Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: MEMORY TEST COMPUTER: GUIDE TO CODING AND MTC OPERATION CODE

To: MTC Engineers

From: Philip R. Bagley

Date: March 4, 1953: Revised July 3, 1953.

Abstract: This note contains a brief guide to coding for the Memory Test computer and the MTC Operation Code. The present instruction code embodies the following operations:

Decimal Number	Binary Equivalent	Abbreviation	Name
0	00000	ha	halt
1	00001	mh	multiply
2	00010	ds	display
3	00011	id	identity check
4	00100	st	store
6	00110	ra	replace address
8	01000	ca	clear and add
9	01001	ad	add
10	01010	cs	clear and subtract
11	01011	su	subtract
16	10000	tr	transfer (control)
17	10001	tp	transfer (control) & pulse
18	10010	tn	transfer on negative
19	10011	np	transfer on negative & pulse
20	10100	md	memory address display
24	11000	cr	cycle right
25	11001	sr	shift right
28	11100	pr	(cycle &) print
30	11110	ri	read in (& cycle)

SECTION 1. GUIDE TO CODING FOR MTC

COMPUTER PROGRAMS

Program. A program is a sequence of actions by which a computer handles a problem. The process of determining the sequence of actions is known as programming.

Flow Diagram. A flow diagram is a series of statements of what the computer has to do at various stages in a program. Lines of flow indicate how the computer passes from one stage of the program to another.

Coded Program. Programs and flow diagrams are somewhat independent of computer characteristics, but instructions for a computer must be expressed in terms of a code. A set of instructions that will enable a computer to execute a program is called a coded program, and the process of preparing a coded program is known as coding. Individual coded instructions call for specific operations such as add, shift, etc.

COMPUTER COMPONENTS

Registers and words. A register has 16 digit positions, each able to store a one or a zero. A word is a set of 16 digits that may be stored in a register. A word can represent an instruction or a number.

Arithmetic element. Arithmetic operations take place in the arithmetic element, whose main components are three flip-flop registers, the A-Register, the Accumulator, and the B-Register (AR, AC, and BR). The 16 digit positions of AR starting from the left are denoted by AR 0, AR 1 . . . AR 15. The digit positions of AC and BR are denoted in a similar fashion. AR is a buffer register, through which all numbers and instructions are transmitted from and to the various registers and memory units. AC is a register which can perform addition, subtraction, and shifting, and can transmit to storage all or part of a 16-digit word. BR is an extension of AC to the right, but BR can only shift.

Storage. The term "register" by itself refers to the main magnetic core storage, which consists of 1024 registers, each of which is identified by an address. These addresses are 11-digit binary numbers from 1024 through 2047. Panel storage is comprised of 32 toggle switch registers, with addresses 0 through 31. A single flip-flop register (termed the "LR" or "live register") may be substituted for any one or more toggle switch registers simultaneously. 32 registers of plug-board storage will be available later, with addresses 32 through 63.

Input-Output. Information entering the computer is temporarily stored in the B-Register (BR). Information leaves the computer via the AC. The computer regulates the flow of information between the internal storage and AC or BR, and also calls for any necessary manipulation of external units.

Control element. The control element controls the sequence of computer operations and their execution. The control element takes its instructions one at a time from storage, where the instructions are stored as individual words.

Inter-connections. The four main elements of the computer (storage, control, arithmetic, and input-output) are connected to the input and output of the A-Register by a parallel communications system, known as "A-Register In" and A-Register Out" busses.

REPRESENTATION OF INSTRUCTIONS

Operation section. When a word is used to represent an instruction the first (left-hand) 5 digits, or operation section, specify a particular operation in accordance with the operation code.

Address section. The remaining 11 digits, or address section, are interpreted as a number with the binary point at the right-hand end. For the majority of instructions this number is the address of the register whose contents will be used in the operation. In the instructions cr, sr, pr, and ri, the number specifies the extent of a shift and whether BR is to be cleared after the shift.

Example. The instruction ca x has the effect of clearing AC (making all the digits zero) and then copying into AC the word that is in the register whose address is x. If q is a quantity in some register, the operation needed to copy q in AC is not ca q but ca x, where x is the address of the register that contains q.

