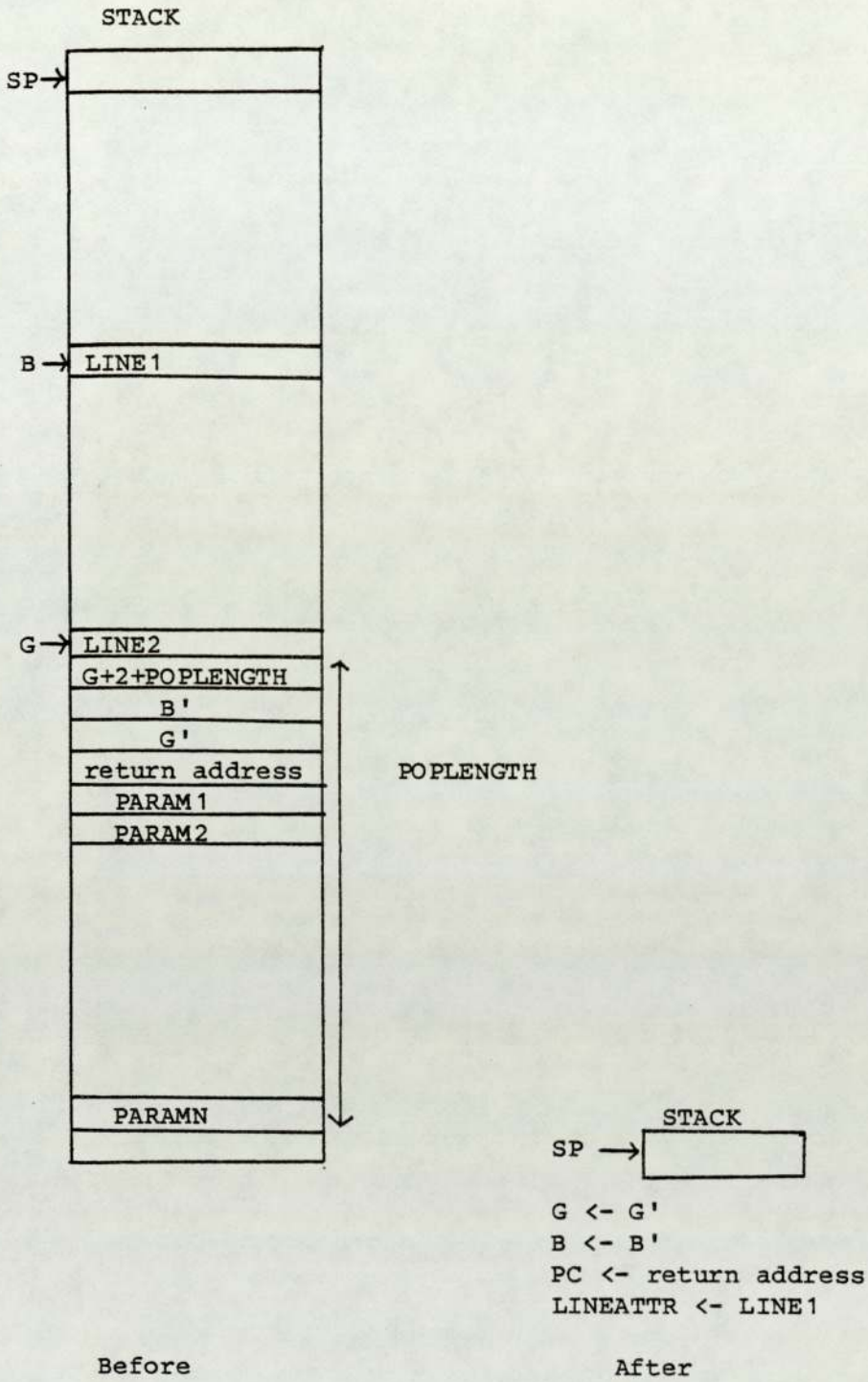


Fig. 7.1. Exception handling.

ADD — ADD WORD

(No operands)

Operation:

ST (SP +2) <- ST (SP + 2) + ST (SP)

SP <- SP + 2

Description:

The top and the top but one stack words are pulled off the stack and added together. The result is pushed onto the stack.

ADR — ADD REAL

(No operands)

Operation:

SP <- SP + 8

Description:

The top four words on the stack are treated as a real number and added to the next four words on the stack where the result is stored. The stack pointer is incremented by 8, thus deleting the top real number.

ANS AND SET

(No operands)

Operation:

```
ST (SP + 16) <- ST (SP) AND ST (SP + 16)
ST (SP + 18) <- ST (SP + 2) AND ST (SP + 18)
ST (SP + 20) <- ST (SP + 4) AND ST (SP + 20)
ST (SP + 22) <- ST (SP + 6) AND ST (SP + 22)
ST (SP + 24) <- ST (SP + 8) AND ST (SP + 24)
ST (SP + 26) <- ST (SP + 10) AND ST (SP + 26)
ST (SP + 28) <- ST (SP + 12) AND ST (SP + 28)
ST (SP + 30) <- ST (SP + 14) AND ST (SP + 30)
```

Description:

The first eight words on the stack are ANDed with the next eight words. The results are stored in the bottom eight of the top sixteen stack words. The stack pointer is incremented by sixteen, thus deleting the top eight words. See Fig. 7.2.

ANW AND WORD

(No operands)

Operation:

```
ST (SP + 2) <- ST (SP + 2) AND ST (SP)
SP <- SP + 2
```

Description:

The top two stack words are popped and ANDed together. The result is pushed onto the stack.

ATR(*CP) GET ATTRIBUTE

(No operands)

Operation:

ST (SP) <- ST (PH) + ST (SP)

Description:

The stack top contains an index to a process attribute. The base address of the process attributes is pointed to by the process head register, PH. The stack top value is replaced by the value of that attribute.

BGC(*CP) BEGIN CLASS

(4 Operands: STACKLENGTH, POPLENGTH = 10,

LINE, VARLENGTH = 0)

Description:

This instruction allocates space on the stack for the initial statement of a class. It is the same as the "ETC" Enter Class instruction. However, it would normally be called with POPLENGTH and VARLENGTH set to constant values of 10 and 0 respectively. This is because an initial statement has no parameters or temporary variables.

BGM(*CP) BEGIN MONITOR

(4 Arguments: STACKLENGTH, POPLENGTH = 10,

LINE, VARLENGTH = 0)

Description:

This instruction is similar to the "ENM" Enter Monitor instruction except that it is used to enter the initial statement of a monitor rather than a routine. POPLENGTH and VARLENGTH will

always have constant values of 10 and 0 respectively since there are no parameters or temporary variables needed.

BGP(*CP) BEGIN PROCESS

(1 Operand: LINE)

Operation:

ST (B) <- ST (PC)

Description:

This instruction is identical to the "NLN" New Line instruction.

BLS BUILD SET

(No operands)

Operation:

ST (SP + 2 + (ST (SP) / 8)) <-

ST (SP + 2 + (ST (SP) / 8)) OR (1 SHL (ST (SP) MOD 8))

SP <- SP + 2

Description:

The top word of the stack contains a number representing an entry in a set. The next eight words contain a set. The appropriate member of the set is set to 1. The stack pointer is incremented by 2, thus removing the top number. The set, however, is left on the stack. If the set member is not in the range 0 <= n <= 127 then a RANGEERROR exception is generated. See Fig. 7.3.

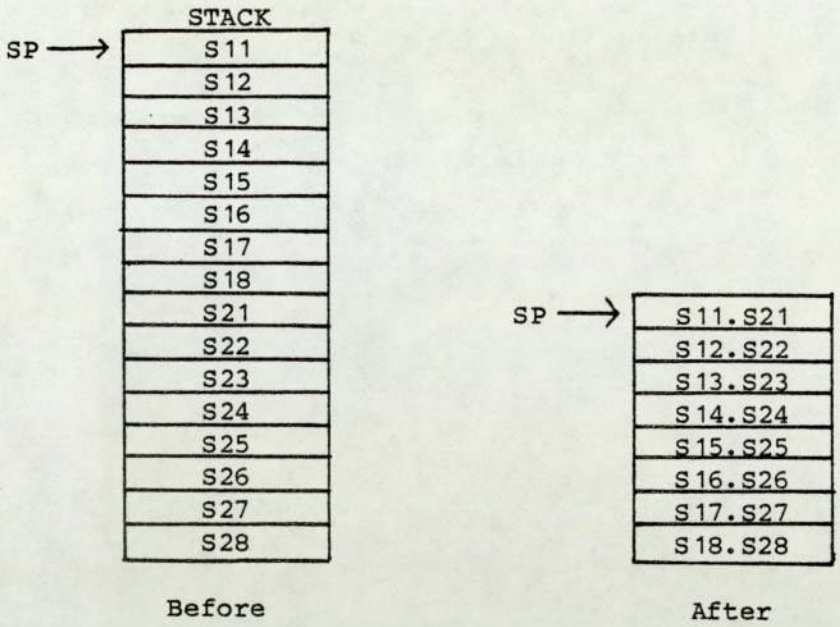


Fig. 7.2. The AND SET instruction.

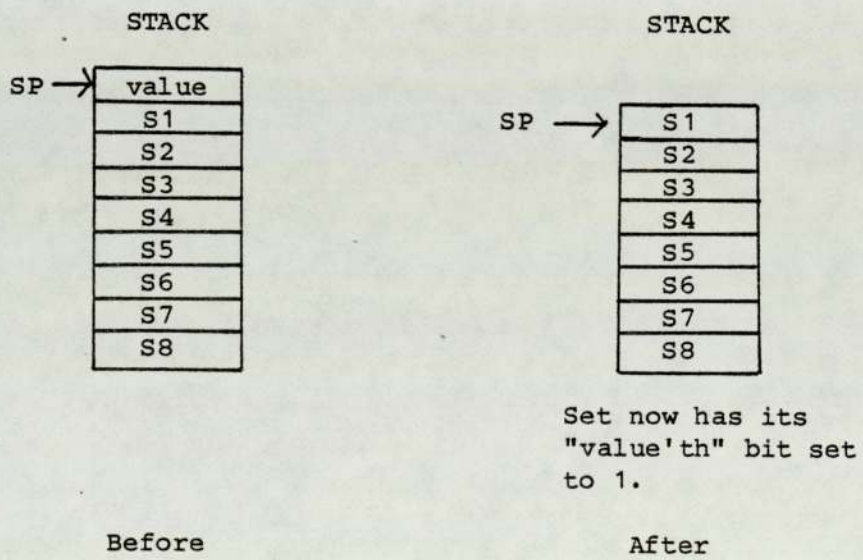


Fig. 7.3. The BUILD SET instruction.

CALL — CALL PROCEDURE

(1 Operand: DISTANCE)

Operation:

```
ST (SP - 2) <- PC
SP <- SP - 2
PC <- PC + ST (PC)
```

Description:

A procedure call is made. In other words, the current program counter value is pushed onto the stack and a relative jump of DISTANCE bytes is made.

CEQ — COMPARE WORD EQUAL TO

(No operands)

Operation:

```
IF ST (SP + 2) = ST (SP)
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

The top word of the stack is compared with the top but one stack word. The two words are also pulled off the stack. If they are equal then a one is pushed onto the stack. If they are unequal then a zero is pushed onto the stack.

CGE COMPARE WORD GREATER THAN

OR EQUAL TO

(No operands)

Operation:

```
IF ST (SP + 2) >= ST (SP)
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

The top word of the stack is compared with the top but one stack word. Both values are pulled off the stack. If the latter is greater than or equal to the former then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CGT COMPARE WORD GREATER THAN

(No operands)

Operation:

```
IF ST (SP + 2) > ST (SP)
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

The top word of the stack is compared with the top but one stack word. Both values are pulled off the stack. If the latter is greater than the former then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CLE COMPARE WORD LESS THAN

OR EQUAL TO

(No operands)

Operation:

```
IF ST (SP + 2) <= ST (SP)
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

The top word of the stack is compared with the top but one stack word. Both values are pulled off the stack. If the latter is less than or equal to the former then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CLP(*CP) CALL PROGRAM

(No operands)

Operation:

```
ST (SP - 2) <- PC
PC <- ST (ST (ST (SP)) - 1) * 2 + ST (ST (SP) + 1)) + 8
ST (SP) <- PC + ST (PC - 6)
SP <- SP - 2
```

Description:

Calls a Sequential Pascal program. The stack top contains the address of a program descriptor structure. This structure points to a table of program addresses. The structure contains an index to the table and also its base address. The program counter is set to the value of the index element of the table plus the base address of the table plus 8 (to skip past the 4 control words). The stack top contents is replaced with the program's large constant address which

is formed by adding the start of the program address to the second word of the program. The old program counter (return address) is pushed onto the stack. See Fig. 7.4.

CLT COMPARE WORD LESS THAN

(No operands)

Operation:

```
IF ST (SP + 2) < ST (SP)
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

The top word on the stack is compared with the top but one stack word. Both values are pulled off the stack. If the latter is less than the former then the value one is pushed onto the stack, otherwise zero is pushed onto the stack.

CNE COMPARE WORD NOT EQUAL

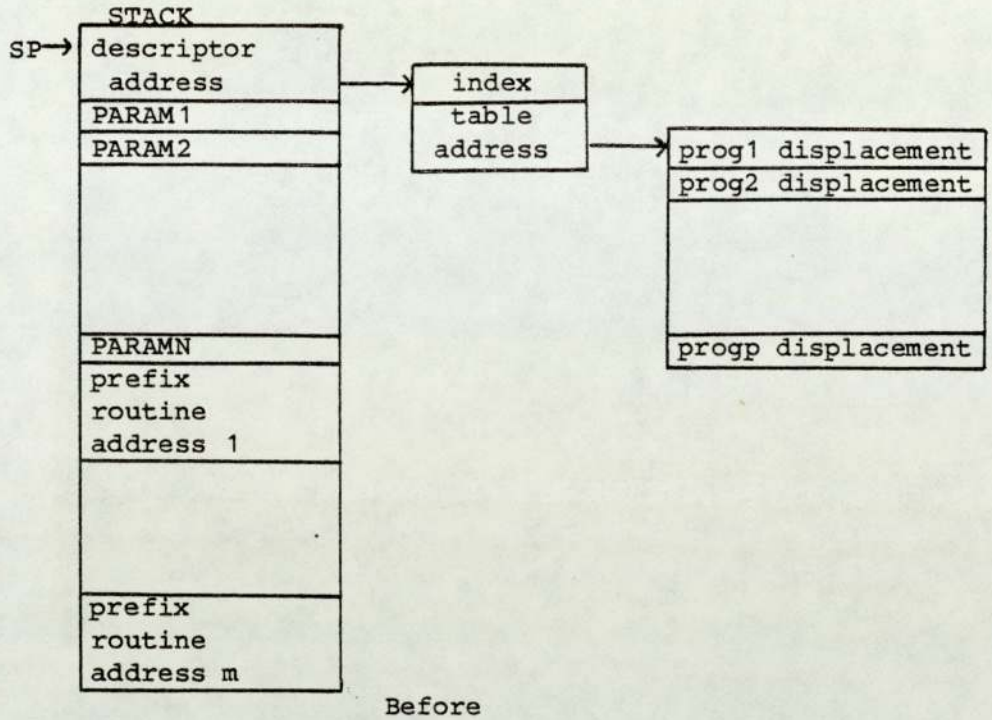
(No operands)

Operation:

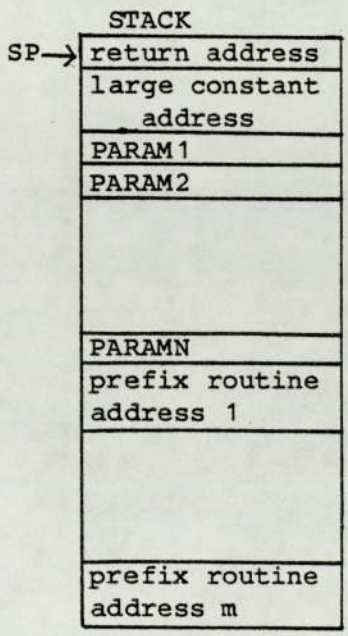
```
IF ST (SP + 2) NOT = ST (SP)
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

The top word of the stack is compared with the top but one stack word. Both words are pulled off the stack. If the latter is not equal to the former then a one is pushed onto the stack, otherwise a



Before



PC <- table address + prog index displacement + 8

After

Fig. 7.4. The CALL PROGRAM Instruction.

zero is pushed onto the stack.

CNT(*CP) CONTINUE PROCESS

(No operands)

Description:

This instruction continues a process that has previously been delayed by a DLY instruction. See Appendix 8.

CPB COPY BYTE

(No operands)

Operation:

ST (ST (SP + 2)) <- l.s. byte ST (SP)

SP <- SP + 4

Description:

The least significant byte of the top stack word is written to the address contained in the top but one stack word. Only one byte is copied and the other byte in the destination word is unaffected. The stack pointer is incremented by four, thus deleting the top two stack entries. See Fig. 7.5.

CPR COPY REAL

(No operands)

Operation:

```
ST (ST (SP + 8)) <- ST (SP)
ST (ST (SP + 8) + 2) <- ST (SP + 2)
ST (ST (SP + 8) + 4) <- ST (SP + 4)
ST (ST (SP + 8) + 6) <- ST (SP + 6)
SP <- SP + 10
```

Description:

A real number is transferred from the top four stack words to a set of locations whose base address is stored in the fifth stack word. The stack pointer is incremented by ten, thus deleting the real number and the address from the stack. See Fig. 7.6.

CPS COPY SET

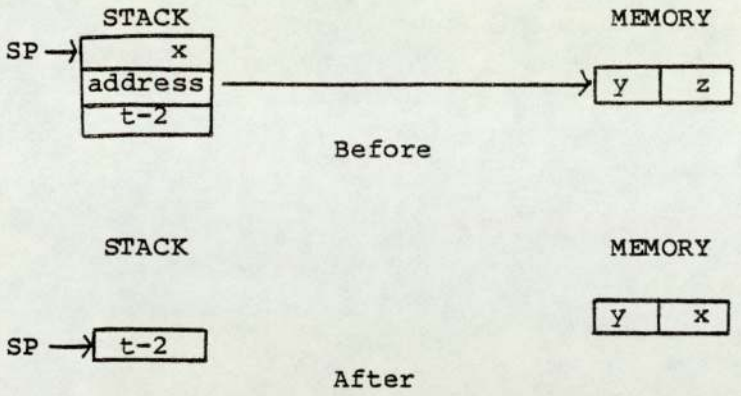
(No operands)

Operation:

```
ST (ST (SP + 16)) <- ST (SP)
ST (ST (SP + 16) + 2) <- ST (SP + 2)
ST (ST (SP + 16) + 4) <- ST (SP + 4)
ST (ST (SP + 16) + 6) <- ST (SP + 6)
ST (ST (SP + 16) + 8) <- ST (SP + 8)
ST (ST (SP + 16) + 10) <- ST (SP + 10)
ST (ST (SP + 16) + 12) <- ST (SP + 12)
ST (ST (SP + 16) + 14) <- ST (SP + 14)
SP <- SP + 18
```

Description:

A set is transferred from the top eight stack words to a set of locations whose base address is contained in the ninth stack word.



Note
This diagram shows the value being written to the l.s. byte. If the address were an even number then the m.s. byte would be written to and the l.s. byte would remain unaffected.

Fig. 7.5. The COPY BYTE Instruction.

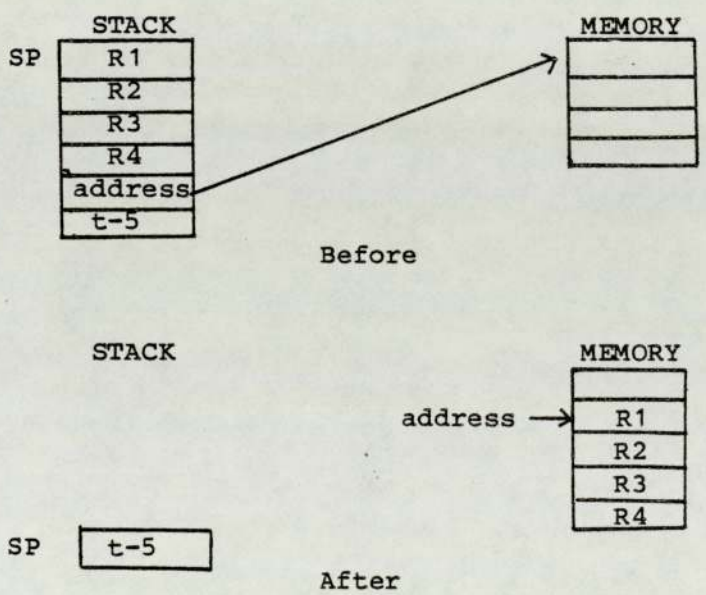


Fig. 7.6. The COPY REAL Instruction.

The stack pointer is incremented by eighteen, thus deleting the set and the address from the stack. See Fig. 7.7.

CPT COPY TAG

(1 Operand: LENGTHWORDS)

Operation:

ST (ST (SP + 2)) <- ST (SP)

CLEAR ((ST (SP + 2) + 1) : (ST (SP + 2) + ST (PC)))

SP <- SP + 4

Description:

The value at the top of the stack (a tag field of a Pascal record) is copied to a location whose address is contained in the top but one stack word. The next LENGTHWORDS words following this address are set to zero. The stack pointer is incremented by 4, thus deleting the tag and the address from the stack. See Fig. 7.8.

CPW COPY WORD

(No operands)

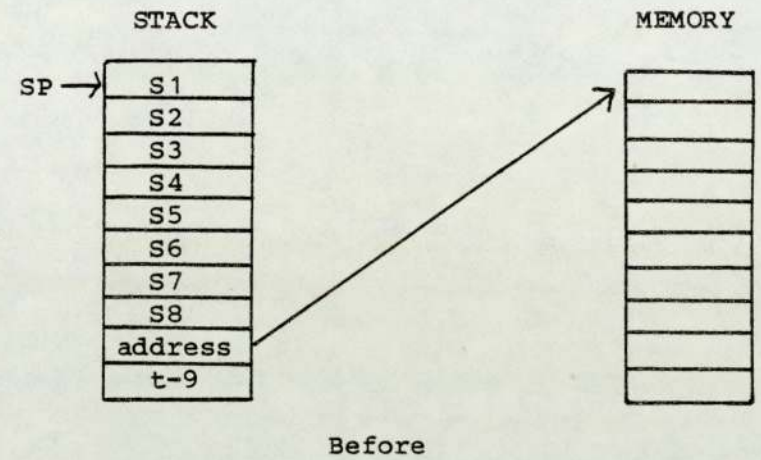
Operation:

ST (ST (SP + 2)) <- ST (SP)

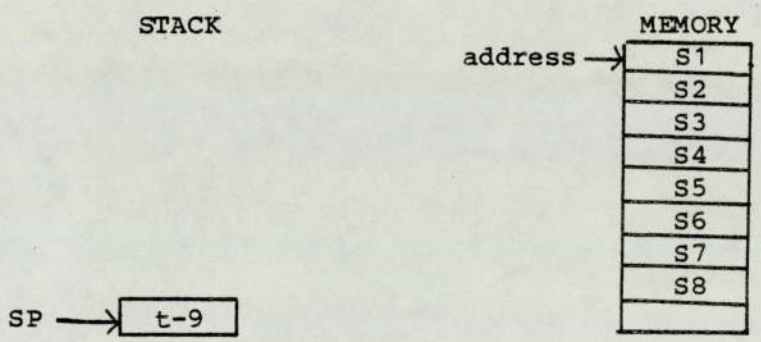
SP <- SP + 4

Description:

The top word of the stack is written to a word in memory whose address is given by the top but one stack word. The stack pointer is incremented by four, thus deleting the value and the address.

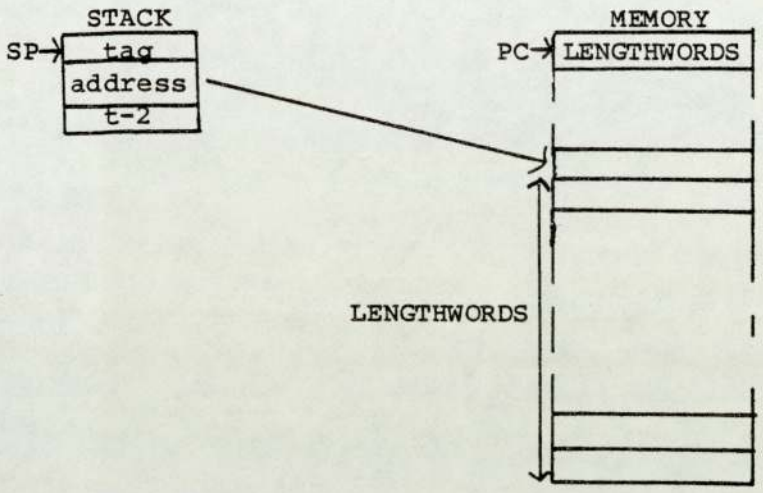


Before

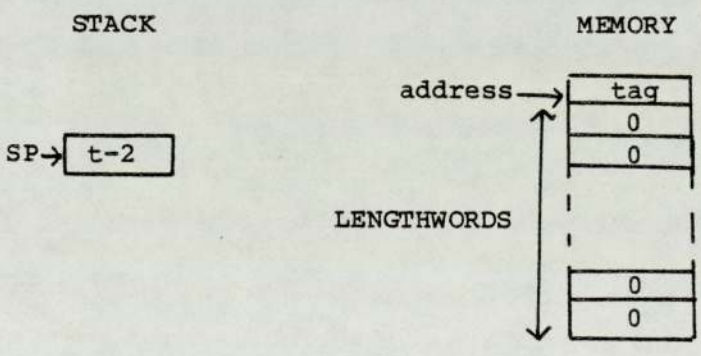


After

Fig. 7.7. The COPY SET Instruction.



Before



After

Fig. 7.8. The COPY TAG Instruction.

CRE COMPARE REAL EQUAL TO

(No operands)

Operation:

SP <- SP + 14

Description:

The top eight words of the stack are compared as two real numbers. Both real numbers are pulled off the stack. If the real number represented by the bottom four words is equal to the real number represented by the top eight words then a one is pushed onto the stack, otherwise a zero is pushed onto the stack. See Fig. 7.10.

CRG COMPARE REAL GREATER THAN

(No operands)

Operation:

SP <- SP + 14

Description:

The top eight stack words are pulled off the stack and compared as two real numbers. If the real number represented by the bottom four words is greater than the real number represented by the top four words then the value one is pushed onto the stack, otherwise zero is pushed onto the stack.

CRGE COMPARE REAL GREATER THAN

OR EQUAL TO

(No operands)

Operation:

SP <- SP + 14

Description:

The top eight stack words are pulled off the stack and compared as two real numbers. If the real number represented by the bottom four words is greater than or equal to the top four words then the value one is pushed onto the stack, otherwise a zero is pushed onto the stack. See Fig. 7.10.

CRLE COMPARE REAL LESS THAN

OR EQUAL

(No Operands)

Operation:

SP <- SP + 14

Description:

The top eight words on the stack are pulled and compared as two real numbers. If the real number represented by the bottom four words is less than or equal to the real number represented by the top four words then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CRLT COMPARE REAL LESS THAN

(No operands)

Operation:

SP <- SP + 14

Description:

The top eight stack words are pulled and compared as two real numbers. If the real number represented by the bottom four words is less than the real number represented by the top four words then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CRNE COMPARE REAL NOT EQUAL

(No operands)

Operation:

SP <- SP + 14

Description:

The top eight words are pulled off the stack and compared as two real numbers. If the real number represented by the bottom four words is not equal to the real number represented by the top four words then a one is pushed onto the stack, otherwise a zero is pushed onto the stack. See Fig. 7.10.

CSE COMPARE STRUCTURE EQUAL TO

(1 Operand: LENGTHWORDS)

Operation:

```
IF ST ((ST (SP)) : (ST (SP) + ST (PC)*2))
 = ST ((ST (SP + 2)) : (ST (SP + 2) + ST (PC)*2))
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

Two structures, each of length LENGTHWORDS, are compared. The address of the source structure is given by the top stack word. The address of the destination structure is given by the top but one stack word. Both of these addresses are pulled off the stack. If every word of the destination structure is equal to every word of the source structure then the value one is pushed onto the stack, otherwise the value zero is pushed onto the stack.

CSG COMPARE STRUCTURE GREATER THAN

(1 Operand: LENGTHWORDS)

Operation:

```
IF ST ((ST (SP + 2)) : (ST (SP + 2) + ST (PC) * 2)
 > ST ((ST (SP)) : (ST (SP) + ST (PC) * 2)
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

Two structures, each of length LENGTHWORDS, are compared. The address of the source structure is contained in the top stack word. The address of the destination structure is given by the top but one

stack word. Both these addresses are pulled off the stack. If every word of the destination structure is greater than every word of the source structure then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CSGE COMPARE STRUCTURE GREATER THAN OR EQUAL TO (1 Operand: LENGTHWORDS)

Operation:

```
IF ST ((ST (SP + 2)) : (ST (SP + 2) + ST (PC) * 2))
>= ST ((ST (SP) : (ST (SP) + ST (PC) * 2))
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

Two structures, each of length LENGTHWORDS, are compared. The address of the source structure is contained in the top stack word. The address of the destination structure is given by the top but one stack word. If every word of the destination structure is greater than or equal to every word of the source structure then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CSLE COMPARE STRUCTURE LESS

THAN OR EQUAL TO

(1 Operand: LENGTHWORDS)

Operation:

```
IF ST ((ST (SP + 2)) : (ST (SP + 2) + ST (PC) * 2))
<= ST ((ST (SP)) : (ST (SP) + ST (PC) * 2))
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

Two structures, each of length LENGTHWORDS, are compared. The address of the source structure is given by the top stack word. The address of the destination structure is given by the top but one stack word. Both of these addresses are pulled off the stack. If every word of the destination structure is less than or equal to every word of the source structure then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CSLT COMPARE STRUCTURE LESS THAN

(1 Operand: LENGTHWORDS)

Operation:

```
IF ST ((ST (SP + 2)) : (ST (SP + 2) + ST (PC) * 2))
 < ST (ST (SP) : ((ST (SP) + ST (PC) * 2))
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

Two structures, each of length LENGTHWORDS, are compared. The address of the source structure is given by the top stack word. The

address of the destination structure is contained in the top but one stack word. Both these addresses are pulled off the stack. If every word of the destination structure is less than every word of the source structure then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

CSNE COMPARE STRUCTURE NOT EQUAL

(1 Operand: LENGTHWORDS)

Operation:

```
IF ST ((ST (SP + 2) : (ST (SP + 2) + ST (PC) * 2))
NOT = ST ((ST (SP)) : (ST (SP) + ST (PC) * 2))
THEN ST (SP + 2) <- 1
ELSE ST (SP + 2) <- 0
SP <- SP + 2
```

Description:

Two structures, each of length LENGTHWORDS, are compared. The address of the source structure is contained in the top stack word. The address of destination structure is given by the top but one stack word. Both these addresses are pulled off the stack. If the two structures are not equal then the value one is pushed onto the stack, otherwise the value zero is pushed onto the stack.

CSP CASE JUMP

(3 Operands: MIN, MAX-MIN,

DISTANCES)

Operation:

PC <- ST (PC + 4 + (ST (SP) - ST (PC))*2)

SP <- SP + 2

Description:

The top word of the stack contains an index. This index must be in the range MIN <= index <= MAX, otherwise a RANGEERROR exception will be generated. The two words following the current instruction word contain the values MIN and MAX-MIN. The words following this contain a list of displacements. A relative jump using one of these displacements is made depending on the value of the index at the top of the stack. If its value is MIN then the first displacement is used, if it is MIN+1 then the second displacement is used and so on. See Fig. 7.9.

CST COPY STRUCTURE

(1 Operand: LENGTHWORDS)

Operation:

ST (ST (SP + 2)) <- ST (ST (SP))

ST (ST (SP + 2) + 2) <- ST (ST (SP) + 2)

.

.

.

ST (ST (SP + 2) + ST (PC)-1) <- ST (ST (SP) + ST (PC) - 1)

SP <- SP + 4

Description:

Copies a structure from one part of memory to another. The

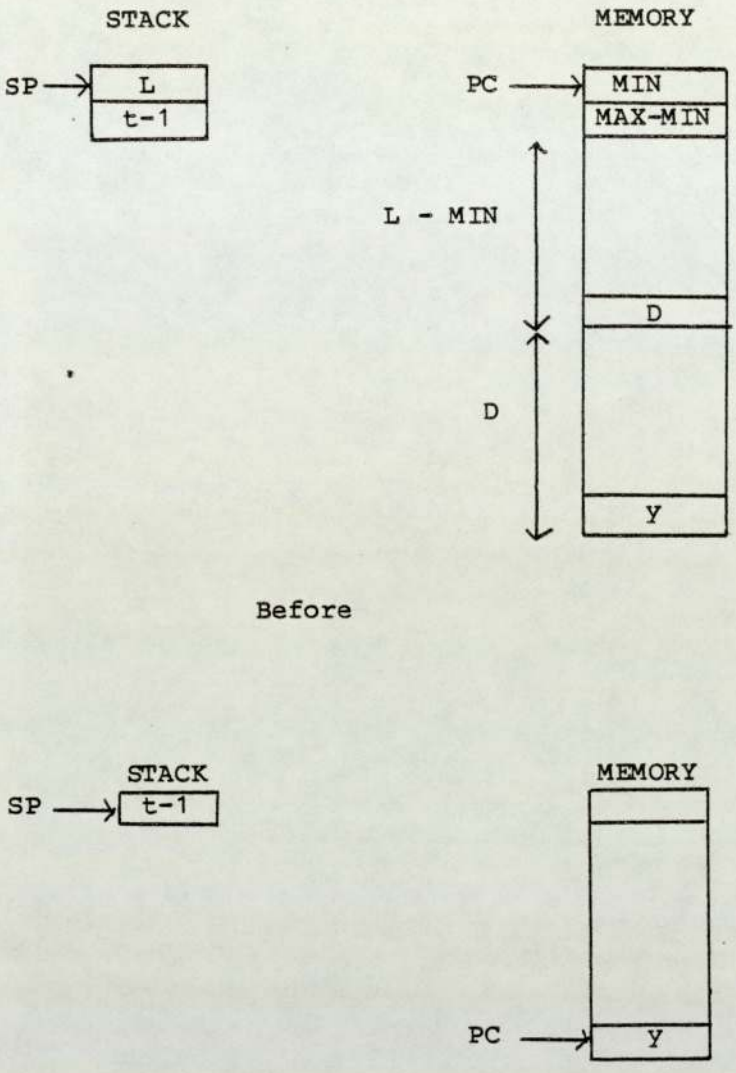


Fig. 7.9. The CASE JUMP Instruction.

source address is contained in the top stack word. The destination address is given by the top but one stack word. Both of these addresses are pulled off the stack. The structure consists of LENGTHWORDS contiguous words. See Fig. 7.11.

CTF CHECK TAG FIELD

(2 Operands: TAGSET, DISPLACEMENT)

Description:

The top stack word contains an address. This address is added to DISPLACEMENT to give the address of a tag field of a variant record (A Pascal data type). This tag must have a valid value which should:-

a) Be in the range 0 to 15.

b) Whatever its value there must be a one in the corresponding bit of TAGSET (numbering from right to left 0 - 15). That is, the tag value must be present in TAGSET.

If the tag is invalid then a VARIANT exception is generated.

CVW CONVERT WORD TO REAL

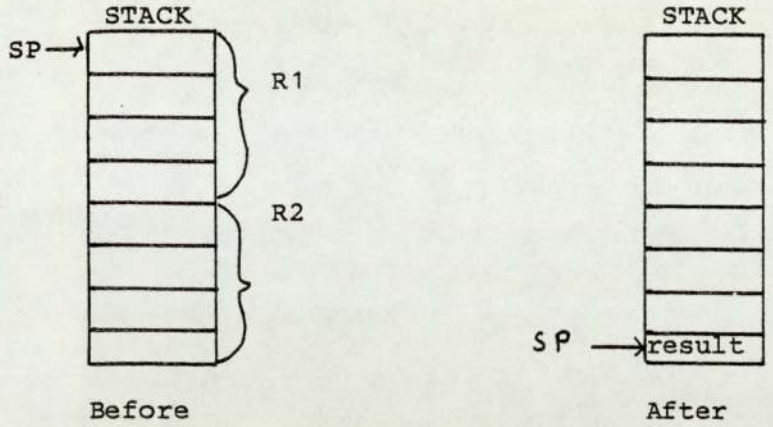
(No operands)

Operation:

SP <- SP - 6

Description:

The top stack word, which is an integer, is pulled off the stack. It is then converted to a real number, consisting of four words, and pushed back onto the stack.



result = 1 if comparison true
= 0 if comparison false

Fig. 7.10. The COMPARE REAL Instructions.

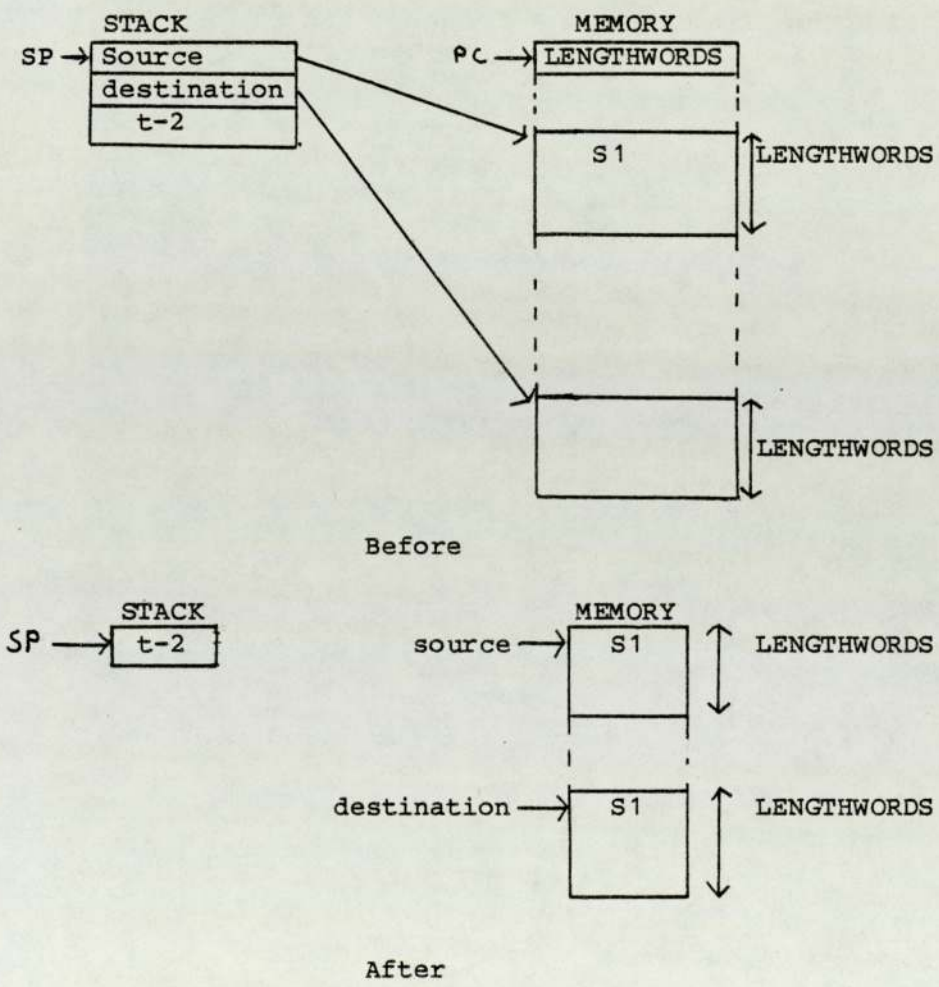


Fig. 7.11. The COPY STRUCTURE Instruction.

DEC DECREMENT WORD

(No operands)

Operation:

ST (ST (SP)) <- ST (ST (SP)) - 1

SP <- SP + 2

Description:

The word whose address is stored in the top stack word is decremented by one. The address is pulled off the stack.

DLY(*CP) DELAY PROCESS

(No operands)

Description:

Pre-empts the current process and puts it in a single process queue. The address of this queue is the top stack word. The process can later be resumed using a "CNT" Continue Process instruction. This corresponds to the high level DELAY primitive. See Appendix 8.

DVR DIVIDE REAL

(No operands)

Operation:

SP <- SP + 8

Description:

The top four words are treated as a real number and divided, as a real number, into the next four stack words where the result is stored. The stack pointer is incremented by eight, thus deleting the top real number. This instruction has not been implemented on the Super Sixteen.

DVW DIVIDE WORD

(No operands)

Operation:

ST (SP + 2) <- ST (SP + 2)/ST (SP)

SP <- SP + 2

Description:

The top but one stack word is divided by the top stack word. The two values are pulled off the stack and the result of the division is pushed onto the stack. If the divisor is zero then an OVERFLOW exception is generated.

ENC(*CP) END CLASS

(No operands)

Description:

This instruction makes an exit from the initial statement of a class. It has the same function as the "EXT" instruction.

ENM(*CP) END MONITOR

(No operands)

Description:

See Appendix 8.

ENP(*CP) END PROCESS

(No operands)

Description:

Stops a process from running. The process is not put in any queue (see Appendix 8) and so it is effectively lost and will never run again.

ENT ENTER PROCEDURE

(4 Operands: STACKLENGTH, POPLENGTH, LINE, VARLENGTH)

Operation:

```
ST (SP - 2) <- G
ST (SP - 4) <- B
ST (SP - 6) <- SP - 6 + ST (PC + 4)
ST (SP - 8) <- ST (PC + 6)
B <- SP - 8
SP <- SP - 8 - ST (PC + 8)
```

Description:

This instruction allocates space on the stack for a procedure. It assumes that the parameters and the return address have already been pushed onto the stack. It pushes the values of the Global and Local Base registers, G and B, onto the stack. It pushes the value of POPLENGTH added to the stack pointer onto the stack. POPLENGTH is a displacement from the current stack position, made up of the parameter length added to 8 (since three pushes have already been made and the return address should already be on the stack). It is effectively a return address for the stack pointer. It then pushes the value of LINE onto the stack, this is used for error diagnostics in the event of a failure. The B register is loaded with the

current stack pointer value (i.e. the address of the word containing LINE). Finally, the stack pointer is decremented by VARLENGTH to leave space for the procedure's temporary variables. If there is not enough space on the stack (this is determined by examining STACKLENGTH) then a STACKLIMIT exception is generated. See Fig. 7.12.

EPP — ENTER PASCAL PROGRAM

(4 Operands: POPLENGTH, LINE, STACKLENGTH, VARLENGTH)

Operation:

Set Mode to User

ST (SP - 2) <- G

ST (SP - 4) <- B

ST (SP - 6) <- SP - 6 + ST (PC)

ST (SP - 8) <- ST (PC + 2)

B <- SP - 8

G <- SP - 8

SP <- SP - 8 - ST (PC)

Description:

This instruction enters a Sequential Pascal program. First, it sets the mode to User (Sequential Pascal). Next the values of the G and B registers are pushed onto the stack. Then, a value formed by adding the current stack pointer to POPLENGTH (= Parameter length + 8) is pushed onto the stack. This is effectively the stack return address when the program terminates. The value of LINE is then pushed onto the stack (this is used if an error occurs). The value of the stack pointer register is put into the Local Base register B and also the Global Base register G. Finally, VARLENGTH bytes are

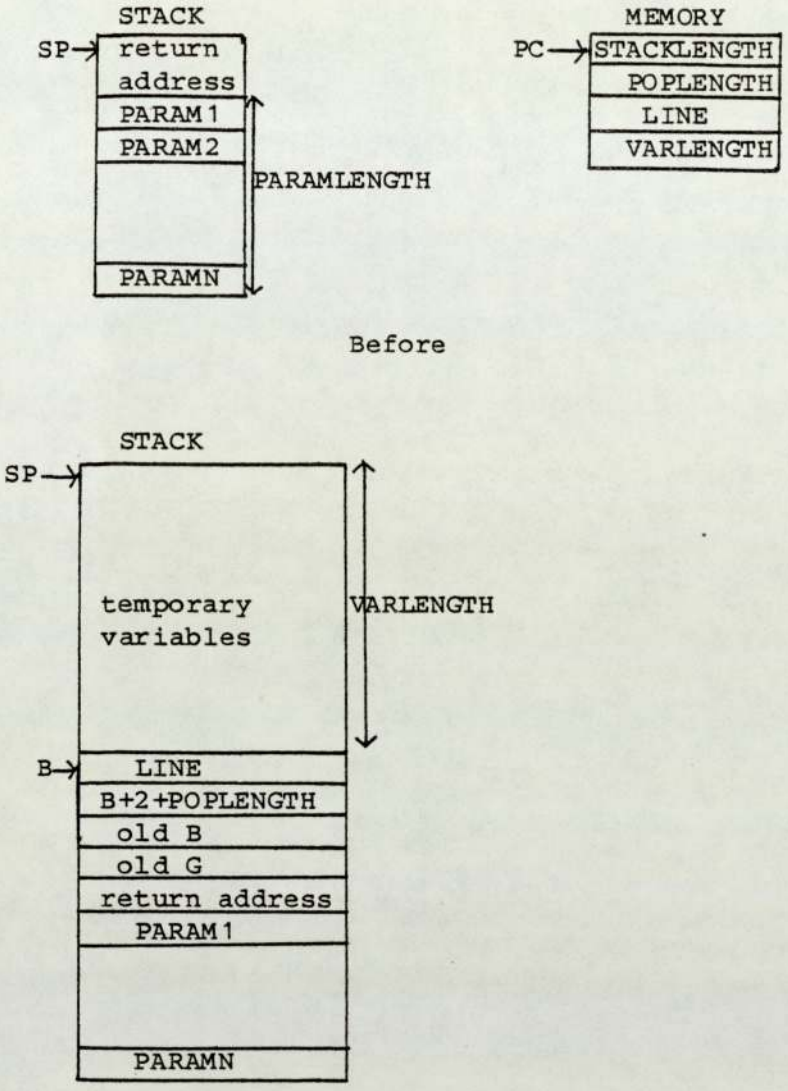


Fig. 7.12. The ENTER PROCEDURE Instruction.

allocated on the top of the stack for the temporary variables. If there is not enough space on the stack then a STACKLIMIT exception occurs. This is determined by examining STACKLENGTH. See Fig. 7.13.

EQS COMPARE SET EQUAL TO

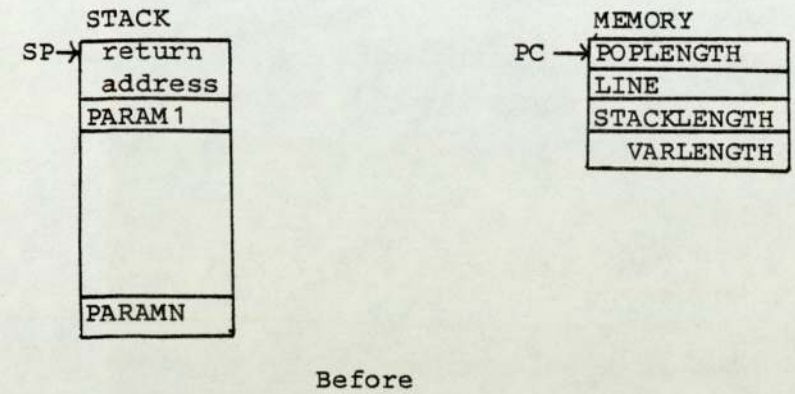
(No operands)

Operation:

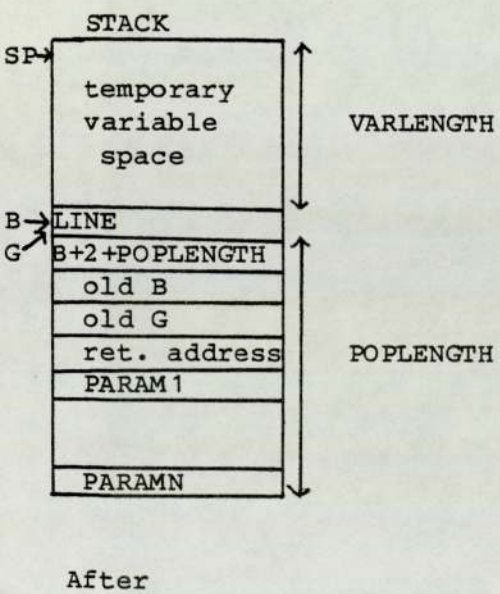
```
IF (ST (SP) : ST (SP + 14)) = (ST (SP + 16) : ST (SP + 30))
THEN ST (SP + 30) <- 1
ELSE ST (SP + 30) <- 0
SP -< SP + 30
```

Description:

The top sixteen stack words are pulled and compared as two sets. If the set represented by the bottom eight words is equal to the set represented by the top eight words then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.



Before



After

Fig. 7.13. The ENTER PASCAL PROGRAM Instruction.

ETC(*CP) ENTER CLASS

(4 Operands: STACKLENGTH, POPLENGTH,

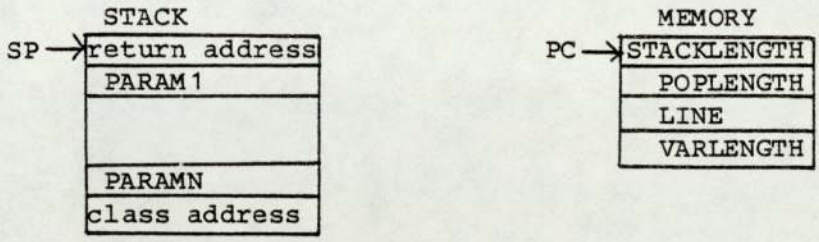
LINE, VARLENGTH)

Operation:

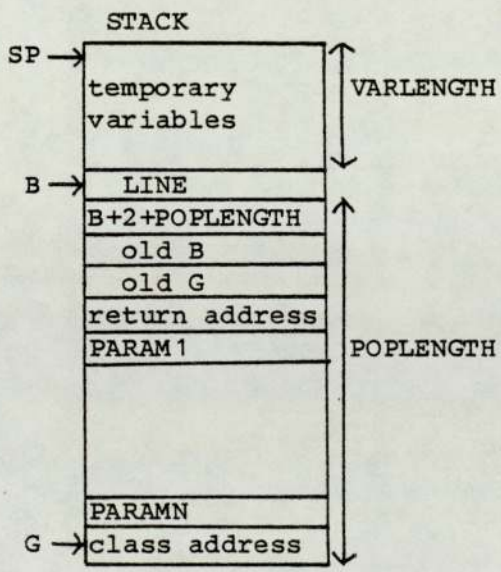
```
ST (SP - 2) <- G

ST (SP - 4) <- B

ST (SP - 6) <- SP - 6 + ST (PC + 4)

ST (SP - 8) <- ST (PC + 6)

B <- SP - 8

G <- ST (ST (SP - 6) - 2)

SP <- SP - ST (PC + 8)
```

Description:

This instruction allocates space on the stack for a class routine. It stacks the G register and the B register. It then stacks a value obtained by adding the current stack pointer to POPLENGTH. This is the stack return address when the class routine is exited. Next, the value of LINE is pushed onto the stack (this is used in the event of an error). Next, the B register is made to point to the stack top and the stack pointer is decremented by VARLENGTH to leave space for the routine's temporary variables. Finally, the G register is made to point to the bottom parameter of the routine plus two. If there is insufficient stack space (STACKLENGTH determines this) then a STACKLIMIT exception is generated. See Fig. 7.14.



Before



After

Fig. 7.14. The ENTER CLASS Instruction.

ETM(*CP) ENTER MONITOR

(4 Operands: STACKLENGTH, POPLENGTH,

LINE, VARLENGTH)

Operation:

ST (SP - 2) <- G

ST (SP - 4) <- B

ST (SP - 6) <- SP + ST (PC + 2) - 6

ST (SP - 8) <- ST (PC + 4)

B <- SP - 8

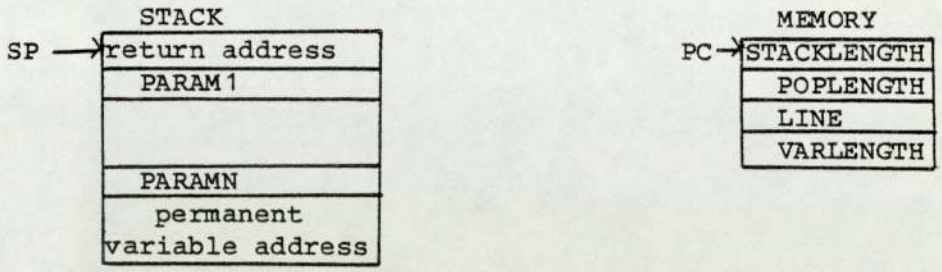
G <- ST (SP - 6) + SP - 6

SP <- SP - 8 - ST (PC + 6)

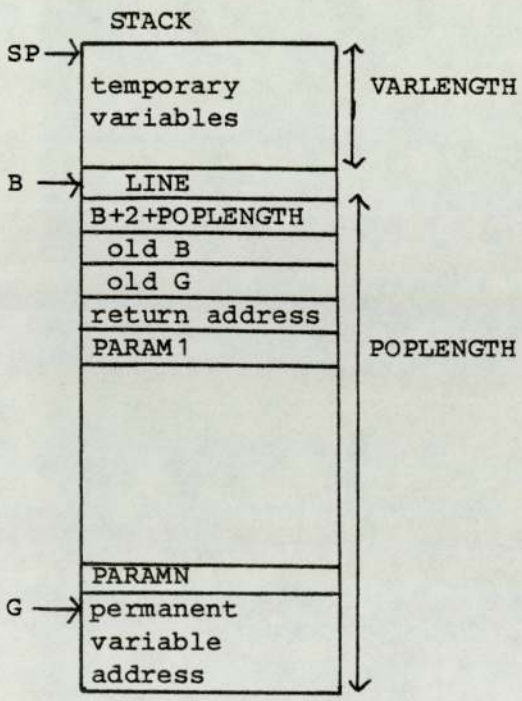
Call Monitor Controller

Description:

This instruction is used to call a monitor routine. The G and B registers are pushed onto the stack, also POPLENGTH is added to the current stack pointer and pushed onto the stack. Next, LINE is pushed onto the stack and space is allocated for the temporary variables on the stack by decrementing the stack pointer by VARLENGTH. A call to the kernel indicating that the monitor resource controller is required is made (see Appendix 8). If there is insufficient space on the stack then a STACKLIMIT exception is generated. See Fig. 7.15.



Before



After

Fig. 7.15. The ENTER MONITOR Instruction.

ETP(*CP) ENTER PREFIX PROCEDURE

(4 Operands: STACKLENGTH, POPLENGTH,

LINE, VARLENGTH)

Operation:

ST (SP - 2) <- G

ST (SP - 4) <- B

ST (SP - 6) <- SP - 6 + ST (PC + 2)

ST (SP - 8) <- ST (PC + 4)

B <- SP - 8

SP <- SP - 8 - ST (PC + 4)

G <- ST (G + 6)

Set Mode to System

Description:

This instruction is similar to an ENT instruction, i.e. it enters a procedure. However, the procedure that it enters is a prefix routine implemented within a Concurrent Pascal process. The G and B registers and POPLENGTH and LINE are all stacked and space is allocated for the temporary variables. The B register is set to the address on the stack containing LINE. Unlike the ENT instruction, however, the G register is set to point to the base address of the permanent variables of the current process. This is stored six bytes below the present value of G. Also, the mode is changed to System. If there is no space on the stack then a STACKLIMIT exception is generated.

EXC EXIT CLASS

(No Operands)

Description:

Makes an exit from a class routine. This instruction is identical to the EXT instruction.

EXM(*CP) EXIT MONITOR

(No Operands)

Description:

Used to exit from a monitor routine, has the same format as the ENM instruction.

EXP(*CP) EXIT PROGRAM

(No Operands)

Description:

Makes an exit from a program. This is done by causing an exception.

EXPC(*CP) EXIT PREFIX PROCEDURE

(No Operands)

Operation:

SP <- B

B <- ST (SP + 4)

G <- ST (SP + 6)

PC <- ST (SP + 8)

SP <- ST (SP + 2)

Set Mode to User

Description:

Makes an exit from a prefix routine implemented within a Concurrent Pascal process. This procedure is the same as the EXT procedure except that the mode is set to User.

EXT EXIT PROCEDURE

(No Operands)

Operation:

B <- ST (B + 4)

G <- ST (B + 6)

PC <- ST (B + 8)

SP <- ST (B + 2)

Description:

Makes an exit from a procedure. The stack is reset as follows. The word after the word pointed to by the B register holds a displacement. When this is added to its address it gives the return stack address. Therefore, this value is put into the stack pointer register, thus deleting all the temporary values, permanent variables and parameters of the routine. Two words after the B register address, the value of the B register before the procedure

was called is held. The word after this holds the corresponding value for the G register. The word after this holds the program return address. These values are loaded up into the B, G and Program Counter registers respectively. See Fig. 7.16.

FLD FIELD

(1 Operand: DISPLACEMENT)

Operation:

ST (SP) <- ST (SP) + ST (PC)

Description:

The top word of the stack is incremented by DISPLACEMENT. This instruction is normally used when the top stack word contains an address.

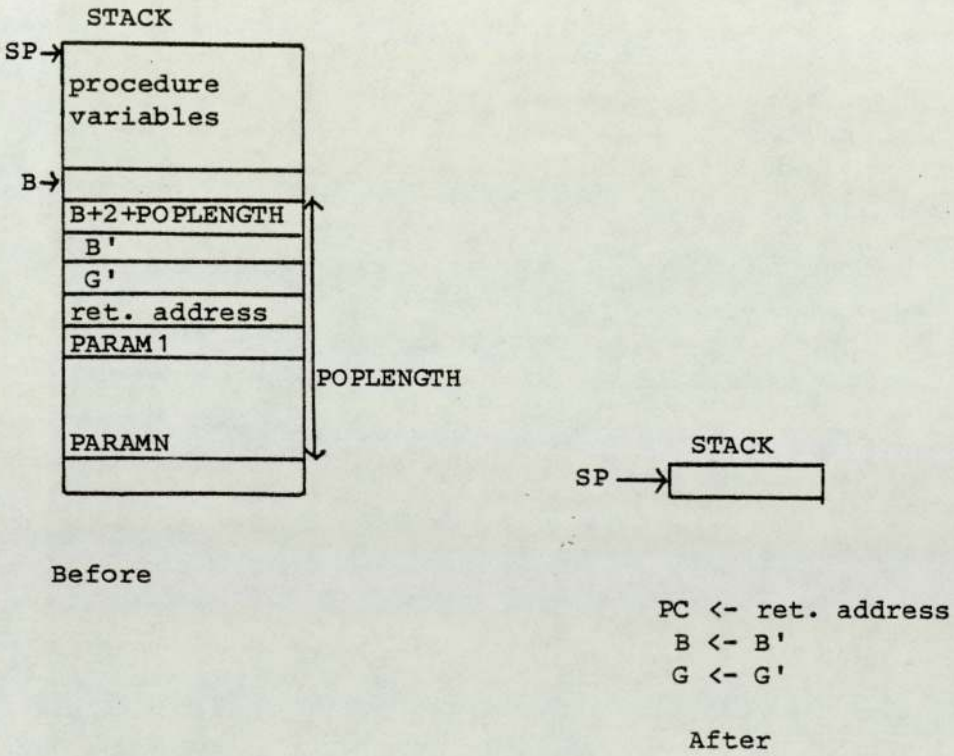


Fig. 7.16. The EXIT PROCEDURE Instruction.

FNV FUNCTION VALUE

(1 Operand: KIND)

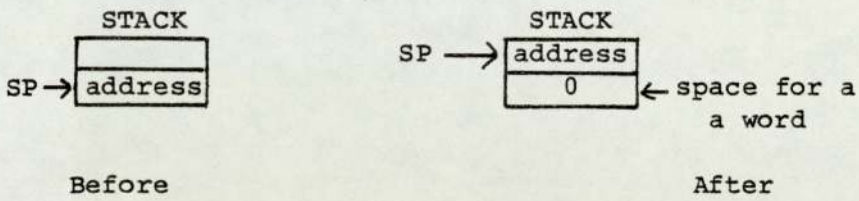
Operation:

```
IF ST (PC + 2) = 0
THEN ST (SP - 2) <- 0
     SP <- SP - 2
ELSE IF ST (PC + 2) = 8
     THEN SP <- SP - 8
     ELSE IF ST (PC + 2) = 16
          THEN ST (SP - 2) <- ST (SP)
               ST (SP) <- 0
               SP <- SP - 2
          ELSE IF ST (PC + 2) = 24
               THEN ST (SP - 8) <- ST (SP)
                    SP <- SP - 8
```

Description:

This instruction allocates space on the top of the stack for a function call. The result of this function can be one of four different types according to KIND. If KIND is 0 then space for an integer must be allocated, i.e. the top word of the stack is decremented by 2. The top stack word is also set to zero. If KIND is 8 then space for a real number is required. This is done by decrementing the stack pointer by 8. If KIND is 16 or 24 then the result is again an integer or a real respectively. However, the function appears within a Class or Monitor. In this case the top stack word contains an address (the base address of the permanent variables for the Class or Monitor). This value is removed from the stack and then put back on again on top of the result location. See Fig. 7.17.

KIND = 16 (CLASS WORD)



KIND = 24 (CLASS REAL)

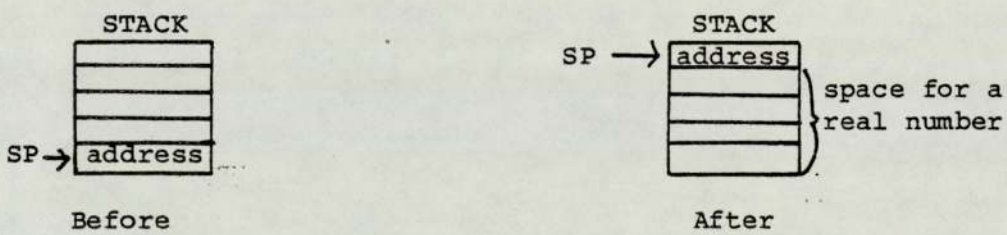


Fig. 7.17. The FUNCTION VALUE Instruction.

FSP FALSE JUMP

(1 Operand: DISTANCE)

Operation:

IF ST (SP) = 0

THEN PC <- PC + ST (PC)

ELSE PC <- PC + 2

Description:

The top word of the stack is examined. If it is zero then a relative jump of DISTANCE bytes is made. If the program executing this instruction is a Sequential Pascal program and it has been flagged (by a STOP instruction) as ready to terminate then the jump is not made and the program is terminated.

ICL(*CP) INITIALISE CLASS

(2 Operands: PARAMLENGTH, DISTANCE)

Description:

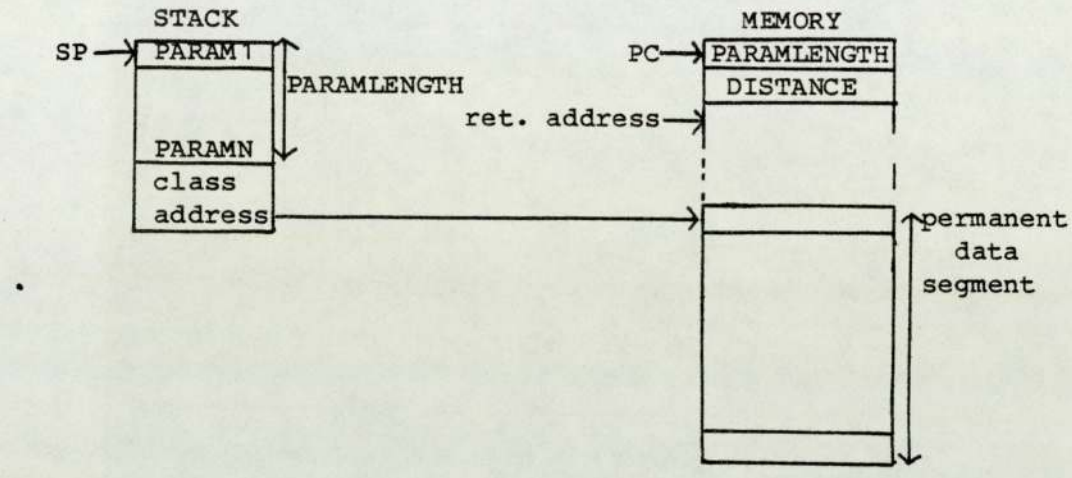
The stack top contains the class parameters, followed by the base address of the class. The parameters are transferred from the stack to the permanent variable space of the stack which starts at the word following the base address of the class. Following this, DISTANCE is added to the program counter so that the initial statement of the class will be executed. PARAMLENGTH indicates the number of parameters (in bytes). See Fig. 7.18.

IMN(*CP) INITIALISE MONITOR

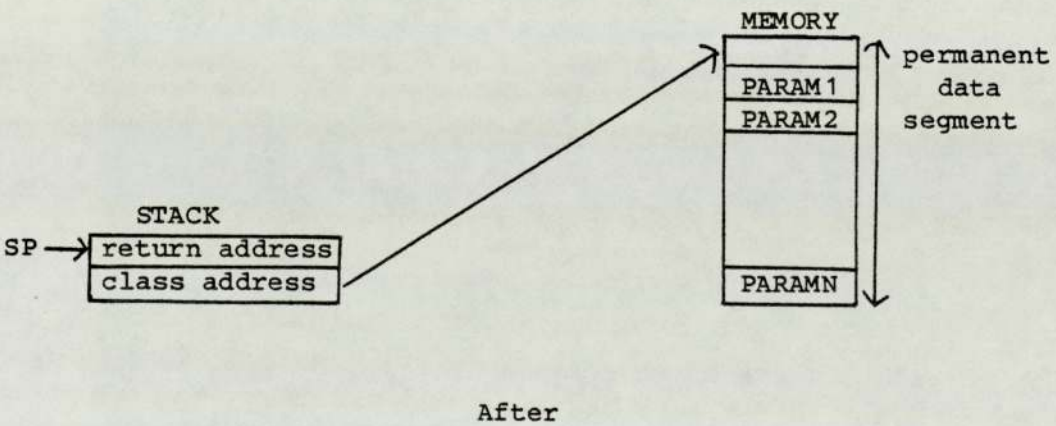
(2 Operands: PARAMLENGTH, DISTANCE)

Description:

Initialises a Monitor. This instruction is the same as the ICL



Before



After

Fig. 7.18. The INITIALISE CLASS Instruction.

instruction.

INC — INCREMENT WORD

(No Operands)

Operation:

ST (ST (SP)) <- ST (ST (SP)) + 1

SP -< SP + 2

Description:

The word whose address is stored in the top stack word is incremented by 1. The address is pulled off the top of the stack.

IOP(*CP) — INPUT/OUTPUT OPERATION

(No Operands)

Description:

See Appendix 8.

IPC(*CP) — INITIALISE PROCESS

(4 Operands: PARAMLENGTH, VARLENGTH, STACKLENGTH, DISTANCE)

Description:

See Appendix 8.

IPV INITIALISE POINTER VARIABLE

(2 Operands: STACKLENGTH+LENGTH,

LENGTH)

Operation:

```
ST (ST (SP)) <- H
H <- H + ST (PC + 2)
SP <- SP + 2
```

Description:

Allocates LENGTH bytes of store on the heap. A pointer is set to the first address of the new store allocated. The address of this pointer is obtained from the top of the stack. If there is not enough space on the heap then a HEAPLIMIT exception is generated. See Fig. 7.19.

IPVC INITIALISE POINTER VARIABLE

AND CLEAR

(2 Operands: LENGTH+STACKLENGTH,

LENGTH)

Operation:

```
ST (ST (SP)) <- H
H <- H + ST (PC + 4)
CLEAR ST ((ST (SP)) : (ST (SP) + ST (PC + 4)))
SP <- SP + 2
```

Description:

This instruction is the same as IPV. However, the new area of the heap is cleared (i.e. every word is set to zero).

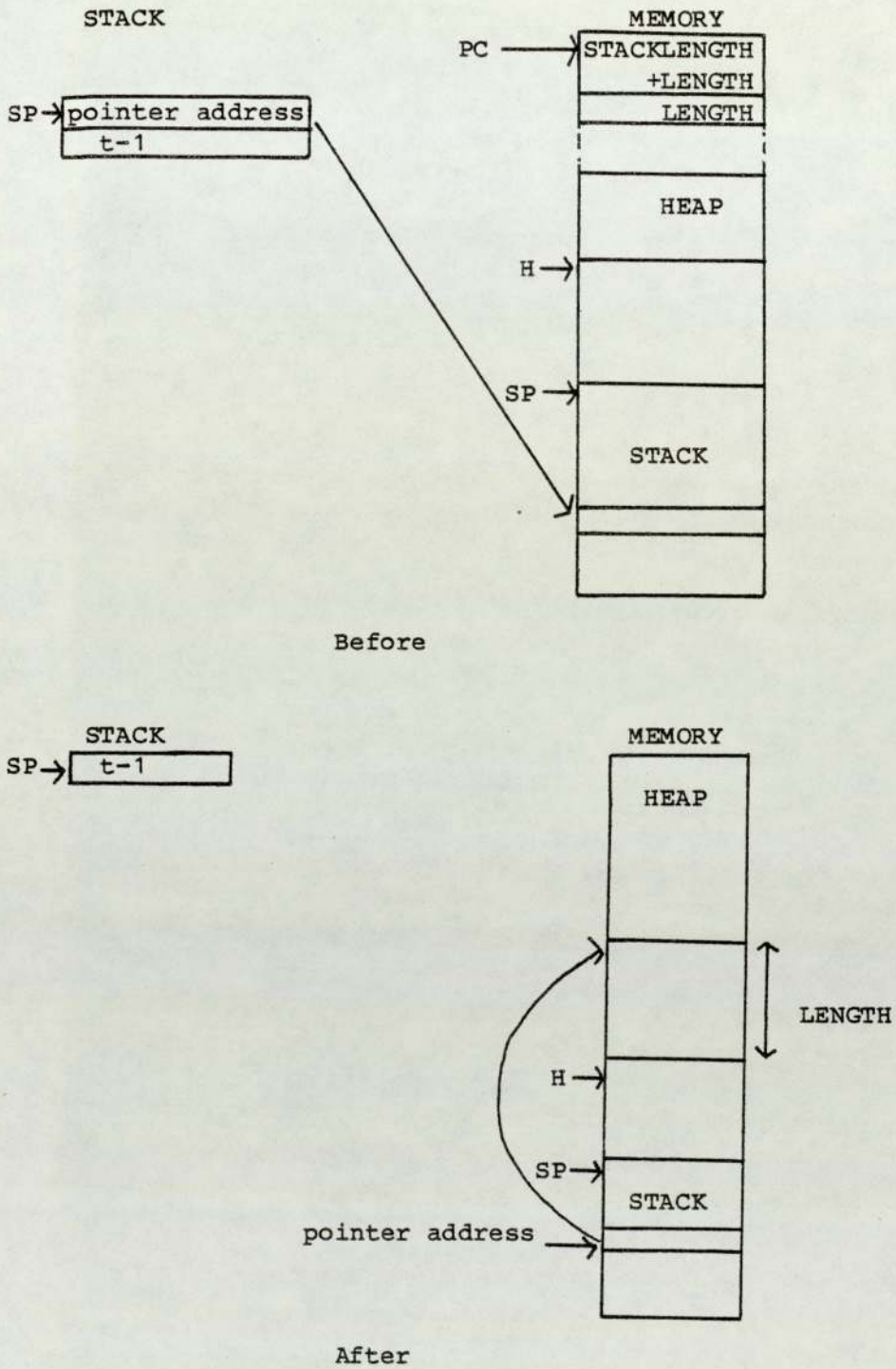


Fig. 7.19. The INITIALISE POINTER VARIABLE Instruction.

IVR INITIALISE VARIABLES

(1 Operand: LENGTHWORDS)

Operation:

Sets the top LENGTHWORDS words of the stack to zero. This instruction is normally used to initialise the temporary variables of a routine.

JMP RELATIVE JUMP

(1 Operand: DISTANCE)

Operation:

PC <- PC + ST (PC)

Description:

A relative jump (i.e. a branch) of DISTANCE bytes is made. In other words, the word following the current instruction word is added to the program counter.

MLR MULTIPLY REAL

(No Operands)

Operation:

SP <- SP + 8

Description:

The top four stack words are treated as a real number and multiplied, as a real number, by the next four words on the stack. Both real numbers are pulled off the stack and the result of the multiplication is pushed onto the stack.

MLW MULTIPLY WORD

(No Operands)

Operation:

ST (SP + 2) <- ST (SP) * ST (SP + 2)

SP <- SP + 2

Description:

The top two words of the stack are pulled and multiplied together. The result is pushed onto the stack.

MOD MODULO WORD

(No Operands)

Operation:

ST (SP + 2) <- Remainder (ST (SP + 2) / ST (SP))

SP <- SP + 2

Description:

The contents of the top but one stack word is divided by the contents of the top stack word. The two values are pulled off the stack and the remainder formed when making the division is pushed onto the stack.

NGR NEGATE REAL

(No Operands)

Description:

The real number which is stored on the top four words of the stack is negated.

NGW NEGATE WORD

(No Operands)

Operation:

ST (SP) <- - ST (SP)

Description:

The top word of the stack is negated.

NLN NEWLINE

(1 Operand: NUMBER)

Operation:

ST (B) <- ST (PC)

Description:

The value of NUMBER is put into the address pointed to by the B register.

NOT NOT WORD

(No Operands)

Operation:

ST (SP) <- NOT ST (SP)

Description:

Inverts the word at the top of the stack.

ORS OR SET

(No Operands)

Operation:

```
ST (SP + 16) <- ST (SP) OR ST (SP + 16)
ST (SP + 18) <- ST (SP + 2) OR ST (SP + 18)
ST (SP + 20) <- ST (SP + 4) OR ST (SP + 20)
ST (SP + 22) <- ST (SP + 6) OR ST (SP + 22)
ST (SP + 24) <- ST (SP + 8) OR ST (SP + 24)
ST (SP + 26) <- ST (SP + 10) OR ST (SP + 26)
ST (SP + 28) <- ST (SP + 12) OR ST (SP + 28)
ST (SP + 30) <- ST (SP + 14) OR ST (SP + 30)
SP <- SP + 16
```

Description:

The top eight stack words are ORed with the next eight stack words. Both sets are pulled off the stack and the result (8 words) is pushed onto the stack. See Fig. 7.20.

ORW OR WORD

(No Operands)

Operation:

