John Corcoran

UNIVAC® SOLID—STATE 90

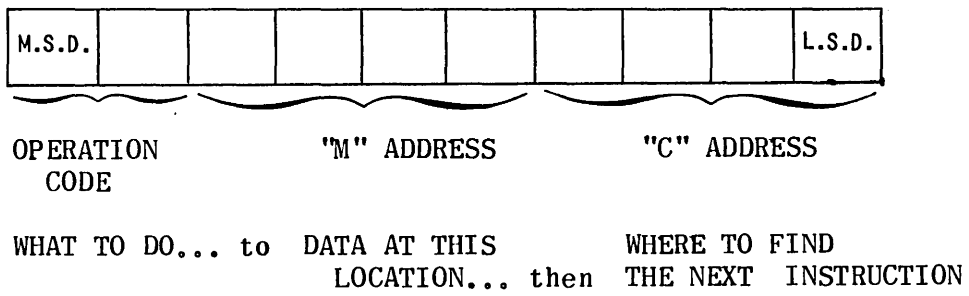
Instruction Codes

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

CONTENTS

GENERAL

The UNIVAC Solid-State 90 employs a 1½-address instruction code system, with one instruction per computer word. The sign digit is not used in defining instructions. The format of an instruction word is illustrated below:



The "m" address is usually the address of a word in storage. The "operation code" tells the computer what to do with this word; and the "c" address is the storage location of the next instruction word. These fields may have dif-derent significance for some special instructions, as noted in the instruction definitions.

When a word is transferred from a storage location or register, the contents of the storage location or register from which the word was transferred remain unchanged.

When a word is transferred into a storage location or register, the previous contents of the storage location or register are erased, except in the 20 and 35 instructions.

INSTRUCTION CYCLE

A three or four step cycle is associated with each instruction, depending upon whether an operand is required from drum storage. If setting-up the instruction is considered the starting point, the instruction cycle is:

Staticize the Instruction	Search for the Operand	Execute the Instruction	Search for the next Instruction
(1)	(2)	(3)	(4)

(1) STATICIZE THE INSTRUCTION:	The instruction located by the previous search (4) is transferred from the drum location to the Static Register (operation code only) and register C (the entire word). This step requires one word time which is 17 microseconds.
(2) SEARCH FOR THE OPERAND:	If the first address part of the instruction does not refer to a drum storage location or a register, this step is ignored and no time is required. If it does refer to drum storage, the address of the next available storage location on the drum is compared with the first address part of the contents of rC every word time until a match is obtained. Register C contains the entire instruction. If an operand is required from storage, this step requires a minimum of one word time and a maximum of 200 word times.
(3) EXECUTE THE INSTRUCTION:	The operation indicated in the instruction is performed. The time required depends upon the type of operation to be performed.
(4) SEARCH FOR THE NEXT INSTRUCTION:	Every word time the address of the next available storage location on the drum is compared with the second address part of the contents of rC until a match is obtained. This step requires a minimum of one word time (when minimum latency coding is used) and a maximum of 200 word times.

In describing the 39 instructions used with the UNIVAC Solid-State 90, the following conventions are used:

m	represents a storage location or register.
c	represents the address of the next instruction.
(m)	represents the contents of a storage location or register.
rA	represents register A
rL	represents register L
rC	represents register C
rX	represents register X

The timing for each instruction is shown in the right-hand column following the description of the instruction. Timing is shown as the number of word times required to execute the entire instruction cycle in minimum latency. Timing in milliseconds can be obtained by multiplying the word times given by 0.017.

<u>TRANSFER INSTRUCTIONS</u>		<u>Word Time</u>
25 m c	Transfer (m) to rA.	4
60 m c	Transfer (rA) to m.	4
05 m c	Transfer (m) to rX.	4
65 m c	Transfer (rX) to m.	4
30 m c	Transfer (m) to rL.	4
50 m c	Transfer (rL) to m.	4
77 m c	Transfer (rA) to rL. Ignore m.	3
06 m c	Clear rX to zero and set its sign storage to plus. Ignore c. Next instruction at m.	3
31 m c	Clear rL to zero and set its sign storage to plus. Ignore c. Next instruction at m.	3
26 m c	Clear rA to zero and set its sign storage to plus. Ignore c. Next instruction at m.	3
36 m c	Clear rA to zero and leave its sign unchanged. Ignore c. Next instruction at m.	3
86 m c	Clear rA and rX to zero. Set the signs of rA and rX to that of rL. Ignore m. Next instruction at c.	14
23 m c	Transfer (rC) to rA. Next instruction at m.	2

<u>Examples:</u>

1. Place the contents of 4309 into registers A and L. Next instruction is at 378. Begin in line 350.

<u>Solution:</u>

```
0350   25  4309  0361      (4309) ——> rA
0361   77  0000  0378        (rA) ——> rL
```

2. Clear memory location 3172 to all zeros. Leave registers A and X undisturbed. Next instruction at 174.

<u>Solution:</u>

```
0167   31  0000  0170      Ø's  ——> rL
0170   50  3172  0174      (rL) ——> 3172
```

ADDRESSING REGISTERS

Often, operands or instructions needed in a program are available in registers A, X or L. In such instances these words can be obtained directly from the registers. The addresses of the three registers are as follows:

000K	(1101)	refers to rA
000T	(1111)	refers to rX
000Y	(0110)	refers to rL

When a register address is given in the "m" portion of an instruction, the operand to be obtained is in the register specified. If a register address is given in the "c" portion of an instruction, it indicates that the next instruction to be executed is in the register designated.

One restriction is placed on the use of register addressing. A register address may not be used with a 60, 50 or 65 instruction.

Examples:

1. Starting in memory location 1301, place the contents of 4511 in registers A, X and L.

Solution:

```
1301   25  4511  1313       (4511) ──► rA
1313   05  000K  1317         (rA) ──► rX
1317   30  000T  1321         (rX) ──► rL
```

2. Place the contents of rL in rA. Next instruction is in rX. Start at 0021.

Solution:

```
0021   25  000Y  000T       (rL) ──► rA, next in-
                                 struction is in rX.
```

ARITHMETIC INSTRUCTIONS

	ARITHMETIC INSTRUCTIONS	Word Time
70 m c	Add algebraically (m) to (rA) and place the sum in rA.	5
75 m c	Subtract algebraically (m) from (rA) and store the difference in rA.	5
85 m c	Multiply (rL) by (m) and store the 10 most significant digits of the product in rA and the 10 least significant digits in rX. Both rA and rX will have the sign of the product. Multiplication can be shortened for multipliers having less than 10 significant digits by placing a sentinel* just to	5 plus the number of digits in the multiplier plus the sum of these digits

* Code 0101 or 1101 (- or K)

85 m c (Con't)		Word Time
	the left of the most significant digit of the multiplier, m. This sentinel stops the multiplication after the last significant multiplier digit is used.	
55 m c	Divide (m) by (rL) and put the unrounded 10 digits of the quotient with sign in rA and the remainder in rX. The remainder has the sign of the dividend. If the divisor is zero, less than or equal to the dividend, overflow occurs.	20 plus the sum of the odd digits of the quotient plus the sum of the tens complements of the even digits.

Examples:

1. Add the contents of 1511 and 4115. Place the sum in 4441.

Solution:

```
0009   25  1511  0013        (1511) → rA
0013   70  4115  0018        (4115)+(rA) → rA
0018   60  4441  0043          (rA) → 4441
```

2. Reduce the contents of memory locations 4591 and 4691 by 1. Place the results in 4600 and 4700.

Solution:

```
 2085   30  2087  2089       0000000001 → rL
[2087   00  0000  0001]
 2089   25  4591  2093       (4591) → rA
 2093   75  000Y  2098         (rA)-(rL) → rA
 2098   60  4600  2102         (rA) → 4600
 2102   25  4691  2143       (4691) → rA
 2143   75  000Y  2148         (rA)-(rL) → rA
 2148   60  4700                (rA) → 4700
```

3. Multiply the contents of 4391 by the contents of 4100. Place the 10 most significant digits of the product in 4720, and the 10 least significant digits in 4730.

Solution:

```
0191   30  4391  0193
0193   85  4100  0253
0253   60  4720  0272
0272   65  4730
```

OVERFLOW

Two types of overflow can occur in the computer. The first is an arithmetic overflow, indicating that the result of an arithmetic operation does not fall within the range $-1 < x < +1$. The second is an overflow signifying an abnormal condition in an input or output unit (e.g., printer out of paper, a card jam in the HSR or RPU, etc.) In both cases the computer continues to operate. However, when such a condition does occur, the next instruction is not found at c but at c + 1.

It should be noted that c and c + 1 must be in the same band on the drum. If the c address is the last location in any band (e.g., 0199, 0399, etc.,) the c + 1 address is the first word of the same band (e.g., 0000, 0200, etc.) When the address of the next instruction is a register address, proceed to that register whether or not overflow occurs.

Example:

Add (3178) to (3182). Place the sum in 3210. If overflow occurs place 0000000001 in 3200 and (rA) in 3210. In both cases, the next instruction (following storage of the sum) is in 0271.

```
Solution:      0176   25  3178  0180      (3178)→rA
               0180   70  3182  0185      (3182) + (rA)→rA
               0185   60  3210  0271       (rA)→3210
               0186   05  0188  0190      0000000001→rX
              [0188   00  0000  0001]
               0190   65  3200  0185       (rX)→3200
```

LOGICAL INSTRUCTIONS

		Word Time
20 m c	Superimpose the 1 bits of (m) onto (rA) and leave the result in rA. The sign of rA is undisturbed.	4

Examples:

```
1. Superimpose m =0020406080
            on rA =0103050709
    Result in rA =0123456789


2. Superimpose m =0000093800
            on rA =9873100000
    Result in rA =9873193800
```

Note:

This superimposition is performed on a bit by bit basis on each digit individually. Many combinations are possible using this instruction. For example, superimposing a 4 (0100) on a 1 (0001) produces a minus (0101), or an 8 (1011) and a 2 (0010) produce an 8 (1011).

		Word Time
35 m c	Superimpose the 0 bits of (m) onto (rA) and leave the result in rA. The sign of rA is undisturbed.	4

Examples:

```
1.  Superimpose m =0TTTTTTTTT
            on rA =1111122222
    Result in  rA =0111122222


2.  Superimpose m =00TTT000TT
            on rA =1234567896
    Result in  rA =0034500096
```

Note:

This instruction is like the 20 m c since it also operates on a bit by bit basis. A zero (0000) in m will erase a whole digit in rA; a T (1111) will retain it.

32 m c	Shift (rA) to the right n places into rX which also is shifting to the right into rA. The sign positions are not involved in this shift. n can vary between 0 and 10, and is a single digit inserted in the next to most significant digit position of m.*	3 + n
37 m c	Shift (rA) to the left n places losing the most significant digits and bringing in zeros in the least significant digit positions on the right. n can vary from 0 to 10 and is a single digit inserted in the next to most significant digit position of m. The sign of rA is not distrubed.*	3 + n

* The special character K (1101) is used to represent 10.