REPRESENTATION OF NUMBERS

Single-word representations. When a word is used to represent a number the first digit indicates the sign and the remaining 15 are numerical digits. For a positive number the sign digit is zero, and the 15 numerical digits with a binary point at their left specify the magnitude of the number. The negative -y of a positive number y is represented by complementing all the digits, including the sign digit, that would represent y. (The complement is formed by replacing every zero by a one and every one by a zero.) In this way a word can represent any multiple of 2^{-15} from $-1 + 2^{-15}$ to $1 - 2^{-15}$. Neither +1 nor -1 can be represented by a single word. Zero has two representations, either 16 zeros or 16 ones, which are called +0 and -0 respectively.

Overflow--increase of range and precision. With a single-word representation numbers are limited to the range $(-1 + 2^{-15})$ through $(1 - 2^{-15})$. Programs

must be so planned that arithmetic operations will not cause an overflow beyond this range. The range may be extended by assuming a scale factor or by using 2 registers (30 digits, not counting sign) to represent a single number. Overflow will stop the computer in an overflow alarm except where special provision has been made to accommodate the overflow. The overflow alarm may be suppressed by a switch on the computer. At some future date, a special instruction may be available to permit addition involving overflow without giving an overflow alarm.

COMPUTER PROCEDURE

Sequence of operations. After the execution of an instruction the program counter in the control element holds the address of the register from which the next instruction is to be taken. Control calls for this instruction and carries out the specified operation. If the operation is not a transfer of control, the address in the program counter then increases by one so that the next instruction will be taken from the next consecutive register. The "tranfer of control" instructions permit resetting the program to a particular number so the next instruction can be taken from any particular register.

Copy and store instructions. The copying of a word, or part of a word, from one location to another affects only the latter location, whose previous content is lost.

Zero. The number zero, if it results from a sequence of operations the last of which is addition, will be represented as negative zero (1.111 111 111 111 111), except that plus zero added to plus zero gives plus zero. If the last operation is a subtraction, the representation will be positive zero (0.000 000 000 000 000), except that plus zero subtracted from minus zero gives minus zero. The sign of a zero resulting from shifting is the same as the sign of the number before shifting. (An exception may occur with the use of the "cycle right" instructions.)

Manipulation of instructions. Words representing instructions may be handled in the arithmetic element as numbers.

Procedure in the arithmetic element. The execution of an addition includes the process of adding in carries; this process treats all 16 digits as if they were numerical digits, a carry from AC 0 ("end-around carry") being added into AC 15. (This compensation is necessary because of the method of representing negative numbers.) A subtraction is executed by complementing the AC, adding the subtrahend, and complementing the AC again. There is no provision for rounding off.

NOTATION FOR CODING

Addresses. A coded program requires certain registers to be used for specified purposes. The addresses of these registers must be chosen before the program can be run on the computer, but for study purposes this final choice is unnecessary, and the addresses can be indicated by a system of symbols or index numbers.

Writing a coded program. Registers from which control obtains instructions may be called action registers, and should be listed separately from registers containing other information, which may be called data registers. A coded program is written out in two columns: the first contains the index number of each action or data register, and the second column indicates the word that is initially stored in that register. In many cases part or all of a word may be immaterial because the contents of the register in question will be changed during the course of the program. This state of affairs is indicated by two dashes, for example, ca--.

Conventional notation. In order to make a program more readily understandable to others and more easily remembered by the author himself, it is desirable to write short descriptions of the functions served by certain key instructions and groups of instructions. It is also desirable to indicate breaks and confluences in the "flow" of the program and to indicate instructions which are altered or otherwise abnormally used during the program. Some generally

accepted symbols for this purpose are exemplified and described below:

