Issue 3 January 2005

Ferranti Mercury Computer
X3. Instruction Sets and Instruction Times.

Index

D2 Power comparisons
D5 Instruction sets
Description of instructions
References

D2 Power comparisons[5,6].

As Mercury was designed for floating point arithmetic only, comparisons based on fixed point addition are unrepresentative. Such fixed point operations as there were - B-operations - were just 10 digits, not comparable to the 30+ bits of other machines.

	Date	Word length	Floating Add	Floating multiply	Fixed add	On-line store
Mercury	1957	40	180 µS	300 µS	60 µS (10 bit)	5 + 80 K bytes
Mark I	1951	40	40 mS	24 mS	1.8 mS (av)	128 + 1024 bytes
UNIVAC I	1951	84			525 µS	
Mark I Star	1953	40			1.2 mS	416 b + 16 Kb
EDVAC	1953	44			0.86 mS	5 Kb+
IBM 701	1953	35			60 µS	
IBM 704	1954	36	84 µS	192 µS	24 µS	128 Kb + 8 Kb / drum
EDSAC I	1949?	36			1.5 mS	4 Kb
EDSAC II	1957	40	75 µS	200 µS	20 µS	5 Kbytes core
KDF9	1961	48	5 - 10 µS	<14 µS	1 µS	32 Kw core
						Fl/fi * <14 µS; / 28µS
IBM 7030	1961	64	1.5 µS	2.4 µS		768 Kb
(Stretch)			(pipeline rate 64 bit)			
Atlas	1963	48	1.61 µS	5 µS	1.59 µS	192 + 576 Kb

Note:-
1. Bytes of 8 bits as used for store sizes in 2004 is not a good comparison, since the ‘bytes’ used in older machines were not 8 bits. Mercury worked with 10-bit chunks which could hold two 5-bit characters if necessary, alphanumeric characters being of ‘minor’ importance. Atlas worked with 6 bit characters (8 characters in 48 bits - 6 (8 bit) bytes per word is a bit odd).

2. Figures for American machines are not easy to find, and different sources are inconsistent, possibly due to confusion between fixed and floating point, possibly due to pipelining, some giving repetitive result, some arithmetic time, some single time through the pipeline.

D5. Instruction sets.

The instruction set table and a description of all instructions is appended to this section. There are a notional 100 instructions numbered 00 to 99.

Addressing could read <u>M</u>edium words of 20 digits (first half of computational store only), and <u>L</u>ong words of 40 digits. <u>H</u>alf words of 10 digits can also be accessed. Floating point numbers consisted of two words, the first in an even address. One half word is the exponent and three half words the argument (mantissa). The exponent, H_0, was base 2 and the argument was $-1 \leq x < 1$ in H_1 to H_3[1], H_3 being the most significant. The exponent had a range of -256 to +255, the two most significant digits being normally the same for overflow detection. This gives a range of 10^{-77} to 10^{76} with about nine decimal digits accuracy. With two exceptions, floating point calculations were rounded by placing a 1 in the least significant digit[2] after the calculation and again, if necessary, after standardisation (normalisation). Thus in practice the argument was $-1 \leq x < -½$; $½ \leq x < 1$.

Eight B-registers were provided for address modification primarily. B_7 was a special fixed point ‘short accumulator’, usually termed the sac, S. A B-test, Bt, and an S-test, St, register were also provided each containing two bits, the sign (≥ 0, < 0) and result of last B/sac computation equal to 0 or not.

A number has the most significant bit equal to -1 and the rest as a number in the range $½ \leq x < 1$; thus 0.11 is ¾ and 1.11 is -1 + ¾ = -¼. This is one definition of true (twos) complement numbers.

Instructions are single address and are 20 digits long. They consist of

Function - 7 binary digits; B-register - 3 digits; address - 10 digits.

Instructions must be in the first 16 pages of the computational store (i.e. half of it). Half words can only be referred to in this half of the store[2]. Instructions include special functions to manipulate the floating point exponent, and an instruction to detect a pre-shift of more than 31 bits in an add/subtract. When summing series this last enables a jump out of the loop when the new addend is too small to make a difference. There are also instructions to read from the tape reader, write to the tape punch and to the display on the **Control Desk** (not console[4]). Other instruction codes (8x) are used for instructions for card input, output, line printer and magnetic tape operation.

Reading from or writing to the drum took two instructions. The first set the sector address in a sector register, T; the second indicated which page in the computational store to write to or read from. Sectors 0 to 63 on the drum held library routines. Those on sectors 0 and 1 could not be overwritten normally.

