I. C. T. 1900 SERIES

SYSTEM DESCRIPTION

(1904, 1905, 1909)

Provisional Information

MARCH 1965

IM 25
ISSUE 2

Engineering Publications Dept.,
Computer Equipment Group,
I.C.T. Limited,
Thomas Street,
West Gorton,
Manchester, 12

FOREWORD

This handbook sets out to explain the 1904, 5, and 9 computers of the 1900 system to both engineers new to computing and engineers with experience of existing computers.

Chapter 1 will be of value to the newcomer and Chapters 2 and 3 to those engineers who already have experience of non-time sharing machines.

The description of the 1904, 5, and 9 computers proper starts in Chapter 4 where a brief account is given of their main features. This account is enlarged upon in the following chapters and while nothing is dealt with in detail the aim has been to give to the reader a good working knowledge of the system as a whole.

The design of some sections of the computers is, at the time of writing, in a state of flux and though no radical changes are likely, some changes and additions may be expected, especially in the area of peripheral control. For this reason the information contained herein must be regarded as provisional and further information will be issued from time to time.

CONTENTS

Frontispiece

CHAPTER Page

1 INTRODUCTION TO A SIMPLE COMPUTER 1/1

1.1 Some simple arithmetic 1/1

1.2 Digital and analogue computers 1/2

1.3 The choice of a numerical system 1/4

1.3.1 General 1/4

1.3.2 The binary system 1/4

1.3.3 The radix of a system 1/5

1.3.4 Binary decimal relationships 1/5

1.4 Logic elements 1/6

1.4.1 General 1/6

1.4.2 The NOR element 1/7

1.4.3 The single entry bistable 1/8

1.4.4 A method of storing and transferring numbers serially 1/9

1.4.5 A method of storing and transferring in parallel 1/10

1.5 The make-up of a simple computer 1/11

1.5.1 General 1/11

1.5.2 The store 1/11

1.5.3 A computer instruction 1/12

1.5.4 The control unit 1/13

1.5.5 The arithmetic unit 1/13

1.5.6 Input and output 1/13

1.5.7 The relationship of control, arithmetic unit, and store 1/14

1.6 Jump instructions 1/15

CHAPTER			Page
	1.7	Address modification or indexing	1/16
	1.8	Multiple address instructions	1/17
	1.9	Forms of storage	1/18
		1.9.1 General	1/18
		1.9.2 Delay lines	1/18
		1.9.3 Magnetic drums	1/19
		1.9.4 Magnetic tape	1/19
		1.9.5 The disc file	1/20
		1.9.6 Bistables	1/21
		1.9.7 The magnetic core store	1/21
		1.9.8 Speeds and capacity of different forms of storage	1/23
	1.10	The wordform of a number	1/24
		1.10.1 General	1/24
		1.10.2 As a signed integer	1/25
		1.10.3 As a signed fraction	1/26
		1.10.4 As a floating point number	1/27
	1.11	Input to a computer	1/29
		1.11.1 Bootstrap	1/29
		1.11.2 Punched paper tape as an input medium	1/29
	1.12	Output from a computer	1/32
		1.12.1 General	1/32
		1.12.2 Line printers	1/33
		1.12.3 Xeronic printers	1/33
	1.13	Internal checking of information accuracy	1/34
2		THE DISADVANTAGES OF A SIMPLE COMPUTER	2/1
	2.1	Peripheral speed limitations	2/1
	2.2	Improvement due to autonomous transfers	2/2

CHAPTER			Page
3	TIME SHARING		3/1
	3.1	Principles of time sharing	3/1
	3.2	Additional features required with a time sharing computer	3/2
4	GENERAL INTRODUCTION TO THE I.C.T. 1904, 5, and 9		4/1
	4.1	The 1900 series	4/1
	4.2	The 1900, 1904, 1905 and, 1909 - general	4/2
	4.3	Organisation of programs and peripherals	4/4
	4.4	Peripheral transfers	4/5
	4.5	Operator control	4/6
	4.6	Engineers' control	4/7
5	PERIPHERALS AVAILABLE FOR THE 1900		5/1
	5.1	Input devices	5/1
		5.1.1 The I.C.T. 1915 and 1916 paper tape readers	5/1
		5.1.2 The I.C.T. 1911 card reader	5/1
		5.1.3 The I.C.T. 1912 card reader	5/2
	5.2	Output devices	5/2
		5.2.1 The I.C.T. 1925 paper tape punch	5/2
		5.2.2 The I.C.T. 1921 card punch	5/3
		5.2.3 The I.C.T. 1920 card punch	5/3
		5.2.4 The I.C.T. 1922 card punch	5/4
		5.2.5 The I.C.T. 1933 line printer	5/4
		5.2.6 The I.C.T. 1932 line printer	5/5
		5.2.7 The I.C.T. 1931 line printer	5/5

CHAPTER

		Page
5.3	File storage devices	5/6
5.3.1	The I.C.T. 1953 exchangeable disc store	5/6
5.3.2	The I.C.T. 1956 Data disc store	5/7
5.3.3	The I.C.T. 1938 magnetic card file	5/8
5.3.4	The I.C.T. 1962, 1963, and 1964 drum storage systems	5/10
5.3.5	The I.C.T. 1974 magnetic tape system	5/10
5.3.6	The I.C.T. 1971, 1972, and 1973 magnetic tape systems	5/12
5.4	Other equipment	5/13
5.4.1	The I.C.T. 1996 program timer and real time digital clock	5/13
5.4.2	The I.C.T. 1830 general purpose visual display unit	5/14
5.4.3	The I.C.T. 1831 display and interrogation unit	5/16
5.4.4	The I.C.T. 1934 digital increment plotter	5/16
5.4.5	The I.C.T. 8500 document sorter/ reader	5/18
5.4.6	The I.C.T. 1992 communications multiplexer	5/18
5.4.7	The I.C.T. 1995 data exchange control	5/19
6	THE CENTRAL COMPUTER	6/1
6.1	General	6/1
6.2	The arithmetic unit	6/1
6.3	The LDG register	6/2
6.4	The microprogram sequence gates	6/3

Page

CHAPTER

6.5 The central control unit 6/4

6.6 The computer clock 6/4

6.7 Hesitation control 6/4

6.8 Peripheral control units 6/5

6.9 The core store 6/6

6.10 The datum and limit tester 6/7

6.11 Parity 6/7

6.12 The engineer's panel 6/7

6.13 Power supplies 6/8

7 THE I.C.T. 1904, 5 AND 9 WORDFORM 7/1

7.1 General 7/1

7.2 Four 6-bit alpha-numeric characters 7/1

7.3 A signed fraction 7/1

7.4 A signed integer 7/2

7.5 Multiple length working 7/2

7.5.1 General 7/2

7.5.2 A double length signed integer 7/2

7.6 A floating point number 7/3

7.7 An instruction 7/4

7.8 A counter modifier 7/5

8 SUMMARY OF THE FUNCTION CODES 8/1

8.1 General 8/1

CHAPTER			Page
	8.2	The function code	8/1
		8.2.1 General	8/1
		8.2.2 Ordinary instructions	8/2
		8.2.3 Macro-instructions	8/2
	8.3	Executive orders	8/9
	8.4	Function code nomenclature	8/10
	8.5	Floating point hardware and software times	8/11
9		THE MICROPROGRAM	9/1
	9.1	General	9/1
	9.2	Conventions on drawing LDP/1900/0092	9/1
	9.3	Sequencing of orders and hesitations	9/3
		9.3.1 Short orders	9/4
		9.3.2 Long orders	9/4
		9.3.3 Very long orders	9/5
	9.4	Hesitations	9/5
	9.5	Macro-instructions	9/6
	9.6	Timing	9/7
	9.7	A microprogram sequence - the 001 instruction	9/7
10		CONTROL OF PERIPHERAL TRANSFERS	10/1
	10.1	General	10/1
	10.2	Control and data lines	10/2
	10.3	Setting up a peripheral transfer	10/3

CHAPTER Page

10.4 Hesitation requests 10/5
10.4.1 Hesitation select 10/5
10.4.2 Identifying the selected peripheral 10/7
10.4.3 Transfer of a word or character 10/8
10.5 Ending a peripheral transfer 10/9
10.6 Reading SRs 64 and 65 and peripheral special registers 10/10
10.6.1 General 10/10
10.6.2 SRs 64 and 65 10/10
10.6.3 Peripheral special registers 10/11
10.7 Loading peripheral special registers 10/11
10.8 The special hesitation facility 10/12

11 THE I.C.T. STANDARD INTERFACE 11/1
11.1 General 11/1
11.2 The scope of the interface 11/2
11.3 Lines crossing the standard interface 11/2
11.4 Transfer of control information over the interface 11/4
11.4.1 General 11/4
11.4.2 Control of single channel, single mechanism peripherals 11/4
11.4.3 Control of single or multichannel, multi-mechanism peripherals 11/5
11.4.4 Control of multiaddress, single or multi-mechanism peripherals 11/6
11.4.5 T Suppression 11/7

CHAPTER		Page
11.5	Transfer of data over the standard interface	11/7
11.5.1	General	11/7
11.5.2	Single mechanism data transfers	11/7
11.5.3	Multichannel, multimechanism data transfers	11/8
11.5.4	Single channel, multimechanism data transfers	11/9
11.5.5	Multicharacter transfers	11/10
11.5.6	End of transfer	11/11
11.6	Special hesitations	11/11
11.6.1	General	11/11
11.6.2	Link board connections	11/11
11.6.3	Operation	11/12
11.7	Peripheral status	11/13
11.7.1	General	11/13
11.7.2	Q and P status	11/13
11.7.3	Direct status response	11/15
12	ARITHMETIC PROCESSES	12/1
12.1	General	12/1
12.2	Addition and subtraction	12/1
12.2.1	General	12/1
12.2.2	Addition	12/2
12.2.3	Subtraction	12/2
12.2.4	Overflow	12/3
12.3	Multiplication	12/4
12.3.1	General	12/4
12.3.2	Multiplication overflow	12/5
12.3.3	Operand movements in the arithmetic unit	12/5

CHAPTER			Page
12.4	Division		12/8
	12.4.1	General	12/8
	12.4.2	Numerical examples of the division method	12/8
	12.4.3	The rules in non-restoring division	12/12
	12.4.4	Operand movement in the arithemtic unit	12/13
	12.4.5	Division overflow	12/16
12.5	Character conversion		12/17
	12.5.1	Decimal to binary conversion using the 043 instruction	12/17
	12.5.2	Operand movement in the arithmetic unit	12/17
	12.5.3	Binary to decimal conversion using the 047 instruction	12/20
	12.5.4	Operand movement in the arithmetic unit	12/21
13	THE FLOATING POINT UNIT		13/1
13.1	General		13/1
13.2	The floating point unit		13/1
13.3	The argument and exponent arithmetic units		13/2
13.4	The floating point unit microprogram		13/3
13.5	Addition and subtraction		13/4
	13.5.1	General	13/4
	13.5.2	Addition	13/5
	13.5.3	Subtraction	13/6
13.6	Multiplication		13/6
13.7	Division		13/9
13.8	Normalising		13/10

CHAPTER Page

13.8.1 General 13/10
13.8.2 Normalising after addition and subtraction 13/11
13.8.3 Normalising after multiplication 13/11
13.8.4 Normalising after division 13/13

13.9 Exponent test 13/13

13.10 End action 13/14

LIST OF DIAGRAMS

DIAGRAM

1 Delay line store

2 Input timing diagram

3 Bistable store

4 Block diagram of a simple computer

5 Magnetising a core

6 B/H characteristic of core material

7 Addressing a core

8 A section of a core store plane

9 Table of speeds and capacity of different types of store

10 Representations of numbers 0 to 9 on 8-hole I.S.O./A.S.A. tape

11 Timing diagram of a computer in time sharing

12 Block diagram of central computer

13 Block diagram of microprogram

14 The 001 microprogram sequence

15 The hesitation selection system (simplified)

16 Hesitation system block diagram

17 Transfer of SR64 to accumulator X

18 Transfer of peripheral special register n to accumulator X

19 Transfer of accumulator X to peripheral special register n

20 Magnetic Tape control system (1974)

DIAGRAM

21 Disc file control system (1956)

22 Standard interface lines (schematic)

23 Transfer of control information over the standard interface

24 Transfer of data over the standard interface

25 The arithmetic unit

26 Floating point unit block diagram

27 The floating point unit argument and exponent arithmetic units

28 Floating point unit microprogram sequence

CHAPTER 1

INTRODUCTION TO A SIMPLE COMPUTER

1.1 Some simple arithmetic

If there was a clerk who could only add, subtract, and recognise zero and positive or negative numbers, he could be used to work out the most complex arithmetical problems by being issued with pencil and paper, and a series of instructions. To multiply x by n he need not know how to multiply: x can be added to itself n-1 times.

A series of instructions to enable the clerkto do this could be set out as below.

	cumulative answer	counter	final answer
Write x in cumulative answer.	x		
Put n-1 in counter.		n-1	
If counter is zero, place cumulative answer in final answer and stop.			
If counter is not zero, add x to cumulative answer.	x+x		
Subtract 1 from counter		n-2	
If counter is zero, place cumulative answer in final answer and stop.			
If counter is not zero, add x to cumulative answer.	x+x+x		
And so on.		n-3	

It will be seen that n is reduced by 1 and x added to the cumulative answer at each step until the condition n = zero is recognised by the clerk. At this stage he puts the cumulative answer into the final answer column and stops the operation.

If x were to be divided by n, the clerk could be given another program (list of instructions) to follow. In this case he would be asked to subtract repeatedly n from x, counting each step until the cumulative answer went negative. In this case the counter column contains the answer, the remainder being the last but one entry in the cumulative answer column.

With the aid of these two programs, the arithmetic prowess of the clerk has now been extended to multiplication and division. If he is now given a program for the solution of a more complex arithmetic program, rather than write out the whole of each multiplication or division process in the program, the orders multiply or divide can be given at the approriate points. He can refer to the mechanism of these himself. This saves time in writing out the program for him to follow. If he could be taught to multiply and divide in the more normal manner, this would save him a lot of time in executing the program.

In summary, any mathematical problem which can be solved by arithmetic means, will break down into a program of simple arithmetic instructions. The execution of these demands an ability to follow the instructions sequentially and do very simple arithmetic and discriminative operations.

1.2 Digital and analogue computers

Like the clerk postulated above, an electronic computer can perform only simple operations in the order laid down by a program of instructions. A computer's whole virtue lies in the speed and accuracy with which it can carry out such operations. These qualities are its only "superhuman" aspect.

There are two types of computer: analogue and digital. Two examples of simple analogue devices are the speedometer in a car and a slide rule.

A speedometer translates the revolutions per minute of a car's wheels into electrical energy and then into the mechanical displacement of a pointer on a scale marked off in m.p.h.

The slide rule has its length marked off in numbers spaced logarithmically. To multiply two numbers, distances analogous to the logarithms of the numbers are added together.

Two examples of digital devices are the odometer in a car and the abacus.

An odometer counts the revolutions of a car's wheels (via suitable gearing) on a digital counter. This is marked in miles.

With an abacus, beads are moved from one end of a wire to the other for the purpose of adding. Each bead represents unity on the wire and each wire represents some multiple of the wire above it.

It will be noted that in the case of the analogue devices, their accuracy is determined entirely by how well they are made and how clearly their scale can be read. In the case of the digital devices, there is no doubt about their answers. Were there sufficient "places" on an odometer, the progress of the car could be measured in feet or even inches.

Some points should be mentioned at this stage.

1. The accuracy of an analogue device depends on the precision of its manufacture.
2. The accuracy of a digital device depends on the number of digits (or places) which can be held inside it.
3. An analogue device measures quantities.
4. A digital device counts quantities.

An analogue computer is well suited to the solution of differential equations. The constants in such an equation can be simulated electrically in terms of inductance, capacitance and resistance and the accuracy of the answer is good enough for many engineering applications.

A digital computer is suited to any problem that can be solved numerically but will generally take longer than an analogue machine to solve problems in differential calculus.

The I.C.T. 1900 is a digital computer and from here onwards, no further mention of analogue computers will be made.

1.3 The choice of a numerical system

1.3.1 General

If a computer is to perform simple arithmetic, it must be able to remember numbers. To hold decimal digits, we need an electrical device with ten separate states in order to represent any of the numbers 0 to 9. It is simpler and cheaper to use two-state components. A switch or a relay is either open or closed and has no half way state, and a valve or transistor can be used with greatest speed and simplicity by switching it between its fully conductive and cut off states. Two-state devices such as these are useless for the direct expression of decimal quantities and so, inside the computer, numbers are represented in binary form.

1.3.2 The binary system

The decimal number 10,496 represents:

$$1 \times 10^4 + 0 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 6 \times 10^0$$

or 10,000 + 400 + 90 + 6

In the binary system the number 2 fulfils the same function as does

the number 10 in the decimal system. With the decimal system the digits are represented by the numbers 0 to 9. As the binary system works in powers of 2, only two digits are required: 0 and 1. These two values can be represented by an electrical or an electronic device being switched to its 'on' or 'off' state.

The binary number 10110 represents:

$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

or in decimal 16 + 0 + 4 + 2 + 0 which is equal to 22.

1.3.3 The radix of a system

As is well known, the decimal system is based on the number 10, and as is now seen, the binary system is based on the number 2. The number on which a system is based is called the radix (pl radices). There are other systems than decimal and binary in common usage, e.g. the weight Xst Ylb Zoz involves the radices 16 (ozs to lbs) and 14(lb to st).

The sum of money £X Ys Zd involves the radices 12 (pennies to shillings)and 20 (shillings to pounds). The above are both examples of mixed radix numbers.

In a binary computer, all mixed radix numbers are often broken down into terms of the lowest denomination e.g. Sterling may be broken down into halfpennies before any calculations are done on it, and then converted back into £ s d. before obtaining an output from the computer.

1.3.4 Binary decimal relationship

The decimal value of each digit of a binary number is worked out and these values are added to give the decimal equivalent.

e.g $101101001_2 = (1\times2^0 + 1\times2^3 + 1\times2^5 + 1\times2^6 + 1\times2^8)_{10}$

$= (1 + 8 + 32 + 64 + 256)_{10}$

$= 361_{10}$

The suffixes 10 & 2 denote the radices of the numbers.

To obtain the binary counterpart of a decimal number, the process shown below is carried out.

2)361			
2)180	remainder	1	
2) 90		0	
2) 45		0	
2) 22		1	
2) 11		0	<-- 361 binary
2) 5		1	
2) 2		1	
2) 1		0	
0		1	

The answer is the remainder column reading from bottom to top

$$361_{10} = 101101001_2$$

1.4 Logic elements

1.4.1 General

A binary number may be represented inside a computer in three forms: serial, parallel, and a combination of the two. When represented in serial form, the bits (binary digits) are represented by a chain of pulses passing through a circuit one after the other, and when represented in parallel form each bit in a number has a separate channel allocated to it.

In a serial computer there is always an electronic clock and it is by reference to this clock that the significance of the bits of a serial number in any part of the computer may be determined. The significance of the digits of a number represented in parallel is determined in quite a different manner, and depends upon which one of the n separate channels is allocated to it.

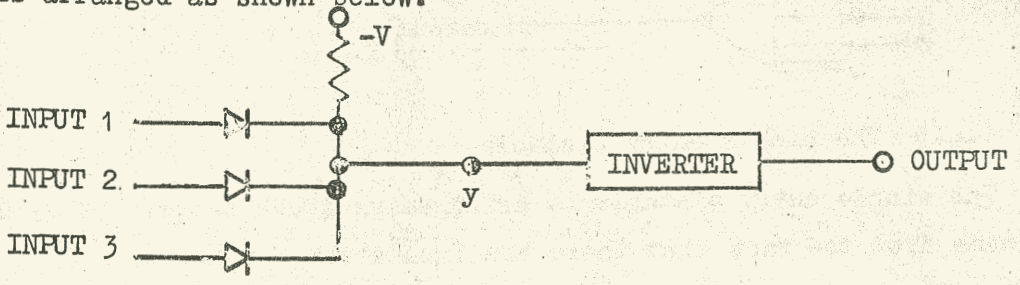
Several basic circuit elements have been designed for computers, but in order to orientate the reader toward the I.C.T. 1900 range, two of the

elements used in the range will be used in the following examples. These are the NOR and the single entry bistable.

Note: It was mentioned in section 1.3.1 that binary data may be represented by the fully conductive and cut-off states of a transistor. This has resulted in data voltages being referred to as positive and negative. Such terminology is very useful but is purely relative, and in the 1900 series for example, corresponds to voltages of 0V and -6V respectively.

1.4.2 The NOR element

The NOR element is a multiple input gate followed by an inverter and is arranged as shown below.



Consider first the effect of INPUT 1, 2 and 3 upon the voltage level at point y.

If any inputs are positive their diodes will conduct and point y will also be positive: any negative inputs will not effect the output because their diodes will be cut off. The only time that point y can be negative is when all of the inputs are negative.

That is:

INPUT 1 and INPUT 2 and INPUT 3 negative .... y is negative

INPUT 1 or INPUT 2 or INPUT 3 positive .... y is positive.

Because of its action, such a diode configuration is referred to as either a negative AND-gate or a positive OR-gate depending upon its logical application.

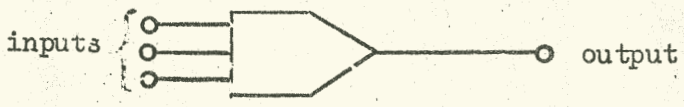
To complete the NOR element the gate is followed by a single transistor inverter whose output is positive for a negative input and vice-versa. So, with inputs which are active when positive neither INPUT 1 nor INPUT 2 nor INPUT 3 (i.e. all inputs negative) make OUTPUT positive.

The overall effect of the NOR element is:

INPUT 1 and INPUT 2 and INPUT 3 negative .... OUTPUT is positive

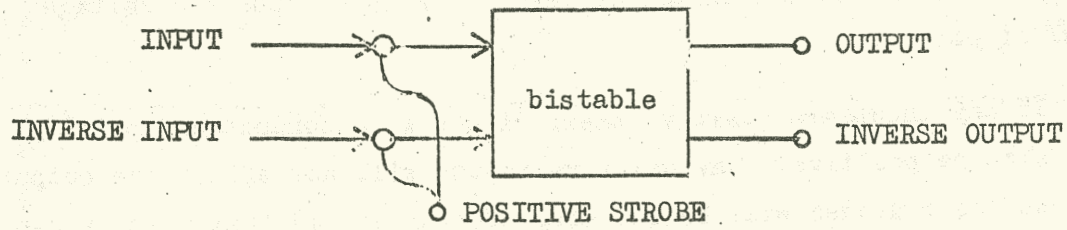
INPUT 1 or INPUT 2 or INPUT 3 positive .... OUTPUT is negative.

In order to simplify logic drawings the NOR element is represented by the symbol shown below in the 1900 series.



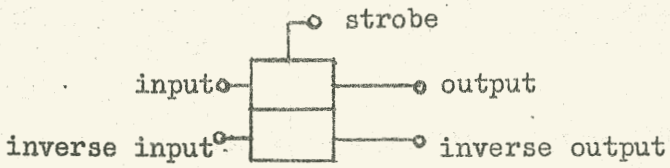
1.4.3 The single entry bistable

The single entry bistable is arranged as shown below. It gains its name from the fact that there are facilities for one pair of inputs only.



INPUT may be either positive or negative and INVERSE INPUT is, as its name implies, in antiphase. The two small circles are positive AND-gates and their outputs are negative except for the time that POSITIVE STROBE goes positive. At that time either INPUT or INVERSE INPUT, depending upon which is positive, is gated through to the bistable. The bistable is triggered by positive inputs only and its outputs take up the states of the inputs at the time of the strobe. Since the bistable only experiences a triggering input at the time of the strobe, its state remains unchanged between strobes regardless of changes in input.

To simplify logic drawings the bistable and its input gates are represented by the symbol shown below.



Bearing in mind the characteristics of the above two elements, the following two examples of serial and parallel logic will easily be understood.

1.4.4 A method of storing and transferring numbers serially

The binary number in this example is taken as being six bits in length, each bit being represented by the presence (1) or absence (0) of a pulse one microsecond in length. The whole serial number is represented by a train of pulses six microseconds in duration recirculating around the circuit shown in Diagram 1.

a) Input

A timing diagram for the input of information is shown in Diagram 2. The six bits of data present at INPUT during the time of INPUT GATING WAVEFORM are gated through NOR 1 as shown in the timing diagram. The delay line may have some information already circulating from a previous transfer so, as the new information enters from NOR 1, any old information is erased by applying NOR 4 (inverse INPUT GATING WAVEFORM) to NOR 3. NOR 1 output emerges from the delay line just as NOR 4 returns to its negative state and so will circulate unimpeded, the circuit acting as a six bit store.

b) Output

OUTPUT GATING WAVEFORM is normally positive. If, at an appropriate time it is taken negative for 6 microseconds, a copy of the delay line's content will emerge at OUTPUT.

1.4.5 A method of storing and transferring numbers in parallel

The binary number in this example is again taken as being 6 bits in length, but this time it is stored in the six bistables shown in Diagram 3.

The six inputs have no effect on the bistables until STROBE is applied. At the time of STROBE, any positive inputs are gated onto the bistables' upper inputs making the corresponding outputs positive. Any negative inputs present are inverted in the NOR elements and are gated by STROBE onto the appropriate bistables' lower inputs, again making the corresponding outputs positive.

The content of INPUT is thus stored in the bistables and is available at OUTPUT. Any change in INPUT will have no effect on OUTPUT until a further STROBE is applied.

These typical circuits serve to demonstrate two points:

1) the serial circuit is slower than the parallel circuit because it takes 6 bit times to deliver or accept a 6-bit number whereas the parallel circuit accepts or transmits all its bits simultaneously in a time governed by the speed of the circuit elements (typically less than 1μs);

2) the parallel circuit uses more circuit elements than the serial circuit because it provides one data path per bit.

Serial computers are normally cheaper than parallel computers, but are at the moment slower. If the bit time of a serial machine could be speeded up to fractional microsecond speeds, a serial machine could be made comparable to present parallel machines.

It is futile at the present time to speed up appreciably the circuits of parallel machines because storage at present available

(discussed in 1.9) cannot take in or give out information at sufficient speed to take advantage of extra speed in a parallel machine. The I.C.T. 1900 is a fully parallel machine in which a 24-bit number can be transferred from one bistable register to another in less than 1 microsecond.

1.5 The make-up of a simple computer

1.5.1 General

To solve a complex problem or a large number of similar simple problems, the clerk originally postulated would need written instructions and a sheet of paper on which to write any intermediate and final answers. In order that an electronic computer may perform the same functions it requires:

a) a store for holding instructions and answers;
b) an arithmetic unit for carrying out any necessary multiplication, addition or division;
c) a control unit to interpret the instructions and move data between the store, the arithmetic unit, and input/output devices;
d) a means of getting information into and out from the computer, e.g, a paper tape reader and punch.

Diagram 4 shows a simple computer in block form incorporating the above units, and the following sections deal briefly with the content of each block and with the way in which they work together.

1.5.2 The store

The store is equally divided into addresses and each address contains a number of bits. The addresses are numbered from nought up to, in the case of modern computers, many thousands. Any one of the store addresses can be selected electronically.

The content of each address is called a word (rather than a number) and is typically between 20 and 50 bits in length. The 1900 range for

instance has a word length of 24 bits and a store capacity of between 4096 (2^{12}) and 32,768 (2^{15}) words. The bigger the store capacity, the more expensive is the computer. A store of 4096 words capacity would be addressed from 0 to 4095.

The program of instructions is fed into the store via the input device and stored in consecutive addresses.

Each instruction basically contains;

1. a function part which tells the computer what arithmetic, transfer or input/output operation to perform,
2. an address part which tells the computer at which address the data relevant to the instruction is stored.

The data is then fed into store locations referred to by the address parts of the instruction.

When program and data are stored the computation can start.

1.5.3 A computer instruction

Before seeing how a computer runs through a program, it is as well to understand more of the make up of a simple instruction. As already mentioned, an instruction specifies what operation is to be performed (function part) and where the data on which the operation is to be carried out is stored (address part). It is required for ease of programming that besides being able to do simple arithmetic, the computer should also be able to carry out transfers, input/output control, sign testing, radix conversion, etc., For this reason, upwards of one hundred instructions are common on modern computers. A seven bit binary number gives 128 different instruction codes. If in a hypothetical computer of 24 bit word length the code for transfer store to arithmetic unit is 1100011 and the content of store line (address) 33 is required in the arithmetic unit, the instruction to implement this is:-

110001100000000000100001

function address (33)

i.e. transfer content of store line 33 to arithmetic unit. It can be seen that the above instruction looks exactly like an operand. The job of the control unit is to separate instructions from operands, break down instructions into their separate parts and act upon them.

1.5.4 The control unit

The main components of the control unit are;

a) a control register which contains the control address - this usually increases by unity after each instruction so that the next instruction may be read from store when required;

b) circuits which decode the function part of an instruction and set up the computer to obey the instruction;

c) circuits which decode the address part of the instructions in order that the required data can be read from the appropriate store address.

1.5.5 The arithmetic unit

The main components of the arithmetic unit are:

a) an adder (or adders in the case of a parallel computer);
b) one or more single word stores into which operands may be transferred whilst being operated upon by the arithmetic unit;
c) facilities for shifting operands left or right in the stores mentioned in b) for such purposes as multiplication and division.

1.5.6 Input and output

Data is fed into a computer via such media as punched paper tape, punched cards, and magnetic tape. Output is obtained on the same media and, in addition, may be obtained directly in printed form on devices which print out a whole line of characters at a time.

No more need be said of input and output at this stage, but the

subject will be discussed in more detail in sections 1.11. and 1.12.

1.5.7 The relationship of control, arithmetic unit, and store

An easy way to gather insight into the way in which the control, arithmetic unit, and store work together is to run through the short program, "add x to y" and place the result in the store.

Assume that;

a) operand x is in store address 3,
b) operand y is in store address 4,
c) the answer is to be placed in store address 5.

Three instructions are required and these are;

a) transfer the content of store address 3 (x) to the arithmetic unit,
b) add the content of store address 4 (y) to the content of the arithmetic unit,
c) transfer the content of the arithmetic unit (x + y) to store address 5.

Assume that these instructions are located in store lines 0, 1, and 2.

All the paths taken by operands, addresses and instructions in the following examples are shown in Diagram 4. The control unit does the following.

First instruction

a). Sends the control address (equal to 0) from the control register to store selection.
b) Receives the content of address 0 (first instruction) and separates the function and address parts.
c) Applies the address part of the instruction to store selection thus selecting store line 3.

d) Decodes the function causing the content of store address 3 (x) to be transferred to the arithmetic unit.

e) Unity is automatically added to the control address in order that the next instruction may be had from store.

Second instruction

a) Sends the control address (equal to 1) from the control register to store selection.

b) Receives the content of address 1 (second instruction) and separates the function and address parts.

c) Applies the address part of the instruction to store selection thus selecting store line 4.

d) Decodes the function causing the arithmetic unit to add the incoming operand (y) from store address 4 to its present content (x).

e) Unity is automatically added to the control address in order that the next instruction may be had from store.

Third instruction

a) Sends the control address (equal to 2) from the control register to store selection.

b) Receives the contents of address 2 (third instruction) and separates the function and address parts.

c) Applies the address part of the instruction to store selection to select store address 5.

d) Decodes the function causing the content of the arithmetic unit (x + y) to be transferred to store address 5.

e) Unity is automatically added to the control address in order that the next instruction may be had from store..

1.6 Jump instructions

It has been stated that the instructions of a program are placed in

consecutive store addresses in order that they may be selected in sequence by control. For a variety of reasons, it is sometimes necessary to break the sequence and jump to another control address. This is done by injecting the address to which it is required to jump into the control register. The control register then carries on counting from the new control address as normal.

There are two types of jump:

a) unconditional,
b) conditional.

The first type is initiated by an instruction of the type "jump to address N" and the second by an instruction of the type "jump to address N if certain conditions are met". These conditions may be that an intermediate result has gone from positive to negative, exceeded a certain value, become zero, etc.

The computer would be programmed to "look" for this condition and if it happened, jump to an alternative part of the program. The conditional jump is necessary in every problem a computer encounters.

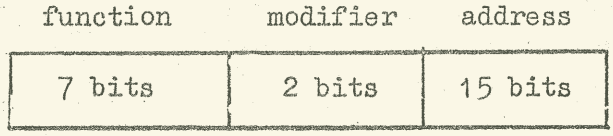
1.7 Address modification or indexing

In some types of application it is necessary to repeat the same program several times using different sets of data. An example of this would be the working out of a company's pay cheques. Each individual's gross salary is different, but the computation to arrive at his nett salary is standard (with due apologies to all salaries departments). It is desirable that the same program can deal with each salary computation, but that each time the program is repeated the address parts of the instruction may be modified or indexed to select the part of the store containing data applicable to the current case. Both the terms modifier and index are in current use, the former in Britain and the latter in the United States and Canada.

Looking back to the three instructions described in the latter part of section 1.5.7, if between steps b) and c) a modifier is added to the address part of the instructions, then the data will be selected from a different store location to that specified by each instruction.

Most computers have several special registers in which modifiers are stored. Part of the instruction coding specifies one of these registers and the modifier is read out and added to the address part of the instruction just before it is applied to the store selection.

Taking a computer of 24-bit word length, 100 instructions, and 3 modifier registers, the instruction might be made up as follows:



The control unit decodes the 2 bits specifying the modifier register and if both bits are zero, no modifying takes place. With values 1, 2, or 3, the content of the appropriate modifier register is released and used in the manner described above. It should be noted that when modifying takes place, the address part of the instruction in the store is in no way altered.

1.8 Multiple address instructions

So far, each instruction has quoted a single store address and an operation such as an addition has required three instructions, i.e. two to bring out the operands used in the addition and one to place the answer back in store. More sophisticated computers can accomplish this with one instruction.

It is common to have additional registers called accumulators and use these to hold operands in the course of a calculation. A program in the I.C.T. 1900 for example, has access to 8 such accumulators and these are the first 8 locations of the program's store area.

The 1900 range instruction has four parts: function, address, modifier and accumulator address. An example of an instruction typical of this type is: add the content of N to the content of accumulator X and place the result in N (or in X).

Two addresses are involved in this type of instruction and a computer capable of this sort of work is called a two address machine.

The I.C.T. Orion computer is an example of a three address machine and is capable of instructions of the type:

add the content of N to the content of X and place the result in Y.

Clearly there are more steps involved in performing instructions in a two or three address machine and as a result of this, central control is much more complex: so much so, that in the cases of both the computers just mentioned, a program called the microprogram is used to guide central control through the operational sequences necessary for each instruction. The microprogram is not a program kept in the computer's store, but is made up of a complex series of logic elements.

The main advantage of multiple address machines is that programmers are saved much of the tedious work involved in shifting operands in and out of the arithmetic unit.

1.9 Forms of storage

1.9.1. General

It has already been shown in section 1.4 how information can be held in a delay line or on a series of bistables. This is a form of storage. There are several types of store in current use and these include delay lines, various forms of magnetic store, and bistables.

1.9.2 Delay lines

Delay lines have been used as the central store in some serial

machines. In the Pegasus computer for instance, the central store consists of 64 one-word delay lines, any one of which can be selected. Delay lines are extensively used as single word registers in serial computers. The content of a delay is lost when the computer is switched off.

1.9.3 Magnetic drums

Magnetic drum storage is quite cheap and is generally used as backing storage to a main store. If a program is too large for a computer's main store, it may be placed in the drum store and transferred down to the main store in sections as required. The drum is coated with a magnetic material and rotates at high speed (2000 - 6000 r.p.m). Stationary heads, similar in principle to those in a tape recorder, are located one or two thousandths of an inch from the drum's surface. Each of these heads, which can be selected electronically, cover a small slice of the drum. This slice is called a track.

Each track is divided into sectors each of which hold several words and special tracks permanently recorded on a drum's surface tell the computer which sector of a drum is under a head at any particular time. By selecting a head, information may thus be transferred to or from addressable sectors on a drum's surface. Information fed to a head results in magnetisation of the drum surface and this can only be removed by further "writing". Information on a drum is not lost if the computer is switched off.

1.9.4 Magnetic tape

Magnetic tape used in computing is $\frac{1}{2}$ or 1 inch wide by 2,000 to 4,000 feet long. Sixteen recording or replay heads can be accomodated across a 1 inch tape, some being used for recording or playing back data and some for purposes of control.

A tape may be divided longitudinally into blocks, and in order that any particular block may be located, each block is preceeded by some sort

of marker. Specific blocks may be found by counting markers from the beginning of a tape, and some systems improve on this method by preceeding each block with an address, the blocks being addressed consecutively from the beginning of a tape.

Magnetic tape, due to its low cost and enormous capacity, is one of the most important storage mediums in use at the present time.

1.9.5 The disc file

The disc is similar in principle to a drum, but by using several rotating discs mounted a few inches apart on a common drive shaft, a much larger surface area is obtained and hence more storage space.

Each disc has flying heads mounted near to both of its faces, and these heads can move radially under the control of a servo-mechanism.

The disc is addressed in much the same way as a drum, but in addition to head and sector addressing, the position of each head (which can be moved over any one of a number of tracks) must also be addressed.

An example of a disc file is the one produced by Data Products. It has up to four disc units each with up to 16 discs. Each disc has 8 heads which can be moved collectively to any of 64 positions. The disc has 256 tracks on each surface (4 heads x 64 positions on both topside and underside), the outer 128 tracks being divided into 16 sectors (the outer zone) while the inner 128 tracks are each divided into 8 sectors (the inner zone). A sector contains 1446 bits of data preceeded by a 17 bit record identification.

Each disc unit has, in addition to its 16 data discs, two control discs which contain permanently recorded sector address information and a clock for writing and control purposes.

In order to locate a sector (or record) on the file, the following 21-bit address must be supplied:

parity	read next sector	head (8) and sector (16 or 8)	head position (64)	disc (16) and unit (4)
1	1	7	6	6

1.9.6 Bistables

These are used extensively as small stores in serial and parallel machines. They are more expensive than delay lines, one bistable being required per bit. They do however have a very fast access time, compared with delay lines.

1.9.7 The magnetic core store

a) General

Magnetic core stores are used in all modern computers with large central stores. Each bit is stored in terms of the two remanent states of magnetisation of a square B/H characteristic magnetic toroid or core.

b) Cores

A toroid or core may be magnetised in either direction by passing current through a wire threaded through the core as illustrated in Diagram 5. The B/H characteristic of the material used in the core is as shown in Diagram 6.

The two states of remenance of a core are arbitrarily labelled the "0" and "1" states. A magnetising current of Im or -Im will switch the core to the "0" or "1" state respectively, but lesser currents such as Im/2 will have no effect because they do not produce sufficient magnetising force to overcome the coercivity of the material. The importance of this characteristic of the material will become apparent in the next section.

c) Addressing and reading cores

If a number of cores are arranged in a matrix as shown in

diagram 7 with X and Y wires threaded through them, any particular core may be selected by addressing the two appropriate X and Y wires. The wires may be selected by decoding binary addresses applied to the store selection electronics.

Current pulses of Im/2 simultaneously applied to the chosen X and Y wires result in a magnetising current of Im linking with the core at the intersection of these two wires. This core will take up the "0" state, and if it were previously in the "1" state, the flux will change direction. If however it was already in the "0" state, there will be only a slight change in flux. A third wire called the read wire is threaded through every core in the matrix and a voltage is induced in this wire when "1" is read from any address (X, Y intersection) in the matrix.

It should be noted that the only core affected by this read cycle is that at the intersection of the selected X and Y wires. Other cores on the two wires experience magnetising currents of Im/2 but, as already mentioned, this is insufficient to change the remanent state of the material.