	120	ra	124	
start -->	121	ca	161	initial entry (i.e., start of program)
	122	ra	132	
139 -->	123	ca	181	re-entry point, showing origin of re-entry
	124	su	(182)	address altered by program, initial value shown
	125	sr	16	
	126	tn	128	conditional short break in consecuitvity (note other form below)
	127	ad	140	
	128	ad	133	
	129	st		addresses indicated by arrow (e.g. addresses = 130 in this case), used primarily at early stages of writing
	130	(ca217/cs217)		word altered by program, alternative values shown
	131	tr	78	no break in consecuitvity, despite tn operation, where a closed subroutine is called in.
(122, 167)	132	st	(-)	addresses altered by program, initial value immaterial, locations of altering instructions shown, alternative values not shown.
	133	ca	217	semi-pseudo instruction, serves both as instruction and number
	134	tr	136	short break in consecutivity, used especially where a closed subroutine with program parameters is called for
	135	su	115	
	136	ad	114	
	137	tn	141	conditional break in consecutivity (note short form above)

138	st	114	
139	sp	123	break in consecutivity (note short form above)
140	‖ ra	0	pseudo-instruction, serves only as a number, not as instruction
137, 171→141	st	171	entry point, showing origins of entry

The abbreviations RC, CR. Abbreviations used in referring to the register that contains a certain word or the word in a certain register are

RC . . . = (Address of) register containing....

CR . . . = Contents of register (whose address is)...

The symbol ha x. When an address forms part of an instruction it is represented by the last 11 digits of a word whose first 5 digits specify an operation. An address that is not part of an instruction is represented by the last 11 digits of a word whose first 5 digits are zero, which is equivalent to specifying the operation ha. Thus the word for an unattached address x may be written ha x. It may also be written as +x or as $+x \times 2^{-15}$.

SECTION 2. MTC OPERATION CODE

NOTES ON THE OPERATION CODE

Introduction. Included under each operation are the average time of execution, the function, the contents (if altered) of AC, BR, and register x after the operation, and possible alarms.

Abbreviations. The abbreviations used are the following:

AC = Accumulator

AR = A-Register

BR = B-Register

MM = Magnetic Memory

x = address of a storage register

n = a positive integer

Contents of various registers. The contents of AC, BR, AR, and the register whose address is x are undisturbed unless the contrary is stated.

Alarms. Overflow, programmed and identity check alarms due to programming cannot be caused except as specifically noted.

Execution times. The times given are estimated average times for the execution of single instructions which are stored in MM and which refer to addresses in MM. The estimates are based on a memory cycle of 4.8 microseconds for the combined read-and-write operation. It should be noted that some 8.3 microseconds of the total execution time are required in each case for

"program timing"; i.e., to extract the instruction from memory and find out what it is.

Abbreviation	Name	Number	Binary	Time	
ha -	halt	#0	00000	15.8 microseconds.	ha

Stop the computer. The address section is of no significance.

mh x	multiply	#1	00001	47 microseconds (min) 55 microseconds (max)	mh

Multiply the contents of AC and the contents of register x, leaving a 30-digit product of proper sign in AC and in BR0-14, and leaving a zero in BR 15. The previous contents of BR is lost.

ds x	display	#2	00010	200 microseconds	ds

Display a point on the output oscilloscope; the horizontal deflection is specified by the digits 0-9 of register x and the vertical deflection is specified by the digits 0-9 of AC.

id x	identity check	#3	00011	19.5 microseconds	id

Compare the contents of AC with the contents of register x. If it is not identical, give an "identity check" alarm and prepare to skip the immediately succeeding instruction (by adding 1 to the program counter). The identity check alarm feature may be suppressed by a switch on the Alarm Suppression Panel. After the instruction has been executed, AC will contain 1's in each digit position where the original digit did not agree with the corresponding digit position of register x. All other digit positions in AC will be zero.

st x	store	#4	00100	19.5 microseconds	st

Store in register x a copy of the contents of AC. The previous contents of register x is destroyed.

ra x	replace address	#6	00110	19.5 microseconds	ra

Replace digits 5-15 of register x with a copy of digits 5-15 of AC. Because digits 5-15 of an instruction are termed the "address section", the instruction ra in effect replaces the address section of the contents of register x with a copy of the address section of the instruction contained in AC.

ca x	clear and add	#8	01000	17.5 microseconds	ca

Clear AC (but not BR), and add in a copy of the contents of register x.

