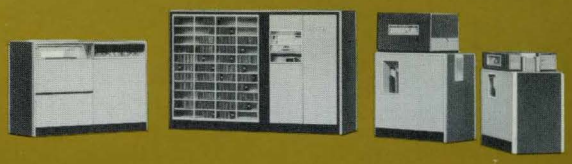


UNIVAC®
Solid-State 80
PROGRAMMING

X-6 ASSEMBLY SYSTEM

...a programming aid

PROGRAMMING

X-6 ASSEMBLY SYSTEM

...a programming aid

Contents

1. Introduction

PURPOSE AND ADVANTAGES

The X-6 Assembly System for the UNIVAC Solid-State 80 Computer is another in the series of Remington Rand relative coding systems designed to simplify the coding of data processing runs. Of the many advantages that can be claimed for the system, probably the most significant is the elimination of many sources of programming errors. X-6 reduces debugging time by permitting the programmer to code small sections of a problem individually, later assembling these segments into a larger unified program. The system also permits division of a large program among several programmers thus reducing overall coding time. Coding time is further reduced because X-6 performs automatically many of the jobs which are normally done by the programmer using computer code.

Automatic minimization, program list preparation, and conversion from a format convenient for coding and key punching, to an efficient format for a standard loading routine are also provided by the system.

THE NATURE OF THE X-6 ASSEMBLY SYSTEM

Programs, by nature, are composed of a number of subsections each designed to perform a definite function. These subsections, or the lines of coding which form them, are commonly called subroutines or operations. The X-6 Assembly System is a program which receives as input a series of these operations, assigns them as an integrated unit to storage locations, and produces as output a complete program deck as well as a parallel printer listing. During the assembly, relative and symbolic addresses are converted to absolute or actual storage locations in the computer.

ADDITIONAL FEATURES AND CONVENIENCES

Included among the many conveniences provided for the programmer by X-6 is a system of relative and symbolic addressing. Basically, a relative address is one which indicates a relationship between a line being referenced and some other line whose location has already been determined. A symbolic address is any arbitrary combination of characters defined within a system to represent a storage location. The value of the particular symbol is provided in some specified way by the programmer or the assembler. Also included in the system are simplified notations for expressing constant and working storage locations.

In addition to the features already mentioned, the X-6 Assembly System provides mnemonic instruction codes. Table I lists this code with its computer code equivalents as well as a description of each instruction.

This manual is intended as a reference manual for the X-6 programmer and a thorough understanding of the UNIVAC Solid-State 80 computer code and programming techniques is assumed.

INSTRUCTION CODES

Minimum Time in wt	Machine Code	Function	X-6	m	c	Notes
4	25	Load rA	LDA			
3	26	Clear rA to zeros	CLA	c	ŷ	
4	60	Store rA	STA			
3	36	Clear rA - Original sign remains	CAA	c	ŷ	
4	30	Load rL	LDL			
3	31	Clear rL to zeros	CLL	c	ŷ	
4	50	Store rL	STL			
4	05	Load rX	LDX			
3	06	Clear rX to zeros	CLX	c	ŷ	
4	65	Store rX	STX			
3	77	rA→rL	ATL	ŷ		
5	70	Add: rA + m	ADD			
5	75	Subtract: rA - m	SUB			
105	85	Multiply: rL x m	MUL			
115	55	Divide: m ÷ rL	DIV			
4	20	Buff: m onto rA	BUF			
4	35	Erase: rA controlled by m	ERS			
4	62	Zero suppress: in rA and rX	ZUP	ŷ		
3	17	Translate: computer-to-card code	MTC	ŷ		
3	12	Translate: card-to-computer code	CTM	ŷ		
3+n	32	Shift right: rA and rX	SHR	n		n = number of places to shift = 0, 1, ...10
3+n	37	Shift left: rA	SHL	n		n = number of places to shift = 0, 1, ...10
3	82	Test equal : rA & rL	TEQ	c'		c' = next instruction if rA = rL
3	87	Test greater: rA & rL	TGR	c'		c' = next instruction if rA > rL
2	00	Jump	JMP	c	ŷ	
Ind.	67	Stop	STP	c'		c' = alternative next - instruction (requires manual intervention)
3(4 if c')	22	RPU Buffer Test	RBT	c'		c' = next instruction if RPU free for use
203*	46	RPU Buffer Unload	RBU	Rnood		n = 0, 1, ...9 for 0th thru 9th RPU input interlace; d = 0, normal translation
203*	81	RPU Card Cycle	RCC	Onood		n = 0, 1, ...9 for 0th thru 9th RPU output interlace; d = 1, "on the fly"
3	57	RPU Select Stacker 1	RSS	ŷ		
3(4 if c')	42	HSR Buffer Test	HBT	c'		c' = next instruction if HSR free for use
203*	96	HSR Buffer Unload	HBU	Hnood		n = 0, 1 ...9 for 0th thru 9th HSR interlace (d is specified as in RBU, RCC)
3(4 if c')	72	HSR Card Cycle	HCC	c'		c' = next instruction if HSR busy
3	47	HSR Stacker Select	HSS	noo		n = 0, 1, 2 for stacker #0, #1, or #2
3(4 if c')	27	Printer test	PBT	c'		c' = next instruction if printer is free for use
3	16	Printer advance	PFD	yy		yy = 00, 01, ..., 79 = no. lines to advance
592	11	Advance & print	PRN	Pnoee		n = 0, 1, ...9 for 0th thru 9th print interlace
						ee as in PFD above
14	86	Clear rA and rX, sign of rL goes to rA and rX	CAX	c		
2	23	rC→rA	CTA	c		
3	02	Load Index Register	LIR			
4	07	Increment Index Register	IIR			

LEGEND:

1. Unless otherwise noted, standard X-6 *m* and *c* addresses may be used.
2. All *m* and *c* addresses use leading spaces to make up the 5 digits allowed.
3. *c* indicates address of next instruction in other than standard position.
4. ŷ indicates that this address is ignored; any 5 digits may be used.

