Instruction set for the Ferranti Mark I computer.

Differences with the Ferranti Mark I* are noted at the end of this section.

The information in this section is mostly taken from reference 12 (see section F1X5), supplemented by cross-checking with reference 11.

Contents.

1. Symbol representation, character codes and the 'backwards binary' notation.

In 1949, When Alan Turing devised the programming system to be used with the Manchester University computer, he chose the standard 5-bit teleprinter code (the *International Telegraphy Alphabet* of the 1930s) to represent 5-bit quantities. All software was then written in terms of these 5-bit characters. Furthermore, in order to achieve a direct correspondence between written bit-patterns and the engineers' waveforms on oscilloscopes, bit-patterns were written in 'backwards binary' with the least-significant digit at the *left-hand* end.

The standard 5-bit teleprinter code is normally used in *Letter Shift* mode or in *Figure Shift* mode. Turing used *Letter Shift* mode for all software purposes, *Figure Shift* being reserved for the printing of a program's numerical results. There are further complications. There are two special bit-patterns in the standard teleprinter code that are used to switch equipment between letter-shift and figure-shift, and special characters that are used for layout purposes, for example 'space', linefeed', etc. and which therefore cannot be used to represent alphabetic characters. Turing overcame these difficulties by assigning so-called *stunt* characters to these special bit-patterns in the standard teleprinter code. The stunt characters were chosen so that their printed appearance corresponded to the characters available on a standard typewriter. This smoothed the off-line preparation of software documentation.

To summarise, the special character-codes for the Ferranti Mark I are as follows. Throughout the present document, teleprinter symbols are placed in **courier type-font,** for clarity.

Decimal equivalent of normal binary value	*'Backwards binary' bit-pattern*	*Printed symbol, when teleprinter is in Letter Shift*	*Printed symbol, when teleprinter is in Figure Shift*
0	00000	/	0
1	10000	E	1
2	01000	@	2
3	11000	A	3
4	00100	:	4
5	10100	S	5
6	01100	I	6
7	11100	U	7
8	00010	½	8
9	10010	D	9
10	01010	R	+
11	11010	J	–
12	00110	N	.
13	10110	F	
14	01110	C	
15	11110	K	
16	00001	T	
17	10001	Z	
18	01001	L	
19	11001	W	
20	00101	H	
21	10101	Y	
22	01101	P	
23	11101	Q	
24	00011	O	
25	10011	B	
26	01011	G	
27	11011	"	
28	00111	M	
29	10111	X	
30	01111	V	
31	11111	£	

2. Representation of operand addresses.

Primary storage addresses are in the form: <row, column>, where $0 \geq$ row ≤ 31 and $0 \geq$ column ≤ 31. It is convenient to express store addresses as symbols in the teleprinter code, so as to be compatible with the symbols used for the Ferranti Mark I's instructions. The (64 x 8) lines, each of 20 bits, contained in each of the eight tubes of the primary memory are thus addressed as follows:

Tube 0		*Tube 1*		*…*	*Tube 7*	
//	/E	/@	/A	…	/C	/K
E/	EE	E@	@A	…	EC	EK
.	.	.	.	…	.	.
.	.	.	.	…	.	.
£/	£E	£@	£A	…	£C	£K

A line-pair, or *long line*, make up a 40-bit operand. Lines `E/` and `EE` form a pair, as do `R/` and `J/` and also `BK` and `GK`, etc. The last line of a page is paired with the first line of the same page so that, for example, `£E` and `//` form a pair for tube 0. (This anomaly of considering adjacent operands to lie on the same page was a hang-over from the early prototyping days when programmers were not sure if more than one Williams/Kilburn tube would be available operationally at any one time).

Operand-addressing is to a 20-bit word boundary. Where the operand is a 40-bit quantity, the contents of addresses s and s+1 are loaded into the accumulator. The less-significant half is held in s and the more-significant half in s + 1. It is considered good practice to start 40-bit operands at an even address.

