CERN - DATA HANDLING DIVISION
DD/75/17
CERN/Erasme/75-2
T. Lingjaerde
May 1975

A FAST MICROPROGRAMMABLE PROCESSOR

Presented at the MIMI 75 Conference,
2-5 June, 1975, Zürich

DD-ag

A FAST MICROGRAMMABLE PROCESSOR

Tor Lingjaerde

CERN, Geneva

ABSTRACT

A fully programmable processor has been built which can be used for numerous tasks requiring simple but fast data treatment in real time.

High speed of operation is obtained using dedicated autonomous sub-units which operate concurrently under control of a common command module. This unit is capable of issuing one 48 bit instruction every 80 ns.

The processor is loaded and controlled by a host mini-puter (PDP-11). The instruction set includes ADD, TRACT, SHIFT, BOOLEANS, MULTIPLY, DIVIDE and CONDITIONAL SKIPS and JUMPS.

Software presently available is a cross-assembler which runs on a PDP-10, a loader, a set of debugging programs and a histogramming routine.

1. INTRODUCTION

In the field of experimental high energy nuclear physics there is an ever-increasing need for very high speed data processing. Frequently large quantities of measured data are generated in bursts, and it is necessary to do some pre-processing in real time before the result is transmitted to an on-line mini-computer for further processing and/or storage on tape.

At CERN-DD considerable effort is being devoted to build very fast special purpose processors using combinatorial LSI circuits. [1,2] These systems use prewired data paths and the program is set up in a diode matrix.

The processor described in this paper uses a different roach. It is a fully programmable sequential machine with an instruction set comparable to that of a modern mini-computer. High speed operation is obtained by keeping data and control path delays to a minimum and by overlapping operations as much as possible.

It is inherently slower than hard wired devices, but it will execute suitable tasks more than one order of magnitude faster than a modern mini-computer.

Originally, the processor was designed as an integrated part of the CERN ERASME project [3,4], which is a system of five semi-automatic scanning and measuring machines for bubble chamber photographs. The prototype [5] was specifically implemented to generate and evaluate histograms of track co-ordinates in the automatic track-following mode of the machine.

Based upon the experience with the prototype which came into operation at the end of 1974, the more powerful version described in this paper was designed. The first unit is being built at present.

Apart from the application described above, the processor is being proposed to be used in data collection and preprocessing for wire-chamber experiments.

It is felt that the device could find numerous other applications in fields where a variety of short routines have to be executed at a speed which would be too high for a mini-computer.

2. GENERAL DESCRIPTION

Fig. 1 shows a simplified block diagram of the processor. It is made up from five main sections which are lodged in separate "NIM" type plug-in units, connected together by a printed circuit mother-board:

a) an instruction unit with a 256 x 48 bit read/write TTL memory

b) a 256 x 16 bit data memory of the same type as above

c) an arithmetic-logic unit which also contains the data flow control

d) a fast shift and bit test unit

e) a fixed point multiply and divide unit.

The two latter units are options and not shown on the block diagram.

There are two main buses in the processor : the 48 bit instruction bus (IB) which controls all operations, and the 16 bit data bus (DB) to which the data memory (DM) and all the processing units are connected. Data to and from external devices also pass via the DB.

The processor is connected on line to a host mini-computer a PDP-11, which is used to load the program, initiate the operation and finally read the processed data.

The machine runs on a single phase clock and one 48 bit instruction is executed per cycle. The prototype runs at 10MHz, but the revised version, which will use Schottky TTL memory, has been designed to work at a nominal cycle time of 80 ns.

To optimize speed of operation, the machine has been designed in such a way that as many activities as possible are executed simultaneously. Data and instructions reside in separate memories; every cycle a new instruction is fetched and at the same time the operands are brought in to the arithmetic unit from the data memory and/or the data file. Meanwhile, the addresses for the next data and instruction are being prepared in the dedicated arithmetic units. Book-keeping is done in parallel, rather than by a separate instruction. Finally, a unique "BREAK" system allows single instruction loops to be executed with up to four way automatic branching.

3. THE ARITHMETIC LOGIC UNIT

The arithmetic logic unit (ALU) is shown in Fig. 2 together with the associated registers and the data memory.

