Printed in Great Britain by
Clifford-Thames Printing Co. Ltd., Shenfield
Published by the City and Guilds of London Institute
76 Portland Place, London, W1N 4AA

City and Guilds of London Institute
INCORPORATED BY ROYAL CHARTER · FOUNDED IN 1878

Mn **319** Certificate in Computer Programming and Information Processing

Revised Mnemonic Code (1968)

76 Portland Place, London, W1N 4AA

319 MNEMONIC CODE (1968)

Note—A revision of the Mnemonic code has been undertaken, following consultations with colleges, in order to ensure that the needs of new equipment installed since the scheme was introduced in 1964 can be met. The revised Mnemonic code follows below.

1 Machine Requirements

The minimum machine configuration necessary for implementation of the Mnemonic code consists of a central processing unit, one reader (paper tape or punched card) and one punch (paper tape or punched card). The central processing unit has a control register, an accumulator and a store. Optional extras are a reader in the other medium, a punch in the other medium, magnetic tapes, discs, drums and line printers.

1.1 Store

The store consists of 1,000 locations, addressed consecutively from 0, 1, 2, 3, . . . 999, or some suitable lower number. With the exception of location 0, each location can hold either one number or one instruction. The first ten locations have "special" applications.

1.1.1 Numbers

Each number will be held to a precision of 7 decimal digits and as far as the student is concerned, the number may be integral, fractional or mixed. The student will consider it a "number machine", and although it will undoubtedly operate in floating point to a precision determined by the implementor, the student will not initially be required to distinguish between fixed and floating point numbers.

1.1.2 Instructions

The instruction word format is:

x x	x	x x x	x
F	Q	n	m

F The function part (2 decimal digits) specifies the operation which takes place between the specified operand in the given address and the contents of the accumulator. In the written form of the instruction (q.v.), the function part will always be 3 or 4 letters.

Q A query digit, which if omitted causes no further action, if present allows tracing of the programme. The operation of this digit is left to the discretion of the implementor.

n The address part refers to any one location in the range 0, . . . 999.

m The modifier digit refers to any one index register in the range 0, 1, . . . 9.

1.1.3 Special Locations

The first ten locations in the store are reserved in the following way:

0 The contents of location 0 will always be zero; hence if no modifier is specified in the instruction, modification by the (0) will take place. 0 is the only location into which information cannot be transferred.

1 This is the address of the accumulator A.

2 } 3 } By convention, these are the normal index registers.

4 The link used in the JSR instruction will be held in this location.
5 A counter used in association with the LOP instruction.
6 Number of places before decimal point } used in association with
7 Number of places after decimal point } CNC instruction
8–9 Additional index registers if provided by implementors.

1.2 Control Register C

The control register is automatically updated by one during execution of an instruction, ready for extraction of the following instruction. Breaks in this sequence only occur when jump instructions are executed.

1.3 Input/Output Equipment

Control of input/output equipment will be determined by individual implementors; for example, the columns on a card which are to be available for data, reference numbers, etc.

1.4 Off-line Equipment

Programs will be prepared, in general, off-line on a paper tape or card punch. In the case of paper tape each instruction will occupy one line, will be opened by its commencing symbol and be terminated by $\triangle$ (space) or crlf. In the case of cards each instruction will occupy one card in fields determined by the implementor. The minimum character set which will have to be available for both program preparation and for off-line printing of results is as follows:

Character	*Numeric Value*	*Character*	*Numeric Value*
Ø	0	Q	26
1	1	R	27
2	2	S	28
3	3	T	29
4	4	U	30
5	5	V	31
6	6	W	32
7	7	X	33
8	8	Y	34
9	9	Z	35
A	10	+	36
B	11	–	37
C	12	.	38
D	13	,	39
E	14	(	40
F	15	)	41
G	16	crlf	42
H	17	lettershift	43
I	18	figureshift	44
J	19	space	45
K	20	erase	46
L	21	=	47
M	22	£	48 (end of block marker)
N	23		
O	24		
P	25		