In order to store an n-bit word, n such matrices are required with their respective X and Y drive wires connected in series. Each such plane has one read wire.

d) Regenerating the read down information

At the conclusion of the read cycle the selected core in each plane (or matrix) of a store has been switched to the "0" state. Had it previously been a "1", then in order to retain the information in the core store address selected, it must be written back immediately. For this reason the read cycle is followed by a regenerate cycle.

e) Regenerating a "1"

During the regenerate cycle the selected X and Y drive wires

have a reverse current of -Im/2 applied to them. The cores at their intersection thus have a magnetising current of -Im which is sufficient to return them to their "1" condition.

f) Regenerating a "0"

In order to prevent the reverse current of -Im/2 affecting cores which were read as "0"s, all the cores in each plane are threaded with a fourth wire called the inhibit wire. If a "0" was read, a current of Im/2 is applied to the inhibit wire coincident with the write cycle X,Y drive currents. The total magnetising current linking with the selected core is then -Im/2 -Im/2 +Im/2 = -Im/2. This is insufficient to switch the core from the "0" condition.

g) Writing to a core store

A write is always preceded by a read, the read erasing the store address's previous content by filling it with "0"s. The technique of writing in information is similar to that of regenerating, the difference being that the inputs applied to the inhibit wires are derived from the information to be written rather than from that just read from store.

A section of a plane of a core store matrix is shown in Diagram 8 complete with all necessary drive, read, and inhibit wires.

1.9.8 Speed and capacity of different forms of storage.

It would be ideal to have the whole of a computer's store as fast as possible. Unfortunately, fast high capacity stores are very expensive. A common compromise is a several thousand word core store backed by a drum store. On calling for information inside a store, the location of the information must be specified. The time which elapses between selecting a store location (address) and the information being available is called the access time. In the case of a drum for example, the average access time is the time taken for the drum to turn through half a revol-

ution. This is normally still several milliseconds. When the information is under a drum's read head it can be read off at speeds comparable to a fast core store.

A simple computer is at a standstill for the period of the access and it is better to transfer whole blocks of consecutively addressed information than randomly located words to or from a drum store. This applies equally to magnetic tape, delay lines, etc.,

The table in Diagram 9 assumes words 24 bits in length, but it must be emphasised that the times are not very accurate, and are only provided for comparision.

1.10 The wordform of a number.

1.10.1 General.

Up to now it has been assumed that numbers are represented in a computer by their true values. This is true only in part and some convention must be arrived at whereby the sign of a number can be represented, and whereby fractional values can be represented.

It is important that the sign of a number can be tested easily as in many processes of arithmetic, sign plays an important role in indicating when a process has finished or when a change of tactics is required.

A convention has been arrived at whereby the most significant digit is used as a sign bit, its presence or absence indicating whether a number is negative or positive respectively. A negative number could be represented merely by adding a sign bit to a positive number, but it is preferable to represent it by taking the complement of its positive counterpart. To subtract A from B we can then merely add the complement of A to B: this calls for just an adder in the computer rather than an adder and a subtractor.

Working inside a six bit framework the complement of the binary

number 0 0 1 1 1 0 is:

$$\begin{array}{r} 0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,1\,1\,0 \;- \\ \hline 1\,1\,0\,0\,1\,0 \\ \hline \end{array}$$

Remembering that the most significant bit is a sign bit and has a negative value:

$0\,0\,1\,1\,1\,0 \;=\; -0+0+8+4+2+0 \;=\; +14$

and $1\,1\,0\,0\,1\,0 \;=\; -32+16+0+0+2+0 \;=\; -14$

The three main forms in which numbers are stored inside a computer are as integers, fractions ,and floating point quantities.

1.10.2 As a signed integer.

The word is looked upon as having a binary point immediately to the right of the least significant digit. Taking a six bit word as an example, the values of its bits are:

B5	B4	B3	B2	B1	B0
-2^5	2^4	2^3	2^2	2^1	2^0

↑ -binary point

It has already been shown that a number A is represented as a negative number by its complement. Another way to arrive at the complement of a binary number is to invert it and add one.

For example:

take 14	0 0 1 1 1 0
invert it	1 1 0 0 0 1
add one	1
answer is -14	1 1 0 0 1 0

This is very useful because it presents an easy way of forming the complement of a number by electronic means.

1.10.3 As a signed fraction.

The word is looked upon as having a binary point between the most and second most significant places. The most significant digit when present has the value - 1.

Again taking a six bit word as an example the values of its bits are:

B5	B4	B3	B2	B1	B0
-2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}

↑——— binary point

So $1 . 1\ 0\ 0\ 1\ 0 = -1 + \frac{1}{2} + \frac{1}{16}$

$= -\frac{7}{16}$

and $0 . 0\ 1\ 1\ 1\ 0 = +\frac{1}{4} + \frac{1}{8} + \frac{1}{16}$

$= +\frac{7}{16}$

The negative fraction is known as the 2s complement because it is the complement of the positive fraction with respect to 2. As with integers the complement may be arrived at simply by inverting and adding this time, the smallest possible fraction to it.

Example:

Take $\frac{7}{16}$	0 . 0 1 1 1 0
invert	1 . 1 0 0 0 1
add smallest fraction	0 . 0 0 0 0 1
gives $-\frac{7}{16}$	1 . 1 0 0 1 0

This is useful for the same reasons as with integers.

1.10.4 As a floating point number.

With integral and fractional representations, the range of numbers that can be represented given an n bit word is:

$$-2^{n-1} \leqslant \text{integer} < +2^{n-1}$$

$$\text{and} \quad -1.0 \leqslant \text{fraction} < +1.0$$

Taking a 24 bit word as an example, the decimal range that can be covered is 10^{7} to 10^{-7}. In some problems, especially those of a technical nature, numbers vastly different in magnitude are present. Such numbers cannot always be kept in the range made available by normal representations and so "floating point" notation is used.

It is common practice to scale down large numbers by multiplying them by powers of 10. For instance, rather than write 1,250,000,000,000 we write 1.25×10^{12}.

A similar technique may be employed inside a computer, powers of 2 being used in place of powers of 10.

A number Y is represented as:-

$Y = R2^{e}$ where R is the argument
e is the exponent

Such a number is represented internally as R and e, the 2 being understood.

The argument is stored as a signed fraction, but lies in the restricted range:-

$$-1.0 \leqslant \text{Argument} < -\tfrac{1}{2}$$

$$\text{and} \quad \tfrac{1}{2} \leqslant \text{Argument} < 1.0$$

It can also be zero.

The exponent is stored as a signed integer. Quite a small exponent gives an enormous range of numbers. For example, a nine bit exponent

gives numbers ranging from 2^{-256} to 2^{255}, or more familiarly, approximately 10^{-76} to 10^{76}.

In binary the argument lies in the ranges:

0.10000 - - - - 00 to 0.111 - - - 11 ($\frac{1}{2}$ to approx.1)
and 1.0000 - - - - 00 to 1.0111 - - 11 (-1 to approx.$-\frac{1}{2}$)

All the numbers in the above ranges have the sign bit and next most significant bit unequal. A number with this characteristic is referred to as a standard number. A computer with facilities for floating point working generally has a built in program which reduces all numbers to this standard form prior to any floating point operations. Also at the end of any arithmetic operation, the argument is shifted to bring it inside the range of standard numbers, the exponent being adjusted accordingly. This standardisation of argument and automatic adjustment of exponent is called normalising.

For an example of normalising, if the answer to an arithmetic operation prior to normalising is:-

0.0000010110 - - - - 000 001000000 (2^{64})
The normalising process changes it to
0.10110000 - - - - 000 000111011 (2^{64-5})
Similarly with a negative number
1.1111101100 - - - - 000 001000000 (2^{64})
becomes 1.01100 - - - - 000 000111011 (2^{64-5})

In this way the argument is shifted as near as possible to the m.s. end consistent with being in standard form, and as many significant digits as possible are retained inside the framework of the number.

As a standard argument takes the form 1.0 - - - or 0.1 - - - it is an easy matter for the normalising process to shift a non-standard argument until the most and second most significant digits present the correct configuration for a standard number.

1.11 Input to a computer

1.11.1 Bootstrap

Input to a computer may be obtained by various means, the most commonly used being punched cards, punched paper tape and magnetic tape.

On first switching on a computer however, its store may be completely empty (or full of rubbish which amounts to the same thing) and, as an input device may only be activated by programmed instruction, there must be some way of getting a simple program into the store which can control the inputting of further information.

This simple program is often called a bootstrap and the way in which it is entered into a computer varies with different computers. For example, on the I.C.T. 1900 there are two switches which, when operated, cause control waveforms to be set up inside the computer which make a paper tape reader read in 128 characters off a paper tape containing a bootstrap routine, into the first 32 lines of the store.

This action is controlled by the computer's logic and requires no guidance from previously stored instructions.

1.11.2 Punched paper tape as an input medium.

In order to demonstrate how instructions and data are put into a computer, the use of paper tape will be discussed.

Characters are readily represented by the presence or absence of holes in a row across a piece of paper tape. The more holes there are across the width of the tape, the more characters may be represented, and tape codes are in use having 5, 6, 7 and 8 holes. As the range 1900 at present uses 8 hole tape, the other types will not be discussed.

Program and data tapes are prepared on a tape editing set which consists of an electric typewriter, a paper tape punch and a paper tape reader all of which are interconnected. Information typed out can be

simultaneously punched out on paper tape in standard I.S.O./A.S.A. seven bit code. The eight hole is used for parity checking which will be discussed later.

The typewriter keyboard may be represented on the paper tape and in addition to the keyboard characters, additional codes are punched out for such functions as carriage return, tabulate etc.

It will be recalled that an instruction is made up of a function part and one or more address parts and in order to prepare a program on paper tape, a series of instructions is typed out. For example, if a programmer wishes to carry out an instruction of the type: transfer the content of line 562 (modified by the content of accumulator 2) plus the content of accumulator 4 to accumulator 4, he might type out the following:

001 4 2 562

where 001 is assumed to be the function code for the particular instruction.

On the paper tape the following characters would appear in sequence in coded form.

Carriage Return
New Line
Tabulate
0
0
1
Tabulate
4
Tabulate
2
Tabulate
5
6
2

It will be appreciated while the characters on the tape are representative of decimal numbers, the computer deals only in binary quantities. In order to convert the decimal input into instructions and binary data, there has to be an assembler program inside the computer and this is designed to read in tape, recognise the various characters and act accordingly. For example, the carriage return/new line tells the assembler program that the previously read instruction is fully read in, and that the information following these characters is to be read into the next store location as the next instruction in the program.

The numbers on the typewriter keyboard have a direct relationship to the value of the binary coded punch tape character and the numbers 0 to 9 are punched out as shown in Diagram 10.

Ignoring bits P and L, it can be seen that the binary value punched on tape is always 48 greater than the value of the number represented. It is simple to remove this extra 48, but the number is still not in the correct form. Taking the address part of the instruction as an example, it can be seen that the computer will read in binary 5, binary 6 and binary 2. It wants binary 562. The assembler program carries out a radix conversion on the binary coded decimal number in the following manner.

Take 101 (5)		101
Multiply it by 1010 (10_{10})	=	110010 (50_{10})
Store answer in X		110010 to X
Take 110 (6_{10})		110
Add it to the content of X 110010+110	=	111000(56_{10})
Replace content of X by 111000		111000 to X
Multiply content of X by 1010 1010.111000	=	1000110000(560_{10})
Replace content of X by 1000110000		1000110000 to X
Take 010 (2_{10})		010
Add it to the content of X 1000110000+010	=	1000110010 (562_{10})
Replace content of X by 10000110010		
Answer is in X (i.e. 562 in binary).		

The different parts of the instructions are formed by the assembler program into a complete instruction in binary form and this is placed into the store address selected by control. The control starting address is initially set up by information punched out at the front end of the paper tape.

P & L bits

P is for parity. With eight hole tape, it is arranged that one of the bits is used to make up the total number of bits in the character to an even number. As the tape is fed into the tape reader, the parity of each character is checked. If there is an odd number of holes in a character the tape reader "parity fails", indicating invalid information. As the most common fault in tape punching and reading devices is the dropping or gaining of one hole from a character, parity is a very powerful check of validity.

L is for location. The location hole informs the tape reader when a character is in position to be read.

1.12 Output from a computer

1.12.1 General

Output from a computer is in many ways a reverse process of input. Data exists inside the computer in binary form and it must be broken down into binary coded characters prior to output. This is done by another section of the assembler program. Like input devices, output devices include punched cards, punched paper tape and magnetic tape, but in addition to these there are devices which print the finished result directly on paper. Output printers exist in various forms. The simplest is the electric typewriter and this is normally used to monitor the progress of a computation rather than print out the results. A programmer may arrange for his program to give a short print out every few minutes as an indication of the way it is running. More ambitious out-

put printers are line printers and Xeronic printers.

1.12.2 Line printers

The Anelex is an example of a line printer and uses 120 print wheels mounted side by side on a common axis. Each wheel has 60 characters embossed around its periphery. A continuous sheet of paper passes between these wheels and a set of 120 hammers - one for each character position.

The print wheels rotate together at high speed and the hammers, activated by information from the computer, strike as the letter to be printed in that character position comes opposite the hammer. Interposed between the paper and the print wheels is a sheet of inked material, and as the print wheels rotate, first any A's in the line and then any B's etc., are printed until the whole line is built up.

The paper is then moved and the next line is printed, speeds of up to 1200 lines per minute being obtained.

1.12.3 Xeronic printers

At the centre of the Xeronic system is a revolving selenium coated drum and a cathode ray tube on which alph-numeric characters are formed as a result of decoding the computer's output. As the drum revolves through a full turn, the following series of events take place.

a) The drum surface receives an electrostatic charge.
b) Images from the C.R.T. are focussed onto the drum, dissipating the charge where the light falls.
c) Charged powder dusted onto the drum and being repelled by the charged drum, sticks only on the discharged characters.
d) The printing paper, moving at the surface speed of the drum, is brought into contact and the powder character transferred from drum to paper by applying a further charge.
e) The powder image is fused onto the paper by heating.
f) The drum is dusted off prior to recycling.

1.13 Internal checking of information accuracy.

The value of an operand or address moving around or stored in a computer can change due to temporary or permanent fault conditions. In order to limit the possibility of a computer "making a mistake" certain checks can be carried out on numbers as they are handled inside the computer.

A very likely fault in a computer is for a digit to be lost or gained. It has already been pointed out how the parity of paper tape characters is checked. In a similar way words kept in a computer store have their parity checked on exit from the store. As a word is written into store its parity is calculated, and in order to accomodate this extra parity bit a store generally has space for one bit more than the word length. When the word is read out of store the parity bit is also read out and a copy of the store's output is automatically fed to the parity checking circuit. If a word with an even parity bit contains an even (or odd in some applications) number of digits, the computer stops and indicates a parity failure.

Another common method of checking information accuracy is to form a check-sum of all the words in a block of information. This is common with magnetic tape. In this application, several hundred words may be transferred in a block from a computer's central store to the tape. As the words are written onto the magnetic tape they are also fed into a cumulative adder. When all the words have been transferred, the check sum is taken from the adder and written onto the magnetic tape immediately after the information. On reading information from the tape, a similar check-sum is formed and compared with the check-sum at the end of the block of information. If all is well the check-sums are identical.

CHAPTER 2

THE DISADVANTAGES OF A SIMPLE COMPUTER

2.1 Peripheral speed limitations

The nature of some computing problems leads to a lot of time being spent in getting information into and out of a computer and not very much time being spent in computing.

The list below shows the speed at which some modern peripheral devices will run.

Line printer		approximately	2000	characters	/second
Punched cards	(reading	"	1600	"	"
	(punching	"	250	"	"
Paper tape	(reading	"	1000	"	"
	(punching	"	300	"	"
Magnetic tape	(reading	"	64000	"	"
	(writing	"	64000	"	"
Teleprinter		"	10	"	"

Some of these are very fast but a modern computer is several orders of magnitude faster than this. Such a computer may be capable of over 100,000 operations per second. If a computer can read in a paper tape character in 10μs and then has to wait for 1ms before another character is in position to be read, it is only using 1% of its time usefully; a computer which is slowed down in this way is said to be peripheral limited. Paper tape punches are slower than tape readers, and the time wasted on paper tape output is even greater than on input.

One way to utilise the computer's time more efficiently is to have faster peripherals (input/output devices). Magnetic tape has the same

speed on writing as on reading and transfer rates of the order of 1 character every 15μs can be realised with $\frac{1}{2}$ inch tape.

It is common with magnetic tape for the tape electronics to assemble a number of characters (or stripes) into a word prior to transferring the word to the computer. With the 1900 range for example, the magnetic tape stripes each contain six information bits and four such stripes are assembled into a four character, 24 bit word prior to being transferred to the computer. Using this technique, in the case of the 1900 range magnetic tape can transmit or receive a word every 60μs. Even at this high speed however, 80 to 90% of the computer's time can be wasted whilst involved in input or output of data.

Magnetic tape is more suitable as a store than as the sole input/output medium of a computer. Paper tape is a good medium on which to prepare programs and data. It is cheap, and the characters are visible, being easily rectified by hand. A converter would have to be used to transfer the information from such a paper tape onto magnetic tape, and unless there were several of these, the converters themselves would become bottlenecks. Such a converter having computer complexity is very expensive.

2.2 Improvement due to autonomous transfers

Some of the time wasted due to the slowness of peripheral devices can be avoided by arranging for transfers to take place autonomously. This technique is outlined below.

Suppose that a programmer requires a transfer of a word from peripheral X to a specific store area. Associated with each peripheral is a special store and into peripheral X's special store is placed information defining the store area to be involved in the transfer. This may take the form of a starting address and the number of words to be transferred, or a starting and finishing address.

Having set up these conditions, peripheral X is told to start transferring and the computer carries on with its program. Every time peripheral X is ready to transfer a word or character it sends a signal to the computer which causes it to hesitate at some suitable point, and the content of peripheral X's special store is used to address the main store at this point. The transfer between the main store and peripheral X then takes place.

The transfer address is modified (in order that the next store position may be selected when peripheral X is ready to do its next transfer) and placed back in the peripheral's special store.

If, at the completion of the word or character transfer, no other peripherals are seeking attention, the computer then carries on with its computations until such time as anyperipheral is ready for a further transfer of information. If however the computer runs out of instructions it can usefully perform before a transfer is completed, time is still wasted.

Ideally, when a system is designed for autonomous transfers, the only way in which the central processor is held up by peripheral transfers is that it may have to wait momentarily for access to the central store. The control boxes of peripherals in such a system are more complex than usual and contain the following additional items:

1) a store for holding the transfer address (the central store address to and from which the next character or word is to be transferred) and either the finishing address or the number of words left to transfer,

2) circuits to modify the transfer address after each transfer and to detect when the required number of words have been transferred.

A computer with provisions for the attachment of many peripherals is expensive using the above system, and for reasons of economy (and greater reliability through the use of less components) a simpler approach

is often used.

A computer's central store and processor are used to hold and process the necessary control information. This lengthens the time required to transfer a word or character to or from a peripheral and may involve temporary removal of operands from the central processor whilst the transfer address is being modified. This technique is used in the 1900 range of computers and, as the central processor is very fast, little computing time is lost.

If a program requires access to a store area which is currently involved in a peripheral transfer, the program must be suspended until the transfer is completed. This can be done either by having the computer check the store transfer area of each busy peripheral every time an access to store is made, or by issuing an order of the type "suspend the program if peripheral X is busy". This type of order would stop the program running until the peripheral involved had filled up the transfer area.

The first method involves considerable additional hardware but no action on the part of the programmer. The second method requires that the programmer is constantly aware of the store locations of any active peripheral transfers which he must guard against disturbing.

Most programs gain some advantage from autonomous transfers, but a program with a lot of input/output, using a few peripherals and not very much computing, can still render such a computer idle for much of its operating time.

In order to utilise a computer's time as fully as possible the technique of time sharing has been evolved. The idea behind this and the additional problems resulting from its use are discussed in the next chapter.

CHAPTER 3

TIME SHARING

3.1 Principles of time sharing

The central and backing store of a computer designed to work in time sharing may contain information relevant to several entirely unconnected programs.

Such a computer is designed so that immediately it comes to an "incident" it switches into a permanently resident organising program. This program determines what type of incident has occurred and then takes appropriate action.

An example of an incident is the initiation of a peripheral transfer. When a program arrives at a point where it requires a transfer of information between a peripheral and the computer, the program is interrupted and the organising program entered. This program arranges for the peripheral to transfer the information and then ascertains whether the original program can proceed. If it cannot, because it is awaiting the end of the transfer, the organising program makes the computer proceed with an alternative program. Many time sharing computers work on a program priority system, and in the above case, the organising program would enter the highest priority alternative program available.

The ideal situation with a time sharing computer is that both the peripherals and the central computer are occupied all the time. The actual situation is usually either

1) that various peripherals are standing waiting because further input is not yet required or further output is not yet available, the central computer being fully occupied;

or 2) that all engaged peripherals are working on transfers, and all programs are held up awaiting the completion of same, the central computer being idle in between autonomous information transfers.

Which of these situations arises depends upon various factors, the main of which are the type and number of programs in the computer and the speed of the peripherals being used. To obtain maximum efficiency from the installation as a whole, some thought has to be given to the type of programs loaded into a computer at any one time. Their relative priorities are important, it being preferable to give high priority to programs involved in many peripherals transfers and low priority to those involved in few. With this arrangement the peripherals are kept very busy and, if the higher priority programs are held up awaiting transfer, lower priority programs requiring few transfers will keep the central computer busy.

A simplified timing diagram of a computer executing three programs in time sharing is shown in Diagram 11. The programs are arranged in order of priority P1, P2, and P3, and the general rules of time sharing may be gathered by following through the sequence of events.

3.2 Additional facilities required with a time sharing computer

Various extras are required in a time sharing computer. These are:

a) a larger store than a non-time-sharing machine to accommodate the extra programs;

b) facilities to ensure that one program cannot interfere with another;

c) facilities for addressing peripheral devices in order that a program may have specific peripherals allocated to it by the organising program;

d) a supervisory program to carry out time sharing, organisation and monitoring of programs;

e) a hesitation system to allow transfer of information;

f) facilities for adding a datum to all addresses in a program in order that a program may be located anywhere in the store,

the datum being allocated by the organising program on input of a program;

g) facilities to ensure that no access is made to a store area involved in peripheral transfer until the transfer is completed.

The way in which some of these facilities are realised varies from machine to machine. How they are realised in the 1900 range of computers is explained in the following chapters.

CHAPTER 4

GENERAL INTRODUCTION TO THE I.C.T. 1904, 5, and 9.

4.1 The 1900 series

The 1900 series is comprised of a range of 7 computers each of which has been developed from the parent computer, the I.C.T. 1900. The computers in the series range from a single program machine with a minimum store size of 4K words (where K = 1,024) up to a 16-program machine with a maximum store size of 256K words. All the peripherals and program languages used in the range are compatible so that any particular installation may be expanded when necessary, either in terms of a more powerful central processor or additional peripheral equipment.

Every computer in the series can work in both fixed and floating point mode, but whereas the 1902, 3, 4, and 6 have floating point operations directed by a program held in their stores, the 1905, 7, and 9 have built in floating point units which give much greater operating speeds.

The range uses the I.C.T. standard interface which allows standard interface peripherals to be easily connected to any computer in the series by a standard 76-way connector. With a minority of peripherals standard interface control cannot or need not be used and special peripheral controls are used instead.

Information about the control of peripherals over both special and standard interfaces is given in Chapters 10 and 11, and Chapter 5 briefly describes each peripheral available for use with mention of the type of interface over which it works.

Some of the attributes of the various models in the series are summarised in the following table. It will be seen that the 1904, 5 and 9 fall into a distinct sub-group within the range and it is these three

models that are discussed collectively in this handbook.

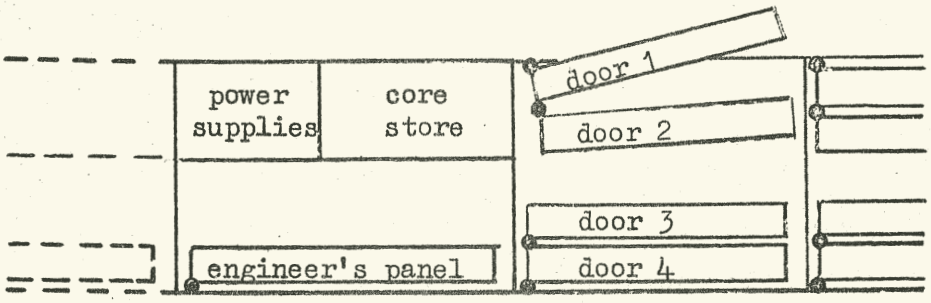
Series number	1902	1903	1904	1905	1906	1907	1909
Store speed (μs)	6	2	2	2	1.1 or 2.2	1.1 or 2.2	6
Store (K words) capacity	4,8 or 16	4,8 or 16	8,16 or 32	8,16 or 32	n.32 up to 256	n.32 up to 256	16 or 32
Number of programs and sub-programs	1 with 2 sub	1 with 2 sub	4each with 2 sub	4each with 2 sub	16each with 3 sub	16each with 3 sub	4each with 2 sub
Floating point unit	no	no	no	yes	no	yes	yes
Input/output channels slow	8	8	18	18	18	18	18
fast			5	5	any	any	5

4.2 The 1900, 1904, 1905, and 1909 - general

The 1900/4/5/9 computers represent the medium sized machines in the 1900 range. Each is a 4-program machine capable of both scientific and commercial applications, the 1905 and 9 being preferable where much scientific work is to be undertaken.

The computers are fully parallel machines with a clock rate of 1 mc/s. Internal working is in binary and the 24-bit word may be used as an instruction, a signed integer or fraction, four 6-bit alpha numeric characters, or a counter modifier. Two or more words may be used together for greater accuracy, and double length floating point numbers may be represented with a 9-bit exponent and a 38-bit argument.

The basic computers are arranged as in the plan view below, and each cabinet is approximately 5 feet × 4 feet × 3 feet deep.



added for greater than 16K store, extra power supplies, or extra monitoring

basic machine with either floating point or a few peripheral controls

peripheral controls

The power supplies are rack mounted and, in the basic machine, consist of 5 racks of equipment providing a.c. control and the necessary d.c. supplies. The power supply racks are easily removed for servicing.

The core store which is a self contained unit with its own power supplies and read write electronics is also readily accessible.

As a result of differing peripheral and floating point requirements, the disposition of the sections of the computer varies from machine to machine. The central processor is usually contained in doors 1 and 2, and part of door 3; and in the 1905 and 9, and those 1900s fitted with floating point units, floating units are contained in door 4 and the rest of door 3.

Hence the basic computer, with or without floating point, is contained in only 2 cabinets. As the 1900/4/5/9 computers are multiprogram machines, a fair number of peripherals is required, and practicable installations have a minimum size of 3 cabinets with as many additional cabinets as are necessary to contain the extra logic involved.

Door pairs are hinged back to back as indicated in the plan view of the machine: this method of construction makes the back wiring particularly accessible. The doors each contain 7 card files each of which has positions for 38 plug-in cards. These cards contain the circuit elements that make up the computer's logic.

The computers use solid state devices throughout and the main logic comprises NOR elements and bistables which fit, six and two respectively, to a card.

4.3 Organisation of programs and peripherals

A program called Executive permanently occupies the first few thousand store lines (its size varies with the size of the installation) of each model and organises the work in the computers. The Executive programs used at present allow the computers to time share between four programs, each with up to two sub-programs. These programs are arranged in a priority list and every time an incident occurs which temporarily prevents the current program from carrying on (for instance, it may be awaiting the completion of a peripheral transfer) Executive is entered. Executive scans the priority list and re-enters the highest priority program available.

Each program, when entered into the computer, has an area of store reserved for it by Executive, and each of these reservations is defined by two addresses, the datum and limit. The first eight addresses of each program's store area are used as accumulators, and three of these may be used as modifier registers.

Any address referred to by a program has the program's datum added to it and is checked for being within the reserved area before being applied to the core store. Both of these operations are completely automatic. If, due to a miscalculation on the part of the programmer, a program refers to an address outside its reserved area, a datum/limit failure occurs and Executive suspends the offending program.

Complete inter-program protection is thus ensured and each program may be written as though it were the only program to run in the computer.

Executive also provides certain macro-instructions for the control of programs and peripherals and, in the case of the 1904, for carrying out floating point arithmetic.

The sequencing of machine orders and the transfer of data to and from peripherals during hesitations is directed by a hardware program called the microprogram.

A large variety of peripheral devices, both conventional and unconventional, may be added to the computers, making the systems versatile and adaptable to data-processing, scientific, and real time applications.

4.4. Peripheral transfers

When a program calls for a peripheral transfer, an interrupt occurs, Executive is entered and the appropriate peripheral is activated. The computer then goes back to executing the program and the peripheral transfer takes place on the hesitation principle: that is, every time a peripheral is ready for a word or character transfer, it causes the central computer to hesitate over the program it is executing whilst the word or character is transferred between the peripheral and the core store. Each hesitation takes up 6μs of the central computer's time in the case of the 1904 and 5, and 16μs in the case of the 1909. As most peripherals do not work at anything like this rate, the computers can handle many peripherals and run programs concurrently. For example, with the 1904 and 5 computers a 300 ch/sec paper tape reader requires 0.2% of the central computer's time and a magnetic tape reader about 10%.

When a peripheral transfer is completed, a further interrupt causes another entry to Executive. Executive notes that the transfer is completed and enters the highest priority program available.

It is necessary, when engaging in transfers, to ensure that the

program connected with the transfer is locked out from the transfer area until the transfer is completed. When a program has reached the end of the word it can do before it requires access to such a transfer area, the instruction "suspend me if my unit X of type N is busy" (which is provided in the function code) must be used. If the unit referred to is still busy, Executive suspends the program and enters the highest priority alternative program available.

Except in the case of very fast peripherals, each peripheral has a control unit associated with it. Each type of fast peripheral, because of its speed, may share one or more control units with others of its type. The control units are addressable and are used to pass data and control information to and fro between peripherals and computer. In addition to the control unit, each peripheral has a core store word or area associated with it and this is used to contain information about the transfer such as the starting address and word or character counts.

In the case of a unbuffered broadside-fed card punch the same block of 80 characters must be transferred from the computer to the card punch 12 times in order to punch out the card row by row.

To avoid the necessity for 12 separate transfer instructions the special hesitation facility is provided, and this allows the necessary repetitive transfers to take place as a result of only one card punch instruction. This facility is used with all unbuffered peripherals which require reiterated block transfers, and as with normal hesitations, is controlled by a section of the hesitation microprogram.

4.5 Operator control

Each computer is normally controlled from an electric typewriter which enables the operator to give instructions to Executive and vice-versa. A large number of these instructions are concerned with manual action on peripheral equipment, but others concern program starting, monitoring, priority changing, etc. In addition to its keyboard, the typewriter has eight illuminated indicators and eight illuminated push button switches.

These are used to indicate which program is running, to call the attention of Executive to an incoming message, to cancel such a message, and so on.

4.6 Engineers' control

Engineers have much more comprehensive access to the computer via the engineers' panel. The contents of the various bistable registers and control bistables are displayed on the panel, and there are facilities for injecting instructions and operands via handkeys, single shot working, and margin testing.

CHAPTER 5

PERIPHERALS AVAILABLE FOR THE 1900/4/5/9

5.1 Input devices

5.1.1 The 1915 and 1916 paper tape readers

The 1915 and 1916 paper tape readers can read data punched in 5, 6, 7, or 8-track paper tape at up to 300 and 1000 characters per second respectively. Characters are read from the tape by means of photo-electric cells, and are transferred to the computer until either the number of characters called for has been transferred or, if specified in the program, an end of block character is detected. Accuracy is checked by a parity bit and when a parity failure occurs, the reader stops at the failing character and Executive informs the operator, via the console typewriter. The type of parity (even or odd), coding of end of block characters, and the number of tracks are determined by interchangeable printed boards, one of which is used at a time.

The standard code is the I.S.O. 8-track (7 data plus parity) code, which is automatically translated into internal 6-bit code by means of hardware. Other codes must be translated by the use of appropriate subroutines in the program.

The readers are designed for use over the standard interface but some early machines and all 1909 models use special control boxes.

5.1.2 The I.C.T. 1911 card reader

The 1911 card reader is a standard interface device and reads standard, 80-column cards at speeds of up to 900 cards per minute. The standard code is the I.C.T. 64-character code which is automatically converted into internal 6-bit code by hardware. A card image feature may also be specified which enables the two 6-bit halves of each 12-bit

column to be transferred directly to the computers store.

The capacity of the input hopper is 3000 cards and that of the output stacker, 2400. The cards are fed sideways from the hopper, into a waiting station with a capacity of 2 cards. On demand, a card is immediately taken from the waiting station and end-fed through the reading station where it is read, column by column, by 12 photo-electric cells. Each column is read twice at the same station for checking purposes, correctly read cards passing into the output stacker and incorrectly read cards passing into the 450-card reject stacker. When a card is mis-read Executive informs the operator via the console typewriter. When a card leaves the waiting station it is quickly replaced by a further card from the hopper so that there is always at least one card in the waiting station immediately available for transport through the reading station.

5.1.3 The I.C.T. 1912 card reader

The 1912 card reader is a standard interface device and reads standard 80 - column cards at speeds of up to 300 cards per minute. The standard code is the I.C.T. 64-character code which is automatically converted into internal 6-bit code by hardware.

The capacity of both the input hopper and the output stacker is 2000 cards. The cards are end fed, one by one, from the hopper to the two photo-electric reading stations, one for reading and the other for checking, and any misread cards are automatically diverted to the 500 card reject hopper. The action for mis-read cards is as with the 1911 reader.

5.2 Output devices

5.2.1 The I.C.T. 1925 paper tape punch

The 1925 paper tape punch is normally a standard interface device

but some early models and all 1909 installations will have special interfaces. Data can be punched in 5, 6, 7, or 8-track paper tape at speeds of up to 110 characters per second. Data are transferred until either the number of characters called for has been transferred or an end of block character is detected.

The standard punching code is the I.S.O. 8-track code and conversion from internal 6-bit code is done by hardware. Other codes are dealt with by appropriate program sub-routines.

5.2.2 The I.C.T. 1921 card punch

The 1921 card punch is a standard interface device and can punch standard 80-column cards at speeds of up to 350 cards per minute. The cards are punched in I.C.T. 64-character code, conversion from the computer's internal 6-bit code being carried out by hardware.

The capacity of the input hopper is 1000 cards and that of the output stacker, 1250. The cards are sideways fed one by one, from the hopper to the punch station where they are held stationary while being punched. The punching is done by 960 knives (80 columns × 12 rows) to which information is fed one row at a time by means of the special hesitation facility. Information accuracy is ensured by checking that the relevant punch knife interposers have operated, and mis-punched cards are diverted into a reject stacker. When a card is mis-punched, an entry is made to Executive and Executive arranges for the data to be repunched on another card. Should persistent mis-punching occur the punch is stopped and the operator is informed via the typewriter.

5.2.3 The I.C.T. 1920 card punch

The 1920 card punch is a standard interface device and punches standard 80-column cards at speeds of up to 100 cards per minute. The cards are punched in I.C.T. 64-character code, conversion from the computer's internal code being carried out by hardware.

Cards are broadside fed from the 800-card input hopper to the punching station. The punching is done by 80 knives which punch out the 12 rows one at a time as the card moves through the punching station. Information is presented to the punch by means of the special hesitation facility, and information checking is done by counting the total number of holes on the card at a brush-type read station situated behind the punching knives. Corresctly punched cards are fed into the 650 card output stacker and mis-read cards into a reject stacker. Mis-punched cards are treated as with the 1921 card punch.

5.2.4 The I.C.T. 1922 card punch

The 1922 card punch is a standard interface device and punches standard 80-column cards at speeds of up to 33 cards per minute. The cards are punched in I.C.T. 64-character code, conversion from the computer's internal code being carried out by hardware.

Cards are endfed from the 450 card input hopper to the punching station. The punching is done by 12 knives which punch out the information column by column. The accuracy of punching is checked by monitoring the displacement of the punch knives. All cards are fed to the output hopper and mis-punched cards cause the punch to stop in order that the mis-punched card may be removed from the top of the pile by the operator. Executive arranges to have the same set of data transferred again when the punch is re-engaged.

5.2.5 The I.C.T. 1933 line printer

The 1933 line printer is a buffered printer which will operate at 1350 lines per minute when 48 consecutive characters are in use and 1100 lines per minute when all 64 characters are in use. The printer is available in three forms:

1933/1 - 96 print positions per line,
1933/2 - 120 print positions per line,
1933/3 - 160 print positions per line.

Character spacing is ten to the inch and characters available are; A to Z, 0 to 9, space, and 27 symbols.

The parity of information is checked in the printer's stand-off electronics. A parity failure causes a lozenge to be printed and an entry to Executive which stops the printer, and informs the operator via the typewriter.

Line spacing may be set at either 6 or 8 lines to the inch by the operator and fast throws, controlled by an 8-channel paper control band, take place at 31 inches per second.

Conversion from 6-bit internal computer code into printed characters is automatically performed by hardware.

5.2.6 The I.C.T. 1932 line printer

The 1932 line printer operates at a speed of 600 lines per minute and can be supplied in buffered or unbuffered form, the special hesitations facility being used in the latter case. Line widths of either 96 or 120 print positions are available with lateral spacing of 10 characters per inch and vertical spacing of 6 lines per inch.

In other respects the printer is similar to the 1933 model.

5.2.7 The I.C.T. 1931 line printer

The 1931 printer is the same as the 1932 except that the speed is 300 lines per minute.

5.3 File storage equipment

5.3.1 The I.C.T. 1953 exchageable disc store.

The I.C.T. 1953 exchangeable disc store consists of an exchangeable disc cartridge, a transport with a disc drive mechanism and heads, and control logic for up to four transports. Control will be effected via the I.C.T. standard interface.

A cartridge provides storage space for 4.03×10^6 characters and contains 6 discs, both sides of which are used. 24 heads, two per disc surface, may be moved radially, all at the same time, to any of 100 positions over the discs' surfaces, providing 200 tracks of data on each side of each disc. Each disc surface is divided into 4 sectors which divide each track into 4 blocks of 420 characters. Thus, the capacity of a cartridge is:

$$420 \text{ characters} \times 4 \text{ blocks} \times 200 \text{ bands} \times 12 \text{ surfaces} = 4.03 \times 10^6 \text{ characters}$$

A seventh disc, the block code disc, generates signals to indicate the start and end of each block, and a block is the smallest addressable area on a disc. A block is addressed in terms of the transport number, the head number, the head position, and the sector number.

The access time to the store is 75ms for head relocation plus an average of 12.5ms for disc rotations. Once the heads are located random or sequential access can be made to 40,320 characters, that is:-

24 (heads) × 4 (blocks per head) × 420 (characters per block)

Data, which are stored serially, can be read from or written to the file at the rate of 66,000 characters per second. Information is checked by means of a parity bit after every four characters, and on a writing operation the parity is checked by reading back during the next rotation of the disc. This check is carried out by the file's control

electronics and does not occupy the central computer. Any error causes an interrupt and Executive will try to rectify the situation. If successive re-reads or re-writes do not clear the parity, the operator is informed via the typewriter.

When it is necessary to load an alternative cartridge into the transport unit, the drive is automatically disengaged and the heads withdrawn until the new cartridge is placed in position by the operator.

5.3.2 The I.C.T 1956 Data disc store

The I.C.T. 1956 data disc store provides storage, in one unit for 31.5×10^6 characters and four such units, providing storage for 126×10^6 characters, may be controlled via one access control. The arrangement is shown in block form on Diagram 21. Random access may be made to any 240 - character sector within the system.

A storage unit contains sixteen 31in magnetic discs both sides of which are used. Associated with each disc are 8 heads, four per side, which may be moved radially, all at the same time, to any of 64 positions on the discs' surface. This provides 256 tracks of data on each side of each disc. The outer 128 tracks are divided into twenty 240-character sectors and the inner 128 tracks into twelve. Thus the capacity of a disc unit is:

$$240 \text{ characters} \times (20 + 12)/2 \text{ sectors} \times 256 \text{ tracks} \times 32 \text{ surfaces} = 31.5 \times 10^6 \text{ characters}$$

In addition to the sixteen data discs there are two smaller control discs which hold sector address information, and a clock for reading and writing timing purposes. A sector is addressed in terms of the unit number, the disc number, the head number, the head position, and the sector number.

The access time to the file is an average of 190ms for head positioning plus an average of 30 milliseconds for disc rotation. Once the set of 8 heads associated with a disc is located and the beginning of the addressed sector is under the read/write heads, data may be passed between the computer and the file, to or from a number of consecutive sectors, at a rate of 80,000 and 48,000 characters per second for outer and inner track zones respectively. If a block of consecutive sectors is spread over two discs or two head positions, Executive arranges for the new positions to be located.