In any of the shift orders, if n is represented by an undigit combination, the following shifts will occur:

0110	2
0111	3
0101	5
1110	7
1111	8
1101	10

		Word Time
00 m c	Skip to next instruction at address m. Ignore c.	2
87 m c	If (rA) is algebraically greater than (rL), the next instruction is in m; if not, the next instruction is in c.	3
82 m c	If (rA) equals (rL), then the next instruction is in m: if not, the next instruction is in c.	3

Examples:

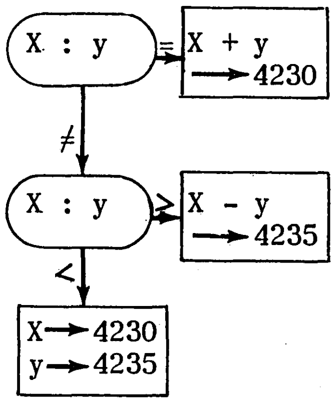
1. 4392 contains the value "A" 4400 contains the value "B". Place the smaller in rX.

```
Solution:          0290   25  4392  0294        A⟶rA
                   0294   30  4400  0302        B⟶rL
                   0302   87  0305  0306      test A against B
           (A>B)   0305   05  000Y                  B⟶rX
           (A<B)   0306   05  000K                  A⟶rX
```

2. 0900 contains X
   0910 contains y

   If X = y, add the 2 numbers and store the sum in 4230. If X > y, subtract y from X and store the difference in 4235. If X < y, merely store X in 4230 and y in 4235.

   Solution:



Solution: (Con't)

```
        0098   25  0900  0102      (900)──►rA
        0102   30  0910  0112      (910)──►rL
        0112   82  0116  0115    {  rA:rL  for equality
        0115   87  0119  0118    {  rA:rL  for magnitude
  X<y   0118   60  4230  0132       X──►4230
        0132   50  4235             Y──►4235
  X=y   0116   70  000Y  0121       X + y──►sum
        0121   60  4230               sum──►4230
  X>y   0119   75  000Y  0124       X-y
        0124   60  4235             difference──►4235
```

67 m c Stop the computer. The computer stops with the stop instruction in rC, but before the search for the next instruction is started. Normally, when the computer is restarted, the first step will be to search for the next instruction specified by the c address. In this case the m digits are ignored and may be used as a code to indicate the reason for stopping. However, if desired, the m address may be used as an alternate restart location by depressing the "m" button on the control panel.

TRANSLATE INSTRUCTIONS

Data enters the computer from punched cards in Remington Rand code. It may be moved from place to place within the computer in this code by instructions. Data must be in Remington Rand code to be correctly punched in output cards or printed. However, the arithmetic operations 70 m c, 75 m c, 85 m c, and 55 m c and the logical operation 87 m c will not give correct results unless the data is in machine code.

The following translate instructions permit the translation of data from one code to the other:

		Word Time
12 m c	Send (rA) and (rX) through the Remington Rand code-to-Machine code translator and deposit result in rA and clear rX to zeros. The sign of rA and rX remain unchanged. rA must contain the unprimed word of the card image and rX the primed word. m is ignored.	3
17 m c	Send (rA) through the Machine code-to-Remington Rand code translator and deposit the two results in rA and rX. The signs of both results are positive. rA will contain the unprimed word and rX the primed word. All Machine code zeros are translated to Remington Rand code punching zeros. m is ignored.	3

The results of either the 12 m c or 17 m c instruction are listed in Table 1 on page 10.

UNIVAC SOLID-STATE 90
(Print Code is Same as Card Code)

Card Code to Machine Code

	Card Code		Machine Code
	Unprime	Prime	
Character	5310	XX97	
0	0001	0000	0000
1	0010	0000	0001
2	0010	0010	0010
3	0100	0000	0011
4	0100	0010	0100
5	1000	0000	1000
6	1000	0010	1001
7	0000	0001	1010
8	0000	0011	1011
9	0000	0010	1100
A	1010	0010	1011
B	1010	0000	1001
C	0001	0001	1010
D	1101	0000	1011
E	0101	0000	0011
F	0010	0011	1011
G	1000	0001	1010
H	0100	0001	1011
I	1100	0000	1011
J	1110	0000	1011
K	1100	0010	1101
L	0001	0010	1100
M	1001	0000	1000
N	1001	0010	1001
O	0110	0000	0011
P	0110	0001	1011
Q	1100	0001	1011
R	0010	0001	1011
S	1010	0001	1011
T	0100	0011	1111
U	1001	0001	1010
V	0101	0010	0100
W	0101	0001	1011
X	0001	0011	1011
Y	0110	0010	0110
Z	1000	0011	1011
Space	0000	0000	0000
:	0110	0011	1111
,	1101	0010	1101
$	1111	0010	1111
-	1101	0001	1011
#	1011	0001	1011
*	0011	0000	0001
%	1011	0000	1001
;	1110	0011	1111
/	1100	0011	1111
+	1010	0011	1011
.	1110	0010	1111
&	1111	0001	1011
'	0111	0011	1111
(	1001	0011	1011
)	1110	0001	1011

Machine Code to Card Code

Machine Code		Card Code	
Bit. Combo.		Unprime	Prime
		5310	XX97
0000	(0)	0001	0000
0001	(1)	0010	0000
0010	(2)	0010	0010
0011	(3)	0100	0000
0100	(4)	0100	0010
1000	(5)	1000	0000
1001	(6)	1000	0010
1010	(7)	0000	0001
1011	(8)	0000	0011
1100	(9)	0000	0010
0101		0110	0010
0110		0110	0010
0111		0100	0010
1101		0000	0010
1110		0000	0011
1111		0000	0011

Table 1. Code Translation Table

INPUT OUTPUT INSTRUCTIONS (READ-PUNCH UNIT)

Four instructions direct the operation of the Read-Punch Unit. They are:

		Word Time
81 m c	Transfer the output card images (in the output interlace pattern) from the m band to the Punch Buffer. When the transfer is completed, the computer is free to operate on other instructions. The Read-Punch Unit will then punch the data from the Punch Buffer area into the card in the punch station. It then reads the cards now in both read stations, storing their images in the Read-Punch Input Buffer in the read interlace pattern. Finally, all cards are advanced one card station to the right in the Read-Punch Unit. m must be a multiple of 200 (i.e., 000, 0200, 0400, etc.) For minimum latency this instruction should be in storage location 0198 + 201n.	203

Note:

If an abnormal condition exists (card jam, empty input hopper, etc.) overflow occurs and the next instruction is found in c+1.

		Word Time
46 m c	Wait until the Read-Punch Input Buffer is loaded, then transfer the input card images to band m. (m must be a multiple of 200.) For minimum latency, this instruction should be in storage location 0198 + 201n.	203
22 m c	This instruction permits the program to test the status of the Read-Punch Input Buffer. If the buffer is loaded, (rC) is transferred to rA and the next instruction is found at m. If the buffer is not loaded the next instruction is found at c. rA is not altered in this case.	3 if c address taken, otherwise 4
57 m c	Select output stacker #1 (Sort). m is ignored. This instruction must be given within 116 ms after the Read-Punch Input Buffer is loaded if it is to operate on the card at the second read station. Otherwise stacker #0 is automatically selected.	3

INPUT INSTRUCTIONS (HIGH-SPEED READER)

Four instructions direct the operation of the High-Speed Reader:

		Word Time
72 m c	Pull a card into the continuously moving rollers of the feed. The card will be read at each station, in turn, and the data stored in the buffer band. The computer is free to operate on other instructions during the moving and reading of the cards. If the HSR is interlocked, the contents of rC are sent to rA and the next instruction is at m.	3 if c address taken, otherwise 4

Note:

If an abnormal condition exists (full output stacker, empty input hopper, etc.) overflow occurs and the next instruction is found in c+1.

96 m c	Wait until the High-Speed Reader Buffer is loaded, then transfer this data from the buffer to the memory band m (m must be a multiple of 200). For minimum latency this instruction should be placed in cell 0198 + 201n.	203
42 m c	This instruction permits the programmer to test the status of the High-Speed Reader Buffer. If the buffer is loaded (rC) is transferred to rA and the next instruction is found at m. If the buffer is not loaded, the next instruction is found at c. rA is not altered in this case.	3 if c address taken, otherwise 4
47 m c	Select output stacker designated by m. If m = 0000 the #0 stacker is selected, if m = 0100 the #1 stacker is selected, and if m = 0200 the #2 stacker is selected. To operate on the card at the second read station, this instruction must be given within 120.8 ms after the image is available in the buffer. If not, the card will enter the previously selected stacker.	3

Note:

If a 96 instruction is given with only one card present in either read station, the buffer interlace locations of the station not occupied will read all binary 1's.

PRINTING INSTRUCTIONS (HIGH-SPEED PRINTER)

The following instructions govern the printing operation:

		Word Time
11 m c	Wait until the previous printer operation is completed, then advance the paper "y" lines. While the paper advance is taking place, transfer the data from the m band print interlace to the print buffer band. The line is printed upon completion of the paper advance. The computer is released for other operations as soon as the buffer band is loaded. The contents of rA and rX will be altered during the transfer of data to the buffer band.	592

The two most significant digits of m specify the print interlace band. The two least significant digits of m specify y. Normally $00 \leq y \leq 49$. However, it is possible to advance the paper as many as 79 lines in the following manner:

Ky	where y = 0 to 9	- moves paper 50-59 lines
Yy	where y = 0 to 9	- moves paper 60-69 lines
Ty	where y = 0 to 9	- moves paper 70-79 lines

For minimum latency, the 11 instruction should be in location 0198 + 789n.

16 m c	Wait until the previous printer operation is completed then advance the paper y lines. Once paper movement is started the computer is free for other operations. y is the same as defined in the 11 instruction above.	4

Note:

Overflow can occur on either the 11 m c or 16 m c instructions if an abnormal condition exists in the High-Speed Printer (out of paper, paper jam, etc.)

27 m c	This instruction tests the status of the printer. If a print or paper advance is in process the next instruction is selected from c, otherwise the next instruction is at m and (rC) are transferred to rA.	3 if c address is taken, otherwise 4

15

		Word Time
62 m c	This is the zero suppression instruction. It is used to suppress printing (or punching) of non-significant zeros or commas. Zero suppression is done to the words in the print (or punch) interlace before the print (or punch) instruction is given. The instruction works as follows: The primed word of the pair must be in rX and the unprimed word in rA. Each of the following digit combinations,	4

Unprime	0001	1101
Prime	0000	0010
	(zero)	(comma)

found to the left of the first non zero or comma digit is replaced by binary zero. This is a non-printing and non-punching code. m is ignored.

ADDRESS MODIFICATION

If an m or c address exceeds 4999, the computer will recognize the address as follows:

1. The band addressed can be calculated by subtracting 5000 from the address. Thus, a reference to 6800 would pertain to a word in the 1800 band.

2. The word addressed can be one of four within the specified band. The interval between words is 50 (i.e. A reference to 8247 would refer to 3247, 3297, 3347 or 3397. All these words are in the 3200 band, 50 words apart.)

3. The particular word that would be affected is the first one to appear on the drum, relative to the instruction address.