The ALU itself consists of three IC's of the type SN74S181 and a carry look-ahead SN74S182. These circuits are strictly combinatorial and the typical propogation time for adding two 16 bit words is 20 ns. The 5 bit opcode and the input carry are held in a dedicated 6 bit register (ALO) which is loaded from the instruction bus (IB). The 64 possible operations include add, subtract, left shift and Booleans. An exhaustive list is found in a semiconductor manufacturer's manual or in this processor's software documentation [6].

The two operands are latched in the registers A and B at the beginning of a cycle whenever their "enable" bits are present in the instruction word (IB <45> and <27>). The A operand is taken from the common data bus of the processor (DB <15-0>). It can thus be loaded directly with data from the host computer, an external device, the data memory or immediately from the instruction bus. The B register is connected to the data file (DF) which contains 8 x 16 bit words (4 x 74S172). The addressing of the latter is done via the IB during the second half of the instruction read-out (Fig.4). Thus, the content of the selected file register can be strobed into the B register at clock time. In other words, the DF combined with the B latch look like 8 separate accumulator registers.

As shown in Fig. 2, the DF has two input ports and two output ports. These ports can operate simultaneously and independently to an extent which is only limited by the addressing structure. Due to the restricted number of bits available in the instruction word, it was not possible to take full advantage of all the addressing possibilities of the DF.

The output of the ALU is connected to the common data bus DB through two sets of tristate gates, one set of which has been connected to swap bytes. From the DB, the information can then be transferred to any of the units connected to this bus, whether internal or external.

Most frequently results will be written into the data memory (DM). As this happens during the second half of a cycle, it is possible to execute an operation, for instance ADD, in the ALU and still have time to write the result into the DM during the same cycle. This means that within a program, we can perform the operation A(ALU)B → C within one cycle. A typical example would be to add a constant to a block of data in the DM. This would be done in a loop with only two instructions, one for read and the next one for the ADD and write. The DM addressing would be executed in parallel and the word count handled in the instruction unit by means of the BREAK logic as explained in a separate chapter.

To perform conditional branching, the carry (CN+4) and A=B outputs of the ALU have been brought out to the BREAK logic of the Instruction Unit (see Fig. 1 and 3). The BREAK logic is treated below in a separate chapter.

4. THE MULTIPLY/DIVIDE AND SHIFT UNITS

The multiply/divide unit is a separate and autonomous option.

The operands are loaded from the common data bus (DB) by a micro-instruction. As soon as the unit has completed its execution,a flag is set which causes a conditional jump in the instruction unit, provided the BREAK system has been activated. Multiplication takes 1,6 μs and division 1.8 μs. The unit works in signed or unsigned integer mode, and executes concurrently with other activities in the processor.

The shift unit uses two levels of the AM 25S10 combinatorial shift network. End-off right or left shift as well as rotate by any number of places up to 15 is carried out in one single 80 ns cycle.

As combinatorial "drop-through" shifters are not well suited for detection of overflow, a separate bit test logic has been incorporated in the shift unit. It works as follows: a bit pattern which is set up in a mask register by program is compared with the data word. If the BREAK system is activated, the processor will skip the next instruction whenever any of the masked bits is present in the data word, and it will skip two instructions if all the masked bits are present.

This unit has been designed, but not yet built.

5. THE DATA MEMORY

The data memory (DM) is shown in Fig. 2 together with its associated addressing and control logic.

The memory itself is made up from 16 TTL memory chips each of which contain 256 static storage cells (SN74200 or equivalent). A read or write cycle takes 45 ns. By using a recent Schottky version, the access time is reduced to 30 ns typically.

The data memory unit has its own dedicated addressing system.

The arithmetic unit (DALU) can perform 8 different operations : A, A+1, A-1, B, A+B, A+B+1, all = 0 and finally all = 1.

The A input of the DALU is taken directly from the Data Address Register (DAR) making address modifications relative to the present address. When the opcode DALO=A+1 the DAR-DALU combination looks like a counter where the strobe SDA acts as the incrementing command. If this strobe is left out (IB <25> = 0), or the clock to the module stopped, the address will remain at its latest value.

The addressing logic has been designed in such a way that it is easy to jump from one block of data to another. The base values are set up directly from the IB in the 8 word Address File (AF). They can then be used as absolute addresses by executing a direct transfer to the DAR via the common bus DAB <7:0> or as indices by tranferring the relevant AF word to the Data Index Register (DAI). The DALO opcodes available for indexing are A+B or A+B+1. The DAR content can be saved at any time in the AF by giving the AF address on IB <38:36> and setting the strobe bit IB <39> SAVDA. As the AF permits read and write at the same time, save and retrieve can be executed within one cycle.

