TX-0 COMPUTER
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE 39, MASSACHUSETTS

M-5001-22

A MORE POWERFUL ORDER CODE FOR THE TX-0

CONTENTS PAGE

23 May 1960

A MORE POWERFUL ORDER CODE FOR THE TX-0

A. Introduction

This note describes the present thoughts of the TX-0 staff regarding the eventual order code for the machine. These additions fall into three major classes. The first group is the expansion of the set of addressable commands from six to twenty-four operation codes. These orders are listed in Table 3, and are discussed in section B below, and make use of a single index register XR, to be added to the computer.

The second group involves the expansion of the operate class commands, by making use of unused combinations of bits 4 through 8 of the instruction word. These are described in section C. Finally, there is a group of instructions which will control the operation of the proposed magnetic tape installation and is given in section C-4.

The philosophy of this order code, as will be evident from the discussion, is that operate class commands will be used to perform the majority of arithmetic and logical operations, while the addressable commands allow convenient access to quantities in memory and provide the more important testing and control functions.

The modified system organization of the TX-0 to provide these features is shown in Figure 1. The important additions to the logic are:

1) The index register consisting of a sign and 13 binary digits.
2) A full adding circuit in the program counter for effective address calculation and index addition.
3) A shifting circuit in the operand register which is required for automatic multiplication and division.
4) A bit selector which allows testing and changing of a specified bit of the MBR.
5) A program sense register, PSR, in which each bit may be tested or set by appropriate instructions. The flip flops in this register will be used to indicate conditions in the magnetic tape system, as flags for external events in user's equipment, and for the light pen input.
6) A program indicator register, PIR, might be installed later for storing one-bit indicators which could be set or tested by a program.
7) A step counting circuit which would provide for shifting n places, and counting steps in multiplication or division.
8) The magnetic tape control circuits.

and the necessary additions to the control logic.

B. The Addressable Commands

Figure 2. gives the structure of a TX-0 addressable command. Bits 0 through 4 give the operation code. Since a one in bit 0 and bit 1 defines the operate class instructions, either or both of these bits being zero defines the set of 24 addressable instructions. Bits 5 through 17 form the address α which specifies which of the 2^{13} memory registers is involved.

A complete listing of all addressable commands is given in Table 3. The instructions STO, ADD, TRA and TRN operate exactly as at present, and SOP and LOP are merely new names for the present SLR and LLR. These instructions need not be discussed here.

LDX α Load Index - Load the index register from bit 0 and bits 5 through 17 of register α. The contents of α are unchanged. This instruction places the signed contents of a memory register in the index register provided its magnitude is within the capacity of XR, that is $|C(y)| \leq 2^{13} - 1$.

ADX α Add to Index - The contents of memory register α are added to XR. The contents of α is unchanged.

Ones complement addition is employed in the index register. Therefore, the result of the addition is taken modulo $2^{14} - 1$. The fourteen bit number added consists of bit 0 and bits 5 through 17 of register α.

Examples:

C(CM(α)) 18 bits	C(XR) 14 bits	
	initial	final
+1 = 000001	-0 = 37777	00001 = +1
-4 = 777773	+2 = 00002	37775 = -2
360000	00000	00000 = +0
417777	00000	37777 = -0

STX α Store Index - Store the digits of the index register in the address portion of register α. The sign of XR is ignored. The contents of XR are unchanged. Bits 0 through 4 of register α are unchanged.

The instructions LDX, ADX, and STX may be used to set addresses in instructions without affecting the operation codes.

TZE α Transfer on Zero - If the contents of the accumulator are either plus zero or minus zero, the next instruction is taken from register α. If the accumulator contents are not plus or minus zero, the next instruction in sequence will be executed.

EXT α Extract - Place selected bits from memory register α in a cleared accumulator. The selected bits are those for which the corresponding bits of the operand register are zero. The contents of register α are unchanged.

INS α — **Insert** - Change selected bits in register α to agree with corresponding bits of the accumulator. The selected bits are those for which the corresponding bits of the operand register are zero.

These instructions allow convenient means for examining and modifying a portion of a word in memory. For example, the sequence

```
llr (000077
ext y
add x
ins y
```

will add together the right hand six bits of registers x and y and return the sum to the right six bits of y, without affecting the other bits of register y, and regardless of the contents of the other bits of register x.

If the operand register is clear

```
ext y
```

will load the accumulator with the content of register y.

STO+X	EXT+X	Indexed instructions
ADD+X	INS+X	
SOP+X	TRA+X	
LOP+X		

These instructions are identical to their non-indexed counterparts, but the content of the index register is added to the address (modulo $2^{14} - 1$) before the instruction is executed. The resulting address is known as the <u>effective address</u> of the instruction.

Examples:

instruction address	index register	effective address
(13 bits)	(14 bits)	(13 bits)
00000	-0 = 37777	17777
00067	-0 = 37777	00067
17777	+1 = 00001	00000
00144	-1 = 37776	00143

TIX α Transfer and Index - If the index register contains plus or minus zero, perform the next instruction in sequence without changing the contents of the index register. If the index register contains a positive number, its contents are reduced by one and the next instruction is taken from register α. If the index register contains a non-zero negative number, its contents are increased by one and the next instruction is taken from register y. A zero result will have the same sign as the initial contents of the index register.

The TIX instruction thus combines the operations of address stepping, counting, and testing for completion. The counting in the index register is illustrated by the following examples:

Index Register Contents		Transfer Executed?
Initial	Final	
00046	00045	Yes
37734	37735	Yes
00000	00000	No
00001	00000	Yes
37777	37777	No

Both of the following programs will add the contents of the operand register into the accumulator n times where $n \geq 1$:

```
ldx (-n+1        ldx (+n-1
lad              lad
tix .-1          tix .-1
```

The program below will clear n registers starting at register tab:

```
cla
ldx (+n-1
sto tab+x
tix .-1
```

As another illustration of the extract instruction, the two programs given next will search through a table of n entries beginning at tab for a number which has zeros in bits 0 through 2 and ones in bits 15 through 17. When such a number is found control is transferred to register yes.

```
    llr (077770            llr (077770
    ldx (-n+1              ldx (+n-1
 a, ext tab+n-1+x       a, ext tab+x
    add (-000007           add (-000007

    tze yes                tze yes
    tix a                  tix a
```

Note that in each program the main loop contains four instructions and takes 42 μsecs per step. The first program searches the table with increasing effective address while the second searches with decreasing effective address. Clearly, a similar program could be written to test for any specified bit combination in a table.

TSX α   Transfer and Set Index - The next instruction is taken from register α and the address of the register following the TSX instruction is placed in the index register.

The instruction is used primarily for entering closed subroutines. Two ways of returning control to the main program from a subroutine entered by the TSX instruction are given below.

```
main program                         main program
     .        subroutine                  .          subroutine
     .                                    .
tsx sub  ---> sub,                  tsx sub ---> sub, stx xit
     .   ^         .                      .   ^               .
     .    \        .                      .    \              .
     .     \ tra+x                        .     \             .
                                                 xit, tra
```

The first procedure would be employed if the subroutine itself does not make use of the index register, the second procedure would be used otherwise.