The following examples indicate this procedure:

in line 0098	25	7952	$(2902) \longrightarrow$ rA
in line 0198	25	7952	$(2802) \longrightarrow$ rA
in line 0077	60	6101	$(rA) \longrightarrow$ 1101
in line 0329	60	6101	$(rA) \longrightarrow$ 1151

The memory locations falling within the 9000 through 9999 range will be interpreted as fast access bands. The actual band interpreted can be computed by subtracting 5000 from the address. Fast access characteristics prevail with these bands.

GENERAL

"Minimum Latency Coding" is a technique by which addresses for instructions and data are assigned in a manner which minimizes memory reference time. The extent to which these techniques are applied depends on the application being coded.

DRUM LAYOUT

The drum provides for the storage of 5000 10-digit "words" and their signs, in 25 bands of 200 words each.

1. The location of each word on the drum is uniquely determined by a four digit address, from 0000 to 4999.

2. All addresses from 0000 to 3999 refer to fast access.
All addresses from 4000 to 4999 refer to high-speed access.

3. In fast access, there is one reading head per band. Addresses differing by an exact multiple of 200 word times are equivalent (i.e., have the same angular position with respect to the reading heads) and may be used interchangeably in coding without affecting instruction execution time.

Since:	0254 - 0054 = 200	Then:	0054 and 0254 have equivalent access times
	1236 - 0436 = 800 = 4 x 200		0436 and 1236 have equivalent access times
But:	0186 - 0045 = 141	Then:	0045 and 0186 are not equivalent
	3473 - 1836 = 1637		1836 and 3473 are not equivalent
	2379 - 2229 = 150		2229 and 2379 are not equivalent

In high-speed access there are four reading heads per band. Addresses differing by an exact multiple of 50 are equivalent and may be used interchangeably without affecting instruction execution.

Since:	$\left.\begin{array}{l}\underline{42}94 - \underline{42}44 = 50 \\ \underline{43}44 - \underline{42}44 = 100 = 2 \times \underline{50} \\ \underline{43}94 - \underline{42}44 = 150 = 3 \times \underline{50}\end{array}\right\}$	Then:	These locations are equivalent addresses in the same band (in high-speed access only).
But: and	$\left.\begin{array}{l}4736 - 4633 = 103 \\ 4814 - 4635 = 179\end{array}\right\}$		These are not equivalent.

4. For each memory access in a program, there are a total of 40 possible locations available for minimum latency coding.

Thus: Fast Access (one per band)	1 x 20 =	20
High-Speed Access (four per band)	4 x 5 =	20
	TOTAL	40

High-speed access is normally used for data storage (input output interlaces, constants, etc.) but may be used for instructions.

BASIC INSTRUCTION CYCLE

There is a three or four step cycle associated with each instruction, depending on whether an operand is required from drum storage. These steps, and the times required are:

1. Staticize the instruction	-	1 word time
2. Search for operand (if needed)	-	1 to 200 word times
3. Execute the instruction	-	variable, depending on operation
4. Search for next instruction	-	1 to 200 word times

The instructions are divided into three types as follows:

1. Instructions requiring one operand from memory.
2. Instructions requiring reference to a buffer.
3. Instructions requiring neither an operand from memory, nor reference to a buffer.

REFERENCE TO MEMORY

Instructions requiring an operand from memory are: Transfer from memory to a register, transfer from a register to memory, arithmetic, superimpose, and extract instructions.

Remarks:

a. A minimum of two word times must always be allowed for staticizing the instruction and search for the operand.

b. If T = total word times for completion of a given instruction of this type, then T - 2 word times must be allowed for execution and search for the next instruction phases of the operation.

c. Additional time needed, either for operand search, or search for next instruction, must be added to the minimum times as determined in a and b.

d. Maximum time for staticizing the instruction and searching for the operand is 201 word times (See Basic Instruction Cycle, above).

REFERENCE TO A BUFFER

Instructions requiring reference to a buffer are: 46, 96, 81, 11.

Remarks:

a. Each of these transfers is initiated at the 0198 level on the drum. Thus 0198 level locations are minimum latency positions for each of these instructions. Placement at any other level adds to execution time.

b. Total transfer time to or from the card buffers is 203 word times. This means that the next instruction may be located in: 0198 + 203 = 0401 (or any equivalent level location) for minimum latency.

c. The Advance and Print (11) instruction requires 591 word times so the next instruction would be placed in 0198 + 591 = 0789 (or its equivalent) for minimum latency.

Instructions requiring neither reference to memory or memory include, comparisons, tests, zero suppress, transfer (rA) to rL, shift (right or left), High-Speed Reader card cycle, select stacker.

Remarks:

a. Times given for these instructions are minimum latency total execution times.

REFERENCE TO A REGISTER

Since no drum search is required when the operand or next instruction is in a register, such addressing is automatically in mimimum latency.

The following examples are given as illustrations of mimimum latency coding:

```
1.  0136  250138  0140 )
    0136  250138  1940 (     4 word times each
    0136  254138  1740 (
    0136  254088  1740 )     Not ••• 4088 1090

2.  1198  814200  1001 )
    1198  814200  0401 )     203 word times

3.  1286  320400  1293 )
    1286  320400  0293 )     7 word times
    1341  821344  1544       3 word times for both paths
```

OVERCAPACITY PUNCHING

GENERAL

Occassionally, extra digits above and beyond the normal 90 available are needed on a card. When such a condition arises, overcapacity punching may be used.

These punches are placed in the zero punching area of each column. The number of columns (or zero punch positions) required to indicate the digit, is dependent upon its potential range of values.

For example, assume that a one-digit code is to be punched in the zero area as an addition to the normal 90 columns. If the range of this code is from zero to one, only one punch is required. The presence of a punch would indicate a one. The absence of a punch would indicate a zero. If the range of the code is from zero to nine, more punches are needed. It could be punched in a bi-quinary form covering four punch positions, (a nine punched in four adjacent zero columns would be 1100, an eight 1011, a seven 1010, etc.) It might be punched in six zero positions, each one corresponding to an actual row on the card (i.e., the first position represents the zero row, 2nd = 1, 2 row, 3rd = 3, 4 row, 4th = 5, 6 row, 5th = 7, 8 row, and 6th = the 9 row.) This system facilitates punching as wide a range of values as is available in the normal punching mode.

Any mode of punching is valid as long as the program interprets the overpunches correctly.

The program must interrogate the overpunch in the unprimed portion of an input word before translation. The punches will fall into the zero-bit of a digit. If the program is to produce output overpunches, they must be placed in the zero-bit of each digit of the unprimed word as well.

Since alphabetic punching requires the use of the zero-punch area, the positions chosen for overcapacity punching must contain purely numeric information. If a zero exists among the numerics, it must be a space or non-punching zero.

EDITING AN INPUT OVERCAPACITY PUNCH

An example of editing an input overcapacity punch follows:

Given:

Memory location 4221 contains the unprimed portion of an input word. Overcapacity punches have been placed in the last 4 digit positions of this word. They are in bi-quinary form. The punch in digit 7 represents the 5-bit, digit 8 represents the 4-bit, digit 9 represents the 2-bit and digit 10 the 1-bit.

Problem:

Edit these 4 bits into the Most Significant Digit Position of rA.

MEMORY LOCAT'N	OP.	M	C	
0019	25	4221	0023	
0023	35	0025	0027	*Isolate 4 overcapacity punches from unprimed portion of word.*
[0025	00	0000	1111]	
0027	70	0029	0032	*Add constant which places bits in 5, 4, 2 and 1 positions if appropriate.*
[0029	00	0000	4310]	
0032	35	0034	0036	*Erase all but 5, 4, 2 and 1 positions*
[0034	00	0000	5421]	
0036	06	0039		
0039	32	0100	0043	
0043	20	000T	0047	
0047	32	0100	0051	
0051	37	0100	0055	
0055	20	000T	0059	
0059	32	0200	0064	*Buff all bits into one digit position.*
0064	37	0200	0069	
0069	20	000T	0073	
0073	32	0300	0079	
0079	37	0300	0086	
0086	20	000T	0090	
0090	35	0092	MAIN PROG	
[0092	T0	0000	0000]	

EDITING AN OUTPUT OVERCAPACITY PUNCH

An example of editing an output overcapacity punch follows:

Given:

Register X contains a word in the form 000000000X
where X is any number from zero to nine.

Problem:

Edit this for overcapacity bi-quinary punching and place it in the four least significant digits of the word in 4221.

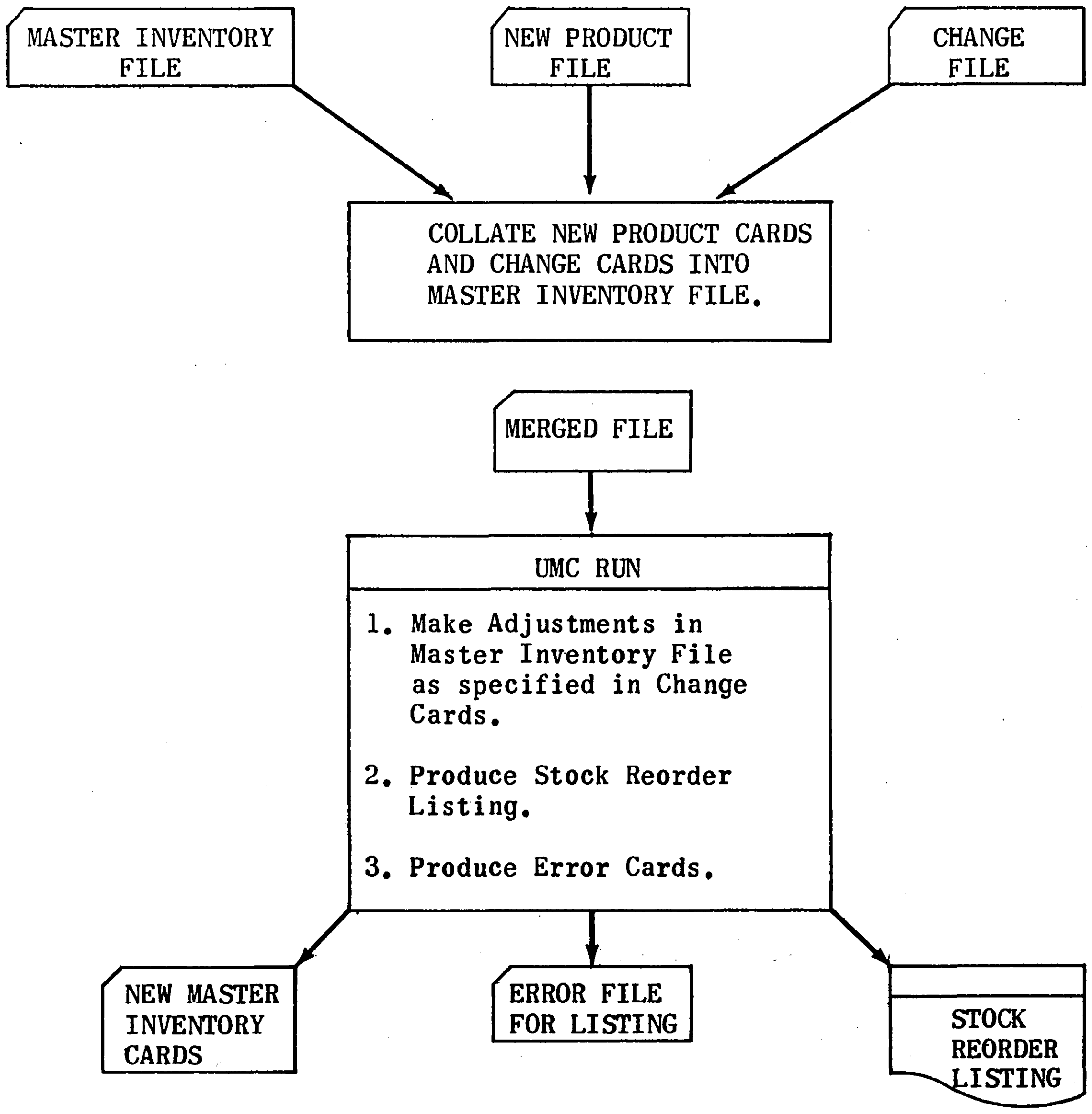
MEMORY LOCAT'N	OP.	M	C	
0120	25	000T	0124	
0124	70	0126	0129	
[0126	00	0000	0005]	
0129	35	0131	0133	*Edit 5-bit into punching position*
[0131	00	0000	0010]	
0133	37	0200	0138	
0138	77		0141	
0141	25	000T	0145	
0145	35	0147	0149	
[0147	00	0000	0004]	
0149	70	0151	0154	
[0151	00	0000	0006]	*Edit 4-bit into punching position*
0154	35	0156	0158	
[0156	00	0000	0010]	
0158	37	0100	0162	
0162	20	0004	0166	
0166	77		0169	
0169	25	000T	0173	
0173	35	0175	0177	*Edit 2-bit into punching position*
[0175	00	0000	0002]	
0177	70	0179	0182	
[0179	00	0000	0008]	
0182	35	0184	0186	*Edit 2-bit (cont.)*
[0184	00	0000	0010]	
0186	20	0004	0190	
0190	77		0193	
0193	25	000T	0197	
0197	35	0199	0210	*Edit 1-bit into punching position*
[0199	00	0000	0001]	
0201	20	0004	0205	
0205	20	4221	0223	*Buff overcapacity punches into unprimed portion of word.*
0223	60	4221	MAIN PROG.	