The lower byte of the DB is connected to the DAB via tristate gates. Therefore all DB sources can be used as addresses. Immediate addressing is obtained by connecting IB <15:0> to the DB, and using the lower byte only. The DAR can in turn be read on to the DB.

External devices can address the DM through two different ports: one port is reserved for the host computer (UNIBUS),one for DMA addressing (HIST BIN NO.). Data is transferred directly on to the DB.

In one mode, the entire DM is connected to the PDP-11 UNIBUS; it then looks like a section of the host computer's own memory. This is particularly convenient when treating processed results, because the data does not have to be moved before it can be further processed by the PDP-11.

The data memory can be expanded in blocks of 256 words by means of paging. The page number is set up via the IB.

So far, the processor has not been used for tasks which need more than the basic page. It is estimated that the fast memory could be expanded to a few thousand words at the most, at the expense of some reduction in speed. Direct connection to a 16k MOS memory with 250 ns access time is being studied at present.

6. THE INSTRUCTION UNIT

6.1. PRINCIPLE

The main parts of the Instruction Unit, which is shown on Fig. 3 are:

a) a fast TTL memory with 256 words of 48 bits

b) the instruction addressing logic, which includes the conditional "BREAK" system

c) a read-in sequencer to load programs from the host computer

d) the UNIBUS interface, part of which is external to the unit

e) a priority register which can be set by external signals to start a given routine.

Although the execution time of the individual instructions is directly related to the memory access time, the effective speed at which a program runs is even more dependent upon the architecture of the machine. Therefore, the Instruction Unit has been built in such a way that operations will overlap in time. Thus, in order to overlap fetch and execute cycles, it has its own arithmetic unit for address calculations. Further, the address for the next instruction is evaluated during the time the present instruction is read from the memory.

The Instruction Unit is autonomous in the sense that it is indifferent to what kind of units it controls. The only incoming signals are the ones that are necessary to load the instructions, start routines and to set up branch operations.

A versatile "BREAK" system allows single instruction loops to be executed with up to four-way branching. Direct jumps are executed without loss of a cycle.

A more detailed description of the logic is given below.

6.2. THE INSTRUCTION MEMORY

Fast, static TTL memory chips of the type SN74200 or a faster Schottky version (82S16 or 27LS00) are used. The total memory capacity of the prototype is 256 words of 48 bits.

The operation of the memory is simple: when a "READ" and "ENABLE" signal is given, the content of the addressed cell will occur on the output 45 ns later (30 ns for Schottky version) and stay until any of the mentioned signals are changed. The processor cycle time is 80 ns, as shown on Fig. 4. When a program is running only READ operations are performed. This means that the Instruction Memory could be replaced by a Read Only Memory once the whole program has been finally frozen.

The programs are loaded from the host computer. A sequencer automatically assembles three incoming 16 bit words into one 48 bit instruction and then advances the address register (IAR).

6.3. THE INSTRUCTION ADDRESSING SYSTEM

As shown in Fig. 4 a new instruction cycle is started each time an address is strobed into the Instruction Address Register (see Fig. 3). This address is prepared in the dedicated instruction addressing logic concurrently with the other activities in the processor. The main parts of this logic are:

1. a dedicated Arithmetic-Logic Unit (IALU)
2. a 4-way multiplixer to select the addressing source
3. a "BREAK" system which controls conditional branching
4. a register to save an address to be used later (SAVIA)
5. the Instruction Address Register (IAR).

All these units are controlled by the program via the Instruction Bus (IB).

The IALU (SN74S181) is used to calculate instruction addresses. The operation code is set up in the IALO register. The code is converted from 3 bits to 6 bits to save space in the instruction word.

One of the inputs to the IALU is taken directly from the IAR. This makes address calculations relative to the present IAR content. Normally the IALO code will be A+1, which means that the IALU acts together with IAR to form a virtual address counter.

Jumps can be executed via two different paths: direct jumps bypass the IALU and are taken directly from the IB to the IAR through the MUX. In this way, direct jumps do not take longer than sequential addressing. Conditional jumps pass through the IALU input B, and are controlled by the "BREAK" system as described below.