```
ST (SP + 2) <- ST (SP + 2) OR ST (SP)
SP <- SP + 2
```

Description:

The top word of the stack is ORed with the top but one stack word. Both the words are pulled off the stack and the result is pushed onto the stack.

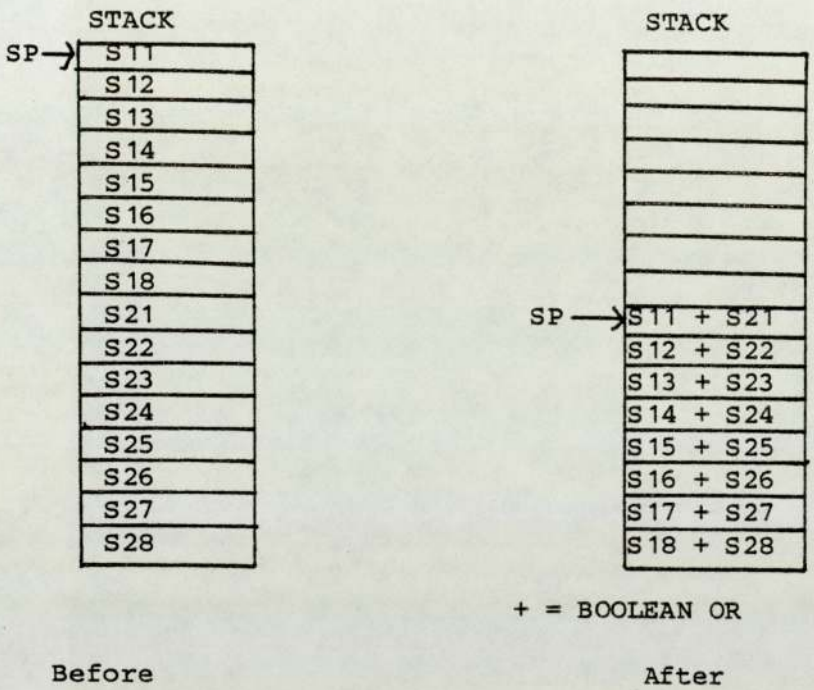


Fig. 7.20. The OR SET Instruction.

PAC PUSH ARRAY COMPONENT

(3 Operands: MIN, MAX-MIN,

LENGTH)

Operation:

ST (SP + 2) <- ST ((SP + 2) + (ST (SP) - ST (PC + 2) * ST (PC + 4)

SP <- SP + 2

Description:

This instruction forms an array element address. The index specifying the element required is on the top of the stack. The base address of the array is stored in the top but one stack word. Both these values are pulled off the stack. If the index is not between MIN and MAX then a RANGEERROR exception is generated. The element address is calculated by multiplying the index displacement from MIN by LENGTH and adding it to the array base address. The address formed by doing this is pushed onto the stack. See Fig. 7.21.

PCA PUSH CONSTANT ADDRESS

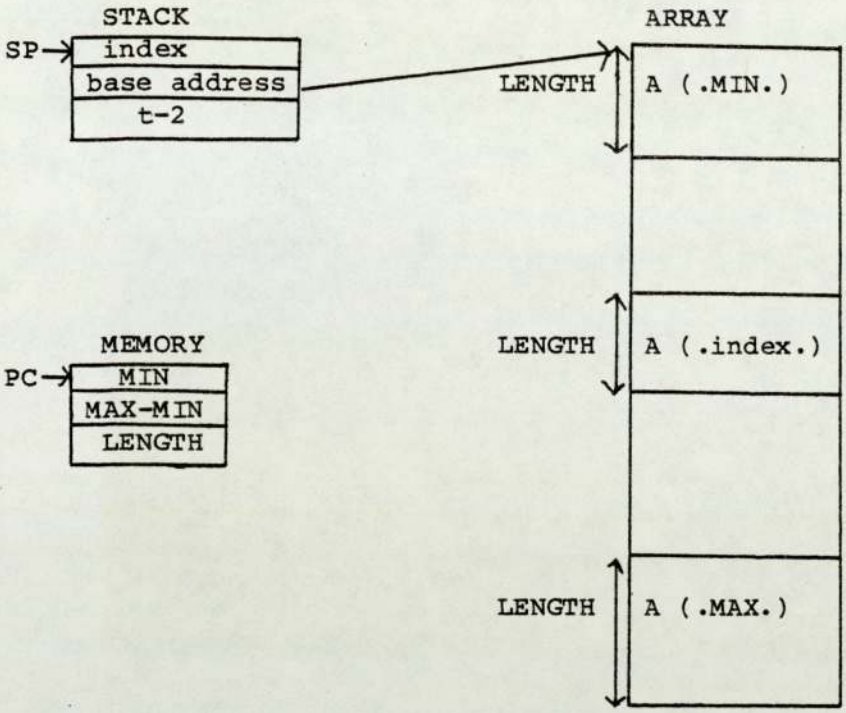
(1 Operand: DISPLACEMENT)

Operation:

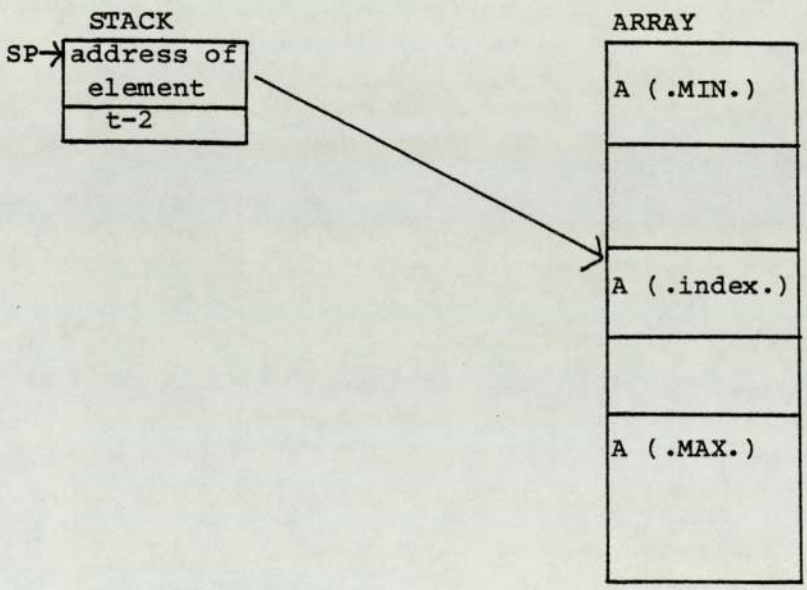
```
IF System Mode
THEN SP <- SP - 2
     ST (SP) <- ST (CONSTATR) + ST (PC)
ELSE SP <- SP - 2
     ST (SP) <- ST (G + 10) + ST (PC)
```

Description:

The address of a large constant (set, real or array of characters) is pushed onto the stack. For a Concurrent Pascal program (i.e. in System Mode) the base address of the large



Before



After

Fig. 7.21. The PUSH ARRAY COMPONENT Instruction.

constants is stored in the CONSTATR attribute (see Appendix 8). For a Sequential Pascal program (User Mode) the base address is stored ten bytes below the address pointed to by the G register. In either case the DISPLACEMENT is added onto this base address and the result is pushed onto the stack.

PGA PUSH GLOBAL ADDRESS

(1 Operand: DISPLACEMENT)

Operation:

ST (SP - 2) <- G + ST (PC)

SP <- SP - 2

Description:

The address of a global variable is pushed onto the stack. This value is obtained by adding the value in the Global Base Register G to DISPLACEMENT.

PLA PUSH LOCAL ADDRESS

(1 Operand: DISPLACEMENT)

Operation:

ST (SP - 2) <- B + ST (PC)

SP <- SP - 2

Description:

The address of a local variable is pushed onto the stack. This address is obtained by adding the value stored in the local base address register B to DISPLACEMENT.

PLB(*CP) PUSH LABEL

(1 Operand: DISTANCE)

Operation:

ST (SP - 2) <- ST (PC) + PC

SP <- SP - 2

Description:

An address which is calculated by adding DISTANCE to the current program counter value is pushed onto the stack.

POP POP STACK

(1 Operand: LENGTH)

Operation:

SP <- SP + ST (PC)

Description:

The stack pointer is incremented by LENGTH.

PRE PREDECESSOR WORD

(No operands)

Operation:

ST (SP) <- ST (SP) - 1

Description:

Decrements the value stored in the top stack word by 1.

PSB PUSH BYTE

(No operands)

ST (SP) <- <BYTE> ST (ST (SP))

Description:

It is assumed that the top word of the stack contains an address. This is replaced with the byte value found at this address. The word on the stack will be right justified with leading zeroes.

PSC PUSH CONSTANT

(1 operand: VALUE)

Operation:

ST (SP - 2) <- ST (PC)

SP <- SP - 2

Description:

The word following the current instruction is pushed onto the stack.

PSG PUSH GLOBAL

(1 operand: DISPLACEMENT)

Operation:

ST (SP - 2) <- ST (G + ST (PC))

SP <- SP - 2

Description:

A Global variable consisting of one word is pushed onto the stack. The address of this variable is obtained by adding the value stored in the Global Base address register G to DISPLACEMENT.

PSL PUSH LOCAL

(1 Operand: DISPLACEMENT)

Operation:

ST (SP - 2) <- ST (B + ST (PC))

SP <- SP - 2

Description:

A local variable is pushed onto the stack. The address of this variable is formed by adding the Local Base register B to DISPLACEMENT.

PSR PUSH REAL

(No Operands)

Operation:

ST (SP - 6) <- ST (ST (SP))

ST (SP - 4) <- ST (ST (SP) + 2)

ST (SP - 2) <- ST (ST (SP) + 4)

ST (SP) <- ST (ST (SP) + 6)

SP <- SP - 6

Description:

An address is pulled off the stack. This address points to a real number consisting of four words. This real number is pushed onto the stack.

PSS PUSH SET

(No operands)

Operation:

ST (SP - 14) <- ST (ST (SP))

ST (SP - 12) <- ST (ST (SP) + 2)

ST (SP - 10) <- ST (ST (SP) + 4)

ST (SP - 8) <- ST (ST (SP) + 6)

ST (SP - 6) <- ST (ST (SP) + 8)

ST (SP - 4) <- ST (ST (SP) + 10)

ST (SP - 2) <- ST (ST (SP) + 12)

ST (SP) <- ST (ST (SP) + 14)

SP <- SP - 14

Description:

An address is pulled off the top of the stack. This value points to a set consisting of eight words. This set is pushed onto the stack.

PSW PUSH WORD

(No operands)

Operation:

ST (SP) <- ST (ST (SP))

Description:

The top word of the stack is treated as an address. It is replaced by the contents of this address.

PTR TEST POINTER

(No operands)

Description:

The top word of the stack, which should be a pointer value, is checked. If it is zero then a POINTER exception is generated since it is invalid. Otherwise, no action is taken.

RNG CHECK RANGE

(2 Operands: MIN, MAX)

Description:

If the top stack word contains a value which falls within the values MIN and MAX then no action is taken. If the top stack word falls outside this range then a RANGEERROR exception is generated.

RTM(*CP) REAL TIME

(No operands)

Operation:

ST (SP - 2) <- time in seconds since system activation

SP <- SP - 2

Description:

The time in seconds since system activation is pushed onto the stack. See Appendix 8.

SBR SUBTRACT REAL

(No operands)

Operation:

SP <- SP + 8

Description:

The top four words are treated as a real number and pulled off the stack. This is then subtracted, as a real number, from the next four words which are also pulled off the stack. The result is pushed back onto the stack.

SBS SUBTRACT SET

(No Operands)

Operation:

ST (SP + 16) <- ST (SP + 16) AND NOT ST (SP)

ST (SP + 18) <- ST (SP + 18) AND NOT ST (SP + 2)

ST (SP + 20) <- ST (SP + 20) AND NOT ST (SP + 4)

ST (SP + 22) <- ST (SP + 22) AND NOT ST (SP + 6)

ST (SP + 24) <- ST (SP + 24) AND NOT ST (SP + 8)

ST (SP + 26) <- ST (SP + 26) AND NOT ST (SP + 10)

ST (SP + 28) <- ST (SP + 28) AND NOT ST (SP + 12)

ST (SP + 30) <- ST (SP + 30) AND NOT ST (SP + 14)

SP <- SP + 16

Description:

For every word in the bottom eight words of the stack a comparison is made with the corresponding word in the top eight stack words. Any bits which are set to zero in the former are unaffected. However, any bits that are set to one are reset to zero if the corresponding bit in the top eight words is also a one. The top set is pulled off the stack. See Fig. 7.22.

SCL(*CP) SYSTEM CALL

(1 Operand: ENTRY)

Operation:

ST (SP - 2) <- PC

SP <- SP - 2

PC <- ST (PC) + ST (G + 2)

Description:

A call of a system (Concurrent Pascal) routine is made from a Sequential Pascal program. The current program counter value (the return address) is pushed onto the stack. The system routines (prefix procedures) are located by an array of routine addresses. The base address of this array is stored in the word below the address pointed to by the G register. The displacement within this array is given by ENTRY which is added on to the base address to give the procedure address. This value is loaded into the program counter.

SGE SET GREATER THAN OR EQUAL TO

(No operands)

Operation:

IF NOT (ST (SP + 16 : SP + 30) AND NOT ST (SP : SP + 14) = 0)

THEN ST (SP + 30) <- 1

ELSE ST (SP + 30) <- 0

SP <- SP + 30

Description:

The top sixteen words are compared as two sets. If the set represented by the bottom eight words is greater than or equal to

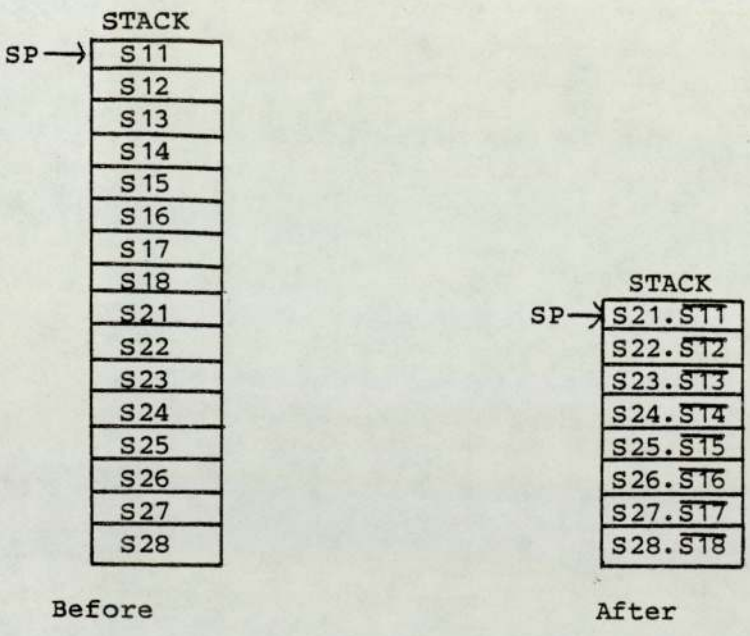


Fig. 7.22. The SUBTRACT SET Instruction.

the set represented by the bottom eight words (i.e. for every 0 in the bottom set there is a zero in the top set) then the result is true, otherwise it is false. Both sets are pulled off the stack. If the result is true then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

SHP(*CP) SET HEAP REGISTER

(No Operands)

Operation:

H <- ST (SP)

SP <- SP + 2

Description:

The top stack word is pulled and put into the Heap register H.

SLE COMPARE SET LESS THAN OR EQUAL TO

Operation:

IF ((ST (SP)) : (ST (SP + 14)) AND NOT (ST (SP + 16) : ST (SP + 30)) = 0

THEN ST (SP + 30) <- 1

ELSE ST (SP + 30) <- 0

SP <- SP + 30

Description:

The top sixteen stack words are compared as two sets. If the set represented by the bottom eight words is less than or equal to the set represented by the top eight words (i.e. for every zero in the top set there is a zero in the bottom set) then the result is true, otherwise it is false. Both sets are pulled off the stack.

If the reult is true then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

SNE COMPARE SET NOT EQUAL

(No operands)

Operation:

```
IF ST ((SP) : (SP + 14)) NOT = ST ((SP + 16) : (SP + 30))
THEN ST (SP + 30) <- 1
ELSE ST (SP + 30) <- 0
SP <- SP + 30
```

Description:

The top sixteen stack words are compare as two sets. Both these sets are pulled off the stack. If the set represented by the top eight words is not equal to the set represented by the bottom eight words then a one is pushed onto the stack, otherwise a zero is pushed onto the stack.

STP(*CP) STOP PROGRAM

(No operands)

Description:

Terminates a Sequential Pascal program. The stack top contains two values which are pulled. The top but one word is the index number of the process running the program and the top word contains a value to be written into the RESULT attribute. The program is stopped by setting CONTATR to zero. For a further description of process attributes, see Appendix 8.

STRT(*CP) START PROGRAM

(No operands)

Description:

Sets the CONTATR process attribute to one. See Appendix 8.

SUB SUBTRACT WORD

(No operands)

Operation:

ST (SP + 2) <- ST (SP + 2) - ST (SP)

SP <- SP + 2

Description:

The top stack word is subtracted from the top but one stack word. Both words are pulled off the stack and the result of the subtraction is pushed onto the stack.

SUC SUCCESSOR WORD

(No operands)

Operation:

ST (SP) <- ST (SP) + 1

Description:

Increments the top stack word by 1.

TEM(*CP) TEST EMPTY

(No operands)

Operation:

```
IF ST (SP) = 0
THEN ST (SP) <- 1
ELSE ST (SP) <- 0
```

Description:

The top stack word is tested. If it is zero then it is set to one, otherwise it is set to zero.

TIS TEST IN SET

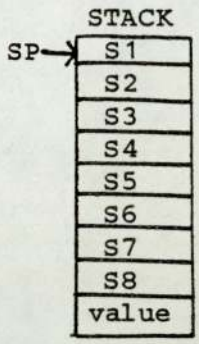
(No operands)

Operation:

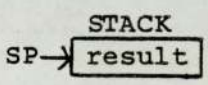
```
ST (SP + 16) <- ((ST (SP + ST (SP + 16)/16))
    SHIFT LEFT  (ST (SP + 16) MOD 16)) AND 1
SP <- SP + 16
```

Description:

The top eight stack words contain a set, the next word in the stack contains a number. Both of these are pulled off the stack. If the number is in the set (i.e. its corresponding bit is a one) then a one is pushed onto the stack, otherwise a zero is pushed onto the stack. If the number is not in the range 0 <= n <= 127 then a RANGEERROR exception is generated. See Fig. 7.23.



Before



After

result = 1 if value is in set
= 0 otherwise

Fig. 7. 23. The TEST IN SET Instruction.

TNR TRUNCATE REAL

(No operands)

Operation:

SP <- SP + 6

Description:

Pulls the real number which is stored in the top four words of the stack and truncates it. The integer result is pushed back onto the stack.

WAIT(*CP) PUT PROCESS IN WAITING QUEUE

(No operands)

Description:

Suspends a process for one second. See Appendix 8.

Appendix 8

Concurrent P-code Documentation

This appendix contains a description of all the microcode that is not concerned with fault-tolerance and cannot be easily described in Appendix 7. This does not necessarily include all the Concurrent P-code instructions as some are no more complex than their Sequential P-code counterparts. This appendix is mainly concerned with process scheduling, monitor usage, real-time operations and input/output.

8.1 System Data Structures

The data structure of the Concurrent Pascal machine memory is shown in Fig. 8.1 and is defined by means of equates at the start of the microcode listing. Starting from the bottom address in memory ($0000) it is structured as follows. First, there are n processes and p priorities, it is assumed at present that these values are 4 and 3 respectively. There are p queues of processes waiting to run, each of these can hold "QLENGTH" processes, at present this is set to n. There is a queue for each priority. Following this there is a further queue which holds processes that have been delayed for one second using the high level Concurrent Pascal "WAIT" primitive. Next comes the system pointers (Fig. 8.2). All of the values in this category are used to allocate data areas whenever a new process or monitor is introduced into the system (using an "Initialise Process" or "Initialise Monitor" P-code Instruction). Once a monitor or process is initialised it remains in the system for ever, thus no garbage collection is required. The

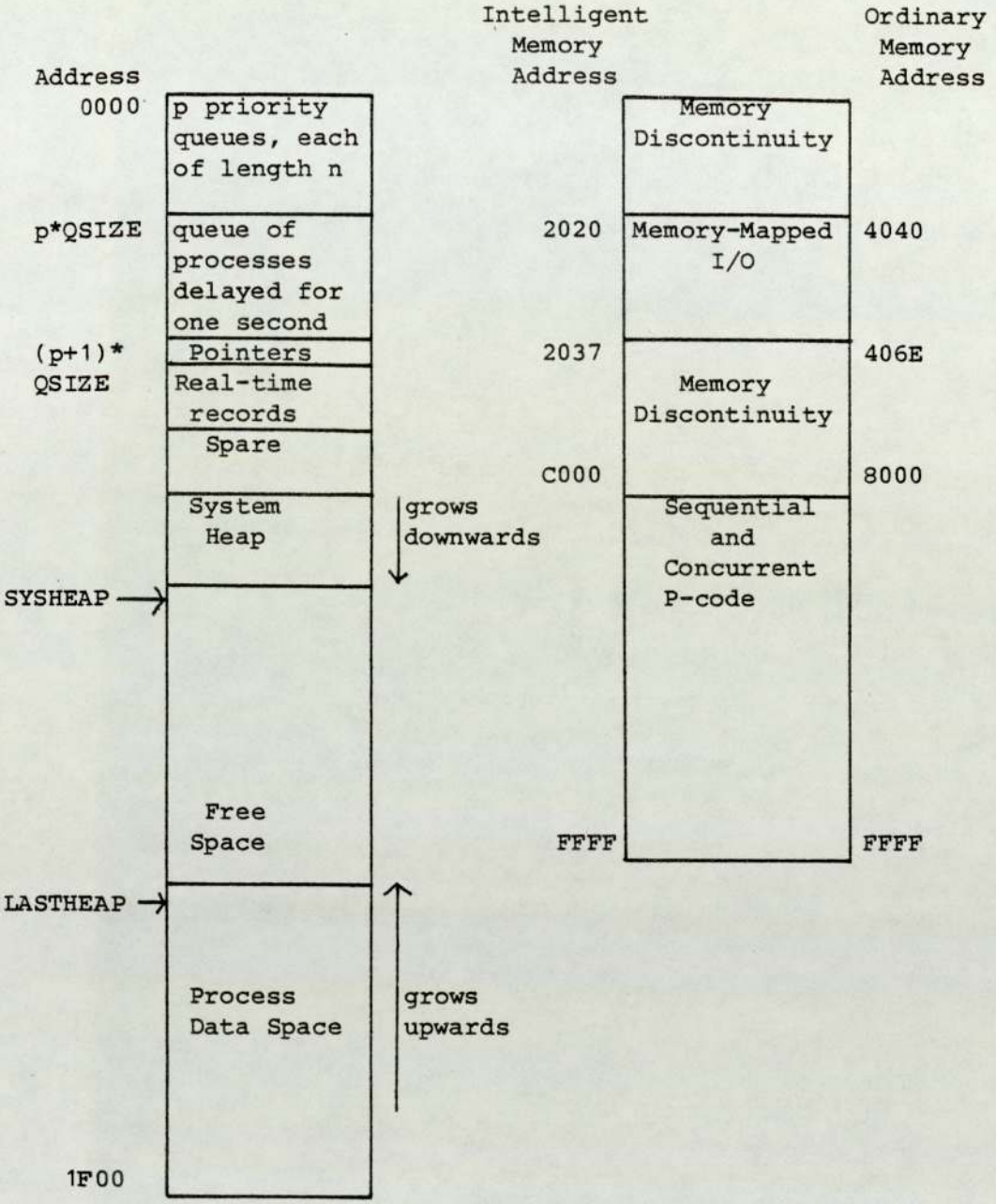


Fig. 8.1. Concurrent Pascal Machine Memory

System Heap Pointer (SYSHEAP) points to the end of the System Heap and the data pointer (LASTHEAP) points to the last Process Data Space allocated. Each process within the system has a unique index. The NEXTINDEX counter indicates the next free index number that can be allocated if a new process enters the system. This is followed by a list of processes corresponding to this indexing system. Thus, the first element of this list contains the process head address for the process whose index is zero, the second element points to process 1 and so on.

Following the system pointers is the real-time record. This consists of two words of data. RTREC is a count of the number of milliseconds since the last second interval, this is incremented by twenty whenever a timer interrupt occurs. As soon as RTREC reaches 1000 it is reset to zero and the number of seconds count, RTSECS, is incremented by 1. This is the value that is accessed when the "RTM" P-code instruction is called.

There is then some unused space for future development, followed by the system heap. This should not be confused with the heap of a Concurrent Pascal process. The system heap is used to store gates (which control monitors) and process heads (which define processes).

The structure of a gate is shown in Fig. 8.3. It consists of a single one-word boolean variable called OPEN which indicates whether the monitor is currently in use or not (1 if in use, 0 otherwise). If a process attempts to call a monitor and it is in use then it is delayed and placed in the process queue where it must wait its turn to access the monitor. A process queue is shown in Fig.8.4, it consists of a HEAD pointer which points to the first process in the

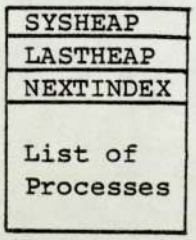


Fig. 8.2. Concurrent P-code System Pointers.

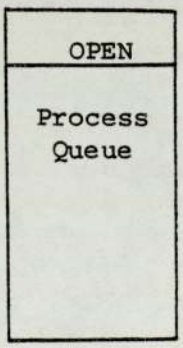


Fig. 8.3. A Gate.

queue and a TAIL pointer which points to the last process. The queue is cyclic and is large enough to accomodate all n processes.

A process head is shown in Fig. 8.5. It consists of various attributes, an area to dump the processor registers when the process is pre-empted (finishes its time slice) and some spare space.

The attributes are accessed by a displacement from the process head register (PHREG = register 3 of PCU) and are defined as follows:-

INDEXATR - The index of this process.

HEAPATR - Not used. In the original implementation this was the heap register. This is now an internal ALU register.

LINEATR - The line number where the last Sequential Pascal program to be run by this process terminated.

RESATR - The result (exception code) of the last Sequential Pascal program to be run by this process. If the process itself generates an exception then its code can be found here.

RUNATR - Not used.

SLICEATR - The time slice that the process has used.

NESTATR - The depth of nesting of monitor calls.

PRIATR - The priority of the process:-

0 - If executing a monitor routine.

1 - If performing I/O.

2 - OtherwIse.

OVERTIME 1 - If a process has used up its allocated time slice. This could happen if it was executing a

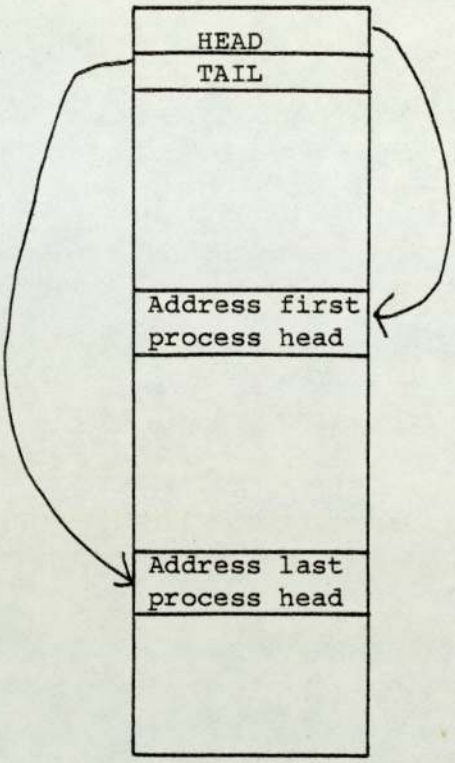


Fig. 8.4. A process Queue.

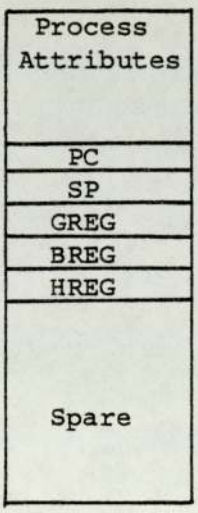


Fig. 8.5. A Process Head.

monitor routine and could not be pre-empted.

0 - Otherwise.

JOBATR — 0 - If executing Concurrent Pascal code.

1 - If executing Sequential Pascal code.

CONTATR - If this attribute is set to zero then the Sequential Pascal program currently being executed by this process will terminate as soon as the next "FSP" false jump instruction is executed.

OPCODE, ARG1-4, OPLINE - No longer used.

CONSTATR - The base address of large Concurrent Pascal constants (strings, real numbers and sets).

DEVICEATR - Used during an I/O operation. Indicates the device to be operated on.

0 = VDU (ACIA)

1 = PROCESSOR (PIA)

OPERATION — The I/O operation to be performed (also depends on DEVICEATR).

0 = INPUT or SEND

1 = OUTPUT or RECEIVE

DATAATR - Where the byte to be input or output is stored.

WAITING - Used when a process is performing I/O to a PIA. If a message has been sent and the process is waiting for a reply then WAITING is set to 1 to indicate this, otherwise it is set to zero.

LASTTIME - The last time (in milliseconds) that this process ran.

The system heap grows downwards. Hopefully, it should never meet the Process Data Space which grows upwards from $1F00. However, if it does then a system error will be generated. The

process data space is allocated contiguously as each new process is initiated (Fig. 8.6). The data space for each individual process consists of a permanent variable area, a stack area and a heap area (Fig. 8.7). Locations $4040 upwards contain the memory-mapped I/O devices as shown in Fig. 8.8. Finally, the Concurrent and Sequential Pascal code is stored in $8000 onwards as shown in Fig. 8.9.

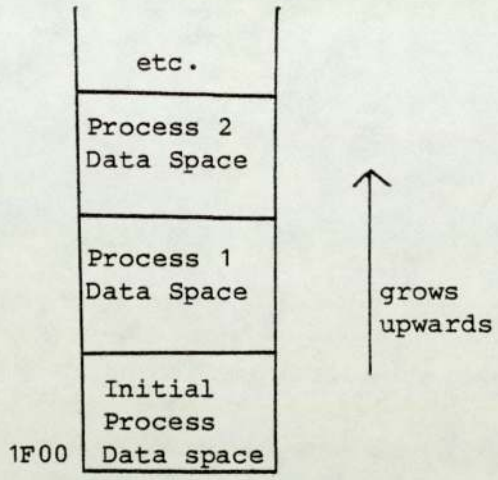


Fig. 8.6. Allocation of Process Data Space.

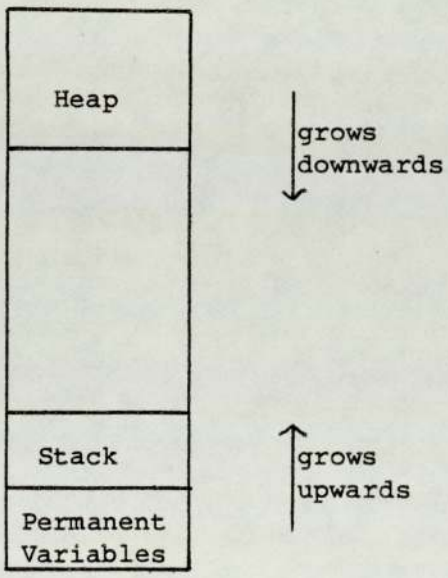


Fig. 8.7. Process Data Space.

Intelligent Memory Address		Ordinary Memory Address
2020	PIAs 1-3 Not Used	4040
2030	PIA 4 Communicates with other processor	4060
2034	ACIA Control reg.	4068
2035	ACIA Data reg.	406A
2038	PIA 5 Not used	4070

Note

Each PIA corresponds to 4 contiguous memory-mapped locations:-

1) A-side Data Register.

2) A-side Control Register.

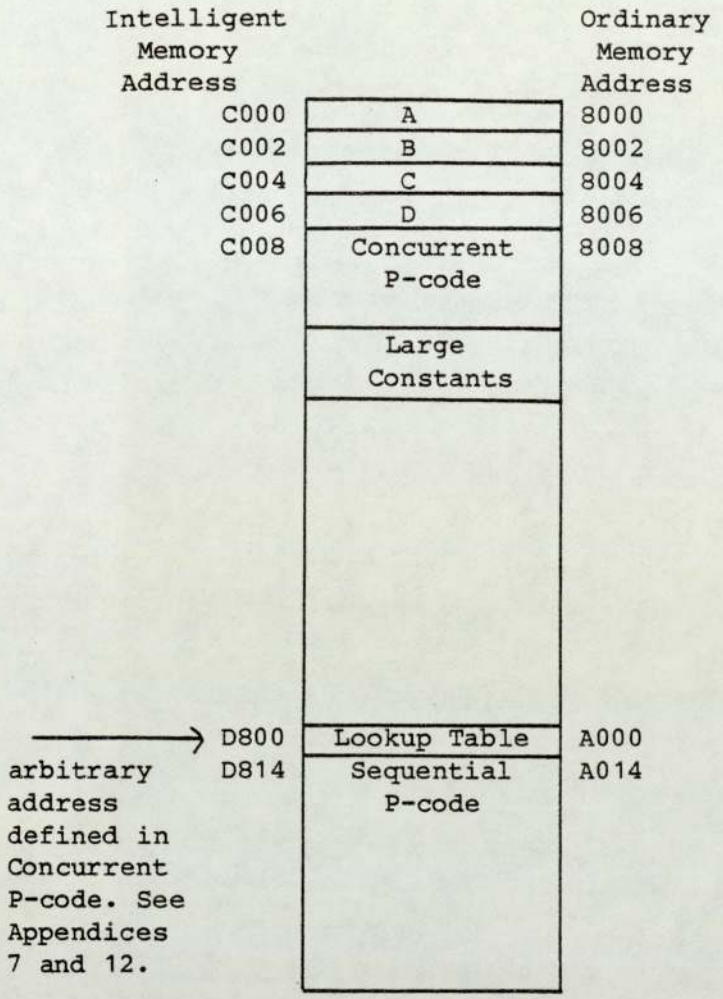
3) B-side Data Register.

4) B-side Control Register.

The A side is used to send a message.

The B side is used to receive a message.

Fig. 8.8. Memory-Mapped I/O Devices.



Note

A, B, C and D are all control data referring to the Concurrent Pascal program.

A = Total Program Size.
B = Total Code Size.
C = Amount of Stack Space needed by the Initial Process.
D = Amount of Permanent Variable Space needed by the Initial Process.

Fig. 8.9. Allocation of Concurrent and Sequential P-code.

8.2 Subroutines

A series of subroutines are used for monitor policing and process scheduling. They are as follows:-

Running Enter (RUNENT)

This subroutine is called when a process enters a monitor. The Priority attribute of the calling process is set to zero and the Nesting attribute is incremented by 1.

Running Leave (RUNLEAVE)

This subroutine is called when a process leaves a monitor. It decrements the nesting attribute by 1. If this makes it zero then the process is no longer in a monitor and so its priority attribute is set to 2.

Initialise Queue (QINIT)

This subroutine sets the head and tail pointers of a process queue to zero.

Put in Queue (PUTINQ)

This procedure puts a process in a queue (either within a gate or one of the ready-to-run queues or the one-second queue). The current process (as defined by the Process Head Register, R3 of the PCU) is put in the next available queue entry. The tail displacement is incremented cyclically to indicate that this entry is now in use.

Examine Queue (EXAMINEQ)

Examines a queue to see if it is empty or not.

Transfer Queues (MPUTQ)

Transfers a process from a monitor process queue to the priority 0 queue. This would be called to allow a process that was waiting for a monitor to access it once it becomes free.

Get Next Queue Entry (GETQ)

Fetches the next process head address from a queue.

Serve Process (SERVE)

This subroutine is called when a process is given processor time. The process's registers are loaded up from the process head.

Pre-empt Process (PREMPT)

Performs the opposite function to SERVE. This is called when a running process is pre-empted to allow another process to run. The process's registers are dumped into the process head.

Update Process Head (UPDATE)

Updates the Slice, Last Time and Overtime attributes for the current process. This subroutine is called during a Timer Interrupt.

8.3 Monitor Policing P-code Instructions

Begin Monitor (BGM)

A monitor is introduced into the system by means of a "BGM" instruction. This performs the same function as a "BGC" Begin Class instruction but it also creates a gate on the System Heap. This is done by adding the length of a gate to the system heap pointer and initialising the process queue. The Begin Monitor instruction is always found at the start of a monitor's initial statement. This means that the process must be executing monitor code and so OPEN must be set to false. Also, the Running Enter subroutine must be called. The address of the gate is also written to the address pointed to by the G register.

Enter Monitor (ENM)

The "ENM" instruction is used to enter a monitor subroutine. It performs the same function as an ordinary "ENT" instruction but it also calls a Running Enter operation and checks the gate to see if it is open. If the gate is open it will be closed and the process will execute the monitor code. If the gate is closed then the process will be put in the queue and pre-empted and a process reschedule operation will take place.

End and Exit Monitor

This instruction is called when a process leaves a monitor routine (either a procedure or the initial routine). As well as performing the standard exit from a procedure routine it also

examines the process queue to see if any processes are waiting to run. If they are then the next process is put in the ready-to-run queue and the gate will stay closed. If there are no processes waiting then the gate will be opened. Following this, a Running Leave operation is made. This will be followed by a Process Reschedule operation.

8.4 Process Scheduling

The process scheduling operations fall into three categories:-

1) Initialising the Initial Process.

2) Timer Interrupts.

3) Operations explicitly called by P-code instructions.

The Initial Process is treated as a parameterless process which will be used to explicitly initialise all the other (child) processes. The first four words of this process contain control data. The first word specifies the total program length (code + large constants). The second word contains the code length alone. This is followed by the amount of stack space that will be required by the process and the amount of space it requires for permanent variables. A process head is set up on the system heap and enough space for the stack and permanent variables is set up in the process data area. The process attributes are initialised as required. When all this has been done the first P-code instruction is executed.

The timer interrupt code consists of some fault-tolerant tests which are described in Chapter 5, reloading the interrupt counter,

updating the real-time records, updating the process attributes and rescheduling the processes. The real-time record (RTREC) is updated by adding twenty to it (an interrupt occurs every 20 ms). If this value reaches 1000 (one second) then it is reset to zero and two things happen. First, the real time in seconds since system activation is incremented by 1. Secondly, the queue of processes that have been delayed for one second is examined. If it is not empty then any entries in it are transferred to the ready-to-run queue (priority zero).

Following this, a reschedule operation takes place according to the following criteria:-

> A process can only be pre-empted by a process of higher priority or a process of the same priority if the Overtime attribute is true. A priority 0 process cannot be pre-empted even if Overtime is true.

If this is not satisfied then execution of the current process will continue from the point where it was interrupted.

The reschedule consists of pre-empting the current process (if there is one) and then scanning each queue in turn (starting at zero) until a process is found. When it is, a Serve operation is performed on it and it will be given a time slice. If no processes are found then the processor loops round waiting for an interrupt in the hope that some processes have been delayed for one second and will appear in the ready-to-run queue when their delay is completed.

The P-code instructions that deal with process scheduling are

outlined below.

Initialise Process (IPC)

The parameters of a process indicate its access rights, i.e. which monitors and classes it can use. When the initial process initialises a child using the high level "INIT" primitive it generates an "IPC" instruction. Any parameters will be on the stack of the initial process. The first step, therefore, is to transfer them to the permanent variable area of the new process.

The next step is to pre-empt the current (parent) process so that the new process will obtain a time slice. Following this, the CONST attribute is set to the same value as its parent since, being part of the same Concurrent Pascal program, they share the same constant address. Next, the process data space is created, some of which has already been used when the parameters were copied across. An entry on the system heap is also made containing a process head. Finally, the process is placed in the priority 2 queue and a reschedule takes place.

Delay Process (DLY) and Continue Process (CONTINUE)

These correspond to the high level DELAY and CONTINUE primitives. They have one parameter, the queue address where the process is delayed. This is a high level Concurrent Pascal queue and consists of one word of memory. A Delay operation pre-empts the process and puts it in the queue. It then tests the monitor queue, if it is empty then the gate is opened, otherwise the next process

waiting to gain access to the monitor is transferred to the ready-to-run queue. In either event a reschedule takes place.

Once a process has been delayed it remains in the queue until it is CONTINUEd. This consists of transferring it to the ready-to-run queue. If the queue is empty (zero) then this will not be done. In either case the End Monitor instruction is then called to make the calling process exit the monitor.

8.5 Real-Time and I/O Instructions

Input/Output (IOP)

The "IOP" instruction reads its three parameters off the stack. These consist of the Device type, the Operation Address and the Data Address (see Attributes description). For historical reasons, these values are transferred to the process head.

The DEVICE parameter is examined. If it is non-zero a jump is made to the section of code dealing with PIA I/O. If not, the operation is read, if it is zero an input operation takes place, otherwise an output occurs. In either case this consists of scanning the ACIA Control Register to see if it is ready to perform the I/O. If it is then the Data Register is written to or read from as appropriate. If the ACIA is not ready then the program counter and stack pointer registers are restored and a reschedule operation will take place. This means that the next time this process gains a time slice it will perform the I/O operation again. This will continue until the ACIA is ready.

If PIA I/O is required then the operation is examined. If it is zero then a message is sent, if not a message will be received. If the Send operation is required then a message will be sent and the WAITING attribute will be set to 1. A reschedule will then take place, the next time the process gains a time slice it will repeat the "IOP" instruction. This time, however, it will not send a message but it will try and receive a reply. If a reply has not arrived then a reschedule takes place again. This will continue until a reply does arrive. When it does, WAITING will be set to zero. If a receive message operation is requested then the process will be rescheduled whenever it examines the PIA and no message has arrived. When a message does arrive a reply will be sent and execution of the "IOP" is complete.

Real Time (RTM)

This instruction puts a copy of the RTSECS real-time record onto the current process's stack.

Wait

The Wait instruction pre-empts the current process, puts it in the one second queue and calls a reschedule operation.

Appendix 9

Microcode Source Listings

These listings have been specially processed to convert them from PMA format into a format similar to AMDASM on the System 29 [3.2]. This is because the PMA format is too bulky. Compacting the source files in this way does not affect the structure of the microcode.

The microcode is contained in three files:-

MCOD

This file contains the Sequential P-code. The Mapping PROM can only address the first 256 locations. Therefore, any instruction which requires more than three microinstructions consists of a jump at its decode address to an area of microcode memory greater than address 256. In the case of Concurrent P-code instructions, a jump is made to microcode contained in another source file.

CPMCOD

This file contains all the Concurrent P-code instructions, together with the timer interrupt code.

FTMCOD

This file contains all the fault-tolerant microcode. This includes the code for processor communication.

```
*
* PRIME AMD MICRO-CODE MACRO ASSEMBLER
* VERSION 8, 2/10/82
        LIST
*
*
* CODE WORKING ON 19/08/81
*
*
*            ************************************************
*            * MICRO-CODE FOR AMD 16-BIT COMPUTER           *
*            *                                              *
*            *    P-CODE INSTRUCTION SET                    *
*            *                                              *
*            *    WRITTEN BY T.E.SHARP                      *
*            ************************************************
*
INTMEM      EQU 2            1 FOR INTELLIGENT MEMORY, 2 OTHERWI
ISTACK      EQU $1F00        INITIAL STACK POINTER VALUE
IHEAP       EQU $400        INITIAL HEAP POINTER VALUE
* EXCEPTION ERROR CODES
TERMINATED    EQU 0
OVERFLOWERR   EQU 1
POINTERERR    EQU 2
RANGEERROR    EQU 3
VARIANTERR    EQU 4
HEAPLIMIT     EQU 5
STACKLIMIT    EQU 6
* PROCESS ATTRIBUTE CODES
INDEXATR      EQU 0*2
HEAPATR       EQU 1*2
LINEATR       EQU 2*2
RESATR        EQU 3*2
RUNATR        EQU 4*2
SLICEATR      EQU 5*2
NESTATR       EQU 6*2
PRIATR        EQU 7*2
OVERTIME      EQU 8*2
JOBATR        EQU 9*2
CONTATR       EQU 10*2
OPCODE        EQU 11*2
ARG1          EQU 12*2
ARG2          EQU 13*2
ARG3          EQU 14*2
ARG4          EQU 15*2
OPLINE        EQU 16*2
CONSTATR      EQU 17*2
DEVICEATR     EQU 18*2
OPERATION     EQU 19*2
DATAATR       EQU 20*2
WAITING       EQU DATAATR+2
LASTTIME EQU WAITING+2
PCREG    EQU LASTTIME+2
SPREG    EQU PCREG+2
REGG     EQU SPREG+2
REGB     EQU REGG+2
REGH     EQU REGB+2
PHLENGTH EQU REGH+10                    ALLOW SPARE
```

```
* MEMORY MAP.
P             EQU 3                    NUMBER OF PRIORITIES
NUMPROCESS    EQU 4                    MAXIMUM NUMBER OF PROCESSE
QSTART        EQU 0                    START OF QUEUE DATA STRUCT
QLENGTH       EQU 2*NUMPROCESS
TOTQLENGTH    EQU QLENGTH+4            TOTAL LENGTH OF QUEUE STRU
ONESECQ       EQU QSTART+P*TOTQLENGTH            ADDRESS OF 1 S
SYSHEAP       EQU (P+1)*TOTQLENGTH     SYSTEM HEAP POINTER ADDRES
LASTHEAP      EQU SYSHEAP+2            NEXT AVAILABLE PROCESS DAT
NEXTINDEX     EQU LASTHEAP+2            NEXT INDEX COUNTER
RTREC         EQU NEXTINDEX+2+2*NUMPROCESS        REAL TIME REC
RTSECS        EQU RTREC+2              REAL TIME IN SECONDS
SPARE         EQU RTSECS+2
SPARELENGTH   EQU 10
GATELENGTH    EQU 2+TOTQLENGTH
* DEVICE EQUATES
VDU           EQU 0
* OTHER EQUATES
PROGSTART     EQU $C000*INTMEM
ACIACS        EQU $2034*INTMEM           ACIA CONTROL REGISTER
ACIADA        EQU ACIACS+INTMEM         ACIA DATA REGISTER ADDRE
          ENT START
          ENT START1
          ENT RNI
          ENT WRTS
          ENT EXT
          ENT ENM
          EXT TERMEXP
          EXT OVFLOWEXP
          EXT  POINTEXP
          EXT  RANGEEXP
          EXT  VARIANTEXP
          EXT  HEAPEXP
          EXT  STACKEXP
          EXT EXCEPTION
*
*             RESET SEQUENCE STARTS HERE
*
RESET:  ALU & CONTROL ,IOEN,INTDIS & CNTLB $7F & DATAPATH &
        MEMCONT ,,HREQ & AM2904 & PCUNOP & JUMP INIT
0000: 48 00 29 1F 30 01 1B FF F0 03 02 5F

*
* INTERRUPT ADDRESS (MUST BE $0001).
*
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & XJUMP TIMER
0001: 48 62 29 1F 30 09 78 07 F0 03 XX XX

*
* INVALID INSTRUCTION
*
INVALID:  ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT & CONTROL & JUMPX SINVMESS
0002: 48 62 29 1F 30 09 78 07 F0 03 XX XX

*
*
```

```
*           ************************************************

*           *     P-CODE INSTRUCTIONS                      *

*           ************************************************

*
*
* PUSH CONSTANT ADDRESS
* PCA DISPLACEMENT                        OP-CODE = 2
*
*        PHREG + JOBATTR -> MAR.
PCA:  ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
      DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
      IMMD JOBATR & CONT
0003: 40 62 29 9E 56 28 78 07 F0 0E 00 12

*        READ JOB MODE, ZREG ("DISPLACEMENT") -> WREG
*        REST OF CODE IS IN CONCURRENT P-CODE FILE
      ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
      JUMP TESTMODE
0004: CA 7B 69 0F 30 3B F9 87 F0 03 01 01

*
* PUSH LOCAL ADDRESS
* PLA DISPLACEMENT              CODE = 4
*
*        FORM ADDRESS, READ INSTR N+1
PLA:  ALU YBUS ADD & DAB & OEY & PCUPUSH & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB &
      CNTLB BREG.LS.4 & CONT
0005: C2 61 A9 CF 98 7B F8 07 F0 0E 00 00

*        WRITE ADDRESS TO STACK
      ALU YBUS PASS & AB & OEY & PCUNEXT & AM2904 &
      CARRYCTL & DATAPATH ZZI ,,,,,LDMAR &
      MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP RNI
0006: 58 62 29 9F 18 3F 78 07 F0 03 01 00

*
* PUSH GLOBAL ADDRESS
* PGA DISPLACEMENT          CODE = 6
*
*        FORM ADDRESS, READ INSTR N+1
PGA:  ALU YBUS ADD & DAB & OEY & PCUPUSH & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB &
      CNTLB GREG.LS.4 & CONT
0007: C2 61 A9 CF 98 7B F8 87 F0 0E 00 00

*        WRITE ADDRESS TO STACK
      ALU YBUS PASS & AB & OEY & PCUNEXT & AM2904 &
      CARRYCTL & DATAPATH ZZI ,,,,,LDMAR &
      MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP RNI
0008: 58 62 29 9F 18 3F 78 07 F0 03 01 00
```

```
*
* PUSH CONSTANT
* PSC VALUE          CODE = 8
*
*         VALUE -> D-REG, READ INSTR N+1
PSC:  ALU YBUS PASSR & DAQ & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0009: 42 63 69 CF 98 7B 78 07 F0 0E 00 00

*         WRITE VALUE TO STACK
       ALU YBUS PASS & AB & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP RNI
000A: 58 62 29 9F 18 3F 78 07 F0 03 01 00

*
* PUSH LOCAL
* PSL DISPLACEMENT              CODE = 10
*
*         FORM ADDRESS, READ INSTR N+1
PSL:  ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & CNTLB BREG.LS.4 & JUMP MSTRAN
000B: CA 61 AB 8F 30 3B F8 07 F0 03 01 10

*
* PUSH GLOBAL
* PSG DISPLACEMENT              CODE = 12
*
*         FORM ADDRESS , READ INSTR N+1
PSG:  ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & CNTLB GREG.LS.4 & JUMP MSTRAN
000C: CA 61 AB 8F 30 3B F8 87 F0 03 01 10

*
* PUSH WORD
* PSW                CODE = 14
*
*         SP -> MAR
PSW:  ALU YBUS PASS & AB & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP PUSHWORD
000D: 58 62 29 9F 32 69 78 07 F0 03 01 08

*
* PUSH BYTE
* PSB                CODE = 16
*
*         SP -> MAR
PSB:  ALU YBUS PASS & AB & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP PUSHBYTE
000E: 58 62 29 9F 32 69 78 07 F0 03 01 0C

*
```

```
* PUSH REAL
* PSR                                    OP-CODE = 18
*
*         SP -> MAR, LOAD AM2910 COUNTER WITH 2 (NO. OF WORDS
          IN A REAL NO. - 2)
*         6 -> WREG (NO. OF BYTES IN A REAL NO. - 2)
PSR:  ALU AUR PASSR & DAQ & OEY & PCUSP & AM2904 SHIFTEN &
       CARRYCTL COEQ1 & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB & CONTROL ROM & CNTLB 0 & RTB &
       BRAM WREG & LDCT 2 & IMMD 2
000F: D2 43 69 9F 32 68 F1 87 F4 0C 00 02

*         READ SOURCE ADDRESS, SP+2 -> SP
       ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT REQB MREQ & CONTROL & JUMP MPUSH
0010: 48 62 29 1F 18 7B 78 07 F0 03 01 16

*
*
* PUSH SET
* PSS                                    OP-CODE = 20
*
*         SP -> MAR, LOAD AM2910 COUNTER WITH 6 (NO. OF WORDS
          IN A SET - 2)
*         14 -> WREG (NO. OF BYTES IN A SET - 2)
PSS:  ALU AUR PASSR & DAQ & OEY & PCUSP & AM2904 SHIFTEN &
       CARRYCTL COEQ1 & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB & CONTROL ROM & CNTLB 0 & RTB &
       BRAM WREG & LDCT 6 & IMMD 6
0011: D2 43 69 9F 32 68 F1 87 F4 0C 00 06

*         READ SOURCE ADDRESS, SP+2 -> SP
       ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT REQB MREQ & CONTROL & JUMP MPUSH
0012: 48 62 29 1F 18 7B 78 07 F0 03 01 16

*
* FIELD
* FLD DISPLACEMENT                                OP-CODE = 22
*
*         SP -> MAR, ZREG ("DISPLACEMENT") -> WREG, READ INSTR
          N+1
FLD:  ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM WREG & CONT
0013: C2 7B 69 8F 32 7B F9 87 F0 0E 00 00

*         READ VALUE FROM STACK, PC+2 -> MAR
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & JUMP FLDENT
0014: 58 62 29 9F 18 3B 78 07 F0 03 01 1D

*
* PUSH ARRAY COMPONENT
* PAC MIN, MAX-MIN, LENGTH       CODE = 24
*
*         READ "MAX-MIN", ZREG ("MIN") -> WREG, PC+2 -> MAR
```

```
PAC:  ALU REG PASSR & DAQ & OEY & PCUNEXT & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB &
      CNTLB WREG.LS.4 & JUMP PSHARRAY
0015: CA 7B 69 8F 18 3B F9 87 F0 03 01 1F

*
* TEST POINTER
* PTR                                          OP-CODE = 26
*
*       SP -> MAR
PTR:  ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
      DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
      CONT
0016: 50 62 29 9F 32 69 78 07 F0 0E 00 00

*       READ VALUE, PC -> MAR
      ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
      DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
      CONT
0017: 40 62 29 9F 30 3B 78 07 F0 0E 00 00

*       ZREG (VALUE) -> YBUS, READ INSTR N+2
*       EXCEPTION IF = 0
      ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 ,OECT &
      CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL & TEST DIRIN UAEQB &
      CJPX POINTEXP
0018: 42 63 69 0F 30 3B 78 03 F3 53 XX XX

*       JUMP MAP, PC+2 -> MAR
      ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
      DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & JMAP
0019: 48 62 29 9F 18 29 78 07 F0 02 00 00

*
* CHECK TAG FIELD
* CTF DISPLACEMENT,TAGSET                      OP-CODE = 28
*
*       SP -> MAR, ZREG ("DISPLACEMENT") -> WREG
CTF:  ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 & CARRYCTL &
      DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB &
      CONTROL ROM & RTB & BRAM WREG & JUMP CHCKTAG
001A: CA 7B 69 8F 32 69 F9 87 F0 03 01 28

*
* CHECK RANGE
* RNG MIN,MAX                                   OP-CODE = 30
*
*       SP -> MAR, READ "MAX", ZREG ("MIN") -> WREG
RNG:  ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 & CARRYCTL &
      DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
      CONTROL ROM & RTB & BRAM WREG & CONT
001B: C2 7B 69 8F 32 7B F9 87 F0 0E 00 00

*       READ VALUE FROM STACK, ZREG ("MAX") -> XREG
      ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
```

```
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
       CONT
001C: C2 7B 69 0F 30 3B FA 07 F0 0E 00 00

*        ZREG (VALUE) - WREG ("MIN") -> YBUS, EXCEPTION IF < 0
         , PC+2 -> MAR
       ALU YBUS SUBS & DAB & OEY & PCUNEXT & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
       TEST DIRIN SALSB & CJPX RANGEEXP
001D: C2 61 29 8F 18 29 F9 83 F7 33 XX XX

*        XREG ("MAX") - ZREG (VALUE) -> YBUS, EXCEPTION IF < 0
         , PC+2 -> MAR, REA
       ALU YBUS SUBR & DAB & OEY & PCUNEXT & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
       TEST DIRIN SALSB & CJPX RANGEEXP
001E: C2 60 A9 8F 18 3B FA 03 F7 33 XX XX

*        READ INSTR N+2, PC+2 -> MAR
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & JMAP
001F: 58 62 29 9F 18 3B 78 07 F0 02 00 00

*
* COPY BYTE
* CPB           CODE = 32
*
*        SP -> MAR
CPB:  ALU YBUS PASS & AB & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP COPYBYTE
0020: 58 62 29 9F 32 69 78 07 F0 03 01 31

*
* COPY WORD
* CPW           CODE = 34
*
*        SP -> MAR
CPW:  ALU YBUS PASS & AB & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP COPYWORD
0021: 58 62 29 9F 32 69 78 07 F0 03 01 35

*
* COPY REAL
* CPR                                        OP-CODE = 36
*
*        NOTE THE COPY OF SP PUT IN THE ALU AND USED TO ADDRES
         S THE STACK.
*        SP+8 -> SP,MAR,WREG, 8 -> QREG
CPR:  ALU RQPT PASSR & DAQ & YOFF & PCU ,,PCUDA A1 B1 &
       AM2904 & CARRYCTL & DATAPATH ZZI ,,,,PCUY LDMAR &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
       IMMD 8 & CONT
0022: D3 3B 68 9F 52 68 F9 87 F0 0E 00 08
```

```
*         READ STARTING ADDRESS, SP+2 -> SP, WREG (SP') - QREG
          (16) -> WREG,MAR
*         LOAD AM2910 COUNTER WITH 6 (NO. OF WORDS IN A REAL NO
          . - 2)
        ALU REG SUBS & AQ & OEY & PCUPOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & ARAM WREG & RTB &
        BRAM WREG & LDCT 2
0023: C0 79 6B 9F 18 7B F9 9F F4 0C 00 02

*         READ FIRST VALUE, WREG (SP') + 2 -> MAR,WREG, ZREG (A
          DDRESS) -> R6 OF P
        SPF14 INCTWO & AB & OEY & PCU 1 0 PCUDZ A6 B6 &
        AM2904 & CARRYCTL COEQ1 &
        DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM WREG & JUMP MPULL
0024: C8 20 2B 8F 7D BB F9 87 F4 03 01 46

*
* COPY SET
* CPS                                          OP-CODE = 38
*
*         NOTE THE COPY OF SP PUT IN THE ALU AND USED TO ADDRES
          S THE STACK.
*         SP+16 -> SP,MAR,WREG, 16 -> QREG
CPS:   ALU RQPT PASSR & DAQ & YOFF & PCU ,,PCUDA A1 B1 &
        AM2904 & CARRYCTL & DATAPATH ZZI ,,,,PCUY LDMAR &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
        IMMD 16 & CONT
0025: D3 3B 68 9F 52 68 F9 87 F0 0E 00 10

*         READ STARTING ADDRESS, SP+2 -> SP, WREG (SP') - QREG
          (16) -> WREG,MAR
*         LOAD AM2910 COUNTER WITH 6 (NO. OF WORDS IN A SET - 2
          )
        ALU REG SUBS & AQ & OEY & PCUPOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & ARAM WREG & RTB &
        BRAM WREG & LDCT 6
0026: C0 79 6B 9F 18 7B F9 9F F4 0C 00 06

*         READ FIRST VALUE, WREG (SP') + 2 -> MAR,WREG, ZREG (A
          DDRESS) -> R6 OF P
        SPF14 INCTWO & AB & OEY & PCU 1 0 PCUDZ A6 B6 &
        AM2904 & CARRYCTL COEQ1 &
        DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM WREG & JUMP MPULL
0027: C8 20 2B 8F 7D BB F9 87 F4 03 01 46

*
* COPY TAG
* CPT LENGTHWORDS                                  OP-CODE =
          40
*
*         READ INSTR N+1, -ZREG ("LENGTHWORDS") -> WREG, SP ->
          MAR
CPT:   ALU REG COMPLR & DAQ & OEY & PCUSP & AM2904 &
```

```
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP COPYTAG
0028: CA 7B E9 8F 32 7B F9 87 F4 03 01 38

*
* COPY STRUCTURE
* CST LENGTHWORDS                                    OP-CODE = 42
*
*        SP -> MAR, -ZREG ("LENGTHWORDS") -> QREG, READ INSTR
         N+1
CST:  ALU QPT COMPLR & DAQ & OEY & PCUSP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP COPYSTRUCT
0029: CA 33 E9 8F 32 7B F9 87 F4 03 01 3E

*
* INITIALISE POINTER VARIABLE
* IPV STACKLENGTH+LENGTH, LENGTH                          OP-COD
         E = 44
*
*        READ "LENGTH", BREG - ZREG ("STACKLENGTH+LENGTH") ->
         QREG, SP -> MAR
IPV:  ALU QPT SUBR & DAB & PCUSP & AM2904 & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM BREG & JSB NEWHEAP
002A: CA 30 A9 8F 32 7B F8 07 F4 01 01 4A

*        PC+2 -> MAR, REAF INSTR N+2
       ALU YBUS PASS & AB & PCUNEXT & AM2904 &
       DATAPATH ,,,,,,LDMAR & CONTROL & MEMCONT REQB MREQ &
       JMAP
002B: 48 62 29 9F 18 3B 78 07 F0 02 00 00

*
* INITIALISE POINTER VARIABLE AND CLEAR
* IPVC STACKLENGTH+LENGTH, LENGTH                         OP-CO
         DE = 46
*
*        READ "LENGTH", BREG - ZREG ("STACKLENGTH+LENGTH") ->
         QREG, SP -> MAR
IPVC:  ALU QPT SUBR & DAB & PCUSP & AM2904 & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM BREG & JSB NEWHEAP
002C: CA 30 A9 8F 32 7B F8 07 F4 01 01 4A

*        READ INSTR N+2, TREG (OLD HEAP VALUE) -> R6 OF PCU,MA
         R, 0 -> DREG
       ALU YBUS LOW & AB & OEY & PCU ,,PCUDZ A6 B6 & AM2904 &
       CARRYCTL & DATAPATH ,ENTREG ,,,,LDMAR LDD &
       MEMCONT REQB MREQ & CONTROL & CONT
002D: 40 64 09 DF 7D BB 78 07 F0 0E 00 00

*        WRITE ZERO TO HEAP, R6 (NEXT ADDRESS) + 2 -> MAR
*        INCREMENT WREG ("LENGTH") BY TWO, LOOP IF < 0
CLHEAP:  SPF14 INCTWO & AB & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 ,OECT & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR &
```

```
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM WREG & TEST DIRIN SALSB & CJP CLHEAP
002E: C0 20 29 9F 19 BF F9 83 F7 33 00 2E

*        PC+2 -> MAR, JUMP MAP
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & JMAP
002F: 48 62 29 9F 18 29 78 07 F0 02 00 00

*
* NOT WORD
* NOT            OP-CODE = 48
*
*        SP -> MAR, 1 -> WREG
NOT:  ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM WREG & IMMD 1 & CONT
0030: D2 7B 69 9F 32 68 F9 87 F0 0E 00 01

*        READ VALUE FROM STACK
       ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
0031: 40 62 29 9F 30 3B 78 07 F0 0E 00 00

*        NEGATE VALUE + 1 -> D-REG, READ INSTR N+2
       ALU YBUS SUBR & DAB & OEY & PCUSP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP WRTS
0032: CA 60 A9 CF 32 7B F9 87 F4 03 01 15

*
* ANW WORD
* ANW             OP-CODE = 50
*
*        SP -> MAR
ANW:  ALU YBUS PASS & OEY & AB & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JSB RW1W2
0033: 58 62 29 9F 32 69 78 07 F0 01 01 13

*        WORD1 + WORD2 -> D-REG, READ INSTR N+2
       ALU YBUS AND & DAB & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR LDD ,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & CNTLB WREG.LS.4 & JUMP WRTS
0034: CA 66 29 CF 32 7B F9 87 F0 03 01 15

*
* AND SET
* ANS                                              OP-CODE = 52
*
*        SP -> MAR, LOAD AM2910 COUNTER WITH 6 (NO. OF WORDS I
         N A SET -2)
ANS:  ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       LDCT 6
0035: 50 62 29 9F 32 69 78 07 F0 0C 00 06
```

```
*         READ FIRST TOP SET WORD, SP+16 -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDA A1 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL & IMMD 16 & CONT
0036: 40 62 29 9F 53 BA 78 07 F0 0E 00 10

*         READ FIRST BOTTOM SET WORD, SP+2 -> MAR, ZREG (FIRST
          TOP SET WORD) -> W
        ALU REG PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        JUMP ANDSET
0037: CA 7B 69 8F 18 7B F9 87 F0 03 01 4E

*
* OR WORD
* ORW             OP-CODE = 54
*
*         SP -> MAR
ORW:  ALU YBUS PASS & OEY & AB & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JSB RW1W2
0038: 58 62 29 9F 32 69 78 07 F0 01 01 13

*         WORD1 + WORD2 -> D-REG, READ INSTR N+2
        ALU YBUS OR & DAB & OEY & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR LDD ,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & CNTLB WREG.LS.4 & JUMP WRTS
0039: CA 67 A9 CF 32 7B F9 87 F0 03 01 15

*
* OR SET
* ORS                                              OP-CODE = 56
*
*         SP -> MAR, LOAD AM2910 COUNTER WITH 6 (NO. OF WORDS I
          N A SET -2)
ORS:  ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
        LDCT 6
003A: 50 62 29 9F 32 69 78 07 F0 0C 00 06

*         READ FIRST TOP SET WORD, SP+16 -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDA A1 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL & IMMD 16 & CONT
003B: 40 62 29 9F 53 BA 78 07 F0 0E 00 10

*         READ FIRST BOTTOM SET WORD, SP+2 -> MAR, ZREG (FIRST
          TOP SET WORD) -> W
        ALU REG PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        JUMP ORSET
003C: CA 7B 69 8F 18 7B F9 87 F0 03 01 53

*
* NEGATE WORD
```

```
* NGW             CODE = 58
*
*         SP -> MAR
NGW:   ALU YBUS PASS & AB & OEY & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
        CONT
003D: 50 62 29 9F 32 69 78 07 F0 0E 00 00

*         READ VALUE FROM STACK
       ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
003E: 40 62 29 9F 30 3B 78 07 F0 0E 00 00

*         NEGATE VALUE -> D-REG, READ INSTR N+2
NG1:   ALU YBUS COMPLR & DAQ & OEY & PCUSP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & JUMP WRTS
003F: 4A 63 E9 CF 32 7B 78 07 F4 03 01 15

*
* NEGATE REAL
* NGR                                   OP-CODE = 60
*
*         SP+4 -> Q,MAR, REAL1L -> IREG, SET EXP1 TO -1, M N ->
          0
NGR:   ALU REG HIGH & AQ & OEY & PCU QEU ,PCUAB A5 B1 &
        AM2904 ,,,,ES ,CEM & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR ,LDI & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM EXP1 &
        TESTF $03 & IREG REAL1L-1 REAL1L & CONT
0040: D0 78 69 BE 1A 69 FD 87 50 3E 00 A9

*         READ FIRST MANTISSA WORD, LOAD AM2910 COUNTER
*         1 -> MICRO C, Q-2 -> R6,MAR,TREG
       ALU YBUS PASS & AQ & YOFF & PCU ,SUB PCUAQ A4 B6 &
       AM2904 ,,,,,,,CEU & CARRYCTL &
       DATAPATH ,,,,,PCUY LDMAR & MEMCONT REQB MREQ &
     TESTF '13         SET CARRY
       CONTROL &         PHLC 1
0041: 49 62 68 9F 89 BB 78 07 E0 B4 00 01

*         -ZREG (NEXT REAL NO. WORD) - 1 + MICRO C -> IREG PORT

*         DECREMENT IREG, READ IN NEXT WORD
*         R6-2 -> MAR, LOAD MICRO C
NEGRL:   ALU REG COMPLR & DAQ & OEY & PCU ,SUB PCUAB A4 B6 &
        AM2904 ,,,EC ,,CEM CEU & CARRYCTL COEQST &
        DATAPATH ,,,ENCTR ,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL & RTB & TESTF '20 & RFCT
0042: C2 7B E1 8F 99 BB 78 06 CD 08 00 00

*         -ZREG (MANTISSA M.S.) + MC -> IREG PORT,QREG
*         IF NO OVERFLOW THEN SKIP NEXT 2 INSTRUCTIONS
*         Q+2 -> MAR (ADDRESS EXPONENT)
       ALU RQPT COMPLR & DAQ & OEY & PCU QEU ,PCUAQ A4 B4 &
       AM2904 ,OECT & CARRYCTL COEQST &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
```

```
        RTB &
        TESTF $36 & CJP NORM
0043: C2 3B E9 8E 09 29 78 03 FF 63 00 45

*         ARRIVED HERE IF OVERFLOW.
*         EXPONENT MUST BE INCREMENTED BY 1.
*         THIS IS DONE BY SUBTRACTING -1.
*         -1 -> EXP1, READ EXPONENT.
        ALU REG HIGH & AQ & OEY & PCUNOP &
        AM2904 ,,,,ES ,CEM & CARRYCTL & DATAPATH &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM EXP1 &
        TESTF $03 & JUMP ADDEXP
0044: C8 78 69 1F 30 3B FD 87 50 33 00 46

* NOTE THAT IF NORMALISATION IS REQUIRED THEN IT IS BECAUSE
* THE ORIGINAL NUMBER WAS 01 FOLLOWED BY ALL 0'S.
* THEREFORE WE ONLY NEED TO NORMALISE THE M.S. WORD AS
* THE OTHER TWO WORDS ARE ZERO.
*         NORMALISE, READ EXPONENT, LOAD MN
NORM:   SPF14 SLN & AB & OEY & PCUNOP &
        AM2904 SHIFTEN OECT ,,ES ,CEM & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT REQB MREQ & CONTROL ROM & RTB &
        CNTLB $10 &
        BRAM EXP1 &
        TESTF $3A & CJP NORM
0045: C0 40 29 1F 30 3B F5 83 57 A3 00 45

*         ZREG (EXPONENT) - EXP1 -> EXP1, RELOAD IREG
ADDEXP:   ALU REG SUBS & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,,LDI ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM EXP1 & IREG EXP1-1 REAL1M &
        CONT
0046: C2 79 29 2F 30 09 FD 87 F4 0E 00 8A

*         SHIFT QREG (REAL1M) RIGHT TO RESTORE SIGN AND PUT IN
          DREG
*         INCREMENT IREG (NOW POINTS TO REAL1I) R6+2 -> R6,MAR
SHSIGN:   ALU LDR PASS & AQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 SHIFTEN & CARRYCTL &
        DATAPATH ,,,ENCTR INC ,LDMAR LDD & MEMCONT REQB &
        CONTROL &
        RTB & CNTLB $05 & CONT
0047: C0 0A 65 DF 19 A9 70 2F F0 0E 00 00

*         WRITE REAL BACK (M.S. MANTISSA ALREADY IN DREG)
*         WRITE RESULT, R6+2 -> MAR, IREG PORT -> DREG
*         INCREMENT IREG, LOOP BACK IF B-PORT <> A-PORT (EXP1 O
          R REAL1L)
RB1:    ALU YBUS PASS & AB & OEY & PCU ,,PCUAB A4 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,ENCTR INC ,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTB &
        TESTF AEQB & CJP RB1
0048: C0 62 25 DF 19 BF 78 07 F0 03 00 48

*         WRITE LAST RESULT, PC -> MAR
        ALU YBUS PASSR & AB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
```

```
0049: 48 63 29 9F 30 3F 78 07 F0 03 01 00

*
* ADD WORD
* ADD              CODE = 62
*
*         SP -> MAR
ADD:   ALU YBUS PASS & OEY & AB & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JSB RW1W2
004A: 58 62 29 9F 32 69 78 07 F0 01 01 13

*         WORD1 + WORD2 -> D-REG, READ INSTR N+2
        ALU YBUS ADD & DAB & OEY & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR LDD ,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & CNTLB WREG.LS.4 & JUMP WRTS
004B: CA 61 A9 CF 32 7B F9 87 F0 03 01 15

*
* ADD REAL
* ADR                                          OP-CODE = 64
*
*         SP -> MAR, -1 -> XREG
ADR:   ALU REG HIGH & AQ & OEY & AM2904 & CARRYCTL & PCUSP &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM XREG & JSB ADRSUB
004C: D8 78 69 9F 32 69 FA 07 F0 01 01 58

*         STAGE 3 - ADD THE TWO NO.'S
*         REAL2L + REAL1L -> REAL1L
*         CARRY -> MICRO STATUS REGISTER, LOAD AM2910 COUNTER W
          ITH 2, PC -> MAR
        ALU REG ADD & AB & OEY & PCUNOP & AM2904 ,,,,,,,CEU &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & ARAM REAL2L & BRAM REAL1L &
        TEST MICRO DLOAD & LDCT 2
004D: C0 79 A9 9F 30 29 FD 77 E0 7C 00 02

*         REAL2I + REAL1I + MACHINE CARRY -> REAL1I,QREG, LOAD
          MICRO STATUS REGIS
*         LOAD I-REG, READ INSTR N+2
        ALU RQPT ADD & AB & OEY & PCUNOP &
        AM2904 ,,,EC ,,CEM & CARRYCTL COEQST &
        DATAPATH ,,,,,,,,LDI & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & ARAM REAL2I & BRAM REAL1I &
        TEST MICRO DLOAD & IREG WREG REAL1L & CONT
004E: C0 39 A9 3F 30 3B FC EE DC 7E 00 A3

*         REAL2M + REAL1M + MACHINE CARRY -> REAL1M, JUMP TO
*         NORMALISE CODE IF NO OVERFLOW, LOAD MSR C-BIT
        ALU REG ADD & AB & OEY & PCUNOP &
        AM2904 ,OECT ,EC ,,CEM & CARRYCTL COEQST & DATAPATH &
        MEMCONT & CONTROL ROM & RTB & ARAM REAL2M &
        BRAM REAL1M &
        TESTF $36 & CJP REALNORM
004F: C0 79 A9 1F 30 09 FC 62 DF 63 01 6A

*         ARRIVED HERE IF OVERFLOW HAS OCCURED
```

```
*         NO. CANNOT BE GREATER THAN 2*ORIGINAL VALUE SO ONLY
*         ONE SHIFT RIGHT REQUIRED.
*         SHIFT REAL1M RIGHT
        ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,,,,,,LDI & MEMCONT &
        CONTROL ROM & RTB &
        BRAM REAL1M & CNTLB $09 & IREG REAL1L-1 REAL1I &
        CONT
0050: C0 0A 29 3F 30 09 F4 4F F0 0E 00 99

*         SHIFT RIGHT 1 PLACE MC -> I-REG PORT -> MC ,INCREMENT
           I-REG AND REPEAT
*         2 TIMES.
        ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,ENCTR INC & MEMCONT & CONTROL &
        RTB & CNTLB $09 & TESTF AEQB & CJP *
0051: C0 0A 25 1F 30 09 70 4F F0 03 00 51

*         I-REG PORT (EXP1) + 1 -> DREG, SP -> MAR, JUMP TO WRI
          TE BACK CODE
*         DECREMENT I-REG (NOW POINTS TO REAL1L)
        ALU YBUS PASS & AB & OEY & PCUPUSH & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,ENCTR ,,LDMAR LDD &
        MEMCONT & CONTROL & RTB & JUMP REALWRT
0052: C8 62 21 DF 98 49 78 07 F4 03 01 71

*
* SUBTRACT WORD
* SUB             CODE = 66
*
*         SP -> MAR
SUB:   ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JSB RW1W2
0053: 58 62 29 9F 32 69 78 07 F0 01 01 13

*         WORD2 - WORD1 ->  D-REG, READ INSTR N+2
        ALU YBUS SUBS & DAB & OEY & PCUSP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB &
        CNTLB WREG.LS.4 & JUMP WRTS
0054: CA 61 29 CF 32 7B F9 87 F4 03 01 15

*
* SUBTRACT REAL
* SBR                                          OP-CODE = 68
*
*         SP -> MAR, -1 -> XREG
SBR:   ALU REG HIGH & AQ & OEY & AM2904 & CARRYCTL & PCUSP &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM XREG & JSB ADRSUB
0055: D8 78 69 9F 32 69 FA 07 F0 01 01 58

*         STAGE 3 - ADD THE TWO NO.'S
*         REAL2L - REAL1L -> REAL1L
*         CARRY -> MICRO STATUS REGISTER, LOAD AM2910 COUNTER W
          ITH 2, PC -> MAR
        ALU REG SUBS & AB & OEY & PCUNOP & AM2904 ,,,,,,,CEU &
```

```
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & ARAM REAL2L & BRAM REAL1L &
        TEST MICRO DLOAD & LDCT 2
0056: C0 79 29 9F 30 29 FD 77 E4 7C 00 02

*         REAL2I - REAL1I + MACHINE CARRY -> REAL1I,QREG, LOAD
          MICRO STATUS REGIS
*         LOAD I-REG, READ INSTR N+2
        ALU RQPT SUBS & AB & OEY & PCUNOP &
        AM2904 ,,,EC ,,CEM & CARRYCTL COEQST &
        DATAPATH ,,,,,,,,LDI & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & ARAM REAL2I & BRAM REAL1I &
        TEST MICRO DLOAD & IREG WREG REAL1L & CONT
0057: C0 39 29 3F 30 3B FC EE DC 7E 00 A3

*         REAL2M - REAL1M + MACHINE CARRY -> REAL1M, JUMP TO
*         NORMALISE CODE IF NO OVERFLOW, LOAD MSR C-BIT
        ALU REG SUBS & AB & OEY & PCUNOP &
        AM2904 ,OECT ,EC ,,CEM & CARRYCTL COEQST & DATAPATH &
        MEMCONT & CONTROL ROM & RTB & ARAM REAL2M &
        BRAM REAL1M &
        TESTF $36 & CJP REALNORM
0058: C0 79 29 1F 30 09 FC 62 DF 63 01 6A

*         ARRIVED HERE IF OVERFLOW HAS OCCURED
*         NO. CANNOT BE GREATER THAN 2*ORIGINAL VALUE SO ONLY
*         ONE SHIFT RIGHT REQUIRED.
*         SHIFT REAL1M RIGHT
        ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,,,,,,LDI & MEMCONT &
        CONTROL ROM & RTB &
        BRAM REAL1M & CNTLB $09 & IREG REAL1L-1 REAL1I &
        CONT
0059: C0 0A 29 3F 30 09 F4 4F F0 0E 00 99

*         SHIFT RIGHT 1 PLACE MC -> I-REG PORT -> MC ,INCREMENT
           I-REG AND REPEAT
*         2 TIMES.
        ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,ENCTR INC & MEMCONT & CONTROL &
        RTB & CNTLB $09 & TESTF AEQB & CJP *
005A: C0 0A 25 1F 30 09 70 4F F0 03 00 5A

*         I-REG PORT (EXP1) + 1 -> DREG, SP -> MAR, JUMP TO WRI
          TE BACK CODE
*         DECREMENT I-REG (NOW POINTS TO REAL1L)
        ALU YBUS PASS & AB & OEY & PCUPUSH & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,ENCTR ,,LDMAR LDD &
        MEMCONT & CONTROL & RTB & JUMP REALWRT
005B: C8 62 21 DF 98 49 78 07 F4 03 01 71

*
* SUBTRACT SET
* SBS                                              OP-CODE = 70
*
*         SP -> MAR, LOAD AM2910 COUNTER WITH 6 (NO. OF WORDS I
          N A SET -2)
SBS:   ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
```

```
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       LDCT 6
005C: 50 62 29 9F 32 69 78 07 F0 0C 00 06

*        READ FIRST TOP SET WORD, SP+16 -> R6,MAR
       ALU YBUS PASS & AB & PCU ,,PCUDA A1 B6 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & IMMD 16 & CONT
005D: 40 62 29 9F 53 BA 78 07 F0 0E 00 10

*        READ FIRST BOTTOM SET WORD, SP+2 -> MAR, ZREG (FIRST
         TOP SET WORD) -> W
       ALU REG COMPLR & DAQ & OEY & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP SUBSET
005E: CA 7B E9 8F 18 7B F9 87 F0 03 01 73

*
* MULTIPLY WORD
* MLW                                                OP-CODE = 72

*
*        0 -> XREG, SP -> MAR
MLW:  ALU REG LOW & AQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM XREG & JUMP MULTWORD
005F: D8 7C 69 9F 32 69 FA 07 F0 03 01 78

*
* MULTIPLY REAL
* MLR                                                    OP-CO
       DE = 74
*
*        SP -> MAR
MLR:  ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP MULTREAL
0060: 58 62 29 9F 32 69 78 07 F0 03 01 7E

*
* DIVIDE WORD
* DVW                  OP-CODE = 76
*
*        SP -> MAR, 0 -> WREG
DVW:  ALU REG LOW & AQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM WREG & JSB DIVSUB
0061: D8 7C 69 9F 32 69 F9 87 F0 01 01 A5

*        PERFORM 15 DIVIDES
DIV15:  SPF14 TCDIV & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL COEQCI & DATAPATH & DATAPATH & MEMCONT &
       CONTROL & RTB & CNTLB $1F & RPCT DIV15
0062: C0 60 29 1F 30 09 70 FF F8 09 00 62

*        PERFORM LAST DIVIDE (NO BIAS CORRECTION)
*        SP -> MAR, SKIP NEXT INSTRUCTION IF MSR N = 0 (I.E.SI
```

```
          GN RESULT +VE)
        SPF14 TCDIV & AB & OEY & PCUSP & AM2904 SHIFTEN OECT &
        CARRYCTL COEQCI & DATAPATH ,,,,,,LDMAR & MEMCONT &
        CONTROL & RTB & CNTLB $1F & TESTF $2E & CJP QTOD
0063: C0 60 29 9F 32 49 70 FB FA E3 00 65

*         ANSWER SHOULD BE -VE SO -QREG -> DREG
        ALU YBUS COMPLS & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
        CONTROL & JUMP WRTS
0064: 48 62 E9 5F 30 29 78 07 F4 03 01 15

*         QREG -> DREG
QTOD:   ALU YBUS PASS & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
        CONTROL & JUMP WRTS
0065: 48 62 69 5F 30 29 78 07 F0 03 01 15

*
* MODULO WORD
* MOD                                              OP-CODE = 80
*
*         SP -> MAR, 0 -> WREG,
MOD:   ALU REG LOW & AQ & OEY & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM WREG & JSB DIVSUB
0066: D8 7C 69 9F 32 69 F9 87 F0 01 01 A5

*         PERFORM 15 DIVIDES
MOD15:   SPF14 TCDIV & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL COEQCI & DATAPATH & MEMCONT & CONTROL & RTB &
        CNTLB $1F & RPCT MOD15
0067: C0 60 29 1F 30 09 70 FF F8 09 00 67

*         LAST OP, NO BIAS CORRECTION, SP -> MAR, REMAINDER ->
          DREG
        SPF14 TCDC & AB & OEY & PCUSP & AM2904 SHIFTEN OECT &
        CARRYCTL COEQCI & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL & RTB &
        CNTLB $13 & TEST DIRIN POSITIVE & CJP WRTS
0068: C0 70 29 DF 32 69 70 9B FB E3 01 15

*         MODIFY THE REMAINDER
        ALU YBUS ADD & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
        CONTROL & RTB & TEST DIRIN NEGATIVE & CJP *
0069: C0 61 A9 5F 30 29 78 03 F3 F3 00 69

*         WRITE RESULT TO STACK
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JMAP
006A: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* BUILD SET
* BLS                                              OP-CODE = 82
*
```

```
*         SP -> MAR, 1 -> VREG
BLS:  ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM VREG & IMMD 1 & CONT
006B: D2 7B 69 9F 32 68 FB 87 F0 0E 00 01

*         READ VALUE, SP+2 -> SP,YREG, LOAD AM2910 COUNTER WITH
           2
       ALU REG SUBS & DAQ & YOFF & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,PCUY & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM YREG & LDCT 2
006C: C3 79 68 1F 18 7B FA 87 F0 0C 00 02