GENERAL

This is a problem designed to show various techniques used when programming the UNIVAC Solid-State 90. Much of it is concerned with editing input information from the High-Speed Reader and preparing output information for the Read-Punch Unit and the High-Speed Printer. There is a minimum of actual processing involved since this would vary so widely from problem to problem.

The sample itself is a simple one. Master Inventory Cards and Change Cards are read into the computer. Where a Change Card refers to a given Master Card, this charge is applied and a new Master Card is punched. At the same time a listing is kept on the High-Speed Printer of any items which must be reordered to keep a sufficient quantity on hand.

A complete description of the run follows:


MASTER INVENTORY FILE
NEW PRODUCT FILE
CHANGE FILE
COLLATE NEW PRODUCT CARDS AND CHANGE CARDS INTO MASTER INVENTORY FILE.
MERGED FILE
UMC RUN
1. Make Adjustments in Master Inventory File as specified in Change Cards.
2. Produce Stock Reorder Listing.
3. Produce Error Cards.
NEW MASTER INVENTORY CARDS
ERROR FILE FOR LISTING
STOCK REORDER LISTING

EXPLANATION OF FILES

A. INPUT

1. Master Inventory File - The Master Inventory File reflects the current status of a given item of inventory. The file is maintained in ascending sequence by its identifier, which consists of the plant and stock numbers.

   Each Master Inventory Card contains the following 12 fields:

Plant	(2 characters): Designates the plant in which the item is stocked.
Stock Number	(8 characters): Numeric.
Amount on Hand	(8 characters): Quantity on hand at a given time.
Description	(22 characters): Alpha-numeric description of the item.
Reorder Amount	(8 characters): The amount to be reordered when the inventory falls low.
Reorder Point	(8 characters): Item must be reordered when amount on hand plus the amount on order falls below this point.
Date	(6 characters): This field reflects the date on which the item was last changed. If the item has not been altered, the date here represents the date the item was incorporated into the file. (Mo.,Da., Yr.)
Unit of Measure	(4 characters): The unit by which an item is ordered, Doz., Each, Gals., etc.

Lead Time, Days	(2 characters): The number of days which an item must be ordered prior to the date on which it is needed.
Unit Cost Price	(6 characters): The price of a unit when bought or manufactured by the company.
Unit Sale Price	(6 characters): The price of a unit when sold by the company.
On Order Amount	(8 characters): The amount of items on order.

2. Change File - The change file consists of cards representing changes to be applied to items of the Master Inventory File. Each change will specify the following:

   1. Which item is to be changed.
   2. Which field of the card is to be changed.
   3. The nature of the change.

   A change card reaches the Processor immediately following its corresponding Master Card on the High-Speed Reader.

   Each change card contains the following four fields:

Plant:	Same as Master Card.
Stock Number:	Same as Master Card.
Change Key	(4 characters): Specifies which field of the Master Card is to be changed. This key will be represented by one of the following:

0004:	Change Reorder Amount
0005:	Change Reorder Point
0008:	Change Unit Cost.
0009:	Change Unit Selling Price.

0040:	Change Unit of Measure.
0050:	Change Lead Time, Days.
0600:	Stock received, add to Balance on Hand.
0700:	Stock sold, to be subtracted from Balance on Hand. (See Error File.)

<u>New Information</u> (8 characters): When a field of the Master Card is to be changed, it will be replaced by the information in this field of the Change Card, and a new Master Card will be punched.

When a field of the Master Card is to be added to or subtracted from, the amount of the change will appear in this field, and a new Master Card will be punched.

B. OUTPUT

1. <u>Updated Master Inventory Card</u>: Represents all changes to the inventory and will be collated into the Master File on the next run.

2. <u>Error Card:</u> One Error Card is produced each time one of the following situations occurs:

   a. If the amount sold results in a negative on-hand amount, the charge is not to be applied.

   b. If a Change Card is encountered whose identifier does not match any identifier in the Master Inventory File, the change cannot be made.

   c. If an invalid code number is given, the change cannot be made.

Each Error Card produced will contain the following three fields:

<u>Plant:</u>	Same as previously discussed.
<u>Stock Number:</u>	Same as previously discussed.

<u>Type of Error</u> (10 characters): In the case of error "a", the 10 characters should be:

OVERSOLDss (where s equals spaces)

In the case of error "b", the 10 characters should be:

NOsMASTERs (where s equal spaces)

In the case of error "c", the 10 characters should be:

WRONGsKEYs (where s equal spaces)

3. <u>Stock Reorder Card:</u> Each time a reorder is required, a list will be printed on the High-Speed Printer. This happens each time a new on hand amount, plus the amount on order, falls below the reorder point.

   Each page will contain <u>one</u> Header Line and <u>12</u> Body Lines. The Date will be the current date.

XX:	Page Number.
SSSSSSSS:	Stock Number.
PP:	Plant Number.
Description:	Of the item to reorder.
AAAAAA:	Reorder Amount.
UUUU:	The unit of measure
LLLL:	Lead time, days.

C. PRINTING FORMAT

<u>Header Line:</u>

```
DD Mon YEAR                 Stock Reorder File                Page XX
```

<u>Body Item:</u>

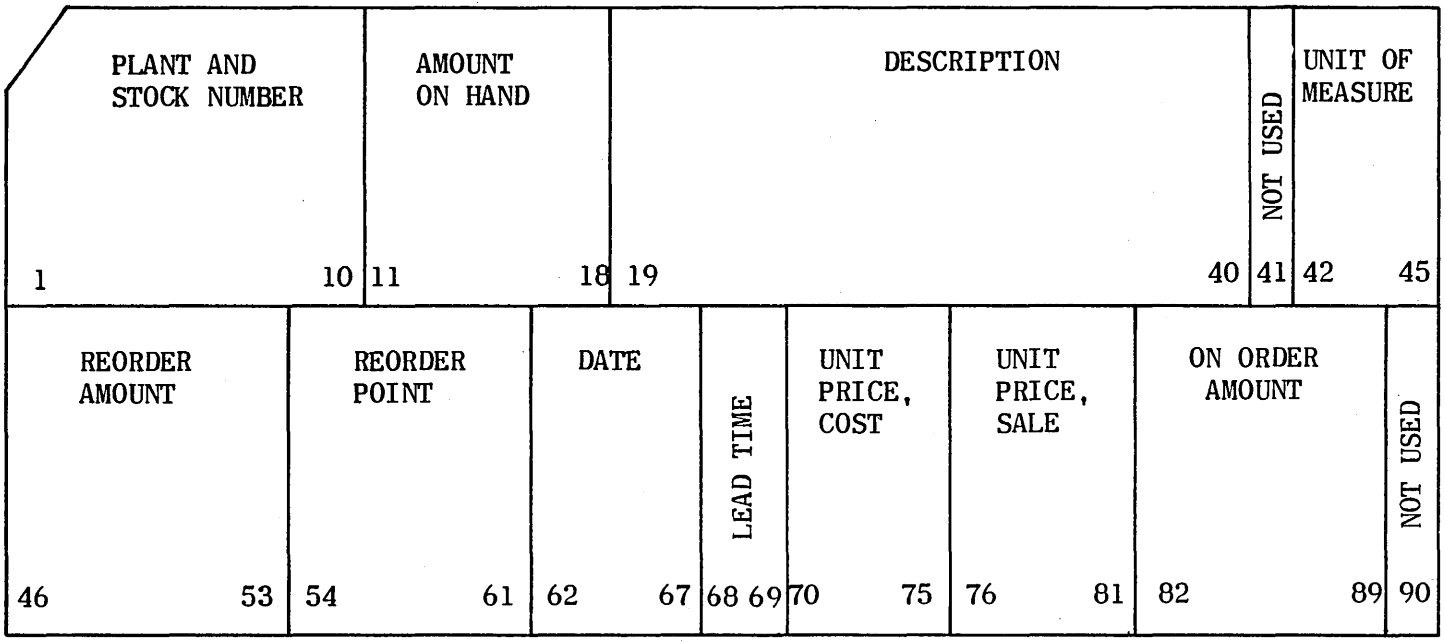
```
No.  SSSSSSSS      Description ---- AAAAAA UUUU
Plant      PP      DD Mon Year               LL
```

The D, X, S, A, P, and L fields will be zero suppressed.

Listing to be double spaced.

This run is basically an Inventory Updating. A given Master Card may have multiple change, or none.