3. Representation of instruction addresses.

Addresses are 10 bits long, so the theoretical upper limit of addressable primary memory is 1K words (lines) of 20 bits each. This would require a physical store of 16 CRTs, each holding 64 lines. Only eight tubes were implemented – (the ninth tube on the Ferranti Mark I* was a special buffer or cache called the *output tube*, used with a fast lineprinter). Whereas *operand* addresses are modulo one tube of primary memory (see above), *control* addresses are modulo the whole of primary memory. The ten bits of the Program Counter are incremented in a straightforward manner so that instruction addresses proceed from top to bottom down the rows of one column, then down the next-most column, etc., until the highest-address in physical memory is reached at which point the sequence wraps around and starts from the lowest address again.

4. Drum secondary storage.

The Ferranti Mark I's magnetic drum has a maximum of 256 tracks of 2560 digits each. In practice, the number of tracks available to programmers was much less than this, so that it is usual to quote the size in terms of 64 tracks, giving 64 x 2560 = 160 K bits or 8K of 20-bit words. (In practice, a 65th line is included in each page when it is transferred down into primary store – see later). The drum is synchronised to the CPU's clock so that, in principle, several drums could be added to the computer. It is believed that, in practice, no Ferranti Mark I or Mark I* system ever had more than one drum attached. The capacity of a Mark I drum, from a users's viewpoint, is equivalent to about 3.2K bytes in modern terms.

Each drum track contains two pages – ie 2 x 64 x 20 = 2560 bits. It takes 36milliseconds for read transfers and 63 msec. for write transfers. Track-selection is electronic for reading (the more frequent operation) and by relay for writing. The arrangements for organising drum transfers are explained later.

5. Programmer-accessible registers and the notation used to describe them.
An 80-bit accumulator is provided, which can be loaded with the contents of four 20-bit lines in memory. When viewed in the Manchester 'backwards binary', the accumulator appears as follows:

less- significant 40 bits	more-significant 40 bits
AL	AM

We use the following abbreviations for registers:

- `A` 80-bit double-length accumulator
- `AL` the less-significant 40 bits of A
- `ALL` the least-significant 20 bits of L
- `ALM` the most-significant 20 bits of L
- `AM` the more-significant 40 bits of A
- `D` 40-bit multiplicand register
- `B` one of 8 modifier (index) registers. B0, by convention, is normally arranged by the programmer to contain zero.
- `QA` the most-significant bit of A
- `QB` the Q digit. This is the most-significant digit of the B line which was last operated upon by one of the B group of instructions.
- `C` 10-bit Control register (Program Counter). C is normally incremented at the *conclusion* of the present instruction, unless the instruction is a control-transfer order for which the jump-to condition has been satisfied.
- `H` the hand-switches on the operator's console
- `P` the 65th line of the CRT specified by the current address. The contents of this line may be taken as the address of this page when the page is held on drum.
- S the address of a 20-bit line in the primary (CRT) memory. When loading a 40-bit quantity into AL, the contents of S goes into ALL and the contents of (S+1) goes into ALM.

We use lower-case letters to indicate 'the contents of'. Thus, d means 'the contents of D'.

Control words, or so-called *key words*, are used for drum transfers and for peripheral transfers. Both transfers are known in the Ferranti Mark I programming manuals as *magnetic instructions.* The word *magnetic* signifies that movement of information to and from the magnetic drum was, historically speaking, perhaps the major operational improvement that distinguished the earlier versions of Manchester University experimental computers from later versions in the period 1948/49. Magnetic instructions are described later.

6. Instruction format and instruction set.
Normal instructions are 20 bits long. The layout is as follows:

10 bits	*3 bits*	*1*	*6 bits*
Address, n	***B***	***Spare***	***Function***
0 9	10 12	13	14 19

In the description below, a prime indicates the new value; lower-case letters represent the contents. Thus, the function:

l′ = l + s

means: 'the new contents of the lower half of the accumulator L become equal to the contents of the line-pair S specified by the address-part of the instruction'.

Unless otherwise indicated, all operands are assumed to be 40-bit quantities. Some operations can either be signed (assuming 2's complement representation) or unsigned, as indicated. The full instruction set for the Ferranti Mark I is given below. It may be inferred that this set was somewhat experimental and that rationalisations took place for the later Ferranti Mark I* machine (see later).