*         ZREG (VALUE) -> XREG,QREG, EXCEPTION IF < 0
       ALU RQPT PASSR & DAQ & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
       CONTROL ROM & RTB & BRAM XREG & TEST DIRIN SALSB &
       CJPX EXCEPTION
006D: C2 3B 69 0F 30 09 FA 03 F3 33 XX XX

*         XREG (VALUE) - 127 -> YBUS, SET MACHINE STATUS REGIST
           ER
       ALU YBUS SUBR & DAB & OEY & PCUNOP &
       AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM XREG &
       IMMD 127 & TEST MACHINE DLOAD & CONT
006E: C2 60 A9 1F 30 08 FA 04 16 7E 00 7F

*         XREG SHR 1 -> XREG, 3 TIMES (I.E. DIVIDE BY 8)
SHIFT:  ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM &
       CNTLB 0 & RTB &
       BRAM XREG & RPCT SHIFT
006F: C0 0A 29 1F 30 09 F2 07 F0 09 00 6F

*         XREG + YREG (SP) -> MAR, EXCEPTION IF MSR > 0
       ALU YBUS ADD & AB & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR & MEMCONT REQB &
       CONTROL ROM & RTB & ARAM YREG & BRAM XREG &
       TEST MACHINE UAGRB & CJPX RANGEEXP
0070: C0 61 AB 9F 30 29 FA 2B F2 C3 XX XX

*         READ SET BYTE, QREG.$0007 -> WREG,TREG (VALUE MOD 8),
           SET MSR
       ALU REG AND & DAQ & OEY & PCUNOP &
       AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL &
       DATAPATH ,,LDTREG & MEMCONT REQB MREQ ,,MBYTE &
       CONTROL ROM & RTB & BRAM WREG & IMMD $0007 &
       TEST MACHINE DLOAD & CONT
0071: C2 7E 79 1F 30 38 F9 84 12 7E 00 07

*         VREG (1) -> QREG, SKIP NEXT INSTRUCTION IF MSR = 0
       ALU QPT PASSR & AQ & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM &
       ARAM VREG & TEST MACHINE UAEQB & CJP WTBACK
0072: 40 33 69 1F 30 09 F8 3B F2 53 00 74

*         DECREMENT TREG, QREG SHL 1 -> QREG, LOOP BACK IF TREG
```

```
          > 0
SHFTST:  ALU LUQ SUBS & DAB & OEY & PCUNOP &
       AM2904 SHIFTEN OECT & CARRYCTL COEQ1 &
       DATAPATH ,ENTREG LDTREG & MEMCONT & CONTROL ROM &
       CNTLB 0 & RTB &
       BRAM VREG & TEST DIRIN SAGRB & CJP SHFTST
0073: C2 69 19 1F 30 09 F3 83 F7 03 00 73

*        QREG OR ZREG (ORIGINAL SET BYTE) -> DREG
WTBACK:  ALU YBUS OR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD ,ENZ0 & MEMCONT REQB &
       CONTROL & CONT
0074: 42 67 E9 4F 30 29 78 07 F0 0E 00 00

*        WRITE NEW SET BYTE, PC -> MAR
       ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & JUMP RNI
0075: 48 62 29 9F 30 3D 78 07 F0 03 01 00

*
* TEST IN SET
* TIS                                          OP-CODE = 84
*
*        SP+16 -> MAR,R6, 1 -> VREG
TIS:   ALU LUR LOW & DAQ & OEY & PCU ,,PCUDA A1 B6 &
       AM2904 SHIFTEN ,,,,,,CEU & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & CNTLB $11 &
       BRAM VREG &
       TESTF $16 & IMMD 16 & CONT
0076: D2 4C 69 9F 53 A8 F3 8F E1 6E 00 10

*        READ VALUE, SP -> YREG, LOAD AM2910 COUNTER WITH 2
*        RESET OVR BIT ON MACHINE STATUS REGISTER
       ALU REG PASS & AB & YOFF & PCUSP &
       AM2904 ,,,,,EOVR CEM & CARRYCTL & DATAPATH ,,,,,PCUY &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM YREG &
       TESTF $02 & LDCT 2
0077: C1 7A 28 1F 32 7B FA 87 90 2C 00 02

*        ZREG (VALUE) -> XREG + QREG, EXCEPTION IF < 0, R6 (SP
         +16) -> SP
       ALU RQPT PASSR & DAQ & OEY & PCU ,,PCUZA A6 B1 &
       AM2904 ,OECT & CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT & CONTROL ROM & RTB & BRAM XREG &
       TEST DIRIN SALSB & CJPX RANGEEXP
0078: C2 3B 69 0F 4C 49 FA 03 F3 33 XX XX

*        XREG (VALUE) - 127 -> YBUS, SET MSR
       ALU YBUS SUBR & DAB & OEY & PCUNOP &
       AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM XREG &
       TEST MACHINE DLOAD & IMMD 127 & PUSH 127
0079: C2 60 A9 1F 30 08 FA 00 16 74 00 7F

*        XREG SHR 3 -> XREG (DIVIDE BY 8)
SHR3:  ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
```

```
        CARRYCTL & DATAPATH ,,,,,,,,LDI & MEMCONT &
        CONTROL ROM & CNTLB $00 & RTB &
        BRAM XREG &
        IREG VREG VREG & RFCT
007A: C0 0A 29 3F 30 09 F2 07 F0 08 00 77

*         XREG + YREG (SP) -> MAR, EXCEPTION IF MSR > 0
        ALU YBUS ADD & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & ARAM XREG & BRAM YREG &
        TEST MACHINE SAGRB & CJPX RANGEEXP
007B: C0 61 AB 9F 30 29 FA A3 F2 03 XX XX

*         READ SET BYTE, QREG (VALUE).$0007 -> WREG,TREG (VALUE
           MOD 8), SET MSR
        ALU REG AND & DAQ & OEY & PCUNOP &
        AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL &
        DATAPATH ,,LDTREG & MEMCONT REQB MREQ ,,MBYTE &
        CONTROL ROM & RTB & BRAM WREG & TEST MACHINE DLOAD &
        IMMD $0007 & CONT
007C: C2 7E 79 1F 30 38 F9 84 12 7E 00 07

*         ZREG (SET BYTE) -> QREG, PC -> MAR, SKIP NEXT INSTRUC
          TION IF MSR = 0
        ALU QPT PASSR & DAQ & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL & TEST MACHINE UAEQB &
        CJP WRT
007D: 42 33 69 8F 30 29 78 03 F2 53 00 7F

*         QREG SHR 1 -> QREG, DECREMENT TREG + LOOP BACK IF > 0

DTREG:   ALU LDQP SUBS & DAB & OEY & PCUNOP &
        AM2904 SHIFTEN OECT & CARRYCTL COEQ1 &
        DATAPATH ,ENTREG LDTREG & MEMCONT REQB &
        CONTROL & CNTLB $00 & RTB & TEST DIRIN SAGRB &
        CJP DTREG
007E: C2 29 19 1F 30 29 70 03 F7 03 00 7E

*         QREG.VREG(1) -> DREG, SP -> MAR, READ INSTR N+2
WRT:   ALU YBUS AND & AQ & OEY & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB MREQ &
        CONTROL ROM & ARAM VREG & JUMP WRTS
007F: 48 66 69 DF 32 7B F8 3F F0 03 01 15

*
* COMPARE WORD LESS THAN
* CLT                                              OP-CODE = 86
*
*         SP -> MAR, 1 -> DREG
CLT:   ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & CNTLB $11 & RTB &
        BRAM WREG & JSB RW1W2
0080: D8 4C 29 DF 32 69 F1 8F F0 01 01 13

*         READ INSTR N+2, ZREG (WORD2) - WORD1 -> YBUS, SP -> M
          AR
```

```
*         SKIP NEXT INSTRUCTION IF <
        ALU YBUS SUBS & DAB & OEY & PCUSP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN SALSB & CJP WRTS
0081: C2 61 29 8F 32 7B F9 83 F7 33 01 15

*          0 -> DREG
        ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL &
        JUMP WRTS
0082: 48 64 29 5F 30 29 78 07 F0 03 01 15

*
*
* COMPARE WORD EQUAL TO
* CEQ                                                OP-CODE = 88
*
*         SP -> MAR, 1 -> DREG
CEQ:   ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & CNTLB $11 & RTB &
        BRAM WREG & JSB RW1W2
0083: D8 4C 29 DF 32 69 F1 8F F0 01 01 13

*         READ INSTR N+2, ZREG (WORD2) - WORD1 -> YBUS, SP -> M
          AR
*         SKIP NEXT INSTRUCTION IF =
        ALU YBUS SUBS & DAB & OEY & PCUSP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN SAEQB & CJP WRTS
0084: C2 61 29 8F 32 7B F9 83 F7 53 01 15

*          0 -> DREG
        ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL &
        JUMP WRTS
0085: 48 64 29 5F 30 29 78 07 F0 03 01 15

*
* COMPARE WORD GREATER THAN
* CGT                                                OP-CODE = 90
*
*         SP -> MAR, 1 -> DREG
CGT:   ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & CNTLB $11 & RTB &
        BRAM WREG & JSB RW1W2
0086: D8 4C 29 DF 32 69 F1 8F F0 01 01 13

*         READ INSTR N+2, ZREG (WORD2) - WORD1 -> YBUS, SP -> M
          AR
*         SKIP NEXT INSTRUCTION IF >
        ALU YBUS SUBS & DAB & OEY & PCUSP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN SAGRB & CJP WRTS
```

```
0087: C2 61 29 8F 32 7B F9 83 F7 03 01 15

*         0 -> DREG
       ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL &
       JUMP WRTS
0088: 48 64 29 5F 30 29 78 07 F0 03 01 15

*
* COMPARE WORD GREATER THAN OR EQUAL TO
* CGE                                              OP-CODE = 92
*
*        SP -> MAR, 1 -> DREG
CGE:  ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
       MEMCONT REQB & CONTROL ROM & CNTLB $11 & RTB &
       BRAM WREG & JSB RW1W2
0089: D8 4C 29 DF 32 69 F1 8F F0 01 01 13

*        READ INSTR N+2, ZREG (WORD2) - WORD1 -> YBUS, SP -> M
         AR
*        SKIP NEXT INSTRUCTION IF >=
       ALU YBUS SUBS & DAB & OEY & PCUSP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       TEST DIRIN SAGEB & CJP WRTS
008A: C2 61 29 8F 32 7B F9 83 F7 23 01 15

*         0 -> DREG
       ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL &
       JUMP WRTS
008B: 48 64 29 5F 30 29 78 07 F0 03 01 15

*
* COMPARE WORD NOT EQUAL TO
* CNE                                              OP-CODE = 94
*
*        SP -> MAR, 1 -> DREG
CNE:  ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
       MEMCONT REQB & CONTROL ROM & CNTLB $11 & RTB &
       BRAM WREG & JSB RW1W2
008C: D8 4C 29 DF 32 69 F1 8F F0 01 01 13

*        READ INSTR N+2, ZREG (WORD2) - WORD1 -> YBUS, SP -> M
         AR
*        SKIP NEXT INSTRUCTION IF NOT =
       ALU YBUS SUBS & DAB & OEY & PCUSP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       TEST DIRIN SANEB & CJP WRTS
008D: C2 61 29 8F 32 7B F9 83 F7 43 01 15

*         0 -> DREG
       ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL &
       JUMP WRTS
```

```
008E: 48 64 29 5F 30 29 78 07 F0 03 01 15

* COMPARE WORD LESS THAN OR EQUAL TO
* CLE                                              OP-CODE = 96
*
*       SP -> MAR, 1 -> DREG
CLE:  ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
      CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
      MEMCONT REQB & CONTROL ROM & CNTLB $11 & RTB &
      BRAM WREG & JSB RW1W2
008F: D8 4C 29 DF 32 69 F1 8F F0 01 01 13

*       READ INSTR N+2, ZREG (WORD2) - WORD1 -> YBUS, SP -> M
        AR
*       SKIP NEXT INSTRUCTION IF <=
      ALU YBUS SUBS & DAB & OEY & PCUSP & AM2904 ,OECT &
      CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
      TEST DIRIN SALEB & CJP WRTS
0090: C2 61 29 8F 32 7B F9 83 F7 13 01 15

*        0 -> DREG
      ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
      DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL &
      JUMP WRTS
0091: 48 64 29 5F 30 29 78 07 F0 03 01 15

*
* COMPARE REAL LESS THAN
* CRL                                    OP-CODE = 98
*
*       SP -> R6,MAR, 0 -> DREG,WREG (ASSUME FALSE)
CRLT:  ALU LUR LOW & AB & OEY & PCU ,,PCUZA A1 B6 &
      AM2904 SHIFTEN & CARRYCTL &
      DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
      CONTROL ROM & RTB & BRAM WREG & CNTLB $10 &
      JUMP CRLESS
0092: D8 4C 29 DF 43 A9 F1 87 F0 03 01 AC

*
* COMPARE REAL EQUAL TO
* CRE                                              OP-CODE = 100
*
*       SP -> R6,MAR, 1 -> DREG,YREG
CRE:  ALU LUR LOW & DAQ & OEY & PCU ,,PCUZA A1 B6 &
      AM2904 SHIFTEN & CARRYCTL &
      DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
      CONTROL ROM & RTB & CNTLB $11 &
      BRAM YREG & JUMP CMPREAL
0093: DA 4C 69 DF 43 A9 F2 8F F0 03 01 D4

*
* COMPARE REAL GREATER THAN
* CRG                                    OP-CODE = 102
*
*       SP -> R6,MAR, 0 -> DREG,XREG (ASSUME FALSE)
CRG:  ALU LUR LOW & AB & OEY & PCU ,,PCUZA A1 B6 &
      AM2904 SHIFTEN & CARRYCTL &
```

```
        DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & CNTLB $10 &
        JUMP CRGREATER
0094: D8 4C 29 DF 43 A9 F1 87 F0 03 01 C0

*
* COMPARE REAL GREATER THAN OR EQUAL TO
* CRGE                                    OP-CODE = 104
*
*         SP -> R6,MAR, 1 -> DREG,XREG (ASSUME FALSE)
CRGE:   ALU LUR LOW & AB & OEY & PCU ,,PCUZA A1 B6 &
        AM2904 SHIFTEN & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & CNTLB $11 &
        JUMP CRLESS
0095: D8 4C 29 DF 43 A9 F1 8F F0 03 01 AC

*
* COMPARE REAL NOT EQUAL TO
* CRNE                                          OP-CODE = 106
*
*         SP -> R6,MAR, 0 -> DREG,YREG
CRNE:   ALU LUR LOW & DAQ & OEY & PCU ,,PCUZA A1 B6 &
        AM2904 SHIFTEN & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & RTB & CNTLB $10 &
        BRAM YREG & JUMP CMPREAL
0096: DA 4C 69 DF 43 A9 F2 87 F0 03 01 D4

*
* COMPARE REAL LESS THAN OR EQUAL TO
* CRLE                                    OP-CODE = 108
*
*         SP -> R6,MAR, 1 -> DREG,WREG (ASSUME FALSE)
CRLE:   ALU LUR LOW & AB & OEY & PCU ,,PCUZA A1 B6 &
        AM2904 SHIFTEN & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & CNTLB $11 &
        JUMP CRGREATER
0097: D8 4C 29 DF 43 A9 F1 8F F0 03 01 C0

*
* COMPARE SET EQUAL TO
* EQS                                           OP-CODE = 110
*
*         SP -> R6,MAR, 1 -> DREG,YREG
EQS:    ALU LUR LOW & DAQ & OEY & PCU ,,PCUZA A1 B6 &
        AM2904 SHIFTEN & CARRYCTL &
        DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & RTB & CNTLB $11 &
        BRAM YREG & JUMP CMPSET
0098: DA 4C 69 DF 43 A9 F2 8F F0 03 01 DC

*
* COMPARE SET GREATER THAN OR EQUAL TO
* SGE                                           OP-CODE = 112
*
*         SP -> R6,MAR, 1 -> DREG
```

```
SGE:  ALU LUR LOW & AQ & OEY & PCU ,,PCUZA A1 B6 &
       AM2904 SHIFTEN & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
       CONTROL ROM & RTB &
       BRAM WREG & CNTLB $11 & JUMP SETGE
0099: D8 4C 69 DF 43 A9 F1 8F F0 03 01 E4

*
* COMPARE SET NOT EQUAL TO
* SNE                                          OP-CODE = 114
*
*        SP -> R6,MAR, 0 -> DREG,YREG
SNE:  ALU LUR LOW & DAQ & OEY & PCU ,,PCUZA A1 B6 &
       AM2904 SHIFTEN & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
       CONTROL ROM & RTB & CNTLB $10 &
       BRAM YREG & JUMP CMPSET
009A: DA 4C 69 DF 43 A9 F2 87 F0 03 01 DC

*
* COMPARE SET LESS THAN OR EQUAL TO
* SLE                                          OP-CODE = 116
*
*        SP -> R6,MAR, 1 -> DREG
SLE:  ALU LUR LOW & AQ & OEY & PCU ,,PCUZA A1 B6 &
       AM2904 SHIFTEN & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR LDD & MEMCONT REQB &
       CONTROL ROM & RTB &
       BRAM WREG & CNTLB $11 & JUMP SETLE
009B: D8 4C 69 DF 43 A9 F1 8F F0 03 01 EA

*
* COMPARE STRUCTURE LESS THAN
* CSL LENGTHWORDS                                    OP-CODE
        = 118
*
*        SP -> MAR, READ INSTR N+1, -ZREG ("LENGTHWORDS")*2 ->
         WREG
CSLT:  ALU AUR COMPLR & DAQ & OEY & PCUSP & AM2904 SHIFTEN &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & CNTLB $10 & RTB &
       BRAM WREG & JSB RSADA
009C: CA 43 E9 8F 32 7B F1 87 F4 01 01 F0

*        READ FIRST SOURCE BYTE, ZREG (DESTINATION ADDRESS) ->
         R7,MAR
       ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ ,,MBYTE & CONTROL & JUMP CMPLSTR
009D: 48 62 29 8F 7F F9 78 07 F0 03 01 F2

*
* COMPARE STRUCTURE EQUAL TO
* CSE LENGTHWORDS
         OP-CODE = 120
*
*        SP -> MAR, -ZREG ("LENGTHWORDS") -> WREG, READ INSTR
         N+1
```

```
CSE:  ALU REG COMPLR & DAQ & OEY & PCUSP & AM2904 &
      CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
      CONT
009E: C2 7B E9 8F 32 7B F9 87 F4 0E 00 00

*       READ SOURCE ADDRESS, SP+2 -> MAR, 1 -> DREG,YREG
      ALU LUR LOW & AB & OEY & PCUPOP & AM2904 SHIFTEN &
      CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
      MEMCONT REQB MREQ & CONTROL ROM & CNTLB $11 & RTB &
      BRAM YREG & JUMP CESTRUCT
009F: D8 4C 29 DF 18 7B F2 8F F0 03 02 0E

*
* COMPARE STRUCTURE GREATER THAN
* CSG LENGTHWORDS                                    OP-CODE
        = 122
*
*       SP -> MAR, READ INSTR N+1, -ZREG ("LENGTHWORDS")*2 ->
         WREG
CSG:  ALU AUR COMPLR & DAQ & OEY & PCUSP & AM2904 SHIFTEN &
      CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & CNTLB $10 & RTB &
      BRAM WREG & JSB RSADA
00A0: CA 43 E9 8F 32 7B F1 87 F4 01 01 F0

*       READ FIRST SOURCE BYTE, ZREG (DESTINATION ADDRESS) ->
         R7,MAR
      ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB MREQ ,,MBYTE & CONTROL & JUMP CMPLSTR1
00A1: 48 62 29 8F 7F F9 78 07 F0 03 01 F9

*
* COMPARE STRUCTURE GREATER THAN OR EQUAL TO
* CSGE LENGTHWORDS                                   OP-CODE
        = 124
*
*       SP -> MAR, READ INSTR N+1, -ZREG ("LENGTHWORDS")*2 ->
         WREG
CSGE:  ALU AUR COMPLR & DAQ & OEY & PCUSP & AM2904 SHIFTEN &
      CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & CNTLB $10 & RTB &
      BRAM WREG & JSB RSADA
00A2: CA 43 E9 8F 32 7B F1 87 F4 01 01 F0

*       READ FIRST SOURCE BYTE, ZREG (DESTINATION ADDRESS) ->
         R7,MAR
      ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB MREQ ,,MBYTE & CONTROL & JUMP CMPLSTR2
00A3: 48 62 29 8F 7F F9 78 07 F0 03 02 00

*
* COMPARE STRUCTURE NOT EQUAL TO
* CSNE LENGTHWORDS
         OP-CODE = 126
*
```

```
*         SP -> MAR, -ZREG ("LENGTHWORDS") -> WREG, READ INSTR
          N+1
CSNE:   ALU REG COMPLR & DAQ & OEY & PCUSP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
00A4: C2 7B E9 8F 32 7B F9 87 F4 0E 00 00

*         READ SOURCE ADDRESS, SP+2 -> MAR, 0 -> DREG,YREG
        ALU LUR LOW & AB & OEY & PCUPOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
        MEMCONT REQB MREQ & CONTROL ROM & CNTLB $10 & RTB &
        BRAM YREG & JUMP CESTRUCT
00A5: D8 4C 29 DF 18 7B F2 87 F0 03 02 0E

*
* COMPARE STRUCTURE LESS THAN OR EQUAL TO
* CSLE LENGTHWORDS                                        OP-CODE
          = 128
*
*         SP -> MAR, READ INSTR N+1, -ZREG ("LENGTHWORDS")*2 ->
           WREG
CSLE:   ALU AUR COMPLR & DAQ & OEY & PCUSP & AM2904 SHIFTEN &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & CNTLB $10 & RTB &
        BRAM WREG & JSB RSADA
00A6: CA 43 E9 8F 32 7B F1 87 F4 01 01 F0

*         READ FIRST SOURCE BYTE, ZREG (DESTINATION ADDRESS) ->
           R7,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL & JUMP CMPLSTR3
00A7: 48 62 29 8F 7F F9 78 07 F0 03 02 07

*
* FUNCTION VALUE
* FNV KIND                                          OP-CODE = 130
*
*         ZREG ("ZIND") -> WREG, SKIP 2 IF > 0, READ INSTR N+1,
           PC+2 -> MAR
FNV:    ALU REG PASSR & DAQ & OEY & PCUNEXT & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN SAGRB & CJP FUNCVL
00A8: C2 7B 69 8F 18 3B F9 83 F3 03 02 17

*         ORDINARY WORD.
*         0 -> DREG, SP - 2 -> MAR, READ INSTR N+2
        ALU YBUS LOW & AB & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
        MEMCONT REQB MREQ & CONTROL & CONT
00A9: 50 64 29 DF 98 7B 78 07 F0 0E 00 00

*         WRITE ZERO TO STORE, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JMAP
```

```
00AA: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* JUMP RELATIVE
* JMP DISPLACEMENT            CODE = 132
*
*        SUBTRACT 2 FROM PC
JMP:  ALU YBUS PASS & PCU ,1 PCUAB B4 A0 & AM2904 &
       CARRYCTL & DATAPATH & CONTROL & MEMCONT & CONT
00AB: 40 62 29 1F 98 09 78 07 F0 0E 00 00

*        ADD DISPLACEMENT TO PC
       ALU YBUS PASS & PCU ,,PCUDA A0 B0 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL & JUMP START1
00AC: 48 62 29 8F 50 29 78 07 F0 03 00 FD

*
* FALSE JUMP
* FSP DISTANCE                                  OP-CODE = 134
*
*        PHREG+CONTATR -> MAR
FSP:  ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD CONTATR & CONT
00AD: 40 62 29 9E 56 28 78 07 F0 0E 00 14

*        READ CONTINUE ATTRIBUTE, ZREG ("DISTANCE") -> WREG
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP FALSEJUMP
00AE: CA 7B 69 0F 30 3B F9 87 F0 03 02 1F

*
* CASE JUMP
* CSP MIN, MAX-MIN, DISTANCES                                OP-CODE
         = 136
*
*        SP -> MAR, READ "MAX-MIN", ZREG ("MIN") -> WREG
CSP:  ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM WREG & JUMP CASEJMP
00AF: CA 7B 69 8F 32 7B F9 87 F0 03 02 26

*
* INITIALISE VARIABLES
* IVR LENGTHWORDS                                            OP-CODE
         = 138
*
*        -ZREG ("LENGTHWORDS") -> WREG, PC+2 -> MAR, READ INST
         R N+1
IVR:  ALU REG COMPLR & DAQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP INITVR
00B0: CA 7B E9 8F 18 3B F9 87 F4 03 02 2B
```

```
*
* CALL PROCEDURE
* CALL DISTANCE                                                    OP
        -CODE = 140
*
*       PC -> DREG, 2 -> QREG
CALL:  ALU QPT PASSR & DAQ & YOFF & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,PCUY ,LDD & MEMCONT &
       CONTROL & IMMD 2 & CONT
00B1: 43 33 68 5F 30 08 78 07 F0 0E 00 02

*       SP - 2 -> MAR, ZREG ("DISTANCE") - QREG (2) -> TREG
       ALU YBUS SUBS & DAQ & OEY & PCUPUSH & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,LDTREG ,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL & CONT
00B2: 42 61 79 8F 98 69 78 07 F4 0E 00 00

*       WRITE RETURN ADDRESS TO STACK, PC+TREG -> MAR
       ALU YBUS PASS & AB & PCU ,,PCUDA A0 B0 & AM2904 &
       CARRYCTL & DATAPATH ,ENTREG ,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP START1
00B3: 48 62 09 9F 50 3F 78 07 F0 03 00 FD

*
* SYSTEM CALL
* SCL ENTRY                                  OP-CODE = 142
*
*       G+2 -> MAR, REST OF CODE IS IN CONCURRENT P-CODE FILE

SCL:   SPF 14 INCTWO & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM GREG &
       XJUMP SYSCALL
00B4: C8 20 2B 9F 30 29 F8 87 F4 03 XX XX

*
* ENTER PROCEDURE
* ENT STACKLENGTH, POPLENGTH, LINE, VARLENGTH                       O
        P-CODE = 144
*
*       HREG + ZREG ("STACKLENGTH") -> QREG, SP -> TREG, READ
         "POPLENGTH"
ENT:   ALU QPT ADD & DAB & YOFF & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ,,LDTREG ,,PCUY ,,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM HREG & JSB ENTER
00B5: CB 31 B8 0F 32 7B F9 07 F0 01 02 2E

*       READ INSTR N+1, SP - ZREG ("VARLENGTH") -> SP, VREG (
        PC) + QREG (2) ->
       ALU YBUS ADD & AQ & OEY & PCU ,SUB PCUDA A1 B1 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & ARAM VREG & CONT
00B6: 40 61 EB 8F D2 7B F8 3F F0 0E 00 00

*       READ INSTR N+2, PC+4 -> MAR, BREG - QREG (2) -> BREG
       ALU REG SUBS & AQ & OEY & PCU ,,PCUAB A5 B0 & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & ARAM BREG &
```

```
       BRAM BREG & JMAP
00B7: D8 79 69 9F 1A 3B F8 07 F4 02 00 00

*
* EXIT PROCEDURE
* EXT                                          OP-CODE = 146
* ALSO "ENDCLASS" AND "EXITCLASS" ARE IDENTICAL.
*
*        BREG + 2 -> MAR,BREG,TREG
ENC      EQU *
EXC      EQU *
EXT:   SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 & CARRYCTL COEQ1 &
       DATAPATH ZZI ,LDTREG ,,YMAR LDMAR & MEMCONT REQB &
       CONTROL ROM & RTB & BRAM BREG & JUMP EXIT
00B8: D8 20 3B 9F 30 29 F8 07 F4 03 02 37

*
* ENTER PASCAL PROGRAM
* EPP POPLENGTH,LINE,STACKLENGTH,VARLENGTH           CODE = 14
        8
*
*        PHREG+JOBATR -> MAR, 1 -> DREG
EPP:  ALU LUR LOW & AB & OEY & PCU QEU ,PCUDA PHREG &
       AM2904 SHIFTEN & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
       CNTLB $11 & IMMD JOBATR & CONT
00B9: C0 4C 29 DE 56 28 F1 8F F0 0E 00 12

*        WRITE NEW MODE = USER TO PROCESS HEAD, PC -> MAR.
       ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & CONT
00BA: 40 62 29 9F 30 3F 78 07 F0 0E 00 00

*        READ "LINE", ZREG("POPLENGTH") -> R6 OF PCU, G -> DRE
         G
       ALU YBUS PASSR & AQ & OEY & PCU 1 0 PCUDZ A0 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,,LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & CNTLB GREG &
       JUMP SCODE
00BB: C8 63 69 4F 71 BB F8 0F F0 03 02 3C

*
* EXIT PASCAL PROGRAM
* EXP                                      OP-CODE = 150
*
*        PHREG+CONTATR -> MAR
EXP:  ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD CONTATR & CONT
00BC: 40 62 29 9E 56 28 78 07 F0 0E 00 14

*        READ CONTINUE ATTRIBUTE
*        REST OF CODE IS IN CONCURRENT P-CODE FILE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ & CONTROL & XJUMP EXTPROG
00BD: 48 62 29 1F 30 3B 78 07 F0 03 XX XX
```

```
*
*         BEGIN CLASS
*         BGC STACKLENGTH, POPLENGTH, LINE, VARLENGTH
             OP-CODE = 152
*         ENTER CLASS (ETC) IS ALSO IDENTICAL
*
*         HREG + ZREG ("STACKLENGTH") -> QREG, SP -> TREG
*         READ "POPLENGTH", JUMP TO ENTER SUBROUTINE
ETC       EQU *
BGC:    ALU QPT ADD & DAB & YOFF & PCUSP & AM2904 &
        CARRYCTL & DATAPATH ,,LDTREG ,,PCUY ,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM HREG &
        JSB ENTER
00BE: CB 31 B8 0F 32 7B F9 07 F0 01 02 2E

*         READ INSTR N+1, BREG -> MAR, SP-ZREG ("VARLENGTH") ->
           SP
*         REST OF CODE IS IN CONCURRENT P-CODE FILE
        ALU YBUS PASS & AB & OEY & PCU ,SUB PCUDA A1 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & CONTROL ROM &
        RTB & BRAM BREG & XJUMP BGNCLASS
00BF: C8 62 2B 8F D2 7B F8 07 F0 03 XX XX

*
*         BEGIN MONITOR
*         BGM STACKLENGTH, POPLENGTH, LINE, VARLENGTH
             OP-CODE = 160
*
*         HREG + ZREG ("STACKLENGTH") -> QREG, SP -> TREG
*         READ "POPLENGTH", JUMP TO ENTER SUBROUTINE
BGM:    ALU QPT ADD & DAB & YOFF & PCUSP & AM2904 & CARRYCTL &
         DATAPATH ,,LDTREG ,,PCUY ,,,ENZ0 & MEMCONT REQB MREQ &
         CONTROL ROM & RTB & BRAM HREG & JSB ENTER
00C0: CB 31 B8 0F 32 7B F9 07 F0 01 02 2E

*         READ INSTR N+1, BREG -> MAR, SP-ZREG ("VARLENGTH") ->
           SP
*         REST OF CODE IS IN CONCURRENT P-CODE FILE
        ALU YBUS PASS & AB & OEY & PCU ,SUB PCUDA A1 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
        XJUMP BGNMON
00C1: C8 62 2B 8F D2 7B F8 07 F0 03 XX XX

*
* END MONITOR
* ENM                                                          OP
          -CODE = 162
* EXIT MONITOR (EXM) IS IDENTICAL.                             OP
          -CODE = 166
*
*         GREG -> TREG, PRIORITY 0 QUEUE ADDRESS - Q OF PCU
EXM     EQU *
ENM:    ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDZ &
        AM2904 & CARRYCTL & DATAPATH ,,LDTREG & MEMCONT &
        CONTROL ROM & ARAM GREG & IMMD QSTART & CONT
00C2: 40 63 79 1E 70 08 F8 0F F0 0E 00 00
```

```
*         BREG + 2 -> MAR,BREG,TREG ,TREG -> R7 OF PCU
*         REST OF CODE IS IN CONCURRENT P-CODE FILE.
        SPF14 INCTWO & AB & OEY & PCU ,,PCUDZ A7 B7 & AM2904 &
        CARRYCTL COEQ1 &
        DATAPATH ,ENTREG LDTREG ,,YMAR LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM BREG & XJUMP ENDMON
00C3: C8 20 1B 9F 7F E9 F8 07 F4 03 XX XX

*
*         ENTER MONITOR
*         ETM STACKLENGTH, POPLENGTH, LINE, VARLENGTH
             OP-CODE = 164
*
*         HREG + ZREG ("STACKLENGTH") -> QREG, SP -> TREG
*         READ "POPLENGTH", JUMP TO ENTER SUBROUTINE
ETM:   ALU QPT ADD & DAB & YOFF & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,LDTREG ,,PCUY ,,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM HREG & JSB ENTER
00C4: CB 31 B8 0F 32 7B F9 07 F0 01 02 2E

*         READ INSTR N+1, BREG -> MAR, SP-ZREG ("VARLENGTH") ->
           SP
*         REST OF CODE IS IN CONCURRENT P-CODE FILE
        ALU YBUS PASS & AB & OEY & PCU ,SUB PCUDA A1 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & CONTROL ROM &
        RTB & BRAM BREG & XJUMP ENTMON
00C5: C8 62 2B 8F D2 7B F8 07 F0 03 XX XX

*
* END PROCESS
* ENP                    CODE = 170
* TEMPORARY TEST CODE
*
*         RESCHEDULE
ENP:   ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & CONTROL & XJUMP RESCHED
00C6: 48 62 29 1F 30 01 78 07 F0 03 XX XX

*
*         ENTER PREFIX PROCEDURE
*         ETP STACKLENGTH ,POPLENGTH ,LINE ,VARLENGTH
             OP-CODE = 172
*
*         HREG + ZREG ("STACKLENGTH") -> QREG, READ "POPLENGTH"

*         SP -> TREG, JUMP TO "ENTER" SUBROUTINE
ETP:   ALU QPT ADD & DAB & YOFF & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,LDTREG ,,PCUY ,,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM HREG & JSB ENTER
00C7: CB 31 B8 0F 32 7B F9 07 F0 01 02 2E

*         READ INSTR N+1, SP-ZREG ("VARLENGTH") -> SP, BREG-2 -
           > BREG
*         REST OF THE CODE IS IN CONCURRENT P-CODE FILE
        ALU REG SUBS & AQ & OEY & PCU ,SUB PCUDA A1 B1 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
```

```
       MEMCONT REQB MREQ & CONTROL ROM & RTB & ARAM BREG &
       BRAM BREG & XJUMP ENTERPROC
00C8: C8 79 69 0F D2 7B F8 07 F4 03 XX XX

*
*        EXIT PREFIX PROCEDURE
*        EXPC                                                OP-C
         ODE = 174
*
*        SET MODE TO USER
*        1 -> DREG, PHREG + JOBATR -> MAR
EXPC:  ALU LUR LOW & AQ & OEY & PCU QEU ,PCUDA PHREG &
       AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
       CNTLB $11 & IMMD JOBATR & CONT
00C9: C0 4C 69 DE 56 28 F1 8F F0 0E 00 12

*        WRITE NEW MODE TO PROCESS HEAD, PC -> MAR, JUMP TO "E
         XT"
       ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JUMP EXT
00CA: 48 62 29 9F 30 3F 78 07 F0 03 00 B8

*
* POP STACK
* POP LENGTH                                        OP-CODE = 176

*
*        READ INSTR N+1, ZREG ("LENGTH") + SP -> SP
POP:   ALU YBUS PASS & AB & PCU ,,PCUDA A1 B1 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
00CB: 40 62 29 0F 52 7B 78 07 F0 0E 00 00

*        PC+2 -> MAR
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP RNI
00CC: 58 62 29 9F 18 29 78 07 F0 03 01 00

*
* NEWLINE
* NLN NUMBER                                 OP-CODE = 178
* BEGIN PROCESS (BGP) IS IDENTICAL.
*
*        READ INSTR N+1, ZREG ("NUMBER") -> DREG, PC+2 -> MAR
BGP    EQU *
NLN:   ALU YBUS PASSR & DAQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
00CD: 42 63 69 CF 18 3B 78 07 F0 0E 00 00

*        READ INSTR N+2, BREG -> MAR
       ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ZZI ,,,,YMAR LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & ARAM BREG & CONT
00CE: 50 63 6B 9F 30 3B F8 07 F0 0E 00 00
```

```
*         WRITE "NUMBER", PC+2 -> MAR, JUMP MAP
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JMAP
00CF: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* INCREMENT WORD
* INC                                          OP-CODE = 180
*
*         SP -> MAR, 1 -> WREG
INC:  ALU AUR LOW & AQ & OEY & PCUSP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & CNTLB $11 & RTB &
        BRAM WREG & JUMP INCDEC
00D0: D8 44 69 9F 32 69 F1 8F F0 03 02 47

*
* DECREMENT WORD
* DEC                                          OP-CODE = 182
*
*         SP -> MAR, -1 -> WREG
DEC:  ALU AUR HIGH & AQ & OEY & PCUSP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & CNTLB $11 & RTB &
        BRAM WREG & JUMP INCDEC
00D1: D8 40 69 9F 32 69 F1 8F F0 03 02 47

*
* INITIALISE CLASS
* ICL PARAMLENGTH, DISTANCE                OP-CODE = 184
* INITIALISE MONITOR (IMN) IS IDENTICAL.
*
*         SP + ZREG ("PARAMLENGTH") -> MAR (BUT NOT SP)
*         READ "DISTANCE", -ZREG ("PARAMLENGTH") -> WREG, SET M
          SR
*         REST OF CODE IS IN CONCURRENT P-CODE FILE
IMN     EQU *
ICL:    ALU REG COMPLR & DAQ & OEY &
        PCU QEU ,PCUDA A1 B1 & AM2904 ,,EZ ,,,CEM &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        TEST MACHINE DLOAD & XJUMP INTCLASS
00D2: CA 7B E9 8E 52 7B F9 85 D6 73 XX XX

*
* INITIALISE PROCESS
* IPC   PARAMLENGTH,VARLENGTH,STACKLENGTH,DISTANCE
              OP-CODE = 188
*
*         LAST HEAP ADDRESS -> R6,MAR, 1 -> YREG
IPC:  ALU LUR LOW & AQ & OEY & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM YREG & CNTLB $11 & IMMD LASTHEAP & CONT
00D3: C0 4C 69 9F 7D A8 FA 8F F0 0E 00 32

*         READ LAST HEAP VALUE, -ZREG ("PARAMLENGTH") -> WREG,Q
```

```
          REG
*         ZREG ("PARAMLENGTH") + SP -> SP
*         REAST OF CODE IS IN CONCURRENT P-CODE FILE.
        ALU RQPT COMPLR & DAQ & OEY & PCU ,,PCUDA A1 B1 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        XJUMP PROCINIT
00D4: CA 3B E9 0F 52 7B F9 87 F4 03 XX XX

*
* PUSH LABEL
* PLB DISTANCE                                          OP-CODE = 1
          90
*
*         ZREG ("DISTANCE") -> TREG, PC-2 -> Q OF PCU, READ INS
          TR N+1
*         REST OF CODE IS IN CONCURRENT P-CODE FILE.
PLB:  ALU YBUS PASSR & DAQ & OEY & PCU QEU SUB PCUAB A4 B0 &
        AM2904 & CARRYCTL & DATAPATH ,,LDTREG ,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & XJUMP PUSHLAB
00D5: 4A 63 79 0E 98 3B 78 07 F0 03 XX XX

*
* CALL PROGRAM
* CLP                                                       OP-CO
          DE = 192
*
*         SP -> MAR, 1 -> WREG
*         REST OF CODE IS IN CONCURRENT P-CODE FILE
CLP:  ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB &
        BRAM WREG & CNTLB $11 & XJUMP CALLPROG
00D6: C8 4C 29 9F 32 69 F1 8F F0 03 XX XX

*
* TRUNCATE REAL
* TNR                                           OP-CODE = 194
*
*         SP -> MAR,128 -> EXP1
TNR:  ALU REG PASSR & DAQ & OEY & PCU QEU SUB PCUAB A4 B0 &
        PCUSP & AM2904 & CARRYCTL & DATAPATH ZZI ,,,,,LDMAR &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM EXP1 &
        IMMD 128 & CONT
00D7: D2 7B 69 9F 32 68 FD 87 F0 0E 00 80

*         READ MANTISSA M.S. (OTHER MANTISSA WORDS DON'T MATTER
          )
*         SP+6 -> MAR, -1 -> XREG
        ALU REG HIGH & AQ & OEY & PCU ,,PCUDA A1 B1 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM XREG & IMMD 6 & CONT
00D8: C0 78 69 9F 52 7A FA 07 F0 0E 00 06

*         READ EXPONENT, ZREG (MANTISSA M.S.) -> WREG,QREG
*         PC -> MAR
        ALU RQPT PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
```

```
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP TRUNREAL
00D9: CA 3B 69 8F 30 3B F9 87 F0 03 02 4B

*
* ABSOLUTE VALUE OF WORD
* ABS                                                    OP-CODE =
         196
*
*        SP -> MAR
ABS:  ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       CONT
00DA: 50 62 29 9F 32 69 78 07 F0 0E 00 00

*        READ VALUE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT REQB MREQ & CONTROL & CONT
00DB: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*        ZREG (VALUE) -> YBUS, JUMP TO "NGW" IF < 0,ELSE DO NO
         THING, PC -> MAR
       ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL & TEST DIRIN SALSB & CJP NG1
00DC: 42 63 69 8F 30 29 78 03 F3 33 00 3F

*        READ INSTR N+2, PC+2 -> MAR, JUMP MAP
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       JMAP
00DD: 48 62 29 9F 18 3B 78 07 F0 02 00 00

*
* ABSOLUTE VALUE OF REAL
* ABR                                                      OP-COD
         E = 198
*
*        SP+4 -> Q,MAR, REAL1L -> IREG, SET EXP1 TO -1, M N ->
          0
ABR:  ALU REG HIGH & AQ & OEY & PCU QEU ,PCUAB A5 B1 &
       AM2904 ,,,,ES ,CEM & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR ,LDI & MEMCONT REQB &
       CONTROL ROM & RTB & BRAM EXP1 &
       TESTF $03 & IREG REAL1L-1 REAL1L & CONT
00DE: D0 78 69 BE 1A 69 FD 87 50 3E 00 A9

*        READ FIRST MANTISSA WORD, LOAD AM2910 COUNTER
*        1 -> MICRO C, Q-2 -> R6,MAR,TREG
       ALU YBUS PASS & AQ & YOFF & PCU ,SUB PCUAQ A4 B6 &
       AM2904 ,,,,,,,CEU & CARRYCTL &
       DATAPATH ,,,,,PCUY LDMAR & MEMCONT REQB MREQ &
     TESTF '13         SET CARRY
       CONTROL &         PHLC 1
00DF: 49 62 68 9F 89 BB 78 07 E0 B4 00 01

*        ZREG (MANTISSA M.S.) -> YBUS, JUMP TO NEGATE REAL IF
         -VE
```

```
       ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
       CONTROL & TEST DIRIN NEGATIVE & CJP NEGRL
00E0: 42 63 69 0F 30 29 78 03 F3 F3 00 42

*        PC -> MAR, NO NEGATE NEEDED
       ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP RNI
00E1: 48 62 29 9F 30 29 78 07 F0 03 01 00

*
* SUCCESSOR WORD
* SUC                                              OP-CODE = 200
*
*        SP -> MAR, 1 -> WREG
SUC:  ALU AUR LOW & AQ & OEY & PCUSP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB &
       CONTROL ROM & CNTLB $11 & RTB &
       BRAM WREG & JUMP WADDS
00E2: D8 44 69 9F 32 69 F1 8F F0 03 01 1C

*
* PREDECESSOR WORD
* PRE                                              OP-CODE = 202
*
*        SP -> MAR, -1 -> WREG
PRE:  ALU AUR HIGH & AQ & OEY & PCUSP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB &
       CONTROL ROM & CNTLB $11 & RTB &
       BRAM WREG & JUMP WADDS
00E3: D8 40 69 9F 32 69 F1 8F F0 03 01 1C

*
* CONVERT WORD TO REAL
* CVW                                              OP-CODE = 204
*
*        SP -> R6,MAR, 0 -> WREG,DREG
CVW:  ALU REG LOW & AQ & OEY & PCU ,,PCUZA A1 B6 & AM2904 &
       CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
       JUMP CONVWORD
00E4: D8 7C 69 DF 43 A9 F9 87 F0 03 02 53

*
* TEST EMPTY
* TEM                                                  OP-CODE =
         206
*
*        SP -> MAR
TEM:  ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       JUMP TESTMT
00E5: 58 62 29 9F 32 69 78 07 F0 03 02 5C

*
* GET ATTRIBUTE
* ATR                                                OP-CO
```

```
        DE = 208
*
*       SP -> MAR, 1 -> WREG
*       REST OF CODE IS IN CONCURRENT P-CODE FILE
ATR:  ALU LUR LOW & AB & OEY & PCUSP & AM2904 SHIFTEN &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM WREG & CNTLB $11 & XJUMP GETATR
00E6: D8 4C 29 9F 32 69 F1 8F F0 03 XX XX

*
* REAL TIME.
* RTM                                                    OP-CODE
        = 210
*
*       REAL TIME SECONDS ADDRESS -> MAR
RTM:  ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RTSECS & CONT
00E7: 50 62 29 9E 70 28 78 07 F0 0E 00 40

*       READ REAL TIME IN SECONDS, PC -> MAR.
*       REST OF CODE IS IN CONCURRENT P-CODE FILE.
       ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       XJUMP REALTIME
00E8: 48 62 29 9F 30 3B 78 07 F0 03 XX XX

*
* DELAY PROCESS
* DLY                                                  OP-CODE =
         212
*
*       SP -> MAR, GREG -> TREG
*       REST OF CODE IS IN CONCURRENT P-CODE FILE.
DLY:  ALU YBUS PASSR & AQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ,,LDTREG ,,,LDMAR & MEMCONT REQB &
       CONTROL ROM & ARAM GREG & XJUMP DELAY
00E9: 48 63 79 9F 32 69 F8 0F F0 03 XX XX

*
* CONTINUE PROCESS
* CNT                                                    OP-CODE
        = 214
*
*       SP -> MAR, 0 -> DREG
*       REST OF CODE IS IN CONCURRENT P-CODE FILE.
CNT:  ALU YBUS LOW & AB & OEY & PCUSP & AM2904 &
       DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB & CONTROL &
       XJUMP CONTINUE
00EA: 48 64 29 DF 32 69 78 07 F0 03 XX XX

*
* INPUT/OUTPUT OPERATION
* IOP                                                    OP-CODE
         = 216
*
*       SP -> Q OF PCU,MAR,TREG
*       REST OF CODE IS IN OTHER FILES
```

```
IOP:  ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUZA A1 &
       AM2904 & DATAPATH ZZI ,LDTREG ,,PCUY LDMAR &
       MEMCONT REQB & CONTROL & XJUMP INPOUTOP
00EB: 59 62 38 9E 42 29 78 07 F0 03 XX XX

*
* START PROGRAM
* STRT                                                    OP-COD
        E = 218
*
*       PHREG+CONTATR -> MAR, 1 -> DREG, READ INSTR N+2
STRT:  ALU LUR LOW & AQ & PCU QEU ,PCUDA PHREG &
       AM2904 SHIFTEN & DATAPATH ZZI ,,,,,LDMAR LDD &
       MEMCONT REQB MREQ & CONTROL ROM & RTB &
       BRAM WREG & CNTLB $11 & IMMD CONTATR & CONT
00EC: D0 4C 69 DE 56 3A F1 8F F0 0E 00 14

*       WRITE 1 TO CONTINUE ATTRIBUTE, PC+2 -> MAR, PC+2 -> M
        AR, JUMP MAP
       ALU YBUS PASS & AB & PCUNEXT & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JMAP
00ED: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* STOP PROGRAM
* STP                                                     OP-CO
        DE = 220
*
*       SP -> MAR, NEXT INDEX ADDRESS/2 -> WREG
STP:  ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM WREG & IMMD NEXTINDEX/2 & CONT
00EE: D2 7B 69 9F 32 68 F9 87 F0 0E 00 1A

*       READ RESULT, SP+2 -> MAR
*       REST OF CODE IS IN CONCURRENT P-CODE FILE
       ALU YBUS PASS & AB & PCUPOP & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       XJUMP STOP
00EF: 48 62 29 9F 18 7B 78 07 F0 03 XX XX

*
* SET HEAP REGISTER
* SHP                                                   OP-CODE =
         222
*
*       SP -> MAR
*       REST OF CODE IS IN CONCURRENT P-CODE FILE.
SHP:  ALU YBUS PASS & AB & PCUSP & AM2904 &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB & CONTROL &
       XJUMP SETHREG
00F0: 58 62 29 9F 32 69 78 07 F0 03 XX XX

*
* WAIT FOR 1 SECOND
* WAIT                                                    OP-CO
        DE = 224
```

```
*
*         CODE IS IN CONCURRENT P-CODE FILE
WAIT:   ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & XJUMP WAIT30
00F1: 48 62 29 1F 30 09 78 07 F0 03 XX XX

*
* SEND MESSAGE                                    OP-CODE = 226

* USED DURING SYSTEM FAILURE
* OP-CODE 226 IS READ FROM THE PIA, IF IT IS SUCCESSFULL
* THEN IT ARRIVES HERE, OTHERWISE SYSERR OR INVALID WILL (EVEN
         TUALLY)
* BE CALLED. TELLS WHETHER MAP AND ZIREG ARE FUNCTIONING CORRE
         CTLY
*
*         CODE IS IN FAULT-TOLERANT MICROCODE FILE.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & XJUMP MAPZITEST
00F2: 48 62 29 1F 30 09 78 07 F0 03 XX XX

*
* READ MEMORY TEST            OP-CODE = 228
*
* 0 -> MAR,YREG (MEMORY ADDRESS)
TST1:   ALU REG LOW & AQ & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,YMAR LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM YREG & JSB TSTSUB
00F3: C8 7C 6B 9F 30 29 FA 87 F0 01 02 66

*         READ, YREG + 2 -> YREG (NEXT MEMORY ADDRESS)
TSTLP1:   SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM YREG & CONT
00F4: C0 20 29 1F 30 3B FA 87 F4 0E 00 00

*         READ, YREG AND WREG ($3FFF) -> YREG,MAR (ENSURES ADDR
          ESS < $4000)
        ALU REG AND & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,YMAR LDMAR & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & ARAM WREG & BRAM YREG &
        JUMP TSTLP1
00F5: C8 7E 2B 9F 30 3B FA 9F F0 03 00 F4

*
* WRITE MEMORY TEST (OP-CODE = 230)
*
*
*         0 -> MAR,XREG
TST2:   ALU REG LOW & AQ & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,YMAR LDMAR & MEMCONT & CONTROL ROM &
        RTB & BRAM XREG & CONT
00F6: C0 7C 6B 9F 30 09 FA 07 F0 0E 00 00

*         1 -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & IMMD 1 & CONT
```

```
00F7: C2 7B 69 5F 30 28 F9 87 F0 0E 00 01

*        WRITE, XREG + 2 -> XREG
TSTLP2:  SPF14 & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB MREQ ,WRITE &
       CONTROL ROM & RTB & BRAM XREG & CONT
00F8: C0 20 29 1F 30 3F FA 07 F4 0E 00 00

*        WRITE, XREG AND $03FF -> XREG,TREG (ADDRESS ALWAYS <
         $4000)
       ALU REG AND & DAB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,LDTREG & MEMCONT REQB MREQ ,WRITE &
       CONTROL ROM & RTB & BRAM XREG & IMMD $03FF & CONT
00F9: C2 7E 39 1F 30 3E FA 07 F0 0E 03 FF

* WRITE, TREG -> MAR (VIA PCU), WREG .SHL. 1 -> WREG,DREG
       ALU LUR PASS & AB & OEY & PCU ,,PCUDZ A0 B0 &
       AM2904 SHIFTEN & CARRYCTL &
       DATAPATH ,ENTREG ,,,,LDMAR LDD &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM WREG & CNTLB $1A & JUMP TSTLP2
00FA: C8 4A 09 DF 70 3F F1 D7 F0 03 00 F8

*
* TST3
* TEST INSTRUCTION, CONTINOUS LOOP, NO MEMORY ACESS MADE
*
TST3:  ALU & CONTROL & DATAPATH & MEMCONT & AM2904 & PCUNOP &
       JUMP TST3COD
00FB: 48 00 29 1F 30 09 78 07 F0 03 02 67

* PEND *
*
* REQUEST BUS FOR INSTR N
*
START:  ALU YBUS PASS & AB & OEY & CARRYCTL & CONTROL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & AM2904 &
       PCUNOP & CONT
00FC: 40 62 29 9F 30 29 78 07 F0 0E 00 00

*
* READ INSTR N
*
START1:  ALU YBUS PASS & AB & OEY & CARRYCTL & CONTROL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & AM2904 &
       PCUNEXT & CONT
00FD: 40 62 29 9F 18 3B 78 07 F0 0E 00 00

*
* READ INSTR N+1, INSTR N -> ZI-REG, DECODE INSTR N
*
START2:  ALU YBUS PASS & DAB & CARRYCTL & OEY & CONTROL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB MREQ & AM2904 &
       PCUNEXT & JMAP
00FE: 5A 62 29 9F 18 3B 78 07 F0 02 00 00

* ARRIVED HERE IF MAP FAULTY, JUMP TO FAULT-TOLERANT CODE
```

```
00FF: 48 62 29 1F 30 09 78 07 F0 03 XX XX

*
* REQUEST NEXT INSTRUCTION
*
RNI:  ALU YBUS PASS & DAB & CARRYCTL & OEY & CONTROL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & AM2904 &
       PCUNEXT & JMAP
0100: 4A 62 29 9F 18 3B 78 07 F0 02 00 00

*
*        CODE USED BY "PCA"
*
*        ZREG (JOB MODE) -> YBUS, PC -> MAR, JUMP IF SYSTEM MO
         DE
TESTMODE:  ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL &
       TEST DIRIN UAEQB & CJP PCSYSTEM
0101: 42 63 69 8F 30 29 78 03 F3 53 01 05

*
* ARRIVED HERE IF USER MODE
*        G+10 -> MAR, READ INSTR N+1
       ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM GREG &
       IMMD 10 & CONT
0102: C2 61 AB 9F 30 3A F8 87 F0 0E 00 0A

*
*        READ CONSTANT ADDRESS, PC+2 -> MAR
READCONST:  ALU YBUS PASS & AB & PCUNEXT & AM2904 &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
0103: 50 62 29 9F 18 3B 78 07 F0 0E 00 00

*        ZREG (CONSTANT ADDRESS) + WREG ("DISPLACEMENT") -> DR
         EG
*        SP-2 -> MAR, READ INSTR N+2, JUMP TO WRTS
       ALU YBUS ADD & DAB & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       JUMP WRTS
0104: CA 61 A9 CF 98 7B F9 87 F0 03 01 15

*        ENTERED HERE IF SYSTEM MODE
*        PHREG+CONSTADDR -> MAR, READ INSTR N+1
PCSYSTEM:  ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG &
       AM2904 & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & IMMD CONSTATR & CONT
0105: 40 62 29 9E 56 3A 78 07 F0 0E 00 22

*        READ CONSTANT ADDRESS, PC+2 -> MAR
       ALU YBUS PASS & AB & PCUNEXT & AM2904 &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
0106: 50 62 29 9F 18 3B 78 07 F0 0E 00 00
```

```
*         ZREG (CONSTANT ADDRESS) + WREG ("DISPLACEMENT") -> DR
          EG
*         SP-2 -> MAR, READ INSTR N+2
        ALU YBUS ADD & DAB & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        JUMP WRTS
0107: CA 61 A9 CF 98 7B F9 87 F0 03 01 15

*
* CODE USED BY "PSW"
*
*         READ ADDRESS FROM STACK TOP
PUSHWORD:  ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ & CONTROL &
        CONT
0108: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*         ADDRESS -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
0109: 42 63 6B 8F 30 29 78 07 F0 0E 00 00

*         READ VALUE FROM ADDRESS
        ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL & CONT
010A: 40 62 29 9F 30 3B 78 07 F0 0E 00 00

*         VALUE -> DREG, SP -> MAR, READ INSTR N+2
        ALU YBUS PASSR & DAQ & OEY & PCUSP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & JUMP WRTS
010B: 4A 63 69 CF 32 7B 78 07 F0 03 01 15

*         READ ADDRESS FROM STACK TOP
PUSHBYTE:  ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ & CONTROL &
        CONT
010C: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*         ADDRESS -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
010D: 42 63 6B 8F 30 29 78 07 F0 0E 00 00

*         READ VALUE FROM ADDRESS
        ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL & CONT
010E: 40 62 29 9F 30 39 78 07 F0 0E 00 00

*         VALUE -> DREG, SP -> MAR, READ INSTR N+2
        ALU YBUS PASSR & DAQ & OEY & PCUSP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
```

```
       MEMCONT REQB MREQ & CONTROL & JUMP WRTS
010F: 4A 63 69 CF 32 7B 78 07 F0 03 01 15

*
* TRANSFER FROM MEMORY TO STACK CODE
* SHARED BY "PSL", "PSG"
*
*           READ VALUE FROM MEMORY
MSTRAN:  ALU YBUS PASS & AB & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL & CONT
0110: 50 62 29 9F 18 3B 78 07 F0 0E 00 00

*        PUT VALUE INTO D-REG, READ INSTR N+2
       ALU YBUS PASSR & DAQ & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0111: 42 63 69 CF 98 7B 78 07 F0 0E 00 00

*        WRITE VALUE TO STACK
       ALU YBUS PASS & AB & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JMAP
0112: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* SUBROUTINE TO READ THE TOP TWO STACK WORDS.
* ASSUMES MAR ALREADY HOLDS THE FIRST ADDRESS.
* USED BY "ADD","SUB", ALL COMPARE WORD INSTRUCTIONS.
*
*        READ WORD1
RW1W2:  ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       CONT
0113: 40 62 29 9F 18 7B 78 07 F0 0E 00 00

*        WORD1 -> RAM, READ WORD2
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB &
       CNTLB WREG.LS.4 & RTN
0114: CA 7B 69 8F 30 3B F9 87 F0 0A 00 00

*
* CODE TO WRITE CONTENTS OF D-REG TO THE STORE.ASSUMES MAR ALR
        EADY
* HOLDS THE ADDRESS.
* USED BY "ADD", "SUB", "NGW" "PSW","CPW", ALL COMPARE WORD IN
        STRUCTIONS.
*
*        WRITE RESULT TO STACK
WRTS:  ALU YBUS PASSR & DAQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JMAP
0115: 4A 63 69 9F 18 3F 78 07 F0 02 00 00

*
* MULIPLE PUSH CODE.
```

```
* SHARED BY "PSS","PSR".
* PUSHES WORDS FROM A SOURCE STARTING ADDRESS ONTO THE STACK.
* SOURCE ADDRESS IS IN THE ZREG, NO. OF WORDS TO BE PUSHED
* IS IN THE AM2910 COUNTER (-1)
*
*        ZREG (STARTING ADDRESS) + WREG (14 or 6) -> MAR,WREG
         (POINTS TO BOTTOM
MPUSH:  ALU REG ADD & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
0116: C2 79 AB 8F 30 29 F9 87 F0 0E 00 00

*        THE ACTUAL PUSHING NOW STARTS, NOTE THAT THERE ARE TH
         REE OPERATIONS INV
*        IN A TRANSFER OF ANY ONE WORD :
*        READ WORD - TRANSFER TO DREG - WRITE TO STACK
*        THIS CAN BE PIPELINED :
*        READ FIRST WORD - (READ NEXT WORD+TRANSFER CURRENT WO
         RD TO DREG
*        - WRITE CURRENT WORD TO STACK)*(N-1) -TRANSFER LAST W
         ORD TO DREG-
*        WRITE LAST WORD.
*
*        READ FIRST WORD, WREG (ADDRESS) - 2 -> MAR,WREG
       ALU REG SUBR & DAB & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       TEST DIRIN UALSB & IMMD 2 & PUSH 2
0117: C2 78 AB 9F 30 3A F9 83 F7 A4 00 02

*        NEXT TWO MICROWORDS ARE A LOOP
*        ZREG (CURRENT WORD) -> DREG, READ NEXT WORD,SP - 2 ->
          MAR
PIPELN:  ALU YBUS PASSR & DAQ & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0118: 42 63 69 CF 98 7B 78 07 F0 0E 00 00

*        WRITE CURRENT WORD, ADDRESS - 2 -> MAR,WREG
       ALU REG SUBR & DAB & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM WREG & IMMD 2 & RFCT
0119: C2 78 AB 9F 30 3E F9 87 F4 08 00 02

*        ZREG (LAST WORD) -> DREG, SP - 2 -> MAR
       ALU YBUS PASSR & DAQ & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB & CONTROL & CONT
011A: 42 63 69 CF 98 69 78 07 F0 0E 00 00

*        WRITE LAST WORD, PC -> MAR
       ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JUMP RNI
011B: 48 62 29 9F 30 3F 78 07 F0 03 01 00

*
```

```
* CODE TO ADD THE CONTENTS OF WREG TO THE TOP STACK WORD.
* ASSUMES SP ALREADY IN MAR.
* USED BY "FLD","SUC","PRE" (ENTRY POINT FOR "FLD" IS SECOND M
        ICROINSTRUCTION.
*
*         READ VALUE FROM STACK, PC + 2 -> MAR
WADDS:   ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       CONT
011C: 40 62 29 9F 30 3B 78 07 F0 0E 00 00

*         READ INSTR N+2, ZREG (VALUE) + WREG -> DREG, SP -> MA
          R
FLDENT:   ALU YBUS ADD & DAB & OEY & PCUSP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       CONT
011D: C2 61 A9 CF 32 7B F9 87 F0 0E 00 00

*         WRITE UPDATED VALUE TO STACK, PC+2 -> MAR
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JMAP
011E: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* CODE USED BY "PAC"
* PUSH ARRAY COMPONENT.
*
*         READ "LENGTH", ZREG ("MAX-MIN") -> XREG, SP -> MAR
PSHARRAY:   ALU REG PASSR & DAQ & OEY & PCUSP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB &
       CNTLB XREG.LS.4 & CONT
011F: C2 7B 69 8F 32 7B FA 07 F0 0E 00 00

*         READ INDEX FROM STACK, ZREG ("LENGTH") -> YREG, PC+2
          -> MAR, LOAD IREG
       ALU REG PASSR & DAQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,LDI ENZ0 &
       IREG YREG XREG & MEMCONT REQB MREQ & CONTROL ROM &
       RTB & CNTLB YREG.LS.4 & CONT
0120: C2 7B 69 AF 18 3B FA 87 F0 0E 00 45

*         READ INSTR N+1, ZREG ("INDEX") - WREG ("MIN") -> QREG

*         EXCEPTION IF < 0
       ALU QPT SUBS & DAB & YOFF & PCU 1 0 PCUDZ A0 B6 &
       AM2904 ,OECT & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & CNTLB WREG.LS.4 &
       TEST DIRIN SALSB & CJPX RANGEEXP
0121: C3 31 29 0F 71 BB F9 83 F7 33 XX XX

*         SP+2 -> MAR, QREG (INDEX - "MIN") - XREG ("MAX-MIN")
          -> YBUS
*         EXCEPTION IF > 0
       ALU YBUS SUBR & AQ & OEY & PCUPOP & AM2904 ,OECT &
```

```
       CARRYCTL COEQ1 & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB & CONTROL ROM & CNTLB XREG &
       TEST DIRIN SAGRB & CJPX RANGEEXP
0122: 50 60 E9 9F 18 69 F8 23 F7 03 XX XX

*        READ BASE ADDRESS, 0 -> XREG, LOAD COUNTER
       ALU REG LOW & AQ & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT REQB MREQ & CONTROL ROM & RTB &
       CNTLB XREG.LS.4 & LDCT $E
0123: C0 7C 69 1F 30 3B FA 07 F0 0C 00 0E

*        TWO'S COMPLEMENT MULTIPLY
* AT START: XREG = 0 (R0),
*           QREG = MULTIPLIER (INDEX-"MIN"),
*           YREG = MULTIPLICAND ("LENGTH") (R0)
FADDR:  SPF14 TCMUL & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH & MEMCONT & CONTROL & RTB &
       CNTLB $0E & RPCT FADDR
0124: C0 10 29 1F 30 09 70 77 F0 09 01 24

*        TWO'S COMPLEMENT MULTIPLY, LAST CYCLE, PC+2 -> MAR
       SPF14 TCMLS & AB & OEY & PCUNEXT & AM2904 SHIFTEN &
       CARRYCTL COEQCI & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB & CONTROL &. RTB & CNTLB $0E & CONT
0125: C0 30 29 9F 18 29 70 77 F8 0E 00 00

*        READ INSTR N+2, ZREG (ADDRESS) + QREG ("LENGTH"*INDEX
         ) -> DREG, SP -> M
       ALU YBUS ADD & DAQ & OEY & PCUSP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR LDD ,ENZ0 & MEMCONT REQB MREQ &
       CONTROL & CONT
0126: 42 61 E9 CF 32 7B 78 07 F0 0E 00 00

*        WRITE  ADDRESS, PC+2 -> MAR
       ALU YBUS PASS & AQ & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JMAP
0127: 48 62 69 9F 18 3F 78 07 F0 02 00 00

*
* CODE USED BY "CTF"
* CHECK TAG FIELD
*
*        READ BASE ADDRESS, 15 -> XREG
CHCKTAG:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM XREG & IMMD 15 & CONT
0128: C2 7B 69 1F 30 3A FA 07 F0 0E 00 0F

*        ZREG (BASE ADDRESS) + WREG ("DISPLACEMENT") -> MAR, L
         OAD IREG
       ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,LDI ENZ0 &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
       IREG WREG WREG & CONT
0129: C2 61 AB AF 30 29 F9 87 F0 0E 00 33

*        READ VALUE, $8000 -> QREG
```

```
        ALU QPT PASSR & DAQ & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT REQB MREQ & CONTROL & IMMD $8000 &
        CONT
012A: 42 33 69 1F 30 3A 78 07 F0 0E 80 00

*         ZREG (VALUE) -> AM2910 COUNTER, PC+2 -> MAR, 0 -> WRE
          G
        ALU REG LOW & AB & OEY & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR ,,,,SHTCNTEN & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & LDCT 0
012B: C0 7C 29 9B 18 29 F9 87 F0 0C 00 00

*         ZREG (VALUE) -> YREG, EXCEPTION IF < 0, PC-2 -> MAR,
          READ INSTR N+1
        ALU REG PASSR & DAQ & OEY & PCU ,SUB PCUAB A4 B0 &
        AM2904 ,OECT & CARRYCTL &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM YREG & TEST DIRIN SALSB &
        CJPX VARIANTEXP
012C: C2 7B 69 8F 98 3B FA 83 F3 33 XX XX

*         YREG (VALUE) - XREG (15) -> YBUS, EXCEPTION IF > 0, R
          EAD "TAGSET", PC+2
        ALU YBUS SUBS & AB & OEY & PCUNEXT & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ZZI & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & ARAM YREG & BRAM XREG &
        TEST DIRIN SAGRB & CJPX VARIANTEXP
012D: D0 61 29 1F 18 3B FA 2B F7 03 XX XX

*         SHIFT WREG. NOTE THE 'TRICK' OF ALREADY HAVING SET TH
          E M.S. BIT OF Q TO
*         THUS ENSURING THAT IF N IS IN THE AM2910 COUNTER THEN
           THE N'TH BIT OF
*         WREG WILL BE SET TO 1 AFTER THE SHIFTING OPERATION CO
          MPLETES.
SHW:  ALU LURQ PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL & CNTLB $14 &
        RPCT SHW
012E: 40 5A 29 1F 30 09 70 A7 F0 09 01 2E

*         WREG.ZREG ("TAGSET") -> YBUS, EXCEPTION IF = 0, PC+2
          -> MAR
        ALU YBUS AND & DAB & OEY & PCUNEXT & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN UAEQB & CJPX VARIANTEXP
012F: C2 66 29 8F 18 29 F9 83 F3 53 XX XX

*         READ INSTR N+2, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        JMAP
0130: 48 62 29 9F 18 3B 78 07 F0 02 00 00

*
* CODE USED BY "CPB"
* COPY BYTE
*
```

```
*        READ VALUE FROM TOP OF STACK
COPYBYTE:  ALU YBUS PASS & AB & OEY & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
0131: 40 62 29 9F 18 7B 78 07 F0 0E 00 00

*        VALUE -> DREG, POP ADDRESS FROM TOP OF STACK
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0132: 42 63 69 CF 30 3B 78 07 F0 0E 00 00

*        ADDRESS -> MAR, READ INSTR N+2
       ALU YBUS PASSR & DAQ & OEY & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0133: 42 63 6B 8F 18 7B 78 07 F0 0E 00 00

* WRITE RESULT TO STACK
       ALU YBUS PASSR & DAQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & JMAP
0134: 4A 63 69 9F 18 3D 78 07 F0 02 00 00

*
* CODE USED BY "CPW"
* COPY WORD
*
*        READ VALUE FROM TOP OF STACK
COPYWORD:  ALU YBUS PASS & AB & OEY & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
0135: 40 62 29 9F 18 7B 78 07 F0 0E 00 00

*        VALUE -> DREG, POP ADDRESS FROM TOP OF STACK
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0136: 42 63 69 CF 30 3B 78 07 F0 0E 00 00

*        ADDRESS -> MAR, READ INSTR N+2
       ALU YBUS PASSR & DAQ & OEY & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & JUMP WRTS
0137: 4A 63 6B 8F 18 7B 78 07 F0 03 01 15

*
* CODE USED BY "CPT"
* COPY TAG
*
*        READ TAG, SP+2 -> MAR
COPYTAG:  ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
       DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
0138: 50 62 29 9F 18 7B 78 07 F0 0E 00 00

*        ZREG (TAG) -> DREG, READ ADDRESS, PC+2 -> MAR
       ALU YBUS PASSR & DAQ & OEY & PCUNEXT & AM2904 &
```

```
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0139: 42 63 69 CF 18 3B 78 07 F0 0E 00 00

*        READ INSTR N+2, ZREG (ADDRESS) -> MAR,TREG, SP+2 -> S
         P
       ALU YBUS PASSR & DAQ & OEY & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,LDTREG ,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
013A: 42 63 7B 8F 18 7B 78 07 F0 0E 00 00

*        WRITE TAG, TREG (ADDRESS) + 2 -> MAR,R6, 0 -> DREG
       ALU YBUS LOW & AB & OEY & PCU ,,PCUDA A4 B6 & AM2904 &
       CARRYCTL & DATAPATH ,ENTREG ,,,,LDMAR LDD &
       MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
013B: 40 64 09 DF 59 BF 78 07 F0 0E 00 00

*        WRITE ZERO TO STORE, R6 (ADDRESS) + 2 -> MAR, WREG +
         1 -> YBUS
*        LOOP AND STOP WHEN = 0
WZTS:  SPF14 INCTWO & AB & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 ,OECT & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM WREG & TEST DIRIN SALSB & CJP WZTS
013C: C0 20 29 9F 19 BF F9 83 F3 33 01 3C

*        PC+2 -> MAR, JUMP MAP
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & JMAP
013D: 48 62 29 9F 18 29 78 07 F0 02 00 00

*
* CODE USED BY "CST"
* COPY STRUCTURE
*
*        SP+2 -> MAR, READ SOURCE ADDRESS, INCREMENT QREG ("LE
         NGTHWORDS") BY 1
COPYSTRUCT:  ALU QPT PASS & AQ & PCUPOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL & CONT
013E: 50 32 69 9F 18 7B 78 07 F4 0E 00 00

*        ZREG (SOURCE ADDRESS) -> XREG,MAR, READ DESTINATION A
         DDRESS, SP+2 -> SP
       ALU REG PASSR & DAQ & OEY & PCUPOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
       CONT
013F: C2 7B 6B 8F 18 7B FA 07 F0 0E 00 00

*        READ FIRST WORD, ZREG (DESTINATION ADDRESS) -> R7 OF
         PCU, XREG + 2 -> T
       SPF14 INCTWO & AB & OEY & PCU ,,PCUDZ A7 B7 & AM2904 &
       CARRYCTL COEQ1 &
       DATAPATH ,,LDTREG ,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
       CONT
0140: C0 20 3B 8F 7F FB FA 07 F4 0E 00 00
```

```
*         R7 (ADDRESS) - 2 -> R7, 1 -> XREG
        ALU REG PASSR & DAQ & OEY & PCU ,SUB PCUAB A4 B7 &
        AM2904 & CARRYCTL & DATAPATH & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM XREG & IMMD 1 & CONT
0141: C2 7B 69 1F 99 E8 FA 07 F0 0E 00 01

*         NOTE THE PIPELINE, SEE "PSS".
*         READ NEXT WORD, ZREG (CURRENT WORD) -> DREG, R7 (DEST
           ADDRESS) + 2 -> M
CPYLOOP:  ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & CONT
0142: 42 63 69 CF 19 FB 78 07 F0 0E 00 00

*         WRITE CURRENT WORD, TREG + 2 -> MAR,TREG, QREG + 1 ->
           QREG, LOOP IF < 0
        ALU QPT ADD & AQ & YOFF & PCU ,,PCUDA A4 B6 &
        AM2904 ,OECT & CARRYCTL &
        DATAPATH ,ENTREG LDTREG ,,PCUY LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM XREG &
        TEST DIRIN SALSB & CJP CPYLOOP
0143: 41 31 D8 9F 59 BF F8 23 F3 33 01 42

*         ZREG (LAST WORD) -> DREG, R7 + 2 -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
0144: 42 63 69 CF 19 E9 78 07 F0 0E 00 00

*         WRITE LAST WORD, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
0145: 48 62 29 9F 18 3F 78 07 F0 03 01 00

*
* MULTIPLE PULL CODE.
* SHARED BY "CPS","CPR".
* PULLS WORDS FROM THE STACK TO A DESTINATION ADDRESS CONTAINE
         D IN R6 OF PCU
* ALSO ASSUMES THAT WREG POINTS TO THE STACK AND THE AM2910 CO
         UNTER HOLDS THE
* NUMBER OF WORDS TO BE PULLED - 2. USES A PIPELINE, SIMILAR T
         O "PSS".
* ASSUMES FIRST WORD IS ALREADY IN ZREG.
*
*         READ NEXT VALUE, ZREG (CURRENT VALUE) -> DREG, R6 (AD
          DRESS) -> MAR
MPULL:   ALU YBUS PASSR & DAQ & OEY & PCU 1 0 PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & CONT
0146: 42 63 69 CF 4D BB 78 07 F0 0E 00 00

*         WRITE NEXT VALUE, WREG (SP') + 2 -> MAR, R6 (ADDRESS)
           + 2 -> R6
        SPF14 INCTWO & AB & OEY & PCU 1 0 PCUAB A4 B6 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
```

```
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM WREG & RPCT MPULL
0147: C0 20 2B 9F 19 BF F9 87 F4 09 01 46

*        ZREG (LAST VALUE) -> DREG, R6 (ADDRESS) -> MAR
       ALU YBUS PASSR & DAQ & OEY & PCU 1 0 PCUZA A6 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB & CONTROL & CONT
0148: 42 63 69 CF 4D A9 78 07 F0 0E 00 00

*        WRITE LAST VALUE, PC -> MAR
       ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP RNI
0149: 48 62 29 9F 30 3F 78 07 F0 03 01 00

*
* SUBROUTINE TO SET UP NEW AREA ON HEAP.
* USED BY "IPV","IPVC".
*
*        QREG - HREG -> YBUS, EXCEPTION IF < 0, SP+2 -> SP
NEWHEAP:  ALU YBUS SUBR & AQ & OEY & PCUPOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB &
       CONTROL ROM & ARAM HREG & TEST DIRIN UALSB &
       CJPPX HEAPEXP
014A: 40 60 E9 1F 18 69 F8 13 F7 AB XX XX

*        -ZREG ("LENGTH") -> WREG, PC+2 -> MAR, READ POINTER A
         DDRESS
       ALU REG COMPLR & DAQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       CONT
014B: C2 7B E9 8F 18 3B F9 87 F4 0E 00 00

*        ZREG (POINTER ADDRESS) -> MAR, HREG -> DREG,TREG, REA
         D INSTR N+1
       ALU YBUS PASSR & AQ & OEY & PCU ,,PCUDZ A7 B7 &
       AM2904 & CARRYCTL &
       DATAPATH ,,LDTREG ,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & ARAM HREG & CONT
014C: 40 63 79 CF 7F FB F8 17 F0 0E 00 00

*        WRITE ADDRESS, HREG  WREG (-"LENGTH") -> HREG, PC+2 -
         > MAR
       ALU REG SUBR & AB & OEY & PCUNEXT & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ZZI ,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       ARAM WREG & BRAM HREG & RTN
014D: D8 78 A9 9F 18 3F F9 1F F4 0A 00 00

*
* CODE USED BY "ANS"
* NOTE THE PIPELINING OF :
* READ S1 - READ S2 - S1 AND S2 -> DREG - WRITE RESULT
* TO :
* READ S1, PREVIOUS S1 AND S2 -> DREG - READ S2 - WRITE RESULT
```

```
*
*         READ NEXT TOP SET WORD, R6+2 -> MAR,
*         ZREG (CURRENT BOTTOM SET WORD) AND WREG (CURRENT TOP
          SET WORD) -> DREG
ANDSET:   ALU YBUS AND & DAB & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
014E: C2 66 29 CF 19 BB F9 87 F0 0E 00 00

*         READ NEXT BOTTOM SET WORD, R6-2 -> MAR (BUT NOT R6)
*         ZREG (NEXT TOP SET WORD) -> WREG
        ALU REG PASSR & DAQ & OEY & PCU QEU SUB PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
014F: C2 7B 69 8E 99 BB F9 87 F0 0E 00 00

*         WRITE CURRENT RESULT, SP + 2 -> MAR
        ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & RPCT ANDSET
0150: 40 62 29 9F 18 7F 78 07 F0 09 01 4E

*         ZREG (LAST BOTTOM SET WORD).WREG (LAST TOP SET WORD)
          -> DREG, R6 -> MAR
        ALU YBUS AND & DAB & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
0151: C2 66 29 CF 4D A9 F9 87 F0 0E 00 00

*         WRITE LAST RESULT, PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
0152: 48 62 29 9F 30 3F 78 07 F0 03 01 00

*
* CODE USED BY "ORS".
* OR SET
*
*
* NOTE THE PIPELINING OF :
* READ S1 - READ S2 - S1 OR S2 -> DREG - WRITE RESULT
* TO :
* READ S1, PREVIOUS S1 OR S2 -> DREG - READ S2 - WRITE RESULT
*
*         READ NEXT TOP SET WORD, R6+2 -> MAR,
*         ZREG (CURRENT BOTTOM SET WORD) OR WREG (CURRENT TOP S
          ET WORD) -> DREG
ORSET:    ALU YBUS OR & DAB & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
0153: C2 67 A9 CF 19 BB F9 87 F0 0E 00 00