The clock rate was 1 Mc/s (Mhz) giving a floating point add time of 180 µS and multiply of 300 µS. Division is by programme (note spelling![4]) and takes 3½ mS. Arithmetic was serial. Organisational (‘B’) instructions took 60 µS[3]. Transfer time for long words was 120 µS and for B-registers, 60 µS.

Issue 3 January 2005

Ferranti Mercury Computer: Description of Instructions

These tables are copied from “A Description of the Ferranti Mercury Computer with Ancillary Equipment”, Appendices 2 and 3, Ferranti List DC30, June 1957, MSIM reference F Series 6 box 26/32, with updates from DC30A, August 1958 (held by Chris Burton) and additional instructions from CS225, Programmers Handbook Nov 1958, MSIM F2 Series 6 Box 4/18.

The ' after a result name indicates the value after the instruction. Note also that ‘digit’ means bit in modern terms. H indicates the address of a half word, L of a long word and n is the value in the address part of the instruction. S is the sac, B_7.

TABLE 1

The Arithmetical Instructions

Function Code	Description	Notation
40	Transfer the long number in the specified address into the accumulator.	A' = L
41	Transfer the long number in the accumulator into the specified address.	L' = A
42(43)	Add (subtract) the long number in the specified address to (from) the accumulator and round off the result.	A' = A ! L
44(45)	As 42(43), without roundoff.	A' = A ! L (unrounded)
50	Replace the long number in the accumulator by the product of itself and the long number in the specified address, and round off the product.	A' = A x L
51	As 50, but change the sign of the product.	A' = - A x L
52	Replace the long number in the accumulator by the (unrounded) most-significant half of the product of itself and the long number in the specified address.	A' = (A x L)m
53	As 52, but also change the sign of the product.	A' = -(A x L)m
54	As 52, but for the least-significant half of the positive product.	A' = (A x L)l
55	As 52, but for the least-significant half of the product, the sign of which has been changed.	A' = -(A x L)l

TABLE 2

Two Jump instructions
(see also Tables 7, 8 and 12)

59	Jump to the specified address.	C' = n
49	Jump to the specified address if the number in the accumulator is positive or zero; if not, obey the next instruction.	A m 0, C' = n

TABLE 3

B-instructions

Function Code	Description	Notation
00	Transfer the integer in the specified address into the specified B-register, and into the B-test register.	B' = Bt' = H
01	Transfer the integer in the specified B-register into the specified address.	H' = B
02(03)	Add (subtract) the integer in the specified address into (from) the specified B-register. Copy the result into the B-test register.	B' = Bt' = B ! H
04	Shift the digits in the specified B-register one place towards the least-significant end. Subtract the integer in the specified address, and copy the result into the B-test register.	B' = Bt' = ½B - H (see Note 1)
05	Collate the two binary numbers in the specified B-register and in the address, digit by digit, placing the result into the specified B-register, and into the B-test register	B' = Bt' = B & H (see Note 2)
06	Perform the operation of non-equivalence on the two binary numbers in the specified B-register and in the address, digit by digit, placing the result into the specified B-register, and into the B-test register.	B' = Bt' = B g H (see Note 3)

Notes on Table 3

1) The notation is suggestive only. If the B-register contains initially an even positive integer, then a shift downwards of one place will halve this number. If the initial number is odd, the “1” digit at the least-significant end will be lost, while if the initial number is negative, the final result will be positive, since the sign digit is not duplicated.

2) The logical operation of collation gives a “1” in the result only in those positions where there is a “1” in both operands. Thus (10110) & (10101) = (10100).

3) The logical operation of non-equivalence give a “1” in the result only in those positions where the numbers in corresponding positions of the operands are different. Thus (10110) g (10101) = (00011).

TABLE 4

B-Instructions (dual form)

Function Code	Description	Notation
10	Transfer the specified integer into the specified B-register and the B-test register.	B' = Bt' = n
12(13)	Add (subtract) the specified integer into (from) the specified B-register. Copy the result into the B-test register.	B' = Bt' = B ! n
14	Shift the digits in the specified B-register one place towards the least-significant end. Subtract the specified integer, and copy the result into the B-test register.	B' = Bt' = ½B - n
15	Collate the number in the specified B-register with the specified integer, placing the result in the specified B-register, and the B-test register.	B' = Bt' = B & n
16	Perform the operation of non-equivalence on the contents of the specified B-register, and the specified integer. Place the result in the specified B-register, and the B-test register.	B' = Bt' = B ɡ n