The index register also simplifies the acquisition of program parameters. Suppose the following calling sequence is used for a subroutine which will print the three flexo characters of a TX-0 word:

```
define
     print A
     tsx pr
     A
     terminate
```

The subroutine could be coded as

```
pr, cla
    add +x
    pnt
    pnt
    pna
    tra 1+x
```

ADO α   Add One - Add one to the contents of memory register α and leave the result in the accumulator and register α.

The addition is performed modulo $2^{18} - 1$ in the same manner as the ADD instruction. This instruction is useful for counting and address stepping when the index register is occupied.

EXC α   Execute - Execute the instruction in register α. If the instruction in register y is a transfer command and the transfer is effective, control will proceed to the indicated register. Otherwise, the instruction following the EXC command (or perhaps the second following for a skip group instruction) will be performed next.

As an example, the following calling sequence and subroutine will print N words of flexo characters from registers A through A+N-1.

calling sequence	subroutine
define	sub, stx a
print N,A	lop 1+x
tsx sub	cla,omb,mbx
add A+N+x	a, exc
-N	pnt
terminate	pnt

(cont.)

pnt

tix a

ldx a

tra 1+x

The EXC instruction at location a in the subroutine executes the add instruction in the calling sequence, which is indexed through the list of flexo words.

The execute command may also be used to execute a second EXC command and so on.

EXC α+x Indexed Execute - Execute the instruction in the register whose address is α plus the contents of XR modulo $2^{14} - 1$.

C. Operate Class Commands

The structure of an operate class instruction is shown in Figure 4. Bits zero and one must be one for any operate instruction. These instructions fall into four groups, each of which will be described separately below. Bits 4 through 8 of the instruction word determine which group is active as specified in Chart 5.

1. Micro-programmed Group

A micro-programmed operate instruction has the structure given in Figure 6. Bits 2, 3 and 9 through 17 select members of a set of 16 micro-orders as indicated in Chart 7. The action effected by each of these orders is given in Table 8. The individual orders are executed in the sequence given. Chart 7 also lists the time pulse on which each micro-command is obeyed. In analysing the effect of any combination of micro-orders, it must be remembered that the states of flip flops change slowly compared with the duration of the pulses which set or clear them. Thus, if two micro-orders are given which occur at the same time pulse, the result will be the same as if each order had acted separately on the initial contents of the arithmetic registers. It follows that using the micro orders CMB and MOP together will interchange the contents of the memory buffer and operand registers.

Some examples of the useful combinations which are made possible by this list are the following:

AMB, CLA, CMB, MOP, PAD	Exchange OR and AC
AMB, CLA, CMB, MOP, PAD, CYR	Exchange, then cycle AC right
AMB, CLA, CMB, MOP, CRY	Exchange, then cycle AC left
AMB, ANB, CLA, PAD	OR $\cap$ AC $\rightarrow$ AC ("and" to AC)
AMB, CLA, ANB, MOP, PAD	AC $\rightarrow$ OR, AC $\cap$ OR $\rightarrow$ AC (Exchange then "and" to AC)

Many others are possible.

<u>In-out commands</u> - The input-output commands which may be used together with the above micro-orders are listed in Table 9.* Except where stated otherwise, these orders are executed during an "in-out stop" interval of appropriate duration, which is inserted between cycle zero and cycle one.

<u>Half-word programming</u> - If bits 4 through 8 of an operate instruction correspond to the RHT, LFT or BTH codes shown in Chart 5, the micro-orders specified by bits 9 through 17 will act on the halves of AC and OR as separate nine bit quantities. Specifically, the orders shift right, cycle right and carry will operate as indicated in Figure 10. The codes RHT, LFT and BTH will cause the following action:

RHT -- Perform the micro-commands indicated by bits 9 through 17 on the right nine bits of AC and OR. The left nine bits of AC and OR will be unaffected.

LFT -- Perform the micro-commands indicated by bits 9 through 17 on the left nine bits of the AC and OR. The right nine bits of the AC and OR will be unaffected.

BTH -- Perform the micro-commands indicated by bits 9 through 17 on both halves of AC and OR as <u>separate</u> nine bit quantities.

The following micro-orders will always operate normally when the codes RHT, LFT or BTH are used:

CLL CLR AMB OMB XMB MBX

---

* The reader will observe that the present orders PEN, R1L and R3L do not appear in this list nor in Chart 7. A skip group command will be provided to test the light pen flip flop as described in section C-2.

During installation, an interim PEN order will be provided by one of the spare combinations of bits 4 through 8. Control of the photo-electric tape reader will eventually be included with the magnetic tape control logic (see section C-4). The present mode of operation will be continued for some time after the new system is operating reliably.

2. Skip and Make Group

An operate instruction of the skip and make group has the form shown in Figure 11. These instructions may be used to test and/or set any bit of the accumulator, operand register, or a register of flip flop indicators which will be called the program sense register PSR. A second group of flip flops which will be called the program indicator register, PIR, might be added in the future.

The specific bit or flip flop tested by a skip and make instruction is determined by the bit address together with bit 8 of the instruction as set forth in Table 12a. Testing is mechanized in the machine by skipping the next instruction in the program or not, according to the state of the selected flip-flop. Four variations are possible as determined by the skip indicator bits of the instruction. These are given in Table 12b. The setting or clearing of the selected flip flop is similarly controlled by the two make indicator bits as in Table 12c. It is important to remember that the skipping is controlled by the initial state of the selected flip flop.

To simplify the writing of skip and make instructions we may make the following definitions:

skp = 600	mkc = 140
skz = 200	mkz = 40
skn = 400	mkn = 100

Then the instruction

ska skn 21

will cause the machine to skip the following instruction if bit 17 (decimal) of the accumulator is one. Bit 17 may also be complemented by the same instruction by writing

ska skn mkc 21

Two extra features are provided by instruction bits 2 and 3. First, any skip and make instruction will also clear the accumulator if bit 2 is one. Second, the bit address will be augmented by the contents of the index register if a one is placed in bit 3. The effective bit address is given by bits 13 through 17, plus the contents of the index register taken modulo 2^5.

Eight special orders are included in the skip and make group which allow the entire program sense or program indicator register to be tested as a unit. These instructions are described in Table 13.

The flip flops of the program sense register will be used to inform a program of certain internal and external events.

Included will bethe following:

Arithmetic

OVF overflow indicator (addition)

DVC divide check indicator

Tape Control

EOR end of record

RWC read write check

TCK parity error

EOT end of tape

Other

LP light pen FF

user's equipment flags

3. The Step Command Group

The instructions of the step group are used to cycle or shift the accumulator, operand register, or both, and arbitrary number of places. Also included in this group are multiplication and division commands, a scaling order, and an order which will count the number of ones in an 18-bit word.

