SLAC-PUB-1723
March 1976
(E/I)

THE LASS HARDWARE PROCESSOR*

Paul F. Kunz

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

ABSTRACT

The problems of data analysis with hardware processors are reviewed and a description is given of a programmable processor. This processor, the 168/E, has been designed for use in the LASS multi-processor system; it has an execution speed comparable to the IBM 370/168 and uses the subset of IBM 370 instructions appropriate to the LASS analysis task.

(Submitted to Nucl. Instrum. Methods.)

*Work supported by the U. S. Energy Research and Development Administration.

1. INTRODUCTION

In the interest of performing systematic studies of multiparticle final states, several large spectrometers have been constructed at CERN, BNL, and SLAC[1]. These spectrometers are capable of taking data at such a rate that the amount of computing time required for the data analysis is becoming a major problem. At SLAC,for example, the Large Aperture Solenoid Spectrometer (LASS) has the capability of recording events on magnetic tape at an average rate of an event every 10 milliseconds[2]. However, the mean time required for processing an event at the SLAC computer center[3] is of the order of 100 milliseconds. The goal of the LASS hardware processor is to preprocess the events so as to cut down significantly the amount of computer time required to support a LASS experiment. With the advent of large detectors and relatively inexpensive read-out electronics, the problem of computer support for the data analysis in LASS is becoming a familiar one faced by many experimenters in high energy nuclear physics[4].

Section 2 of this paper discusses the criteria imposed on hardware processing in general while section 3 reviews the components available for implementation. Section 4 describes the programmable processor designed for use in LASS. Finally, section 5 is a summary.

2. CRITERIA

In order to specify the criteria for the LASS hardware processor, a study was made of the data analysis task. It was quickly realized that the inherent structure of the task lent itself naturally to a multi-processor system since the overall task can be broken down into distinct sub-tasks such as unpacking the raw data, finding space points, finding line segments, etc. These sub-tasks generally try all combinations of a pair of coordinates in two planes and search

for a match with coordinates in the remaining planes using only simple basic operations such as addition, comparison, and multiplication. This search leads to a nested loop structure of the sub-tasks which accounts for the considerable amount of computer time required for execution.

It was also realized that most of the analysis task could be performed with the data in integer format. A word size of 16 bits yields a precision of 1 part in 65,000 which, for a least significant bit of 0.1 mm, allows a maximum detector size of 6.5 meters. This data format is sufficient for track reconstruction since the detector resolution is on the order of 0.5 mm and in most programs the comparison of predicted coordinates with measured coordinates has a window size significantly larger than the detector resolution.

The question remained, however, as to what set of criteria one should use in selecting the individual processors. The criteria used for LASS, outlined in the following sections, are similar to those one might use in other projects whether they be single or multi-processor systems.

2.1 Speed of Execution. The effective execution speed of the overall system must be about an order of magnitude faster than a large scale computer such as SLAC's IBM 370/168. Such speeds are difficult to achieve since the 370/168 has a cycle time of 80 nanoseconds and can do a memory to register ADD in 4 cycles.

2.2 Speed of Programming. An aspect frequently underestimated in hardware processing projects is the programming time. Since the intention is to duplicate in hardware a complex algorithm which normally requires a considerable effort in software on a large computer, the means by which one will understand, write, debug, and support the program in a hardware processor is an important consideration.

2.3 Flexibility. Program algorithms frequently change as one gains experience, encounters unforeseen problems, or changes the focus of the experiment in light of preliminary data. The programs may also change as new or modified detectors are brought into the apparatus, or different experiments are run on the same apparatus. It is important therefore that a processor's program can be easily modified.

2.4 Reliability. The processors and the system in which they are contained should be as simple as possible in order to achieve a high degree of reliability. A modular system would allow easy replacement of faulty modules or introduction of upgraded ones. The modules should be made of components which can be replaced, if faulty, by parts readily available from stock.

2.5 Speed of Fabrication. The fabrication time, which includes the time it takes to design, build and debug the processing system, must take into account the talents of the people involved in the project. To be practical one should make maximum use of technologies which are already well known.

2.6 Cost and Size. The cost and size are important criterion if one is going to have many parallel processors.

2.7 Compatibility. One would like to have a system which has maximum compatibility with existing equipment including the format in which the data is presented by the detectors and the physical configuration of the apparatus.