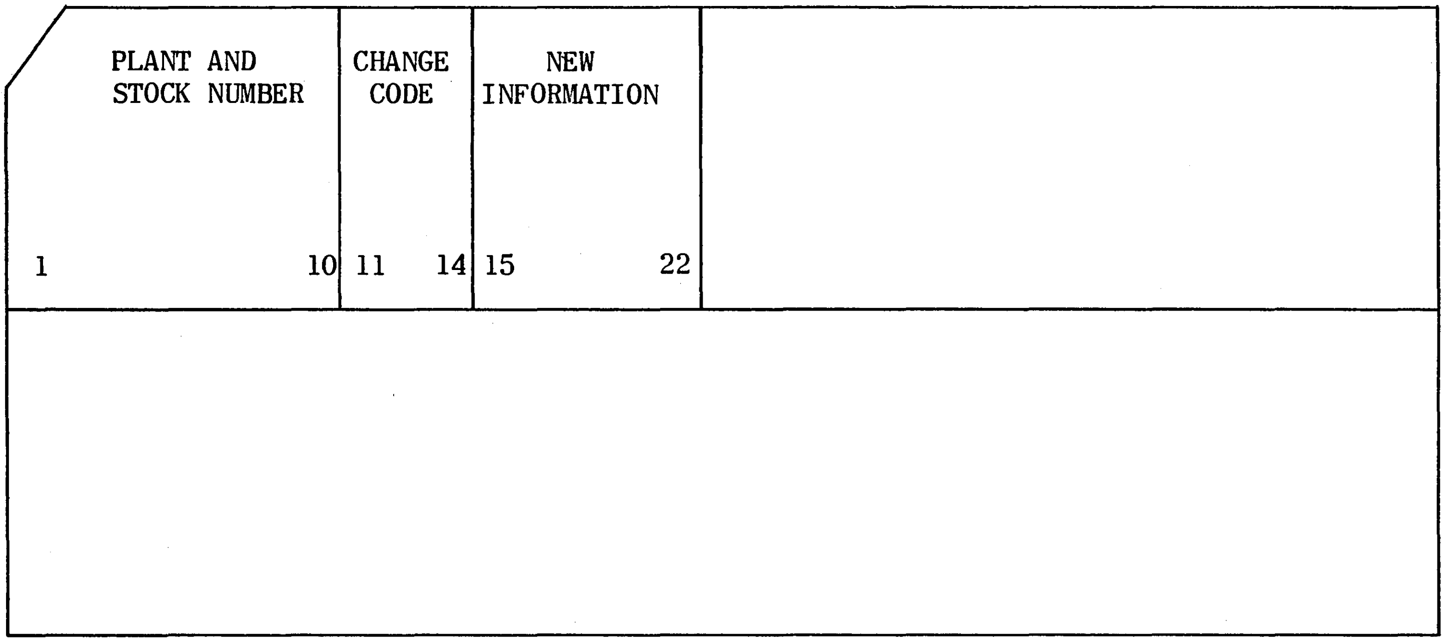
At the start of the run, the current date is keypunched and added to the rest of the deck. The Date should be put into each updated Master Card being changed.



PLANT AND STOCK NUMBER	AMOUNT ON HAND	DESCRIPTION	NOT USED	UNIT OF MEASURE
1 10	11 18	19 40	41	42 45

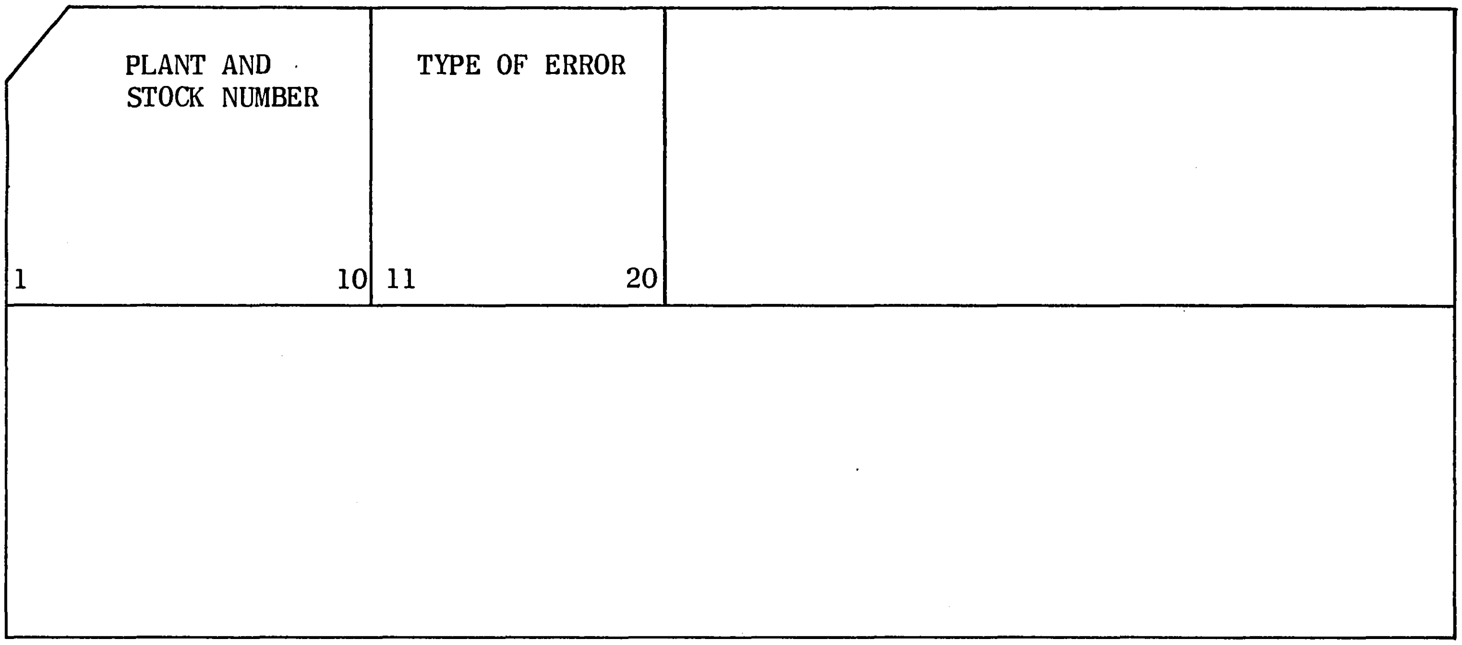
REORDER AMOUNT	REORDER POINT	DATE	LEAD TIME	UNIT PRICE, COST	UNIT PRICE, SALE	ON ORDER AMOUNT	NOT USED
46 53	54 61	62 67	68 69	70 75	76 81	82 89	90

SAMPLE MASTER INVENTORY CARD



PLANT AND STOCK NUMBER	CHANGE CODE	NEW INFORMATION	
1 10	11 14	15 22	

SAMPLE CHANGE CARD



SAMPLE ERROR CARD

INPUT-OUTPUT ROUTINES

There are three routines in the memory along with the one on the following pages. These are the High-Speed Reader Routine, the Read-Punch Routine and the High-Speed Printer Routine. They are entered in various portions of the Inventory Run. Their functions are:

1. High-Speed Reader Routine.

   a. Feeds a card.

   b. Reads a card (i.e., unloads the buffer to an interlace).

   c. Compares first and second reading of each card.

   d. Transfers a card image from a Reserve Storage Area to a Working Storage Area.

2. Read-Punch Routine.

   a. Moves an image from Working Storage to an Output Interlace.

   b. Punches a card.

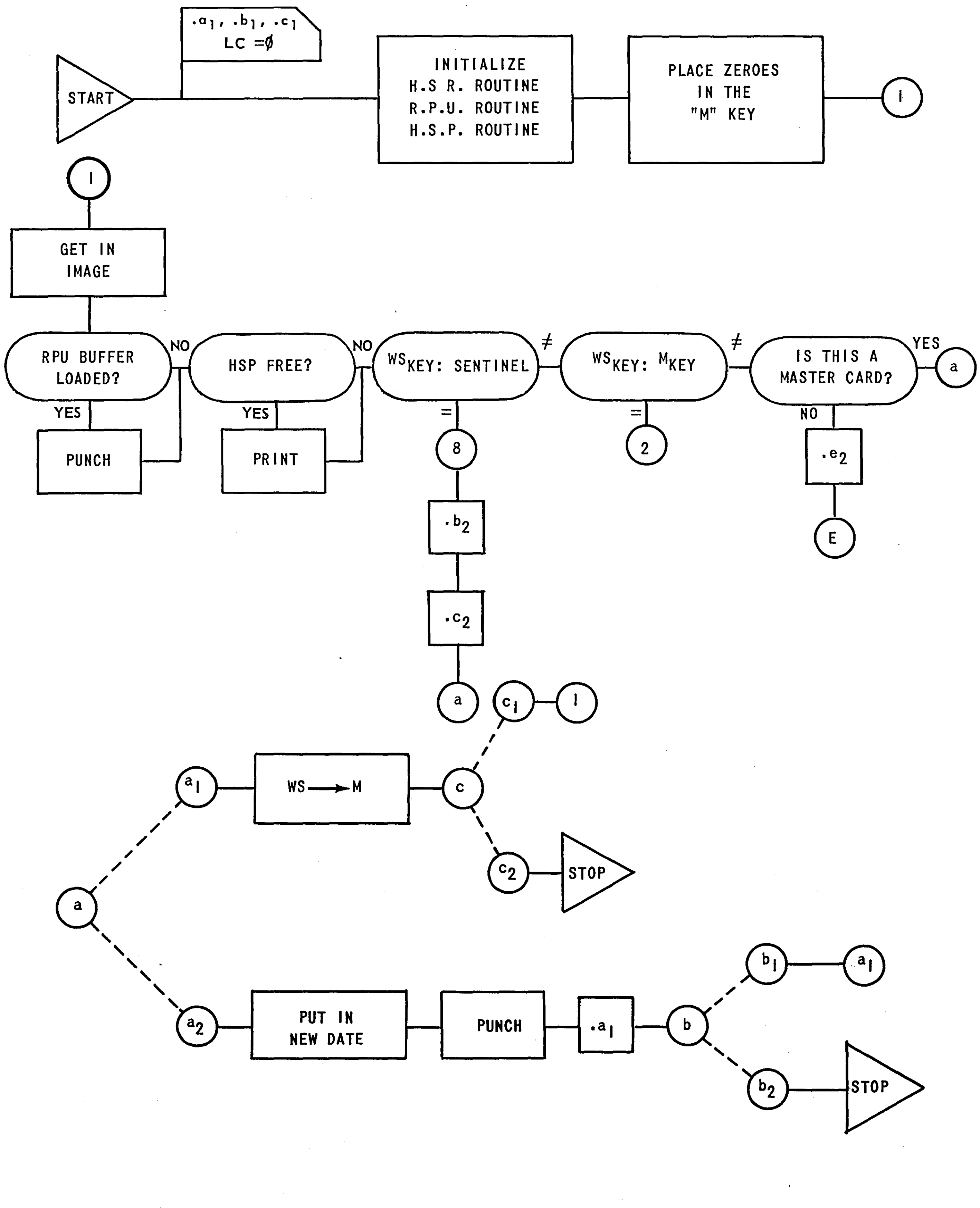
   c. Check reads the card punched.