The structure of a step group command is given in Figure 14. Bits 12 through 17 specify the number of steps to be executed, and may range from 0 to 63 (decimal). If bit 3 is one, this number will be augmented by the contents of the index register where the sum is taken modulo 2^6. If a one is placed in bit 2, the accumulator will be cleared before the stepped operation is executed. Bits 8 through 11 specify the particular step group command being performed.

Shifting Commands

The seven shifting instructions are stated and explained by Table 15.

Other Step Commands

The remaining step group instructions are given in Table 16 and are discussed below with examples.

CODE	SYMBOL	FUNCTION
606021	MPY	Multiply
606221	DVF	Divide Fractions
606321	DVL	Logical Divide
606500	SCA	Scale
607500	CBT	Count Bits

Table 16. Miscellaneous instructions of the operate step group

Multiply Command

The instruction MPY multiplies the contents of the accumulator by the contents of the operand register, leaving the result in AC and OR. The multiplicand and multiplier are interpreted as a sign and 17 bits in the usual one's complement fashion. The product is a 34-bit signed number and major and minor 17-bit portions will appear in the AC and OR respectively, and will agree in sign. The usual sign rule applies for all cases.

Examples:

Initial		Final	
AC	OR	AC	OR
000021 +17	000101 +65	000000	002121 +1105
477777 -3/4	577777 -1/2	140000 +3/8	000000

Divide Fractions

The instruction DVF takes as dividend the signed 34-bit fraction consisting of the sign and 17 bits of AC with 17 zeros or ones appended to the right. The divisor is the signed 17 bit fraction initially in the OR. The quotient will appear as an appropriately signed 17-bit fraction in the OR and the remainder will be left in the AC with the same sign as the dividend. The divide check indicator, DVC, will be set if the magnitude of the divisor is less than or equal to the magnitude of the dividend. However, the division will be executed even though the results are not meaningful.

Examples:

Initial		Final		Divide Check
AC	OR	AC	OR	
020000 +1/16	200000 +1/2	000000 +0	040000 +1/8	No
577777 -1/2	577777 -1/2	577777 -1/2	377777 ≈ 1	Yes

Logical Divide

The instruction DVL divides the signed integer in the AC by the signed integer in the OR. The appropriately-signed quotient is left in the OR and the remainder is left in the AC with the sign of the dividend. The divide check indicator will be set if the divisor is plus or minus zero.

Examples:

Initial		Final		DVC
AC	OR	AC	OR	
777756 -17	000003 +3	777775 -2	777772 -5	No
000021 +17	777774 -3	000002 +2	777772 -5	No
777777 -0	000000 +0	777777 -0	400000 ≈ -1	Yes

Scale

The instruction SCA shifts the contents of AC and OR left in the manner of LIS (see Table 15) until bit 1 of AC is the opposite of bit 0. The number of shifts necessary to accomplish this is placed in the index register. If the contents of AC is plus or minus zero the divide check indicator is set and no shifts are executed.

Examples:

Initial		Final			DVC
AC	OR	AC	OR	XR	
002000	204000	200041	000000	000006	No
775777	573777	577736	777777	000006	No
000000	220000	000000	220000	000000	Yes

Count Bits

The instruction CBT counts the number of ones in the AC, places this result in the index register, and clears the AC.

4. <u>In-Out Select Group</u>

This group of instructions is provided to control up to three magnetic tape transports and a photo-electric tape reader, although other equipment may be added. The select instruction determines the unit which is to communicate information through the arithmetic element, and establishes the desired mode of operation. Copy instructions control the actual transfer of data. The structure of an in-out select instruction is given in Figure 17. A one in bit 2 will cause the AC to be cleared by the instruction. Placing a 1 in bit 3 will cause the mode code to be augmented by the contents of the index register modulo 2^5. The presently contemplated select orders are listed in Table 18. The magnetic tape orders are discussed in M-5001-17. The operation of the "load" push button is described in section below.