3. REVIEW OF AVAILABLE COMPONENTS

A frequently used approach to hardware processing is to build hardwired boxes with point to point logic[5]. It allows one to design an extremely fast processor. For example, one can do the calculation

$$X_p = a\ X_i + b\ Y_j$$

in the time it takes to do one multiplication and one addition by building two parallel multipliers and separate memory banks for X and Y. Furthermore, 16 by 16 bit multiplication time can be reduced to under 200 nanoseconds by using a large number of specialized integrated circuits. This approach has been made considerably easier in recent years with the availability of MSI and LSI integrated circuits. But since the program is effectively contained in the point to point wiring, it suffers from certain severe disadvantages. For example, the writing of a program takes a considerable logic design effort and the debugging or changing of the program usually involves rewiring sections within the box. Consequently, these processors take a long time to build and debug. Flexibility is limited when the algorithm one would like to use has been simplified in order to be implemented in hardware and only a limited range of program changes are allowed without a major reworking of the box. Also, reliability is impaired by the fact that the boxes are one of a kind and hence cannot be easily replaced and must be repaired by an expert when faulty. Cost and size of such processors may be reasonable but frequently they are not compatible with existing equipment. For the above reasons it was decided that hardwired processors were undesirable for a large spectrometer facility such as LASS.

The required fast effective execution speed can also be achieved by an array of programmable processors. There are many inexpensive programmable processors commercially available today which one might consider as elements in a multi-processor system. Since in recent years the cost of mini-computers has dropped considerably and their speed has increased, one might also consider their use in such a system.

In order to compare various processors, a study was made on the execution time of a simple DO-LOOP which frequently occurs in the data analysis task as the innermost DO-LOOP of many of the sub-tasks. Our studies show that for a space-point or line-finding subroutine, the repeated execution of this DO-LOOP can account for about half of the total execution time. The equivalent FORTRAN statements for the DO-LOOP studied are:

```
      DO  100  I=1,N
      IF  (X(I) .LT. XP)  GO TO 200
100   CONTINUE
      .
      .
      .
200   X1 = ...
```

In machine code the DO-LOOP consists of only four operations: 1) a COMPARE of a measured coordinate with a predicted coordinate in memory; 2) a BRANCH if the compare was low; 3) a DECrement on the coordinate index; and 4) a BRANCH to the top of the loop if one has not exhausted the coordinate list. Table I shows the execution time of this simple loop for various programmable processors. For each processor, the code was optimized in assembly language with 16 bit integer arithmetic. The approximate cost of each processor, relative to the Intel 8080, is also given.

The two popular MOS micro-processors suffer in this comparison because of their 8-bit word size, thus the LSI-11 has a clear advantage over them. The execution time of a typical mini-computer is represented by the PDP-11/40 while that of an advanced mini-computer with fast MOS memory by the PDP-11/45. The PDP-11's are both micro-programmed processors, so they also represent roughly the kind of performance one could achieve by designing a mini-computer

with an LSI bipolar micro-processor chip set such as the Intel 3000 series. The execution time on an IBM 370/168 is shown for comparison. One should bear in mind that to meet the real-time data rate of LASS one needs a system which is an order of magnitude faster than the 370/168. Thus in a system of parallel processors one would need 10 370/168's, 30 PDP-11/45's, 70 PDP-11/40's, 160 LSI-11's or 280 Intel 8080's. None of these options is within our budget and even if it were it is deemed extremely difficult to organize the interconnection of so many processors into a workable system.

Another aspect of this comparison of processors is the differences in their instruction set. For example, the 6800 matches the performance of the 8080, in spite of its longer cycle time, because it requires only 7 instructions, rather than 9, for the DO-LOOP. The LSI-11, PDP-11's, and 370/168 require only 4 instructions. In general, an average programmer can produce faster and more efficient code with a processor that has a more flexible instruction set. The IBM 370/168 certainly has the most powerful instruction set of all the processors considered, having 16 working registers which can be used either as accumulators or index registers.

None of the available inexpensive processors are fast enough nor do they have a sufficiently powerful instruction set. Consequently a programmable processor has been designed which would meet our needs. The remainder of this paper discusses the features of this processor.

4. THE LASS PROGRAMMABLE PROCESSOR: 168/E

The LASS hardware processors have been designed so that they are very fast, easily programmed, and relatively low in cost. Each processor has the execution speed comparable to an IBM 370/168 and, in order to minimize the programming task, the processors have been designed to efficiently emulate a

subset of the 370 machine instructions. Their low cost and high speed makes if feasible to build a small array of them (about 10) to meet our goal of executing at speeds an order of magnitude greater than the 370/168.

