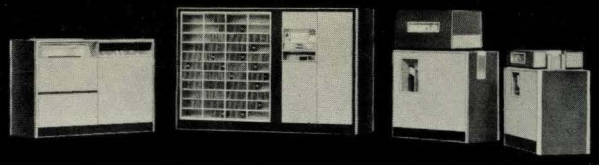


UNIVAC®
Solid-State 90

PROGRAMMING

Instruction Codes and Programming Examples

PROGRAMMING

Instruction Codes and Programming Examples

Contents

1. INSTRUCTION CODES AND EXAMPLES 1
- Instruction Format 1
- Instruction Cycle 2
- Address Modification with Index Registers 3
- Instruction Conventions 4
- Transfer Instructions 4
- Addressing Registers 5
- Arithmetic Instructions 5
- C + 1 Conditions 6
- Logical Instructions 6
- Translate Instructions 8
- Input-Output Instructions (Read-Punch Unit) 9
- Input Instructions (High-Speed Reader) 9
- Printing Instructions (High-Speed Printer) 10

2. OVERCAPACITY PUNCHING 11
- General 11
- Editing an Input Overcapacity Punch 11
- Editing an Output Overcapacity Punch 12
- Suggested Coding for Overcapacity Input-Editing Routine 14
- Suggested Coding for Overcapacity Output-Editing Routine 15

APPENDIX 16
- Key to Card Interlace Table 16
- Interlace Table — Read-Punch Unit 17
- Interlace Table — High-Speed Reader 17
- Key to Print Interlace Table 18
- Interlace Table—High-Speed Printer 19
- Translation Table 20

1. Instruction Codes and Examples

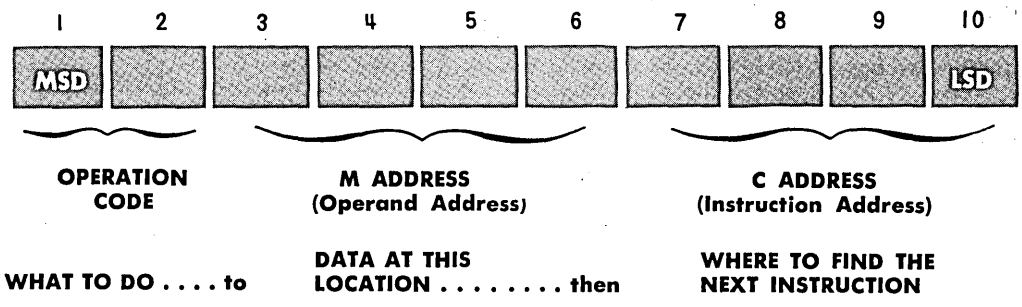
INSTRUCTION FORMAT

The UNIVAC Solid-State 90 Computing System employs a 1½-address instruction code system, with one instruction per computer word. The digit positions in a 10-digit computer word are numbered, left to right, 1 through 10. Digit position 1 is considered the most significant digit (MSD) and digit position 10 is considered the least significant digit (LSD). 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 Processor what to do with this word, and the *c* address is the storage location of the next instruction word. The *m* and *c* address portions have different 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-phase cycle (with an additional phase if index registers are used) 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:

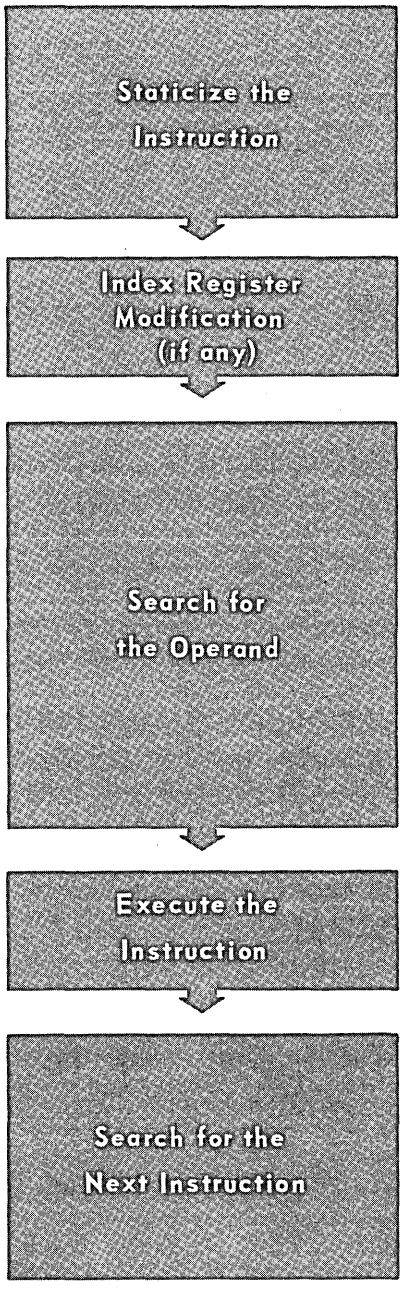
(1) Staticize the Instruction: The instruction located by the previous search (5) is transferred from the drum location to the static register (operation code only) and register *C* (the entire word). This step requires one word time, which is 17 microseconds.

(2) Index Register Modification: When modification is indicated, the *m* address of the instruction is altered by the index register specified. This step requires one word time.

(3) 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 a drum location, the address of the next available storage location on the drum is compared with the first address part of the contents of register *C* 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.

