

Durham E-Theses

The design and construction of a microprocessor-based educational computer

Lumsdon, Bryan

How to cite:

Lumsdon, Bryan (1976) *The design and construction of a microprocessor-based educational computer*, Durham theses, Durham University. Available at Durham E-Theses Online:
<http://etheses.dur.ac.uk/9014/>

Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a [link](#) is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the [full Durham E-Theses policy](#) for further details.

ABSTRACT

Since digital computers came into widespread use in the early 1960's there has been the need to educate Engineers and Scientists in the areas of logic, machine organisation and programming. The computer described is aimed primarily at the teaching of machine organisation while offering useful facilities in the other areas. In the past, machines demonstrating computer architecture have been special purpose machines, they are now rather dated and have proved to be very expensive. The new machine uses a microprocessor to simulate the operation of an educational computer in which the content of all registers and states of gates are simultaneously displayed. The computer has an order code and architecture which is typical of modern small computers and has four modes of operation, viz:-

'Manual' in which individual parts of the machine can be manipulated manually by push buttons.

'One bit' in which the machine will obey an instruction one-step at a time and return to manual mode on completion of the instruction.

'One Instruction' when the machine will carry out the sequence of steps forming an instruction at a selected speed and will return to manual mode on completion of the instruction.

'Continuous' in which the machine obeys sequential instructions taken from the store until it is stopped manually or reaches a 'halt' instruction when it returns to manual mode.

While such a computer could be a specially built machine as in the past, use of a microprocessor reduces the display to a series of lamps and push buttons interfaced to the processor and the apparent operation of the computer is determined by the program held in Read-only memory. Thus changes in the architecture of the order code of the educational computer can be achieved by re-writing part or all of the program and it is therefore anticipated that the machine described will be capable of enhancement both easily and cheaply.

THE DESIGN AND CONSTRUCTION OF A MICROPROCESSOR-
BASED EDUCATIONAL COMPUTER

A Thesis submitted to the University of Durham for the
Degree of Master of Science

Bryan Lumsdon, B.Sc.

Department of Applied Physics
and Electronics,
Science Laboratories,
South Road,
Durham.

June, 1976.

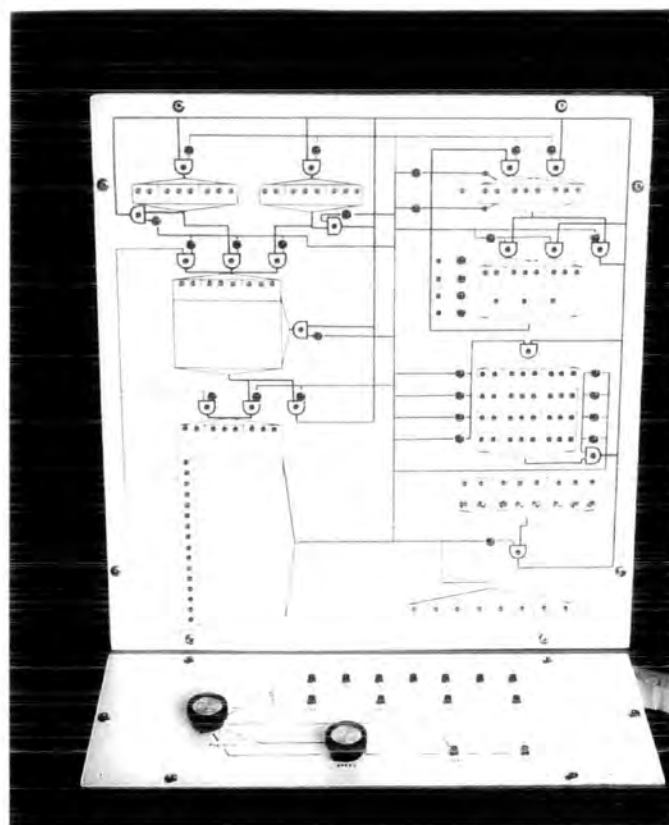


Plate 1. General View of the Educational Computer.

CONTENTS

	Page
Abstract	i
Contents	i
List of Illustrations	iii
Acknowledgements	iv
CHAPTER 1: Introduction	1
CHAPTER 2: Specification of the Machine	5
2.1 General	5
2.2 Front Panel Operation	7
2.3 Front Panel Indicators	8
2.4 Architecture of the New Machine	9
2.5 Consideration of an 8-bit, Direct Address Machine	12
2.6 Consideration of an 8-bit Computer having Seven Registers and Indirect Addressing	15
2.7 More Detailed Specification of the Computer .	17
2.8 The Order Code	19
CHAPTER 3: Hardware	21
3.1 Microprocessor 8008	21
3.2 Prototyping Unit SIM 8	24
3.3 The Processor Unit	25
3.4 The Interface	28
CHAPTER 4: The Program of the Microprocessor	32
4.1 General	32
4.2 Mode of Operation	36
4.3 Manual Operation	37
4.3.1 PANEL	37
4.3.2 DOUT	38
4.3.3 DIN	38
4.3.4 OPONA	39
4.3.5 LOGIC	39
4.3.6 INSTN	40
4.3.7 DECODI	40
4.3.8 STGATE	42
4.4 The Microprogram	42

Cont'd

	Page
CHAPTER 5: Conclusions and Review of Project	46
5.1 Speed	46
5.2 Possible Improvements in the Hardware	48
5.3 Cost	50
5.4 Two Recent Machines	51
5.5 Postscript to 5.3	52
APPENDICES	
Appendix 1: Listing of Microprocessor Program ...	53
Appendix 2: Some Sample Programs	69
References and Bibliography	73

LIST OF ILLUSTRATIONS

Plates:-

- 1 The Educational Computer
- 2 The Interface Unit
- 3 The Computer Board

Figures:-

- 2.1 Organisation of the Educational Computer
- 3.1 Instruction set of Microprocessor 8008
- 3.2 Timing diagram of Microprocessor 8008
- 3.3 Layout of Computer Board
- 3.4 Data flow - Computer Board
- 3.5 Table of bus pin connections
- 3.6 Part-circuit of Intel unit 8212 and symbol
- 3.7 Memory arrangement - Computer Board
- 3.8 Control logic - Computer Board
- 3.9 Layout of Interface Board
- 3.10 Organisation of Interface
- 3.11 Shift Register Detail
- 3.12 Panel Display Matrix
- 3.13 Logic Control Unit - Interface Board
- 3.14 Panel Switch Matrix
- 3.15 Bit-drive Amplifier for Display
- 3.16 Word-drive Amplifier for Display
- 4.1 Flowchart for Mode Selection
- 4.2 Flowchart for Subroutine 'PANEL'
- 4.3 Flowchart for Subroutine 'DOUT'
- 4.4 Flowchart for Subroutine 'DIN'
- 4.5 Flowchart for Subroutine 'OPONA'
- 4.6 Flowchart for Subroutine 'LOGIC'
- 4.7 Flowchart for Subroutine 'INSTN'
- 4.8 Flowchart for the decode and display sections
 of the Subroutine 'DECODE'
- 4.9 Flowchart for Subroutine 'STGATE'
- 4.10 Microprogram, states 1,2,3,4
- 4.11 Flowchart for microprogram
- 4.12 Flowcharts for microprogram states 2 and 3
- 4.13 Microprogram for state 1 (following display
 section of 'DECODE')

ACKNOWLEDGEMENTS

It is with great pleasure that I acknowledge several sources of help and encouragement in this project. First and foremost I thank Dr B. J. Stanier who suggested and supervised the project and who could be relied upon for lively and critical discussions. His help and guidance are greatly appreciated.

My thanks are due to Professor D. A. Wright who made available the facilities of the Department of Applied Physics and Electronics for the work to be undertaken and to the Rector, Dr M. Hutton, and my Head of Department, Mr J. H. Jones, who provided time free of other Polytechnic duties for this research activity.

I am grateful to my colleague, Mr M. Todd, who developed a cross-assembler program which proved invaluable in the development of the microprocessor program and produced the listing of Appendix 1.

Finally, my grateful thanks are extended to Miss E. Moss for her tolerance and excellent typing in the final presentation of this thesis.

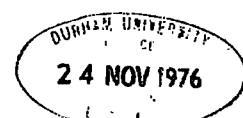
B. Lumsdon,
Department of Electrical, Electronic
& Control Engineering,
Sunderland Polytechnic.

CHAPTER 1:

Introduction

The Department of Applied Physics and Electronics has long been interested in the teaching of digital electronics and its extension in the organisation of Digital Computers. To this end, the design and construction of an Educational digital computer was undertaken in 1966 by R. Brumskill and the M.Sc. Thesis describing that machine was published in 1969. (Ref. 1). Other machines existed before that date, notably the Abacus Computer, initially designed by Elliott Bros. and subsequently adopted by Feedback Ltd. This machine is still available at a current price of about £4,000 but, because of its early design, uses serial arithmetic processes and its architecture and order code are rather dated.

The design undertaken in 1966 was an attempt to produce a cheaper and more modern machine. At that time, T.T.L. integrated circuits were becoming widely available and it was anticipated that a cheaper machine could be produced and a more modern architecture adopted at the same



time. In the event the machine was not exploited commercially because its cost/benefits were not sufficiently favourable to displace the established machine.

In 1972 as a result of work in the calculator field, Intel Corporation of the U.S.A. introduced a 4-bit general purpose micro-processor unit and followed this fairly quickly with an 8-bit unit, the '8008'. As with integrated circuits in the mid-1960's, it was anticipated that widespread use of the new devices would result in significant cost reductions and indeed this has proved to be the case, the unit cost of the 8008 having fallen from about £70 in 1973 to £20 in 1975.

It was appreciated by University staff in 1972-73 that use of such a device would permit a fresh approach to the Educational Computer with every chance that the result would be both cheap and modern and with the further advantage that because the architecture of the new machine would be determined by the microprocessor program, it could be modified and updated simply by writing a new program.

In the course of discussion between Drs. Morant and Stanier of the University and the author in the early part of 1973, it was decided that such a development would be undertaken and work was begun in October 1973.

No attempt was made initially to assess the relative merits of the 4- and 8-bit processors 4004 and 8008 in this application. It was obvious that the 8008 would be of more general use within the Department for undergraduate laboratory work and other postgraduate projects than the 4004 and therefore this project was also based on it. Since the

computer which has been produced is observably slow, it is clear that use of the 4004 which would have required a longer program, would have resulted in an unacceptably slow machine.

The early part of the project was concerned with establishing the architecture of the teaching machine:-

a) The Abacus computer by Feedback and the 1966 Durham Computer were examined and compared in their structure and order code. Programs were written and run on the Durham machine. At the same time commercially available machines, notably the Argus 600 by Ferranti (Ref. 3), PDP8e and PDP11 by Digital Equipment Corporation (Refs. 4, 5) and SPC 16 by General Automation Inc. (Ref. 6) were examined with a view to discerning the trends in modern small computer design.

b) With the additional constraint that the word length should be as short as possible, designs based on a single working register were considered and programs written in order to test the usefulness of the proposed order codes. By the end of 1973 it was clear that only a very limited machine could be built on that basis and a multi-register approach with indirect addressing of store was formulated. The architecture adopted thus resembles the later machines PDP11, SPC16 and the Intel microprocessor rather than the earlier single-address, single register types.

January to June 1974 was devoted to the design and construction of the computer panel and the interface to the microprocessor prototyping unit SIM - 8 which is a general purpose unit supplied by Intel and incorporating a microprocessor 8008 together with Read-only memory (R.O.M.), Random Access memory (R.A.M.), logic necessary for controlling the 8008, two input and four output ports.

With the interface working correctly, the program was written over the period June 1974-June 1975; the final program contains 927 8-bit words and therefore occupies 3½ Programmable Read-only memories, type 1702. The period of twelve months to write a program of 1000 words may be considered rather long but it is in keeping with the general industrial experience that Software costs are surprisingly high and often greater than the hardware cost. In this case it was found that the program could be simplified in parts if the display interface was re-arranged and this was generally done. With experience of writing program in the machine code, it became possible to write more efficient code and consequently sub-routines which were written in the early part of this period could be re-examined later and shortened without loss of any essential features. In some cases, the improvement represented a 50% saving of instructions.

With the program complete it only remained to make the Educational Computer independent of the commercial item SIM-8 by designing and constructing a circuit board containing the microprocessor 8008 and the minimum necessary R.O.M., R.A.M. and control logic. Since this was a prototype circuit, various monitoring and control facilities were also included which would not be necessary in a production model. This circuit was completed in July 1975 and when augmented by a purpose-built power supply designed and constructed by the Electronics Workshop the project was complete in September 1975.

CHAPTER 2:

Specification of the Machine

2.1 General

Certain essential and other desirable features of a teaching machine were enumerated at the time of the design of the 1966 machine and since these are still largely applicable today they have been incorporated in the present machine. These features with some observations are:-

- i) The teaching machine must be a computer rather than a complex logic trainer.
- ii) The operation of the machine must be demonstrable in much greater detail than is possible with commercially available computers. This implies a large display panel with a comprehensive mimic diagram and lamps indicating the flow of information and states of all major control gates. Push button switches should provide for manual operation. The further suggestion at that time that the logic circuitry should be accessible for detailed examination is simply not possible in a simulated machine of the present type.

- iii) The machine should be capable of step by step operation over a range of speeds.
- iv) It should be cheap enough for purchase by University and College departments. The upper price limit in 1969 was thought to be £2,000. In the event this aim was not met and presently it is felt that a marketing price of not more than £1,000 would be necessary to ensure commercial acceptance. This implies a cost of components of about £300.
- v) The educational aspect should be borne in mind throughout the project and instruction manuals produced for the machine.

Also intended at that time was the incorporation of the then new integrated circuits, printed wiring, an up-to-date architecture and some provision for extension at a future date. It is now generally recognised that the preparation of a printed circuit is not commercially justified unless several circuits of a particular type are to be made and since only a prototype is involved here, the more appropriate method of construction using Veroboard was used.

Integrated circuits are no longer novel, they are used as a matter of course. The various technologies are all used side by side, this being aided by the general tendency of manufacturers to make their various units T.T.L. compatible. In this particular project the micro-processor and Read-only memory are P-channel M.O.S. devices, the Random-access memory is N-channel silicon-gate M.O.S., general logic is T.T.L, the more complex M.S.I. units being used wherever possible while some of the Intel units designed specifically for interfacing to microprocessors are Schottky T.T.L.

Since the prototype was built, the range of Complementary M.O.S. devices has increased and the price has fallen. It is therefore likely that a new design would use these elements instead of T.T.L.

The aims of an up-to-date architecture and the provision of spare capacity to allow 'stretch' remain valid. These topics are dealt with fully in the later sections.

2.2 Front Panel Operation

The 1966 machine had a total of 58 push-button switches of which some 40 were actually used. Mounted on the consoles at the foot of the panel, they were linked with the panel by reference numbers, so that, if one wished to clear a register it was necessary to trace the 'clear' line of the register to the edge of the panel where it was numbered and then to locate and operate the push-button of corresponding number. With experience, the buttons which were frequently required came to be remembered but few students would spend long enough with the machine for this to occur.

An attempt has been made to improve this situation in the new machine by locating most push-buttons on the panel itself alongside the gates which they control. This has not been possible in every case and the console at the foot of the panel still carries eleven push-buttons but the remaining 24 have been dispersed to their functional positions. In the course of development it was realised that an operator would only have two hands and those of limited span, so that the number of push-button switches which could be simultaneously operated was limited. The final design only requires two push-buttons to be operated simultaneously and to achieve this it was necessary to imply the presence of memory in two of the computer functions where none need exist in a real computer. These functions are:-

- a) The Arithmetic and Logic Unit where the required operation is selected by push-button or instruction and is remembered by the unit until a new operation is selected.
- b) The store address which could simply be gated from one of three registers is, instead, remembered in a store-address register.

2.3 Front Panel Indicators

The traditional logic indicator on earlier teaching machines and logic tutors was the filament lamp. This has four main disadvantages:-

- a) Heavy power consumption, approximately 1 W per lamp so that a panel of 80 lamps all lighted consumes some 80 W.
- b) Following from (a) a lamp driving transistor capable of switching perhaps 40 mA at 25 V must be provided for every lamp.
- c) Low reliability. The failure rate of incandescent lamps is very much greater than that of integrated circuits and other electronic components in properly designed circuits.

Therefore it is likely that the most common fault in a teaching computer will be the failure of the indicator lamps. This has certainly been the case with the 1966 machine.
- d) High cost. The lamp itself is quite cheap but it is generally mounted in a bezel with a coloured lens which makes the total cost high.

The alternative, available since about 1972 is the Light Emitting Diode (L.E.D.). These were initially of interest to the Electronics Industry because of their high reliability (of the same order as that of a transistor) and their low power dissipation. (1.8 V., 20 mA).

Their initial high cost has fallen so that the cheapest have now a lower cost than a 1 W incandescent bulb. A type with consistent and adequate light output now costs about 30p and at this price although more expensive, is preferable to a filament lamp as an indicator. At the time that the display panel was designed and constructed only red LEDs were suitable, other colours were available but they were expensive and inefficient (i.e. lower light output and a larger current). The main disadvantage of the LED is that the light output is low compared to a 1 W filament lamp and most LEDs are fitted with an integral lens which projects the available light forward through a limited angle. The result of this is that a display panel using these lamps must be viewed from the front and high ambient light avoided.

The panel must also be covered at the back to prevent light entering the LEDs from the back and giving the impression that the lamps are on.

2.4 Architecture of the New Machine

The machine designed and constructed in 1966-69 has been in regular use in the teaching laboratory of the Department of Applied Physics and Electronics. Its performance in this role has been satisfactory and there was no feeling that it was old-fashioned and due for replacement, but rather that the microprocessor had so sufficiently changed the situation that a new machine based on it could be cheap enough to find widespread application. Nevertheless there were certain aspects of the machine that it was felt could be improved.

- a) Word length. This was originally chosen to be 12-bits so that the multiplication of two numbers, one of 4-bits or less, the other of

7-bits or less could be demonstrated. Due to the high cost of display it was felt that a shorter word length should significantly reduce the cost and at the same time still demonstrate arithmetic processes adequately. Thus an aim of fairly high priority in the new design was a word length of 8-bits.

- b) Order Code. In Memory Reference Instructions four bits of the word were used to specify the instruction, the remainder specified the address but in machine-operating instructions these extra bits were often unused. This led to a rather inefficient order code which lacked some of the instructions which one would expect to find, e.g. an unconditional jump or a logical operation. It can be noted in passing that the order code of this machine is similar to that of the 'Abacus' teaching computer and of the same general structure as the PDP8. By microprogramming of the machine instructions, the PDP8 is enabled to have a much wider range of instructions than either of the other machines.
- c) Input/output instructions. Since these instructions were never implemented on the computer it was impossible to demonstrate any real-time program.
- d) Modifier. An event of great significance in the development of the early computers was the introduction of the 'B-line Modifier' since it greatly facilitated the access to sequential store locations and thus simplified list-processing. While the name no longer appears in computer literature, the function is still provided by some form of auto-indexing register which can be used to address store. A modifier register was provided in the 1966

machine and instructions provided for loading it and program branching dependent on its state. Experience with the machine has shown however that little use was made of this feature and little importance was attached to its implementation in the new machine.

- e) Sub-routine linkage. All commercial machines have some provision for linking sub-routines to a main program. The means adopted can be basically either hardware or software although generally there is some of each. In the former case a common arrangement is for the computer to have a push-down stack (or last-in, first-out address register) in which the current address is held in the first location; when a 'Call sub-routine' instruction is obeyed, the first address of the sub-routine is pushed on to the stack and all addresses previously in the stack move down one level. Since the stack must be finite, the lowest address is lost.

In a software implementation, a common system provides for sub-routine return addresses to be held in a reserved part of the store. A stack-pointer indicates the next vacant location in this area.

When a sub-routine is called, the address of the next instruction of the current program is stored in the location addressed by the stack pointer, the stack pointer is incremented and the sub-routine starting address is loaded into the program counter. On return from the sub-routine the stack pointer is decremented and the content of the addressed location is loaded into the program counter.

While no sub-routine provision was made on either the 1966 or the Abacus machines, serious consideration was given to the desirability of providing a single level of sub-routine on the new machine. It was finally decided that it should not be provided since the machine

would inevitably be slow and the programmes which experience had shown were used were quite short and not generally sufficiently complex to require sub-routines.

It can be noted that when program is held in random-access memory, sub-routines can be implemented by loading the return address into the final return jump of the sub-routine before entering the sub-routine. This system, while tedious, could be used if a sub-routine was considered essential in some particular program.

2.5 Consideration of an 8-bit, direct-address machine

With an 8-bit machine based on the architecture of the PDP8 or the previous educational computers, some of the 8-bits are used for the instruction, the remainder for the address. Thus if two bits are used for the instruction the remaining 6 bits allow 64 (2^6) memory locations to be directly addressed. If three bits specify the instruction, 32 memory locations can be directly addressed. The only reasonable compromise between an adequate instruction repertoire and adequate direct address field is thus 3:5.

The address field could be extended by a separate Page-address register of 4 bits which would be altered in value by an operating instruction and would provide a total store of 16 pages each of 32 locations or lines. The line directly addressed by the memory reference instruction would be on the page currently addressed by the page address register.

If the input/output instructions were small in number so that no peripheral address need be incorporated in the instruction, then the three bits for the instruction would provide a repertoire of seven memory reference instructions and a group of 17 operating instructions.

The memory reference instructions could be:-

<u>Code</u>	<u>Action</u>
0	Transfer the contents of the accumulator to the specified store location.
1	Add contents of specified store location to the accumulator.
2	Subtract contents of specified store location from the accumulator.
3	Multiply content of the specified store location by the accumulator.
4	Divide content of the specified store location by the accumulator.
5	Form logical AND between specified store location and the accumulator.
6	Jump to specified line of page specified by the succeeding byte.

Of the possible 32 operating instructions specified by code 7, half would be absorbed by the literal instruction 'load the page address register with ...,' leaving 16 for other purposes. These would have to include instructions of the following types:-

Skip the next two instructions.

Shift accumulator right.

Shift accumulator left.

Complement accumulator.

Input to accumulator.

Output from accumulator.

While this system is based on the structure often employed in commercial computers of the 1960's it incorporates two types of instruction which did not generally appear in small computers of that time.

- a) The 2-byte jump instruction; the first byte specifies 'jump' and the line number, the second specifies page number. The multi-byte instruction is much more common in small computers which have become available in the 1970's.
- b) The literal instruction 'Load page address register with ...'
This again is much more common in the later designs and while it is only a single length instruction in this case, it could equally be a double length instruction if one wished to load an 8-bit number.

When this code was postulated it was intended that the machine would use signed-binary notation in its arithmetic processes, test programs of the type which had been used on the earlier computer were written in order to test the code.

The difference between this and the earlier machine was in the word length and while previously the square root of a number up to 2048 could be calculated, the limit now was 128 and the answer was limited to the range 1 to 11; a rounding error in the final bit would result in an error of at least 9%. In view of this reduced accuracy it was felt that the machine should be programmable for double-length working and in trying to achieve this it became apparent that signed-binary notation was incompatible with this aim. Two solutions were considered, the first that the machine should have a concealed pre-selector switch to select signed-binary or two's complement was rejected because it was felt that confusion would be caused when the switch was found to be in the wrong position during the course of a demonstration. The second, that the order code be extended to deal with both types of notation was rejected because it would result in a large proportion of the possible instruction codes being used, thus precluding further development and furthermore, that many of the orders would be obscure.

2.6 Consideration of an 8-bit computer having seven registers and indirect store addressing

If the need to specify a store address can be removed from the instruction, then an 8-bit machine immediately has a possible repertoire of 256 instructions. There is still the need of course for some register which addresses store and which can be set up to point to any particular location; this will be called the 'store-addressing register.' Thus the memory access system proposed is such that when an instruction specifies 'fetch from store' or 'put in store' the location used will be determined by the value of the store-addressing register. Use of an 8-bit register in this application provides a total store of 256 words which was considered to be quite adequate.

Other specific registers which are needed are the Program Counter, and an Accumulator. Since access to the store has been made more difficult it is necessary to make it less frequently if economy of program code is to be maintained; this means that there must be a series of general-purpose registers to which access is easily obtained and which can hold data that is currently required. If data is to be transferred between these registers and to and from store, a certain number of bits is required in the instruction to specify the source and destination registers. Allocation of two bits to this purpose only allows accumulator, store, store-addressing register and program counter to be specified and the system is minimal and very inflexible. If four bits are allocated, all 256 instructions are used for inter-register transfers leaving none for any other purpose. Thus a total of eight registers requiring three bits to specify each is the only possible compromise and they consist of the four already listed plus an extra four forming a scratch-pad memory.

When six of the eight bits have been used to specify the source and destination registers, 64 of the possible 256 instructions have been allocated. The remainder can be divided into those specifying a single register and machine instructions independent of the registers. Examples of the former are 'Clear register X' and 'Add contents of register X to the accumulator' while examples of the latter are 'Halt' and 'Shift accumulator one place right.'

If all the remaining instructions were of the former type there could be a total of 24; if that number were reduced to 16 there would be codes available for 64 machine instructions of the latter type.

It is also apparent that two-byte literal instructions of the type 'Load register X with ...' could also be provided.

The wide range and large number of instructions available with this type of architecture were extremely attractive: the problem of number notation could be solved simply by providing both; a general purpose register could be made auto-indexing so that a modifier register would be provided; by associating an adder with the program counter, program-relative jumps could be provided. Multiplication and division could be implemented with single length machine instructions if two of the general registers were assigned to the task.

As a result of these considerations it was decided to adopt this architecture and the remainder of the project was concerned with its implementation. In the course of this the detail of the computer gradually crystallised, certain features which had initially seemed desirable were discarded and replaced by others, some features which were quite feasible were not implemented because it was felt that the resulting complication would detract from the merit of the machine as a teaching aid and a large block of instruction codes were reserved for future extension.