The processors, which have been given the name 168/E, are divided into three parts: a program memory 24 bits wide, a data memory 16 bits wide, and a processing unit. The separation of program and data memories, which allows simultaneous access to them, is an important feature for the speed of execution. Figure 1 shows a block diagram of the processing unit, and the following paragraphs discuss its various features.

The most significant bits of the program memory determine the control section within the processing unit which will execute the instruction, with data for the instruction in the remaining bits. The basic sections of the processing unit are the micro-processor slice array with its control logic, the branch control logic, and the data memory control logic.

The heart of the processing unit is an array of four bipolar LSI micro-processor 4-bit slices, the 2901[6]. As shown in fig. 2, it consists of an 8 function Arithmetic Logic Unit; 16 addressable registers with dual port read-out; an auxiliary register Q which is used for double precision shifts and multiplication; and a shifting network at the input ports of the register files. In addition there are status outputs to indicate CARRY or OVERFLOW conditions and ZERO or NEGATIVE results. The chips cost about $30 each in small quantities.

The micro-processor slice requires 18 bits of information to execute an instruction: 3 bits to define the source operands, 3 bits to define the function, 1 bit for the CARRY input, 3 bits to define the destination, and 4 bits each to define the two read addresses of the register file. These 18 bits are provided

by the data of the program memory. A latch is used between the program memory and the micro-processor slice so that the next program instruction can be fetched while the slice is executing the present instruction. The simultaneous fetch and execution feature of this processor is another of the important factors for its speed of execution.

External to the slice array is a 14 bit binary program counter. Its output is the program memory address so that the processor clock steps the processor sequentially through the program memory. A JUMP instruction is executed by a parallel load to the counter from the data of the program memory or the data output of the slice. A conditional BRANCH instruction is executed by placing the counter in the parallel load mode if the status bits of the slice match those in the program memory instruction. The programs for these processors can almost always be written in such a way that the BRANCH or JUMP addresses are known at load time, and this address can be part of the program memory data. Thus, a BRANCH or JUMP can be executed in one machine cycle.

A hardware multiplication has been implemented in the 168/E processors. It is done by momentarily stopping the program counter clock while cycling the slice through conditional ADD and SHIFT instructions. With this additional circuitry in the slice control logic, 16 by 16 bit multiplication time has been reduced to about 2 microseconds.

The address and data busses of the data memory are separate from those of the program memory. In order to allow efficient indexing of data memory addresses, the data memory address is formed by an ADD of bits from the program memory and from the output of the slice. Data may be written to the memory from the slice or from the program memory and similarly data may be presented to the direct input of the slice from the data memory or from the program memory.

Important aspects of the processor's structure will become apparent by comparison of the program code generated to perform the DO-LOOP described above. Table II shows the DO-LOOP implemented on the IBM 370/168 and the 168/E processor. The first instruction on the 370/168, cf. Table II, is a COMPARE between the contents of a memory location and register 0. The memory address is formed by the sum of register 9 (the index register), register 10 (the base register), and 12 bits from the instruction (the displacement ED). The 168/E performs the same operation in three instruction cycles. In the first cycle, the slice executes an instruction which places the sum of registers 9 and 10 at its output. In the second, the displacement from the program memory is added to the output of the slice and loaded to the memory address register. In the same cycle, the memory is switched to the read mode and the data are latched into the direct data latch at a time which overlaps the third cycle. In the third cycle, the comparison is made in the slice between register 0 and the direct inputs. As shown in Table II, the remaining three instructions on the 370/168 can be implemented in one cycle each on the 168/E. Thus one sees that the structure of the LASS processor allows it to emulate the IBM 370 efficiently. Emulation is possible because both processors have the same number of working registers and can perform the same arithmetic and logic operations.

Not all of the 370 instruction set can be emulated by the 168/E, but all those instructions needed for track reconstruction have been emulated. In fact, the IBM FORTRAN compiler requires about the same subset of 370 instructions as those implemented in the 168/E when dealing with 16 bit integer arithmetic. Those instructions not implemented deal with floating point, decimal arithmetic, character manipulation, system interrupt, and I/O. By not implementing all the instructions of the 370 one has reduced the cost and complexity of this

processor while increasing its speed. The goal of this project is to build fast programmable processors for physics applications and not to build a general purpose computer with the entire instruction set of the IBM 370.

