alwac III-E

DATA PROCESSING SYSTEMS

DESCRIPTION OF OPERATIONS

ALWAC III-E

ELECTRONIC DIGITAL COMPUTER

LOGISTICS RESEARCH INC.

141 so. pacific ave. redondo beach, california

DESCRIPTION OF OPERATIONS

ALWAC III-E

ELECTRONIC DIGITAL COMPUTER

INTRODUCTION

The ALWAC III-E is a general purpose, electronic digital computer designed for an extraordinary range of application to the problems of scientific-engineering-research and business-industrial-financial organizations. If a problem can be explicitly formulated --- ALWAC can solve it at electronic speed.

ALWAC is able to organize, calculate, and make decisions. By using an internally stored sequence of instructions set up by the operator to apply to simple or complex problems, ALWAC can absorb facts (words or numbers), store, sort, distribute, rearrange, and then call them when needed for calculation, modification, or for output.

ALWAC can store and obey a sequence of up to 16,000 instructions, and by adding, multiplying, subtracting or dividing is capable of calculating required answers of any degree of complexity, in any desired sequence.

Individual steps in a routine can be judged or monitored, and the process guided by decision or break points in the routine. ALWAC's power is based on its ability to choose different courses of action depending on the nature of some intermediate results --- even to the extent of changing criteria by which it decides.

When decision criteria are logical or quantitative, they can be coded into standardized routines, placed in permanent storage, and used repeatedly in miscellaneous problems as required.

The machine's "eyes", "ears", and "voice" are inputs and outputs --- punched paper tape, magnetic tape, punched cards, typewriter keyboard or high-speed reader.

The following revision should be incorporated in this manual:

Page 15 ..... last paragraph - third line - should read "is one whose last digit is three greater, in time,"

instead of "four greater" as it now appears.

The ALWAC III-E obtains its code list of instructions from the hexadecimal number system. A discussion of number systems in general is beyond the scope of this text, but the following list of numbers representing the basic characters of the binary, decimal and hexadecimal systems and their relationship is given for convenience.

BINARY	DECIMAL	HEXADECIMAL
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	a
1011	11	b
1100	12	c
1101	13	d
1110	14	e
1111	15	f
10000	16	10

It is easily seen that each group of four (4) binary characters represents one hexadecimal character.

The arithmetic registers of the ALWAC III-E computer are circulating loops of information one word in length. The information is changed when the magnetic writing head is switched to an input device or to some reading head other than the one which makes up the loop with it. In addition to three one-word registers, referred to as A, B and D, the drum has four 32-word channels of working storage, referred to by Roman numerals I, II, III and IV, from which instructions are picked up to be carried out, and data picked up to be operated on in the arithmetic registers. The working storage, through what is essentially a loop arrangement, has faster access than the remainder of the drum, the main storage.

Main storage consists of 256 channels or blocks, 32 words in each. These are numbered hexadecimally 00, 01, 02 .... ff and may be copied as a unit to one of the working-storage channels, and a working-storage channel may be copied to any main channel.

Since block-copy operations always transfer 32 words between working storage and main storage, individual words in main storage are not addressed. (Main memory channel 00 is a single exception and will be discussed later). In working storage, however, not only words, but half-words may be addressed, since each instruction and associated address occupies a half-word.

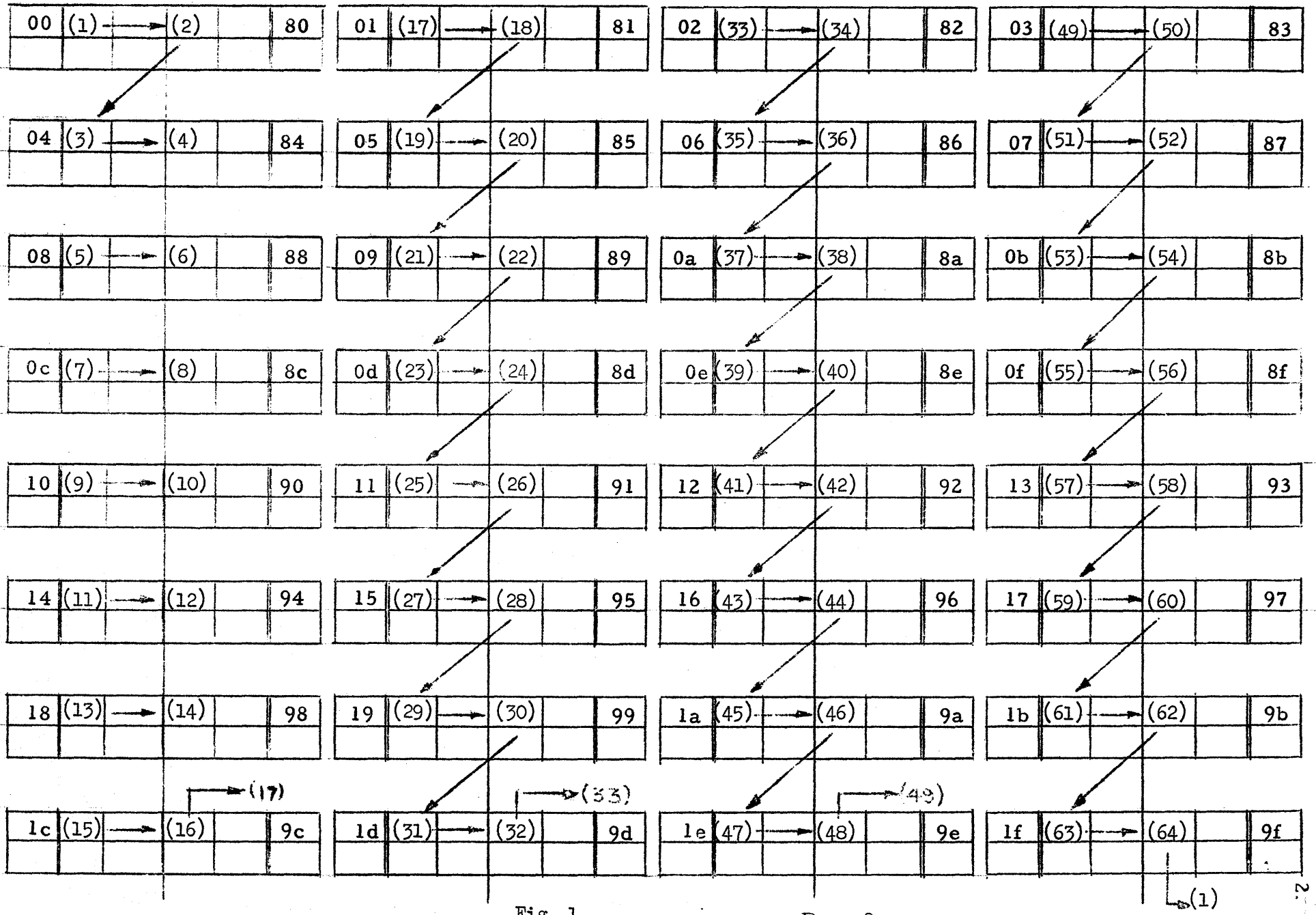

00 (1) (2) 80
01 (17) (18) 81
02 (33) (34) 82
03 (49) (50) 83
04 (3) (4) 84
05 (19) (20) 85
06 (35) (36) 86
07 (51) (52) 87
08 (5) (6) 88
09 (21) (22) 89
0a (37) (38) 8a
0b (53) (54) 8b
0c (7) (8) 8c
0d (23) (24) 8d
0e (39) (40) 8e
0f (55) (56) 8f
10 (9) (10) 90
11 (25) (26) 91
12 (41) (42) 92
13 (57) (58) 93
14 (11) (12) 94
15 (27) (28) 95
16 (43) (44) 96
17 (59) (60) 97
18 (13) (14) 98
19 (29) (30) 99
1a (45) (46) 9a
1b (61) (62) 9b
(17)
(33)
(49)
1c (15) (16) 9c
1d (31) (32) 9d
1e (47) (48) 9e
1f (63) (64) 9f
(1)

Fig. 1

The sketch below shows the easily read manner in which words appear on the face of the oscilloscope. The staggering of the alternate syllables permits a ready inspection of recorded orders as well as the immediate determination of stored numbers in a hexadecimal form.

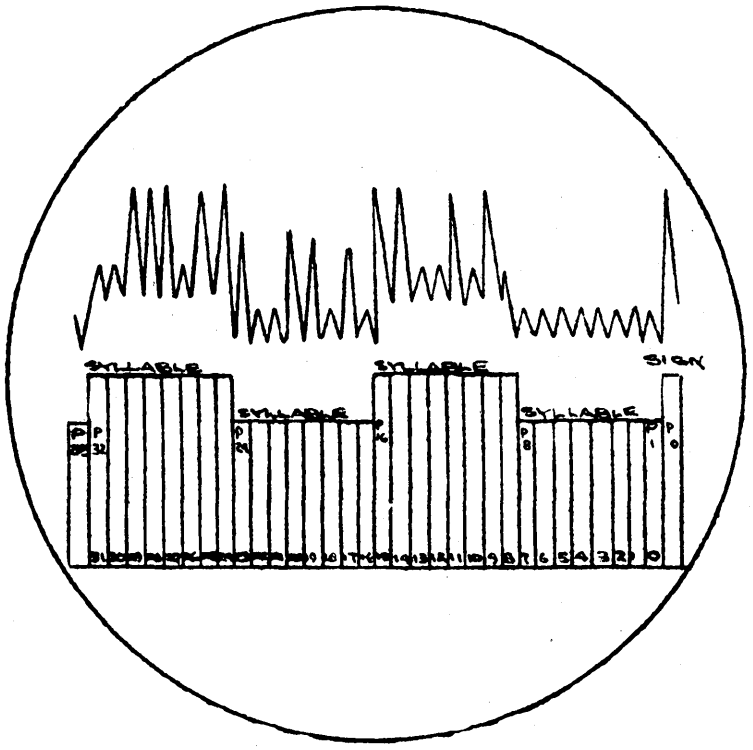


Figure 2.