An address can be saved any time in the SAVIA register by means of a single bit command and retrieved later by selecting the relevant MUX input.

The host computer has direct access to the IAR over the IAB-bus. This port is used to load programs or to initiate a routine.

To skip one cycle, the clock to the IAR can be inhibited by means of the single bit "NOP" command. The program will resume one clock period later.

6.4. The "BREAK" SYSTEM

The "BREAK" logic controls conditional skip and jump operations. In order to illustrate its principle, let us consider a simple routine which performs a tight loop in the ALU with three possible branches. The sequence, as it would occur in a conventional mini-computer could typically look like this:

1. set up necessary parameters
2. load working register with data word
3. perform operation
4. branch to 8 if result = 0
5. increment address register for the data
6. go to 9 if all done
7. return to 2
8. do subroutine, then return to 5
9. continue with next instruction.

For simplicity, let us also assume that step 3 merely consists of one single instruction, such as COMPARE, and that we do not store the result. Normally, most of the other steps within the loop require one instruction, which means that the overhead is more time consuming than the actual operation itself. Further, most of the instructions will need two memory cycles.

How many of these steps can in principle overlap in time? Obviously, step 3 is the essential operation. However, the address of the third data word could be calculated during the read out of the second while the first one is being treated in the ALU. Thus no real time is lost within the loop in fetching the data.

It remains to take care of the branch instructions, step 4 and step 6. For the latter, we need a counter which can be preset to a given value, and then decremented once per revolution in the loop. When it reaches zero, a conditional "BREAK" system could force the instruction address to be modified. Additional inputs to such a system could handle the conditional outputs of the ALU.

By now it should be obvious that in fact it is possible to overlap all the steps in the loop with the ALU activity.

The main parts which form the BREAK system are shown on Fig. 3. A conditional jump function is prepared and executed as follows: a mask is set up by the program in the conditional break register BR which selects one or several of the open collector gates which are marked <,=,> or W. The output of these gates are tied to the Instruction Address Bus IAB which normally contains all "0's". Whenever a condition becomes true, a "1" is placed on the relevant bit of the IAB; in other words, a number appears on the IAB and consequently at the B input of the IALU. If the code IALO=A+B, this number is added to the present instruction address, and a jump is executed. The magnitude of the jump will lie between 1 and 5, and depend upon the selection mask in BR as well as the condition which has occurred. Note that for a given condition the magnitude of the jump is programmable. In this way, one can compose multiway branching. For instance, one could choose to add 1 to the present instruction address when the wordcounter reaches zero, 2 if the "=" condition is met in the ALU, 3 for the "<" condition and stay with the same instruction if none of these conditions are met. The wordcounter empty condition (W) has been given priority over all other conditions.

Skip instructions can be composed in a similar way by setting the IALO code = A+B+1.

Without going into further details, the timing problem needs a few comments: the path from the input of one of the registers of the ALU to the IAR is longer than the clock period of 80 ns. It is therefore necessary to skip one strobe pulse to the IAR when a condition is met, in order to allow for the jump address to be evaluated in the IALU. This is controlled by the two flip-flops CKCO and ENBK. The latter is also used to enable the BREAK system so that one can leave the content of the BR if it is needed again.

When a condition has been met in the ALU, and a branch is executed, the content of the data address register, ALU registers etc. must be frozen, or else the return to the loop becomes cumbersome. Therefore, the clock to the ALU is inhibited within the cycle in which the condition was met. This is the task of the gate G3.

7. INTERFACE TO THE HOST COMPUTER

The processor is intimately connected to the UNIBUS, and the PDP-11 controls its state via a command register. To load a program, the instruction memory is made to look like a single register on the UNIBUS and the incoming 16 bit words are automatically assembled into sequential 48 bits instructions. Once loaded, the content of the instruction memory may be read back and checked.

Execution of a program is triggered by moving its start address to a virtual UNIBUS register, and the processor can be programmed to operate in a mode in which the data memory looks like part of the PDP-11's own memory. This means that results do not have to be transferred back to the PDP-11 memory for further processing.