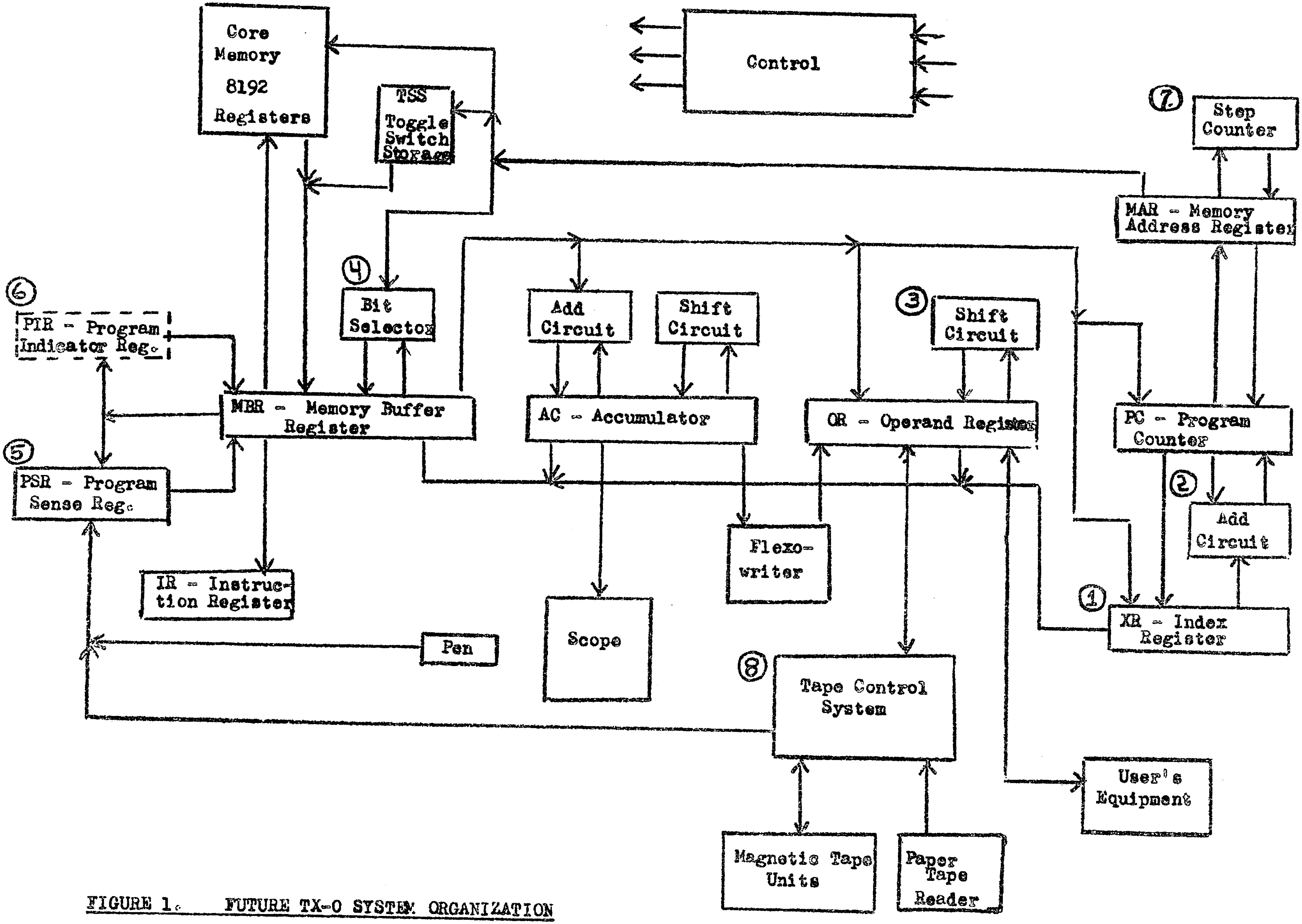


FIGURE 1. FUTURE TX-0 SYSTEM ORGANIZATION

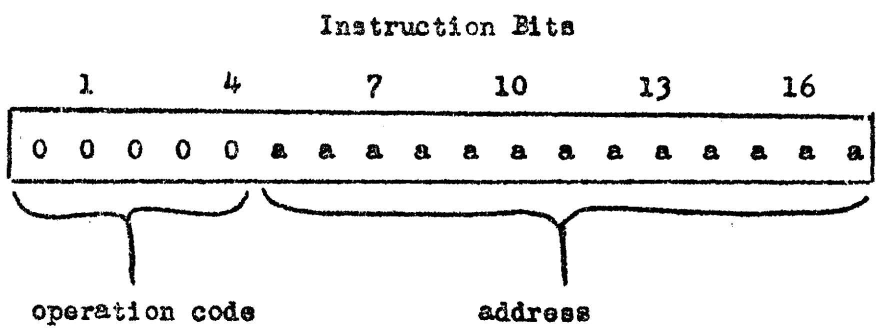


Figure 2. Structure of an addressable instruction

CODE	SYMBOL	NAME	FINAL CONTENTS OF			
			PC*	CM(α)	CM(β)	AC
00000	STO	Store Accumulator	PC + 1	AC		
00001	STO + X	Indexed Store Accumulator	PC + 1	CM	AC	
00010	INS	Insert	PC + 1	$(CM\ (\alpha) \cap OR) \cup (AC \cap \overline{OR})$		$AC \cap \overline{OR}$
00011	INS + X	Indexed Insert	PC + 1		$(CM\ (\beta) \cap OR) \cup (AC \cap \overline{OR})$	$AC \cap \overline{OR}$
00100	SOP	Store Operand	PC + 1	OR		
00101	SOP + X	Indexed Store Operand	PC + 1		OR	
00110	STX	Store Index in Address	PC + 1	$(CM\ (\alpha) \cap 760000) \cup (XR \cap 017777)$		
00111	ADO	Add One	PC + 1	$CM\ (\alpha) + 1^{c}$		$CM\ (\alpha) + 1^{c}$

* Addition in PC is modulo 2^{13}. This quantity will actually appear in MAR rather than PC.

$\cup$ means union of logical sum. $\cap$ means intersect or logical product. ^{c} Modulo $2^{18}-1$.

α = address portion of instruction (bits 5-17). $\beta = [\alpha + XR]$ modulo $2^{14}-1$.

TABLE 3A - ADDRESSABLE INSTRUCTIONS - STORE CLASS

CODE	SYMBOL	NAME	FINAL CONTENTS OF			
			PC	AC	OR	XR
01000	ADD	Add to Accumulator	PC + 1	$AC + CM(\alpha)^{\circ}$		
01001	ADD + X	Indexed Add	PC + 1	$AC + CM(\beta)^{\circ}$		
01010	EXT	Extract	PC + 1	$\overline{OR} \cap CM(\alpha)$		
01011	EXT + X	Indexed Extract	PC + 1	$\overline{OR} \cap CM(\beta)$		
01100	LOP	Load Operand	PC + 1		$CM(\alpha)$	
01101	LOP + X	Indexed Load Operand	PC + 1		$CM(\beta)$	
01110	LDX	Load Index	PC + 1			$CM(\alpha)$ $[0, 5\text{-}17]$
01111	ADX	Add to Index	PC + 1			$XR + \{CM(\alpha)\ [0, 5\text{-}17]\}^{\square}$

$^{\square}$ Modulo $2^{14}-1$

TABLE 3B - ADDRESSABLE INSTRUCTIONS - ADD CLASS

CODE	SYMBOL	NAME	FINAL CONTENTS OF		CONDITION
			PC*	XR	
10000	TRN	Transfer if negative	PC + 1		AC − 0 = 0
			α		AC − 0 = 1
10001	TZE	Transfer if zero	PC + 1		AC $\langle 0 - 17 \rangle \neq \pm 0$
			α		AC $\langle 0 - 17 \rangle = \pm 0$
10010	TSX	Transfer and set index	α	PC + 1	
10011	TIX	Transfer and Index	PC + 1		XR $= \pm 0$
			α	XR $\pm$ 1 □	XR $\neq \pm 0$
10100	TRA	Transfer	α		
10101	TRA + X	Indexed Transfer	β		
10110	EXC	Execute	Execute instruction in register α		
10111	EXC + X	Indexed Execute	Execute instruction in register β		

□ The magnitude of contents of XR is reduced by 1. The result will have the same sign as the initial contents of XR.

TABLE 3C – ADDRESSABLE INSTRUCTIONS – TRANSFER CLASS

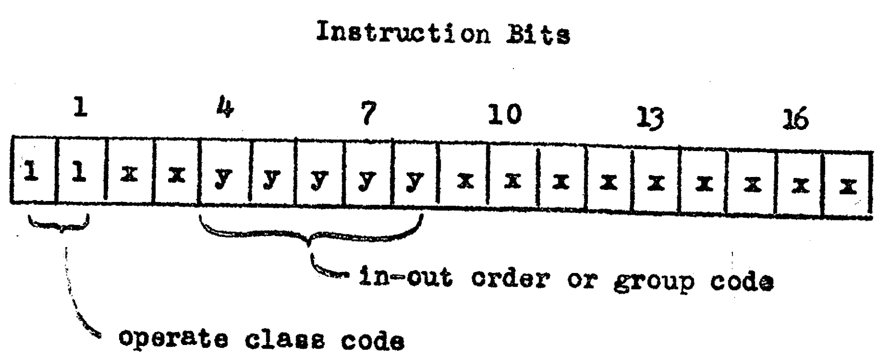


Figure 4. Structure of operate class instructions

INSTRUCTION BITS SIX, SEVEN AND EIGHT

INSTRUCTION BITS FOUR AND FIVE	000	001	010	011	100	101	110	111
00	NOP	TAC	TBR	SPARE	SKIP AND MAKE GROUP		STEP COMMAND GROUP	
01	EX0	EX1	EX2	EX3	EX4	EX5	EX6	EX7
10	CPY	SPARE	DIS	SPARE	PRT	SPARE	P6H	P7H
11	HLT	LFT	RHT	BTH	IN-OUT SELECT GROUP	← SPARE →		