2 Instruction Set

Numeric Function	*Mnemonic Instruction*	*Operation*	*Remarks*
00	LDA n, m	$(n+(m)) \rightarrow A$	Load operand into cleared accumulator.
01	ADD n, m	$(A)+(n+(m)) \rightarrow A$	Add operand.
02	SUB n, m	$(A)-(n+(m)) \rightarrow A$	Subtract operand.
03	MLT n, m	$(A)\times(n+(m)) \rightarrow A$	Multiply by operand.
04	DIV n, m	$(A)\,/(n+(m)) \rightarrow A$	Divide by operand.
10	LDAN n, m	$n+(m) \rightarrow A$	Load integer n+(m) into cleared accumulator.
11	ADDN n, m	$(A)+n+(m) \rightarrow A$	Add integer.
12	SUBN n, m	$(A)-n+(m) \rightarrow A$	Subtract integer.
13	MLTN n, m	$(A)\times n+(m) \rightarrow A$	Multiply by integer.
14	DIVN n, m	$(A)\,/n+(m) \rightarrow A$	Divide by integer.
20	STA n, m	$(A) \rightarrow n+(m)$	Store (A) without clearing the accumulator.
30	JUN n, m	$n+(m) \rightarrow C$	Jump unconditionally.
31	JEQ n, m	If $(A)=0$ $n+(m) \rightarrow C$	Jump if $(A)=0$.
32	JNE n, m	If $(A)\neq 0$ $n+(m) \rightarrow C$	Jump if $(A)\neq 0$.
33	JLE n, m	If $(A)\leqslant 0$ $n+(m) \rightarrow C$	Jump if $(A)<0$.
34	JGE n, m	If $(A)\geqslant 0$ $n+(m) \rightarrow C$	Jump if $(A)\geqslant 0$.
35	JLT n, m	If $(A)<0$ $n+(m) \rightarrow C$	Jump if $(A)<0$.
36	JGR n, m	If $(A)>0$ $n+(m) \rightarrow C$	Jump if $(A)>0$.
37	JSR n, m	Link$\rightarrow 4$; $n+(m) \rightarrow C$ (Link is the address of the next instruction to be obeyed on return i.e. that immediately following JSR.)	Set link and jump.
38	JST n, m	Wait; $n+(m) \rightarrow C$	Wait; jump when start button operated.
39	LOP n, m	If $(5)>0$ after$(5)-1$, $n+(m) \rightarrow C$	Jump if after subtracting 1 from location 5, $(5)>0$.
40	SQT n, m	$\sqrt{(A)} \rightarrow A$	If $(A)<0$, jump to $n+(m)$.
41	EXP n, m	$\exp(A) \rightarrow A$	If (A) too large, jump to $n+(m)$.
42	LGN n, m	$\ln(A) \rightarrow A$	If $(A)<0$, jump to $n+(m)$.
43	SIN	$\sin(A) \rightarrow A$	(A) in radians.
44	COS	$\cos(A) \rightarrow A$	(A) in radians.
45	ARC	$\arctan(A) \rightarrow A$	
46	ENT	$\text{entier}(A) \rightarrow A$	Integral part of $(A) \rightarrow A$.
50	ARD n, m	Allocate input device n+(m) to program	
51	AWD n, m	Allocate output device n+(m) tc program	

Device Code	Device
0–9	Typewriters
10–19	Paper Tape Readers
20–29	Paper Tape Punches
30–39	Card Readers
40–49	Card Punches
50–59	Magnetic Tape Units
60–69	Disc Drives
70–79	Drums
80–89	Line Printers
90–999	etc. etc.

Numeric Function	*Mnemonic Instruction*		
52	RNA	n, m	Read number to accumulator. Jump to n+(m) if error in number.
53	WNA	n, m	Write number from accumulator, with n integral, and m fractional digits, if n and m are zero, floating point output is implied.
60	RCH	n, m	Read 1 character to store at location n + (m).
61	WCH	n, m	Write 1 character from store at location n + (m).
62	RNB	n, m	Read next block of characters to store beginning at location n + (m).
63	WNB	n, m	Write block of characters from store beginning at location n + (m).
64	WNL	n, m	Write n + (m) new line characters.
65	WSS	n, m	Write n + (m) spaces.
66	CNN	n, m	Convert character string beginning at location n + (m) to number in accumulator.
67	CNC	n, m	Convert number in accumulator to character string beginning at location n + (m). Location 6 contains number of places before point. Location 7 contains number of places after point. If both locations are zero floating point representation is implied.
70	ACB	n, m	Access block n + (m). For direct access devices n + (m) defines the sector address. On other devices the instruction has the effect of skipping n + (m) blocks.
71	BSP	n, m	Backspace device by n + (m) blocks and stop ready to read or write.
72	RWD		Rewind device and stop ready to read or write first block.
99	STOP		Return control to operating system.