* *Applicable only when d = 0.*

2. X-6 Addressing

ABSOLUTE ADDRESSES

In the X-6 Assembly System, the *a*, *m*, and *c* addresses contain a five-digit field in the form:

aaaaa hhh mmmmm ccccc

where aaaaa is the address of the instruction in storage.

hhh is the mnemonic instruction code.

mmmmm is the address of an operand; the location of the next instruction to be executed; or can be ignored depending upon the instruction to be executed.

ccccc is the address to which control is to be transferred.

When referencing an absolute storage address, the address is placed in the least significant digit positions of that part of the instruction to which it applies. The unused portion of the most significant digits is filled with deltas.

For example, an instruction in 3465 to load register A with the contents of storage location 1959 and then go to storage location 0372 would appear as:

Δ3465 LDA Δ1959 ΔΔ372

The delta is used as an empty column or space indicator and is inserted mainly for the convenience of the Key Punch operator. However, as greater familiarity is gained with X-6, the filler symbol may be omitted, thus reducing the burden on the programmer. It should be noted that zeros may be substituted for spaces, and

Δ3465

may be written

03465

Any numeric or space in the first digit position of an address will cause the last four digits to be regarded as absolute and not to be modified in any way.

Certain instructions require parameters which are classified here as absolute addresses merely to indicate that X-6 does not process them.

1. The *m* address of a shift instruction is written as

   ΔΔΔΔN or ΔΔΔNN

   where N is 0 through 9 and NN is 10.

2. The *m* address of stacker select instructions is entered as

   ΔΔN00 or Δ0N00

   where N is 0, 1, or 2.

3. The *m* address of a paper feed is entered as

   ΔΔΔYY or Δ00YY

   where YY can be 00-79.

INTERLACES

A five-digit address in the *m* portion of an instruction references a specific word location in an interlace. The five digits have the following significance.

Digit 1 may be:

H for the read interlace of the High-Speed Reader.

R for the read interlace of the Read-Punch Unit.

O for the punch interlace of the Read-Punch Unit.

P for the print interlace of the High-Speed Printer.

Digit 2 may be:

0-9 thus allowing ten different interlaces for each type of input-output unit.

Digit 3 may be:

U or P, or D for unprimed, primed, or duo-primed.
N or Z for the numeric or zone word.

H, R, or O interlaces are specified in card type 4* as either two or three part images. If a two part image has been specified, only N or Z should be used to reference the particular words.

Digits 4 and 5 may be:

10-17 for read station 1 of the High-Speed Reader or the read interlace of the Read-Punch Unit or 20-27 for the second read station.

10-17 for the punch interlace.

01-13 for the print interlace.

To address the primed image in the seventh word location of a card at the first read station of the High-Speed Reader and stored in an input band assigned to contain interlace number two, the following five-digit address would be used:

H2P17

The address of the same data after it has been read at the second read station would be:

H2P27

Referencing a location in the print interlace is slightly different in that no indication is made to a particular station. Instead only the desired word location is entered in the last two digits of the address. Since the print interlace consists of 13 word locations, the last two digits may be any in a range of 01 to 13. The address of the unprimed image in the fourth word location of a print interlace assigned as interlace number two, would be:

P2U04

However, when addressing a complete interlace, digits 3, 4, and 5 are specified differently. For example, the instruction to advance the paper two lines and print the contents of the P2 interlace is entered as

PRN P2002 *c*

It should be noted that the H, R, and O interlaces have zeros in digits 3, 4, and 5 when data is in its three part form. It should be further noted that if digit 5 is a 1, the instructions RCC, RBU, or HBU will be interpreted as a call for translation 'on the fly'.

TABLES

A table may be defined as a series of values put into storage at regularly spaced intervals. To address entries made in tables, the programmer need only indicate the type and number of the table and the number of the entry within the table. This address is in the form:

inxxx

See pages 13 and 14.

where i is the type of table (S, U or V).*

n is the number assigned to that table.

xxx is the number of the entry in the table.

For example, the address of the fifteenth entry under a table assigned the symbol S and designated as table number two would be:

S2015

X-6 allows 30 tables of up to 1000 words each with their successive entries separated by an increment. This increment must remain constant within a table.

SPACES

Spaces may be used to denote the next instruction. This method reduces the amount of writing for the programmer. If spaces are used in either the *m* or *c* portion, the next *a* address must also be spaces. For example,

a	op	*m*	*c*
00249	LDA	ΔΔΔΔΔ	00251
ΔΔΔΔΔ	[CONSTANT]		
00251	STA	00365	ΔΔΔΔΔ
ΔΔΔΔΔ	[NEXT INSTRUCTION]		

Note: this device does not apply to instructions where *m* or *c* is ignored and spaces are used as fillers.

K AND W AREAS

The K and W areas contain constants and working storage addresses, respectively, whose locations are not dependent upon their relationship to each other. Addressing of these entries is in five-digit notation and in the form:

iΔxxx

where i is the letter assigned to an area.

Δ is ignored.

xxx is the number of the desired entry.

**X-6 makes no distinction between these three types of tables.*

Any operation may reference these K or W areas in the *m* or *c* addresses. The programmer may enter the initial value in any working storage by writing an operation, usually WWW with the five-digit notation as the *a* addresses.

Similarly, the constants themselves may be entered as an operation, usually KKK, with KΔΔΔ0, KΔΔΔ1, and so on, as the *a* addresses.

There are 400 pooled constants and 400 working storages allowed in X-6. All pooled constants and all working storages are automatically assigned to high-speed access storage unless it has been filled. However, as far as the constant pool is concerned, better access time will occur if it is assembled after the operations referencing it. If constants contain relative or symbolic addresses they may not be placed in the pool.

TAGS

Tags are used to identify instructions in the program to which reference may be made. These tags are used as *a* addresses for lines of coding or constants which are referenced in the *m* and *c* portions of other lines of coding. The programmer may assign them to high-speed or standard access storage. X-6 provides for two types of tags, temporary and permanent.