One could have built a processor with its own unique instruction set, tailored to our needs. Instead, the 168/E is based on the architecture of the 370 for several important reasons. First of all, the writing and debugging of programs for the 168/E can be done on the 370/168 at the SLAC computer center. Once a program is running on real or simulated data, it can be easily translated to the instruction set of the 168/E. In fact, the translation is so simple that it can be done by a program written and executed on the 370/168. Secondly, programmers who are familiar with the 370 do not require any special understanding of the 168/E processor in order to produce fast and efficient code. In addition, they do not need to debug their programs on a hardware box with limited I/O capabilities.

Emulation of the 370 has greatly reduced the burden of programming the LASS processors. The question remaining, however, is how much emulation of the 370 has cost us in execution speed and the dollar cost of the processors. The cycle time of the 168/E will be 125 nanoseconds, which is slower than the 370/168, but, as Table I shows, the execution time of the 168/E is actually competitive with the 370/168. Fast execution of the 168/E comes mainly from the fact that BRANCH instructions can be done in one cycle. The 370/168 operates in a multi-programming environment so that it spends many cycles calculating the absolute address of the BRANCH. The basic DO-LOOP of Table I is biased in favor of the 168/E because of the two Branch instructions. A better comparison was made with a complete program which found line segments in a spark chamber system. The program was written on the 370/168 using 16 bit

integer data. With real data the execution time for the program on the 168/E should be about 10% slower than the 370/168.

The cost of the 168/E with program and data memories of 1K words each is less than $2000. This cost includes components, circuit board, and power supplies but excludes labor for assembly. About two-thirds of the cost is in the two memories. High memory cost is not as serious as it might seem because the routines which take long execution times are fairly small in size and the data set is also relatively small. The line segment program mentioned above, for example, was less than 500 words and a DO-LOOP such as the one used in Table I was executed over 2000 times in a typical event. The important point is that the speed-cost ratio of the 168/E is sufficiently high that a multi-processor system that meets our goals is economically feasible.

5. SUMMARY

The computer support for the data analysis in a high data rate physics experiment is becoming a familiar problem. The fact that the analysis task can be broken down into many simple sub-tasks and that the data can be represented as 16 bit integers has led many experimenters to thinking about using hardware processors. The processor should have, however, both high execution speed and programmability. Hardwired processors can be extremely fast but take a considerable effort to design and maintain. Commercially available programmable processors are either too slow or too costly to meet our needs, even in a multi-processor system.

The hardware processing system in LASS is based on an array of fast programmable processors. Each processor has the execution speed comparable to an IBM 370/168 and emulates a subset of the 370 machine instructions. The program for the processors can thus be written and debugged on an IBM 370

before translation and loading to the hardware processors. The task of programming the processors appears to be no more difficult than that of programming a large computer. Only a small number of processors are needed to meet the goal of executing at speeds an order of magnitude greater than a large computer such as the IBM 370/168.

ACKNOWLEDGEMENTS

I would like to thank D.W. G.S. Leith for his support and encouragement. Many useful discussions were held with several members of Group B at SLAC: B. Bertolucci, M. Gravina, D. Hutchinson, G. Oxoby, and S. Shapiro.

REFERENCES

1) A. Michelini, 1973 Int. Conf. on Instrumentation for High Energy Physics, Frascati, Italy, May 1973.

2) G. Armstrong, R. G. Friday, D. Hutchinson, K. Mauro, S. Shapiro, and S. H. Williams, IEEE Trans. Nucl. Sci. NS-20, No. 1 (Feb 1973).

3) The SLAC computer center consists of a triplex system with two IBM 370/168's and one IBM 360/91.

4) S. Dhawan, F. Kirsten, P. Kunz, R. Larsen, R. Thomas, L. Costrell, and D. Mack, Report of the NIM/CAMAC Executive Committee on Data Rate Requirements for Physics Applications, Nov 1975 (unpublished).

5) For an excellent review with a large bibliography, see C. Verkerk, "Special Purpose Processors," Proc. 1974 CERN School of Computing, Godøysund, Norway, Aug 1974.

6) This circuit is available from Advanced Micro Devices, Sunnyvale, California. It is also second-sourced by three other companies.