*         READ NEXT BOTTOM SET WORD, R6-2 -> MAR (BUT NOT R6)
*         ZREG (NEXT TOP SET WORD) -> WREG
```

```
        ALU REG PASSR & DAQ & OEY & PCU QEU SUB PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
0154: C2 7B 69 8E 99 BB F9 87 F0 0E 00 00

*         WRITE CURRENT RESULT, SP + 2 -> MAR
        ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & RPCT ORSET
0155: 40 62 29 9F 18 7F 78 07 F0 09 01 53

*         ZREG (LAST BOTTOM SET WORD).WREG (LAST TOP SET WORD)
          -> DREG, R6 -> MAR
        ALU YBUS OR & DAB & OEY & PCU ,,PCUZA A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
0156: C2 67 A9 CF 4D A9 F9 87 F0 0E 00 00

*         WRITE LAST RESULT, PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
0157: 48 62 29 9F 30 3F 78 07 F0 03 01 00

*
* SUBROUTINE USED BY "ADR","SBR"
* CODE IS IN 5 PARTS, 1) READ IN BOTH REAL NO.'S
*                     2) SELECT SMALLEST EXPONENT
*                     3) SHIFT DOWN
*                     4) ADD/SUBTRACT
*                     5) NORMALISE
*
*         READ FIRST WORD, SP+2 -> MAR, LOAD I-REG, -1 -> TREG
ADRSUB:   ALU YBUS HIGH & AB & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,LDTREG ,,,LDMAR ,LDI &
        MEMCONT REQB MREQ & CONTROL & IREG REAL2L-1 REAL1M &
        CONT
0158: 40 60 39 BF 18 7B 78 07 F0 0E 00 8D

*         ZREG (NEXT WORD) -> I-REG PORT, LOOP BACK IF A<>B SP+
          2 -> MAR
*         INCREMENT I-REG
        ALU REG PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,,ENCTR INC ,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & RTB & TESTF AEQB &
        CJP *
0159: C2 7B 65 8F 18 7B 78 07 F0 03 01 59

*         ZREG (LAST WORD) -> I-REG PORT (EXP2), LOAD I-REG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,LDI ENZ0 & MEMCONT &
        CONTROL & RTB & IREG REAL2M REAL2M & CONT
015A: C2 7B 69 2F 30 09 78 07 F0 0E 00 CC

*         2) SELECT THE SMALLEST EXPONENT
*         EXP2 - EXP1 -> TREG, JUMP AHEAD IF >0, SET MSR
*         TREG (-1) -> R6 OF PCU
```

```
        ALU YBUS SUBS & AB & OEY & PCU ,,PCUDZ A6 B6 &
        AM2904 ,OECT EZ ,,,CEM & CARRYCTL COEQ1 &
        DATAPATH ,,LDTREG & MEMCONT & CONTROL ROM & RTB &
        ARAM EXP2 & BRAM EXP1 & TEST DIRIN SAGRB &
        CJP ADRGRT
015B: C0 61 39 1F 7D 89 FD F9 D7 03 01 5F

*         IF MSR = 0 THEN RETURN (EXPONENTS ARE EQUAL)
*         EXP1 -> DREG
        ALU YBUS PASS & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM EXP1 & TEST MACHINE SAEQB &
        CRTN
015C: C0 62 29 5F 30 09 FD 83 F2 5A 00 00

*         REAL2M -> VREG - 1
        ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        ARAM REAL2M & BRAM VREG-1 & CONT
015D: C0 7B 69 1F 30 09 FB 67 F0 0E 00 00

*         1 -> WREG (VIA PCU), JUMP PAST CODE, REAL2L -> QREG
        ALU RQPT PASSR & AQ & YOFF & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,PCUY & MEMCONT &
        CONTROL ROM & RTB & ARAM REAL2L & BRAM WREG &
        JUMP TTOY
015E: C9 3B 68 1F 19 89 F9 F7 F0 03 01 62

*         R6 OF PCU (-1) -> WREG, REAL1L -> QREG, LOAD I-REG
ADRGRT:   ALU RQPT PASSR & AQ & YOFF & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,PCUY ,,LDI &
        MEMCONT & CONTROL ROM & RTB & ARAM REAL1L &
        BRAM WREG & IREG REAL1M REAL1M & CONT
015F: C1 3B 68 3F 4D 89 F9 D7 F0 0E 00 88

*         REAL1M -> VREG-1
        ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        ARAM REAL1M & BRAM VREG-1 & CONT
0160: C0 7B 69 1F 30 09 FB 47 F0 0E 00 00

*         EXP2 -> EXP1 (LARGEST EXPONENT NOW IN EXP1)
        ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & ARAM EXP2 & BRAM EXP1 & CONT
0161: C0 7B 69 5F 30 09 FD FF F0 0E 00 00

*         TREG (EXP2 - EXP1) -> YREG
TTOY:   ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,ENTREG & MEMCONT & CONTROL ROM &
        RTB & BRAM YREG & CONT
0162: C2 7B 49 1F 30 09 FA 87 F0 0E 00 00

*         3) SHIFT DOWN NOW STARTS
*         LOAD MSR C BY SHIFTING VREG-1
        ALU LDR PASS & AB & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM VREG-1 & CNTLB $02 & CONT
```

```
0163: C0 0A 29 1F 30 09 F3 17 F0 0E 00 00

*         SHIFT B-PORT (MANTISSA M.S.), KEEP A COPY IN TREG, IN
          C B-PORT
SHBPRT:   ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,LDTREG ENCTR INC & MEMCONT &
        CONTROL & RTB &
        CNTLB $0E & CONT
0164: C0 0A 35 1F 30 09 70 77 F0 0E 00 00

*         ADD WREG (1 OR -1) TO YREG, JUMP OUT OF LOOP IF = 0
        ALU REG ADD & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        ARAM WREG & BRAM YREG & TEST DIRIN SAEQB &
        CJP LASTBSHT
0165: C0 79 A9 1F 30 09 FA 9B F3 53 01 68

*         SHIFT B-PORT,QREG,DECREMENT B-PORT
        ALU LDRQ PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,ENCTR & MEMCONT & CONTROL &
        RTB &
        CNTLB $04 & CONT
0166: C0 1A 21 1F 30 09 70 27 F0 0E 00 00

*         TREG (COPY OF MANTISSA M.S.) SHIFT DOWN -> VREG-1
*         (SHIFT SETS MSR C).LOOP BACK.
        ALU LDR PASSR & DAQ & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,ENTREG & MEMCONT & CONTROL ROM &
        RTB &
        BRAM VREG-1 & CNTLB $02 & JUMP SHBPRT
0167: CA 0B 49 1F 30 09 F3 17 F0 03 01 64

*         LAST SHIFT.
*         SHIFT B-PORT,QREG, INCREMENT B-PORT
LASTBSHT:   ALU LDRQ PASS & AB & OEY & PCUNOP &
        AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,ENCTR INC &
        MEMCONT & CONTROL & RTB & CNTLB $04 & CONT
0168: C0 1A 25 1F 30 09 70 27 F0 0E 00 00

*         QREG -> B-PORT, RETURN TO CALLING CODE
        ALU REG PASS & AQ & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT & CONTROL & RTB & RTN
0169: C8 7A 69 1F 30 09 78 07 F0 0A 00 00

*
* CODE TO NORMALISE A 48-BIT NO. IN THE REAL1 REGISTER.
* ASSUMES I-REG POINTS TO REAL1L ON ENTRY AND QREG CONTAINS
* A COPY OF REAL1I (INTERMEDIATE 16 BITS).ALSO ASSUMES XREG =
          -1
*
*         SHIFT I-REG PORT (MANTISSA L.S.), LOAD I-REG
REALNORM:   ALU LUR PASS & AB & OEY & PCUNOP &
        AM2904 SHIFTEN & DATAPATH ,,,,,,,,LDI & MEMCONT &
        CONTROL & RTB & CNTLB $10 & IREG REAL1M REAL1M &
        CONT
016A: C0 4A 29 3F 30 09 70 87 F0 0E 00 88

*         DOUBLE LENGTH NORMALISE (MANTISSA M.S. <- QREG <- MSR
```

```
          CARRY)
*         LOAD MSR, JUMP OUT OF LOOP IF COMPLETE, INCREMENT I-R
          EG
        SPF14 DLN & AB & OEY & PCUNOP &
        AM2904 SHIFTEN OECT ,,ES ,CEM &
        DATAPATH ,,,ENCTR INC & MEMCONT & CONTROL & RTB &
        CNTLB $1C &
        TESTF $3B & CJP RNLAST
016B: C0 50 25 1F 30 09 70 E3 53 B3 01 6D

*         EXP1 + XREG (-1) -> EXP1,DREG, LOOP BACK, INCREMENT I
          -REG
        ALU REG ADD & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,ENCTR INC ,,LDD & MEMCONT & CONTROL ROM &
        RTB & ARAM XREG & BRAM EXP1 & JUMP REALNORM
016C: C8 79 A5 5F 30 09 FD A7 F0 03 01 6A

*         FINAL SHIFT BACK NOW STARTS. THIS TAKES 4 MICROCYCLES
          .
*         1) LOAD MC WITH QREG L.S. BIT, BUT DON'T SHIFT QREG
RNLAST:   ALU LDR PASS & AQ & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM XREG & CNTLB $02 & CONT
016D: C0 0A 69 1F 30 09 F2 17 F0 0E 00 00

*         SHIFT MN -> MANTISSA M.S. -> QREG (MANTISSA I.S.)
        ALU LDRQ PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM REAL1M & CNTLB $05 & LDCT 2
016E: C0 1A 29 1F 30 09 F4 2F F0 0C 00 02

*         3) SHIFT MANTISSA L.S., LOAD I-REG
        ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,,,,,,LDI & MEMCONT &
        CONTROL ROM & RTB &
        BRAM REAL1L & CNTLB $04 & IREG REAL1L REAL1L & CONT
016F: C0 0A 29 3F 30 09 F5 27 F0 0E 00 AA

*         4) QREG GOES BACK TO MANTISSA I.S., SP -> MAR, JUMP T
          O WRITE BACK CODE
        ALU REG PASS & AQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL1I & JUMP REALWRT
0170: C8 7A 69 9F 98 69 FC 87 F0 03 01 71

*
* CODE TO PUT THE CONTENTS OF THE REAL1 REGISTERS ON THE STACK
          .
* ASSUMES I-REG POINTS TO EXP1 AND SP POINTS EXP1 MEMORY
* DESTINATION ADDRESS.
*
*         WRITE NEXT NO., DECREMENT I-REG, SP-2 -> MAR, LOAD DR
          EG
REALWRT:   ALU YBUS PASS & AB & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,ENCTR ,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTB &
        RPCT REALWRT
0171: C0 62 21 DF 98 7F 78 07 F0 09 01 71
```

```
*         WRITE LAST RESULT, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & RTB & JMAP
0172: C8 62 29 9F 18 3F 78 07 F0 02 00 00

*
* CODE USED BY "SBS".
* SUBTRACT SET
*
*
* NOTE THE PIPELINING OF :
* READ S1 - READ S2 - S1 SUBTRACT S2 -> DREG - WRITE RESULT
* TO :
* READ S1, PREVIOUS S1 SUBTRACT S2 -> DREG - READ S2 - WRITE R
          ESULT
*
*         READ NEXT TOP SET WORD, R6+2 -> MAR,
*         ZREG (CURRENT BOTTOM SET WORD) SUBTRACT WREG (CURRENT
           TOP SET WORD) ->
SUBSET:   ALU YBUS AND & DAB & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
0173: C2 66 29 CF 19 BB F9 87 F0 0E 00 00

*         READ NEXT BOTTOM SET WORD, R6-2 -> MAR (BUT NOT R6)
*         INVERT ZREG (NEXT TOP SET WORD) -> WREG
        ALU REG COMPLR & DAQ & OEY & PCU QEU SUB PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
0174: C2 7B E9 8E 99 BB F9 87 F0 0E 00 00

*         WRITE CURRENT RESULT, SP + 2 -> MAR
        ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & RPCT SUBSET
0175: 40 62 29 9F 18 7F 78 07 F0 09 01 73

*         ZREG (LAST BOTTOM SET WORD).WREG (LAST TOP SET WORD)
          -> DREG, R6 -> MAR
        ALU YBUS AND & DAB & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
0176: C2 66 29 CF 4D A9 F9 87 F0 0E 00 00

*         WRITE LAST RESULT, PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
0177: 48 62 29 9F 30 3F 78 07 F0 03 01 00

*
* CODE USED BY "MLW"
* MULTIPLY WORD
*
```

```
*         READ WORD 1,SP+2 -> MAR
MULTWORD:  ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       CONT
0178: 40 62 29 9F 18 7B 78 07 F0 0E 00 00

*         READ WORD2, ZREG (WORD1) -> YREG, PC -> MAR, LOAD IRE
          G
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,LDI ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM YREG &
       IREG YREG XREG & CONT
0179: C2 7B 69 AF 30 3B FA 87 F0 0E 00 45

*         READ INSTR N+2, ZREG (WORD2) -> QREG, LOAD AM2910 COU
          NTER WITH 14
       ALU QPT PASSR & DAQ & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ & CONTROL &
       LDCT $E
017A: 42 33 69 0F 30 3B 78 07 F0 0C 00 0E

*         TWO'S COMPLEMENT MULTIPLY
* AT START    XREG = 0
*         QREG = MULTIPLIER (WORD2)
*         YREG = MULTIPLICAND (WORD1)
MLTPLY:  SPF14 TCMUL & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH & MEMCONT & CONTROL & RTB &
       CNTLB $0E & RPCT MLTPLY
017B: C0 10 29 1F 30 09 70 77 F0 09 01 7B

*         TWO'S COMPLEMENT MULTIPLY, LAST CYCLE
       SPF14 TCMLS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL COEQCI & DATAPATH & MEMCONT & CONTROL & RTB &
       CNTLB $0E & CONT
017C: C0 30 29 1F 30 09 70 77 F8 0E 00 00

*         QREG (L.S. WORD) -> DREG, SP -> MAR
       ALU YBUS PASS & DAQ & OEY & PCUSP &
       DATAPATH ,,,,,,LDMAR LDD & AM2904 & CARRYCTL &
       MEMCONT REQB & CONTROL & JUMP WRTS
017D: 4A 62 69 DF 32 69 78 07 F0 03 01 15

*
* CODE USED BY "MLR".
* TO MULTIPLY TWO REAL NUMBERS WE DO THE FOLLOWING :-
*     1) ADD THE EXPONENTS.
*     2) MULTIPLY THE MANTISSAS. THIS IS A 48 BIT MULTIPLY,
*        HOWEVER AT EACH STAGE WE ARE ONLY INTERESTED IN THE
*        L.S. BIT OF THE MULTIPLIER. THIS MEANS THAT WE CAN
*        PERFORM THE OPERATION IN 3 * 16 OPERATIONS, PAUSING
*        ONLY TO REFILL THE 16 BIT Q REGISTER.
*     3) NORMALISE. ONLY 1 48 BIT SHIFT AT MOST IS NEEDED.
*
*         READ FIRST WORD, LOAD I-REG, SP+2 -> MAR
MULTREAL:  ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR ,LDI & MEMCONT REQB MREQ &
       CONTROL & IREG REAL2L-1 REAL1M & CONT
017E: 40 62 29 BF 18 7B 78 07 F0 0E 00 8D
```

```
*         ZREG (NEXT WORD) -> I-REG PORT, INCREMENT I-REG, SP+2
           -> MAR,
*         READ NEXT WORD, LOOP BACK IF A-PORT OF I-REG <> B-POR
          T
        ALU REG PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,,ENCTR INC ,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & RTB & TESTF AEQB &
        CJP *
017F: C2 7B 65 8F 18 7B 78 07 F0 03 01 7F

*         EXP1 - 128 -> EXP1, SP-2 -> SP
        ALU REG SUBR & DAB & OEY & PCUPUSH & AM2904 &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
        RTB & BRAM EXP1 & IMMD 128 & CONT
0180: C2 78 A9 1F 98 48 FD 87 F4 0E 00 80

*         ZREG (MULTIPLICAND EXPONENT) + EXP1 -> EXP1, LOAD MSR

*         IF NO OVERFLOW THEN SKIP THE NEXT 5 INSTRUCTIONS
        ALU REG ADD & DAB & OEY & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL &
        DATAPATH ,,,,,,,,ENZ0 & MEMCONT & CONTROL ROM & RTB &
        BRAM EXP1 &
        TESTF $36 & CJP CLRSLT
0181: C2 79 A9 0F 30 09 FD 80 13 63 01 87

*         ARRIVED HERE IF OVERFLOW, EXCEPTION IF MSR<0 (MEANS N
          O. TOO LARGE
*         AS OPPOSED TO TOO SMALL), 0 -> DREG, SP-2 -> MAR
        ALU YBUS LOW & AB & OEY & PCUPUSH & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL & TEST MACHINE NEGATIVE & CJPX EXCEPTION
0182: 40 64 29 DF 98 69 78 03 F2 F3 XX XX

*         ARRIVED HERE IF SUM OF EXPONENTS IS TOO SMALL, RESULT
           IS ZERO.
*         WRITE MANTISSA L.S., 128 -> DREG, SP+2 -> MAR.
        ALU YBUS PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL & IMMD 128 & CONT
0183: 42 63 69 DF 18 7E 78 07 F0 0E 00 80

*         WRITE EXPONENT, SP-4 -> MAR, 0 -> DREG
        ALU YBUS LOW & AB & OEY & PCU ,SUB PCUAB A5 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
0184: 40 64 29 DF 9A 7F 78 07 F0 0E 00 00

*         WRITE MANTISSA I.S., SP-2 -> MAR
        ALU YBUS PASS & AB & PCUPUSH & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & CONT
0185: 40 62 29 9F 98 7F 78 07 F0 0E 00 00

*         WRITE MANTISSA M.S., PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
```

```
       CONTROL & JUMP RNI
0186: 48 62 29 9F 30 3F 78 07 F0 03 01 00

*        CLEAR RESULT REGISTERS (MULT1-MULT3).
*        0 -> MULT1, LOAD I-REG
CLRSLT:  ALU REG LOW & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,LDI & MEMCONT &
       CONTROL ROM & RTB & BRAM MULT1 & IREG MULT2 MULT2 &
       CONT
0187: C0 7C 29 3F 30 09 FA 87 F0 0E 00 66

*        CLEAR MULT2 AND MULT3
*        0 -> I-REG PORT, INCREMENT I-REG, LOOP BACK IF A <> B

       ALU REG LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,LDTREG ENCTR INC & MEMCONT & CONTROL &
       RTB & TESTF AEQB & CJP *
0188: C0 7C 35 1F 30 09 78 07 F0 03 01 88

*        REAL1L -> QREG, 15 -> AM2910 COUNTER, PC -> MAR
       ALU QPT PASSR & AQ & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT & CONTROL ROM &
       ARAM REAL1L & LDCT 15
0189: 40 33 69 9F 30 09 F8 57 F0 0C 00 0F

*        FIRST 16 CYCLES OF THE MULTIPLICATION.
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB MULT16
018A: 48 62 29 1F 30 09 78 07 F0 01 01 9C

*        REAL1I -> QREG, 15 -> AM2910 COUNTER, READ INSTR N+2
       ALU QPT PASSR & AQ & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT REQB MREQ & CONTROL ROM & RTB &
       ARAM REAL1I & LDCT 15
018B: C0 33 69 1F 30 3B F8 4F F0 0C 00 0F

*        SECOND 16 CYCLES OF THE MULTIPLICATION OPERATION.
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB MULT16
018C: 48 62 29 1F 30 09 78 07 F0 01 01 9C

*        REAL1M -> QREG, 14 -> AM2910 COUNTER
       ALU QPT PASSR & AQ & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & ARAM REAL1M &
       LDCT 14
018D: C0 33 69 1F 30 09 F8 47 F0 0C 00 0E

*        LAST 15 MULTIPLY OPERATIONS
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB MULT16
018E: 48 62 29 1F 30 09 78 07 F0 01 01 9C

*        TWO'S COMPLEMENT LAST MULTIPLY OPERATION STARTS NOW.
*        SHIFT L.S. BIT Q AND TEST IN ONE CYCLE USING THE "TCM
         UL" INSTRUCTION
*        WREG IS A DUMMY REGISTER.
       SPF14 TCMLS & AB & PCUNOP & AM2904 ,OECT & CARRYCTL &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM WREG &
```

```
       TEST DIRIN UANEB & CJP JSUB
018F: C0 30 29 1F 30 09 F9 83 F3 43 01 93

*        MULT3 - REAL2L -> MULT3, LOAD MICRO STATUS REGISTER
       ALU REG SUBR & AB & OEY & PCUNOP & AM2904 ,,,,,,,CEU &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & ARAM REAL2L & BRAM MULT3 & TEST MICRO DLOAD &
       CONT
0190: C0 78 A9 1F 30 09 FB F7 E4 7E 00 00

*        MULT2 - REAL2I -> MULT2, LOAD MICRO STATUS REGISTER
       ALU REG SUBR & AB & OEY & PCUNOP & AM2904 ,,,,,,,CEU &
       CARRYCTL COEQST & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & ARAM REAL2I & BRAM MULT2 & TEST MICRO DLOAD &
       CONT
0191: C0 78 A9 1F 30 09 FB 6F EC 7E 00 00

*        MULT1 - REAL2M -> MULT1,TREG, JOIN LAST MULTIPLY OPER
         ATION SUBROUTINE.
       ALU REG SUBR & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQST & DATAPATH ,,LDTREG & MEMCONT &
       CONTROL ROM & RTB & ARAM REAL2M & BRAM MULT1 &
       TESTF $00 & CONT
0192: C0 78 B9 1F 30 09 FA E7 FC 0E 00 00

JSUB:  ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT & CONTROL & CONT
0193: 40 62 29 1F 30 09 78 07 F0 0E 00 00

*        MULT1 -> QREG
       ALU QPT PASSR & AQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,LDI & MEMCONT &
       CONTROL ROM & ARAM MULT1 & IREG WREG XREG & CONT
0194: 40 33 69 3F 30 09 F8 2F F0 0E 00 43

*        NORMALISE AND SKIP NEXT 3 INSTRUCTION IF NOT REQUIRED

       SPF14 SLN & AB & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
       BRAM WREG & TEST DIRIN UAGEB & CJP PUTEXP
0195: C0 40 29 1F 30 09 F9 83 F3 B3 01 99

*        SHIFT MULT3 LEFT, LOAD I-REG
       ALU LUR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ,,,,,,,,LDI & MEMCONT &
       CONTROL ROM & RTB &
       BRAM MULT3 & CNTLB $10 & IREG MULT1+1 MULT2 & CONT
0196: C0 4A 29 3F 30 09 F3 87 F0 0E 00 66

*        SHIFT I-REG PORT LEFT, DECREMENT I-REG, LOOP BACK IF
         A<>B
       ALU LUR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ,,,ENCTR & MEMCONT & CONTROL &
       CNTLB $19 & RTB & TESTF AEQB & CJP *
0197: C0 4A 21 1F 30 09 70 CF F0 03 01 97

*        EXP1 - 1 -> EXP1
       ALU REG SUBR & DAB & OEY & PCUNOP & AM2904 &
```

```
       CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & BRAM EXP1 & IMMD 1 & CONT
0198: C2 78 A9 1F 30 08 FD 87 F4 0E 00 01

*        SP -> MAR, EXP1 -> DREG, LOAD I-REG
PUTEXP:  ALU YBUS PASSR & AQ & OEY & PCUSP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD LDI &
       MEMCONT REQB & CONTROL ROM & ARAM EXP1 &
       IREG MULT1+1 MULT3 & CONT
0199: 40 63 69 FF 32 69 F8 5F F0 0E 00 76

*        WRITE NEXT WORD, I-REG PORT -> DREG, SP-2 -> MAR
*        DECREMENT I-REG AND LOOP IF A <> B
       ALU YBUS PASS & AB & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,ENCTR ,,LDMAR LDD &
       MEMCONT REQB MREQ ,WRITE & CONTROL & RTB &
       TESTF AEQB & CJP *
019A: C0 62 21 DF 98 7F 78 07 F0 03 01 9A

*        WRITE LAST WORD, PC+2 -> MAR
       ALU YBUS PASS & AB & PCUNEXT & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JMAP
019B: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
*
* SUBROUTINE TO PERFORM N CYCLES OF A 48-BIT MULTIPLY.
* N-1 IS IN THE AM2910 COUNTER ON ENTRY AND N <= 16.
* QREG = MULTIPLIER (16 BITS)
* MULT1-MULT3 = RESULT (48 BITS)
* REAL2M-REAL2L = MULTIPLICAND (48 BITS)
*
*        "TCMUL" TO SHIFT QREG DOWN, WREG IS A DUMMY.
*        SKIP THE 48-BIT REGISTER ADD CODE IF Z IS LOW.
MULT16:  SPF14 TCMUL & AB & OEY & PCUNOP &
       AM2904 ,OECT & CARRYCTL & DATAPATH & MEMCONT &
       CONTROL ROM & RTB & BRAM WREG & TEST DIRIN UANEB &
       CJP MRENT
019C: C0 10 29 1F 30 09 F9 83 F3 43 01 A0

*        ARRIVED HERE IF ADD NEEDED.
*        MULT3 + REAL2L -> MULT3, LOAD MICRO STATUS REGISTER.
       ALU REG ADD & AB & OEY & PCUNOP & AM2904 ,,,,,,,CEU &
       CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
       ARAM REAL2L & BRAM MULT3 & TESTF $00 & CONT
019D: C0 79 A9 1F 30 09 FB F7 E0 0E 00 00

*        MULT2 + REAL2I + MICRO CARRY -> MULT2, LOAD MICRO STA
         TUS REGISTER
       ALU REG ADD & AB & OEY & PCUNOP & AM2904 ,,,,,,,CEU &
       CARRYCTL COEQST & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & ARAM REAL2I & BRAM MULT2 & TESTF $00 & CONT
019E: C0 79 A9 1F 30 09 FB 6F EC 0E 00 00

*        MULT1,TREG + REAL2M + MICRO CARRY -> MULT1
       ALU REG ADD & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQST & DATAPATH ,,LDTREG & MEMCONT &
```

```
       CONTROL ROM & RTB & ARAM REAL2M & BRAM MULT1 &
       TESTF $00 & CONT
019F: C0 79 B9 1F 30 09 FA E7 FC 0E 00 00

*        NOW SHIFT MULT1-MULT3 DOWN
*        SHIFT TREG(COPY OF MULT1) -> XREG TO LOAD MSR, LOAD I
         -REG
MRENT:  ALU LDR PASSR & DAQ & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ,ENTREG ,,,,,,LDI & MEMCONT &
       CONTROL ROM & RTB & BRAM XREG & CNTLB $02 &
       TEST MACHINE DLOAD & IREG MULT3 MULT1 & CONT
01A0: C2 0B 49 3F 30 09 F2 17 F2 7E 00 57

*        SHIFT SIGN INTO MULT1,TREG, INC I-REG
       ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ,,LDTREG ENCTR INC & MEMCONT &
       CONTROL & RTB & CNTLB $0E & CONT
01A1: C0 0A 35 1F 30 09 70 77 F0 0E 00 00

*        SHIFT MULT2, INCREMENT I-REG
       ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH ,,,ENCTR INC & MEMCONT & CONTROL &
       RTB & CNTLB $09 & CONT
01A2: C0 0A 25 1F 30 09 70 4F F0 0E 00 00

*        SHIFT MULT3, LOOP BACK
       ALU LDR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH & MEMCONT & CONTROL & RTB &
       CNTLB $04 & RPCT MULT16
01A3: C0 0A 29 1F 30 09 70 27 F0 09 01 9C

*        RETURN
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB & CONTROL & RTN
01A4: 48 62 29 1F 30 29 78 07 F0 0A 00 00

*
* SUBROUTINE USED BY "DVW" AND "MOD"
*
*        READ TOP STACK WORD (DIVISOR), SP+2 -> MAR
*        LOAD AM2910 WITH 14 (NO. OF DIVIDES - 1)
DIVSUB:  ALU YBUS PASS & AQ & PCUPOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       LDCT 14
01A5: 40 62 69 9F 18 7B 78 07 F0 0C 00 0E

*         ZREG (DIVISOR) -> XREG,TREG, READ BOTTOM STACK WORD
*        PC -> MAR, SKIP NEXT INSTRUCTION IF >= 0
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,LDTREG ,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
       TEST DIRIN SAGRB & CJP TSTDVD
01A6: C2 7B 79 8F 30 3B FA 03 F3 03 01 A8

*        NEGATE XREG (DIVISOR), EXCEPTION IF = 0
       ALU REG COMPLS & AB & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & BRAM XREG & TEST DIRIN UAEQB & CJPPX OVFLOWEXP
```

```
01A7: C0 7A A9 1F 30 09 FA 03 F7 5B XX XX

*        ZREG (DIVIDEND) -> QREG, SET MSR N, LOAD I-REG
TSTDVD:  ALU QPT PASSR & DAQ & PCUNOP & AM2904 ,,,,ES ,CEM &
       CARRYCTL & DATAPATH ,,,,,,,,LDI ENZ0 & MEMCONT &
       CONTROL & IREG XREG WREG & TEST MACHINE DLOAD & CONT
01A8: 42 33 69 2F 30 09 78 07 52 7E 00 34

*        TREG EXOR QREG -> YBUS (GETS SIGN BIT OF RESULT)
*        LOAD MSR N AND SKIP NEXT INSTRUCTION IF MSR N = 0, PC
          -> MAR
       ALU YBUS EXOR & DAQ & OEY & PCUNOP &
       AM2904 ,OECT ,,ES ,CEM & CARRYCTL &
       DATAPATH ,ENTREG ,,,,LDMAR & MEMCONT REQB & CONTROL &
       TEST MACHINE POSITIVE & CJP FIRSTOP
01A9: 42 65 C9 9F 30 29 78 03 52 E3 01 AB

*        NEGATE QREG
       ALU QPT COMPLS & AQ & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB & CONTROL &
       CONT
01AA: 40 32 E9 1F 30 29 78 07 F4 0E 00 00

*        FIRST DIVIDE OP, READ INSTR N+2.
FIRSTOP:  SPF14 DLN & AB & OEY & PCUNOP & AM2904 SHIFTEN &
       CARRYCTL & DATAPATH & MEMCONT REQB MREQ & CONTROL &
       RTB & CNTLB $1F & RTN
01AB: C8 50 29 1F 30 3B 70 FF F0 0A 00 00

*
* CODE USED BY "CRL" AND "CRGE"
*
*        READ TOP STACK MANTISSA M.S., SP+8 -> MAR, 1 -> YREG
CRLESS:  ALU LUR LOW & AQ & OEY & PCU ,,PCUDA A1 B1 &
       AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM YREG &
       CNTLB $11 & IMMD 8 & CONT
01AC: C0 4C 69 9F 52 7A F2 8F F0 0E 00 08

*        READ BOTTOM STACK MANTISSA M.S.
*        ZREG(TOP MANTISSA M.S.) -> REAL2M, SP-2 -> MAR (BUT N
          OT SP)
       ALU REG PASSR & DAQ & OEY & PCU QEU SUB PCUAB A4 B1 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM REAL2M &
       CONT
01AD: C2 7B 69 8E 98 7B FE 07 F0 0E 00 00

*        ZREG (BOTTOM MANTISSA M.S.) -> REAL1M, READ TOP EXPON
          ENT
*        SP+4 -> Q,MAR.
       ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A5 B1 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM REAL1M &
       CONT
01AE: C2 7B 69 0F 1A 7B FC 07 F0 0E 00 00

*        REAL1M EXOR REAL2M -> YBUS (TESTS SIGN BITS SAME OR N
```

```
          OT)
*         SKIP NEXT INSTRUCTION IF SIGN BIT = 0, Q+2 -> MAR.
        ALU YBUS EXOR & AB & OEY & PCU QEU ,PCUAB A4 B1 &
        AM2904 ,OECT & CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB & CONTROL ROM & RTB & ARAM REAL1M &
        BRAM REAL2M & TEST DIRIN POSITIVE & CJP *+18
01AF: C0 65 A9 9E 18 69 FE 43 F3 E3 01 B2

*         ARRIVED HERE IF SIGNS DIFFERENT, JUMP TO CHANGE RESUL
          T IF REAL1M -VE
*         PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL1M &
        TEST DIRIN NEGATIVE & CJP CHNGRSLT
01B0: C0 62 29 9F 30 29 FC 03 F3 F3 01 BF

*         READ THE NEXT INSTRUCTION AND THEN WRITE THE RESULT T
          O THE STACK
RNIWRTS:  ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        JUMP WRTS
01B1: 48 62 29 9F 18 7B 78 07 F0 03 01 15

*         ZREG (TOP EXPONENT) -> EXP2, READ BOTTOM EXPONENT, R6
          +2 -> MAR.
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM EXP2 &
        CONT
01B2: C2 7B 69 8F 19 BB FF 87 F0 0E 00 00

*         ZREG (EXP1) - EXP2 -> YBUS, JUMP TO TEST MANTISSA COD
          E IF =
*         R6+2 -> MAR, READ TOP MANTISSA I.S.
        ALU REG SUBR & DAB & PCU ,,PCUAB A4 B6 &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM EXP2 & TEST DIRIN SAEQB &
        CJP TESTMANT
01B3: C2 78 A9 8F 19 BB FF 83 F7 53 01 B9

*         TEST REAL1M, IF +VE SKIP NEXT 2 INSTRUCTIONS, PC -> M
          AR.
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL1M &
        TEST DIRIN POSITIVE & CJP *+18
01B4: C0 62 29 9F 30 29 FC 03 F3 E3 01 B7

*         EXP1 - EXP2 -> YBUS, JUMP TO CHANGE RESULT IF <0, REA
          D INSTR N+2
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM EXP2 & TEST DIRIN SALSB &
        CJP CHNGRSLT
01B5: C0 62 29 1F 30 3B FF 83 F3 33 01 BF
```

```
*         ARRIVED HERE IF EXP1 > EXP2, RESULT IS FALSE, SP -> M
          AR.
        ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMP WRTS
01B6: 48 62 29 9F 18 69 78 07 F0 03 01 15

*         EXP1-EXP2 -> YBUS, JUMP TO CHANGE RESULT IF >, READ I
          NSTR N+2
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM EXP2 & TEST DIRIN SAGRB &
        CJP CHNGRSLT
01B7: C0 62 29 1F 30 3B FF 83 F3 03 01 BF

*         ARRIVED HERE IF EXP1 < EXP2 AND REAL1M +VE, RESULT =
          FALSE, SP -> MAR
        ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMP WRTS
01B8: 48 62 29 9F 18 69 78 07 F0 03 01 15

*         ARRIVED HERE IF WE MUST TEST THE MANTISSAS.
*         ZREG (TOP MANTISSA I.S.) -> REAL2I, READ TOP MANTISSA
           L.S.
*         Q-4 -> Q,MAR.
TESTMANT:  ALU REG PASSR & DAQ & OEY & PCU QEU SUB PCUAQ A5 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM REAL2I &
        CONT
01B9: C2 7B 69 8E 8A 3B FE 87 F0 0E 00 00

*         ZREG (TOP MANTISSA L.S.) -> REAL2L
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL2L & CONT
01BA: C2 7B 69 0F 30 29 FF 07 F0 0E 00 00

*         COMPARE REAL1M AND REAL2M, JUMP TO END TEST IF <>, SE
          T MSR
*         READ BOTTOM MANTISSA I.S., Q+2 -> MAR.
        ALU YBUS SUBS & AB & PCU QEU ,PCUAQ A4 B4 &
        AM2904 ,OECT EZ EC ES EOVR & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & ARAM REAL1M & BRAM REAL2M &
        TEST DIRIN UANEB & CJP ENDTEST
01BB: C0 61 29 8E 09 3B FE 40 17 43 01 BE

*         COMPARE ZREG (BOTTOM MANTISSA I.S.) WITH REAL2I
*         JUMP TO END TEST IF <>, SET MSR
*         READ BOTTOM MANTISSA L.S.
        ALU YBUS SUBS & DAB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM REAL2I & TEST DIRIN UANEB &
        CJP ENDTEST
01BC: C2 61 29 0F 30 3B FE 80 17 43 01 BE
```

```
*         COMPARE ZREG (BOTTOM MANTISSA L.S.) AND REAL2L, SET M
          SR
        ALU YBUS SUBS & DAB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL ROM & RTB &
        BRAM REAL2L & TEST MACHINE DLOAD & CONT
01BD: C2 61 29 0F 30 09 FF 00 16 7E 00 00

*         END OF TEST, IF MSR >= THEN RESULT IS FALSE, PC -> MA
          R
ENDTEST:  ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL & TEST MACHINE UAGEB & CJP RNIWRTS
01BE: 40 62 29 9F 30 29 78 03 F2 B3 01 B1

*         CHANGE RESULT.
*         TREG (1) - WREG (0 OR 1) -> DREG (CHANGES 0-> 1 OR 1-
          > 0)
*         SP+2 -> MAR, READ INSTR N+2
CHNGRSLT:  ALU YBUS SUBS & AB & OEY & PCUPOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & ARAM YREG &
        BRAM WREG & JUMP WRTS
01BF: C8 61 29 DF 18 7B F9 AF F4 03 01 15

*
* CODE USED BY "CRG" AND "CRLE"
*
*         READ TOP STACK MANTISSA M.S., SP+8 -> MAR, 1 -> YREG
CRGREATER:  ALU LUR LOW & AQ & OEY & PCU ,,PCUDA A1 B1 &
        AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM YREG &
        CNTLB $11 & IMMD 8 & CONT
01C0: C0 4C 69 9F 52 7A F2 8F F0 0E 00 08

*         READ BOTTOM STACK MANTISSA M.S.
*         ZREG(TOP MANTISSA M.S.) -> REAL2M, SP-2 -> MAR (BUT N
          OT SP)
        ALU REG PASSR & DAQ & OEY & PCU QEU SUB PCUAB A4 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM REAL2M &
        CONT
01C1: C2 7B 69 8E 98 7B FE 07 F0 0E 00 00

*         ZREG (BOTTOM MANTISSA M.S.) -> REAL1M, READ TOP EXPON
          ENT
*         SP+4 -> Q,MAR.
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A5 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM REAL1M &
        CONT
01C2: C2 7B 69 0F 1A 7B FC 07 F0 0E 00 00

*         REAL1M EXOR REAL2M -> YBUS (TESTS SIGN BITS SAME OR N
          OT)
*         SKIP NEXT INSTRUCTION IF SIGN BIT = 0, Q+2 -> MAR.
        ALU YBUS EXOR & AB & OEY & PCU QEU ,PCUAB A4 B1 &
        AM2904 ,OECT & CARRYCTL & DATAPATH ,,,,,,LDMAR &
```

```
        MEMCONT REQB & CONTROL ROM & RTB & ARAM REAL1M &
        BRAM REAL2M & TEST DIRIN POSITIVE & CJP *+18
01C3: C0 65 A9 9E 18 69 FE 43 F3 E3 01 C6

*         ARRIVED HERE IF SIGNS DIFFERENT, JUMP TO CHANGE RESUL
          T IF REAL1M +VE
*         PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL1M &
        TEST DIRIN POSITIVE & CJP CHNGRSLT1
01C4: C0 62 29 9F 30 29 FC 03 F3 E3 01 D3

*         READ THE NEXT INSTRUCTION AND THEN WRITE THE RESULT T
          O THE STACK
RNIWRTS1:  ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        JUMP WRTS
01C5: 48 62 29 9F 18 7B 78 07 F0 03 01 15

*         ZREG (TOP EXPONENT) -> EXP2, READ BOTTOM EXPONENT, R6
          +2 -> MAR.
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM EXP2 &
        CONT
01C6: C2 7B 69 8F 19 BB FF 87 F0 0E 00 00

*         ZREG (EXP1) - EXP2 -> YBUS, JUMP TO TEST MANTISSA COD
          E IF =
*         R6+2 -> MAR, READ TOP MANTISSA I.S.
        ALU REG SUBR & DAB & PCU ,,PCUAB A4 B6 &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM EXP2 & TEST DIRIN SAEQB &
        CJP TESTMANT1
01C7: C2 78 A9 8F 19 BB FF 83 F7 53 01 CD

*         TEST REAL1M, IF +VE SKIP NEXT 2 INSTRUCTIONS, PC -> M
          AR.
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL1M &
        TEST DIRIN POSITIVE & CJP *+18
01C8: C0 62 29 9F 30 29 FC 03 F3 E3 01 CB

*         EXP1 - EXP2 -> YBUS, JUMP TO CHANGE RESULT IF >0, REA
          D INSTR N+2
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM EXP2 & TEST DIRIN SAGRB &
        CJP CHNGRSLT1
01C9: C0 62 29 1F 30 3B FF 83 F3 03 01 D3

*         ARRIVED HERE IF EXP1 > EXP2, RESULT IS FALSE, SP -> M
          AR.
        ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
```

```
        JUMP WRTS
01CA: 48 62 29 9F 18 69 78 07 F0 03 01 15

*         EXP1-EXP2 -> YBUS, JUMP TO CHANGE RESULT IF <, READ I
          NSTR N+2
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM EXP2 & TEST DIRIN SALSB &
        CJP CHNGRSLT1
01CB: C0 62 29 1F 30 3B FF 83 F3 33 01 D3

*         ARRIVED HERE IF EXP1 < EXP2 AND REAL1M +VE, RESULT =
          FALSE, SP -> MAR
        ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMP WRTS
01CC: 48 62 29 9F 18 69 78 07 F0 03 01 15

*         ARRIVED HERE IF WE MUST TEST THE MANTISSAS.
*         ZREG (TOP MANTISSA I.S.) -> REAL2I, READ TOP MANTISSA
           L.S.
*         Q-4 -> Q,MAR.
TESTMANT1:  ALU REG PASSR & DAQ & OEY &
        PCU QEU SUB PCUAQ A5 & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM REAL2I & CONT
01CD: C2 7B 69 8E 8A 3B FE 87 F0 0E 00 00

*         ZREG (TOP MANTISSA L.S.) -> REAL2L
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL2L & CONT
01CE: C2 7B 69 0F 30 29 FF 07 F0 0E 00 00

*         COMPARE REAL1M AND REAL2M, JUMP TO END TEST IF <>, SE
          T MSR
*         READ BOTTOM MANTISSA I.S., Q+2 -> MAR.
        ALU YBUS SUBS & AB & PCU QEU ,PCUAQ A4 B4 &
        AM2904 ,OECT EZ EC ES EOVR & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & ARAM REAL1M & BRAM REAL2M &
        TEST DIRIN UANEB & CJP ENDTEST1
. 01CF: C0 61 29 8E 09 3B FE 40 17 43 01 D2

*         COMPARE ZREG (BOTTOM MANTISSA I.S.) WITH REAL2I
*         JUMP TO END TEST IF <>, SET MSR
*         READ BOTTOM MANTISSA L.S.
        ALU YBUS SUBS & DAB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM REAL2I & TEST DIRIN UANEB &
        CJP ENDTEST1
01D0: C2 61 29 0F 30 3B FE 80 17 43 01 D2

*         COMPARE ZREG (BOTTOM MANTISSA L.S.) AND REAL2L, SET M
          SR
        ALU YBUS SUBS & DAB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
```

```
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL ROM & RTB &
       BRAM REAL2L & TEST MACHINE DLOAD & CONT
01D1: C2 61 29 0F 30 09 FF 00 16 7E 00 00

*        END OF TEST, IF MSR <= THEN RESULT IS FALSE, PC -> MA
         R
ENDTEST1:  ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
       CONTROL & TEST MACHINE UALEB & CJP RNIWRTS1
01D2: 40 62 29 9F 30 29 78 03 F2 D3 01 C5

*        CHANGE RESULT.
*        TREG (1) - WREG (0 OR 1) -> DREG (CHANGES 0-> 1 OR 1-
         > 0)
*        SP+2 -> MAR, READ INSTR N+2
CHNGRSLT1:  ALU YBUS SUBS & AB & OEY & PCUPOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & ARAM YREG &
       BRAM WREG & JUMP WRTS
01D3: C8 61 29 DF 18 7B F9 AF F4 03 01 15

*
* SUBROUTINE TO COMPARE TWO REAL NUMBERS ON THE STACK.
* USED BY "CRE","CRNE"
*
*        READ FIRST TOP SET WORD, SP+8 -> R7,MAR, 1 -> XREG
CMPREAL:  ALU LUR LOW & DAQ & OEY & PCU ,,PCUDA A1 B7 &
       AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & CNTLB $11 & RTB &
       BRAM VREG & IMMD 8 & CONT
01D4: C2 4C 69 9F 53 FA F3 8F F0 0E 00 08

*        READ FIRST BOTTOM SET WORD, R6+2 -> MAR, LOAD COUNTER
          WITH 6 AND PUSH
*        ZREG (FIRST TOP SET WORD) -> WREG
       ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       PHLC 2
01D5: CA 7B 69 8F 19 BB F9 87 F0 04 00 02

*        NEXT TWO WORDS ARE A LOOP
*        READ NEXT TOP SET WORD, R7 + 2 -> MAR,
*        ZREG (CURRENT BOTTOM SET WORD) - WREG (CURRENT TOP SE
         T WORD) -> YBUS
*        EXIT LOOP IF NOT EQUAL
       ALU YBUS SUBS & DAB & OEY & PCU ,,PCUAB A4 B7 &
       AM2904 ,OECT & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM WREG & TEST DIRIN UANEB &
       CJPP RETURN1
01D6: C2 61 29 8F 19 FB F9 83 F7 4B 01 D9

*        READ NEXT BOTTOM SET WORD, R6 + 2 -> MAR, ZREG (CURRE
         NT TOP SET WORD) -
       ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
```

```
       RFCT
01D7: C2 7B 69 8F 19 BB F9 87 F0 08 00 00

*        ARRIVED HERE IF EQUAL SO FAR.
*        ZREG (LAST BOTTOM SET WORD) - WREG (LAST TOP SET WORD
         ) -> YBUS
*        SKIP NEXT INSTRUCTION IF =, PC -> MAR
       ALU YBUS SUBS & DAB & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
       TEST DIRIN UAEQB & CJP *+12
01D8: C2 61 29 8F 30 29 F9 83 F7 53 01 DA

*        ARRIVED HERE IF NOT EQUAL.
*        -YREG + VREG (1) -> DREG (CHANGES 0->1 or 1 -> 0)
*        PC -> MAR
RETURN1:  ALU YBUS SUBR & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB & CONTROL ROM & RTB & ARAM YREG &
       BRAM VREG & CONT
01D9: C0 60 A9 DF 30 29 FB AF F4 0E 00 00

*        SP + 14 -> MAR, READ INSTR N+2
       ALU YBUS PASS & AB & OEY & PCU ,,PCUDA A1 B1 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL & IMMD 14 & CONT
01DA: 40 62 29 9F 52 7A 78 07 F0 0E 00 0E

*        WRITE RESULT TO STACK
       ALU YBUS PASS & AB & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JMAP
01DB: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
*
* SUBROUTINE TO COMPARE TWO SETS ON THE STACK.
* USED BY "EQS","SNE"
*
*        READ FIRST TOP SET WORD, SP+16 -> R7,MAR, 1 -> XREG
CMPSET:  ALU LUR LOW & DAQ & OEY & PCU ,,PCUDA A1 B7 &
       AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & CNTLB $11 & RTB &
       BRAM VREG & IMMD 16 & CONT
01DC: C2 4C 69 9F 53 FA F3 8F F0 0E 00 10

*        READ FIRST BOTTOM SET WORD, R6+2 -> MAR, LOAD COUNTER
          WITH 6 AND PUSH
*        ZREG (FIRST TOP SET WORD) -> WREG
       ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       PHLC 6
01DD: CA 7B 69 8F 19 BB F9 87 F0 04 00 06

*        NEXT TWO WORDS ARE A LOOP
*        READ NEXT TOP SET WORD, R7 + 2 -> MAR,
*        ZREG (CURRENT BOTTOM SET WORD) - WREG (CURRENT TOP SE
```

```
          T WORD) -> YBUS
*         EXIT LOOP IF NOT EQUAL
        ALU YBUS SUBS & DAB & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM WREG & TEST DIRIN UANEB &
        CJPP RETURN
01DE: C2 61 29 8F 19 FB F9 83 F7 4B 01 E1

*         READ NEXT BOTTOM SET WORD, R6 + 2 -> MAR, ZREG (CURRE
          NT TOP SET WORD) -
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        RFCT
01DF: C2 7B 69 8F 19 BB F9 87 F0 08 00 00

*         ARRIVED HERE IF EQUAL SO FAR.
*         ZREG (LAST BOTTOM SET WORD) - WREG (LAST TOP SET WORD
          ) -> YBUS
*         SKIP NEXT INSTRUCTION IF =, PC -> MAR
        ALU YBUS SUBS & DAB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN UAEQB & CJP *+12
01E0: C2 61 29 8F 30 29 F9 83 F7 53 01 E2

*         ARRIVED HERE IF NOT EQUAL.
*         -YREG + VREG (1) -> DREG (CHANGES 0->1 or 1 -> 0)
*         PC -> MAR.
RETURN:   ALU YBUS SUBR & AB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & RTB & ARAM YREG &
        BRAM VREG & IMMD 30 & CONT
01E1: C0 60 A9 DF 30 28 FB AF F4 0E 00 1E

*         SP + 30 -> MAR, READ INSTR N+2
ABC:    ALU YBUS PASS & AB & OEY & PCU ,,PCUDA A1 B1 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL & IMMD 30 & CONT
01E2: 40 62 29 9F 52 7A 78 07 F0 0E 00 1E

*         WRITE RESULT TO STACK
        ALU YBUS PASS & AB & OEY & PCUNEXT & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JMAP
01E3: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* CODE USED BY "SGE"
*
*         READ FIRST TOP SET WORD
*         SP+16 -> MAR, 1 -> VREG
SETGE:    ALU LUR LOW & AQ & OEY & PCU ,,PCUDA A1 B7 &
        AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $11 & IMMD 16 & CONT
01E4: C0 4C 69 9F 53 FA F1 8F F0 0E 00 10
```

```
*         READ FIRST BOTTOM SET WORD
*         R6+2 -> MAR, LOAD AM2910 COUNTER WITH 6 AND PUSH
*         ZREG (FIRST TOP SET WORD) -> WREG
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        PHLC 6
01E5: CA 7B 69 8F 19 BB F9 87 F0 04 00 06

*         READ NEXT TOP SET WORD, R7+2 -> MAR
*         NOT ZREG (CURRENT BOTTOM SET WORD) AND WREG (CURRENT
          TOP SET WORD) -> Y
*         EXIT LOOP IF <> 0
        ALU YBUS NOTRS & DAB & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 ,OECT & CARRYCTL &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM WREG & TEST DIRIN UANEB &
        CJPP SGENE
01E6: C2 64 A9 8F 19 FB F9 83 F3 4B 01 E9

*         READ NEXT BOTTOM SET WORD
*         R6+2 -> MAR, ZREG (CURRENT TOP SET WORD) -> WREG
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        RFCT
01E7: C2 7B 69 8F 19 BB F9 87 F0 08 00 00

*         ZERO SO FAR
*         NOT ZREG (LAST BOTTOM SET WORD) AND WREG (LAST TOP SE
          T WORD) -> YBUS
*         SKIP NEXT INSTRUCTION IF = 0, PC -> MAR
        ALU YBUS NOTRS & DAB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN UAEQB & CJP ABC
01E8: C2 64 A9 8F 30 29 F9 83 F3 53 01 E2

*         NOT EQUAL, 0 -> DREG
SGENE:   ALU YBUS LOW & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL & JUMP ABC
01E9: 48 64 69 DF 30 29 78 07 F0 03 01 E2

*
* CODE USED BY "SLE"
*
*         READ FIRST TOP SET WORD
*         SP+16 -> MAR, 1 -> VREG
SETLE:   ALU LUR LOW & AQ & OEY & PCU ,,PCUDA A1 B7 &
        AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $11 & IMMD 16 & CONT
01EA: C0 4C 69 9F 53 FA F1 8F F0 0E 00 10

*         READ FIRST BOTTOM SET WORD
*         R6+2 -> MAR, LOAD AM2910 COUNTER WITH 6 AND PUSH
```

```
*         ZREG (FIRST TOP SET WORD) -> WREG
        ALU REG COMPLR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        PHLC 6
01EB: CA 7B E9 8F 19 BB F9 87 F0 04 00 06

*         READ NEXT TOP SET WORD, R7+2 -> MAR
*         NOT ZREG (CURRENT BOTTOM SET WORD) AND WREG (CURRENT
          TOP SET WORD) -> Y
*         EXIT LOOP IF <> 0
        ALU YBUS AND & DAB & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 ,OECT & CARRYCTL &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM WREG & TEST DIRIN UANEB &
        CJPP SLENE
01EC: C2 66 29 8F 19 FB F9 83 F3 4B 01 EF

*         READ NEXT BOTTOM SET WORD
*         R6+2 -> MAR, ZREG (CURRENT TOP SET WORD) -> WREG
        ALU REG COMPLR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        RFCT
01ED: C2 7B E9 8F 19 BB F9 87 F0 08 00 00

*         ZERO SO FAR
*         NOT ZREG (LAST BOTTOM SET WORD) AND WREG (LAST TOP SE
          T WORD) -> YBUS
*         SKIP NEXT INSTRUCTION IF = 0, PC -> MAR
        ALU YBUS AND & DAB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN UAEQB & CJP ABC
01EE: C2 66 29 8F 30 29 F9 83 F3 53 01 E2

*         NOT EQUAL, 0 -> DREG
SLENE:   ALU YBUS LOW & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL & JUMP ABC
01EF: 48 64 69 DF 30 29 78 07 F0 03 01 E2

*
* SUBROUTINE TO READ TWO ADDRESSES OFF THE TOP OF THE STACK.
* USED BY "CSL","CSLE","CSG","CSGE"
*
*         READ SOURCE ADDRESS, SP+2 -> MAR, 1 -> DREG,YREG
RSADA:   ALU REG PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM YREG &
        IMMD 1 & CONT
01F0: D2 7B 69 DF 18 7A FA 87 F0 0E 00 01

*         READ DESTINATION ADDRESS, ZREG (SOURCE ADDRESS) -> R6
           OF PCU,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & RTN
```

```
01F1: 48 62 29 8F 7D BB 78 07 F0 0A 00 00

*
* CODE USED BY "CSL".
*
*         READ FIRST DESTINATION BYTE, ZREG (FIRST SOURCE BYTE)
           -> XREG, R6 + 1 -
CMPLSTR:  ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & CONT
01F2: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*         READ NEXT SOURCE BYTE, ZREG (CURRENT DESTINATION BYTE
          ) - XREG (CURRENT
*         JUMP OUT OF LOOP IF NOT EQUAL, R7 + 1 -> MAR
RNSB:   ALU QPT SUBS & DAB & PCU ,,PCUAB A2 B7 &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & TEST DIRIN UANEB & CJP LEQ
01F3: C2 31 29 8F 15 F9 FA 03 F7 43 01 F6

*         READ NEXT DESTINATION BYTE, ZREG (CURRENT SOURCE BYTE
          ) -> XREG, R6 + 1
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & CONT
01F4: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*         INCREMENT WREG AND LOOP BACK IF < 0
        SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM WREG & TEST DIRIN SALSB & CJP RNSB
01F5: C0 20 29 1F 30 29 F9 83 F3 33 01 F3

*         QREG -> YBUS, IF < 0 THEN SKIP NEXT INSTRUCTION, SP -
          > MAR
LEQ:    ALU YBUS PASS & AQ & OEY & PCUSP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL & TEST DIRIN SALSB & CJP WRJMP
01F6: 40 62 69 9F 32 69 78 03 F3 33 01 F8

*         0 -> DREG
        ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL & CONT
01F7: 40 64 29 5F 30 29 78 07 F0 0E 00 00

*         WRITE RESULT, PC+2 -> MAR
WRJMP:   ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
01F8: 48 62 29 9F 18 3F 78 07 F0 03 01 00

*
* CODE USED BY "CSG".
*
```

```
*         READ FIRST DESTINATION BYTE, ZREG (FIRST SOURCE BYTE)
           -> XREG, R6 + 1 -
CMPLSTR1:   ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & CONT
01F9: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*         READ NEXT SOURCE BYTE, ZREG (CURRENT DESTINATION BYTE
          ) - XREG (CURRENT
*         JUMP OUT OF LOOP IF NOT EQUAL, R7 + 1 -> MAR
RNSB1:   ALU QPT SUBS & DAB & PCU ,,PCUAB A2 B7 &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & TEST DIRIN UANEB & CJP LEQ1
01FA: C2 31 29 8F 15 F9 FA 03 F7 43 01 FD

*         READ NEXT DESTINATION BYTE, ZREG (CURRENT SOURCE BYTE
          ) -> XREG, R6 + 1
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & CONT
01FB: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*         INCREMENT WREG AND LOOP BACK IF < 0
        SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM WREG & TEST DIRIN SALSB & CJP RNSB1
01FC: C0 20 29 1F 30 29 F9 83 F3 33 01 FA

*         QREG -> YBUS, IF < 0 THEN SKIP NEXT INSTRUCTION, SP -
          > MAR
LEQ1:   ALU YBUS PASS & AQ & OEY & PCUSP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL & TEST DIRIN SAGRB & CJP WRJMP1
01FD: 40 62 69 9F 32 69 78 03 F3 03 01 FF

*         0 -> DREG
        ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL & CONT
01FE: 40 64 29 5F 30 29 78 07 F0 0E 00 00

*         WRITE RESULT, PC+2 -> MAR
WRJMP1:   ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
01FF: 48 62 29 9F 18 3F 78 07 F0 03 01 00

*
* CODE USED BY "CSGE".
*
*         READ FIRST DESTINATION BYTE, ZREG (FIRST SOURCE BYTE)
           -> XREG, R6 + 1 -
CMPLSTR2:   ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
```

```
       BRAM XREG & CONT
0200: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*        READ NEXT SOURCE BYTE, ZREG (CURRENT DESTINATION BYTE
         ) - XREG (CURRENT
*        JUMP OUT OF LOOP IF NOT EQUAL, R7 + 1 -> MAR
RNSB2:  ALU QPT SUBS & DAB & PCU ,,PCUAB A2 B7 &
       AM2904 ,OECT & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
       BRAM XREG & TEST DIRIN UANEB & CJP LEQ2
0201: C2 31 29 8F 15 F9 FA 03 F7 43 02 04

*        READ NEXT DESTINATION BYTE, ZREG (CURRENT SOURCE BYTE
         ) -> XREG, R6 + 1
       ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
       BRAM XREG & CONT
0202: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*        INCREMENT WREG AND LOOP BACK IF < 0
       SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM WREG & TEST DIRIN SALSB & CJP RNSB2
0203: C0 20 29 1F 30 29 F9 83 F3 33 02 01

*        QREG -> YBUS, IF < 0 THEN SKIP NEXT INSTRUCTION, SP -
         > MAR
LEQ2:  ALU YBUS PASS & AQ & OEY & PCUSP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
       CONTROL & TEST DIRIN SAGEB & CJP WRJMP2
0204: 40 62 69 9F 32 69 78 03 F3 23 02 06

*        0 -> DREG
       ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL & CONT
0205: 40 64 29 5F 30 29 78 07 F0 0E 00 00

*        WRITE RESULT, PC+2 -> MAR
WRJMP2:  ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JUMP RNI
0206: 48 62 29 9F 18 3F 78 07 F0 03 01 00

*
* CODE USED BY "CSLE".
*
*        READ FIRST DESTINATION BYTE, ZREG (FIRST SOURCE BYTE)
          -> XREG, R6 + 1 -
CMPLSTR3:  ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
       BRAM XREG & CONT
0207: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*        READ NEXT SOURCE BYTE, ZREG (CURRENT DESTINATION BYTE
         ) - XREG (CURRENT
```

```
*         JUMP OUT OF LOOP IF NOT EQUAL, R7 + 1 -> MAR
RNSB3:   ALU QPT SUBS & DAB & PCU ,,PCUAB A2 B7 &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & TEST DIRIN UANEB & CJP LEQ3
0208: C2 31 29 8F 15 F9 FA 03 F7 43 02 0B

*         READ NEXT DESTINATION BYTE, ZREG (CURRENT SOURCE BYTE
          ) -> XREG, R6 + 1
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A2 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM XREG & CONT
0209: C2 7B 69 8F 15 B9 FA 07 F0 0E 00 00

*         INCREMENT WREG AND LOOP BACK IF < 0
        SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM WREG & TEST DIRIN SALSB & CJP RNSB3
020A: C0 20 29 1F 30 29 F9 83 F3 33 02 08

*         QREG -> YBUS, IF < 0 THEN SKIP NEXT INSTRUCTION, SP -
          > MAR
LEQ3:   ALU YBUS PASS & AQ & OEY & PCUSP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL & TEST DIRIN SALEB & CJP WRJMP3
020B: 40 62 69 9F 32 69 78 03 F3 13 02 0D

*         0 -> DREG
        ALU YBUS LOW & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL & CONT
020C: 40 64 29 5F 30 29 78 07 F0 0E 00 00

*         WRITE RESULT, PC+2 -> MAR
WRJMP3:   ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
020D: 48 62 29 9F 18 3F 78 07 F0 03 01 00

*
* CODE TO COMPARE TWO STRUCTURES.
* USED BY "CSE","CSNE"
*
*         READ DESTINATION ADDRESS, ZREG (SOURCE ADDRESS) -> R6
          ,MAR
CESTRUCT:   ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & CONT
020E: 40 62 29 8F 7D BB 78 07 F0 0E 00 00

*         READ FIRST SOURCE WORD, ZREG (DESTINATION ADDRESS) ->
           R7,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & CONT
020F: 40 62 29 8F 7F FB 78 07 F0 0E 00 00
```

```
*         READ FIRST DESTINATION WORD, ZREG (FIRST SOURCE WORD)
           -> XREG, R6 + 2
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
        CONT
0210: C2 7B 69 8F 19 BB FA 07 F0 0E 00 00

*         NEXT 3 WORDS ARE A LOOP.
*         READ NEXT SOURCE WORD, ZREG (CURRENT DESTINATION WORD
          ) - XREG (CURRENT
*         JUMP OUT OF LOOP IF NOT EQUAL, R7 + 2 -> MAR
RNSW:   ALU YBUS SUBS & DAB & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM XREG & TEST DIRIN UANEB &
        CJP NOTEQ
0211: C2 61 29 8F 19 FB FA 03 F7 43 02 15

*         READ NEXT DESTINATION WORD, ZREG (NEXT SOURCE WORD) -
          > XREG, R6 + 2 ->
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
        CONT
0212: C2 7B 69 8F 19 BB FA 07 F0 0E 00 00

*         INCREMENT WREG AND LOOP BACK IF < 0
        SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM WREG & TEST DIRIN SALSB & CJP RNSW
0213: C0 20 29 1F 30 29 F9 83 F3 33 02 11

*         ARRIVED HERE IF STRUCTURES EQUAL, PC + 2 -> MAR, SKIP
           NEXT INSTRUCTION
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMP *+12
0214: 48 62 29 9F 18 29 78 07 F0 03 02 16

*         ARRIVED HERE IF NOT EQUAL, 1 - YREG -> DREG (CHANGES
          0 -> 1 or 1 -> 0),
NOTEQ:   ALU YBUS SUBS & DAB & OEY & PCUNEXT & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM YREG &
        IMMD 1 & CONT
0215: C2 61 29 DF 18 28 FA 87 F4 0E 00 01

*         READ INSTR N+2, SP -> MAR
        ALU YBUS PASS & AB & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        JUMP WRTS
0216: 48 62 29 9F 32 7B 78 07 F0 03 01 15

*
* CODE USED BY "FNV".
* DEALS WITH THE LAST 3 CASES (REAL, CLASS WORD + CLASS REAL)
*
```

```
*         8 -> XREG,R6 OF PCU
FUNCVL:   ALU REG PASSR & DAQ & OEY & PCU ,,PCUDZ A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ZZI & MEMCONT &
        CONTROL ROM & RTB & BRAM XREG & IMMD 8 & CONT
0217: D2 7B 69 1F 7D 88 FA 07 F0 0E 00 08

*         WREG ("KIND" - 8) - XREG (8) -> WREG, SKIP NEXT INSTR
          UCTION IF > 0, SP
        ALU REG SUBR & AB & OEY & PCU ,SUB PCUAB A6 B1 &
        AM2904 ,OECT & CARRYCTL COEQ1 & DATAPATH &
        MEMCONT REQB & CONTROL ROM & RTB & ARAM XREG &
        BRAM WREG & TEST DIRIN SAGRB & CJP CWORRL
0218: C0 78 A9 1F 9C 69 F9 A3 F7 03 02 1A

*         SIMPLE REAL.
*         PC + 2 -> MAR, READ INSTR N+2
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        JMAP
0219: 48 62 29 9F 18 3B 78 07 F0 02 00 00

*         0 -> DREG, SP + 8 -> MAR
CWORRL:   ALU YBUS LOW & AQ & OEY & PCU ,,PCUAB A6 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL & CONT
021A: 40 64 69 DF 1C 69 78 07 F0 0E 00 00

*         WREG ("KIND"-16) - XREG (8) -> WREG, SKIP 2 INSTRUCTI
          ON IF > 0
*         READ ADDRESS FROM STACK
        ALU REG SUBR & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & ARAM XREG & BRAM WREG &
        TEST DIRIN SAGRB & CJP CLASSR
021B: C0 78 A9 1F 30 3B F9 A3 F7 03 02 1E

*         CLASS WORD
*         CLEAR STORE, ZREG (ADDRESS) -> DREG, SP - 2 -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
021C: 42 63 69 CF 98 7F 78 07 F0 0E 00 00

*         WRITE ADDRESS BACK,PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMP RNI
021D: 48 62 29 9F 30 3F 78 07 F0 03 01 00

*         CLASS REAL.
*         ZREG (ADDRESS) -> DREG, SP - 8 -> MAR
CLASSR:   ALU YBUS PASSR & DAQ & OEY & PCU ,SUB PCUAB A6 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL &
021E: 4A 63 69 CF 9C 69 78 07 F0 03 02 1D

*
* CODE USED BY "FJP".
```

```
* FALSE JUMP.
*
*       PHREG+JOBATR -> MAR
FALSEJUMP:  ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG &
       AM2904 & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
       CONTROL & IMMD JOBATR & CONT
021F: 40 62 29 9E 56 28 78 07 F0 0E 00 12

*       ZREG (CONTINUE) -> YBUS, SKIP NEXT INSTRUCTION IF <>
        0
*       READ JOB, PC -> MAR.
       ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & TEST DIRIN UANEB &
       CJP FSP1
0220: 42 63 69 8F 30 3B 78 03 F3 43 02 22

*       ZREG (JOB) -> YBUS, EXCEPTION IF <> 0
       ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
       CONTROL & TEST DIRIN UANEB & CJPX EXCEPTION
0221: 42 63 69 0F 30 29 78 03 F3 43 XX XX

*       SP -> MAR, READ INSTR N+1 (JUST IN CASE)
FSP1:  ALU YBUS PASS & DAQ & OEY & PCUSP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & CONT
0222: 42 62 69 9F 32 7B 78 07 F0 0E 00 00

*       READ TEST FLAG, PC+2 -> MAR, WREG ("DISTANCE") - 4 ->
         TREG
       ALU REG SUBR & DAB & OEY & PCUNEXT & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ZZI ,LDTREG ,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       IMMD 4 & CONT
0223: D2 78 B9 9F 18 3A F9 87 F4 0E 00 04

*       ZREG (TEST FLAG) -> YBUS, EXIT IF NOT ZERO, SP+2 -> S
        P
       ALU YBUS PASSR & DAQ & PCUPOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
       CONTROL & TEST DIRIN SANEB & CJP RNI
0224: 42 63 69 0F 18 69 78 03 F3 43 01 00

*       MAKE THE JUMP
*       TREG ("DISTANCE"-4) + PC -> MAR
       ALU YBUS PASS & AB & PCU ,,PCUDA A0 B0 & AM2904 &
       CARRYCTL & DATAPATH ,ENTREG ,,,,LDMAR & MEMCONT REQB &
       CONTROL & JUMP START1
0225: 48 62 09 9F 50 29 78 07 F0 03 00 FD

*
* CODE USED BY "CJP"
*
*       READ INDEX, SP+2 -> SP, 2*ZREG ("MAX-MIN") -> YREG
CASEJMP:  ALU AUR PASSR & DAQ & OEY & PCUPOP &
       AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & CNTLB $10 & RTB &
```

```
       BRAM YREG & CONT
0226: C2 43 69 0F 18 7B F2 87 F0 0E 00 00

*           2*(ZREG(INDEX) - WREG("MIN")) -> WREG,TREG, PC+2 -
          > PC, EXCEPTION IF
       ALU AUR SUBS & DAB & OEY & AM2904 SHIFTEN OECT &
       CARRYCTL COEQ1 & PCUNEXT &
       DATAPATH ,,LDTREG ,,,,,,ENZ0 & MEMCONT & CONTROL ROM &
       CNTLB $10 & RTB &
       BRAM WREG & TEST DIRIN SALSB & CJPX RANGEEXP
0227: C2 41 39 0F 18 09 F1 83 F7 33 XX XX

*        WREG (2*(INDEX - "MIN") - YREG (2*"MAX-MIN") -> YBUS,
          PC+TREG -> MAR
*        EXCEPTION IF > 0
       ALU YBUS SUBS & AB & OEY & PCU ,,PCUDA A0 B0 &
       AM2904 ,OECT & CARRYCTL COEQ1 &
       DATAPATH ,ENTREG ,,,,LDMAR & MEMCONT REQB &
       CONTROL ROM & RTB & ARAM WREG & BRAM YREG &
       TEST DIRIN SAGRB & CJPX RANGEEXP
0228: C0 61 09 9F 50 29 FA 9B F7 03 XX XX

*        READ "DISTANCE"
       ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
       DATAPATH & MEMCONT REQB MREQ & CONTROL & CONT
0229: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*        ZREG ("DISTANCE") + PC -> MAR
       ALU YBUS PASS & AB & PCU ,,PCUDA A0 B0 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL & JUMP START1
022A: 48 62 29 8F 50 29 78 07 F0 03 00 FD

*
* CODE USED BY "IVR"
*
*        READ INSTR N+2, SP -> R6,MAR, 0-> DREG
INITVR:  ALU YBUS LOW & AQ & OEY & PCU ,,PCUZA A1 B6 &
       AM2904 & CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
       MEMCONT REQB MREQ & CONTROL & CONT
022B: 50 64 69 DF 43 BB 78 07 F0 0E 00 00

*        INCREMENT WREG, WRITE ZERO TO STACK, LOOP BACK IF < 0
          , R6 + 2 -> MAR
       SPF14 INCTWO & AB & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 ,OECT & CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM WREG & TEST DIRIN SALSB & CJP *
022C: C0 20 29 9F 19 BF F9 83 F3 33 02 2C

*        PC + 2 -> MAR, JUMP MAP
       ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & JMAP
022D: 48 62 29 9F 18 29 78 07 F0 02 00 00

*
* SUBROUTINE USED BY "ENT"
*
```

```
*         QREG (HEAP + "STACKLENGTH") - TREG (SP) -> YBUS, EXCE
          PTION IF >=, PC+2
ENTER:   ALU YBUS SUBR & DAQ & PCUNEXT & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,ENTREG ,,,,LDMAR &
        MEMCONT REQB & CONTROL & TEST DIRIN UAGEB &
        CJPPX STACKEXP
022E: 42 60 C9 9F 18 29 78 03 F7 BB XX XX

*         READ "LINE", ZREG ("POPLENGTH") -> WREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
022F: C2 7B 69 0F 30 3B F9 87 F0 0E 00 00

*         SP - 2 -> MAR, GREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & ARAM GREG & CONT
0230: 40 63 69 DF 98 69 F8 0F F0 0E 00 00

*         WRITE GREG, SP - 2 -> MAR, BREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM BREG &
        CONT
0231: 40 63 69 DF 98 7F F8 07 F0 0E 00 00

*         WRITE BREG, SP - 2 -> MAR,TREG, 2 -> QREG
        ALU QPT PASSR & DAQ & YOFF & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,LDTREG ,,PCUY LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL & IMMD 2 & CONT
0232: 43 33 78 9F 98 7E 78 07 F0 0E 00 02

*         TREG (SP) + WREG ("POPLENGTH") -> DREG
        ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,ENTREG ,,,,,LDD & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & CONT
0233: C2 61 89 5F 30 29 F9 87 F0 0E 00 00

*         WRITE SP+"POPLENGTH", SP - 2 -> MAR, ZREG ("LINE") ->
           DREG
        ALU YBUS PASSR & DAQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
0234: 42 63 69 CF 98 7F 78 07 F0 0E 00 00

*         WRITE "LINE", PC+2 -> MAR, TREG -> BREG
        ALU REG PASSR & DAQ & OEY & PCUNEXT & AM2904 &
        CARRYCTL & DATAPATH ,ENTREG ,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM BREG & CONT
0235: C2 7B 49 9F 18 3F F8 07 F0 0E 00 00

*         READ "VARLENGTH", PC+2 -> MAR,VREG
        ALU REG PASS & AB & YOFF & PCUNEXT & AM2904 &
        CARRYCTL & DATAPATH ,,,,,PCUY LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM VREG &
```

```
       RTN
0236: C9 7A 28 9F 18 3B FB 87 F0 0A 00 00

*
* CODE USED BY "EXT"
*
*        READ SP+POPLENGTH, BREG + 2 -> MAR, TREG + 2 -> R6
EXIT:  SPF14 INCTWO & AB & OEY & PCU ,,PCUDA A4 B6 & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,ENTREG ,,,YMAR LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
       CONT
0237: C0 20 0B 9F 59 BB F8 07 F4 0E 00 00

*        ZREG (SP+POPLENGTH) -> SP, READ OLD B, BREG + 2 -> MA
         R
       SPF14 INCTWO & AB & OEY & PCU ,,PCUDZ A1 B1 & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
       CONT
0238: C0 20 2B 8F 72 7B F8 07 F4 0E 00 00

*        R6 + 4 -> R6,MAR, ZREG (OLD B) -> BREG, READ OLD G
       ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A5 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
       CONT
0239: C2 7B 69 8F 1B BB F8 07 F0 0E 00 00

*        ZREG (OLD G) -> GREG, READ RETURN ADDRESS
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM GREG &
       CONT
023A: C2 7B 69 0F 30 3B F8 87 F0 0E 00 00

*        ZREG (RETURN ADDRESS) -> PC,MAR
       ALU YBUS PASS & AB & PCU ,,PCUDZ A0 B0 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL & JUMP START1
023B: 48 62 29 8F 70 29 78 07 F0 03 00 FD

*
* CODE TO PUT THE REGISTERS ON THE STACK, USED BY "EPP"
*
*        PC+2 -> MAR, HREG -> XREG
SCODE:  ALU REG PASSR & AQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
       CONTROL ROM & RTB & ARAM HREG & BRAM XREG & CONT
023C: C0 7B 69 9F 18 29 FA 17 F0 0E 00 00

*        READ "STACKLENGTH", SP - 2 -> MAR, ZREG ("LINE") -> W
         REG
       ALU REG PASSR & DAQ & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB &
       CNTLB WREG.LS.4 & CONT
023D: C2 7B 69 8F 98 7B F9 87 F0 0E 00 00
```

```
*         WRITE G, SP - 2 -> MAR, B -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & CNTLB BREG &
        CONT
023E: 40 63 69 DF 98 7F F8 07 F0 0E 00 00

*         WRITE B, PC+2 -> MAR, ZREG ("STACKLENGTH") + XREG ->
          XREG
        ALU REG ADD & DAB & OEY & PCUNEXT & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        CNTLB XREG.LS.4 & CONT
023F: C2 79 A9 8F 18 3F FA 07 F0 0E 00 00

*         READ "VARLENGTH", SP - 2 -> YREG, MAR
        ALU REG PASSR & AQ & YOFF & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,PCUY LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB &
        CNTLB YREG.LS.4 & CONT
0240: C1 7B 68 9F 98 7B FA 87 F0 0E 00 00

*         SP + R6 ("POPLENGTH") -> DREG, 2-> QREG
        ALU QPT PASSR & DAQ & YOFF & PCU 1 0 PCUAB A1 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,PCUY ,LDD &
        MEMCONT REQB & CONTROL & IMMD 2 & CONT
0241: 43 33 68 5F 13 A8 78 07 F0 0E 00 02

*         WRITE SP+POPLENGTH,SP - 2 -> MAR, WREG ("LINE") -> DR
          EG
        ALU YBUS PASSR & AQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & CNTLB WREG &
        CONT
0242: 40 63 69 DF 98 7F F8 1F F0 0E 00 00

*         WRITE "LINE", SP - ZREG ("VARLENGTH") -> SP, YREG - 2
           -> BREG
        ALU REG SUBS & AQ & OEY & PCU 1 1 PCUDA A1 B1 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        ARAM YREG & BRAM BREG & CONT
0243: C0 79 69 0F D2 7F F8 2F F4 0E 00 00

*         YREG - 2 -> YREG, PC+2 -> MAR
        ALU REG SUBS & AQ & OEY & PCUNEXT & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        RTB & CONTROL ROM & ARAM YREG & BRAM YREG &
        IMMD STACKLIMIT & CONT
0244: C0 79 69 9F 18 28 FA AF F4 0E 00 06

*         READ INSTR N+1, PC+2 -> MAR, XREG (HEAP + "STACKLENGT
          H") - YREG (SP) ->
*         EXCEPTION IF SP - "STACKENGTH" < HEAP
*                   I.E. HEAP + "STACKLENGTH" - SP > 0
        ALU YBUS SUBS & AB & OEY & PCUNEXT & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & ARAM XREG &
```

```
      BRAM YREG & TEST DIRIN UAGRB & CJPX STACKEXP
0245: C0 61 29 9F 18 3B FA A3 F7 C3 XX XX

*       READ INSTR N+2, PC+2 -> MAR, BREG -> GREG
      ALU REG PASSR & AB & PCUNEXT & AM2904 & CARRYCTL &
      DATAPATH ZZI ,,,,,LDMAR & MEMCONT REQB MREQ &
      CONTROL ROM & RTB & ARAM BREG & BRAM GREG & JMAP
0246: D8 7B 29 9F 18 3B F8 87 F0 02 00 00

*
* CODE SHARED BY "INC","DEC"
*
*       READ ADDRESS, SP+2 -> SP
INCDEC:  ALU YBUS PASS & AB & PCUPOP & AM2904 & CARRYCTL &
      DATAPATH & MEMCONT REQB MREQ & CONTROL & CONT
0247: 40 62 29 1F 18 7B 78 07 F0 0E 00 00

*       ZREG (ADDRESS) -> R6 OF PCU,MAR
      ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
      CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
      MEMCONT REQB & CONTROL & CONT
0248: 40 62 29 8F 7D A9 78 07 F0 0E 00 00

*       READ VALUE, PC+2 -> MAR
      ALU YBUS PASS & AB & PCUNOP & AM2904 & CARRYCTL &
      DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
      CONT
0249: 40 62 29 9F 30 3B 78 07 F0 0E 00 00

*       READ INSTR N+2, ZREG (VALUE) + WREG (1 OR -1) -> DREG
        , R6 OF PCU (ADDRE
      ALU YBUS ADD & DAB & OEY & PCU ,,PCUZA A6 B6 &
      AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
      JUMP WRTS
024A: CA 61 A9 CF 4D BB F9 87 F0 03 01 15

*
* CODE USED BY "TNR". TRUNCATE REAL.
*
*       READ INSTR N+2, ZREG (EXPONENT) - EXP1 (128) -> EXP1,
         EXIT IF EXPONENT
TRUNREAL:  ALU REG SUBS & DAB & OEY & PCUNOP & AM2904 ,OECT &
      CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
      MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM EXP1 &
      TEST DIRIN UALSB & CJP TRUNZERO
024B: C2 79 29 0F 30 3B FD 83 F7 A3 02 52

*       EXP1-15 -> YBUS, SET MICRO SR (CAUSES EXCEPTION LATER
        )
      ALU YBUS SUBR & DAB & PCUNOP & AM2904 ,,,,,,,CEU &
      CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
      RTB & BRAM EXP1 & TEST MICRO DLOAD & IMMD 15 & CONT
024C: C2 60 A9 1F 30 08 FD 87 E4 7E 00 0F

*       PUT SIGN OF WREG IN MSR CARRY BY SHIFTING
      ALU LUR PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
      CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM &
```

```
        CNTLB $10 & RTB & BRAM WREG & CONT
024D: C0 4A 29 1F 30 09 F1 87 F0 0E 00 00

*         PROPOGATE SIGN QREG (MANTISSA M.S.) -> WREG
*         EXCEPTION IF MICRO SR > 0 (MEANS THAT EXPONENT >15)
        ALU SINEX & AQ & OEY & PCUNOP & AM2904 SHIFTEN OECT &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM &
        CNTLB $19 & RTB &
        BRAM WREG & TEST MICRO UAGRB & CJPX OVFLOWEXP
024E: C0 70 69 1F 30 09 F1 CB F0 C3 XX XX

*         32 BIT SHIFT LEFT
SHB32:   ALU LURQ PASS & AB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $16 & CONT
024F: C0 5A 29 1F 30 09 F1 B7 F0 0E 00 00

*         DECREMENT EXP1 BY ADDING XREG (-1), LOOP BACK IF >= 0

        ALU REG ADD & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        ARAM XREG & BRAM EXP1 & TEST DIRIN UAGEB & CJP SHB32
0250: C0 79 A9 1F 30 09 FD A3 F3 B3 02 4F

*         WREG -> DREG, SP -> MAR.
        ALU YBUS PASS & AB & OEY & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & JUMP WRTS
0251: C8 62 29 DF 32 69 F9 87 F0 03 01 15

*         ARRIVED HERE IF EXPONENT <0 ,SO INTEGER CONVERSION MU
          ST BE ZERO
*         0 -> DREG, SP -> MAR.
TRUNZERO:   ALU YBUS LOW & AQ & OEY & PCUSP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL & JUMP WRTS
0252: 48 64 69 DF 32 69 78 07 F0 03 01 15

*
* CODE USED BY "CVW". CONVERT WORD TO REAL.
*
*         READ WORD, PC -> MAR, 144 (128+16) -> EXP1
CONVWORD:   ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM EXP1 & IMMD 144 & CONT
0253: C2 7B 69 9F 30 3A FD 87 F0 0E 00 90

*         READ INSTR N+2, ZREG (WORD TO BE CONVERTED) -> QREG
*         SP-2 -> MAR, JUMP TO SPECIAL ZERO CODE IF WORD = 0
        ALU QPT PASSR & DAQ & PCUPUSH & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & TEST DIRIN UAEQB &
        CJP ZEROCODE
0254: 42 33 69 8F 98 7B 78 03 F3 53 02 5A

*         NORMALISE.
        SPF14 SLN & AB & PCUNOP &
        AM2904 SHIFTEN OECT ,,ES ,CEM & CARRYCTL COEQ1 &
```

```
        DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $12 &
        TESTF $3A & CJP *
0255: C0 40 29 1F 30 09 F1 93 57 A3 02 55

*         SHIFT BACK + PUT IN D-REG, WRITE ZERO TO MANTISSA L.S
          . PS'N, SP-4 -> MA
        ALU LDR PASS & AQ & OEY & PCU ,SUB PCUAB A5 B1 &
        AM2904 SHIFTEN & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM XREG & CNTLB $05 & CONT
0256: C0 0A 69 DF 9A 7F F2 2F F0 0E 00 00

*         WRITE MANTISSA M.S., EXP1 - WREG -> DREG, R6 -> MAR.
        ALU YBUS SUBS & AB & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        ARAM EXP1 & BRAM WREG & CONT
0257: C0 61 29 DF 4D BF F9 DF F4 0E 00 00

*         WRITE EXPONENT, R6-4 -> MAR, 0 -> DREG.
WTEXP:   ALU YBUS LOW & AB & OEY & PCU ,SUB PCUAB A5 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
0258: 40 64 29 DF 9B BF 78 07 F0 0E 00 00

*         WRITE ZERO TO MANTISSA I.S. POS'N, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JMAP
0259: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*         ARRIVED HERE IF "CVW" WORD IS ZERO.
*         128 -> EXP1, WRITE ZERO TO MANTISSA L.S. POS'N,SP-4 -
          > MAR.
ZEROCODE:   ALU REG PASSR & DAQ & OEY & PCU ,SUB PCUAB A5 B1 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM EXP1 & IMMD 128 & CONT
025A: C2 7B 69 9F 9A 7E FD 87 F0 0E 00 80

*         WRITE ZERO TO MANTISSA M.S., EXP1 -> DREG, R6 -> MAR.