(4) Execute the Instruction: The operation indicated in the instruction is performed. The time required depends upon the type of operation performed.

(5) 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 register *C* until a match is obtained. This step requires a minimum of one word time and a maximum of 200 word times.



ADDRESS MODIFICATION WITH INDEX REGISTERS

If an instruction is to be modified by the use of index registers, it must contain an indication of the index register to be used. The indication is a combination of the sign bit and the second, or least significant, digit of the operation code. Only the 4 bit of the 5 4 2 1 bits of this digit is used. Since this bit is not utilized in the least significant digit of any operation code, the normal execution of an instruction is not otherwise affected. The index register must be loaded initially with the desired increment.

The four combinations of the two bit positions are interpreted as shown in the table below:

Band Modification Feature

A band modification feature enables modification with index registers to be restricted to the same band in which it was initiated. If, during modification, the *m* address is altered to refer to another band, the Processor will cause a return to the corresponding location in the original band. This band modification occurs when at least two of the following conditions are present:

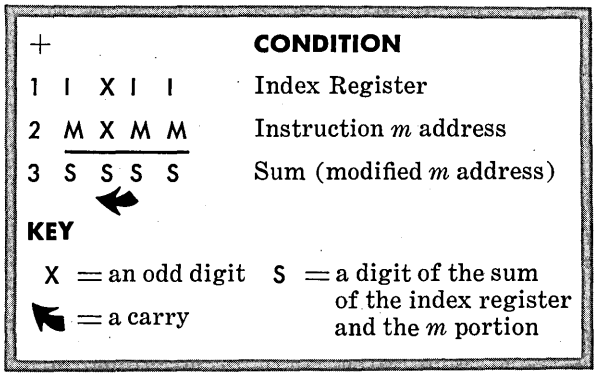
1. The hundreds digit of the contents of the index register is odd.
2. The hundreds digit of the instruction's *m* address is odd.
3. During modification, a carry occurs from the tens digit to the hundreds digit.

MODIFICATION TABLE

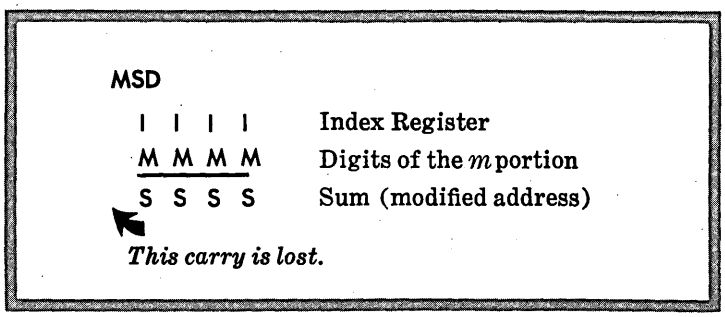
SIGN BIT	0 5421	C 5421	CONDITION	TYPE OF MODIFICATION
0	XXXX	X0XX	0	No Modification
0	XXXX	X1XX	1	Modify with Index Register 1 (rB_1)
1	XXXX	X0XX	2	Modify with Index Register 2 (rB_2)
1	XXXX	X1XX	3	Modify with Index Register 3 (rB_3)

During the staticizing phase, all instructions are examined for the presence of one of the above conditions. If condition 0 is found to be present, the modification phase is bypassed, and the instruction is executed in the normal manner with a three- or four-phase cycle. If condition 1, 2, or 3 exists, however, the second phase of the instruction cycle is the addition of the specified index register's contents to the *m* address of the contents of register *C*. The Processor then searches for the new *m* address during the third phase, and the execution of instructions proceeds in the normal manner.

The figure below represents the conditions which, in combination, cause band modification to be effected.



Note: When the four digits of the m *address are added to those of the index register, any carry from the most significant digit position is lost.*



INSTRUCTION CONVENTIONS

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

- *m* represents the address of a storage location or register which usually contains the operand.
- *c* represents the address of a storage location or register which usually contains the next instruction.
- ( ) represents the contents of a storage location or register; for example (4309) = contents of storage location 4309.
- *rA* represents register *A*.
- *rL* represents register *L*.
- *rX* represents register *X*.
- *rC* represents register *C*.

A dash substituted for the *m* or *c* portion of an instruction means the Processor ignores that portion when the instruction is executed.

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.

TRANSFER INSTRUCTIONS		WORD TIME
25 m c	Transfer (*m*) to *rA*.	4
05 m c	Transfer (*m*) to *rX*.	4
30 m c	Transfer (*m*) to *rL*.	4
*60 m c	Transfer (*rA*) to *m*.	4
*65 m c	Transfer (*rX*) to *m*.	4
*50 m c	Transfer (*rL*) to *m*.	4
77 – c	Transfer (*rA*) to *rL*.	3
26 m –	Clear *rA* to zero and set its sign storage to plus. The next instruction is at *m*.	3
06 m –	Clear *rX* to zero and set its sign storage to plus. The next instruction is at *m*.	3
61 m –	Clear *rL* to zero and set its sign storage to plus. The next instruction is at *m*.	3
36 m –	Clear *rA* to zero and leave its sign unchanged. The next instruction is at *m*.	3
86 m –	Clear *rA* and *rX* to zero. Set the sign of *rA* and *rX* to that of *rL*. The next instruction is at *m*.	14
23 m –	Transfer (*rC*) to *rA*. The next instruction is at *m*.	3

**Registers A, X, or L cannot be used as the* m *address.*

Examples:

1. Place the contents of 4302 into registers A and L. The next instruction is at 0357. Begin in line 0350.

Solution:

0350	25	4302	0354	(4302) ⟶ rA
0354	77	0000	0357	(rA) ⟶ rL

2. Clear storage location 3172 to all zeros. Leave registers A and X undisturbed. The next instruction is at 0174. Begin in line 0167.