8. EXTERNAL I/O AND FLAGS

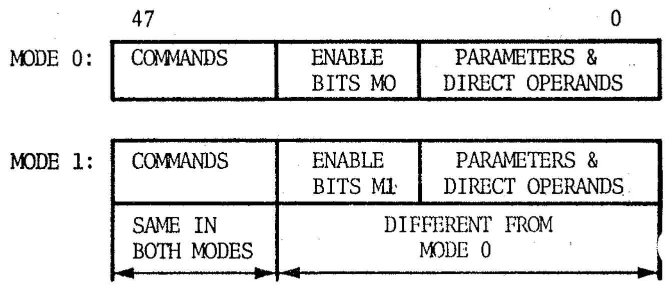
Data to and from an external device pass via the common data bus DB. The access to the DB, as well as the memory commands, are controlled by program. The device can either send the addresses through the "HIST BIN" port (Fig. 2) or use the internal addressing system. The transfer is synchronous with a cycle time of 80 ns or multiples thereof.

In order to start a given routine in the processor from an external device, a set of four FLAG inputs are connected to a priority register.(see Fig. 3). The outputs of the latter are connected to the BREAK system. Each flag will force the program register to a different address, from where the routines can be started. To make the system still more flexible, the lowest priority flag will enable a set of six gates on to the IAB (not shown on Fig. 3). In this way, up to 64 different peripherals can be handled directly at the expense of some additional external interfacing logic.

Obviously, the external flag system is program controlled. The priority register must be regularly checked to detect incoming signals. On the other hand, routines are normally short and executed at high speed, making the response time supportable in all practical cases. A genuine interrupt system would require an appreciable amount of hardware and most likely introduce additional time delays in the control paths of the processor.

9. PROGRAMMING AND SOFTWARE

The 48 bit instruction word is divided into three main groups: commands, enable bits and their associated parameter fields as shown below: (see also ref. 6).



As all parameters and direct operands do not fit into the framework of 48 bits, two different instructions are used which are distinguished by means of a single "mode" bit. Most of the commands are single independent bits and may be used at the same time. However, this does not always apply to the enable bits as sometimes their data fields overlap.

The programs for the processor are written in a symbolic language. A cross assembler which runs on a PDP-10 computer composes the 48 bit instruction words from individual micro commands and stores the result on disc. The programs are then transferred to the host PDP-11 computer via a direct link or paper tape. A resident loader in the PDP-11 transfers the binary program to the microprocessor for execution.

The assembler includes all the usual features such as symbolic addressing, labels, literals etc.

In addition to the assembler, a complete set of debugging and checking routines has been created.

10. PRESENT STATUS AND PERFORMANCE

The prototype was installed in one of the ERASME [1] measuring units in December 1974 and has been running continuously ever since without failure. Its task so far has been to generate four different histograms in real time. Actual timing measurements by means of an oscilloscope show that the prototype executes this particular routine 11 times faster than the PDP-11-45.

The corresponding program for the more powerful new version has been written and it is estimated that the gain will increase to around 20 when compared with a PDP 11-45, or 60 when compared with a PDP 11-20.

The fastest possible routine is a single loop instruction containing a multi-way branch. A typical example would be to search through the data memory for words which match a given sample. (In fact, the processor then simulates a content addressable memory). In this mode of operation the processor is about 35 times faster than a PDP 11-45.

Although further experience with various kinds of programs are needed before a good estimate can be found, it seems that the factor of 20 times PDP 11-45 can be expected for tasks which are reasonably suitable for the processor. Routines which include a lot of multiplications and/or divisions will run slower.

REFERENCES

1. C. Verkerk : "Special Purpose Processors", CERN - DD/74/27, September 1974 (contains extensive reference list)

2. A. Fucci, M. F. Letheren, F. H. Sumner, C. Verkerk, W. G. Vree : "Design of Hardware Processors for Use in High-Energy Physics" CERN - DD/74/32, November 1974

3. T. Lingjaerde et al. : "ERASME : A Scanning and Measuring System for Large Bubble Chamber Photographs", CERN - DD/71/19, 6 October 1971

4. J-C Gouache : "Description and Status Report of the Erasme System", Proc. of the Oxford Conference on Computer Scanning, 2-5 April 1974