TEMPORARY TAGS

The temporary tag is used to identify references within an operation and are meaningful *only within that operation.* These tags are in the form:

ΔΔxxi

where ΔΔ is ignored.

xx is the tag identifier. It may be numeric, alphabetic, or alpha-numeric.

i is N or F indicating that the line is to be assigned to standard or high-speed access storage.

is O in a *c* address indicating an instruction the execution of which could result in overflow. O in an *a* address indicates the next instruction to be executed if overflow does not occur.

is P indicating the line to which control will be transferred should overflow occur (*c*+1)*.

The following is an example illustrating a possible use of O and P in an overflow condition:

*O and P are automatically assigned to high-speed storage.

```
  a      op      m       c
ΔΔΔΔΔ  ADD   WΔΔΔ1   ΔΔΔ2O      (Indicates overflow condition)
ΔΔΔ2O  [if no overflow]
ΔΔΔ2P  [if overflow occurs]
ΔΔΔΔΔ  [next instruction]
```

Note: Temporary tags do not have to be in any particular sequence. Only 50 temporary tags may be used in any one operation, usually Δ1 through 50 to assist the programmer to avoid exceeding the limit.

PERMANENT TAGS

These tags serve the same function as temporary tags but are not restricted to singular operations.* They provide a means of communication among different operations or different elements within any one operation. Permanent tags can be assigned absolute addresses. These tags are in the form:

nnnxi

where nnnx is the tag identifier and may be alphabetic or numeric. It is often desirable to enter nnn as the operation number.

i is N for standard access.

F for high-speed access.

O or P to indicate whether the tag applies to a *c* or *c* + 1 line in case of overflow.**

The permanent tag

2054F

indicates that this tag, tag 4 in operation 205, is to be assigned an address in high-speed access storage.

The tag

SIN5N

is tag 5 in operation SIN and is to be assigned an address in standard access storage.

REGISTER ADDRESSES

ΔΔΔRA, ΔΔΔRL, and ΔΔΔRX are used as register designations in X-6. It is recommended that when an instruction is being executed in a register, this line of coding be listed on the coding paper with the register as the *a* address. This line will not be punched in the output but it will show on the listing and will allow X-6 to do a more efficient minimization.

**See card type 3, page 13.*
***O and P are automatically assigned to high-speed storage.*

LIBRARY ROUTINES

An X-6 library routine differs from any other operation only in that it may contain variable addresses in the *a*, *m*, or *c* portion of an instruction. Variables are written in the form:

XΔΔnn

where nn may be any number from Δ1 to 20 for any single operation. Variable addresses permit the programmer to supply parameters for any library routine he uses. It should be noted that any X-6 operation may be written as a library routine. Furthermore, the individual programmer can increase the usefulness of the system by initially coding commonly used functions as library routines thereby making them available for use by others.

SAMPLE PROBLEMS

1. 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.

2. Given:

Data	Form	Location
Income	GGGGGGGG00	WΔΔΔ1
Number of dependents	0NN0000000	WΔΔΔ2
Deductions other than for dependents	00AAAAAA00	WΔΔΔ3

A deduction of $600 is allowed for each dependent. The tax is 20% of taxable income. Store the tax in location 4073 in the form:

00TTTTTTTT

3. Given:

Data	Form	Location
Quantity ordered	00QQQQ0000	WΔΔΔ1
Unit price	PPPP000000	WΔΔΔ2

If the quantity is greater than or equal to 100, apply a discount of 40%. Otherwise, apply a discount of 30%. Store the charge in location 4053 in the form:

00CCCCCCCC

3. Assembly Preparation

GENERAL INFORMATION

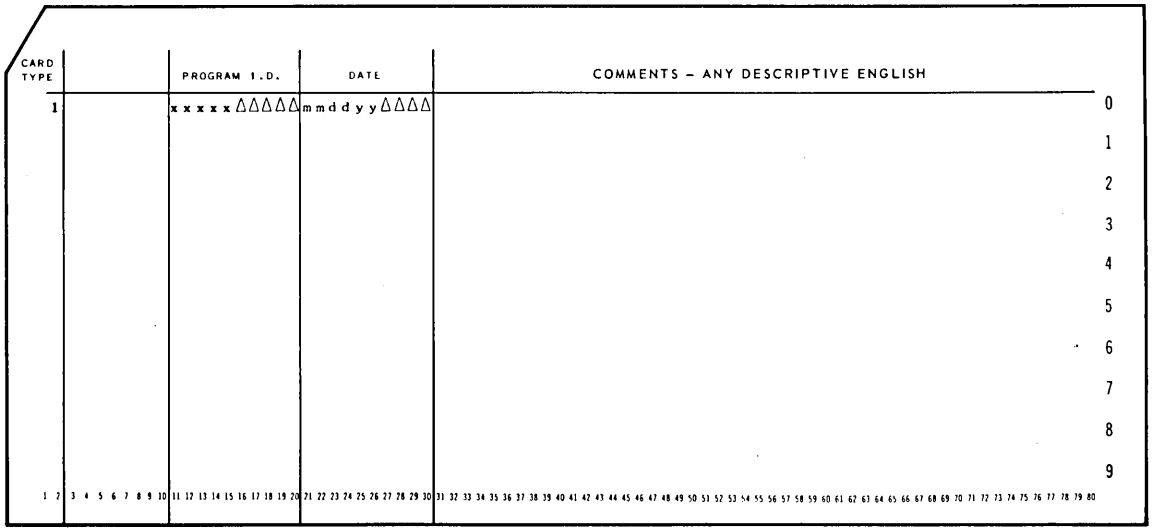
Each operation in a program has a header card, detail cards (cards containing coding and constants), and an end-operation sentinel card. Library routines are considered operations. However, they may have additional specification cards which serve the purpose of particularizing any variables within them. In addition, summary cards are introduced for control purposes or to increase the efficiency of the routines.