3 Input Directives

(STORE n′)	Commence storing program sequentially from location n′.
(WAIT)	Temporary stop during input for change of tapes.
(EXECUTE n′)	Commence execution of program at instruction in location n′.
(TITLE) "String"	The alpha-numeric string immediately following the right bracket will be copied on to output tape.

4 Notes Concerning Instructions

The written form of each instruction will be, in general, as follows: LDA n, m where L is the opening symbol of the instruction, n is the address and m the modifier address. In the description (m) denotes "the contents of location m".

If there is no modification necessary, the ",m" may be omitted and modification by location 0 will then be implied. Specimen instructions are:

LDA 100	(100) unmodified$\rightarrow$A
MLT 200,3	$(A)\times$(200 modified by (3))$\rightarrow$A

The comma is used to separate the addresses n and m.
Note that n should precede m.
The addition of Q (query) to any instruction will cause the query digit to be set during input. Then on execution, when a query digit is encountered, the address of that instruction together with the contents of the accumulator after execution will be monitored giving a program trace. The operations associated with Q will be left to the discretion of the implementor as it has no direct bearing on the use of the code during examinations.

4.1 **Instructions 00 to 04**

All instructions leave the specified store location unchanged.

4.2 **Instructions 10 to 14**

In these instructions the address part n is treated as a positive integer in the range $0 \leqslant n \leqslant 999$.

4.3 **Instruction 20**

This instruction will overwrite any previous contents of the specified location.

4.4 **Instruction 30 to 39**

It is expected that in general, m will be zero. It should be noted that all these instructions cause jumps to actual addresses; there is no symbolic addressing. As the implementation of this scheme will be in floating point, the selection of a suitable "zero" for the JEQ instruction is left to the implementor.
JSR, which is intended only for subroutine entry, will set its link in 4. Hence exit from the subroutine will always be via (4). It is not expected that students will "nest" subroutines, but if the need arises, they can extract (4) and put to safety elsewhere. The link will always hold the current store address plus one, i.e. the one following the JSR instruction.

4.5 **Instructions 40 to 46**

These should be considered as very powerful machine instructions, which calculate the "standard" functions. The addresses are used for escape routes if the arguments are not within the prescribed range. In general these will be at the discretion of the implementor.

4.6 **Instructions 50 to 72**

The first two instructions in this set are for use in conjunction with the generalised input/output instruction 52–65 and 70–72. Once an "allocate" instruction has been given, the device remains allocated, i.e. all subsequent and relevant input/output instructions refer to that device, until a further "allocate" instruction is given.
This facility allows for implementation to be tailored to fit the peripheral device available. There is no requirement to implement those instructions which are not relevant to a given installation.
Characters will be stored as their numeric values in the specified store locations. The organisation of "character" instructions, when dealing with checking or non-checking codes, 5, 6 or 7 track tapes etc. will be left to the implementor.
Numbers should be opened by a decimal digit or sign, have any normal format and be terminated by space or crlf.
Each number as punched should contain no more than 7 decimal digits.
In general a block of characters is taken to be 80 characters for card devices and a string terminated by a "block-end" character for other devices. Shorter blocks for card devices may be used at the implementor's discretion provided the string is terminated by a block-end character.
The characters read from a peripheral device will be placed in successive locations commencing at that specified in the instruction.

5 **Notes Concerning Directives**

The purpose of the directives is to inform the assembler or compiler that it should perform certain functions immediately, i.e. during the input process. They are intended to be self-explanatory although their specific functions will depend on whether an assembler or compiler is being written. This choice is left to the implementor.
Each directive is enclosed in parentheses. The "string" following (TITLE) must commence and terminate with crlf, and the whole of that string will then be copied on to the output tape, preceding any results. Care should be taken not to exceed one line of printing.

6 **Input of Numbers with Program**

Numbers may be stored together with instructions as long as each number is headed by a + or − sign. They may be assembled from any of the usual ways of writing numbers, e.g.

+123
−0.456
+7.89

7 **Input of Characters with Program**

Characters may be stored together with instructions as long as each character is preceded by an = sign. Each character will occupy a separate location. An = sign with nothing following will be construed as a space.