OSCILLOSCOPE DISPLAY

This shows the easily read hexadecimal configuration 3b9aca00 which is equal to the decimal number 1,000,000,000. In a like manner commands and addresses can be visually inspected.

The word addresses for the 128 words of working storage are numbered consecutively, 00 through 1f for channel I, 20 through 3f for channel II, 40 through 5f for channel III and 60 through 7f for channel IV. These same addresses also refer to the left half of each word, but various instructions which require these addresses make the correct interpretation. The right half-word addresses are also numbered consecutively, 80 through 9f for channel I, a0 through bf for channel II, c0 through df for channel III, and e0 through ff for channel IV. Reference to the programming forms at back of this manual will help to clarify this pattern.

The sequence in which ALWAC III-E picks-up each instruction and address from a set program or routine is shown in Figure 1, using channel I as an example. It is seen that the first (1) operation to be performed is located in half-word 00, the second (2) in half-word 80, the third (3) in half-word 04, the fourth (4) in half-word 84, the fifth (5) in half-word 08, and so on through half-word 9c. From half-word 9c the sequence goes to half-word 01 and proceeds down this column of words in a similar order until the operation contained in 1d is performed. From half-word 1d the sequence goes to half-word 02 and proceeds similarly throughout the rest of the channel. From half-word 1f the sequence returns to 00. This is the normal sequence, but it can be broken at any time in a routine by use of any of various jump instructions (break point). Also, the sequence can be started in any half-word. The above holds true for the other three working channels, and jumps can be made from one channel to another.

The ALWAC III-E word may be viewed on the oscilloscope in a pattern similar to that shown in Figure 2, as shown on page 3. Magnitude of the word is 32 binary digits (quite frequently referred to as binary bits or pulses). There is another bit or pulse, designated as P_0, to the extreme right, which is the sign of the word. For plus, P_0 is up (contains a one (1)); for minus, P_0 is down (contains a zero (0)). At the extreme left of the word is an overflow bit designated as P_{33}. The 32 binary digits (bits), P_1, P_2 ... P_{32}, are divided into four (4) groups of 8. Each group is called a syllable. Thus, one (1) syllable comprises two (2) hexadecimal digits, two (2) syllables comprise a half-word, and so on. The decimal numbers 0, 1 ... 31, refer to powers of the base 2, which is, of course, the radix in the binary system. $2^{32}-1$ is the decimal equivalent of a word containing all ones (hexadecimally, ffffffff), the maximum capacity. P_{32} is the high order bit, P_{31} next, and so on down.

For programming purposes words are used not only to contain data, but also to hold the various instructions and their associated addresses. The instruction word can be interpreted according to Figure 3.

Bin.	0	1111	0001	0000	1000	0100	1001	0010	0000	1
Hex.	P_{33}	f	1	0	8	4	9	2	0	P_0
		Instr.		Address		Instr.		Address		

Figure 3 - Instruction Word

It is seen that the instruction portion of the word occupies eight (8) binary, or two (2) hexadecimal digits. The same is true regarding addresses. While it may appear, at this point, that only two (2) instructions per word are allowed, it will be shown in the next section that as many as four (4) instructions can be put in one word.

Key to Symbols Used

A	Accumulator; holds sums, differences, remainders.
B	Holds multiplicands, dividends, quotients.
D	Holds multipliers and divisors.
E	A half-word tally or counting register.

One Word Recirculating Registers

AB	Dougle length accumulator; the A and B registers combined; A is most significant part and B is the least significant.
M	Address of channels (32-word blocks of information) in the main memory.
N	Any hexadecimal number.
P_o	Sign bit.
P_{32}	The highest bit of magnitude in a word.
P_{33}	A bit to the extreme left of a word, used in A to hold overflows.
W	Any word address of the 128 in working storage.
W/2	Any half-word address.
Z	OVERFLOW indicator (flip-flop)

M, N, W and W/2 always occupy the address syllable of the instruction that uses them.

Note: The E register is also used for AUTOMATIC ADDRESS MODIFICATION, in addition to its counting feature, but this is covered in a special section on page 18.

NO-ADDRESS INSTRUCTIONS WHICH MODIFY ACCUMULATOR

22	ROUND OFF	Round off the double length number [AB] to a single length number in A; if P_{32} in B (left hand bit) is not zero, add 1 to A. B remains unchanged. If capacity of A is exceeded Z comes on.
28	CLEAR A	Put plus zero in A. (P_o = 1)
2c	ABSOLUTE VALUE	Make sign of A positive. (P_o = 1)
2e	REVERSE SIGN A	If P_o = 1, make it 0. If P_o = 0, make it 1. The magnitude is unchanged.
3e	COMPLEMENT A	Make a 1's complement of the number in A, including P_o. (Change all 1's to 0, all 0's to 1).

NO ADDRESS COPY AND EXCHANGE

30	EXCHANGE A and B	32	COPY B to A. (B unchanged)
3a	EXCHANGE A and D	38	COPY D to A. (D unchanged)
36*	EXCHANGE A and E	34*	COPY E to A. (E unchanged)

*While E, like A, B and D, is a circulating loop of information, one word in length, it has no effective sign. When viewing E on the oscilloscope a pulse in the sign position (P_o) may be seen but it is completely ignored by the computer. The absolute value of a count or tally is kept in the left half and this is the only part of E which is available to the programmer.

Thus, in both the 36 and 34 instructions, only the left half of the respective registers is considered. The right half of A is unchanged.

PROCEDURE FOR DOUBLING

The instructions described thus far (all even-numbered) require no specific address in the address syllable of the half-word in which they are used. For this reason they may be doubled-up to yield three or four instructions per word. To do this it is necessary to modify the instruction which is to be in the instruction syllable by adding 1 to its code. The second instruction can then be used, without alteration, in the address syllable. For example, let the instructions 28, 30, 38 and 32 be a desired sequence of operations. Instead of using two words, thus:

2 8 0 0 3 0 0 0

3 8 0 0 3 2 0 0

They may be packed into one word, thus:

2 9 3 0 3 9 3 2

If an even-numbered instruction is so modified, and is used in an instruction syllable, any instruction except the four (4) shift operations may be used in the address syllable.

CAUTION: IF THE SECOND INSTRUCTION IS ONE WHICH REQUIRES AN ADDRESS (W, W/2, N, etc.) IT WILL USE ITSELF AS THE ADDRESS.

It is, therefore, advisable that only those instructions which require no address be chosen for use in an address syllable.

SHIFT INSTRUCTIONS

The shift instructions in the ALWAC shift the number in A, or [AB] N binary places, where N, a hexadecimal number, is less than 40 but greater than 0. This gives as many as 63 binary shifts. Regardless of the bits used in the address syllable for N, only the right hand six (6) will be considered. (The bits of N are interpreted "modulo sixty-four").

ab	FLOAT	Copy sign of B to A. Shift [AB] to the left until the high order bit of A (P_{32}) is 1. Record in D the positive value of the number of binary shifts required. If P_{32} of A is already 1, or [AB] is zero, record in D a minus zero.
a1	DOUBLE SHIFT RIGHT	Shift AB , excluding signs, N binary bits to the right, where N is in the address syllable. Bits leaving A enter the left end of B. 0's are shifted into P_{33} but if there is a bit in P_{33} it will shift into P_{32}. N bits to the right in B are lost.
a3	DOUBLE SHIFT LEFT	Shift [AB] , excluding signs, N binary bits to the left, where N is in the address syllable. Bits leaving B enter the right end of A. The right N bits of B are filled with 0's and N bits to the left in A are lost. There is no overflow indication, and if there is a bit in P_{33} it will be lost.
a5	SHIFT RIGHT	Shift A, excluding sign, N bits to the right, where N is in the address syllable. The right N bits are lost. 0's are shifted into P_{33} but if there is a bit in P_{33} it will shift into P_{32}.
a7	SHIFT LEFT	Shift A, excluding sign, N bits to the left, where N is in the address syllable. The right N bits of A are filled with 0's and the left N bits are lost. There is no overflow indication, and if there is a bit in P_{33} it will be lost.

ARITHMETIC

61	ADD	Add W to A and put the sum in A.
63	MINUS ADD	Add W to A, put sum in A and reverse sign of A.
65	MINUS SUBTRACT	Subtract W from A, put difference in A and reverse sign of A.
67	SUBTRACT	Subtract W from A and put difference in A.

All four (4) operations are algebraic. If the result of any of the above arithmetic instructions exceeds the capacity of A, Z will come on and P_{33} will be 1.

bd	LONG ADD	Copy sign of B to A, add W to [AB] and put sum in [AB] .
bf	LONG SUBTRACT	Copy sign of B to A, subtract W from [AB] and put difference in [AB] .