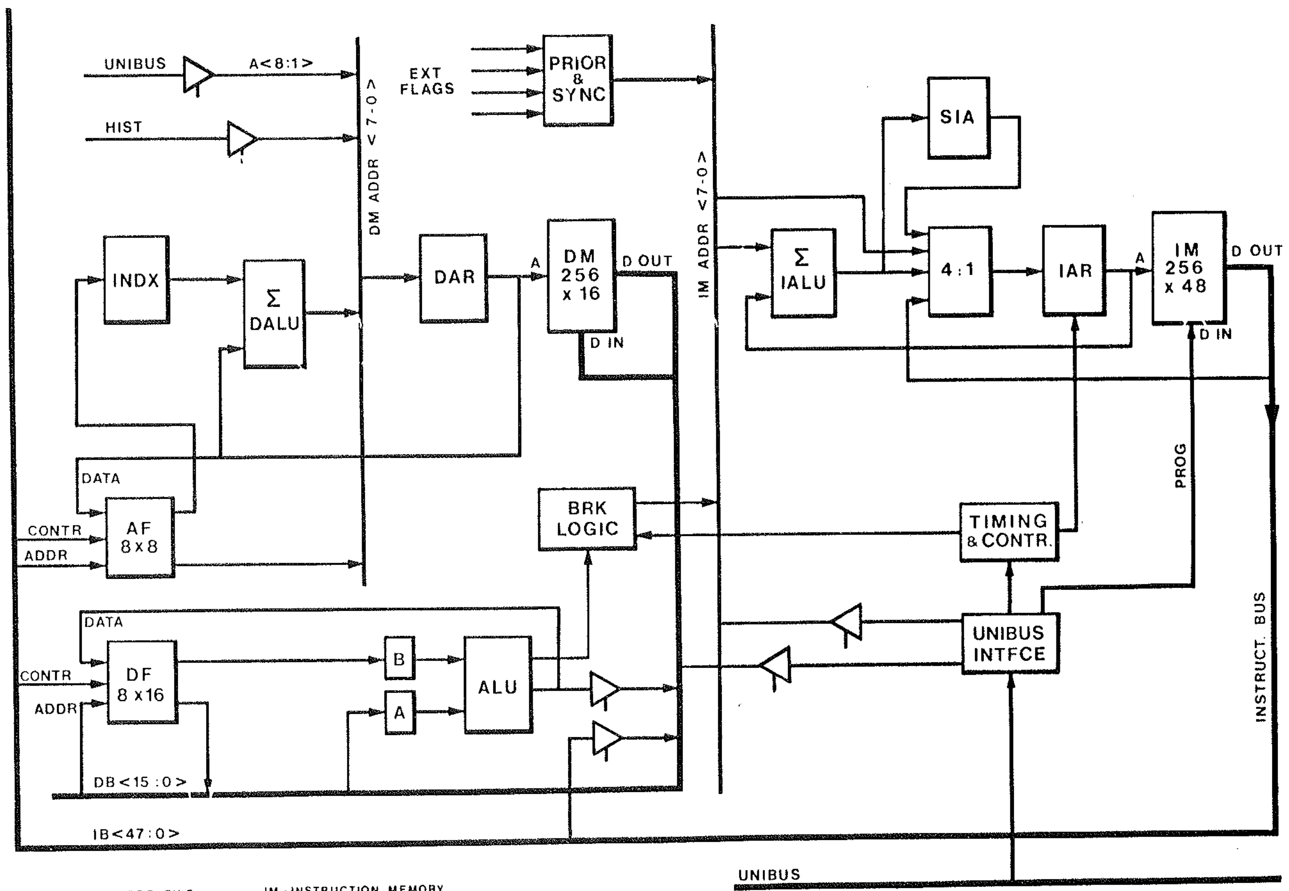
5. T. Lingjaerde : "A Fast Microprogrammed Processor", Proc. of the European DECUS Symposium Zurich 11 - 13 September 1974, pp 331-334

6. D. Myers : Software Documentation for the ERASME Microprocessor (Listings of instruction repertoire, assembler, loader and various routines, not yet published.)

ACKNOWLEDGEMENTS

The detailed construction and debugging of the hardware was carried out by D. Marland and the software package was written by Dr. D. Myers. The multiply/divide unit has been designed by C. Ljuslin.

I want to thank Dr. D. Jacobs for his valuable suggestions and finally Dr. D. Lord for his encouragement and support of the project.