*         REJOIN MAIN CODE
        ALU YBUS PASS & AB & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM EXP1 & JUMP WTEXP
025B: C8 62 29 DF 4D BF FD 87 F0 03 02 58

*
* CODE USED BY "TEM"
*
*         READ VALUE, PC  -> MAR, 1 -> DREG
TESTMT:   ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ & CONTROL & IMMD 1 & CONT
025C: 42 63 69 DF 30 3A 78 07 F0 0E 00 01
```

```
*         READ INSTR N+2, SP -> MAR, ZREG (VALUE) -> YBUS, SKIP
           NEXT INSTRUCTION
        ALU YBUS PASSR & DAQ & PCUSP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & TEST DIRIN UAEQB &
        CJP WRTS
025D: 42 63 69 8F 32 7B 78 03 F3 53 01 15

*         0 -> DREG
        ALU YBUS LOW & AQ & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL &
        JUMP WRTS
025E: 48 64 69 5F 30 29 78 07 F0 03 01 15

*
*
*             ************************************************

*             *       INITIALISATION                         *

*             ************************************************

*
*
*         SET G REGISTER
INIT:   ALU REG PASSR & DAQ & OEY & CARRYCTL & DATAPATH &
        MEMCONT ,,HREQ & CONTROL ROM & RTB & CNTLB GREG.LS.4 &
        IMMD ISTACK+2 & CONT
025F: C2 7B 69 1E 00 00 F0 80 00 0E 1F 02

*         SET B REGISTER
        ALU REG PASSR & DAQ & OEY & CARRYCTL & DATAPATH &
        MEMCONT ,,HREQ & CONTROL ROM & RTB & CNTLB BREG.LS.4 &
        IMMD ISTACK & CONT
0260: C2 7B 69 1E 00 00 F0 00 00 0E 1F 00

*
* INITIALISE REGISTERS IN AM2901A
* R1 = STACK, R2 = 1,  R4 = 2, R5 = 5
*
*         R1 = STACK
        ALU YBUS PASS & CONTROL ,,,INTRIEN & CNTLB $F8 &
        DATAPATH & MEMCONT ,,HREQ & AM2904 &
        PCU ,,PCUDZ A1 B1 & IMMD ISTACK & CONT
0261: 40 62 29 1F 72 40 6F C7 F0 0E 1F 00

*         R2 = 1
        ALU YBUS PASS & CONTROL & DATAPATH & MEMCONT ,,HREQ &
        AM2904 & PCU ,,PCUDZ A2 B2 & IMMD 1 & CONT
0262: 40 62 29 1F 74 80 78 07 F0 0E 00 01

*         R4 = 2
        ALU YBUS PASS & PCU ,,PCUDZ A4 B4 & CARRYCTL &
        DATAPATH & MEMCONT ,,HREQ & IMMD 2 & CONT
0263: 40 62 29 1F 79 00 00 00 00 0E 00 02

*         R5 = 4
```

```
        ALU YBUS PASS & PCU ,,PCUDZ A5 B5 & CARRYCTL &
        DATAPATH & MEMCONT ,,HREQ & IMMD 4 & CONT
0264: 40 62 29 1F 7B 40 00 00 00 0E 00 04

        ALU YBUS PASS & CONTROL & DATAPATH & MEMCONT &
        AM2904 & PCUNOP & XJUMP INITPIA
0265: 48 62 29 1F 30 09 78 07 F0 03 XX XX

*         $3FFF -> WREG
TSTSUB:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM WREG & IMMD $3FFF & RTN
0266: CA 7B 69 1F 30 28 F9 87 F0 0A 3F FF

* CODE USED BY TST3COD
TST3COD:  ALU REG PASSR & DAQ & OEY & CARRYCTL & DATAPATH &
        MEMCONT ,,HREQ & CONTROL ROM & RTB & CNTLB GREG.LS.4 &
        IMMD ISTACK & CONT
0267: C2 7B 69 1E 00 00 F0 80 00 0E 1F 00

*         SET B REGISTER
        ALU REG PASSR & DAQ & OEY & CARRYCTL & DATAPATH &
        MEMCONT ,,HREQ & CONTROL ROM & RTB & CNTLB BREG.LS.4 &
        IMMD ISTACK & CONT
0268: C2 7B 69 1E 00 00 F0 00 00 0E 1F 00

*         SET H REGISTER
        ALU REG PASSR & DAQ & OEY & CARRYCTL & DATAPATH &
        MEMCONT ,,HREQ & CONTROL ROM & RTB & CNTLB HREG.LS.4 &
        IMMD IHEAP & CONT
0269: C2 7B 69 1E 00 00 F1 00 00 0E 04 00

*
* INITIALISE REGISTERS IN AM2901A
* R0 = PC, R1 = STACK, R2 = 1,  R4 = 2, R5 = 5
*
*         R0 = PC
        ALU YBUS PASS & CONTROL ,,,INTRIEN & CNTLB $F0 &
        DATAPATH & MEMCONT ,,HREQ & AM2904 &
        PCU ,,PCUDZ A0 B0 & IMMD 8 & CONT
026A: 40 62 29 1F 70 00 6F 87 F0 0E 00 08

*         R1 = STACK
        ALU YBUS PASS & CONTROL ,,,INTRIEN & CNTLB $F8 &
        DATAPATH & MEMCONT ,,HREQ & AM2904 &
        PCU ,,PCUDZ A1 B1 & IMMD ISTACK & CONT
026B: 40 62 29 1F 72 40 6F C7 F0 0E 1F 00

*         R2 = 1
        ALU YBUS PASS & CONTROL & DATAPATH & MEMCONT ,,HREQ &
        AM2904 & PCU ,,PCUDZ A2 B2 & IMMD 1 & CONT
026C: 40 62 29 1F 74 80 78 07 F0 0E 00 01

*         R4 = 2
        ALU YBUS PASS & PCU ,,PCUDZ A4 B4 & CARRYCTL &
        DATAPATH & MEMCONT ,,HREQ & IMMD 2 & CONT
026D: 40 62 29 1F 79 00 00 00 00 0E 00 02
```

```
*         R5 = 4
        ALU YBUS PASS & PCU ,,PCUDZ A5 B5 & CARRYCTL &
        DATAPATH & MEMCONT ,,HREQ & IMMD 4 & CONT
026E: 40 62 29 1F 7B 40 00 00 00 0E 00 04

        ALU YBUS PASS & CONTROL & DATAPATH & MEMCONT &
        AM2904 & PCUNOP & JUMP TST3
026F: 48 62 29 1F 30 09 78 07 F0 03 00 FB
0000 ERRORS (PMA-REV18.3)
```

```
*
* PRIME AMD MICRO-CODE MACRO ASSEMBLER
* VERSION 8, 2/10/82
        LIST
*
*
*
*
*
*                    ****************************************
*                    *          CONCURRENT PASCAL P-CODE     *
*                    ****************************************
*
*
INTMEM     EQU 2          1 FOR INTELLIGENT MEMORY, 2 OTHERWISE
* EXCEPTION ERROR CODES
TERMINATED   EQU 0
OVERFLOWERR  EQU 1
POINTERERR   EQU 2
RANGEERROR   EQU 3
VARIANTERR   EQU 4
HEAPLIMIT    EQU 5
STACKLIMIT   EQU 6
* MEMORY MAP.
P            EQU 3                  NUMBER OF PRIORITIES
NUMPROCESS   EQU 4                  MAXIMUM NUMBER OF PROCESSE
QSTART       EQU 0                  START OF QUEUE DATA STRUCT
QLENGTH      EQU 2*NUMPROCESS
TOTQLENGTH   EQU QLENGTH+4          TOTAL LENGTH OF QUEUE STRU
ONESECQ      EQU P*TOTQLENGTH       ADDRESS ONE SECOND QUEUE
SYSHEAP      EQU (P+1)*TOTQLENGTH   SYSTEM HEAP POINTER ADDRES
LASTHEAP     EQU SYSHEAP+2          NEXT AVAILABLE PROCESS DAT
NEXTINDEX    EQU LASTHEAP+2         NEXT INDEX COUNTER
RTREC        EQU NEXTINDEX+2+2*NUMPROCESS        REAL TIME REC
RTSECS       EQU RTREC+2            REAL TIME IN SECONDS
SPARE        EQU RTSECS+2
SPARELENGTH  EQU 10
GATELENGTH   EQU 2+TOTQLENGTH
* PROCESS ATTRIBUTE CODES
INDEXATR     EQU 0*2
HEAPATR      EQU 1*2
LINEATR      EQU 2*2
RESATR       EQU 3*2
RUNATR       EQU 4*2
SLICEATR     EQU 5*2
NESTATR      EQU 6*2
PRIATR       EQU 7*2
OVERTIME     EQU 8*2
JOBATR       EQU 9*2
CONTATR      EQU 10*2
OPCODE       EQU 11*2
ARG1         EQU 12*2
ARG2         EQU 13*2
ARG3         EQU 14*2
ARG4         EQU 15*2
OPLINE       EQU 16*2
CONSTATR     EQU 17*2
DEVICEATR    EQU 18*2
```

```
OPERATION     EQU 19*2
DATAATR       EQU 20*2
WAITING       EQU DATAATR+2
LASTTIME      EQU WAITING+2
PCREG     EQU LASTTIME+2
SPREG     EQU PCREG+2
REGG      EQU SPREG+2
REGB      EQU REGG+2
REGH      EQU REGB+2
PHLENGTH EQU REGH+10                        ALLOW SPARE
* DEVICE EQUATES
VDU           EQU 0
* MESSAGE EQUATES
TFMESS     EQU 1          TIMER FAULTY
IRMESS     EQU TFMESS+1   INTERRUPT RECEIVED
JMTSUCMESS EQU IRMESS+1   JOINT MEMORY TEST SUCCESSFULL
JMTUNMESS  EQU JMTSUCMESS+1   JOINT MEMORY TEST UNSUCCESSFULL
ALUMTSUCMESS    EQU JMTUNMESS+1       ALU MEMORY TEST SUCCESSFUL
ALUMTUNMESS     EQU ALUMTSUCMESS+1    ALU MEMORY TEST UNSUCCESSF
PCUMTSUCMESS    EQU ALUMTUNMESS+1     PCU MEMORY TEST SUCCESFULL
PCUMTUNMESS     EQU PCUMTSUCMESS+1    PCU MEMORY TEST UNSUCCESSF
PCUALUSUCMESS   EQU PCUMTUNMESS+1     PCU/ALU TEST SUCCESSFULL
PCUALUUNMESS    EQU PCUALUSUCMESS+1   PCU/ALU TEST UNSUCCESSFULL
ALUSUCMESS      EQU PCUALUUNMESS+1    ALU MEMORY TEST SUCCESSFUL
ALUUNMESS       EQU ALUSUCMESS+1      ALU TEST UNSUCCESSFULL
BMTSUCMESS      EQU ALUUNMESS+1       BYTE MEMORY TEST SUCCESSFU
BMTUNMESS       EQU BMTSUCMESS+1      BYTE MEMORY TEST UNSUCCESS
MAPZIMESS       EQU BMTUNMESS+1       MAP/ZI TEST SUCCESSFULL
INVMESS         EQU MAPZIMESS+1       INVALID INSTRUCTION MESSAG
SYSERRMESS      EQU INVMESS+1         SYSTEM ERROR MESSAGE
INTMESS         EQU SYSERRMESS+1      PERIODIC TIMER INTERRUPT M
IEMESS          EQU INTMESS+1         INTERRUPT ERROR MESSAGE
* OTHER EQUATES
A$INCREG      EQU 2*INTMEM       = A2 (1) OR A4 (2)
PROGSTART     EQU $C000*INTMEM
CNTVAL        EQU $5555                    TIMER INTERRUPT PE
ACIACS        EQU $2034*INTMEM    ACIA CONTROL REGISTER ADDRESS
ACIADA        EQU ACIACS+INTMEM        ACIA DATA REGISTER ADDRE
CLOCKSPEED    EQU 1
INTERVAL      EQU 40000          20 ms. = 40,000 cycles
CLOCKINCR     EQU 20           CLOCK INCREMENT VALUE
MAXSLICE      EQU 20
          ENT SYSCALL
          ENT BGNCLASS
          ENT BGNMON
          ENT ENDMON
          ENT ENTMON
          ENT INTCLASS
          ENT PUSHLAB
          ENT SETHREG
          ENT REALTIME
          ENT CALLPROG
          ENT GETATR
          ENT ENTERPROC
          ENT EXTPROG
          ENT PROCINIT
          ENT DELAY
          ENT CONTINUE
```

```
          ENT STOP
          ENT WAIT30
          ENT RESCHED
          ENT IPINIT
          ENT TIMER
          ENT TERMEXP
          ENT OVFLOWEXP
          ENT POINTEXP
          ENT RANGEEXP
          ENT VARIANTEXP
          ENT HEAPEXP
          ENT STACKEXP
          ENT EXCEPTION
          ENT IOPRES1
          EXT START
          EXT START1
          EXT WRTS
          EXT RNI
          EXT EXT
          EXT ENM
          ENT INPOUTOP
          ADDR $270
*
* SYSTEM CALL
*
*         READ BASE ADDRESS, PC -> DREG, ZREG ("ENTRY") -> QREG

SYSCALL:  ALU QPT PASSR & DAQ & YOFF & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,PCUY ,LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & CONT
0270: 43 33 68 4F 30 3B 78 07 F0 0E 00 00

*         ZREG (BASE ADDRESS) + QREG ("ENTRY") -> MAR.
        ALU YBUS ADD & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
0271: 42 61 EB 8F 30 29 78 07 F0 0E 00 00  '

*           READ PROCEDURE ADDRESS, SP-2 -> MAR, GREG-2 -> GREG
           (RESTORE FROM PRE
        ALU REG SUBR & DAB & OEY & PCUPUSH & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM GREG &
        IMMD 2 & CONT
0272: C2 78 A9 9F 98 7A F8 87 F4 0E 00 02

*           WRITE RETURN ADDRESS TO STACK
*           ZREG (PROCEDURE ADDRESS) -> PC,MAR,JUMP START1
        ALU YBUS PASS & AB & PCU ,,PCUDZ A0 B0 & AM2904 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JUMPX START1
0273: 48 62 29 8F 70 3F 78 07 F0 03 XX XX

*
*         EXIT PASCAL PROGRAM
*
*
*         ZREG (CONTINUE ATTRIBUTE) -> YBUS, JUMP TO EXCEPTION
```

```
          IF = 0
EXTPROG:  ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL & TEST DIRIN UAEQB & CJP EXCEPTION
0274: 42 63 69 0F 30 09 78 03 F3 53 03 31

*         PROGRAM HAS TERMINATED NORMALLY
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP TERMEXP
0275: 48 62 29 1F 30 09 78 07 F0 03 03 1C

*
* BEGIN CLASS
*
*         READ POINTER TO CLASS ADDRESS, BREG-2 -> BREG, -2 ->
          Q OF PCU
BGNCLASS:  ALU REG SUBR & DAB & OEY & PCU QEU SUB PCUDZ &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ZZI &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
        IMMD 2 & CONT
0276: D2 78 A9 1E F0 3A F8 07 F4 0E 00 02

*         ZREG (POINTER TO CLASS ADDRESS) + Q OF PCU (-2) -> MA
          R
        ALU YBUS PASS & DAQ & PCU ,,PCUDQ A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
0277: 42 62 69 8F 6D A9 78 07 F0 0E 00 00

*         READ CLASS ADDRESS, PC+2 -> MAR, 2 -> QREG
        ALU QPT PASSR & DAQ & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        IMMD 2 & CONT
0278: 42 33 69 9F 18 3A 78 07 F0 0E 00 02

*         ZREG (CLASS ADDRESS) - > GREG, READ INSTR N+2, PC+2 -
          > MAR
        ALU REG PASSR & DAQ & OEY & PCUNEXT & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM GREG &
        JMAP
0279: CA 7B 69 8F 18 3B F8 87 F0 02 00 00

*
* BEGIN MONITOR
*
*         READ POINTER TO MONITOR ADDRESS, BREG-2 -> BREG, -2 -
          > Q OF PCU
BGNMON:  ALU REG SUBR & DAB & OEY & PCU QEU SUB PCUDZ &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ZZI &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
        IMMD 2 & CONT
027A: D2 78 A9 1E F0 3A F8 07 F4 0E 00 02

*         ZREG (POINTER TO MONITOR ADDRESS) + Q OF PCU (-2) ->
          MAR
        ALU YBUS PASS & DAQ & PCU ,,PCUDQ A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
```

```
       MEMCONT REQB & CONTROL & CONT
027B: 42 62 69 8F 6D A9 78 07 F0 0E 00 00

*        READ MONITOR ADDRESS, SYSTEM HEAP POINTER ADDRESS ->
         R6,MAR
       ALU YBUS PASS & AQ & PCU ,,PCUDZ A6 B6 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
       CONTROL & IMMD SYSHEAP & CONT
027C: 40 62 69 9F 7D BA 78 07 F0 0E 00 30

*         ZREG (MONITOR ADDRESS) -> GREG
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
       CONTROL ROM & RTB & BRAM GREG & CONT
027D: C2 7B 69 0F 30 29 F8 87 F0 0E 00 00

* ALLOCATE SPACE ON THE SYSTEM HEAP FOR A GATE.
* THIS IS DONE BY ADDING GATELENGTH TO THE SYSTEM HEAP,
* SETTING OPEN TO FALSE AND INITIALISING THE QUEUE.
*        READ HEAP POINTER, GATELENGTH -> WREG.
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM WREG & IMMD GATELENGTH &
       CONT
027E: C2 7B 69 1F 30 3A F9 87 F0 0E 00 0E

*        ZREG (HEAP POINTER) -> R6 OF PCU, ZREG + WREG (GATELE
         NGTH) -> DREG (NEW
*        HEAP VALUE)
       ALU YBUS ADD & DAB & OEY & PCU ,,PCUDZ A6 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,,LDD ,ENZ0 &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
027F: C2 61 A9 4F 7D A9 F9 87 F0 0E 00 00

*        WRITE UPDATED HEAP POINTER ADDRESS, GREG -> MAR.
       ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,YMAR LDMAR &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM GREG &
       CONT
0280: 40 63 6B 9F 30 3F F8 0F F0 0E 00 00

*        R6 OF PCU (OLD HEAP POINTER ADDRESS) -> DREG (GATE AD
         DRESS)
       ALU YBUS PASS & AB & YOFF & PCU ,,PCUZA A6 B6 &
       AM2904 & DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB &
       CONTROL & CONT
0281: 41 62 28 5F 4D A9 78 07 F0 0E 00 00

*        WRITE GATE ADDRESS TO MONITOR, JUMP TO RUNNING.ENTER
         SUBROUTINE.
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JSB RUNENT
0282: 48 62 29 1F 30 3F 78 07 F0 01 03 37

*        0 -> DREG, R6 OF PCU (GATE ADDRESS) -> MAR
       ALU YBUS LOW & AQ & OEY & PCU ,,PCUZA A6 B6 & AM2904 &
       DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB & CONTROL &
       CONT
```

```
0283: 40 64 69 DF 4D A9 78 07 F0 0E 00 00

*         SET OPEN TO FALSE, I.E. WRITE ZERO TO THE GATE
*         R6 + 2 -> R6 (POINTS TO QUEUE START)
*         JUMP TO INITIALISE QUEUE SUBROUTINE
        ALU YBUS PASS & AB & PCU ,,PCUAB A4 B6 & AM2904 &
        DATAPATH & MEMCONT REQB MREQ ,WRITE & CONTROL &
        JSB QINIT
0284: 48 62 29 1F 19 BF 78 07 F0 01 03 42

*         PC+2 -> MAR, JUMP RNI
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMPX RNI
0285: 48 62 29 9F 18 29 78 07 F0 03 XX XX

*
* ENTER MONITOR
*
*         READ POINTER TO MONITOR ADDRESS, BREG-2 -> BREG, -2 -
          > Q OF PCU
ENTMON:   ALU REG SUBR & DAB & OEY & PCU QEU SUB PCUDZ &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ZZI &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
        IMMD 2 & CONT
0286: D2 78 A9 1E F0 3A F8 07 F4 0E 00 02

*         ZREG (POINTER TO MONITOR ADDRESS) + Q OF PCU (-2) ->
          MAR
        ALU YBUS PASS & DAQ & PCU ,,PCUDQ A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
0287: 42 62 69 8F 6D A9 78 07 F0 0E 00 00

*         READ MONITOR ADDRESS, SYSTEM HEAP POINTER -> R6,MAR
        ALU YBUS PASS & AQ & PCU ,,PCUDZ A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL & IMMD SYSHEAP & CONT
0288: 40 62 69 9F 7D BA 78 07 F0 0E 00 30

*         ZREG (MONITOR ADDRESS) -> GREG, JUMP RUNNING.ENTER SU
          BROUTINE
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM GREG & JSB RUNENT
0289: CA 7B 69 0F 30 09 F8 87 F0 01 03 37

*         GREG -> MAR
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR & MEMCONT REQB &
        CONTROL ROM & ARAM GREG & CONT
028A: 40 63 6B 9F 30 29 F8 0F F0 0E 00 00

*         READ GATE ADDRESS
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & CONT
028B: 40 62 29 1F 30 3B 78 07 F0 0E 00 00
```

```
*         ZREG (GATE ADDRESS) -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
        CONT
028C: 40 62 29 8F 7D A9 78 07 F0 0E 00 00

*         THE FOLLOWING TWO CYCLES FORM A TEST AND RESET OPERAT
          ION
*         READ FIRST LOACATION OF GATE (I.E. OPEN FLAG), 0 -> D
          REG
        ALU YBUS LOW & AQ & OEY & PCUNOP & AM2904 &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ HREQ &
        CONTROL & CONT
028D: 40 64 69 5F 30 33 78 07 F0 0E 00 00

*         WRITE ZERO TO GATE (I.E. MAKE IT CLOSED), ZREG (OPEN
          FLAG) -> YBUS,
*         IF NOT ZERO THEN SKIP NEXT 2 INSTRUCTIONS.
        ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL &
        TEST DIRIN UANEB & CJP GATEOPEN
028E: 42 63 69 0F 30 3F 78 03 F3 43 02 91

*         ARRIVED HERE IF GATE IS CLOSED.
*         PUT PROCESS IN MONITOR QUEUE.
        ALU YBUS PASS & AB & PCU ,,PCUAB A4 B6 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & JSB PUTINQ
028F: 48 62 29 1F 19 89 78 07 F0 01 03 45

*         RESCHEDULE
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT & CONTROL &
        JUMP IOPRES
0290: 48 62 29 9F 18 09 78 07 F0 03 03 0C

*         ARRIVED HERE IF GATE IS OPEN.
*         FETCH NEXT INSTRUCTION
*         PC+2 -> MAR, JUMP RNI
GATEOPEN:  ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMPX RNI
0291: 48 62 29 9F 18 29 78 07 F0 03 XX XX

*
* CODE USED BY "ENM".
* END MONITOR (+ EXIT MONITOR).
*
*         READ SP+POPLENGTH, BREG + 2 -> MAR, TREG + 2 -> R6
ENDMON:  SPF14 INCTWO & AB & OEY & PCU ,,PCUDA A4 B6 &
        AM2904 & CARRYCTL COEQ1 &
        DATAPATH ,ENTREG ,,,YMAR LDMAR & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM BREG & CONT
0292: C0 20 0B 9F 59 BB F8 07 F4 0E 00 00

*         ZREG (SP+POPLENGTH) -> SP, READ OLD B, BREG + 2 -> MA
          R
        SPF14 INCTWO & AB & OEY & PCU ,,PCUDZ A1 B1 & AM2904 &
```

```
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
        CONT
0293: C0 20 2B 8F 72 7B F8 07 F4 0E 00 00

*         R6 + 4 -> R6,MAR, ZREG (OLD B) -> BREG, READ OLD G
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A5 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
        CONT
0294: C2 7B 69 8F 1B BB F8 07 F0 0E 00 00

*         ZREG (OLD G) -> GREG, READ RETURN ADDRESS
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM GREG &
        CONT
0295: C2 7B 69 0F 30 3B F8 87 F0 0E 00 00

*         ZREG (RETURN ADDRESS) -> PC, JUMP TO EXAMINE QUEUE RO
          UTINE.
        ALU YBUS PASS & AB & PCU ,,PCUDZ A0 B0 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        JSB EXAMINEQ
0296: 48 62 29 0F 70 09 78 07 F0 01 03 4D

*         TEST MSR, IF = THEN SKIP NEXT INSTRUCTION
*         1 -> DREG, R7 (GATE ADDRESS) -> MAR
        ALU LUR LOW & AQ & PCU ,,PCUZA A7 B7 &
        AM2904 SHIFTEN OECT & DATAPATH ,,,,,,LDMAR LDD &
        CONTROL ROM & RTB & BRAM WREG & CNTLB $11 &
        TEST MACHINE UAEQB & CJP *+18
0297: C0 4C 69 DF 4F C1 F1 8B F2 53 02 9A

*          R7 OF PCU -> R6 (QUEUE ADDRESS),
*         JUMP TO PUT NEXT PROCESS IN QUEUE ROUTINE.
        ALU YBUS PASS & AB & PCU ,,PCUZA A7 B6 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & JSB MPUTQ
0298: 48 62 29 1F 4F 89 78 07 F0 01 03 55

*         PC+4 -> PC
        ALU YBUS PASS & AB & PCU ,,PCUAB A5 B0 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & JUMP RUNLEAVE
0299: 48 62 29 1F 1A 09 78 07 F0 03 03 3C

*         SET OPEN TO TRUE (WRITE TO IT)
*         JUMP TO RUNNING.LEAVE, PC+4 -> PC
        ALU YBUS PASS & AB & PCU ,,PCUAB A5 B0 & AM2904 &
        DATAPATH & MEMCONT REQB MREQ ,WRITE & CONTROL &
        JUMP RUNLEAVE
029A: 48 62 29 1F 1A 3F 78 07 F0 03 03 3C

*
* ENTER PREFIX PROCEDURE
*
*         GREG+6 -> MAR
ENTERPROC:  ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ZZI ,,,,YMAR LDMAR &
```

```
       MEMCONT REQB & CONTROL ROM & RTB & BRAM GREG &
       IMMD 6 & CONT
029B: D2 61 AB 9F 30 28 F8 87 F0 0E 00 06

*        READ NEW G VALUE, 0 -> DREG, PHREG + JOBATR -> MAR
       ALU YBUS LOW & AQ & OEY & PCU QEU ,PCUDA PHREG &
       AM2904 & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB MREQ & CONTROL & IMMD JOBATR & CONT
029C: 40 64 69 DE 56 3A 78 07 F0 0E 00 12

*        WRITE NEW MODE = SYSTEM
*        PC+2 -> MAR, ZREG (NEW G VALUE) -> GREG
       ALU REG PASSR & DAQ & OEY & PCUNEXT & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM GREG & JUMPX RNI
029D: CA 7B 69 8F 18 3F F8 87 F0 03 XX XX

*
* INITIALISE CLASS
*
*        READ CLASS ADDRESS, ZREG ("DISTANCE") -> TREG, SP ->
         MAR
*        JUMP OUT OF MSR = 0 (I.E. "PARAMLENGTH" = 0)
INTCLASS:  ALU YBUS PASSR & DAQ & OEY & PCUSP &
       AM2904 ,OECT & CARRYCTL &
       DATAPATH ,,LDTREG ,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & TEST MACHINE UAEQB &
       CJP PLEQZERO
029E: 42 63 79 8F 32 7B 78 03 F2 53 02 A7

*        ZREG (CLASS ADDRESS) -> R6
       ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB & CONTROL &
       CONT
029F: 40 62 29 0F 7D A9 78 07 F0 0E 00 00

*        NOTE THE PIPELINE
*        READ FIRST PARAMETER, WREG+2 -> WREG
*        JUMP PAST LOOP IF ZERO (MEANS ONLY 1 PARAMETER)
*        SP+2 -> MAR
       SPF14 INCTWO & AB & OEY & PCUPOP & CARRYCTL COEQ1 &
       AM2904 ,OECT & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       TEST DIRIN UAEQB & CJP LOOPOVER
02A0: C0 20 29 9F 18 7B F9 83 F7 53 02 A3

*        THE FOLLOWING TWO WORDS ARE A LOOP
*        READ NEXT PARAMETER, ZREG (CURRENT PARAMETER) -> DREG
         , R6+2 -> MAR
ICLLOOP:  ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
02A1: 42 63 69 CF 19 BB 78 07 F0 0E 00 00

*        WRITE CURRENT PARAMETER, WREG+2 -> WREG, LOOP BACK IF
          <> 0
*        SP+2 -> MAR
```

```
        SPF14 INCTWO & AB & OEY & PCUPOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM WREG & TEST DIRIN UANEB & CJP ICLLOOP
02A2: C0 20 29 9F 18 7F F9 83 F7 43 02 A1

*         ZREG (LAST PARAMETER) -> DREG, R6+2 -> MAR
LOOPOVER:  ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
02A3: 42 63 69 CF 19 A9 78 07 F0 0E 00 00

*         WRITE LAST PARAMETER, PC+2 -> DREG (RETURN ADDRESS)
        ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUAB A4 B0 &
        AM2904 & DATAPATH ,,,,,PCUY ,LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
02A4: 41 62 28 5E 18 3F 78 07 F0 0E 00 00

*         SP - 2 -> MAR
        ALU YBUS PASS & AB & PCUPUSH & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & CONT
02A5: 40 62 29 9F 98 69 78 07 F0 0E 00 00

*         PC + TREG ("DISTANCE") -> PC,MAR, WRITE RETURN ADDRES
          S, JUMP TO START1
        ALU YBUS PASS & AB & PCU ,,PCUDA A0 B0 & AM2904 &
        DATAPATH ,ENTREG ,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JUMPX START1
02A6: 48 62 09 9F 50 3F 78 07 F0 03 XX XX

*         CODE FOR THE SPECIAL CASE "PARAMLENGTH" = 0
*         PC+2 -> DREG (BUT NOT PC)
PLEQZERO:  ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUAB A0 B4 &
        AM2904 & DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB &
        CONTROL & CONT
02A7: 41 62 28 5E 11 29 78 07 F0 0E 00 00

*         SP-2 -> MAR
        ALU YBUS PASS & AB & PCUPUSH & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        CONT
02A8: 40 62 29 9F 98 7B 78 07 F0 0E 00 00

*         PC+TREG ("DISTANCE") -> MAR, WRITE RETURN ADDRESS TO
          STACK
        ALU YBUS PASS & AB & PCU ,,PCUDA A0 B0 & AM2904 &
        DATAPATH ,ENTREG ,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JUMPX START1
02A9: 48 62 09 9F 50 3F 78 07 F0 03 XX XX

*
* INITIALISE PROCESS (CHILD)
*
*         1) TRANSFER PARAMETERS FROM PARENTS ADDRESS SPACE TO
          CHILDS'.
*         ZREG (LAST HEAP VALUE) -> R7,ZREG - YREG (1) - 1 -> Y
          REG
PROCINIT:  ALU REG SUBS & DAB & PCU ,,PCUDZ A7 B7 & AM2904 &
```

```
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM YREG & CONT
02AA: C2 79 29 0F 7F C9 FA 87 F0 0E 00 00

*         SP -> Q, 2 -> REAL1M
        ALU REG PASSR & DAQ & PCU QEU ,PCUZA A1 & AM2904 &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM REAL1M &
        IMMD 2 & CONT
02AB: C2 7B 69 1E 42 08 FC 07 F0 0E 00 02

*         WREG+2 -> WREG, JUMP OUT OF LOOP IF > 0, Q-2 -> MAR
COPYPARAM:  SPF14 INCTWO & AB & OEY & PCU QEU SUB PCUAQ A4 &
        AM2904 ,OECT & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN SAGRB & CJP PREIP
02AC: C0 20 29 9E 88 29 F9 83 F7 03 02 B0

*         READ NEXT PARAMETER FROM PARENTS STACK
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & CONT
02AD: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*         ZREG (NEXT PARAMETER) -> DREG, R7-2 -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCU ,SUB PCUAB A4 B7 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
02AE: 42 63 69 CF 99 E9 78 07 F0 0E 00 00

*         WRITE NEXT PARAMETER TO CHILDS ADDRESS SPACE
*         R7 -> YREG, LOOP BACK
        ALU REG PASS & AB & YOFF & PCU ,,PCUZA A7 B7 &
        AM2904 & DATAPATH ,,,,,PCUY &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM YREG & JUMP COPYPARAM
02AF: C9 7A 28 1F 4F FF FA 87 F0 03 02 AC

*         STEP 2) PRE-EMPT THE PARENT
*         YREG - REAL1M (2) -> YREG
*         SP+2 -> SP (POP'S PROCESS ADDRESS), PRE-EMPT PARENT P
          ROCESS
PREIP:   ALU REG SUBR & AB & PCUPOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
        RTB & ARAM REAL1M & BRAM YREG & JSB PREEMPT
02B0: C8 78 A9 1F 18 49 FA C7 F4 01 03 6F

*         STEP 3) FETCH THE PARENTS' LARGE CONSTANT ADDRESS FOR
           LATER
*         PHREG +CONSTATR -> MAR, YREG -> GREG
        ALU REG PASSR & AQ & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
        RTB & ARAM YREG & BRAM GREG & IMMD CONSTATR & CONT
02B1: C0 7B 69 9E 56 28 F8 AF F0 0E 00 22

*         READ CONSTANT ADDRESS, PHREG -> REAL2M (USED AS A SCR
          ATCH REGISTER)
        ALU REG PASS & AB & YOFF & PCU ,,PCUZA PHREG PHREG &
        AM2904 & DATAPATH ,,,,,PCUY & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM REAL2M & CONT
```

```
02B2: C1 7A 28 1F 46 FB FE 07 F0 0E 00 00

*         ZREG (CONSTANT ADDRESS) -> REAL1M (USED AS A SCRATCH
          REGISTER)
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM REAL1M & CONT
02B3: C2 7B 69 0F 30 09 FC 07 F0 0E 00 00

*         STEP 4) CREATE MEMORY SPACE FOR HEAP, STACK AND PERMA
          NENT VARIABLES.
*         SYSTEM HEAP ADDRESS -> MAR, -QREG -> QREG ("PARAMLENG
          TH")
        ALU QPT COMPLS & AQ & PCU QEU ,PCUDZ & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL & IMMD SYSHEAP & CONT
02B4: 40 32 E9 9E 70 28 78 07 F4 0E 00 30

*         READ SYSTEM HEAP, PROCESS HEAD LENGTH -> WREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM WREG & IMMD PHLENGTH & CONT
02B5: C2 7B 69 1F 30 3A F9 87 F0 0E 00 40

*         WREG (PHLENGTH) + ZREG (SYSTEM HEAP) -> WREG,DREG
*         ZREG -> PHREG
        ALU REG ADD & DAB & OEY & PCU ,,PCUDZ PHREG PHREG &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,,LDD ,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
02B6: C2 79 A9 4F 76 E9 F9 87 F0 0E 00 00

*         WRITE NEW SYSTEM HEAP VALUE, PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & CONT
02B7: 40 62 29 9F 30 3F 78 07 F0 0E 00 00

*         READ VARLENGTH, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONT
02B8: 40 62 29 9F 18 3B 08 07 F0 0E 00 00

*         READ STACKLENGTH, ZREG (VARLENGTH) + QREG (PARAMLENGT
          H) + 1 -> HREG
*         ZREG -> R6 OF PCU
        ALU REG ADD & DAQ & OEY & PCU ,,PCUDZ A6 B6 & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM HREG &
        CONT
02B9: C2 79 E9 0F 7D BB F9 07 F4 0E 00 00

*         LAST HEAP ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD LASTHEAP & CONT
02BA: 40 62 29 9E 70 28 78 07 F0 0E 00 32

*         ZREG (STACKLENGTH) + HREG + 1 -> HREG, READ LAST HEAP
```

```
        ALU REG ADD & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM HREG &
        CONT
02BB: C2 79 A9 0F 30 3B F9 07 F4 0E 00 00

*         ZREG(LAST HEAP) - HREG -> DREG,HREG
        ALU REG SUBS & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD ,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM HREG & CONT
02BC: C2 79 29 4F 30 29 F9 07 F4 0E 00 00

*         WRITE NEW LAST HEAP, HREG (LAST HEAP) - WREG (SYSTEM
          HEAP) -> YBUS
*         SYSTEM ERROR IF <=
        ALU YBUS SUBR & AB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB MREQ ,WRITE &
        CONTROL ROM & RTB & ARAM WREG & BRAM HREG &
        TEST DIRIN UALEB & CJP SYSERR
02BD: C0 60 A9 1F 30 3F F9 1B F7 D3 03 DF

*         R6 OF PCU (VARLENGTH) + 2 -> TREG
        ALU YBUS PASS & AB & YOFF & PCU ,,PCUAB A4 B6 &
        AM2904 & DATAPATH ,,LDTREG ,,PCUY & MEMCONT &
        CONTROL & CONT
02BE: 41 62 38 1F 19 89 78 07 F0 0E 00 00

*         GREG - TREG ("VARLENGTH") -> TREG
        ALU YBUS SUBR & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,ENTREG LDTREG & MEMCONT &
        CONTROL ROM & RTB & BRAM GREG & CONT
02BF: C2 60 99 1F 30 09 F8 87 F4 0E 00 00

*         TREG -> SP, GREG -> BREG, JUMP TO INITIALISE ATTRIBUT
          ES SUBROUTINE.
        ALU REG PASSR & AQ & YOFF & PCU ,,PCUDZ A1 B1 &
        AM2904 & CARRYCTL & DATAPATH ,ENTREG ,,,PCUY &
        MEMCONT & CONTROL ROM & RTB & ARAM GREG & BRAM BREG &
        JSB ATRINIT
02C0: C9 7B 48 1F 72 49 F8 0F F0 01 03 AD

*         PHREG+CONSTATR -> MAR, REAL1M (PARENTS CONSTANT ADDRE
          SS) -> DREG
        ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & ARAM REAL1M &
        IMMD CONSTATR & CONT
02C1: 40 63 69 DE 56 28 F8 47 F0 0E 00 22

*         WRITE CONSTANT ATTRIBUTE,PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & CONT
02C2: 40 62 29 9F 18 3F 78 07 F0 0E 00 00

*         READ "DISTANCE", PRIORITY 2 QUEUE ADDRESS -> R7
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
```

```
        DATAPATH & MEMCONT REQB MREQ & CONTROL &
        IMMD QSTART+2*TOTQLENGTH & CONT
02C3:  40 62 29 1F 7F FA 78 07 F0 0E 00 18

*         PC+4 -> VREG
        ALU REG PASS & AB & YOFF & PCU ,,PCUAB A5 B0 &
        AM2904 & DATAPATH ,,,,,PCUY & MEMCONT & CONTROL ROM &
        RTB & BRAM VREG & CONT
02C4:  C1 7A 28 1F 1A 09 FB 87 F0 0E 00 00

*         ZREG ("DISTANCE") + PC -> PC, PRE-EMPT
        ALU YBUS PASS & AB & PCU ,,PCUDA A0 B0 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        JSB PREEMPT
02C5:  48 62 29 0F 50 09 78 07 F0 01 03 6F

*         REAL2M (PARENTS PHREG) -> TREG, JUMP TO PUTINQ, R7 ->
           R6
        ALU YBUS PASSR & AQ & OEY & PCU ,,PCUZA A7 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,LDTREG & MEMCONT &
        CONTROL ROM & ARAM REAL2M & JSB PUTINQ
02C6:  48 63 79 1F 4F 89 F8 67 F0 01 03 45

*         TREG -> PHREG, VREG (PARENTS PC) + 2 -> DREG
        SPF14 INCTWO & AB & OEY & PCU ,,PCUDZ PHREG PHREG &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,ENTREG ,,,,,LDD &
        MEMCONT & CONTROL ROM & RTB & BRAM VREG & CONT
02C7:  C0 20 09 5F 76 C9 FB 87 F4 0E 00 00

*         PHREG + PC ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD PCREG & CONT
02C8:  40 62 29 9E 56 28 78 07 F0 0E 00 2E

*         WRITE PARENTS PC, PUT PARENT IN QUEUE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JSB PUTINQ
02C9:  48 62 29 1F 30 3F 78 07 F0 01 03 45

*         RESCHEDULE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP RESCHED
02CA:  48 62 29 1F 30 09 78 07 F0 03 03 8A

*
* PUSH LABEL
*
*         TREG ("DISTANCE") + Q OF PCU (PC-2) -> DREG (LABEL AD
           DRESS)
PUSHLAB:   ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDQ A6 B6 &
        AM2904 & DATAPATH ZZI ENTREG ,,,PCUY ,LDD & MEMCONT &
        CONTROL & CONT
02CB:  51 62 08 5E 6D 89 78 07 F0 0E 00 00

*         SP-2 -> MAR
        ALU YBUS PASS & AB & PCUPUSH & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & CONT
```

```
02CC: 40 62 29 9F 98 69 78 07 F0 0E 00 00

*         WRITE LABEL, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMPX RNI
02CD: 48 62 29 9F 18 3F 78 07 F0 03 XX XX

*
* CALL PROGRAM
*
*         READ DESCRIPTOR ADDRESS, PC -> YREG
CALLPROG:  ALU REG PASS & AB & YOFF & PCUNOP & AM2904 &
        DATAPATH ,,,,,PCUY & MEMCONT REQB MREQ & CONTROL ROM &
        RTB & BRAM YREG & CONT
02CE: C1 7A 28 1F 30 3B FA 87 F0 0E 00 00

*         ZREG (DESCRIPTOR ADDRESS) -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
        CONT
02CF: 40 62 29 8F 7D A9 78 07 F0 0E 00 00

*         READ INDEX, R6+2 -> MAR, YREG(PC) - 2 -> YREG
        ALU REG SUBR & DAB & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM YREG &
        IMMD 2 & CONT
02D0: C2 78 A9 9F 19 BA FA 87 F4 0E 00 02

*         READ TABLE ADDRESS, (INDEX-1)*2 -> WREG
        ALU LUR SUBS & DAB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CNTLB $10 & CONT
02D1: C2 49 29 0F 30 3B F1 87 F4 0E 00 00

*         ZREG (TABLE BASE ADDRESS) + WREG ((INDEX-1)*2) -> MAR

*         ZREG (TABLE BASE ADDRESS) + 2 -> PC
        ALU YBUS ADD & DAB & OEY & PCU ,,PCUDA A4 B0 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
02D2: C2 61 AB 8F 58 29 F9 87 F0 0E 00 00

*         READ PROG ADDRESS
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & CONT
02D3: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*         ZREG (PROG ADDRESS) + PC -> PC,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDA A0 B0 & AM2904 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
        CONT
02D4: 40 62 29 8F 50 29 78 07 F0 0E 00 00

*         READ CONSTANT ADDRESS DISPLACEMENT, PC+6 -> MAR,VREG
        ALU REG PASS & AB & YOFF & PCU ,,PCUDA A0 B0 &
```

```
        AM2904 & DATAPATH ,,,,,PCUY LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM VREG &
        IMMD 6 & CONT
02D5: C1 7A 28 9F 50 3A FB 87 F0 0E 00 06

*         ZREG (CONSTANT ADDRESS DISPLACEMENT) + VREG (PC) -> D
          REG
*         SP -> MAR, READ INSTR N+1
        ALU YBUS ADD & DAB & OEY & PCUSP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR LDD ,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM VREG & CONT
02D6: C2 61 A9 CF 32 7B FB 87 F0 0E 00 00

*         WRITE LARGE CONSTANT ADDRESS, SP-2 -> MAR
*         YREG (OLD PC = RETURN ADDRESS) -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUPUSH & AM2904 &
        CARRYCTL & DATAPATH ZZI ,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM YREG &
        CONT
02D7: 50 63 69 DF 98 7F F8 2F F0 0E 00 00

*         WRITE RETURN ADDRESS, PC+2 -> MAR
        ALU YBUS PASS & AB & PCUNEXT & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JUMPX RNI
02D8: 48 62 29 9F 18 3F 78 07 F0 03 XX XX

*
* GET ATTRIBUTE
*
*         READ INDEX FROM STACK, PHREG + 2 -> R6
GETATR:   ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B6 & AM2904 &
        DATAPATH & MEMCONT REQB MREQ & CONTROL & IMMD 2 &
        CONT
02D9: 40 62 29 1F 57 BA 78 07 F0 0E 00 02

*          2*(ZREG (INDEX) - WREG (1)) -> YBUS,TREG
*         JUMP TO SPECIAL CODE IF HEAP ATTRIBUTE REQUIRED, PC -
          > MAR
        ALU LUR SUBS & DAB & PCUNOP & AM2904 SHIFTEN OECT &
        CARRYCTL COEQ1 & DATAPATH ,,LDTREG ,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG &
        CNTLB $10 & TEST DIRIN UAEQB & CJP SHCODE
02DA: C2 49 39 8F 30 29 F1 83 F7 53 02 DE

*         TREG (2*(INDEX-1)) + R6 (PHREG+2) -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDA A6 & AM2904 &
        DATAPATH ,ENTREG ,,,,LDMAR & MEMCONT REQB & CONTROL &
        CONT
02DB: 40 62 09 9E 5C 29 78 07 F0 0E 00 00

*         READ ATTRIBUTE, PC -> MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        CONT
02DC: 40 62 29 9F 30 3B 78 07 F0 0E 00 00

*         ZREG (ATTRIBUTE) -> DREG, SP -> MAR, READ INSTR N+2
```

```
       ALU YBUS PASSR & DAQ & OEY & PCUSP & AM2904 &
       DATAPATH ,,,,,,LDMAR LDD ,ENZ0 & MEMCONT REQB MREQ &
       CONTROL & JUMPX WRTS
02DD:  4A 63 69 CF 32 7B 78 07 F0 03 XX XX

*        SPECIAL CODE FOR HEAP ATTRIBUTE
*        HREG -> DREG, SP -> MAR, READ INSTR N+2
SHCODE:  ALU YBUS PASSR & AQ & OEY & PCUSP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB MREQ & CONTROL ROM & ARAM HREG &
       JUMPX WRTS
02DE:  48 63 69 DF 32 7B F8 17 F0 03 XX XX

*
* REAL TIME.
*
*        ZREG (REAL TIME IN SECONDS) -> DREG, SP-2 -> MAR, REA
         D INSTR N+2
REALTIME:  ALU YBUS PASSR & DAQ & OEY & PCUPUSH & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
02DF:  42 63 69 CF 98 7B 78 07 F0 0E 00 00

*        WRITE REAL TIME TO STACK, PC+2 -> MAR, JUMP MAP
       ALU YBUS PASS & AB & PCUNEXT & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JMAP
02E0:  48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* DELAY PROCESS
*
*        PC+2 -> PC
DELAY:  ALU YBUS PASS & AB & PCUNEXT & AM2904 & DATAPATH &
       MEMCONT REQB & CONTROL & CONT
02E1:  40 62 29 1F 18 29 78 07 F0 0E 00 00

*        READ QUEUE ADDRESS, SP+2 -> SP, PRE-EMPT PROCESS.
       ALU YBUS PASS & AB & OEY & PCUPOP & AM2904 &
       DATAPATH & MEMCONT REQB MREQ & CONTROL & JSB PREEMPT
02E2:  48 62 29 1F 18 7B 78 07 F0 01 03 6F

*        PHREG -> DREG
       ALU YBUS PASS & AB & YOFF & PCU ,,PCUZA PHREG PHREG &
       AM2904 & DATAPATH ,,,,,PCUY ,LDD & MEMCONT & CONTROL &
       CONT
02E3:  41 62 28 5F 46 C9 78 07 F0 0E 00 00

*        ZREG (QUEUE ADDRESS)  -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
       CONT
02E4:  40 62 29 8E 70 29 78 07 F0 0E 00 00

*        WRITE PHREG TO QUEUE, TREG (MONITOR ADDRESS) -> R7 OF
          PCU
*        JUMP TO EXAMINE QUEUE ROUTINE.
       ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
```

```
        DATAPATH ,ENTREG & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JSB EXAMINEQ
02E5:  48 62 09 1F 7F FF 78 07 F0 01 03 4D

*         TEST MSR, IF <> THEN SKIP NEXT INSTRUCTION, 1 -> DREG

*         R7 (GATE ADDRESS) -> R6,MAR
        ALU LUR LOW & DAQ & OEY & PCU ,,PCUZA A7 B6 &
        AM2904 SHIFTEN OECT & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $11 & TEST MACHINE UANEB &
        CJP *+12
02E6:  C2 4C 69 DF 4F A9 F1 8B F2 43 02 E8

*         ARRIVED HERE IF MONITOR QUEUE EMPTY, WRITE 1 TO GATE
          (OPEN = TRUE), RES
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP RESCHED
02E7:  48 62 29 1F 30 3F 78 07 F0 03 03 8A

*         ARRIVED HERE IF SOMETHING IN MONITOR QUEUE.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB MPUTQ
02E8:  48 62 29 1F 30 09 78 07 F0 01 03 55

*         RESCHEDULE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP RESCHED
02E9:  48 62 29 1F 30 09 78 07 F0 03 03 8A

*
* CONTINUE PROCESS
*
*         READ QUEUE ADDRESS, SP+2 -> SP
CONTINUE:  ALU YBUS PASS & AB & PCUPOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & CONT
02EA:  40 62 29 1F 18 7B 78 07 F0 0E 00 00

*         ZREG (QUEUE ADDRESS) -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
        CONT
02EB:  40 62 29 8E 70 29 78 07 F0 0E 00 00

*         READ QUEUE VALUE, PHREG -> TREG
        ALU YBUS PASS & AB & YOFF & PCU ,,PCUZA PHREG PHREG &
        AM2904 & DATAPATH ,,LDTREG ,,PCUY &
        MEMCONT REQB MREQ & CONTROL & CONT
02EC:  41 62 38 1F 46 FB 78 07 F0 0E 00 00

*         WRITE 0 TO QUEUE (NIL), GREG -> MAR
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM GREG &
        CONT
02ED:  40 63 6B 9F 30 3F F8 0F F0 0E 00 00

*         ZREG (QUEUE VALUE) -> R7 OF PCU,YBUS, JUMP TO "ENM" I
```

```
          F = 0,
*         READ GATE ADDRESS
        ALU YBUS PASSR & DAQ & PCU ,,PCUDZ A7 B7 &
        AM2904 ,OECT & CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & TEST DIRIN UAEQB &
        CJPX ENM
02EE: 42 63 69 0F 7F FB 78 03 F3 53 XX XX

*         ZREG (GATE ADDRESS) + 2 -> R6 OF PCU
        ALU YBUS PASS & AB & PCU ,,PCUDA A4 B6 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL & CONT
02EF: 40 62 29 0F 59 89 78 07 F0 0E 00 00

*         R7 OF PCU -> PHREG, PUT IN QUEUE
        ALU YBUS PASS & AB & PCU ,,PCUZA A7 PHREG & AM2904 &
        DATAPATH & MEMCONT & CONTROL & JSB PUTINQ
02F0: 48 62 29 1F 4E C9 78 07 F0 01 03 45

*         TREG -> PHREG, JUMP TO END MONITOR
        ALU YBUS PASS & AB & PCU ,,PCUDZ PHREG PHREG &
        AM2904 & DATAPATH ,ENTREG & MEMCONT & CONTROL &
        JUMPX ENM
02F1: 48 62 09 1F 76 C9 78 07 F0 03 XX XX

*
* INPUT/OUTPUT OPERATION
* INITIAL STEP IS TO TRANSFER PARAMETERS FROM STACK TO PROCESS
          HEAD
* STACK :-
*         DEVICE
*         OPERATION ADDRESS
*         DATA ADDRESS
*
*         READ DEVICE, SP+2 -> MAR
INPOUTOP:  ALU YBUS PASS & AB & PCUPOP & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        CONT
02F2: 40 62 29 9F 18 7B 78 07 F0 0E 00 00

*         ZREG (DEVICE) -> DREG, SP+2 -> MAR, READ OPERATION AD
          DRESS
        ALU YBUS PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & CONT
02F3: 42 63 69 CF 18 7B 78 07 F0 0E 00 00

*         ZREG (OPERATION ADDRESS) -> WREG, SP+2 -> SP, READ DA
          TA ADDRESS
        ALU REG PASSR & DAQ & OEY & PCUPOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
02F4: C2 7B 69 0F 18 7B F9 87 F0 0E 00 00

*         PHREG+DEVICEATR -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD DEVICEATR & CONT
```

```
02F5: 40 62 29 9F 57 A8 78 07 F0 0E 00 24

*         WRITE DEVICE, R6+2 -> MAR, WREG (OPERATION ADDRESS) -
          > DREG
        ALU YBUS PASSR & AQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM WREG &
        CONT
02F6: 40 63 69 DF 19 BF F8 1F F0 0E 00 00

*         WRITE OPERATION ADDRESS
*         ZREG (DATA ADDRESS) -> DREG, R6+2 -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
02F7: 42 63 69 CF 19 BF 78 07 F0 0E 00 00

*         WRITE DATA ADDRESS
*         A JUMP IS NOW MADE TO A PIECE OF CODE WHICH READS
*         THESE ATTRIBUTES BACK
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP IOATR
02F8: 48 62 29 1F 30 3F 78 07 F0 03 02 F9

*
* READ THE I/O ATTRIBUTES FROM THE PROCESS HEAD
*
*         PHREG+DEVICEATR -> R6,MAR
IOATR:   ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD DEVICEATR & CONT
02F9: 40 62 29 9F 57 A8 78 07 F0 0E 00 24

*         READ DEVICE TYPE, R6+2 -> MAR
        ALU YBUS PASS & AB & PCU ,,PCUAB A4 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        CONT
02FA: 40 62 29 9F 19 BB 78 07 F0 0E 00 00

*         ZREG (DEVICE TYPE) -> WREG
*         READ OPERATION ADDRESS, R6+2 -> MAR
*         JUMP OUT IF NOT ZERO (I.E. DEVICE <> VDU)
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 ,OECT & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        TEST DIRIN UANEB & XCJP NOTVDU
02FB: C2 7B 69 8F 19 BB F9 83 F3 43 XX XX

*         ARRIVED HERE IF DEVICE = VDU
*         ZREG (OPERATION ADDRESS) -> R7,MAR, READ CHARACTER AD
          DRESS
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL & CONT
02FC: 40 62 29 8F 7F FB 78 07 F0 0E 00 00

*         READ OPERATION, ZREG (CHARACTER ADDRESS) -> R6 OF PCU
```

```
*         LOAD COUNTER WITH VDU INPUT ADDRESS
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ & CONTROL &
        LDCTA VDUINPUT
02FD: 40 62 29 0F 7D BB 78 07 F0 0C 02 FF

*         ZREG (OPERATION) -> YBUS, JUMP TO VDU INPUT CODE IF =
           0,
*         JUMP TO VDU OUTPUT CODE IF <>0
        ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL & TEST DIRIN UANEB & JRP VDUOUTPUT
02FE: 42 63 69 0F 30 09 78 03 F3 47 03 06

*
* VDU INPUT CODE
*
*         ACIA CONTROL REGISTER ADDRESS -> Q,MAR
VDUINPUT:  ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD ACIACS & CONT
02FF: 40 62 29 9E 70 28 78 07 F0 0E 40 68

*         READ ACIA CONTROL REGISTER, R6+1 (ACIA DATA REGISTER
          ADDRESS) -> MAR
*         $0001 -> WREG (MASK FOR LATER)
        ALU REG PASSR & DAQ & PCU QEU ,PCUAQ INCREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,,MBYTE &
        CONTROL ROM & RTB & BRAM WREG & IMMD $0001 & CONT
0300: C2 7B 69 9E 08 38 F9 87 F0 0E 00 01

*         ZREG (ACIA CR) AND WREG ($0001) -> YBUS, RETURN TO SC
          HEDULER IF SIGN BI
*         DATA AVAILABLE YET).
        ALU YBUS AND & DAB & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & TEST DIRIN UAEQB &
        CJP IOPRES1
0301: C2 66 29 0F 30 29 F9 83 F3 53 03 0B

*         READ ACIA DATA REGISTER
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL & CONT
0302: 40 62 29 1F 30 39 78 07 F0 0E 00 00

*         ZREG (CHARACTER INPUT) -> DREG, R6 (CHARACTER ADDRESS
          ) -> MAR.
        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
0303: 42 63 69 CF 4D A9 78 07 F0 0E 00 00

*         WRITE CHARACTER,1 -> DREG, R7 (OPERATION ADDRESS) ->
          MAR
        ALU LUR LOW & AB & OEY & PCU ,,PCUZA A7 B7 &
        AM2904 SHIFTEN & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $11 & CONT
```

```
0304: C0 4C 29 DF 4F FF F1 8F F0 0E 00 00

*         WRITE NEW OPERATION = OUTPUT TO PROCESS HEAD
*         JUMP TO VDU OUTPUT
*         THIS CAUSES THE CHARACTER TO BE ECHOED
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP VDUOUTPUT
0305: 48 62 29 1F 30 3F 78 07 F0 03 03 06

*
* VDU OUTPUT
*
*         ACIA CONTROL REGISTER ADDRESS -> R7,MAR
VDUOUTPUT:  ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD ACIACS & CONT
0306: 40 62 29 9F 7F E8 78 07 F0 0E 40 68

*         READ ACIA CONTROL REGISTER, R6 (CHAR ADDRESS) -> MAR
*         $0002 -> WREG (USED AS A MASK LATER)
        ALU REG PASSR & DAQ & PCU ,,PCUZA A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,,MBYTE &
        CONTROL ROM & RTB & BRAM WREG & IMMD $0002 & CONT
0307: C2 7B 69 9F 4D B8 F9 87 F0 0E 00 02

*         ZREG (ACIACR) AND XREG ($0002) -> YBUS, PC -> MAR, RE
          AD CHARACTER
*         RETURN TO SCHEDULER IF = 0 (I.E. DATA NOT READY YET)
        ALU YBUS AND & DAB & PCUNOP & AM2904 SHIFTEN OECT &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB MREQ &
        CONTROL ROM & RTB &
        BRAM WREG & TEST DIRIN UAEQB & CJP IOPRES1
0308: C2 66 29 8F 30 3B F1 83 F3 53 03 0B

*         ZREG (CHARACTER) -> DREG, R7+1 -> MAR, READ INSTR N+2

        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUAB INCREG B7 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL & CONT
0309: 42 63 69 CF 19 FB 78 07 F0 0E 00 00

*         WRITE RESULT TO ACIA
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & JMAP
030A: 48 62 29 9F 18 3D 78 07 F0 02 00 00

*         TREG (OLD SP) -> SP
IOPRES1:  ALU YBUS PASS & AB & PCU ,,PCUDZ A1 B1 & AM2904 &
        DATAPATH ,ENTREG & MEMCONT & CONTROL & CONT
030B: 40 62 09 1F 72 49 78 07 F0 0E 00 00

*
* FALL INTO IOPRES
*
*
* COPY PROCESS PRIORITY INTO VREG AND THEN JUMP TO RESCHEDULE
*
```

```
*         PHREG+PRIATR -> MAR
IOPRES:   ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD PRIATR & CONT
030C: 40 62 29 9E 56 28 78 07 F0 0E 00 0E

*         READ PRIORITY, 1 -> DREG
        ALU LUR LOW & AQ & PCUNOP & AM2904 SHIFTEN &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
        CONTROL ROM & RTB &
        BRAM VREG & CNTLB $11 & CONT
030D: C0 4C 69 5F 30 3B F3 8F F0 0E 00 00

*         ZREG (PRIORITY) - 1 -> VREG, IF >= 0 THEN SKIP NEXT I
          NSTRUCTION
        ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM VREG & TEST DIRIN SAGEB &
        CJP PREPARE
030E: C2 61 A9 0F 30 29 FB 83 F3 23 03 84

*         WRITE PRIORITY = 1
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP PREPARE
030F: 48 62 29 1F 30 3F 78 07 F0 03 03 84

*
* STOP PROGRAM
*
*         READ PROCESS NO., ZREG (RESULT) -> DREG,SP+2 -> SP
STOP:   ALU YBUS PASSR & DAQ & PCUPOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD ,ENZ0 & MEMCONT REQB MREQ &
        CONTROL & CONT
0310: 42 63 69 4F 18 7B 78 07 F0 0E 00 00

*         2*(ZREG(PROCESS NO.) + WREG (NEXT INDEX ADDRESS/2)) -
          > MAR
        ALU LUR ADD & DAB & OEY & PCUNOP & AM2904 SHIFTEN &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $10 & CONT
0311: C2 49 AB 8F 30 29 F1 87 F0 0E 00 00

*         READ PROCESS HEAD ADDRESS, RESULT ATTRIBUTE DISPLACEM
          ENT -> R6
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH & MEMCONT REQB MREQ & CONTROL & IMMD RESATR &
        CONT
0312: 40 62 29 1F 7D BA 78 07 F0 0E 00 06

*         ZREG (PROCESS HEAD ADDRESS) + R6 -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDA A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
        CONT
0313: 40 62 29 8F 5D A9 78 07 F0 0E 00 00

*         WRITE RESULT, CONTINUE ATTRIBUTE - RESULT ATTRIBUTE +
           R6 -> MAR, 0 -> D
```

```
       ALU YBUS LOW & AQ & OEY & PCU ,,PCUDA A6 B6 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB MREQ ,WRITE & CONTROL &
       IMMD CONTATR-RESATR & CONT
0314: 40 64 69 DF 5D BE 78 07 F0 0E 00 0E

*        WRITE 0 TO CONTINUE, PC -> MAR, JUMP RNI
       ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & JUMPX RNI
0315: 48 62 29 9F 30 3F 78 07 F0 03 XX XX

*
* SET HEAP REGISTER
*
*        READ NEW HEAP VALUE FROM STACK, SP+2 -> SP
SETHREG:  ALU YBUS PASS & AB & PCUPOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ & CONTROL & CONT
0316: 40 62 29 1F 18 7B 78 07 F0 0E 00 00

*        ZREG (NEW HEAP VALUE) -> HREG, PC -> MAR
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM HREG &
       JUMPX RNI
0317: CA 7B 69 8F 30 29 F9 07 F0 03 XX XX

*
* PUT PROCESS IN 1 SECOND QUEUE.
*
*          PC+2 -> PC (MAKES PC POINT TO NEXT INSTRUCTION AFTE
         R "SERVE" SUBROUTI
* PRE-EMPT
WAIT30:  ALU YBUS PASS & AB & PCUNEXT & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB PREEMPT
0318: 48 62 29 1F 18 09 78 07 F0 01 03 6F

*        1 SEC QUEUE ADDRESS -> R6
       ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD ONESECQ & CONT
0319: 40 62 29 1F 7D 88 78 07 F0 0E 00 24

*        PUT IN ONE SECOND QUEUE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB PUTINQ
031A: 48 62 29 1F 30 09 78 07 F0 01 03 45

*        RESCHEDULE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP RESCHED
031B: 48 62 29 1F 30 09 78 07 F0 03 03 8A

*
* TERMINATED EXCEPTION
*
*        RESULT = TERMINATED -> DREG
TERMEXP:  ALU YBUS PASSR & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDD & MEMCONT & CONTROL &
```

```
       IMMD TERMINATED & CONT
031C:  42 63 29 5F 30 08 78 07 F0 0E 00 00

*        PHREG+RESATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RESATR & CONT
031D:  40 62 29 9E 56 28 78 07 F0 0E 00 06

*        WRITE RESULT, JUMP TO EXCEPTION
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP EXCEPTION
031E:  48 62 29 1F 30 3F 78 07 F0 03 03 31

*
* OVERFLOW EXCEPTION
*
*        RESULT = OVERFLOW -> DREG
OVFLOWEXP:  ALU YBUS PASSR & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT & CONTROL &
       IMMD OVERFLOWERR & CONT
031F:  42 63 29 5F 30 08 78 07 F0 0E 00 01

*        PHREG+RESATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RESATR & CONT
0320:  40 62 29 9E 56 28 78 07 F0 0E 00 06

*        WRITE RESULT, JUMP TO EXCEPTION
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP EXCEPTION
0321:  48 62 29 1F 30 3F 78 07 F0 03 03 31

*
* POINTER ERROR EXCEPTION
*
*        RESULT = POINTERERR -> DREG
POINTEXP:  ALU YBUS PASSR & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT & CONTROL &
       IMMD POINTERERR & CONT
0322:  42 63 29 5F 30 08 78 07 F0 0E 00 02

*        PHREG+RESATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RESATR & CONT
0323:  40 62 29 9E 56 28 78 07 F0 0E 00 06

*        WRITE RESULT, JUMP TO EXCEPTION
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP EXCEPTION
0324:  48 62 29 1F 30 3F 78 07 F0 03 03 31

*
* RANGE ERROR EXCEPTION
*
*        RESULT = RANGEERROR -> DREG
```

```
RANGEEXP:  ALU YBUS PASSR & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT & CONTROL &
       IMMD RANGEERROR & CONT
0325: 42 63 29 5F 30 08 78 07 F0 0E 00 03

*        PHREG+RESATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RESATR & CONT
0326: 40 62 29 9E 56 28 78 07 F0 0E 00 06

*        WRITE RESULT, JUMP TO EXCEPTION
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP EXCEPTION
0327: 48 62 29 1F 30 3F 78 07 F0 03 03 31

*
* VARIANT ERROR EXCEPTION
*
*        RESULT = VARIANTERR -> DREG
VARIANTEXP:  ALU YBUS PASSR & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT & CONTROL &
       IMMD VARIANTERR & CONT
0328: 42 63 29 5F 30 08 78 07 F0 0E 00 04

*        PHREG+RESATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RESATR & CONT
0329: 40 62 29 9E 56 28 78 07 F0 0E 00 06

*        WRITE RESULT, JUMP TO EXCEPTION
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP EXCEPTION
032A: 48 62 29 1F 30 3F 78 07 F0 03 03 31

*
* HEAP LIMIT EXCEPTION
*
*        RESULT = HEAPLIMIT -> DREG
HEAPEXP:  ALU YBUS PASSR & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT & CONTROL &
       IMMD HEAPLIMIT & CONT
032B: 42 63 29 5F 30 08 78 07 F0 0E 00 05

*        PHREG+RESATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RESATR & CONT
032C: 40 62 29 9E 56 28 78 07 F0 0E 00 06

*        WRITE RESULT, JUMP TO EXCEPTION
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP EXCEPTION
032D: 48 62 29 1F 30 3F 78 07 F0 03 03 31

*
* STACK LIMIT EXCEPTION
```

```
*
*         RESULT = STACKLIMIT -> DREG
STACKEXP:  ALU YBUS PASSR & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT & CONTROL &
       IMMD STACKLIMIT & CONT
032E: 42 63 29 5F 30 08 78 07 F0 0E 00 06

*         PHREG+RESATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RESATR & CONT
032F: 40 62 29 9E 56 28 78 07 F0 0E 00 06

*         WRITE RESULT, JUMP TO EXCEPTION
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JUMP EXCEPTION
0330: 48 62 29 1F 30 3F 78 07 F0 03 03 31

*
* EXCEPTION
*
*         BREG -> MAR, LINE ATTR -> R7 OF PCU
EXCEPTION:  ALU YBUS PASSR & AQ & OEY & PCU ,,PCUDZ A7 B7 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,YMAR LDMAR &
       MEMCONT REQB & CONTROL ROM & ARAM BREG &
       IMMD LINEATR & CONT
0331: 40 63 6B 9F 7F E8 F8 07 F0 0E 00 04

*         READ LINE, PHREG+JOBATR -> R6,MAR
       ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B6 & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       IMMD JOBATR & CONT
0332: 40 62 29 9F 57 BA 78 07 F0 0E 00 12

*         ZREG (LINE) -> DREG, PHREG+R7 (LINEATR) -> MAR, READ
          JOB MODE
       ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUAB PHREG B7 &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & CONT
0333: 42 63 69 CF 17 FB 78 07 F0 0E 00 00

*         WRITE LINE, ZREG (JOB MODE) -> YBUS, SYSTEM ERROR IF
          = 0
       ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ ,WRITE & CONTROL &
       TEST DIRIN UAEQB & CJP SYSERR
0334: 42 63 69 0F 30 3F 78 03 F3 53 03 DF

*         R6 (JOB ATTRIBUTE ADDRESS) -> MAR, 0 -> DREG
       ALU YBUS LOW & AQ & OEY & PCU ,,PCUZA A6 B6 & AM2904 &
       DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB & CONTROL &
       CONT
0335: 40 64 69 DF 4D A9 78 07 F0 0E 00 00

*         WRITE NEW JOB MODE, GREG -> BREG, JUMP TO EXT
       ALU REG PASSR & AB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH & MEMCONT REQB MREQ ,WRITE &
```

```
       CONTROL ROM & RTB & ARAM GREG & BRAM BREG &
       JUMPX EXT
0336: C8 7B 29 1F 30 3F F8 0F F0 03 XX XX

*
* SUBROUTINES
*
* RUNNING.ENTER
* SET PRIORITY TO 0
* INCREMENT NESTING
* CORRUPTS Q OF PCU
*
*        0-> DREG, PHREG+PRIATR -> MAR
RUNENT:  ALU YBUS LOW & AQ & OEY & PCU QEU ,PCUDA PHREG &
       AM2904 & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
       CONTROL & IMMD PRIATR & CONT
0337: 40 64 69 DE 56 28 78 07 F0 0E 00 0E

*        WRITE 0 TO PRIORITY, PHREG+NESTATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
       CONTROL & IMMD NESTATR & CONT
0338: 40 62 29 9E 56 3E 78 07 F0 0E 00 0C

*        READ NESTING ATTRIBUTE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ & CONTROL & CONT
0339: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*        ZREG (NESTING) + 1 -> DREG
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD ,ENZ0 &
       MEMCONT REQB & CONTROL & CONT
033A: 42 63 69 4F 30 29 78 07 F4 0E 00 00

*        WRITE NESTING ATTRIBUTE, RETURN.
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
033B: 48 62 29 1F 30 3F 78 07 F0 0A 00 00

*
* RUNNING.LEAVE
*
* DECREMENTS NESTING
* IF NESTING = 0 (I.E. PROCESS IS NO LONGER IN A MONITOR)
* THE PRIORITY IS SET TO 2 AND THE PROCESS IS RESCHEDULED
* CORRUPTS Q OF PCU
*
*        PHREG+NESTATR -> MAR
RUNLEAVE:  ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG &
       AM2904 & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
       CONTROL & IMMD NESTATR & CONT
033C: 40 62 29 9E 56 28 78 07 F0 0E 00 0C

*        READ NESTING, 1 -> WREG,QREG
       ALU RQPT PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM WREG & IMMD 1 & CONT
```

```
033D: C2 3B 69 1F 30 3A F9 87 F0 0E 00 01

*         ZREG (NESTING) - 1 -> DREG, SET MSR
        ALU YBUS SUBS & DAB & OEY & PCUNOP &
        AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,,LDD ,ENZ0 & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & TEST MACHINE DLOAD &
        CONT
033E: C2 61 29 4F 30 29 F9 84 16 7E 00 00

*         WRITE NESTING, PC -> MAR, JUMP TO START1 IF MSR <> 0
*         WREG + 1 -> WREG
        ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        ARAM WREG & BRAM WREG & TEST MACHINE UANEB &
        CJPX START1
033F: C0 7B 69 9F 30 3F F9 9B F6 43 XX XX

*         PHREG+PRIATR -> MAR, WREG (2) -> VREG,DREG
        ALU REG PASSR & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM VREG &
        ARAM WREG & IMMD PRIATR & CONT
0340: C0 7B 69 DE 56 28 FB 9F F0 0E 00 0E

*         WRITE NEW PRIORITY, JUMP TO RESCHEDULE
*         VREG (PRIORITY) - QREG (1) -> VREG
        ALU REG SUBS & AQ & PCUNOP & AM2904 & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT REQB MREQ ,WRITE & CONTROL ROM &
        RTB & ARAM VREG & BRAM VREG & JUMP PREPARE
0341: C8 79 69 1F 30 3F FB BF F4 03 03 84

*
* INITIALISE QUEUE.
* SET HEAD DISPLACEMENT = TAIL DISPLACEMENT = 0
* AT START    R6 = QUEUE ADDRESS
* CORRUPTS    Q OF PCU
*
*         R6 -> MAR (HEAD ADDRESS), 0 -> DREG
QINIT:   ALU YBUS LOW & AB & PCU ,,PCUZA A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB & CONTROL &
        CONT
0342: 40 64 29 DF 4D A9 78 07 F0 0E 00 00

*         R6+2 -> MAR (TAIL ADDRESS), WRITE HEAD VALUE
        ALU YBUS PASS & AB & PCU QEU ,PCUAB A4 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & CONT
0343: 40 62 29 9E 19 BF 78 07 F0 0E 00 00

*         WRITE TAIL VALUE, RETURN
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
0344: 48 62 29 1F 30 3F 78 07 F0 0A 00 00