In both cases, the signs of A and B will be alike in the result. If the capacity of [AB] is exceeded Z will come on but no 1 is put in P_{33}. The operations are algebraic.

e1	ADD MULTIPLY BY D	Multiply B by D and put product in [AB] ; make the signs of both A and B the correct sign of the product. Any number previously in A is given the same sign as the product and is added to the least significant part of this product. D remains unchanged.
e3	ADD MULTIPLY BY W	Copy W to D and perform e1.
e5	MULTIPLY BY D	Clear A and perform e1.
e7	MULTIPLY BY W	Copy W to D, clear A and perform e1.
e9	LONG DIVIDE BY D	Divide [AB] by D. The sign of B is considered to be the sign of the dividend. Put quotient in B, with correct sign: remainder, with sign of dividend, in A. D is unchanged.
eb	LONG DIVIDE BY W	Copy W to D and perform e9.
ed	DIVIDE BY D	Clear A and perform e9.
ef	DIVIDE BY W	Copy W to D, clear A and perform e9.

Just before actual division takes place, the computer tests that D is greater than A (absolute values), since A is the most significant part of the dividend. If D has the greater magnitude, then a proper division can be made and the operation is performed. When D is less than A (absolute

values) the resulting quotient would exceed the capacity of B. Thus Z comes on, (indicating improper division), the division is not performed and the computer proceeds to the next instruction.

The 22 instruction and any one of the fourteen (14) arithmetic operations will not be performed if the overflow (Z) is already on. Instead, the computer will stop and sound an alarm. 22, e1, e5, e9 and ed are exceptions but only when these are used in an address syllable.

A Table of Arithmetic Operations is given on Page 10 for ready reference.

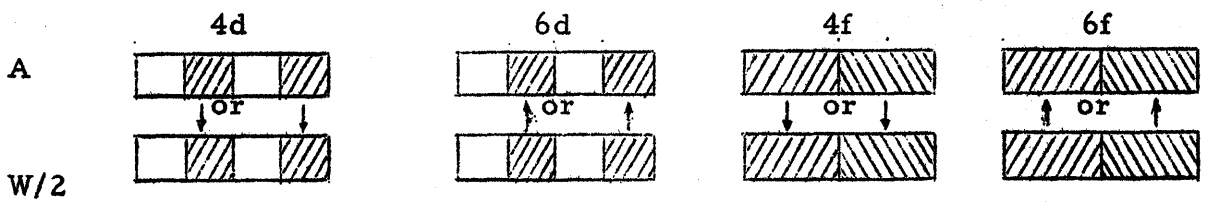
COPY AND EXCHANGE WITH ADDRESSES

69	EXCHANGE A and W	79	COPY W to A
49	COPY A to W	41	COPY W to B
c5	COPY B to W	5b	COPY W to D
c7	COPY D to W	57	COPY W to E
c3	COPY E to W		

In the 57 and c3 instructions only the left half of W is affected; the right half remains unchanged.

4d	COPY ADDRESS TO W/2	6d	COPY ADDRESS FROM W/2
4f	COPY HALF TO W/2	6f	COPY HALF FROM W/2

In 4d, 4f, 6d and 6f, W/2 refers to a half-word address, but 4d and 6d affect only the address syllable, while 4f and 6f affect both the instruction syllable and the address syllable. Copying proceeds between W/2 and the syllable (s) which are in the same relative position of A.



b5 COPY MAIN TO A

This instruction requires W as its associated address. The W refers to a word in channel 00 of the main storage. These words have the same addresses as words in channel I of working storage.

At least two drum revolutions, or 34 milliseconds, must be allowed between the b5 operation and any preceding BLOCK COPY operation (explained later) except one which copies to channel 00. This allows time for head-switching relays to drop out.

TABLE OF ARITHMETIC OPERATIONS

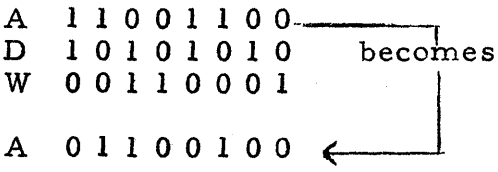
Instruction	Address	Operation	Location of Result	Remarks
61	W	A + W	A	If capacity of A is exceeded Z comes on and P_{33} will contain a one (1).
67	W	A - W	A	
63	W	-(A + W)	A	
65	W	-A + W	A	
bd	W	[AB] + W	[AB]	Same except no P_{33} indication.
bf	W	[AB] - W	[AB]	
e5	*Optimum	B x D	[AB]	D left unchanged.
e1	*Optimum	$(\lvert B \times D \rvert + \lvert A \rvert)\frac{B \times D}{\lvert B \times D \rvert}$	[AB]	If B or D = 0 |A| still takes sign of product and is added in.
e7	W	W copied to D and e5 performed	[AB]	As e5
e3	W	W copied to D and e1 performed	[AB]	As e1
ed	*Optimum	$B/_D$	B	Remainder, in A, takes sign of dividend.
e9	*Optimum	$[AB]/_D$		
ef	W	W copied to D and ed performed		D unchanged
eb	W	W copied to D and e9 performed		

* While these instructions require no specific address, they do, by their very nature, search for that address which appears in the address syllable of the half-word in which they are used. This address, however, is not an operand; and by choosing one which is optimum, operation time is decreased. How to determine optimum addresses will be discussed in its proper place later.

71 EXTRACT (D)

Extract according to D. Wherever D contains a 1 bit, copy the corresponding bit of W to A. Wherever D contains a 0 bit leave the corresponding bit of A in A.

Example:



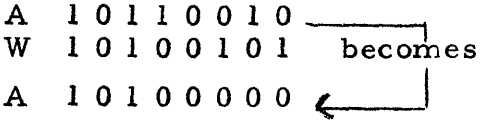
Expressed logically this is:

(A X D') + (D X W)

75 EXTRACT

Wherever A contains a 1 bit, copy the corresponding bit of W to A. Wherever A contains a 0 bit, leave the corresponding bit of A in A.

Example:



This is nothing more than logical multiplication; A X W

BLOCK COPY

81	COPY M to I	89	COPY I to M
83	COPY M to II	8b	COPY II to M
85	COPY M to III	8d	COPY III to M
87	COPY M to IV	8f	COPY IV to M

Each of these 8 BLOCK COPY operations copies a block of 32 words (including signs) to or from channels in main memory storage (M) from or to channels in working storage (I, II, III or IV) as specified.

All copy operations leave the information in the location from which it was copied.

JUMP AND RELATED

11	JUMP	Carry out the instruction contained in W/2.
13	CONTROL JUMP 1	If jump switch 1 on control panel, is ON, carry out the instruction contained in W/2; if it is OFF, continue in the normal sequence of instructions.
15	CONTROL JUMP 2	If jump switch 2, on control panel, is ON, carry out instruction contained in W/2; if it is OFF continue in the normal sequence of instructions.
17	COUNT DOWN	Subtract 1 from the absolute value of the left half of E. If the result is not zero, carry out the instruction contained in W/2; if the result is zero continue in the normal sequence of instructions.
19	NON-ZERO JUMP	If A is not zero, carry out the instruction contained in W/2; if A is zero continue in the normal sequence of instructions.

1b	STOP	Wherever this instruction occurs, the computer will stop, if the START-NORMAL switch (located on control panel) is in the NORMAL position. When this switch is in the START position the instruction acts as a JUMP instruction (11) and carries out the instruction located in W/2.
1d	LESS THAN ZERO JUMP	If A is less than zero, carry out the instruction contained in W/2; if A is plus or minus zero, or greater than zero, continue in the normal sequence of instructions.
1f	OVERFLOW JUMP	If Ƶ is ON, turn it OFF and carry out the instruction contained in W/2; if Ƶ is OFF, leave it OFF and continue in the normal sequence of instructions.

Whenever a jump is executed, the normal sequence of instructions continues from W/2 (the half-word address to which the jump was made).

02	REVERSE OVERFLOW	If Ƶ is on, turn it off; if it is off, turn it on. (This instruction may be used as one of a doubled pair. The rule for doubling up is explained on page 6.
51	OVERFLOW IF A SMALLER	Consider only absolute values of A and W. If A is less than W, turn Ƶ ON, if A is greater than or equal to W, do nothing to Ƶ. In either case, continue in the normal sequence of instructions. If, however, Ƶ is already ON, stop and sound an alarm.

While 51 and 02 are not jump instructions, they are included here because of their relationship with the OVERFLOW JUMP (1f).

INPUT-OUTPUT

The following d and f series instructions are used for input-output operations, and, as will be described, can be used in conjunction with a photoelectric high speed tape reader, an electromechanical high speed punch and/or the Flexowriter (with associated paper tape read-punch mechanism). Should the high speed units be inoperative, the computer will automatically refer to the Flexowriter for execution of these instructions.

N, as used below, indicates the number of times the given operation is to be performed; and will occupy only the right four (4) binary bits of the eight (8) required in any address syllable. N is any number between and including 0 and 8, (unless otherwise noted). If N = 0 it will be interpreted as 8; if N > 8, it will be interpreted modulo 8.