TABLE 5

Sac Instructions

20	Transfer the integer in the specified address into the sac and the sac-test registers.	S' = St' = H
21	Transfer the integer in the sac into the specified address.	H' = S
22(23)	Add (subtract) the integer in the specified address into (from) sac. Copy the result into the sac-test register.	S' = St' = S ! H
24	Shift the digits in sac one place towards the least-significant end. Subtract the integer in the specified address, and copy the result into the sac-test register.	S' = St' = ½S - H
25	Collate the two binary numbers in sac, and the specified address, placing the result in sac and the sac-test register	S' = St' = S & H
26	Perform the operation of non-equivalence on the two binary numbers in sac and in the specified address, placing the result in sac, and the sac-test register.	S' = St' = S ɡ H

TABLE 6

Sac Instructions (dual form)

Function Code	Description	Notation
30	Transfer the specified integer into sac and the sac-test register.	S' = St' = n
32(33)	Add (subtract) the specified integer into (from) the sac. Copy the result into the sac-test register.	S' = St' = S ! n
34	Shift the digits in sac one place towards the least-significant end. Subtract the specified integer, and copy the result into the sac-test register.	S' = St' = ½S - n
35	Collate the number in sac with the specified integer, placing the result in sac and the sac-test register.	S' = St' = S & n
36	Perform the operation of non-equivalence on the contents of sac and the specified integer, placing the result in sac and the sac-test register.	S' = St' = S g n

TABLE 7

The B-test Instructions

08	Jump to the specified address if the number in the B-test register is not zero.	Bt g 0, C' = n
09	Jump to the specified address if the number in the B-test register is zero or positive.	Bt ≥ 0, C' = n
18	Jump to the specified address if the number in the B-test register is not zero. Add 1 to the specified B-register, and copy the result into the B-test register.	Bt g 0, C' = n B' = Bt' = B + 1

TABLE 8

The Sac-Test Instructions

28	Jump to the specified address if the number in the sac-test register is not zero.	St g 0, C' = n
29	Jump to the specified address if the number in the sac-test register is zero or positive.	St ≥ 0, C' = n
38	Jump to the specified address if the number in the sac-test register is not zero or positive. Add 1 to sac, and copy the result into the sac-test register.	St g 0, C' = n S' = St' = S + 1

TABLE 9

The comparison Instructions

Function Code	Description	Notation
07	Place into the B-test register the difference between the integer in the specified B-register and the integer in the specified address.	Bt' = B - H
17	Place into the B-test register the difference between the integer in the specified B-register and the integer in the address part of the instruction.	Bt' = B - n
27	Place into the sac-test register the difference between the integer In sac and the integer in the specified address.	St' = S - H
37	Place into the sac-test register the difference between the integer in sac and the integer in the address part of the instruction.	St' = S - n

TABLE 10

Instructions for Transfers Between Computing and Backing Stores

67	Transfer the integer specified in the address part of the instruction into the sector register.	T' = n
68	Transfer the sector indicated by the sector register to the page specified by the integer in the address part of the instruction.	P' = D
69	Transfer the page specified by the integer in the address part of the instruction to the sector indicated by the sector, register.	D' = P

TABLE 11

Input/Output Instructions

60	Copy the next character from the input tape into the least-significant half of the specified address, clearing the other half.	H' = t.i.
61	Copy the hand switches into the specified address.	H' = h.s.
62	Punch one tape character from the five least-significant binary digits in the given integer.	to = n
63	Punch one tape character from the five least-significant binary digits in the specified address.	to = H

TABLE 12

Miscellaneous Instructions

Function Code	Description	Notation
46	Add into the accumulator the floating point number whose fractional part is zero and whose exponent is the exponent of the floating point number whose address is given in the address part of the instruction. The result is not standardised.	$A' = A + 0 \times 2^Y$
48	Jump to the specified address if the last accumulator addition or subtraction involved a relative shift of less than 31 binary places.	C' = n if Shift < 31
57	Dummy	
58	Hoot	
64	Display on the monitors (which have to be selected) the specified long word.	Display = L
99	Stop. The machine will proceed to the next instruction when the prepulse button is depressed.	

TABLE 13

Instructions Relating to Accumulator Exponent and B-registers

70	Transfer the augmented exponent into the specified B-register and into the B-test register.	B' = Bt' = Exp + n
71	Transfer the integer in the specified B-register into the exponent of the accumulator.	Exp' = B
72 (73)	Add (subtract) the augmented exponent into (from) the specified B-register. Copy the result into the B-test register.	B' = Bt' = B +/- (Exp + n)
74	Shift the digits in the specified B-register one place towards the least-significant end. Subtract the augmented exponent and copy the result into the B-test register.	B' = Bt' = ½B - (Exp + n)
75	Collate the contents of the specified B-register with the augmented exponent, placing the result in the specified B-register and the B-test register.	B' = Bt' = B & (Exp + n)
76	Perform the operation of non-equivalence on the contents of the specified B-register and the augmented exponent, placing the result in the specified B-register and in the B-test register	B' = Bt' = B (Exp + n)
77	Place into the B-test register the difference between the integer in the specified B-register and the exponent of the accumulator.	Bt' = B - Exp

Note: In all cases except 71, one of the operands is the sum (modulo 1024) of the accumulator exponent and the integer in the address part of the instruction. The sum is referred to as the “augmented exponent” in the above definitions.