CARD TYPES

The following is a list and description of the various input cards used in an operation. It should be noted that card types 1, 7, 8, 9, and 10 are always used while the other summary cards can be considered optional and dependent upon the specific operation.

CARD TYPE 1 - LABEL

A label card contains the output program identification and any title information desired on the printer listing.

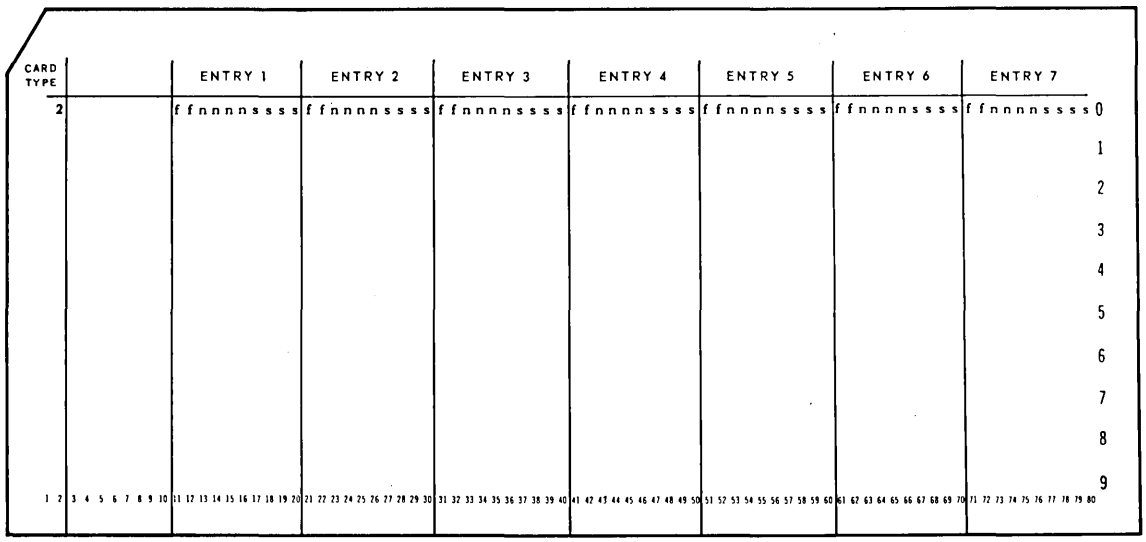


Here, xxxxx is the five-digit program identification.

mmddyy is month, day, year.

CARD TYPE 2 - RESTRICTS

Restrict cards mark off certain areas or locations as unavailable for the assembly. This means that the system will not assign a relative or symbolic address to a storage location in a restricted area. The programmer, however, *may* specify an *absolute address,* an *interlace,* or a *table* within this area if he deems it necessary. Up to *seven* entries may be made on the card with no limit on the number of cards used. The last valid entry is followed by a word of 9's as a sentinel.



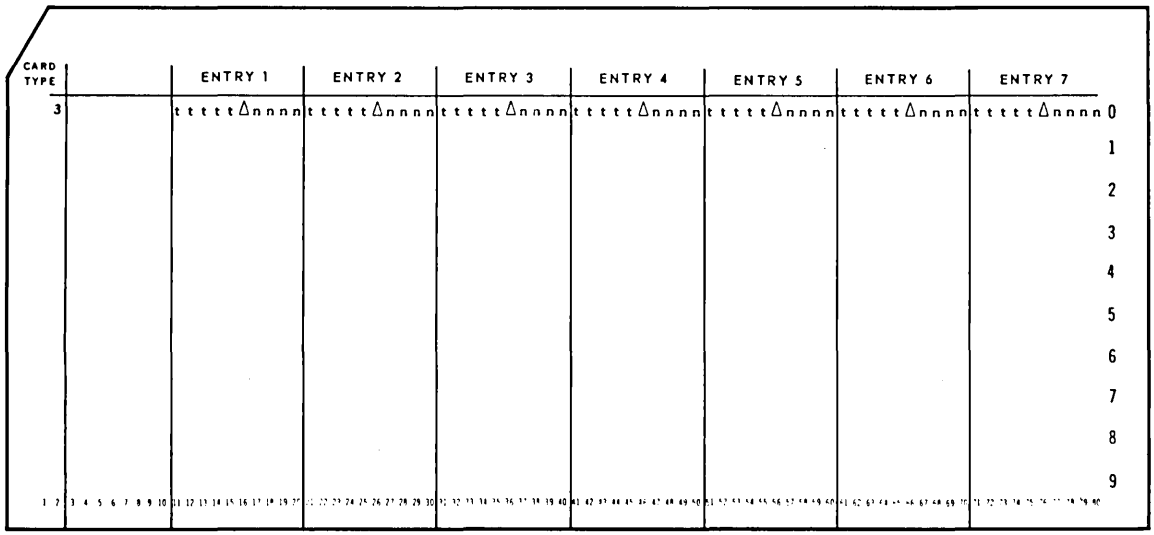
Here, ff is the increment.

nnnn is the total number of restricted addresses.

ssss is the absolute starting address.

CARD TYPE 3 - TAG EQUALS

This card makes it possible to assign absolute addresses to permanent tags used in a program. Up to seven entries may be made with no restriction on the number of cards used. The last valid entry is followed by a word of 9's.