Of the left group of four (4) binary bits used in the address syllable (following the d and f series instructions) the high order bit is used to distinguish between the Flexowriter and either of the high speed tape units (read or punch). The next binary bit is used to indicate whether or not a conversion is to be made (from decimal to binary for input and from binary to decimal for output). The remaining two (2) bits of this group are ignored.

f1 HEXADECIMAL IN

f18N	Flexowriter
f10N	High speed reader

Shift A left (excluding P_o) four (4) binary places and copy into the four (4) right-hand bits the digit received from the input station designated. Perform this operation N times.

f1cN	Flexowriter
f14N	High speed reader

Convert the N decimal digits of the input to their binary equivalent. A must be cleared first!

f3 ALPHABET IN

f38N	Flexowriter
f30N	High speed reader

Shift A left (excluding P_o) six (6) binary places and copy into the second through seventh right-hand bits the code symbol received from the input station designated. Perform this operation N times, where $0 < N \leq 5$. If $N > 5$, the first N-5 inputs will be lost.

f5 HEXADECIMAL OUT

f58N	Flexowriter
f50N	High speed punch

Copy the four (4) left-hand bits of A to output station designated and shift A left (excluding P_o) four (4) binary places. Perform this operation N times.

f5cN	Flexowriter
f54N	High speed punch

Convert the binary number in A to an N-digit, decimal equivalent. This operation must be preceded by a long division (of the number to be converted) by $10^N + 1$, where the number (as a dividend) is in A with B cleared. The resulting quotient must then be put in A, followed by the desired f5 for conversion. ($N \leq 8$)

f7 ALPHABET OUT

f78N	Flexowriter
f70N	High speed punch

Copy the code symbol represented by the 26th through 31st bits of A to the designated output station, and shift A left (excluding P_o) six binary places. (6 zeros will come into the right-hand side of A). Perform this operation N times, where $0 < N < 5$. If $N > 5$, the last N-5 outputs will be spaces.

In the following, any hexadecimal digit can be used for X since it is ignored by the computer.

f9 SIGN IN

f98X	Flexowriter
f90X	High speed reader

Clear A and read the designated input station for the sign code (space or plus for positive, and minus for negative) and make P_o agree (1 if positive, 0 if negative).

d5	SIGN OUT	If the sign of A is positive ($P_o = 1$) transmit a space to the designated output station; if the sign of A is negative ($P_o = 0$) transmit a minus sign.
	d58X Flexowriter d50X High speed punch	
dd	NUMBER OUT	Copy the number represented by the four (4) right-hand bits of A ($P_1 \ldots P_4$) to the output station designated.
	dd8X Flexowriter dd0X High speed punch	

At the Flexowriter input-output station there are two three-position switches designated as Type and Punch switches. The forward positions (away from operator) are TYPE and PUNCH respectively; the center (vertical) position of both is OFF; and the backward position (toward operator) of both is COMPUTE.

These two switches, when in the COMPUTE positions, are used in conjunction with certain special instructions (described below) to edit the form of computer output to the Flexowriter. The TYPE and PUNCH positions, however, will override any of the form-editing instructions and cause an output to be typed and/or punched, according to their settings. In the OFF position, of course, all Flexowriter outputs are lost.

9b	TYPE	Type out information according to the d and f series output instructions, unless overruled by Flexowriter switch settings.
9d	PUNCH	Punch information, according to the d and f series output instructions, unless overruled by Flexowriter switch settings.
9f	BOTH	Type and punch information, according to the d and f series output instructions, unless overruled by Flexowriter switch settings.
99	NEITHER	Neither type nor punch information, regardless of d and f series output instructions, unless overruled by Flexowriter switch settings.

Thus, it can be seen, that with the use of the two PUNCH and TYPE switches together with the Flexowriter editing instructions, complete flexibility, with regard to form of output, can be maintained.

OPTIMUM ADDRESSING

When using instructions which require W and/or W/2 addresses, choice of the address determines whether or not an optimum condition exists. Optimizing, quite naturally, results in speedier computation and, therefore, an efficient coder will attempt to create this condition wherever possible. ALWAC III-E logic is of such a nature that two optimum conditions are possible; namely, SINGLY OPTIMUM or DOUBLY OPTIMUM.

Even numbered instructions, used singly, automatically create optimum conditions and the following three general rules will show other ways to bring them about.

(1) OPTIMUM (left half)

If the instruction appears in the left half of a word, an optimum address is one whose last digit is one greater, in time, than the address of the word in which the instruction appears.

Example: Instruction appears in address 05 (working channel I); optimum addresses are 06, a6, 16, b6 and addresses in the same relative positions in the other three working channels.
Fig. 4a

(2) DOUBLY OPTIMUM (whole word)

If an optimum address has been used with the instruction in the left half of a word (as above), an optimum address for the instruction in the right half is one whose last digit is three greater, in time, than the last digit of the word in which the instruction appears.

Example: Instruction appears in right half of word 05 (half-word 85, working channel I); optimum addresses are 08, 88, 18, 98 and relative addresses in the other working channels.
Fig. 4a

(3) SINGLE OPTIMUM (right half)

If a non-optimum address has been used with the instruction in the left half of a word, an optimum address for the instruction in the right half is one whose last digit is at least two greater, in time, than the last digit of the non-optimum address, but less, in time, than the address in which the next instruction appears, in the normal sequence (described on page 2).

Example: Instruction in left half of word 57 (working channel III) uses 6f for its non-optimum address; optimum addresses for the instruction in the right half are those whose last digits fall within the range of 1 to a, inclusive, i.e., 01, 81, 11, 91...1a 9a and relative addresses in the other working channels.
Fig. 4b

Note: If, however, the address in the left half of the word is optimum, or 1 greater than optimum, a singly optimum condition automatically exists.

Jump instructions are automatically optimum if no jump is executed. An optimum address for a jump instruction used in the left half of a word, when a jump is made, is one whose last digit is ~~four~~ three greater, in time, than the last digit of the word in which the jump instruction appears. If the jump instruction appears in the right half of a word and a jump is made, an optimum address is one whose last digit is four greater, in time, than the last of the address used with the instruction in the left half. If an even numbered instruction, used singly, appears in the left half, then an optimum address for a jump instruction in the right half is one whose last digit is five greater than the last digit of the address in which the jump instruction appears.

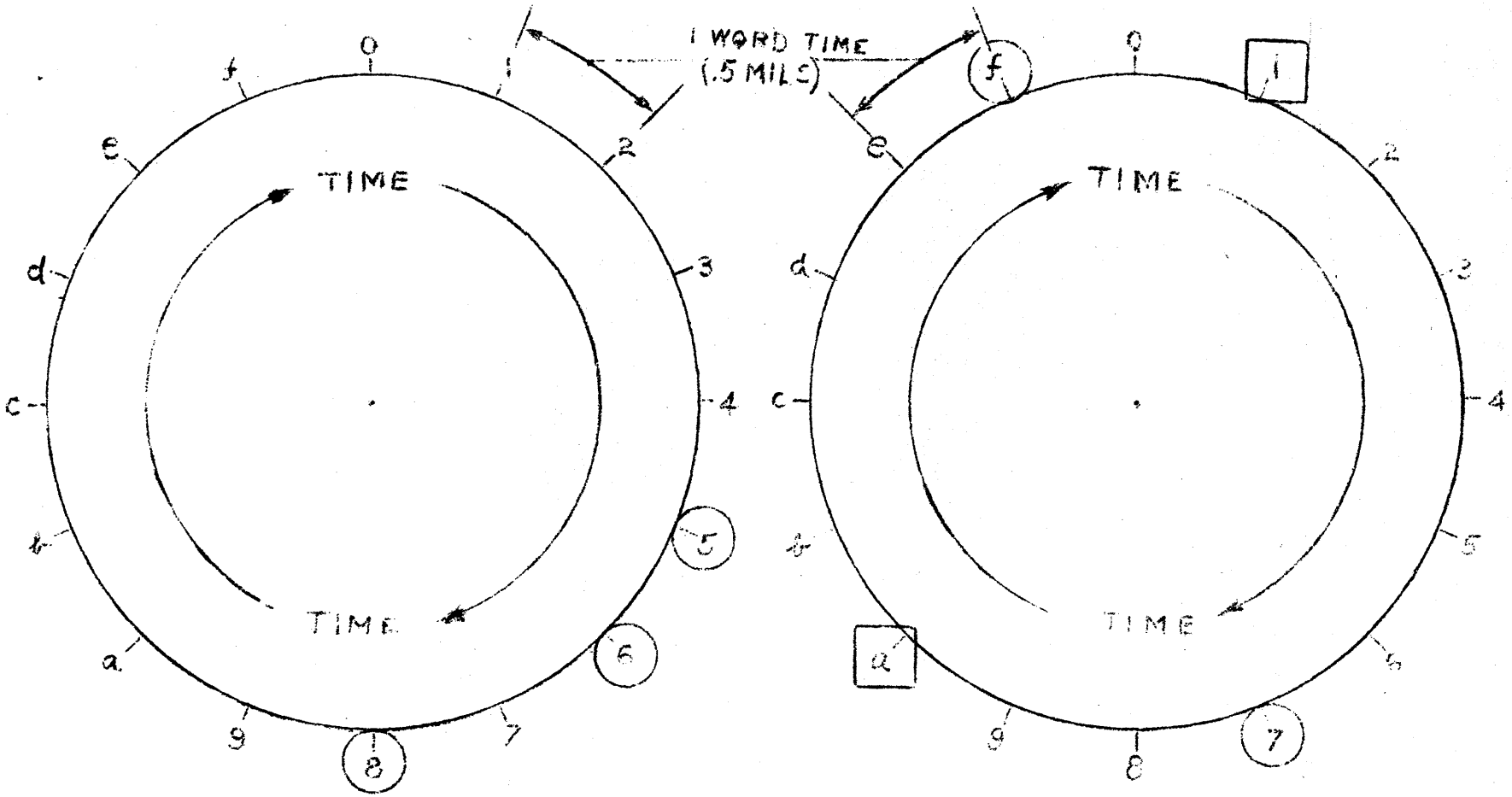


Figure 4
OPTIMUM ADDRESS DIAGRAM
Digits are LAST digits of addresses

a DOUBLY OPTIMUM
(1) and (2)

Left (5) Location of instruction
(6) Optimum for left instruction
(8) Optimum for right instruction if left has an optimum.

b SINGLY OPTIMUM

(f) A non-optimum address of instruction in left half of any word whose last address-digit is (7)
[1] through [a] are optimum addresses.