Code	***Backwards binary***	***Function***	***Comment***
//	000000	h is obeyed as a magnetic instruction	H is the pattern set up on the handswitches
/E	010000	s′ = am	Store M
/@	001000	am′ = am + i, where i = 63 if s = 0; else, $2^i \leq s < 2^{i+1}$	Indicates the position of the ms '1' in S.
/A	011000	s′ = am; AM cleared.	Store and clear
/:	000100	s is obeyed as a magnetic instruction	S contains a drum or I/O transfer Control Word
/S	010100	s′ = al	Store Acc
/I	001100	am′ = al; al′ = am	swap
/U	011100	s′ = al; al′ = am; AM cleared	40-bit move and clear
/½	000010	a′ = a − d x s, (unsigned)	Multiply & subtract (unsigned)
/D	010010	a′ = a − d x s, (signed)	Multiply & subtract (signed)
/R	001010	am′ = am + (number of 1s in s)	Sideways add into m ('population count')
/J	011010	am′ = am + s	Add (upper)
/N	000110	a′ = a + d x s, (unsigned)	Multiply and add (unsigned)
/F	010110	a′ = a + d x s, (signed)	Multiply and add (signed)
/C	001110	d′ = s; d treated as an unsigned no.	Load multiplicand (unsigned)
/K	011110	d′ = s; d treated as a signed number	Load multiplicand (signed)
/T	000001	c′ = s if b ≥ 0, else c′ = c + 1	Conditional indirect absolute jump on the last-referenced B line
/Z	010001	s′ = h	Copy the digits set up on

			the handswitches into s
/L	001001	Stop if switch /L is set on the console.	*Known as a 'dummy Stop'*
/W	011001	all = 20 random digits	Hardware random number generator
/H	000101	c′ = s if a ≥ 0, else c′ = c + 1	Conditional indirect absolute jump
/Y	010101	*unassigned*	
/P	001101	c′ = s	Unconditional indirect absolute jump
/Q	011101	c′ = c + s + 1	Unconditional indirect relative jump
/O	000011	c′ = c + s + 1 if b ≥ 0, else c = c + 1	Conditional indirect relative jump on the last-referenced B line
/B	010011	*unassigned*	
/G	001011	Stop if switch /G is set on the console.	*Known as a 'dummy Stop'*
/"	011011	*unassigned*	
/M	000111	c′ = c + s + 1 if a ≥ 0, else c = c + 1	Conditional indirect relative jump
/X	010111	*unassigned*	
/V	001111	Hoot	Pulse the console's Audio amplifier
/£	011111	*unassigned*	
T/	100000	al′ = s; am cleared	(unsigned)
TE	110000	*unassigned*	
T@	101000	all′ = 65th line of tube addressed by S; alm & am are cleared	Load drum addr of page
TA	111000	s′ = al; al is cleared	Store Acc & clear
T:	100100	a′ = 0	Clear acc
TS	110100	*unassigned*	
TI	101100	al′ = s + al, any carry being added into am	Add (unsigned)
TU	111100	*unassigned*	
T½	100010	a′ = s with sign-extn into am	Load acc (signed)
TD	110010	a′ = extended s OR a	Logical OR (signed)
TR	101010	a′ = extended s AND a	Logical AND (signed)
TJ	111010	a′ = extended s NEQ a	Logical NEQ (signed)
TN	100110	a′ = a − extended s	Subtract (signed)
TF	110110	a′ = - extended s	Load acc negatively (signed)
TC	101110	a′ = a + extended s	Signed addition
TK	111110	a′ = 2 x extended s	Arithmetic shift up by One place
TT	100001	b′= s	Load B *(This instruction may itself be modified)*

TZ	110001	s′= b	Store B *(This instruction may itself be modified)*
TL	101001	b′= b - s	B subtract *(This instruction may itself be modified)*
TW	111001	*Same action as TL*	
TH	100101	*unassigned*	
TY	110101	*unassigned*	
TP	101101	*unassigned*	
TQ	111101	*unassigned*	
TO	100011	b′= s	Load B *(This instruction is **not** itself modified)*
TB	110011	s′= b	Store B *(This instruction is **not** itself modified)*
TG	101011	b′= b - s	B subtract *(This instruction is **not** itself modified)*
T″	111011	*Same action as TG*	B subtract *(This instruction is **not** itself modified)*
TM	100111	*unassigned*	*(This instruction, if used, is **not** itself modified)*
TX	110111	*unassigned*	*(This instruction, if used, is **not** itself modified)*
TV	101111	*unassigned*	*(This instruction, if used, is not itself modified)*
T£	111111	*Dummy order*	*(This instruction is not itself modified)*

Note: for instructions TO to T£, no B-modification takes place. It is believed that B modification has no effect on the address-part of the address-less functions such as /W, /G, /V and T:.