Solution:

0167	31	0170	0000	0's ⟶ rL
0170	50	3172	0174	(rL) ⟶ 3172

ADDRESSING REGISTERS

As a result of their addressability, registers A, X, or L may be the m address of many, and the c address of all, instructions. The sole restriction is that they cannot be used in the m portion of the 50, 60, and 65 orders, nor in the m portion of instructions which do not address a one-word storage location. The arithmetic registers are addressed by nonnumeric digits in the least significant digit of c or m. The addresses of the three registers are as follows:

	LSD of m or c	
	BITS	CHARACTER
rA =	0101	4/1
rX =	0111	4/3
rL =	0110	4/2

Examples:

1. Starting in storage location 1301, place the contents of 4503 in registers A, X, and L.

Solution:

1301	25	4503	1305	(4503) ⟶ rA
1305	05	0004/1	1309	(rA) ⟶ rX
1309	30	0004/3	1313	(rX) ⟶ rL

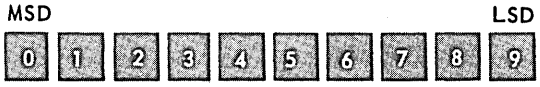
2. Place the contents of rL in rA. The next instruction is in rX. Start at 0021.

Solution:

0021	25	0004/2	0004/3	(rL) ⟶ rA, next instruction is in rX

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 the left of the most significant digit of the multiplier, m. This sentinel stops the multiplication after the last significant multiplier digit is used.	5 plus the number of digits in the multiplier plus the sum of these digits (maximum = 105)
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. If the absolute value of the contents of register L are less than or equal to the contents of the storage location specified, overflow occurs. A sentinel may be placed in rX to stop the division after the desired number of digits has been developed in the quotient. For the placement of the sentinel, the digit positions are numbered, *from the MSD to the LSD, 0 through 9, instead of in the usual way.*	5 + 2 times the number of digits in the quotient + the sum of the digits in the odd positions in the quotient + the sum of the nines complement of the digits in the even positions in the quotient (maximum = 115)



Digit positions for division sentinel only

The sentinel (0101) is placed as follows:

PURPOSE	POSITION
To develop 2 digits in the quotient	2
To develop 4 digits in the quotient	4
To develop 6 digits in the quotient	6
To develop 8 digits in the quotient	8

All other positions must contain zeros.

*Code 0101 or 1101.

When 10 digits are to be developed, the contents of rX need not be changed, provided that there is no possibility that any character in register X has a bit in both the 4 and the 1 bit positions. If there is a possibility that the combination is present, rX should be cleared to all zeros.

Note: Only an even number of digits should be developed in the quotient.

Examples:

1. Add the contents of 1411 and 4115. Place the sum in 4420. Begin in 0009.

Solution:

0009	25	1411	0013	(1411)→rA
0013	70	4115	0018	(4115) + (rA)→rA
0018	60	4420	0022	(rA)→4420

2. Reduce the contents of storage locations 4491 and 4691 by 1, which is a constant stored in 2087. Place the results in 4600 and 4700. Begin in 2085.

Solution:

2085	30	2087	2089	0000000001→rL
2087	00	0000	0001	CONSTANT
2089	25	4491	2093	(4491)→rA
2093	75	0004/2	2098	(rA) − (rL)→rA
2098	60	4600	2102	(rA)→4600
2102	25	4691	2143	(4691)→rA
2143	75	0004/2	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 4705, and the 10 least significant digits in 4709. Begin in 0189.

Solution:

0189	30	4391	0193	(4391)→rL
0193	85	4100	0203	(rL) x (4100)→rA; rX
0203	60	4705	0207	(rA)→4705
0207	65	4709		(rX)→4709

C + 1 CONDITIONS

Two types of $c+1$ conditions can occur in the Processor. The first is an arithmetic overflow, indicating that the result of an arithmetic operation exceeds the capacity of the register. The second is an abnormal condition in an input or output unit (for example, Printer out of paper, a card jam in the High-Speed Reader or Read-Punch Unit). In both cases, the system continues to operate. However, when such a condition does occur, the next instruction is found not 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 (for example, 0199, 0399, and so forth), the $c+1$ address is the first word of the same band (for example, 0000, 0200, and so forth). When the address of the next instruction is a register address, proceed to that register whether or not overflow occurs. In this instance, overflow causes a delay of one word time.

Example:

Add (3178) to (3182). Place the sum in 3187. If overflow occurs, place 0000000001 in 3192 and (rA) in 3187. In both cases, the next instruction (following storage of the sum) is in 0189. Begin in 0176.