Data accuracy is checked in the same manner as with the 1953 exchangeable disc file.

Areas of the disc file may be reserved for programs in much the same way as are areas of core store. Disc file reservations are controlled by Executive and result in the disc file being available to any program running in the computer. Relative addressing is used so that a programmer may address his file from zero, a datum being automatically allocated by Executive to any file recorded on the disc file. A list is available to Executive relating the titles of files currently stored on a disc file to their sector starting addresses. A programmer need only quote his file's title and Executive will refer to this and use the sector starting address as a datum to add to the relative sector addresses quoted in this program. The file name/sector address list may itself be stored on the disc file.

5.3.3 The I.C.T. 1958 magnetic card file

One magnetic card file unit contains 1 or 2 retrieval units which each hold up to eight magazines. A magazine holds 256 magnetic cards each of which is 4 by 16 inches and contains sixty four bands of information. One such band consists of two 1300 character tracks which are simultaneously read from or written to by two magnetic heads. Up to four magnetic card file units may be controlled by one magnetic card file control unit. On-line capacity is thus,

4 (file unit)× 2 (retrieval unit) × 8 (magazines) × 256 (cards) × 64 (bands) × 2600 (characters)

$= 2{,}730 \times 10^6$ characters

As card magazines may be exchanged by the operator, off-line capacity is virtually unlimited.

Each card in a magnetic card file can be addressed in terms of its file unit, retrieval unit, magazine number, and position therein. When a transfer of information is required, the addressed card is extracted from its magazine by its retrieval unit and passed to a rotating drum. The card travels round with the drum and data are transferred to or from the card by means of a pair of magnetic heads adjacent to the drum's surface. There are four such head pairs and they can be moved, together, to any of sixteen positions, so accounting for the 64 (4 × 16) bands of data on a card. The head pair and position also form part of the address, resulting in the 2600-character band being the smallest addressable area inside a card file. The access time to a card is, on average, 325ms, and the data transfer rate is 80,000 characters per second.

When the card has passed round the drum it can be either returned to the magazine or recirculated round the drum. The head assembly may be repositioned so that successive bands are transferred during successive revolutions of the drum and, if desired, the entire content of a card may be transferred before returning it to its magazine.

Data are recorded as 6 - bit serial characters each with a parity bit. When a parity error occurs an interrupt causes entry to Executive as usual and Executive takes similar action to that taken with the exchangeable disc file.

5.3.4 The I.C.T. 1962, 1963, and 1964 drum storage systems

The drum store system provides a range of storage from 131,072 characters to 8,338,608 characters and, in all cases, direct access can be made to individual 24-bit words stored within the system.

Drums are attached to a 1900 range computer via a drum control and a standard interface channel, and up to four drums of any particular type can be controlled by a drum control.

Data are recorded magnetically in serial form by the means of fixed read/write heads and, for checking purposes, a parity bit is inserted after each 6-bit character by the drum control unit.

The main characteristics of the storage system are summarised in the following table.

Drum type	Number of tracks or heads	Capacity of single drum (characters)	Average access time (ms)	Maximum transfer rate (characters)
1962	128	131,072	10	50,000
1963	256	524,288	10	100,000
1964	512	2,097,152	20	100,000

5.3.5 The I.C.T. 1974 magnetic tape system

The 1974 magnetic tape system can read or write to half inch, 7-track magnetic tape in even or odd parity mode and, providing that information is written onto a tape at the appropriate packing density, the tape will be compatible with the other 1900 tape systems as well as with most internationally used tape systems. Tapes are stored on 2400 foot spools.

Up to twelve decks may be connected to a computer and they are connected via special control units situated at the computer. Each deck

has a unit control and the decks share up to three channel controls giving simultaneity of operation to up to three tape decks at any one time, and access to all decks over a period of time. Diagram 20 shows the arrangement in block form.

The tape read/write speed is 120 inches per second, and at the packing densities of 800, 556, and 200 characters per inch the data transfer rates are 96000, 66667, and 24000 characters per second respectively. Decks are fitted with a switch to give the user alternative densities of 800 and either 556 or 200 characters per inch.

The length of a block of information (i.e. information transferred as the result of a single transfer instruction) is specified by the programmer. Blocks of between 1 and 32,768 words in length may be used, making possible the transfer of the total content of the core store with the use of only one instruction.

The parity of data in a block is generated by the magnetic tape channel control unit and is written in two ways: the lateral (character) parity is recorded on the seventh track and the longitudinal (track) parity is recorded in an additional stripe called the check character at the end of the block. Longitudinal parity is always even and, on tapes originating within the system, lateral parity is usually odd. Even lateral parity is however, available for use when necessary. The mechanism of parity checking is identical on reading and writing in that data are read back as they are recorded during a write operation. On the completion of a block transfer any parity failure is noted by Executive and a repeat transfer is arranged. On a writing operation, after several repeated failures Executive erases the faulty information and tries again further along the tape. Insurmountable failures on read or write operations result in a message to the operator via the console typewriter.

A block of information on a tape may contain several records, and a number of blocks make up a file. A file may be spread over several reels of tape. Divisions between significant sections of information on a tape are indicated by means of a tape marker (octal 17) which,

though termed an end of file (E.O.F.) marker, may indicate the beginning or end of a section of a file. The programmer has facilities enabling him to skip forward or backward to the next or previous E.O.F. mark respectively.

The first block of information on each tape is a 16-word label containing such information as the tape's name, the date on which it was written, the minimum number of days for which the information should be retained, etc. The last mentioned defines the time for which Executive, which has access to the current date, should protect the tape. Further identification of content is provided by 5-word qualifier blocks which normally follow E.O.F. marks and give the context in which an E.O.F. mark is used.

Tapes are normally assigned to a program by Executive at the point in the program at which they are first required. At this point, Executive carries out a deck by deck search until the tape label whose name part corresponds to that given in the program is found. The information given in the label is then made available to the program. Should the required tape not be mounted on any deck, a request for the tape is delivered to the operator via the console typewriter. When a tape has been located it is left in a position where it is ready to read or write to the block that follows the label.

Manual assignment of tapes is also possible and in this case, the operator types a message to Executive assigning the tape deck upon which the desired tape is mounted to the program requiring it.

5.3.6 The I.C.T. 1971, 1972, and 1973 magnetic tape systems

The 1971, 1972, and 1973 magnetic tape systems are each available as a group of 2, 4, or 6 decks and each group is contained in a single unit with its own control electronics. The control electronics are designed to I.C.T. standard interface requirements.

The read/write speed and recording rates of each system are shown in the following table.

System number	Read/write speed (in/s)	Recording densities (ch/in)	Character transfer rates (ch/s)
1973	75	200	15,000
		556	41,700
		800	60,000
1972	75	200	15,000
		556	41,700
1971	37.5	200	7,500
		556	20,800

The three systems are compatible with the 1974 system in terms of both information layout on tape and programming so that information on these aspects in the previous section can be applied equally here.

5.4 Other equipment

5.4.1 The I.C.T. 1996 program timer and real time clock

The program timer and real time clock are when provided, contained in the computer proper, the former being used for measuring the central computer time occupied by each program and the latter for program dating and log-keeping purposes.

The program timer is a 21 - stage binary counter which counts off the computer clock and has a capacity of approximately 2 seconds. Executive sets the timer to zero each time a program is entered and accumulates the value that is in the timer when the program is interrupted.

Should a program run free of interrupts for the time that the timer takes to run up to half of its capacity (approximately 1 second), the counter causes an interrupt and Executive adds the timer value to the time already accumulated for the program. This interrupt also provides an opportunity for any higher priority program to be re-entered.

The timer does not operate during hesitations as most of these are usually connected with programs other than that running at the time. Executive makes suitable allowances based on each program's use of peripherals and adds these to the programs' running times. When a program is deleted, its time total is printed out.

The real time clock provides the local time in hours, minutes and seconds. This time is available to programs and, in addition, is periodically typed out on the console typewriter to form a part of the computer's operating log.

5.4.2 The I.C T. 1830 general purpose visual display unit

This unit contains a 21-inch C.R.T. which may be used for graph and vector plotting and the display of alpha-numeric characters; and a keyboard by which the operator may communicate with the central computer. Each unit is self-contained, no separate control unit being necessary and is connected to the computer via a standard interface channel.

The amount of information displayed on the screen can be as high as one hundred and twenty eight 256-character lines. This amount would however, be used only in conjunction with the 35mm camera described later. For visual work, up to sixty four 128-character lines of double sized characters are used.

Characters are formed on the C.R.T. by combining sections of various Lissajous figures. The various gated sine and cosine waveforms necessary for these are produced by a complex of electronics, the outputs of which are dependent upon the character code applied to the input.

The outputsare applied to a pair of minor deflection coils mounted on the C.R.T. adjacent to the main deflection coils. The position of the resulting character on the face of the C.R.T. depends upon the currents in the main deflection coils and these currents each have 1024 discrete magnitudes giving rise to a 1024 square matrix. Co-ordinate points in the matrix can be specified for graph plotting and as the starting points of vectors.

All information to be visually displayed is contained in a 4096 - character buffer store. As the data are read from this in sequence, the current in the C.R.T. main deflection coils is stepped in such a manner as to write the data at the appropriate points of the scan matrix. The buffer store is cycled at least ten times per second in order to provide a flicker free display.

Optional extras to be used with the 1830 unit are a 35mm camera and a light pen. The former can take pictures on demand, at rates of up to 20 frames per second and for this purpose, an additional high definition 5 inch C.R.T. is used. When the camera is in use, each group of characters need be displayed on the C.R.T. for one scan only and masses of information can be transferred onto film in a very short time.

The light pen is a photo-electric probe connected to the 1830 unit by a flying lead. When placed against the C.R.T. face, its position may be ascertained in terms of the time at which it is illuminated in the scanning cycle of the C.R.T. An immediate use for it will be to indicate characters which need changing or deleting but future use, subject to programming problems being overcome, may include the recognition, by the computer, of "drawn" shapes.

The keyboard has 55 keys which give a similar variety of symbols to those on an ordinary typewriter. The keyboard is used for issuing directives to the computer.

5.4.3 The 1831 display and interrogating unit

This unit is similar in principle to the 1830 unit but has a small screen, a 15-key keyboard, and is capable of reproducing characters only.

The unit is intended for use where operators require to retrieve specific items from file storage devices in connection with such activities as banking, seat reservations, etc. The reference number of the file will be typed on the keyboard and the information displayed, in groups of up to 256 characters, on a 4in × 2in screen.

Several 1831 units may be connected to a standard interface channel via a control unit. Two types of control unit are available: one type has a 4096 word buffer store and the other type uses the special hesitation facility. The former can supply up to sixty four 1831 units and the latter up to 8, and in both cases different information can be displayed on all units.

5.4.4 The I.C.T. 1934 digital increment plotter

This unit consists of a Calcomp plotter unit, Model 563, 564, 565, or 566, mounted on a pedestal-type cabinet which contains power supplies and control electronics designed to work over the standard interface.

The paper on which plotting takes place is wound on two rollers and passes over a rotatable plotting drum. The plotting pen carriage is mounted over the drum on a traverse arm which enables it to move parallel to the drum axis. The plotting action is obtained by step motions of the drum, and/or the pen, permitting each step to be in any one of the eight directions listed.

Drum step forward	-X direction
Drum step backward	+X direction
Pen step left	+Y direction
Pen step right	-Y direction
Simultaneous drum and pen steps	four diagonal directions
Pen down	pen on paper
Pen up	pen off paper

Each operation is represented by a data character, and the sequence of operations is effected by the transfer of characters from the computer to the plotter.

The characteristics of each plotter in the series are summarised in the following table

Model Number	563	564	565	566	
Speed: X-axis	12,000	18,000	18,000	18,000	steps/min
Y-axis	12,000	18,000	18,000	18,000	steps/min
Pen	600	600	600	600	operations /min
Step size	0.01	0.005	0.01	0.005	inch
Paper width	31	31	12	12	inches
Plotting width	29½	29½	11	11	inches
Paper length	120	120	120	120	feet
Sprocket diameter	0.188	0.188	0.130	0.130	inch
Sprocket spacing	0.5	0.5	0.375	0.375	inch
Data transfer rate	200	300	300	300	ch/s

5.4.5 The I.C.T. 8500 document sorter/reader

This unit sorts and selects bank cheques and similar documents which are coded in magnetic ink and conform to the requirements established by the London Clearing Banks. The range of acceptable document sizes is from $6 \times 2\frac{3}{4}$ inches to $8\frac{3}{4} \times 4\frac{1}{2}$ inches. When the device is connected to a 1900 system, sorting and selection is controlled by the computer and data read from the documents may be edited, listed on a printer, stored on magnetic tape, etc, as required, under the control of a program.

Sorting takes place at 1200 documents per minute. The magnetic ink code is read and, according to instructions, the documents can be sorted on any characters position, to any one of 18 stackers, one of which is for documents classified as rejects.

5.4.6 The I.C.T. 1992 communications multiplexer

The I.C.T. 1992 communications multiplexer is a multi-channel control device permitting on-line data input and output between the central computer and up to 63 remote devices. These are connected to the multiplexer by ordinary telegraph, telephone, or internal multi-typewriter line facilities; or any combination of these.

The multiplexer is modular to cater for any number of lines from 2 to 63, and each line can be one-way (simplex) or non-simultaneous two-way (half duplex). Line termination units are necessary at both ends of each line. The speed of transmission of data is limited, for all practical purposes, not by the multiplexer but by the speed of the transmission line or central computer.

Each line input to the multiplexer contains a single character buffer and these are sequentially scanned, any characters present being transferred to an area in the computer's store which has been allocated to each terminal. Transfers take place on the hesitation principle and an interrupt is caused only when a complete message has

been received or transmitted, or the allotted store area filled or emptied. Then, subject to program priorities, the communications program deals with the message.

The I.S.O./A.S.A. 8 track code, used as the standard paper tape code throughout the 1900 system, is decoded or encoded in the multiplexer unit. Other codes are handled without modification and codes of greater than 8 bits are split up into 6 bit groups for transfer to or from the computer. The parity of all incoming messages, if present, is checked in the multiplexer on the line transmission unit.

5.4.7 The I.C.T. 1955 data exchange control

This unit will enable a 1900 series computer to communicate with another computer via a standard interface channel. Information is passed over the data exchange control in 4-character bursts. With the 1902 (6μs store) and 1904 (2μs store) the rates of transfer can be as high as 76,000 and 114,000 characters per second respectively. The speed is limited to that of the slower computer of a linked pair.

A use for the unit is to link a small computer to a large computer and use the small machine for the editing and validification of input and output data. This raises overall efficiency by reducing the load on the main computer and ensures that all input information is validified and presented in the most useable form.

CHAPTER 6

THE CENTRAL COMPUTER

6.1 General

The central computers of the 1904, 1905, and 1909 are very similar and they are shown in block form in Diagram 12. The content and relationship of the blocks is decribed in the following sections.

6.2 The arithmetic unit

At the centre of the arithmetic unit, shown in Diagram 25, is the mill. This is a 24-bit parallel adder which can carry out logical and arithmetic operations on the contents of the various registers gated onto its inputs, the K and L highways. Its output may be gated to the various registers either directly or with a single right or left shift. A mill operation is accomplished in 1 microsecond.

The sizes of the various registers and their interconnections are shown clearly on Diagram 25 and need little further explanation. Some of their uses are outlined as follows.

a) Register A is used as a working register. In addition to the normal output it has an inverse output to assist in the formation of negative numbers (see section 12.2.3).

b) Register B is used as a working register, and to send and receive operands to and from store. Parity is formed from the content of B when store accesses are made.

c) Register P is used to hold the instruction address (bits 0 to 14), special Executive modes (bits 15 to 21) the carry from multiple length operations (bit 22), and a record of any overflow that occurs during the execution of a program (bit 23).

In more complex instructions such as multiplication, register P is required as a working register, and its ability to shift operands internally, left or right, is used. When P is used as a working register, its normal contents are temporarily stored in store location 8 of the program.

d) The content of the datum and limit registers is self explanatory and the use made of the datum and limit is explained in section 6.10.

e) Register N contains various quantities as an instruction progresses. Initially it contains the instruction address and then the n-address part of the instruction. It also has the ability to count down in decrements of unity and is used as a counter in instructions such as multiply, shift, etc.

f) Register F, X and M are used to contain the function, accumulator address and modifier address of an instruction respectively. Register X has an additional output, X + 1: this is used to specify the second accumulator used in double length operations.

Peripherals are addressed via the BAD highway, and information is passed to and from peripherals on the BOUT and BIN highways respectively.

6.3 The LDG register

The LDG register can only be loaded by special Executive orders and consists of the datum and limit registers in the arithmetic unit, and the 4-bit G register in the central control unit.

The G register bits are numbered 0, 1, 3, and 4. Bits 3 and 4 are used to indicate which of the four possible programs are running in the machine and their decoded output is used to illuminate the appropriate indicator lamp on the console typewriter.

A program called Trace may be entered into the computer as an aid to the development of new programs. This program works in conjunction with Executive to provide monitoring facilities, and the monitoring facilities provided depend upon the monitoring mode set up in bits 0 and 1 of G. The four possible modes are:

00 no monitoring required,
01 monitor after each order obeyed,
10 monitor after each 123 order with X = 7,
11 monitor after each successful jump and after each 123 order with X = 7.

A steering tape, which is prepared by the programmer, is used in conjunction with the Trace program. The tape indicates the points at which the program to be tested requires monitoring, and the type of monitoring action required. According to the information on the steering tape Executive, in conjunction with Trace, loads the G register with the different monitor modes required as the program progresses.

The 123 order is normally a dummy instruction, but when given with X = 7, and 2 or 3 set in G 0 and 1, a monitoring interrupt is forced. 123, X = 7 orders may only be given by Trace. They are inserted by Trace, under the direction of the Trace steering tape, into the program to be monitored at the points where monitoring is required.

6.4 The microprogram sequence gates

The microprogram gates are responsible for sequencing the control bistables in central control. The microprogram sequence required is indicated either by the function register (which is in the arithmetic unit) or hesitation control.

The microprogram is broken down into sub units, each one of which is responsible for sequencing a particular order, a set of similar orders, or hesitation. The microprogram, via central control, defines the sequence of actions necessary in the arithmetic unit to carry out an instruction.

During the sequencing of an order or hesitation, the function code and the present setting of the central control bistables are used to set the microprogram gates and determine what has to be done next. For more complex orders such as division and shift orders, which contain several repetitive operations, additional bistables in central control and counters in the arithmetic unit, are available to the microprogram.

6.5 The central control unit

All operations and facilities in the arithmetic unit are controlled by signals originating in the central control unit. Central control consists mainly of a set of control bistables, each one of which in general, controls a specific sub-operation in the arithmetic unit. Usually, several of these bistables are set to control a complete arithmetic unit operation. Central control is also used to control core store operations and, on some occasions, the machine clock rate.

6.6 The computer's clock

The clock produces timed pulses which set control and data bistables in all parts of the computer. The clock normally has a p.r.f. of 1 mc/s but is slowed down for certain operations such as core store cycles, by central control. For maintenance purposes the clock can be operated manually on single shot, double shot or single instruction.

6.7 Hesitation control

A hesitation is an operation which causes the computer to hesitate for 6μs (or 16μs with the 6μs core store) over the program it is executing in order that a transfer of data may take place between the core store and a peripheral.

Hesitation control provides an interlock between peripherals, so that if several data transfer requests occur at the same time, the computer

can deal with them one by one. The microprogram hesitation sequencing varies with different types of peripheral and hesitation control indicates which type of peripheral is requiring attention and therefore, which sequence is to be entered.

Hesitation control also generates a predetermined core store address for each peripheral. This is the control word address and it contains information concerning the store area being used for the transfer.

6.8 Peripheral control units

Most peripherals used on the 1904, 5, and 9 whether designed for standard or special interface operation, each have a control unit: the exceptions to this rule are very fast peripherals which share control units.

Special control units generally consist of a word or character buffer, a set of control bistables and gates and, when appropriate, parity checking and character decoding facilities.

Standard interface control units are considerably simpler than special controls, but as the same problems of control and data handling are met with, this simplicity is off-set by having more stand-off logic at the peripheral itself.

A peripheral control unit can only be selected and told to begin operating by special orders of the Executive program. Once a peripheral control has been selected and activated it continues a transfer on the hesitations principle.

Peripheral control will cause hesitations whenever it has information in its word or character buffer ready to be stored in the core store. On an output operation, it causes a hesitation whenever a transfer has been completed from the input buffer to the peripheral. Another word or character is then transferred from the core store.

A peripheral control continues to cause hesitations until a transfer is completed, at which time it causes an interrupt. The interrupt causes Executive to take over and determine what to do next.

6.9 The core store

For those readers unfamiliar with core stores, a short account of the principles involved can be found in section 1.9.7. The I.C.T. 1904, 5 and 9 computers have conventional core stores, the sizes and speeds of which are summarised in the table in section 4.1 There are 26 planes in each store: 24 for information, 1 for parity, and 1 spare. The stores have three modes of operation which are:

a) read regenerate: in which information is read from the core store and regenerated for further use,

b) clear write: in which the selected store location is cleared during the read half cycle and information is written in during the write half cycle,

c) read pause write: in which information is read from the core store and different information is written back.

The store is fully self contained using its own power supplies, selection and drive logic, and the power on/off switching is automatically sequenced in a manner that retains stored information. It also has built-in testing facilities. These enable fixed patterns to be written sequentially right through the store, and then to be read back again in the same sequence, checking the read-out against the pattern generator. Patterns available are all '1's, all '0's, worst pattern and inverse worst pattern.

The temperature of the core store is thermostatically controlled in order to ensure that its working point does not vary due to changes in ambient temperature.

6.10 The datum and limit tester

A region of core store is allocated to each program by Executive and each time a program is entered, the datum and limit of the program's store area are entered into the datum and limit registers.

The values of datum and limit are multiples of 64 and the datum and limit register are each made up of nine bistables which hold the most significant nine digits of a 15-bit address. When a program other than Executive is running, each address is automatically checked against the datum and limit prior to being applied to store selection. This ensures that an address inside the programs allocated area is selected. X and M do not require checking since they lie in the range:-

$$D \leqslant X \text{ or } M < D + 8$$

This is inside the minimum area of 64 lines that can be occupied by a program.

6.11 Parity

All information going in to and out from the core store is momentarily present on the arithmetic unit's B register and, on both store input and output, the information is extracted from this register for parity forming purposes. The parity bit is stored in the 25th plane of the store, and on store output is used to set the parity bistable. The setting of this bistable is compared with the parity bit formed from the content of the B register and, if there is disagreement, the computer stops and the failing address is displayed on the engineers' panel.

6.12 The engineer's panel

The engineer's panel in the basic machine provides the engineer with very comprehensive monitoring and control of the whole computer.

On the panel are displayed:

1) the contents of all the arithmetic unit registers;

2) the contents of the arithmetic unit data transfer paths (known as highways);

3) the settings of the bistables in central control;

4) the operating modes of the various peripherals.

In order to facilitate fault finding, a switch is provided to run the computer on single shot, double shot or single instruction, and there is also a set of handkeys which may be used either to inject an order number or a whole instruction into the arithmetic unit.

Two more switches, FREEZE and RESET, are provided which, when operated, activate control waveforms causing the computer to read in a 128 character bootstrap via a predetermined peripheral (usually paper tape) into the first 32 core store locations. This bootstrap can then be used to read in Executive. FREEZE and RESET facilities are to be expanded on future machines in order to provide three primary input channels.

Margin checking facilities are also provided and take the form of a variable power supply and a variable clock rate.

On multi-cabinet machines and those with floating point units, an extra engineers' panel is provided to monitor the extra peripherals and registers.

6.13 Power supplies

The power supply for a two cabinet computer consists of an a.c. control unit controlling four rack mounted d.c. power units. The outputs of these are, +6V, -6V, -28V and a variable supply giving two outputs which are set to +6V and -6V. Each of these supplies has voltage and current monitoring. The core store has its own supply but is also controlled by the a.c. control unit.

The existing supplies have sufficient capacity to supply an extra cabinet but if more than three cabinets are used, additional supplies must be installed. The power consumption of a two cabinet computer is approximately 2.5 kVA.

CHAPTER 7

THE I.C.T. 1904, 5, and 9 WORDFORM

7.1 General

The basic word is 24 bits in length. For the purposes of programmers the bits are numbered from B0 to B23 starting at the most significant end, and for the purposes of engineers they are numbered from B23 to B0 starting at the most significant end. The pros and cons of these two systems are numerous, but as this manual is more likely to be read by engineers than by programmers, engineers' notation will be used.

The 24-bit word can have various interpretations according to the way in which it is to be used.

7.2 Four 6 -bit alpha-numeric characters.

The word is split into four characters as shown below

N_0	N_1	N_2	N_3
Bits 23 to 18	17 to 12	11 to 6	5 to 0

The notation used is 123.0 for character "0" of the word in line 123, 123.1 for character "1", and so on.

7.3 A signed fraction.

The word is regarded as having a binary point between B23 and B22. B23 is negative and B22 to B0 are positive, their values being as shown below.

B23 .	B22	B21	----------	B1	B0
Value -1	2^{-1}	2^{-2}	- - - - - - - - - -	2^{-22}	2^{-23}

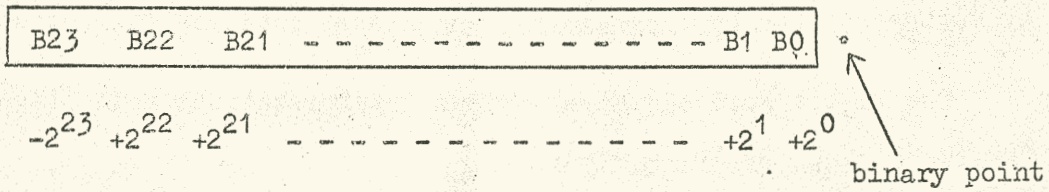
Binary point

Such a fraction lies in the range:-

$$-1 \leqslant \text{fraction} < 1$$

7.4 A signed integer

The word is regarded as having a binary point immediately to the right of the least significant digit. B23 is again negative and B22 to B0 positive, but different values are now assigned.



Such an integer lies in the range:

$$-2^{23} \leqslant \text{integer} < 2^{23} - 1$$

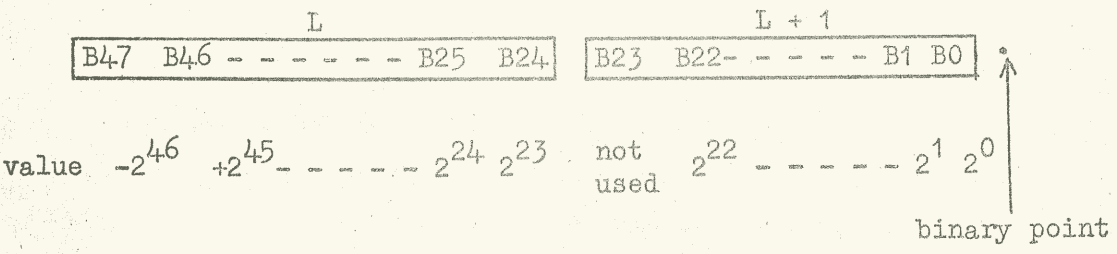
7.5 Multiple length working

7.5.1 General

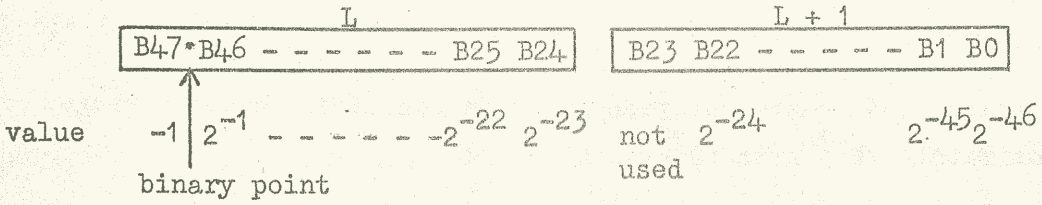
A single word can represent a number about as big as 10^{7} in the case of an integer and about as small as 10^{-7} in the case of a fraction. If a larger range is required multiple length working is available.

Two or more words in adjacent store locations are used to represent one number, extending the range of numbers to approximately 10^{14} to 10^{-14} in the case of double length working, 10^{21} to 10^{-21} in the case of treble length working, and so on. An example of a double length signed integer and a double length signed fraction is shown below.

7.5.2 A double length signed integer



7.5.3 A double length signed fraction

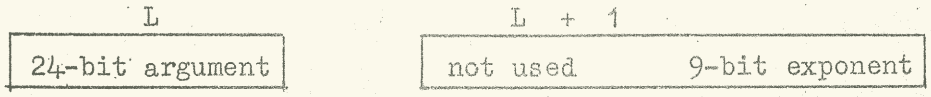


Additional instructions are used for the lesser significant words of multiple length numbers.

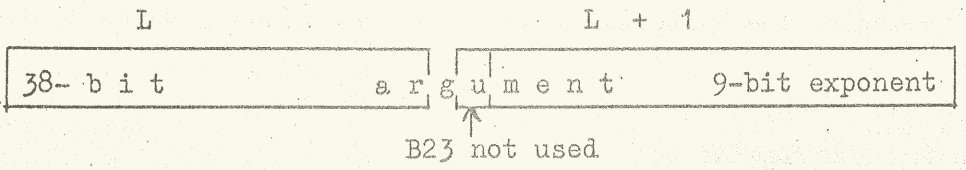
7.6 A floating point number

A number y is expressed as $y = R2^{e}$ where R is the argument
e is the exponent

Two store lines are used to hold a floating point number and the number is made up as shown below.



or if more accuracy is required.



The argument is stored as a standard number and so must lie in the ranges:-

$1 > R \geq \frac{1}{2}$ or 0 or $-\frac{1}{2} > R \geq -1$

The exponent store contains, not e , but e + 256, which is a positive integer lying in the range 0 to 511. The content of an exponent store is referred to as the characteristic. This is intended to avoid confusing e with e + 256. The constant 256 is automatically allowed for when floating point arithmetic is performed and when the exponent is -256 the content of the exponent store is zero. This condition is easily detected.

Floating point representation gives an equivalent decimal range of approximately 10^{-76} to 10^{76}.

7.7 An instruction

A normal instruction is made up as below:

X	F	M	N
3 bits	7 bits	2 bits	12 bits

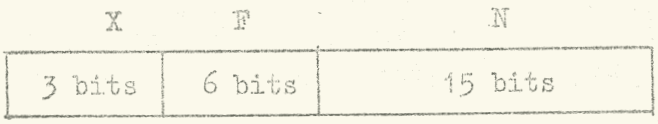
X specifies one of eight accumulators. These are the first eight store lines of any program and are used to store one of the operands used by the instruction.

F is the function and specifies which operation the instruction is to perform.

M is the address of accumulator 1, 2, or 3 whose content is to be used to modify the N address. If M = 0 no modifying takes place.

N is the address of the store line containing the other operand on which the instruction acts.

The other main type of instruction is the branch or jump instruction. This is made up as follows:-

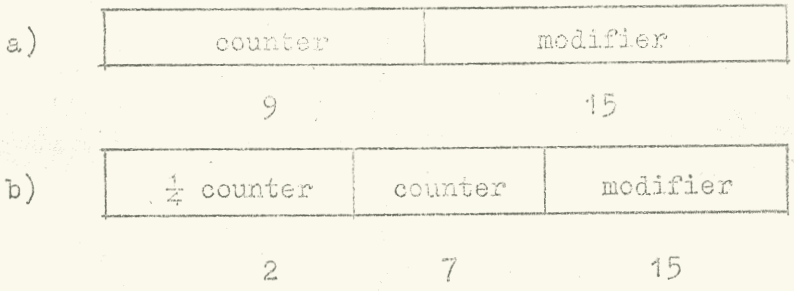


Here N is the address to which the branch is to be made.

There are many other types of instruction and these are all listed in the function code summary. The programmers handbook gives full details of the use to which the various instructions can be put.

7.8 A counter modifier

The counter modifier word configuration may be used to assist in the control of program loops or, as a control word, to assist in character and word at a time peripheral transfers. The counter modifier takes two forms and these are:



Whether the counter modifier is controlled by the appropriate instruction (in a program) or by the hesitation microprogram (in a peripheral transfer), the effect is the same: the counter decreases and the modifier increases.

In the case of a) the counter decreases by one and the modifier increases by the same amount. In the case of b), every time one is subtracted from the counter, one is added to the 2-bit ¼ counter. This overflows into the modifier-part so that four is subtracted from the counter for every one added to the modifier. The ¼ counter may thus refer to one of four characters to be packed, unpacked, etc., in

an address arrived at via the modifier, and the counter will count the characters as each is dealt with.

CHAPTER 8

SUMMARY OF THE FUNCTION CODES

8.1 General

Each function is given an octal (radix = 8) number and this corresponds to the actual binary configuration of the number inside the computer. For example, the internal representation of the 152 function is:

$$\underbrace{1}_{1}\ \underbrace{1\ 0\ 1}_{5}\ \underbrace{0\ 1\ 0}_{2}$$

it will be seen that $152_8 = 1101010_2$

The function code is made up so that similar functions are grouped consecutively, and such groups are referred to by the two most significant digits of the digit octal number assigned to the function. For example, functions 041 to 047 which comprise the multiplication and division instructions are referred to as group 4, and the 041 function would be defined as group 4, function 1.

8.2 The function code

8.2.1 General

The following few pages show the whole of the function code and section 8.4 contains a key to the nomenclature.

The speeds of operation refer to the 1904, 1905, and 1909, the 1909 having the 6μs core store. While the other machines in the series can use the same function code, the 1902 and 3 perform orders more slowly

and the 1906 and 7 more quickly.

The 13 to 17 group instructions are macro-instructions involving Executive sub-routines: the method of entry to these is explained in section 8.5. Exceptions to this rule are 13 group, functions 2 to 7 on machines with floating point units.

8.2.2 Ordinary instructions

Function	Description	Time (μsecs)* 6μs store	Time (μsecs)* 2μsec store	Notes
000	$x' = n + c$	18	7	Overflow may be set on exit but Carry is cleared
001	$x' = x + n + c$	18	7	
002	$x' = -n - c$	18	7	
003	$x' = x - n - c$	18	7	
004	$x' = n + c$	18	7	Overflow cannot be set. Sign of result always positive. Carry set if appropriate.
005	$x' = x + n + c$	18	7	
006	$x' = -n - c$	18	7	
007	$x' = x - n - c$	18	7	
010	$n' = x + c$	18	7	As 000-003 but with n and x interchanged
011	$n' = n + x + c$	18	7	
012	$n' = -x - c$	18	7	
013	$n' = n - x - c$	18	7	
014	$n' = x + c$	18	7	As 004-007 but with n and x interchanged
015	$n' = n + x + c$	18	7	
016	$n' = -x - c$	18	7	
017	$n' = n - x - c$	18	7	
020	$x' = x \,\&\, n$	18	7	Logical AND
021	$x' = x \vee n$	18	7	Logical INCLUSIVE OR
022	$x' = x \not\equiv n$	18	7	Logical EXCLUSIVE OR
023	Obey n as an instruction	7	3	
024	$x' = n_j$	18	7	Extract character

Function	Description	Time (μsecs)* 6μs store	2μsec store	Notes
025	$x' = n_e$	18	7	Extract exponent
026	Set C if $n \neq x$ or $c = 1$	18	7	Test equality
027	Set C if $n + x\ c > X$	18	7	Test equality
030	$n' = n \,\&\, x$	18	7	As 020-022, result in N
031	$n' = n \vee x$	18	7	
032	$n' = n \not\equiv x$	18	7	
033	$n' = 0$	18	7	Clear n
034	$n_j' = x_3$	18	7	Insert character (6 bits)
035	$n_e' = x_e$	18	7	Insert exponent (9 bits)
036	$n_a' = x_a$	18	7	Insert xa (12 bits)
037	$n_m' = x_m$	18	7	Insert x_m (15 bits)

*modifying increases these time by 6 μsecs or 2 μs respectively.

Function	Description	Time (μsecs)* 6μsec store	2 μsec store	Notes
040	$x:' = n.x$	67	40	Unrounded multiply
041	$x:' = n.x + 2^{-24}$	67	40	Rounded multiply
042	$x:' = n.x + x^*$	72	41	Semi-cumulative multiply
043	$x:' = 10.x: + n_j$	58	27	Decimal-binary conversion
044	$x^{*}' = x:/n$	76	45	Unrounded double length division (remainder to X)
045	$x^{*}' = x:/n + 2^{-24}$	79	48	Rounded double length division (remainder to X)
046	$x^{*}' = x^*/n$	71	44	Unrounded single length division (remainder to X)
047	$x' = 10.x:$ $nj' =$ Character	58	27	Binary-decimal conversion
050	Branch if $x = 0$	13	5	
052	Branch if $x \neq 0$	13	5	
054	Branch if $x \geqslant 0$	13	5	
056	Branch if $x < 0$	13	5	
060	Single word modify	13	5	Add 1 to modifier
062	Alternate word modify	13	5	Add 2 to modifier
064	Character modify	13	5	Add $\frac{1}{4}$ to modifier (subtract 1 from counter branch if counter non-zero)
070	Subroutine entry	15	8	Store ONR and V in x, clear V, branch to N
072	Subroutine exit	12	4	Branch to N + x, V reset as at entry unless set by subroutine
074	Branch on condition X	7	3	

Function	Description	Time (μsecs)* 6μsec store	2μsec store	Notes
	X = 0 unconditionally			
	X = 1 if V set			
	X = 2 if V set, clear V			
	X = 3 if V clear			
	X = 4 if V clear, clear V			
	X = 5 if C set			
	X = 6 if C clear			
	X = 7 if V clear, invert V			
100	$x' = N + c$	12	5	
101	$x' = x + N + c$	12	5	As 000-003 but N
102	$x' = -N - c$	12	5	used in place of n
103	$x' = x - N - c$	12	5	
104	$x' = N + c$	12	5	
105	$x' = x + N + c$	12	5	As 004-007 but N
106	$x' = -N - c$	12	5	used in place of n
107	$x' = x - N - c$	12	5	
110	Left Shift x, N_s places	18+NS	6 + N_s	N_t = 0, Cycle shift
111	Left Shift x:, N_s places	42+NS	15 + N_s	N_t = 1, Logic shift
112	Right Shift x, N_s places	18+NS	6 + N_s	N_t = 2, Arithmetic shift
113	Right Shift x:, N_s places	42+NS	15 + N_s	N_t = 3, Special shift
114	Normalise x	37+Sh	15 + Shift	
115	Normalise x:	42+Sh	16 + Shift	
120	$x' = x \,\&\, N$	12	5	As 020-022 but N
121	$x' = x \vee N$	12	5	used in place of n
122	$x' = x \neq N$	12	5	
123	No operation	7	3	Dummy instruction
124	$x_c' = N$, $x_m' = 0$	13	5	Set counter

Function	Description	Time (μsec)* 6μsec store	Time (μsec)* 2μsec store	Notes
125	Set Mode N	8	4	
126	Block Transfer	32+12N	13 + 4N	N words from address x to address x*
127	x' = Checksum	32+ 7N	14 + 3N	Sum N words from address x* ignoring overflow

8.2.3 Macro-instructions

Function		Description
130		convert fixed to floating point
131		convert floating point to fixed
132		a:' = a: + n: } if bit 0 of X is "1", don't round.
133		a:' = a: - n: } if bit 1 of X is "1", don't
134		a:' = a: . n: } normalise. If bit 2 of X is
135		a:' = a: / n: } "1", reverse positions of a: and n:
136		a:' = n: Bit 0 of X is "0".
		Clear a: Bit 0 of X is "1".
137		n:' = c: Bit 0 of X is "0".
		n:' = a: and clear a: } Bit 0 of X is "1".
150		Suspend me if my unit X of type N is busy
151		Release my unit X of type N
152		Disengage my unit X of type N
153		Place in my register 9 the control word of my unit X of type N
154		Read more program from my unit X of type N
155		Suspend and dump on my unit X of type N
156		Allocate to me a unit X of type N
157		Peripheral operation: operation defined in the control area N to N+3
160	X = 0	Suspend the program awaiting operator message and print out the message in the area defined by control N
	X = 1	As above without program suspension
	X = 2	Delete this program treating the message in the area defined by control word N as though input by typewriter
161	X = 0	Suspend this program awaiting operator message and type out N as two characters.