TABLE I

Comparison of Programmable Processors

Manufacturer / Model / Program Code	Intel 8080	Motorola 6800	DEC LSI-11	DEC PDP-11/40	DEC PDP-11/45	IBM 370/168	SLAC 168/E
COMPARE X_i and X_p	16.0 μs	16.0 μs	4.9 μs	2.5 μs	0.9 μs	0.32 μs	0.375 μs
BRANCH Low	5.0	4.0	3.5	1.4	0.5	0.24	0.125
DECrement i	2.5	4.0	4.2	1.0	0.5	0.08	0.125
BRANCH Greater	5.0	4.0	3.5	1.8	0.9	0.36	0.125
Total Time	28.0 μs	28.0 μs	16.1 μs	6.7 μs	2.8 μs	1.00 μs	0.750 μs
Relative Cost	1	1	1.2	14	36	3000	2

TABLE II

Comparison of Program Code Generated for 370/168 and 168/E

Program Step	Code for 370/168			Action of code for 168/E
COMPARE X_i and X_p	LOOP	C	0,ED(9,10)	Slice: Reg. 9 + Reg. 10 → Slice-Out
				Memory: ED + Slice-Out → MAR, and Data Memory → Direct Latch
				Slice: Reg. 0 - Direct (COMPARE)
BRANCH Low		BL	GOTE	Branch: If less than 0, GOTE → PC
DECrement i		SR	9,1	Slice: Reg. 9 - Reg. 1 → Reg. 9
BRANCH Greater		BNM	LOOP	Branch: If greater than 0, LOOP → PC

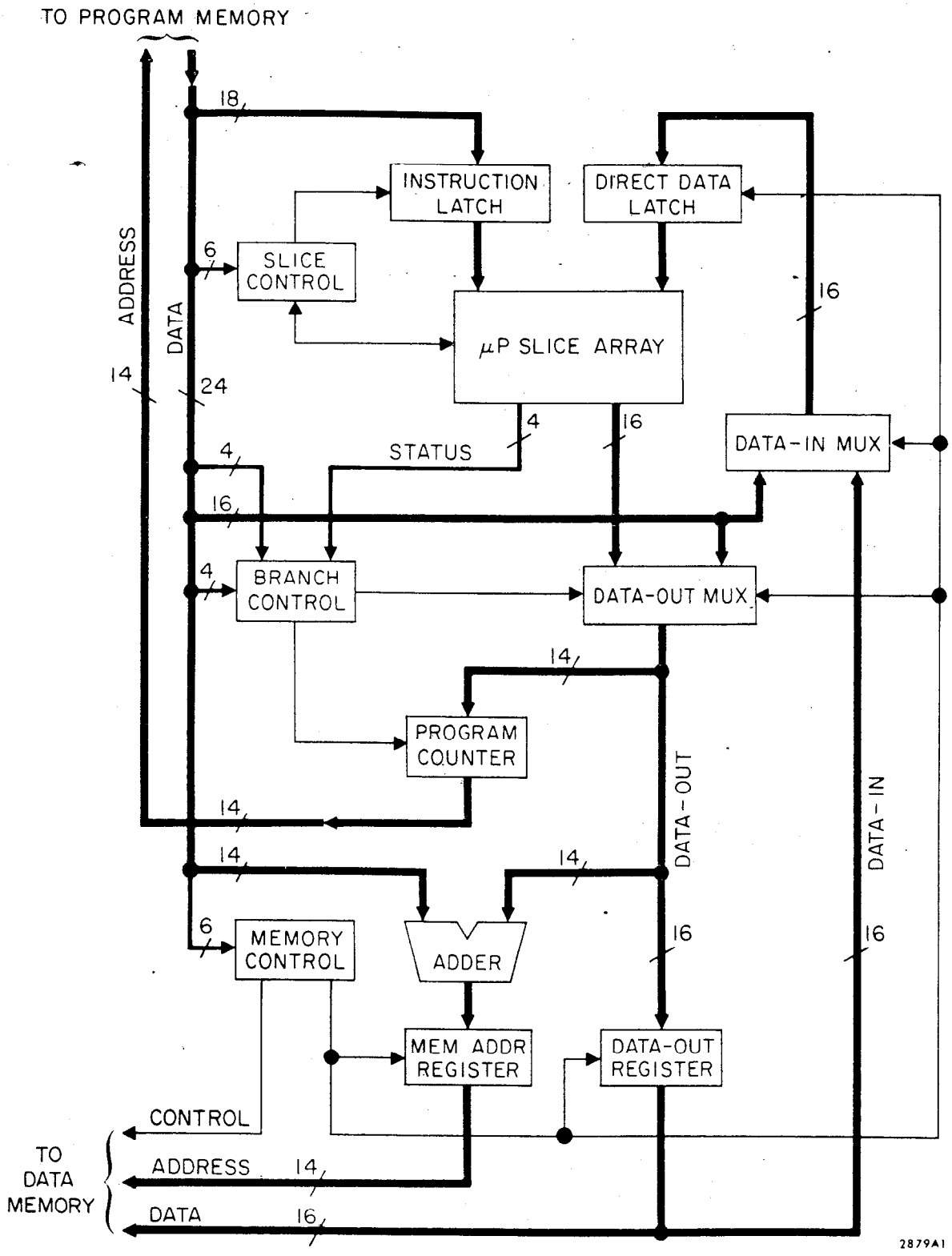

Fig. 1

Block diagram of LASS programmable processor, 168/E.