INSTRUCTION CODES ALWAC III-E

Sequentially listed with Basic Operation Times in Milliseconds

02	Reverse overflow	1.0
11	Jump	1.0-1.5
13	Control jump 1	1.0-1.5
15	Control jump 2	1.0-1.5
17	Count down	1.0-1.5
19	Non zero jump	1.0-1.5
1b	Stop	
1d	Less than zero jump	1.0-1.5
1f	Overflow jump	1.0-1.5
22	Round off	1.0
28	Clear A	1.0
2c	Absolute A	1.0
2e	Reverse Sign A	1.0
30	Exchange A and B	1.0
32	Copy B to A	1.0
34	Copy E to A	1.0
36	Exchange A and E	1.0
38	Copy D to A	1.0
3a	Exchange A and D	1.0
3e	Complement A	1.0
41	Copy W to B	1.0
49	Copy A to W	1.0
4d	Copy address to W	1.0
4f	Copy half to W	1.0
51	Overflow if |A| smaller	1.0
57	Copy W to E	1.0
5b	Copy W to D	1.0
61	Add	1.0
63	Minus Add	1.0
65	Minus subtract	1.0
67	Subtract	1.0
69	Exchange A and W	1.0
6d	Copy address to A	1.0
6f	Copy half to A	1.0
71	Extract (D)	1.0
75	Extract	1.0
79	Copy W to A	1.0
81	Copy to I	91.0 Av.
83	Copy to II	91.0 "
85	Copy to III	91.0 "
87	Copy to IV	91.0 "
89	Copy from I	107.0 "
8b	Copy from II	107.0 "
8d	Copy from III	107.0 "
8f	Copy from IV	107.0 "

99	Neither	24.0 Av.
9b	Type	24.0 "
9d	Punch	24.0 "
9f	Both	24.0 "
a1	Double shift right	0.5 + 0.5N
a3	Double shift left	0.5 + 0.5N
a5	Shift right	0.5 + 0.5N
a7	Shift left	0.5 + 0.5N
ab	Float	1.0 + 0.5N
b5	Copy M to A	9Ms Av
bd	Long add	1.0
bf	Long subtract	1.0
c3	Copy E to W	1.0
c5	Copy B to W	1.0
c7	Copy D to W	1.0
d5	Sign out	*
dd	Number out	*
e1	Add multiply by D	17.0
e3	Add multiply	17.0
e5	Multiply by D	17.0
e7	Multiply	17.0
e9	Long divide by D	17.0
eb	Long divide	17.0
ed	Divide by D	17.0
ef	Divide	17.0
f1	Hexadecimal in	*
f3	Alphabet in	*
f5	Hexadecimal out	*
f7	Alphabet out	*
f9	Sign in	*

* Flexowriter, 100 milliseconds per character
Photoelectric reader, 2.5 milliseconds per character
High speed punch, 17 milliseconds per character

To cause automatic modification of an instruction address by the E register subtract 1 from the related instruction code, which is normally an odd number. Such an alteration of the instruction code causes the left half of E (modulo 256) to be subtracted from the address. In its use as a tally, the E register is counted down (the 17 instruction), so that the effect of the subtraction is to assign successive addresses, in ascending order, to repetitive operations. For example, if the basic form of an instruction calls for adding word 28, and E contains 28, the instruction operates, when decreased by 1, on word 00 because of the subtraction of E from the address. When E is now counted down, the next occurrence of the same altered instruction will operate on word 01, because 27 is now subtracted from the address.

The following example shows the kind of economy in coding that can be effected by making use of the automatic address modification feature.

WITHOUT AUTOMATIC ADDRESS MODIFICATION

Arithmetic used: sum of eight numbers, a_0, a_1 . . . a_7

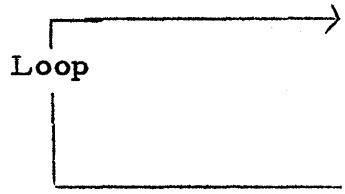
Stored Data

In 01	0 0 0 8 0 0 0 0
In 09	0 0 0 0 0 0 0 1
In 20, 21. . . 27	a_0, a_1. . . a_7

LOCATION OF INSTRUCTION		INSTRUCTION AND ADDRESS	OPERATION
00		57 01	Copies contents of 01 to E (left half)
80		41 04	Copies contents of 04 to B
04		28 00	Clears A register
84	→	61 20	Adds a_i to contents of A
08		bd 09	Adds 00000001 to B, increasing instruction address (of 61 instruction) by 1.
88		c5 04	Copies B to 04 putting the new instruction address in place for further addition.
	Loop		
0c	←	17 84	Counts E down 1 (left half) jumping to 84 if result is not zero.
8c		etc.	

WITH AUTOMATIC ADDRESS MODIFICATION

Same data storage except for 00000001 in 09, which is no longer needed.

LOCATION OF INSTRUCTION		INSTRUCTION AND ADDRESS	OPERATION
00		57 01	Copies contents of 01 to E. (left half)
80		28 00	Clears A register
04		60 28	Adds contents of 28 - E to A register. (Since the instruction in 04 is 60, E register (left half) is subtracted from 28.)
84		17 04	Counts E down 1 (left half) jumping to 04 if result is not zero. (The next 60 operation will be carried out on the contents of the subsequent word).
08		etc.	

Automatic address modification has saved five instruction spaces (or half words), and eliminated one operation plus two more per loop. The time saved, therefore, increases proportionately with the number of times through a given loop.

INSTRUCTION CODES, ALWAC III-E BY GROUPS

ARITHMETIC

61 Add
67 Subtract
63 Minus add
65 Minus subtract
bd Long add
bf Long subtract

e7 Multiply
e5 Multiply by D
e3 Add multiply
e1 Add multiply by D

ef Divide
ed Divide by D
eb Long divide
e9 Long divide by D

ACCUMULATOR

22 Round off
28 Clear A
2c Absolute value
2e Reverse A sign
3e Complement A

SHIFT

a1 Double shift right
a3 Double shift left
a5 Shift right
a7 Shift left
ab Float

INPUT-OUTPUT

f1 Hex. in
f3 Alphabet in
f5 Hex. out
f7 Alphabet out
f9 Sign in
d5 Sign out
dd Number out
9b Type
9d Punch
9f Both
99 Neither

COPY & EXCHANGE

69 Exhange A and W
49 Copy A to W
79 Copy W to A
b5 Copy M to A
30 Exchange A and B
c5 Copy B to W
41 Copy W to B
32 Copy B to A
3a Exchange A and D
c7 Copy D to W
5b Copy W to D
38 Copy D to A
36 Exchange A and E
c3 Copy E to W
57 Copy W to E
34 Copy E to A
4d Copy address to W
4f Copy half to W
6d Copy address to A
6f Copy half to A
71 Extract (D)
75 Extract

BLOCK COPY

81 Copy to I
83 Copy to II
85 Copy to III
87 Copy to IV
89 Copy from I
8b Copy from II
8d Copy from III
8f Copy from IV

JUMP & RELATED

11 Jump
13 Control jump 1
15 Control jump 2
17 Count down
19 Non-zero jump
1b Stop
1d Less than zero jump
1f Overflow jump
51 Overflow if |A| smaller
02 Reverse overflow

Two instructions can be doubled up if the first is an even-numbered instruction; but it must be made odd by adding 1. The second can be any instruction not requiring an address.

Odd numbered instructions will have their addresses automatically modified if the instruction is made even by subtracting 1.

ALWAC III-E MAGNETIC TAPE UNITS

Magnetic tape employed with the ALWAC III-E computer as high-capacity storage or high-speed input consists of the following units:

Buffer and Control Unit: Provides high-speed random access memory for magnetic tape blocks of 32 words (each 32 bits plus sign). Controls modes of operation of tape transports, and furnished interlock signals to the computer for maximum simultaneous utilization of search and rewind times by the computer program. Can control up to 16 tape transports.

Tape Transport Unit: Passes one-half inch magnetic tape past read-record head at 100 inches per second forward or backward under control of Buffer Unit. Records and reads pulses on seven tracks (4 information, one check, one clock, one block marker) at 100 pulses per inch. Starts and stops tape in approximately 10 ms. Rewinds at greater than 500 inches per second. Holds up to 32 bits of search and comparison information for locating an individual block on the tape.

The ALWAC III-E commands associated with the magnetic tape units, as well as operation times and modes are listed in detail below:

91 Instruction

91 0N	Rewind tape on transport N.
91 1N	Prepare to read from transport N.
91 2N	Set transport N to search in the first mode, and prepare to read.
91 3N	Set transport N to search in the second mode, and prepare to read.
91 4N	If transport N is searching, turn on overflow.
91 5N	Prepare to write in transport N.
91 6N	Set transport N to search in the first mode, and prepare to write.
91 7N	Set transport N to search in the second mode, and prepare to write.

93 Instruction

a. The least significant address digit chooses the transport to be used.

b. The most significant address digit selects the type of operation.

93 1N	Read the last block, if transport is set to read.
93 2N	Read the next block, if transport is set to read.
93 3N	Read this block, if transport is set to read.
93 5N	Write the last block, if transport is set to write.
93 6N	Write the next block, if transport is set to write.
93 7N	Write this block, if transport is set to write.

95 Instruction

The most significant address digit selects the type of operation.

95 1X	Copy the contents of the buffer to working storage IV.
95 2X	Copy the contents of working storage IV to the buffer.
95 3X	Exchange the contents of the buffer with working storage IV.