3. High-Speed Printer Routine.

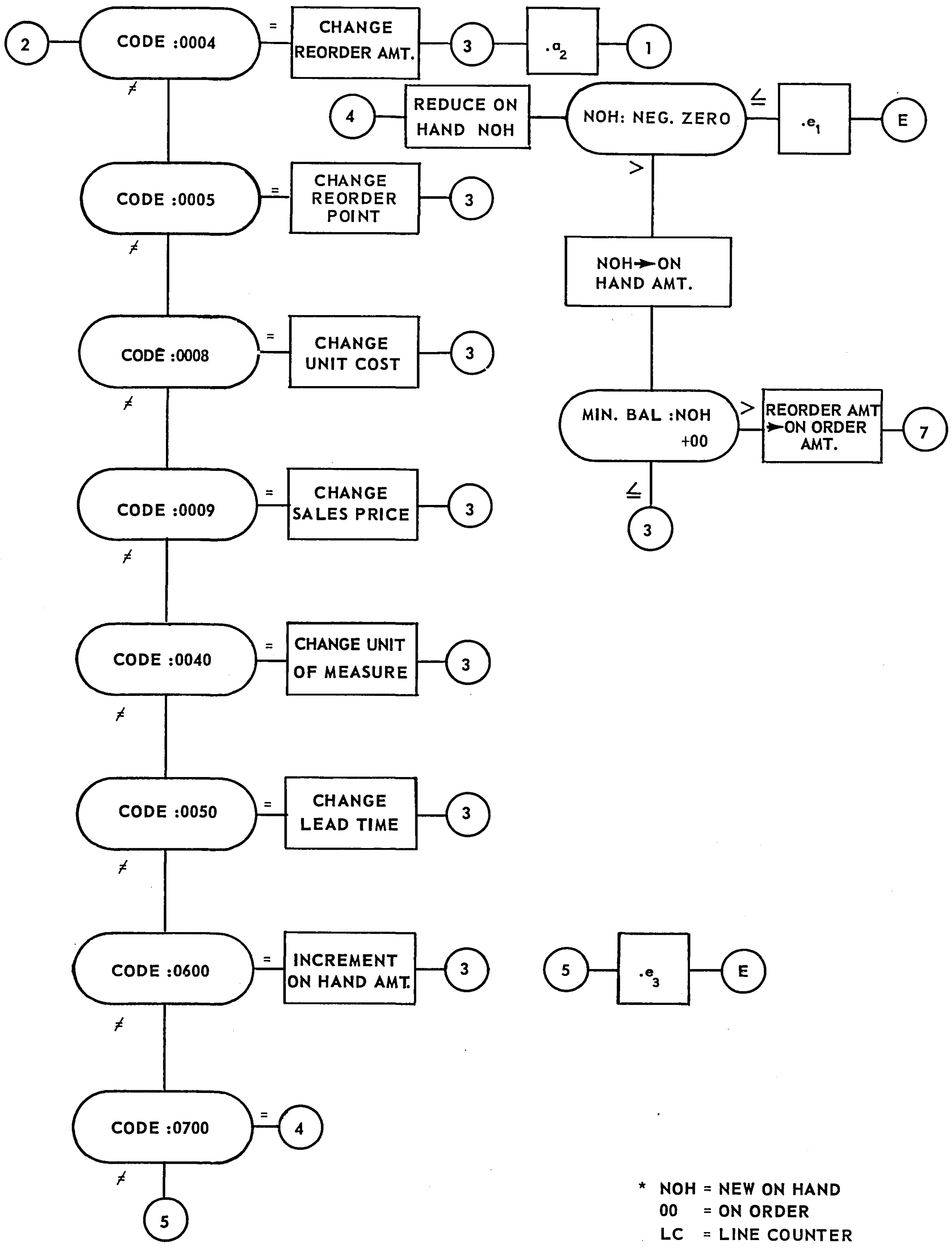
   a. Prints a line.

   b. Advances the paper.

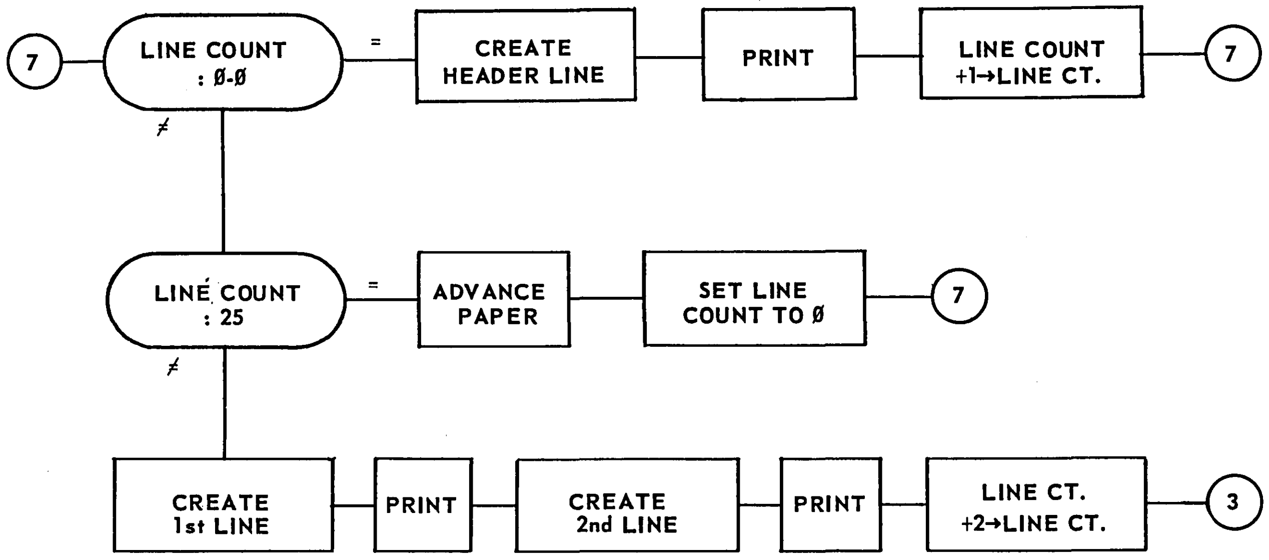
   c. Clears the Print Interlace.

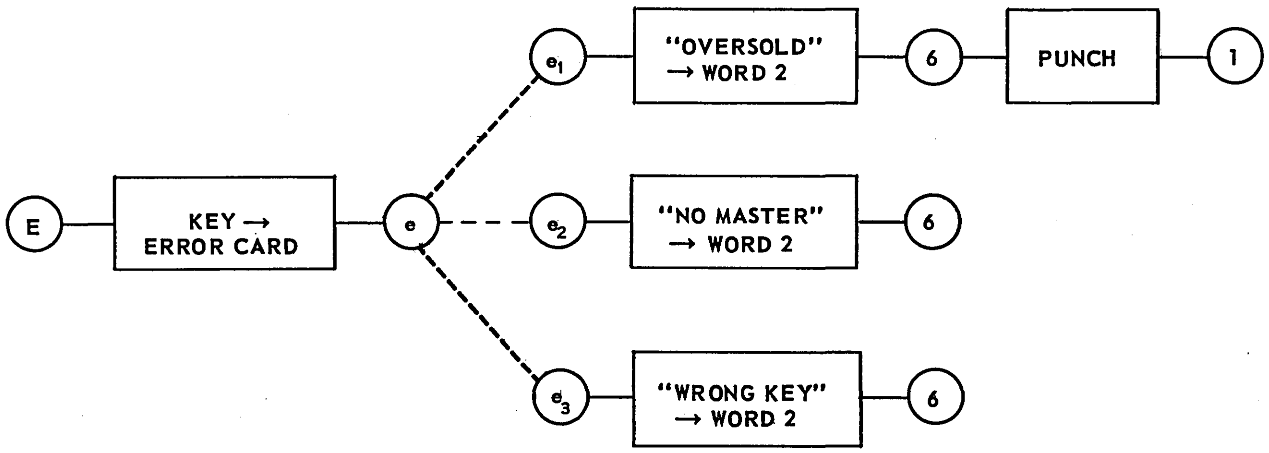


.a1, .b1, .c1
LC =∅
START
INITIALIZE
H.S R. ROUTINE
R.P.U. ROUTINE
H.S.P. ROUTINE
PLACE ZEROES
IN THE
"M" KEY
1
1
GET IN
IMAGE
RPU BUFFER
LOADED?
NO
YES
PUNCH
HSP FREE?
NO
YES
PRINT
WSKEY: SENTINEL
≠
=
8
.b2
.c2
a
WSKEY: MKEY
≠
=
2
IS THIS A
MASTER CARD?
YES
a
NO
.e2
E
c1
1
a1
WS → M
c
c2
STOP
a
a2
PUT IN
NEW DATE
PUNCH
.a1
b
b1
a1
b2
STOP



* NOH	=	NEW ON HAND
00	=	ON ORDER
LC	=	LINE COUNTER





MEMORY ALLOCATIONS

The 4200 band
contains the Print Interlace and a Working Storage Area for an input card image. The Print Interlace is positioned in exact accord with the required locations. The input card image begins in 4211 (unprimed) and 4216 (primed) for word 0. Each subsequent word is placed in memory location positions with addresses 20 words greater than the address of the last word. (Word 1 is in 4231 and 4236, word 2 is in 4251 and 4256, etc.)

The 4400 band
contains the "M" area. Word 0 is in 4421 and 4426. Each subsequent word is found memory location positions with addresses 20 words greater than the address of the last word. Word 9 is found in 4401 and 4406.

CODING

	MEMORY LOCAT'N	OP.	M	C	
	820	00	0633		Initialize the High-Speed Reader Routine.
	728	00	0420		Initialize the Read-Punch Routine.
	652	00	0769		Initialize the High-Speed Printer Routine.
	711	06	0914		
	914	65	4421	0923	Place zeroes in the Key Word of the m area.
	923	65	4426	0717	
(1)	717	GET AN IMAGE			High-Speed Reader Routine obtains next card image.
	533	22	0376	0540	Is the RPU buffer loaded?
	540	27	1000	0371	Is the Printer free?
	1000	25	4211	1013	INPUT EDITING: translate key word into machine code.
	1013	05	4216	1018	
	1018	12		1021	
	1021	30	1023	1025	
	[1023	kk	kkkk	kkkk]	PROCESSING: test Key Word against the sentinel all K's.
	1025	92	1280	1028	
	1028	77		1031	PROCESSING: Key Word put in rL.
	1031	25	4421	1073	
	1073	05	4426	1078	INPUT EDITING: translate Key Word in m to machine code.
	1078	12		1081	
	1081	82	1479	1084	PROCESSING: Key Word in m compared to Key in ws .
	1084	25	4291	1093	
	1093	05	4296	1098	INPUT EDITING: translate ws 19.
	1098	12		1101	
	1101	31	1104		PROCESSING: Is this a Master Card?
	1104	92	3507	1107	
(a_1)	1107	25	4211	1113	
	1113	05	4216	1118	
	1118	60	4421	1123	PROCESSING: image in ws sent to m
	1123	65	4426	1128	
	1128	25	4231	1133	

MEMORY LOCAT'N	OP.	M	C
1133	05	4236	1138
1138	60	4441	1143
1143	65	4446	1148
1148	25	4251	1153
1153	05	4256	1158
1158	60	4461	1163
1163	65	4466	1168
1168	25	4271	1173
1173	05	4276	1178
1178	60	4481	1183
1183	65	4486	1188
1188	25	4291	1193
1193	05	4296	1198
1198	60	4501	1203
1203	65	4506	1208
1208	25	4311	1213
1213	05	4316	1218
1218	60	4521	1223
1223	65	4526	1228
1228	25	4331	1233
1233	05	4336	1238
1238	60	4541	1243
1243	65	4546	1248
1248	25	4351	1253
1253	05	4356	1258
1258	60	4561	1263
1263	65	4566	1268
1268	25	4371	1273
1273	05	4376	1278
1278	60	4581	1283