Solution:

	0176	25	3178	0180	(3178)→rA
	0180	70	3182	0194	(3182) + (rA)→rA
	0185	60	3187	0189	(rA)→3187
C+1	0186	05	0188	0190	0000000001→rX
	0188	00	0000	0001	CONSTANT
	0190	65	3192	0185	(rX)→3192

LOGICAL INSTRUCTIONS

WORD TIME

Buff

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 4 (0100) on 1 (0001) produces the non numeric (0101), and 6 (1001) on 2 (0010) produces 8 (1011).

Erase — WORD TIME

35 m c Erase the contents of the bit positions of register A corresponding to the bit positions in the specified storage location that contains zero bits. The sign of rA is undisturbed. — 4

Examples:

1. Erase (rA)	=1	1	1	1	1	2	2	2	2	2
on the basis of (m)	=0	8/4	8/4	8/4	8/4	8/4	8/4	8/4	8/4	8/4
Result in rA	=0	1	1	1	1	2	2	2	2	2
2. Erase (rA)		=12	3	4	5	678	9	6		
on the basis of (m)		=00	8/4	8/4	8/4	000	8/4	8/4		
Result in rA		=00	3	4	5	000	9	6		

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; an 8/4 (1111) will retain it.

Right Circular Shift

32 0N00 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. The 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

Left Shift

37 0N00 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. The 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 disturbed. — 3+N

Skip

00 m – Skip to the next instruction at address m. — 2

Magnitude Comparison

87 m c If (rA) are algebraically greater than (rL), the next instruction is in m; if not, the next instruction is in c. — 3

Equality Comparison

82 m c If (rA) equal (rL), then the next instruction is in m; if not, the next instruction is in c. — 3

Examples:

1. 4392 contains the value **A** and 4396 contains the value **B**. If **A** is the same as or less than **B**, place it in **rX**. Begin in 0290.

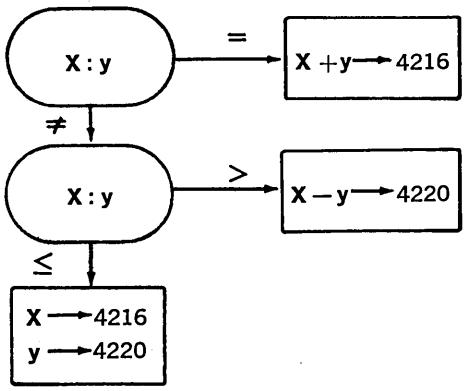
Solution:

	0290	25	4392	0294	A → rA
	0294	30	4396	0298	B → rL
	0298	87	0301	0302	test A against B
(A > B)	0301	05	0004/2		B → rX
(A ≤ B)	0302	05	0004/1		A → rX

2. 0900 contains **X**
   0904 contains **y**

If **X**=**y**, add the two numbers and store the sum in 4216. If **X** is greater than (>) **y**,subtract **y** from **X** and store the difference in 4220. If **X** is equal to or less than (≤) **y**, merely store **X** in 4216 and **y** in 4220. Begin in 0098.

Solution:



Solution:

	0098	25	0900	0102	(0900) → rA
	0102	30	0904	0106	(0904) → rL
	0106	82	0109	0309	rA : rL for equality
	0309	87	0312	0112	rA : rL for magnitude
X > y	0112	60	4216	0118	X → 4216
	0118	50	4220		y → 4220
X = y	0109	70	0004/2	0114	X + y → sum
	0114	60	4216		sum → 4216
X ≤ y	0312	75	0004/2	0317	X – y → rA
	0317	60	4220		difference → 4220

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

Zero Suppression

		WORD TIME
62 – c	This instruction is used to suppress printing (or punching) of nonsignificant zeros or commas to the left of the most significant digit of a field. Zero suppression is performed on the words in the print (or punch) interlace* before the print (or punch) instruction is given. The primed word of the pair must be in *rX* and the unprimed word in *rA*.	4

Stop

67 m c	Stop the Processor. The Processor stops with the stop instruction in *rC*, but before the search for the next instruction is started. Normally, when the Processor 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.

Load Index Register

02 m c	Transfer the four digits composing *m* into the specified index register. The same combination (see *Note*) which indicates normal index register modification specifies the index register to be loaded.	3

Increment and Unload Index Register

07 m c	Using the address modification method, add the four digits composing *m* to the contents of the specified index register. The sum is placed in the index register specified and also in the *m* portion of *rA*. The remainder of *rA* is cleared to zeros, and the sign of *rA* is set to plus.	4

Note: These two instructions will serve all three index registers. The same combination which indicates normal modification with index registers is used to specify the index register to be used. This combination of the sign bit and the 4 bit of the operation code's least significant digit has been explained under "Address Modification with Index Registers."

TRANSLATE INSTRUCTIONS

Data enters the Central Processor from punched cards in Remington Rand card code. This data may be moved from place to place within the Processor in this code by instructions. For the arithmetic operations 70 *m c*, 75 *m c*, 85 *m c* and 55 *m c*, and the logical operation 87 *m c*, the data must be translated into computer code* to give correct results. However, before this data can be punched in output cards, or printed, it must be translated back to card code.

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

		WORD TIME
12 – c	Translate from Remington Rand card code to computer code,* deposit the result in *rA*, and clear *rX* to zeros. The sign of *rA* and *rX* remains unchanged. Before this command is given, *rA* must contain the unprimed word of the card image and *rX* the primed word.	3
17 – c	Translate from computer code to Remington Rand card code* and deposit the two resulting words in *rA* and *rX*. The signs of both results are positive. *rA* will contain the unprimed word and *rX* the primed word. All	3

**The interlace is the pattern in which data is placed in a storage band. See the interlace patterns in the Appendix.*

**See Translation Table in the Appendix.*

WORD TIME

computer-code zeros are translated to card code punching zeros. Before this command is given, *rA* must contain the computer-coded word.