2.7 More detailed specification of the computer

The machine has eight registers which are addressed by number:-

0	Accumulator
1,2,3,4	General purpose registers
5	Store-addressing register
6	Program counter
7	Store, addressed by register 5.

With the exception of the store, each register can be cleared, complemented, incremented and decremented.

Data is transferred between the registers by means of the Data-bus; each register being gated so that its data can be impressed on the bus and also so that it can accept data from the bus. Also gated to the data-bus is a set of eight toggle switches which represent one of the machine peripherals and are known as 'input 0'.

Working in conjunction with the accumulator is the 'Arithmetic and Logic Unit' (ALU) which requires two sources of input data; the first of these is the accumulator and the second is the data-bus. The result of the ALU operation is loaded directly into the accumulator replacing the data originally held there. The ALU is capable of four operations:-

Add	The two 8-bit words are added in 2's complement format
Subtract	The data from the bus is subtracted from the accumulator data in 2's complement format
AND	The logical AND between the data words is formed
OR	The logical OR between the data words is formed

The operation being performed by the ALU is indicated by a lamp which remains set until changed to another function for the next operation. This 'memory' is not functionally required here but manual operation would be impossible without it.

The ALU contains three flags which are set according to the result of the ALU operation, they are:-

Carry	If there is an overflow from the most significant bit
Negative	If the most significant bit is 1
Zero	If all eight bits are zero

The carry flag is also affected by the shift instruction as detailed in the order code listing in Section 2.8.

The store can be addressed from registers 5 & 6 (Store-addressing register and Program counter) and, in manual mode only, from input 0. A store address register is provided which accepts data via a gate from each of these three sources and maintains its value until new data is gated to it. In the case of semiconductor memory it is not necessary to hold the address in this way but this is another case where manual operation would be impossible without it.

As well as being gated to the data-bus, the store output is gated to the 'Instruction register and Decoder' where instructions are decoded into one of sixteen types. In manual mode only, the instruction register can be set up from the switches, input 0.

The general organisation of the processor is shown in Figure 2.1 while Plate 1 shows the general appearance and layout of the prototype machine.

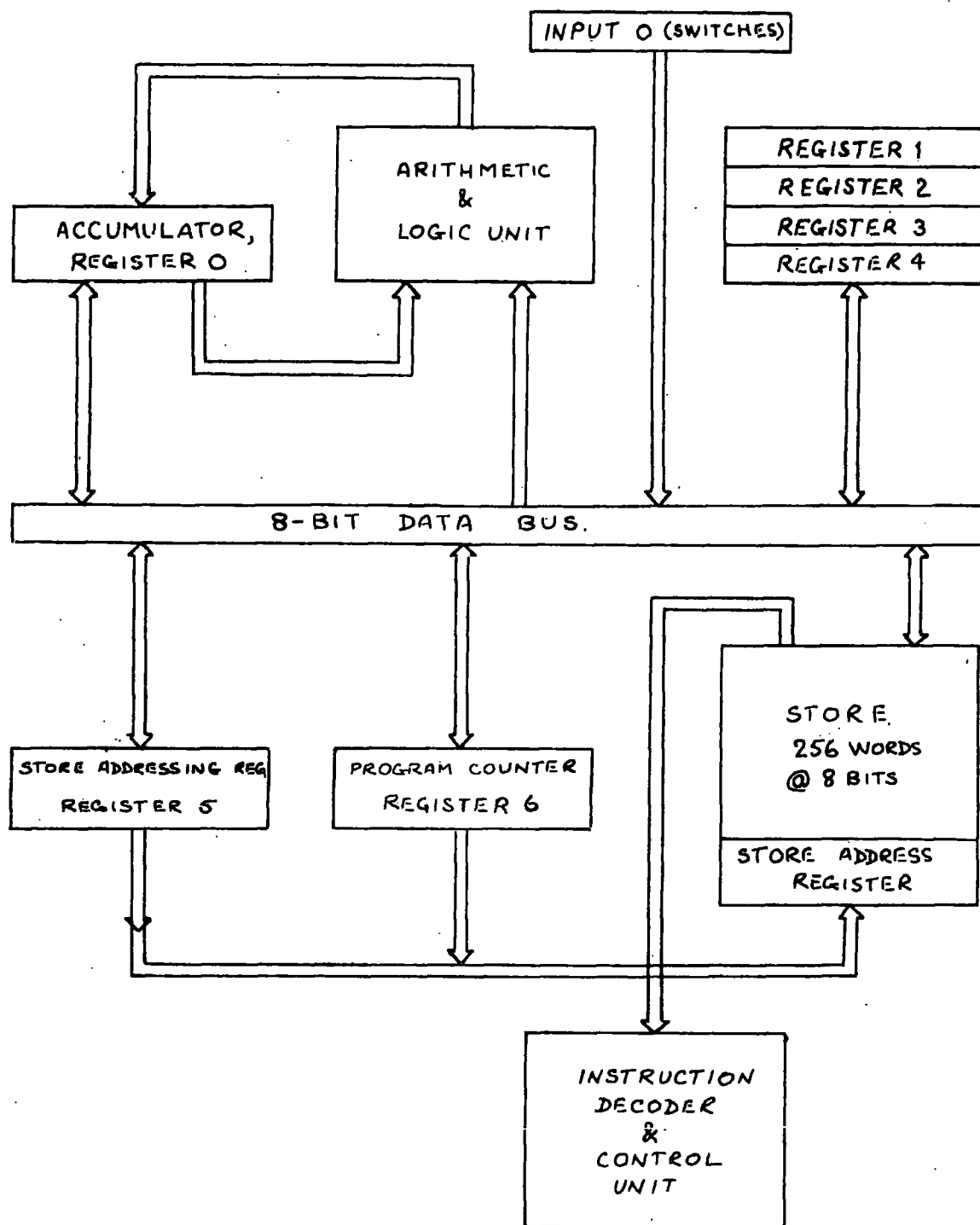


Figure 2.1 Organisation of Educational Computer.

2.8 The Order Code

The order code as finally implemented on the machine is given here. Most instructions are given in Octal format but in one or two cases certain bits have to be specified in binary for complete clarity.

<u>Code</u>	<u>Operation</u>
000 ₈ or 377 ₈	Halt.
1XY ₈	Copy the content of register Y into register X.
20X ₈	Clear register X. If X = 7, no operation.
21X ₈	Complement register X. If X = 7, no operation.
22X ₈	Increment register X. If X = 7, no operation.
23X ₈	Decrement register X. If X = 7, no operation.
24X ₈	Add content of register X to accumulator. Set Carry, Negative and Zero flags according to result.
25X ₈	Subtract content of register X from accumulator. Set Carry, Negative and Zero flags according to result.
26X ₈	Form logical AND between content of register X and original content of accumulator. Result in Accumulator. Set Negative and Zero flags according to result. Carry flag to zero.
27X ₈	Form logical OR between content of register X and original content of accumulator. Result in Accumulator. Set Negative and Zero flags according to result. Carry flag to zero.
30X ₈	Load switch setting (input 0) into register X.

<u>Code</u>	<u>Operation</u>
31X, 32X, 33X ₈	Reserved for input and output instructions. No operation at present.
34X ₈	Load register X immediately with the following byte of data. (a two-byte instruction)
35 ₈ CNZ ₂	(a) 354, Load the program counter with the following byte if the Carry flag is 1. (b) 352, Load the program counter with the following byte if the Negative flag is 1. (c) 351, Load the program counter with the following byte if the Zero flag is 1. If more than one flag is specified, e.g. by 355, the jump will occur if either flag is set. If the specified flag(s) is not set, the following byte is skipped.
36 ₈ CNZ ₂	Load the program counter if the specified flag is zero, otherwise the same as the previous instruction.
370 ₈ , 371 ₈	Shift accumulator one bit right or one bit left. If shift right, the Carry is copied into the MSB and the original LSB is lost. If shift left, the LSB becomes zero, the MSB moves into the Carry and the original carry is lost.
OXY ₈	With the exception of Halt (000) this subset is not used at present and the computer interprets it as 'no operation.'

CHAPTER 3:

Hardware

The hardware of the project was constructed in two phases:-

- a) the display panel and the interface during the early part of 1974. These are shown in plates 1 and 2.
- b) the processor unit in mid-1975. This is shown in plate 3.

These items are described in this chapter in the reverse order because the design of the processor follows naturally from the sections on the microprocessor 8008 and the prototyping unit SIM 8.

3.1 Microprocessor 8008

The 8008 is a single chip MOS 8-bit parallel central processor unit and requires control logic and memory to form a microcomputer system. The processor communicates over an 8-bit data and address bus and uses two input leads (ready and interrupt) and four output leads (S_0 , S_1 , S_2 and sync) for control. Time multiplexing of the data-bus allows control information, 14-bit addresses and data to be transmitted between CPU and memory.

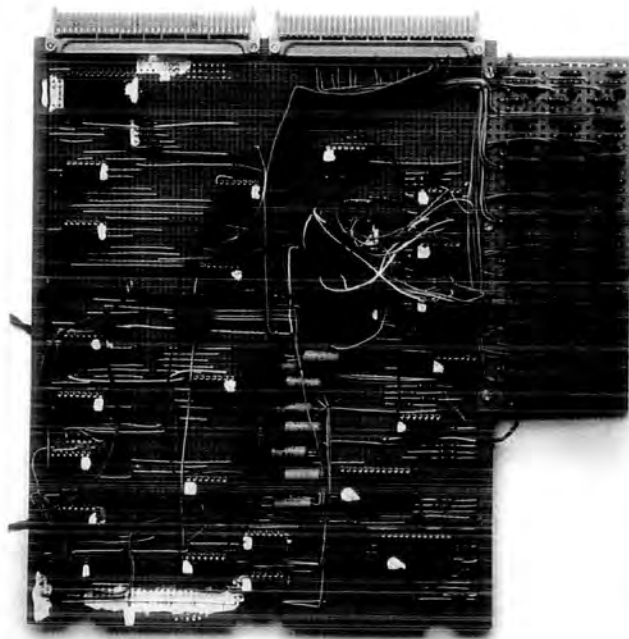


Plate 2. The Interface Board.

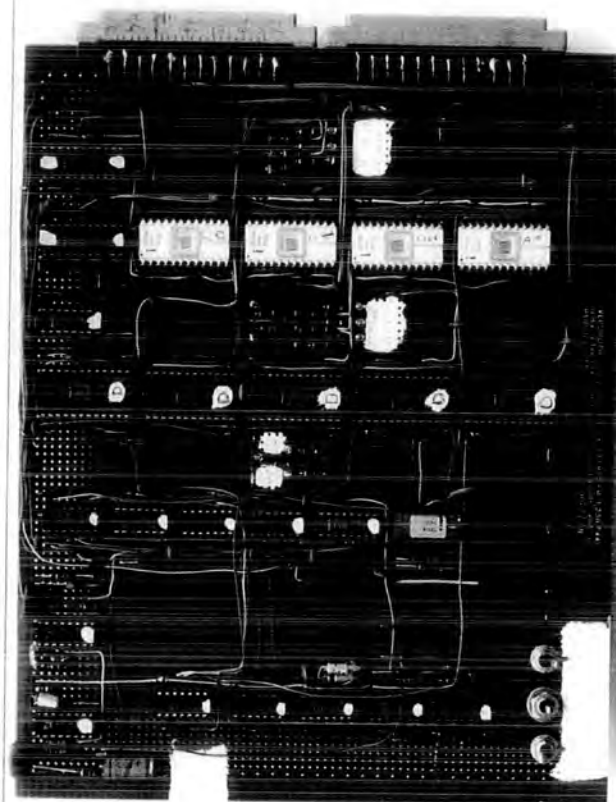


Plate 3. The Computer Board.

The CPU contains six 8-bit data registers, an 8-bit accumulator, two 8-bit temporary registers, four flag bits and an 8-bit parallel ALU which implements addition, subtraction and logical operations. A memory stack containing a 14-bit program counter and seven 14-bit words is used internally to store program and sub-routine addresses. The 14-bit address permits direct addressing of 16K words of memory which may be any mix of RAM, ROM or Shift register.

The chip is internally microprogrammed to implement a variety of inter-register transfer, arithmetic, control and logical instructions. Most instructions are coded in one byte (8-bits); data immediate instructions take two bytes; jump instructions use three bytes. Operating with a 500 KHz clock, the shortest instructions take 12 μ S for execution; most common instructions take between 20 and 32 μ S while the longest takes 44 μ S.

The instruction set consists of 48 instructions and is shown in Figure 3.1, a more detailed specification can be found in the Manual Ref. 7.

All inputs are TTL compatible and all outputs are Low-power TTL compatible.

Typically, a machine cycle consists of five states; two states (T_1 , T_2) in which addresses are output from the processor and latched externally, one state (T_3) for the instruction or data fetch from memory and two states (T_4 , T_5) for the execution of the instruction. If the processor is used with slow memories, the 'ready' line at logic 0 induces a 'wait' state after T_2 which persists until ready goes to 1 and allows the processor to proceed to state T_3 .

MCS-8 Instruction Set

INDEX REGISTER INSTRUCTIONS

The load instructions do not affect the flag flip-flops. The increment and decrement instructions affect all flip-flops except the carry.

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE D ₇ D ₆ D ₅ D ₄ D ₃ D ₂ D ₁ D ₀	DESCRIPTION OF OPERATION
ILI r ₂	(5)	1 1 0 0 0 1 1 1	Load index register r ₂ with the content of index register r ₂ .
ILI M	(8)	1 1 0 0 0 1 1 1	Load index register r ₂ with the content of memory register M.
LMi	(7)	1 1 1 1 1 1 1 1	Load memory register M with the content of index register r ₂ .
ILI r ₁	(8)	0 0 0 0 0 1 1 0	Load index register r ₁ with data B ... B.
LMi	(9)	0 0 1 1 1 1 1 0	Load memory register M with data B ... B.
INr	(5)	0 0 0 0 0 0 0 0	Increment the content of index register r (r ≠ A).
DCr	(5)	0 0 0 0 0 0 0 1	Decrement the content of index register r (r ≠ A).

ACCUMULATOR GROUP INSTRUCTIONS

The result of the ALU instructions affect all of the flag flip-flops. The rotate instructions affect only the carry flip-flop.

ADr	(5)	1 0 0 0 0 1 1 1	Add the content of index register r, memory register M, or data B ... B to the accumulator. An overflow (carry) sets the carry flip-flop.
ADM	(8)	1 0 0 0 0 1 1 1	
ADi	(8)	0 0 0 0 0 1 0 0	
ACr	(5)	1 0 0 0 1 1 1 1	Add the content of index register r, memory register M, or data B ... B to the accumulator with carry. An overflow (carry) sets the carry flip-flop.
ACM	(8)	1 0 0 0 1 1 1 1	
ACi	(8)	0 0 0 0 1 1 0 0	
SUr	(5)	1 0 0 1 0 1 1 1	Subtract the content of index register r, memory register M, or data B ... B from the accumulator. An underflow (borrow) sets the carry flip-flop.
SUM	(8)	1 0 0 1 0 1 1 1	
SUi	(8)	0 0 0 1 0 1 0 0	
SBr	(5)	1 0 0 1 1 1 1 1	Subtract the content of index register r, memory register M, or data B ... B from the accumulator with borrow. An underflow (borrow) sets the carry flip-flop.
SBM	(8)	1 0 0 1 1 1 1 1	
SBi	(8)	0 0 0 1 1 1 0 0	
NDr	(5)	1 0 1 0 0 1 1 1	Compute the logical AND of the content of index register r, memory register M, or data B ... B with the accumulator.
NOM	(8)	1 0 1 0 0 1 1 1	
NDi	(8)	0 0 1 0 0 1 0 0	
XRr	(5)	1 0 1 0 1 1 1 1	Compute the EXCLUSIVE OR of the content of index register r, memory register M, or data B ... B with the accumulator.
XRM	(8)	1 0 1 0 1 1 1 1	
XRI	(8)	0 0 1 0 1 1 0 0	
ORr	(5)	1 0 1 1 0 1 1 1	Compute the INCLUSIVE OR of the content of index register r, memory register M, or data B ... B with the accumulator.
ORM	(8)	1 0 1 1 0 1 1 1	
ORI	(8)	0 0 1 1 0 1 0 0	
CPr	(5)	1 0 1 1 1 1 1 1	Compare the content of index register r, memory register M, or data B ... B with the accumulator. The content of the accumulator is unchanged.
CPM	(8)	1 0 1 1 1 1 1 1	
CPI	(8)	0 0 1 1 1 1 0 0	
RLC	(5)	0 0 0 0 0 0 1 0	Rotate the content of the accumulator left.
RRC	(5)	0 0 0 0 0 1 0 1	Rotate the content of the accumulator right.
RAL	(5)	0 0 0 0 1 0 0 1	Rotate the content of the accumulator left through the carry.
RAR	(5)	0 0 0 0 1 1 0 1	Rotate the content of the accumulator right through the carry.

PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS

JMP	(11)	0 1 X X X 1 0 0	Unconditionally jump to memory address B ₃ ... B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂
JFc	(9 or 11)	0 1 0 C ₄ C ₃ 0 0 0	Jump to memory address B ₃ ... B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
JTc	(9 or 11)	0 1 1 C ₄ C ₃ 0 0 0	Jump to memory address B ₃ ... B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
CAL	(11)	0 1 X X X 1 1 0	Unconditionally call the subroutine at memory address B ₃ ... B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ . Save the current address (up one level in the stack).
CFc	(9 or 11)	0 1 0 C ₄ C ₃ 0 1 0	Call the subroutine at memory address B ₃ ... B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ if the condition flip-flop c is false, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
CTc	(9 or 11)	0 1 1 C ₄ C ₃ 0 1 0	Call the subroutine at memory address B ₃ ... B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ B ₂ if the condition flip-flop c is true, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
RET	(5)	0 0 X X X 1 1 1	Unconditionally return (down one level in the stack).
RFc	(3 or 5)	0 0 0 C ₄ C ₃ 0 1 1	Return (down one level in the stack) if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
RTc	(3 or 5)	0 0 1 C ₄ C ₃ 0 1 1	Return (down one level in the stack) if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
HST	(5)	0 0 A A A 1 0 1	Call the subroutine at memory address AA000 (up one level in the stack).

INPUT/OUTPUT INSTRUCTIONS

INP	(8)	0 1 0 0 M M M 1	Read the content of the selected input port (MMM) into the accumulator.
OUT	(8)	0 1 R R M M M 1	Write the content of the accumulator into the selected output port (RRMMM, RR ≠ 00).

MACHINE INSTRUCTION

HLT	(4)	0 0 0 0 0 0 0 X	Enter the STOPPED state and remain there until interrupted.
HLT	(4)	1 1 1 1 1 1 1 1	Enter the STOPPED state and remain there until interrupted.

NOTES:

- (1) SSS - Source Index Register. These registers, r_i, are designated Accumulator (000), D001, C010, D011, E100, H101, L1101.
- (2) DDU - Destination Index Register. These registers, r_i, are designated Accumulator (000), D001, C010, D011, E100, H101, L1101.
- (3) Memory registers are addressed by the contents of registers H & L.
- (4) Additional bytes of instruction are designated by 88888888.
- (5) X - "Don't Care".
- (6) Flag flip-flops are defined by C₄ C₃ carry (00-overflow or underflow), zero (01 result is zero), sign (11 MSB of result is "1"), parity (11 parity is even).

intel

Figure 3.1. Instruction Set of Microprocessor 8008.
(Reproduced with permission of Intel Corporation (UK) Ltd.)

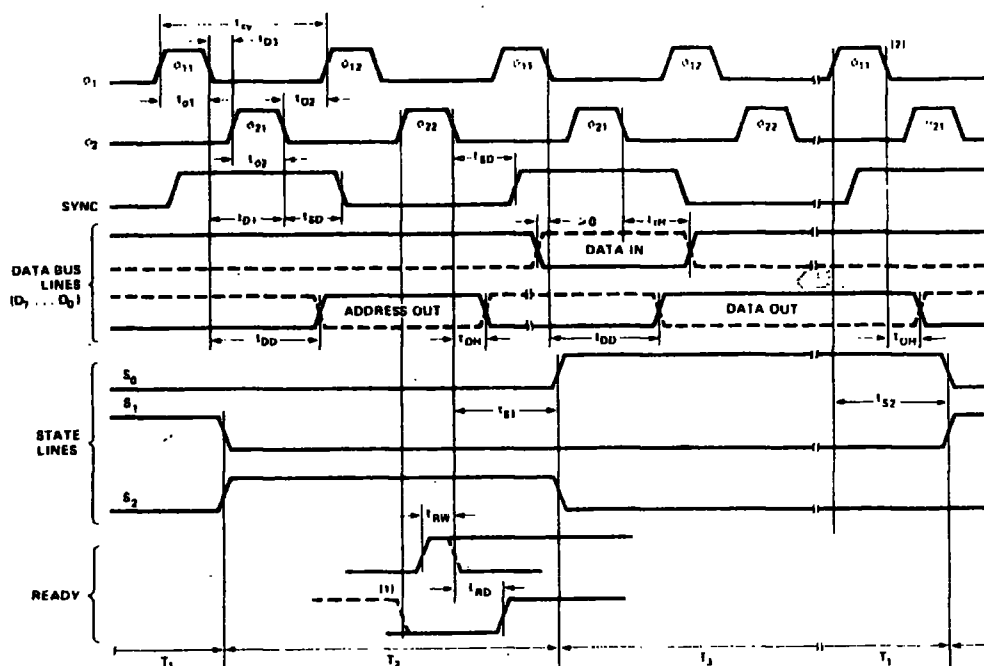
When a 'Halt' instruction is received, the processor enters the 'stopped' state after T_3 . The only escape from this state is by application of the interrupt signal which causes the processor to enter the T_{1I} state. This is an alternative to T_1 in which the lower eight bits of the address are sent out as usual but the program counter is not incremented, thus, in the absence of any special arrangement the instruction following the halt will be obeyed twice, consequently it is normally a no-operation instruction.

Being an MOS device, the 8008 is fairly slow and output signals can be delayed by up to $1.1 \mu\text{s}$ from the initiating change. The complete timing diagram and timing tolerances are shown in Figure 3.2.

With reference to the timing diagram, the following observations can be made:-

- a) Each state of the microprocessor is split into two parts, the first in which $\text{sync} = 1$, the second where $\text{sync} = 0$. The sync transition is initiated by the trailing edge of ϕ_2 and occurs after a delay t_{SD} which may be as long as 700 nS . The pulses ϕ_1 and ϕ_2 which occur in the first part of the cycle are designated ϕ_{11} and ϕ_{21} respectively, those in the second part are ϕ_{12} and ϕ_{22} . Due to the sync delay, the sync transition may occur during the ϕ_{11} pulse and therefore it is not possible to distinguish simply between ϕ_{11} and ϕ_{12} . In the case of ϕ_2 the combinations $\phi_2.\text{sync}$, $\phi_2.\overline{\text{sync}}$ allow this distinction.
- b) Data is placed on the output lines by the trailing edge of ϕ_{11} and removed by ϕ_{22} . It could be latched by either ϕ_{12} or ϕ_{22} , but because ϕ_{22} is the more easily derived it is used. Thus both high and low addresses are latched by ϕ_{22} , the distinction between them being made by T_1 and T_2 respectively.

TIMING DIAGRAM



- Notes: 1. READY line must be at "0" prior to ϕ_{22} of T_2 to guarantee entry into the WAIT state.
2. INTERRUPT line must not change levels within 200ns (max.) of falling edge of ϕ_1 .

A.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$ to 70°C ; $V_{CC} = +5V \pm 5\%$, $V_{DD} = -9V \pm 5\%$. All measurements are referenced to 1.5V levels.

SYMBOL	PARAMETER	8008		8008-1		UNIT	TEST CONDITIONS
		LIMITS		LIMITS			
		MIN.	MAX.	MIN.	MAX.		
t _{CY}	CLOCK PERIOD	2	3	1.25	3	μs	t _R , t _F = 50ns
t _R , t _F	CLOCK RISE AND FALL TIMES		50		50	ns	
t _{φ1}	PULSE WIDTH OF φ ₁	.70		.35		μs	
t _{φ2}	PULSE WIDTH OF φ ₂	.55		.35		μs	
t _{D1}	CLOCK DELAY FROM FALLING EDGE OF φ ₁ TO FALLING EDGE OF φ ₂	.90	1.1		1.1	μs	
t _{D2}	CLOCK DELAY FROM φ ₂ TO φ ₁	.40		.35		μs	
t _{D3}	CLOCK DELAY FROM φ ₁ TO φ ₂	.20		.20		μs	
t _{DO}	DATA OUT DELAY		1.0		1.0	μs	C _L = 100pF
t _{OH}	HOLD TIME FOR DATA BUS OUT	.10		.10		μs	
t _{IH}	HOLD TIME FOR DATA IN	(1)		(1)		μs	
t _{SD}	SYNC OUT DELAY		.70		.70	μs	C _L = 100pF
t _{S1}	STATE OUT DELAY (ALL STATES EXCEPT T1 AND T11) [2]		1.1		1.1	μs	C _L = 100pF
t _{S2}	STATE OUT DELAY (STATES T1 AND T11)		1.0		1.0	μs	C _L = 100pF
t _{RW}	PULSE WIDTH OF READY DURING φ ₂₂ TO ENTER T3 STATE	.35		.35		μs	
t _{RD}	READY DELAY TO ENTER WAIT STATE	.20		.20		μs	

(1) $t_{IH} \text{ MIN} \geq t_{SD}$

(2) If the INTERRUPT is not used, all states have the same output delay, t_{S1} .

Figure 3.2. Timing Diagram of Microprocessor 8008.

(Reproduced with permission of Intel Corporation (UK) Ltd.)