7. Peripheral transfer control word: the so-called *magnetic instruction.*

A 20-bit control word is used to specify two classes of data transfers:
(a) between the primary memory (CRT store) and the secondary memory (magnetic drum);
(b) between the primary memory and paper tape input/output equipment.

The general format of the 20-bit control word is:

8 bits	*2*	*5*	*5*
Track number	***Spare***	***Function***	***Tube number***
0 7	8 9	1 0 14	15 19

The function-bits are designated as follows:
Bit 10: if = 0, then the left-half drum track in a two-tube drum transfer goes to the first tube.
If = 1, then the right-half track in a two-tube drum transfer goes to the first tube.
Bit 11: one-tube transfer (=0) or two-tube transfer (=1). For one-tube transfers, bits 15 – 19 give either of the column-numbers of the two columns in the desired tube. For two-tube transfers, the pair of tubes specified must be an ‘odd-plus-even’ pair.
Bits 12 and 13: 00 = read: transfer data, including the 65th line, from drum to CRT;
01 = write: transfer data, excluding 65th line, from CRT to drum;
10 = compare data on specified CRT, including 65th line, with data on the

specified half-track. If data is equivalent, `c' = c + 3;` if there is disagreement, `c' = c + 1` as usual at the end of this instruction. A programmer typically issues this check after a *read* transfer.

11 = compare data as above, but exclude the 65th line. A programmer typically issues this check after a *write* instruction.

Bit 14: if = 0, the control word specifies a drum transfer, as above.

If = 1, the control word specifies an input/output transfer from/to paper tape equipment, etc. In this case, bits 0 to 9 of the control word are ignored. See later for interpretation of the five function-bits for paper tape transfers.

The 65th line in each half-track contains fixed information specific to that track. The contents of a 65th line is in fact the control-word required to read from the half-track into the first tube of primary store. It is arranged as follows:

digits 0 -> 7 contain the number of that track;
digits 10 -> 14 contain 00000, which is the bit-pattern for *read from this track*;
digits 15 -> 19 contain 00000, thus identifying the first tube.

Digits 0 -> 7 therefore give the address of a page of primary storage when that page is stored on the drum. At Manchester University this concept was later to lead to the idea of a page's *Virtual Address* and the hardware-assisted memory-management of virtual memory on the Manchester/Ferranti Atlas computer.

Drum transfers cause a 'hesitation' in the regular rhythm of fetch-execute. In other words, computation is suspended whilst the drum transfer takes place. Writing transfers take about 90 milliseconds; reading and checking transfers each take about 35 milliseconds.

8. Input and output from/to paper tape equipment, etc.

For input/output, the control word has bit 14 = 1. 0 – 9 of the control word are then irrelevant. Only nine of the 16 possible combinations of Bits 10 – 13 have been allocated for the Ferranti Mark I. The allowable values of bits 10 - 13 are as follows:

Bits 10 – 13	***Symbol***	***Meaning***
00011	O	input a character from the paper tape reader – (see note (a))
00001	T	output a character (see note (b))
10011	B	test output (see note (c))
10001	Z	teleprinter space
01001	L	teleprinter carriage return
11001	W	teleprinter line feed
00101	H	switch teleprinter to Figure Shift (see note (d).
10101	Y	switch teleprinter to Letter Shift (see note (d)).
11111	£	no action.

Notes.

(a). During input, a 5-bit character is transferred from the paper tape reader and the tape is moved on to the next character-position. The logical OR is performed between this 5-bit character and the five ms digits of the accumulator, the result being placed in the five ms digits of the accumulator.