MICRO-PROGRAMMED GROUP

CHART 5 IN-OUT ORDERS AND SPECIAL OPERATE GROUPS

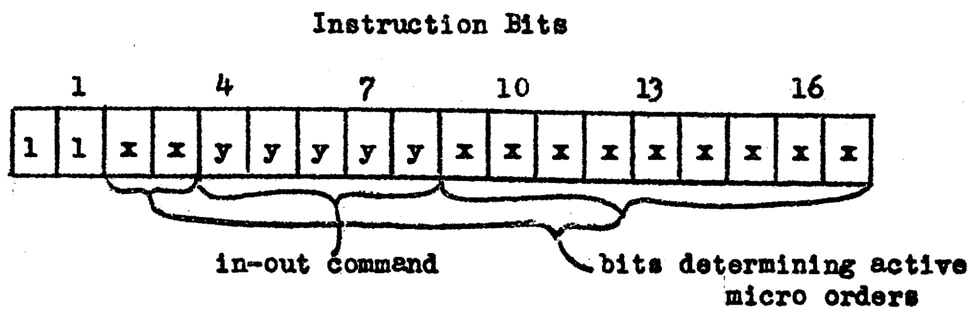
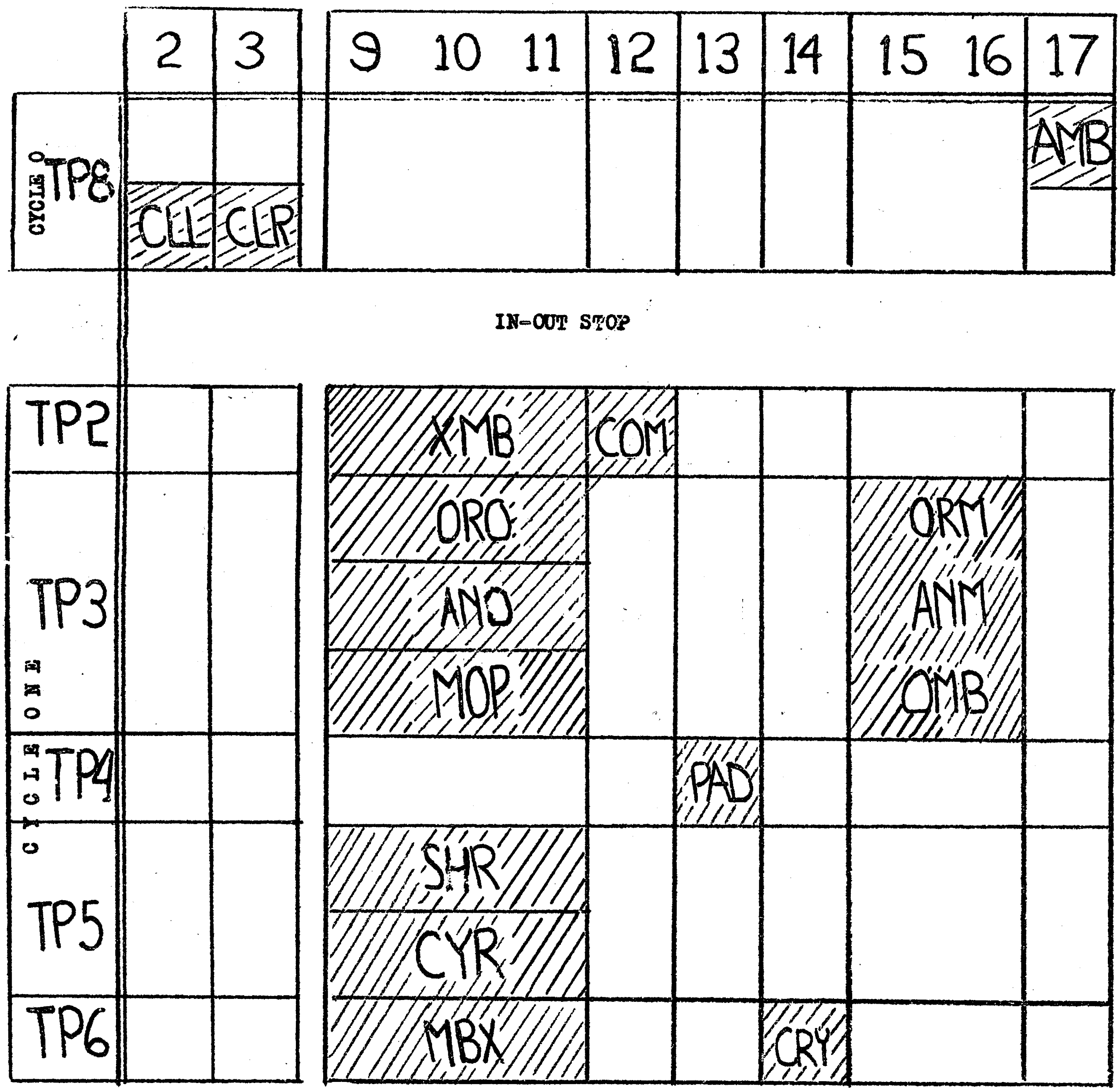


Figure 6. Structure of a micro programmed order

INSTRUCTION WORD BITS



//// bit must be one. //// bit must be zero.

Digits (9, 10, 11) and (15, 16) are decoded together.

<u>CHART 7</u> <u>SEQUENCE AND CODE BITS FOR OPERATE MICRO ORDERS</u>

CODE	SYMBOL	FUNCTION	
		The MBR is initially clear.	
600001	AMB	transfer AC contents to MBR (occurs before CLL or CLR	
700000	CLL	clear left 9 bits of AC	
640000	CLR	clear right 9 bits of AC	
600040	COM	complement AC	
600500	XMB	Transfer contents of XR to MBR (logical sum). The sign of XR, bit 4, will be the input for bits 0 - 4 of MBR.	
600200	MOP	transfer contents of MBR to OR	(a)
600300	ANO	AND the contents of MBR and OR and leave in OR	(a)
600100	ORO	OR (inclusive) the contents of MBR and OR and leave in OR	(a)
600002	OMB	transfer contents of OR to MBR	(b)
600006	ANM	AND the contents of MBR and OR and leave in MBR	(b)
600004	ORM	OR the contents of MBR and OR and leave in MBR	(b)
		Note: Any member of group (a) above may be used simultaneously with any member of group (b).	
600020	PAD	partial add MBR to AC (for each MBR one, complement the corresponding AC bit)	
600400	SHR	shift AC contents right one binary position (bit 0 is unchanged, bit 17 is lost)	
600600	CYR	cycle AC contents right one binary position (AC bit 17 goes to AC bit 0)	
600010	CRY	PAD plus carry completes a full add. Carry is also useful by itself. (A carry digit is a ONE if in the next least significant digit either AC = 0 and MBR = 1, or AC = 1 and carry digit = 1. The carry digits so determined are partial added to the AC by CRY.)	
600700	MBX	Place the contents of bits 0 and 5 through 17 of MBR in bits 4 - 17 of XR.	