INPUT-OUTPUT INSTRUCTIONS (READ-PUNCH UNIT)

Four instructions direct the operation of the Read-Punch Unit:

Card Cycle—Buffer Load

81 XX00 c Initiate card movement in the Read-Punch Unit and a storage-to-buffer transfer. The two most significant digits of the *m* address (*XX*) designate the storage band. When the transfer is completed, the unit will punch the data from the punch buffer area into the card in the punch station and read 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 station, and the end card goes into its preselected stacker. For minimum latency, this instruction should be placed on level 0198. **203**

Note: If an abnormal condition exists, the next instruction is found in *c* + 1. The abnormal conditions in the Read-Punch Unit may be: no card in read station 1 or 2, empty input magazine, full stacker, full chip box, and card jam.

Buffer Unload

46 XX00 c Transfer the contents of the Read-Punch input buffer into the storage band designated by the two most significant digits of *m* (*XX*). The data is not translated. In storage, it is arranged according to an interlace pattern. **203**

Buffer Test

22 m c Test the Read-Punch input buffer. If the buffer is loaded, (*rC*) are transferred to *rA* and the next instruction is found at *m*. If the buffer is not loaded, the next instruction is found at *c*. The contents of *rA* are not altered in this case. **3 if c address taken; otherwise 4**

Stacker Select

57 – c Select output stacker 1 (sort). **3**

INPUT INSTRUCTIONS (HIGH-SPEED READER)

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

Card Cycle – Buffer Load

72 m c Initiate card movement 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 Central Processor is free to operate on other instructions during the moving and reading of the cards. If an interlock* occurs, 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, the next instruction is found in *c* + 1. Abnormal conditions in the High-Speed Reader may be: output bin full, card jam, input magazine empty, and registration check failure.

**See Read-Punch Unit interlace pattern in the Appendix.*

**Interlock is the postponement of an instruction until the previous instruction has completed its operation.*

Buffer Unload

		WORD TIME
96 XX00 c	Transfer the contents of the High-Speed Reader buffer into the storage band according to its predetermined interlace pattern.* The data is transferred untranslated. Only the two most significant digits of m (XX) designate the storage band.	203

Buffer Test

42 m c	Test the High-Speed Reader buffer. If the buffer is loaded, (rC) are transferred to rA and the next instruction is found at m. If the buffer is not loaded, the next instruction is found at c. Register A is not altered in this case.	3 if c address taken; otherwise 4

Stacker Select

47 0N00 c	Select the output stacker on the High-Speed Reader (N = stacker 0, 1, or 2).	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 by a card will read all binary 1's.

PRINTING INSTRUCTIONS (HIGH-SPEED PRINTER)

The following instructions govern the printing operation:

Advance and Print

		WORD TIME
11 XXNN c	Advance the paper NN lines (NN = 00 through 79), and print one line. While the paper advance is taking place, data is transferred from the storage band indicated by the two most significant digits of m (XX) to the print buffer. Registers A and X are used for the transfer, and their previous contents are therefore destroyed. Before this instruction is given, the data to be printed must be arranged in the storage band (XX) according to the print interlace pattern.* It is possible to advance the paper as many as 79 lines in the following manner:	592

4/1N where N = 0 to 9 moves paper 50-59 lines
4/2N where N = 0 to 9 moves paper 60-69 lines
4/3N where N = 0 to 9 moves paper 70-79 lines

For minimum latency, the 11 instruction should be placed on level 0197.

Advance Paper

16 00NN c	When the previous Printer operation is completed, advance the paper NN lines. Once paper movement is started, the Processor is free for other operations. The advance is the same as defined in the 11 instruction above.	4

Note: On either the 11 m c or 16 m c instruction, if an abnormal condition exists in the High-Speed Printer, the next instruction is found in c + 1. The abnormal conditions in the High-Speed Printer may be: out of paper, out of ribbon, print-code error, one-line print, carriage out, paper jam, and buffer parity error.

Printer Test

27 m c	Test 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

**See High-Speed Reader interlace pattern in the Appendix.*

2. Overcapacity Punching

GENERAL

Occasionally, extra digits, beyond the normal ninety, are needed on a card.When such need 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 ninety columns. If the range of this code is from zero to one, only one punch is required. The presence of a punch would indicate one. The absence of a punch would indicate zero. If the range of the code is from zero to nine, more punches are needed. The one-digit code could be punched in a biquinary form covering four punch positions (nine punched in four adjacent zero columns would be 1100, eight 1011, seven 1010, and so on). The code might be punched in six zero positions, each one corresponding to an actual row on the card (that is, the first position represents the zero row, second = 1, 2 row, third = 3, 4 row, fourth = 5, 6 row, fifth = 7, 8 row, and sixth = the 9 row). This system facilitates punching as wide a range of values as 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 nonpunching zero.

EDITING AN INPUT OVERCAPACITY PUNCH

An example of editing an input overcapacity punch follows:

Given:

Storage location 4221 contains the unprimed portion of an input word. Overcapacity punches have been placed in the last four digit positions of this word. They are in biquinary 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 four bits into the most significant digit position of *rA*.