*
* PUT IN QUEUE.
```

```
* SET QUEUE TAIL ENTRY TO PROCESS HEAD REGISTER
* INCREMENT TAIL DISPLACEMENT (MOD QUEUE LENGTH)
* AT START    R6 = QUEUE ADDRESS
* CORRUPTS    Q OF PCU,QREG,XREG,EXP2.
*
*         R6+2 -> VREG,MAR (TAIL ADDRESS)
PUTINQ:  ALU REG PASS & AB & YOFF & PCU QEU ,PCUAB A4 B6 &
        AM2904 & DATAPATH ,,,,,PCUY LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM EXP2 & CONT
0345: C1 7A 28 9E 19 A9 FF 87 F0 0E 00 00

*         READ TAIL DISPLACEMENT, PHREG -> DREG, EXP2+2 -> QREG

        ALU QPT ADD & DAB & YOFF & PCU ,,PCUZA PHREG PHREG &
        AM2904 & CARRYCTL & DATAPATH ,,,,,PCUY ,LDD &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM EXP2 &
        IMMD 2 & CONT
0346: C3 31 A8 5F 46 FA FF 87 F0 0E 00 02

*         ZREG (TAIL DISPLACEMENT) + QREG (START OF QUEUE) -> M
          AR
        ALU YBUS ADD & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
0347: 42 61 EB 8F 30 29 78 07 F0 0E 00 00

*         WRITE PROCESS HEAD TO QUEUE TAIL, ZREG (TAIL DISPLACE
          MENT) + 2 -> XREG,
        ALU REG PASS & AB & YOFF & PCU QEU ,PCUDA A4 &
        AM2904 & DATAPATH ,,,,,PCUY ,LDD ,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM XREG & CONT
0348: C1 7A 28 4E 58 3F FA 07 F0 0E 00 00

*         XREG - QUEUE LENGTH -> XREG, SET MSR
*         R6+2 -> MAR (TAIL ADDRESS)
        ALU REG SUBR & DAB & OEY & PCU QEU ,PCUAB A4 B6 &
        AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR & MEMCONT & CONTROL ROM & RTB &
        BRAM XREG & TEST MACHINE DLOAD & IMMD QLENGTH & CONT
0349: C2 78 A9 9E 19 88 FA 04 16 7E 00 08

*         TEST MSR, IF <0 THEN SKIP NEXT INSTRUCTION
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT REQB & CONTROL &
        TEST MACHINE NEGATIVE & CJP *+12
034A: 40 62 29 1F 30 29 78 03 F2 F3 03 4C

*         XREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
        CONTROL ROM & ARAM XREG & CONT
034B: 40 63 69 5F 30 29 F8 27 F0 0E 00 00

*         WRITE NEW TAIL DISPLACEMENT, RETURN.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
034C: 48 62 29 1F 30 3F 78 07 F0 0A 00 00
```

```
*
* EXAMINE QUEUE TO SEE IF IT IS EMPTY OR NOT.
* MSR SET TO <> IF ANY ENTRIES FOUND
* QUEUE EMPTY IF HEAD DISPLACEMENT = TAIL DISPLACEMENT
* AT START   R7 = MONITOR ADDRESS
* AT END   R7 = GATE ADDRESS, WREG = NO. OF ENTRIES * 2, QREG
        CORRUPTED
*
*       R7 (MONITOR ADDRESS) -> MAR
EXAMINEQ:  ALU YBUS PASS & AB & PCU ,,PCUZA A7 B7 & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & CONT
034D: 40 62 29 9F 4F E9 78 07 F0 0E 00 00

*       READ GATE ADDRESS, 2 -> QREG
       ALU QPT PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
       CONTROL ROM & IMMD 2 & CONT
034E: 42 33 69 1F 30 3A F8 07 F0 0E 00 02

*       ZREG (GATE ADDRESS) -> R7, ZREG (GATE ADDRESS) + 2 ->
         TREG,MAR (HEAD AD
       ALU YBUS ADD & DAQ & OEY & PCU ,,PCUDZ A7 B7 &
       AM2904 & CARRYCTL &
       DATAPATH ,,LDTREG ,,YMAR LDMAR ,,ENZ0 & MEMCONT REQB &
       CONTROL ROM & RTB & BRAM WREG & JUMP EXQ2
034F: CA 61 FB 8F 7F E9 F9 87 F0 03 03 51

*       ALTERNATE ENTRY POINT FOR R7 = QUEUE ADDRESS
*       R7 -> TREG,MAR, 2 -> QREG
EXQ1:  ALU QPT PASSR & DAQ & YOFF & PCU ,,PCUZA A7 B7 &
       AM2904 & CARRYCTL & DATAPATH ,,LDTREG ,,PCUY LDMAR &
       MEMCONT REQB & CONTROL & IMMD 2 & CONT
0350: 43 33 78 9F 4F E8 78 07 F0 0E 00 02

*       READ HEAD DISPLACEMENT, TREG+2 -> MAR (TAIL ADDRESS)
EXQ2:  ALU YBUS ADD & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,ENTREG ,,,YMAR LDMAR &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       CONT
0351: C2 61 CB 9F 30 3B F9 87 F0 0E 00 00

*       ZREG (HEAD) -> WREG, READ TAIL
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
       CONT
0352: C2 7B 69 0F 30 3B F9 87 F0 0E 00 00

*       ZREG (TAIL) - WREG (HEAD) -> WREG, SET MSR, RETURN IF
         > 0
       ALU REG SUBS & DAB & OEY & PCUNOP &
       AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM WREG &
       TEST DIRIN POSITIVE & CRTN
0353: C2 79 29 0F 30 29 F9 80 17 EA 00 00
```

```
*         WREG IS NEGATIVE, ADD QUEUE LENGTH
        ALU REG ADD & DAB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT REQB & CONTROL ROM & RTB &
        BRAM WREG & IMMD QLENGTH & RTN
0354: CA 79 A9 1F 30 28 F9 87 F0 0A 00 08

*
* TRANSFER FROM MONITOR QUEUE TO PRIORITY 0 QUEUE.
* AT START    R7 = GATE ADDRESS
* CORRUPTED    R6,XREG,VREG,TREG
* 1) FETCH FROM MONITOR QUEUE
*
*         SAVE PHREG IN TREG
MPUTQ:   ALU YBUS PASS & AB & YOFF & PCU ,,PCUZA PHREG PHREG &
        AM2904 & DATAPATH ,,LDTREG ,,PCUY & MEMCONT &
        CONTROL & CONT
0355: 41 62 38 1F 46 C9 78 07 F0 0E 00 00

*         R6 + 2 -> R6 (MONITOR QUEUE START), JUMP TO GET QUEUE
           SUBROUTINE.
        ALU YBUS PASS & AB & PCU ,,PCUAB A4 B6 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        JSB GETQ
0356: 48 62 29 0F 19 89 78 07 F0 01 03 5A

* 2) PROCESS IS NOW IN PHREG, PUT IT IN PRIORITY 0 QUEUE.
*         PRIORITY 0 QUEUE ADDRESS -> R6 OF PCU
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & IMMD QSTART & CONT
0357: 40 62 29 1F 7D 88 78 07 F0 0E 00 00

*         JUMP TO PUT QUEUE SUBROUTINE.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & JSB PUTINQ
0358: 48 62 29 1F 30 09 08 07 F0 01 03 45

*         TREG (CURRENT PHREG) -> PHREG, RETURN
        ALU YBUS PASS & AB & PCU ,,PCUDZ PHREG PHREG &
        AM2904 & DATAPATH ,ENTREG & MEMCONT & RTN
0359: 48 62 09 1F 76 C9 08 07 F0 0A 00 00

*
* GET NEXT QUEUE ENTRY
* AT START    R6 = QUEUE ADDRESS
* AT END    PHREG = PROCESS HEAD ADDRESS
* CORRUPTS    QREG, R6 OF PCU, Q OF PCU, XREG.
*
*         R6 (QUEUE ADDRESS) -> MAR
GETQ:   ALU YBUS PASS & AB & PCU ,,PCUZA A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONT
035A: 40 62 29 9F 4D A9 08 07 F0 0E 00 00

*         READ HEAD, 2 -> QREG, R6 + 4 -> Q OF PCU
        ALU QPT PASSR & DAQ & PCU QEU ,PCUAB A5 B6 & AM2904 &
        DATAPATH & MEMCONT REQB MREQ & IMMD 2 & CONT
035B: 42 33 69 1E 1B BA 08 07 F0 0E 00 02

*         ZREG (HEAD) + Q OF PCU -> MAR, ZREG (HEAD DISPLACEMEN
```

```
          T) + 2 -> QREG,DRE
        ALU QPT ADD & DAQ & PCU QEU ,PCUDQ & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
035C: 42 31 E9 CE 60 29 78 07 F0 0E 00 00

*         READ QUEUE ENTRY, QREG - QUEUE LENGTH -> XREG, SET MS
          R, R6+2 -> MAR.
        ALU REG SUBR & DAQ & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM XREG & TEST MACHINE DLOAD &
        IMMD QLENGTH & CONT
035D: C2 78 E9 9F 4D BA FA 04 16 7E 00 08

*         ZREG (ENTRY) -> PHREG, IF MSR <0 THEN SKIP NEXT INSTR
          UCTION.
        ALU YBUS PASS & AB & PCU ,,PCUDZ PHREG PHREG &
        AM2904 ,OECT & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB &
        CONTROL & TEST MACHINE NEGATIVE & CJP *+12
035E: 40 62 29 0F 76 E9 78 03 F2 F3 03 60

*         XREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL ROM &
        ARAM XREG & CONT
035F: 40 63 69 5F 30 29 F8 27 F0 0E 00 00

*         WRITE NEW HEAD ADDRESS, RETURN.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
0360: 48 62 29 1F 30 3F 78 07 F0 0A 00 00

*
* SERVE SUBROUTINE
* LOADS UP HREG,BREG,GREG,SP,PC FROM PROCESS HEAD
* CORRUPTS R6, Q OF PCU, QREG, TREG
*
*         PHREG+PCREG -> R6,MAR
SERVE:   ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD PCREG & CONT
0361: 40 62 29 9F 57 A8 78 07 F0 0E 00 2E

*         READ PC, R6+2 -> MAR,WREG, 2 -> QREG
        ALU RQPT PASSR & DAQ & YOFF & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,PCUY LDMAR &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        IMMD 2 & CONT
0362: C3 3B 68 9F 19 BA F9 87 F0 0E 00 02

*         ZREG -> PC, WREG + QREG (2) -> WREG,MAR, READ SP
        ALU REG ADD & AQ & OEY & PCU ,,PCUDZ A0 B0 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & ARAM WREG &
        BRAM WREG & CONT
0363: C0 79 EB 8F 70 3B F9 9F F0 0E 00 00
```

```
*         ZREG -> SP, WREG + QREG (2) -> TREG, READ GREG
        ALU YBUS ADD & AQ & OEY & PCU ,,PCUDZ A1 B1 & AM2904 &
        CARRYCTL & DATAPATH ,,LDTREG ,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & ARAM WREG &
        CONT
0364: C0 61 F9 0F 72 7B F8 1F F0 0E 00 00

*         TREG -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,ENTREG ,,,,LDMAR & MEMCONT REQB & CONTROL &
        CONT
0365: 40 62 09 9F 7D A9 78 07 F0 0E 00 00

*         R6+2 -> MAR, ZREG -> GREG, READ BREG
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM GREG &
        CONT
0366: C2 7B 69 8F 19 BB F8 87 F0 0E 00 00

*         ZREG -> BREG, READ HREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
        CONT
0367: C2 7B 69 0F 30 3B F8 07 F0 0E 00 00

*         ZREG -> HREG, PC - 4 -> PC
        ALU REG PASSR & DAQ & OEY & PCU ,SUB PCUAB A5 B0 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM HREG & CONT
0368: C2 7B 69 0F 9A 09 F9 07 F0 0E 00 00

*         PHREG+SLICE -> MAR, 0 -> DREG
        ALU YBUS LOW & AB & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL & IMMD SLICEATR & CONT
0369: 40 64 29 DE 56 28 78 07 F0 0E 00 0A

*         WRITE 0 TO SLICE, PHREG+OVERTIME -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD OVERTIME & CONT
036A: 40 62 29 9E 56 3E 78 07 F0 0E 00 10

*         WRITE 0 TO OVERTIME, RTREC ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD RTREC & CONT
036B: 40 62 29 9E 70 3E 78 07 F0 0E 00 3E

*         READ RTREC, LASTTIME ADDRESS -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ &
        CONTROL & IMMD LASTTIME & CONT
036C: 42 63 69 9E 56 3A 78 07 F0 0E 00 2C

*         ZREG (RTREC) -> DREG
```

```
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        DATAPATH ,,,,,,,LDD ,ENZ0 & MEMCONT REQB & CONTROL &
        CONT
036D: 42 63 69 4F 30 29 78 07 F0 0E 00 00

*         WRITE LASTTIME,RETURN
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
036E: 48 62 29 1F 30 3F 78 07 F0 0A 00 00

*
* PRE-EMPT PROCESS SUBROUTINE.
* DUMP GREG,BREG,HREG,SP,PC TO PROCESS HEAD
*
*         PHREG+PCREG -> MAR
PREEMPT:  ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG &
        AM2904 & DATAPATH ,,,,,,LDMAR & MEMCONT & CONTROL &
        IMMD PCREG & CONT
036F: 40 62 29 9E 56 08 78 07 F0 0E 00 2E

*         PC -> DREG
        ALU YBUS PASS & AB & YOFF & PCUNOP & AM2904 &
        DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB & CONTROL &
        CONT
0370: 41 62 28 5F 30 29 78 07 F0 0E 00 00

*         WRITE PC, SP -> DREG
        ALU YBUS PASS & AB & YOFF & PCUSP & AM2904 &
        DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB MREQ ,WRITE &
        CONTROL & CONT
0371: 41 62 28 5F 32 7F 78 07 F0 0E 00 00

*         PHREG+SPREG -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD SPREG & CONT
0372: 40 62 29 9F 57 A8 78 07 F0 0E 00 30

*         WRITE SP, PHREG+REGG -> MAR, GREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM GREG &
        IMMD REGG & CONT
0373: 40 63 69 DE 56 3E F8 0F F0 0E 00 32

*         WRITE GREG, PHREG + REGB -> MAR, BREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM BREG &
        IMMD REGB & CONT
0374: 40 63 69 DE 56 3E F8 07 F0 0E 00 34

*         WRITE BREG, PHREG+REGH -> MAR, HREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM HREG &
        IMMD REGH & CONT
0375: 40 63 69 DE 56 3E F8 17 F0 0E 00 36
```

```
*         WRITE HREG, RETURN
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
0376: 48 62 29 1F 30 3F 78 07 F0 0A 00 00

*
* UPDATE PROCESS HEAD
*
* NEWSLICE = OLD SLICE + (RTREC-LASTTIME)
* LASTTIME = RTREC
* OVERTIME SET TO TRUE IF SLICE > MAXSLICE
* CORRUPTS Q OF PCU,R6,R7,WREG,XREG
*
*         SLICE ATTRIBUTE ADDRESS -> R7,MAR
UPDATE:   ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B7 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD SLICEATR & CONT
0377: 40 62 29 9F 57 E8 78 07 F0 0E 00 0A

*         READ SLICE, LASTTIME ADDRESS -> R6,MAR
        ALU YBUS PASS & AB & PCU ,,PCUDA PHREG B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        IMMD LASTTIME & CONT
0378: 40 62 29 9F 57 BA 78 07 F0 0E 00 2C

*         READ LASTTIME, ZREG (SLICE) -> XREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
        CONT
0379: C2 7B 69 0F 30 3B FA 07 F0 0E 00 00

*         RTREC ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD RTREC & CONT
037A: 40 62 29 9E 70 28 78 07 F0 0E 00 3E

*         READ RTREC, XREG (SLICE) - ZREG (LASTTIME) -> XREG
        ALU REG SUBR & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM XREG &
        CONT
037B: C2 78 A9 0F 30 3B FA 07 F4 0E 00 00

*         ZREG (RTREC) + XREG -> XREG,DREG
*         R7 (SLICE ADDRESS) -> MAR
*         SKIP NEXT INSTRUCTION IF >0
        ALU REG ADD & DAB & OEY & PCU ,,PCUZA A7 B7 &
        AM2904 ,OECT & CARRYCTL &
        DATAPATH ,,,,,,LDMAR LDD ,ENZ0 & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM XREG & TEST DIRIN POSITIVE &
        CJP *+12
037C: C2 79 A9 CF 4F E9 FA 03 F3 E3 03 7E

*         XREG + 10000 -> XREG,DREG
        ALU REG ADD & DAB & OEY & PCUNOP & AM2904 & CARRYCTL &
```

```
        DATAPATH ,,,,,,,LDD & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM XREG & IMMD 1000 & CONT
037D: C2 79 A9 5F 30 28 FA 07 F0 0E 03 C8

*         WRITE NEW SLICE, ZREG (RTREC) -> DREG, R6 -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD ,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
037E: 42 63 69 CF 4D BF 78 07 F0 0E 00 00

*         WRITE LAST TIME, XREG (NEW SLICE) - MAXSLICE -> YBUS,
           RETURN IF < 0
        ALU YBUS SUBR & DAB & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB MREQ ,WRITE &
        CONTROL ROM & RTB & BRAM XREG & IMMD MAXSLICE &
        TEST DIRIN UALSB & CRTN
037F: C2 60 A9 1F 30 3E FA 03 F7 AA 00 1A

*         1 -> DREG, OVERTIME ADDRESS -> MAR
        ALU LUR LOW & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 SHIFTEN & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $11 & IMMD OVERTIME & CONT
0380: C0 4C 69 DE 56 28 F1 8F F0 0E 00 10

*         WRITE OVERTIME, RETURN
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
0381: 48 62 29 1F 30 3F 78 07 F0 0A 00 00

*
* TEST FOR NULL PROCESS (PHREG = 0)
* IF SO THEN SKIP PREPARE CODE.
*
*         PHREG -> TREG
TESTNULL:  ALU YBUS PASS & AB & YOFF & AM2904 &
        PCU ,,PCUZA PHREG PHREG & DATAPATH ,,LDTREG ,,PCUY &
        MEMCONT & CONTROL & CONT
0382: 41 62 38 1F 46 C9 78 07 F0 0E 00 00

*         TREG -> YBUS, JUMP TO RESCHEDULE IF = 0
        ALU YBUS PASSR & DAQ & PCUNOP & AM2904 ,OECT &
        DATAPATH ,ENTREG & MEMCONT & CONTROL &
        TEST DIRIN UAEQB & CJP RESCHED
0383: 42 63 49 1F 30 09 78 03 F3 53 03 8A

*
* PREPARE A PROCESS FOR RESCHEDULING.
*
* PRE-EMPT IT AND PUT IT IN THE APPROPRIATE PRIORITY QUEUE.
* ASSUMES VREG = PRIORITY - 1
* STEP 1) CALCULATE THE APPROPRIATE QUEUE ADDRESS.
* PRIORITY 0 QUEUE ADDRESS -> R6,MAR, -VREG - 1 -> VREG, AM291
          0 PUSH
PREPARE:  ALU REG COMPLR & AQ & PCU ,,PCUDZ A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT &
        CONTROL ROM & RTB & ARAM VREG & BRAM VREG &
        IMMD QSTART &
```

```
        PUSH QSTART
0384: C0 7B E9 9F 7D 88 FB BF F0 04 00 00

*         VREG+1 -> VREG, SKIP NEXT INSTRUCTION IF <= 0
        ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
        RTB & ARAM VREG & BRAM VREG & TEST DIRIN SALEB &
        CJP *+12
0385: C0 7B 69 1F 30 09 FB BB F7 13 03 87

*         STRANGE LOOP STRUCTURE IS TO STOP THE
*         MICROCODE STICKING IN THE EVENT OF AM2904 FAILURE
*         EXIT LOOP HERE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JPOP PQ1
0386: 48 62 29 1F 30 09 78 07 F0 0B 03 88

*         R6 + QUEUE LENGTH -> R6, LOOP BACK

        ALU YBUS PASSR & DAQ & PCU ,,PCUDA A6 B6 &
        AM2904 ,OECT & CARRYCTL & DATAPATH & MEMCONT &
        CONTROL &
        TEST DIRIN NEGATIVE & IMMD TOTQLENGTH & LOOP
0387: 42 63 69 1F 5D 88 78 03 F3 FD 00 0C

*         STEP 2) PUT IN QUEUE
PQ1:   ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB PUTINQ
0388: 48 62 29 1F 30 09 78 07 F0 01 03 45

*         STEP 3) PRE-EMPT CURRENT PROCESS
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB PREEMPT
0389: 48 62 29 1F 30 09 78 07 F0 01 03 6F

*
* FALL INTO RESCHEDULE
*
*
***********
*
* RESCHEDULE
*
* SCAN EACH QUEUE UNTIL A READY-TO-RUN PROCESS IS FOUND
* IF ALL QUEUES ARE EMPTY THEN LOOP ROUND, UNLESS THERE IS AN
         INTERRUPT
*
*         PRIORITY 0 QUEUE ADDRESS -> R7
RESCHED:   ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD QSTART & CONT
038A: 40 62 29 1F 7F C8 78 07 F0 0E 00 00

*         TEST PRIORITY 0 QUEUE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB EXQ1
038B: 48 62 29 1F 30 09 78 07 F0 01 03 50

*         IF MSR <> THEN PROCESS FOUND, EXIT
```

```
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE UANEB &
        CJP PROCFOUND
038C:  40 62 29 1F 30 09 78 03 F2 43 03 95

*         PRIORITY 1 QUEUE ADDRESS -> R7
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH & MEMCONT & CONTROL &
        IMMD QSTART+TOTQLENGTH & CONT
038D:  40 62 29 1F 7F C8 78 07 F0 0E 00 0C

*         TEST PRIORITY 1 QUEUE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB EXQ1
038E:  48 62 29 1F 30 09 78 07 F0 01 03 50

*         IF MSR <> THEN PROCESS FOUND, EXIT
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE UANEB &
        CJP PROCFOUND
038F:  40 62 29 1F 30 09 78 03 F2 43 03 95

*         PRIORITY 2 QUEUE ADDRESS -> R7
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH & MEMCONT & CONTROL &
        IMMD QSTART+2*TOTQLENGTH & CONT
0390:  40 62 29 1F 7F C8 78 07 F0 0E 00 18

*         TEST PRIORITY 2 QUEUE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB EXQ1
0391:  48 62 29 1F 30 09 78 07 F0 01 03 50

*         IF MSR <> THEN PROCESS FOUND, EXIT
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE UANEB &
        CJP PROCFOUND
0392:  40 62 29 1F 30 09 78 03 F2 43 03 95

*         ARRIVED HERE IF ALL QUEUES EMPTY
*         IF NO INTERRUPT THEN WAIT, 0 -> TREG
        ALU YBUS LOW & AQ & OEY & PCUNOP & AM2904 &
        DATAPATH ,,LDTREG & MEMCONT & CONTROL &
        XJSB TIMERTEST
0393:  48 64 79 1F 30 09 78 07 F0 01 XX XX

*         TIMER INTERRUPT HAS OCCURED
*         TREG (0) -> PHREG
        ALU YBUS PASS & AB & PCU ,,PCUDZ PHREG PHREG &
        AM2904 & DATAPATH ,ENTREG & MEMCONT & CONTROL &
        JUMP TIMER
0394:  48 62 09 1F 76 C9 78 07 F0 03 03 BD

*         ARRIVED HERE IF THERE IS A READY TO RUN PROCESS
*         R7 CONTAINS THE QUEUE ADDRESS
*         R7 -> R6, GET PROCESS FROM QUEUE
PROCFOUND:  ALU YBUS PASS & AB & PCU ,,PCUZA A7 B6 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & JSB GETQ
```

```
0395: 48 62 29 1F 4F 89 78 07 F0 01 03 5A

*         SERVE PROCESS
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB SERVE
0396: 48 62 29 1F 30 09 78 07 F0 01 03 61

*         PC -> MAR, JUMP TO START1
        ALU YBUS PASS & AB & PCUNOP & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMPX START1
0397: 48 62 29 9F 30 29 78 07 F0 03 XX XX

*
* INITIALISATION
*
* 1) INITIALISE P+1 QUEUES.
*         QUEUE START ADDRESS -> R6,MAR.
IPINIT:   ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT & CONTROL &
        IMMD QSTART & CONT
0398: 40 62 29 9F 7D 88 78 07 F0 0E 00 00

*         P -> AM2910 COUNTER AND PUSH
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & PHLC P
0399: 48 62 29 1F 30 09 78 07 F0 04 00 03

*         JUMP TO INITIALISE QUEUE SUBROUTINE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & JSB QINIT
039A: 48 62 29 1F 30 09 08 07 F0 01 03 42

*         R6 + QUEUE LENGTH -> R6, REPEAT LOOP FOR NEXT QUEUE
        ALU YBUS PASS & AB & PCU ,,PCUDA A6 B6 & AM2904 &
        DATAPATH & MEMCONT & IMMD TOTQLENGTH & RFCT
039B: 40 62 29 1F 5D 88 08 07 F0 08 00 0C

* 2) SET REAL TIME RECORD TO ZERO.
*         0 -> DREG, REAL TIME RECORD ADDRESS -> R7,MAR.
        ALU YBUS LOW & AB & OEY & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB & CONTROL &
        IMMD RTREC & CONT
039C: 40 64 29 DF 7F E8 78 07 F0 0E 00 3E

*          WRITE REAL TIME RECORD, R7+2 -> MAR, SYSTEM HEAP INI
          TIAL VALUE -> WREG
        ALU REG PASSR & DAQ & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM WREG & IMMD SPARE+SPARELENGTH+PHLENGTH & CONT
039D: C2 7B 69 9F 19 FE F9 87 F0 0E 00 8C

* 3) SET THE VARIOUS REGISTERS AND POINTERS.
*         WRITE ZERO TO REAL TIME IN SECONDS, SYSTEM HEAP ADDRE
          SS -> MAR
*         WREG (SYSTEM HEAP INITIAL VALUE) -> DREG
        ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDZ & AM2904 &
```

```
       CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM WREG &
       IMMD SYSHEAP & CONT
039E:  40 63 69 DE 70 3E F8 1F F0 0E 00 30

*        WRITE SYSTEM HEAP INITIAL VALUE,
*        INITIAL PROCESS HEAD ADDRESS -> PHREG.
       ALU YBUS PASS & AB & PCU ,,PCUDZ PHREG PHREG &
       AM2904 & DATAPATH & MEMCONT REQB MREQ ,WRITE &
       CONTROL & IMMD SPARE+SPARELENGTH & CONT
039F:  40 62 29 1F 76 FE 78 07 F0 0E 00 4C

*        NEXT INDEX ADDRESS -> MAR, 0 -> DREG
       ALU LUR LOW & AB & OEY & PCU QEU ,PCUDZ &
       AM2904 SHIFTEN & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB & CONTROL ROM & RTB &
       BRAM WREG & CNTLB $10 & IMMD NEXTINDEX & CONT
03A0:  C0 4C 29 DE 70 28 F1 87 F0 0E 00 34

*        WRITE ZERO TO NEXT INDEX,
*        JUMP TO INITIALISE ATTRIBUTES SUBROUTINE.
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & JSB ATRINIT
03A1:  48 62 29 1F 30 3F 78 07 F0 01 03 AD

*        PROGSTART+6 -> PC,MAR (VARLENGTH ADDRESS).
       ALU YBUS PASS & AB & PCU ,,PCUDZ A0 B0 & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD PROGSTART+6 & CONT
03A2:  40 62 29 9F 70 28 78 07 F0 0E 80 06

*        READ VARIABLE LENGTH, PC-2  (PROGSTART+4 = STACKLENGT
         H ADDRESS) -> MAR
       ALU YBUS PASS & AB & PCU ,SUB PCUAB A4 B0 & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
       CONT
03A3:  40 62 29 9F 98 3B 78 07 F0 0E 00 00

*        SP - ZREG (VARIABLE LENGTH) -> SP,BREG, READ STACKLEN
         GTH
       ALU REG PASS & AB & YOFF & PCU ,SUB PCUDA A1 B1 &
       AM2904 & DATAPATH ,,,,,PCUY ,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM BREG &
       CONT
03A4:  C1 7A 28 0F D2 7B F8 07 F0 0E 00 00

*        BREG - ZREG (STACKLENGTH) -> QREG, PC - 2 (PROGSTART+
         2) -> MAR.
       ALU QPT SUBR & DAB & PCU ,SUB PCUAB A4 B0 & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL ROM & RTB & BRAM BREG & CONT
03A5:  C2 30 A9 8F 98 29 F8 07 F4 0E 00 00

*        READ CODE LENGTH, PC+6 -> PC (PROGSTART+8), QREG -> H
         REG,DREG
       ALU REG PASS & AQ & OEY & PCU ,,PCUDA A0 B0 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM HREG & IMMD 6 & CONT
```

```
03A6: C0 7A 69 5F 50 3A F9 07 F0 0E 00 06

*         LAST HEAP POINTER ADDRESS -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        DATAPATH ,,,,,YMAR LDMAR & MEMCONT REQB & CONTROL &
        IMMD LASTHEAP & CONT
03A7: 42 63 6B 9F 30 28 78 07 F0 0E 00 32

*         ZREG (CODELENGTH) + PC -> DREG (LARGE CONSTANT ADDRES
          S), WRITE LAST HEA
*         LOAD UP INTERRUPT COUNTER.
        ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDA A0 &
        AM2904 & DATAPATH ,,,,,PCUY ,LDD ,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL &
        INTLOAD -INTERVAL & CONT
03A8: 01 62 28 4E 50 3F 78 07 F0 0E 63 C0

*         PHREG + CONSTATR -> MAR.
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD CONSTATR & CONT
03A9: 40 62 29 9E 56 28 78 07 F0 0E 00 22

*         WRITE LARGE CONSTANT ADDRESS
*         SYSTEM HEAP POINTER ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD SYSHEAP & CONT
03AA: 40 62 29 9E 70 3E 78 07 F0 0E 00 30

*         READ SYSTEM HEAP POINTER, LOAD COUNTER WITH SYSTEM ER
          ROR
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & LDCTA SYSERR
03AB: 40 62 29 1F 30 3B 78 07 F0 0C 03 DF

*          HREG - ZREG (SYSTEM HEAP POINTER) -> YBUS
*          SYSTEM ERROR IF <= 0, PC -> MAR, ELSE JUMP TO START1

        ALU YBUS SUBR & DAB & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM HREG &
        TEST DIRIN UAGRB & XJRP START1
03AC: C2 60 A9 8F 30 29 F9 03 F7 C7 XX XX

*
* INITIALISE ATTRIBUTES SUBROUTINE.
*
* INDEX = NEXT INDEX, NEXT INDEX +:= 1
* PRIORITY = 2
* SLICE,NESTING,OVERTIME,JOB,CONTINUE,RUNTIME = 0
*          NEXT INDEX ADRESS -> MAR
ATRINIT:   ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD NEXTINDEX & CONT
03AD: 40 62 29 9E 70 28 78 07 F0 0E 00 34

*         READ NEXT INDEX, QUEUE LENGTH/2 -> XREG, 2+Q OF PCU -
```

```
          > Q OF PCU
        ALU REG PASSR & DAQ & OEY & PCU QEU ,PCUAQ A4 &
        AM2904 & CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM XREG & IMMD QLENGTH/2 &
        CONT
03AE: C2 7B 69 1E 08 3A FA 07 F0 0E 00 04

*         ZREG (NEXT INDEX) + 1 -> WREG,DREGZREG (NEXT INDEX) +
           Q OF PCU (NEXT IN
        ALU REG PASSR & DAQ & OEY & PCU QEU ,PCUDQ & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD ,ENZ0 &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM WREG & CONT
03AF: C2 7B 69 4E 60 29 F9 87 F4 0E 00 00

*         WRITE NEXT INDEX, WREG - XREG (QUEUE LENGTH) -> YBUS.

*         ZREG (NEXT INDEX) + Q OF PCU -> MAR (GENERATES PROCES
          S HEAD TABLE ENTRY
*         SYSTEM ERROR IF YBUS > 0.
        ALU YBUS SUBS & AB & OEY & PCU QEU ,PCUDQ &
        AM2904 ,OECT & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        ARAM WREG & BRAM XREG & TEST DIRIN UAGRB &
        CJPP SYSERR
03B0: C0 61 29 8E 60 3F FA 1B F7 CB 03 DF

*         PHREG -> DREG
        ALU YBUS PASS & AB & YOFF & PCU ,,PCUZA PHREG PHREG &
        AM2904 & DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB &
        CONTROL & CONT
03B1: 41 62 28 5F 46 E9 78 07 F0 0E 00 00

*         PHREG+INDEXATR -> MAR,WRITE PHREG, WREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM WREG &
        IMMD INDEXATR & CONT
03B2: 40 63 69 DE 56 3E F8 1F F0 0E 00 00

*         0 -> DREG,WREG, PHREG+RUNTIMEATR -> MAR, WRITE INDEX
        ALU LUR LOW & AB & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 SHIFTEN & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM WREG & CNTLB $10 & IMMD RUNATR & CONT
03B3: C0 4C 29 DE 56 3E F1 87 F0 0E 00 08

*         PHREG+SLICE -> MAR, WRITE RUNTIME
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD SLICEATR & CONT
03B4: 40 62 29 9E 56 3E 78 07 F0 0E 00 0A

*         PHREG+NESTING -> MAR, WRITE SLICE ATTRIBUTE
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD NESTATR & CONT
03B5: 40 62 29 9E 56 3E 78 07 F0 0E 00 0C
```

```
*         PHREG+OVERTIME -> MAR, WRITE NESTING ATTRIBUTE
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD OVERTIME & CONT
03B6: 40 62 29 9E 56 3E 78 07 F0 0E 00 10

*         PHREG+JOB -> MAR, WRITE OVERTIME ATTRIBUTE
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD JOBATR & CONT
03B7: 40 62 29 9E 56 3E 78 07 F0 0E 00 12

*         PHREG+WAITING -> MAR, WRITE JOB ATTRIBUTE
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD WAITING & CONT
03B8: 40 62 29 9E 56 3E 78 07 F0 0E 00 2A

*         PHREG + LASTTIME -> MAR, WRITE WAITING ATTRIBUTE
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD LASTTIME & CONT
03B9: 40 62 29 9E 56 3E 78 07 F0 0E 00 2C

*         PHREG + CONTINUE -> MAR, WRITE LASTTIME ATTRIBUTE,1 -
          > DREG
        ALU LUR LOW & AQ & PCU QEU ,PCUDA PHREG &
        AM2904 SHIFTEN & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM YREG & CNTLB $11 & IMMD CONTATR & CONT
03BA: C0 4C 69 DE 56 3E F2 8F F0 0E 00 14

*         PHREG+PRIATR -> MAR, 2 -> DREG ((WREG+1)*2)
*         WRITE CONTINUE ATTRIBUTE
        ALU LUR PASS & AB & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 SHIFTEN & CARRYCTL COEQ1 &
        MEMCONT REQB MREQ ,WRITE & DATAPATH ,,,,,,LDMAR LDD &
        CONTROL ROM & RTB &
        BRAM WREG & CNTLB $10 & IMMD PRIATR & CONT
03BB: C0 4A 29 DE 56 3E F1 87 F4 0E 00 0E

*         WRITE PRIORITY ATTRIBUTE, RETURN
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONTROL & RTN
03BC: 48 62 29 1F 30 3F 78 07 F0 0A 00 00

*
*               ********************************************
*               *    TIMER INTERRUPT CODE                  *
*               ********************************************
*
* 1) SEND A MESSAGE TO NEIGHBOUR
* 2) UPDATE REAL TIME DATA
* 3) SEE IF THE CURRENT PROCESS NEEDS RESCHEDULING
*
*         IF THERE IS NO INTERRUPT ON THE TEST TREE
*         THEN SOMETHING'S WRONG.
```

```
TIMER:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & TESTF INTERRUPT & XCJP STIMESS
03BD: 40 62 29 1F 30 09 78 07 F0 73 XX XX

*         PERFORM A TEST ON THE PCU HERE.
*         THIS IS MAINLY BECAUSE A FAULTY PCU
*         COULD CAUSE THE SAME INSTRUCTION TO
*         BE REPEATEDLY EXECUTED. AN ERROR WOULD
*         ONLY BE DETECTED WHEN A TIMER INTERRUPT OCCURED.
*         THIS TEST PREVENTS THE MICROCODE FROM
*         STICKING HERE.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & XJSB PATEST
03BE: 48 62 29 1F 30 09 78 07 F0 01 XX XX

*         RELOAD INTERRUPT COUNTER, PHREG -> REAL1I
        ALU REG PASS & AB & YOFF & PCU ,,PCUZA PHREG PHREG &
        AM2904 & DATAPATH ,,,,,PCUY & CONTROL ROM & RTB &
        BRAM REAL1I & MEMCONT & INTLOAD -INTERVAL & CONT
03BF: 81 7A 28 1F 46 C9 FC 87 F0 0E 63 C0

*         RTREC ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD RTREC & CONT
03C0: 40 62 29 9E 70 28 78 07 F0 0E 00 3E

*         READ REAL TIME RECORD, CLOCK INCREMENT VALUE - 1000
          -> WREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM WREG & IMMD CLOCKINCR-1000 &
        CONT
03C1: C2 7B 69 1F 30 3A F9 87 F0 0E FC 2C

*         ZREG (REAL TIME RECORD) + WREG -> WREG,DREG, SKIP NEX
          T INSRUCTION IF >=
*         STRANGE CODE STRUCTURE IS TO PREVENT
*         THE MICROCODE FROM "STICKING" IN THE
*         EVENT OF PROCESSOR FAILURE.
        ALU REG ADD & DAB & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,LDD ,ENZ0 & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & TEST DIRIN POSITIVE &
        CJP *+12
03C2: C2 79 A9 4F 30 29 F9 83 F3 E3 03 C4

*         NOT 1 SEC., JUMP FORWARD
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP NOT1SEC
03C3: 48 62 29 1F 30 09 78 07 F0 03 03 CD

*         ARRIVED HERE IF NEXT 1 SEC. INTERVAL HAS OCCURED.
*         WRITE REAL TIME RECORD, RTSECS ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & IMMD RTSECS & CONT
03C4: 40 62 29 9E 70 3E 78 07 F0 0E 00 40
```

```
*         READ REAL TIME SECONDS ADDRESS
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & CONT
03C5:  40 62 29 1F 30 3B 78 07 F0 0E 00 00

*         ZREG (REAL TIME IN SCEONDS) + 1 -> DREG
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD ,ENZ0 &
        MEMCONT REQB & CONTROL & CONT
03C6:  42 63 69 4F 30 29 78 07 F4 0E 00 00

*         WRITE REAL TIME IN SECONDS, 1 SEC. QUEUE ADDRESS -> R
          7, 1 -> YREG
        ALU LUR LOW & AQ & OEY & PCU ,,PCUDZ A7 B7 &
        AM2904 SHIFTEN & DATAPATH & MEMCONT REQB MREQ ,WRITE &
        CONTROL ROM & RTB &
        BRAM YREG & CNTLB $11 & IMMD ONESECQ & CONT
03C7:  C0 4C 69 1F 7F FE F2 8F F0 0E 00 24

*         EXAMINE THE ONE SECOND QUEUE, 2 -> YREG
        SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM YREG &
        JSB EXQ1
03C8:  C8 20 29 1F 30 09 FA 87 F0 01 03 50

*         TEST MSR, IF = THEN SKIP CODE TO FETCH PROCESSES OUT
          OF QUEUE, R7 -> R6
TRAN1SEC:  ALU YBUS PASS & AB & PCU ,,PCUZA A7 B6 &
        AM2904 ,OECT & DATAPATH & MEMCONT & CONTROL &
        TEST MACHINE UAEQB & CJP TESTCP
03C9:  40 62 29 1F 4F 89 78 03 F2 53 03 CF

*         PERFORM A TEST ON THE PCU HERE.
*         THIS IS MAINLY BECAUSE A FAULTY PCU
*         COULD CAUSE THE SAME INSTRUCTION TO
*         BE REPEATEDLY EXECUTED. AN ERROR WOULD
*         ONLY BE DETECTED WHEN A TIMER INTERRUPT OCCURED.
*         THIS TEST PREVENTS THE MICROCODE FROM
*         STICKING HERE.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & XJSB PATEST
03CA:  48 62 29 1F 30 09 78 07 F0 01 XX XX

*         TRANSFER FROM 1 SEC. QUEUE
*         R6+2 -> R6 (FOOLS MPUTQ INTO THINKING THAT
*         ONE SECOND QUEUE IS A MONITOR).
        ALU YBUS PASS & AB & PCU ,SUB PCUAB A4 B6 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & JSB MPUTQ
03CB:  48 62 29 1F 99 89 78 07 F0 01 03 55

*         WREG - YREG (2) -> WREG (QUEUE SIZE COUNT)
*         SET MSR, LOOP BACK
        ALU REG SUBR & AB & OEY & PCUNOP &
        AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & ARAM YREG &
        BRAM WREG & TEST MACHINE DLOAD & JUMP TRAN1SEC
03CC:  C8 78 A9 1F 30 09 F9 AC 16 73 03 C9
```

```
*         ARRIVED HERE IF NEXT 1 SEC. INTERVAL HAS NOT OCCURED
*         WREG + 1000 -> DREG
NOT1SEC:  ALU YBUS ADD & DAB & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
       CONTROL ROM & RTB & BRAM WREG & IMMD 1000 & CONT
03CD: C2 61 A9 5F 30 28 F9 87 F0 0E 03 C8

*         WRITE REAL TIME RECORD
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE & CONTROL & CONT
03CE: 40 62 29 1F 30 3F 78 07 F0 0E 00 00

*         NOW TO SEE IF THE CURRENT PROCESS MAY WANT RESCHEDULI
          NG
*         REAL1I -> YBUS, SET MSR, IF <>0 THEN CALL UPDATE
TESTCP:   ALU YBUS PASSR & AQ & PCUNOP &
       AM2904 ,OECT EZ EC ES EOVR CEM & DATAPATH & MEMCONT &
       CONTROL ROM & ARAM REAL1I &
       TEST DIRIN UANEB & CJS UPDATE
03CF: 40 63 69 1F 30 09 F8 48 13 41 03 77

*         PHREG+PRIATR -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD PRIATR & CONT
03D0: 40 62 29 9E 56 28 78 07 F0 0E 00 0E

*         READ PRIORITY, -1 -> VREG
*         IF MSR = 0 THEN EXIT (NULL PROCESS RUNNING)
       ALU REG HIGH & AQ & OEY & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT REQB MREQ & CONTROL ROM & RTB &
       BRAM VREG & TEST MACHINE UAEQB & CJP RESCHED
03D1: C0 78 69 1F 30 3B FB 83 F2 53 03 8A

*         ZREG (PRIORITY) -1 -> VREG, IF < 0 THEN THIS IS PRIOR
          ITY 0 AND
*         CANNOT BE PRE-EMPTED,EXIT. SET MSR.
       ALU REG ADD & DAB & OEY & PCUNOP &
       AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL ROM & RTB &
       BRAM VREG & TEST DIRIN NEGATIVE & CJP ENDTIMER
03D2: C2 79 A9 0F 30 09 FB 80 13 F3 03 D9

*         IF MSR > 0 THEN PRIORITY = 2
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE POSITIVE &
       CJP PRIEQ2
03D3: 40 62 29 1F 30 09 78 03 F2 E3 03 DA

*         ARRIVED HERE IF PRIORITY = 1, PRIORITY 0 QUEUE ADDRES
          S -> R7
       ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD QSTART & CONT
03D4: 40 62 29 1F 7F C8 78 07 F0 0E 00 00

*         EXAMINE PRIORITY 0 QUEUE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB EXQ1
```

```
03D5: 48 62 29 1F 30 09 78 07 F0 01 03 50

*         PHREG+OVERTIME -> MAR
PRIEQ1:   ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD OVERTIME & CONT
03D6: 40 62 29 9E 56 28 78 07 F0 0E 00 10

*         READ OVERTIME ATTRIBUTE, PREPARE FOR RESCHEDULE ADDRE
          SS -> AM2910 COUNT
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & CONT
03D7: 40 62 29 1F 30 3B 78 07 F0 0E 00 00

*         ZREG (OVERTIME ATTRIBUTE) OR WREG (NO. OF QUEUE ENTRI
          ES ) -> YBUS
*         RESCHEDULE IF <> 0
        ALU YBUS OR & DAB & PCUNOP & AM2904 ,OECT &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL ROM & RTB &
        BRAM WREG & TEST DIRIN UANEB & CJP TESTNULL
03D8: C2 67 A9 0F 30 09 F9 83 F3 43 03 82

*         PROCESS DOES NOT NEED RESCHEDULING
*         PC - 4 -> MAR, JUMP TO START1
ENDTIMER:  ALU YBUS PASS & AB & PCU ,SUB PCUAB A5 B0 &
        AM2904 & DATAPATH ,,,,,,LDMAR & CONTROL &
        MEMCONT REQB & JUMPX START1.
03D9: 48 62 29 9F 9A 29 78 07 F0 03 XX XX

*         ARRIVED HERE IF PRIORITY = 2, PRIORITY 0 QUEUE ADDRES
          S -> R7
PRIEQ2:   ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH & CONTROL & MEMCONT & IMMD QSTART & CONT
03DA: 40 62 29 1F 7F C8 78 07 F0 0E 00 00

*         EXAMINE PRIORITY 0 QUEUE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB EXQ1
03DB: 48 62 29 1F 30 09 78 07 F0 01 03 50

*         WREG -> XREG, PRIORITY 1 QUEUE ADDRESS -> R7
        ALU REG PASSR & AQ & OEY & PCU ,,PCUDZ A7 B7 &
        AM2904 & CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM &
        RTB & ARAM WREG & BRAM XREG & IMMD QSTART+TOTQLENGTH &
        CONT
03DC: C0 7B 69 1F 7F C8 FA 1F F0 0E 00 0C

*         EXAMINE PRIORITY 1 QUEUE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB EXQ1
03DD: 48 62 29 1F 30 09 78 07 F0 01 03 50

*         WREG OR XREG -> WREG
*         JOIN PRIORITY = 1 TEST SEQUENCE
        ALU REG OR & AB & OEY & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL ROM & RTB & ARAM XREG & BRAM WREG &
        JUMP PRIEQ1
03DE: C8 7F A9 1F 30 09 F9 A7 F0 03 03 D6
```

```
*
* SYSTEM ERROR
*
SYSERR:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & XJUMP SSYMESS
03DF: 48 62 29 1F 30 09 78 07 F0 03 XX XX

0000 ERRORS (PMA-REV18.3)
```

```
*
* PRIME AMD MICRO-CODE MACRO ASSEMBLER
* VERSION 8, 2/10/82
        LIST
*
*
*
*       ******************************************************
*       *         AMD SUPER SIXTEEN BIT-SLICE PROCESSOR      *
*       *                FAULT-TOLERANT MICROCODE            *
*       ******************************************************
*
*
INTMEM     EQU 2              1 FOR INTELLIGENT MEMORY, 2 OTHERW
* MEMORY MAP.
P             EQU 3                    NUMBER OF PRIORITIES
NUMPROCESS    EQU 4                    MAXIMUM NUMBER OF PROCESSE
QSTART        EQU 0                    START OF QUEUE DATA STRUCT
QLENGTH       EQU 2*NUMPROCESS
TOTQLENGTH    EQU QLENGTH+4            TOTAL LENGTH OF QUEUE STRU
ONESECQ       EQU P*TOTQLENGTH         ADDRESS ONE SECOND QUEUE
SYSHEAP       EQU (P+1)*TOTQLENGTH     SYSTEM HEAP POINTER ADDRES
LASTHEAP      EQU SYSHEAP+2            NEXT AVAILABLE PROCESS DAT
NEXTINDEX     EQU LASTHEAP+2           NEXT INDEX COUNTER
RTREC         EQU NEXTINDEX+2+2*NUMPROCESS        REAL TIME REC
RTSECS        EQU RTREC+2              REAL TIME IN SECONDS
SPARE         EQU RTSECS+2
SPARELENGTH   EQU 10
GATELENGTH    EQU 2+TOTQLENGTH
* PROCESS ATTRIBUTE CODES
INDEXATR      EQU 0*2
HEAPATR       EQU 1*2
LINEATR       EQU 2*2
RESATR        EQU 3*2
RUNATR        EQU 4*2
SLICEATR      EQU 5*2
NESTATR       EQU 6*2
PRIATR        EQU 7*2
OVERTIME      EQU 8*2
JOBATR        EQU 9*2
CONTATR       EQU 10*2
OPCODE        EQU 11*2
ARG1          EQU 12*2
ARG2          EQU 13*2
ARG3          EQU 14*2
ARG4          EQU 15*2
OPLINE        EQU 16*2
CONSTATR      EQU 17*2
DEVICEATR     EQU 18*2
OPERATION     EQU 19*2
DATAATR       EQU 20*2
WAITING       EQU DATAATR+2
LASTTIME      EQU WAITING+2
PCREG     EQU LASTTIME+2
SPREG     EQU PCREG+2
REGG      EQU SPREG+2
REGB      EQU REGG+2
REGH      EQU REGB+2
```

```
PHLENGTH EQU REGH+10                        ALLOW SPARE
* DEVICE EQUATES
VDU          EQU 0
* MESSAGE EQUATES
TFMESS     EQU 1          TIMER FAULTY
IRMESS     EQU TFMESS+1   INTERRUPT RECEIVED
JMTSUCMESS EQU IRMESS+1   JOINT MEMORY TEST SUCCESSFULL
JMTUNMESS  EQU JMTSUCMESS+1   JOINT MEMORY TEST UNSUCCESSFULL
ALUMTSUCMESS    EQU JMTUNMESS+1       ALU MEMORY TEST SUCCESSFUL
ALUMTUNMESS     EQU ALUMTSUCMESS+1    ALU MEMORY TEST UNSUCCESSF
PCUMTSUCMESS    EQU ALUMTUNMESS+1     PCU MEMORY TEST SUCCESFULL
PCUMTUNMESS     EQU PCUMTSUCMESS+1    PCU MEMORY TEST UNSUCCESSF
PCUALUSUCMESS   EQU PCUMTUNMESS+1     PCU/ALU TEST SUCCESSFULL
PCUALUUNMESS    EQU PCUALUSUCMESS+1   PCU/ALU TEST UNSUCCESSFULL
ALUSUCMESS      EQU PCUALUUNMESS+1    ALU MEMORY TEST SUCCESSFUL
ALUUNMESS       EQU ALUSUCMESS+1      ALU TEST UNSUCCESSFULL
BMTSUCMESS      EQU ALUUNMESS+1       BYTE MEMORY TEST SUCCESSFU
BMTUNMESS       EQU BMTSUCMESS+1      BYTE MEMORY TEST UNSUCCESS
MAPZIMESS       EQU BMTUNMESS+1       MAP/ZI TEST SUCCESSFULL
INVMESS         EQU MAPZIMESS+1       INVALID INSTRUCTION MESSAG
SYSERRMESS      EQU INVMESS+1         SYSTEM ERROR MESSAGE
INTMESS         EQU SYSERRMESS+1      PERIODIC TIMER INTERRUPT M
IEMESS          EQU INTMESS+1         INTERRUPT ERROR MESSAGE
MFMESS          EQU IEMESS+1          MAPPING PROM FAULTY MESSAG
STARTMESS       EQU 100                SYSTEM INITIALISATION MES
* OTHER EQUATES
A$INCREG     EQU 2*INTMEM          = A2 (1) OR A4 (2)
PROGSTART    EQU $C000*INTMEM
CNTVAL       EQU $5555                      TIMER INTERRUPT PE
ACIACS       EQU $2034*INTMEM     ACIA CONTROL REGISTER ADDRESS
ACIADA       EQU ACIACS+INTMEM          ACIA DATA REGISTER ADDRE
CORRECTREPLY EQU $AA              PROCESSOR REPLY CODE
SENDPIADR    EQU $2030*INTMEM     A SIDE OF PIA DATA REGISTER -
SENDPIACR    EQU SENDPIADR+INTMEM      A SIDE OF PIA CONTROL RE
RECPIADR     EQU SENDPIACR+INTMEM      B SIDE OF PIA DATA REG.
RECPIACR     EQU RECPIADR+INTMEM       B SIDE OF PIA CONTROL RE
CLOCKSPEED   EQU 1
INTERVAL     EQU 1000
CLOCKINCR    EQU 2000            CLOCK INCREMENT VALUE
MAXSLICE     EQU CLOCKSPEED*INTERVAL
MEMMAX       EQU 7*1024                          TOP RAM ADDRE
DLY1         EQU 1500                      MAX VALUE = 4095 ($0
         EXT IOPRES1
         EXT START1
         EXT RNI
         ENT INITPIA
         ENT NOTVDU
         ENT TIMERTEST
         ENT JMEMTEST
         ENT JSENDSUB
         ENT MAPZITEST
         ENT SINVMESS
         ENT SSYMESS
         ENT STIMESS
         ENT MAPFAULTY
         ENT PATEST
         ADDR $3E0
*
```

```
*
* SEND A MESSAGE TO OTHER PROCESSOR
* CALLED FROM HIGH LEVEL "IO" COMMAND
* PIA :    A-PORT = SENDER, B-PORT = RECEIVER
* 1) SEND A MESAGE AND THEN 2) WAIT FOR REPLY
* WAITING ATTRIBUTE IS SET TO 1 IF PART 1) HAS BEEN
* COMPLETED, THIS ALLOWS RE-SCHEDULING TO TAKE PLACE IF NO REP
        LY
* IS RECEIVED.
* ON ENTRY  WREG = DEVICE TYPE
*           ZREG = OPERATION ADDRESS
*            MAR = R6 = DATA ADDRESS
*
*         WREG (DEVICE TYPE) - 1 -> WREG, LOAD MSR
*         R6+2 -> MAR ("WAITING" ATTRIBUTE ADDRESS)
NOTVDU:   ALU REG SUBR & DAB & OEY & PCU ,,PCUAB A4 B6 &
       AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
       RTB & BRAM WREG & TEST MACHINE DLOAD & IMMD 1 & CONT
03E0: C2 78 A9 9F 19 A8 F9 84 16 7E 00 01

*         READ WAITING ATTRIBUTE, JUMP OUT IF MSR <> 0
*         ZREG (OPERATION ADDRESS) -> R7,MAR
       ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 &
       AM2904 ,OECT & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & TEST MACHINE UANEB &
       CJP SSYMESS
03E1: 40 62 29 8F 7F FB 78 03 F2 43 04 08

*         ZREG (WAITING ATTRIBUTE) -> YBUS, LOAD MSR, READ OPER
          ATION
       ALU YBUS PASSR & DAQ & PCUNOP &
       AM2904 ,,EZ EC ES EOVR CEM & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ & CONTROL & TEST MACHINE DLOAD &
       CONT
03E2: 42 63 69 0F 30 3B 78 04 12 7E 00 00

*         ZREG (OPERATION) -> YBUS, JUMP TO RECEIVE CODE IF <>
          0
*         R6-2 -> MAR
       ALU YBUS PASSR & DAQ & PCU ,SUB PCUAB A4 B6 &
       AM2904 ,OECT & DATAPATH ,,,,,,LDMAR ,,ENZ0 &
       MEMCONT REQB & CONTROL & TEST DIRIN UANEB &
       CJP RECEIVE
03E3: 42 63 69 8F 99 A9 78 03 F3 43 03 F0

*         IF MSR <> 0 THEN GOTO WAITREPLY, READ DATA ADDRESS
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT REQB MREQ & CONTROL &
       TEST MACHINE UANEB & CJP WAITREPLY
03E4: 40 62 29 1F 30 3B 78 03 F2 43 03 E8

*         ZREG (DATA ADDRESS) -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,,LDMAR ,,ENZ0 & MEMCONT REQB & CONTROL &
       CONT
03E5: 40 62 29 8E 70 29 78 07 F0 0E 00 00
```

```
*         READ DATA, R6+2 -> MAR (WAITING ATTRIBUTE ADDRESS)
*         1-> DREG
        ALU LUR LOW & AQ & OEY & PCU QEU ,PCUAB A4 B6 &
        AM2904 SHIFTEN & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ & CONTROL ROM & RTB &
        BRAM YREG & CNTLB $11 & CONT
03E6: C0 4C 69 DE 19 BB F2 8F F0 0E 00 00

*         WRITE 1 TO WAITING ATTRIBUTE, ZREG (DATA) -> WREG,DRE
          G
*         JUMP TO "SEND" SUBROUTINE.
        ALU REG PASSR & DAQ & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH ,,,,,,,LDD ,ENZ0 & MEMCONT REQB MREQ ,WRITE &
        CONTROL ROM & RTB & BRAM WREG & JSB JSENDSUB
03E7: CA 7B 69 4F 30 3F F9 87 F0 01 04 8D

*
* FALL INTO WAITREPLY
*
*         JUMP TO RECEIVE SUBROUTINE
WAITREPLY:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JRECSUB
03E8: 48 62 29 1F 30 09 78 07 F0 01 04 91

*         IF MSR = 0 THEN NO MESSAGE RECEIVED YET, RESCHEDULE
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
        CJPX IOPRES1
03E9: 40 62 29 1F 30 09 78 03 F2 53 XX XX

*         WREG - CORRECT REPLY -> YBUS, LOAD MSR, PC -> MAR
        ALU YBUS SUBR & DAB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM WREG &
        TEST DIRIN UANEB & IMMD CORRECTREPLY & CONT
03EA: C2 60 A9 9F 30 28 F9 80 17 4E 00 AA

*         R7+2 -> MAR ("FAULTY" ADDRESS), 0 -> DREG, IF MSR = 0
           THEN SKIP NEXT IN
*         READ INSTR N+2
        ALU LUR LOW & AQ & OEY & PCU ,,PCUAB A4 B7 &
        AM2904 SHIFTEN OECT & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ & CONTROL ROM & CNTLB $10 & RTB &
        BRAM YREG & TEST MACHINE UAEQB & CJP WVF1
03EB: C0 4C 69 DF 19 FB F2 83 F2 53 03 EE

*         ARRIVED HERE IF REPLY IS BAD.
*         SEND A REPLY TO FREE THE FAULTY PROCESSOR.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSENDSUB
03EC: 48 62 29 1F 30 09 78 07 F0 01 04 8D

*         1 -> DREG, R7 -> MAR ("FAULTY ADDRESS")
        ALU YBUS PASSR & AQ & OEY & PCU ,,PCUZA A7 B7 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & ARAM YREG & CONT
03ED: 40 63 69 DF 4F E9 F8 2F F4 0E 00 00
```

```
*         WRITE VALUE TO FAULTY, R6+2 -> MAR (WAITING ADDRESS),
           0 -> DREG
WVF1:   ALU YBUS LOW & AQ & OEY & PCU ,,PCUAB A4 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB MREQ ,WRITE &
        CONTROL & CONT
03EE: 40 64 69 DF 19 BF 78 07 F0 0E 00 00

*         WRITE ZERO TO WAITING, PC+2 -> MAR, JUMP MAP
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JMAP
03EF: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
* RECEIVE A MESSAGE FROM OTHER PROCESSOR
* 1) WAIT FOR MESSAGE
* 2) SEND A REPLY
* WAITING ATTRIBUTE IS SET TO 1 IF PART 1) HAS
* ALREADY BEEN DONE, THIS ALLOWS RE-SCHEDULING
* TO TAKE PLACE IF NO REPLY IS RECEIVED
*
*         IF MSR <> 0 THEN GOTO SEND REPLY, READ DATA ADDRESS
RECEIVE:  ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT REQB MREQ & CONTROL &
        TEST MACHINE UANEB & CJP SENDREPLY
03F0: 40 62 29 1F 30 3B 78 03 F2 43 03 F6

*         ARRIVED HERE IF WAITING FOR MESSAGE, JUMP TO RECEIVE
          SUBROUTINE
*         ZREG (DATA ADDRESS) -> R6
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        JSB JRECSUB
03F1: 48 62 29 0F 7D 89 78 07 F0 01 04 91

*         IF MSR = 0 THEN NO MESSAGE READY YET, RESCHEDULE
*         R6 (DATA ADDRESS) -> MAR, WREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 ,OECT & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB & CONTROL ROM & ARAM WREG &
        TEST MACHINE UAEQB & CJPX IOPRES1
03F2: 40 63 69 DF 4D A9 F8 1B F2 53 XX XX

*         WRITE MESSAGE RECEIVED TO DATA ADDRESS, 0 -> DREG
*         R7+2 -> MAR (FAULTY ADDRESS)
        ALU REG LOW & AQ & OEY & PCU ,,PCUAB A4 B7 & AM2904 &
        DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB MREQ ,WRITE &
        CONTROL ROM & RTB & BRAM YREG & CONT
03F3: C0 7C 69 DF 19 FF FA 87 F0 0E 00 00

*         WRITE 0 TO FAULTY, 1 -> DREG,
*         WAITING ATTRIBUTE ADDRESS -> MAR
        ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & ARAM YREG &
        IMMD WAITING & CONT
03F4: 40 63 69 DE 56 3E F8 2F F4 0E 00 2A
```

```
*         WRITE 1 TO WAITING
W1W:   ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE & CONT
03F5: 40 62 29 1F 30 3F 08 07 F0 0E 00 00

*
* FALL INTO SEND REPLY
*
*         CORRECT REPLY -> WREG,DREG , PC -> MAR
SENDREPLY:   ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM WREG & IMMD CORRECTREPLY &
        CONT
03F6: C2 7B 69 DF 30 28 F9 87 F0 0E 00 AA

*         JUMP TO SEND SUBROUTINE, READ INSTR N+2
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ & CONTROL & JSB JSENDSUB
03F7: 48 62 29 1F 30 3B 78 07 F0 01 04 8D

*         0 -> DREG, WAITING ATTRIBUTE ADDRESS -> MAR
        ALU YBUS LOW & AQ & OEY & PCU QEU ,PCUDA PHREG &
        AM2904 & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
        CONTROL & IMMD WAITING & CONT
03F8: 40 64 69 DE 56 28 78 07 F0 0E 00 2A

*         WRITE 0 TO WAITING, PC+2 -> MAR, JUMP MAP
        ALU YBUS PASS & AB & PCUNEXT & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ ,WRITE &
        CONTROL & JMAP
03F9: 48 62 29 9F 18 3F 78 07 F0 02 00 00

*
*
* CODE TO TEST THE TIMER INTERRUPT HARDWARE
*
*
*         IF NO INTERRUPT IS PRESENT AFTER THE MAXIMUM
*         NO. OF CYCLES THEN THE HARDWARE IS FAULTY.
*         AFTER AN ERROR IS DETECTED, A MESSAGE IS SENT TO THE
          NEIGHBOUR PROCESSO
*         IF AN INTERRUPT IS ENCOUNTERED AFTER THIS A MESSAGE
*         IS SENT TO THE NEIGHBOUR PROCESSOR. THIS ENABLES
*         THE USER TO TEST THE PUSH BUTTON FACILITY.
*
*
*         -TIMER INTERRUPT PERIOD/2 -> WREG
TIMERTEST:   ALU REG PASSR & DAQ & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM WREG & IMMD -(INTERVAL/2) & CONT
03FA: C2 7B 69 1F 30 08 F9 87 F0 0E FE 0C

*         INCREMENT WREG, JUMP TO TIMER FAULTY CODE IF = 0
TFLOOP:   ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
        RTB & BRAM WREG & ARAM WREG & TEST DIRIN UAEQB &
        CJP TIMERFAULTY
```

```
03FB: C0 7B 69 1F 30 09 F9 9B F7 53 03 FE

*         IF NO INTERRUPT THEN LOOP BACK
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & TESTF INTERRUPT & CJP TFLOOP
03FC: 40 62 29 1F 30 09 78 07 F0 73 03 FB

*         RETURN, INTERRUPT HAS OCCURED
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & RTN
03FD: 48 62 29 1F 30 09 78 07 F0 0A 00 00

*         ARRIVED HERE IF TIMER IS FAULTY
*         TIMER FAULTY MESSAGE -> WREG,DREG
TIMERFAULTY:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD TFMESS-1 & CONT
03FE: C2 7B 69 5F 30 08 F9 87 F4 0E 00 00

*         SEND A MESSAGE
STF:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
03FF: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*         IF NO INTERRUPT THEN LOOP
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & TESTF INTERRUPT & CJP *
0400: 40 62 29 1F 30 09 78 07 F0 73 04 00

*         CLEAR THE INTERRUPT
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & INTLOAD -1 & CONT
0401: 00 62 29 1F 30 09 78 07 F0 0E FF FF

*         INTERRUPT RECEIVED MESSAGE -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD TFMESS-1 & CONT
0402: C2 7B 69 5F 30 08 F9 87 F4 0E 00 00

*         LOOP BACK
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP JMEMTEST
0403: 48 62 29 1F 30 09 78 07 F0 03 04 17

*
* ARRIVED HERE IF INVALID OP-CODE HAS BEEN ENCOUNTERED
* SEND APPROPRIATE MESSAGE AND THEN JOIN MAIN TEST SEQUENCE
*
*         KICK THE PIA
SINVMESS:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JKICK
0404: 48 62 29 1F 30 09 78 07 F0 01 04 95

*         INVALID MESSAGE CODE -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD INVMESS-1 &
```

```
       CONT
0405: C2 7B 69 5F 30 08 F9 87 F4 0E 00 0F

*        SEND THE MESSAGE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSFSENDMES
0406: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*        JOIN MAIN TEST SEQUENCE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP JMEMTEST
0407: 48 62 29 1F 30 09 78 07 F0 03 04 17

*
* ARRIVED HERE IF SYSTEM ERROR HAS OCCURED.
* SEND APPROPRIATE MESSAGE AND THEN JOIN MAIN TEST SEQUENCE.
*
*        KICK THE PIA
SSYMESS:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JKICK
0408: 48 62 29 1F 30 09 78 07 F0 01 04 95

*        SYSERR MESSAGE CODE -> WREG,DREG
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL ROM & RTB & BRAM WREG & IMMD SYSERRMESS-1 &
       CONT
0409: C2 7B 69 5F 30 08 F9 87 F4 0E 00 10

*        SEND THE MESSAGE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSFSENDMES
040A: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*        JOIN THE MAIN TEST SEQUENCE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP JMEMTEST
040B: 48 62 29 1F 30 09 78 07 F0 03 04 17

*
* ARRIVED HERE IF MAPPING PROM FAULTY
*
*        KICK THE PIA
MAPFAULTY:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JKICK
040C: 48 62 29 1F 30 09 78 07 F0 01 04 95

*        MAP FAULTY MESSAGE CODE -> WREG,DREG
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL ROM & RTB & BRAM WREG & IMMD MFMESS-1 & CONT
040D: C2 7B 69 5F 30 08 F9 87 F4 0E 00 13

*        SEND THE MESSAGE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSFSENDMES
040E: 48 62 29 1F 30 09 78 07 F0 01 04 C4
```

```
*         SEND THE MESSAGE AGAIN IN CASE IT HAS BEEN
*         TREATED AS AREPLY TO A PREVIOUSLY SENT MESSAGE.
*         MAP FAULTY MESSAGE CODE -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD MFMESS-1 & CONT
040F: C2 7B 69 5F 30 08 F9 87 F4 0E 00 13

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
0410: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*         JOIN THE MAIN TEST SEQUENCE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP JMEMTEST
0411: 48 62 29 1F 30 09 78 07 F0 03 04 17

*
* ARRIVED HERE IF ERROR DETECTED ON ENTRY TO INTERRUPT CODE
*
*         KICK THE PIA
STIMESS:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JKICK
0412: 48 62 29 1F 30 09 78 07 F0 01 04 95

*         FAULTY INTERRUPT MESSAGE CODE -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD IEMESS-1 & CONT
0413: C2 7B 69 5F 30 08 F9 87 F4 0E 00 12

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
0414: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*         SEND THE MESSAGE AGAIN IN CASE IT HAS BEEN
*         TREATED AS A REPLY TO A PREVIOUSLY SENT MESSAGE.
*         FAULTY INTERRUPT MESSAGE CODE -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD IEMESS-1 & CONT
0415: C2 7B 69 5F 30 08 F9 87 F4 0E 00 12

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
0416: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*
* FALL INTO JOINT MEMORY TEST CODE
*
*
*
* MEMORY TEST CODE.
*
*
```

```
*         MEMORY IS TESTED BY REPEATEDLY CALLING A SUBROUTINE
*         WHICH TESTS 1K OF MEMORY. AS SOON AS A FAULTY BLOCK
*         OF MEMORY IS DISCOVERED A MESSAGE IS SENT FOLLOWED
*         BY THE BLOCK OF MEMORY WHICH IS FAULTY. IF MEMORY IS
*         O.K. THEN A MESSAGE IS ALSO SENT.
*
*
*         TOP RAM ADDRESS -> XREG
JMEMTEST:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       DATAPATH & CARRYCTL & MEMCONT & CONTROL ROM & RTB &
       BRAM XREG & IMMD MEMMAX & CONT
0417: C2 7B 69 1F 30 08 FA 07 F0 0E 1C 00

*         0 -> ,YREG, JUMP TO MEMORY TEST SUBROUTINE
       ALU REG LOW & AQ & OEY & PCU ,SUB PCUAB A6 B6 &
       AM2904 & DATAPATH & MEMCONT & CONTROL ROM & RTB &
       BRAM YREG & JSB JMTSUB
0418: C8 7C 69 1F 9D 89 FA 87 F0 01 04 E4

*         R6+1K -> R6, YREG+1K -> YREG (R6,YREG IDENTICAL)
JMTLOOP:  ALU REG ADD & DAB & OEY & PCU ,,PCUDA A6 B6 &
       AM2904 & CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & BRAM YREG & IMMD 1024 & CONT
0419: C2 79 A9 1F 5D 88 FA 87 F0 0E 04 00

*         IF MSR <> 0 THEN MEMORY IS FAULTY
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
       CJP *+12
041A: 40 62 29 1F 30 09 78 03 F2 53 04 1C

*         ARRIVED HERE IF MEMORY IS FAILTY
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP JMEMFAULTY
041B: 48 62 29 1F 30 09 78 07 F0 03 04 23

*         XREG - YREG -> YBUS, IF = 0 THEN TEST IS OVER
       ALU YBUS SUBS & AB & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & ARAM XREG & BRAM YREG & TEST DIRIN UANEB &
       CJP JMTESTEND
041C: C0 61 29 1F 30 09 FA A3 F7 43 04 20

*         TEST IS OVER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP JMTESTEND
041D: 48 62 29 1F 30 09 78 07 F0 03 04 20

*         JUMP TO MEMORY TEST SUBROUTINE.
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JMTSUB
041E: 48 62 29 1F 30 09 78 07 F0 01 04 E4

*         LOOP BACK.
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP JMTLOOP
041F: 48 62 29 1F 30 09 78 07 F0 03 04 19
```

```
*         ARRIVED HERE IF ALL BLOCKS OF RAM HAVE BEEN SUCCESSFU
          LLY TESTED.
*         TEST SUCCESSFULL MESSAGE CODE -> WREG,DREG
JMTESTEND:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD JMTSUCMESS-1 &
        CONT
0420: C2 7B 69 5F 30 08 F9 87 F4 0E 00 02

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
0421: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*         JUMP TO PCU/ALU TEST CODE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP ALUMEMTEST
0422: 48 62 29 1F 30 09 78 07 F0 03 04 25

*         ARRIVED HERE IF MEMORY FAULTY
*         MEMORY TEST UNSUUCCESSFULL MESSAGE -> WREG,DREG
JMEMFAULTY:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD JMTUNMESS-1 &
        CONT
0423: C2 7B 69 5F 30 08 F9 87 F4 0E 00 03

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
0424: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*
* FALL INTO ALU MEMORY TEST CODE
*
*
* CODE TO TEST MEMORY USING THE ALU ONLY.
* MEMORY IS TESTED 1K AT A TIME BY CALLING A SUBROUTINE
*
* 8K -> XREG
ALUMEMTEST:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM XREG & IMMD MEMMAX & CONT
0425: C2 7B 69 1F 30 08 FA 07 F0 0E 1C 00

*         0 -> YREG, JUMP TO MEMORY TEST SUBROUTINE
        ALU REG LOW & AQ & OEY & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL ROM & RTB & BRAM YREG &
        JSB ALUMTSUB
0426: C8 7C 69 1F 30 09 FA 87 F0 01 04 F3

*         YREG+1K -> YREG
ALUMTLOOP:  ALU REG ADD & DAB & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM YREG & IMMD 1024 & CONT
0427: C2 79 A9 1F 30 08 FA 87 F0 0E 04 00

*         IF MSR <> 0 THEN MEMORY IS FAULTY
```

```
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
        CJP *+12
0428: 40 62 29 1F 30 09 78 03 F2 53 04 2A

*         ARRIVED HERE IF MEMORY IS FAULTY
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP ALUMEMFAULTY
0429: 48 62 29 1F 30 09 78 07 F0 03 04 33

*         XREG - YREG -> YBUS, IF = 0 THEN TEST OVER
        ALU YBUS SUBS & AB & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
        RTB & ARAM XREG & BRAM YREG & TEST DIRIN UANEB &
        CJP *+12
042A: C0 61 29 1F 30 09 FA A3 F7 43 04 2C

*         ARRIVED HERE IF TEST IS OVER
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP ALUMTESTEND
042B: 48 62 29 1F 30 09 78 07 F0 03 04 2E

*         JUMP TO MEMORY TEST SUBROUTINE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB ALUMTSUB
042C: 48 62 29 1F 30 09 78 07 F0 01 04 F3

*         LOOP BACK
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP ALUMTLOOP
042D: 48 62 29 1F 30 09 78 07 F0 03 04 27

*         ARRIVED HERE IF TEST IS PASSED.
*         TEST PASSED MESSAGE CODE -> DREG
ALUMTESTEND:  ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL & IMMD ALUMTSUCMESS-1 & CONT
042E: 42 63 69 5F 30 08 78 07 F4 0E 00 04

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB ALUSFSENDMES
042F: 48 62 29 1F 30 09 78 07 F0 01 04 CC

*         SEND THE MESSAGE AGAIN IN CASE IT HAS BEEN TREATED AS
           A REPLY.
*         TEST PASSED MESSAGE CODE -> DREG
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL & IMMD ALUMTSUCMESS-1 & CONT
0430: 42 63 69 5F 30 08 78 07 F4 0E 00 04

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB ALUSFSENDMES
0431: 48 62 29 1F 30 09 78 07 F0 01 04 CC

*         JUMP TO PCU MEMORY TEST CODE
```

```
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP PCUMEMTEST
0432: 48 62 29 1F 30 09 78 07 F0 03 04 35

*        ARRIVED HERE IF MEMORY IS FAULTY, TEST FAILED MESSAGE
          -> DREG
ALUMEMFAULTY:  ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL & IMMD ALUMTUNMESS-1 & CONT
0433: 42 63 69 5F 30 08 78 07 F4 0E 00 05

*        SEND THE MESSAGE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB ALUSFSENDMES
0434: 48 62 29 1F 30 09 78 07 F0 01 04 CC

*
* FALL INTO PCU MEMORY TEST CODE
*
* CODE TO TEST MEMORY USING PCU ONLY
* TESTS 1K BLOCK AT A TIME BY CALLING A SUBROUTINE.
* STOPS WHEN ALL RAM OR FAILURE DETECTED
*
*        8K -> R1
PCUMEMTEST:  ALU YBUS PASS & AB & PCU ,,PCUDZ A1 B1 &
       AM2904 & DATAPATH & MEMCONT & CONTROL & IMMD MEMMAX &
       CONT
0435: 40 62 29 1F 72 48 78 07 F0 0E 1C 00

*        IF R1 = 0 THEN THERE IS AN ERROR
*        THIS IS TO ENSURE THAT THE TEST TREE
*        IS WORKING.
       ALU YBUS PASS & AB & PCU ,,PCUZA A1 B1 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & TESTF PCUNZ &
       CJP PCUMT1
0436: 40 62 29 1F 42 49 78 07 F0 63 04 38

*        ARRIVED HERE IF THERE IS AN ERROR
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP PCUMEMFAULTY
0437: 48 62 29 1F 30 09 78 07 F0 03 04 43

*        0 -> R6, JUMP TO MEMORY TEST SUBROUTINE
PCUMT1:  ALU YBUS PASS & AB &
       PCU ,SUB PCUAB A6 B6 & AM2904 & DATAPATH & MEMCONT &
       CONTROL & JSB PCUMTSUB
0438: 48 62 29 1F 9D 89 78 07 F0 01 05 06

*        R6+1K -> R6
PCUMTLOOP:  ALU YBUS PASS & AB & PCU ,,PCUDA A6 B6 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD 1024 & CONT
0439: 40 62 29 1F 5D 88 78 07 F0 0E 04 00

*        IF R0 <> 0 THEN MEMORY IS FAULTY
       ALU YBUS PASS & AB & PCU ,,PCUZA A0 B0 & AM2904 &
       DATAPATH & CONTROL & MEMCONT & TESTF PCUNZ &
       CJP PCUMEMFAULTY
043A: 40 62 29 1F 40 09 78 07 F0 63 04 43
```

```
*         R1 - R6 -> Q, IF = 0 THEN TEST IS OVER, OTHERWISE JUM
          P FORWARD.
          ALU YBUS PASS & AB & PCU QEU SUB PCUAB A6 B1 &
          AM2904 & DATAPATH & MEMCONT & CONTROL & TESTF PCUNZ &
          CJP JMT
043B:  40 62 29 1E 9C 49 78 07 F0 63 04 41

*         TEST OVER, TEST SUCCESSFULL MESSAGE -> R6
          ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
          DATAPATH & MEMCONT & CONTROL & IMMD PCUMTSUCMESS &
          CONT
043C:  40 62 29 1F 7D 88 78 07 F0 0E 00 07

*         SEND THE MESSAGE
          ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
          MEMCONT & CONTROL & JSB PCUSENDMES
043D:  48 62 29 1F 30 09 78 07 F0 01 04 C0

*         SEND THE MESSAGE AGAIN IN CASE IT HAS BEEN TREATED AS
           A REPLY.
*          TEST SUCCESSFULL MESSAGE -> R6
          ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
          DATAPATH & MEMCONT & CONTROL & IMMD PCUMTSUCMESS &
          CONT
043E:  40 62 29 1F 7D 88 78 07 F0 0E 00 07

*         SEND THE MESSAGE
          ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
          MEMCONT & CONTROL & JSB PCUNCSEND
043F:  48 62 29 1F 30 09 78 07 F0 01 04 C2

*         JUMP FORWARD
          ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
          MEMCONT & CONTROL & JUMP REPLACEREG
0440:  48 62 29 1F 30 09 78 07 F0 03 04 45

*         JUMP TO MEMORY TEST SUBROUTINE
JMT:   ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
          MEMCONT & CONTROL & JSB PCUMTSUB
0441:  48 62 29 1F 30 09 78 07 F0 01 05 06

*         LOOP BACK
          ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
          MEMCONT & CONTROL & JUMP PCUMTLOOP
0442:  48 62 29 1F 30 09 78 07 F0 03 04 39