- c) Data into the processor must be present before the trailing edge of ϕ_{11} in state T_3 . Due to the state out delay t_{s_1} this edge can occur before T_3 can be recognised. An extra state must therefore be generated (T_{3A}) which is initiated by the trailing edge of ϕ_{22} in T_2 and terminated by $\overline{\text{sync}}$ of T_3 . This state bridges any wait that occurs between T_2 and T_3 and gates either memory or input data to the microprocessor.
- d) The interrupt line must not change within 200 nS of the trailing edge of ϕ_1 . Since the interrupt request may occur at any time it is necessary to hold it until an appropriate time, apply it to the interrupt line and then to cancel it after it has been recognised.

3.2 Prototyping Unit SIM 8

This unit was designed by Intel with the aim of providing all the features that would be needed by a prospective user of the 8008. The circuit diagram and layout can be found in the User Manual, Ref. 7

It provides 1 K of RAM using page addresses 010₈ to 013₈, space for 2 K of PROM type 8702 using page addresses 000 to 007₈, two input and one interrupt port ready multiplexed and four output ports ready latched. An interface is provided for a teletype so that program can be loaded into RAM from keyboard or paper tape.

A comprehensive range of signals including high and low address lines, memory input and output data lines, processor state, clock and sync lines are available from the board for use in specialised applications and were used for developing the interface unit and program before the processor board was constructed.

3.3 The processor unit

The SIM8 unit provides many features which are not necessary in the present application and since it was designed during 1972 it would be surprising if the parts-count could not be reduced by using integrated circuits which have become available since that time. Both of these avenues have been explored in the design of the unit to be described.

Diagrams relevant to this section are figures 3.3 to 3.8. Figure 3.3 shows the layout of the unit and its connectors and assigns reference numbers prefixed with B to all integrated circuits. These references are also given where appropriate in the other diagrams so that the location of any functional unit can quickly be found.

Data flow within the unit is shown in Figure 3.4. Data is supplied to the 8008 via the 'input data-bus' and data output from the 8008 flows via the buffer to the 'output data-bus.' Pin numbers used by all the buses shown on this diagram are given in the table of Figure 3.5. The important unit for transmitting data to and from the 8008 is the Intel unit 8212 and this is shown in greater detail in Figure 3.6. Basically it is an 8-bit latch with tri-state gated output and four control lines. Its truth table is:-

<u>Required Operation</u>		<u>Control inputs</u>			
	MP	$\overline{DS1}$	DS2	STB	
Enable output gate	(1	X	X	X	
	or (X	0	1	X	
Write into latches	(0	X	X	1	
	(1	0	1	X	

The Clear input (CLR) is not used in this application.

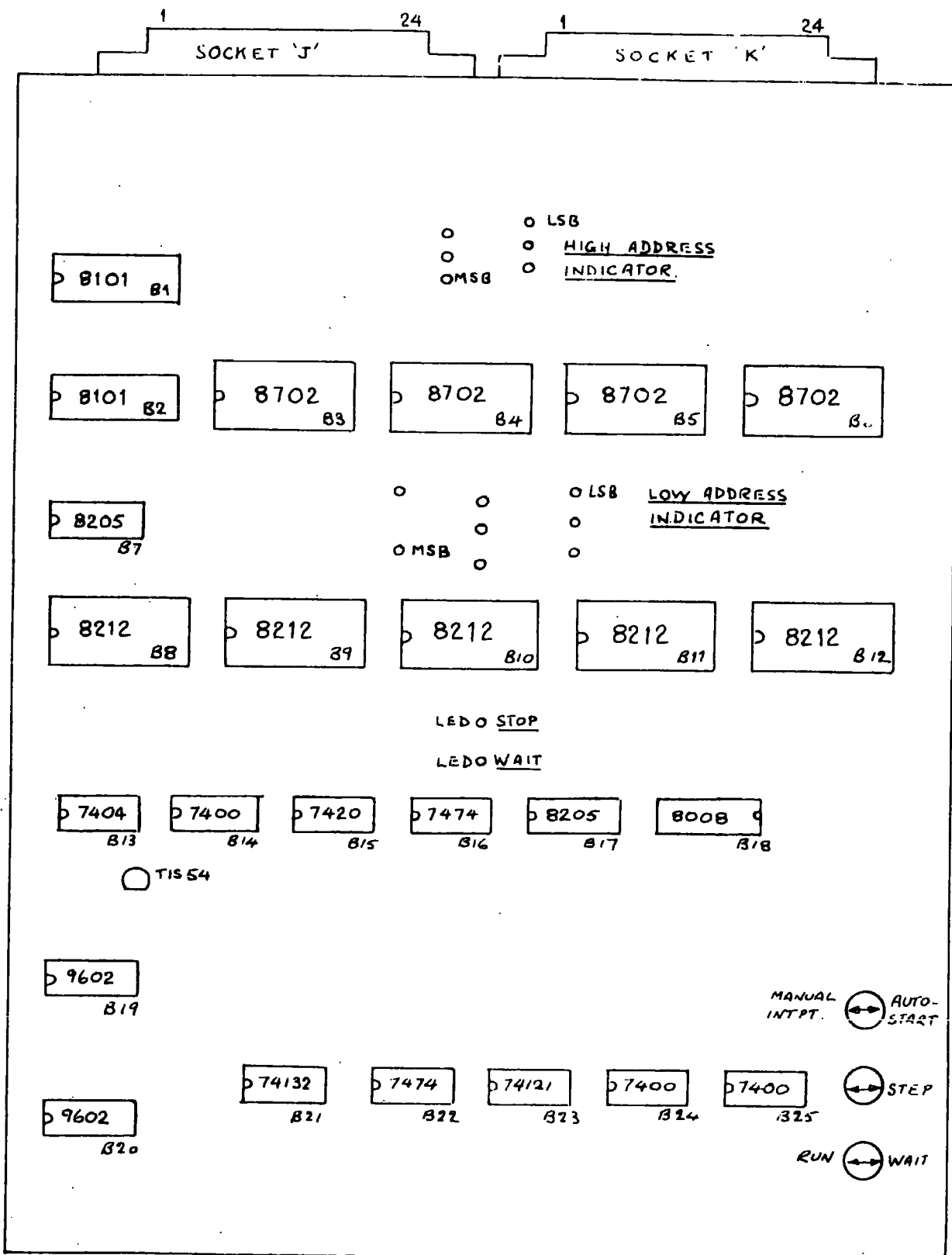


Figure 3.3. Layout of Computer Board.

BUS (Refer to Fig 3.4)	UNIT	TYPE	REFERENCE NUMBER	PIN CONNECTION FOR BIT NUMBER															
				1	2	3	4	5	6	7	8								
Input Data Bus	Microprocessor	8008	B15	9	8	7	6	5	4	3	2								
	Input Port	8212	B12	4	6	3	10	15	17	19	21								
	Memory Port	8212	B11	4	6	8	10	15	17	19	21								
	Buffer	8212	B9	3	5	7	9	16	18	20	22								
Output Data Bus	Buffer	8212	B9	4	6	8	10	15	17	19	21								
	High Add. Latch	8212	B8	3	5	7	9	16	18	21	22								
	Low Add. Latch	8212	B10	3	5	7	9	16	18	21	22								
	RAM	8101	B2	9	11	13	15	-	-	-	-								
	RAM	8101	B1	-	-	-	-	9	11	13	15								
Memory Data Bus	Memory Port	8212	B11	3	5	7	9	16	18	20	22								
	Read-only Memory	8702	B3,4,5,6	4	5	6	7	8	9	10	11								
	RAM	8101	B2	10	12	14	16	-	-	-	-								
	RAM	8101	B1	-	-	-	-	10	12	14	16								
Memory Address Bus	Low Add. Latch	8212	B10	4	6	8	10	15	17	19	21								
	Read-only Memory	8702	B3,4,5,6	3	2	1	21	20	19	18	17								
	RAM	8101	B1,2	4	3	2	1	21	21	5	6								
	RAM	8101	B1,2	4	3	2	1	21	21	5	6								

Figure 3.5. Table of Bus pin connections.

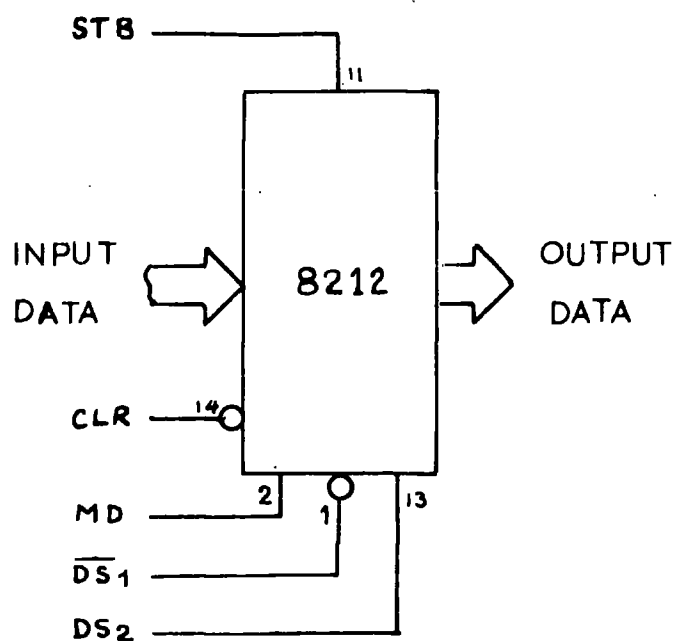
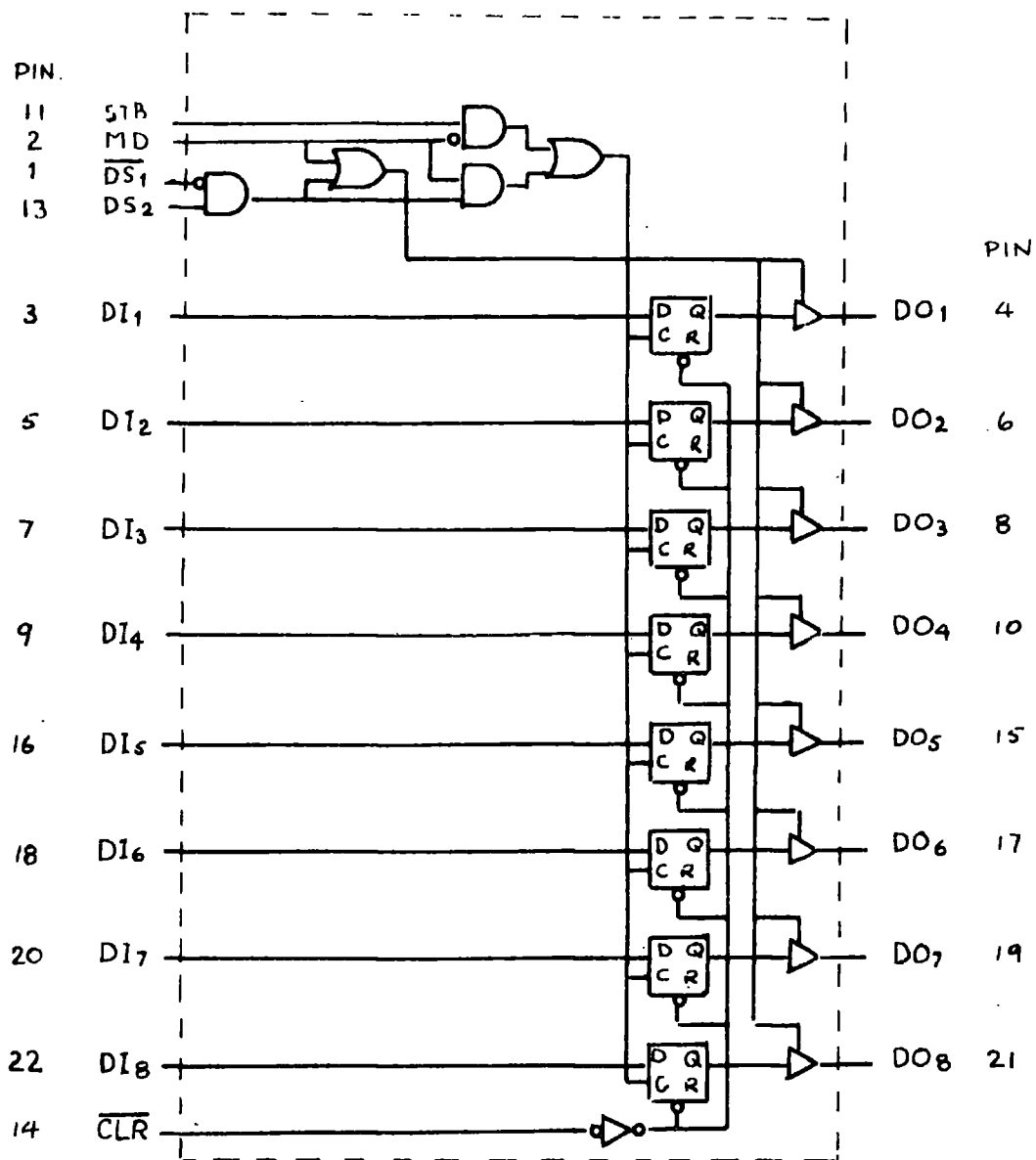


Figure 3.6.
Part-circuit of Intel
Unit 8212 and Symbol.

The units 1 and 2, input and memory ports respectively, have $MD = 0$, $STB = 1$, so that the latches continually follow the input data and when it is required that this data be applied to the Input data-bus, the appropriate device is selected by $\overline{DS1} = 0$, $DS2 = 1$.

The unit 3 is simply used as a buffer and the control inputs accordingly are $MD = 1$, $\overline{DS1} = 0$, $DS2 = 1$, $STB = X$.

Units 4 and 5 have their outputs permanently enabled while latching the inputs at the appropriate times, therefore $MD = 1$, $STB = X$, $DS2 = \phi_{22}$ and $\overline{DS1} = \overline{T_1}$ for unit 4, $\overline{T_2}$ for unit 5.

The outputs of the address latches and the data buffer are fed to the memory and the interface unit. The arrangement of the memory is shown in Figure 3.7. Certain of the high-address lines are decoded to select the memory chips and because only four pages of ROM, one page of RAM and one page of shift-register (in the interface) are required, only the three least significant bits need be decoded. However, the RAM used in development had been page 013 in the SIM 8 and this character had already been written into ROM, furthermore, decoding for X77 (the shift-register page) was included in the interface. Therefore to maintain compatibility between SIM8 and the processor, bits 1, 2 and 4 were used as inputs to the 3-line decoder, bit 3 was applied to an Enable input. The result is that addresses 0, 1, 2, 3 are decoded directly, 013 appears as output 7 and X77 disables the decoder while enabling the shift-register. All the memory chips have tri-state output gates so that the chip-select signals simply enable these gates and impose the corresponding data on the memory data-bus.

The control logic of the system is shown in Figure 3.8. The outputs are all down the right hand side. Considering these in sequence:-

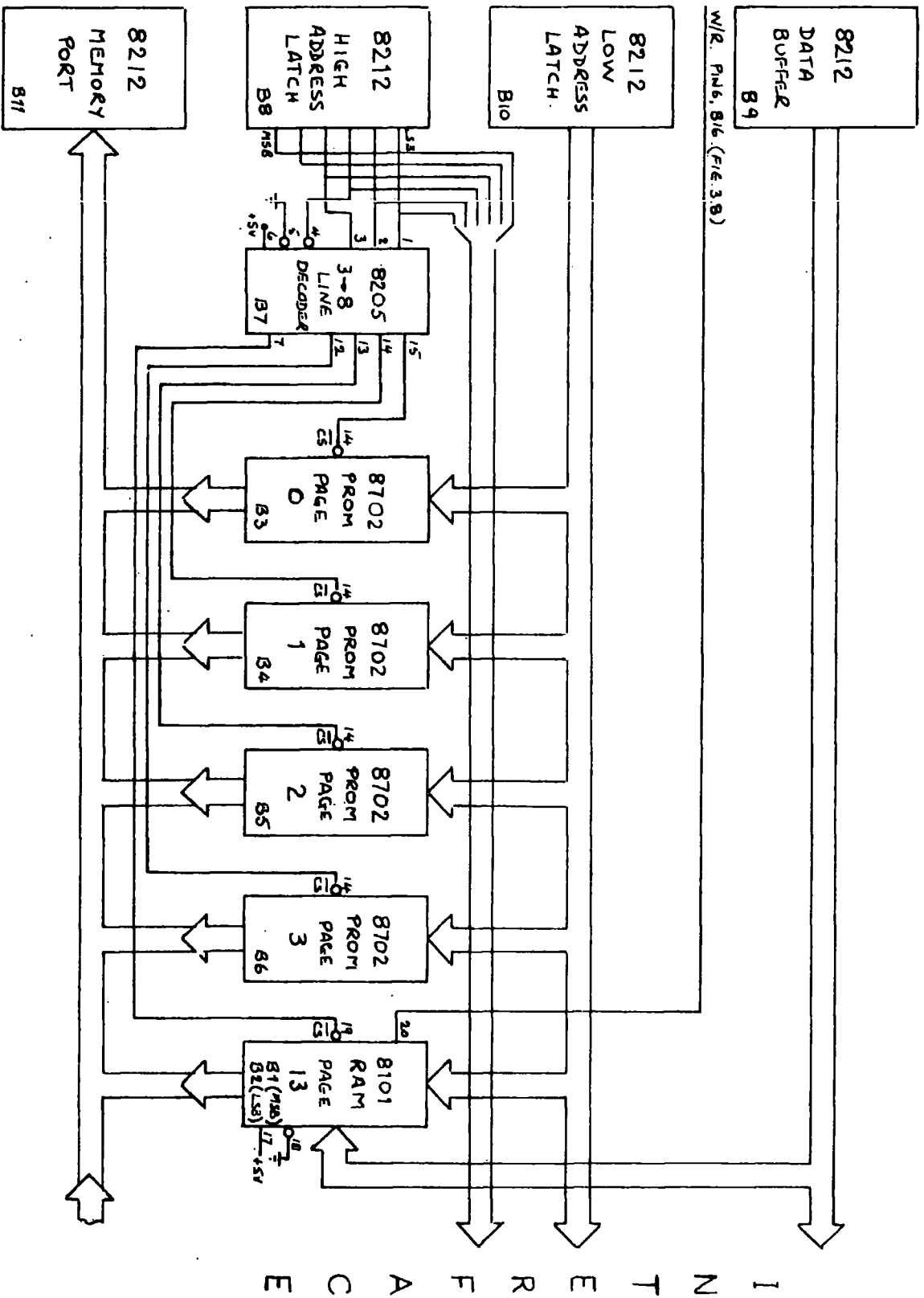


Figure 3.7. Memory Arrangement - Computer Board.

The Clock signals are generated by two dual retriggerable monostables type 9602, the pulse widths are preset by paralleling suitable resistors. This circuit is identical to that used in the SIM 8 unit.

Units B8, B10 (High and low address latches) are latched by ϕ_{22} applied to DS2 and $\overline{T_1}$, $\overline{T_2}$ to $\overline{DS1}$ as already explained. The state signals are obtained from the 3-line decoder B17, type 8205. The sync output of the microprocessor is the only one requiring a buffer and rather than introduce a low-power T.T.L. inverter with limited fan-out, an emitter-follower with standard T.T.L. was used.

The need for the bridging state T_{3A} was explained in section 3.1. It is provided by a D-type flip-flop clocked to 1 at ϕ_{22} of T_2 on all except write cycles. The expression for the clock pulse is:-

$$\overline{\phi_{22} \cdot \text{sync} \cdot T_2 \cdot \overline{DO_7 \cdot DO_8}}$$

which goes high on the trailing edge of ϕ_{22} . T_{3A} is reset by $\overline{T_3 \cdot \text{sync}}$ i.e. at the end of the first part of T_3 .

T_{3A} enables DS2 of both the input and memory ports, $\overline{DS1}$ is used to discriminate between them. On a memory-read cycle when $DO_7DO_8 = 0X$, $\overline{DS1}$ of the memory port (B11) is enabled; on an input cycle when $DO_7DO_8 = 10$, $\overline{DS1}$ of the input port (B12) is enabled.

When data is to be written to store, the W/R lines are enabled by the 4-input gate B15 which presets the D-type flip-flop when

$$DO_7 \cdot DO_8 \cdot T_3 \cdot \text{sync} \cdot \phi_2 = 1$$

i.e. the ϕ_{21} pulse of T_3 in a PCW cycle.

The flip-flop is reset by the following ϕ_{22} pulse applied to the clock input.

The 'ready' line of the 8008 is controlled by the interface when the Wait/Run switch is in the Run position. Otherwise the 'step' switch applies a 5 μ S pulse to the ready line when it coincides with the interface also being ready.

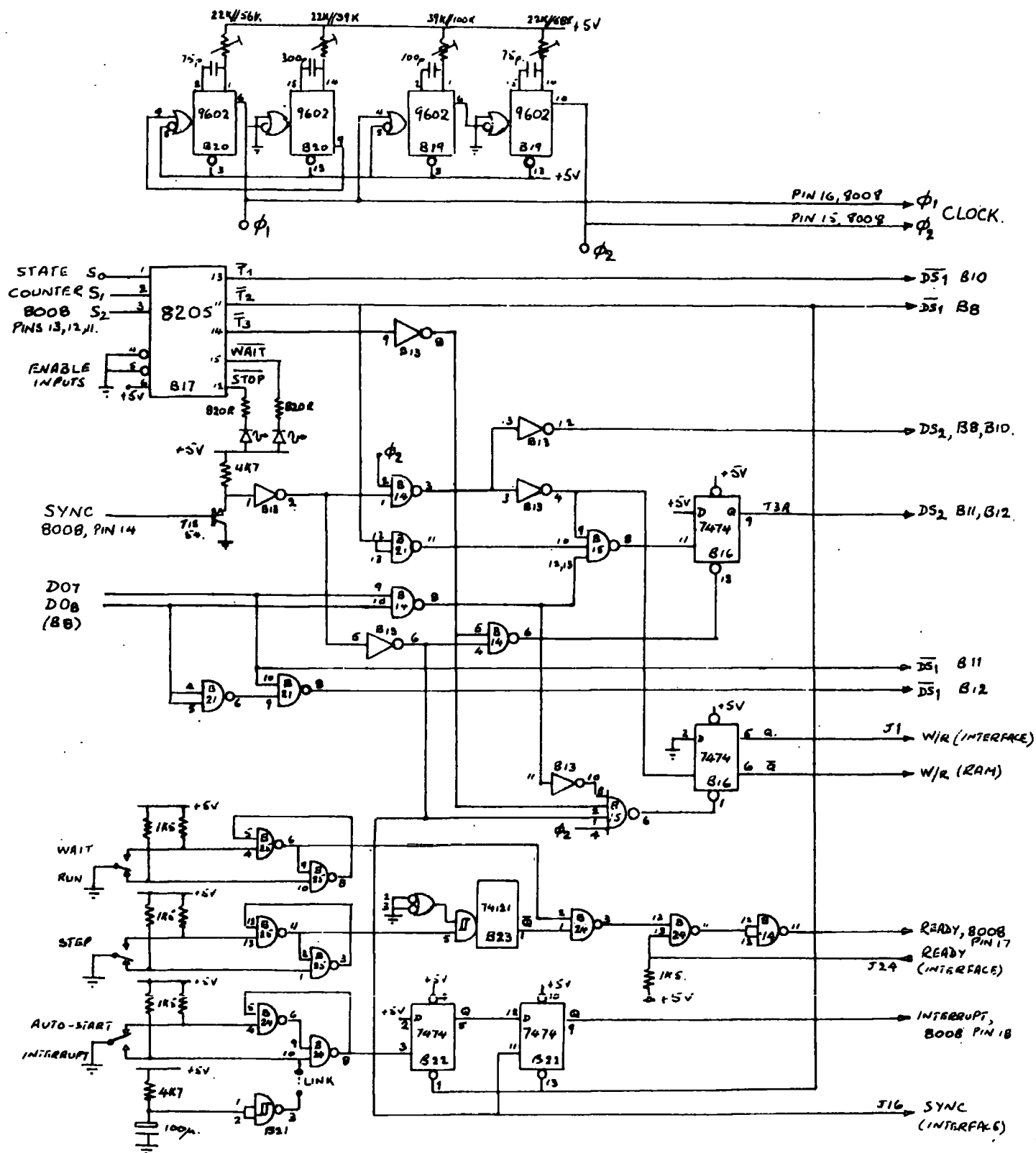


Figure 3.8. Control Logic of Computer Board.

An interrupt can be applied to the processor manually, for program development purposes or automatically, for starting under normal conditions. A clock pulse is applied in both cases to the first D-type, the second is then set by the following sync pulse and applies the interrupt. This arrangement meets the timing requirement of the interrupt given in the timing chart, Figure 3.2. Both flip-flops are reset when the interrupt is recognised and the processor enters state T_2 .

3.4 The Interface

Diagrams relating to the interface are Figures 3.9 - 3.16. Figure 3.9 shows the layout of the board and its connectors and assigns reference numbers prefixed by C to all the integrated circuits. These numbers are also shown in the other diagrams and enable functional elements to be quickly located.

The general organisation of the interface is shown in Figure 3.10; data to be displayed on the panel is stored in a 32-bit shift-register, eight bits wide. Normally the shift-register recirculates and as each word of eight bits appears at the output it controls the bit-lines of the LED matrix (Figure 3.12) via the Bit Drivers (Figure 3.15) while a 5-bit binary counter which is keeping track of the shift-register energises the appropriate word line. Not all of the 32 locations of the shift-register are needed for display and some of these non-display locations are used as ordinary memory locations within the program. When access to the shift-register for read or write is needed, the processor is forced into a wait state until the correct location is reached.