Function		Description
	X = 1	As X = 0 without program suspension
	X = 2	Delete this program, type out N as two characters
162		Suspend me if my sub-program X is active
163		Activate my sub-program X at instruction N
164		Suspend the sub-program awaiting activation by the main program and release main program if waiting
165	N = 0	Place binary date (days from 1st Jan 1900) in X
	N = 1	Place date in character form in X:
	N = 2	Place amount of core store allocated to this program in X

8.3 Executive orders

When Executive is entered the Executive mode bistable is set (further described in section 9.5), and certain instructions in the order code take on different meanings to those set out in the previous section. The instruction effected are the 125 and 170 to 173, and their new meanings are set out below.

125E	$P_{15-21}' = N_{0-6}$	read the contents of bits 0 to 6 of the N register to bits 15 to 21 of the P register.

The use of bits 15 to 21 of the P register when the computer is in Executive mode is as follows.

P_{0-14}	order number
P_{15}	set zero suppression on 047 instruction
P_{16-18}	not used
P_{19}	add datum to N
P_{20}	add datum to X except in groups 04, 11, 12, and 074 order
P_{21}	add datum to M
P_{22}	carry
P_{23}	overflow

170E	x' = SRn	read the contents of special register n to accumulator X.
171E	SRn' = x	read the contents of accumulator X to special register n.
172E	LDG' = n reset EXM bistable P' = datum + 8	exit from Executive
173E	LDG' = n	read the contents of store location N to registers L, D, and G.

Special registers (SRn) are numbered in the range 0 to 65. Special registers 0 to 63 are assigned to peripheral devices and contain control data, and special registers 64 and 65 are the two reasons for interrupt registers: their use is described in Chapter 9 but need not be considered for the time being.

8.4 Function code nomenclature

a refers to the content of the floating point accumulator.

C is the carry register and c is used in order 000 to 017 and 100 to 107. C is always left clear by any order unless that order sets C.

M is a modifier register (registers 1 to 3).

m is the content of register M.

N is a core store address or number

Nt and Ns comprise the most significant 2 bits and the least significant 10 bits respectively of the 12-bit address.

n is the content of register N.

n' = n + x means that the result of adding n and x replaces n, only n being altered.

n_j is any one of the four 6-bit characters of n,

ONR is the order number register (bits 0 to 14 of P),

S is the sign bit or the most significant bit.

V is the arithmetic unit overflow register. V remains set until cleared.

X is an accumulator (registers 0 to 7)

x is the content of register X.

xa is the least significant 12 bits of X.

xe (the signed floating point exponent) is the least significant 9 bits of the contents of X + 1.

xm is the least significant 15 bits of X.

x* is the content of X + 1.

x: is the content of X and X + 1.

8.5 Floating point hardware and software times

The use of a floating point hardware unit speeds up floating point operations considerably. This is shown in the following table comparing times by the hardware unit with times taken by the Executive.

	F.P. hardware	Executive
Addition	10 µsecs average	400 µsecs
Subtraction	10 " "	400 "
Multiplication	22 " "	435 "
Division	45 " "	450 "

The anove times are for machines with 2µs core stores.

CHAPTER 9

THE MICROPROGRAM

9.1 General

The microprogram is a hardware program responsible for the sequencing of machine orders and hesitations.

The sequencing of any machine order may be broken into three parts:

1) preparation of order,
2) execution of order,
3) end of order.

Any of the above may be broken into by a fourth sequencing, hesitations.

Many blocks of sequences are available to the microprogram, and these are shown on drawing LDP/1900/0092, now known as LS0 to LS11.

9.2 Conventions on drawing LDP/1900/0092

Drawing LDP/1900/0092 provides a convenient method of working through the microprograms of the various order groups and hesitations of the computer. Once an engineer is familiar with the method of reading the drawing, it is probable that he need refer only occasionally to the detailed explanation given in the Technical Manual. Each function, group of functions, or hesitations has a separate sequence block, and these are indicated in the boxed titles above sequence blocks.

Points of entry or exit from these blocks are located at the large circles which contain the name of the waveform which is activated to make such an entry or exit possible. For example, on completion of any order group ending at circled waveform ME0, the end of order sequence is entered

at the corresponding circled waveform.

The boxes within the blocks contain the name of a microprogram waveform (prefixed by M) and one or more central control waveforms. The microprogram waveform in any box is set up as a result of the central control waveforms, function codes etc., which were present previously, and it in turn primes the bistables which produce the central control waveforms shown in the box. These bistables are activated by the arrival of a clock pulse, and as there is normally a clock pulse every microsecond, it will be seen that any instruction is carried a step forward once every microsecond.

The square brackets within the sequence blocks indicate various actions and these are as set out below.

a) operations in which the microprogram waveforms shown result in information being strobed, at the clock time, into the register specified: e.g. highway Q to A register.

b) operations in which the microprogram waveform shown results in information being passed to a highway by forcing the control waveform mentioned: e.g. register A to L highway. The control waveform is forced in the sense that the bistable which produces the waveform is triggered at the collector rather than base so that no strobe pulse is required.

c) operations in which the microprogram waveform shown is responsible for allowing or inhibiting the action specified e.g. inhibit register B15-23 to K highway.

The small circles which lie on the lines connecting blocks, boxes, and brackets are NOR gates, and the conditions indicated have to be met before proceeding past them.

Where two or more short lines are drawn towards such a gate, all the conditions indicated have to be met. The small letter "v" indicates an "or" action where only one of the indicated conditions need be met:

in such a case, only one line will be drawn toward the gate.

A line drawn over any of the inputs to these gates indicates that that input must not be present: e.g. $\overline{G4}$.

9.3 Sequencing of orders and hesitations

The microprogram sequencing is shown in full on drawing LDP/1900/0092 and a simplified block diagram is shown on Diagram 13.

The starting point of any sequence is at the conclusion of end of order in the extreme left centre of LDP/1900/0092. The first decision to be made is whether or not to enter the hesitation sequence. If there is a hesitation request signal (HR) from hesitation control, the hesitation sequence is entered, and a word or character is transmitted to or from a peripheral. If there is no hesitation request signal, the next consideration is whether there is any reason to interrupt i.e. enter Executive. This decision is based on a signal from any of the possible reasons for interrupt.

If neither interrupt nor hesitation request signals are present, the computer may continue with the current program and enter preparation or order. Preparation of order reads the next instruction in the program from store and sets it up in the instruction registers.

Whereas some instructions require only two of the 24 bit registers in the arithmetic unit for working space, others require all three, and the points of exit from preparation or order from these two classes of instructions are at MEP1 and, assuming there is no fast hesitation request present, MEP2.

Entry to the order sequence are likewise at points MEP1 or MEP2, and which order sequence is entered will depend upon the function code of the present instruction.

Orders are divided into three types, and these are short, long, and

very long orders.

9.3.1 Short orders

Short orders can be accomplished at such a speed that they do not warrant hesitation sequence break points. These orders do not require three working registers in the arithmetic unit, and exit from preparation of order at MEP1. Upon completion of the order, short orders generally enter the end of order sequence at ME0, two exceptions being an illegal operation and an 023 instruction where end of order is entered at ME01. An example of a short order is an 001 instruction, there being no hesitation break point from start to finish.

9.3.2 Long orders

Long orders take longer to complete than the permissible waiting time for a fast peripheral and, as this is the case, a break point is provided part way through the sequence where a fast hesitation request may cause entry to hesitations.

These orders require the use of register P as a working register, and preparation of order stores the content of register P in the core store prior to exiting at MEP2 into appropriate sequence. Entry to end of order is generally at ME0 0, exceptions to this rule being in the case of the 113 and 111 instructions with a shift of zero places. In these two cases entry is at ME0.

Long orders are never interrupted by hesitation requests from slow peripherals.

An example of a long order is the 040 multiply instruction. With this order there is an opportunity for a fast hesitation immediately prior to exit from preparation of order, and further opportunities each time a circuit is made of the loop half way down the multiply sequence block.

9.3.3 Very long orders

Very long orders take more time to complete than the permissible waiting time of either fast or slow peripherals, and hesitation break points are provided for both types.

Very long orders leave the preparation of order at MEP2 and enter the end of order sequence at ME0 0.

An example of a very long order is the 126 block transfer instruction. As with the long order, there is an opportunity for a fast hesitation request prior to the exit from preparation of order, but at each circuit of the loop in the 126 instruction sequence block, there is a test for both fast and slow hesitation requests.

9.4 Hesitations

Due to the differing data transfer rates of various peripherals they are divided into two classes and the relevant hesitation requests are designated:

1) fast < 100 microseconds,
2) slow > 100 microseconds.

Hesitation requests are initiated at random times and may occur immediately after one of the longer types of instruction has started. This is why, in long and very long orders, there are hesitation break points spaced sufficiently close in time to accommodate any fast periph-erals' maximum waiting time.

The hesitation sequence block is located at the right hand side of drawing LDP/1900/0092 and has three points of entry:

1 MHES 0 - hesitation entry for fast or slow peripherals originating either between the finish and start of two successive orders or during the execution of very long orders 126 and 127.

2 MHES 10 - hesitation entry for fast peripherals originating during the execution of long orders,

3 MHES 11 - hesitation entry for fast peripherals originating at the end of preparation of order.

If a hesitation occurs during a long or very long order, the contents of the B and N registers are stored in the core store so that the order may continue from where it left off when the hesitation is completed. If a hesitation occurs just before the end of order sequence, no storing of operands is necessary.

9.5 Macro-instructions

The 13 to 17 group instructions are, with a few exceptions, macro-instructions which are sequenced by Executive sub-routines. The exceptions are the 13 group instructions 2 to 7 on machines fitted with a floating point unit.

The macro-instructions cover floating point operations, and peripheral and program control: in all cases they involve an entry to Executive at absolute store location 32, the entry being termed voluntary as it is the result of a programmed instruction.

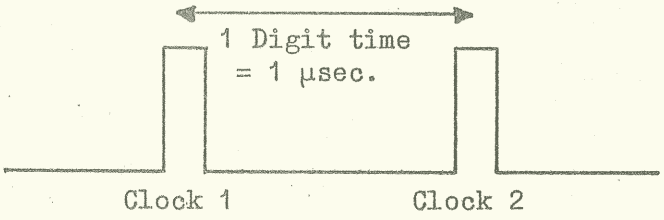
Exit from preparation of order is at MEP1, from where the group 13 to 17 sequence is entered. This sequence stores the modified N-part of the instruction in store location 1, and the instruction itself in absolute store location 2. Exit from the 13 to 17 group sequence is at ME0 from where the end of order sequence is entered. As a result of the end of order sequence, the next instruction number in the program is stored in datum + 8, the Executive mode bistable is set, and Executive is entered at absolute store location 32. Executive then performs the sub-routine indicated by the instruction stored in absolute address 2.

The Executive mode bistable is set for all the time that the computer is performing Executive routines. It results in certain instructions

taking on a different meaning (see section 8.3) and the suspension of datum addition and datum/limit testing except as specified by P19 to 21 (see section 8.3).

9.6 Timing

Timing throughout the computer is controlled by a one megacycle clock and the interval of time separating the leading edges of consecutive clock pulses is called the digit time.



Progress through each box of a sequence takes one digit time, and any operations associated with that box are accomplished during that time. The clock timing is modified slightly for core store operations.

9.7 A microprogram sequence - the 001 instructions.

The microprogram sequence shown and explained on Diagram 14 is intended to familiarise the engineer with the drawing conventions used, and the timing involved, in a typical sequence. The sequence is extracted from drawing LDP/1900/0092 and is that of a non-indexed 001 instruction.

The instruction being executed determines the route taken through the overall microprogram sequence. The 001 instruction is classified as group 0, function 1, and it will be seen that all the gates (small circles) in the route taken on diagram 14 respond to that group and function. A different instruction would activate other gates and a different sequence would result.

CHAPTER 10

CONTROL OF PERIPHERAL TRANSFERS

10.1 General

It was mentioned in Chapter 4 that the 1900 series of computers has a mixture of standard and special controls.

The parent computer, the I.C.T. 1900, originally had a different type of control box for each type of peripheral, and these were designed to each peripheral's specific requirements. On all but a few early machines the 1900 type control boxes are superseded by standard interface controls and only in a few instances are the original control boxes, now referred to as special controls, to be retained.

The reasons for the retention of the special controls vary. For instance, the high speed magnetic tape system (1974) uses a special control because it is too fast for the standard interface at the time of writing. However, when impending improvements in the hesitation system take place, it will be possible to build future 1974 tape systems to standard interface requirements. At the other end of the speed scale, the console typewriter uses a special control because it is used on every machine and gains no advantage from being made into a standard peripheral.

Where the central computer logic is concerned, both types of peripheral control are treated in much the same manner. However, peripherals with special controls use the control and data signals in the form in which they are produced by the central computer, whereas standard interface peripherals use them in a modified form which complies with the requirements for standard interface devices.

The control of peripheral transfers is described in this chapter as applied to both types of peripheral control, and Chapter 11 describes the way in which the computers' peripheral control signals are converted

into I.C.T. standard interface signals and the somewhat different Executive procedures which are necessary.

10.2 Control and data lines

The diagram of the arithmetic unit (Diagram 25) shows the BIN (peripheral in) highway gated onto the Q highway and the BOUT (peripheral out) highway gated off the K highway. These two highways are used to pass data and control information from each peripheral control box to the computer and from the computer to the peripheral control boxes respectively.

Gated off the N register is the BAD (peripheral address) highway which is used by Executive to address peripherals, each peripheral control box having a decoder which responds to one particular pattern on the BAD highway.

Other control lines between peripheral control boxes and the computer carry the signals HR (hesitation request), HS (hesitation select), BRST (peripheral reset), BSTOP (stop transfer) and PINT (peripheral interrupt) and these are used as follows.

a) HR - the hesitation request lines are used to carry hesitation request signals from the peripheral control boxes to hesitation control. Such a signal is transmitted when a peripheral is ready to send or receive a word or character.

b) HS - a hesitation select signal is returned to the requesting peripheral's control box as soon as the computer is able to handle the request. This enables the selected peripheral to transmit or receive data and also gates in BRST and BSTOP.

c) BRST - used in conjunction with HS to produce RESET inside the requesting peripheral's control box. This resets certain bistables inside the control box in order to prepare it for a further hesitation. In the case of output devices it is also used to gate data into the selected control box.

d) BSTOP - when the requisite number of words or characters have been transferred, the hesitation microprogram sends out the signal BSTOP which is gated into the selected peripheral by HS. BSTOP prevents the peripheral control box asking for further hesitations and results in an interrupt.

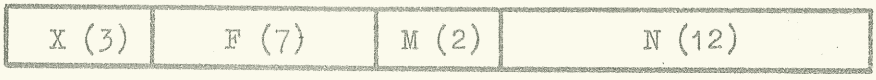
e) PINT - a peripheral control box sends out this signal when a peripheral transfer stops for any reason (section 10.5). The PINT lines from the various peripherals are mixed and any one becoming active causes an entry to Executive which then determines the cause of the interrupt.

10.3 Setting up a peripheral transfer

When a program is first entered, any necessary peripheral units which are not being used by other programs may be allocated to it by Executive. In some applications such as the printing of results, a peripheral may not be needed during the early parts of the program and it is more efficient to allocate the peripheral to the program immediately before it is required. The 156 order is provided for this purpose and allocates a peripheral of type N to a program, naming it unit X. The reverse is true of input peripherals which may only be required at the start of a program: the 151 order is available which removes the peripheral unit X of type N from the peripheral list of the program.

A programmer numbers his units of any particular type from zero, and Executive types out on the console typewriter the correspondence between the program's numbering and the numbers actually on the peripherals allocated to it by Executive.

When, during the course of a program, a peripheral action is required, a 157 order is given. This takes the form shown below.

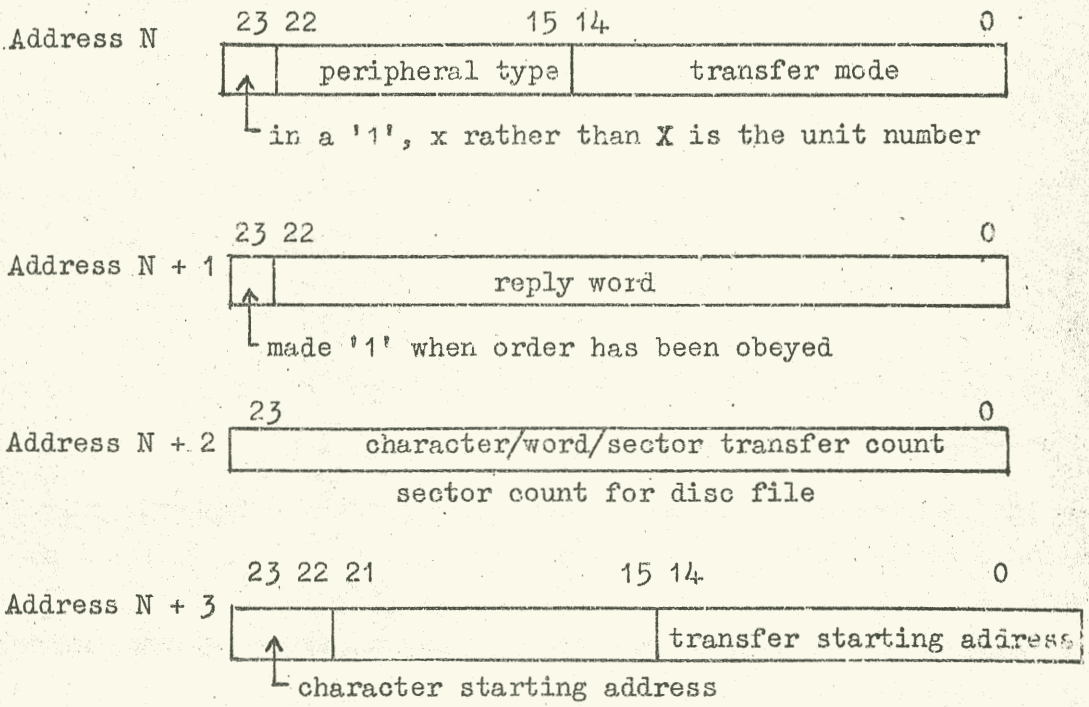


F is the function number and calls for a peripheral operation

X specifies the unit number of the peripheral

M modifies N in the normal manner

N specifies the first address of a 4-word control area in the core store: this contains the following information.



The control area must be set up by the programmer somewhere inside his program prior to the transfer order being given.

The 157 instruction is a macro-instruction and, if common with all other macro-instructions, the N-part is placed in absolute store address 1, the instruction in absolute store address 2, and the content of register P in the program's datum + 8. Executive is then entered at store location 32. This type of entry to Executive is referred to as a voluntary entry because it is the result of a programmed instruction.

Each peripheral control unit has a special register for holding control data and also a core store word permanently associated with it. This word is called the control word. Using the information available in the instruction and the 4-word control area, Executive addresses the peripheral

to be involved in the transfer via the BAD highway and ensures that it is not busy. If it is busy, 1 is subtracted from the content of datum + 8 (i.e. instruction number + 1 - 1) in order that the program may be re-entered at the transfer instruction at a later date. Another program is then entered.

If the peripheral is not busy, Executive picks up the information in N + 2 and N + 3 of the control area which defines the store transfer area, and places it in the peripheral's control word. Executive also picks up the transfer mode from word N of the control area, loads it into the special register in the control unit and starts the transfer. The program is then re-entered at the instruction number stored in datum + 8.

Peripherals such as magnetic tape and disc file require a larger control area than simpler peripherals as additional quantites such as the transfer area on the peripheral must be specified.

It should be noted that since the starting address of a transfer area, takes up 15 bits of a peripheral's control word, only 9 bits are left to specify the number of words, characters, or sectors to be transferred. This limits a peripheral transfer instruction to order a maximum transfer of 128 characters, or 512 words or sectors (see section 7.8 for counter modifier/control word format).

10.4 Hesitation requests

10.4.1 Hesitation select

When a peripheral control wishes to have access to the central computer in order to extract or insert a word or character, it sends a hesitation request to hesitation control in the central computer. Various peripherals may send in hesitation within a space of microseconds and, when this happens, they engage in a "race" in which fast peripherals are given precedence.

Diagram 15 shows the hesitation select system in simplified form. Hesitation requests enter at the top of the diagram, the fast requests being OR-gated together in one block and the slow requests in the other. The diagram is simplified in that in practice, each block accepts up to 10 hesitation request lines and there is normally one block for slow hesitation request lines and two blocks for fast ones.

When a hesitation request arises, the computer can be at any point in the execution of any microprogram sequence and, in the course of these sequences, there are fast and slow hesitation break points (as explained in section 9.4). It will be appreciated that fast peripherals require more immediate service than slow ones and, because of this, fast hesitation break points are more numerous. The lines CSLHR (slow hesitation request) and CFHR (fast hesitation request) inform the microprogram of the type of hesitation to be processed and an exit is made from the current instruction into the hesitation microprogram via a fast or slow break point as appropriate to the requesting peripheral.

The hesitation microprogram gives rise to a signal CHM (hesitation mode) and this primes NOR B immediately and NOR A after a slight delay (approximately 70 ns). If a fast hesitation request arises after a slow one, but prior to the entry to the hesitation microprogram, the fast hesitation OR output, CHRB 1 (hesitation request block 1), appears at the output of NOR B before the slow request output appears from NOR A. A fast peripheral may thus gain precedence over a slow one right up to the time of entry to the hesitation microprogram. The outputs of NOR B and NOR A are mutually exclusive, i.e. the one that outputs first inhibits the other.

Assuming that NOR A is activated, CSB 0 (select block 0) is applied to NOR 1,2, and 3. If more than one slow hesitation request is present, then more than one BHS (hesitation select) line will tend to be activated. The outputs of NORs 1,2, and 3 are, however, mutually exclusive and, of the ones with PHR applied at their input, the fastest (the one which, by chance, has the fastest switching time) will output and hold off the other

two. If only one PHR is present it will give rise to the corresponding BHS.

So, as both the block selection and the hesitation select selection are mutually exclusive actions, a number of hesitation requests all occuring within a short time will result in one peripheral being selected, and if any of the requesting peripherals are fast, a fast peripheral will be selected.

10.4.2 Identifying the selected peripheral

In order to proceed with a hesitation, the type (input, word, etc.) of the selected peripheral and the address of its core store control word must be available to the hesitation microprogram. These quantities are generated in hesitation control (top of Diagram 16) and the logic producing them is activated by BHS via a line, BEN.

The control words' addresses are generated by a group of 6 NOR elements giving up to 64 control words, and the address generated during a hesitation depends upon which of the NORs are driven by the BEN line associated with the peripheral taking part in the hesitation. The BEN line of each peripheral is also taken to the word and/or input, or card mixes, and the output of these mixes indicates the correct hesitation sequence for the peripheral.

In order to standardise installations, a table of standard allocations has been prepared which lists BEN waveforms (which are numbered) and the control addresses they produce, against peripherals. The number of BEN waveforms and associated control words considerably exceeds the number of peripherals likely to be found on the average installation, and when a further peripheral is attached to any machine, it is only necessary to connect it to the next available section of the selection logic and link the BHS line to the BEN line allocated to that peripheral.

For example, paper tape reader 1 is always associated with BEN 4 which, when activated, always results in control word address 68 being

generated. The control word's address is the BEN line number plus 64 as indicated on Diagram 16.

10.4.3 Transfer of a word or character

The hesitation microprogram controls the following four actions:

a) transfer of operands in use in the current instruction from registers B and N to the program's store locations 9 and 10.

b) extraction of the control word from the control word's address and its return in suitable modified form.

c) transfer of a word or character between peripheral and store.

d) return of operands in use in current instruction from program's store locations 9 and 10 to registers B and N.

The first and last of these actions only take place in the case of hesitations which start part of the way through a long order (section 9.4). They are by-passed in the case of hesitations which start between successive orders. They are shown as part of the hesitation microprogram on Diagram 16 but are not part of the actual transfer of data.

The hesitation microprogram initiates a read pause write store cycle and, as the control word's address has arrived on the SAD highway from hesitation control, the control word is read from store during the read pause. In the arithmetic unit, the address part of the control word is increased by 1 (or $\frac{1}{4}$ as explained in section 7.8) and the counter part decreased by 1. During the write part, the modified control word is returned to the store, a copy being retained in the B register. The address of the data to be transferred, which is arrived at by subtracting unity from the address part of the modified control word retained in the B register, is then applied to the store on the SAD highway, and the second read pause write store cycle is initiated by the hesitation microprogram.

During the second read pause write cycle the data is gated, under control of the hesitation microprogram, either from the store during the

read part and out to the peripheral via SIN and BOUT; or in from the peripheral, via BIN and SOUT to the store, during the write part. In both cases BRST is issued by the hesitation microprogram around the time of the transfer and resets the peripheral and control box in preparation for the next word or character transfer.

The content of registers 9 and 10 are then restored, if necessary, to B and N and, if no further hesitation request is awaiting attention, the general microprogram is re-entered.

If, during the hesitation, the counter becomes zero, indicating the last word or character in the transfer, the hesitation microprogram sends out the signal B STOP to the peripheral, and the peripheral causes an interrupt and entry to Executive. This is discussed further in the next section.

10.5 Ending a peripheral transfer

When a peripheral transfer stops, the peripheral control box sends out a signal PINT (peripheral interrupt). The current program is then interrupted, the next control address stored in datum + 8, and Executive entered. The entry made to Executive in the case of a peripheral interrupt is at store location 16 and is known as an involuntary entry because it is not a programmed entry.

In the computer are two master "reason for entry" registers known as special registers (SR,) 64 and 65. Any cause of involuntary entry to Executive gives rise to a one being present in SR64 or SR65 in a position corresponding to the reason for interrupt. Most of these reasons are peripheral incidents, and the PINT lines from the various slow peripheral control boxes are each allocated a position on SR64, while the PINT lines from the fast peripherals' control boxes are each allocated a position on SR65.

A peripheral transfer may cause interrupt either because a transfer has finished or because some fault condition has arisen. Either of

these reasons is stored in the peripheral control unit's special register (not to be confused with SRs 64 and 65).

Executive examine SRs 64 and 65 and identifies the control unit causing the interrupt. The special register associated with the peripheral control unit is then interrogated and Executive takes action appropriate to the reason for peripheral interrupt. For example, if the transfer has been successful, any program which has been suspended because it was awaiting the peripheral is reactivated and, if unsuccessful, the program making the transfer is suspended and a typewriter message outputted.

Having dealt with the peripheral interrupt, Executive takes a further look at SRs 64 and 65 and, if there are no further reasons for interrupt, enters the highest priority program available.

10.6 Reading SRs 64 and 65 and peripheral special registers

10.6.1 General

The 170 instruction has a different assignation when the computer is in Executive mode, and is used to order the transfer of the content of the special register specified by the n-part of the instruction into the accumulator specified by the x-part.

10.6.2 SRs 64 and 65

As already explained, the cause of an interrupt is ascertained by interrogating SRs 64 and 65. On entry to Executive, a 170 order is given with its N part equal to 65, and the content of SR65 is transferred to accumulator X via the BIN highway. If the cause for interrupt is indicated in SR65 (it could be in SR64), a "1" will be present in accumulator X. Executive determines the whereabouts of the "1", and hence the reason for interrupt, by doing a normalising count on the content of accumulator X. If there is no indication of interrupt in SR65, SR64 is treated in the same way.

SR64 is shown near the top of Diagram 17, and the 170 EXM microprogram order sequence produces the various waveforms which gate the content of SR64 onto the BIN highway and thence to the B register via the Q highway.

10.6.3 Peripheral special registers

If the cause for interrupt is a peripheral incident, then as a result of interrogating SRs 64 and 65, Executive loads the address of the interrupting peripheral control box into the n-part of a further 170 instruction. This instruction is given and results in the content of the special register of peripheral control box n being transferred to accumulator X. Executive may then examine the content of X and decide why the peripheral has stopped.

Diagram 18 shows the peripheral address n gated from the N register onto the BAD highway. The BAD decoder in peripheral control box n responds to this particular address and generates Executive Select (ES) which gates SRn onto the BIN highway and thence to the B register via the Q highway. Again all the gating waveforms are produced by the 170 EXM microprogram sequence.

10.7 Loading peripheral special registers

As already stated, when the programmer requires a peripheral transfer he must supply the computer with such details as the type of peripheral required and the transfer mode.

When a transfer request comes up in the program, and Executive has made sure that the peripheral is not still engaged with some previous transfer, Executive makes available the transfer mode in accumulator X and the peripheral number (say n) in register N. A 171 EXM instruction is then given.

When the computer is operating in Executive mode, the 171 instruction

orders the transfer of the content of the accumulator specified by the x-part of the instruction to the peripheral special register specified by the n-part of the instruction. This results in the mode being transferred to the selected peripheral's special register.

Diagram 19 shows the required peripheral special register being selected in just the same way as with the 170 EXM instruction. This time however the mode is read from accumulator X to the B register, and then via the K highway, to the BOUT highway and the special register of the peripheral control box whose BAD decoder is producing E.S. in response to n. All the gating waveforms are produced by the 171 EXM microprogram sequence.

10.8 The special hesitation facility

Some peripherals, such as unbuffered versions of broadside-fed card punches and line printers, require the same block of characters presenting to them several times in order that data can be output on them.

For example, each column of a standard 80-column, 12-row card represents a character. When such cards are punched out on a broadside-fed punch, they are punched row by row, and the same 80 characters must be used to activate the appropriate knives for each of the 12 rows punched.

If the card punch has a buffer store of 80 characters capacity then once the buffer has been loaded, punching of a card can proceed autonomously. With unbuffered punches however, the 80-character block must be transferred from the core store 12 times: once per row. In order to facilitate this without Executive action, each card punch has two core store control words associated with it.

The first control word contains the original character count and starting address, as copied from the program by Executive,

throughout the transfer; while the second is used by the hesitation microprogram to bring out the 80 characters in the normal manner. As each row is punched out, the second control word counts down to zero as usual, but prior to the next row being punched, the second control word is reset by having the content of the first copied into it.

The resetting of the second control word is accomplished by a special hesitation, and in the case of the card punch is carried out as follows.

When a card punch operation is requested in a program, Executive loads the first of the punch's control words with the character count (< 80) and starting address as usual. The card punch control unit has two hesitation requests and two hesitation select lines, and one of these pairs is used in conjunction with the special hesitations sequence.

At the start of a card punch operation the first hesitation request is sent to the computer on the special hesitation request line. This is answered by a special hesitation select. The select signal causes the hesitation microprogram to run through a special routine which copies the content of the first control word into the next store address up.

The special hesitation select signal also enables the card punch control unit to activate its ordinary hesitation request line, and there follows a series of ordinary hesitations in which the characters specified by the second control word are transferred to the punch.

When the control word counter reaches zero, BSTOP is sent to the peripheral as usual. As there are more rows yet to be punched, BSTOP does not have its usual effect of ending the transfer. Instead, a further special hesitation is initiated which results in the content of the first control word, which has remained unchanged, being re-copied into the next store address, and a repeat of the ordinary hesitations specified by the second control word.

When the last row has been punched, the BSTOP sent out by the computer has its normal effect and PINT goes to hesitation control as usual.

A peripheral using the special hesitation facility takes up considerably more of the computer's time than the equivalent peripheral fitted with a buffer store. An unbuffered card punch for example, needs 972 (80 x 12 ordinary plus 12 special hesitations) hesitations per card, as opposed to 80 for its buffered counterpart; and an unbuffered 64-character line printer with 120 characters per line needs 7744 (120 x 64 ordinary plus 64 special hesitations) hesitations per line, as opposed to 120 for its buffered counterpart. This increase in hesitations is often considered worth while compared with the expense of buffer storage.

CHAPTER 11

THE I.C.T. STANDARD INTERFACE

11.1 General

The I.C.T. standard interface has been introduced in order that as many peripherals as possible will have the same type of control and data signals, with the effect that almost any I.C.T. peripheral may be plugged into any I.C.T. computer with no modification.

In order to convert signals that pass between 1900 series computers and their peripherals into standard interface signals, the early 1900 control boxes are being replaced by standard interface controls. These controls are equipped for straight through transfers only and any decoding of control or data characters must take place at the peripherals themselves. Thus, while the standard interface controls at the computer are quite simple, the peripherals' stand-off logics are complex and capable of receiving, interpreting, and transmitting control information, and decoding or coding data.

As hesitation and peripheral control signals are dealt with one at a time, it is possible to multiplex some of them in a single piece of equipment called a central control unit (C.C.U.). Signals unique to each device are handled by individual control units (I.C.U.) and up to twenty five of these may be driven by one C.C.U. Any I.C.U. can be adopted to work in the special hesitation mode.

A C.C.U. and one I.C.U. occupy one card file and the configuration is known as a WGA 15 Main Control. Further I.C.U.s are then grouped together three to a card file to make up Triple Individual Control Units known as WGA 18 Interface Controls. The third control unit of each triple configuration will, by the addition of a few extra packages, operate a device with up to 64 channels.

The standard interface requirements are fully explained in the I.C.T. Standard Interface Specification, Drawing Number 1110020, published by Design Communication Group, but the following sections describe briefly the use of the standard interface in the 1904, 5 and 9 computers.

11.2 The scope of the interface

The types of peripherals which may be connected to the computers via a single interface connector are as follows:

a) single channel, single mechanism, e.g. a paper tape reader;

b) single channel, multimechanism, e.g. a group of tape decks working via one tape control unit;

c) multichannel, multimechanism, e.g. a number of typewriters working via a multiplexer;

d) data exchange, e.g. a device for transferring information between two computers;

e) single or multimechanism, multiaddress, e.g. a magnetic card file.

A single channel, multimechanism peripheral differs from a multichannel, multimechanism peripheral in that the former must complete a block transfer from one of its mechanisms before starting another, while the latter can carry out simultaneous transfers from all of its channels.

11.3 Lines crossing the standard interface

The lines that pass between each standard interface peripheral

and a 1900-range computer, and the correspondence between the various lines and existing computer signals, are shown on Diagram 22.

The interface lines basically comprise two 8-bit (plus parity) highways, Di (data into computer) and Do (data out from computer), along which data, control, and multimechanism addressing information are multiplexed. The seventh and eighth Di and Do lines are not normally used by the 1904, 5, and 9 machines. The type of information present on these highways at any one time is defined by the states of the A, C, No, and Ni lines.

The A line is used to address the peripheral, and if the A signal is absent, all other signals are ignored. With only the A signal present, information is interpreted as data, and with A and C present, as a control character. No is used with single or multichannel, multimechanism peripherals in connection with control (C="1") and data (C="0") identifiers on the Do and Di highways respectively, and Ni is used with multichannel, multimechanism peripherals to indicate that a data identifier is available.

The direction of a transfer is indicated by the state of the J line, which is a "1" for inward transfers and a "0" for outward ones. The line F, when a "1", indicates that the peripheral obeys faster-than-normal timing rules. The specification for such peripherals is not yet set out, and the F lines from all current peripherals are terminated by resistors at their I.C.U.s.

The uses of the other lines, and of those mentioned above, will become more apparent in sections 11.4 and 11.5 of this chapter.

Physically, each line is a twin twisted pair: a current of 25mA in one leg represents a "1", and in the other, a "0". A common return path is provided for all the signals associated with a particular peripheral. The lines are terminated at each end by a 75-way connector and form the only connection between the computer and peripheral.

11.4 Transfer of control information over the interface

11.4.1 General

It has already been shown in Chapter 10, sections 10.6 and 10.7, how the 171 and 170 EXM orders are used to transfer control information to and from peripheral special registers. When a standard interface peripheral causes an interrupt or is required to perform some action, a 6-bit control code of the type "send your status" or "perform this action" is sent to the peripheral by Executive.

The peripheral is equipped to decode such control codes and also to send back an appropriate response to the computer. In the former case for example, the peripheral replies with an indication of its status (discussed further in section 11.7), while in the latter, the reply would indicate to Executive whether or not the peripheral was available to carry out the action required of it.

All transfers of control information to standard interface peripherals result in a response, and in order to simplify Executive and perform control operations in as short a time as possible, the 174 EXM order has been introduced. The 174 EXM order microprogram sequence is split into four parts: preparation of order, read x to SRn, write SRn to x, and end of order. Transfer of x to SRn and vice-versa is accomplished by utilising the appropriate parts of the 171 and 170 order microprograms.

The N-part of the 174 EXM order is used to address the peripheral via the BAD highway, and the X-part specifies the accumulator from which the control code is read and to which the response is written.

11.4.2 Control of single channel, single mechanism peripherals

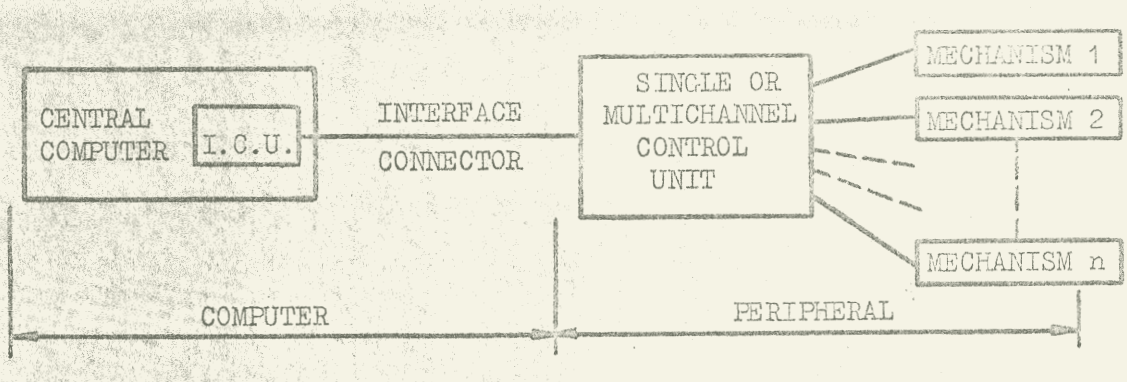
The N-part of the 174 EXM order is placed on the BAD highway as usual and activates the appropriate BAD decoder which, in the case of a standard interface peripheral, is located in its I.C.U. (see Diagram

23). The BAD decoder produces the signal ES from which the signals A and C are derived. ES also gates the control code, transferred from accumulator X by the 171 -part of the 174 EXM order, onto the Do register and hence, to the peripheral. As A and C are active, the peripheral treats the content of Do as a control code and places an appropriate response on Di. The 170-part of the 174 order, gates the response on through to accumulator X.

ITS is produced from IC at the time that the response is written into the core store. ITS produces T at the I.C.U., and the effect of T depends upon the control code and the status of the peripheral. If the control code is of the mode setting type (read, write, etc.), then provided that the peripheral is not otherwise engaged, T allows it to take up the required mode. Alternatively, with a control code of the type "send your status", provided that the status is in the status group asked for, T stops the peripheral interrupting and it enters the set 2 or unset state as explained in section 11.7.

11.4.3 Control of single or multichannel, multimechanism peripherals.

The connection of a single channel, multimechanism peripheral or multichannel, multimechanism peripheral to the computer is shown below in block form.



It is necessary with such a peripheral, to first identify the mechanism to which any command is to be given. This is accomplished by issuing two 174 EXM orders: the first order identifies the mechanism and the second issues a control code to the selected mechanism.

When the A, C, and No lines of a single or multichannel multimechanism peripheral are active, the information on Do at that time is interpreted as a control identifier, and to this end, the signal No is derived from BOUT 8 and ES (see Diagram 23). The accumulator specified by the X-part of the first 174 EXM order contains the control identifier and a "1" in the bit 8 position, while the accumulator specified by the second 174 EXM order contains the control code.

As a result of the first 174 EXM order the peripheral control unit has its A, C, and No lines activated and so "switches" to the mechanism specified on Do. The control code is then fed to the specified mechanism by the second 174 EXM order.

11.4.4 Control of multiaddress, single or multimechanism peripherals

With this type of peripheral the address within the mechanism is selected by sending a control code of a type "the following N characters are the address", followed by an appropriate number of characters which constitute the address. This information is supplied by 174 EXM orders and is accompanied by C activation. When the address has been supplied and selected within the mechanism, the transfer of data takes place as described in section 11.5.