PROCESSING: (cont.):

Image in ws: sent to m.

	memory LOCAT'N	OP.	M	C	
	1283	65	4586	1288	
	1288	25	4391	1293	
	1293	05	4396	1298	
	1298	60	4401	1303	
	1303	65	4406	717	$\rightarrow C_1 \rightarrow 1$
②	1479	25	4231	1483	
	1483	05	4236	1488	
	1488	12		1491	*INPUT EDITING: translate code and isolate it.*
	1491	35	1493	1495	
	[1493	77	7700	0000]	
	1495	30	1497	1499	*PROCESSING: test the value of the Code Word.*
	[1497	00	0400	0000]	*Code: 004*
	1499	82	1702	1502	
	1502	30	1504	1506	
	[1504	00	0500	0000]	*Code: 0005*
	1506	82	2709	1509	
	1509	30	1511	1513	
	[1511	00	0800	0000]	*Code: 0008*
	1513	82	1916	1516	
	1516	30	1518	1520	
	[1518	00	0900	0000]	*Code: 0009*
	1520	82	2123	1523	
	1523	30	1525	1527	
	[1525	00	4000	0000]	*Code: 0040*
	1527	82	2530	1530	
	1530	30	1532	1534	
	[1532	00	5000	0000]	*Code: 0050*
	1534	82	2337	1537	
	1537	30	1539	1541	*Code: 0600*
	[1539	06	0000	0000]	

	MEMORY LOCAT'N	OP.	M	C	
	1541	82	2944	1544	*Code: 0600 (cont.)*
	1544	30	1546	1548	
	[1546	07	0000	0000]	*Code: 0700*
	1548	82	3151	3551	
change reorder amount	1702	25	4231	1733	*INPUT AND OUTPUT EDITING:*
	1733	05	4251	1753	
	1753	32	0600	1762	
	1762	25	4521	1773	
	1773	35	1775	1777	
	[1775	00	0000	00TT]	
	1777	20	000T	1781	
	1781	77		1784	
	1784	25	4236	1788	
	1788	05	4256	1608	
	1608	32	0600	1617	
	1617	50	4521	1623	
	1623	25	4526	1628	
	1628	35	1630	1632	
	[1630	00	0000	00TT]	
	1632	20	T	1636	
	1636	60	4526	1678	
③	1678	25	1680	1682	
	[1680	82	3507	3398]	*PROCESSING: .a2*
	1682	60	1104	717	→ ①
change reorder point	2709	06	2712		*INPUT AND OUTPUT EDITING:*
	2712	25	4231	2733	
	2733	32	0400	2740	
	2740	35	2742	2744	
	[2742	00	0000	00TT]	
	2744	30	000k	2748	

INPUT AND OUTPUT EDITING:
Align new reorder amount with its correct position; erase old reorder amount and replace it with the new. Do this for unprimed and primed portion of the word.

INPUT AND OUTPUT EDITING:
Align new minimum balance correctly; erase old minimum balance and replace it with the new. Do this for unprimed and primed portion of the word.

MEMORY LOCAT'N	OP.	M	C
2748	25	4521	2773
2773	35	2775	2777
[2775	TT	TTTT	TT00]
2777	20	0004	2781
2781	60	4521	2623
2623	30	000T	2627
2627	25	4251	2653
2653	32	0400	2660
2660	20	0004	2664
2664	05	000K	2668
2668	25	4541	2693
2693	35	2695	2697
[2695	00	0000	TTTT]
2697	20	000T	2701
2701	60	4541	2743
2743	06	2746	
2746	25	4236.	2788
2788	32	0400	2795
2795	35	2797	2799
[2797	00	0000	00TT]
2799	30	000K	2603
2603	25	4526	2628
2628	35	2630	2632
[2630	TT	TTTT	TT00]
2632	20	0004	2636
2636	60	4526	2678
2678	30	000T	2682
2682	25	4256	2708
2708	32	0400	2715
2715	20	0004	2719

Continue editing reorder point.

	MEMORY LOCAT'N	OP.	M	C	
	2719	05	000k	2723	
	2723	25	4546	2749	
	2749	35	2751	2753	
	[2751	00	0000	TTTT]	
	2753	20	000T	2757	
	2757	60	4546	1678	→ ③
change unit cost	1916	25	4231	1933	
	1933	35	1935	1937	
	[1935	00	0000	TTTT]	
	1937	05	4251	1953	
	1953	30	4561	1963	
	1963	32	0800	1974	
	1974	25	0004	1981	
	1981	35	1983	1985	
	[1983	TT	TT00	0000]	
	1985	20	000T	1989	
	1989	60	4561	1813	
	1813	25	4236	1838	
	1838	35	1840	1842	
	[1840	00	0000	TTTT]	
	1842	05	4256	1858	
	1858	30	4566	1868	
	1868	32	0800	1879	
	1879	25	0004	1883	
	1883	35	1885	1887	
	[1885	TT	TT00	0000]	
	1887	20	000T	1891	
	1891	60	4566	1678	→ ③
change selling price	2123	25	4231	2133	
	2133	05	4251	2153	

	MEMORY LOCA T'	OP.	M	C	
	2153	32	0400	2160	
	2160	25	4581	2183	
	2183	35	2185	2187	
	[2185	00	0000	TTTT]	
	2187	20	000T	2191	
	2191	77		2194	
	2194	25	4236	2038	
	2038	05	4256	2058	
	2058	32	0400	2065	
	2065	50	4581	2083	
	2083	25	4586	2088	
	2088	35	2090	2092	
	[2090	00	0000	TTTT]	
	2092	20	000T	2096	
	2096	60	4586	1678	→
change unit of measure	2530	05	4251	2553	
	2553	25	4231	2583	
	2583	35	2585	2587	
	[2585	00	0000	00TT]	
	2587	32	0800	2598	
	2598	65	4501	2403	
	2403	25	4236	2438	
	2438	05	4256	2458	
	2458	35	2460	2462	
	[2460	00	0000	00TT]	
	2462	32	0800	2473	

	MEMORY LOCAT'N	OP.	M	C	
	2473	60	4506	1678	→ ③
change lead time	2337	26	2340		
	2340	05	4251	2353	
	2353	32	0200	2358	
	2358	25	4561	2363	
	2363	35	2365	2367	
	[2365	TT	00TT	TTTT]	
	2367	20	000T	2371	
	2371	77		2374	
	2374	05	4256	2208	
	2208	50	4561	2213	
	2213	26	2216		
	2216	32	0200	2221	
	2221	25	4566	2268	
	2268	35	2270	2272	
	[2270	TT	00TT	TTTT]	
	2272	20	000T	2276	
	2276	60	4566	1678	→ ③
stock receipt	2944	25	4231	2983	
	2983	05	4236	2988	
	2988	12		2991	
	2991	77		2994	
	2994	25	4251	2803	
	2803	05	4256	2808	
	2808	12		2811	
	2811	05	K	2815	

INPUT EDITING: Align charge amount with on-hand amount; translate it, store it in r L.

	MEMORY LOCAT'N	OP.	M	C	
	2815	25	0004	2819	
	2819	32	0600	2828	
	2828	30	000T	2832	
	2832	25	4441	2843	*INPUT EDITING: translate on hand amount; isolate this field.*
	2843	05	4446	2848	
	2848	12		2851	
	2851	35	2853	2855	
	[2853	TT	TTTT	TT00]	
	2855	70	0004	2860	*PROCESSING: add on hand amt. and receipt amount.*
	2860	17		2863	*OUTPUT EDITING: translate updated on hand amount; erase old on hand amount and replace it with the new.*
	2863	35	2865	2867	
	[2865	TT	TTTT	TT00]	
	2867	77		2870	
	2870	25	4441	2883	
	2883	35	2885	2887	
	[2885	00	0000	00TT]	
	2887	20	0004	2891	
	2891	60	4441	2943	
	2943	25	4446	2948	
	2948	35	2950	2952	
	[2950	00	0000	00TT]	
	2952	20	000T	2956	
	2956	60	4446	1678	$\rightarrow$ ③
Stock Sold	3151	25	4541	3193	*INPUT EDITING: align and translate reorder point for future testing; store it in 3042.*
	3193	05	4546	3198	
	3198	12		3001	
	3001	35	3003	3005	
	[3003	TT	TTTT	0000]	
	3005	77		3008	
	3008	25	4521	3023	

MEMORY LOCAT'N	OP.	M	C	
3023	05	4526	3028	
3028	12		3031	
3031	05	0004	3035	
3035	32	0200	3040	
3040	65	3042	3044	
3044	25	4251	3053	*INPUT EDITING: Align and translate change amount; store it in rL.*
3053	05	4256	3055	
3055	12		3058	
3058	77		3061	
3061	25	4231	3083	
3083	05	4236	3088	
3088	12		3091	
3091	35	3093	3095	
[3093	00	00TT	TTTT]	
3095	05	0004	3099	
3099	32	0600	3108	
3108	30	000T	3112	
3112	25	4441	3143	*INPUT EDITING: Translate and isolate on hand amount.*
3143	05	4446	3148	
3148	12		3152	
3152	35	3154	3156	
[3154	TT	TTTT	TT00]	
3156	75	0004	3161	*PROCESSING: On hand - sales amount = New On Hand (in r A)*
3161	30	3163	3165	
[3163	$\bar{0}0$	0000	0000]	*New On Hand: - 0000000000*
3165	82	3168	3568	
3168	60	3170	3172	*New On Hand$\longrightarrow$T.S. 3170*
3172	17		3175	*OUTPUT EDITING: Translate new on hand amount; erase old on hand amount and replace it with the new.*
3175	35	3177	3180	
[3177	TT	TTTT	TT00]	

MEMORY LOCAT'N	OP.	M	C
3180	77		3184
3184	25	4441	3194
3194	35	3196	3199
[3196	00	0000	00TT]
3199	20	0004	3004
3004	60	4441	3043
3043	25	4446	3048
3048	35	3050	3052
[3050	00	0000	00TT]
3052	20	000T	3056
3056	60	4446	3098
3098	25	4401	3103
3103	05	4406	3109
3109	12		3113
3113	37	0100	3117
3117	77		3120
3120	25	4581	3133
3133	05	4586	3138
3138	12		3141
3141	37	0600	3150
3150	20	0004	3155
3155	70	3170	3176
3176	77		3179
3179	25	3042	3045
3045	42	4288	3051
3051	87	3054	1678
3054	06	3057	
3057	25	3060	3062
3060	TT	TTTT	TT00
3062	35	4521	3073

INPUT EDITING: Align and translate On Order amount.

PROCESSING: New On Hand + On Order$\longrightarrow$rA

H.S.R. buffer test

Test reorder point against New On Hand & On Order

OUTPUT EDITING: Erase old On Order amount; replace it with reorder amount.