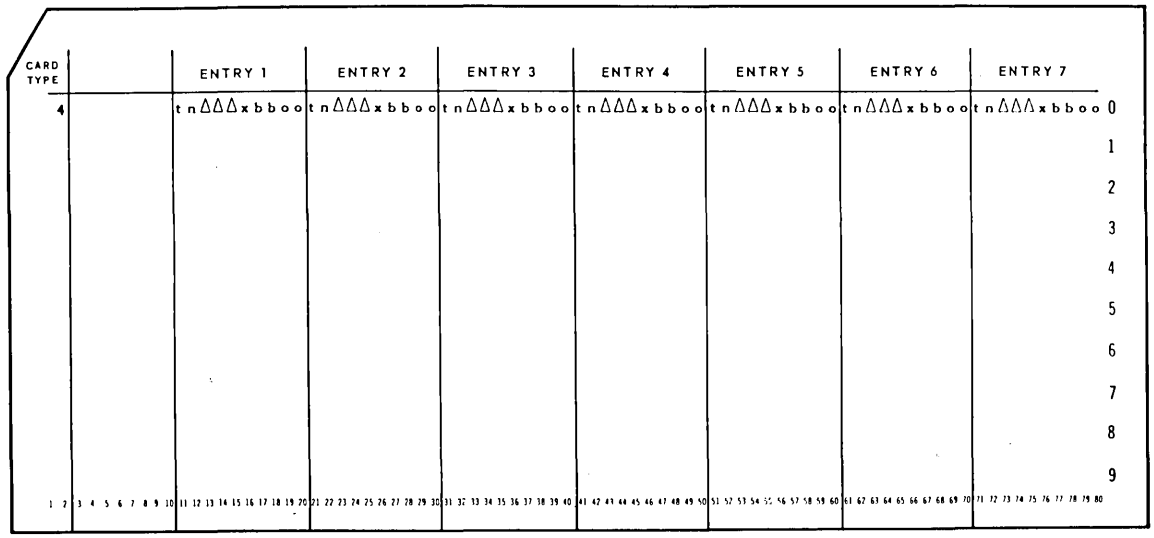


Here, ttttt is the tag.

nnnn is the absolute address.

CARD TYPE 4 - INTERLACES

All input-output interlaces required in the problem are entered here. Up to seven entries may be made with no restriction on the number of cards. The last valid entry is followed by a word of 9's.



Here x is 0 for a three part interlace.*

1 for a two part interlace.

2 if both kinds are specified.

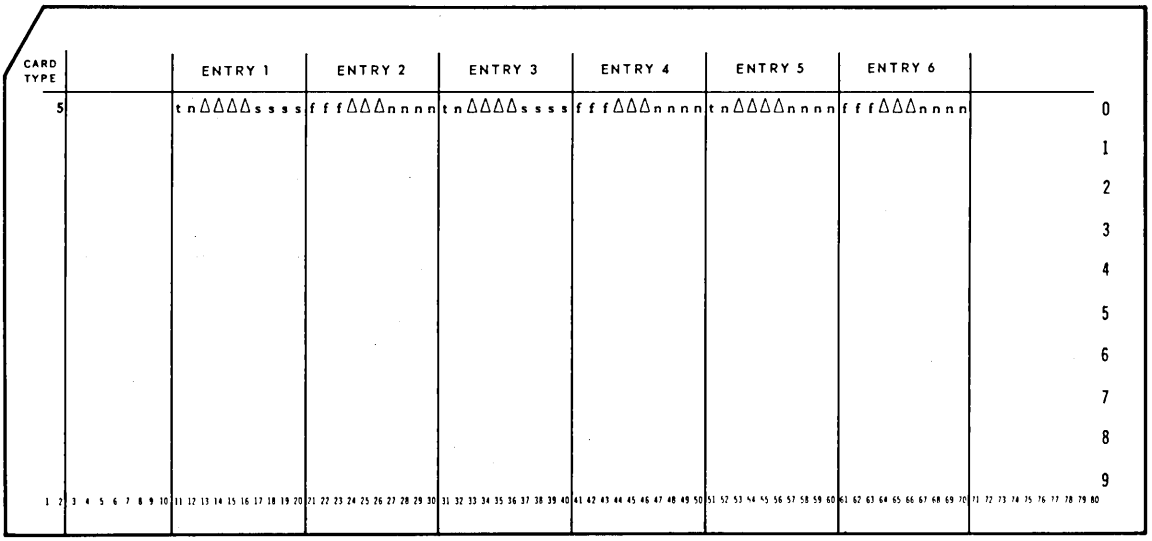
t is the type of interlace, R, P, O, or H.

n is interlace number (0-9).

bboo is the absolute address of the band. (bb must be an even number).

CARD TYPE 5 - TABLES

Tables are arrays of numbers separated by a fixed increment. The entries are in pairs with a word of 9's following the last valid entry.



Here, t is the type of table; S, U, or V.

n is table number (0-9).

ssss is the absolute starting address.

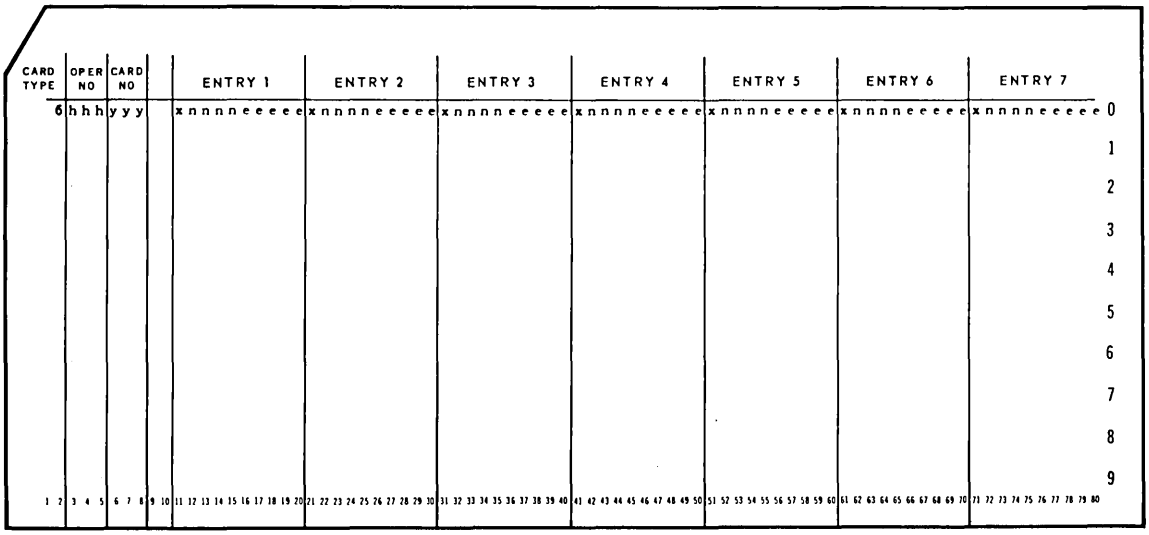
fff is the increment.

nnnn is the total number of entries in the table.