TABLE 8 ACTION OF OPERATE MICRO ORDERS

SYMBOL	CODE	NAME	FUNCTION
DIS	622000	Display	Display a point on the CRT with x and y coordinates specified by the left and right halves of the AC, respectively, interpreted as one's complement nine bit numbers.
PRT	624000	Print	Print the flexowriter character specified by bits 2, 5, 8, 11, 14 and 17 of the AC.
P6H	626000	Punch 6 holes	Punch holes 1 - 6 on paper tape according to bits 2, 5, 8, 11, 14 and 17 of the AC. Do not punch hole 7.
P7H	627000	Punch 7 holes	Punch holes 1 - 6 on paper tape according to bits 2, 5, 8, 11, 14 and 17 of the AC. Always punch hole 7.
CPY	620000	Copy	Control transfer of information to or from a magnetic tape unit or from the paper tape reader according to last in-out select instruction (Table) given. For timing and other details see M-5001-
EXO	610000		Operate user's external equipment. For details see M-5001- (applies to EXO–EX7)
EX1	611000		
EX2	612000		
EX3	613000		
EX4	614000		
EX5	615000		
EX6	616000		
EX7	617000		
TAC	601000	Examine TAC	TAC ∪ AC → AC. (logical sum) Occurs on time pulse 1 of cycle one.
TBR	602000	Examine TBR	TBR ∪ MBR → MBR. (logical sum) Occurs on time pulse 2 of cycle one.
NOP	600000	No Operation	No in-out function is to be executed.
HLT	630000	Halt	Stop the computer after completing this instruction (time pulse 8 of cycle one).

TABLE 9. IN-OUT COMMANDS WHICH MAY BE USED IN MICRO-PROGRAMMED OPERATE CLASS INSTRUCTIONS

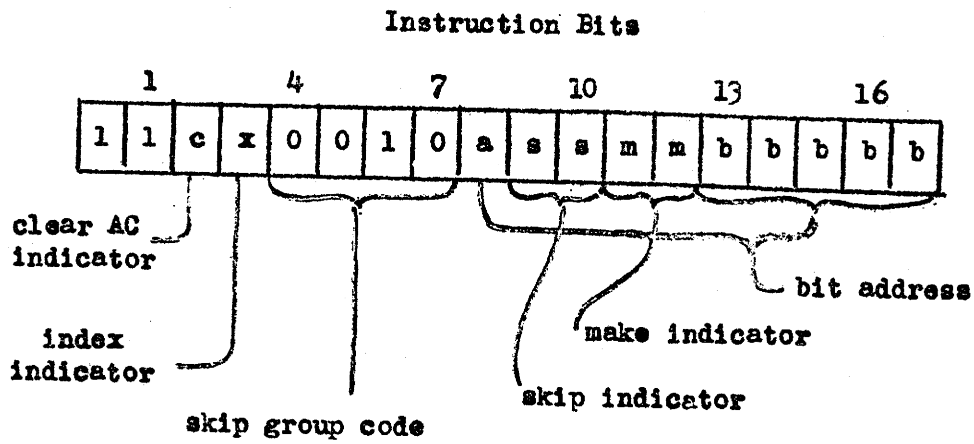


Figure 11. Structure of a skip and make instruction

CODE	SYMBOL	SELECTED BIT
604000 + N	SKA + N	Bit n of AC, $0 \leq n \leq 21_8$
605000 + N	SKO + N	Bit n of OR, $0 \leq n \leq 21_8$
604024 + N	SKS + N	Bit n of PSR, $0 \leq n \leq 13$
605024 + N	SKI + N	Bit n of PIR, $0 \leq n \leq 13$
604022	SKA 22	Parity of AC*
605022	SKO 22	Parity of OR*

*May only be examined, of course.

a) Bit selection by skip group instructions

Skip Indicator, Bits 9, 10	ACTION
0 0	Never skip
0 1	Skip if selected bit is zero
1 0	Skip if selected bit is one
1 1	Always skip

b) Skip indicator bits

Make Indicator Bits 11 and 12	ACTION
0 0	Do not change selected bit
0 1	Set selected bit to zero
1 0	Set selected bit to one
1 1	Complement selected bit

c) Make indicator bits

Table 12. Skip group instruction possibilities

CODE	SYMBOL	NAME	ACTION*
604023	SSE	Set Sense Register	Sets selected bits of PSR to agree with corresponding bits of AC.
704023	SSE + CL	Set Sense and Clear	As above, and leave AC clear.
604423	TSE	Test Sense Register	Partial add selected bits of PSR to corresponding bits of AC.
704423	TSE + CL	Clear and Test Sense	As above, but clear AC first.
605023	SIN	Set Indicator Register	Same as above group, but refers to PIR.
705023	SIN + CL	Set Indicators and Clear	Same as above group, but refers to PIR.
605423	TIN	Test Indicator Register	Same as above group, but refers to PIR.
705423	TIN + CL	Test Indicators and Clear	Same as above group, but refers to PIR.

*Selected bits are those positions in which the OR contains zeros.

Table 13. Instructions for setting and testing PSR and PIR

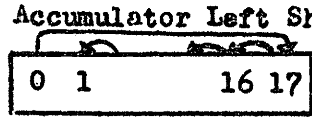
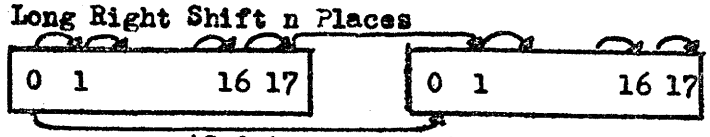
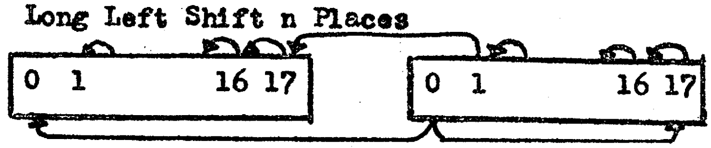
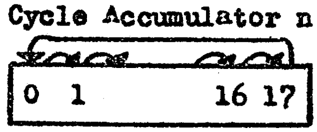
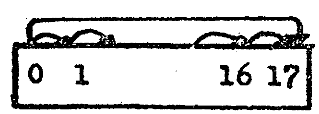
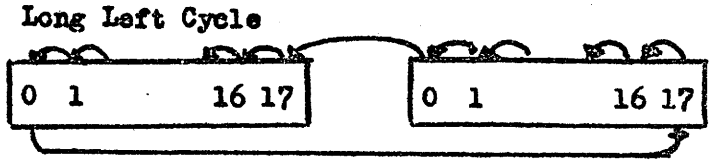
CODE	SYMBOL	NAME AND FUNCTION (AC / OR)
607400 + N	ARS + N	Accumulator Right Shift n Places  AC-0 is unchanged
607600 + N	ALS + N	Accumulator Left Shift n Places  AC-0 is unchanged
606400 + N	LRS + N	Long Right Shift n Places  AC-0 is unchanged
606600 + N	LLS + N	Long Left Shift n Places  OR-0 is unchanged
607100 + N	CYA + N	Cycle Accumulator n Places 
607700 + N	CYO + N	Cycle Operand n Places 
606500 + N	LLC + N	Long Left Cycle 

Table 15. Shifting Instructions of the Operate Step Group

CODE*	SYMBOL*	NAME	FUNCTION
634011 + N	REW + N	Rewind	Rewind magnetic tape unit n to its load point.
634015 + N	WRT + N	Write Tape	Start Magnetic tape unit n forward in the record mode.
634005 + N	RDT + N	Read Tape	Start magnetic tape unit n forward in the read mode.
634001 + N	BST + N	Backspace Tape	Run magnetic tape unit n to the beginning of the previous record.
634004	R3L	Read three line	Start paper tape reader in the three line mode.
634014	R1L	Read one line	Start paper tape reader in the one line mode.
634025	IMT + N °	Load Mag. Tape	Initiate read in mode from magnetic tape unit n. Unit n will first be rewound to its load point.
634024	LPT °	Load Paper Tape	Initiate read in mode from paper tape reader.
634000	STP	Stop	Deselect any selected unit. This order must be given to stop the paper tape reader.