*         ARRIVED HERE IF MEMORY IS FAULTY
*         TEST FAILED MESSAGE CODE -> R6
PCUMEMFAULTY:  ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 &
          AM2904 & DATAPATH & MEMCONT & CONTROL &
          IMMD PCUMTUNMESS & CONT
0443:  40 62 29 1F 7D 88 78 07 F0 0E 00 08

*         SEND THE MESSAGE
          ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
          MEMCONT & CONTROL & JSB PCUSENDMES
0444:  48 62 29 1F 30 09 78 07 F0 01 04 C0
```

```
*         REPLACE THE PCU REGISTERS WHICH HAVE BEEN CORRUPTED
*         R2 = 1, R5 = 4, R4 = 2
*         1-> R2
REPLACEREG:  ALU YBUS PASS & AB & PCU ,,PCUDZ A2 B2 &
       AM2904 & DATAPATH & MEMCONT & CONTROL & IMMD 1 &
       CONT
0445: 40 62 29 1F 74 88 78 07 F0 0E 00 01

*         4 -> R5
       ALU YBUS PASS & AB & PCU ,,PCUDZ A5 B5 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD 4 & CONT
0446: 40 62 29 1F 7B 48 78 07 F0 0E 00 04

*         2 -> R4
       ALU YBUS PASS & AB & PCU ,,PCUDZ A4 B4 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD 2 & CONT
0447: 40 62 29 1F 79 08 78 07 F0 0E 00 02

*
* FALL INTO PCU/ALU TEST
*
* PASS VALUES FROM ALU TO PCU AND BACK AGAIN.
* SEE IF THE VALUES ARE CORRUPTED AFTER TRANSFER USING THE AM2
       904.
* VALUES PASSED ARE :- $0000,$5555,$AAAA,$FFFF
*         0 -> WREG,TREG
PCUALU:  ALU REG LOW & AQ & OEY & PCUNOP & AM2904 &
       DATAPATH ,,LDTREG & MEMCONT & CONTROL ROM & RTB &
       BRAM WREG & LDCTA PCUALUERR
0448: C0 7C 79 1F 30 09 F9 87 F0 0C 04 5A

*         TREG -> R6 OF PCU -> XREG
       ALU REG PASS & AB & YOFF & PCU ,,PCUDZ A6 B6 &
       AM2904 & DATAPATH ,ENTREG ,,,PCUY & MEMCONT &
       CONTROL ROM & RTB & BRAM XREG & CONT
0449: C1 7A 08 1F 7D 89 FA 07 F0 0E 00 00

*         WREG - XREG -> YBUS, ERROR IF <> 0
       ALU YBUS SUBS & AB & PCUNOP &
       AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & ARAM WREG &
       BRAM XREG & TEST DIRIN UAEQB & JRP *+6
044A: C0 61 29 1F 30 09 FA 18 17 57 04 4B

*         IF MSR IS -VE THEN THERE IS AN ERROR
*         THIS TEST DETECTS A FAULTY ALU M.S. CHIP.
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE POSITIVE &
       JRP ALUPCUOK1
044B: 40 62 29 1F 30 09 78 03 F2 E7 04 4C

*         $5555 -> YREG,TREG
ALUPCUOK1:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,LDTREG & MEMCONT & CONTROL ROM &
       RTB & BRAM YREG & IMMD $5555 & CONT
044C: C2 7B 79 1F 30 08 FA 87 F0 0E 55 55
```

```
*         TREG -> R7 OF PCU -> VREG
        ALU REG PASS & AB & YOFF & PCU ,,PCUDZ A7 B7 &
        AM2904 & DATAPATH ,ENTREG ,,,PCUY & MEMCONT &
        CONTROL ROM & RTB & BRAM VREG & CONT
044D: C1 7A 08 1F 7F C9 FB 87 F0 0E 00 00

*         VREG - YREG -> YBUS, ERROR IF <> 0
        ALU YBUS SUBR & AB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & ARAM VREG &
        BRAM YREG & TEST DIRIN UAEQB & JRP *+6
044E: C0 60 A9 1F 30 09 FA B8 17 57 04 4F

*         IF MSR IS -VE THEN THERE IS AN ERROR
*         THIS TEST DETECTS A FAULTY ALU M.S. CHIP.
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE POSITIVE &
        JRP ALUPCUOK2
044F: 40 62 29 1F 30 09 78 03 F2 E7 04 50

*         $AAAA -> REAL1M,TREG
ALUPCUOK2:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,LDTREG & MEMCONT & CONTROL ROM &
        RTB & BRAM REAL1M & IMMD $AAAA & CONT
0450: C2 7B 79 1F 30 08 FC 07 F0 0E AA AA

*         TREG -> Q OF PCU -> REAL2M
        ALU REG PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,ENTREG ,,,PCUY & MEMCONT & CONTROL ROM &
        RTB & BRAM REAL2M & CONT
0451: C1 7A 08 1E 70 09 FE 07 F0 0E 00 00

*         REAL1M - REAL2M -> YBUS, ERROR IF <> 0
        ALU YBUS SUBS & AB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & DATAPATH &
        CARRYCTL COEQ1 & MEMCONT & CONTROL ROM & RTB &
        ARAM REAL1M & BRAM REAL2M & TEST DIRIN UAEQB &
        JRP *+6
0452: C0 61 29 1F 30 09 FE 40 17 57 04 53

*         IF MSR IS -VE THEN THERE IS AN ERROR
*         THIS TEST DETECTS A FAULTY ALU M.S. CHIP.
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE POSITIVE &
        JRP ALUPCUOK3
0453: 40 62 29 1F 30 09 78 03 F2 E7 04 54

*         $FFFF -> EXP1 -> TREG
ALUPCUOK3:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,LDTREG & CONTROL ROM & RTB &
        BRAM EXP1 & IMMD $FFFF & CONT
0454: C2 7B 79 1F 30 00 FD 87 F0 0E FF FF

*         TREG -> R0 OF PCU, EXP2
        ALU REG PASS & AB & YOFF & PCU ,,PCUDZ A0 B0 &
        AM2904 & DATAPATH ,ENTREG ,,,PCUY & MEMCONT &
        CONTROL ROM & RTB & BRAM EXP2 & CONT
0455: C1 7A 08 1F 70 09 FF 87 F0 0E 00 00
```

```
*         EXP2 - EXP1 -> YBUS, ERROR IF <> 0
        ALU YBUS SUBS & AB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & ARAM EXP2 &
        BRAM EXP1 & TEST DIRIN UAEQB & JRP *+6
0456: C0 61 29 1F 30 09 FD F8 17 57 04 57

*         IF MSR IS -VE THEN THERE IS AN ERROR
*         THIS TEST DETECTS A FAULTY ALU M.S. CHIP.
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE POSITIVE &
        JRP ALUPCUOK4
0457: 40 62 29 1F 30 09 78 03 F2 E7 04 58

*         ARRIVED HERE IF TEST PASSED, SUCCESS MESSAGE -> WREG,
          DREG
ALUPCUOK4:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD PCUALUSUCMESS-1 &
        CONT
0458: C2 7B 69 5F 30 08 F9 87 F4 0E 00 08

*         SKIP NEXT INSTRUCTION
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP *+12
0459: 48 62 29 1F 30 09 78 07 F0 03 04 5B

*         ARRIVED HERE IF TEST HAS FAILED
*         FAILURE MESSAGE CODE -> WREG,DREG
PCUALUERR:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD PCUALUUNMESS-1 &
        CONT
045A: C2 7B 69 5F 30 08 F9 87 F4 0E 00 09

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
045B: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*
* FALL INTO ALU TEST
*
*
* CODE TO TEST THE ALU BY PASSING A VALUE THROUGH IT TO THE DR
          EG,
* SENDING IT TO THE NEIGHBOUR PROCESSOR,
* SHIFTING IT DOWN 8 BITS AND THEN SENDING THE M.S.
* BYTE TO THE NEIGHBOUR PROCESSOR.
*
*         RE-INITIALISE THE PIA'S
*         THIS IS IN CASE A FAULTY PCU HAS FILLED
*         THEM WITH RUBBISH.
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB ALUINITPIA
045C: 48 62 29 1F 30 09 78 07 F0 01 04 D4
```

```
*         $5555 -> QREG,WREG,DREG
        ALU RQPT PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD $5555-1 & CONT
045D: C2 3B 69 5F 30 08 F9 87 F4 0E 55 54

*         SEND THE VALUE ACROSS TO NEIGHBOUR PROCESSOR
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB ALUSFSENDMES
045E: 48 62 29 1F 30 09 78 07 F0 01 04 CC

*         7 -> AM2910 COUNTER (N. OF SHIFTS - 1)
*         QREG -> XREG
        ALU REG PASS & AQ & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & ARAM WREG &
        BRAM XREG & PHLC 7
045F: C8 7A 69 1F 30 09 FA 1F F0 04 00 07

*         XREG SHIFT RIGHT -> XREG
*         LOOP BACK IF AM2910 COUNTER <> 0
        ALU LDR PASS & AB & OEY & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM XREG &
        RFCT
0460: C0 0A 29 1F 30 09 FA 07 F0 08 00 00

*         XREG -> DREG, SEND MESSAGE
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & ARAM XREG & JSB ALUSFSENDMES
0461: 48 63 69 5F 30 09 F8 27 F0 01 04 CC

*
*
* MEMORY TEST CODE (BYTE)
*
*
* MEMORY IS TESTED BY REPEATEDLY CALLING A SUBROUTINE
* WHICH TESTS 1K OF MEMORY, AS SOON AS A FAULTY BLOCK
* OF MEMORY IS DISCOVERED A MESSAGE IS SENT.
* IF MEMORY IS ALLRIGHT THEN A MESSAGE IS ALSO SENT.
*
*
*
*         8K -> XREG
BMEMTEST:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        DATAPATH & CARRYCTL & MEMCONT & CONTROL ROM & RTB &
        BRAM XREG & IMMD MEMMAX & CONT
0462: C2 7B 69 1F 30 08 FA 07 F0 0E 1C 00

*         0 -> R6,YREG, JUMP TO MEMORY TEST SUBROUTINE
        ALU REG LOW & AQ & OEY & PCU ,SUB PCUAB A6 B6 &
        AM2904 & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM YREG & JSB BMTSUB
0463: C8 7C 69 1F 9D 89 FA 87 F0 01 05 15

*         R6+1K -> R6, YREG + 1K -> YREG (R6,YREG IDENTICAL)
BMTLOOP:  ALU REG ADD & DAB & OEY & PCU ,,PCUDA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM &
```

```
       RTB & BRAM YREG & IMMD 1024 & CONT
0464: C2 79 A9 1F 5D 88 FA 87 F0 0E 04 00

*        IF MSR <> 0 THEN MEMORY IS FAULTY
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
       CJP *+12
0465: 40 62 29 1F 30 09 78 03 F2 53 04 67

*        ARRIVED HERE IF MEMORY IS FAULTY
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP BMEMFAULTY
0466: 48 62 29 1F 30 09 78 07 F0 03 04 6E

*        XREG - YREG -> YBUS, IF = 0 THEN TEST IS OVER
       ALU YBUS SUBS & AB & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & ARAM XREG & BRAM YREG & TEST DIRIN UANEB &
       CJP *+12
0467: C0 61 29 1F 30 09 FA A3 F7 43 04 69

*        ARRIVED HERE IF TEST IS OVER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP BMTESTEND
0468: 48 62 29 1F 30 09 78 07 F0 03 04 6B

*        JUMP TO MEMORY TEST SUBROUTINE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB BMTSUB
0469: 48 62 29 1F 30 09 78 07 F0 01 05 15

*        LOOP BACK
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP BMTLOOP
046A: 48 62 29 1F 30 09 78 07 F0 03 04 64

*        ARRIVED HERE IF ALL BLOCKS OF RAM HAVE BEEN SUCCESSFU
         LLY TESTED.
*        TEST SUCCESSFULL MESSAGE CODE -> WREG,DREG
BMTESTEND:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL ROM & RTB & BRAM WREG & IMMD BMTSUCMESS-1 &
       CONT
046B: C2 7B 69 5F 30 08 F9 87 F4 0E 00 0C

*        SEND THE MESSAGE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSFSENDMES
046C: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*        JUMP TO "WAIT FOR OP-CODE" CODE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP WAITOP
046D: 48 62 29 1F 30 09 78 07 F0 03 04 70

*        ARRIVED HERE IF MEMORY FAULTY
*        MEMORY TEST UNSUCCESSFULL MESSAGE -> WREG,DREG
BMEMFAULTY:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
```

```
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL ROM & RTB & BRAM WREG & IMMD BMTUNMESS-1 &
       CONT
046E: C2 7B 69 5F 30 08 F9 87 F4 0E 00 0D

*        SEND THE MESSAGE
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSFSENDMES
046F: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*
* NOW WAIT FOR MESSAGE CONTAINING TEST OP-CODE TO ARRIVE
* FROM NEIGHBOUR PROCESSOR
*
*        JUMP TO RECSUB
WAITOP:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSFRECMES
0470: 48 62 29 1F 30 09 78 07 F0 01 04 C5

*        IF MSR = 0 THEN OP-CODE HASN'T ARRIVED YET, JUMP BACK

       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH ZZI & MEMCONT & CONTROL &
       TEST MACHINE UAEQB & CJP WAITOP
0471: 50 62 29 1F 30 09 78 03 F2 53 04 70

*       REPLY CODE -> WREG,DREG
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL ROM & RTB & BRAM WREG & IMMD CORRECTREPLY &
       CONT
0472: C2 7B 69 5F 30 08 F9 87 F0 0E 00 AA

*        SEND THE REPLY
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSENDSUB
0473: 48 62 29 1F 30 09 78 07 F0 01 04 8D

*        DELAY TO ALLOW THE MONITOR PROCESSOR TO CATCH UP
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB DLY2910
0474: 48 62 29 1F 30 09 78 07 F0 01 05 27

*        CLEAR THE INTERRUPT COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & INTLOAD -1 & CONT
0475: 00 62 29 1F 30 09 78 07 F0 0E FF FF

*        JUMP MAP, VALID OP-CODE WILL JUMP TO MAPZITEST
*        OTHERWISE THE TEST SEQUENCE WILL (SOONER OR
*        LATER) BE ENTERED.
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & JMAP
0476: 48 62 29 1F 30 09 78 03 F0 02 00 00

*
* IF JMAP IS SUCCESSFULL THEN WE WILL ARRIVE HERE.
* SEND A MESSAGE
```

```
*
*         MESSAGE CODE -> WREG
MAPZITEST:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & IMMD MAPZIMESS-1 &
        CONT
0477: C2 7B 69 5F 30 08 F9 87 F4 0E 00 0E

*         SEND THE MESSAGE
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JSB JSFSENDMES
0478: 48 62 29 1F 30 09 78 07 F0 01 04 C4

*
* NOW TO SEND ACROSS THE RESULT AND LINE NO. OF THE
* CURRENT PROCESS. THIS IS IN CASE THERE IS A GENUINE
* SOFTWARE ERROR. NO PRECAUTIONS ARE TAKEN IN SENDING THE
* MESSAGE SINCE, IF A UNIT IS FAULTY THEN IT DOESN'T MATTER
* WHETHER THE MESSAGES ARRIVE AT THEIR DESTINATION OR NOT.
*
*         RESULT ATTRIBUTE ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD RESATR & CONT
0479: 40 62 29 9E 56 28 78 07 F0 0E 00 06

*         READ RESULT, LINE ATTRIBUTE ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDA PHREG & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB MREQ & CONTROL &
        IMMD LINEATR & CONT
047A: 40 62 29 9E 56 3A 78 07 F0 0E 00 04

*         READ LINE, ZREG (RESULT) -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD ,ENZ0 &
        MEMCONT REQB MREQ & CONTROL ROM & RTB & BRAM WREG &
        CONT
047B: C2 7B 69 4F 30 3B F9 87 F0 0E 00 00

*         ZREG (LINE) -> VREG, SEND RESULT
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM VREG & JSB JSENDMES
047C: CA 7B 69 0F 30 09 FB 87 F0 01 04 B8

*         VREG (LINE) -> WREG,DREG, SEND MESSAGE
        ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT &
        CONTROL ROM & RTB & ARAM VREG & BRAM WREG &
        JSB JSENDMES
047D: C8 7B 69 5F 30 09 F9 BF F0 01 04 B8

*         LOOP FOR EVER
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP *
047E: 48 62 29 1F 30 09 78 07 F0 03 04 7E

*
```

```
*
* SUBROUTINES
*
*
* SUBROUTINE TO TEST THE PCU WITH THE ALU.
* CALLED FROM TIMER INTERRUPT MICROCODE.
*
* CORRUPTS: Q OF PCU, EXP1
*
*         $FFFF -> EXP1 VIA PCU, $FFFF -> QREG
PATEST:   ALU RQPT PASSR & DAB & YOFF & PCU QEU ,PCUDZ &
        AM2904 & DATAPATH ,,,,,PCUY & MEMCONT & CONTROL ROM &
        RTB & BRAM EXP1 & IMMD $FFFF & CONT
047F: C3 3B 28 1E 70 08 FD 87 F0 0E FF FF

*         EXP1 - QREG($FFFF) -> YBUS, ERROR IF -VE
        ALU YBUS SUBS & AQ & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT & CONTROL ROM & ARAM EXP1 &
        TEST DIRIN NEGATIVE & CJPP SSYMESS
0480: 40 61 69 1F 30 09 F8 58 17 FB 04 08

*         ERROR IF <>
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
        CJP *+12
0481: 40 62 29 1F 30 09 78 03 F2 53 04 83

*         ARRIVED HERE IF TEST HAS FAILED
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JPOP SSYMESS
0482: 48 62 29 1F 30 09 78 07 F0 0B 04 08

*         $0000 -> EXP1 VIA PCU, $0000 -> QREG
        ALU RQPT PASSR & DAQ & YOFF & PCU QEU ,PCUDZ &
        AM2904 & DATAPATH ,,,,,PCUY & MEMCONT & CONTROL ROM &
        RTB & BRAM EXP1 & IMMD $0000 & CONT
0483: C3 3B 68 1E 70 08 FD 87 F0 0E 00 00

*         EXP1 - QREG ($0000) -> YBUS, ERROR IF -VE
        ALU YBUS SUBS & AQ & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT & CONTROL ROM & ARAM EXP1 &
        TEST DIRIN NEGATIVE & CJPP SSYMESS
0484: 40 61 69 1F 30 09 F8 58 17 FB 04 08

*         ERROR IF <>
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
        CJP *+12
0485: 40 62 29 1F 30 09 78 03 F2 53 04 87

*         ARRIVED HERE IF TEST HAS FAILED
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JPOP SSYMESS
0486: 48 62 29 1F 30 09 78 07 F0 0B 04 08

*         0 -> TREG
```

```
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,LDTREG & MEMCONT & CONTROL &
       IMMD 0 & CONT
0487: 42 63 79 1F 30 08 78 07 F0 0E 00 00

*        TREG (0) - Q OF PCU (0) -> Q OF PCU, ERROR IF <> 0
       ALU YBUS PASS & AB & PCU QEU SUB PCUDQ & AM2904 &
       DATAPATH ,ENTREG & MEMCONT & CONTROL & TESTF PCUNZ &
       CJPP SSYMESS
0488: 40 62 09 1E E0 09 78 07 F0 6B 04 08

*        PROGSTART+8 -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD PROGSTART+8 & CONT
0489: 40 62 29 9E 70 28 78 07 F0 0E 80 08

*        READ FROM MEMORY, $0084 -> EXP1
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM EXP1 & IMMD $0084 & CONT
048A: C2 7B 69 1F 30 3A FD 87 F0 0E 00 84

*        ZREG - EXP1 ($0084) -> YBUS, ERROR IF <> 0
       ALU YBUS SUBS & DAB & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
       CONTROL ROM & RTB & BRAM EXP1 & TEST DIRIN UANEB &
       CJPP SSYMESS
048B: C2 61 29 0F 30 09 FD 83 F7 4B 04 08

*        RETURN
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RTN
048C: 48 62 29 1F 30 09 78 07 F0 0A 00 00

*
* SEND A MESSAGE TO OTHER PROCESSOR USING BOTH ALU AND PCU
* WREG,DREG = MESSAGE  CORRUPTS Q OF PCU
* 1) PUT MESSAGE IN PIA DATA REGISTER
* 2) SET CA2 TO 1
* 3) RESET CA2 TO 0
*
*
*        SEND PIA DATA REG ADDRESS -> Q OF PCU,MAR
JSENDSUB:  ALU YBUS PASSR & AQ & OEY & PCU QEU ,PCUDZ &
       AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB & CONTROL ROM & ARAM WREG &
       IMMD SENDPIADR & CONT
048D: 40 63 69 9E 70 28 F8 1F F0 0E 40 60

*        WRITE VALUE TO SEND PIA, Q+1 OF PCU -> MAR (SEND PIA
         CONTROL REG ADDRES
*        $3C -> DREG
       ALU YBUS PASSR & DAQ & OEY & PCU QEU ,PCUAQ INCREG &
       AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
       IMMD $3C-1 & CONT
048E: 42 63 69 DE 08 3C 78 07 F4 0E 00 3B
```

```
*         WRITE $3C TO SEND PIA CONTROL REG (SETS CA2)
*         $34 -> DREG
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ ,WRITE MBYTE &
        CONTROL & IMMD $34-1 & CONT
048F: 42 63 69 5F 30 3C 70 00 04 0E 00 33

*         WRITE $34 TO SEND PIA CONTROL REG (CLEARS CA2), RETUR
          N
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & RTN
0490: 48 62 29 1F 30 3D 78 07 F0 0A 00 00

*
* RECEIVE SUBROUTINE USING BOTH ALU AND PCU
* RECEIVES A MESSAGE FROM THE OTHER PROCESSOR
* PUTS RESULT IN WREG
* SETS MSR = 0 IF CONTROL REG READY
* CORUPTS WREG , Q OF PCU
*
* 1) CHECK CONTROL REGISTER TO SEE IF READY, EXIT IF NOT
* 2) READ VALUE FROM PIA DATA REG AND PUT IN WREG
*
*         RECEIVE PIA CONTROL REGISTER ADDRESS -> MAR
JRECSUB:  ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD RECPIACR & CONT
0491: 40 62 29 9E 70 28 78 07 F0 0E 40 66

*         READ CONTROL REGISTER, $40 -> WREG
        ALU REG PASSR & DAQ & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM WREG & IMMD $40-1 & CONT
0492: C2 7B 69 1F 30 38 F9 87 F4 0E 00 3F

*         ZREG (CONTROL REGISTER CONTENTS) AND WREG ($40) -> YB
          US
*         LOAD MSR, RETURN IF = 0
        ALU YBUS AND & DAB & PCUNOP &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT & CONTROL ROM & RTB & BRAM WREG &
        TEST MACHINE DLOAD & CONT
0493: C2 66 29 0F 30 09 F9 84 12 7E 00 00

*         ZREG (CONTROL REGISTER CONTENTS) -> XREG
*         RETURN IF MSR = 0
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM XREG & TEST MACHINE UAEQB &
        CRTN
0494: C2 7B 69 0F 30 09 FA 03 F2 5A 00 00

*         ARRIVED HERE IF DATA AVAILABLE
*         RECEIVE PIA DATA REG ADDRESS -> MAR
JKICK:  ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
```

```
       IMMD RECPIADR & CONT
0495: 40 62 29 9E 70 28 78 07 F0 0E 40 64

*        READ PIA DATA REGISTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,,MBYTE & CONTROL & CONT
0496: 40 62 29 1F 30 39 78 07 F0 0E 00 00

*        ZREG (PIA DATA) -> WREG, RETURN
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
       CONTROL ROM & RTB & BRAM WREG & RTN
0497: CA 7B 69 0F 30 09 F9 87 F0 0A 00 00

*
* SUBROUTINE TO SEND A MESSAGE VIA THE ALU ONLY.
* ON ENTRY: DREG = MESSAGE
* CORRUPTED: XREG
*
*        SEND PIA DATA REG. ADDRESS -> XREG,MAR
ALUSENDSUB:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR & MEMCONT &
       CONTROL ROM & RTB & BRAM XREG & IMMD SENDPIADR-1 &
       CONT
0498: C2 7B 6B 9F 30 08 FA 07 F4 0E 40 5F

*        XREG+1 -> MAR (ADDRESS CONTROL REG.)
*        WRITE VALUE TO SEND PIA
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
       IMMD SENDPIACR-1 & CONT
0499: 42 63 6B 9F 30 3C 78 07 F4 0E 40 61

*        $3C -> DREG
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
       CONTROL & IMMD $3C-1 & CONT
049A: 42 63 69 5F 30 28 78 07 F4 0E 00 3B

*        WRITE $3C TO SEND PIA CONTROL REG (SETS CA2)
*        $34 -> DREG
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
       IMMD $34-1 & CONT
049B: 42 63 69 5F 30 3C 78 07 F4 0E 00 33

*        WRITE $34 TO SEND PIA CONTROL REG. (CLEARS CA2), RETU
         RN
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & RTN
049C: 48 62 29 1F 30 3D 78 07 F0 0A 00 00

*
* SUBROUTINE TO RECEIVE A MESSAGE FROM THE PIA USING
* THE ALU ONLY.
* ON EXIT: WREG = MESSAGE (IF READY), XREG = CONTROL REG. VALU
```

```
          E
*           MSR = 0 IF MESSAGE NOT READY, <> 0 IF READY.
*
*         RECEIVE PIA CONTROL REG. ADDRESS -> MAR
ALURECSUB:  ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB & CONTROL & IMMD RECPIACR-1 & CONT
049D: 42 63 6B 9F 30 28 78 07 F4 0E 40 65

*         READ CONTROL REGISTER, $40 -> WREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM WREG & IMMD $40-1 & CONT
049E: C2 7B 69 1F 30 38 F9 87 F4 0E 00 3F

*         ZREG (CONTROL REGISTER CONTENTS) AND WREG ($40) -> YB
          US
*         LOAD MSR
        ALU YBUS AND & DAB & PCUNOP &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH ,,,,,,,,,ENZ0 &
        MEMCONT & CONTROL ROM & RTB & BRAM WREG &
        TEST MACHINE DLOAD & CONT
049F: C2 66 29 0F 30 09 F9 84 12 7E 00 00

*         ZREG (CONTROL REG.) -> XREG, RETURN IF MSR = 0
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 ,OECT &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM XREG & TEST MACHINE UAEQB &
        CRTN
04A0: C2 7B 69 0F 30 09 FA 03 F2 5A 00 00

*         RECEIVE PIA DATA REG. ADDRESS -> MAR
ALUKICK:  ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB & CONTROL & IMMD RECPIADR-1 & CONT
04A1: 42 63 6B 9F 30 28 78 07 F4 0E 40 63

*         READ PIA DATA REGISTER
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL & CONT
04A2: 40 62 29 1F 30 39 78 07 F0 0E 00 00

*         ZREG (PIA DATA) -> WREG, RETURN
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 & MEMCONT &
        CONTROL ROM & RTB & BRAM WREG & RTN
04A3: CA 7B 69 0F 30 09 F9 87 F0 0A 00 00

*
*
* SUBROUTINE TO SEND A MESSAGE USING THE PCU ONLY.
* ON ENTRY: R6 = MESSAGE
* CORRUPTED: Q OF PCU
*
*         RE-INITIALISE THE PIA'S
*         THIS IS IN CASE A FAULTY ALU HAS FILLED
*         THEM WITH RUBBISH.
```

```
PCUSENDSUB:  ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL & JSB PCUINITPIA
04A4: 48 62 29 1F 30 09 78 07 F0 01 04 DC

*        R6 -> DREG
PCUS1:  ALU YBUS PASS & AB & YOFF & PCU ,,PCUZA A6 B6 &
       AM2904 & DATAPATH ,,,,,PCUY ,LDD & MEMCONT & CONTROL &
       CONT
04A5: 41 62 28 5F 4D 89 78 07 F0 0E 00 00

*        SEND PIA DATA REG.ADDRESS -> Q OF PCU,MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD SENDPIADR & CONT
04A6: 40 62 29 9E 70 28 78 07 F0 0E 40 60

*        WRITE VALUE TO SEND PIA, Q+1 -> MAR (CONTROL REG. ADD
         RESS)
       ALU YBUS PASS & AB & PCU QEU ,PCUDQ & AM2904 &
       DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
       IMMD INTMEM & CONT
04A7: 40 62 29 9E 60 3C 78 07 F0 0E 00 02

*        $3C -> DREG
       ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB & CONTROL &
       IMMD $3C & CONT
04A8: 41 62 28 5E 70 28 78 07 F0 0E 00 3C

*        WRITE $3C TO CONTROL REG. (SETS CA2), $34 -> DREG
       ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,PCUY ,LDD &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & IMMD $34 &
       CONT
04A9: 41 62 28 5E 70 3C 78 07 F0 0E 00 34

*        WRITE $34 TO CONTROL REG. (CLEARS CA2), RETURN
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & RTN
04AA: 48 62 29 1F 30 3D 78 07 F0 0A 00 00

*
* SUBROUTINE TO RECEIVE A MESSAGE FROM THE PIA USING THE PCU O
        NLY.
* ON EXIT: R6 = MESSAGE, R7 = CONTROL REG. CONTENTS
* CORRUPTED: Q OF PCU
* IF THE 6 L.S. BITS OF THE CONTROL REG. ARE NOT $14
* (THEIR INITIALISATION VALUE) THEN A DELAY IS EXECUTED SINCE
* THERE IS MOST LIKELY A FAULT IN THE DATAPATH (E.G. ZREG OR
* Z0REG). THE DELAY GIVES THE NEIGHBOUR PROCESSOR TIME TO ACCE
        SS THE
* MESSAGE BEFORE ANOTHER ONE IS SENT.
*
*        RECEIVE PIA CONTROL REG. ADDRESS -> MAR
PCURECSUB:  ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
      DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RECPIACR & CONT
```

```
04AB: 40 62 29 9E 70 28 78 07 F0 0E 40 66

*         READ CONTROL REGISTER, $54 -> R7 OF PCU
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH & MEMCONT REQB MREQ ,,MBYTE & CONTROL &
        IMMD $54 & CONT
04AC: 40 62 29 1F 7F F8 78 07 F0 0E 00 54

*         R7 ($54) - ZREG (CONTROL REG. CONTENTS) -> R7
*         IF = THEN DATA IS READY
        ALU YBUS PASS & AB & PCU ,SUB PCUDA A7 B7 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        TESTF PCUZERO & CJP PCURDATA
04AD: 40 62 29 0F DF C9 78 07 F0 53 04 B5

*         $D4 -> R7 OF PCU
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & IMMD $D4 & CONT
04AE: 40 62 29 1F 7F C8 78 07 F0 0E 00 D4

*         ZREG (CONTROL REG.) - R7 ($D4) -> R7
*         IF = 0 THEN DATA IS READY
        ALU YBUS PASS & AB & PCU ,SUB PCUDA A7 B7 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        TESTF PCUZERO & CJP PCURDATA
04AF: 40 62 29 0F DF C9 78 07 F0 53 04 B5

*         $14 -> R7 OF PCU
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & IMMD $14 & CONT
04B0: 40 62 29 1F 7F C8 78 07 F0 0E 00 14

*         ZREG (CONTROL REG.) - R7 OF PCU ($14) -> R7
*         IF = 0 THEN CONTROL REG. IS O.K. BUT DATA IS
*         NOT READY SO LOOP BACK.
        ALU YBUS PASS & AB & PCU ,SUB PCUDA A7 B7 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        TESTF PCUZERO & CJP PCURECSUB
04B1: 40 62 29 0F DF C9 78 07 F0 53 04 AB

*         $94 -> R7 OF PCU
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH & MEMCONT & CONTROL & IMMD $94 & CONT
04B2: 40 62 29 1F 7F C8 78 07 F0 0E 00 94

*         ZREG (CONTROL REG.) - R7 ($94) -> R7
*         IF = 0 THEN CONTROL REG IS O.K. BUT
*         DATA IS NOT READY SO LOOP BACK.
        ALU YBUS PASS & AB & PCU ,SUB PCUDA A7 B7 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        TESTF PCUZERO & CJP PCURECSUB
04B3: 40 62 29 0F DF C9 78 07 F0 53 04 AB

*         ZREG -> R7 OF PCU, JUMP TO DELAY SUBROUTINE
        ALU YBUS PASS & AB & PCU ,,PCUDZ A7 B7 & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL &
        JSB DLY2910
04B4: 48 62 29 0F 7F C9 78 07 F0 01 05 27
```

```
*         NOTE: ENTER HERE AFTER DLY2910 TO KICK THE PIA
*         RECEIVE PIA DATA REG. ADDRESS -> MAR
PCURDATA:  ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
       IMMD RECPIADR & CONT
04B5: 40 62 29 9E 70 28 78 07 F0 0E 40 64

*         READ PIA DATA REGISTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT REQB MREQ ,,MBYTE & CONTROL & CONT
04B6: 40 62 29 1F 30 39 78 07 F0 0E 00 00

*         ZREG (PIA DATA) -> R6, RETURN
       ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT & CONTROL & RTN
04B7: 48 62 29 0F 7D 89 78 07 F0 0A 00 00

*
* SUBROUTINE TO SEND A MESSAGE USING ALU AND PCU
* WREG,DREG = MESSAGE
*      1) SEND A MESSAGE
*      2) WAIT FOR A REPLY
*
*         CALL JSENDSUB
JSENDMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JSENDSUB
04B8: 48 62 29 1F 30 09 78 07 F0 01 04 8D

*         CALL JRECSUB
JRECMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JRECSUB
04B9: 48 62 29 1F 30 09 78 07 F0 01 04 91

*         IF MSR = 0 THEN NO REPLY RECEIVED, LOOP BACK
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
       CJP JRECMES
04BA: 40 62 29 1F 30 09 78 03 F2 53 04 B9

*         RETURN
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RTN
04BB: 48 62 29 1F 30 09 78 07 F0 0A 00 00

*
* SUBROUTINE TO SEND A MESSAGE USING ALU ONLY
* DREG = MESSAGE
*      1) SEND A MESSAGE
*      2) WAIT FOR A REPLY
*
*         CALL ALUSENDSUB
ALUSENDMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL & JSB ALUSENDSUB
04BC: 48 62 29 1F 30 09 78 07 F0 01 04 98

*         CALL ALURECSUB
ALURECMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
```

```
      MEMCONT & CONTROL & JSB ALURECSUB
04BD: 48 62 29 1F 30 09 78 07 F0 01 04 9D

*        IF MSR = 0 THEN NO REPLY RECEIVED, RETURN
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE UANEB &
       CJP ALURECMES
04BE: 40 62 29 1F 30 09 78 03 F2 43 04 BD

*        LOOP BACK
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RTN
04BF: 48 62 29 1F 30 09 78 07 F0 0A 00 00

*
* SUBROUTINE TO SEND A MESSAGE USING PCU ONLY
* R6 = MESSAGE
*     1) SEND A MESSAGE
*     2) WAIT FOR A REPLY
*
*        CALL PCUSENDSUB
PCUSENDMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL & JSB PCUSENDSUB
04C0: 48 62 29 1F 30 09 78 07 F0 01 04 A4

*        CALL PCURECSUB
PCURECMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP PCURECSUB
04C1: 48 62 29 1F 30 09 78 07 F0 03 04 AB

*
* SUBROUTINE TO SEND A MESSAGE USING PCU ONLY (NO INITIALISATI
        ON)
* R6 = MESSAGE
*     1) SEND A MESSAGE
*     2) WAIT FOR A REPLY
*
*        CALL PCUSENDSUB
PCUNCSEND:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB PCUS1
04C2: 48 62 29 1F 30 09 78 07 F0 01 04 A5

*        CALL PCURECSUB
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP PCURECSUB
04C3: 48 62 29 1F 30 09 78 07 F0 03 04 AB

*
* SUBROUTINE TO SEND A MESSAGE USING THE ALU AND THE PCU
* WHEN A FAILURE OF THE ZREG IS CRITICAL.
* 1) CALL SENDSUB.
* 2) CALL RECSUB
* 3) IF XREG (= CONTROL REG.) IS NOT $14 OR $54 (I.E. CORRECT
* INITIALISATION VALUE WITH CA2 = DON'T CARE) THEN AN ERROR
* HAS OCCURED AND A DELAY IS MADE TO ALLOW THE NEIGHBOUR
* PROCESSOR TO PROCESS THE MESSAGE
* CORRUPTED: WREG,XREG
*
```

```
*         SEND THE MESSAGE
JSFSENDMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL & JSB JSENDSUB
04C4: 48 62 29 1F 30 09 78 07 F0 01 04 8D

*         CALL THE RECEIVE SUBROUTINE TO SEE IF A REPLY
*         HAS ARRIVED.
JSFRECMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB JRECSUB
04C5: 48 62 29 1F 30 09 78 07 F0 01 04 91

*         XREG AND $7F -> XREG (MASKS OUT CA1)
JSFCODE:  ALU REG AND & DAB & OEY & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM XREG &
       IMMD $7F & CONT
04C6: C2 7E 29 1F 30 08 FA 07 F0 0E 00 7F

*         XREG - $14 -> YBUS, IF = 0 THEN CONTROL REG. IS O.K.
*         BUT REPLY IS NOT RECEIVED YET SO LOOP BACK
       ALU YBUS SUBR & DAB & PCUNOP &
       AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM XREG &
       TEST MACHINE DLOAD & IMMD $14 & CONT
04C7: C2 60 A9 1F 30 08 FA 04 16 7E 00 14

*         IF MSR = 0 THEN LOOP BACK
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
       CJP JSFRECMES
04C8: 40 62 29 1F 30 09 78 03 F2 53 04 C5

*         XREG - $54 -> YBUS, RETURN IF = 0
       ALU YBUS SUBS & DAB & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & BRAM XREG & TEST DIRIN UAEQB & IMMD $54 & CRTN
04C9: C2 61 29 1F 30 08 FA 03 F7 5A 00 54

*         ARRIVED HERE IF THERE IS AN ERROR, JUMP TO DELAY
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB DLY2910
04CA: 48 62 29 1F 30 09 78 07 F0 01 05 27

*         KICK THE PIA
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP JKICK
04CB: 48 62 29 1F 30 09 78 07 F0 03 04 95

*
* SUBROUTINE TO SEND A MESSAGE USING THE ALU ONLY
* WHEN A FAILURE OF THE ZREG IS CRITICAL.
* 1) CALL SENDSUB.
* 2) CALL RECSUB
* 3) IF XREG (= CONTROL REG.) IS NOT $14 OR $54 (I.E. CORRECT
* INITIALISATION VALUE WITH CA2 = DON'T CARE) THEN AN ERROR
* HAS OCCURED AND A DELAY IS MADE TO ALLOW THE NEIGHBOUR
* PROCESSOR TO PROCESS THE MESSAGE
* CORRUPTED: WREG,XREG
*
```

```
*         SEND THE MESSAGE
ALUSFSENDMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL & JSB ALUSENDSUB
04CC: 48 62 29 1F 30 09 78 07 F0 01 04 98

*         CALL THE RECEIVE SUBROUTINE TO SEE IF A REPLY
*         HAS ARRIVED.
ALUSFRECMES:  ALU YBUS PASS & AB & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL & JSB ALURECSUB
04CD: 48 62 29 1F 30 09 78 07 F0 01 04 9D

*         XREG AND $7F -> XREG (MASKS OUT CA1)
ALUSFCODE:  ALU REG AND & DAB & OEY & PCUNOP & AM2904 &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM XREG &
       IMMD $7F & CONT
04CE: C2 7E 29 1F 30 08 FA 07 F0 0E 00 7F

*         XREG - $14 -> YBUS, IF = 0 THEN CONTROL REG. IS O.K.
*         BUT REPLY IS NOT RECEIVED YET SO LOOP BACK
       ALU YBUS SUBR & DAB & PCUNOP &
       AM2904 ,,EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM XREG &
       TEST MACHINE DLOAD & IMMD $14 & CONT
04CF: C2 60 A9 1F 30 08 FA 04 16 7E 00 14

*         IF MSR = 0 THEN LOOP BACK
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT & CONTROL & TEST MACHINE UAEQB &
       CJP ALUSFRECMES
04D0: 40 62 29 1F 30 09 78 03 F2 53 04 CD

*         XREG - $54 -> YBUS, RETURN IF = 0
       ALU YBUS SUBS & DAB & PCUNOP & AM2904 ,OECT &
       CARRYCTL COEQ1 & DATAPATH & MEMCONT & CONTROL ROM &
       RTB & BRAM XREG & TEST DIRIN UAEQB & IMMD $54 & CRTN
04D1: C2 61 29 1F 30 08 FA 03 F7 5A 00 54

*         ARRIVED HERE IF THERE IS AN ERROR, JUMP TO DELAY
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JSB DLY2910
04D2: 48 62 29 1F 30 09 78 07 F0 01 05 27

*         KICK THE PIA
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP ALUKICK
04D3: 48 62 29 1F 30 09 78 07 F0 03 04 A1

*
* SUBROUTINE TO INITIALISE THE PIA'S USING
* THE ALU ONLY.
* NOTHING CORRUPTED.
*
*         $30 -> DREG
ALUINITPIA:  ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL & IMMD $30-1 & CONT
04D4: 42 63 69 5F 30 08 78 07 F4 0E 00 2F
```

```
*         SEND PIA CONTROL REG. ADDRESS -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB & CONTROL & IMMD SENDPIACR-1 & CONT
04D5: 42 63 6B 9F 30 28 78 07 F4 0E 40 61

*         WRITE 0 TO SEND PIA CONTROL REG. (ACCESS DDR)
*         SEND PIA DATA REG. ADDRESS -> MAR.
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD SENDPIADR-1 & CONT
04D6: 42 63 6B 9F 30 3C 78 07 F4 0E 40 5F

*         $FF -> DREG
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
        CONTROL & IMMD $FF-1 & CONT
04D7: 42 63 69 5F 30 28 78 07 F4 0E 00 FE

*         WRITE $FF TO DDR (SET FOR OUTPUT)
*         SEND PIA CONTROL REG. ADDRESS -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD SENDPIACR-1 & CONT
04D8: 42 63 6B 9F 30 3C 78 07 F4 0E 40 61

*         $34 -> DREG
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT REQB &
        CONTROL & IMMD $34-1 & CONT
04D9: 42 63 69 5F 30 28 78 07 F4 0E 00 33

*         WRITE $34 TO SEND PIA CONTROL REG.
*         REC PIA DATA REG. ADDRESS -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD RECPIADR-1 & CONT
04DA: 42 63 6B 9F 30 3C 78 07 F4 0E 40 63

*         KICK THE PIA,RETURN
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL & RTN
04DB: 48 62 29 1F 30 39 78 07 F0 0A 00 00

*
* SUBROUTINE TO INITIALISE THE PIA'S USING THE
* PCU ONLY.
* CORRUPTED: Q OF PCU
*
*         $30 -> DREG
PCUINITPIA:  ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ &
        AM2904 & DATAPATH ,,,,,PCUY ,LDD & MEMCONT & CONTROL &
        IMMD $30 & CONT
04DC: 41 62 28 5E 70 08 78 07 F0 0E 00 30
```

```
*         SEND PIA CONTROL REG. ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD SENDPIACR & CONT
04DD: 40 62 29 9E 70 28 78 07 F0 0E 40 62

*         WRITE 0 TO SEND PIA CONTROL REG. (ACCESS DDR)
*         SEND PIA DATA REG.ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD SENDPIADR & CONT
04DE: 40 62 29 9E 70 3C 78 07 F0 0E 40 60

*         $FF -> DREG
        ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB & CONTROL &
        IMMD $FF & CONT
04DF: 41 62 28 5E 70 28 78 07 F0 0E 00 FF

*         WRITE $FF TO DDR (SET FOR OUTPUT)
*         SEND PIA CONTROL REG.ADDRESS -> MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD SENDPIACR & CONT
04E0: 40 62 29 9E 70 3C 78 07 F0 0E 40 62

*         $34 -> DREG
        ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB & CONTROL &
        IMMD $34 & CONT
04E1: 41 62 28 5E 70 28 78 07 F0 0E 00 34

*         WRITE $34 TO SEND PIA CONTROL REG.
*         REC. PIA DATA REG ADDRESS -> MAR.
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD RECPIADR & CONT
04E2: 40 62 29 9E 70 3C 78 07 F0 0E 40 64

*         KICK THE PIA, RETURN
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL & RTN
04E3: 48 62 29 1F 30 39 78 07 F0 0A 00 00

*
* SUBROUTINE TO TEST 1K BLOCK OF MEMORY.
*
* 1) READ PRESENT MEMORY CONTENTS AND SAVE.
* 2) WRITE + READ BACK THE VALUE $AAAA.
* 3) WRITE + READ BACK THE VALUE $5555.
* 4) RESTORE THE ORIGINAL CONTENTS.
* ABOVE PROCEDURE CARRIED OUT FOR EACH WORD.
*
* AT START : R6 = MEMORY BLOCK START ADDRESS
* CORRUPTED : R6,R7,Q OF PCU, YREG,VREG,REAL1M
```

```
* MSR = 0 IF MEMORY O.K., <> 0 IF MEMORY NOT O.K.
*
* R6+1K -> R7
JMTSUB:  ALU YBUS PASS & AB & PCU ,,PCUDA A6 B7 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD 1024 & CONT
04E4: 40 62 29 1F 5D C8 78 07 F0 0E 04 00

*        R6 -> MAR, $5555 -> REAL2M
       ALU REG PASSR & DAQ & PCU ,,PCUZA A6 B6 & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
       CONTROL ROM & RTB & BRAM REAL2M & IMMD $5555 & CONT
04E5: C2 7B 69 9F 4D A8 FE 07 F0 0E 55 55

*        READ NEXT WORD, $AAAA -> VREG,DREG
       ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
       CONTROL ROM & RTB & BRAM VREG & IMMD $AAAA & CONT
04E6: C2 7B 69 5F 30 3A FB 87 F0 0E AA AA

*        ZREG (NEXT WORD) -> REAL1M, WRITE $AAAA TO MEMORY
JMTRDNW:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,,,ENZ0 &
       MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
       BRAM REAL1M & CONT
04E7: C2 7B 69 0F 30 3F FC 07 F0 0E 00 00

*        READ $AAAA BACK, REAL2M ($5555) -> DREG
       ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
       CONTROL ROM & ARAM REAL2M & CONT
04E8: 40 63 69 5F 30 3B F8 67 F0 0E 00 00

*        ZREG - VREG -> YBUS, ERROR IF -VE, WRITE $5555
       ALU YBUS SUBS & DAB & PCUNOP &
       AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ ,WRITE &
       CONTROL ROM & RTB & BRAM VREG & TEST DIRIN NEGATIVE &
       CJP JMTSERR
04E9: C2 61 29 0F 30 3F FB 80 17 F3 04 F2

*        IF MSR IS <> THEN THERE IS AN ERROR
       ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
       DATAPATH & MEMCONT REQB & CONTROL &
       TEST MACHINE UAEQB & CJP JMOK1
04EA: 40 62 29 1F 30 29 78 03 F2 53 04 EC

*        ARRIVED HERE IF MEMORY FAULTY
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & JUMP JMTSERR
04EB: 48 62 29 1F 30 09 78 07 F0 03 04 F2

*        READ $5555 BACK, REAL1M (ORIGINAL WORD) -> DREG
JMOK1:  ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
       CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
       CONTROL ROM & ARAM REAL1M & CONT
04EC: 40 63 69 5F 30 3B F8 47 F0 0E 00 00

*        ZREG - REAL2M -> YBUS, JUMP TO ERROR IF -VE, WRITE BA
```

```
          CK ORIGINAL WORD
*         THIS TEST WILL DETECT A FAULTY ALU M.S.
*         R6+2 -> MAR
        ALU YBUS SUBS & DAB & PCU ,,PCUAB A4 B6 &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR ,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM REAL2M & TEST DIRIN NEGATIVE & CJP JMTSERR
04ED: C2 61 29 8F 19 BF FE 00 17 F3 04 F2

*         IF MSR IS <> THEN THERE IS AN ERROR
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT REQB & CONTROL &
        TEST MACHINE UAEQB & CJP JMOK2
04EE: 40 62 29 1F 30 29 78 03 F2 53 04 F0

*         ARRIVED HERE IF MEMORY FAULTY
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP JMTSERR
04EF: 48 62 29 1F 30 09 78 07 F0 03 04 F2

*         READ NEXT WORD, VREG -> DREG, R7 - R6 -> Q, LOOP BACK
           IF <> 0
JMOK2:   ALU YBUS PASSR & AQ & OEY & PCU QEU SUB PCUAB A6 B7 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ & CONTROL ROM & ARAM VREG &
        TESTF PCUNZ & CJP JMTRDNW
04F0: 40 63 69 5E 9D FB F8 3F F0 63 04 E7

*         TEST PASSED, SET MSR = 0, RETURN, R6 - 1K -> R6
        ALU YBUS PASS & AB & PCU ,SUB PCUDA A6 B6 &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH & MEMCONT &
        CONTROL & TESTF $01 & IMMD 1024 & RTN
04F1: 48 62 29 1F DD 88 78 04 10 1A 04 00

*         TEST FAILED, SET MSR <> 0, RETURN, R6 - 1K -> R6
JMTSERR:   ALU YBUS PASS & AB & PCU ,SUB PCUDA A6 B6 &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH & MEMCONT &
        CONTROL & TESTF $03 & IMMD 1024 & RTN
04F2: 48 62 29 1F DD 88 78 04 10 3A 04 00

*
* SUBROUTINE TO TEST 1K BLOCK OF MEMORY USING THE ALU ONLY
* ON ENTRY: YREG = MEMORY BLOCK START ADDRESS
* CORRUPTED: VREG,QREG,REAL1M,REAL2M,EXP1
* ON EXIT: MSR = 0 IF TEST PASSED, <> 0 IF FAILED
*
*         YREG+1K -> QREG
ALUMTSUB:   ALU QPT ADD & DAB & PCUNOP & AM2904 & CARRYCTL &
        DATAPATH & MEMCONT & CONTROL ROM & RTB & BRAM YREG &
        IMMD 1024 & CONT
04F3: C2 31 A9 1F 30 08 FA 87 F0 0E 04 00

*         YREG -> EXP1,MAR
        ALU REG PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,YMAR LDMAR & MEMCONT &
        CONTROL ROM & RTB & ARAM YREG & BRAM EXP1 & CONT
04F4: C0 7B 6B 9F 30 09 FD AF F0 0E 00 00
```

```
*         $5555 -> REAL2M
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL ROM & RTB &
        BRAM REAL2M & IMMD $5555 & CONT
04F5: C2 7B 69 1F 30 08 FE 07 F0 0E 55 55

*         READ NEXT WORD, $AAAA -> VREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
        CONTROL ROM & RTB & BRAM VREG & IMMD $AAAA & CONT
04F6: C2 7B 69 5F 30 3A FB 87 F0 0E AA AA

*         ZREG (NEXT WORD) -> REAL1M, WRITE $AAAA TO MEMORY
ALUMTRDNW:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE & CONTROL ROM & RTB &
        BRAM REAL1M & CONT
04F7: C2 7B 69 0F 30 3F FC 07 F0 0E 00 00

*         READ $AAAA BACK, REAL2M ($5555) -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
        CONTROL ROM & ARAM REAL2M & CONT
04F8: 40 63 69 5F 30 3B F8 67 F0 0E 00 00

*         ZREG - VREG -> YBUS, ERROR IF -VE, WRITE $5555 TO MEM
          ORY
*         THIS TEST WILL DETECT A FAULTY ALU M.S.
        ALU YBUS SUBS & DAB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ ,WRITE &
        CONTROL ROM & RTB & BRAM VREG & TEST DIRIN NEGATIVE &
        CJP ALUMTERR
04F9: C2 61 29 0F 30 3F FB 80 17 F3 05 05

*         IF MSR IS <> THEN THERE IS AN ERROR
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT REQB & CONTROL &
        TEST MACHINE UAEQB & CJP ALUMOK1
04FA: 40 62 29 1F 30 29 78 03 F2 53 04 FC

*         ARRIVED HERE IF MEMORY FAULTY
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP ALUMTERR
04FB: 48 62 29 1F 30 09 78 07 F0 03 05 05

*         READ $5555 BACK, REAL1M (ORIGINAL WORD) -> DREG
ALUMOK1:  ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
        CONTROL ROM & ARAM REAL1M & CONT
04FC: 40 63 69 5F 30 3B F8 47 F0 0E 00 00

*         ZREG - REAL2M -> YBUS, ERROR IF -VE
*         THIS TEST WILL DETECT A FAULTY ALU M.S.
*         WRITE BACK ORIGINAL WORD
        ALU YBUS SUBS & DAB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
```

```
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ ,WRITE &
        CONTROL ROM & RTB & BRAM REAL2M &
        TEST DIRIN NEGATIVE & CJP ALUMTERR
04FD: C2 61 29 0F 30 3F FE 00 17 F3 05 05

*         IF MSR IS <> THEN THERE IS AN ERROR
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT REQB & CONTROL &
        TEST MACHINE UAEQB & CJP ALUMOK2
04FE: 40 62 29 1F 30 29 78 03 F2 53 05 00

*         ARRIVED HERE IF MEMORY FAULTY
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP ALUMTERR
04FF: 48 62 29 1F 30 09 78 07 F0 03 05 05

*         EXP1+2 -> EXP1,MAR (NEXT MEMORY ADDRESS)
ALUMOK2:  SPF14 INCTWO & AB & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,YMAR LDMAR &
        MEMCONT REQB & CONTROL ROM & RTB & BRAM EXP1 & CONT
0500: C0 20 2B 9F 30 29 FD 87 F4 0E 00 00

*         READ NEXT WORD, VREG -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD & MEMCONT REQB MREQ &
        CONTROL ROM & ARAM VREG & CONT
0501: 40 63 69 5F 30 3B F8 3F F0 0E 00 00

*         QREG - EXP1 -> YBUS, EXIT IF = 0
        ALU YBUS SUBR & AQ & PCUNOP & AM2904 ,OECT &
        CARRYCTL COEQ1 & DATAPATH & MEMCONT REQB &
        CONTROL ROM & ARAM EXP1 & TEST DIRIN UAEQB &
        CJP ALUMTPASS
0502: 40 60 E9 1F 30 29 F8 5B F7 53 05 04

*         LOOP BACK
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP ALUMTRDNW
0503: 48 62 29 1F 30 09 78 07 F0 03 04 F7

*         TEST PASSED IF ARRIVED HERE, SET MSR = 0, RETURN
ALUMTPASS:  ALU YBUS PASS & AB & PCUNOP &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH & MEMCONT &
        CONTROL & TESTF $01 & RTN
0504: 48 62 29 1F 30 09 78 04 10 1A 00 00

*         ARRIVED HERE IF TEST FAILED, SET MSR <> 0, RETURN
ALUMTERR:  ALU YBUS PASS & AB & PCUNOP &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH & MEMCONT &
        CONTROL & TESTF $03 & RTN
0505: 48 62 29 1F 30 09 78 04 10 3A 00 00

*
* SUBROUTINE TO TEST 1K BLOCK OF MEMORY USING THE PCU ONLY.
* ON ENTRY: R6 = START ADDRESS OF MEMORY BLOCK TO BE TESTED.
* CORRUPTED: R2,R5,R6,R7,Q
* ON EXIT: R0 = 0 IF TEST PASSED, <> 0 IF TEST FAILS
*
```

```
*         R6+1K -> R7
PCUMTSUB:  ALU YBUS PASS & AB & PCU ,,PCUDA A6 B7 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD 1024 & CONT
0506: 40 62 29 1F 5D C8 78 07 F0 0E 04 00

*         R6 -> R2,MAR
       ALU YBUS PASS & AB & PCU ,,PCUZA A6 B2 & AM2904 &
       DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & CONT
0507: 40 62 29 9F 4C A9 78 07 F0 0E 00 00

*         READ NEXT WORD, $AAAA -> DREG
       ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB MREQ &
       CONTROL & IMMD $AAAA & CONT
0508: 41 62 28 5E 70 3A 78 07 F0 0E AA AA

*         ZREG (NEXT WORD) -> R5, WRITE $AAAA TO MEMORY
PCUMTRDNW:  ALU YBUS PASS & AB & PCU ,,PCUDZ A5 B5 & AM2904 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ ,WRITE &
       CONTROL & CONT
0509: 40 62 29 0F 7B 7F 78 07 F0 0E 00 00

*         READ $AAAA BACK, $5555 -> DREG
       ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB MREQ &
       CONTROL & IMMD $5555 & CONT
050A: 41 62 28 5E 70 3A 78 07 F0 0E 55 55

*         $AAAA -> Q OF PCU
       ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD $AAAA & CONT
050B: 40 62 29 1E 70 08 78 07 F0 0E AA AA

*         Q OF PCU ($AAAA) - ZREG -> Q OF PCU, ERROR IF <> 0
*         WRITE $5555
       ALU YBUS PASS & AB & PCU QEU SUB PCUDQ & AM2904 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ ,WRITE &
       CONTROL & TESTF PCUNZ & CJP PCUMTERR
050C: 40 62 29 0E E0 3F 78 07 F0 63 05 14

*         READ $5555 BACK, R5 (ORIGINAL WORD) -> DREG
       ALU YBUS PASS & AB & YOFF & PCU ,,PCUZA A5 B5 &
       AM2904 & DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB MREQ &
       CONTROL & CONT
050D: 41 62 28 5F 4B 7B 78 07 F0 0E 00 00

*         $5555 -> Q OF PCU
       ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH & MEMCONT REQB & CONTROL & IMMD $5555 &
       CONT
050E: 40 62 29 1E 70 28 78 07 F0 0E 55 55

*         Q OF PCU ($5555) - ZREG -> Q OF PCU, ERROR IF <> 0,
*         WRITE BACK ORIGINAL WORD
       ALU YBUS PASS & AB & PCU QEU SUB PCUDQ & AM2904 &
       DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB MREQ ,WRITE &
       CONTROL & TESTF PCUNZ & CJP PCUMTERR
050F: 40 62 29 0E E0 3F 78 07 F0 63 05 14
```

```
*         R2+2 -> MAR
        ALU YBUS PASS & AB & PCU ,,PCUAB A4 B2 & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL & CONT
0510: 40 62 29 9F 18 A9 78 07 F0 0E 00 00

*         READ NEXT WORD, $AAAA -> DREG
        ALU YBUS PASS & AB & YOFF & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,PCUY ,LDD & MEMCONT REQB MREQ &
        CONTROL & IMMD $AAAA & CONT
0511: 41 62 28 5E 70 3A 78 07 F0 0E AA AA

*         R7 - R2 -> Q, LOOP BACK IF <> 0
        ALU YBUS PASS & AB & PCU QEU SUB PCUAB A2 B7 &
        AM2904 & DATAPATH & MEMCONT REQB & CONTROL &
        TESTF PCUNZ & CJP PCUMTRDNW
0512: 40 62 29 1E 95 E9 78 07 F0 63 05 09

*         TEST PASSED, 0 -> R0,RETURN
        ALU YBUS PASS & AB &
        PCU ,SUB PCUAB A0 B0 & AM2904 & DATAPATH & MEMCONT &
        CONTROL & RTN
0513: 48 62 29 1F 90 09 78 07 F0 0A 00 00

*         TEST FAILED, $FFFF -> R0, RETURN
PCUMTERR:  ALU YBUS PASS & AB & PCU ,,PCUDZ A0 B0 & AM2904 &
       DATAPATH & MEMCONT & CONTROL & IMMD $FFFF & RTN
0514: 48 62 29 1F 70 08 78 07 F0 0A FF FF

*
* SUBROUTINE TO TEST 1K BLOCK OF MEMORY (BYTE MODE).
*
* 1) READ PRESENT MEMORY CONTENTS AND SAVE.
* 2) WRITE + READ BACK THE VALUE $AA.
* 3) WRITE + READ BACK THE VALUE $55.
* 4) RESTORE THE ORIGINAL CONTENTS.
* ABOVE PROCEDURE CARRIED OUT FOR EACH BYTE.
*
* AT START : R6 = MEMORY BLOCK START ADDRESS
* CORRUPTED : R6,R7,Q OF PCU, YREG,VREG,REAL1M,REAL2M,QREG,EXP
          2
* MSR = 0 IF MEMORY O.K., <> 0 IF MEMORY NOT O.K.
*
* R6+1K -> R7
BMTSUB:  ALU YBUS PASS & AB & PCU ,,PCUDA A6 B7 & AM2904 &
      DATAPATH & MEMCONT & CONTROL & IMMD 1024 & CONT
0515: 40 62 29 1F 5D C8 78 07 F0 0E 04 00

*         R6 -> MAR, $55 -> REAL2M
        ALU REG PASSR & DAQ & PCU ,,PCUZA A6 B6 & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR & MEMCONT REQB &
        CONTROL ROM & RTB & BRAM REAL2M & IMMD $55 & CONT
0516: C2 7B 69 9F 4D A8 FE 07 F0 0E 00 55

*         $00FF -> QREG (USED AS A MASK LATER)
        ALU QPT PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH & MEMCONT & CONTROL & IMMD $00FF &
        CONT
```

```
0517: 42 33 69 1F 30 08 78 07 F0 0E 00 FF

*         READ NEXT BYTE, $AA -> VREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & RTB &
        BRAM VREG & IMMD $AA & CONT
0518: C2 7B 69 5F 30 38 FB 87 F0 0E 00 AA

*         ZREG (NEXT BYTE) -> REAL1M, WRITE $AA TO MEMORY
BMTRDNW:  ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,,ENZ0 &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL ROM & RTB &
        BRAM REAL1M & CONT
0519: C2 7B 69 0F 30 3D FC 07 F0 0E 00 00

*         READ $AA BACK, REAL2M ($55) -> DREG
        ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM &
        ARAM REAL2M & CONT
051A: 40 63 69 5F 30 39 F8 67 F0 0E 00 00

*         ZREG AND QREG ($00FF) -> EXP2 (MASKS OUT M.S. BYTE)
        ALU REG AND & DAQ & OEY & PCUNOP & AM2904 &
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM EXP2 & CONT
051B: C2 7E 69 0F 30 29 FF 87 F0 0E 00 00

*         EXP2 - VREG -> YBUS, ERROR IF -VE, WRITE $55
*         THIS TEST WILL DETECT A FAULTY ALU M.S.
        ALU YBUS SUBS & AB & PCUNOP &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH & MEMCONT REQB MREQ ,WRITE MBYTE &
        CONTROL ROM & RTB & ARAM EXP2 & BRAM VREG &
        TEST DIRIN NEGATIVE & CJP BMTSERR
051C: C0 61 29 1F 30 3D FB F8 17 F3 05 26

*         IF MSR IS <> THEN THERE IS AN ERROR
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT REQB & CONTROL &
        TEST MACHINE UAEQB & CJP BMOK1
051D: 40 62 29 1F 30 29 78 03 F2 53 05 1F

*         ARRIVED HERE IF TEST FAILED
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP BMTSERR
051E: 48 62 29 1F 30 09 78 07 F0 03 05 26

*         READ $55 BACK, REAL1M (ORIGINAL BYTE) -> DREG
BMOK1:  ALU YBUS PASSR & AQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM &
        ARAM REAL1M & CONT
051F: 40 63 69 5F 30 39 F8 47 F0 0E 00 00

*         ZREG AND QREG ($00FF) -> EXP2 (MASKS OUT M.S. BYTE)
        ALU REG AND & DAQ & OEY & PCUNOP & AM2904 &
```

```
        DATAPATH ,,,,,,,,,ENZ0 & MEMCONT REQB & CONTROL ROM &
        RTB & BRAM EXP2 & CONT
0520: C2 7E 69 0F 30 29 FF 87 F0 0E 00 00

*         EXP2 - REAL2M -> YBUS, JUMP TO ERROR IF -VE, WRITE BA
          CK ORIGINAL BYTE
*         THIS TEST WILL DETECT A FAULTY ALU M.S.
*         R6+1 -> MAR
        ALU YBUS SUBS & AB & PCU ,,PCUAB A2 B6 &
        AM2904 ,OECT EZ EC ES EOVR CEM & CARRYCTL COEQ1 &
        DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL ROM & RTB &
        ARAM EXP2 & BRAM REAL2M & TEST DIRIN NEGATIVE &
        CJP BMTSERR
0521: C0 61 29 9F 15 BD FE 78 17 F3 05 26

*         IF MSR IS <> THEN THERE IS AN ERROR
        ALU YBUS PASS & AB & PCUNOP & AM2904 ,OECT &
        DATAPATH & MEMCONT REQB & CONTROL &
        TEST MACHINE UAEQB & CJP BMOK2
0522: 40 62 29 1F 30 29 78 03 F2 53 05 24

*         ARRIVED HERE IF TEST FAILED
        ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & JUMP BMTSERR
0523: 48 62 29 1F 30 09 78 07 F0 03 05 26

*         READ NEXT BYTE, VREG -> DREG, R7 - R6 -> Q, LOOP BACK
           IF <> 0
BMOK2:   ALU YBUS PASSR & AQ & OEY & PCU QEU SUB PCUAB A6 B7 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL ROM & ARAM VREG &
        TESTF PCUNZ & CJP BMTRDNW
0524: 40 63 69 5E 9D F9 F8 3F F0 63 05 19

*         TEST PASSED, SET MSR = 0, RETURN, R6 - 1K -> R6
        ALU YBUS PASS & AB & PCU ,SUB PCUDA A6 B6 &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH & MEMCONT &
        CONTROL & TESTF $01 & IMMD 1024 & RTN
0525: 48 62 29 1F DD 88 78 04 10 1A 04 00

*         TEST FAILED, SET MSR <> 0, RETURN, R6 - 1K -> R6
BMTSERR:   ALU YBUS PASS & AB & PCU ,SUB PCUDA A6 B6 &
        AM2904 ,,EZ EC ES EOVR CEM & DATAPATH & MEMCONT &
        CONTROL & TESTF $03 & IMMD 1024 & RTN
0526: 48 62 29 1F DD 88 78 04 10 3A 04 00

*
* DELAY USING THE AM2910 COUNTER.
* THE CODE IS INEFFICIENT SINCE THE COUNTER
* IS ONLY 12 BITS WIDE BUT IT WILL WORK
* WHEN THE ALU, THE AM2904 OR THE PCU
* ARE FAULTY.
*
*          DLY1 -> AM2910 COUNTER
DLY2910:   ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
        MEMCONT & CONTROL & PHLC DLY1
0527: 48 62 29 1F 30 09 78 07 F0 04 05 DC
```

```
*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0528:  40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0529:  48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
052A:  40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
052B:  48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
052C:  40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
052D:  48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
052E:  40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
052F:  48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0530:  40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0531:  48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0532:  40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
```

```
       MEMCONT & CONTROL & PHLC DLY1
0533: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0534: 40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0535: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0536: 40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0537: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0538: 40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0539: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
053A: 40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
053B: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
053C: 40 62 29 1F 30 09 78 07 F0 08 00 00

*          DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
053D: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*          DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
053E: 40 62 29 1F 30 09 78 07 F0 08 00 00
```

```
*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
053F: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0540: 40 62 29 1F 30 09 78 07 F0 08 00 00

*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0541: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0542: 40 62 29 1F 30 09 78 07 F0 08 00 00

*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0543: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0544: 40 62 29 1F 30 09 78 07 F0 08 00 00

*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0545: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0546: 40 62 29 1F 30 09 78 07 F0 08 00 00

*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0547: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
0548: 40 62 29 1F 30 09 78 07 F0 08 00 00

*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
0549: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
```

```
054A: 40 62 29 1F 30 09 78 07 F0 08 00 00

*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
054B: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
054C: 40 62 29 1F 30 09 78 07 F0 08 00 00

*         DLY1 -> AM2910 COUNTER
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & PHLC DLY1
054D: 48 62 29 1F 30 09 78 07 F0 04 05 DC

*         DECREMENT COUNTER AND LOOP IF <> 0
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RFCT
054E: 40 62 29 1F 30 09 78 07 F0 08 00 00

*        RETURN
       ALU YBUS PASS & AB & PCUNOP & AM2904 & DATAPATH &
       MEMCONT & CONTROL & RTN
054F: 48 62 29 1F 30 09 78 07 F0 0A 00 00

*
*
* INITIALISE PIA'S AND ACIA
*
*
*        RECEIVE PIA CONTROL REG. ADDRESS -> R7,MAR
INITPIA:  ALU YBUS LOW & AQ & OEY & PCU ,,PCUDZ A7 B7 &
       AM2904 & DATAPATH ,,,,,,LDMAR LDD & MEMCONT REQB &
       CONTROL & IMMD RECPIACR & CONT
0550: 40 64 69 DF 7F E8 78 07 F0 0E 40 66

*        WRITE RECEIVE PIA CONTROL REG (ACCESS DDR)
*        RECEIVE PIA DATA REG ADDRESS -> MAR
       ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
       DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
       IMMD RECPIADR & CONT
0551: 40 62 29 9E 70 3C 78 07 F0 0E 40 64

*        WRITE ZERO TO RECEIVE PIA DDR (SET FOR INPUT)
*        SEND PIA CONTROL REG ADDRESS -> R6,MAR
       ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
       DATAPATH ,,,,,,LDMAR &
       MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
       IMMD SENDPIACR & CONT
0552: 40 62 29 9F 7D BC 78 07 F0 0E 40 62

*        30 -> DREG
       ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
       CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD & MEMCONT &
       CONTROL & IMMD $30-1 & CONT
```

```
0553: 42 63 69 5F 30 08 78 07 F4 0E 00 2F

*         WRITE $30 TO SEND PIA CONTROL REG (ACCESS DDR)
*         $FF -> DREG, SEND PIA DATA REG ADDRESS -> MAR
        ALU YBUS HIGH & AQ & OEY & PCU QEU ,PCUDZ & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD SENDPIADR & CONT
0554: 40 60 69 DE 70 3C 78 07 F0 0E 40 60

*         WRITE $FF TO SEND PIA DDR (SET FOR OUTPUT)
*         R6 (SEND PIA CONTROL REG ADDRESS) -> MAR, $34 -> DREG

        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD $34-1 & CONT
0555: 42 63 69 DF 4D BC 78 07 F4 0E 00 33

*         WRITE SEND PIA CONTROL REGISTER
*         R7 (RECEIVE PIA CONTROL REG ADDRESS) -> MAR, $14 -> D
          REG
        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUZA A7 B7 &
        AM2904 & CARRYCTL COEQ1 & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD $34-1 & CONT
0556: 42 63 69 DF 4F FC 78 07 F4 0E 00 33

*         WRITE RECEIVE PIA CONTROL REG.
*         SET CB2 FOR OUTPUT TO CLEAR IT
*         $14 -> DREG
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD $14-1 & CONT
0557: 42 63 69 5F 30 3C 78 07 F4 0E 00 13

*         WRITE RECEIVE PIA CONTROL REG, ACIA CONTROL REG ADDRE
          SS -> R6,MAR
*         CB2 NOW CONFIGURED FOR INPUT
        ALU YBUS PASS & AB & PCU ,,PCUDZ A6 B6 & AM2904 &
        DATAPATH ,,,,,,LDMAR &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL &
        IMMD ACIACS & CONT
0558: 40 62 29 9F 7D BC 78 07 F0 0E 40 68

*         KICK THE PIA
*         RECEIVE PIA DATA REG. ADDRESS ->MAR
        ALU YBUS PASS & AB & PCU QEU ,PCUDZ & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        IMMD RECPIADR & CONT
0559: 40 62 29 9E 70 28 78 07 F0 0E 40 64

*         3 -> DREG, READ RECEIVE PIA DATA REGISTER, R6 -> MAR
        ALU YBUS PASSR & DAQ & OEY & PCU ,,PCUZA A6 B6 &
        AM2904 & CARRYCTL & DATAPATH ,,,,,,LDMAR LDD &
        MEMCONT REQB MREQ ,,MBYTE & CONTROL & IMMD 3 & CONT
055A: 42 63 69 DF 4D B8 78 07 F0 0E 00 03
```

```
*         WRITE 3 TO ACIA (MASTER RESET), 2 -> DREG
        ALU YBUS PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL & IMMD 2 &
        CONT
055B: 42 63 69 5F 30 3C 78 07 F0 0E 00 02

*         WRITE 2 TO ACIA (BAUD RATE = 9600),
*         SYSTEM INITIALISATION MESSAGE CODE -> WREG,DREG
        ALU REG PASSR & DAQ & OEY & PCUNOP & AM2904 &
        CARRYCTL COEQ1 & DATAPATH ,,,,,,,LDD &
        MEMCONT REQB MREQ ,WRITE MBYTE & CONTROL ROM & RTB &
        BRAM WREG & IMMD STARTMESS-1 & CONT
055C: C2 7B 69 5F 30 3C F9 87 F4 0E 00 63

*         PC ->MAR
        ALU YBUS PASS & AB & PCUNOP & AM2904 &
        DATAPATH ,,,,,,LDMAR & MEMCONT REQB & CONTROL &
        JUMPX IPINIT
055D: 48 62 29 9F 30 29 78 07 F0 03 XX XX

0000 ERRORS (PMA-REV18.3)
```

Appendix 10

Amendments to Concurrent Pascal

The version of Concurrent Pascal on the Super Sixteen is virtually identical to the original implementation except for two features; the calling of Sequential Pascal programs and input/output. The full definition of the language (except for these amendments) can be found in Chapter 8 of [2.9].

10.1 Sequential Pascal Programs

A program is declared as before but the last parameter must be of type "prog_descriptor".

Where,

```
type prog_descriptor =
        record
                index, table_address : integer;
        end;
```

Before calling the program "table_address" must be set to the address in memory of an array of programs and "index" must be set to the index of the program within this array.

e.g.

```
program diagnose ( var  r: result_code;s: message_code;
                   table: prog_descriptor);
entry rcv,snd,wln,pint;
var pd: prog_descriptor;
   .
   .
pd.table_address := -8192;      "HEX E000"
pd.index := 2;
   .
   .
diagnose (result,sequence_start,pd);
   .
```

After compiling a Sequential Pascal program(s) it must be linked into an array of programs before being called as described above. This is done on the PR1ME 250.

The user types :

```
R *PLOAD
```

The system responds with :

```
$
```

The user then types in a list of compiled Sequential Pascal program file names, each terminated by carriage return. When this is done the user types "QUIT" and the program returns to the operating system. PLOAD will have produced a file called PROM containing an array of programs. Their index will correspond to the order that they have been typed in, starting from 1.

10.2 The io command

The io command is used to input or output a single character. It has three parameters:-

```
io (x,y,z)
```

Where x is type CHAR or MESSAGE_CODE

y is type IOPARAM

z is type DEVICE

```
  and type  ioparam =
            record
                  operation: iooperation;
                  faulty: boolean;
            end

and   type iooperation = (input,output);
      const send = input; receive = output;
      type device = (vdu,processor);
      type message_code = any valid message;
```

If the device is VDU then the iooperation must be INPUT or OUTPUT and a single character is input or output using the x parameter. The calling process is delayed until the operation is completed. If the device is PROCESSOR then the operation must be either SEND or RECEIVE and a single 8-bit message is either sent or received. If a message is sent then the calling process is delayed until a reply is received. If the operation is RECEIVE then the calling process is delayed until a message has been received and a reply to it has been sent. If a message is sent and a bad reply is received then "y.faulty" will be set to TRUE, otherwise it will be set to FALSE.

Appendix 11

FTOS Documentation

The system components (Processes, Monitors and Classes) have already been defined in Chapter 4. They are:-

TERMINAL	- A class to communicate with the terminal.
PROCCOMM	- A class to access the PIA and hence communicate with the neighbour processor.
NEIGHBOUR	- A monitor to post message results between the DEBUG process and the COMMUNICATOR process.
WELL_UPDATES	- A monitor holding the database and system information.
RESOURCE	- A monitor giving a process exclusive access to the terminal.
COMMUNICATOR	- A process to send and receive messages from the neighbour processor.
DEBUG	- A process to perform fault detection and analysis on the neighbouring processor.
USER	- A process to perform any user requests.

TERMINAL is a class containing several ENTRY procedures, they are:

```
PROCEDURE ENTRY READ (VAR C:  CHAR);
PROCEDURE ENTRY OCH (C:  CHAR);
```

To, respectively, input and output a single character C.