STORAGE LOCAT'N	OP.	M	C
0019	25	4221	0023
0023	35	0025	0027
[0025	00	0000	1111]
0027	70	0029	0032
[0029	00	0000	4310]
0032	35	0034	0036
[0034	00	0000	5421]
0036	06	0039	
0039	32	0100	0043
0043	20	0004/3	0047
0047	32	0100	0057
0057	37	0100	0055
0055	20	0004/3	0059
0059	32	0200	0064
0064	37	0200	0069
0069	20	0004/3	0073
0073	32	0300	0079
0079	37	0300	0086
0086	20	0004/3	0090
0090	35	0092	MAIN PROG.
[0092	70	0000	0000]

Isolate four overcapacity punches from unprimed portion of word.

Add constant which places bits in 5, 4, 2 and 1 positions if appropriate.

Erase all but 5, 4, 2 and 1 positions.

Buff all bits into one digit position.

EDITING AN OUTPUT OVERCAPACITY PUNCH

An example of editing an output overcapacity punch follows:

Given:

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

Problem:

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

STORAGE LOCAT'N	OP.	M	C	
0120	25	0004/3	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	0004/3	0145	
0145	35	0147	0149	
[0147	00	0000	0004]	
0149	70	0151	0154	
[0151	00	0000	0006]	
0154	35	0156	0158	Edit 4 bit into punching position.
[0156	00	0000	0010]	
0158	37	0100	0162	
0162	20	0004/2	0166	
0166	77		0169	
0169	25	0004/3	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	0004/2	0190	
0190	77		0193	
0193	25	0004/3	0197	
0197	35	0199	0201	Edit 1 bit into punching position.
[0199	00	0000	0001]	
0201	20	0004/2	0205	
0205	20	4221	0223	Buff overcapacity punches into unprimed portion of word.
0223	60	4221	MAIN PROG.	

SUGGESTED CODING FOR OVER-CAPACITY INPUT-EDITING ROUTINE

(1) Unprimed Word to *rA* as *HHHHH-XXXXX* (must be +) (*XXXXX* = portion containing overcapacity; *HHHHH* = irrelevant).
(2) Exit Line to *rX*.

a	OC	m	c	Remarks
1015	31	1018	—	0 to rL
1018	35	1020	1022	Erase all but o/c digits
1020	00	0001	1111	CONSTANT
1022	82	1026	1025	Test for Zero
1025	75	1027	1030	Subtract 9999
1027	00	0000	9999	CONSTANT
1030	87	0004/3	1033	>=1 or 2 in LSD of rA
1033	70	1035	1038	Add 9002
1035	00	0000	9002	CONSTANT
1038	87	0004/3	1041	>=3 or 4 in LSD of rA
1041	70	1043	1046	Add 902
1043	00	0000	0902	CONSTANT
1046	87	0004/3	1049	>=5 or 6 in LSD of rA
1049	70	1051	1054	Add 92
1051	00	0000	0092	CONSTANT
1054	87	0004/3	1057	>=7 or 8 in LSD of rA
1057	25	1059	0004/3	9 to rA and EXIT
1059	00	0000	0009	CONSTANT
1026	26	0004/3	—	Zero to rA and EXIT

Notes:

(1) At the end of the routine, the appropriate overcapacity digit (0 through 9) is in LSD position of *rA*.
(2) It is presumed that the main program will shift left and buff in the balance of the field to complete the editing.
(3) Total time is 46 word times.
(4) The instruction locations shown are not intended to be compatible with the existing or forthcoming Library Routines. The programmer should choose suitable locations compatible with his own program.
(5) It is conceivable that this subroutine can be effectively coded in the regular bands if the programmer will time his entrances and exits appropriately.

Explanation: The overcapacity combinations ranging 1 through 0 are shown in the column labeled *o/c* (overcapacity).

Step 1. Subtract 9999.
If answer is plus, difference is *o/c* digit (1 or 2).
If answer is minus, perform Step 2.

Step 2. Add 9002.
If answer is plus, sum is *o/c* digit (3 or 4).
If answer is minus, perform Step 3.

Step 3. Add 902.
If answer is plus, sum is *o/c* digit (5 or 6).
If negative, perform Step 4.

Step 4. Add 92.
If answer is plus, sum is *o/c* digit (7 or 8).
If negative, bring 9.

Dec	o/c	Subtract	Diff	Add	Sum	Add	Sum	Add	Sum
1	10000	— 9999 =	1+						
2	10001	=	2+						
3	01000	=	8999—	+ 9002 =	3+				
4	01001	=	8998—	=	4+				
5	00100	=	9899—	=	897—	+902	= 5+		
6	00101	=	9898—	=	896—		= 6+		
7	00010	=	9989—	=	987—		=85—	+92	=7+
8	00011	=	9988—	=	986—		=84—		=8+
9	00001	=	9998—	=	996—		=94—		=2—
0	00000			Eliminate by initial test					

SUGGESTED CODING FOR OVER-CAPACITY OUTPUT-EDITING ROUTINE

(1) Overcapacity digit, in computer code, to rA as 000000000X.