In the case of multiaddress peripherals with more than one mechanism (e.g. a multitransport exchangeable disc file), the appropriate mechanism is first selected as described in the previous section.

11.4.5 T Suppression

When BOUT 9 is activated in conjunction with a 174 EXM order, the T strobe is suppressed (see Diagram 23). The facility is not yet used in the 1904, 5, or 9 computers but is included for future system development.

11.5 Transfer of data over the standard interface

11.5.1 General

Data is transferred between the computer and standard interface peripherals on the same highways, Di and Do, as is control information. In the case of data transfers however, the C line is not activated and the A line signal is derived from HS rather than ES. Diagram 24 shows the data and control paths for both single and multimechanism data transfers.

11.5.2 Single mechanism data transfers

When a standard interface peripheral requires a data transfer it activates its R and J lines. The R line signal is converted by the I.C.U. into a normal hesitation request, while activation of the J line indicates an inward transfer with J a "1", and an outward transfer with J a "0", so effecting the appropriate hesitation microprogram sequence.

The resulting HS (hesitation select) is applied to the peripheral's I.C.U. where it activates the A line, so selecting the peripheral for a character transfer. HS also activates the appropriate BEN line which generates the control word address in the normal way.

On an input operation, A results in the data character appearing on Di. Di highways from all standard interface peripherals go to the C.C.U., and are gated from there onto BIN by a derivitive of HS. The

character on BIN is written into the core store during the second read pause write cycle of the hesitation microprogram.

ITS (interface timing strobe) is produced off the trailing edge of BRST. ITS results in the strobe T, being sent out to all peripherals, only the selected one responding to it. T resets the peripheral resulting in removal of the character from Di.

With an output operation, the data character is read from store during the read pause part of the second read pause write section of the hesitation microprogram, and placed on BOUT.

T, derived from BRST, strobes the data on Do into the peripheral. T, however, is generated after the time that the outgoing character is present on BOUT, and to overcome this difficulty, the content of BOUT is temporarily stored in the Do register.

When A becomes active, the signal N is removed unless another character can be put on Di or accepted from Do in less than 2μs after the leading edge of T. The latter condition will only apply to burst mode peripherals described in section 11.5.5.

11.5.3 Multichannel, multimechanism data transfers

A multichannel, multimechanism peripheral is, as far as the hesitation system is concerned, a single device. Only one hesitation request line connects it to the hesitation selection system, and only one hesitation select line returns it to the peripheral's I.C.U. However, with such a peripheral several of its mechanisms may be involved in block transfers simultaneously, and to ensure that data is transferred to or from the appropriate buffer area in the core store, a data identifier is issued with each character transferred.

As already mentioned, the third I.C.U. of each Triple Individual Control Unit (WGA18) may be fitted out to accommodate a multichannel

peripheral, and the dotted parts of Diagram 24 represent, in simplified form, the necessary additions.

When a multichannel, multimechanism peripheral has a character for the computer, it activates its R, J, and Ni lines (see Diagram 24). R and J would normally activate the appropriate hesitation request line to the hesitations system immediately, but when Ni is active the request is delayed by about 2μs.

During the delay period the signals A and No result from Ni, and the peripheral, on receiving them, places the data identifier appropriate to the transferring mechanism on Di from where it is gated through to the data identifier register. The HR signal is then sent to hesitation control and there follows a normal hesitation sequence - with one difference. The HS line for the multimechanism peripheral is not connected to a BEN line. Instead, it is used to gate the content of the data identifier register to the control address gates, where the hesitation microprogram uses it to address the mechanism's control word, in the core store, in the normal manner.

11.5.4 Single channel, multimechanism data transfer

As block transfer from the mechanisms of a single channel, multimechanism peripheral are mutually exclusive, then provided that the contents of the control word are changed to suit each block transfer a single control word can be used for all of the mechanisms of such a peripheral.

So, while it is necessary to use a control identifier when initiating a transfer from a single channel, multimechanism peripheral, the use of data identifiers is unnecessary, and the control word address is generated via the usual HS/BEN line system.

To this end, each I.C.U. has an No line (see Diagram 23), making possible the control of single channel multimechanisms via any I.C.U.

11.5.5 Multicharacter transfers

Multicharacter transfers enable standard interface peripherals to transfer four data characters at a time, thus transferring a 24-bit word as four 6-bit characters in rapid succession. Since the transfer time for four characters is reduced from 24μs to 13μs and 15μs (24μs and 22μs with 6μs store) for output and input devices respectively, almost 50% is saved on hesitation time. This facility may be used, to obvious advantage, with fast peripherals.

In addition to the word, input, and card mixes mentioned in the previous chapter, a multicharacter mix has been introduced. Due to the multicharacter mix output, the control word of a multicharacter device is updated by the first part of the hesitation microprogram as if it was a word device, and then the hesitation microprogram branches into a character transfer loop sequence not previously mentioned.

Four circuits of the character transfer loop are made, resulting in four character transfers to or from (input or output) successive character positions of the B register, and four BRST signals. Each BRST produces a T strobe which, in conjunction with A, which is active for the whole of the four character transfer, effects the necessary action at the peripheral.

NOTE: On installations using many fast standard interface peripherals, multicharacter interface control units (WGA 16) will be available. These will each contain a word buffer into or from which four characters may be quickly transferred over the interface. The contents of the word buffer will be transferred to or from the control units by an ordinary word hesitation. Information will be supplied on the WGA 16 controls at a later date.

11.5.6 End of transfer

When the control word counter reaches zero BSTOP is sent out by the hesitation microprogram and IL is generated in the C.C.U. (see Diagram 24). The effect of L at the selected peripheral is to set it to the "terminated" status and cause a B line interrupt. This will be followed up by "send status" control codes from Executive, as described in sections 11.4.1 and 11.7.

11.6 Special hesitations

11.6.1 General

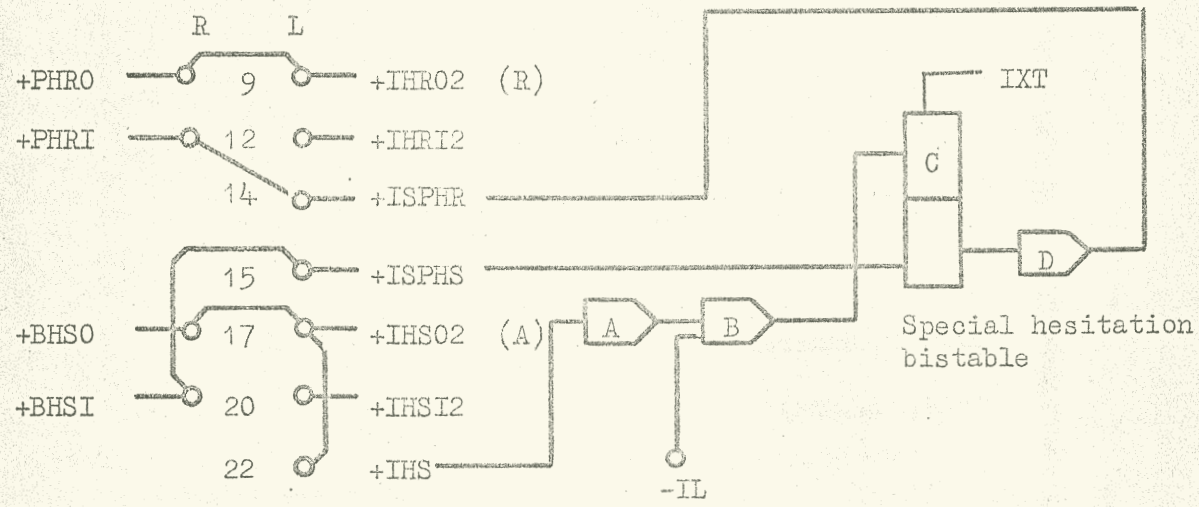
An I.C.U. may be made to operate in the special hesitation mode merely by changing a few wires on a link board. No extra plug-in cards are necessary since the small amount of additional logic involved is already incorporated elsewhere in the card file. There is one such link board per Main Control Unit and one per Triple Individual Control Unit: this allows one I.C.U. per card file to be operated in the special hesitations mode.

11.6.2 Link board connections

The link boards are shown on logic Diagrams 15_6 and 18_{10}, and with each I.C.U. operating in normal mode, the links 8 to 13 and 16 to 21 are made straight across with no additional links elsewhere.

When a peripheral requiring the special hesitation facility is connected to an I.C.U., its HRI and HRO lines are no longer used to indicate the direction of the requested transfer. Instead, by suitably re-wiring the link board, one is used for ordinary and the other for special hesitation requests. (In the case of input/output peripherals running in the special hesitation mode, e.g. an 1831 display and interrogating unit, the input and output sections will be treated as

two separate peripherals). Take, for example, an unbuffered I.C.T. 1920 card punch in use on I.C.U. 2 of a WGA 18 configuration. The links associated with I.C.U. 2 are then arranged as shown below.



Extract from Logic Diagram 18_{10}

11.6.3 Operation

Each time the card punch requires a character it activates its R line which results in the signal +IHR0 2 and, via link 9 (in the above figure), the hesitation request signal +PHR0 +PHR0 is answered by the select signal, +BHS0, and this is directed, via link 17, to the card punch I.C.U. There it results in activation of the A line and a character transfer to the punch. It should be noted that each IHS0 2 signal primes NOR B.

When the control word counter reaches zero, i.e. when a row has been punched, BSTOP is produced by the hesitation microprogram. The resulting signal, IL, enables NOR B, and bistable C is set by the next clock pulse.

The + PHRI line is activated by bistable C and the resulting select signal, BHSI, resets the bistable. The + BHSI signal, in this case, is applied to the hesitation system so as to cause entry to the special hesitation branch of the hesitation microprogram, rather than a data transfer. The sequence involved was discussed in Chapter 10, section 10.8 but briefly, results in the card punch's control word being reset so that the same block of data may be re-read from store for the next row of the card.

When the card punch is ready, a further set of hesitations take place again using + PHRO and + BHSO as request and select lines until, all in all, the 80 (or less) characters have been transferred to the punch 12 times: once per row.

11.7 Peripheral status

11.7.1 General

In Chapter 10 it was explained how, as the result of a 170 EXM order, a peripheral's status could be read from the peripheral's special register to the computer via the 24-bit BIN highway. In the cases of standard interface peripherals their special registers are contained at the peripherals, and the peripherals' status must be transmitted via the 8-bit Di highway. As many peripherals' status stretch to more than eight bits, a system has been evolved whereby they are transmitted in more than one step.

11.7.2 Q and P status

The various status are represented by 6-bit patterns: each bit represents a particular status. Normal causes of interrupt, such as end of transfer, make up a 6-bit group called Q status, and abnormal causes such as tape low, parity, etc., make up further 6-bit groups called P1, P2, etc., status. How many status groups there are depends

upon the complexity of a peripheral, but a line printer for example, has only Q and P1 status groups.

Executive is so written that when a peripheral causes an interrupt its Q, P1, P2, etc., status are examined in that order. The most significant bit of each group is the OR of all the status conditions in the following groups, and when active, results in Executive interrogating the next group - and so on.

There are two types of status condition: continuing and transitory. Transitory status are those which can be dealt with in a short time by Executive. An example of this type is an "end of transfer", for with this condition, the peripheral is available for further work almost immediately. Continuing status are those which involve a peripheral in operator action. Examples are tape low and parity failure; in both of these cases transfer of data ceases until appropriate action is taken.

When due to an incident, a peripheral's status changes, it is said to go from the unset to the set 1 state. This is an interrupting condition and Executive interrogates the peripheral in order to establish the reason for interrupt. If the reason is transitory, then after appropriate Executive action the peripheral is put back in the unset state where it is available for further work. If, however, the reason for interrupt is continuing, the peripheral is put in the set 2 state.

In the set 2 state the peripheral no longer interrupts. The condition originally causing the interrupt is, however, still present, and the peripheral awaits operator attention. For instance, in the case of tape low, the set 2 state would be "disengaged", and the peripheral would remain in this state until the operator loaded a new roll of paper into the magazine.

When the peripheral's needs have been attended to, it is returned to the unset state where it is available for further use.

11.7.3 Direct status response

To avoid having to ascertain the status of a peripheral before issuing it with operational control code (such as read, rewind, etc.), a limited indication of the status of a peripheral is available in direct response to such a code. There are three such responses and these are:

1) inoperable - peripheral is unable to operate until the operator or engineer has removed this condition,

2) rejected - peripheral is busy but can accept the control code later,

3) accepted - peripheral will accept control code and will initiate the operation.

In the third case the peripheral proceeds with the transfer and the program requesting the transfer is re-entered. In the second and first cases the transfer cannot take place at the time of asking, and Executive takes action appropriate to the direct response.

CHAPTER 12

ARITHMETIC PROCESSES

12.1 General

This chapter sets out to explain the mechanism of the arithmetic processes in the I.C.T. 1900/4/5/9 computers. This is done at a general level, no attempt being made to bring the detailed logic of the computer into the explanations.

A block diagram of the arithmetic unit is shown in Diagram 25 and the mill or adder at the top of the diagram is used in all arithmetic processes. The uses to which the various registers may be put during arithmetic instructions has already been discussed in section 6.2 and this will be further clarified as the various processes are explained in the following sections. Constant reference to Diagram 25 will prove useful when following operands around the registers and highways of the arithmetic unit.

12.2 Addition and subtraction

12.2.1 General

There are two sets of instructions for addition and subtraction, one set resulting in overflow being registered if the answer exceeds the capacity of the mill and the other set resulting in carry being registered.

The first set are Functions 0 to 3 of Groups 0, 1 and 10, and they are used for single length working and for the most significant word of a multiple length operand. The second set are Functions 4 to 7 of Groups 0, 1 and 10, and are used for all but the most significant word of multiple length operands.

If the basic 24-bit word does not provide sufficient places, other words can be added. The process of arithmetic using multiple length operands is to first add the least significant words of the two operands, then the second least significant, and so on. The necessary carries between sections of a multiple addition are automatically stored in bit 22 of the P register from one operation to the next. These carries are only generated when instructions resulting in carry are used.

When overflow is set, the fact is recorded in Bit 23 of the P register, and only certain instructions can clear it. Although a program's operands overflow, the computer carries on normally and it is up to a programmer to check the state of overflow periodically. He can do this at any time by the use of appropriate instructions.

12.2.2 Addition

In all types of addition one of the two operands is placed in register A. The other operand is brought out of store and placed on the K highway coincident with the content of register A being placed on the L highway. The mill output is then routed to the B register and into store.

It is arranged that the address of the second operand read from store is the address to which the answer is to be written. The operand is read out on a store read pause operation and the answer written back on the store write operation.

12.2.3 Subtraction

To arrive at the negative value of a signed binary number, the number can be inverted and one added to it.

Example

$$
\begin{array}{lll}
 & 010110_2 & = \quad 22_{10} \\
\text{Inverting} & 101001 & \\
\text{Adding "1"} & \underline{\quad\quad 1} \quad + & \\
 & 101010_2 & = \quad (-32 + 8 + 2)_{10} \\
 & & = \quad -22_{10}
\end{array}
$$

This technique is used in the computer for subtraction.

In subtraction, as in addition, the A register is used to hold one of the operands. The A register has both a normal and an inverse output and to subtract a from b, a is introduced to the A register and issued to the mill in inverse form. INV a + b + 1 is applied to the mill on the L and K highways and the least significant carry input.

In a multiple length subtraction there may be a "borrow" in the carry register. The next partial answer to be formed must, as a result of this, be made one less than it would have been had the borrow not been present. This is accomplished simply by not adding "1" to INV A.

12.2.4 Overflow

The mill has 24 adders in order that it can add together two 24-bit signed numbers. In addition to this the mill has a 25th stage which is located in the central control unit. This is called Mill 24 and, in conjunction with the sign bit of the mills output, it can be used to detect overflow.

The following two additions may clarify the technique used.

	Mill 24	Mill				
bits	24	23	22	21	20	
		0	1	0	0	positive input
		0	1	0	1	positive input
	0	1	0	0	1	apparent negative answer
		1	0	0	1	negative input
		1	1	0	0	negative input
	1	0	1	0	1	apparent positive answer

In both the above cases the answers to the additions are too big for the mill to handle, and the carry from bit 22 has altered the sign bit. Mill 24 however always shows the correct sign and for this reason is known as the true sign bit.

When the true sign bit differs from the actual sign bit (B23) overflow has occurred and is detected by a comparison of these two digits in the control unit.

12.3 Multiplication

12.3.1 General

In multiplication 24 partial products are formed by multiplying together the multiplicand and each separate digit of the multiplier. Taking into account their relative significances, the partial products are cumulatively added to form a running total as the operation proceeds.

With a negative multiplier, the partial product formed by multiplying the multiplier by the multiplicand's sign bit is negative, and is subtracted from the sum of the less significant partial products.

The instructions used for multiplication are the 040, 041, and 042 instructions.

The 040 instruction multiplies the content of N by the content of X and places the double length product in X and X + 1.

The 041 instruction multiplies the content of N by the content of X and produces a double answer to which a "1" is added having the significance of bit 22 of X + 1. This causes a carry if necessary, and a single length rounded answer is available in X.

The 042 instruction multiplies the content of N by the content of X, adds the content of X + 1 to the least significant half of the product, and makes the answer available in X and X + 1.

12.3.2 Multiplication overflow

Overflow can only occur when the fraction, -1, is squared.

12.3.3 Operand movements in the arithmetic unit

In each of the multiply instructions the process of multiplying is identical, and the following example, though simplified by the use of 4-bit rather than 24-bit operands, illustrates the movement of operands in the arithmetic unit during a multiply instruction.

Starting conditions: n (multiplicand) is fed to the A register,
x (multiplier) is fed to the P register,
4 (normally 24) is fed to the N register,
the B register is cleared.

The number 4 (24 in practice) placed in the N register is used to count the number of partial products formed.

Take the multiplication $0101_2 \times 0010_2$ (i.e. $5_{10} \times 2_{10} = 10_{10}$)

1. P l.s. (least significant digit of P) recorded and P shifted one place right. Mill l.s. transferred to P m.s. (most significant). Mill transferred direct to B register.

2. Previous P l.s. examined and cleared. As it was a "1", A is added to B (i.e. this partial product is 1 x 0010). P l.s. recorded and P shifted one place right. Mill l.s. transferred to P m.s. Mill output shifted one place right into B register.

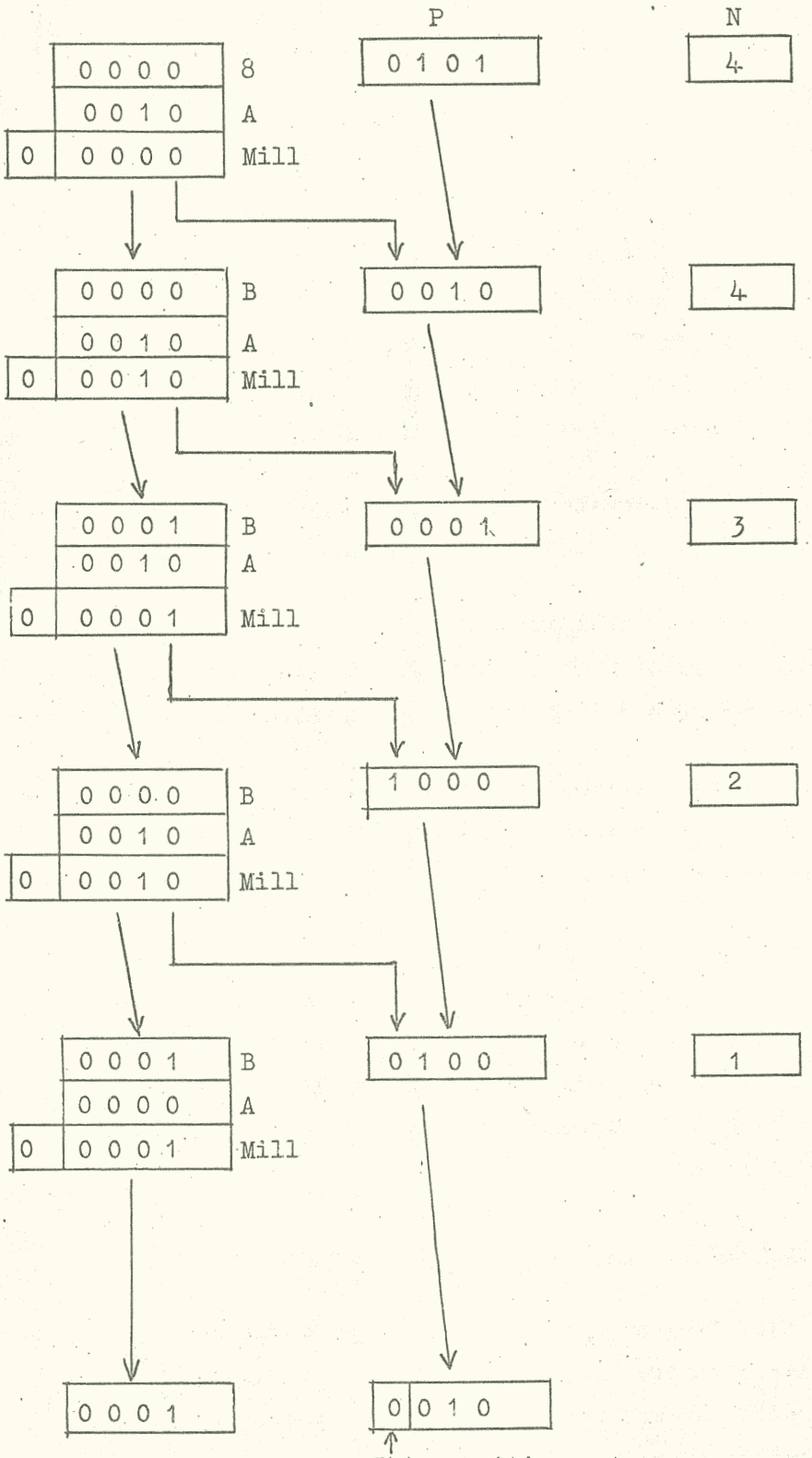
3. Previous P l.s. examined and cleared. As it was a "0", A is not added to B (i.e. this partial product is zero). P l.s. recorded and P shifted one place right. Mill l.s. transferred to P m.s. Mill output shift one place right into B register and "1" subtracted from content of N.

4. Previous P l.s. examined and cleared. As it was a "1", A is added to B (this partial product is 1 x 0010). P l.s. recorded and P shifted one place right. Mill l.s. transferred to P m.s. Mill output shifted one place right into B register and "1" subtracted from content of N.

5. Previous content of N was 2, indicating that sign bit of multiplier was in P l.s. Previous P l.s. examined and, if a "1", partial product $\overline{A} + 1$ formed and added to partial answer. In this case P l.s. is not a "1" and $\overline{A} + 1$ is not added to B. P shifted one place right. Mill transferred direct to B.

6. Double length answer in B and P registers.

The content of B and P is then transferred to X and X + 1.


P
N
0 0 0 0
8
0 0 1 0
A
0 0 0 0
Mill
0 1 0 1
4
0 0 0 0
B
0 0 1 0
A
0 0 1 0
Mill
0 0 1 0
4
0 0 0 1
B
0 0 1 0
A
0 0 0 1
Mill
0 0 0 1
3
0 0 0 0
B
0 0 1 0
A
0 0 1 0
Mill
1 0 0 0
2
0 0 0 1
B
0 0 0 0
A
0 0 0 1
Mill
0 1 0 0
1
0 0 0 1
0 0 1 0
This position not used in double length numbers.

12.4 Division

12.4.1 General

The division instruction deals with both the division of a double length dividend by a single length divisor and of a single length dividend by a single length divisor. The dividing process in both of these cases is identical, the single length dividend being made double length before the division process starts.

The dividend is stored in accumulators X and X + 1 and the divisor in store line N. A single length dividend is stored in X + 1.

The answer to a division takes the form of a single length quotient (rounded or unrounded) and a single length remainder, the quotient being placed in X + 1 and the remainder in X.

Division is carried out by a different method to that normally employed when working out a division problem on paper. The method is that of non-restoring and is illustrated by the following examples.

For the sake of simplicity, 4-bit words are used in the examples, the computer carrying out exactly the same steps using 24-bit words. The double length dividend is 7 rather than 8 bits in length. Again this lines up with the computer, which in multiple length working does not use the sign bit of other than the most significant word.

12.4.2 Numerical examples of the division method

In the division method used, the remainder always has the same sign as the divisor. The following four cases illustrate the types of answers to be expected with the various combinations of sign of divisor and dividend.

1) $\frac{29}{5}$ = 5 and 4 remainder.

2) $\frac{27}{-7}$ = -4 and -1 remainder.

3) $\frac{-24}{5}$ = -5 and 1 remainder.

4) $\frac{-29}{-7}$ = 4 and -1 remainder.

Special cases:

Due to the steps involved in producing answers of the above type, when the computer meets a division involving a negative divisor and zero remainder it tends to produce an answer of the type:

$\frac{25}{-5}$ = -6 and -5 remainder OR $\frac{-30}{-6}$ = 4 and -6 remainder.

In order to adjust this type of answer a check is made when the divisor is negative to ensure that the divisor and remainder are not equal. If they are, one is added to the quotient and the remainder is made up to zero prior to the answer being written to accumulators X and X + 1.

EXAMPLE 1

Divide 29 by 5 i.e. 0011101 / 0101

Bits	6 5 4 3 2 1 0		
Quotient	0 1 0 1		
0101	0 0 1 1 1 0 1	-	Shift up the divisor to the same order of magnitude as the dividend (multiply it by 2^3) and subtract: The answer (PR1) is negative so 0101 x 2^3 is bigger than the dividend. Put 0 in Q3 (bit 3 of quotient). Now add 0101 x 2^2 to PR1. PR2 = dividend -2^3.0101 + 2^2.0101 = dividend -2^2.0101
	0 1 0 1		
PR1	1 1 1 0 1 0 1		
	0 1 0 1	+	
PR2	0 0 0 1 0 0 1		

			PR2 is positive so 2^2,0101 is smaller than the dividend i.e. there is a 2^2.0101 in the dividend so put 1 in Q2 ($=2^2$).
	0 1 0 1	−	Now subtract 2^1.0101 from PR2.
PR3	1 1 1 1 1 1 1		PR3 is negative so 2^1.0101 is bigger than PR2 i.e. there is no 2^1.0101 in the dividend. Put 0 in Q1.
	0 1 0 1	+	Now add 2^0.0101 to PR3
PR4	0 0 0 0 1 0 0		PR4 = PR2 - 2^1.0101 + 2^0.0101 = PR2 - 2^0.0101
			PR4 is positive so 2^0,0101 is smaller than PR2 i.e. there is a 2^0.0101 in the dividend so put 1 in Q0.
Answer 0101 and 0100 remainder			PR4 is the remainder

The last step of adding the divisor (0101) to PR3 while determining the value of Q0, will not always give the true remainder. If Q0 is a "1" the remainder is correct, but if Q0 is a "0" the remainder will be the complement of the true remainder with respect to the divisor. This is because in the case when Q0 is zero, the additive/subtractive process has, as far as the remainder is concerned, been taken to far. This can be remedied and in order to arrive back at the true remainder, the divisor is added to the apparent remainder, PR4.

EXAMPLE 2

Divide 27 i.e. 0011011

−7 1001

Bit	6 5 4 3 2 1 0		
Quotient	1 1 0 0		
1001	0 0 1 1 0 1 1		
	1 0 0 1	+	Shift up the divisor to the same
PR1	1 1 0 0 0 1 1		order of magnitude as the dividend

			(multiply it by 2^3) and as it is negative add to it the dividend. The answer (PR1) is negative so 2^3.1001 is bigger than the dividend i.e. it will not go into it. Insert a sign bit at Q3.
	1 1 0 0 1	-	Now subtract 2^2.1001 from PR1.
PR2	1 1 1 1 1 1 1		Note that the sign bit of 2^2.1001 is extended.
			PR2 = dividend + 2^3.1001 - 2^2.1001
			= dividend + 2^2.1001
			PR2 is negative so there is no 2^2.1001 in the dividend. Put 1 in Q2 in order to reduce the quotient (which is negative) by 2^2.
	1 1 1 0 0 1	-	Now subtract 2^1.1001 from PR2
PR3	0 0 0 1 1 0 1		PR3 = dividend + 2^3.1001 - 2^2.1001 - 2^1.1001
			= dividend + 2^1.1001
			PR3 is positive so 2^1.1001 is smaller than the dividend and goes into it. Put 0 in Q1 in order to increase the quotient negatively by 2^1.
	1 1 1 1 0 0 1	+	Now add 2^0.1001 to PR3. PR4 is
PR4	0 0 0 0 1 1 0		the same sign as PR3 so 2^0.1001 is smaller than PR3 and goes into it. Put 0 in Q0 in order to increase the quotient negatively by 2^0.
			As Q0 is zero take process one
	1 1 1 1 0 0 1		step further and add divisor to
True Rem.	1 1 1 1 1 1 1		PR4 in order to arrive at true remainder.

Answer is -4 and -1 remainder

12.4.3 The rules in non-restoring division

The steps taken in the above two examples may be carried out with little thought if the following rules are applied and the computer makes its decisions at each step in a division by following these same rules.

Rule 1 - formation of partial remainders

sign of previous partial remainder	=	sign of divisor	Subtract divisor (suitably shifted) from previous partial remainder.
sign of previous partial remainder	≠	sign of divisor	Add divisor (suitable shifted) to previous partial remainder.

Rule 2 - formation of the quotient

If the present operation involves subtracting the divisor (suitably shifted) from the previous partial remainder and the new partial remainder so formed is different in sign to the previous partial remainder, insert a zero in the quotient. If the signs are the same, insert a one in the quotient.

If the operation involves addition, reverse the rule.

Rule 3 - formation of the remainder

If Q0 is a "1", the last partial remainder is the true remainder.

If Q0 is a "0", add the divisor to the last partial remainder in order to arrive at the true remainder.

The computer makes its decisions at each step in a division by following the above rules.

12.4.4 Operand movement in the arithmetic unit

The following example illustrates typical movements of operands in the arithmetic unit. The quotient and remainder for the division 29/5 finally end up in Registers P and B prior to being stored away.

Starting conditions: the divisor (n) is read into register A,
the dividend (x and x + 1) is read into registers B and P,
4 (normally 24) is read into register N.

The number fed into the N register is used to count the number of operations in the division process.

The B register has an additional bit position which is normally used for the store parity bit. During the division process, it is used at each stage in the division to store the sign bit of the previous partial remainder.

In addition to this, an extra bistable is used to hold the quotient bits prior to their being fed into the l.s. end of register P.

1. The operands are placed as shown and the content of B is present at the mill output. Sign bit of divisor (A) and dividend (B) are compared. As they are equal, set $\overline{A}$ → L and 1 to mill at next stage.

2. $\overline{A}$ → L set so subtract A (2^3 x divisor) from B. B4 ≠ M3 (sign of previous and present partial remainder) and B-A is set, so put zero in QUO at next stage. As A3 ≠ M3, set A → L at next stage. Shift mill and P left onto B and P. Note that P2 goes to P3 and B0.

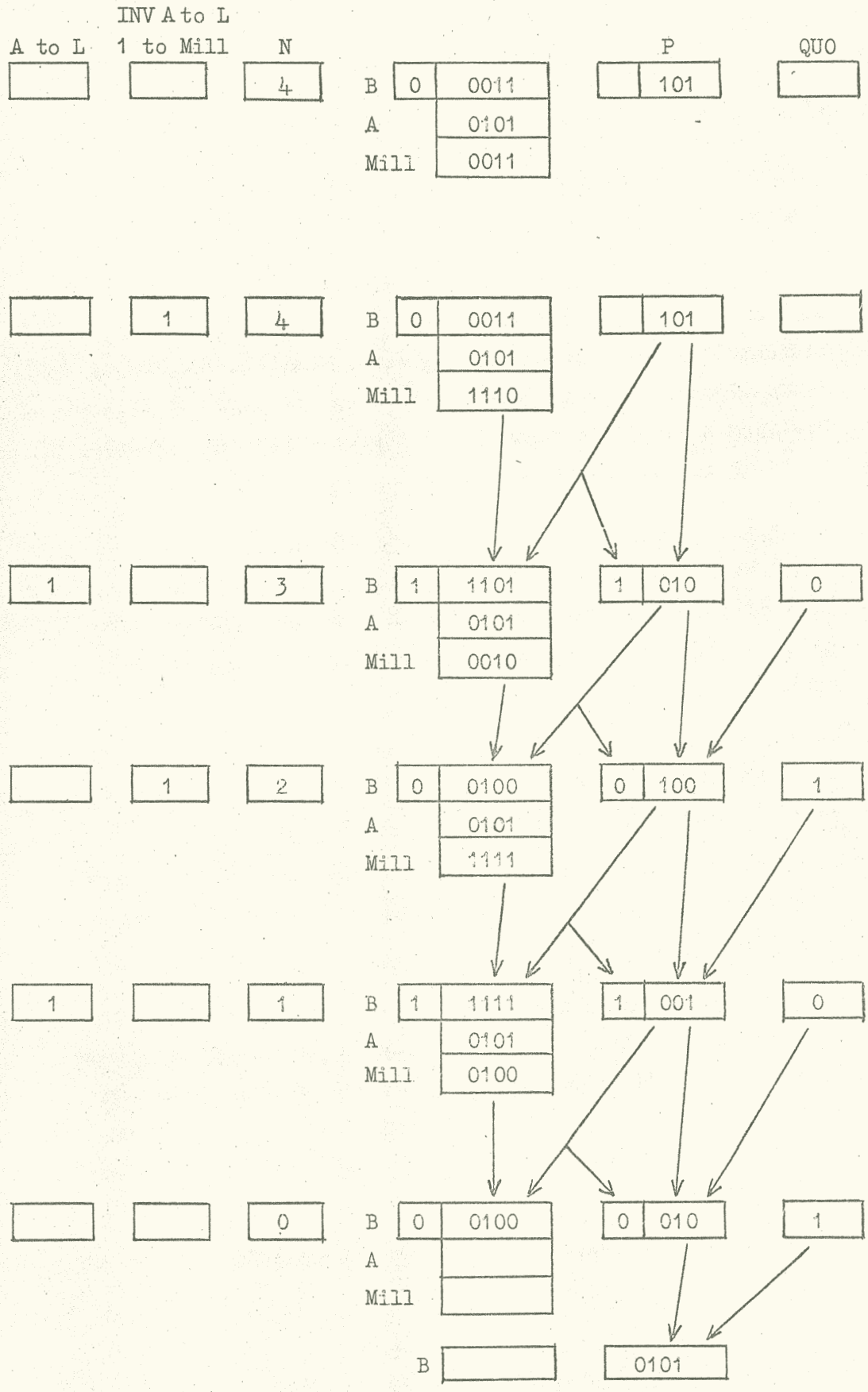
3. A → L set so add A (now effectively divisor x 2^2 because B is shifted left) to B (PR1. B4 ≠ M3 and B+A is set, so put one in QUO at next stage. A3 = M3 so set $\overline{A}$ → L and 1 → mill at next stage. Shift mill, P and QUO left into B and P.

4. $\overline{A}$ → L set so subtract A (Div x 2^1) from B (PR2). B4 ≠ M3 and B-A set, so put zero in quotient at next stage. A3 ≠ M3 so set A → L at next stage. Shift mill, P and QUO left onto B and P.

5. A → L set so add A (Div x 2^0) to B (PR3). B4 = M3 and B+A set so put one in QUO next stage. Computer senses one in register N and now proceeds on different lines in order to finish off the process. Shift P and Q left onto P but shift mill direct onto B.

6. Shift P and QUO left onto P.

   P now contains the quotient and as P0 is a one, B contains the remainder. Had P0 been zero, the divisor would be added to the contents of B in order to arrive at the true remainder.


A to L
INV A to L
1 to Mill
N
P
QUO
4
B 0 0011
A 0101
Mill 0011
101
1
4
B 0 0011
A 0101
Mill 1110
101
1
3
B 1 1101
A 0101
Mill 0010
1 010
0
1
2
B 0 0100
A 0101
Mill 1111
0 100
1
1
1
B 1 1111
A 0101
Mill 0100
1 001
0
0
B 0 0100
A
Mill
0 010
1
B
0101

12.4.5 Division overflow

Overflow in division is detected at the following four points.

1. Before the division proper starts, the divisor is tested for being non-zero.

2. At the beginning of the division process the modulus of the divisor $.2^{23}$ (i.e. shifted 23 places to the m.s. end of the dividend) is checked for being greater than the modulus of the dividend. This is done by checking the sign bit of the dividend against the sign bit of the first partial product. Overflow is set if they are equal.

3. If $-|x|$ is divided by $-|x|.\ 2^{-23}$ the answer is 2^{23} and overflow should be set. Test 2 does not however detect this case because the first partial remainder becomes zero which is different in sign to the dividend and so passes the test.

   This case produces an intermediate answer of the type where the remainder is equal to the quotient and one must be added to the quotient in order to correct for this.

   When one is added, the quotient is changed from 01111 - - - 111 to 10000 - - -000 and overflows, overflow being detected at this point in the process.

4. When rounded division (045 instruction) is carried out, 1 is added to the quotient if the remainder is equal to, or greater than, half the divisor. If the quotient prior to rounding is 01111 - - -111 then overflow will occur when the rounding factor is added. Such an overflow is detected at that stage in the process.

   The types of numbers producing this condition are:

$$\frac{|X| - |\delta|}{|X|} \quad \text{and} \quad \frac{-|X| + |\delta|}{|X|}$$

   δ being the smallest non-zero number.

12.5 Character conversion

12.5.1 Decimal to binary conversion using the 043 instruction

A group of B.C.D. (binary coded decimal) characters as input from, say, paper tape must be converted into a pure binary number before it can be used in computing, and the way in which this is done was described in Chapter 1, section 3.4.

The 043 instruction multiplies the content of X: by ten, adds on nj, and places the result back in X:. The address of nj is specified by the N-address and the counter modifier, M. The following example illustrates the arithmetic method used.

Example: Convert the B.C.D. number 000111, 000010 (72_{10}) to binary

Take 000111	000111	- - - - (a)	7_{10}
Shift left (2a)	0001110		14_{10}
Shift left (4a)	00011100		28_{10}
Add a	000111		7_{10}
Answer is 5a	00100011		35_{10}
Shift left (10a)	001000110		70_{10}
Add next character	000010		2_{10}
	001001000		72_{10}

Fractions are interpreted as integers by the 043 instruction and when converting B.C.D. fractions, it is necessary to divide the binary result by 10^r, where r was the number of B.C.D. characters in the fraction.

12.5.2 Operand movement in the arithmetic unit

The following example shows the movement of operands in the arithmetic unit when the B.C.D. representation of 72_{10} is converted to binary. It is assumed that 000111 (7_{10}) is already in X: and that the B.C.D. characters, 000111 and 000010, are characters 0 and 1 of address N.

1. Select address N and read character 1 to mill

2. Read previous mill to P one place left. Read X + 1 to B to Mill. (X + 1 contains 7).

3. Shift P one place left and transfer previous M22 to P0. Shift previous mill left onto reg.A. Put Zero in A23. Digits shifted into P0 will eventually be added to m.s. half of the double length binary number being accumulated. Transfer A (2 x 7) to mill

4. Shift P one place left and transfer previous M22 to P0. Shift previous mill left onto A register. Put zero in A23. Add A (4 x 7) to B (1 x 7) in mill.