Instructions added later (provisional).

Card Input and Output: two buffers each capable of holding 80 columns of 12 rows are provided.

80 Conditioning Type 1. The data on the card is transferred to the buffer. The address part of the instruction indicates whether an exact binary copy of the contents of the computing store/buffer is required, when 80 columns are transferred to 8 short registers row by row; or for disciplined code punching the binary equivalents of the characters in each of the 80 columns are transferred to 80 short registers. The address part also indicates whether card or line printer is used.

81 Read card Type 1. Copies the contents of the input buffer to the page specified by the address part of the instruction and reads the next card to the buffer.

82 Punch card/Line print Type 1 The page specified by the address part of the instruction is copied to the buffer. The exact contents of the buffer store are punched/printed. In the case of the printer, the paper is advanced.

83 Paper throw Type 1. Paper is fed at approx. 10 inches per second to a preset position.

Magnetic Tape backing store. Up to 8 Decks in two groups of four. Blocks of information are addressed sequentially in four-page mode, but can be in Pegasus mode.

86 Mag-operate Type 4. Read/write from/to consecutive long registers of the computing store, beginning at the long register specified (which must be the beginning of a page).
Search. The long register specified must contain the address on the magnetic tape as an unstandardised 40 digit number with exponent 29.

87 Select deck and operation. Type 1. The least significant three digits of the address part specify the deck (0-7) and the most significant three digits the operation, viz.

101	Rewind	001	Search
000	Read from following block	110	Write to following block
110	Write to preceding block	100	Read from preceding block

Apart from rewind, this must be followed by an 86 instruction.

88 TC 1 busy, C’ = n. Control jump if magnetic tape transfer control unit is busy
89 TC 2 busy, C’ = n ditto

Manchester University Graphical Output.

56 G’ = L Type 4. The contents of the third and fourth short registers of the long register are the co-ordinates of a pint displayed on a special CRT (which can be viewed by an operator or photographed).

65 Open Shutter Type 1. Open the shutter of the camera on the graphical output.

66 Close shutter Type 1 Close the shutter of the camera on the graphical output and advance one frame.

ICI Input/output.

Up to seven input units ad seven output units.

90 $S' = I_i$ 5-digit tape character read to sac.

91 $I_o' = S$ Least significant 5 digits of sac written to tape output

Manchester University Magnetic tape input/output.

92 $M_o' = n$ The least significant 5 digits of the address part of the instruction is written to the magnetic tape output

93 H’ = M_i The character under the read head of the magnetic tape input is read into the least significant five digits of the short register of the computing store, clearing the most significant 5 digits. The tape is set in motion.

The Following instructions are found in CS327, Function code sheet, 1962, MSIM F2 Series 6 Box 4/23.
There are three possibilities for each instruction according as the B digit is 0, 1-3, or 4-7

90	S = St’ = Exp + n	I + n	I	I is information from special input channel
91	Exp’ = S + n	O’ = S + n	O’ = S	O is information to special output channel
92	S’ = St’ = S + (Exp + n)	S + (I + n)	S + I	
93	S’ = St’ = S - (Exp + n)	S - (I + n)	S - I	
94	S’ = St’ = ½S - (Exp + n)	½S - (I + n)	½S - I	
95	S’ = St’ = S & (Exp + n)	S & (I + n)	S & I	
96	S’ = St’ = S $\underline{g}$ (Exp + n)	S $\underline{g}$ (I + n)	S $\underline{g}$ I	
97	St’ = S - (Exp + n)	S - (I + n)	S - I	

D3. References.

[1] *An Introduction to the Ferranti Mercury Computer*. Ferranti List DC 22A, July 1957; MSIM reference F2 Series 6 Box 26/17.

[2] *Programming Manual*. Ferranti List CS 158, July 1957; MSIM reference F2 Series 6 Box 18/12. Has an early version of instruction code - see DC 30A for a more up to date one.

[3] *Ferranti Mercury Computer - Questions and Answers*. Ferranti List CS 120a, August 1957; MSIM reference F2 Series 6 Box 4/5.

[4] *Ferranti Mercury Computer - Recommended Terminology*. Ferranti List CS 205; MSIM reference F2 Series 6 Box 4/13.

[5] Lavington, SH; *History of Manchester Computers*. NCC Publications, 1975.

[6] Lavington, SH; *Early British Computers*. Manchester University Press, 1980.