(2) Exit Line to rX.

a	OC	m	c	Remarks
1215	37	0400	1222	Shift left 4
1222	70	1224	0004/1	Form Bring Order
1224	25	1229	0004/3	CONSTANT
(0004/1	25	1229	0004/3)	Not punched. Brings
(		thru	)	appropriate constant
(		1238	)	to rA
1229	00	0000	0000	"0"
1230	00	0001	0000	"1"
1231	00	0001	0001	"2"
1232	00	0000	1000	"3"
1233	00	0000	1001	"4"
1234	00	0000	0100	"5"
1235	00	0000	0101	"6"
1236	00	0000	0010	"7"
1237	00	0000	0011	"8"
1238	00	0000	0001	"9"

(0004/3 EXIT. Not punched. Drum level=1240 max.)

Notes:

(1) At the end of the subroutine, the five-level card code is formed in the 1 level of the five LSD positions of rA. The main program would arrange to position these bits and buff with the appropriate card image.

(2) Total maximum is 25 word times.

(3) The instruction locations given are not intended to be compatible with existing or forthcoming Library Routines. The programmer should choose locations compatible with his own program.

(4) It is possible that this subroutine can be effectively coded in the regular bands if the programmer will time his entrances and exits appropriately.

(5) Words 1229 thru 1238 can be readily coded to punch any type code: biquinary (5-4-2-1) or straight binary (8-4-2-1) as well as card code (1-3-5-7-9).

Appendix

KEY TO CARD INTERLACE TABLE

The 90-column punched card is represented in the Central Processor as twenty words. Every group of ten columns forms a data word of two images called the unprimed and primed images or a word-pair. (Columns 41-45 and 86-90 are each treated as 10-column groups and are placed into the five least significant digit positions in the computer words.) Each image is a computer word and is an exact representation of the holes appearing on a particular section of the card—a punch equals a 1 bit. The signs of all images are positive.

Row					
0 1 3 5 7 9	0	1	2	3	4
	0'	1'	2'	3'	4'
Columns	1-10	11-20	21-30	31-40	41-45
0 1 3 5 7 9	5	6	7	8	9
	5'	6'	7'	8'	9'
Columns	46-55	56-65	66-75	76-85	86-90

Any combination of punches may be represented within the system; however, translation instructions are provided only for the standard Remington Rand code.

To locate the interlace factor, the base of the symbol is

I=input on Read-Punch Unit

J=input on High-Speed Reader

O=output on Read-Punch Unit

The first subscript added to this base indicates the station; e.g., J_2 refers to input from the second read station of the High-Speed Reader. The second subscript refers to the column group on the card (see above).

The absence of an apostrophe refers to the unprimed word, and a prime indicates the primed word; e.g., O'_{12} refers to the primed word of card-column group 2 which is output on the first (and only) punch station of the Read-Punch Unit.

With a symbol such as this, the corresponding interlace factor can be added to the m band of the input or output instruction to determine the exact location of the card group.

INTERLACE TABLE
Read-Punch Unit

STORAGE LOCATION 00XX		STORAGE LOCATION 01XX	
00	50	00	50
01	51	01 I10	51 I22
02 I15	52 I27	02	52
03	53	03 O'19	53 O'12
04	54	04	54
05	55	05	55
06	56	06 I'10	56 I'22
07 I'15	57 I'27	07	57
08	58	08 O10	58 O17
09	59	09	59
10	60	10	60
11	61	11 I20	61 I13
12 I25	62 I18	12	62
13	63	13 O'10	63 O'17
14	64	14	64
15	65	15	65
16	66	16 I'20	66 I'13
17 I'25	67 I'18	17	67
18	68	18 O15	68 O13
19	69	19	69
20	70	20	70
21	71	21 I11	71 I23
22 I16	72 I28	22	72
23	73	23 O'15	73 O'13
24	74	24	74
25	75	25	75
26	76	26 I'11	76 I'23
27 I'16	77 I'28	27	77
28	78	28 O11	78 O18
29	79	29	79
30	80	30	80
31	81	31 I21	81 I14
32 I26	82 I19	32	82
33	83	33 O'11	83 O'18
34	84	34	84
35	85	35	85
36	86	36 I'21	86 I'14
37 I'26	87 I'19	37	87
38	88	38 O16	88 O14
39	89	39	89
40	90	40	90
41	91	41 I12	91 I24
42 I17	92 I29	42	92
43	93	43 O'16	93 O'14
44	94	44	94
45	95	45	95
46	96	46 I'12	96 I'24
47 I'17	97 I'29	47	97
48	98	48 O12	98 O19
49	99	49	99