**The notations here for x are not applicable to the print interlace.*

CARD TYPE 6 - SPECIFICATIONS

Specification cards precede library routines and are used to modify coding within the routine before it is assembled. These cards will use the operation number of the library routine. Therefore, the first card is card number 1. The last valid entry is followed by a word of 9's.



Here, hhh is the operation number.

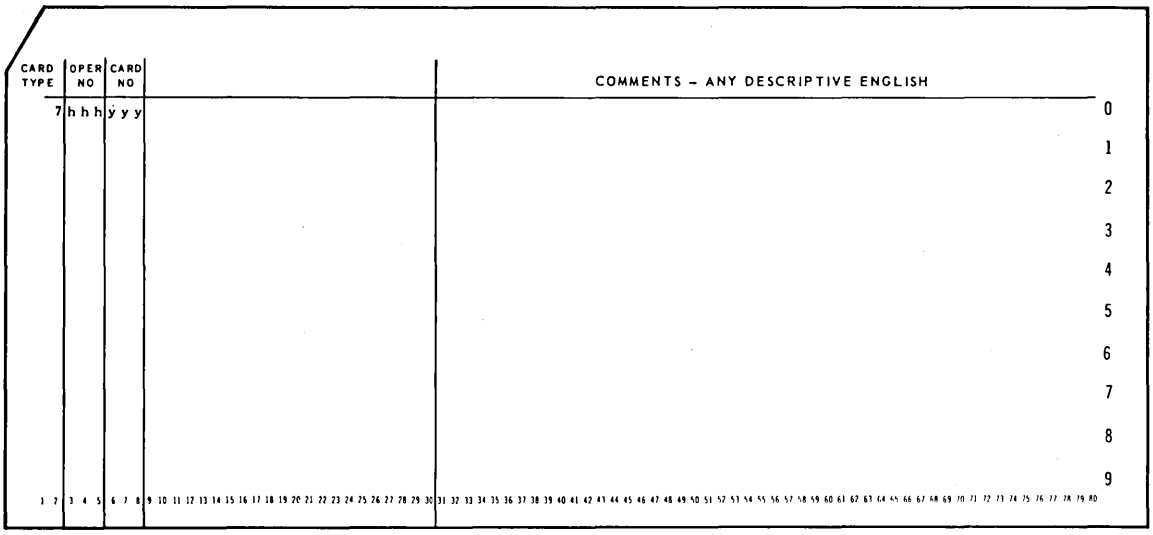
yyy is the card number.

xnnnn is the nth variable, as x0001.

eeeee is the X-6 equivalent address.

CARD TYPE 7 - HEADER

A header card begins each operation (unless preceded by specifications) and may contain, for listing purposes, a description of the function performed by the operation.

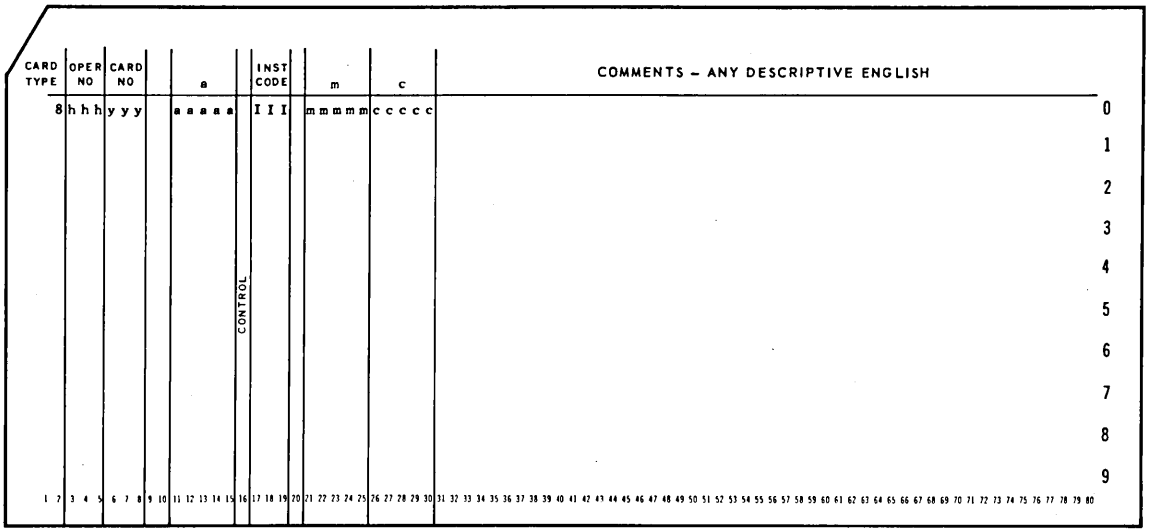


Here, hhh is the operation number.

yyy is the next card number (001 if no specification cards are used).

CARD TYPE 8 - DETAIL

These cards are the lines of coding and the constants. The card numbers are in ascending sequence starting one higher than the header. An operation may not exceed 999 lines.



Here, hhh is the operation number.

yyy is the card number.

aaaaa is the X-6 *a* address.