(b). When a character is output, the five ms digits of the accumulator may be routed according to the setting of a three-position switch on the operator's console. The three possibilities are:

(i) send the character to the paper tape punch;
(ii) send the character to the teleprinter;
(iii) send the character to both devices.

(c). for (b) above, the last 5-bit character that was sent to the output equipment is ORed with the ms five digits of L and the result is placed in the five ms digits of L. This enables a programmer to check that the correct five digits were last sent to the output devices. However, this instruction does not check that the output equipment itself was working correctly.

(d). The teleprinter remains in either the letter-shift or the figure-shift state until a counter-command is received. In letter-shift, the teleprinter prints the symbols of the Ferranti Mark I's code, namely: `/, E, @, A, … £` (refer to table given earlier). When in figure-shift, the decimal digits 0 – 9 can be printed, along with: `+, -, .` Physically, the teleprinter is a modified standard Creed device.

Once one of the input/output instructions has been initiated, the computer will continue to obey other instructions until a further input or output instruction is encountered. At this point the machine pauses until the first is complete.

9. Operator's console.

The following are the main user-oriented displays (consisting of four smaller screens B, C, A and D, and two larger screens S) and the several switches.

Screen B: 8 B lines
Screen C: Control (PC) and another 20-bit line
Screen A: the 80-bit accumulator
Screen D: the 40-bit multiplicand and two other lines
Screens S: Each of the two S tubes can be switched to display the contents of a tube of the primary memory (CRT store).

The main switches on the console are:

/G, /L: these determine whether each of the `/G` and `/L` instructions cause the computer to halt.

3-way *Print/Punch/Both* switch: used for selecting PTP, teletype, or both.

Tape, Print, and the three-way switch marked *k, P/ a, P/ t, P*: these enable the teletype keyboard to be placed in manual mode so that, for example, a heading or comment can be typed manually on the output page prior to, or perhaps during, the automatic printing of computed results. Similarly, tape can be punched manually.

KAC, KBC, KCC, KMC, KSC: these switches (keys) allow, respectively, the accumulator, the entire B store, the control register (Program Counter), the D lines and the entire primary memory to be cleared (ie set to zero) manually. A further switch called KEC has the same effect as pressing all five of the previous switches. KEC stands for 'Key Everything Clear'.

KLC: line-clear.

Write: this switch enables an operator to inhibit the writing current to the drum, thus inhibiting all write-transfers. When not inhibited (the normal state), then

changes can be made to information on the drum. When writing current is enabled, an indicator light is lit on the console.

The machine can be switched to operate at *single-shot, semi-continuous* or *continuous* rates. *Semi-continuous* implies a slow clock-rate and *continuous* implies the standard full-speed rate.

A row of 20 hand-switches is provided, allowing a 20-bit binary quantity to be set up in preparation for use with either the `/Z` or // instruction. Since the primary memory can be cleared manually and since an instruction of all zeros is interpreted as 'obey the number set on the hand-switches as a magnetic instruction', bootstrapping a program from the drum is a simple matter.

A row of 20 manual instruction switches is provided. These enable an instruction to be set up and obeyed repeatedly, for diagnostic purposes. There are certain restrictions; for example, the manual instruction cannot be modified by the contents of a B line.

20 'Typewriter' buttons. Together with an *erase/insert* switch, these enable digits to be inserted into, or removed from, a specified line of the primary store.

10. Instruction speeds for the Ferranti Mark I.

The digit frequency is 100 KHz, giving a nominal 10 microsecond digit-period. 24 digit-periods (240 microseconds) is known as a *beat.* Computational instructions normally take an integral number of beats, as follows:

Simple arithmetical and logical functions:	5 beats (total 1.2 millisecs.)
Multiplication instructions:	9 beats (total 2.16 millisecs.)
Most other instructions:	4 beats (total 0.96 millisec.)

The `/W` instruction takes 5.8milliseconds to generate a 20-bit random number.

11. Instruction-set differences between the Ferranti Mark I and Mark I*.

The Mark I has 26 instructions that operate, in the broadest sense, on the contents of the accumulator. These include the usual loading, storing, adding, subtracting, multiplying, ANDing, ORing, NEQing, and the following 'specials': sideways add, random number, load 65th line. These instructions operate variously on the whole 80-bit double-length accumulator, the upper 40 bits of acc, the lower 40 bits of acc, etc. Some of the operations are offered in two forms: signed and unsigned.