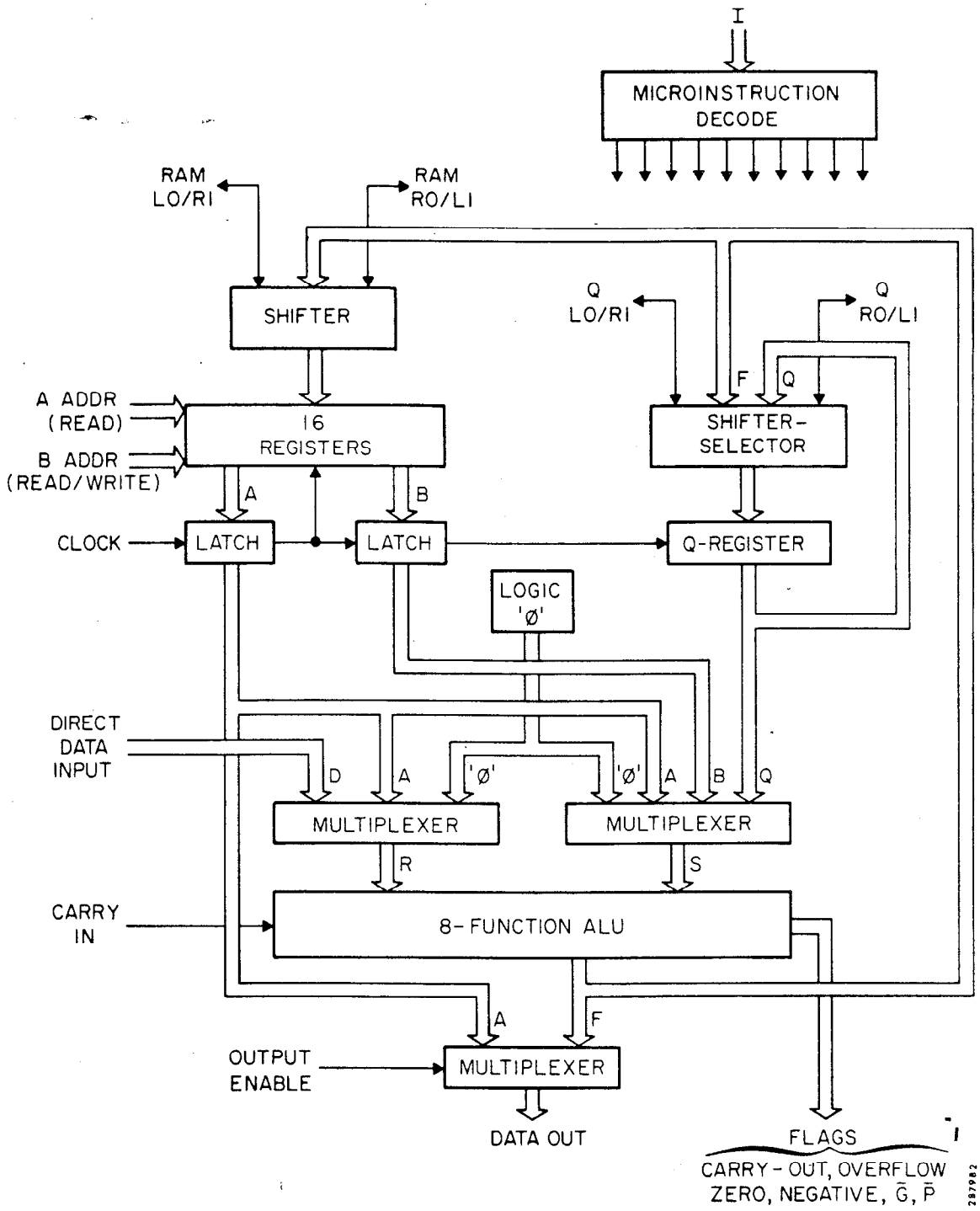

Fig. 2

Block diagram of the micro-processor slice: Am2901.