INTERLACE TABLE
High-Speed Reader

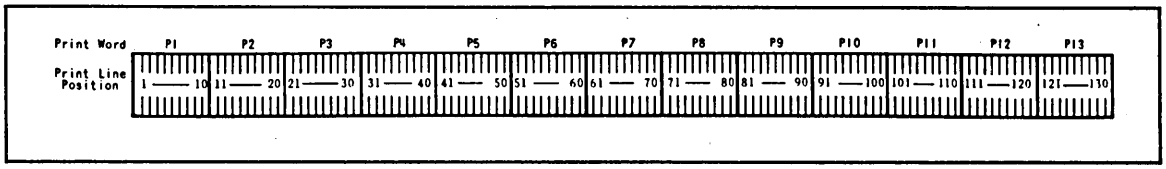
STORAGE LOCATION 00XX		STORAGE LOCATION 01XX	
00	50	00	50
01 J10	51 J22	01	51
02	52	02 J15	52 J27
03	53	03	53
04	54	04	54
05	55	05	55
06 J'10	56 J'22	06	56
07	57	07 J'15	57 J'27
08	58	08	58
09	59	09	59
10	60	10	60
11 J20	61 J13	11	61
12	62	12 J25	62 J18
13	63	13	63
14	64	14	64
15	65	15	65
16 J'20	66 J'13	16	66
17	67	17 J'25	67 J'18
18	68	18	68
19	69	19	69
20	70	20	70
21 J11	71 J23	21	71
22	72	22 J16	72 J28
23	73	23	73
24	74	24	74
25	75	25	75
26 J'11	76 J'23	26	76
27	77	27 J'16	77 J'28
28	78	28	78
29	79	29	79
30	80	30	80
31 J21	81 J14	31	81
32	82	32 J26	82 J19
33	83	33	83
34	84	34	84
35	85	35	85
36 J'21	86 J'14	36	86
37	87	37 J'26	87 J'19
38	88	38	88
39	89	39	89
40	90	40	90
41 J12	91 J24	41	91
42	92	42 J17	92 J29
43	93	43	93
44	94	44	94
45	95	45	95
46 J'12	96 J'24	46	96
47	97	47 J'17	97 J'29
48	98	48	98
49	99	49	99

KEY TO PRINT INTERLACE TABLE

The 130 characters that can be printed on one line are divided into thirteen ten-digit print words. These words, and their corresponding print positions, are shown in the diagram. Before a print instruction is executed, the thirteen print words are accumulated, by program instructions, on a specific main-storage band, in fixed word locations of the print interlace pattern.

To locate the interlace factor, the base of the symbol is *P*. The first subscript indicates the print-line position, 1 through 13 (see diagram). The absence of a prime refers to the unprimed word of a word-pair; a prime denotes the primed word.

The interlace factor corresponding to the symbol can be added to the *m* band of the print instruction to determine the placement of data to be printed.



INTERLACE TABLE
High-Speed Printer

STORAGE LOCATION 00XX				STORAGE LOCATION 01XX			
00	P1	50	P4	00		50	
01		51		01		51	
02		52		02		52	
03		53		03	P7	53	
04		54		04		54	
05	P'1	55	P'4	05		55	
06		56		06		56	
07		57		07		57	
08		58		08	P'7	58	
09	P10	59		09		59	
10		60		10		60	
11		61		11		61	
12		62	P13	12		62	
13		63		13		63	
14	P'10	64		14		64	
15		65		15		65	P3
16		66		16		66	
17		67	P'13	17		67	
18	P6	68		18		68	
19		69		19		69	
20		70		20		70	P'3
21		71		21		71	
22		72		22		72	
23	P'6	73		23		73	
24		74		24		74	
25		75		25	P9	75	
26		76		26		76	
27		77		27		77	
28		78		28		78	P12
29		79		29		79	
30		80		30	P'9	80	
31		81	P2	31		81	
32		82		32		82	
33		83		33		83	P'12
34		84		34	P5	84	
35		85		35		85	
36		86	P'2	36		86	
37		87		37		87	
38		88		38		88	
39		89		39	P'5	89	
40		90		40		90	
41	P8	91		41		91	
42		92		42		92	
43		93		43		93	
44		94	P11	44		94	
45		95		45		95	
46	P'8	96		46		96	
47		97		47		97	
48		98		48		98	
49		99	P'11	49		99	

TRANSLATION TABLE UNIVAC SOLID-STATE 90

	Remington Rand Card Code		Computer Code
	Primed	Unprimed	
Character	XX97	5310	5421
0	0000	0001	0000
1	0000	0010	0001
2	0010	0010	0010
3	0000	0100	0011
4	0010	0100	0100
5	0000	1000	1000
6	0010	1000	1001
7	0001	0000	1010
8	0011	0000	1011
9	0010	0000	1100
A	0010	1010	1011
B	0000	1010	1001
C	0001	0001	1010
D	0000	1101	1011
E	0000	0101	0011
F	0011	0010	1011
G	0001	1000	1010
H	0001	0100	1011
I	0000	1100	1011
J	0000	1110	1011
K	0010	1100	1101
L	0010	0001	1100
M	0000	1001	1000
N	0010	1001	1001
O	0000	0110	0011
P	0001	0110	1011
Q	0001	1100	1011
R	0001	0010	1011
S	0001	1010	1011
T	0011	0100	1111
U	0001	1001	1010
V	0010	0101	0100
W	0001	0101	1011
X	0011	0001	1011
Y	0010	0110	0110
Z	0011	1000	1011
Space	0000	0000	0000
:	0011	0110	1111
,(comma)	0010	1101	1101
$	0010	1111	1111
-	0001	1101	1011
#	0001	1011	1011
*	0000	0011	0001
%	0000	1011	1001
;	0011	1110	1111
/	0011	1100	1111
+	0011	1010	1011
.	0010	1110	1111
&	0001	1111	1011
'	0011	0111	1111
(	0011	1001	1011
)	0001	1110	1011
	0010	0110	0101
	0010	0110	0110
	0010	0100	0111
	0010	0000	1101
	0011	0000	1110
	0011	0000	1111

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

U-1741.4 REV. 2 PRINTED IN U.S.A.