The Mark I* has only 13 such accumulator instructions, including the following 'specials': sideways add, standardise. There is no logical OR, and no hardware random number generator instruction.

Numbers in the Mark I* are written down and displayed in conventional binary, rather than the 'backwards binary' of the Mark I. Tables giving the character-codes are given in Section F1X3.

The Mark I has 15 unassigned instructions and 8 B instructions, some of which are duplicates and some of which are alternatives which allow the instruction itself to be modified. The Mark I* rationalised this situation by having only two unassigned instructions and only four B instructions, none of which may itself be modified.

Further points of operational detail between the two types of computer are now listed.

(a). When addressing operands in an ascending sequence, the Mark I* increments addresses from page to page throughout the memory, whereas the Mark I treats ascending addresses as modulo the page of the first-occurring address in the sequence.
(b). The Ferranti Mark I's jumps are all indirect, whereas in the Mark I* the jump-to address is given in the instruction itself. Also, the Mark I adds + 1 to the jump-to address, whereas the Mark I * does not.
(c). The Mark I distinguished between unsigned and signed arithmetic operations. In the Mark I*, only signed arithmetic is performed. Both machines use the two's complement representation for negative numbers. In the Mark I, the binary point for signed quantities is one place from the end; in the Mark I* it is *two* places from the left-hand (more-significant) end, so that data is held in fractional form.
(d). The Mark I* only uses the so-called magnetic instruction for drum transfers. Input/output from/to paper tape is performed by the separate instructions `F` and `$` and not (as in the Mark I) by using an appropriately-configured magnetic instruction. The *magnetic instructions* have already been explained.

12. Number representation and the instruction set for the Ferranti Mark I*.
The main operational differences between the Mark I and the Mark I* have been summarised above. Before giving the complete instruction set for the Ferranti Mark I*, it is necessary to give the character code for this machine.

Numbers in the Mark I* are written down and displayed in conventional binary, rather than the 'backwards binary' of the Mark I. The teleprinter symbols corresponding to each five-bit pattern have also been re-arranged, as shown in the Table below.

Decimal equivalent	***Binary***	***Teleprinter symbol***
0	00000	f
1	00001	£
2	00010	l
3	00011	O
4	00100	@
5	00101	:
6	00110	$
7	00111	A
8	01000	B
9	01001	C
10	01010	D
11	01011	E
12	01100	F
13	01101	G

14	01110	H
15	01111	I
16	10000	J
17	10001	K
18	10010	L
19	10011	M
20	10100	N
21	10101	P
22	10110	Q
23	10111	R
24	11000	S
25	11001	T
26	11010	U
27	11011	V
28	11100	W
29	11101	X
30	11110	Y
31	11111	Z

All but two of the symbols in the third column are printed by the teleprinter when in *Letter Shift.* The exceptions are `f` and `l`. These two cannot be printed because the bit-pattern 00000 is the teleprinter code for *Figure Shift* and the bit-pattern 00010 is the teleprinter code for *Letter Shift*. When in *Figure Shift*, a teleprinter can print the decimal digits 0 to 9, certain other symbols such as +, -, %, ?, etc., and can perform the actions *Carriage Return, Line Feed, Space.* Programmers wishing to print the symbols `f` and `l` have to choose alternative representations in *Figure Shift* such as ? and %.

Number representations for both the Mark I and the Mark I* use two's complement representation for negative numbers. In the Mark I*, however, the binary point is *two* places from the left-hand (more-significant) end, so that data is held in fractional form. Here are some simple five-bit illustrative examples.

Decimal number	*representation in the Ferranti Mark I**
+ 0	00.000
+ 0.5	00.100
+ 0.25	00.010
- 2	10.000
- 1	11.000
- 0.5	11.100

Both the Mark I and the Mark I* have similar instruction formats, although the written form appears reversed since we now use normal binary rather than 'backwards binary'. The Mark I* only has five function bits, so that the layout is as follows:

Function		***Spare***		***B***		***Address***	
19	15	14	13	12	10	9	0

When a 40-bit operand is considered, the contents of the first-occurring line-address (the one given in the instruction) goes into the ls end of the 40-bit accumulator and the contents of the second line goes into the ms end of the accumulator.