```
PROCEDURE ENTRY WRITE (TEXT:  LINE);
```

Where,

TYPE LINE = ARRAY (.1..LINELENGTH.) OF CHAR;

Outputs a complete line of text.

PROCEDURE ENTRY WRITELN (TEXT : LINE);

Performs the same operation but also outputs a new line.

PROCEDURE ENTRY LINEFEED;

Outputs a new line.

FUNCTION ENTRY READINT: INTEGER;

Reads a decimal integer from the terminal and returns its value.

PROCEDURE PRINTINT (N: UNIV INTEGER);

Outputs the integer number as defined by the 'N' parameter.

PROCCOMM is a class. It contains two routines to communicate with a processor via a PIA.

PROCEDURE ENTRY SEND_MESSAGE (VAR M:NEIGHBUFF);

Sends a message as defined by the M parameter.

Where,

```
TYPE NEIGHBUFF = RECORD
                   VALUE: MESSAGE_CODE;
                   OP: IOOPERATION;
                   FAULTY,FINISHED: BOOLEAN
                 END;
```

and TYPE IOOPERATION = (SEND,RECEIVE);

FAULTY is set to TRUE if a bad reply is received and FINISHED is set

when a reply or a message is received. VALUE is the actual message sent.

PROCEDURE ENTRY RECEIVE_MESSAGE (VAR M: NEIGHBUFF);

Receives a message.

NEIGHBOUR is a monitor which is used as a postbox between DEBUG and COMMUNICATOR.

ENTRY_TABLE: NEIGHBUFF;

Is the current message being posted and:-

TABLE_EMPTY: BOOLEAN;

Defines whether or not there is a message currently in the postbox.

PROCEDURE ENTRY GET_ENTRY (VAR MESSAGE: NEIGHBUFF);

Is used by COMMUNICATOR to get the next message. If there is no message in NEIGHBOUR then the process is delayed.

PROCEDURE ENTRY PUT_ENTRY (MESSAGE: NEIGHBUFF);

Overwrites ENTRY_TABLE with MESSAGE. This is used by COMMUNICATOR to put a processed message back into the "post box".

PROCEDURE ENTRY ADD_ENTRY (MESSAGE: NEIGHBUFF);

Is used by DEBUG to put a message request into ENTRY_TABLE. A continue operation is performed in case COMMUNICATOR has been delayed by GET_ENTRY.

PROCEDURE ENTRY ABORT_ENTRY;

Deletes a message request. Used by COMMUNICATOR after sending a

message.

PROCEDURE ENTRY ENTRY_READY (VAR MSG: NEIGHBUFF);

Is used by DEBUG to take a copy of the message request. DEBUG would then examine the FINISHED field to see if the message request had been processed.

The WELL_UPDATES monitor is used to communicate between DEBUG and USER. It contains three basic types of data:-

1) The well-field information:-

WELL_DATA : ARRAY (.1..WELL_ENTRIES.) OF WELL_RECORD;

The type of WELL_RECORD is irrelevant for the moment since WELL_UPDATES passes across the entire structure. Altering the structure of the data base would not affect WELL_UPDATES.

2) The system information:-

MAIN: BOOLEAN;

NFAULTY: BOOLEAN;

Which, respectively, indicates whether the processor is in main/backup mode and whether its neighbour processor is faulty.

3) Any updates made to the database in order to inform the backup processor:-

MESSAGE_QUEUE: ARRAY (.1..WELL_ENTRIES.) OF INTEGER;

MESSAGE_HEAD,MESSAGE_TAIL: INTEGER;

Is a cyclic queue of index numbers of elements of WELL_DATA that

have been updated.

The operations on this data are:-

FUNCTION ENTRY ENTRY_PRESENT: BOOLEAN;

Tells whether MESSAGE_QUEUE is empty or not.

PROCEDURE ENTRY SET_MAIN;

PROCEDURE ENTRY SET_BACKUP;

FUNCTION ENTRY MAIN_MODE: BOOLEAN;

Respectively, sets the processor to main mode, sets the processor to backup mode and indicates whether the processor is in main mode.

FUNCTION ENTRY NEIGHBOUR_UP: BOOLEAN;

PROCEDURE ENTRY SET_FAULTY;

PROCEDURE ENTRY CLEAR_FAULTY;

Respectively, indicates whether the neighbour processor is operational, sets the neighbour processor mode to faulty and sets the neighbour processor mode to operational.

PROCEDURE ENTRY CLEAR_QUEUE;

Initialises the cyclic well update queue, this is used for a re-instate operation.

PROCEDURE ENTRY PUT_MESSAGE (M: INTEGER);

FUNCTION ENTRY GET_MESSAGE: INTEGER;

Respectively puts and gets an entry to or from the cyclic well update queue.

PROCEDURE ENTRY GET_ENTRY (VAR WELL_ENTRY: WELL_RECORD;

INDEX: INTEGER);

PROCEDURE ENTRY PUT_ENTRY (VAR WELL_ENTRY: WELL_RECORD;

INDEX: INTEGER);

Respectively, gets or puts an entry into the well field database. The INDEX parameter gives the record index and the WELL_ENTRY parameter is the well-record contents.

RESOURCE is a monitor used to obtain exclusive access to the terminal. It contains two procedures:-

PROCEDURE ENTRY ACCESS;

Obtains access to the terminal. If it is already in use then the calling process is delayed.

PROCEDURE ENTRY RELEASE;

Relinquishes access to the terminal. If another process has been delayed whilst waiting for the terminal then it is continued.

COMMUNICATOR is a very simple process consisting of a continuous loop (CYCLE) which merely gets a message request from NEIGHBOUR, either sends or receives it as required and then puts the updated message request back in NEIGHBOUR.

The DEBUG process is responsible for continuously sending and receiving message requests to NEIGHBOUR. These are either periodic "I'm still alive" messages or backup updates that have not yet been completed. If the processor is in main mode then these updates are sent. If the processor is in backup mode then they are received. If an error is detected in the message transmission then a Sequential Pascal program called DIAGNOSE is called to analyse the

fault. No more message passing (except for the fault analysis) will take place until the Faulty Processor is certified operational. This is done by the use of the Re-instate command (called from the USER process).

DEBUG has a function to receive a message:-

```
FUNCTION RECEIVE_MESSAGE (EXPECTED_MESSAGES: UNIV
                    SET_OF_MESSAGE): MESSAGE_CODE;
```

The EXPECTED_MESSAGES parameter is a set of valid messages that can be received. Up to a limit of three tries, any message that is not in the set is ignored. The procedure puts a message request in NEIGHBOUR and waits for a certain time period. If no message is received within this time limit then an error has occured.

```
PROCEDURE SEND_MESSAGE (M: UNIV MESSAGE_CODE);
```

Sends a message and waits for a reply. If this is not received within a certain time limit then there is an error.

There are two procedures for sending and receiving well updates to a backup processor or a main processor respectively.

```
PROCEDURE SEND_WELL_UPDATES;
PROCEDURE RECEIVE_WELL_INFO;
```

The process for sending a message consists of sending a special start of transfer message followed by the entire data structure (8 bits at a time). For the sake of modularity and ease of altering the user functions it would be better if the entire structure was sent as one typeless block. However, this would allow no error

checking. If the neighbour processor were to fail during a transfer sequence then any diagnosis messages would be treated as part of the update sequence. Consequently, they would be lost. It is for this reason that each individual field of WELL_INFO is sent seperately. This allows greater error checking. For example, when the well name is sent a check can be made at each stage to determine whether the message received is a valid character.

The main body of DEBUG consists of an infinite loop to send and receive a message to the neighbour processor. If an error is detected then the program DIAGNOSE is called. This program evaluates the messages sent by the Faulty Processor. The method of doing this is described in Chapter 4. DIAGNOSE requires prefix procedures to communicate with the terminal and the Faulty Processor.

```
FUNCTION RCV (M: UNIV SET_OF_MESSAGE): MESSAGE_CODE;
PROCEDURE SND (M: UNIV MESSAGE_CODE);
```

Respectively, receives and sends a message by calling RECEIVE_MESSAGE and SEND_MESSAGE which were described earlier.

```
PROCEDURE WLN (TEXT: LINE);
```

Outputs a message to the terminal, this is used to inform the user that a fault has occured and what it has been diagnosed to be.

The USER process consists of an infinite loop calling the Sequential Pascal program COMMAND_INTERPRETER. This executes a command typed in at the terminal. It requires several prefix procedures:-

```
PROCEDURE OUTCH (C: CHAR);
PROCEDURE WRITE (TEXT: LINE);
PROCEDURE WRITELN (TEXT: LINE);
PROCEDURE LINEFEED;
PROCEDURE PRINTINT (N: UNIV INTEGER);
FUNCTION READINT: INTEGER;
```

These procedures call their counterparts in the TERMINAL class to perform input and output to the terminal. In addition, however, they call the RESOURCE monitor to acquire exclusive access to the terminal and ensure that a complete line of text is output without interruptions.

The other prefix procedures are:-

```
FUNCTION BACKUP_MODE: BOOLEAN;
```

Indicates whether the processor is in backup mode.

```
FUNCTION NEIGHBOUR_UP: BOOLEAN;
```

Indicates whether the neighbour processor is operational.

```
PROCEDURE CLEAR_FAULTY;
```

Marks the neighbour processor as being operational.

```
PROCEDURE CLEAR_QUEUE;
```

Flushes the backup update message queue in NEIGHBOUR (used during a Re-instate operation).

```
FUNCTION WELL_UPDATE_PRESENT: BOOLEAN;
```

Is no longer used.

```
PROCEDURE UPDATE_BACKUP (VALUE: INTEGER);
```

Instructs DEBUG that an update has been made and the backup processor, if operational, should also be updated.

```
PROCEDURE GET_WELL_ENTRY (VAR WELL_ENTRY: WELL_RECORD;
                          INDEX: INTEGER);
PROCEDURE PUT_WELL_ENTRY (VAR WELL_ENTRY: WELL_RECORD;
                          INDEX: INTEGER);
```

Calls the corresponding GET_ENTRY and PUT_ENTRY procedures in the WELL_UPDATES monitor.

The COMMAND_INTERPRETER program consists of inputting a character. If it corresponds to a valid command then it is executed (in a CASE statement), otherwise an error is detected. The valid commands are:-

'W' - Display Well Data.

'C' - Change Well Data.

'D' - Delete Well Entry.

'P' - Display Processor Status.

'I' - Insert New Well.

'R' - Re-instate Neighbour Processor.

In the case of the 'C', 'D' and 'I' commands an error is detected if the processor is in backup mode.

The 'W' command gets every well entry (using GET_WELL_ENTRY) and, if an entry is marked as being in use, displays it at the terminal.

The 'C' command asks the user to input the well name, attempts

to match it with an existing entry (using GET_WELL_ENTRY) and performs the update required. If the entry does not exist then an error message is output.

The 'D' command attempts to match the well name, if it exists its USED field is set to FALSE. If it doesn't exist an error message is output.

The 'P' command extracts the system information (using BACKUP_MODE and NEIGHBOUR_UP) and outputs it to the terminal.

The 'I' command searches for the first unused entry in the database (using GET_WELL_ENTRY). It then inserts a new record in that position as specified by the user. If the database is full or the well name already exists then an error message is output.

The 'R' command resets the system data. This is done by marking the neighbour processor as operational (using CLEAR_QUEUE and CLEAR_FAULTY). It also marks the entire database as being updated to force DEBUG to copy the database onto the backup processor.

Appendix 12

FTOS Source Listings

```
0001 (NUMBER,CHECK)
0002 "CONCURRENT PASCAL SUPER SIXTEEN PROCESSOR.
0003  REAL TIME OPERATING SYSTEM AND FAULTY PROCESSOR
0004  DIAGNOSIS"
0005 CONST NL = '(:10:)'; CR = '(:13:)';MESSAGE_END ='£';
0006 CONST LINELENGTH = 80;BUFLEN = 5;IDLENGTH = 14;
0007 CONST DIAG_ADDRESS = 24576;      "HEX A000"
0008 CONST CMND_ADDRESS = 24576;      "HEX A000"
0009 CONST DIAG_INDEX = 1;
0010 CONST CMND_INDEX = 2;
0011 TYPE IDENTIFIER = ARRAY (.1..IDLENGTH.) OF CHAR;
0012 TYPE LINE = ARRAY (.1..LINELENGTH.) OF CHAR;
0013 TYPE DEVICE = (VDU,PROCESSOR);
0014 TYPE IOOPERATION = (INPUT,OUTPUT);
0015 CONST SEND = INPUT; RECEIVE = OUTPUT;
0016 TYPE IOPARAM = RECORD OPERATION : IOOPERATION ; FAULTY : BOOLEAN END;
0017 TYPE PROG_DESCRIPTOR = RECORD
0018                              INDEX,TABLE_ADDRESS : INTEGER
0019                          END;
0020 TYPE RESULT_CODE = (SOFTWARE_ERROR,ALU_FAULTY,PCU_FAULTY,
0021                     AM2904_FAULTY,TREG_FAULTY,TTLATCH_FAULTY,
0022                     TESTTREE_FAULTY,ZREGMS_FAULTY,ZREGLS_FAULTY,
0023                     Z0REGLS_FAULTY,ZIREG_FAULTY,MAP_FAULTY,
0024                     PCUTRAN_FAULTY,OTHER);
0025 TYPE MESSAGE_CODE = (FIRSTMESS,
0026             TFMESS,             "TIMER FAILURE MESSAGE"
0027             IRMESS,             "INTERRUPT MESSAGE"
0028             JMTSUCMESS,         "JOINT MEMORY TEST SUCCESSFULL"
0029             JMTUNMESS,          "JOINT MEMORY TEST UNSUCCESSFULL"
0030             ALUMTSUCMESS,       "ALU MEMORY TEST SUCCESSFULL"
0031             ALUMTUNMESS,        "ALU MEMORY TEST UNSUCCESSFULL"
0032             PCUMTSUCMESS,       "PCU MEMORY TEST SUCCESSFULL"
0033             PCUMTUNMESS,        "PCU MEMORY TEST UNSUCCESSFULL"
0034             PCUALUSUCMESS,      "PCU/ALU TEST SUCCESSFULL"
0035             PCUALUUNMESS,       "PCU/ALU TEST UNSUCCESSFULL"
0036             ALUSUCMESS,         "ALU TEST SUCCESSFULL"
0037             ALUUNMESS,          "ALU TEST UNSUCCESSFULL"
0038             BMTSUCMESS,         "BYTE MEMORY TEST SUCCESSFULL"
0039             BMTUNMESS,          "BYTE MEMORY TEST UNSUCCESSFULL"
0040             MAPZIMESS,          "MAP/ZI TEST SUCCESSFULL"
0041             INVMESS,            "INVALID INSTRUCTION MESSAGE"
0042             SYSERRMESS,         "SYSTEM ERROR MESSAGE"
0043             INTMESS,            "PERIODIC INTERRUPT MESSAGE"
0044             IEMESS,             "INTERRUPT ERROR MESSAGE"
0045             MAPERRMESS,         "MAP DECODER ERROR SUSPECTED"
0046             WELLMESS,           "WELL UPDATE MESSAGE"
0047             LASTMESS,
0048             A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,A33,A34,A35,A36,A37,
0049             A38,A39,A40,A41,A42,A43,A44,A45,A46,A47,A48,A49,A50,A51,A52,A53,
0050             A54,A55,A56,A57,A58,A59,A60,A61,A62,A63,A64,A65,A66,A67,A68,A69,
0051             A70,A71,A72,A73,A74,A75,A76,A77,A78,A79,A00,A81,A82,A83,A84,
0052             A85,A86,A87);
0053 TYPE SET_OF_MESSAGE = SET OF MESSAGE_CODE;
```

```
0054
0055 CONST NBUFFSIZE = 15;      "SIZE OF 'NEIGHBOUR' BUFFER"
0056 TYPE NEIGHBUFF = RECORD
0057                      VALUE : MESSAGE_CODE;
0058                      OP : IOOPERATION;
0059                      FAULTY,FINISHED : BOOLEAN
0060                    END;
0061
0062 CONST WELL_ENTRIES = 15;
0063 TYPE WELL_RECORD = RECORD
0064                           NAME : IDENTIFIER;
0065                           SIZE : INTEGER;
0066                           LOCATION : IDENTIFIER;
0067                           USED : BOOLEAN
0068                         END;
0069
0070 TYPE TERMINAL = CLASS
0071
0072 VAR NEWLINE,CARRIAGE_RETURN,SPACE : CHAR;
0073     LOOKUP : ARRAY (.0..15.) OF CHAR;
0074
0075 PROCEDURE INCH (VAR C : CHAR);
0076 "INPUT A CHARACTER FROM THE VDU"
0077 VAR PARAM : IOPARAM;
0078 BEGIN
0079       PARAM.OPERATION := INPUT;
0080       IO (C,PARAM,VDU)
0081 END;
0082
0083 PROCEDURE ENTRY READ (VAR C :CHAR);
0084 BEGIN INCH (C) END;
0085
0086 PROCEDURE OUTCH (C : CHAR);
0087 "OUTPUT A CHARACTER TO THE VDU"
0088 VAR PARAM : IOPARAM;
0089     CH : CHAR;
0090 BEGIN
0091       CH := C;
0092       PARAM.OPERATION := OUTPUT;
0093       IO (CH,PARAM,VDU)
0094 END;
0095
0096 PROCEDURE ENTRY OCH (C : CHAR);
0097 BEGIN OUTCH (C) END;
0098
0099 PROCEDURE OUTTEXT (TEXT : LINE);
0100 VAR I : INTEGER; C : CHAR;
0101 BEGIN
0102       I := 1;
0103       WHILE TEXT (.I.) <> MESSAGE_END
0104       DO
0105          BEGIN
0106               C := TEXT (.I.);
0107               OUTCH (C);
```

```
0108                 I := SUCC (I)
0109             END
0110 END;
0111
0112
0113 PROCEDURE OUTLINE;
0114 BEGIN
0115       OUTCH (CARRIAGE RETURN);
0116       OUTCH (NEWLINE)
0117 END;
0118
0119 PROCEDURE ENTRY WRITE (TEXT : LINE);
0120 BEGIN OUTTEXT (TEXT) END;
0121
0122 PROCEDURE ENTRY WRITELN (TEXT : LINE);
0123 BEGIN
0124       OUTTEXT (TEXT);
0125       OUTLINE
0126 END;
0127
0128 PROCEDURE ENTRY LINEFEED;
0129 BEGIN OUTLINE END;
0130
0131 FUNCTION ENTRY READINT : INTEGER;
0132 VAR DIGIT,VALUE : INTEGER;
0133     CONTINUE,NEG : BOOLEAN;
0134     C : CHAR;
0135 BEGIN
0136   REPEAT
0137       DIGIT := 0;
0138       VALUE := 0;
0139       INCH (C);
0140       NEG := FALSE;
0141       IF C = '+' THEN INCH (C)
0142       ELSE IF C = '-' THEN BEGIN NEG := TRUE; INCH (C) END;
0143       WHILE (ORD(C))= ORD('0')) AND (ORD(C) (= ORD('9'))
0144       DO
0145         BEGIN
0146             DIGIT := ORD(C) - ORD('0');
0147             VALUE := 10*VALUE + DIGIT;
0148             INCH (C)
0149         END;
0150       IF C () CR THEN BEGIN OUTCH ('?');OUTCH (CR); OUTCH (NL) END;
0151   UNTIL C = CR;
0152   OUTCH (NEWLINE);
0153   IF NEG THEN READINT := -VALUE ELSE READINT := VALUE
0154 END;
0155
0156 PROCEDURE ENTRY PRINTINT (N : UNIV INTEGER);
0157 VAR DIVISOR,SUBTRACT,VALUE,BASE,NUMBER : INTEGER;
0158     NOT LEADING : BOOLEAN;
0159 BEGIN
0160     BASE := 10;SUBTRACT := 0;
0161     DIVISOR := 10000;VALUE := 0;
```

```
0162     OUTCH (SPACE);
0163     NOT_LEADING := FALSE;
0164     NUMBER := N;
0165     IF N < 0 THEN
0166     BEGIN
0167          IF N = (-32767-1) " -32768 CAUSES OVERFLOW WHEN NEGATED"
0168          THEN
0169            BEGIN
0170              OUTCH ('3');
0171              OUTCH ('2');
0172              OUTCH ('7');
0173              OUTCH ('6');
0174              OUTCH ('3');
0175              DIVISOR := 0;   " STOPS WHILE LOOP "
0176            END
0177          ELSE
0178            BEGIN
0179              NUMBER := -N;
0180              OUTCH ('-')
0181            END;
0182    END;
0183    IF N = 0 THEN OUTCH ('0')
0184    ELSE
0185     WHILE DIVISOR <> 0 DO
0186     BEGIN
0187        VALUE := NUMBER DIV DIVISOR - SUBTRACT*BASE;
0188        IF (VALUE <> 0) OR (NOT_LEADING) THEN
0189        BEGIN
0190             NOT_LEADING := TRUE;
0191             OUTCH (LOOKUP (.VALUE.))
0192        END;
0193        DIVISOR := DIVISOR DIV BASE;
0194        SUBTRACT := SUBTRACT*BASE + VALUE
0195     END
0196 END;
0197 BEGIN NEWLINE := NL; CARRIAGE_RETURN := CR ;
0198       SPACE := ' ';
0199       LOOKUP := '0123456789ABCDEF'
0200 END;
0201
0202 "COMMUNICATES WITH THE NEIGHBOUR PROCESSOR"
0203 "FACILITIES FOR SENDING AND RECEIVING A SINGLE MESSAGE"
0204 TYPE PROCCOMM = CLASS
0205
0206 VAR PRM1,PRM2 : IOPARAM;
0207
0208 PROCEDURE ENTRY SEND_MESSAGE (VAR M :NEIGHBUFF);
0209 BEGIN
0210       IO (M.VALUE,PRM1,PROCESSOR);
0211       M.FINISHED := TRUE;
0212       M.FAULTY := PRM1.FAULTY
0213 END;
0214
0215 PROCEDURE ENTRY RECEIVE_MESSAGE (VAR M :NEIGHBUFF);
```

```
0216
0217 VAR I,A : INTEGER;
0218
0219 BEGIN
0220       IO (M.VALUE,PRM2,PROCESSOR);
0221       M.FINISHED := TRUE;
0222       M.FAULTY := PRM1.FAULTY;
0223       IF M.VALUE = INTMESS THEN
0224       FOR I := 1 TO 150 DO A := A
0225 END;
0226
0227 BEGIN
0228      PRM1.OPERATION := SEND;
0229      PRM2.OPERATION := RECEIVE
0230 END;
0231
0232
0233 TYPE NEIGHBOUR = MONITOR
0234
0235 VAR ENTRY_TABLE : NEIGHBUFF;
0236     Q : QUEUE;
0237     TABLE_EMPTY : BOOLEAN;
0238
0239 PROCEDURE ENTRY GET_ENTRY (VAR MESSAGE : NEIGHBUFF);
0240 "GET THE NEXT ENTRY FROM THE ENTRY TABLE
0241 WITHOUT REMOVING IT"
0242 BEGIN
0243      IF TABLE_EMPTY THEN DELAY (Q);
0244      MESSAGE := ENTRY_TABLE
0245 END;
0246
0247 PROCEDURE ENTRY PUT_ENTRY (MESSAGE : NEIGHBUFF);
0248 "OVERWRITES THE FIRST ENTRY OF THE TABLE WITH
0249  THE MESSAGE PARAMETER"
0250 BEGIN
0251      ENTRY_TABLE  := MESSAGE;
0252      TABLE_EMPTY := TRUE
0253 END;
0254
0255 PROCEDURE ENTRY ADD_ENTRY (MESSAGE:NEIGHBUFF);
0256 "ADDS AN ENTRY TO THE END OF THE TABLE"
0257 BEGIN
0258      ENTRY_TABLE  := MESSAGE;
0259      ENTRY_TABLE .FAULTY := TRUE;
0260      ENTRY_TABLE.FINISHED := FALSE;
0261      TABLE_EMPTY := FALSE;
0262      CONTINUE (Q)
0263 END;
0264
0265 PROCEDURE ENTRY ABORT_ENTRY;
0266 "DELETES AN ENTRY FROM THE TABLE"
0267 BEGIN TABLE_EMPTY := TRUE END;
0268
0269 PROCEDURE ENTRY ENTRY_READY (VAR MSG : NEIGHBUFF);
```

```
0270
0271 BEGIN
0272       MSG := ENTRY TABLE
0273 END;
0274
0275 BEGIN
0276       TABLE_EMPTY := TRUE
0277 END;
0278
0279
0280 TYPE WELL_UPDATES = MONITOR
0281
0282 VAR MAIN : BOOLEAN;
0283     NFAULTY : BOOLEAN;
0284     MESSAGE_QUEUE : ARRAY (.1..WELL_ENTRIES.) OF INTEGER;
0285     MESSAGE_HEAD,MESSAGE_TAIL : INTEGER;
0286     WELL_DATA : ARRAY (.1..WELL_ENTRIES.) OF WELL_RECORD;
0287     LOOP : INTEGER;
0288     DIAGNOSED : BOOLEAN;
0289
0290 FUNCTION ENTRY PRESENT : BOOLEAN;
0291 BEGIN PRESENT := (MESSAGE_HEAD <> MESSAGE_TAIL) AND (NOT NFAULTY) END;
0292
0293 FUNCTION ENTRY NOT_DIAGNOSED : BOOLEAN;
0294 BEGIN NOT_DIAGNOSED := NOT DIAGNOSED END;
0295
0296 PROCEDURE ENTRY SET_DIAGNOSED (RESULT : BOOLEAN);
0297 BEGIN DIAGNOSED := RESULT END;
0298
0299 PROCEDURE ENTRY SET_MAIN;
0300 "SETS THE PROCESSOR MODE TO MAIN"
0301 BEGIN MAIN := TRUE END;
0302
0303 PROCEDURE ENTRY SET_BACKUP;
0304 "SETS PROCESSOR MODE TO BACKUP"
0305 BEGIN MAIN := FALSE END;
0306
0307 FUNCTION ENTRY MAIN_MODE : BOOLEAN;
0308 "RETURNS THE CURRENT PROCESSOR STATUS (MAIN = TRUE, BACKUP = FALSE)"
0309 BEGIN MAIN_MODE := MAIN END;
0310
0311 FUNCTION ENTRY NEIGHBOUR_UP : BOOLEAN;
0312 BEGIN NEIGHBOUR_UP := NOT NFAULTY END;
0313
0314 PROCEDURE ENTRY SET_FAULTY;
0315 BEGIN NFAULTY := TRUE END;
0316
0317 PROCEDURE ENTRY CLEAR_FAULTY;
0318 BEGIN NFAULTY := FALSE END;
0319
0320 PROCEDURE ENTRY CLEAR_QUEUE;
0321 BEGIN
0322       MESSAGE_HEAD := 1;
0323       MESSAGE_TAIL := 1
```

```
0324 END;
0325
0326 PROCEDURE ENTRY PUT_MESSAGE (M : INTEGER);
0327 BEGIN
0328       MESSAGE_QUEUE (.MESSAGE_TAIL.) := M;
0329       MESSAGE_TAIL := (MESSAGE_TAIL MOD WELL_ENTRIES) + 1
0330 END;
0331
0332 FUNCTION ENTRY GET_MESSAGE : INTEGER;
0333
0334 VAR PTR : INTEGER;
0335
0336 BEGIN
0337       PTR := MESSAGE_HEAD;
0338       MESSAGE_HEAD := (MESSAGE_HEAD MOD WELL_ENTRIES) + 1;
0339       GET_MESSAGE := MESSAGE_QUEUE (.PTR.)
0340 END;
0341
0342 PROCEDURE ENTRY GET_ENTRY (VAR WELL_ENTRY : WELL_RECORD; INDEX : INTEGER);
0343 BEGIN WELL_ENTRY := WELL_DATA (.INDEX.) END;
0344
0345 PROCEDURE ENTRY PUT_ENTRY (VAR WELL_ENTRY : WELL_RECORD; INDEX : INTEGER);
0346 BEGIN WELL_DATA (.INDEX.) := WELL_ENTRY END;
0347
0348 BEGIN
0349       DIAGNOSED := FALSE;
0350       NFAULTY := FALSE;
0351       MESSAGE_HEAD := 1;
0352       MESSAGE_TAIL := 1;
0353       FOR LOOP := 1 TO WELL_ENTRIES
0354       DO WELL_DATA (.LOOP.).USED := FALSE
0355 END;
0356
0357 TYPE RESOURCE = MONITOR
0358
0359 VAR Q : QUEUE;
0360     FREE : BOOLEAN;
0361
0362 PROCEDURE ENTRY ACCESS;
0363 BEGIN
0364       IF NOT FREE THEN DELAY (Q);
0365       FREE := FALSE
0366 END;
0367
0368 PROCEDURE ENTRY RELEASE;
0369
0370 BEGIN
0371       IF NOT EMPTY (Q)
0372       THEN CONTINUE (Q)
0373       ELSE FREE := TRUE
0374 END;
0375
0376 BEGIN
0377       FREE := TRUE
```

```
0378 END;
0379
0380
0381 TYPE COMMUNICATOR = PROCESS (MESSAGE_BUFFER : NEIGHBOUR);
0382 "NOTE THAT IF THE NEIGHBOUR PROCESSOR GOES DOWN AND STOPS
0383  SENDING MESSAGES THEN THIS PROCESS WILL ALSO GO DEAD BUT
0384  THE REST OF THE O.S. WILL CONTINUE TO FUNCTION"
0385
0386 VAR MESSAGE : NEIGHBUFF;
0387     PRCM : PROCCOMM;
0388
0389 BEGIN
0390       INIT PRCM;
0391       WITH PRCM,MESSAGE_BUFFER
0392       DO
0393         CYCLE
0394               GET ENTRY(MESSAGE);
0395               IF MESSAGE.OP = SEND
0396               THEN
0397                 BEGIN
0398                    SEND_MESSAGE (MESSAGE);
0399                    ABORT_ENTRY
0400                 END
0401               ELSE RECEIVE_MESSAGE (MESSAGE);
0402               PUT_ENTRY (MESSAGE)
0403          END
0404 END;
0405
0406
0407
0408 TYPE DEBUG = PROCESS (TERMINAL_RESOURCE :RESOURCE; NGH : NEIGHBOUR; WUP : WELL_UPDATES);
0409 "PROGRAM DATA SPACE = "+1000
0410
0411 VAR TERM : TERMINAL;
0412     ERROR_MESSAGES : SET_OF_MESSAGE;
0413     M : NEIGHBUFF;
0414     MESSAGE, SEQUENCE_START : MESSAGE_CODE;
0415     RESULT : RESULT_CODE;
0416     PD : PROG_DESCRIPTOR;
0417     LOOP : INTEGER;
0418     CH : CHAR;
0419     B : BOOLEAN;
0420     JUST_ONLINE : BOOLEAN;
0421
0422 PROGRAM DIAGNOSE (VAR R : RESULT_CODE; S : MESSAGE_CODE;TABLE : PROG_DESCRIPTOR);
0423 ENTRY RCV,SND,WLN,PINT;
0424
0425 PROCEDURE DLY;
0426 VAR I,A : INTEGER;
0427 BEGIN FOR I := 1 TO 60 DO A := A END;
0428
0429 FUNCTION RECEIVE_MESSAGE (EXPECTED_MESSAGES : UNIV SET_OF_MESSAGE) : MESSAGE_CODE;
0430
0431 CONST NUMTRIES = 30; NUMBAD = 3;
```

```
0432
0433 VAR M : NEIGHBUFF;
0434     BAD_MESSAGE_COUNT,NOT_RECEIVED_COUNT : INTEGER;
0435     MESSAGE_RECEIVED : MESSAGE_CODE;
0436
0437 BEGIN
0438      M.VALUE := LASTMESS;
0439      MESSAGE_RECEIVED := LASTMESS;
0440      IF FIRSTMESS IN EXPECTED_MESSAGES
0441      THEN BAD_MESSAGE_COUNT := NUMBAD -1
0442      ELSE BAD_MESSAGE_COUNT := 0;
0443      WHILE ( NOT((MESSAGE_RECEIVED IN EXPECTED_MESSAGES)))
0444        AND (BAD_MESSAGE_COUNT < NUMBAD)
0445      DO
0446       BEGIN
0447            M.OP := RECEIVE;
0448            NGH.ADD_ENTRY (M);
0449            NOT_RECEIVED_COUNT := 0;
0450            NGH.ENTRY_READY (M);
0451            WHILE (NOT M.FINISHED) AND (NOT_RECEIVED_COUNT < NUMTRIES) DO
0452            BEGIN
0453                 DLY;
0454                 NOT_RECEIVED_COUNT := SUCC (NOT_RECEIVED_COUNT);
0455                 NGH.ENTRY_READY (M);
0456            END;
0457            IF M.VALUE > A87 THEN MESSAGE_RECEIVED := LASTMESS
0458                             ELSE MESSAGE_RECEIVED := M.VALUE;
0459            IF NOT M.FINISHED THEN NGH.ABORT_ENTRY;
0460            BAD_MESSAGE_COUNT := SUCC (BAD_MESSAGE_COUNT)
0461       END;
0462       IF M.VALUE <> INTMESS THEN
0463       BEGIN
0464            TERMINAL_RESOURCE.ACCESS;
0465       TERM.WRITE ('MSG RECEIVED£ ');TERM.PRINTINT (M.VALUE);TERM.LINEFEED;
0466            TERMINAL_RESOURCE.RELEASE;
0467       END;
0468        IF ( M.FINISHED) AND ((MESSAGE_RECEIVED IN (EXPECTED_MESSAGES - ERROR_MESSAGES))
0469                          OR (FIRSTMESS IN EXPECTED_MESSAGES))
0470                                AND (WUP.NEIGHBOUR_UP)
0471        THEN
0472        ELSE WUP.SET_FAULTY;
0473       SEQUENCE_START := M.VALUE;
0474       RECEIVE_MESSAGE := SEQUENCE_START
0475 END;
0476
0477 PROCEDURE SEND_MESSAGE (M : UNIV MESSAGE_CODE);
0478
0479 CONST NUMSEND = 30;
0480
0481 VAR MESSAGE : NEIGHBUFF;
0482     NOT_SENT_COUNT,LOOP : INTEGER;
0483
0484 BEGIN
0485      IF M <> INTMESS THEN
```

```
0486      BEGIN
0487      TERMINAL_RESOURCE.ACCESS;
0488      TERM.WRITE ('SEND£ '); TERM.PRINTINT (M);TERM.LINEFEED;
0489      TERMINAL_RESOURCE.RELEASE;
0490      END;
0491      NOT_SENT_COUNT := 0;
0492      MESSAGE.OP := SEND;
0493      MESSAGE.VALUE := M;
0494      NGH.ADD_ENTRY (MESSAGE);
0495      NGH.ENTRY_READY (MESSAGE);
0496      WHILE (NOT MESSAGE.FINISHED) AND (NOT_SENT_COUNT (NUMSEND) DO
0497      BEGIN
0498           DLY;
0499           NOT_SENT_COUNT := SUCC (NOT_SENT_COUNT);
0500           NGH.ENTRY_READY (MESSAGE)
0501      END;
0502      IF (WUP.NEIGHBOUR_UP) AND (NOT MESSAGE.FAULTY)
0503      THEN
0504      ELSE WUP.SET_FAULTY
0505 END;
0506
0507 PROCEDURE CONVMTOCH (M : UNIV CHAR);
0508 BEGIN CH := M END;
0509
0510 PROCEDURE CONVMTOINT (M : UNIV INTEGER);
0511 BEGIN LOOP := M END;
0512
0513 PROCEDURE CONVMTOBOOL (M : UNIV BOOLEAN);
0514 BEGIN B := M END;
0515
0516 PROCEDURE SEND_WELL_UPDATES;
0517
0518 VAR INDEX : INTEGER;
0519     WELL_INFO : WELL_RECORD;
0520
0521 BEGIN
0522      WITH WUP DO
0523      BEGIN
0524           SEND_MESSAGE (WELLMESS);
0525           IF NEIGHBOUR_UP THEN
0526           BEGIN
0527                INDEX := GET_MESSAGE;
0528                GET_ENTRY (WELL_INFO, INDEX);
0529                SEND_MESSAGE (INDEX);
0530                LOOP := 1;
0531                WHILE (LOOP (= IDLENGTH) AND (NEIGHBOUR_UP)
0532                DO
0533                BEGIN
0534                     CH := WELL_INFO.NAME (.LOOP.);
0535                     SEND_MESSAGE (CH);
0536                     LOOP := SUCC (LOOP)
0537                END;
0538                IF NEIGHBOUR_UP THEN SEND_MESSAGE (WELL_INFO.SIZE DIV 256);
0539                IF NEIGHBOUR_UP THEN SEND_MESSAGE (WELLMESS);
```

```
0540                IF NEIGHBOUR_UP THEN SEND_MESSAGE (WELL_INFO.SIZE);
0541                LOOP := 1;
0542                WHILE (LOOP <= IDLENGTH) AND NEIGHBOUR_UP
0543                DO
0544                BEGIN
0545                     CH := WELL_INFO.LOCATION (.LOOP.);
0546                     SEND_MESSAGE (CH);
0547                     LOOP := SUCC (LOOP)
0548                END;
0549                IF NEIGHBOUR_UP THEN SEND_MESSAGE (WELL_INFO.USED)
0550           END
0551      END
0552 END "SEND WELL UPDATES";
0553
0554 PROCEDURE RECEIVE_WELL_INFO;
0555
0556 VAR M : MESSAGE_CODE; SIZE,INDEX : INTEGER;
0557     WELL_INFO : WELL_RECORD;
0558
0559 BEGIN
0560      WITH WUP DO
0561      BEGIN
0562           M := RECEIVE_MESSAGE ((.FIRSTMESS.));
0563           CONVMTOINT (M);
0564           INDEX := LOOP;
0565           LOOP := 1;
0566           WHILE (LOOP <= IDLENGTH) AND (NEIGHBOUR_UP) DO
0567           BEGIN
0568                M := RECEIVE_MESSAGE ((.FIRSTMESS.));
0569                CONVMTOCH (M);
0570                WELL_INFO.NAME (.LOOP.) := CH;
0571                LOOP := SUCC (LOOP);
0572                IF CH <> ' ' THEN WUP.SET_FAULTY
0573           END;
0574           IF NEIGHBOUR_UP THEN M := RECEIVE_MESSAGE ((.FIRSTMESS.));
0575           CONVMTOINT (M);
0576           SIZE := LOOP * 256;
0577           IF NEIGHBOUR_UP THEN M := RECEIVE_MESSAGE ((.WELLMESS.));
0578           IF NEIGHBOUR_UP THEN M := RECEIVE_MESSAGE ((.FIRSTMESS.));
0579           CONVMTOINT (M);
0580           WELL_INFO.SIZE := SIZE + LOOP;
0581           LOOP := 1;
0582           WHILE (LOOP <= IDLENGTH) AND (NEIGHBOUR_UP) DO
0583           BEGIN
0584                M := RECEIVE_MESSAGE ((.FIRSTMESS.));
0585                CONVMTOCH (M);
0586                WELL_INFO.LOCATION (.LOOP.) := CH;
0587                LOOP := SUCC (LOOP);
0588                IF CH <> ' ' THEN WUP.SET_FAULTY
0589           END;
0590           IF NEIGHBOUR_UP
0591           THEN
0592           BEGIN
0593                M := RECEIVE_MESSAGE ((.FIRSTMESS.));
```

```
0594                CONVMTOBOOL (M);
0595                WELL_INFO.USED := B
0596           END;
0597           IF NEIGHBOUR_UP
0598           THEN PUT_ENTRY (WELL_INFO,INDEX)
0599      END
0600 END "RECEIVE WELL INFO";
0601
0602 FUNCTION ENTRY RCV ( M : UNIV SET_OF_MESSAGE) : MESSAGE_CODE;
0603 BEGIN RCV := RECEIVE_MESSAGE (M) END;
0604
0605 PROCEDURE ENTRY SND (M : UNIV MESSAGE_CODE);
0606 VAR MSG : NEIGHBUFF;
0607 BEGIN
0608 SEND_MESSAGE(M)
0609 END;
0610
0611 PROCEDURE ENTRY WLN (TEXT : LINE);
0612 BEGIN TERMINAL_RESOURCE.ACCESS;
0613       TERM.WRITELN (TEXT);
0614       TERMINAL_RESOURCE.RELEASE
0615 END;
0616
0617 PROCEDURE ENTRY PINT (N : INTEGER);
0618 BEGIN TERMINAL_RESOURCE.ACCESS;
0619       TERM.PRINTINT (N);
0620       TERMINAL_RESOURCE.RELEASE
0621 END;
0622
0623 BEGIN
0624      INIT TERM;
0625      ERROR_MESSAGES := (.INVMESS,SYSERRMESS,IEMESS,MAPERRMESS,
0626                         JMTSUCMESS,JMTUNMESS,ALUMTSUCMESS,ALUMTUNMESS,
0627                         PCUMTSUCMESS,PCUMTUNMESS.);
0628      WUP.CLEAR_FAULTY;
0629      PD.TABLE_ADDRESS := -DIAG_ADDRESS;
0630      PD.INDEX := DIAG_INDEX;
0631      JUST_ONLINE := FALSE;
0632      CYCLE
0633           "SEND A PERIODIC MESSAGE AND THEN RECEIVE ONE
0634            VICE VERSA IF BACKUP MODE"
0635           IF WUP.NEIGHBOUR_UP
0636           THEN
0637               IF WUP.MAIN_MODE
0638               THEN
0639               BEGIN
0640                    IF JUST_ONLINE
0641                    THEN
0642                    BEGIN
0643                         JUST_ONLINE := FALSE;
0644                         SEND_MESSAGE (INTMESS);
0645                         IF NOT WUP.NEIGHBOUR_UP
0646                         THEN
0647                         BEGIN
```

```
0648                              WUP.CLEAR_FAULTY;
0649                              FOR LOOP := 1 TO 600 DO B:= B
0650                         END
0651                    END
0652                    ELSE
0653                         IF WUP.PRESENT
0654                         THEN SEND_WELL_UPDATES
0655                         ELSE
0656                         BEGIN
0657                            SEND_MESSAGE (INTMESS)
0658                         END;
0659                    IF WUP.NEIGHBOUR_UP THEN
0660                    MESSAGE := RECEIVE_MESSAGE ((.INTMESS.) OR ERROR_MESSAGES);
0661               END
0662               ELSE "BACKUP MODE"
0663               BEGIN
0664                    MESSAGE := RECEIVE_MESSAGE ((.WELLMESS,INTMESS.) OR ERROR_MESSAGES);
0665                    IF MESSAGE = WELLMESS
0666                    THEN RECEIVE_WELL_INFO;
0667                    IF WUP.NEIGHBOUR_UP THEN SEND_MESSAGE (INTMESS)
0668               END;
0669          IF (NOT WUP.NEIGHBOUR_UP) AND (WUP.NOT_DIAGNOSED)
0670          THEN
0671          BEGIN
0672               WUP.SET_MAIN;
0673               WUP.SET_DIAGNOSED (TRUE);
0674               DIAGNOSE (RESULT,SEQUENCE_START,PD);
0675               WHILE NOT WUP.NEIGHBOUR_UP DO;  "WAIT FOR PROCESSOR TO BE REINSTATED"
0676               JUST_ONLINE := TRUE
0677          END
0678     END "CYCLE"
0679 END "DEBUG";
0680
0681
0682 TYPE USER = PROCESS (TERMINAL_RESOURCE : RESOURCE; WELLS: WELL_UPDATES);
0683
0684 "PROGRAM DATA SPACE =" +500
0685
0686 VAR PD : PROG_DESCRIPTOR;
0687    TERM : TERMINAL; ALREADY_ACCESSED : BOOLEAN;
0688
0689 PROGRAM COMMAND_INTERPRETER (TABLE : PROG_DESCRIPTOR);
0690 ENTRY INCH,OUTCH,WRITE,WRITELN,LINEFEED,PRINTINT,READINT,BACKUP_MODE,
0691       NEIGHBOUR_UP,CLEAR_FAULTY,CLEAR_QUEUE,WELL_UPDATE_PRESENT,
0692       UPDATE_BACKUP,GET_WELL_ENTRY,PUT_WELL_ENTRY;
0693
0694 PROCEDURE GET_TERMINAL;
0695 BEGIN IF NOT ALREADY_ACCESSED
0696       THEN
0697       BEGIN
0698            ALREADY_ACCESSED := TRUE;
0699            TERMINAL_RESOURCE.ACCESS
0700       END
0701 END;
```

```
0702
0703 PROCEDURE LEAVE_TERMINAL;
0704 BEGIN
0705       ALREADY_ACCESSED := FALSE;
0706       TERMINAL_RESOURCE.RELEASE
0707 END;
0708
0709 PROCEDURE ENTRY INCH (VAR C : CHAR);
0710 BEGIN
0711       IF ALREADY_ACCESSED
0712       THEN LEAVE_TERMINAL;
0713       TERM.READ (C)
0714 END;
0715
0716 PROCEDURE ENTRY OUTCH (C : CHAR);
0717 BEGIN GET_TERMINAL; TERM.OCH (C) END;
0718
0719 PROCEDURE ENTRY WRITE (TEXT : LINE);
0720 BEGIN GET_TERMINAL; TERM.WRITE (TEXT) END;
0721
0722 PROCEDURE ENTRY WRITELN (TEXT : LINE);
0723 BEGIN GET_TERMINAL; TERM.WRITELN (TEXT); LEAVE_TERMINAL END;
0724
0725 PROCEDURE ENTRY LINEFEED;
0726 BEGIN GET_TERMINAL; TERM.LINEFEED; LEAVE_TERMINAL END;
0727
0728 PROCEDURE ENTRY PRINTINT ( N : UNIV INTEGER);
0729 BEGIN GET_TERMINAL; TERM.PRINTINT (N) END;
0730
0731 FUNCTION ENTRY READINT : INTEGER;
0732 BEGIN GET_TERMINAL; READINT := TERM.READINT; LEAVE_TERMINAL END;
0733
0734 FUNCTION ENTRY BACKUP_MODE : BOOLEAN;
0735 BEGIN BACKUP_MODE := NOT (WELLS.MAIN_MODE) END;
0736
0737 FUNCTION ENTRY NEIGHBOUR_UP : BOOLEAN;
0738 BEGIN NEIGHBOUR_UP := WELLS.NEIGHBOUR_UP END;
0739
0740 PROCEDURE ENTRY CLEAR_FAULTY;
0741 BEGIN WELLS.CLEAR_FAULTY; WELLS.SET_DIAGNOSED (FALSE) END;
0742
0743 PROCEDURE ENTRY CLEAR_QUEUE;
0744 BEGIN WELLS.CLEAR_QUEUE END;
0745
0746 FUNCTION ENTRY WELL_UPDATE_PRESENT : BOOLEAN;
0747 BEGIN WELL_UPDATE_PRESENT := WELLS.PRESENT END;
0748
0749 PROCEDURE ENTRY UPDATE_BACKUP (VALUE : INTEGER);
0750 BEGIN WELLS.PUT_MESSAGE (VALUE) END;
0751
0752 PROCEDURE ENTRY GET_WELL_ENTRY (VAR WELL_ENTRY : WELL_RECORD; INDEX : INTEGER);
0753 BEGIN WELLS.GET_ENTRY (WELL_ENTRY, INDEX) END;
0754
0755 PROCEDURE ENTRY PUT_WELL_ENTRY (VAR WELL_ENTRY : WELL_RECORD; INDEX : INTEGER);
```

```
0756 BEGIN WELLS.PUT_ENTRY (WELL_ENTRY, INDEX) END;
0757
0758 BEGIN
0759       INIT TERM;
0760       ALREADY_ACCESSED := FALSE;
0761       PD.INDEX := CMND_INDEX;
0762       PD.TABLE_ADDRESS := CMND_ADDRESS;
0763       CYCLE
0764            COMMAND_INTERPRETER (PD)
0765       END
0766 END; "USER PROCESS"
0767
0768
0769 "
0770 INITIAL PROCESS
0771 "
0772
0773 VAR T : TERMINAL; W : WELL_UPDATES;
0774     N : NEIGHBOUR; R : RESOURCE;
0775     D : DEBUG; C : COMMUNICATOR;U : USER;
0776     CH : CHAR;
0777
0778 BEGIN
0779       INIT T,W,R;
0780       T.WRITELN ('SUPER SIXTEEN PROCESSOR£' );
0781       T.WRITELN ('FAULT TOLERANT CONCURRENT PASCAL ENGINE£' );
0782       T.WRITELN ('TYPE M FOR MAIN, B FOR BACKUP£' );
0783       T.READ (CH);
0784       IF CH = 'M' THEN W.SET_MAIN
0785                   ELSE W.SET_BACKUP;
0786       T.LINEFEED;
0787         INIT N,C (N), D(R,N,W), U (R,W)
0788 END.
```

```
0001 (NUMBER)
0002 " SEQUENTIAL PASCAL PROGRAM TO INTERPRET THE COMMANDS
0003   TYPED IN AT THE SUPER SIXTEEN TERMINAL"
0004 CONST CR ='(:13:)';NL = '(:10:)'; SPACE =' ';
0005 CONST LINELENGTH = 80; CONST IDLENGTH = 14; CONST WELL_ENTRIES = 15;
0006 TYPE LINE = ARRAY (.1..LINELENGTH.) OF CHAR;
0007 TYPE IDENTIFIER = ARRAY (.1..IDLENGTH.) OF CHAR;
0008 TYPE WELL_RECORD = RECORD
0009                          NAME : IDENTIFIER;
0010                          SIZE : INTEGER;
0011                          LOCATION : IDENTIFIER;
0012                          USED : BOOLEAN
0013                    END;
0014
0015 CONST BACKUP = 'BACKUP £';
0016 CONST MAIN = 'MAIN £';
0017 CONST WELL_NAME = 'WELL NAME? £';
0018 CONST WELL_NOT_FOUND = 'WELL NOT FOUND£ ';
0019 CONST WELL_SIZE = 'WELL SIZE £ ';
0020 CONST WELL_LOCATION = 'LOCATION £';
0021 CONST NEW_VALUE = ' NEW VALUE? £ ';
0022 CONST UPDATE_COMPLETE = 'UPDATE COMPLETE£';
0023
0024 "PREFIX PROCEDURES"
0025 PROCEDURE INCH (VAR C : CHAR);
0026 PROCEDURE OUTCH (C : CHAR);
0027 PROCEDURE WRITE (TEXT : LINE);
0028 PROCEDURE WRITELN (TEXT : LINE);
0029 PROCEDURE NEWLINE;
0030 PROCEDURE PRINTINT (N : UNIV INTEGER);
0031 FUNCTION READINT : INTEGER;
0032 FUNCTION BACKUP_MODE : BOOLEAN;
0033 FUNCTION NEIGHBOUR_UP : BOOLEAN;
0034 PROCEDURE CLEAR_FAULTY;
0035 PROCEDURE CLEAR_QUEUE;
0036 FUNCTION WELL_UPDATE_PRESENT : BOOLEAN;
0037 PROCEDURE UPDATE_BACKUP (VALUE : INTEGER);
0038 PROCEDURE GET_WELL_ENTRY (VAR WELL_ENTRY : WELL_RECORD;INDEX : INTEGER);
0039 PROCEDURE PUT_WELL_ENTRY (WELL_ENTRY : WELL_RECORD; INDEX : INTEGER);
0040
0041 PROGRAM COMMAND_INTERPRETER;
0042
0043 VAR CURRENT_WELL : WELL_RECORD;
0044     CH,COMMAND : CHAR;
0045     ANY_WELLS,FOUND : BOOLEAN;
0046     INDEX,LOOP : INTEGER;
0047     STRING : IDENTIFIER;
0048
0049 PROCEDURE INVALID;
0050 BEGIN WRITELN ('INVALID COMMAND£') END;
0051
0052 PROCEDURE NOT_ALLOWED;
0053 BEGIN
```

```
0054       WRITELN ('COMMAND NOT ALLOWED IN £');
0055       IF BACKUP MODE
0056       THEN WRITE (BACKUP)
0057       ELSE WRITE (MAIN);
0058       WRITELN ('MODE£ ')
0059  END;
0060
0061  PROCEDURE READID (VAR STRING : IDENTIFIER);
0062  VAR LOOP : INTEGER;
0063  BEGIN
0064       FOR LOOP := 1 TO IDLENGTH DO STRING (.LOOP.) := ' ';
0065       STRING (.IDLENGTH-1.) := '£';
0066       LOOP := 0;
0067       CH := CHR (0);
0068       WHILE (CH <> CR) AND (LOOP < IDLENGTH - 2) DO
0069       BEGIN
0070            LOOP := SUCC (LOOP);
0071            INCH (CH);
0072            IF CH <> CR THEN STRING (.LOOP.) := CH
0073       END;
0074       IF CH <> CR THEN OUTCH (CR);
0075       OUTCH (NL)
0076  END;
0077
0078  PROCEDURE MATCH NAME (VAR WELL : WELL RECORD; VAR INDEX : INTEGER);
0079  VAR LOOP : INTEGER;
0080  BEGIN
0081       WRITE (WELL NAME);
0082       READID (STRING);
0083       FOUND := FALSE;
0084       LOOP := 0;
0085       WHILE (LOOP < WELL ENTRIES) AND (NOT FOUND) DO
0086       BEGIN
0087            LOOP := SUCC (LOOP);
0088            GET WELL ENTRY (WELL,LOOP);
0089            IF (WELL.NAME = STRING) AND (WELL.USED)
0090            THEN FOUND := TRUE
0091       END;
0092       IF FOUND THEN INDEX := LOOP
0093                ELSE INDEX := -1
0094  END;
0095
0096  *
0097  START OF PROGRAM
0098  *
0099
0100  BEGIN
0101       "INPUT A CHARACTER FROM THE TERMINAL AND CHECK IT'S VALIDITY"
0102       WRITE ('@ £ ');     "PROMPT"
0103       READID (STRING);
0104       COMMAND := STRING (.1.);
0105       IF (COMMAND >= 'a') AND (COMMAND <= 'z')
0106       "CONVERT TO UPPER CASE"
0107       THEN COMMAND := CHR (ORD (COMMAND) - 32);
```

```
0108        IF (COMMAND < 'C') OR (COMMAND >'W')
0109        THEN INVALID
0110        ELSE
0111        CASE COMMAND
0112        OF 'E','F','G','H','J','K','L','M','N','O','Q','S',
0113           'T','U','V' : INVALID;
0114           'W' : "DISPLAY ALL WELL DATA"
0115                 BEGIN
0116                      WRITE ('WELL NAME     £');
0117                      WRITE (' LOCATION  £');
0118                      WRITELN (' SIZE£');
0119                      WRITELN ('-----------------------------------£');
0120                      ANY WELLS := FALSE;
0121                      FOR LOOP := 1 TO WELL ENTRIES DO
0122                      BEGIN
0123                           GET WELL ENTRY (CURRENT WELL,LOOP);
0124                           IF CURRENT WELL.USED
0125                           THEN
0126                           BEGIN
0127                                ANY WELLS := TRUE;
0128                                WRITE (CURRENT WELL.NAME);
0129                                OUTCH (SPACE);
0130                                WRITE (CURRENT WELL.LOCATION);
0131                                PRINTINT (CURRENT WELL.SIZE);
0132                                NEWLINE
0133                           END
0134                      END;
0135                      IF NOT ANY WELLS
0136                      THEN WRITELN ('NO WELLS PRESENT£ ')
0137                 END;
0138           'C' :" CHANGE WELL STATUS"
0139                 BEGIN
0140                      IF BACKUP MODE
0141                      THEN NOT ALLOWED
0142                      ELSE
0143                      BEGIN
0144                           MATCH NAME (CURRENT WELL,INDEX);
0145                           IF INDEX = -1 THEN WRITELN (WELL NOT FOUND)
0146                           ELSE
0147                           BEGIN
0148                                WRITE (WELL SIZE);
0149                                PRINTINT (CURRENT WELL.SIZE);
0150                                WRITE (NEW VALUE);
0151                                CURRENT WELL.SIZE := READINT;
0152                                NEWLINE;
0153                                WRITE (WELL LOCATION);
0154                                WRITE (CURRENT WELL.LOCATION);
0155                                WRITE (NEW VALUE);
0156                                READID (CURRENT WELL.LOCATION);
0157                                NEWLINE;
0158                                PUT WELL ENTRY (CURRENT WELL,INDEX);
0159                                UPDATE BACKUP (INDEX)
0160                           END
0161                      END
```

```
0162              END "CHANGE WELL STATUS";
0163         'D' : "DELETE WELL ENTRY"
0164              IF BACKUP MODE
0165              THEN NOT ALLOWED
0166              ELSE
0167              BEGIN
0168                   MATCH NAME (CURRENT WELL, INDEX);
0169                   IF INDEX <> -1
0170                   THEN
0171                   BEGIN
0172                        CURRENT WELL.USED := FALSE;
0173                        PUT WELL ENTRY (CURRENT WELL, INDEX);
0174                        UPDATE BACKUP (INDEX)
0175                   END
0176                   ELSE WRITELN (WELL NOT FOUND)
0177              END;
0178         'P' : "OUTPUT PROCESSOR STATUS"
0179              BEGIN
0180                   WRITE ('PROCESSOR STATUS = £');
0181                   IF BACKUP MODE
0182                   THEN WRITELN (BACKUP)
0183                   ELSE WRITELN (MAIN);
0184                   WRITE ('NEIGHBOUR PROCESSOR £ ');
0185                   IF NEIGHBOUR UP
0186                   THEN WRITELN ('OPERATIONAL£')
0187                   ELSE WRITELN ('FAULTY£ ')
0188              END;
0189         'I' : "INSERT NEW WELL"
0190              IF BACKUP MODE
0191              THEN NOT ALLOWED
0192              ELSE
0193              BEGIN
0194                   "FIND THE FIRST FREE ENTRY"
0195                   FOUND := FALSE;
0196                   LOOP := 0;
0197                   WHILE (NOT FOUND) AND (LOOP < WELL ENTRIES) DO
0198                   BEGIN
0199                        LOOP := SUCC (LOOP);
0200                        GET WELL ENTRY (CURRENT WELL, LOOP);
0201                        IF NOT CURRENT WELL.USED
0202                        THEN FOUND := TRUE
0203                   END;
0204                   IF NOT FOUND
0205                   THEN WRITELN ('NO FREE SPACE£')
0206                   ELSE
0207                   BEGIN
0208                        WRITE (WELL NAME);
0209                        READID (STRING);
0210                        NEWLINE;
0211                        FOUND := FALSE;
0212                        INDEX := 0;
0213                        WHILE (INDEX < WELL ENTRIES) AND (NOT FOUND) DO
0214                        BEGIN
0215                             INDEX := SUCC (INDEX);
```

```
0216                              GET WELL ENTRY (CURRENT WELL,INDEX);
0217                              IF (CURRENT WELL.USED) AND (CURRENT WELL.NAME =STRING)
0218                              THEN FOUND := TRUE
0219                         END;
0220                         IF FOUND
0221                         THEN WRITELN ('NAME ALREADY EXISTS£')
0222                         ELSE
0223                         BEGIN
0224                              CURRENT WELL.NAME := STRING;
0225                              WRITE (WELL SIZE);
0226                              CURRENT WELL.SIZE := READINT;
0227                              NEWLINE;
0228                              WRITE (WELL LOCATION);
0229                              READID (CURRENT WELL.LOCATION);
0230                              NEWLINE;
0231                              CURRENT WELL.USED := TRUE;
0232                              PUT WELL ENTRY (CURRENT WELL,LOOP);
0233                              UPDATE BACKUP (LOOP);
0234                              WRITELN (UPDATE COMPLETE)
0235                         END
0236                    END
0237               END"INSERT";
0238          'R' :"RE-INSTATE NEIGHBOUR PROCESSOR"
0239               BEGIN
0240                    CLEAR QUEUE;
0241                    CLEAR FAULTY;
0242                    FOR LOOP := 1 TO WELL ENTRIES DO
0243                    BEGIN
0244                         GET WELL ENTRY (CURRENT WELL,LOOP);
0245                         IF CURRENT WELL.USED
0246                         THEN UPDATE BACKUP (LOOP)
0247                    END
0248               END " RE-INSTATE"
0249          END "CASE"
0250 END "PROGRAM".
```

```
0001 (NUMBER)
0002 "PREFIX FOR DEBUG PROGRAMS"
0003 CONST NL = '(:10:)'; CR = '(:13:)';MESSAGE_END ='£';
0004 CONST LINELENGTH = 80;BUFLEN = 5;
0005 TYPE LINE = ARRAY (.1..LINELENGTH.) OF CHAR;
0006 TYPE RESULT_CODE = (SOFTWARE_ERROR,ALU_FAULTY,PCU_FAULTY,
0007                      AM2904_FAULTY,TREG_FAULTY,TTLATCH_FAULTY,
0008                      TESTTREE_FAULTY,ZREGMS_FAULTY,ZREGLS_FAULTY,
0009                      ZOREGLS_FAULTY,ZIREG_FAULTY,MAP_FAULTY,
0010                      PCUTRAN_FAULTY,OTHER);
0011 TYPE MESSAGE_CODE = (FIRSTMESS,
0012             TFMESS,        .     "TIMER FAILURE MESSAGE"
0013             IRMESS,              "INTERRUPT MESSAGE"
0014             JMTSUCMESS,          "JOINT MEMORY TEST SUCCESSFULL"
0015             JMTUNMESS,           "JOINT MEMORY TEST UNSUCCESSFULL"
0016             ALUMTSUCMESS,        "ALU MEMORY TEST SUCCESSFULL"
0017             ALUMTUNMESS,         "ALU MEMORY TEST UNSUCCESSFULL"
0018             PCUMTSUCMESS,        "PCU MEMORY TEST SUCCESSFULL"
0019             PCUMTUNMESS,         "PCU MEMORY TEST UNSUCCESSFULL"
0020             PCUALUSUCMESS,       "PCU/ALU TEST SUCCESSFULL"
0021             PCUALUUNMESS,        "PCU/ALU TEST UNSUCCESSFULL"
0022             ALUSUCMESS,          "ALU TEST SUCCESSFULL"
0023             ALUUNMESS,           "ALU TEST UNSUCCESSFULL"
0024             BMTSUCMESS,          "BYTE MEMORY TEST SUCCESSFULL"
0025             BMTUNMESS,           "BYTE MEMORY TEST UNSUCCESSFULL"
0026             MAPZIMESS,           "MAP/ZI TEST SUCCESSFULL"
0027             INVMESS,             "INVALID INSTRUCTION MESSAGE"
0028             SYSERRMESS,          "SYSTEM ERROR MESSAGE"
0029             INTMESS,             "PERIODIC INTERRUPT MESSAGE"
0030             IEMESS,              "INTERRUPT ERROR MESSAGE"
0031             MAPERRMESS,          "MAP DECODER ERROR SUSPECTED"
0032             WELLMESS,            "WELL UPDATE MESSAGE"
0033             LASTMESS,
0034             A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,A33,A34,A35,A36,A37,
0035             A38,A39,A40,A41,A42,A43,A44,A45,A46,A47,A48,A49,A50,A51,A52,A53,
0036             A54,A55,A56,A57,A58,A59,A60,A61,A62,A63,A64,A65,A66,A67,A68,A69,
0037             A70,A71,A72,A73,A74,A75,A76,A77,A78,A79,A80,A81,A82,A83,A84,
0038             A85,A86,A87);
0039 TYPE SET OF MESSAGE = SET OF MESSAGE_CODE;
0040 FUNCTION RECEIVE ( M : SET_OF_MESSAGE) : MESSAGE_CODE;
0041 PROCEDURE SEND (M : UNIV MESSAGE_CODE);
0042 PROCEDURE WRITELN (TEXT : LINE);
0043 PROCEDURE PRINTINT (N : UNIV INTEGER);
0044 PROGRAM DIAGNOSE (VAR RESULT :  RESULT_CODE; SEQUENCE_START : MESSAGE_CODE);
0045 VAR TEST_RESULTS : ARRAY (.FIRSTMESS..LASTMESS.) OF BOOLEAN;
0046     MESSAGE_RECEIVED : MESSAGE_CODE;
0047     EXPECTED_MESSAGES : SET OF MESSAGE_CODE;
0048     MESSAGE_COUNT : INTEGER;
0049     JTEST,ALUMTTEST,PCUMTTEST,PCUALUTEST,ALUTEST,BYTETEST,MAPZITEST : BOOLEAN;
0050     EXCEPTION,LINE : INTEGER;
0051 PROCEDURE INITIALISE;
0052 VAR INDEX : MESSAGE_CODE;
0053 BEGIN
```

```
0054        ALUTEST := FALSE;
0055        FOR INDEX := FIRSTMESS TO LASTMESS
0056        DO TEST_RESULTS (.INDEX.) := FALSE
0057 END;
0058 PROCEDURE RECEIVE_MESSAGE (M : UNIV SET_OF_MESSAGE;VAR VALUE : UNIV MESSAGE_CODE);
0059 BEGIN
0060       VALUE := RECEIVE (M);
0061       IF (VALUE )= FIRSTMESS) AND (VALUE (= LASTMESS)
0062       THEN TEST_RESULTS (.VALUE.) := TRUE;
0063       WRITELN ('MESSAGE RECEIVED£ ');
0064       IF (VALUE ) FIRSTMESS) AND (VALUE ( LASTMESS)
0065       THEN BEGIN
0066              CASE VALUE OF
0067                    TFMESS: WRITELN ('TIMER INTERRUPT HARDWARE FAULTY£');
0068                    IRMESS: WRITELN ('PUSH BUTTON INTERRUPT£');
0069                    JMTSUCMESS: WRITELN ('PROCESSOR FAULT :- JOINT MEMORY OK£ ');
0070                    JMTUNMESS: WRITELN ('PROCESSOR FAULT:- JOINT MEMORY FAILURE£ ');
0071                    ALUMTSUCMESS: WRITELN ('ALU MEMORY TEST PASSED£ ');
0072                    ALUMTUNMESS: WRITELN ('ALU MEMORY TEST FAILED£ ');
0073                    PCUMTSUCMESS: WRITELN ('PCU MEMORY TEST PASSED£ ');
0074                    PCUMTUNMESS: WRITELN ('PCU MEMORY TEST FAILED£ ');
0075                    PCUALUSUCMESS: WRITELN ('ALU/PCU TEST PASSED£');
0076                    PCUALUUNMESS: WRITELN ('ALU/PCU TEST FAILED£');
0077                    ALUSUCMESS: WRITELN ('ALU TEST PASSED£');
0078                    ALUUNMESS: WRITELN ('ALU TEST FAILED£');
0079                    BMTSUCMESS: WRITELN ('BYTE MEMORY TEST PASSED£');
0080                    BMTUNMESS: WRITELN ('BYTE MEMORY TEST FAILED£');
0081                    MAPZIMESS: WRITELN ('MAPZI OP-CODE RECEIVED£ ');
0082                    INVMESS: WRITELN ('INVALID INSTRUCTION£');
0083                    SYSERRMESS: WRITELN ('SYSTEM ERROR£ ');
0084                    INTMESS: WRITELN ('PERIODIC INTERRUPT MESSAGE£ ');
0085                    IEMESS: WRITELN ('INTERRUPT ERROR MESSAGE£');
0086                    MAPERRMESS: WRITELN ('MAP ERROR SUSPECTED£')
0087               END
0088           END
0089           ELSE PRINTINT (VALUE);
0090 END;
0091 PROCEDURE TESTMAP;
0092 BEGIN
0093       SEND (226);
0094       RECEIVE_MESSAGE ((.MAPZIMESS.),MESSAGE_RECEIVED);
0095       MAPZITEST := MESSAGE_RECEIVED = MAPZIMESS
0096 END;
0097 *
0098 START OF PROGRAM
0099 *
0100 BEGIN
0101       WRITELN ('DIAGNOSIS STARTED£');
0102       INITIALISE;
0103       MESSAGE_COUNT := 0;
0104       EXPECTED_MESSAGES := (.IEMESS,MAPERRMESS,JMTSUCMESS,JMTUNMESS,ALUMTSUCMESS,ALUMTUNME
SS,
0105                                  PCUMTSUCMESS,PCUMTUNMESS.);
0106       RECEIVE_MESSAGE (EXPECTED_MESSAGES,MESSAGE_RECEIVED);
```

```
0107      WHILE (MESSAGE_COUNT (= 7)
0108        AND (MESSAGE_RECEIVED () BMTSUCMESS)
0109        AND (MESSAGE_RECEIVED () BMTUNMESS)
0110      DO
0111      BEGIN
0112           MESSAGE_COUNT := SUCC (MESSAGE_COUNT);
0113           CASE MESSAGE_RECEIVED OF
0114                IEMESS, MAPERRMESS, LASTMESS : ;
0115                JMTSUCMESS, JMTUNMESS : EXPECTED_MESSAGES:= (.ALUMTSUCMESS, ALUMTUNMESS,
0116                                                             PCUMTSUCMESS, PCUMTUNMESS,
0117                                                             PCUALUSUCMESS, PCUALUUNMESS.);
0118                ALUMTSUCMESS, ALUMTUNMESS : EXPECTED_MESSAGES:= (.PCUMTSUCMESS, PCUMTUNMESS,
0119                                                                 PCUALUSUCMESS, PCUALUUNMESS,
0120                                                                 A85.);
0121                PCUMTSUCMESS, PCUMTUNMESS : EXPECTED_MESSAGES:= (.PCUALUSUCMESS, PCUALUUNMES
0122                                                             A85.);
0123                PCUALUSUCMESS, PCUALUUNMESS : EXPECTED_MESSAGES:= (.A85,
0124                                                               BMTSUCMESS, BMTUNMESS.);
0125                A85 : BEGIN
0126                           RECEIVE_MESSAGE ((.FIRSTMESS.), MESSAGE_RECEIVED);
0127                           IF (MESSAGE_RECEIVED = A85)
0128                           OR (MESSAGE_RECEIVED = A87)
0129                           THEN ALUTEST := TRUE
0130                           ELSE ALUTEST := FALSE;
0131                           EXPECTED_MESSAGES := (.BMTSUCMESS, BMTUNMESS.);
0132                           MESSAGE_COUNT := SUCC (MESSAGE_COUNT)
0133                      END;
0134
0135                A87 : EXPECTED_MESSAGES:= (.BMTSUCMESS, BMTUNMESS.);
0136                BMTSUCMESS, BMTUNMESS : "SKIP"
0137           END;
0138           RECEIVE_MESSAGE (EXPECTED_MESSAGES, MESSAGE_RECEIVED)
0139 END "OUTER WHILE LOOP";
0140      WRITELN ('MESSAGES RECEIVED£');
0141      JTEST := (TEST_RESULTS (.JMTSUCMESS.)) OR (SEQUENCE_START = JMTSUCMESS);
0142      ALUMTTEST := (TEST_RESULTS (.ALUMTSUCMESS.)) OR (SEQUENCE_START = ALUMTSUCMESS);
0143      PCUMTTEST := (TEST_RESULTS (.PCUMTSUCMESS.)) OR (SEQUENCE_START = PCUMTSUCMESS);
0144      PCUALUTEST := TEST_RESULTS (.PCUALUSUCMESS.);
0145      BYTETEST := TEST_RESULTS (.BMTSUCMESS.);
0146      RESULT := OTHER;
0147      IF JTEST AND ALUMTTEST AND PCUMTTEST AND PCUALUTEST AND ALUTEST AND BYTETEST
0148      THEN "SOFTWARE ERROR OR MAP/ZI FAILURE OR TEST TREE LATCH FAILURE"
0149      BEGIN
0150           IF (SEQUENCE_START = IEMESS) OR (TEST_RESULTS (.IEMESS.))
0151           THEN RESULT := TTLATCH_FAULTY
0152           ELSE BEGIN
0153             TESTMAP;
0154             IF MAPZITEST
0155             THEN RESULT := SOFTWARE_ERROR
0156             ELSE BEGIN
0157               IF (SEQUENCE_START = MAPERRMESS) OR (TEST_RESULTS (.MAPERRMESS.))
0158               THEN RESULT := MAP_FAULTY
0159               ELSE RESULT := ZIREG_FAULTY
```

```
0160              END
0161            END
0162       END
0163       ELSE IF (NOT JTEST) AND (NOT ALUMTTEST) AND (PCUMTTEST) AND (NOT PCUALUTEST) AND (NO
T ALUTEST)
0164       THEN RESULT := ALU_FAULTY
0165       ELSE IF (NOT JTEST) AND (ALUMTTEST) AND (NOT PCUMTTEST) AND
0166               (NOT PCUALUTEST) AND (ALUTEST)
0167            THEN RESULT := PCU_FAULTY
0168       ELSE IF (NOT JTEST) AND (NOT ALUMTTEST) AND (PCUMTTEST)
0169            AND (NOT PCUALUTEST) AND (ALUTEST)
0170       THEN RESULT := AM2904_FAULTY
0171       ELSE IF ((JTEST) AND (NOT ALUMTTEST) AND (NOT PCUMTTEST) AND
0172                (NOT PCUALUTEST))
0173            OR ((JTEST) AND (ALUMTTEST) AND (NOT PCUMTTEST) AND
0174                (NOT PCUALUTEST))
0175            THEN RESULT := PCUTRAN_FAULTY
0176       ELSE IF (JTEST) AND (ALUMTTEST) AND (PCUMTTEST) AND (NOT PCUALUTEST) AND (ALUTEST)
0177            THEN RESULT := TREG_FAULTY
0178       ELSE IF(NOT JTEST) AND (NOT ALUMTTEST) AND (NOT PCUMTTEST)
0179            AND (PCUALUTEST) AND (ALUTEST) AND (BYTETEST)
0180            THEN RESULT := ZREGMS_FAULTY
0181       ELSE IF (JTEST) AND (ALUMTTEST) AND (NOT PCUMTTEST) AND
0182               (PCUALUTEST) AND (ALUTEST)
0183            THEN RESULT := TESTTREE_FAULTY
0184       ELSE IF (NOT JTEST) AND (NOT ALUMTTEST) AND (NOT PCUMTTEST) AND
0185               (NOT PCUALUTEST) AND (ALUTEST)
0186            THEN RESULT := OTHER
0187       ELSE
0188         BEGIN
0189              TESTMAP;
0190              IF (NOT JTEST) AND (NOT ALUMTTEST) AND (NOT PCUMTTEST)
0191              AND (PCUALUTEST) AND (ALUTEST) AND (NOT BYTETEST)
0192              THEN
0193                BEGIN
0194                     IF MAPZITEST
0195                     THEN RESULT := ZOREGLS_FAULTY
0196                     ELSE RESULT := ZREGLS_FAULTY
0197                END
0198          END;
0199          CASE RESULT OF
0200          SOFTWARE_ERROR: BEGIN
0201                            RECEIVE MESSAGE ((.FIRSTMESS.),EXCEPTION);
0202                            RECEIVE MESSAGE ((.FIRSTMESS.),LINE);
0203                            WRITELN ('SOFTWARE ERROR£ ');
0204                            CASE EXCEPTION OF
0205                                 0: WRITELN ('TERMINATED EXCEPTION£ ');
0206                                 1: WRITELN ('OVERFLOW ERROR£ ');
0207                                 2: WRITELN ('POINTER ERROR£');
0208                                 3: WRITELN ('RANGE ERROR£');
0209                                 4: WRITELN ('VARIANT ERROR£');
0210                                 5: WRITELN ('HEAP LIMIT EXCEEDED£');
0211                                 6: WRITELN ('STACK LIMIT EXCEEDED£ ')
0212                             END;
```

```
0213                            WRITELN ('LINE NO.£ ');
0214                            PRINTINT (LINE)
0215                          END;
0216          ALU_FAULTY: WRITELN ('FAULTY ALU£ ');
0217          PCU_FAULTY: WRITELN ('FAULTY PCU£ ');
0218          AM2904_FAULTY: WRITELN ('FAULTY AM2904£');
0219          TREG_FAULTY: WRITELN ('FAULTY TREG£');
0220          TTLATCH_FAULTY: WRITELN ('TEST TREE LATCH FAULTY£ ');
0221          TESTTREE_FAULTY: WRITELN ('TEST TREE FAULTY£ ');
0222          ZREGMS_FAULTY: WRITELN ('DREG M.S. OR ZREG M.S. OR ZOREG M.S. FAULTY£');
0223          ZREGLS_FAULTY: WRITELN ('FAULTY L.S. ZREG£ ');
0224          Z0REGLS_FAULTY: WRITELN ('FAULTY L.S. Z0REG£');
0225          ZIREG_FAULTY: WRITELN ('FAULTY ZIREG£ ');
0226          MAP_FAULTY: WRITELN ('FAULTY MAP DECODER£ ');
0227          PCUTRAN_FAULTY: WRITELN ('FAULTY PCUTRAN£ ');
0228          OTHER: BEGIN
0229                       WRITELN ('UNDIAGNOSED FAILURE:-£');
0230                       WRITELN ('FAULTY CCU, CLOCK CONTROL, MEMORY OR MULTIPLE FAILURE£')
0231                 END
0232        END
0233 END.  "OF PROGRAM"
```

Appendix 13

FTOS User Guide

When the reset button on the Super Sixteen is pressed, the system is initialised and FTOS will be called. This will output the following prompt:-

```
SUPER SIXTEEN PROCESSOR
FAULT-TOLERANT CONCURRENT PASCAL ENGINE
TYPE M FOR MAIN, B FOR BACKUP
```

The user should type the character 'M' to put the processor in Main Mode and anything else to put it in Backup Mode. Assuming that there are two processors, this should always be performed on the Backup Processor first. If not, the Main Processor may mark the Backup Processor (which has not yet completely started up FTOS) as being faulty. As soon as the initialisation is complete, the Command Interpreter will be called and also fault detection will commence.

13.1 The Command Interpreter

The Command Interpreter outputs the prompt '@'. The user can then type in any one of six commands. All of these can be abbreviated to one letter, however FTOS will accept any multi-letter abbreviation of the full command name. The commands deal with oil well entries which are stored in the computer. Each record has three parameters; the well name, its location and its size. No two

entries may have the same well name but they may be at the same location and their sizes may be equal. The system can hold a maximum of fifteen oil well entries. The oil well and location names may be up to twelve characters long. They may consist of any characters. When inputting the well size, the user must type numerical (0-9) characters only. If this is not done then FTOS will output a question mark '?' and the user must re-input the number. Any commands which alter the well data are not allowed in backup mode. If the user attempts to call them then an error message will be output:-

COMMAND NOT ALLOWED IN BACKUP MODE

If an unknown command is typed in then an error message will be output:-

INVALID COMMAND

The commands are as follows.

WELL DATA (W) Command

This command displays all the well field data in the form:-

WELLNAME	LOCATION	SIZE
WELL1	LOCATION1	SIZE1
WELL2	LOCATION2	SIZE2
.		
.		
.		
WELLN	LOCATIONN	SIZEN

If there are no entries stored then it will output the message:-

NO WELLS PRESENT

CHANGE WELL STATUS (C) Command

This command is not allowed in backup mode. When it is called successfully it responds with:-

WELL NAME?

The user then types in the name of an oil well whose parameters he wishes to alter. If the well does not exist then an error message will be output and the command will terminate:-

WELL NOT FOUND

If the well does exist then the processor outputs:-

WELL SIZE xxx NEW VALUE?

Where 'xxx' is the present well size. The user types in the new well size and the processor responds with:-

LOCATION aaaaaaa NEW VALUE?

Where 'aaaaaa' is the present location. The user then types in the new location to complete the operation.

DELETE WELL ENTRY (D) Command

This command is not allowed in backup mode. If successfully invoked it invites the user to type in the name of the well entry to be deleted:-

WELL NAME?

If the well entry exists then it will be deleted, if not then an error message will be output:-

WELL NOT FOUND

INSERT NEW WELL (I) Command

This command is not allowed in backup mode. It inserts a new well entry. If there is no free space available to do this it outputs an error message and the command terminates:-

NO FREE SPACE

Otherwise, it outputs the message:-

WELL NAME?

The user then types in the name of the well he wants to be inserted. If an entry already exists with the same name then an error message is displayed and the command terminates:-

NAME ALREADY EXISTS

If not, then the processor outputs the prompt:-

WELL SIZE

and the user should type in the size of the well. The processor then outputs the prompt:-

WELL LOCATION

and the user types in the location of the well. Finally, a message is output and the command terminates:-

UPDATE COMPLETE

PROCESSOR STATUS (P) Command

This command outputs the status of both processors in the form:-

PROCESSOR STATUS = x

NEIGHBOUR PROCESSOR y

Where 'x' is the status of the processor on which the command was invoked and is either 'BACKUP' or 'MAIN'. Also, 'y' is the status of the other processor and is either 'OPERATIONAL' or 'FAULTY' depending on whether a fault has been detected.

RE-INSTATE (R) Command

This command is not allowed in backup mode. It should be used when a Faulty Processor has been repaired and should now be regarded by the Main Processor as an operational backup system.

13.2 Fault Detection and Analysis

If a fault is detected, then the message:-

NEIGHBOUR PROCESSOR IS FAULTY

DIAGNOSIS STARTED

is output. This is followed by a message indicating the result of the diagnosis. These messages are self-explanatory and correspond to the test sequences given in Table 4.2 of Chapter 4. If all tests pass, however, then it is assumed that there is a software error in the Faulty Processor and the following message is output:-

SOFTWARE ERROR

x

LINE NO.

y

Where 'x' is the type of exception and will be one of the following:-

TERMINATED EXCEPTION

OVERFLOW ERROR

POINTER ERROR

RANGE ERROR

VARIANT ERROR

HEAP LIMIT EXCEEDED

STACK LIMIT EXCEEDED

Also, 'y' is the line number (in the program source file) where the exception occured.