Time for Operations:

1. All 91 and 93 operations take 1 ms of computer time, unless computer must wait on interlocks.
2. 95 instructions take from 16 to 32 ms. unless computer must wait for interlocks.
3. Read/write operations take 43 to 86 ms. transport time.

Search Modes

1. When a search order is given, contents of the A register are stored in the transport addressed, after which the computer may continue with its program. This word is used in the search comparison. The tape transport starts forward, and compares the first word of each block with the given word. When the correct block is found, the transport positions the tape to read or write.

2. Search Mode I: Tape transport will choose the first block whose first word is greater than or equal to the given word.

3. Search Mode II: The tape transport will choose the first block which has the least significant hexadecimal digit of the first word equal to the least significant digit of the given word.

4. The 914N order turns on the overflow flip-flop if tape unit N is searching. This permits the computer to continue a program if the tape unit is still searching.

Read/Write Modes

1. The tape transport must be set to write, if a write order is given, or to read, if a read order is given. Alarm No. 2 will be turned on in the computer if this procedure is not followed.

2. When a 933N or 937N order is given, the tape unit moves forward, reads or writes the first block, and moves the tape into position to read or write the same block.

3. When a 932N or 936N order is given, the transport advances one block forward and proceeds as in the 933N or 937N order.

4. When a 931N or 935N order is given, the transport reverses one block and proceeds as in 933N or 937N.

ALWAC III-E CARD CONVERTER

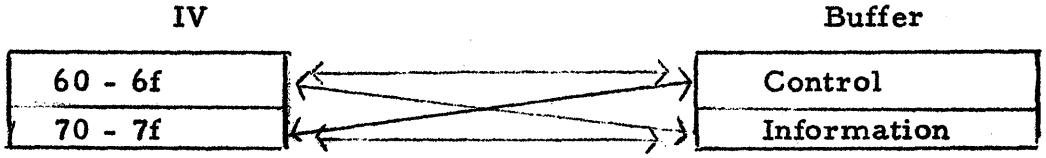
The ALWAC III-E can punch or read IBM cards at a rate of one hundred per minute when connected by its Card Converter attachment either to a 514 Reproducing Punch or to a pair of 523 Summary Card Punches --- one to read, and one to punch. Each word can be read or punched in any of three forms:

Alphabetic (up to five characters per word)

Hexadecimal (up to eight characters per word)

Decimal (up to eight digits per word, automatically converted binary-to-decimal or decimal-to-binary.)

The Card Converter utilizes an extra channel on the drum as a buffer storage for information being read or punched, so that computation may continue during the card cycle. This buffer channel is divided into a control half and an information half, either of which may be exchanged with either half of Working Channel IV:



The control half contains pulses that indicate the card columns which correspond to each word of information, and also pulses that indicate in which of the three forms each word is to be interpreted --- alphabetic, hexadecimal, or decimal.

Signs are all taken to be positive when interpretation is alphabetic. When interpretation is hexadecimal or decimal, negative signs are indicated by an X-overpunch in the same column as the least significant character of the word.

All card operations are accomplished with operation code 97. Different addresses are used with this code to specify the different operations. Each operation is performed in two steps:

1.) Exchange between Working Channel IV and buffer.

2.) Read or punch and convert if decimal.

The conversion and reading or punching is done according to what is in the control half of the buffer after the exchanges are over. Likewise, the information half is punched on a card or filled from a card after the exchanges are over, so that when a card has been read a subsequent 97 operation is required to place the information in Working Channel IV.

The effects of the different address digits are these:

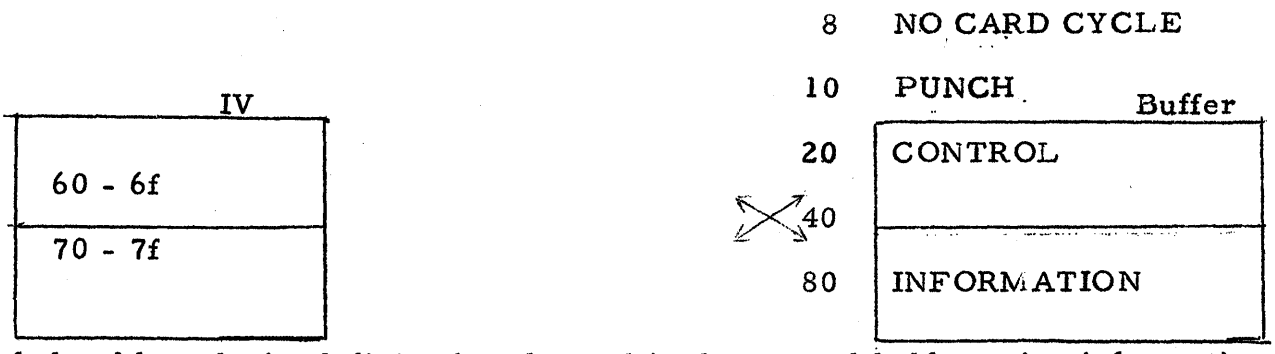
An 8-bit prevents the card machine from executing a card cycle.

A 10-bit causes punching and also causes all conversions to be from binary to decimal. If no 10-bit is present, reading will occur and all conversions will be from decimal to binary.

A 20-bit causes the control half of the buffer to be exchanged with half of Working Channel IV. The 40-bit indicates which half of Working Channel IV this is.

A 40-bit causes the upper half of the buffer to be exchanged with the lower half of Working Channel IV, and vice versa.

An 80-bit causes the information half of the control channel to be exchanged with half of Working Channel IV. The 40-bit indicates which half of Working Channel IV this is.



The left-hand hexadecimal digit of each word in the control half carries information about alphabetic, hexadecimal, or decimal interpretation. The other twenty-eight bits and the sign bit of words in the control half govern card column selection.

No indication is needed if an information word is to be interpreted alphabetically. If it is to be interpreted hexadecimally, then a 4 must be placed in the left-hand hexadecimal digit of the word in the control half which corresponds to the word numbered one less than the word in question.

For example, suppose that Working Channel IV is going to be straightforwardly exchanged with the buffer and punched. This will take a 97b0 operation. If it is desired to punch word 78 in hexadecimal form, then the left-hand hexadecimal digit of word 67 should be 4. If it is desired to punch word 70 in hexadecimal form, then the left-hand digit of word 6f should be 4.

An identical rule applies to decimal interpretation, except that the digit 8 should be used in place of the digit 4.

Decimal conversion is on an integer basis. When a number is to be punched in decimal form, it is interpreted as an integer, converted to decimal, and the eight least significant digits of this decimal number are stored in the eight hexadecimal positions of the word in question. Column selection for the punching of these digits is the same as for hexadecimal punching (described below.) Conversion from decimal to binary follows the exact reverse process.

By blocking the card cycle with an 8-bit in the address of the 97 operation, the Converter may be used to convert between binary and decimal for other forms of input and output.

Conversion takes some time, which must be allowed for in setting up the card machine interlocks. It sometimes happens that a second 97 instruction is given before the first one is finished, but the second card cycle can be started immediately provided conversion has proceeded far enough to be completed before the new information must be handled. To assure that no time is lost in these cases, the coder should count the number of words that are to be decimally interpreted, and add a 1 to the left-hand hexadecimal digit of the word this many positions down in the control channel. Consider the example given above. If three of the sixteen words are to be decimally interpreted, then 1 should be added to the left-hand digit of control word 62. If one or none of the words are decimal, a 1 should be added to the left-hand digit of word 60. The number of decimal words is of course just the number of 8's that appear in the left ends of control words.

If the procedure outlined is not followed, the card machine may run on alternate cycles, giving a card rate of fifty instead of one hundred per minute. The card machine will also run slow if successive 97 operations are separated by more than about four hundred and eighty milliseconds of routine.

There is not complete flexibility in the way card columns are assigned to words by the control channel, but any desired rearrangement of columns can be made on the wiring boards of the 523's or 514.

To understand the assignment of card columns by pulses in the control channel, it is necessary to bear in mind that the buffer storage is of a serial-serial recirculating type. The words are recirculated one after another in numerical order, and the binary digits within words are recirculated one after another, but in inverse order of significance. Thus the last bit in the control channel is the most significant bit of word 6f, and the first bit in the control channel is the sign bit of word 60.

With this time-relation established, successive card columns starting with column 1 are assigned to successive control pulses (not counting those in the left-hand hexadecimal character of any word.) Thus if control word 60 has a pulse in the sign position --- i. e., is positive --- then this pulse will correspond to card column 1. If word 60 is negative, the least significant bit in it will correspond to card column 1. If it is negative zero (with the possible exception of the left-hand hexadecimal digit) then the least significant bit in word 61 will correspond to card column 1.

In the information half of the buffer, alphabetic card columns correspond to six consecutive bits, while hexadecimal or decimal card columns correspond to four consecutive bits. These bits are the six (or four) immediately to the left of the bit corresponding in position to the control bit for that column. The coding of these six (or four) bits is as shown on the "ALWAC III-E Code Conversion Chart." Control bits may be spaced as widely as desired, but should not be placed closer together than six (or four) bits, or nonsensical results will occur.

For example, if it were desired to punch word 7a as eight hexadecimal columns, then word 69 would have a 4 in the left-hand position, and word 6a would be 08888888+ (the left-hand character being immaterial to word 7a, since it determines the form of interpretation of word 7b.)

After eighty pulses for the eighty card columns have been placed in the control channel, later pulses are disregarded and have no effect, with the exception of the left-hand hexadecimal character of the last word, which determines the form of interpretation of the first word.