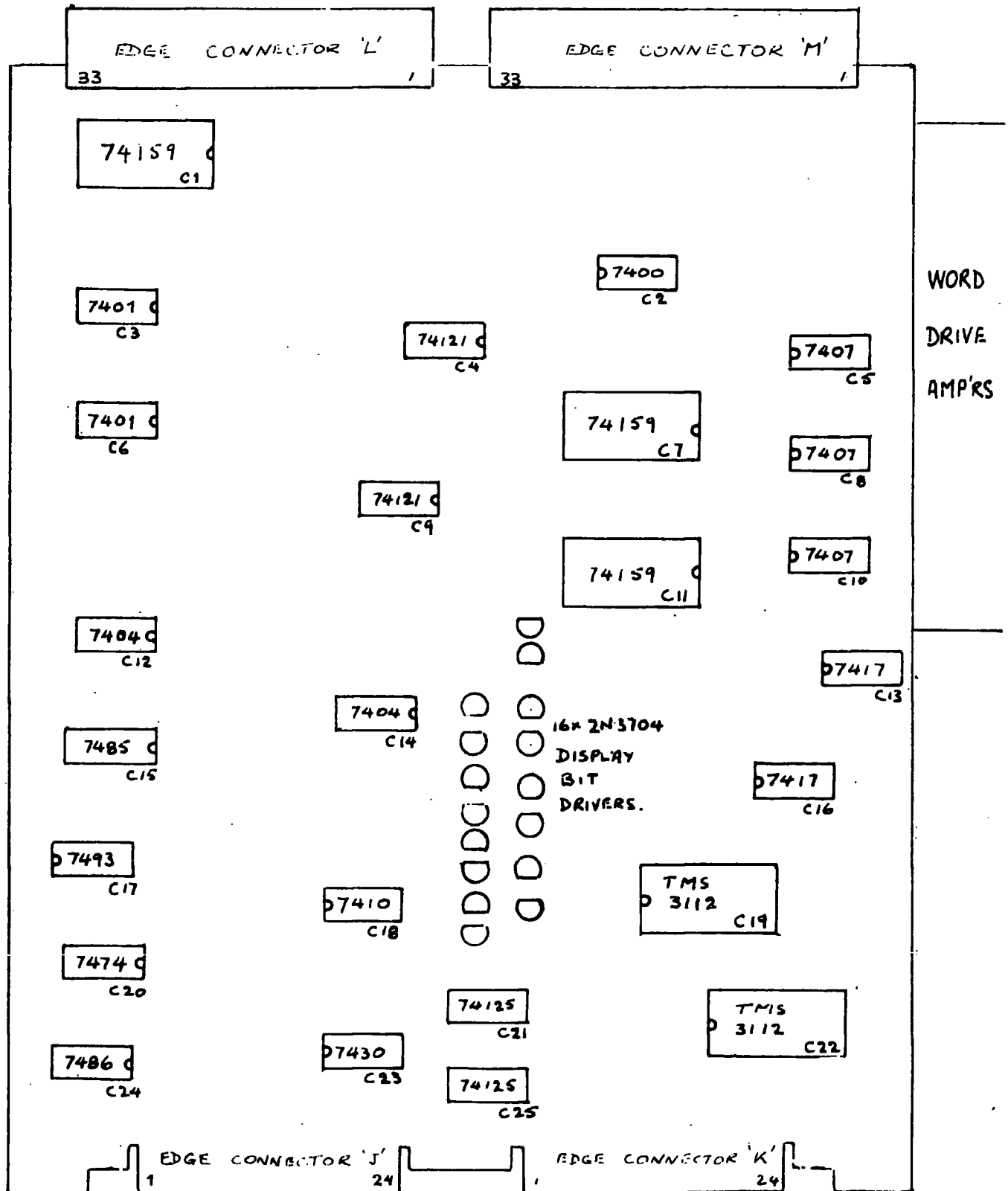


Figure 3.9. Layout of Interface Board.

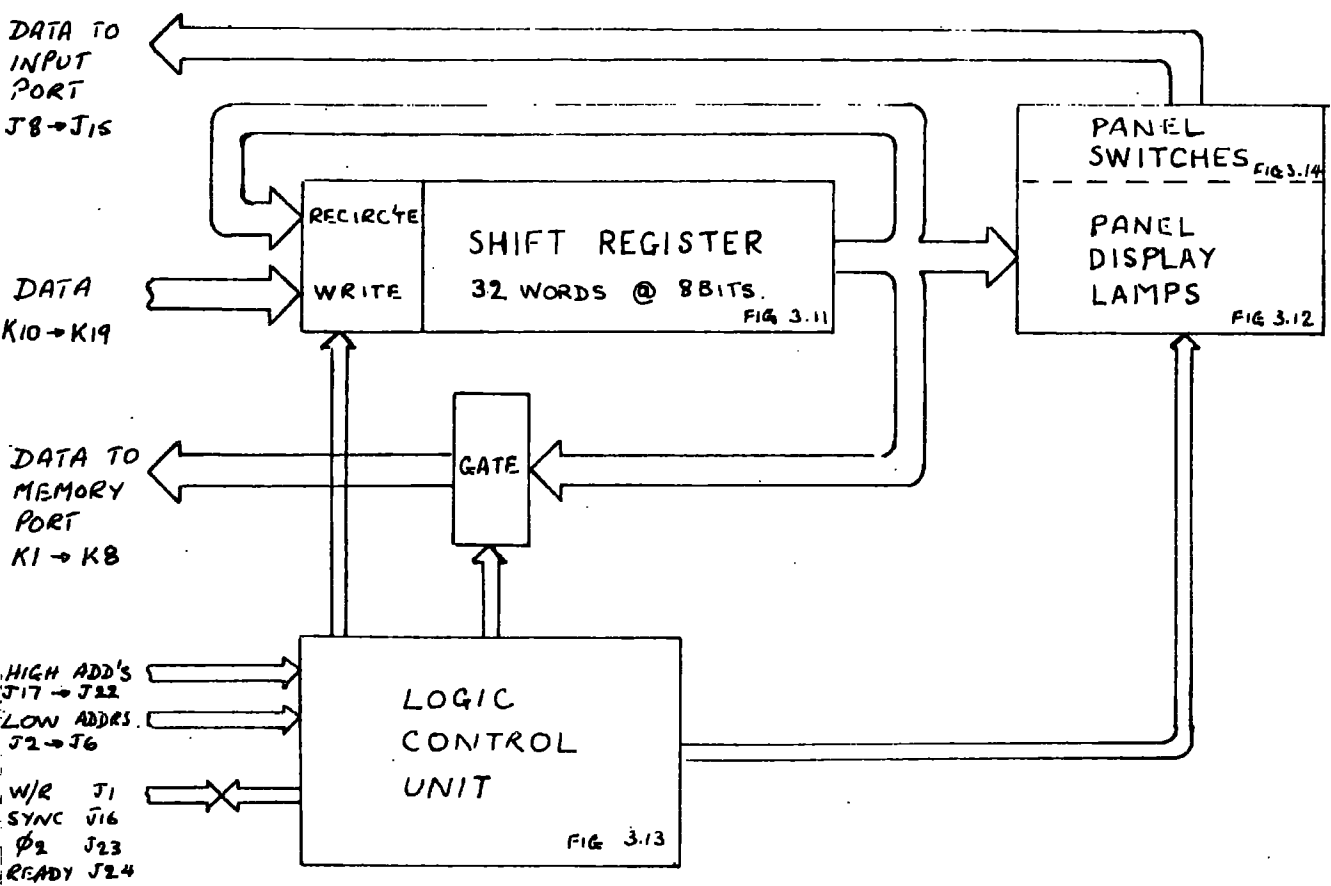


Figure 3.10. Organisation of Interface.

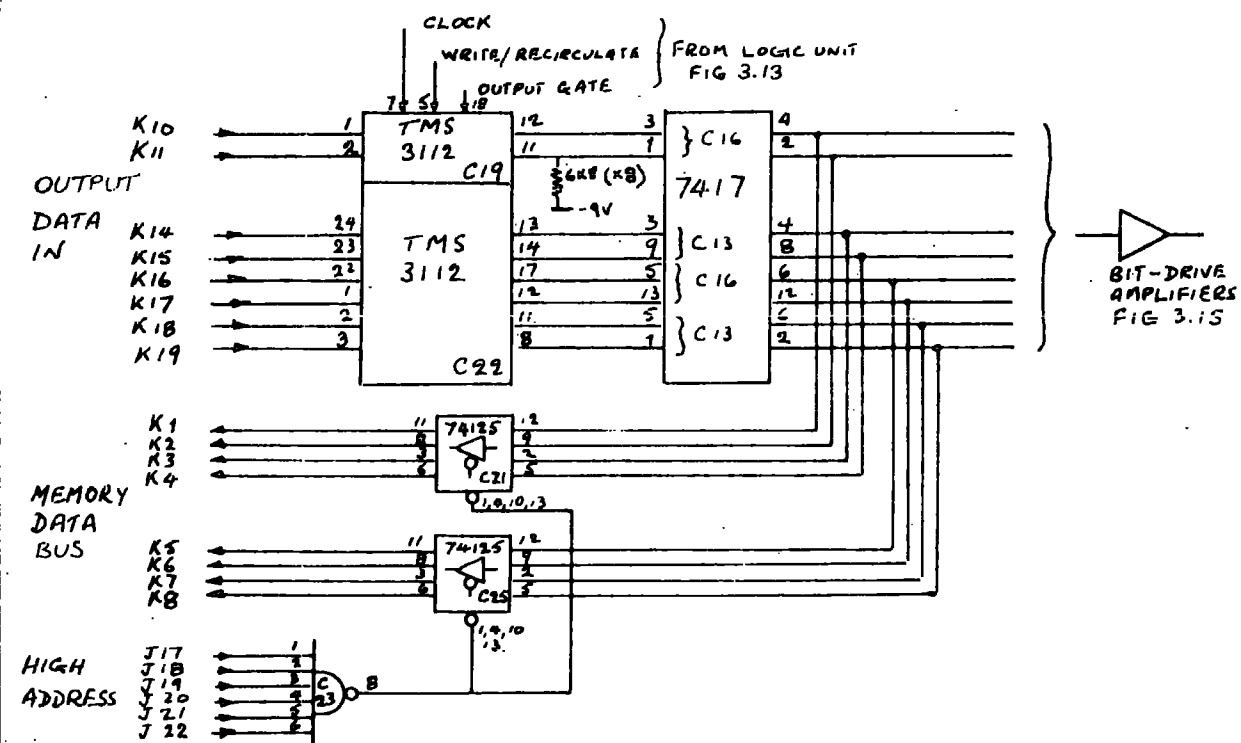


Figure 3.11. Shift Register Detail.

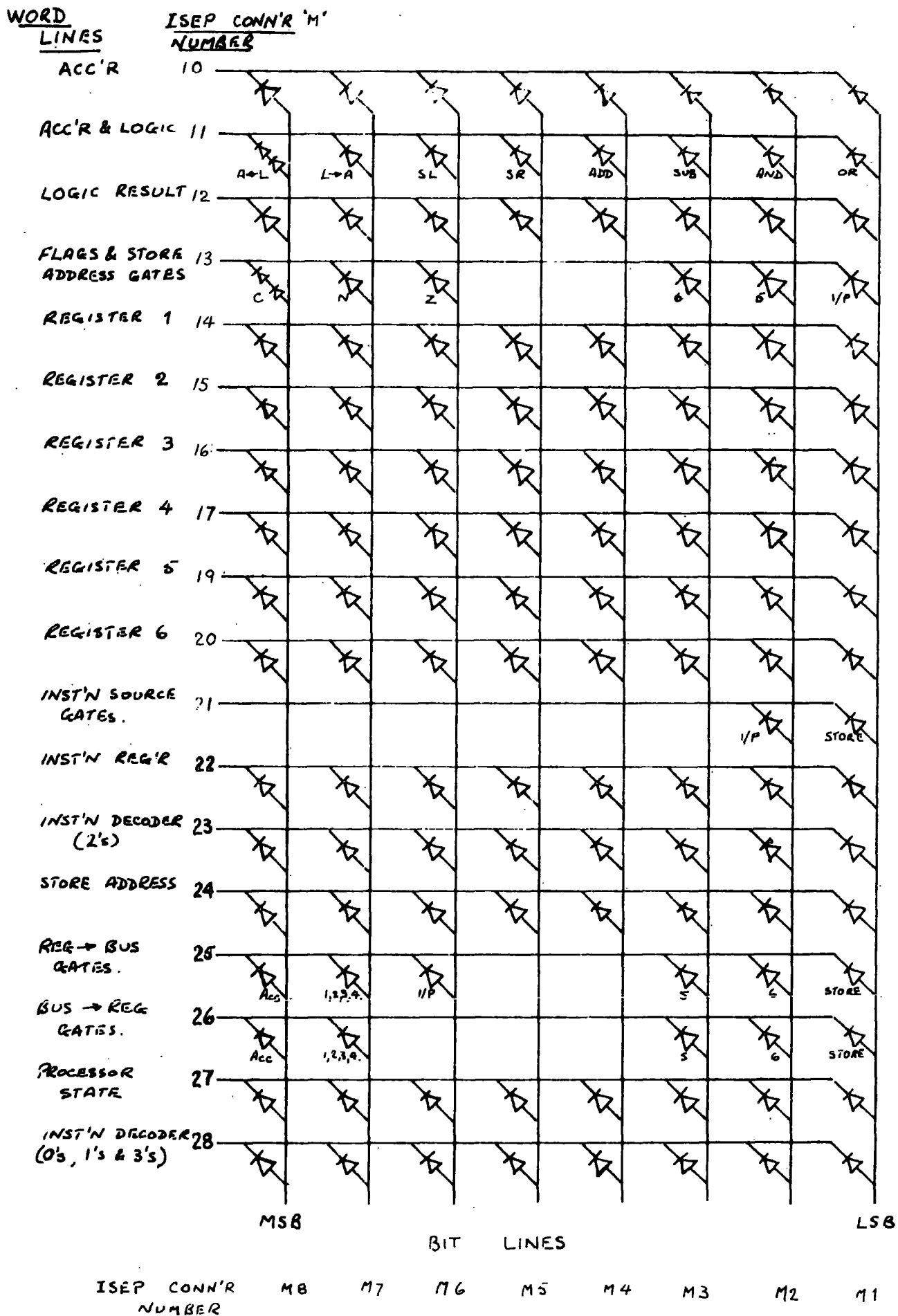


Figure 3.12. Panel Display Matrix.

The shift-register is shown in greater detail in Figure 3.11; it is controlled by three signals generated in the logic unit (Figure 3.13) which are clock, write/recirculate and output gate enable. The output of the MOS shift-register is only able to drive a single T.T.L. load so it is buffered by the SN7417 units which drive in turn the Bit drive amplifiers and the tri-state gates SN74125. When the high address X77 is latched, the gate C23 enables the tri-state gates and imposes the shift-register output on the memory data-bus.

The logic control unit, Figure 3.13, performs several functions; taking them from the top:-

The first two items are connected with the Switch Matrix shown in Figure 3.14. When the processor 8008 executes an input instruction it outputs the accumulator contents at time T_1 and this is held by the low address latch, the peripheral address is held by the high address latch. By decoding four bits of the low address, a single peripheral address can be multiplexed sixteen ways. The unit SN74159 performs this function, the outputs each pulling one of the word lines to earth. If a push-button connected to the earthed line is operated, the corresponding bit line is earthed while the rest remain high. The signals from the bit-lines are inverted in the units SN7404 and applied to the input port. The toggle switches at the top of Figure 3.14 must be gated to the bit lines because they operate simultaneously and for long periods. The effect is to connect several bit lines together.

The counter keeping track of the shift register consists of the D-type SN7474 and the 4-bit binary counter SN7493. The LSB of the counter is compared with that of the low address in the Exclusive-OR

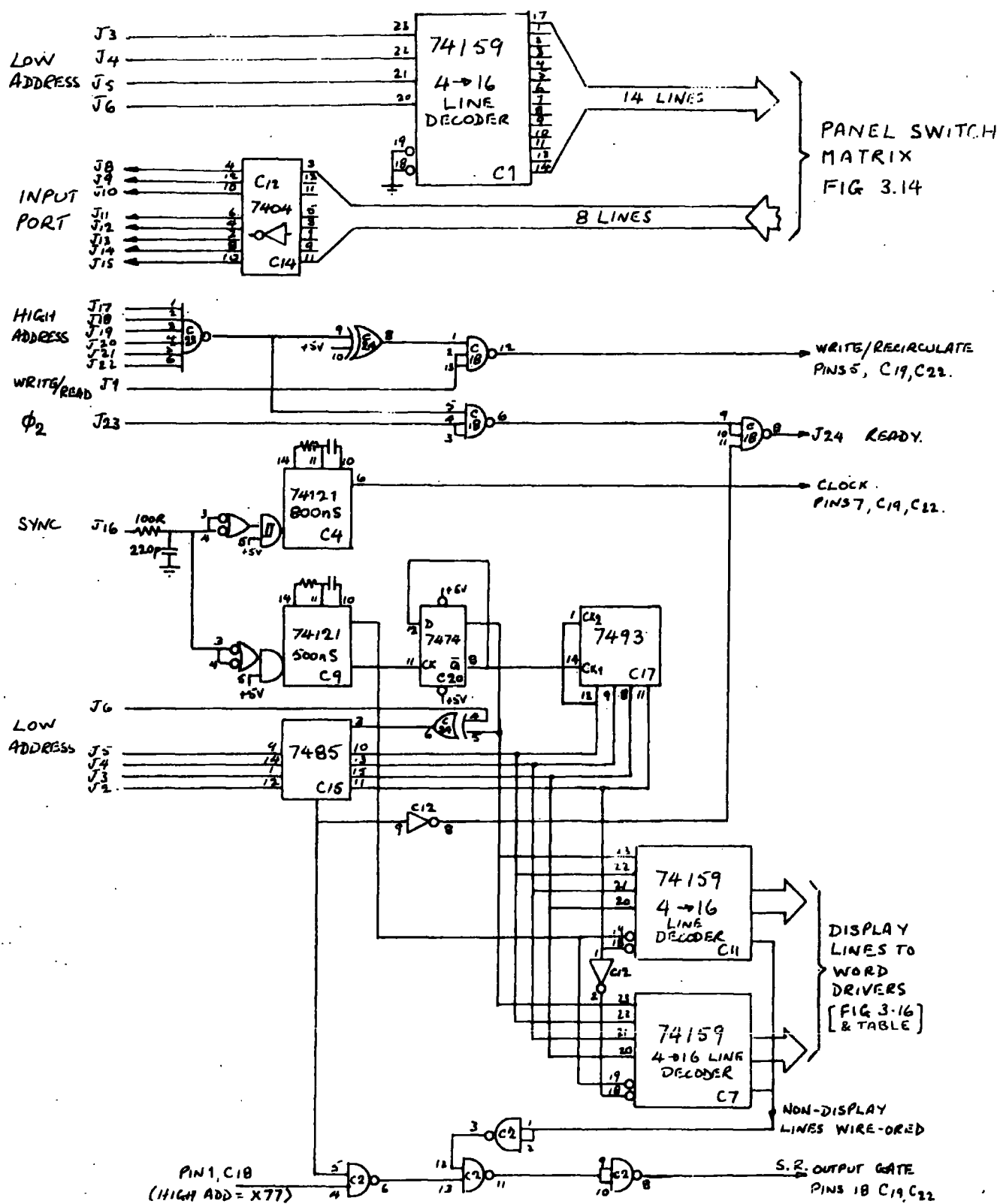


Figure 3.13. Logic Control Unit - Interface Board.

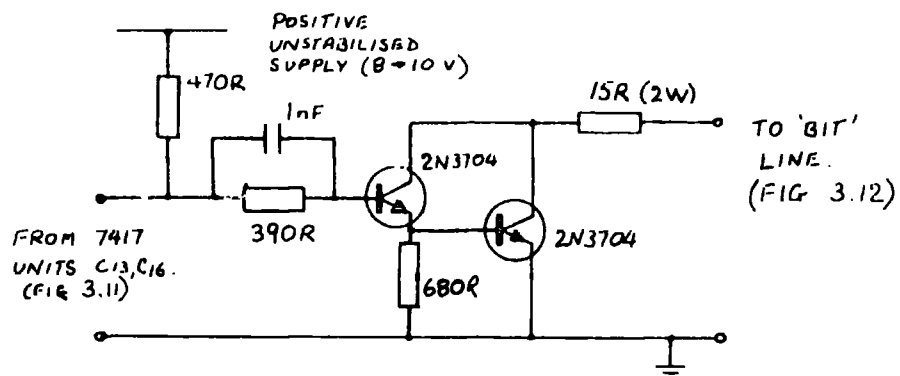


Figure 3.15. Bit-drive Amplifier for Display.

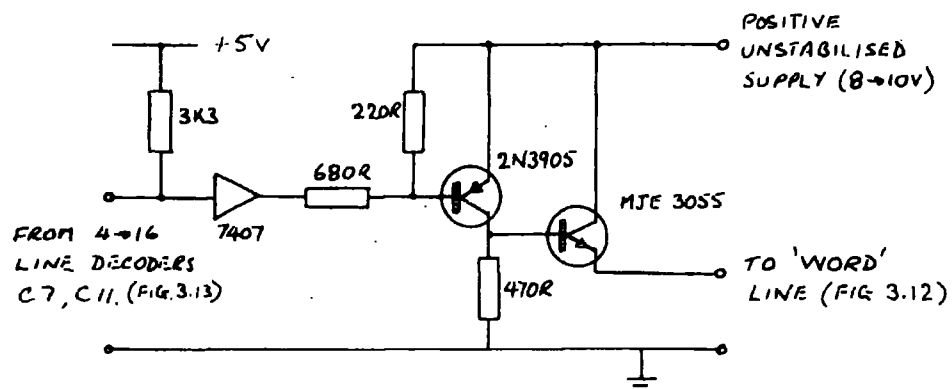


Figure 3.16. Word-drive Amplifier for Display.

Connections of 18 Word-drivers.

7407 REF. N°	INPUT PIN	OUTPUT PIN	1SEP CON'R N° (FIG 3.9)
C10 ↑ ↓	1	2	M12
	3	4	M11
	5	6	M10
	9	8	M13
	11	10	M14
	13	12	M15
C8 ↑ ↓	1	2	M19
	3	4	M17
	5	6	M16
	9	8	M20
	11	10	M21
	13	12	M22
C5 ↑ ↓	1	2	M25
	3	4	M24
	5	6	M23
	9	8	M26
	11	10	M27
	13	12	M28

gate C24, the result being applied to the A = B cascading input of the 4-bit binary comparator SN7485 which compares the remaining four bits. The ready line of the 8008 is enabled when equality is indicated by the SN7485 or by ϕ_{22} when the high address is other than X77.

Access to the shift register may be summarised:-

- a) High address X77 is output at T_2 time, ready is disabled and the shift register assumes control of the memory data-bus.
- b) Shift register and counter are clocked until equality between counter and low address is reached, ready line is enabled.
- c) Processor enters T_3 , accepting data from the shift register on a read cycle. On a write cycle the W/R signal is generated and together with high address X77 this switches the shift register to 'write.'

The five outputs of the counter are fed to two '4 to 16 line decoders' SN74159 which energise the word lines of the LED matrix, Figure 3.12, via the 'word line drivers' Figure 3.16.

Outputs corresponding to locations in the shift register which are not to be displayed are wire-ORed and used to disable the output gate of the shift register except when read/write access is required. The LED display matrix operates by one of the word lines being held high while one or more of the bit lines go low, thus energising the LEDs at the intersections. The word and bit drivers must be synchronised if crosstalk between adjacent words of the display is to be avoided. Three measures have been adopted to minimise such crosstalk:-

- a) The clocks of the counter and the (relatively) slow shift register were offset by about 300 nS by means of the two monostables C4, C9, in order to synchronise their output transitions.
- b) The word drivers were disabled for about 500 nS following the clock pulse of the counter in order to allow the charge stored in the output transistor of the word drive amplifier to decay.
- c) The bit drivers were set to zero on non-display locations by controlling the output gate of the shift register as explained earlier.

By these means, crosstalk in the prototype was reduced to an acceptable degree but it is felt that the real solution lies in the use of a shift register and counter of comparable speed and a faster power transistor in the output stage of the word line driver. This problem is further discussed in the concluding chapter.

CHAPTER 4:

The Program of the Microprocessor

4.1 General

The complete listing of the program with addresses and comments is given in Appendix 1. The description given in this chapter essentially refers to flow diagrams on which labels and addresses used in the program are given wherever necessary.

The program breaks down into three major functional parts:-

- a) The Mode-selection section which determines mode and speed of operation, starting and stopping. This is described in Section 4.2.
- b) Manual operation section, consisting of a series of sub-routines which service the various panel push-buttons. These sub-routines are also used by the microprogram section wherever it is most economical to do so. This section is described in 4.3.
- c) The Microprogram which determines the step by step operation when executing instructions. This is described in Section 4.4.

Labels used in the program were intended to be meaningful but, inevitably, mnemonics are more meaningful to their inventor than anyone else. The following points may be helpful:-

- a) The program counter is called G in some of the older parts of the program, e.g. INCC is a label meaning Increment G or Increment PC.
- b) The store-addressing register 5 is referred to as F. e.g. XFRF = transfer F.
- c) The input switch register was called K. e.g. RDK = Read K.

Useful constants in the program are 000_8 and 377_8 , the former is kept in register C but does not sterilise it because it can still be used for counts and delays which have a final value of zero. Constant 377 is held in register H where it provides the high address for the shift register in the interface. It only needs to be changed in two cases; (a) when reading from RAM in the sub-routine INSTOR and (b) when loading RAM in the sub-routine LDSTOR. The high address in these cases is 013.