* In octal. n = 0, 1 or 2. °Giving the instruction IMT 0 or LPT is equivalent to pushing the "Load Mag. Tape" or Load Paper Tape" push button, respectively.

Table 18. In-out select order group

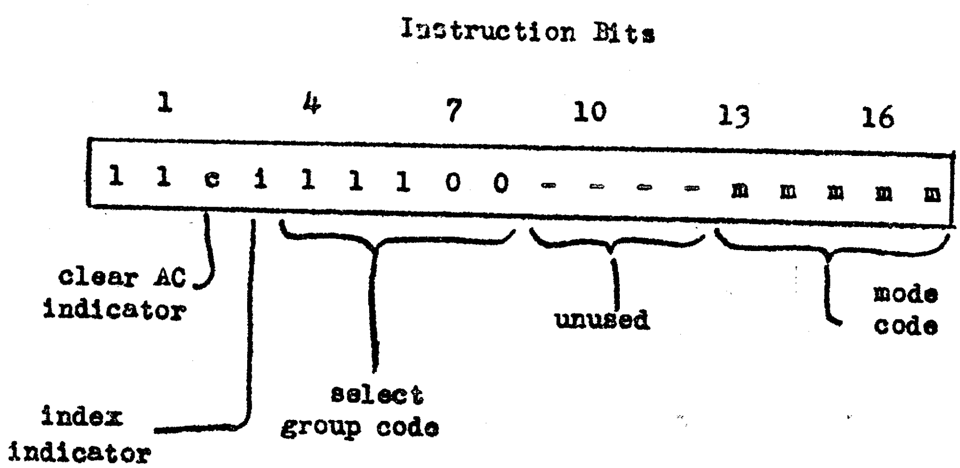


Figure 17. Structure of an in-out select instruction

TX-0 COMPUTER
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE 39, MASSACHUSETTS

May 25, 1960

To the Ad Hoc Committee on Experimental Computation and other interested persons:

Enclosed is a proposal for a comprehensive enlargement of the TX-0 instruction code. The staff feels that many of the features of this instruction code offer a great advantage over possible alternatives and should certainly be installed. However, there are some features included in the proposal which, perhaps, do not make the best use of available operation codes. Your ideas and comments are solicited -- perhaps there are useful alternatives which have not been considered.

The reader will notice that the proposal suggests the name "operand register" as more appropriate than "live register". Unfortunately, this suggestion leads to a rather unaesthetic set of symbolic operation codes (as given in the memo). Criticism of this suggestion has already been loudly voiced, and, therefore, the peculiar designation "live register" will probably be retained.

Since the memo was prepared, a different scheme of organizing the micro-programmed operate orders has been devised. This scheme retains as many of the current possibilities as the arrangement set forth in Chart 7 and Table 8 of the enclosed memo, yet allows even more flexibility. For instance, the new scheme allows the contents of LR and XR to be interchanged. Charts for this alternative will be prepared shortly.

It is planned to proceed this summer with the installation of the portions of the proposal which we are satisfied make good use of the available operation codes. The following table lists these features and the approximate date they will be available to users. Of course, the dates assume that adequate funds and technical assistance will be available.

Features:	Expected to be in operation:
New addressable commands TZE LDX TIX ADX TSX STX STO + X SLR + X (SOP + X) ADD + X LLR + X (LOP + X) TRA + X	September 1960

New schedule of micro-programmed operate commands	September 1960
Magnetic Tape orders: (one unit)	November 1960
Certain Skip Group commands associated with Magnetic Tape operation	November 1960
Certain Step Group commands including Multiply	December 1960

The remaining skip group and step group operate orders will be installed at a rate depending upon the amount of time subsequently available for modifications.

The value of the remaining addressable instructions, and the half-word provision for the operate micro-orders, is not clear. Therefore, active consideration of these features has been dropped for the present.

Any criticism or suggestions with respect to these plans will be given due consideration.

Sincerely,

J B Dennis/dbh

J. B. Dennis

JBD/dbh
Enc.: M-5001-22

TX-0 COMPUTER
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE 39, MASSACHUSETTS

June 15, 1960

To the Ad Hoc Committee on Experimental Computation:

During the past two weeks two meetings have been held with the important users of the TX-0 Computer to discuss the usefulness of the proposed additions to the TX-0 order code, and to discuss the affect of the changes on user's programs. The principal problem arose concerning the order of installation of the changes in the coding of the operate command, and the index register with its associated new instructions. The following points were brought out:

(1) User's are more enthusiastic about the index register than the changes in the coding of the operate command as an aid in improving and speeding up their programs.

(2) A number of important programs do not have up-to-date English (external language) versions. Hence maintaining operability in these cases is not merely a matter of recompiling. Thus, if the index register were not available until some time after the changes in the operate command coding, two reprogramming jobs would be required--one to keep programs working with the new operate structure, and one to take advantage of the index instructions.

(3) The difficulty of new programmers coping with changes made during the fall term has probably been over-emphasized. They can easily avoid the pitfalls if they are warned in advance.

For these reasons it was suggested that the index register be given a relatively high priority, and that the changes in the coding of the operate command be delayed until the indexing instructions are operating.

From the hardware point of view the following facts are pertinent:

(1) It is desired to eventually replace the present vacuum tube time pulse generator with a transistorized equivalent (cost--about $2,000). This would be much more reliable and would yield a considerable saving in space. Also meeting the needs of new control circuitry would be relatively difficult if the present time pulse distribution scheme is retained.

(2) The above change requires a switch from the presently used register driver circuit to a pulse amplifier circuit. The cost of components to manufacture these amplifiers is $2,500 for a sufficient number of units to cover all TX-0 circuits with spares. To build a smaller number of units

is economically unsound. The units would be assembled by our technitians. A supply of these units will be necessary to place the index register in operation.

(3) At present, it does not seem likely that the index register could be installed and operating by the end of the summer. The principal reason is the time required to have additional TX-2 plug-in units assembled bu an outside vendor.