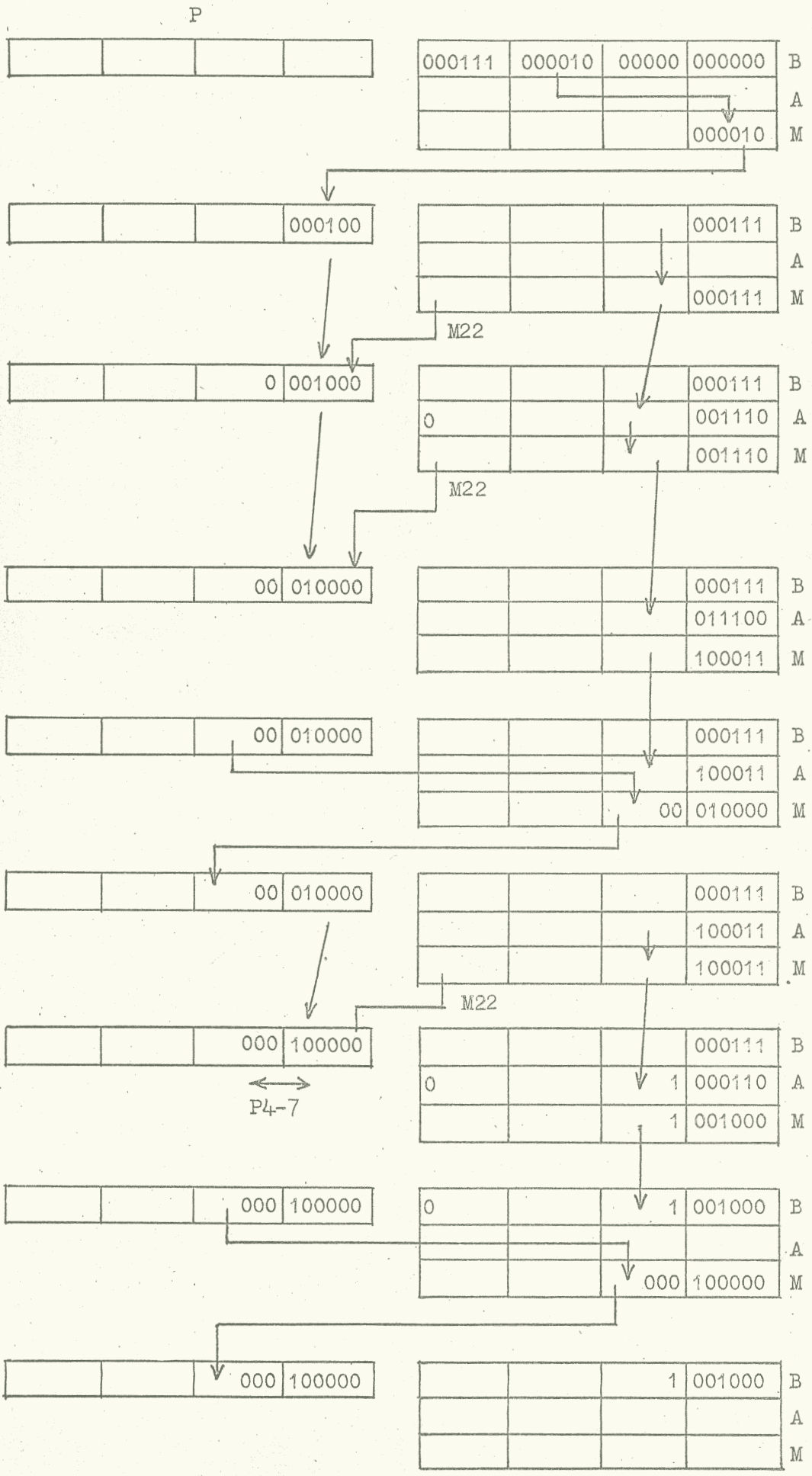
5. Transfer mill to A direct. Transfer P to mill and add 1 if previous M22 = 1. (in this case M22 = 0 so don't add 1).

6. Transfer previous mill to P and then transfer A (5 x 7) to mill

7. Shift P one place left and transfer previous M22 to P0. Shift previous mill left into A register and put zero in A23. Add A (10 x 7) to P4-7 (Character 2).

8. Transfer previous mill which now contains binary 72 to B, and put zero in B23. Transfer P to mill and add 1 if previous M22 = 1.

9. Transfer previous mill to P and write B (binary 72) to accumulator X + 1.


P
000111 000010 00000 000000 B
A
000010 M
000100
000111 B
A
000111 M
M22
0 001000
000111 B
0 001110 A
001110 M
M22
00 010000
000111 B
011100 A
100011 M
00 010000
000111 B
100011 A
00 010000 M
00 010000
000111 B
100011 A
100011 M
M22
000 100000
P4-7
000111 B
0 1 000110 A
1 001000 M
0 1 001000 B
A
000 100000 M
000 100000
1 001000 B
A
M

The stage has now been reached where the original content of X + 1 (binary 7) has been replaced by binary 72, any overflow from X + 1 being held in P0-3. The content of accumulator X is now read out and the process repeated from stage 3 onwards. No character is added in during this second half cycle as it has already been added to the l.s. half words of the double length number. Any overflow from the l.s. half word is, however, present in register P and takes up the same position as did the character during the first cycle of events. In the same way that the character was added to 10 (X + 1) at stage 7, the overflow is added to 10X.

At the end of the second half cycle, the content of register B is written into accumulator X and a check is made to ensure that there are no ones in P0-3. If there are, overflow is set.

12.5.3 Binary to decimal conversion using the 047 instruction

Before results are taken from the computer it is necessary to convert the pure binary content to binary coded decimal characters.

Binary integers can be converted by a continual process of dividing by 10, the remainder at each stage being a character.

Binary fractions require a different approach and are continually multiplied by 10. The integral part produced at each stage is one of the required characters. The same technique can be used for multiplying by 10 as was used with the 043 instruction and, as this is faster than a division process and the hardware already exists, this is the method used.

All numbers to be converted from binary to decimal must be in a fractional form and it is necessary to reduce integers to fractions by dividing by 10^r before a conversion is carried out. (r is the number of decimal characters in the binary integer). The arithmetic process involved in the 047 instruction is illustrated in the following examples.

Example: Convert the binary fraction 0.1100--00 to binary coded decimal characters.

Take 0.110	0.110 - - - - - (a)
Shift left (2a)	1.10
Shift left (4a)	11.0
Add a	0.110
Answer is 5a	11.110
Shift left (10a)	111.100

Extract integral part, 111, as first binary coded decimal character

Take fractional part	0.10 - - - - - (b)
Shift left (2b)	1.00
Shift left (4b)	10.00
Add b	0.10
Answer is 5b	10.10
Shift left (10b)	101.00

Extract integral part, 101, as second binary coded decimal character.

The fractional part is zero so the conversion is completed.

$$.110_2 = .111_2, 101_2$$
$$= .75_{10}$$

12.5.4 Operand movement in the arithmetic unit

The binary fraction to be converted is placed in accumulators X and X + 1, and the characters which result from the conversion are placed in address N. The movement of operands is much the same as with the 043 instruction, the main differences being:

1) there is no character to read from address N so the first step is omitted;

2) there is no character to add to 10 (X + 1) so that step is omitted;

3) the contents of register B are transferred to accumulators X + 1 and X as with the 043 instruction, but in this case they are the fractional part of 10 (X and X + 1);

4) the end of the process is different and routes the content of register P, which contains the integral part of 10 (X and X + 1), into store address N. The character position taken up is determined by the value of the index part of the instruction.

CHAPTER 13

THE FLOATING POINT UNIT

13.1 General

Floating point arithmetic enables the computer to handle a wider range of numbers than is possible with the fixed point arithmetic described in Chapter 12. The floating point number is represented by an argument (a signed fraction) and an exponent (a signed integer). A floating point operation involves argument arithmetic, exponent arithmetic, and normalising of the resultant argument and exponent.

The three ways in which floating point arithmetic can be carried out in the I.C.T. 1900 series are:

1) by Executive,
2) by a library routine,
3) by hardware (the floating point unit).

Floating point operations may be accomplished by either of the first two methods on the 1904 machine, but the 1905 and 1909 are fitted with floating point units which, while involving eleven extra card files, give a ten fold saving in time, and are an economic proposition on installations where much floating point work is to be done.

13.2 The floating point unit

The unit, which is shown in block form in Diagram 26, has its own clock, function register, control bistables and microprogram, and both an argument and an exponent arithmetic unit.

Both normalised and non-normalised operands can be dealt with but the range of quotients in non-normalised division must lie in the range $0 < \text{Quotient} < 2$.

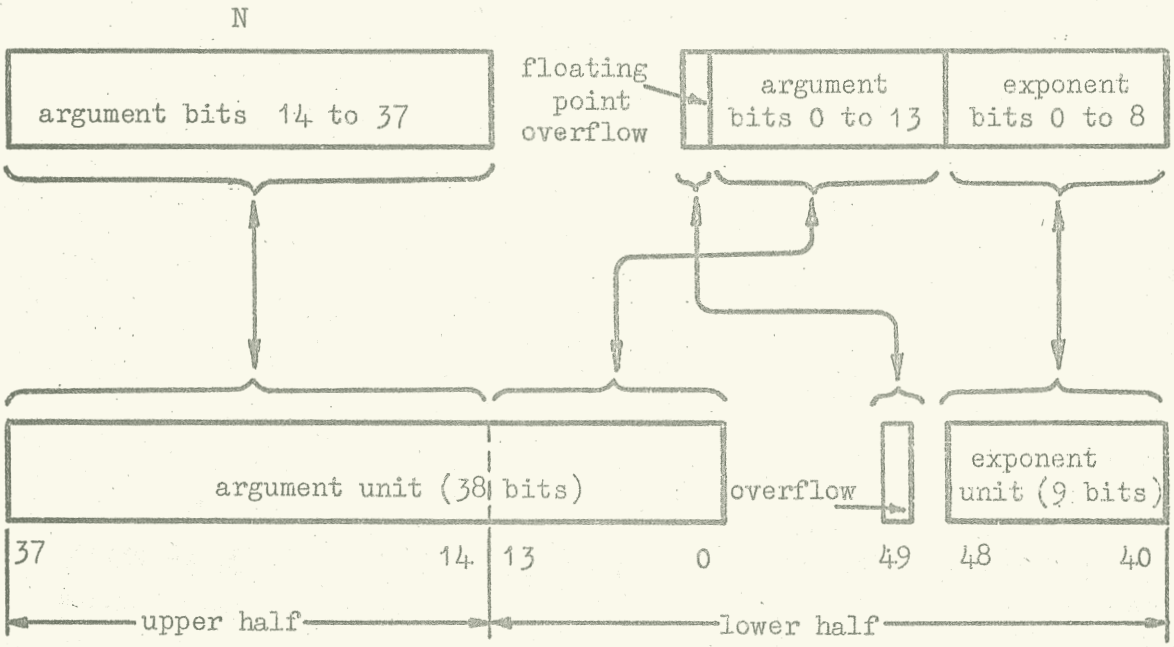
The unit works autonomously so that the central machine is held up only if a further floating point instruction is called for when the unit is still busy with a previous one. Under these circumstances the waiting time would be only a few microseconds and no interrupt is necessary. The central computer clock, while being nominally 1μs in period, is slowed down for read, write, and read pause write store operations. In order that such variations do not slow down floating point operations, the floating point unit has its own clock which runs independently of the central machine clock when floating point operations are being performed.

13.3 The argument and exponent arithmetic units

The argument and exponent arithmetic units are shown on Diagram 27 and it will be seen that they are quite similar to the main arithmetic unit. The slices of the argument and exponent units are numbered 0 to 37 and 40 to 48 respectively, bits 38 and 39 being taken up by the two overflow bits of the argument mill.

Registers FB and FD, and the exponent mill, each have a single overflow bit, the prefix F being used to differentiate between registers and highways in the floating point arithmetic unit and those in the central arithmetic unit.

The units are loaded and unloaded via the SOUT and BIN highways. For purposes of loading and unloading, the floating point arithmetic unit is split into two halves, upper and lower, the demarcation line appearing between bits 12 and 11 of the argument unit. First there is an exchange between store address N and the upper half and then between store address N+1 and the lower half.



13.4 The floating point unit microprogram

Diagram 28 is a block diagram of the floating point unit microprogram. Floating point instructions on machines with hardware units are called G28 orders and such orders lead to the floating point microprogram rather than to macro-instruction entry. Exceptions to this rule are the 130 and 131 instructions which are macro-instructions handled by Executive in all machines (see Chapter 8).

Preparation of order at the top of Diagram 28 is done in the central machine as usual and results in the instruction being placed on both the central arithmetic unit B register, and the F,X,A, and N registers.

If the floating point unit is already engaged in an arithmetic operation the signal BUSY, which is always active for the duration of floating point unit arithmetic operations, holds the preparation/load/unload microprogram sequence in a closed loop in which fast hesitation requests only are allowed. If the unit is not engaged, or when it ceases to be busy, the floating point microprogram causes the function and X-address to be gated into the floating point unit F and X registers and their values determine the rest of the microprogram sequence.

Functions 6 and 7 (load and unload) result in the exponent and argument arithmetic units being loaded from store or unloaded into the

store respectively at the address N: and exit is made from the preparation/load/unloaded sequence to the end of order sequence.

Functions 2, 3, 4, or 5 result in the second operand of the pair needed for an arithmetic operation being loaded into the exponent and argument arithmetic units, and exit is made, at FST, into the arithmetic sequence appropriate to the function. While the floating point arithmetic sequence is being executed the central computer carries on with further instructions. Should a further G28 order occur during the time that the floating point unit is engaged in the present order, overwriting of the content of the floating point unit is prevented by a bistable which is set at the time of FST and produces the signal BUSY.

When a floating point arithmetic operation is completed the normalise sequence is entered; at the end of this, the bistable producing the signal BUSY is reset and the normalised answer is available for either a further arithmetic operation or an unload operation.

Operands used in the floating point unit are taken from the store address N: specified by the N-part of the instruction. The preparation /load/unload microprogram sequence gives rise to control waveforms both in the floating point unit and in the central machine and the sequence makes access to addresses N and N+1 so that the 48-bit argument and exponent can be loaded or unloaded from the floating point unit via the SOUT and BIN highways respectively.

13.5 Addition and subtraction

13.5.1 General

Before two floating point numbers can be added or subtracted, they must be expressed in terms of the same exponent. The two exponents are compared and the argument of the smaller exponent shifted down by a number of places equal to their difference. Except when there is overflow into the sign bit position (one bit overflow), the larger

exponent is then the exponent of the answer.

Examples

Argument	Exponent	
0110100	101	Difference between exponents is
0101000	011	2, so shift argument of smaller exponent right by 2 places and add arguments.
0110100	101	
0001010	101	
0111110	101	Answer

Argument	Exponent	
0100100	011	Difference between exponents is
0110100	011	zero so add. One bit overflow occurs and the answer is converted into the correct form by the normalise sequence.
1011000	011	

13.5.2 Addition

The operands are loaded in by a 136 instruction followed by a 132 instruction. The first operand is loaded into the FA, FB, FC, FD, and FE registers and the second into the FA and FE registers, so leaving the first operand in the FB, FC, and FD registers. The exponent in FE is subtracted from that in FD and subsequent actions depend upon the difference.

1) FD - FE = 0 — add arguments directly.
2) FD - FE > 37 — answer is the operand in FB and FD.
3) 0 < FD - FE < 38 — shift argument in FA right by FD - FE places and add arguments.

4) $0 > FD - FE > -38$ shift arguments in FB right by FD - FE places and add arguments.

5) $FD - FE < -37$ answer is the operand in FA and FE.

In cases where the exponents are equal the answer is accurate, so rounding, when specified (FX0 = 0), only takes place when an argument is shifted. If the bit shifted off the end by the final shift is a "1", carry is generated and added to the newly shifted argument.

13.5.3 Subtraction

Again the first operand is loaded by the 136 instruction into all registers, but the placing of the second operand depends upon the value of bit 2 of X. With X2 = 0 the second operand is placed into FA and FE; with X2 = 1, into FB and FD. The content of FA and FE is then subtracted from that of FB and FD by the same process as addition except that $1 + \overline{FA}$ is added to FB. Also, in case 5, the operand in FA and FE must be negated to provide the answer.

13.6 Multiplication

The argument and exponent of the product of the two numbers $x.2^n$ and $y.2^m$ is $xy.2^{n+m}$.

i.e. $(x.2^n)\ (y.2^m) = xy.2^{n+m}$.

The 136 and 134 instructions place the multiplier in FC and FD and the multiplicand in FA and FE. The exponents are added at an early stage in the operation and the final exponent prior to normalising is stored in FD. FE is used to hold the partial product counter as was register N in the 040 to 042 multiply instructions, and the product is built up in FB and FC.

The method of multiplying is similar to the sequence described in section 12.3.1 except that the multiplier is considered 2 bits at a time in order to speed up the process; that is 19 rather than 38 partial

products are formed and the cumulative sum of the partial products is given a double shift right prior to each successive partial product being added.

Example

0101	(5_{10})	
0110	(6_{10})	
00001010		first partial product is 10 × 0101
00010100		second partial product is 01(00)× 0101
00011110	(30_{10})	Answer

Examination of Diagram 27 will reveal that gating of the multiplicand from the FA register is restricted to FA, 2FA and -FA (i.e. $\overline{FA}$ + 1). As this is the case, 3x multiples of the multiplicand must be obtained by devious means: -FA is subtracted from the sum of the partial products at the present stage and then, by means of carry, F4A is added at the next stage to give -FA + 4FA which is 3FA.

Example

0101	(5_{10})	
0111	(7_{10})	
11111011		first p.p. is -1 × 0101 and set carry
00101000		second p.p. is 1 + carry (=10) × 0101
00100011	(35_{10})	

When the multiplicand is negative the multiple for the last stage is obtained by treating the m.s. two bits of the multiplier as a 2-bit signed integer (+ carry is set).

Example

0101	(5_{10})
1011	(-5_{10})
11111011	first p.p. is -1 × 0101 and set carry
11101100	second p.p. is -2 + carry (= -1) × 0101
11100111	(-25_{10})

When the most significant two bits of the multiplier have the value 10 and carry is not set, the multiple needed is -2x. As with the 3x case there is no gating to accomplish this directly, and again it must be tackled by other means. At the end of a multiply sequence, the product is always given a final single shift right and this feature is used to obtain the -2x multiple: -1x the multiplicand is added to the sum of the partial products and this has the same effect as adding -2x prior to the single right shift. This manipulation is effected by setting carry when the last multiple is -2x.

Example

0101	(5_{10})
1001	(-7_{10})
0101	first p.p. is 1 × 0101
	m.s. multiple is -2x so form no p.p.
	at this stage but set carry
0101	sum of p.p.s given last double shift
0101	final single shift right
1011	as carry is set, add -1x multiplicand
1011101	(-35_{10})

13.7 Division

The argument and exponent of the quotient of two numbers $x.2^n$ and $y.2^m$ is $(x/y).2^{n-m}$.

$$\text{i.e.} \quad \frac{x.2^n}{y.2^m} = \frac{x}{y} .2^{n-m}$$

The 136 and 135 instructions load the first operand into FB and FD, and the second operand into FA and FE. Which of these contains the dividend and which the divisor depends upon the value of bit 2 or X (see function code summary in Chapter 8).

The division process is that of non-restoring and the division sequence is much the same as that employed in fixed point division. The main differences are that in floating point division dividend, divisor and quotient arguments are 38-bit numbers and, except for rounding purposes, the remainder is ignored.

Normalised arguments lie in the ranges,

$\frac{1}{2} \leqslant$ fraction < 1 and $-\frac{1}{2} >$ fraction $\geqslant -1$.

This gives quotient argument ranges, prior to normalising, of
$<$ quotient argument < 2 and $-2 <$ quotient argument $< -\frac{1}{2}$.

Answers lying between $|1.0|$ and $|2.0|$ lie outside the range of a 38-bit argument and cause a one-bit overflow (called provisional overflow) which is stored on a bistable and corrected for by the normalising sequence. Division using non-normalised operands is somewhat restricted in that quotient arguments whose moduli are greater than 2 produce overflow.

Early in the division sequence the exponents are subtracted and their difference is stored in FD leaving FE free to accept the sequence counter. As each partial remainder is formed, 1 is subtracted from the content of FE until, when it becomes zero, the division end effects sequence (formation of the true remainder and quotient, and rounding)

is entered. The end effects are dealt with in the same way as are those for fixed point division, the only added complication being the effect of provisional overflow on rounding.

Rounding, when specified by bit 0 of X, is carried out by the division sequence in cases where one-bit overflow has not occurred, and by the normalising sequence in cases where it has occurred. This is because in the former case the rounding is decided by the remainder, and in the latter case by the least significant bit of the quotient which is moved off the end as a result of the normalising correction for one bit overflow. Except for purposes of rounding, the remainder of the argument of a floating point quotient is not used.

Diagram 29 is a flow diagram which covers both the fixed and floating point division sequences.

13.8 Normalising

13.8.1 General

All floating point arithmetic operations are terminated by the normalising sequence, and the normalising procedure depends upon the type of operation just performed and the form of the argument and exponent at the time of entry to the normalising sequence. This may take four forms:

a) zero argument,
b) normalising required,
c) one bit overflow,
d) satisfactory final form.

After the argument has been normalised the exponent is tested to ensure that it falls in the range 256 > exponent ≤ -256. The normalised floating point is then available either for transfer to the core store or for further arithmetic operations within the unit.

13.8.2 Normalising after addition and subtraction

a) Zero argument - addition and subtraction

On entry to the normalise sequence the argument is on the argument mill output and if all 40 bits are zero the argument and exponent registers are cleared. A "1" is forced into bits 48 and 49 of the exponent making it -256 in value.

b) Normalising required - addition and subtraction

When bits 36, 37, and 38 of the argument are equal and the argument is not zero, normalising is required. Bits 35 and 36 are checked and the argument is shifted one place left, 1 being subtracted from the exponent. If bits 35 and 36 were unequal when checked, the normalising action is finished; if they were equal the process is repeated until the argument has been shifted into the normalised state.

c) One bit overflow - addition and subtraction

When argument bits 37 and 38 are unequal, one bit overflow is indicated (refer back to second examples in section 13.5.1). The argument is shifted one place right and one is added to the exponent.

d) Satisfactory final form - addition and subtraction

When argument bit 37 equals 38 and 36 is not equal to 37, or when the argument is not zero and normalising is not specified (FX1 = 1), no normalising action is required.

12.8.3 Normalising after multiplication

a) Zero argument - multiplication

On entry to the normalising sequence, the m.s. half of the product is in register FB and on the argument mill output,

and the"l.s."half is in register FB. If the mill output is zero the"l.s." half of the product is transferred to register FB, and -37 is added to the exponent to correct for the effective 37-place shift from FC to FB. The content of FB is checked for zero as was the previous content, and if zero, the action is as for addition/subtraction. If it is not zero the result, if not already normalised, is normalised regardless of the state of FX1.

b) Normalising required - multiplication

Normalising action for both single and double length arguments is identical except for the 37-place shift already described when the m.s. half of the product argument is zero. In both cases the content of FB is continually shifted left and the exponent decremented as in addition normalising, a test being made of bits 35 and 36 of FB at each stage. The content of FC is also shifted left at each stage, FC 36 being transferred to FB0.

When FB 35 and 36 are unequal, the normalising shift is finished and rounding, if specified, is carried out when FC 35 is "1". Rounding may cause one-bit multiplication and this is dealt with as an addition.

c) One-bit - multiplication

Correction is as for addition with the added complication that if rounding is specified, unity is added to the argument prior to the single shift right. Rounding may cause a further one-bit overflow and this is treated as for addition.

d) Satisfactory final form - multiplication

When rounding is required and the argument is already in satisfactory final form, "1" is added if FC 36 = 1. This may cause one-bit overflow and, if so, correction is made for it

in the normal manner. Otherwise the action is as for addition.

13.8.4 Normalising after division

a) Zero argument - division

As for addition

b) Normalising required - division

As for addition

c) One bit overflow - division

One bit overflow in division is indicated either by argument bits 37 and 38 being unequal or by provisional overflow set but not by both of these conditions (special case: see section d). The action is as for multiplication.

d) Satisfactory final form - division

Except for the special case mentioned in c), the action is as in multiplication. The special case, where both argument bits 37 and 38, and provisional overflow are set, arises as a result of dividing a positive dividend by a negative divisor of equal magnitude (answer -1). This combination produces an answer which is already correct and requires no normalising.

13.9 Exponent test

When the argument is non-zero the final exponent is checked for underflow or overflow.

The conditions of underflow or overflow are usually detected by examining bits 48 and 49 (true sign bit) of the exponent: these are "0" and "1" for underflow, and "1" and "0" for overflow respectively.

An exception to the rule is when the final exponent is less than -512. This can only happen when operands with exponents of the appropriate extreme values are multiplied or divided, resulting in an exponent, before normalising, of equal to or greater than -512. Such an exponent may become less than -512 when its argument is normalised, such values resulting in a " " in bit 48 and a "0" in bit 49. This would be interpreted as overflow were it not for the fact that on entry to the normalising sequence, a bistable records the true sign of the exponent, and the state of this bistable is used to differentiate between overflow and the extreme cases of underflow.

When underflow has occurred the procedure is similar to that for zero argument normalising as any number with an exponent of less than -256 can be considered as zero. Overflow is set up on the floating point unit's overflow bistable and, when an unload order is given, P23 in the central arithmetic unit is set.

13.10 End action

The final answer is placed in all three argument registers and both exponent registers, and registers FX and FF are cleared. The busy bistable is then reset in order that any further G28 orders may proceed and the floating point unit clock is stopped until such orders are given.

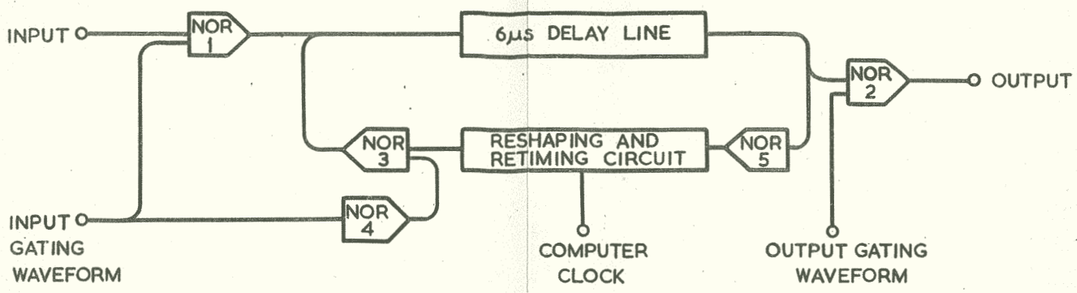


DIAGRAM 1 DELAY LINE STORE

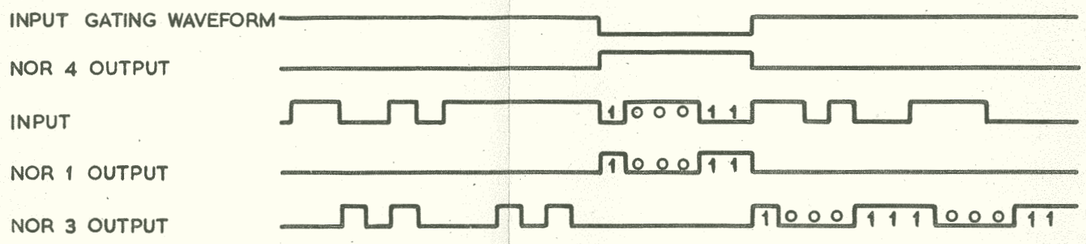


DIAGRAM 2 INPUT TIMING DIAGRAM

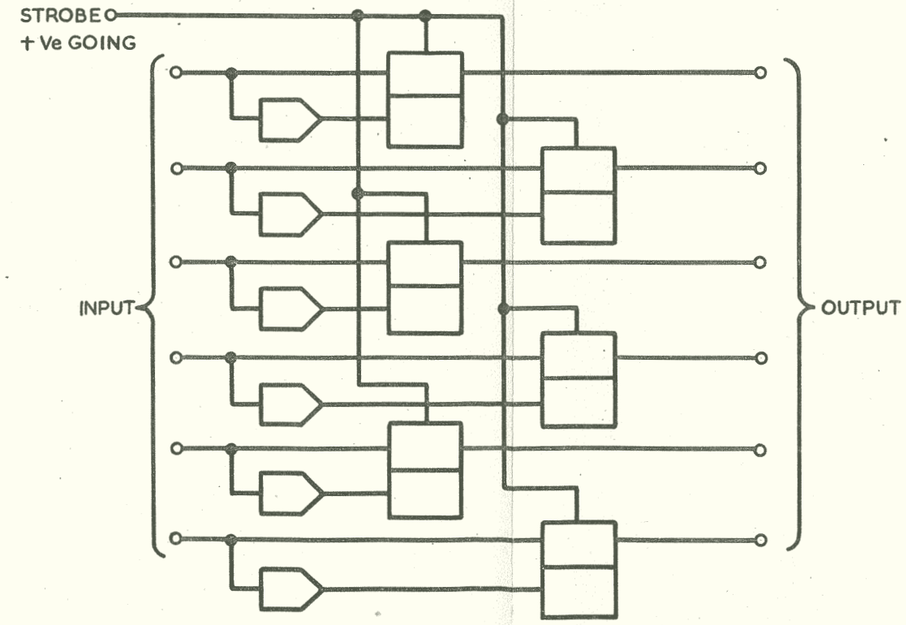
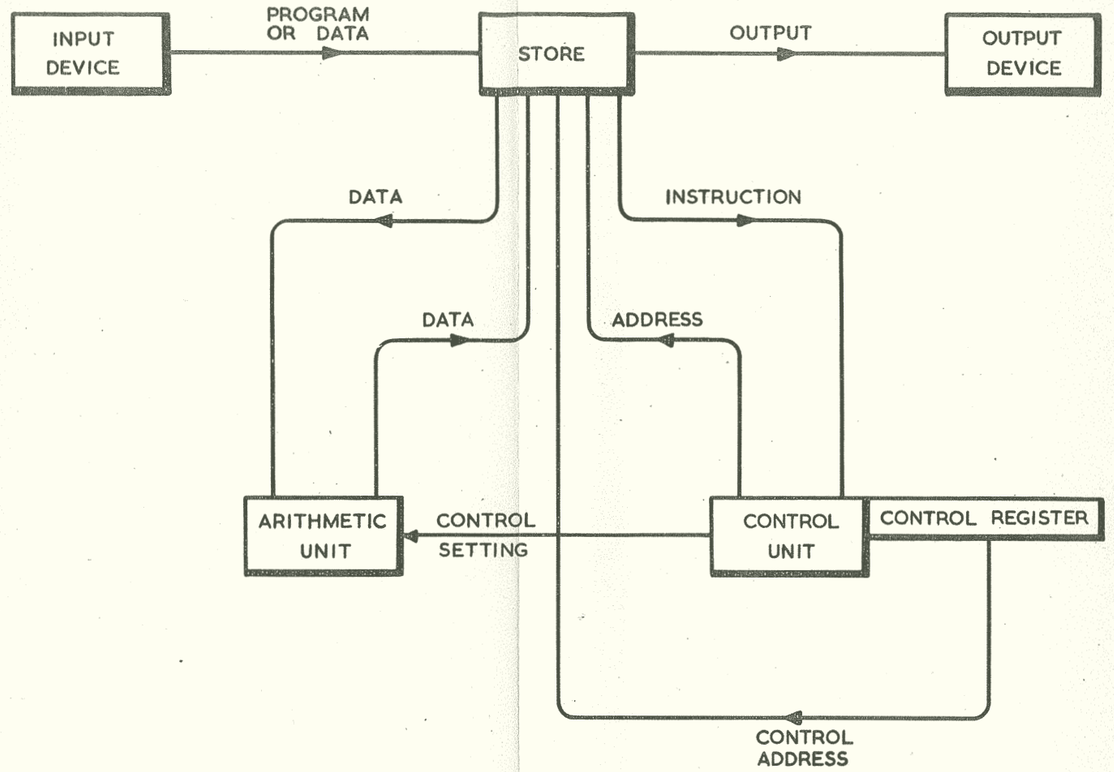


DIAGRAM 3 BISTABLE STORE

DIAGRAMS 1, 2 & 3



BLOCK DIAGRAM OF A SIMPLE COMPUTER

DIAGRAM 4

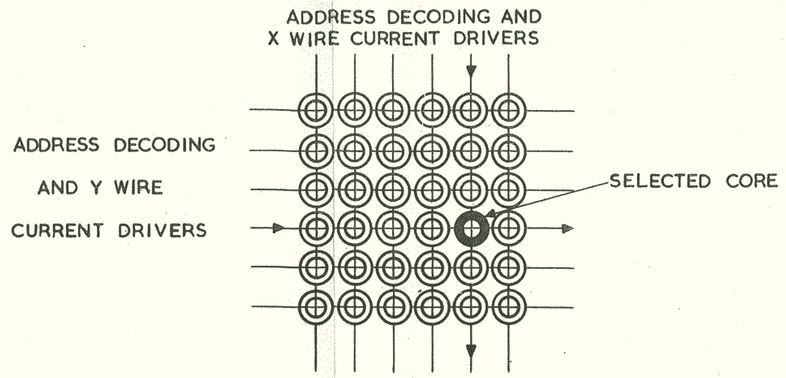


DIAGRAM 7 ADDRESSING A CORE

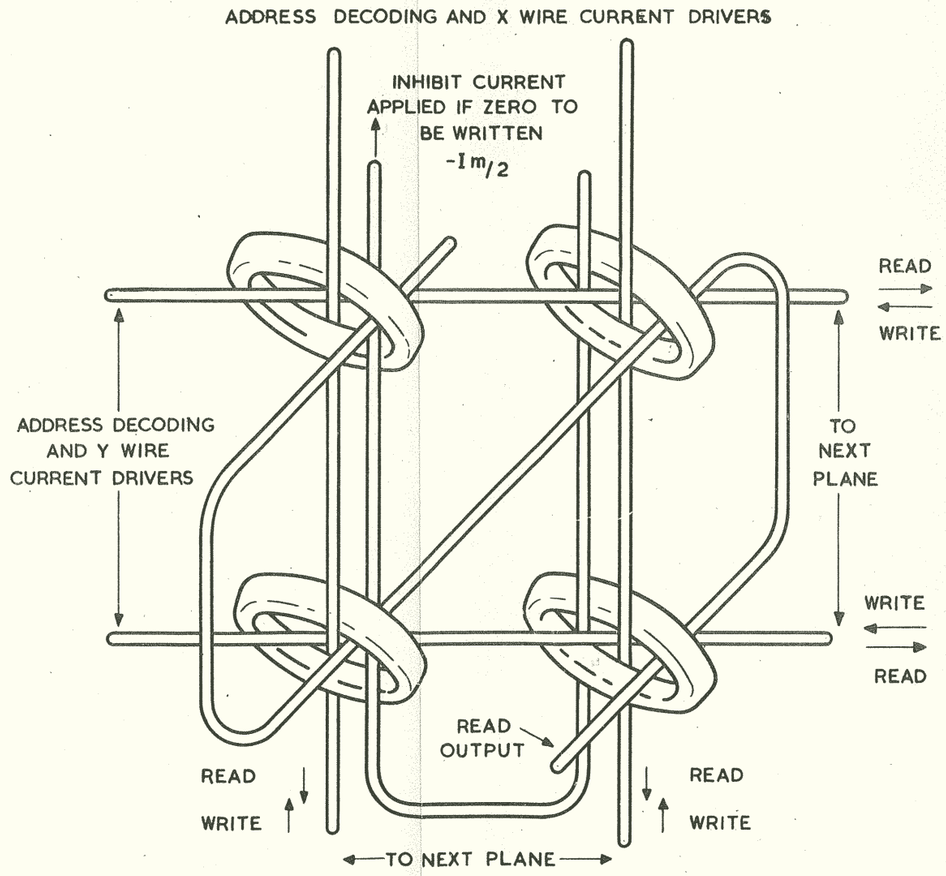
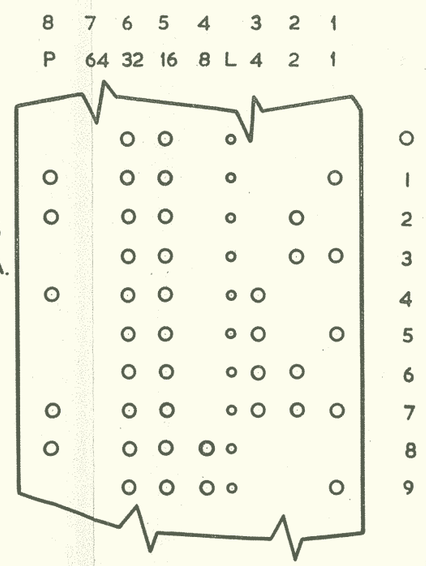


DIAGRAM 8 A SECTION OF A CORE STORE PLANE

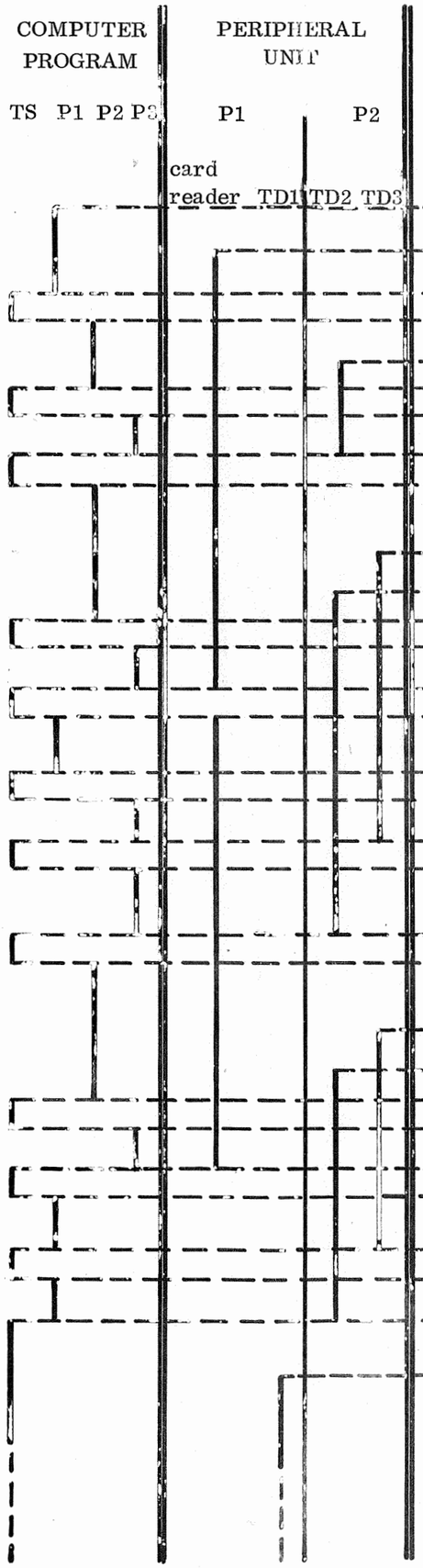
TYPE OF STORE	READ/ WRITE SPEED	AVERAGE ACCESS TIME	APPROXIMATE CAPACITY	COST PER WORD
CORE STORE	MICROSECONDS PER WORD	MICROSECONDS PER WORD	LIMITED BY COST TO SEVERAL THOUSAND	HIGH
DRUM	MICROSECONDS PER WORD FOR PARALLEL DRUM	SEVERAL MILLI — SECONDS	50,000 WORDS PER DRUM	LOW
MAG. TAPE	60 MICRO — SECONDS PER WORD	MILLI — SECONDS TO MINUTES	1/2" TAPE 10. 1/4" SPOOL SEVERAL MILLION WORDS	VERY LOW
DELAY LINE	MICROSECONDS PER BIT	HALF DELAY LINE LENGTH	ONE TO SEVERAL HUNDRED WORDS	MEDIUM
BISTABLE	FRACTIONAL MICROSECOND	FRACTIONAL MICROSECOND	1 BIT PER 2 VALVES OR TRANSISTORS	VERY HIGH
DISC FILE	40 MICRO — SECONDS PER WORD	200 MILLI — SECONDS	SEVERAL MILLION WORDS	VERY LOW

DIAGRAM 9 TABLE OF SPEEDS AND CAPACITY OF DIFFERENT TYPES OF STORE

DIAGRAM 10
REPRESENTATION OF NUMBERS 0 TO 9 ON 8 HOLE I.S.O./ A.S.A. TAPE



DIAGRAMS 9 & 10



Program 1 - transfer punched card to magnetic tape: two cards per block.

Program 2 - update magnetic tape and write results back onto magnetic tape.

Program 3 - mill limited program.

REMARKS

P1 entered

P1 starts card transfer: 1st of pair of cards.

P1 suspended waiting end of card read; TS entered.

P2 entered

P2 starts tape read. Read one block from TD2.

P2 suspended waiting end of tape read; TS entered.

P3 entered.

TD2 finished; P3 interrupted and TS entered.

P2 entered because it is highest priority available and updates information from TD2.

P2 ends updating, initiates write to TD3.

P2 starts tape read. Read one block from TD2.