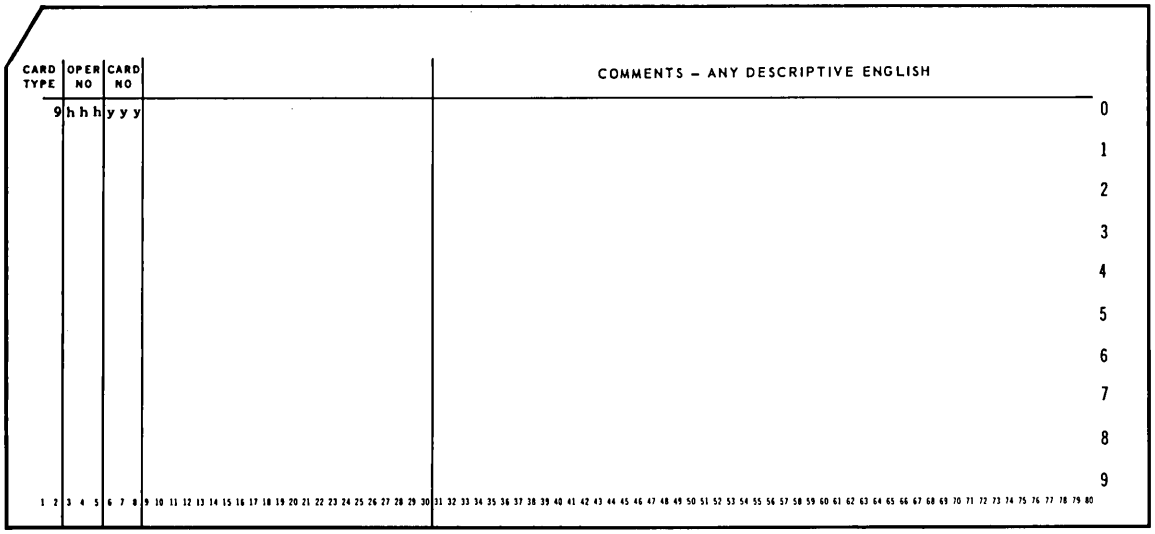
III is the mnemonic instruction code.

mmmmm is the X-6 *m* address.

ccccc is the X-6 *c* address.

Control is 1, 2, or 3 for the appropriate index register.
U, P, or D for three part alphabetic constants.
N or Z for two part alphabetic constants.
2 for a negative constant.

CARD TYPE 9 - END-OPERATION SENTINEL

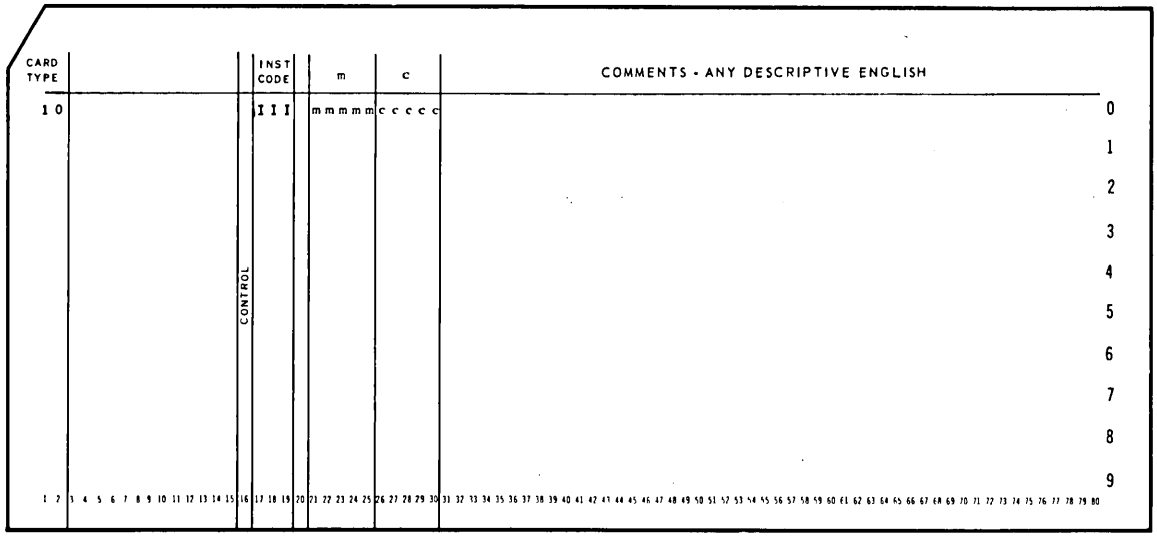


Here, hhh is the operation number

yyy is the card number.

CARD TYPE 10 - END INPUT

This is an end-of-input sentinel card containing the first instruction of the assembled program which will be executed after the program is fully loaded.



Here, fields III, mmmmm, and ccccc are the same as for the detail card.

4. Constants

INTRODUCTORY CONSIDERATIONS

Whether pooled or included with the coding, constants are recognized by the blanks in the columns reserved for mnemonic instruction codes. The 10-digit constant is listed in the *m* and *c* address columns on the coding sheet. Positive constants are indicated by a space (Δ) in the control column and negative ones by a two (2).

Sometimes alphabetic constants are needed in either a two or three part form for printer or punch output. As a convenience, X-6 allows the programmer to write the alphabetic constant twice with N and Z or U, P, and D in the control column. This makes it unnecessary for the coder to break up the alphabetic characters into the machine pulse patterns which will re-create the desired alphabetics when printed or punched.

NON-NUMERICS

There are six non-numeric computer-coded characters. The alphabetic equivalents for these six characters are represented in the following manner.

0101	A
0110	B
0111	C
1101	D
1110	G
1111	H

Again, a space (Δ) or a two (2) in the control column will indicate a positive or negative value.