The processor board does not decode the input address and therefore any INP instruction enables the input port; however, in order to maintain compatibility with the SIM 8 unit, input address 1 is used throughout. Prior to the INP instruction being used it is necessary to load the accumulator with the proper code to select the required input switches. The codes and the switches they select are:-

<u>Code (octal)</u>	<u>Switches selected</u>
000	Clear, complement, increment, decrement.
001	Register switches on lower console.
002	Mode selector.

<u>Code (octal)</u>	<u>Switches selected</u>
003	Speed selector.
004	Bus to register gates.
005	Accumulator and Bus to ALU gates.
006	ALU functions.
007	Register to Bus gates.
010	Input 0 to Bus gate.
011	Store address source gates.
012	Instruction source gates.
013	Start and Stop.
014	ALU result to acc. gate, shift right, left.
017	Input 0 (toggle switch settings)

This table may alternatively be deduced from Figure 3.14.

The functions stored in the 32 words of the shift register are:-

<u>Location (octal)</u>	<u>Function stored</u>
000	Accumulator content.
001	Temp. store used in PANEL and DOBIT sub-routines.
002	Content of register 1.
003	Temp. store for PANEL sub-routine.
004	Content of register 2.
005	Temp. store for PANEL sub-routine.
006	Content of register 3.
007	Spare.
010	Content of register 4.
011	Spare.
012	Content of register 5.
013	Temp. store for ST/ST (Start/stop) sub-routine.

<u>Location (octal)</u>	<u>Function stored</u>
014	Content of register 6 (Program counter).
015	Spare.
016	Temp. store for OPONA sub-routine.
017	Accumulator to ALU gate indicator.
020	ALU result and also temp. store in LOGIC sub-routine.
021	Temp. store in LOGIC sub-routine.
022	Content of store address register.
023	Spare.
024	Content of instruction register.
025	Spare.
026	Register to Bus gate indicators.
027	Bus to register gate indicators.
030	ALU to acc. gate, shift left, shift right indicators.
031	ALU function indicator.
032	ALU flags.
033	Store address source gate indicators.
034	Instruction decoder indicator. (Inst'ns 0,1,3.)
035	Instruction decoder indicator. (Inst'ns 2)
036	Instruction source gate indicators.
037	Processor state indicator.

While the locations given here have been specified in the range 0 to 37, the hardware does not decode the three most significant bits so that each location can be addressed by eight different characters. It is convenient to address 037 by 377 for example. Not all the bits are used in every location, the actual usage may be deduced from Figure 3.12.

4.2 Mode of operation

The flow chart of this section of the program is shown in Figure 4.1. The machine has four modes of operation which are selected by a four position mode selector switch, viz:-

Manual	in which the machine responds to all the push-buttons on the panel and lower console.
One-bit	in which the machine executes one step of the current instruction when the start button is pressed.
One-instruction	in which a complete instruction is executed step by step following operation of the start button. Operation may be frozen at an intermediate step by holding down the stop button.
Continuous	when the machine takes sequential instructions from store and obeys them until such time as the stop button is pressed or a halt instruction is obeyed.

In all except the manual mode, the machine must stop at the end of an instruction in a special state in which the panel push-buttons are operative as they are in the manual mode. In the last two modes the rate at which the microprogram is executed is variable from about two seconds per step to about 10 mS per step.

The flowchart consists basically of a loop in which one micro-instruction is executed on each traverse. Entry to the loop is only gained if the mode is not manual and if the start button has been operated; operation within the loop depends on the mode selected, the state of start and stop buttons and the processor state indicator. Exit from the loop is always to the manual mode of operation.

MODE @ 0135

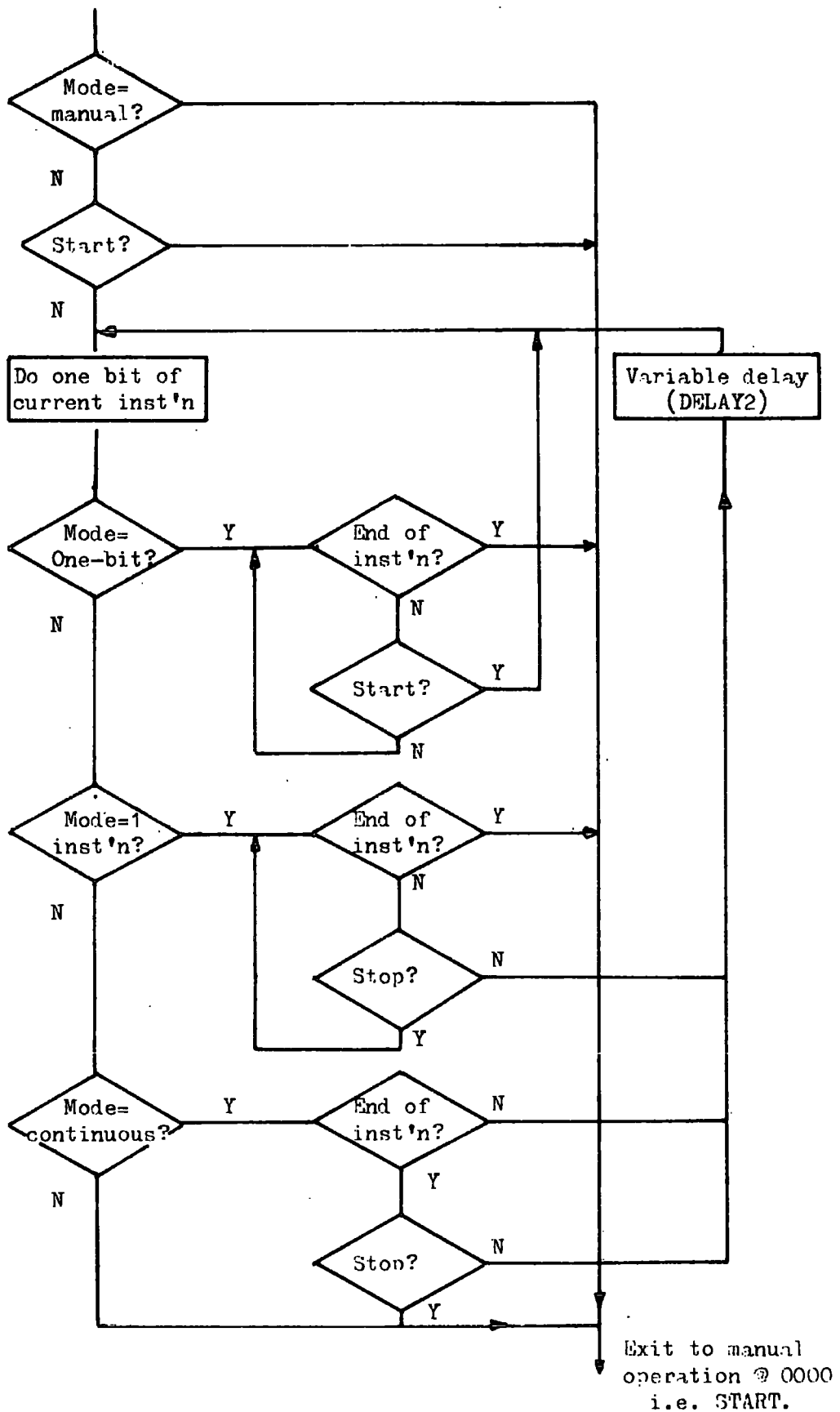


Figure 4.1 Flowchart for Mode Selection.

4.3 Manual operation

This part of the program starts at line 000 of page 000; the initial conditions H = 377, C = 000, processor state = 001 are set up and then a series of sub-routines are called. These are:-

PANEL	services the switches of the lower console.
DOUT (Data OUT)	services the register to bus gate switches.
DIN (Data IN)	services the bus to register gate switches.
OPONA (OPerate ON Acc)	services accumulator operations.
LOGIC	services the ALU.
INSTN (INSTructionN)	services the instruction source gate switches.
DECODI (DECODE Instruction)	decodes the instruction register.
STGATE (STore GATES)	services the store address gate switches.

These sub-routines are considered in turn:-

4.3.1 PANEL

The flow chart is shown in Figure 4.2.

Two sets of switches are involved, the four register function switches and the seven register selectors. In order to eliminate the effect of switch bounce, both sets of switches are ignored unless successive samples have identical non-zero values.

Of the function selectors (Clear, complement etc.) only one should be operated at any one time. This requirement frequently occurs in the manual mode of operation and the sub-routine ONLY1 is designed to deal with it. This sub-routine counts the number of operated switches in the sample and returns control to the calling program if the result is unity, otherwise it causes lamps associated with the operated switches to flash once before returning to the start of the manual sequence. This cycle is repeated for so long as two or more switches in the group are operated and the lamps flash continuously during this time.

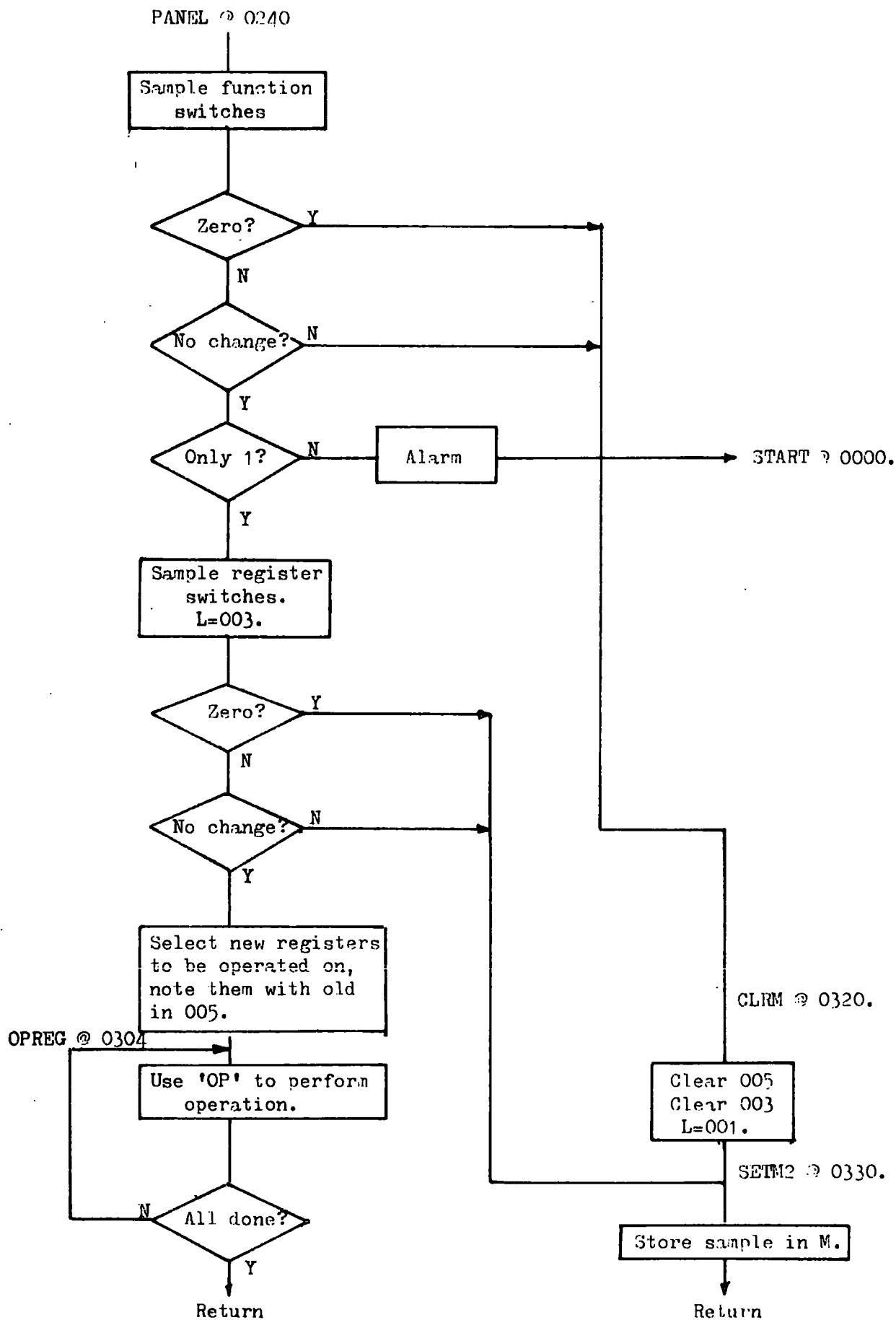


Figure 4.2. Flowchart for subroutine 'PANEL'

In the case of the function selectors there are no lamps associated with the switches, the visual indication cannot be given and the computer simply appears to ignore the switches when two or more are operated.

When the required function has been uniquely established, the registers to be operated on are determined and modified. A record is kept of those changed so that if the switch bounces or is released and re-operated the register is not changed a second time.

4.3.2 DOUT (Data OUT)

The flow chart is shown in Figure 4.3.

This sub-routine samples the switches controlling the gates which transfer data from a register to the bus. If one of them is found to be operated the corresponding lamp is lit and data copied from the register into register B of the microprocessor. If no switches are operated, the lamps are extinguished and 3778 loaded into register B. If more than one switch is found operated, the corresponding lamps are flashed and the sub-routine abandoned.

4.3.3 DIN (Data IN)

The flow chart is shown in Figure 4.4.

This sub-routine samples the switches controlling the transfer of data from bus to registers or store. If one or more are operated, the corresponding lamps are lit and data from register B copied into the selected registers.

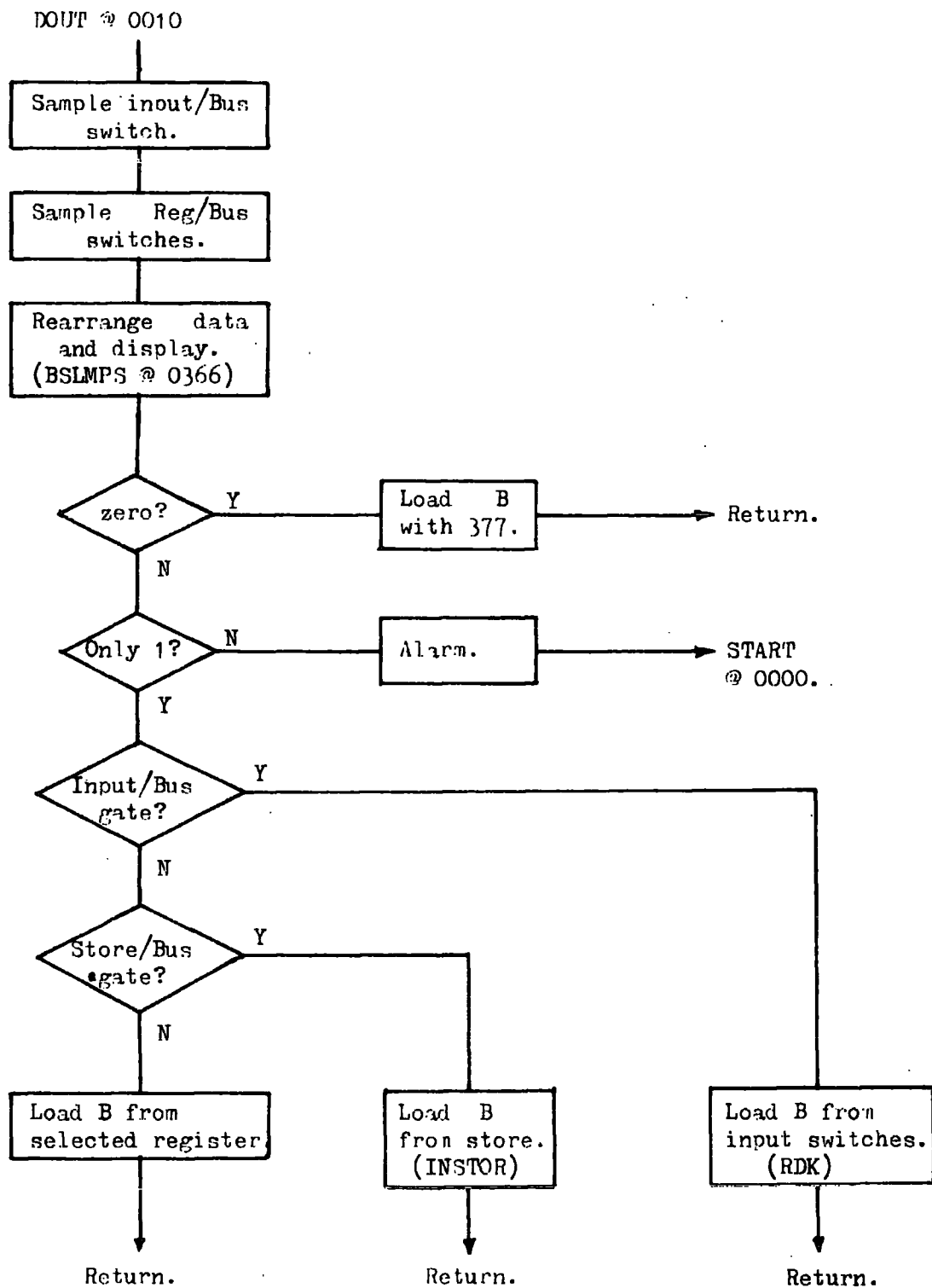


Figure 4.3. Flowchart for DOUBT subroutine.

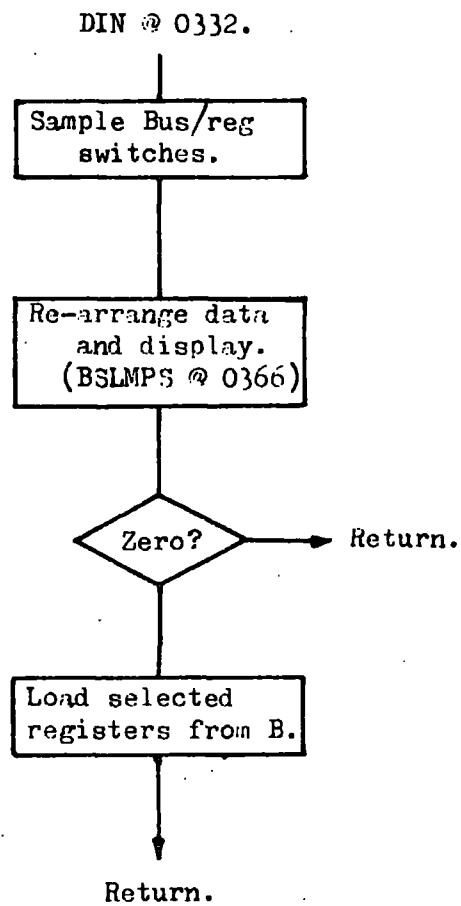


Figure 4.4 Flowchart for DIN subroutine.

4.3.4 OPONA (OPerate ON Accumulator)

The flow chart is shown in Figure 4.5.

Operations on the accumulator are:-

Transfer ALU result to accumulator (LTOA)

Shift accumulator left (SHFTL)

Shift accumulator right.

These switches are sampled and the corresponding lamp(s) lit. If all are off, the temporary store at 016 is cleared and the sub-routine abandoned. The three operations are mutually exclusive of each other and of the operation 'load accumulator from bus,' therefore this extra switch is sampled and combined with the others for the ONLY1 test. The shift instructions must only be performed once while the switch is operated and to indicate this, the store location 016 is loaded from H when the operation is performed. Subsequently, the sub-routine is abandoned when the location 016 is found to be non-zero.

4.3.5 LOGIC

The flowchart for this sub-routine is shown in Figure 4.6.

The operation to be performed by the ALU is selected by four push-buttons, if none is operated the previous setting is used and if there is no previous setting the display is set to ADD. If more than one switch is operated, the lamps are flashed and the whole program restarted.

The switch controlling the input gates is sampled and displayed, if zero the sub-routine is abandoned.

The return instruction (007) is loaded in location 021 of the shift register, the ORB instruction (261) is loaded in E and the required ALU operation in A and D. A is rotated until the single 1 reaches the carry and on each shift prior to this, 020 is subtracted from E thus changing

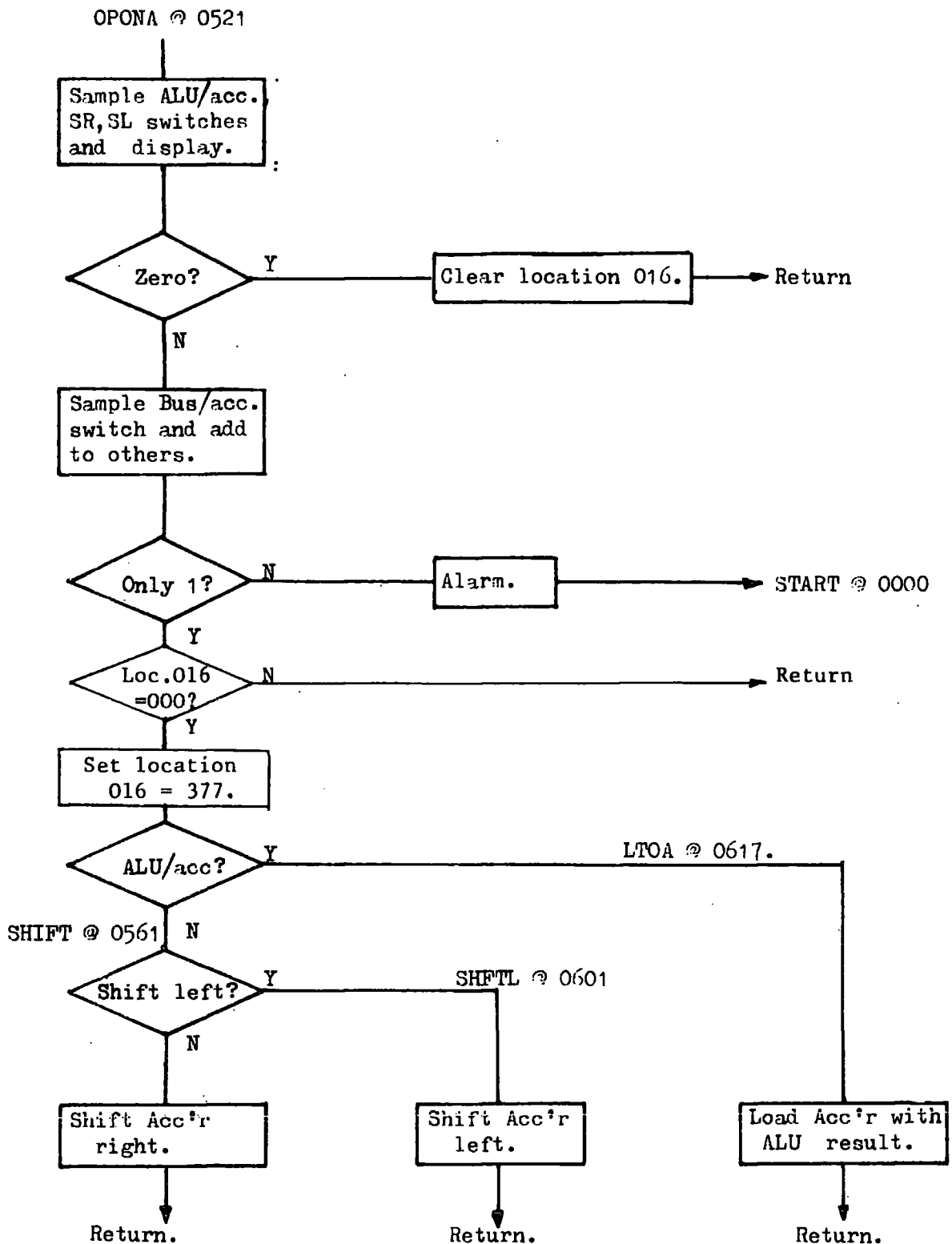


Figure 4.5. Flowchart for OPONA subroutine.

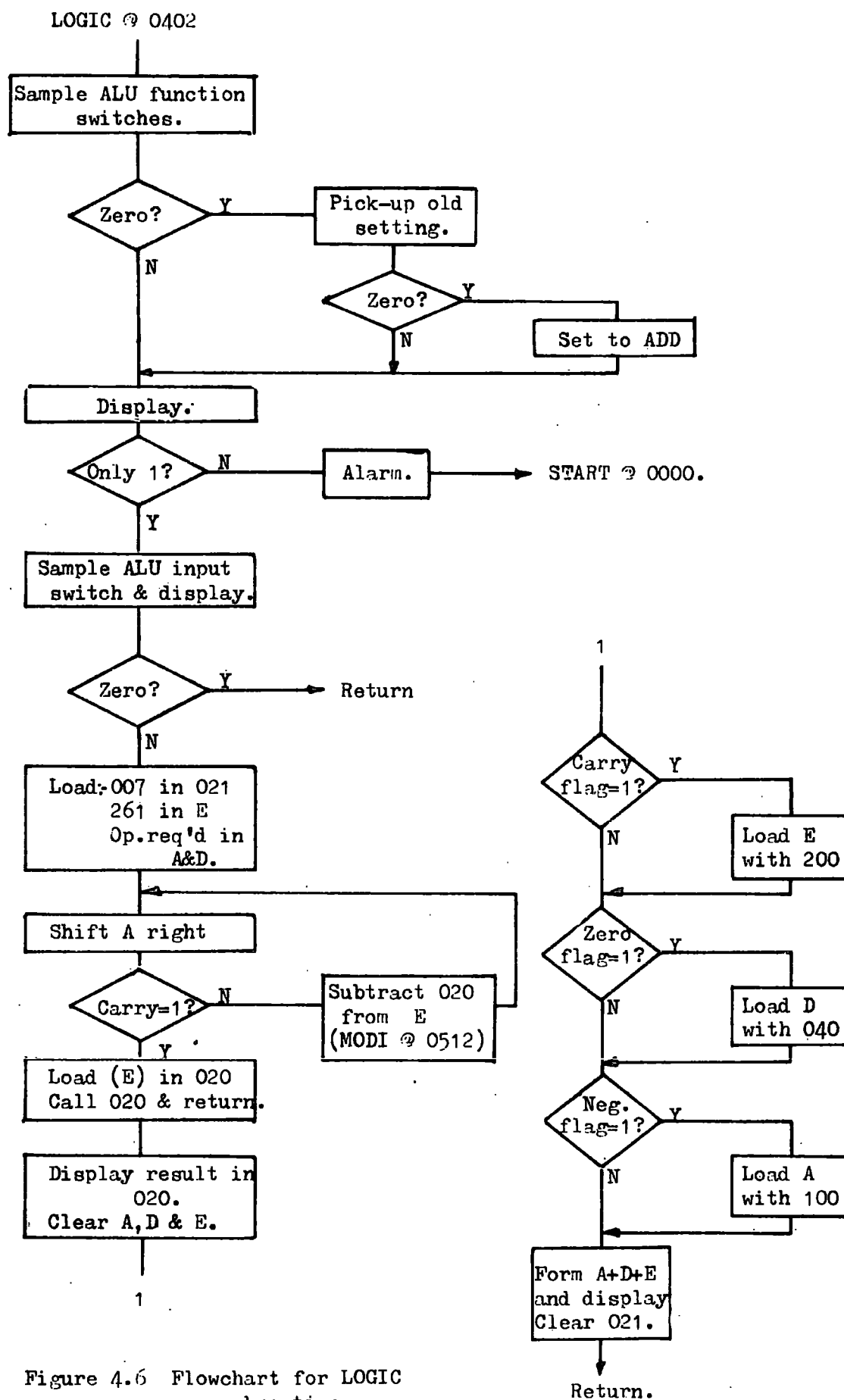


Figure 4.6 Flowchart for LOGIC subroutine.

the instruction from ORB to NDB to SUB to ADB. When the carry is set, the instruction from E is loaded in 020 of the shift register.