P2 suspended waiting end of tape read; TS entered.

P3 entered: highest priority available.

Card reader finished; P3 interrupted and TS entered.

P1 entered: highest priority available. Starts card read transfer: second of a pair of cards.

P1 suspended awaiting end of card read; TS entered.

P3 entered: highest priority available.

TD3 finished; TS entered; fact recorded.

P3 re-entered: no action required on TD3. P3 still highest priority.

TD2 finished; P3 interrupted and TS entered.

P2 entered: highest priority available. Updates information from TD2.

P2 finished updating; initiates write to TD3.

P2 starts tape read. Read one block from TD2.

P2 suspended awaiting end of read; TS entered.

P3 entered: highest priority available.

Card reader finished; P3 interrupted and TS entered.

P1 entered: highest priority available. Compiles information off cards into form suitable for mag. tape.

TD3 finished; TS entered; fact recorded.

P1 re-entered: highest priority available.

TD2 finished; TS entered; and fact recorded.

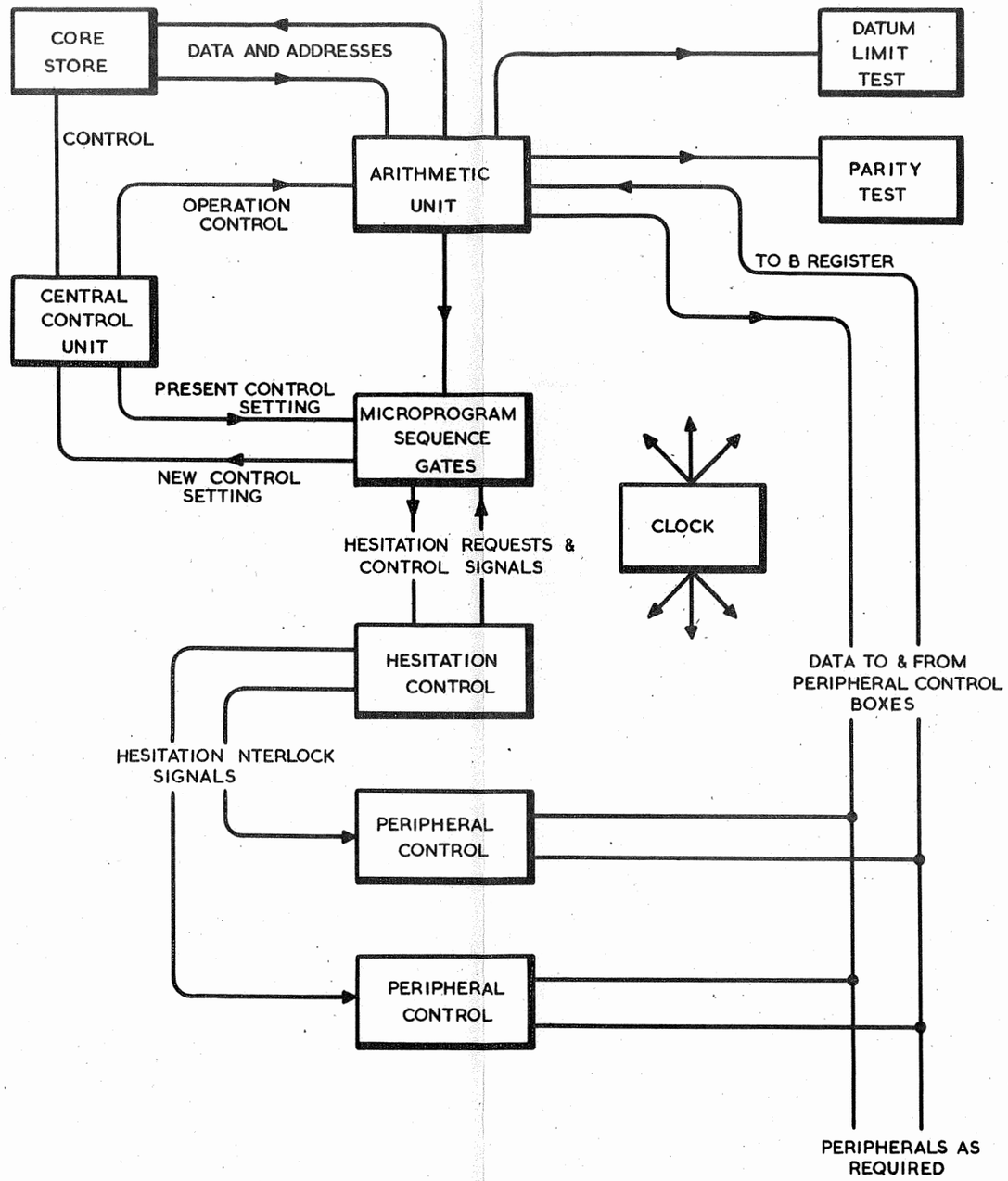
P1 re-entered: highest priority available.

Compiling of card data finished: P1 starts transfer to TD1.

And so on . . .

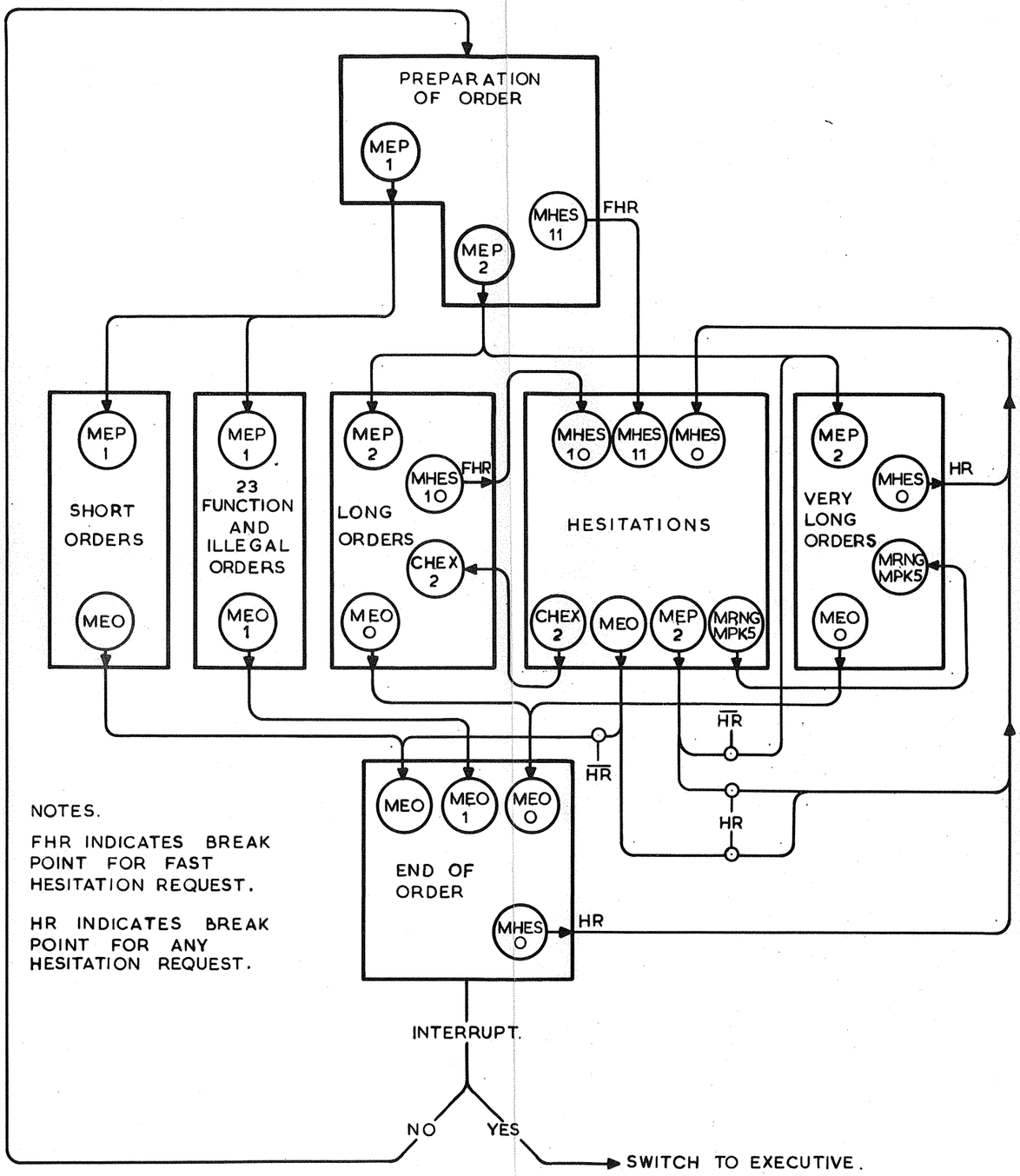
Diagram 11

Timing Diagram for a computer in Time Sharing

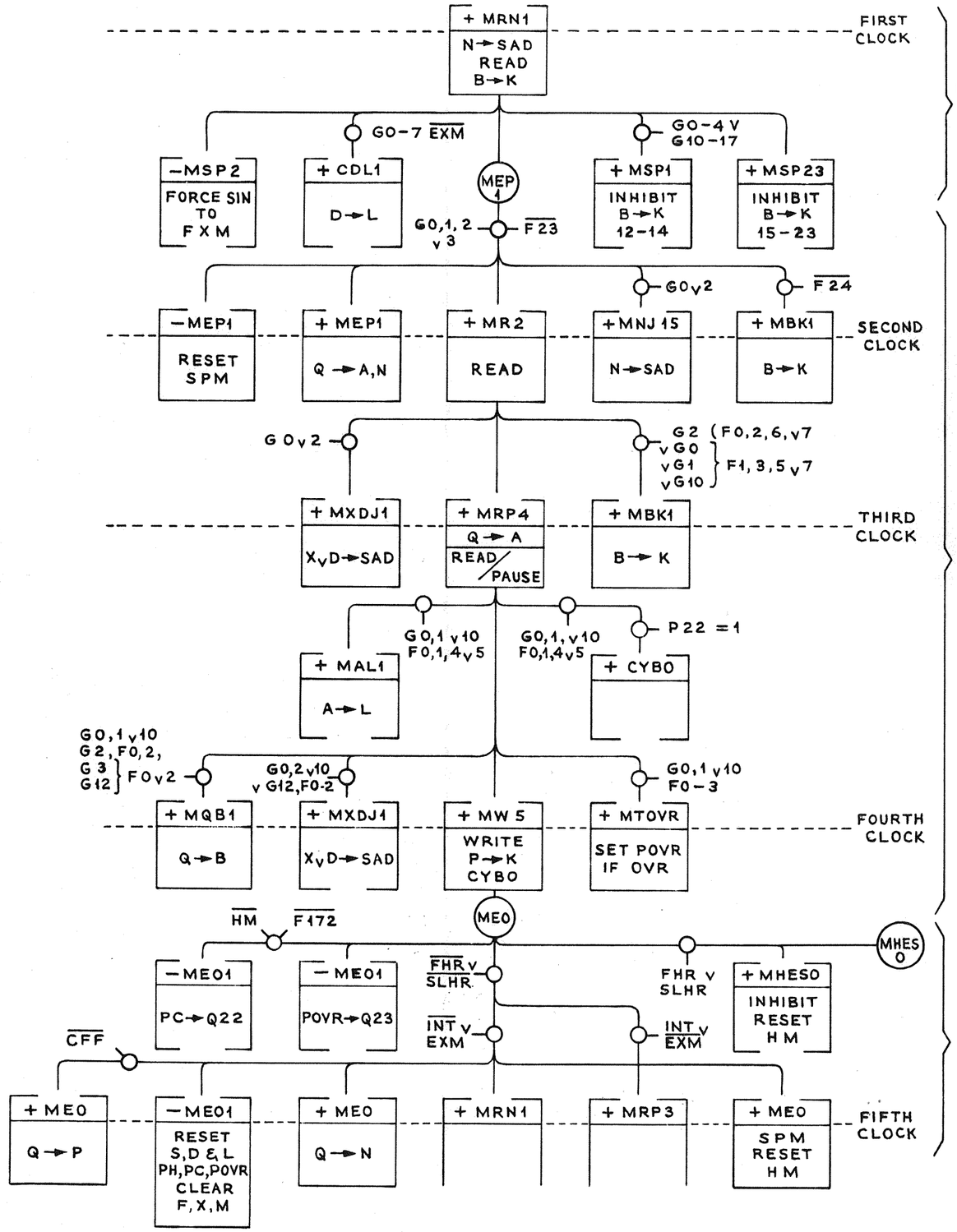


BLOCK DIAGRAM OF CENTRAL COMPUTER

DIAGRAM 12



BLOCK DIAGRAM OF MICROPROGRAM. DIAGRAM 13.



a) Starting conditions: assuming no hesitation request or interrupt, end of order produces +MRN1 which primes control bistables N → SAD, Read and B → K.

b) At first clock: read, N → SAD and B → K bistables set. N contains instruction No.

c) During first digit time: waveforms +MSP1, +CDL1, +MSP23, -MSP2, +MEP1, -MEP1 generated by READ, B → K, N → SAD and function; waveforms +MNJ15, +MR2, +MEP1 generated by -MEP1 and function. Register N's content applied to store (N → SAD) and instruction read from store (READ) to register B and K highway (B → K). Bits 12 - 23 inhibited (+MSP1 and +MSP23), gating N-address only to K. Datum gated onto L highway (CDL1). Mill output will be datumned N-address. Function, and accumulator and modidier address forced onto F, X, and M registers.

d) At second clock: mill output Q (datumned N-address) strobed to A and N (+MEP1) and N → SAD, Read and B → K bistables set. SPM (control bistable contained in microprogram to generating certain microprogram waveforms) reset.

e) During second digit time: datumned N-address, register N's content, applied to store (N → SAD) and operand n read out (READ) to register B and K highway (B → K). Waveforms +MRP4, +MBK1 generated by READ, INV SPM and function; waveforms MXDJ1 generated by READ, INV SPM, function and INV PH. PH, a bistable in central control, fulfills similar function to SPM.

f) At third clock time: mill output Q (operand n) strobed into register A (Q → A); bistables Read/Pause, XvD → SAD, B → K set.

g) During third digit time: datumned X-address applied to store (XvD → SAD) and operand x read out (READ/PAUSE) to register B and K highway (B → K). Waveform +MAL1 generated by READ/PAUSE, INV SPM and function, and n gated onto L highway (+MAL1, applied to collector or control bistable A → L, forces bistable to set without necessity of a strobe pulse). n now on L input to mill, x on K input. If this is part of a multiple length addition there may be a carry stored in bit 22 of register P, from less significant additions. CYBO, generated by P22, READ·PAUSE, INV SPM and function, puts a '1' in l. s. carry input of mill. +MQB1, +MXDJ1, +MTOVR generated as a result of READ/PAUSE and function; +MAL1 generated as a result of READ/PAUSE, INV SPM and function; +MW5 generated as a result of READ/PAUSE, function and INV PH.

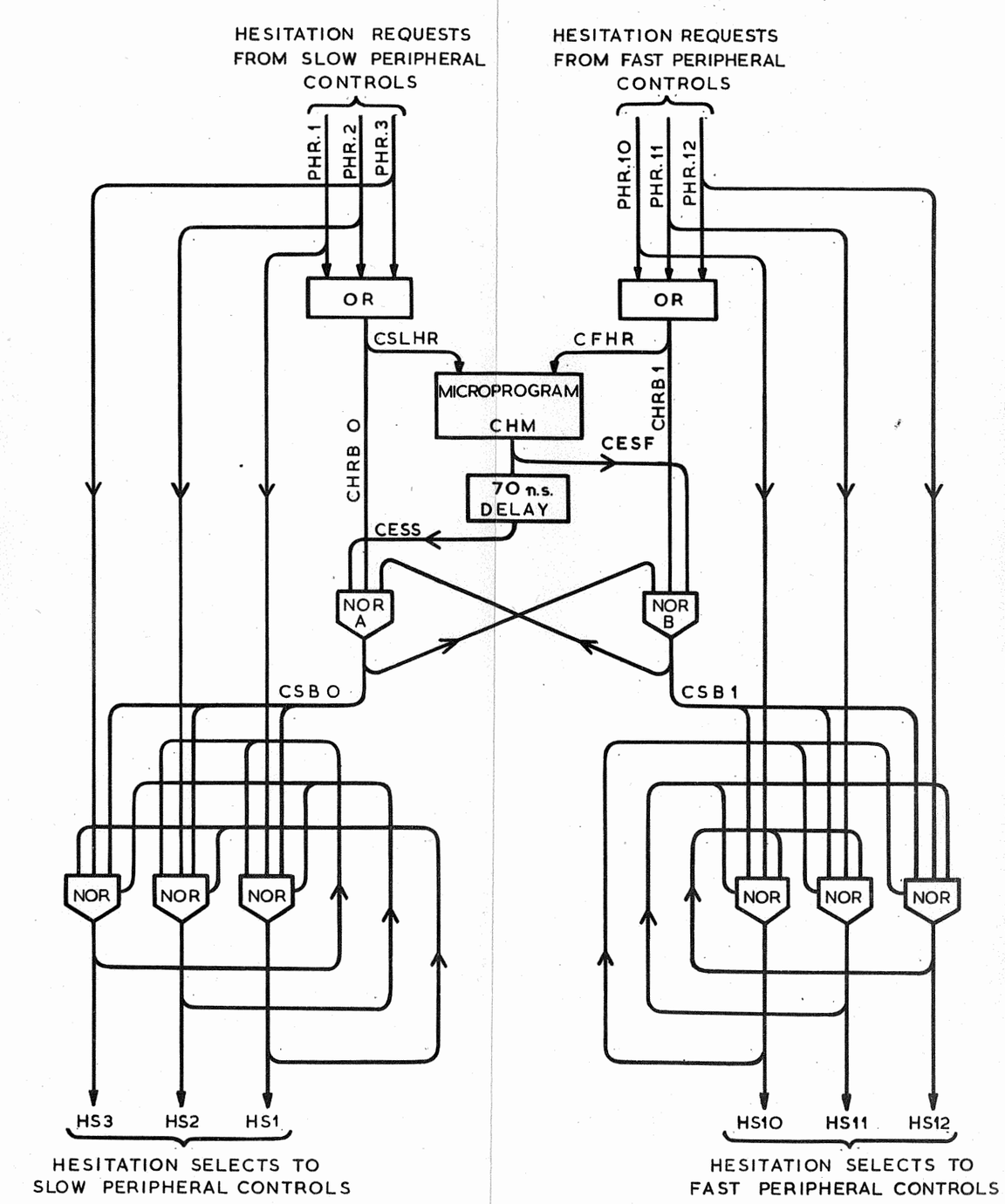
h) At fourth clock: mill output Q (x+n+carry) strobed onto register B and bistables Write, P → K, CYBO, XvD → SAD set. Bistable POVR set if overflow conditions exist.

i) During fourth digit time: datumned X-address applied to store (XvD → SAD), register B's content (x+n+c) written away (WRITE). Register P's content (instruction number) gated onto K highway (P → K); CYBO puts a '1' into l. s. carry input of mill. Mill output will be instruction no. +1. Waveform +MEO derived from WRITE and function; +MEO1, -MEO1 derived from MEO. +MRN1 derived from -MEO1, INV H. R. and INV INT. POVR gated to Q 23by -MEO1 (POVR → Q23); PC (carry bistable output) similarly gated to Q22. PC zero: 001 instruction produces overflow not carry.

j) At fifth clock: mill output Q (instruction no.+1) strobed into register P and N (Q → P, Q → N). Bistables PH, HM and Suspend D and L reset; bistable SPM set. These bistables reset automatically at end of an order as a 'clearing up' process. Whether they require resetting depends on order and any hesitations just completed.
Process now returned to stage b) and new instruction executed.

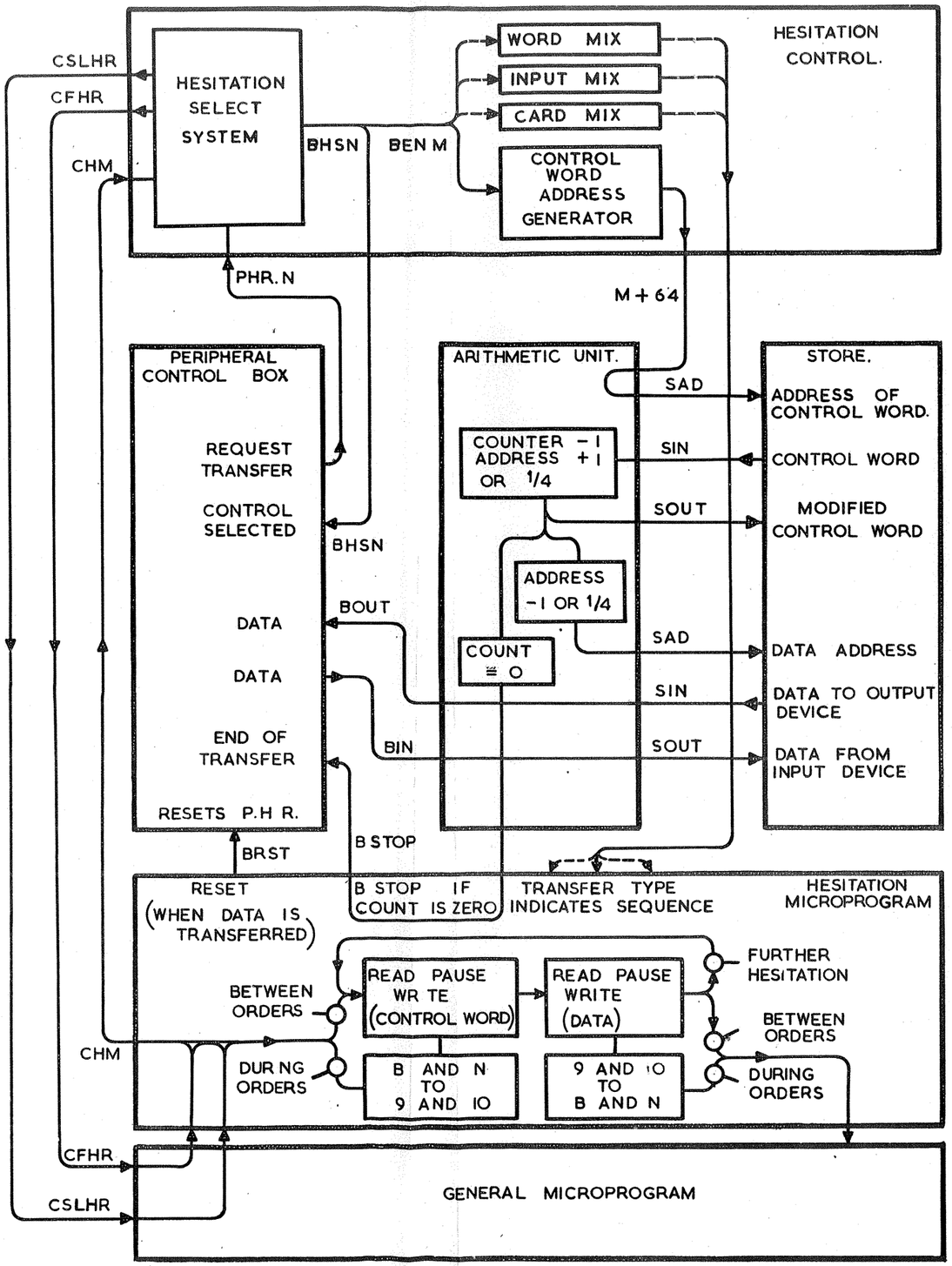
THE 001 INSTRUCTION MICROPROGRAM SEQUENCE (UNMODIFIED)

DIAGRAM 14

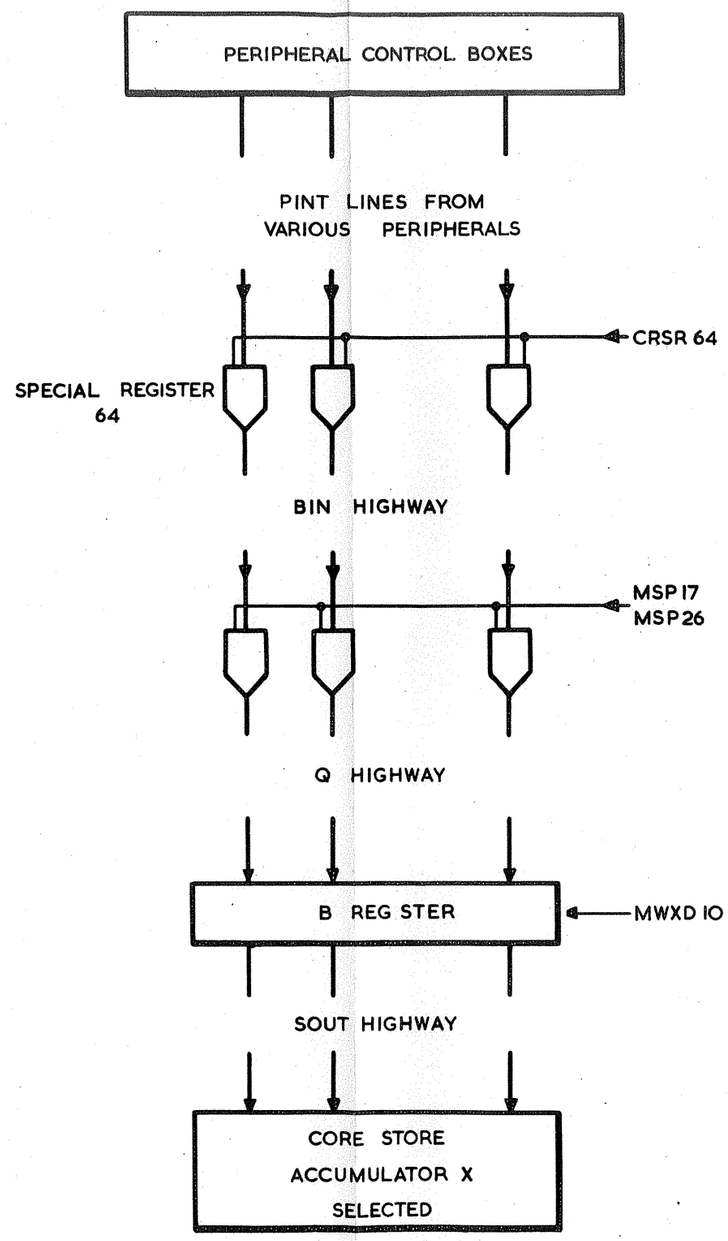


HESITATION SELECTION SYSTEM
(SIMPLIFIED)

DIAGRAM 15



HESITATION SYSTEM BLOCK DIAGRAM. DIAGRAM 16.



TRANSFER OF SR64 TO ACCUMULATOR X

DIAGRAM 17

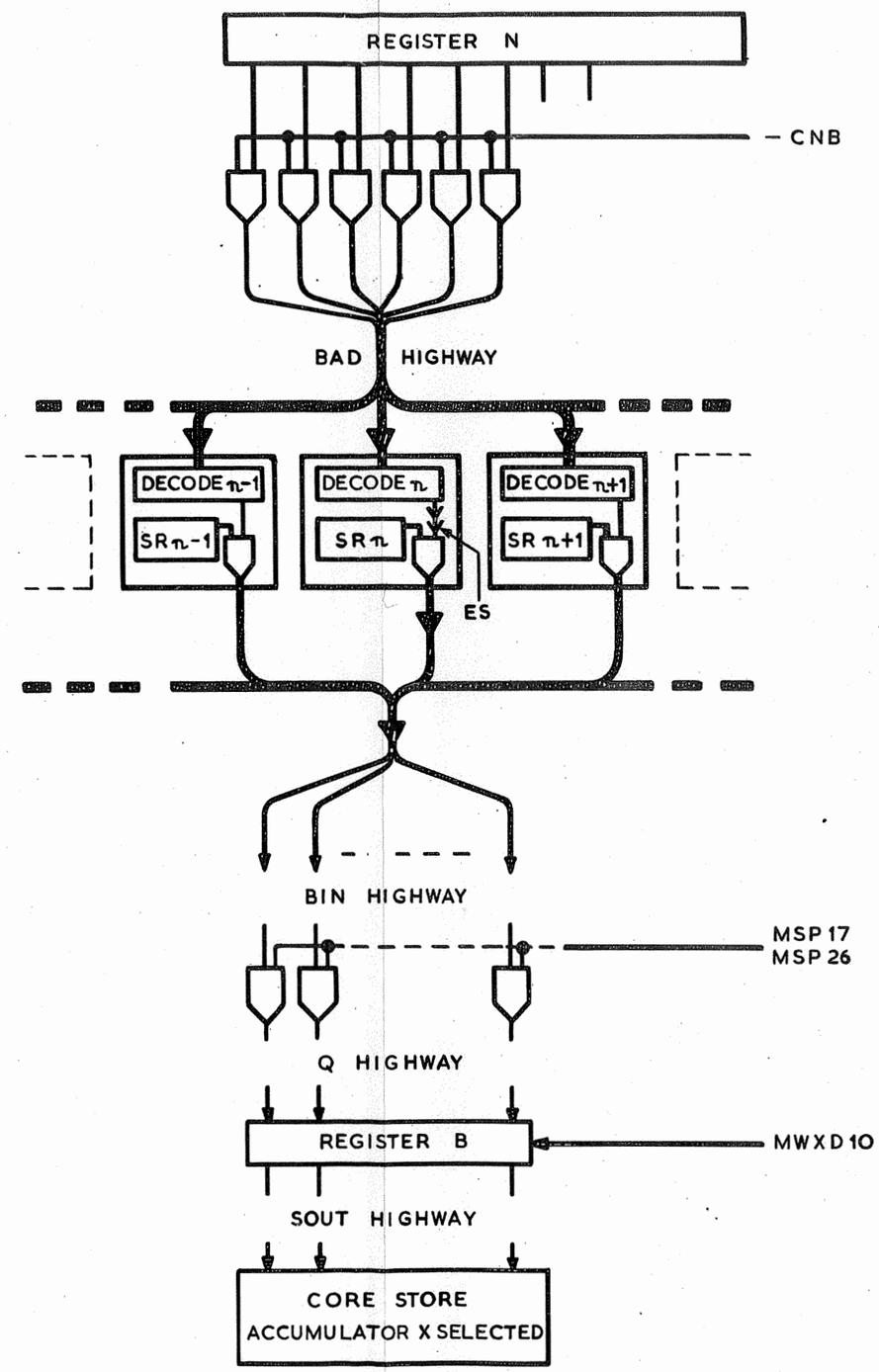
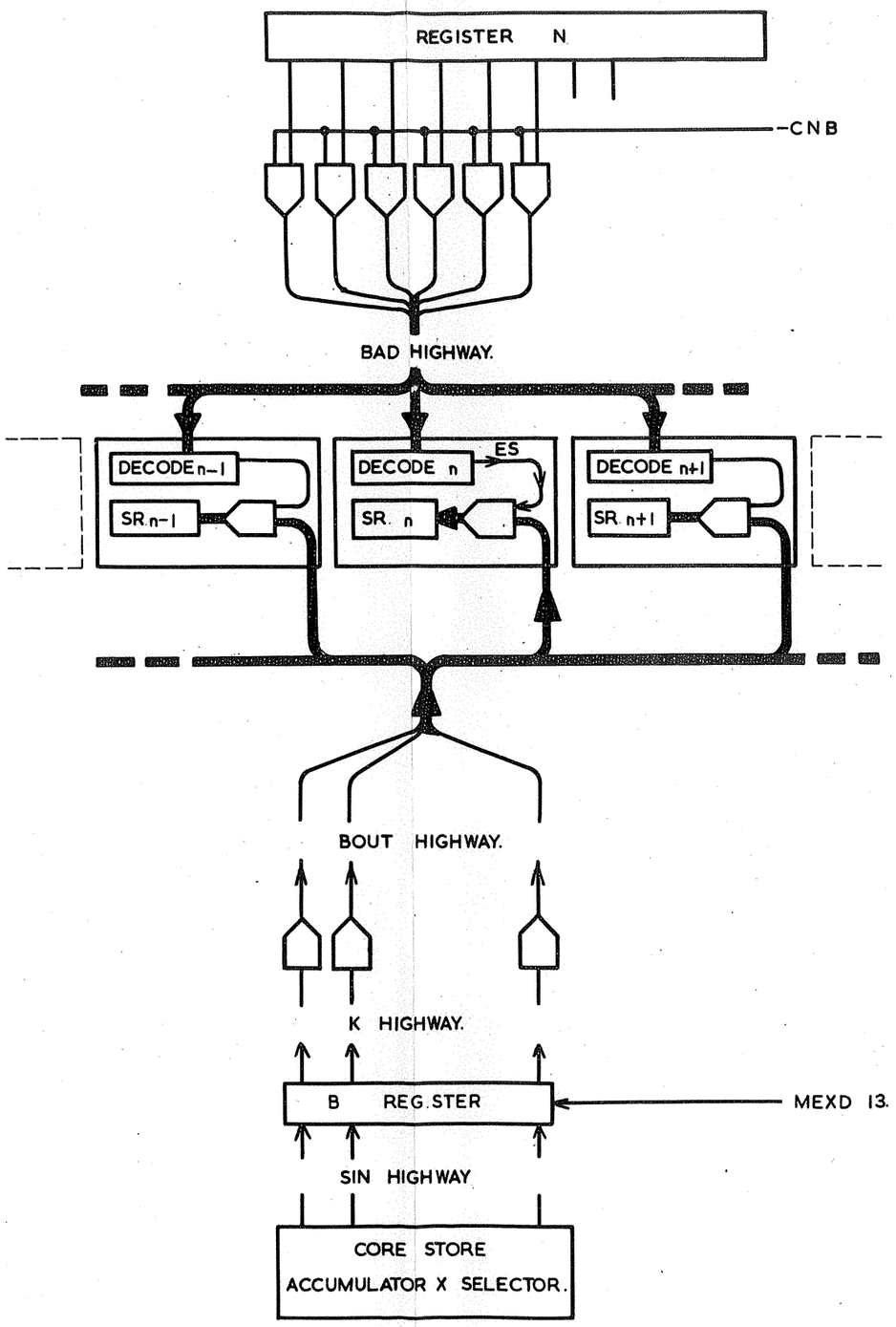
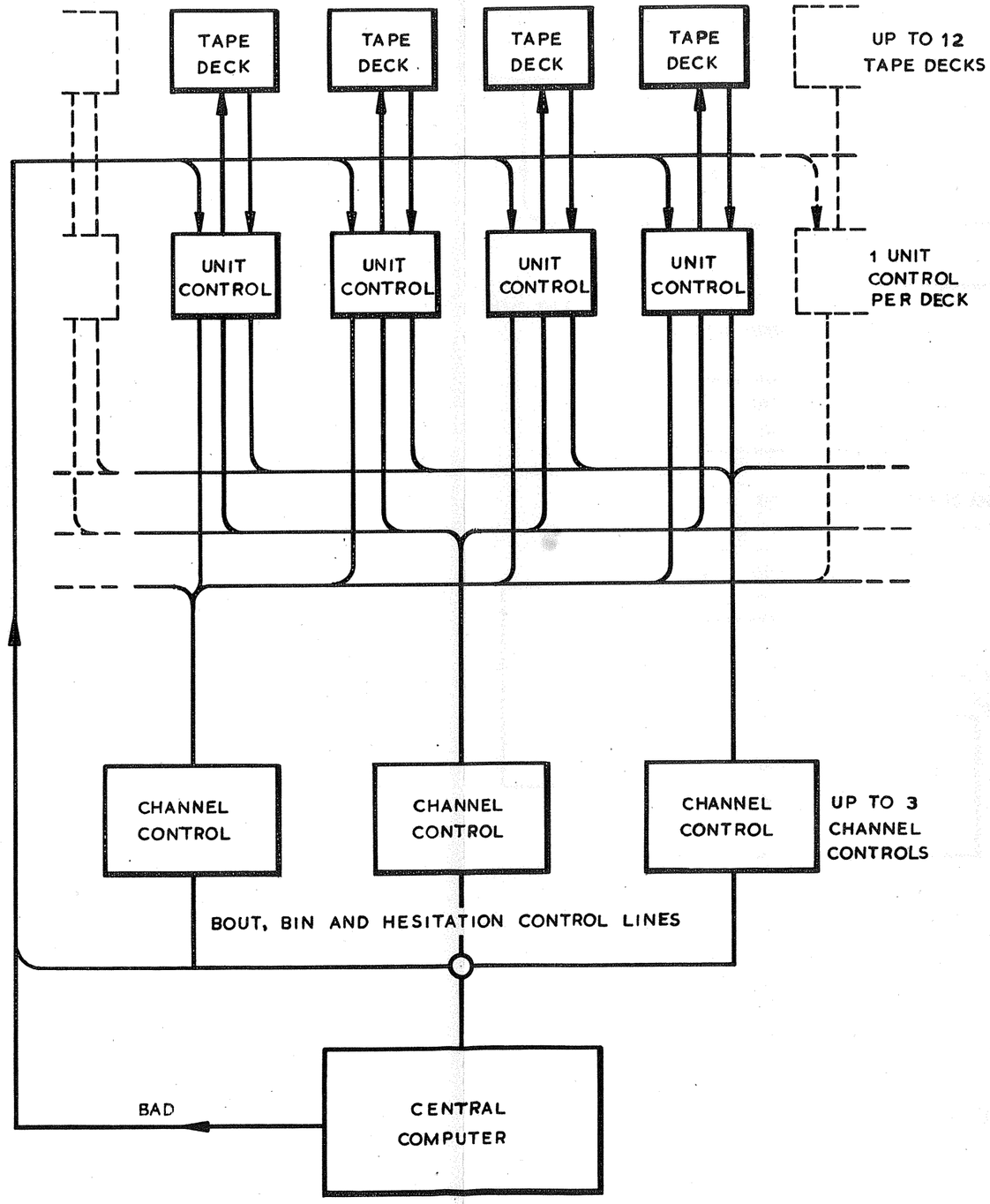


DIAGRAM 18 TRANSFER OF PERIPHERAL SPECIAL REGISTER n TO ACCUMULATOR X

DIAGRAM 18

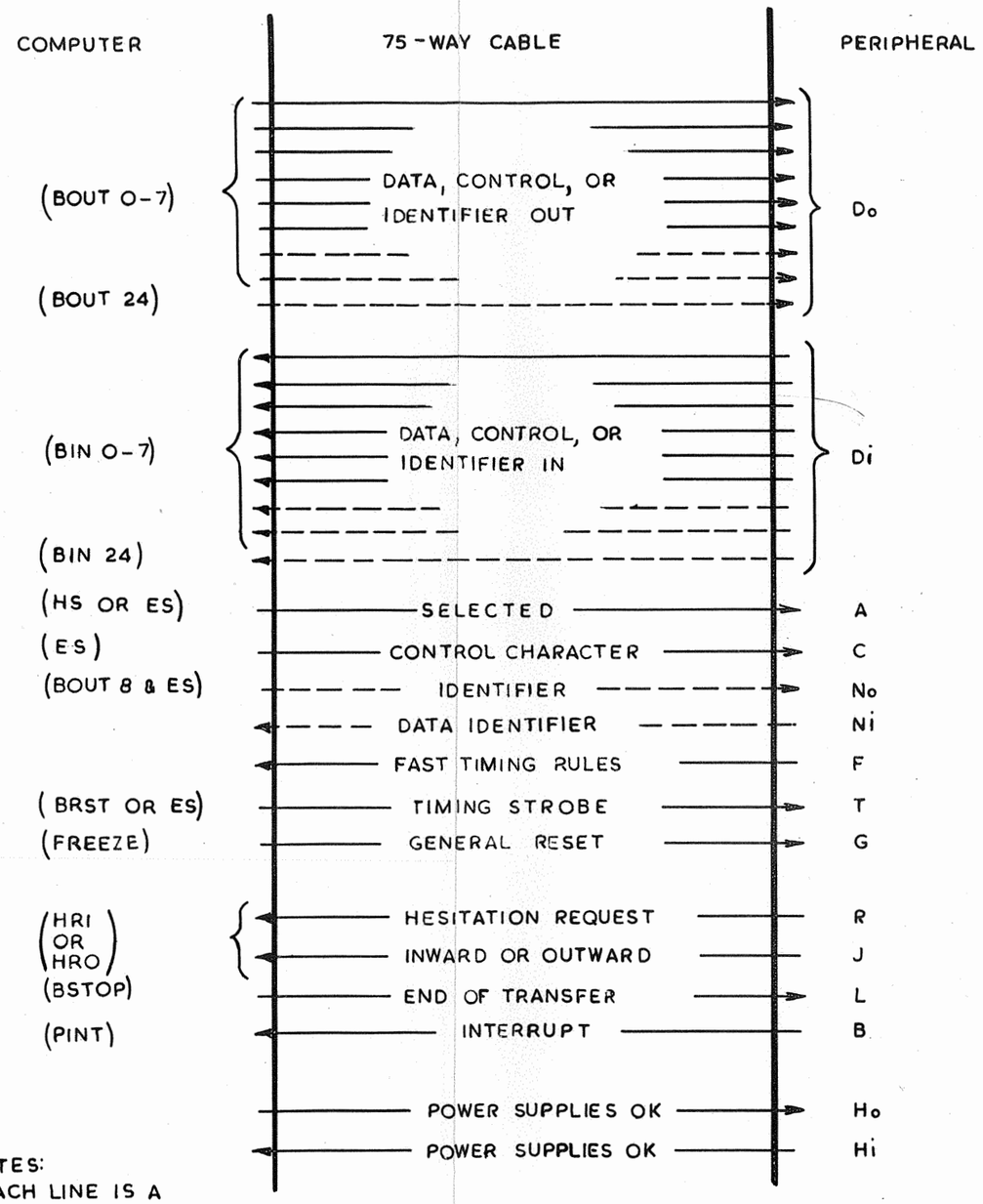


TRANSFER OF ACCUMULATOR X TO PERIPHERAL
SPECIAL REGISTER π DIAGRAM 19



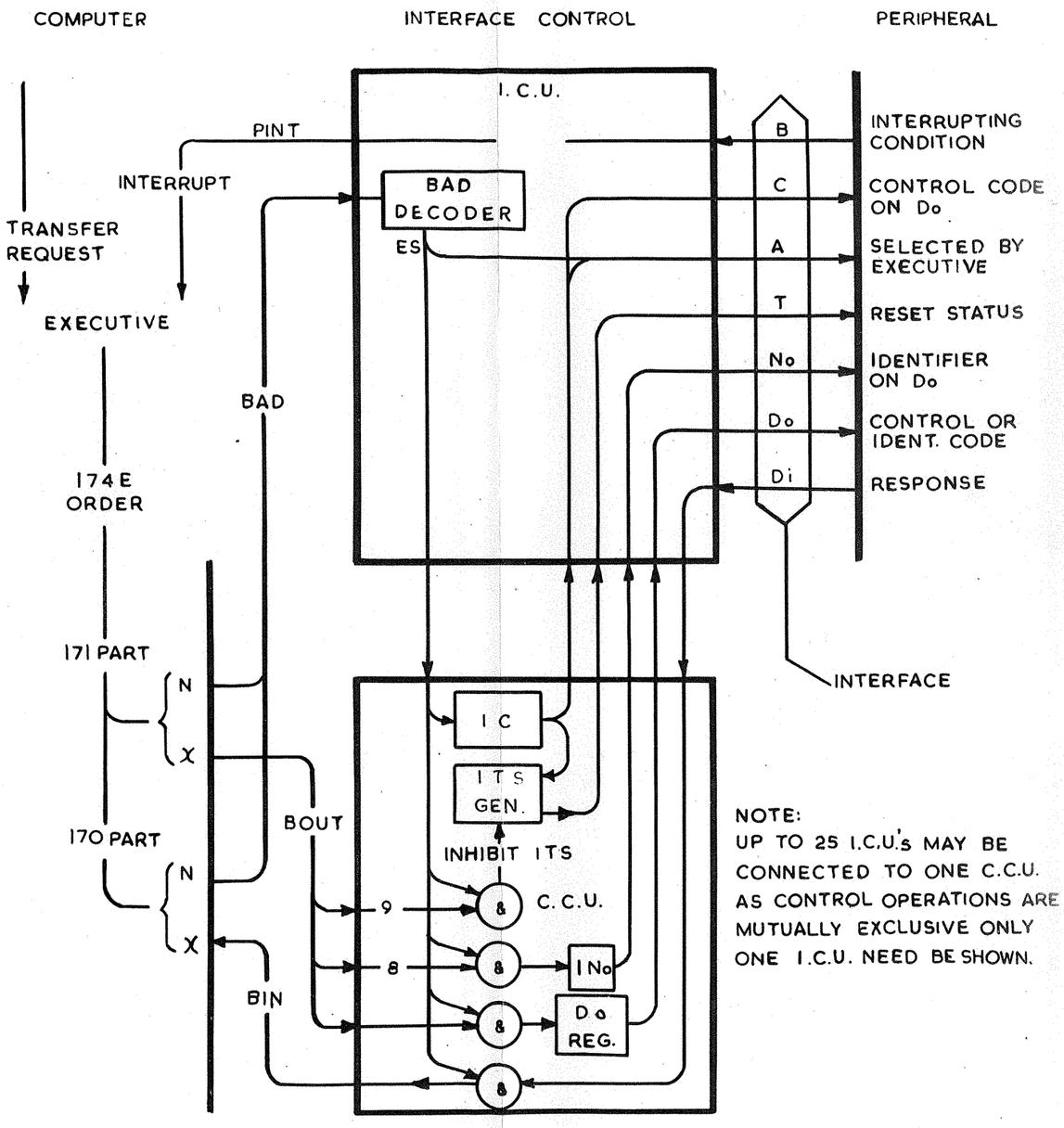
MAGNETIC TAPE CONTROL SYSTEM.