(4) With the pulse amplifiers available, all of the changes in TX-0 control circuits required for the tape unit, new operate orders, and most of the changes required for the index register installation could be made this summer with <u>no</u> effect on currently operating programs. The actual change in the coding of the present orders could be accomplished at a convenient time with a shut-down of a couple of days. The installation of the indexing instructions would require about two weeks of down time.

On the basis of this information the following schedule of installation is proposed. The estimated down time for each job is indicated in parentheses.

<u>Summer 1960</u>:

(1) Build pulse amplifier units (0)

(2) Wire panel for new program counter. This is the major addition required for the operation of the indexing instructions. (0)

(3) Wire tape unit control panel. (0)

(4) Make changes in control circuits of TX-0 for tape unit, index register and new schedule of operate commands. (30 days in groups of 1, 2 or 3).

(5) Mechanical changes in control panel, assembling new rack, running cables, etc. (10 days in short pieces). New orders available: TZE - transfer on zero; perhaps ADO - add one to memory register; select and copy orders for control of a second paper tape reader. The present PETR will remain operating as at present.

<u>Fall 1960</u>:

(1) Construct TX-2 type plug-in units for new program counter. (0)

(2) Continue changes in control if not completed during summer. (?)

<u>December 1960</u>:

(1) Check out magnetic tape unit. (10 days in bits)
New orders: Tape control orders as indicated in M-5001-22.

<u>Xmas vacation or between terms</u>:

(1) Install new program counter (2 weeks)
New orders: Indexed STO, SLR, ADD, LIR and TRA; TSX, TIX, LDX, ADX, STX as given in M-5001-22.

(2) Change operate command to agree with attached schedule. (2 days)

Funds currently have been procured for all of the above work except for the construction of pulse amplifier units and the construction of a new rack to provide space for magnetic tape equipment. The former will cost $2,500. The majority of the parts for the new rack are expected to be available from Lincoln Laboratory, and the remaining parts we hope can be constructed from standard stock in our shops.

Respectfully submitted,

Jack B. Dennis (eab)

Jack B. Dennis

JBD:EAB

SYMBOL	CODE	NAME	FUNCTION
PEN	603000	Read Light Pen	set AC bit 0 from light pen FF, and AC bit 1 from light gun FF. (FF's contain ONE if pen or gun saw displayed point.)
R1L	621000	Read One Line	Read one line of tape from PETR into AC bits 0, 3, 6, 9, 12, 15, with CYR before read
R3L	623000	Read Three Lines	Read three lines of tape from PETR into AC, with CYR before each read

TABLE 9 - Continued

Instruction bits 6, 7, and 8

Instruction bits 4 and 5	000	001	010	011	100	101	110	111
00	NOP	TAC	TBR	PEN	In-Out Select Group			
01	EX0	EX1	EX2	EX3	EX4	EX5	EX6	EX7
10	CPY	R1L	DIS	R3L	PRT	TYP	PCH	P7H
11	HLT	CLL	CLR		Skip Make	and Group	Step Group	Command

Micro-Programmed Group

<u>Chart 5</u> **<u>In-Out Orders and Special Operate Groups</u>**

Instruction bits

Cycle		2	3	9	10	11	12	13	14	15	16	17
Cycle Zero	7		1 AMB									
	8	1 CLA										
	In - Out Stop											
Cycle One	2			0	X XMB	1	1 COM					
	3									1	1 ANB	1
	3									1	0 ORB	1
	4			0	1 MBL	X				0	1 IMB	x
	5							1 PAD				
	5B			1	0 SHR	0						
	5B			1	1 CYR	0						
	7								1 CRY			
	8									0	X MBX	1

X - Bit may be either zero or one

CHART 7 Sequence and Code Bits for Operate Micro-orders

CODE	SYMBOL	FUNCTION
		The MBR is initially clear.
640000	AMB	transfer AC contents to MBR (occurs before CLA)
700000	CLA	clear the AC
600040	COM	complement AC
600500	XMB	Transfer contents of XR to MBR (logical sum). The sign of XR, bit 4, will be the input for bits 0 - 4 of MBR.
600200	MBL	transfer contents of MBR to LR
600002	LMB	transfer contents of LR to MBR
		Note: MBL and LMB when used together will interchange the contents of MBR and LR
60007	ANB	AND the contents of MBR and LR and leave in MBR
60005	ORB	OR the contents of MBR and LR and leave in MBR
600020	PAD	partial add MBR to AC (for each MBR one, complement the corresponding AC bit)
600400	SHR	shift AC contents right one binary position (bit 0 is unchanged, bit 17 lost)
600600	CYR	cycle AC contents right one binary position (AC bit 17 goes to AC bit 0)
600010	CRY	PAD plus carry completes a full add. Carry is also useful itself. (A carry digit is a ONE if in the next least significant digit either AC = 0 and MBR = 1, or AC = 1 and carry digit = 1. The carry digits so determined are partial added to the AC by CRY.)
600001	MBX	Place the contents of bits 0 and 5 through 17 of MBR in bits 4 - 17 of XR

TABLE 8 ACTION OF OPERATE MICRO ORDERS

SYMBOL	CODE	NAME	FUNCTION
CLL	631000	Clear Left	Clear bits 0-8 of AC
CLR	632000	Clear Right	Clear bits 9-17 of AC
DIS	622000	Display	Display a point on the CRT with x and y coordinates specified by the left and right halves of the AC, respectively, interpreted as one's complement nine-bit numbers.
PRT	624000	Print	Print the flexowriter character specified by bits 2, 5, 8, 11, 14 and 17 of the AC.
TYP	625000	Type	Read the last 6-bit character typed into bits 0, 3, 6, 9, 12 and 15 of the LR. A flip-Flop indicator will be set when a character is typed.
P6H	626000	Punch 6 holes	Punch holes 1 - 6 on paper tape according to bits 2, 5, 8, 11, 14 and 17 of the AC. Do not punch hole 7.
P7H	627000	Punch 7 holes	Punch holes 1 - 6 on paper tape according to bits 2, 5, 8, 11, 14 and 17 of the AC. A.ways punch hole 7.
CPY	620000	Copy	Control transfer of information to or from a magnetic tape unit or from the paper tape reader according to last in-out select instruction (Table) given. For timing and other details see M-5001-
EX0	610000		Operate user's external equipment. For details see M-50010
EX1	611000		
EX2	612000		
EX3	613000		
EX4	614000		
EX5	615000		
EX6	616000		
EX7	617000		
TAC	601000	Examine TAC	TAC $\cup$ AC → AC. (logical sum) Occurs on time pulse 1 of cycle one.
TBR	602000	Examine TBR	TBR $\cup$ MBR → MBR. (logical sum) Occurs on time pulse 2 of cycle one.
NOP	600000	No operation	No in-out function is to be executed.
HLT	630000	Halt	Stop the computer after completing this instruction (time pulse 8 of cycle one).

TABLE 9. IN-OUT COMMANDS WHICH MAY BE USED IN MACRO-PROGRAMMED OPERATE CLASS INSTRUCTIONS