The instruction in 020 is called and upon the return it is overwritten with the result of the ALU operation, 021 is cleared. During this operation the microprocessor flags are set and the carry, negative and zero flags are collected and displayed in 032 of the shift register.

4.3.6 INSTN (INSTRUction register source gates)

The flow chart is shown in Figure 4.7.

The instruction source is either the store or the input toggle switches. The push-buttons controlling these gates are sampled and displayed. If neither is operated, the sub-routine is abandoned: if both are operated the alarm sequence is initiated. Otherwise, data is taken from the appropriate source and displayed in 024 of the shift register.

4.3.7 DECODI (DECODE Instruction)

It is necessary to decode the instruction for two distinct reasons:-

- a) to display the meaning of the instruction
- b) to execute the appropriate microinstructions

In a sub-routine for the first of these purposes the bulk of the instructions are required for decoding the instruction so there is no great penalty in repeating it, complete with the display section, for the second purpose. When only the display is required the sub-routine is abandoned at that point by a conditional return to the calling program, the flowchart for the sub-routine up to that point is shown in Figure 4.8; otherwise, the program continues in eight separate branches into the microprogram section to be described later and to which Figure 4.13 refers.

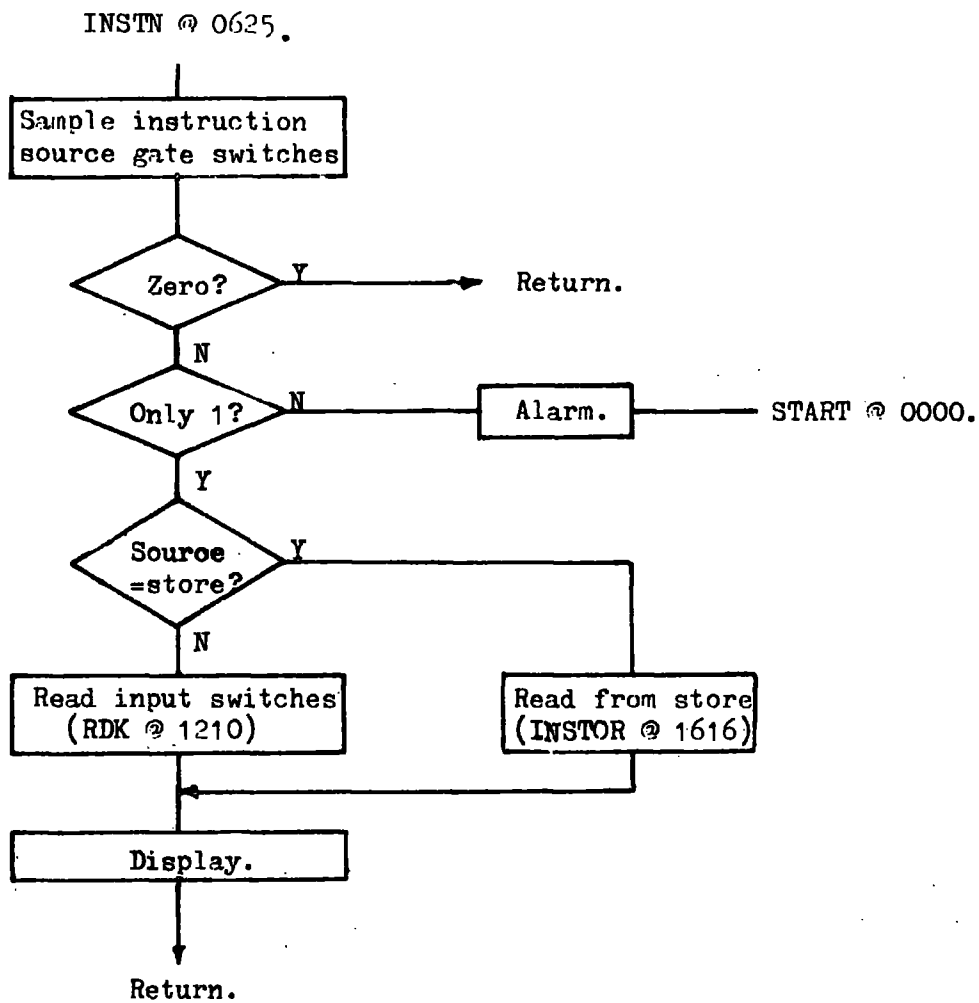
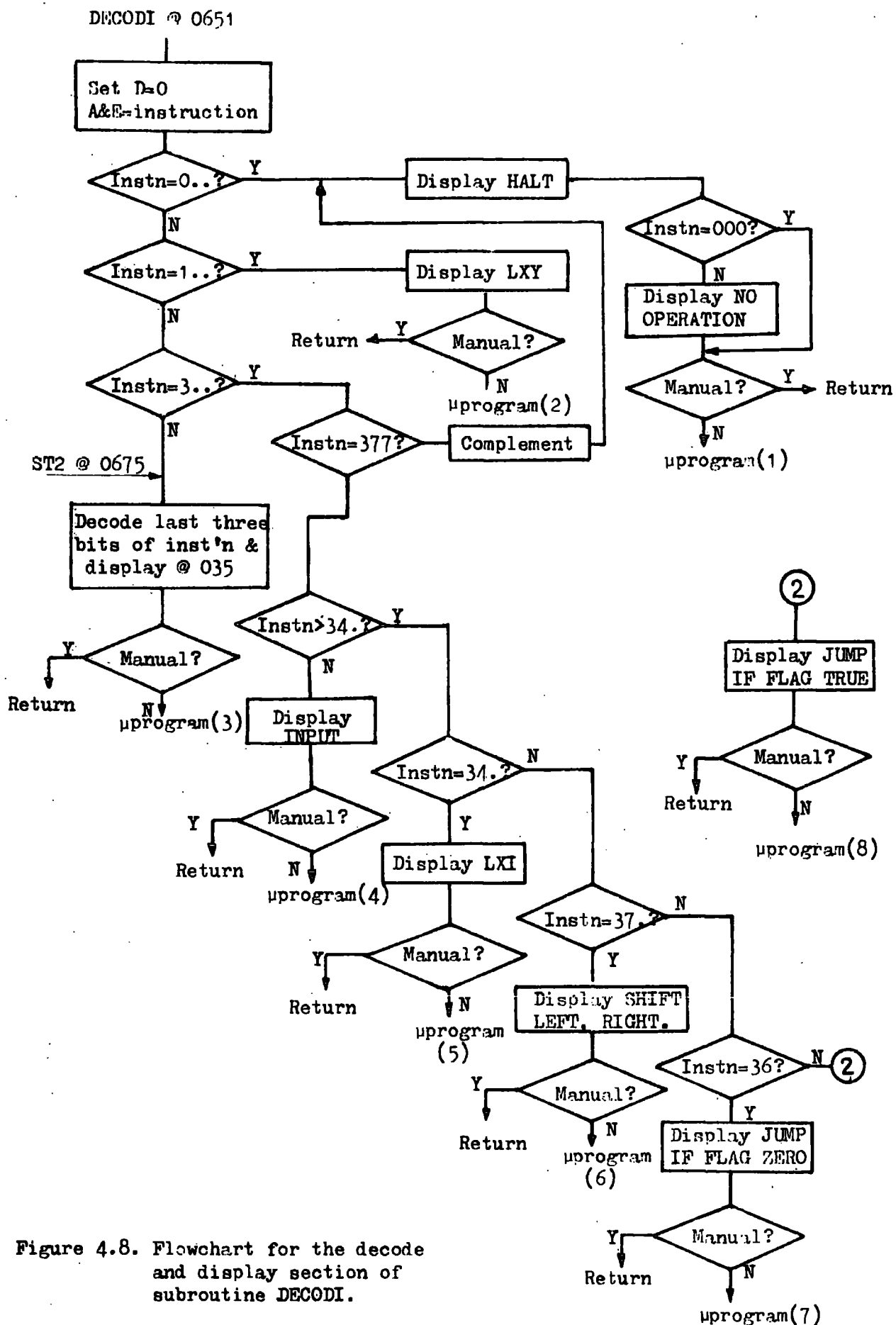


Figure 4.7. Flowchart for INSTN subroutine.



The decoded instructions are displayed in shift register locations 034 and 035 according to the following table:-

<u>Bit Number</u>	<u>Location 034</u>	<u>Location 035</u>
8 (MSB)	000, 377, Halt.	20X, Clear register X
7	1XY, Load reg X from Y	21X, Complement reg X
6	30X-33X, Input/output	22X, Increment reg X
5	34X, Load reg X immediately	23X, Decrement reg X
4	35, Jump if condition true	24X, Add (X) to Acc.
3	36, Jump if condition false	25X, Subtract (X) from Acc.
2	37, Shift left or right	26X, AND (X) with Acc.
1 (LSB)	OXY, No operation	27X, OR (X) with Acc.

The register D is used to distinguish between a display only use of the sub-routine and the extended use, in the former D is loaded with 000, in the latter case it is loaded with 001.

The first two bits of the instruction are examined for:-

- a) true zero - if so, jump to ST ϕ (first octal digit = 0)
- b) false sign - if so, jump to ST1 (first octal digit = 1)
- c) true parity - if so, jump to ST3 (first octal digit = 3)

Otherwise the first octal digit of the instruction is 2 and will be displayed in 035, 034 is cleared; the three middle bits are decoded to one of eight to determine which of the instructions to indicate.

If the first digit is 0, it is further decoded to 000 or OXY.

If the first digit is 1, the instruction is 'load X from Y'

If the first digit is 3, it is further decoded to:-

377	Halt
34	Input/output
34X	Load X immediately
35,36	Jump on true/false condition
37	Shift right or left

4.3.8 STGATE (Store address source GATES)

The flowchart for this sub-routine is shown in Figure 4.9.

The store can be addressed from the input toggle switches, the store-addressing register (5) or the program counter (6). The switches controlling these gates are sampled and displayed, then checked for zero and more than one as in previous cases. The appropriate data is then transferred to the store-address register in 022 of the shift register.

4.4 The Microprogram

The Educational Computer is represented as having eight processor states. Operation in the last four of these states is common to all instructions and is:-

- | | |
|---------|--|
| State 8 | Manual operation, all panel controls operative. |
| State 7 | Program counter incremented, gate from store to instruction register closed. |
| State 6 | Gate from store to instruction register opened, new instruction decoded. |
| | Gate from program counter to store-address register closed. |
| State 5 | Gate from program counter to store-address register opened. |
| | All gates open in state 4 closed. |

Operations in the remaining four states are shown in Figure 4.10.

It will be noticed that several instructions have operations in state 1 followed by a skip to state 4, the time spent in state 1 is very short while that in state 4 is determined by the speed control and may be much longer. The visual impression is given that states 1, 2 and

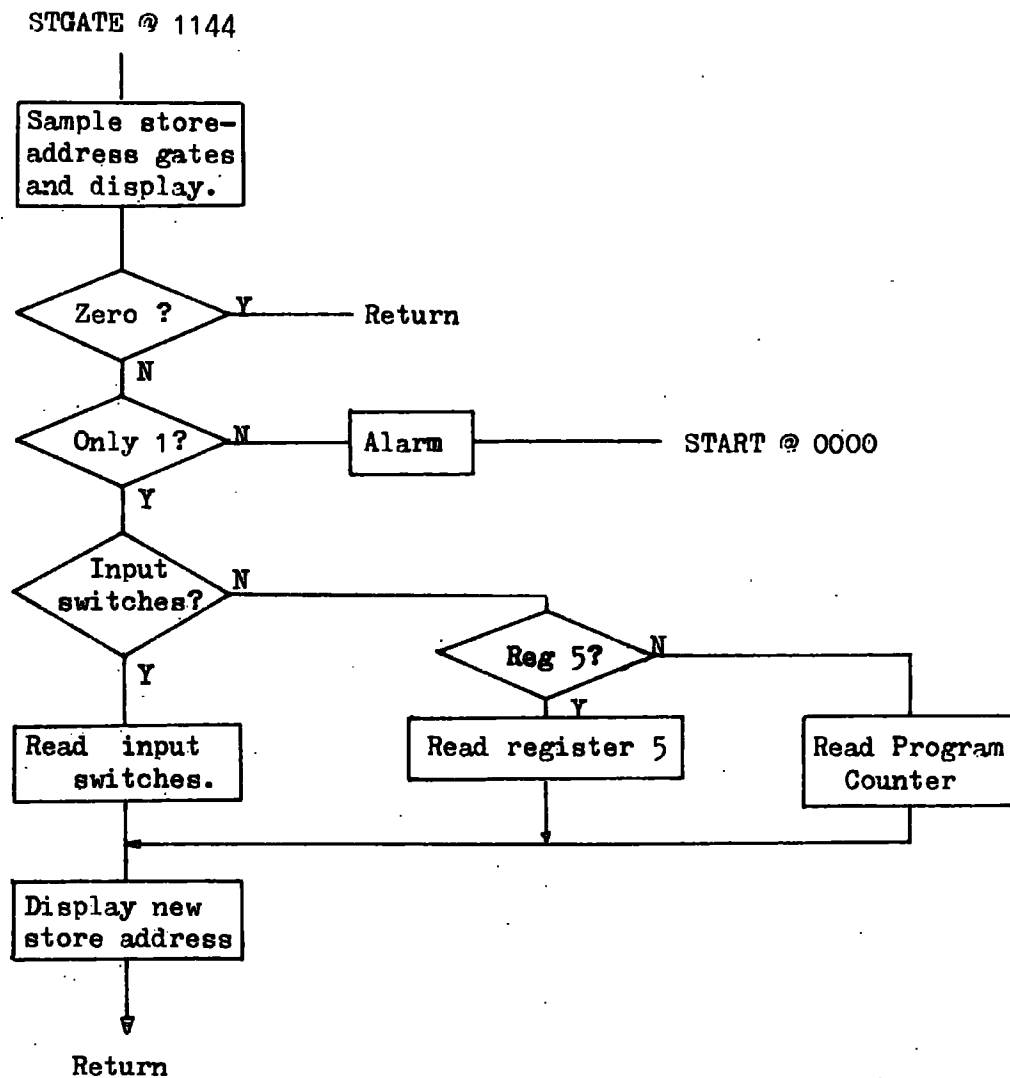


Figure 4.9 Flowchart for STGATE subroutine.

Instruction	State 1	State 2	State 3	State 4
000,377, Halt. 00Y, No operation. 31,32,33, Unused inputs	Load state counter with 020 i.e. skip to state 4. —————>			
1XY, Load X from Y.	Reg 5/store address gate open	Close.	Reg X/Bus gate open. Bus/Reg Y gate open.	Close
20X, Clear X. 21X, Complement X. 22X, Increment X. 23X, Decrement X.	Perform operation and skip. —————>			
24X, Add X to Acc. 25X, Subtract X from Acc. 26X, AND X with Acc. 27X, OR X with Acc.	Select ALU function. Reg 5/store address gate open. Reg X/Bus gate open.	Bus/ALU gate open. Close.	Close	ALU/Acc. gate open.
30X, Input to Reg X.	Reg 5/store address gate open	Close	Input/Bus gate open. Bus /Reg X gate open.	Close.
34X, Load reg X immediately	P.C./store address gate open.	Increment P.C. Close.	Store/Bus gate open. Bus/reg X gate open.	Close.
35,36, Conditional jumps.	Jump fails, increment P.C. and skip to state 4. —————>			
	P.C./store address gate open.	Close.	Store/Bus gate open. Bus/P.C. gate open.	Close.
37, Shift right, left.	Shift and skip to state 4. —————>			

Figure 4.10. Microprogram, states 1,2,3,4.

3 have been skipped rather than the actual skipping of states 2, 3 and 4. This skipping of states greatly simplifies the decoding of instructions in the various bit-times because many instructions simply do not occur in bit-times 2, 3 and 4. Only in bit-time 1 is full decoding of the instruction necessary and the consequent operations at this time are added to the sub-routine DECODI after the conditional return as has been explained in the previous section. In all other states, partial decoding is quite adequate and is generally brief when full account is taken of non-occurring conditions.

The flowchart of the microprogram with bits 4 to 8 in detail is shown in Figure 4.11. The operation required in any particular state can either be performed directly in this section or the accumulator can be loaded with a pattern which simulates the panel switches and a call made to one of the Manual sub-routines. If the operation can be performed in five machine instructions or less, the first course is preferable.

The flowcharts for states 2 and 3 are shown in Figure 4.12.

Due to the skip arrangement, the processor cannot enter state 3 unless the instruction is (a) one of the four ALU instructions starting with 2, (b) starts with 1 or (c) starts with 30 (input), 34 (Load X immediately) or 35, 36 a successful conditional jump.

The first of these requires the isolation of the ALU unit and this can be done without effect on the other operations. The remaining operations are all similar, being data transfers between registers and/or store. In each case the required initial conditions are set in the accumulator and the sub-routines DOUT, DIN called to effect the transfers.

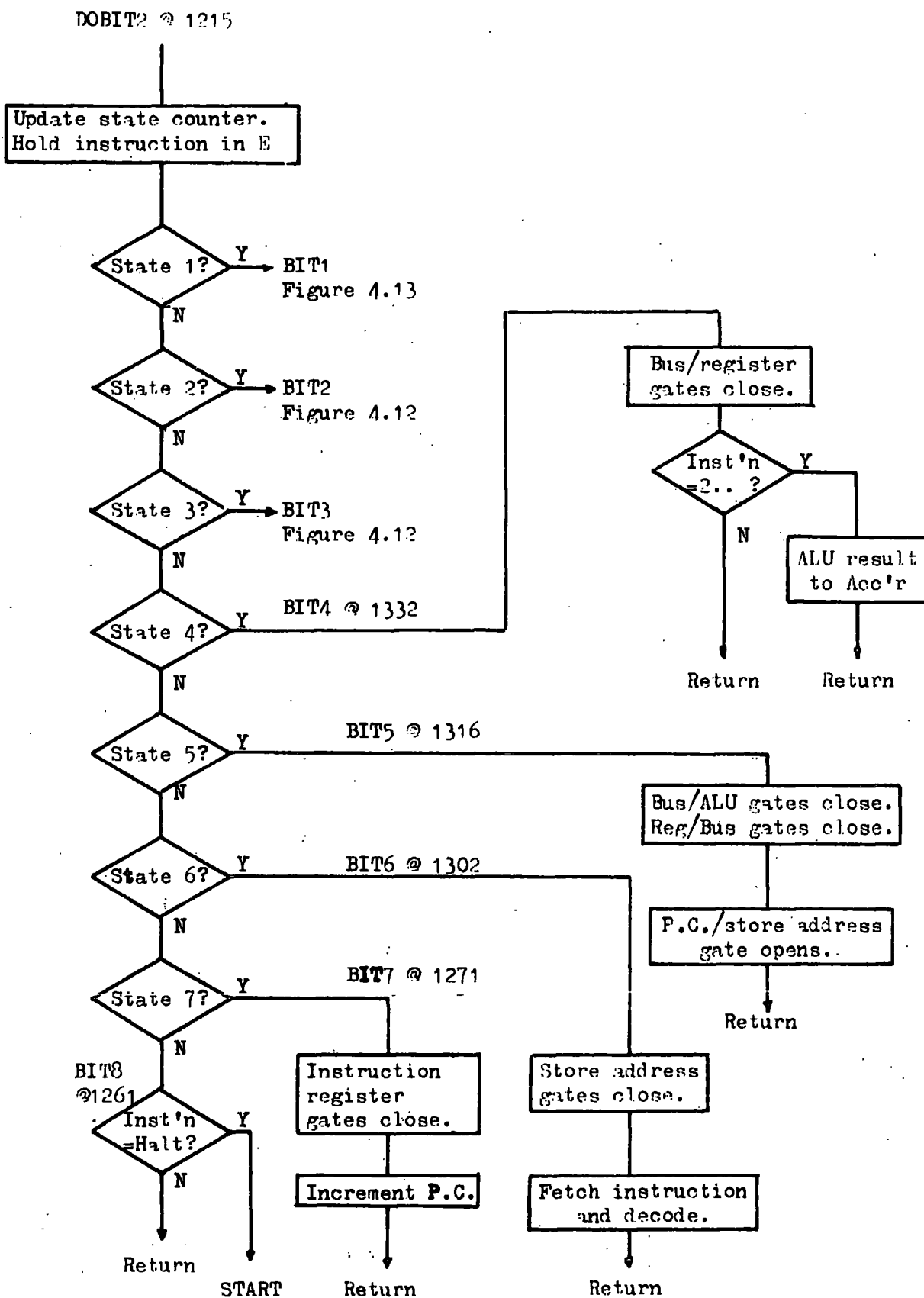


Figure 4.11. Flowchart for microprogram.

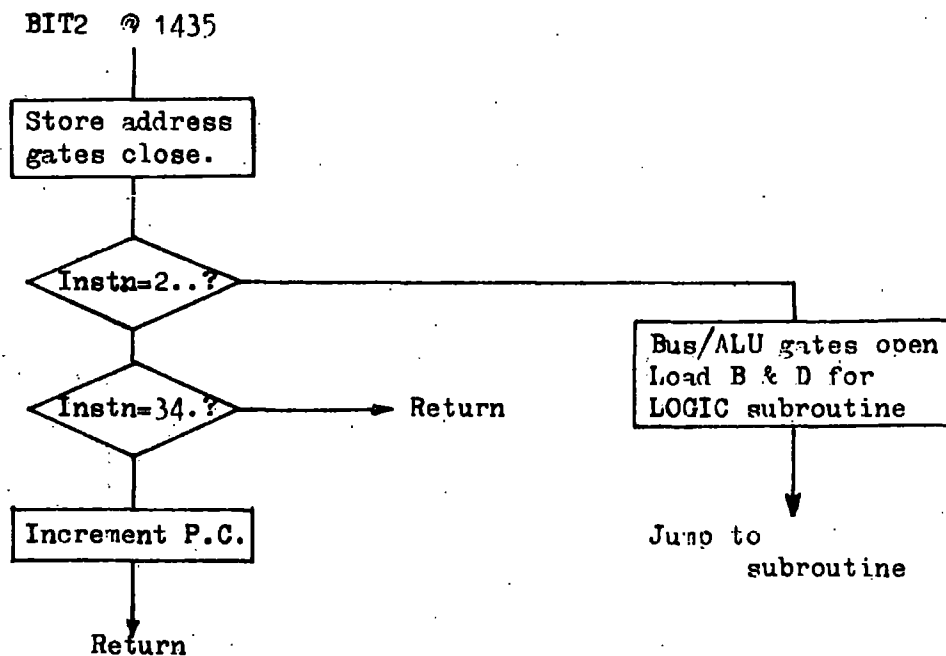
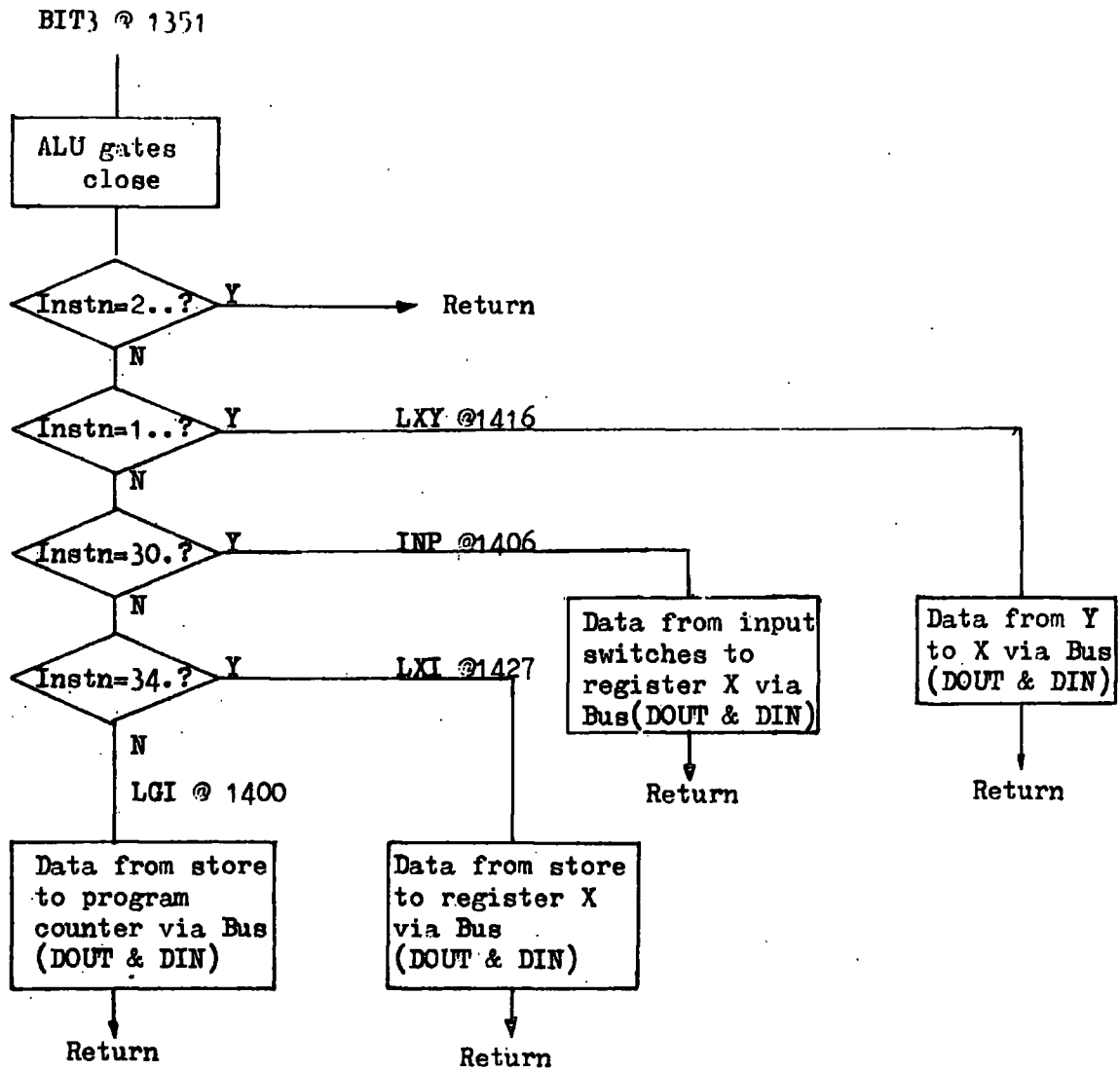


Figure 4.12. Flowcharts for microprogram states 2 & 3.