DIAGRAM 20



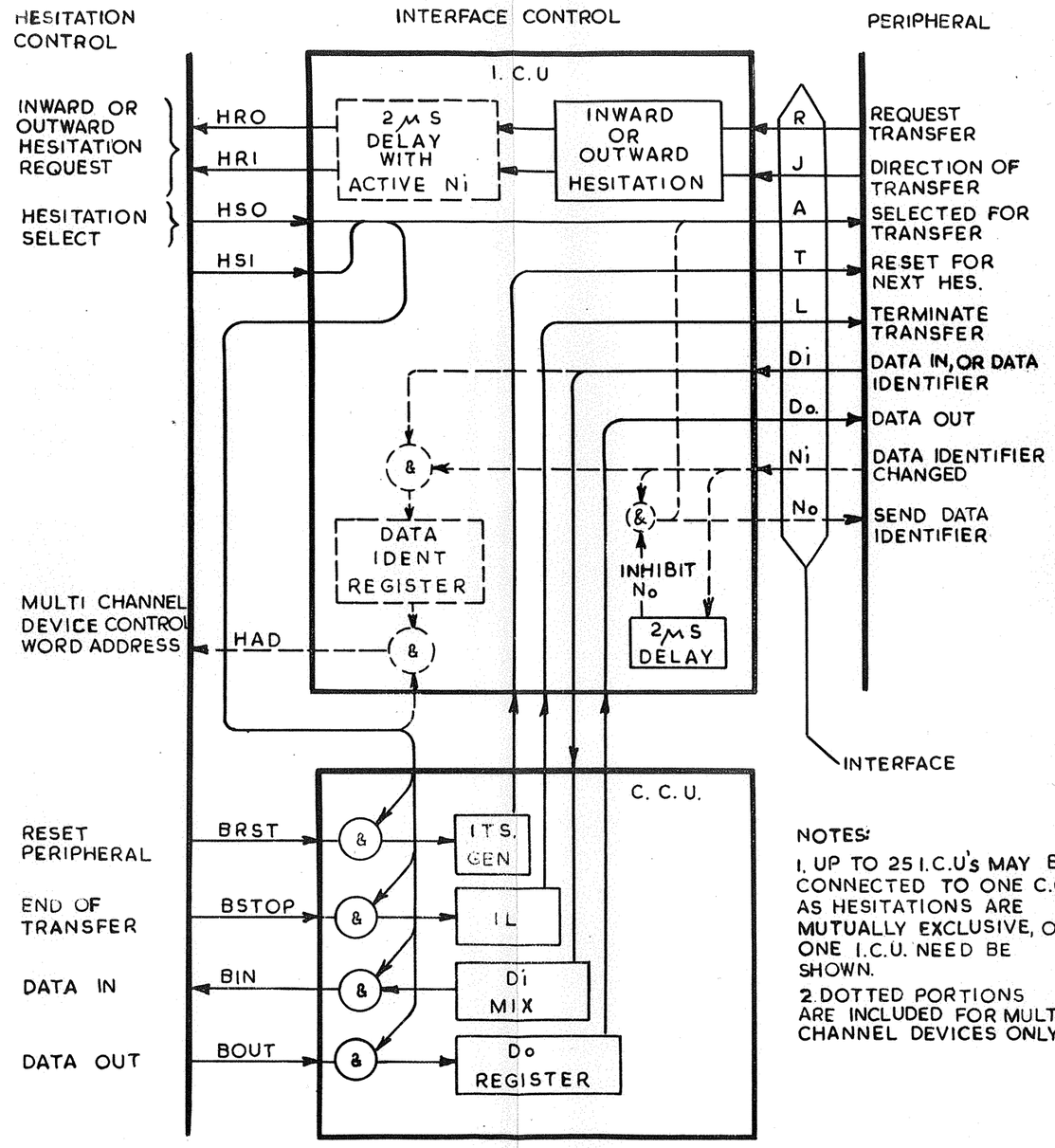
STANDARD INTERFACE LINES (SCHEMATIC)

DIAGRAM 22



TRANSFER OF CONTROL INFORMATION OVER THE STANDARD INTERFACE

DIAGRAM 23

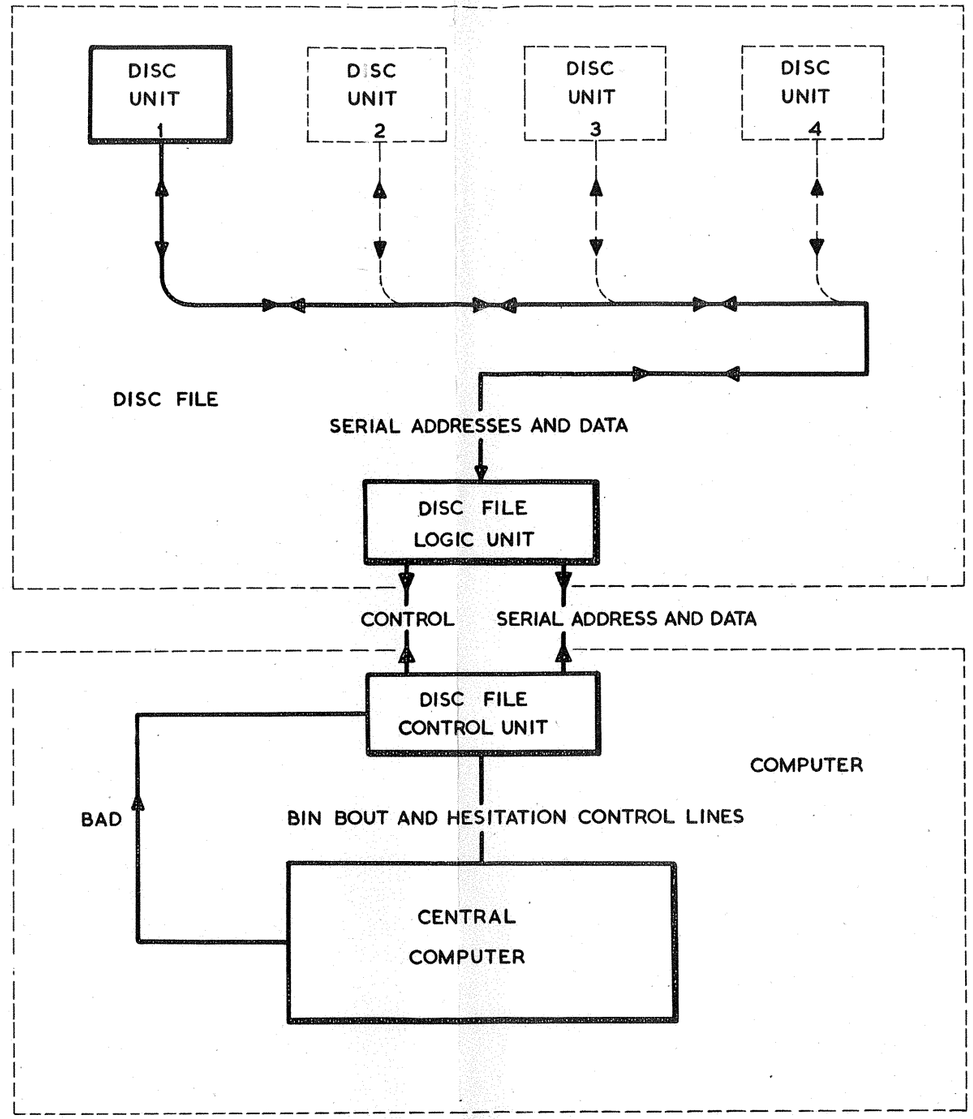


NOTES:

1. UP TO 25 I.C.U's MAY BE CONNECTED TO ONE C.C.U. AS HESITATIONS ARE MUTUALLY EXCLUSIVE, ONLY ONE I.C.U. NEED BE SHOWN.
2. DOTTED PORTIONS ARE INCLUDED FOR MULTI-CHANNEL DEVICES ONLY.

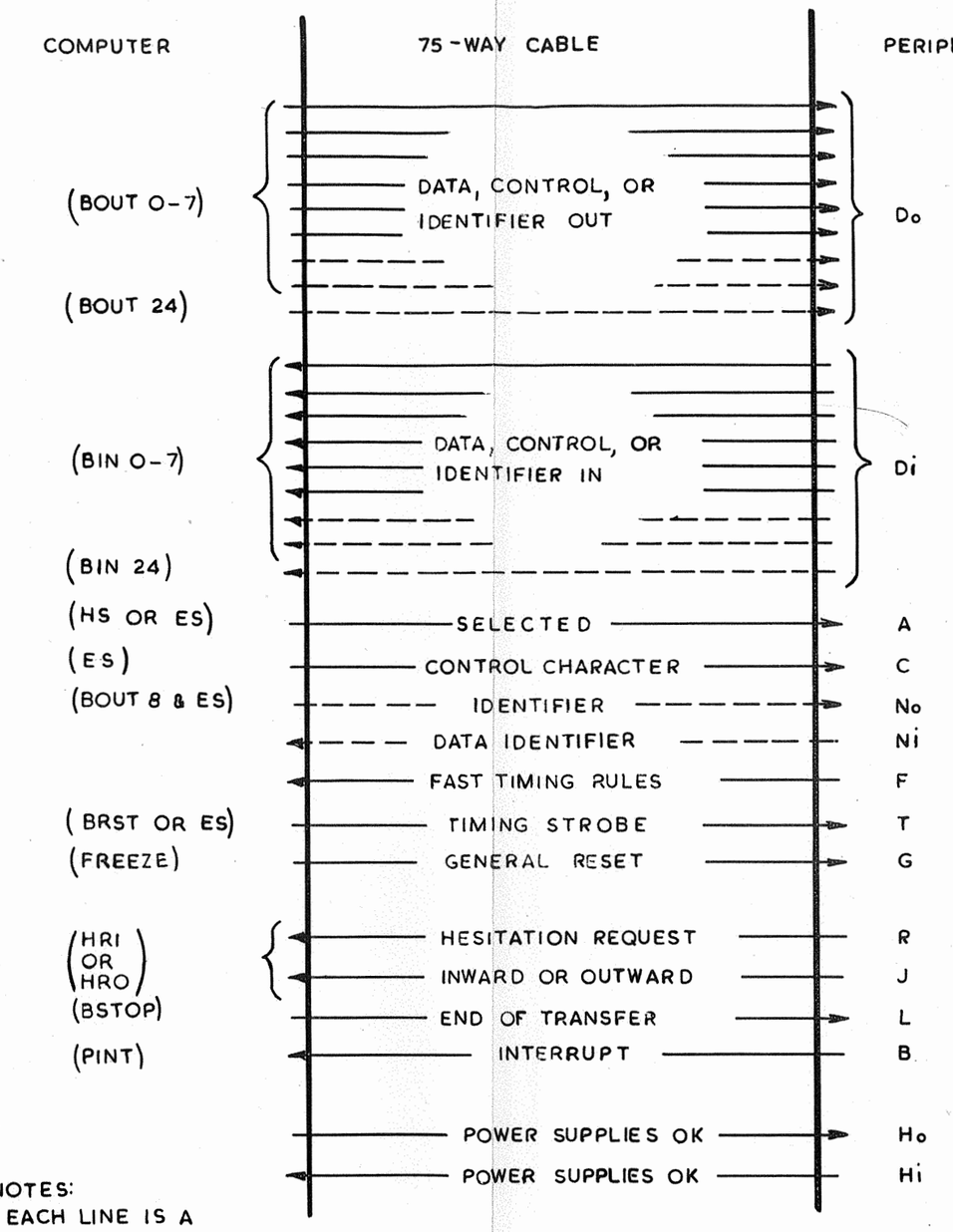
TRANSFER OF DATA OVER THE STANDARD INTERFACE

DIAGRAM 24



DISC FILE CONTROL SYSTEM

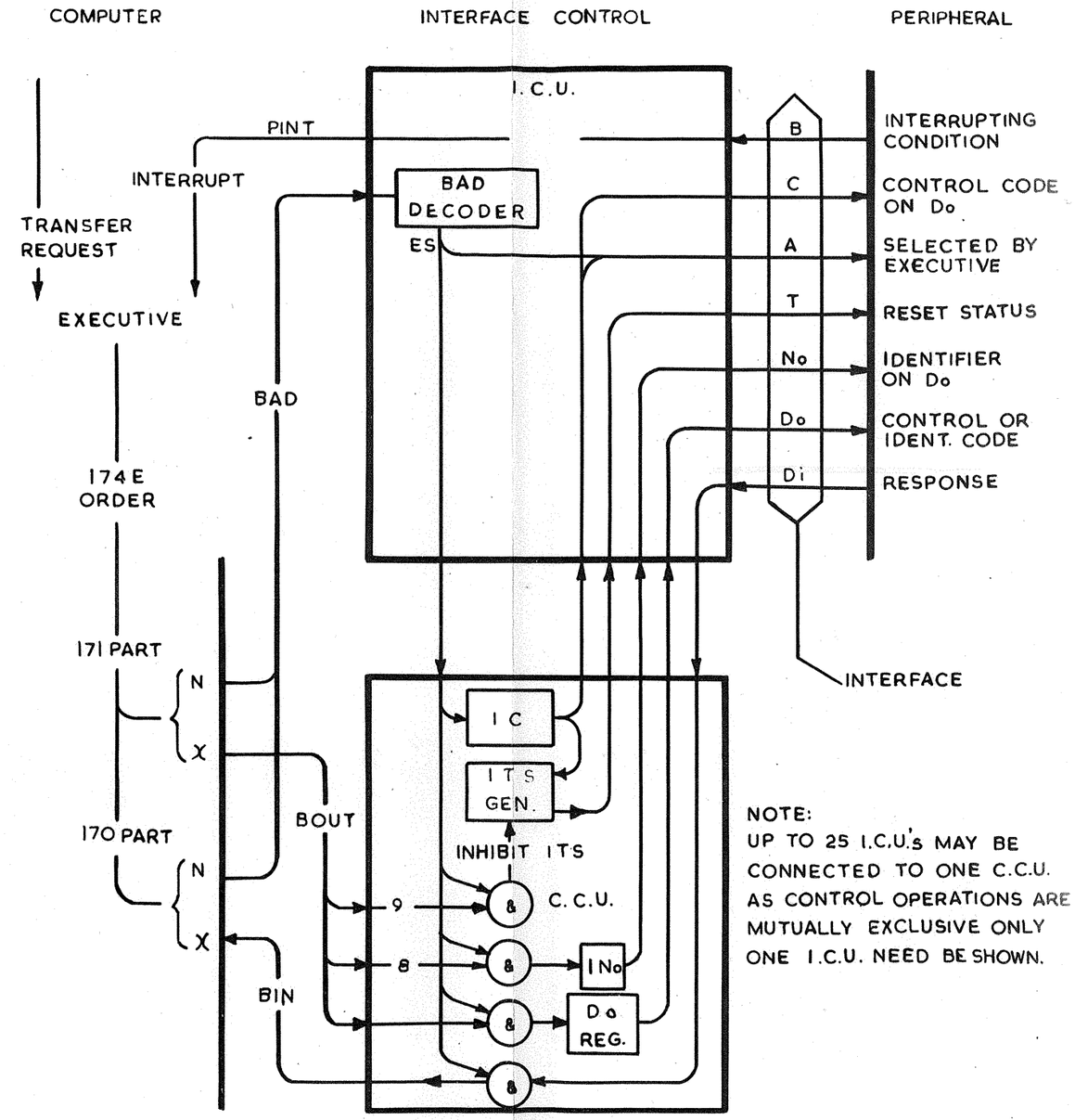
DIAGRAM 21



NOTES:
1. EACH LINE IS A TWISTED PAIR
2. MANDATORY LINES ——
3. LINES NOT NEEDED FOR ALL PERIPHERALS – – – –

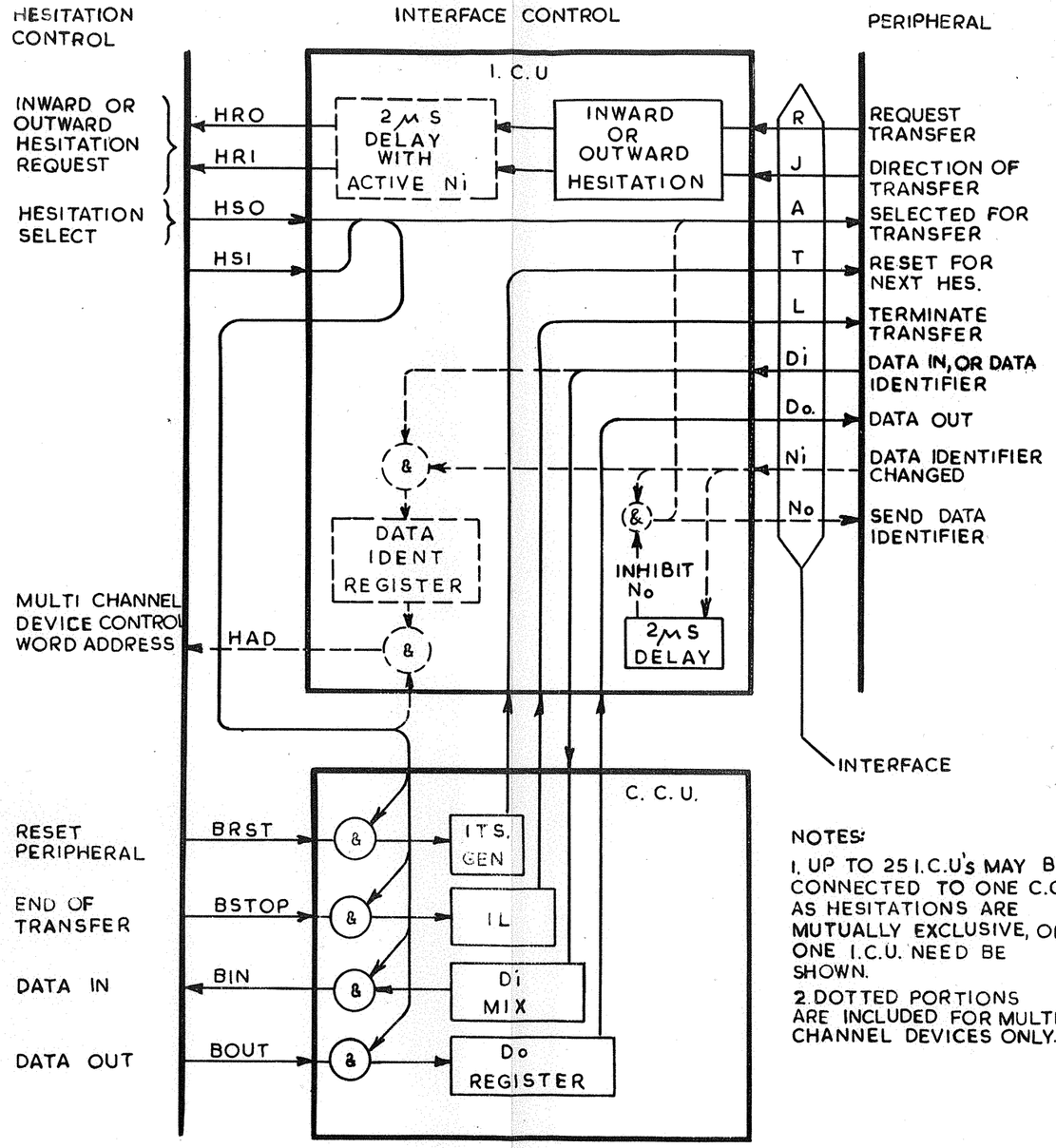
STANDARD INTERFACE LINES (SCHEMATIC)

DIAGRAM 22



TRANSFER OF CONTROL INFORMATION OVER THE STANDARD INTERFACE

DIAGRAM 23

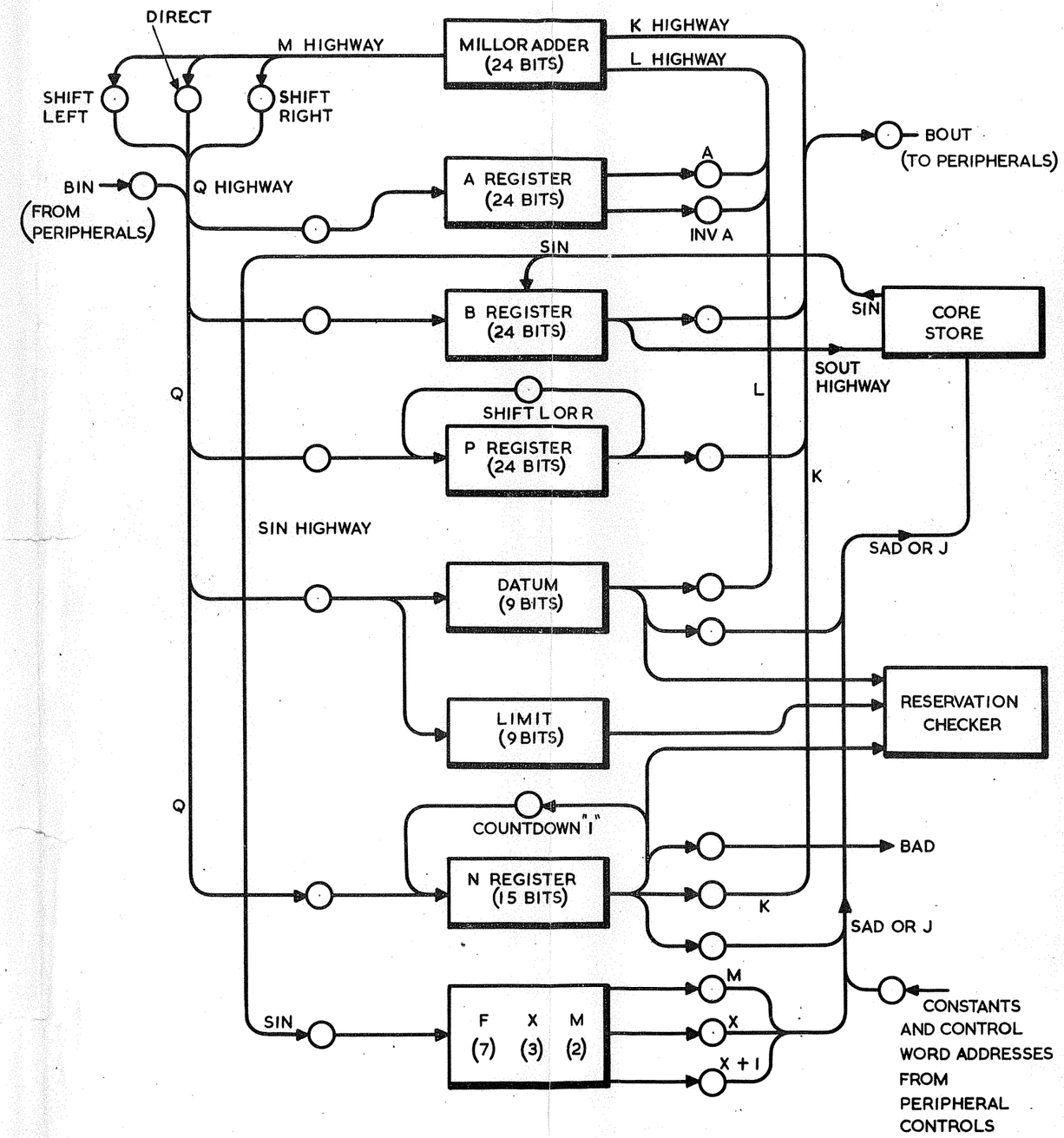


NOTES:

1. UP TO 25 I.C.U's MAY BE CONNECTED TO ONE C.C.U. AS HESITATIONS ARE MUTUALLY EXCLUSIVE, ONLY ONE I.C.U. NEED BE SHOWN.
2. DOTTED PORTIONS ARE INCLUDED FOR MULTI-CHANNEL DEVICES ONLY.

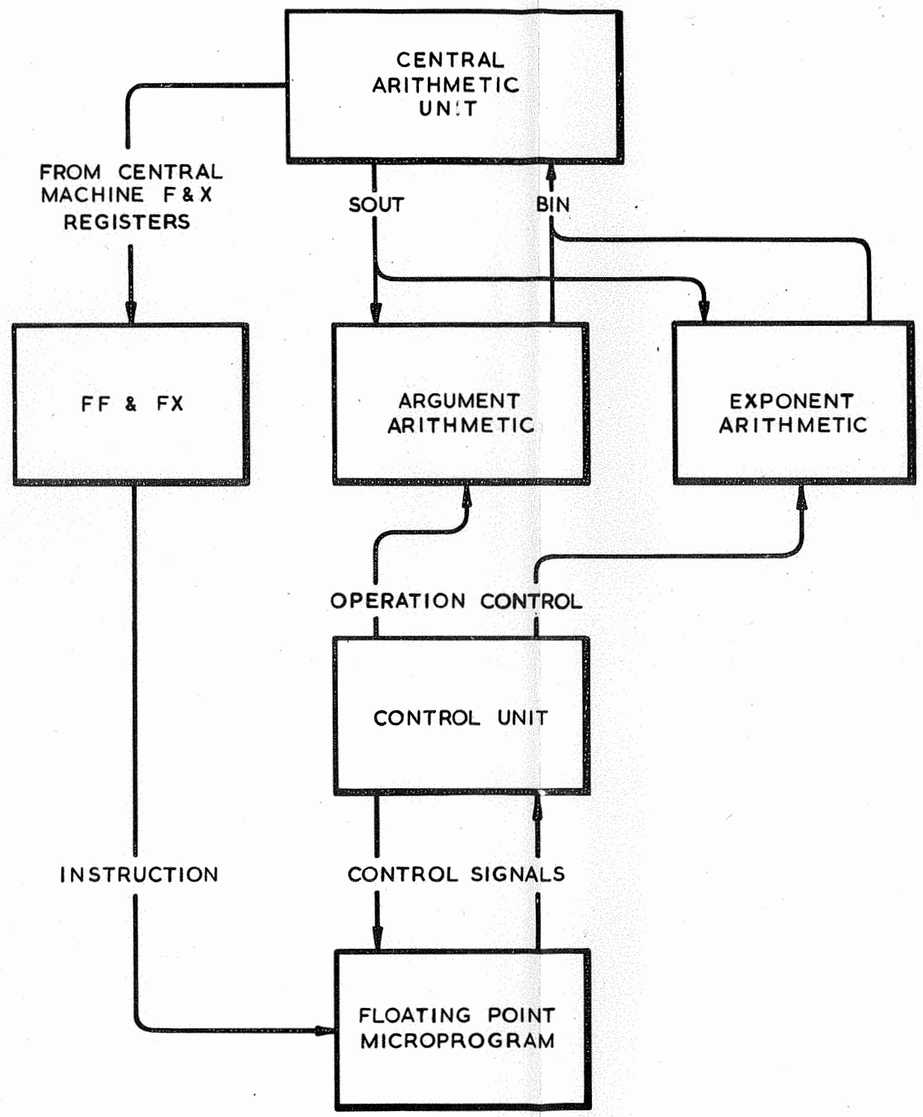
TRANSFER OF DATA OVER THE STANDARD INTERFACE

DIAGRAM 24



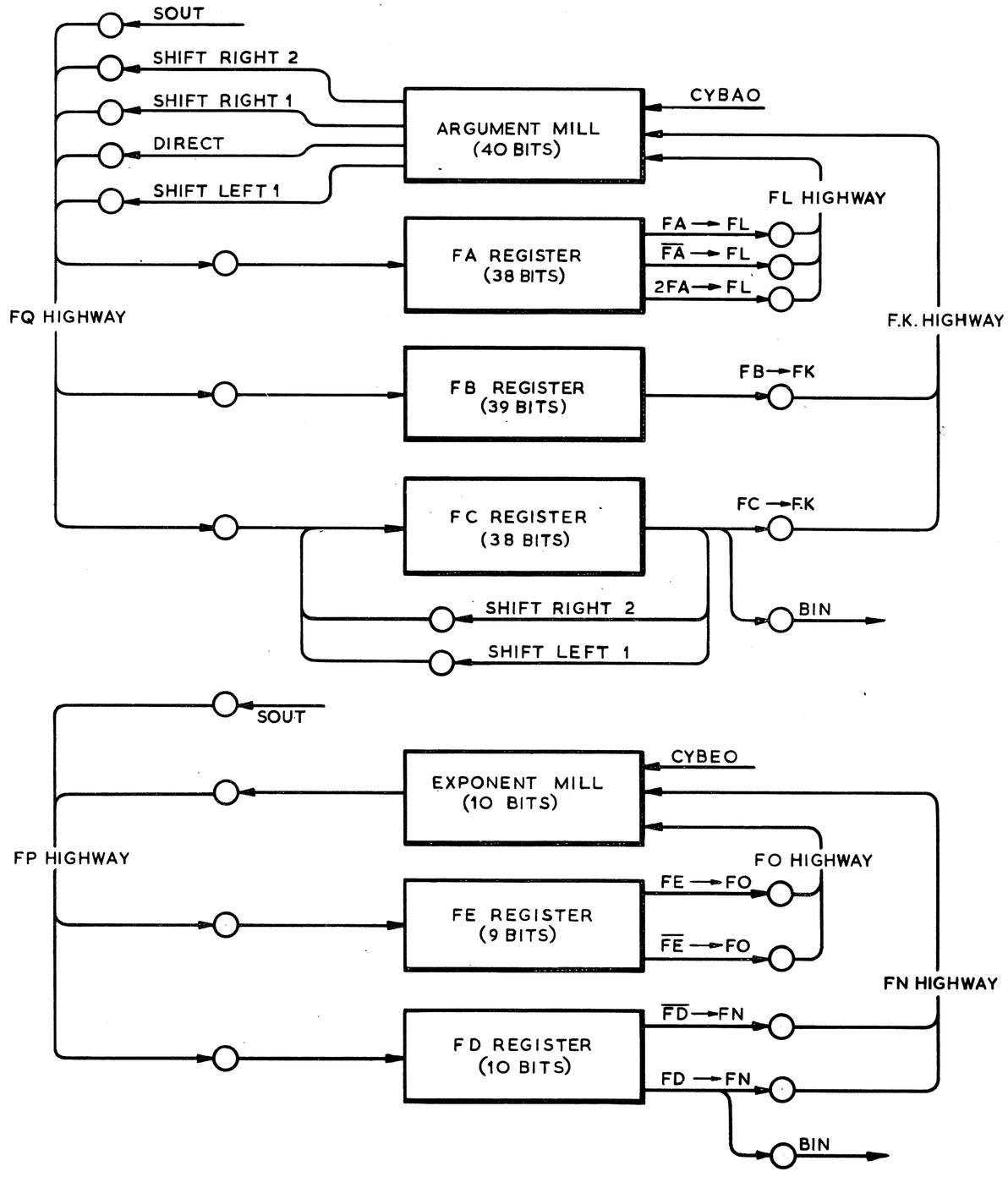
THE ARITHMETIC UNIT

DIAGRAM 25



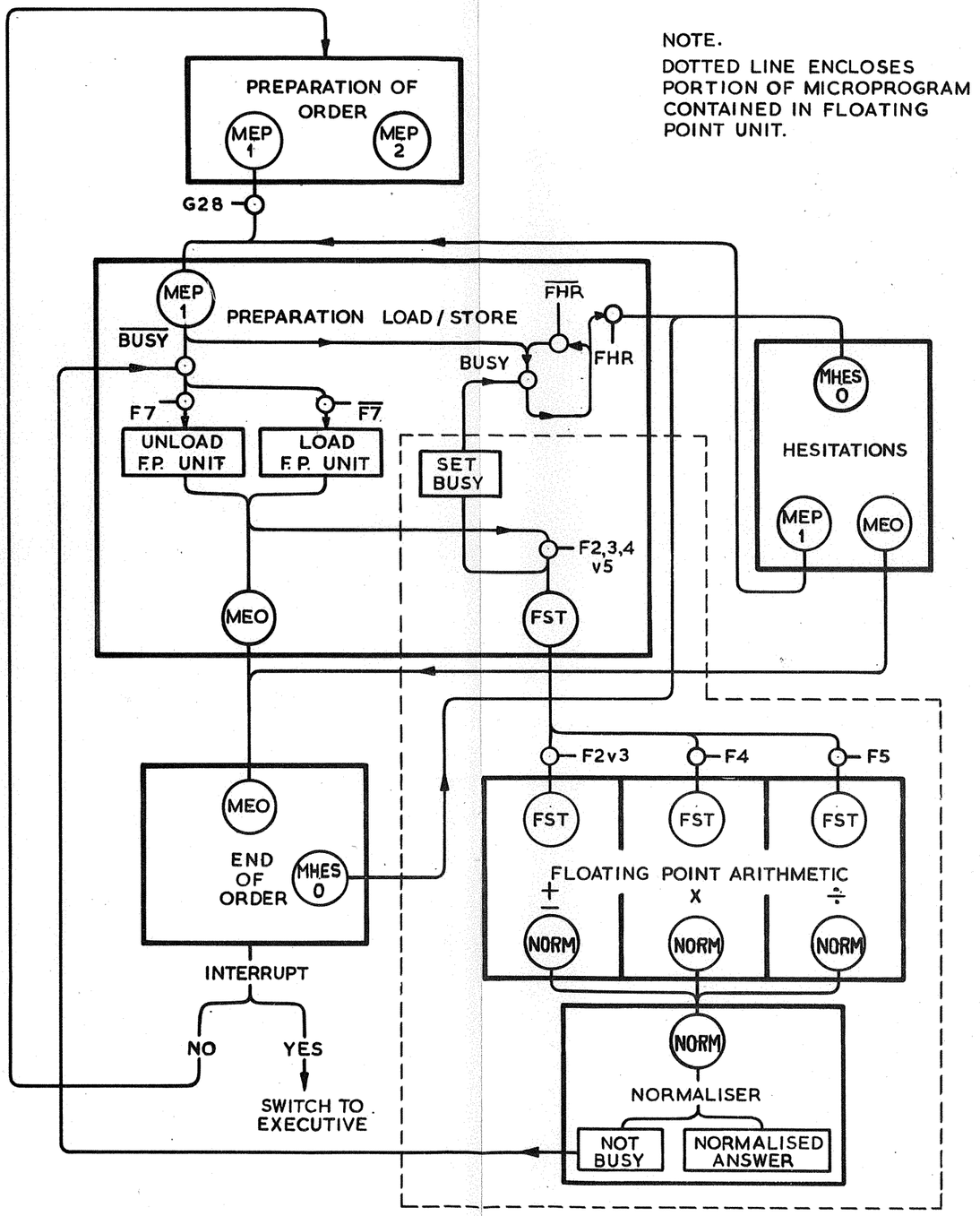
BLOCK DIAGRAM OF FLOATING POINT UNIT

DIAGRAM 26



THE FLOATING POINT UNIT ARGUMENT & EXPONENT ARITHMETIC UNITS.

DIAGRAM 27.



FLOATING POINT UNIT MICROPROGRAM SEQUENCE.

DIAGRAM 28