AF: ADDR. FILE
DF: DATA FILE
DM: DATA MEMORY

IM: INSTRUCTION MEMORY
DB: DATA BUS
IB: INSTR. BUS

FIG. 1 - MICROPROCESSOR BLOCK DIAGRAM

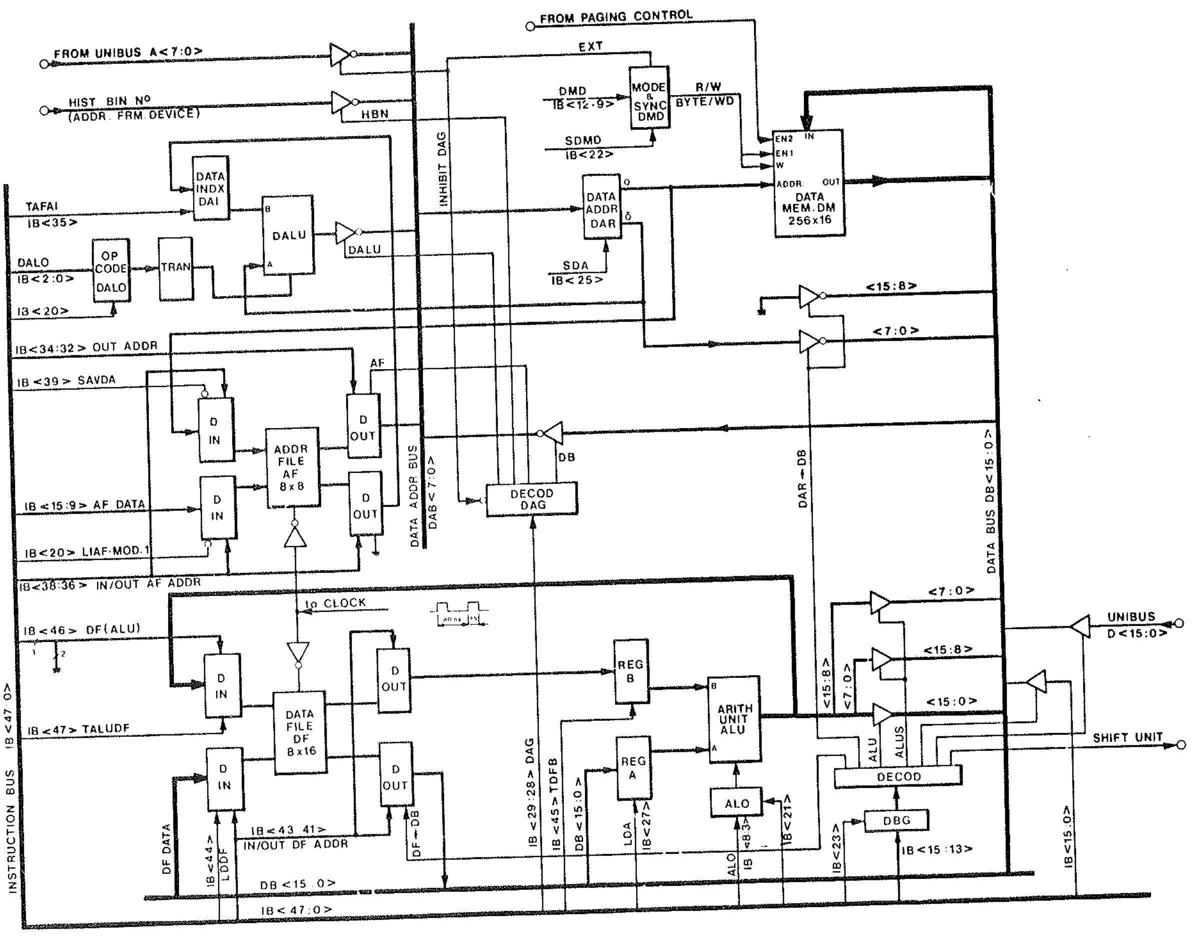


FIG. 2 - DATA MEMORY AND ARITHMETIC UNIT