The connections on the wiring boards of the 523's are as follows: Information to be punched from the computer is presented on the "Comp Mag or Ctr Tot Exit or M S Out" hubs and should be wired to the "Punch Magnets" hubs. Information to be read into the computer should be wired from the "Punch Brushes" to the "Comp Mag or Ctr Tot Exit or M S Out" hubs.

The connections on the wiring board of the 514 are as follows: Information to be punched should be wired from the "Comp Mag or Ctr Tot Exit or M S Out" hubs to the "Punch Magnets" hubs. Information to be read should be wired from the "Reproducing Brushes" to the "Selector 1" and "Selector 2" hubs.

In all cases it should be remembered that since columns are counted from right to left within a word, wiring must exchange end-for-end to present digits in their proper order.

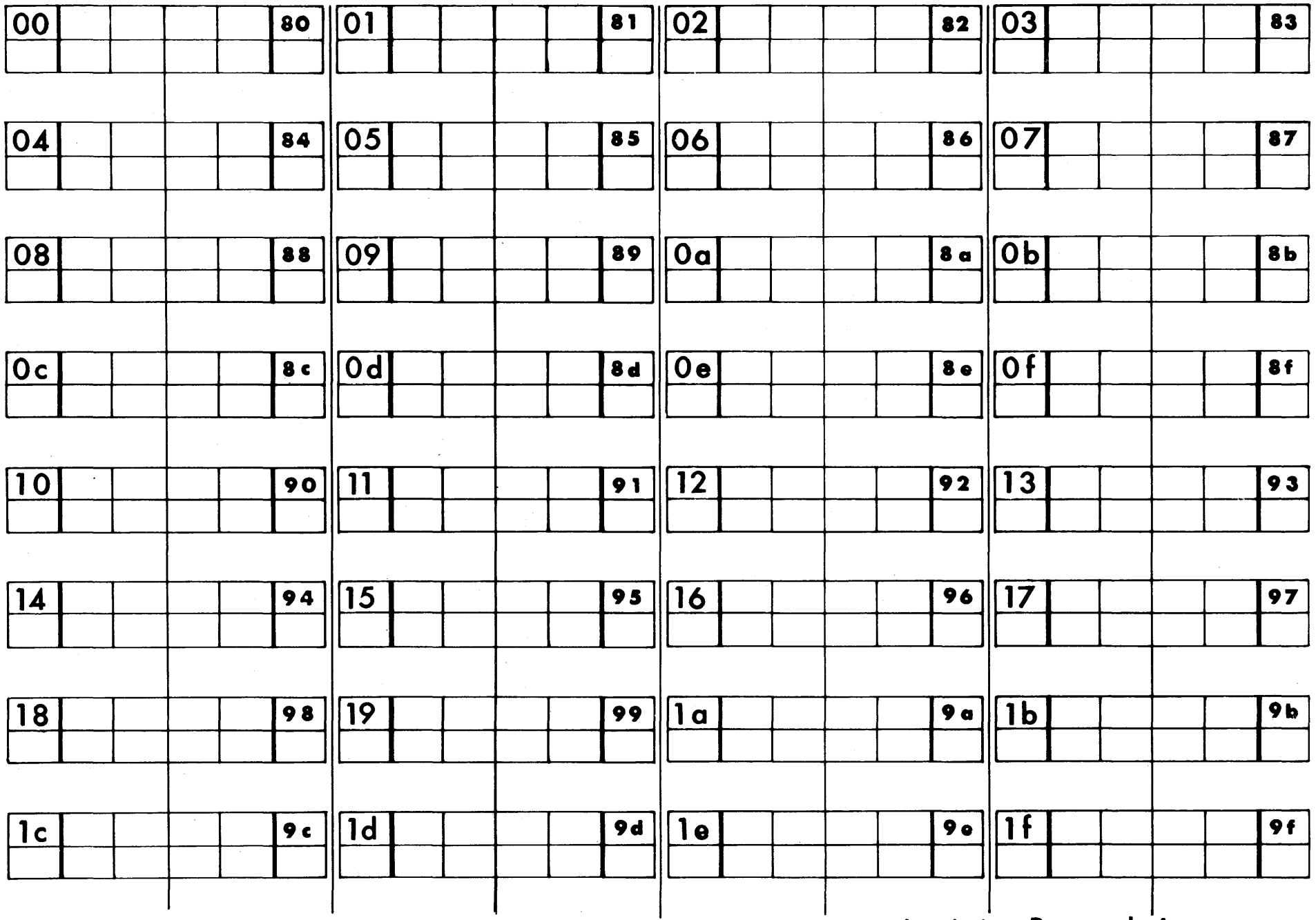
ALWAC III-E CODE CONVERSION

FLEXOWRITER PUNCH	FLEXOWRITER PRINT	HEX	ALWAC	CARDS	HEX
.3 5	Space	0	00 0000	Space	0
1 .3 5	° 1	1	00 0001	I	1
2.3 5	" 2	2	00 0010	2	2
12.3 5	+ 3	3	00 0011	3	3
. 5	= 4	4	00 0100	4	4
1 . 5	% 5	5	00 0101	5	5
2. 5	? 6	6	00 0110	6	6
12. 5	! 7	7	00 0111	7	7
.345	∑ 8	8	00 1000	8	8
1 .345	(9	9	00 1001	9	9
2.345	A a	A	00 1010		A
12.345	B b	B	00 1011	#	B
. 45	C c	C	00 1100	@	C
1 . 45	D d	D	00 1101		D
2. 45	E e	E	00 1110		E
12. 45	F f	F	00 1111		F
.3	G g		01 0000	&	
1 .3	H h		01 0001	A	
2.3	I i		01 0010	B	
12.3			01 0011	C	
.	Tape feed		01 0100	D	
1 .			01 0101	E	
2.			01 0110	F	
12.			01 0111	G	
.34			01 1000	H	
1 .34			01 1001	I	
2.34			01 1010		
12.34			01 1011	.	
. 4			01 1100	¤	
1 . 4			01 1101		
2. 4	_ -		01 1110		
12. 4	J j		01 1111		

FLEXOWRITER PUNCH	FLEXOWRITER PRINT	ALWAC	CARDS
6 .3 5	K k	100000	-
61 .3 5	L l	100001	J
6 2.3 5	M m	100010	K
612.3 5	N n	100011	L
6 . 5	O o	100100	M
61 . 5	P p	100101	N
6 2. 5	Q q	100110	O
612. 5	R r	100111	P
6 .345	Lower case	101000	Q
61 .345	Upper case	101001	R
6 2.345	Color shift	101010	
612.345	Code delete	101011	$
6 . 45	Tabulate	101100	*
61 . 45	Carriage return	101101	
6 2. 45	Back space	101110	
612. 45		101111	
6 .3	} 0	110000	0
61 .3	' /	110001	/
6 2.3	S s	110010	S
612.3	T t	110011	T
6 .	U u	110100	U
61 .	V v	110101	V
6 2.	W w	110110	W
612.	X x	110111	X
6 .34	Y y	111000	Y
61 .34	Z z	111001	Z
6 2.34	△ $	111010	
612.34	* |	111011	,
6 . 4	Stop	111100	%
61 . 4		111101	
6 2. 4	; ,	111110	
612. 4	: .	111111	