	MEMORY LOCAT'N	OP.	M	C	
	3073	32	0600	3082	
	3082	77		3085	
	3085	25	4581	3134	
	3134	35	3136	3139	
	[3136	TT	TTTT	0000]	
	3139	20	0004	3144	
	3144	77		3147	
	3147	26	3153		
	3153	32	0500	3162	
	3162	50	4581	3183	
	3183	65	4401	3006	
	3006	06	3009		
	3009	25	3011	3013	
	[3011	TT	TTTT	TT00]	
	3013	35	4526	3029	
	3029	32	0600	3038	
	3038	77		3041	
	3041	25	4586	3089	
	3089	35	3092	3094	
	[3092	TT	TTTT	0000]	
	3094	20	0004	3104	
	3104	77		3107	
	3107	26	3110		
	3110	32	0500	3118	
	3118	50	4586	3140	
	3140	65	4406	3578	$\longrightarrow$ (7)
(a_2)	3398	06	3201		*OUTPUT EDITING: Change the date.*
	3201	25	3203	3205	
	[3203	00	00DD	MMYY]	} *Date supplied by programmer.*

MEMORY LOCAT'N	OP.	M	C
3205	32	0200	3210
3210	65	3212	3214
3214	17		3217
3217	35	3219	3222
[3219	00	0000	TTTT]
3222	77		3225
3225	25	4541	3243
3243	35	3245	3247
[3245	TT	TTTT	0000]
3247	20	0004	3251
3251	60	4541	3293
3293	25	4546	3298
3298	35	3300	3302
[3300	TT	TTTT	0000]
3302	20	000T	3306
3306	60	4546	3348
3348	25	3212	3215
3215	17		3218
3218	35	3220	3223
[3220	TT	0000	0000]
3223	77		3226
3221	25	4561	3263
3263	35	3265	3267
[3265	00	TTTT	TTTT]
3267	20	0004	3271
3271	60	4561	3313
3313	25	4566	3318
3318	35	3320	3322
[3320	00	TTTT	TTTT]
3322	20	000T	3326

	MEMORY LOCAT'N	OP.	M	C	
	3326	60	4566	3368	
	3368	PUNCH	CARD	A	Read-Punch routine punches and check. reads a card.
	3372	25	3374	3376	
	[3374	82	3507	1107]	$.a_1$
	3376	60	1104	1107	$(.b_1) \rightarrow (a_1)$
(8)	1280	25	1282	1284	
	[1282	65	4406	1320]	$.c_2$
	1284	60	1303	1305	
	1305	25	1307	1309	
	[1307	60	1104	1320]	$.b_2$
	1309	60	3376	1378	
	1378	25	1104	1306	Place 'a' exit (a, or a_2) in register a as the m portion of a skip order. Execute this skip.
	1306	37	0400	1313	
	1313	35	1315	000k	$\rightarrow (a)$
	[1315	00	TTTT	0000]	
	1320	67	TTTT	TTTT	Final Stop.
(E) Error Card	3400	47	0100	0004	$S.S_1$ for the H.S.R. Next instruction in r l (entrance to "e")
(e_1)	0004	25	3409	3411	
	[3409	Δ−	−27Δ	1r00]	Oversold $\rightarrow$ word $_2$
	3411	05	3413	3429	
	[3413	02	0110	2000]	
(6)	3429	60	4231	3433	$Word_2 \rightarrow$ output
	3433	65	4236	3438	
	3438	06	3441		
	3441	65	4251	3453	Erase $Word_3$
	3453	65	4256	3458	
	3458	PUNCH	CARD	A	Read punch routine punches and check reads.
	3462	57		3466	
	3466	47	0000	0717	$\rightarrow (1)$

	MEMORY LOCAT'N	OP.	M	C	
e_2	0001	25	3408	3410	
	[3408	6Δ	0677	4-20]	No Master→Word$_2$
	3410	05	3412	3429	
	[3412	20	0021	3010]	→ 6
e_3	0001	25	3414	3416	
	[3414	-2	Δ650	9-Δ0]	Wrong Key→Word$_2$
	3416	05	3418	3429	
	[3418	11	0210	2020]	→ 6
	3507	30	3509	3400	. e_2
	[3509	25	3408	3410]	→ E
	3551	30	3553	3400	. e_3 Setting entrances into e . The entrance is placed in r l and at this switching point in the routine, what rL determines the direction of the switch.
	[3553	25	3414	3416]	→ E
	3568	30	3570	3400	. e_1
	[3570	25	3409	3411]	→ E
7	3758	42	4288	3762	H.S.R. buffer test.
	3762	25	3764	3766	
	[3764	00	0000	0000]	Line count–originally set to zeros.
	3766	31	3769		Is the line count equal to zeros?
	3769	82	3992	3772	
	3772	30	3774	3776	
	[3774	00	0000	0025]	Is the line count equal to 25?
	3776	82	3979	3779	
Create 1st line	3779	25	4481	3783	PRINT EDITING
	3783	05	4486	3788	
	3788	60	4334	3836	Third part of description sent to Print Interlace 5 and 5'.
	3836	65	4339	3841	
	3841	25	4461	3863	
	3863	05	4466	3868	Second part of description sent to Print Interlace 4 and 4'.
	3868	60	4250	3902	
	3902	65	4255	3907	

MEMORY LOCAT'N	OP.	M	C
3907	25	4211	3913
3913	05	4216	3918
3918	30	3920	3922
[3920	TT	TTTT	TT00]
3922	62		3926
3926	35	000Y	3930
3930	32	0K00	3943
3943	35	000Y	3947
3947	65	4281	3983
3983	60	4286	3988
3988	25	3990	3993
[3990	6Δ	¢000	0000]
3993	05	3995	3997
[3995	20	2000	0000]
3997	60	4200	3602
3602	65	4205	3607
3607	25	4521	3623
3623	05	4526	3628
3628	62		3632
3632	30	3634	3636
[3634	TT	TTTT	TT00]
3636	35	000Y	3640
3640	32	0K00	3653
3653	35	000Y	3657
3657	65	4303	3705
3705	60	4308	3710
3710	42	4288	3714

Stock number field isolated and zero suppressed; sent to Print Interlace 2 and 2'.

No. sent to Print Interlace 1 and 1'.

Reorder amount isolated and zero suppressed, sent to Print Interlace 7 and 7'.

H.S.R. buffer test.

MEMORY LOCAT'N	OP.	M	C
3714	30	3716	3718
[3716	00	0000	00TT]
3718	25	4441	3743
3743	05	4446	3748
3748	35	000Y	3752
3752	32	0K00	3765
3765	35	000Y	3771
3771	65	4365	3818
3818	60	4370	3773
3773	25	4501	3803
3803	05	4506	3810
3810	60	4241	3843
3843	65	4246	3848
3848	PRINT A LINE		
3852	42	4288	3856
3856	25	4211	3864
3864	05	4216	3869
3869	30	3871	3873
[3871	00	0000	00TT]
3873	35	000Y	3877
3877	32	0K00	3890
3890	35	000Y	3894
3894	32	0K00	3908
3908	62		3912
3912	60	4281	3933
3933	65	4286	3938

First part of description sent to Print Interlace 3 and 3'.

Unit of measure sent to Print Interlace 8 and 8'.

High Speed Printer routine prints, advances paper, clears interlace.

create 2nd line

H.S.R. buffer test.

Plant number isolated and sent to Print Interlace 2 and 2'.

MEMORY LOCAT'N	OP.	M	C
3938	25	3940	3942
[3940	Δ1	7640	0000]
3942	05	3944	3946
[3944	12	2230	0000]
3946	60	4200	3952
3952	65	4205	3957
3957	25	4561	3963
3963	05	4566	3968
3968	35	3970	3972
[3970	00	TT00	0000]
3972	32	0K00	3984
3984	35	3986	3989
[3986	00	TT00	0000]
3989	62		3991
3991	77		3994
3994	25	3203	3605
3605	65	4303	3610
3610	37	0400	3617
3617	17		3620
3620	62		3624
3624	35	3626	3629
[3626	TT	TTTT	0000]
3629	32	0K00	3642
3642	35	3644	3646
[3644	TT	TTTT	0000]
3646	50	4308	3662

Plant sent to Print Interlace 1 and 1'.

Lead time isolated and zero suppressed; sent to Print Interlace 7 and 7'.

Date translated, zero suppressed and sent to Print Interlace 5 and 5'.

	MEMORY LOCAT'N	OP.	M	C	
	3662	65	4334	3686	*Date (cont.)*
	3686	60	4339	3691	
	3691	42	4288	3695	*H.S.R. buffer test.*
	3695	PRINT	A LINE		
	3720	25	3764	3770	
	3770	70	3780	3790	*Line count* +2→*line count.*
	[3780	00	0000	0002]	
	3790	60	3764	1678	→ ③
line count =25	3979	ADVANCE	PAPER		*High-Speed Printer Routine advances paper to form a margin.*
	3600	06	3660		*ϕ→line counter.*
	3660	65	3764	3758	→ ⑦
create header	3992	25	3203	3606	
	3606	37	0400	3613	
	3613	17		3616	
	3616	62		3621	
	3621	30	3625	3627	
	[3625	TT	TTTT	0000]	*Date translated, zero suppressed and sent to Print Interlace 1 and 1'.*
	3627	35	0004	3631	
	3631	32	0k00	3643	
	3643	35	0004	3647	
	3647	65	4200	3652	
	3652	60	4205	3658	
	3658	25	3661	3663	
	3661	00	0000	0000	*Page number: initially ϕ.*
	3663	70	3665	3668	
	[3665	00	0100	0000]	
	3668	77		3671	*Page number incremented, translated, zero suppressed and sent to Print Interlace 8 and 8'.*
	3671	17		3674	
	3674	62		3678	
	3678	35	3680	3682	

MEMORY LOCAT'N	OP.	M	C	
[3680	00	TT00	0000]	
3682	32	0K00	3693	
3693	35	3696	3698	*Page number (cont.)*
[3696	00	TT00	0000]	
3698	65	4241	3744	
3744	60	4246	3749	
3749	25	3751	3753	
[3751	r-	2029	1-00]	
3753	05	3755	3757	*Second Part of Title sent to Print Interlace 5 and 5'*
[3755	00	1030	2000]	
3757	60	4334	3786	
3786	65	4339	3791	
3791	25	3793	3795	
[3797	Δ7	5-00	0000]	
3795	05	3797	3799	*Page sent to Print Interlace 7 and 7'*
[3795	12	1000	0000]	
3799	60	4303	3805	
3805	65	4308	3811	
3811	25	3813	3815	
[3813	74	Δ190	2-Δ2]	
3815	05	3817	3819	
[3817	13	0120	1001]	*First Part of title sent to Print Interlace 4 and 4'*
3819	60	4250	3853	
3853	65	4255	3857	
3857	50	3661	3870	
3870	PRINT A LINE			*H.S.P. routine prints, clears interlace, advances paper.*
3880	25	3764	3781	
3781	70	3784	3792	*Line count + 1 → line count*
[3784	00	0000	0001]	
3792	60	3764	3758	→ ⑦

UNIVAC®—The FIRST Name in Electronic Computing Systems