ad x add #9 01001 17.5microseconds ad

Add to the contents of AC a copy of the contents of register x. If an overflow occurs, an overflow alarm will be given and the computer will stop. If the result in AC after the execution of this instruction is zero, it will be a negative zero; exception: (+0) + (+0) = +0

cs x clear and subtract #10 01010 17.5 microseconds cs

Clear AC (but not BR), and put in AC the negative of the contents of register x, except that cs RC (+0) will leave +0 in AC.

su x subtract #11 01011 17.5 microseconds

Add to contents of AC the negative of the contents of register x. If an overflow occurs, an overflow alarm will be given and the computer will stop. If the result in AC after the execution of this instruction is zero, it will be positive zero; exception: (-0)-(+0) = -0

tr x transfer #16 10000 15.0 microseconds tr

Prepare to take the next instruction from (i.e. transfer control to) register x.

tp x transfer/pulse #17 10001 15.0 microseconds tp

Prepare to take the next instruction from (i.e., transfer control to) register x. Provide a test pulse.
NOTE: On instructions tp and np, the test pulse is normally used to give a "programmed alarm" and stop the computer. If the "suppress programmed alarm" switch in ON, the alarm will not occur and the test pulse may be used for a scope sync pulse.

tn x transfer on negative #18 10010 15.0 microseconds tn

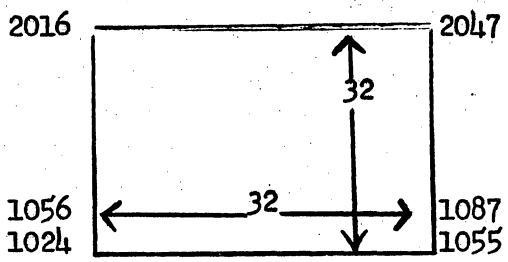
If the contents of AC is negative (has a "1" in digit position 0) prepare to take the next instruction from register x. If the contents of AC is positive, this instruction has no effect.

np x transfer on neg./pulse #19 10011 15.0 microseconds np

If the contents of AC is negative, (has a "1" in digit-position 0) prepare to take the next instruction from (or transfer control to) register x, and provide a test pulse. (See NOTE under tp above). If the contents of AC is positive, this instruction does nothing.

md x memory address display #20 10100 100 microseconds md

Display a point on the memory address scope, the position of the point corresponding to the MM address x. The correspondence is as follows:



cr n cycle right #24 11000 13.1 +1.7N microseconds cr

Shift the entire contents of AC and BR to the right n places. The digits shifted out of BR 15 being carried around into AC 0 so that no digits are lost. If digit 5 of the instruction (i.e., the first digit of the address n) is a "1", BR will be cleared after the shift is executed.

sr n shift right #25 11001 13.1 + 1.7N microseconds sr

Shift the contents of AC and BR, excepting the sign digit, to the right n places, the digits shifted right out of BR 15 being lost. The sign digit will be introduced into every digit position left vacant by the shift. If digit 5 of the instruction (i.e., the first digit of the address n) is a "1", BR will be cleared after the shift is executed.

pr n (cycle &) print #28 11100 150 milliseconds pr

Perform the same function as cr n, then actuate a key on the output printer corresponding to the contents of digits 10-15 of AC.

ri n read in (& cycle) #30 11110 150 milliseconds ri

Read one line of punched paper tape. Put in BR 10-15 the 6-digit binary combination corresponding to the pattern of punches. After the read has been executed, perform the same function as cr n. Since the read-in process reads in only ones and does not clear any digits in BR, BR should, if necessary, be cleared in advance of the read-in instruction.

Future possible instructions. Here is a list of instructions which might be made available at some future date, together with their abbreviations and their probably positions in the instruction code.

5	ao	add one	21	si	select in-out unit
12	am	add magnitude	22	rs	reset
14	lm	logical multiple	23	ro	round off
13	ea	extended add (retains overflow, if any)	27	sl	shift left

Signed: P.R. Bagley
P.R. Bagley

PRB:jmc
cc: N.H. Taylor
R.R. Everett
C.W. Adams

Approved: W. Ogden
W. Ogden

N.H. Taylor