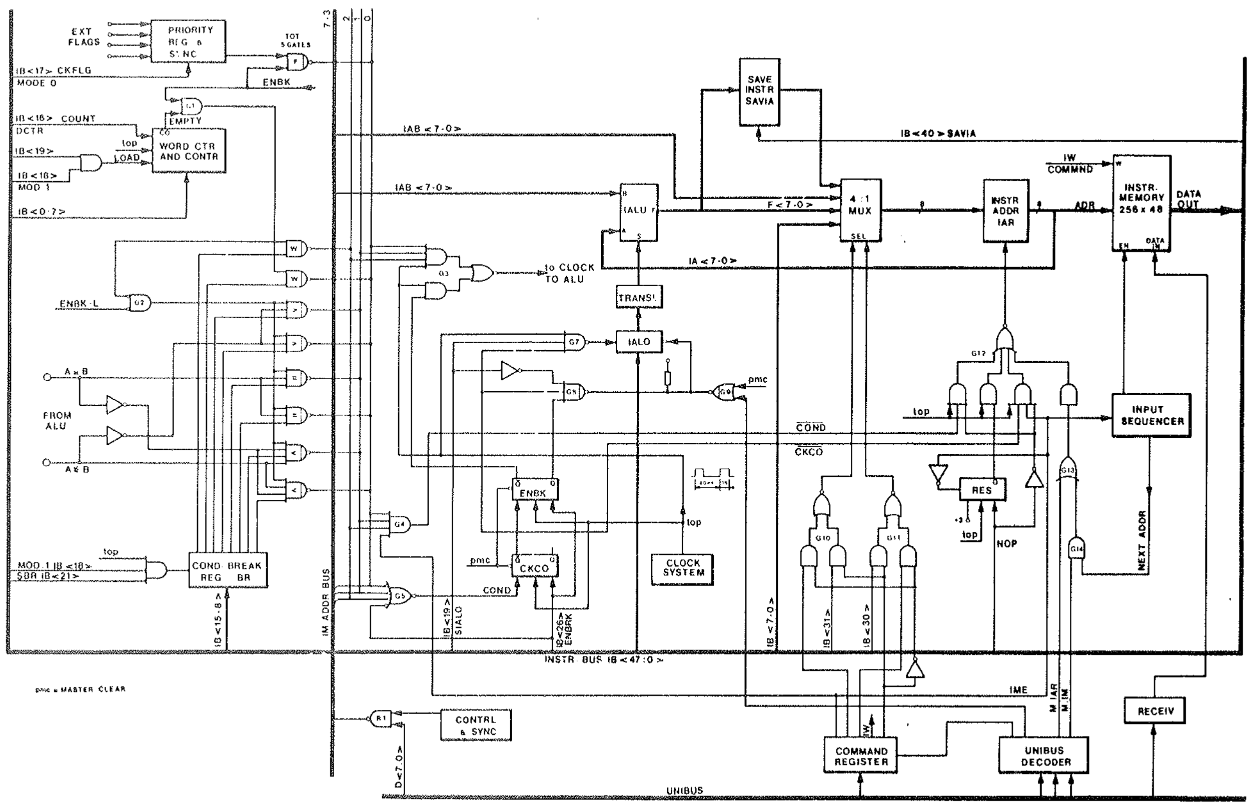
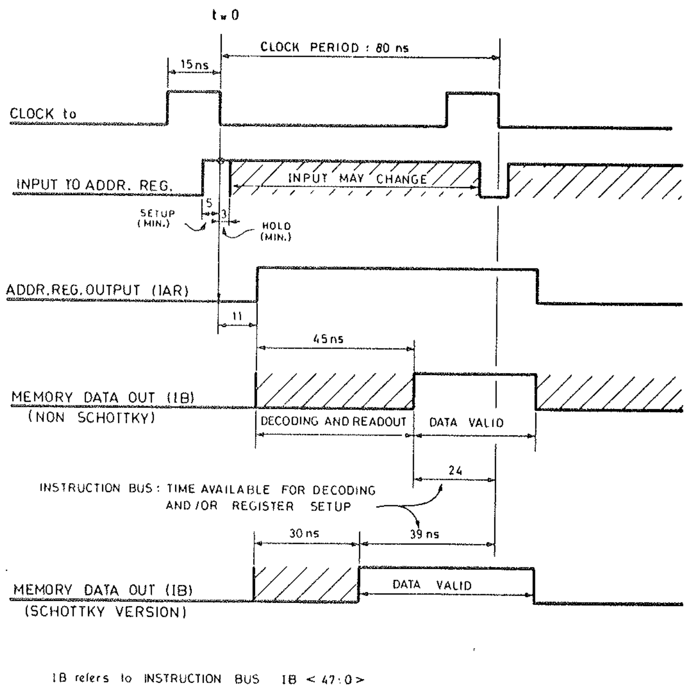


FIG. 3 - INSTRUCTION UNIT



IB refers to INSTRUCTION BUS IB < 47:0 >

FIG.4 MAJOR TIMING DIAGRAM