The same instructions occur in state 2; in the ALU instructions conditions are set up so that the LOGIC sub-routine can perform the operation, of the others, only 34X (load X immediately) requires an action: that the program counter be incremented.

In state 1 the instruction is fully decoded by the sub-routine DECODI because action is taken on every instruction at this time. Before calling DECODI the marker register D is set to 1 so that the conditional return after the display section is ignored. There are eight exits from the display section into the microprogram section and the flowcharts for these branches are shown in Figure 4.13.

If the instruction starts with zero (Halt or No operation) entry 1 is used, the state counter is advanced to 020 thus giving the effect of skipping states 1, 2 and 3.

Instructions starting with 1 use entry 2 in which the store is addressed from register 5, sub-routine STGATE is used for this after the code 002 has been set in the accumulator.

Instructions starting with 2 use entry 3 and are first sorted into the groups 20 - 23 and 24 - 27. In the first group, operations on memory are first excluded and then conditions set for sub-routine OP to implement the operation, this is followed by the skip to state 4. In the second group (the ALU instructions) the ALU function is determined from the instruction indicator, store is addressed from register 5 (STGATE) and DOUT is used to lift data from the register specified by the instruction. This data is stored in location 001 of the shift-register until required in state 2.

The remaining instructions are those starting with 3, the first group 30 - 33 are input/output of which only 30 is presently implemented.

Microprogram entries from DECODI (Figure 4.8.).

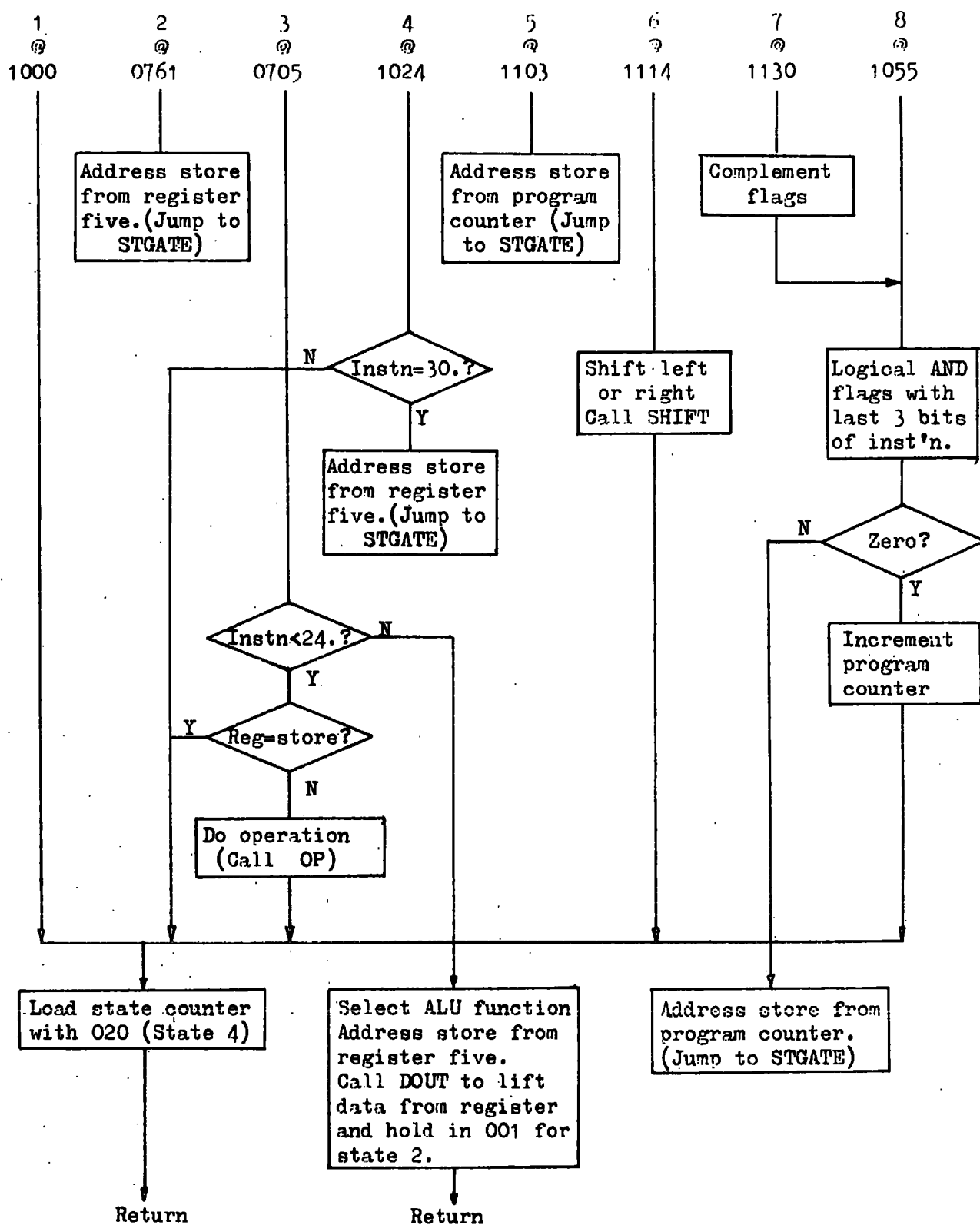


Figure 4.13. Microprogram for State 1 (following display section of DECODI)

These use entry 4 and instructions 31 - 33 are immediately separated and treated as no-operations. For 30, the store is addressed from register 5 by use of STGATE.

Instruction 34X, load X immediately, uses entry 5 when STGATE is used to address the store from the Program counter.

Instructions 35 and 36 are conditional jumps using entries 8 & 7 respectively. In 36 the flags are complemented before joining the 35 path to compare them with the flags specified in the instruction. If the comparison fails, the program counter is incremented and the state counter advanced to state 4; otherwise STGATE is used to address the store from the Program counter.

The remaining instruction, shift, uses entry 6. The direction depends on the LSB of the instruction so this is shifted into the Carry and sub-routine SHIFT called to perform the operation. This is followed by the skip to state 4.

CHAPTER 5:

Conclusions and Review of Project

An Educational Computer has been designed and a prototype built: this has been in use in the Department of Applied Physics and Electronics since September 1975 without major complaint. The main purpose of the machine is to provide a range of controls and modes of operation permitting study of the architecture and microprogram of a typical modern small computer. It is believed that this aim has been fully met. The machine is nevertheless capable of improvement in a number of respects.

5.1 Speed

When operating in the single instruction or continuous modes the machine has a range of speeds selected by an eight position rotary switch. At its slowest it executes one bit of an instruction in about two seconds so that a complete instruction with eight states takes sixteen seconds. This is sufficiently slow for all the steps to be followed visually on the panel if the operator knows what to look for.

At low speeds the processor spends most of its time in the delay shown in Figure 4.1 and this delay is successively halved as the speed setting is advanced until, at the maximum setting, the delay is entirely removed. At this stage the time per instruction varies with the number of instructions which the microprocessor has to carry out in emulating the instruction. It averages at about 10 instructions per second. This is very slow indeed by modern computer standards and when executing a repetitive program such as those of examples 2 and 5 of Appendix 2 it is noticeably so.

If it is decided that this slow speed is a major disadvantage, it can be enhanced either by program or by changes in the hardware. There are some changes in the hardware which are desirable in their own right and these would lead to an increase in speed anyway, they are reviewed in the next section.

In the program it would be possible to provide a new segment so that at the maximum speed setting all the intermediate microprogram steps would be eliminated, only the major registers would be updated on each instruction. The loss of the sequence of gates opening and closing would not be serious because they cannot be followed visually even at the present speeds.

Whether the speed needs to be increased depends on the use to which the machine is put. It is the author's opinion that serious machine-code programming should be undertaken on a microprocessor or minicomputer and that the extensive display panel of the present machine is irrelevant once the basic principles of number systems, architecture, micro-programming and order-code have been understood. From this point of view, only short programs will ever be used on the machine, the slow speed is no disadvantage and there is no pressing need to increase it.

5.2 Possible improvements in the Hardware

There can be little doubt that the multiplexed display led to a significant reduction both in the parts cost of the instrument and the area of printed circuit board required to carry the components. The use of a shift-register to present the words serially to the display was dictated by the fact that at that time read/write RAM was considerably more expensive and was organised as a large number of addresses each with only one bit. This is no longer the situation and presently a RAM unit of 256 words of 4-bits costs no more than one of the shift-register units used. Two advantages would accrue from the use of RAM; firstly the speed would be higher and more compatible with the TTL counter and secondly, random access would obviate the present need for the processor to enter a wait state until the correct location of the data comes round. This in itself would provide a significant speed improvement in the operation of the machine.

Had the project been undertaken a year later it is likely that the later Intel processor 8080 would have been used. This has a significantly higher speed and a wider range of instructions but would have required a different method of multiplexing the input push-buttons. Obviously the higher speed would have led to a higher speed in the emulated machine and the extended range of instructions could be expected to lead to a shorter program. The use of a higher speed processor would necessitate changes in the design of the multiplexed display which at present is synchronised with the processor and works at about its highest possible speed. Indeed, in contrast to the processor, a reduction in the display speed is desirable since that could be expected to reduce the crosstalk between adjacent display

words. By the use of RAM in place of the shift-register the display and processor clocks could be entirely divorced although it might be convenient for the display to use a sub-multiple of the processor clock.

There are a number of microprocessors available now and it may well be that one of the others would have led to a more efficient emulation. Without undertaking an extensive programming comparison it is impossible to know whether this is the case, experience in programming the 8008 showed that practice and familiarity led to a marked increase in the efficiency of the code written. The main competitor to Intel in the microprocessor field is Motorola and they claim that their order code is significantly more efficient than that of the 8080 so that a program for the 6800 is only approximately 70% of the length of an equivalent 8080 program.

Within the limitations of the microprocessor used it is believed that an efficient emulation program has been written. Many of the sub-routines have been revised in the course of development, always with a reduction in their length. It is impossible to know when a minimal program has been reached and while the program listed was thought to be minimal it is now known that there are several places in the program where instructions are redundant or where a jump rather than a call would save an instruction. In an Industrial situation a compromise must be reached between the excess cost in production of an inefficient program and the extra cost of time spent refining it. In this instance the PROM comes in 256-word units and the program occupies $3\frac{1}{2}$ of these, it is most unlikely that continued refinement would reduce the PROM requirement to three and even if this could be achieved, the extra PROM would have to be re-introduced if any enhancement were attempted.

5.3 Cost

The cost of components (November 1975) in this machine can be roughly broken down as follows:-

	£
Processor	20
PROM (4)	80
LEDs (130)	39
Switches	22
Power supply comp's	10
Power transistors	9
ICs (random logic)	30
PCB, Connectors etc.	<u>10</u>
Total	<u>220</u>

This could be reduced to below £200 by the use of cheaper LEDs but the penalty would be a less evenly lit display.

It is appreciated that the computer could have been built to the same specification from random logic and in doing so the expensive processor and PROMs would be removed from the cost list. In their place would be a substantial increase in the cost of random logic elements and in the size or number of PCBs and connectors. An overall reduction of component cost of about £50 would not seem unreasonable. However the component cost of electronic equipment is but a small part of the total and the hard-wired approach would lead to greater mechanical design and assembly costs. The greatest disadvantage of that system however would be the redesign necessary in the event of changes in the microprogram or enhancements being made. In the present machine major

changes or enhancements could be made simply by writing new program for the microprocessor. It can also confidently be anticipated that the cost of microprocessor chips and PROM will continue to fall and thereby erode any apparent cost advantage that a hard-wired design presently enjoys.

5.4 Two Recent Machines

During the course of this project the Open University introduced a small computer, OPUS, which is supplied to every student taking the course and is used in the study of machine-code programming (Ref. 8).

In order to keep the cost of each unit to a minimum the controls and information provided on the front panel are minimal. It is an 8-bit machine with 128 words of store (some of which are equivalent to registers) which can be loaded and monitored from a bank of eight toggle switches and lamps; the precise function of switches and lamps is determined by a further three toggle switches. The modes of operation, stop, single-shot and run are controlled by another two toggle switches.

The computer has two output ports connected to Minitrons and a single input port fed from a keyboard. These items are used in more advanced work after a loader program has been entered in the machine. The order code is very extensive and provides features which are found in mini-computers such as direct and indirect addressing, sub-routine linkage (by means of a stack pointer) and an interrupt.

It is clear that the design aims of this machine reflect the conditions under which many O.U. students work and insofar as these conditions do not obtain in a conventional university or college so the

teaching computer can be expected to have a different emphasis. Thus in the O.U. case large numbers of these machines are necessary so that each student can have his own and this dictates a minimum cost unit; in a conventional case only one may be required, the students using it in small groups in turn and it is free of the minimum cost constraint. More advanced aspects of programming can be taught on a minicomputer and the teaching machine can be aimed at the fundamental concepts. In the O.U. case the cost of providing the more advanced computer is prohibitive and it makes economic sense to squeeze some features of these machines into the basic tutor.

An alternative low cost method of teaching minicomputer programming and use is described by Sommer (Ref. 9) in which several computer panels are interfaced to a minicomputer (a NOVA 1200) using it on a time-sharing basis. Each panel can have peripherals attached to it and appears to the operator as a complete NOVA computer with facilities such as standard editor, assembler and BASIC programs although with a reduction of speed and memory. While this equipment has a role to play where large numbers of students have to use minicomputers simultaneously it is not applicable to the basic study of computer architecture at which the present machine is aimed.

5.5. Postscript to 5.3

Since section 5.3 was written, the cost of the 8702 PROM has fallen from £20 to £11 (February 1976). This reduces the parts cost by about £40 and there can now be little or no price advantage in a random logic design.

APPENDIX 1: Listing of microprocessor program

In the following fourteen pages the complete listing of the program is given. The first entry on each line is the address given in pure octal, thus the first PROM occupies 0000 to 0377

the 2nd PROM occupies 0400 to 0777

the 3rd PROM occupies 1000 to 1377

and the 4th PROM occupies 1400 to 1777.

The second entry on the line is the content of the address in octal.

The program is listed in pages of 100₈ lines each.

```

      DAT 0000
0000 300      START  LAA      / THIS IS START FOR MANUAL SEQUENCE
0001 056      LHI 377    / SET H ALL 1'S
0002 377
0003 026      LCI 000    / SET C ALL 0'S
0004 000
0005 104      JMP START2
0006 104
0007 000
0010 006      DOUT  LAI 010
0011 010
0012 103      INP 1
0013 310      LBA      /SAMPLE BUS-K SWITCH & STORE IN B
0014 006      LAI 007
0015 007
0016 103      INP 1    / SAMPLE BUS-REG SWITCHES
0017 300      LAA
0020 106      DOUT2  CAL BSLMPS  /& REARRANGE. ENTER HERE FROM DOBIT.
0021 366
0022 000
0023 066      LLI 026
0024 026
0025 104      JMP DOUT3
0026 074
0027 003
0030 307      TEST  LAM
0031 012      RRC
0032 140      JTC START
0033 000
0034 000
0035 055      RST 5    / ST/ST
0036 007      RET
0037 300      LAA
0040 330      ONLY1  LDA  / KEEP COPY OF A IN D
0041 022      RAL
0042 100      JFC ONLY1'+1
0043 127
0044 003
0045 104      JMP ONLY1'
0046 126
0047 003
0050 066      ST/ST  LLI 013    / THIS SUBROUTINE SAMPLES THE START
0051 013
0052 306      LAL      / AND STOP BUTTONS. IF START HAS
0053 103      INP 1    / GONE 0 TO 1 SINCE LAST SAMPLE IT
0054 340      LEA      /SETS THE SIGN FLAG AND RETURNS. IF
0055 257      XRM      / STOP IS 1 IT SETS THE CARRY FLAG
0056 244      NDE      / AND RETURNS.
0057 374      LME
0060 304      LAE
0061 032      RAR
0062 365      LLH
0063 007      RET
0064 304      CYCLE1  LAE  /PICK UP INSTN
0065 032      RAR
0066 032      RAR
0067 032      RAR
0070 044      NDI 007    / TAKE R.H. THREE BITS
0071 007
0072 320      LCA
0073 006      LAI 001  /THIS SUBROUTINE CONVERTS AN OCTAL
0074 001
0075 012      RRC      / NUMBER TO A SINGLE 1 IN THE PROPER
0076 021      DCC      / BIT POSITION OF THE ACCUMULATOR
0077 120      JFS *-2

```

0100	075	
0101	000	
0102	020	INC
0103	007	RFT
0104	365	START2 LLH
0105	076	LMI 001 / PROCESSOR STATE INDICATOR IS 001
0106	001	
0107	106	CAL PANEL / SERVICES LOWER PANEL SWITCHES
0110	240	
0111	000	
0112	015	RST 1 / =DOOT / SERVICES BUS-REG SWITCHES
0113	106	CAL DIN / SERVICES REG-BUS SWITCHES
0114	332	
0115	000	
0116	106	CAL LOGIC / SERVICES A & LU.
0117	002	
0120	001	
0121	106	CAL OPONA / SERVICES A-L, SR, SL SWITCHES
0122	121	
0123	001	
0124	106	CAL INSTN / SERVICES INSTN REGISTER GATES
0125	225	
0126	001	
0127	106	CAL DECODI / DECODES INSTN REGISTER
0130	251	
0131	001	
0132	106	CAL STGATE / SERVICES STORE ADDRESS GATES
0133	144	
0134	002	
0135	006	MODE LAI 002
0136	002	
0137	103	INP 1
0140	022	RAL
0141	140	JTC START / MODE SWITCH AT MANUAL
0142	000	
0143	000	
0144	055	RST 5 / = ST/ST, NOT MANUAL, BEEN STARTED?
0145	120	JFS START / NO, GO TO MANUAL
0146	000	
0147	000	
0150	106	DOBIT1 CAL DOBIT2
0151	215	
0152	002	
0153	365	LLH
0154	006	LAI 002
0155	002	
0156	103	INP 1
0157	022	RAL
0160	022	RAL
0161	140	JTC ONEBIT
0162	177	
0163	000	
0164	022	RAL
0165	140	JTC ONEINS
0166	206	
0167	000	
0170	022	RAL
0171	140	JTC C15
0172	215	
0173	000	
0174	076	LMI 000
0175	000	
0176	005	RST START
0177	035	ONEBIT RST 3 / (TEST), GOES TO MANUAL AT END OF INSTN

0200	160	JTS DOBIT1
0201	150	
0202	000	
0203	104	JMP ONEBIT / WAIT FOR START
0204	177	
0205	000	
0206	035	ONINS RST 3 / (TEST), MANAL IF END OF INSTN
0207	100	JFC DELAY2 / DO BIT OF INSTN IF STOP NOT 1
0210	226	
0211	000	
0212	104	JMP ONFINS / OR WAIT FOR START
0213	206	
0214	000	
0215	307	CTS LAM
0216	012	RMC
0217	100	JFC DELAY2 / SAMPLE STOP AT END OF INSTN ONLY
0220	226	
0221	000	
0222	055	RST 5 / (ST/ST)
0223	140	JTC START
0224	000	
0225	000	
0226	006	DELAY2 LAI 003
0227	003	
0230	103	INP 1
0231	310	LBA / SAMPLE SPEED CONTROL
0232	106	CAL DELAY+6
0233	156	
0234	003	
0235	104	JMP DOBIT1
0236	150	
0237	000	
0240	302	PANEL LAC
0241	066	LLI 001
0242	001	
0243	103	INP 1 / SAMPLE CLR CPL INC & DEC SWITCHES
0244	260	ORA / SET FLAGS
0245	150	JTZ CLRM / QUIT IF NONE OPERATED
0246	320	
0247	000	
0250	277	CFM / SWITCHES SAME AS LAST SAMPLE?
0251	110	JFZ CLRM / QUIT IF NOT
0252	320	
0253	000	
0254	045	RST 4 / (ONLY1)
0255	306	LAL
0256	066	LLI 003
0257	003	
0260	103	INP 1 / SAMPLE REGISTER SELECT SWITCHES
0261	260	ORA / SET FLAGS
0262	150	JTZ SETM2 / QUIT IF NONE OPERATED
0263	330	
0264	000	
0265	277	CFM / COMPARE WITH LAST SAMPLE
0266	110	JFZ SETM2 / QUIT IF DIFFERENT
0267	330	
0270	000	
0271	066	LLI 005
0272	005	
0273	340	LEA / E HAS SWITCH SETTINGS
0274	257	XRM / A HAS CHANGES FROM LAST SAMPLE
0275	244	NDE / A HAS 1-0 CHANGES ONLY
0276	310	LBA / KEEP IN B
0277	267	ORM / OR WITH PREVIOUS 1-0 CHANGES

0300	370	LMA	/ AND STORE
0301	301	LAB	
0302	066	LLI	016
0303	016		
0304	012	OPREG	RRC
0305	310	LBA	/ THE 1-0 CHANGES ARE ROTATED THROUGH
0306	142	CTC	OP / THE ACC AND THE CORRESPONDING
0307	163		
0310	003		
0311	301	LAB	/ REGISTERS OPERATED ON BY S.R.
0312	061	DCL	/ 'OP'
0313	061	DCL	
0314	063	RTS	
0315	104	JMP	OPREG
0316	304		
0317	000		
0320	066	CLRM	LLI 005 / SWITCHES ARE ZERO OR CHANGED
0321	005		
0322	372	LMC	/ FROM LAST SAMPLE SO CLEAR TEMP
0323	061	DCL	/ STORE POSITIONS
0324	061	DCL	
0325	372	LMC	
0326	061	DCL	
0327	061	DCL	
0330	370	SETM2	LMA
0331	007		RET
0332	006	DIN	LAI 004
0333	004		
0334	103	INP	1 /SAMPLE BUS TO REGISTER SWITCHES
0335	106	CAL	BSLMPS / & REARRANGE. ENTER HER FROM DOBIT
0336	366		
0337	000		
0340	066	LLI	027
0341	027		
0342	370	LMA	/ AND DISPLAY
0343	053	RTZ	/ QUIT IF ALL SWITCHES OFF
0344	303	LAD	
0345	012	RRC	
0346	142	CTC	LDSTOR /PUT BUS DATA IN
0347	227		
0350	003		
0351	066	LLI	016 / SELECTED REGISTERS
0352	016		
0353	012	RRC	
0354	061	DCL	
0355	061	DCL	
0356	063	RTS	/ QUIT IF ALL DONE
0357	100	JFC	*-4
0360	353		
0361	000		
0362	371	LMB	
0363	104	JMP	*-10
0364	353		
0365	000		
0366	330	BSLMPS	LDA
0367	044		NDI 170
0370	170		
0371	150		JTZ**5
0372	376		
0373	000		
0374	006	LAI	100
0375	100		
0376	263	ORD	
0377	044	NDI	307