In the following list of Mark I* instructions, the notation is much the same as before.

Code	***Binary***	***Function***	***Comment***	***Is there a roughly equiv. Mark I instruction?***
f	00000	Stop	Halts the machine; a steady hoot issues from the console's speaker	No (but see /L and /G)
£	00001	b′ = b - s	B subtract. *(This instruction is not itself modified)*	Yes: TG
l	00010	-	(Not to be used; effect undefined)	-
O	00011	b′ = b + s	B addition. *(This instruction is not itself modified)*	No
@	00100	s′ = c	The contents of C are placed in the ls ten digits of s; the ms ten digits of S are cleared.	No
:	00101	s′ = b	B store. *(This instruction is not itself modified)*	Yes: TZ
$	00110	Output b	The ms five bits of b are sent to the output equipment	No (but see the use of a 'magnetic instruction')
A	00111	b′ = s	Load b. *(This instruction is not itself modified)*	Yes: TO
B	01000	Print a line on the Bull lineprinter	The 64 characters to be printed are assumed to have been pre-loaded into the output tube. The instruction then causes the contents of this tube to be sent to the Bull lineprinter.	No. There is no provision for a lineprinter on the Mark I.
C	01001	If a ≥ 0, goto s	Conditional absolute jump	Yes, but see Note (a)
D	01010	If a < 0, goto s	Conditional absolute jump	No
E	01011	c′ = s	Unconditional absolute jump	Yes, but see Note (a)
F	01100	Input	The five-bit character from the paper tape reader are placed in the ls five positions of s. The rest of s is cleared and the tape reader moves on to the next position.	No (but see the use of a 'magnetic instruction')
G	01101	s′ = store switches	Copy the bit-pattern set up on the 20 'store' switches on the console into s	No
H	01110	-	(Not to be used; effect undefined)	-
I	01111	c′ = s if b ≥ 0, else c′ = c + 1	Conditional jump on the ms digit of the last-referenced B line.	Similar: see /T and /O
J	10000	am′ = am + s		Yes: /J
K	10001	am′ = am - s		Similar: see TN
L	10010	am′ = am NEQ s	Logical 'not equivalent to' (exclusive OR)	Similar: TJ
M	10011	s′ = am		Yes: /A

N	10100	al′ = al + s; am cleared	Add	Similar: see TC
P	10101	am′ = s; al cleared	Load 40 bits into upper 40 bits of the double-length accumulator	Similar: see T/
Q	10110	am′ = am AND s	Logical AND	Similar to TR
R	10111	s′ = am; a cleared		Yes: /A
S	11000	am′ = am + (number of 1s in s)	Sideways add into m (‘population count’). I’m unclear whether this is an ‘add’ or just a ‘load’.	Yes: see /R
T		a′ = a − d x s	Multiply & subtract	Yes: /D
U		A shifted up or down by a function of n	Logical shift up or down, according to the value of n when Treated as a signed number	No (but see TK)
V		A standardised	Shift until a ‘1’ appears in the appropriate digit-position	No
W		s is obeyed as a magnetic instruction	Used for reading/writing between the primary memory (CRT) and the secondary memory (magnetic drum).	Yes: /: See also note (c)
X		a′ = a + d x s	Multiply and add	Yes: /F
Y		d′ = s		Yes: /K
Z		Dummy stop	Halt if console switch set	Yes: /L and /G

Notes.

(a). The Mark I’s jumps are all indirect, whereas in the Mark I* the jump-to address is given in the instruction itself. Also, the Mark I adds + 1 to the jump-to address, whereas the Mark I * does not.

(b). The Mark I distinguished between unsigned and signed arithmetic operations. In the Mark I*, only signed arithmetic is performed.

(c). The Mark I* only uses the so-called magnetic instruction for drum transfers. Input/output from/to paper tape is performed by the separate instructions `F` and `$` and not (as in the Mark I) by using an appropriately-configured magnetic instruction.