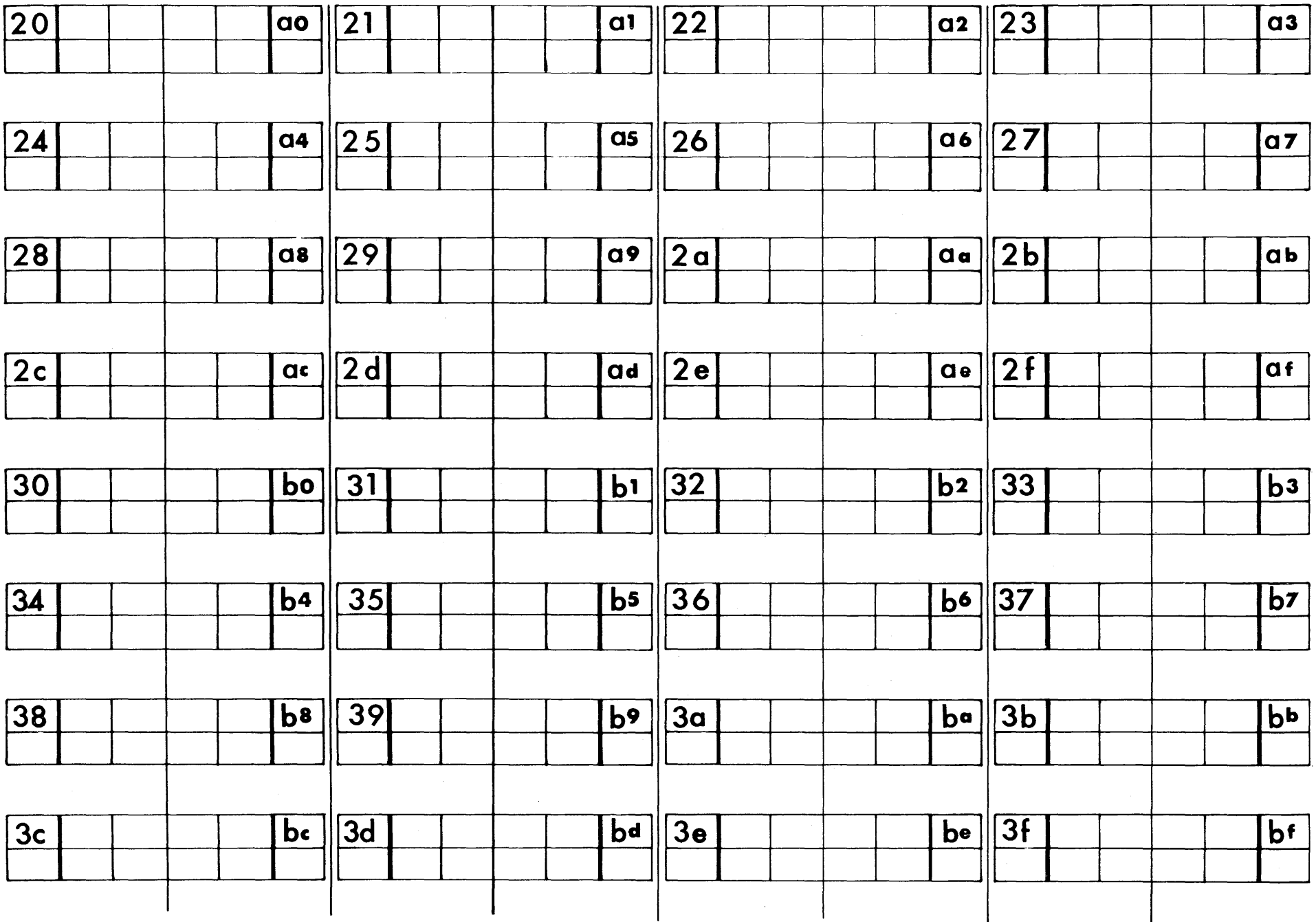
AB/nb 6/28/56

working channel I **no.** _____



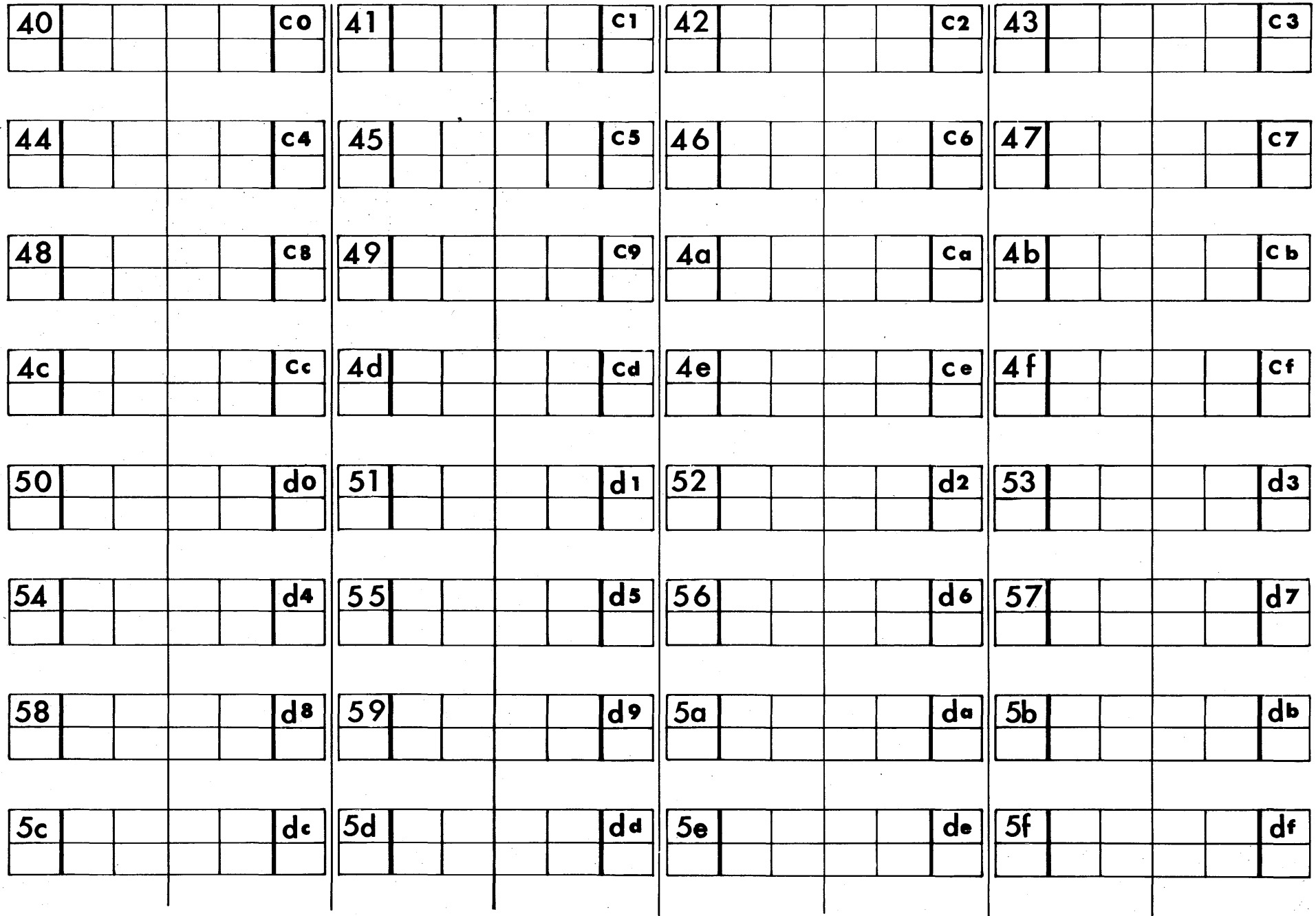
Logistics Research Inc.
Redondo Beach, Calif.

working channel II **no. _____**



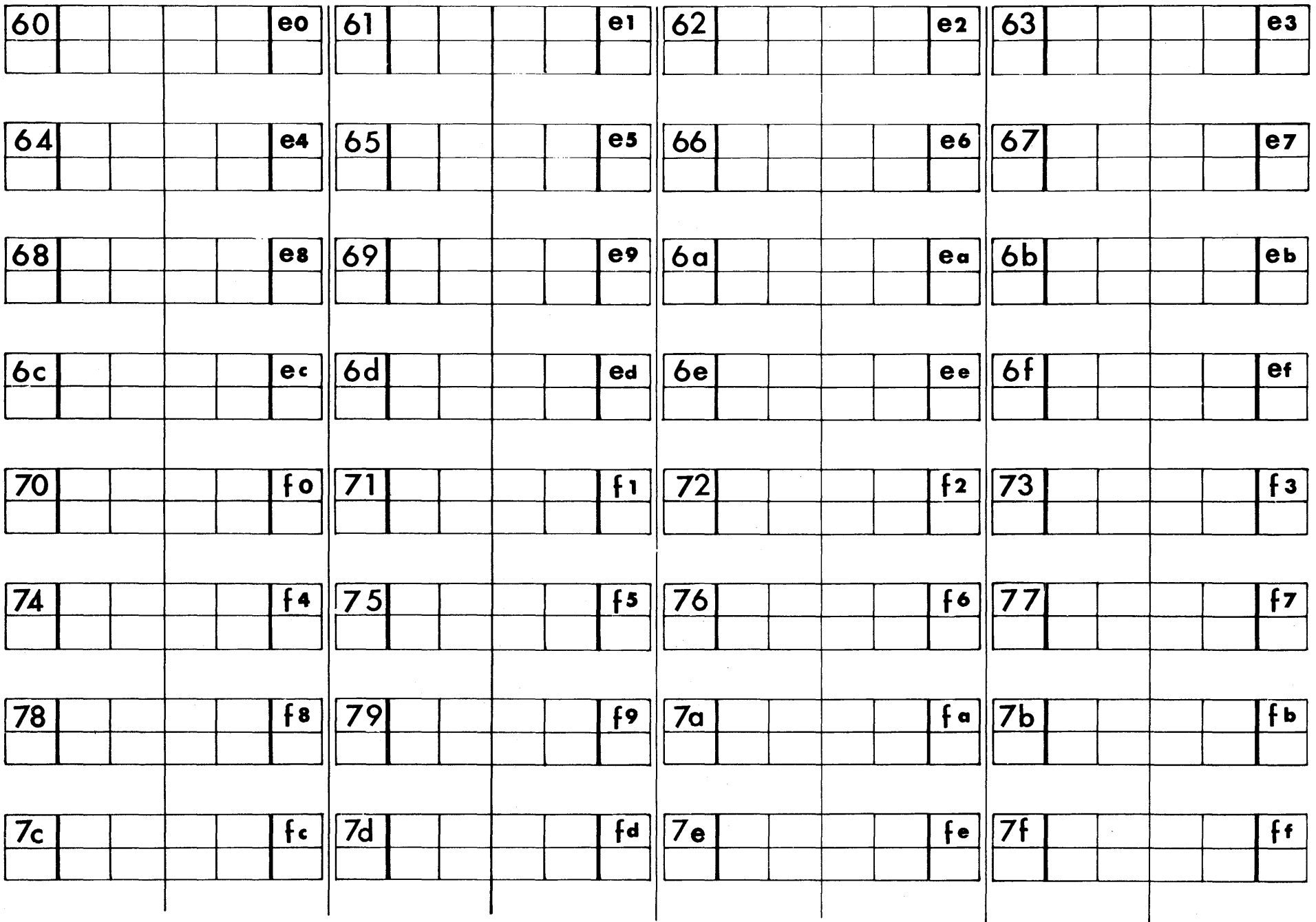
Logistics Research Inc.
Redondo Beach, Calif.

working channel III **no.** _____



Logistics Research Inc.
Redondo Beach, Calif.

working channel IV **no.** _____



Logistics Research Inc.
Redondo Beach, Calif.