0400	307		RET
0401	007		
0402	066	LOGIC	LLI 031
0403	031		
0404	006		LAI 006
0405	006		
0406	103		INP 1 / SAMPLE ADD, SUB, AND, OR SWITCHES
0407	260		ORA / SET FLAGS
0410	150		J1Z **4
0411	014		
0412	001		
0413	370		LMA / SWITCHES NOT ZERO, CHANGE DISPLAY
0414	267		ORM / IF ZERO USE OLD DISPLAY
0415	110		JFZ **5
0416	022		
0417	001		
0420	076		LMI 010 / IF STILL ZERO SET TO ADD
0421	010		
0422	045		RST 4 / (ONLY1)
0423	006	ATOL	LAI 005
0424	005		
0425	103		INP 1 / SAMPLE A TO L BUTTON
0426	066		LLI 017
0427	017		
0430	370		LMA / AND DISPLAY
0431	260		ORA
0432	053		RTZ
0433	066		LLI 021
0434	021		
0435	076		LMI 007 / LOAD RET IN 021
0436	007		
0437	046		LEI 261 / FORM ORB INSTN IN REG E
0440	261		
0441	303	FLAGS	LAD
0442	032		RAR / 'A' HAS ADD SUB ETC
0443	330		LDA
0444	100		JFC MOD1 / MODIFY ORB TO NDB ETC IF C=0
0445	112		
0446	001		
0447	362		LLC
0450	307		LAM / LOAD 'A' FROM ACC
0451	066		LLI 020
0452	020		
0453	374		LME / PUT THE MODIFIED INSTN INTO 077 020
0454	106		+106 / CALL THE OPERATION
0455	020		+020
0456	077		+077
0457	370		LMA / DISPLAY LOGIC RESULT
0460	302		LAC
0461	332		LDC / CLEAR A, D, AND E FOR FLAGS
0462	342		LFC
0463	100		JFC ZEROF
0464	070		
0465	001		
0466	046		LEI 200 / E HAS CARRY FLAG
0467	200		
0470	110	ZEROF	JFZ NEGF
0471	075		
0472	001		
0473	036		LDI 040 / D HAS ZERO FLAG
0474	040		
0475	120	NEGF	JFS SHOFLG
0476	102		
0477	001		

0500	006		LAI 100 / 'A' HAS SIGN FLAG
0501	100		
0502	264	SHOFLG	ORE
0503	263		ORE / COMBINE FLAGS IN 'A'
0504	060		INL
0505	372		LMC / CLEAR THE RETURN INSTRUCTION FROM 077 021
0506	066		LLI 032
0507	032		
0510	370		LMA / DISPLAY FLAGS
0511	007		RET
0512	304	MODI	LAE
0513	024		SUI 020 / CHANGE OR TO AND TO SUB TO ADD
0514	020		
0515	340		LEA
0516	104		JMP FLAGS
0517	041		
0520	001		
0521	066	OPONA	LLI 030
0522	030		
0523	006		LAI 014 / SAMPLE AND DISPLAY A-L, SK SL SWITCHES
0524	014		
0525	103		INP 1
0526	370		LMA
0527	260		ORA / SET FLAGS
0530	110		JFZ **7 / IF SWITCH(S) OPERATED
0531	137		
0532	001		
0533	066		LLI 016
0534	016		
0535	372		LMC / CLEAR TEMPORARY STORE
0536	007		RET / & QUIT
0537	006		LAI 004
0540	004		
0541	103		INP 1 / SAMPLE A-BUS SWITCH
0542	022		KAL
0543	307		LAM
0544	045		RST 4 / (ONLY 1)
0545	066		LLI 016
0546	016		
0547	247		NDM / IF M NOT ZERO, OPERATION ALREADY DONE
0550	013		RFZ / SO QUIT
0551	375		LMH / SET M=377 & DO OPERATION
0552	303		LAD
0553	022		KAL
0554	022		KAL
0555	140		JTC LTOA
0556	217		
0557	001		
0560	022		KAL
0561	066	SHIFT	LLI 030
0562	030		
0563	140		JTC SHFTL
0564	201		
0565	001		
0566	076		LMI 020 / SHIFT ACC ONE PLACE RIGHT
0567	020		
0570	066		LLI 032
0571	032		
0572	307		LAM
0573	022		KAL
0574	362		LLC
0575	307		LAM
0576	032		KAL
0577	370		LMA

0600	007		RFT
0601	076	SHFTL	LMI 040 / SHIFT ACC ONE PLACE LEFT
0602	040		
0603	362		LLC
0604	307		LAM
0605	260		ORA
0606	022		RAL
0607	370		LMA
0610	066		LLI 032
0611	032		
0612	307		LAM
0613	022		RAL
0614	012		RRC
0615	370		LMA
0616	007		RET
0617	066	LTOA	LLI 020 / TRANSFER LOGIC RESULT TO ACC.
0620	020		
0621	307		LAM
0622	362		LLC
0623	370		LMA
0624	007		RET
0625	006	INSTN	LAI 012
0626	012		
0627	103		INP 1 / SAMPLE INSTN REG GATE SWITCHES
0630	066		LLI 036
0631	036		
0632	370		LMA / AND DISPLAY
0633	260		ORA / SET FLAGS
0634	053		RTZ / AND QUIT IF ZERO
0635	045		RST 4 / (ONLY 1)
0636	032		RAR
0637	142		CTC INSTOR / LOAD REG FROM STORE, OR
0640	216		
0641	003		
0642	102		CFC RDK / LOAD FROM SWITCHES
0643	210		
0644	002		
0645	066		LLI 024
0646	024		
0647	371		LMB
0650	007		RET
0651	332	DECODI	LDC / D=0 ON MANUAL OP, D=1 ON AUTO OP
0652	066		LLI 024 / ADDRESS OF INSTN
0653	024		
0654	307		LAM
0655	340		LEA / INSTN IN A & F
0656	066		LLI 035
0657	035		
0660	372		LMC / CLEAR INSTN INDICATOR 035
0661	061		DCL / L POINTS TO 0, 1 & 3 INSTNS IN 034
0662	044		NDI 300
0663	300		
0664	150		JTZ ST0 / INSTN STARTS WITH 0
0665	366		
0666	001		
0667	120		JFS ST1 / INSTN STARTS WITH 01
0670	355		
0671	001		
0672	170		JTP ST3 / INSTN STARTS WITH 11
0673	004		
0674	002		
0675	106	ST2	CAL CYCLE1 / DECODE THE 8 INSTNS WHICH START 2
0676	064		
0677	000		

0700	372	LMC / CLEAR INSTN INDICATOR 034
0701	060	INL
0702	370	LMA / DISPLAY IN 077 035
0703	031	DCD
0704	013	RFZ / QUIT IF NOT BIT 1, I.E. MANUAL OP'N
0705	044	NDI 360 / FIRST FOUR BITS?
0706	360	
0707	150	JTZ LAST4
0710	334	
0711	001	
0712	330	LDA / D HAS OPERATION CLR,CPL,INC,DEC.
0713	304	LAE / PICK UP INSTN
0714	044	NDI 007
0715	007	
0716	074	CPI 007 / EXCLUDE MEMORY
0717	007	
0720	150	JTZ ST3-4 / NO OPERATION
0721	000	
0722	002	
0723	002	RLC
0724	360	LLA / L HAS ADDRESS OF REGISTER FOR OPERATION
0725	303	LAD / A HAS REQUIRED OPERATION
0726	106	CAL OP+1
0727	164	
0730	003	
0731	104	JMP ST3-4
0732	000	
0733	002	
0734	307	LAST4 LAM
0735	066	LLI 031
0736	031	
0737	370	LMA / ADD, SUB, AND, OR, SET
0740	006	LAI 002
0741	002	
0742	106	CAL STGATE+3 / STORE ADDRESS-REG F
0743	147	
0744	002	
0745	304	LAE / A HAS INSTN
0746	075	RST 7 / (CYCLE1+4)
0747	312	LBC / CLEAR B
0750	025	RST 2 / (DOUT2)
0751	066	LLI 001
0752	001	
0753	371	LMB / KEEP RECORD OF BUS DATA FOR BIT2
0754	007	RET
0755	076	ST1 LMI 100
0756	100	
0757	031	DCD
0760	013	RFZ /QUIT IF NOT BIT 1
0761	006	LAI 002
0762	002	
0763	104	JMP STGATE+3 / ADDRESS STORE FROM F
0764	147	
0765	002	
0766	204	ST0 ADE /ADD INSTN TO EMPTY ACC
0767	076	LMI 200
0770	200	
0771	150	JTZ **5
0772	376	
0773	001	
0774	076	LMI 001
0775	001	
0776	031	DCD
0777	013	RFZ

1000	365		LLH / START HERE IF 'NO OPERATION'
1001	076		LMI 020
1002	020		
1003	007		REF
1004	304	ST3	LAE
1005	255		XRH
1006	150		JTZ ST0+1 / INSTN IS 377
1007	367		
1010	001		
1011	304		LAE
1012	022		RAL
1013	022		RAL
1014	022		RAL
1015	140		JTC >34 / INSTN IS 34, 35, 36, OR 37
1016	036		
1017	002		
1020	076	<34	LMI 040 /INSTN IS INPUT OR OUTPUT
1021	040		
1022	031		DCD
1023	013		RFZ / QUIT IF NOT BIT1
1024	044		NDI 300 / INPUT OR OTHER?
1025	300		
1026	110		JFZ ST3-4 / NO OPERATION
1027	000		
1030	002		
1031	006		LAI 002
1032	002		
1033	104		JMP STGATE +3
1034	147		
1035	002		
1036	044	>34	NDI 300
1037	300		
1040	150		JTZ =34 /INSTN IS 34
1041	077		
1042	002		
1043	170		JTP =37 /INSTN IS 37
1044	110		
1045	002		
1046	160		JTS =36 /INSTN IS 36
1047	124		
1050	002		
1051	076	=35	LMI 010 /INSTN IS 35 (JMP IF COND'N TRUE
1052	010		
1053	031		DCD
1054	013		RFZ / QUIT IF NOT BIT1
1055	066		LLI 032
1056	032		
1057	307		LAM / PICK UP FLAGS
1060	002		RLC
1061	002		RLC
1062	002		RLC
1063	244		NDE / & COMPARE WITH INSTRUCTION
1064	006		LAI 004
1065	004		
1066	110		JFZ STGATE+3 /ADDRESS STORE FROM PC IF JUMP
1067	147		
1070	002		
1071	106		CAL INCG
1072	274		
1073	002		
1074	104		JMP ST3-4 /INC PC AND SKIP STATES IF COND FAILS
1075	000		
1076	002		
1077	076	=34	LMI 020 /INSTN IS LOAD X IMMEDIATELY

1100	020		
1101	031		DCD
1102	013		RFZ / QUIT IF NOT BIT1
1103	006		LAI 004
1104	004		
1105	104		JMP STGATE+3 / ADDRESS STORE FROM PC
1106	147		
1107	002		
1110	076	=37	LMI 002 /INSTN IS SHIFT RIGHT OR LEFT
1111	002		
1112	031		DCD
1113	013		RFZ / QUIT IF NOT BIT1
1114	304		LAE /PICK UP INSTN
1115	032		RAR
1116	106		CAL SHIFT
1117	161		
1120	001		
1121	104		JMP ST3-4
1122	000		
1123	002		
1124	076	=36	LMI 004 /INSTN IS JUMP IF CONDITION FALSE
1125	004		
1126	031		DCD
1127	013		RFZ
1130	066		LLI 032
1131	032		
1132	307		LAM
1133	002		RLC
1134	002		RLC
1135	002		RLC
1136	255		XRH
1137	044		NDI 007
1140	007		
1141	104		JMP =35+12
1142	063		
1143	002		
1144	006	STGATE	LAI 011 / STORE ADDRESS GATE SWITCHES
1145	011		
1146	103		INP 1
1147	066		LLI 033
1150	033		
1151	370		LMA / SAMPLE SWITCHES AND DISPLAY
1152	260		ORA /SET FLAGS
1153	053		RTZ / & QUIT IF ZERO
1154	045		RST 4 / (ONLY1)
1155	032		RAR
1156	100		JFC XFRF
1157	167		
1160	002		
1161	106		CAL RDK / READ INPUT SWITCHES
1162	210		
1163	002		
1164	104		JMP STADD / & LOAD ADDRESS REGISTER
1165	204		
1166	002		
1167	032	XFRF	RAR
1170	100		JFC XFRG
1171	201		
1172	002		
1173	066		LLI 012 / READ REG '5'
1174	012		
1175	307		LAM
1176	104		JMP STADD / & LOAD ADDRESS REGISTER
1177	204		

1200	002		
1201	066	XFRG	LLI 014 / READ PROGRAM COUNTER
1202	014		
1203	307		LAM
1204	066	STADD	LLI 022 / LOAD ADDRESS REGISTER
1205	022		
1206	370		LMA
1207	007		RET
1210	305	RDK	LAH / READ THE INPUT SWITCH REGISTER
1211	103		INF 1
1212	255		XRH
1213	310		LBA / & HOLD IN 'B'
1214	007		RET
1215	365	DOBIT2	LLH
1216	307		LAM
1217	012		RHC
1220	260		OKA / CLEAR CARRY
1221	370		LMA / PROCESSOR STATE UPDATED & HELD IN 'A'
1222	066		LLI 024
1223	024		
1224	347		LEM / E HAS ORIGINAL INSTN
1225	022		KAL
1226	140		JTC BIT1
1227	067		
1230	003		
1231	022		KAL
1232	140		JTC BIT2
1233	035		
1234	003		
1235	022		KAL
1236	140		JTC BIT3
1237	351		
1240	002		
1241	022		KAL
1242	140		JTC BIT4
1243	332		
1244	002		
1245	022		KAL
1246	140		JTC BIT5
1247	316		
1250	002		
1251	022		KAL
1252	140		JTC BIT6
1253	302		
1254	002		
1255	022		KAL
1256	140		JTC BIT7
1257	271		
1260	002		
1261	066	BIT8	LLI 034
1262	034		
1263	307		LAM
1264	260		OKA
1265	160		JTS START / GO TO MANUAL OP. IF INSTN IS HLT
1266	000		
1267	000		
1270	007		RET
1271	066	BIT7	LLI 036
1272	036		
1273	372		LMC / CLOSE STORE TO INSTN GATE
1274	066	INCG	LLI 014
1275	014		
1276	347		LEM
1277	040		INE

1300	374		LME / INCREMENT 'G' (PROGRAM COUNTER)
1301	007		RET
1302	066	BIT6	LLI 033
1303	033		
1304	372		LMC / CLOSE STORE ADDRESS GATE
1305	006		LAI 001
1306	001		
1307	106		CAL INSTN+3 /FETCH INSTN FROM STORE
1310	230		
1311	001		
1312	106		CAL DECODI / AND DECODE INSTN
1313	251		
1314	001		
1315	007		RET
1316	066	BIT5	LLI 030
1317	030		
1320	372		LMC /CLOSE L TO A, A TO L ETC. GATES
1321	066		LLI 026
1322	026		
1323	372		LMC / CLOSE REGISTER TO BUS GATES
1324	006		LAI 004
1325	004		
1326	106		CAL STGATE+3 /ADDRESS STORE FOR NEXT INSTN
1327	147		
1330	002		
1331	007		RET
1332	066	BIT4	LLI 027
1333	027		
1334	372		LMC / CLOSE BUS TO REGISTER GATES
1335	066		LLI 035
1336	035		
1337	267		ORM / ACC HAS 2'S INSTN INDICATOR
1340	053		RTZ / RETURN IF INSTN NOT 2
1341	066		LLI 030
1342	030		
1343	076		LMI 100
1344	100		
1345	106		CAL LTOA /LOGIC RESULT TO ACC
1346	217		
1347	001		
1350	007		RET
1351	066	BIT3	LLI 030
1352	030		
1353	372		LMC /CLOSE A-L,L-A,SR,SL GATES
1354	066		LLI 034
1355	034		
1356	267		ORM / ACC HAS 0,1,3, INSTN INDICATOR
1357	053		RTZ / QUIT IF 2 INSTN
1360	312		LBC / CLEAR B COS 'DOUT' USES B TO REMEMBER K
1361	022		RAL
1362	022		RAL
1363	140		JTC LXY
1364	016		
1365	003		
1366	022		RAL
1367	140		JTC INP
1370	006		
1371	003		
1372	022		RAL
1373	006		LAI 001 /FOR BUS-STORE IN LXI,LGI
1374	001		
1375	140		JTC LXI
1376	027		
1377	003		

1400	025	LGI	RST 2 / (DOUT2), INSTN IS CONDITIONAL JUMP
1401	006		LAI 002
1402	002		
1403	104		JMP DIN+3 /TO LOAD G AND DISPLAY GATE
1404	335		
1405	000		
1406	016	INP	LBI 040 /TO OPEN BUS-K GATE
1407	040		
1410	025		RST 2 / (DOUT2),
1411	304		LAE
1412	075		RST 7 / (CYCLE1+4)
1413	104		JMP DIN+3
1414	335		
1415	000		
1416	304	LXY	LAE
1417	075		RST 7 / (CYCLE1+4)
1420	025		RST 2 / (DOUT2)
1421	106		CAL CYCLE1
1422	064		
1423	000		
1424	104		JMP DIN+3
1425	335		
1426	000		
1427	025	LXI	RST 2 / (DOUT2)
1430	304		LAE
1431	075		RST 7 / (CYCLE1+4)
1432	104		JMP DIN+3
1433	335		
1434	000		
1435	066	BIT2	LLI 033
1436	033		
1437	372		LMC / STORE ADDRESS GATES CLOSED
1440	060		INL
1441	267		ORM /A HAS 0,1,3 INSTNS
1442	150		JTZ INSTN2
1443	053		
1444	003		
1445	044		NDI 020
1446	020		
1447	053		RTZ / QUIT IF NOT LXI
1450	104		JMP INCG
1451	274		
1452	002		
1453	060	INSTN2	INL
1454	337		LDM / D HAS DECODED 2 INSTN
1455	066		LLI 001 / TEMP STORAGE FROM BIT 1
1456	001		
1457	317		LBM
1460	066		LLI 030
1461	030		
1462	076		LMI 200
1463	200		
1464	104		JMP FLAGS-6
1465	033		
1466	001		
1467	036	BIT1	LDI 001
1470	001		
1471	104		JMP DECODI+1 /USE D AS MARKER IN DECODE S.R.
1472	252		
1473	001		
1474	261	DOUT3	ORB / ADD IN K SWITCH
1475	370		LMA / & DISPLAY
1476	301		LAB
1477	315		LBH / B=377 IF NO SWITCH OPERATED

1500	053		RTZ	/ & QUIT
1501	205		ADH	/ SET CARRY IF K SWITCH OPERATED
1502	303		LAD	
1503	045		RST	4 / (ONLY 1)
1504	260		ORA	/ SET FLAGS
1505	150		JTZ	RDK
1506	210			
1507	002			
1510	012	CYCLE2	RRC	/ THIS SUBROUTINE USES A SINGLE 1
1511	140		JTC	INSTOR / IN THE ACCUMULATOR TO LIFT
1512	216			
1513	003			
1514	066		LLI	074 / DATA FROM A REGISTER INTO
1515	074			
1516	002		RLC	/ REGISTER 'B'
1517	060		INL	
1520	060		INL	
1521	100		JFC	* - 3
1522	116			
1523	003			
1524	317		LBM	
1525	007		RET	
1526	020	ONLY1'	INC	/ S.R. TO CHECK THAT CHAR'R IN ACC & CARRY
1527	260		ORA	/ ONLY CONTAINS A SINGLE 1
1530	110		JFZ	ONLY1+1 / LOOP IF ACC NOT ZERO
1531	041			
1532	000			
1533	021		DCC	
1534	303		LAD	/ RESTORE ACC TO ORIGINAL
1535	053		RTZ	/ & RETURN IF ONLY 1 1 FOUND
1536	307	ALARM	LAM	/ OTHERWISE FLASH DISPLAY
1537	106		CAL	DELAY
1540	150			
1541	003			
1542	372		LMC	
1543	106		CAL	DELAY
1544	150			
1545	003			
1546	370		LMA	
1547	005		RST	START / AND TRY AGAIN
1550	016	DELAY	LBI	003 / DOUBLE LOOP FOR FLASH PERIOD
1551	003			
1552	020		INC	
1553	110		JFZ	* - 1
1554	152			
1555	003			
1556	011		DCB	
1557	110		JFZ	* - 5
1560	152			
1561	003			
1562	007		RET	
1563	303	OP	LAD	/ D HAD CLR,CPL,INC,DEC SWITCH SETTINGS
1564	022		RAL	
1565	100		JFC	CPL / GOTO CPL IF INSTN NOT CLR
1566	172			
1567	003			
1570	372		LMC	
1571	007		RET	
1572	022	CPL	RAL	
1573	100		JFC	INC / GOTO INC IF INSTN NOT CPL
1574	202			
1575	003			
1576	307		LAM	
1577	255		XRH	

1600	370		LMA
1601	007		RET
1602	022	INC	RAL
1603	100		JFC DEC /GOTO DEC IF INSTN NOT INC
1604	212		
1605	003		
1606	347		LEM
1607	040		INE
1610	374		LME
1611	007		RET
1612	347	DEC	LEM / INSTN MUST BE DEC
1613	041		DCE
1614	374		LME
1615	007		RET
1616	066	INSTOR	LLI 022 / LOAD B FROM STORE ADDRESSED
1617	022		
1620	367		LLM
1621	056		LHI 013 / BY STORE ADDRESS REGISTER
1622	013		
1623	317		LBM
1624	056		LHI 377 / & RESTORE H
1625	377		
1626	007		RET
1627	066	LDSTOR	LLI 022 / LOAD INTO STORE (ADDRESS FROM
1630	022		
1631	367		LLM
1632	056		LHI 013 / STORE ADDRESS REGISTER) FROM B
1633	013		
1634	371		LMB
1635	056		LHI 377 / & RESTORE H
1636	377		
1637	007		RET
		END	

APPENDIX 2: Some sample programs

1. The Bootstrap

A program to take data from the toggle switches, load it into store from a starting address initially set by hand in register 5, to display the loaded data in the accumulator, increment the store address and wait for the next data.

<u>Address</u>	<u>Instruction</u>	<u>Comment</u>
000	000	Halt
	307	Input to store
	107	Load accumulator from store
	225	Increment register 5
	346,000	Load PC with 000

2. Stored program examination routine

A program to display the contents of sequential store locations in the accumulator, starting from an address which is initially set by hand in register 5.

<u>Address</u>	<u>Instruction</u>	<u>Comment</u>
010	107	Load accumulator from store
	000	Halt
	225	Increment register 5
	346,010	Load PC with 010

3. A pattern-recognition routine

A program to take a binary pattern, say 252_8 , from the toggle switches into a convenient register and then to compare subsequent switch settings with the stored pattern. If the patterns are different,

load zeros into registers 1, 2 3 and 4, otherwise flash the lamps of those registers for as long as the patterns remain the same.

<u>Address</u>	<u>Instruction</u>	<u>Comment</u>
100	345,252	Load register 5 with 252
	201	Clear register 1
	121	and 2
	131	and 3
	141	and 4
	300	Input 0 to accumulator
	255	Subtract stored pattern
110	351,120	Jump to 120 if zero
	346,102	Otherwise jump to 102
120	211	Complement register 1
	346,103	Jump to 103

4. A parity conversion routine

A program to take data from the toggle switches and to display it with even parity in the accumulator, then to wait for fresh data.

<u>Address</u>	<u>Instruction</u>	<u>Comment</u>
200	203	Clear register 3
	341,001	Load register 1 with 001
	141	& copy in register 4
	302	Input to register 2
	102	& copy in accumulator
	261	Logical AND with 001 in register 1

<u>Address</u>	<u>Instruction</u>	<u>Comment</u>
207/210	351,212	Jump to 212 if result zero
	223	Increment register 3
	101	Register 1 to accumulator
	371	Shift Left
	354,221	Jump to 221 if Carry set
	110	Accumulator to register 1
217/220	346,205	Jump to 205
	103	Register 3 to accumulator
	264	AND with register 4
	351,226	Jump to 226 if zero
	101	Register 1 to accumulator
	242	Add register 2 to accumulator
	000	Halt
230	346,200	Jump to start

5. Multiplication of 4-bit numbers

A program to take two 8-bit numbers from the toggle switches, to mask out the four most significant bits of each and then to multiply the remaining 4-bit numbers, displaying the product in the accumulator.

<u>Address</u>	<u>Instruction</u>	<u>Comment</u>
300	344,004	Load register 4 with 004
	201	Clear register 1
	342,017	Register 2 = 017
	300	Input to accumulator
	262	AND with 017
	130	Keep in register 3

<u>Address</u>	<u>Instruction</u>	<u>Comment</u>
310	000	Halt
	300	Input to accumulator
	262	AND with 017
	371	
	371	
	371	Shift accumulator left, 5 times
	371	
	371	
320	120	& keep in register 2
	364,326	Jump if Carry = 0
	101	Register 1 to accumulator
	243	Add register 3
	110	Put back in 1
	200	Clear accumulator
	234	Decrement register 4
330	244	Add register 4
	351,341	Jump if zero
	101	Register 1 to accumulator
	371	Shift left
	110	Put back
	102	Register 2 to accumulator
337/340	346,317	Jump to 317
	101	Register 1 to accumulator
	000	Halt
	346,300	Repeat

REFERENCES AND BIBLIOGRAPHY

1. R. H. Brunskill, 'The Design and Construction of the Control and Arithmetic Units of a Demonstration Educational Computer.' M.Sc. Thesis, Durham University, 1969.
2. Feedback Ltd, 'ABACUS Educational Computer - Operating Manual.'
3. Ferranti Ltd, 'ARGUS 600' User Manual,' 1970.
4. Digital Equipment Corporation, 'PDP8E Small Computer Handbook,' 1971.
5. Digital Equipment Corporation, 'PDP11 Processor Handbook,' 1972.
6. General Automation Inc., 'The Value of Power,' 1972.
7. Intel Corporation, 'Users Manual for the 8008 Central Processor Unit,' 1973.
8. Open University, 'The OPOS Computer Reference Manual TM221 ORM,' Open University Press, 1975.
9. Sommer, R., 'Timesharing System of Computer-like Peripherals for Teaching Minicomputer Use,' I.E.E. Conference Publication 121, October 1974.
10. Sobel, 'Introduction to Digital Computer Design,' Addison Wesley, 1970.
11. Foster, C. C., 'Computer Architecture,' Van Nostrand, 1970.
12. Lewin, D., 'Theory and Design of Digital Computers,' Nelson, 1972.