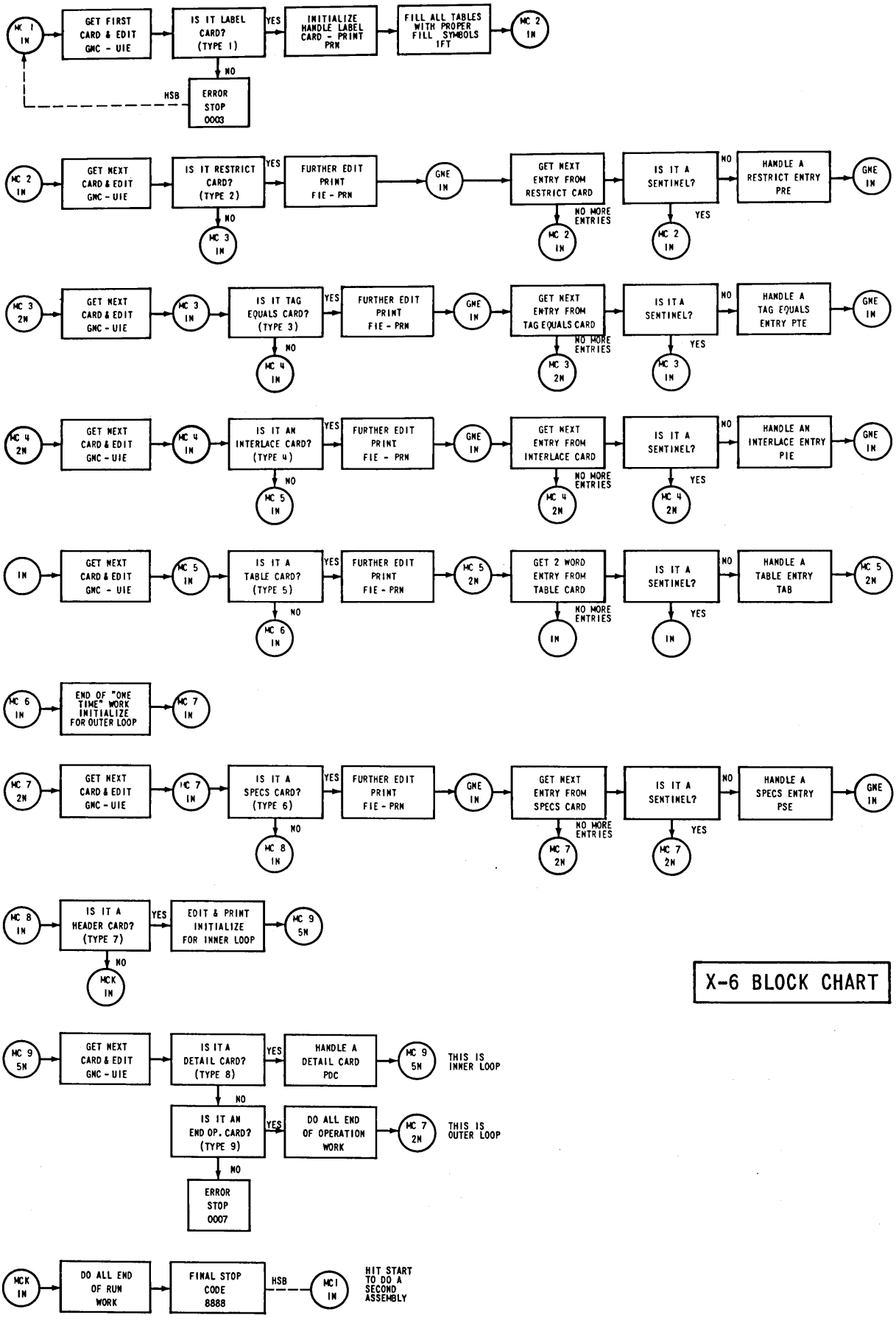
SUMMARY

a. Positive numeric constants have a space (Δ) in the control column.

b. Negative numeric constants have a two (2) in the control column.

c. A, B, C, F, G, and H are equivalents for the six non-numeric computer-coded characters.

d. Alphabetic constants (positive only) have a U, P, D, N, or Z in the control column.


MC 1 IN
GET FIRST CARD & EDIT GNC - UIE
IS IT LABEL CARD? (TYPE 1)
YES
INITIALIZE HANDLE LABEL CARD - PRINT PRN
FILL ALL TABLES WITH PROPER FILL SYMBOLS IFT
MC 2 IN
NO
HSB
ERROR STOP 0003
MC 2 IN
GET NEXT CARD & EDIT GNC - UIE
IS IT RESTRICT CARD? (TYPE 2)
YES
FURTHER EDIT PRINT FIE - PRN
GNE IN
GET NEXT ENTRY FROM RESTRICT CARD
IS IT A SENTINEL?
NO
HANDLE A RESTRICT ENTRY PRE
GNE IN
NO
MC 3 IN
NO MORE ENTRIES
MC 2 IN
YES
MC 2 IN
MC 3 2N
GET NEXT CARD & EDIT GNC - UIE
MC 3 IN
IS IT TAG EQUALS CARD? (TYPE 3)
YES
FURTHER EDIT PRINT FIE - PRN
GNE IN
GET NEXT ENTRY FROM TAG EQUALS CARD
IS IT A SENTINEL?
NO
HANDLE A TAG EQUALS ENTRY PTE
GNE IN
NO
MC 4 IN
NO MORE ENTRIES
MC 3 2N
YES
MC 3 IN
MC 4 2N
GET NEXT CARD & EDIT GNC - UIE
MC 4 IN
IS IT AN INTERLACE CARD? (TYPE 4)
YES
FURTHER EDIT PRINT FIE - PRN
GNE IN
GET NEXT ENTRY FROM INTERLACE CARD
IS IT A SENTINEL?
NO
HANDLE AN INTERLACE ENTRY PIE
GNE IN
NO
MC 5 IN
NO MORE ENTRIES
MC 4 2N
YES
MC 4 2N
IN
GET NEXT CARD & EDIT GNC - UIE
MC 5 IN
IS IT A TABLE CARD? (TYPE 5)
YES
FURTHER EDIT PRINT FIE - PRN
MC 5 2N
GET 2 WORD ENTRY FROM TABLE CARD
IS IT A SENTINEL?
NO
HANDLE A TABLE ENTRY TAB
MC 5 2N
NO
MC 6 IN
NO MORE ENTRIES
IN
YES
IN
MC 6 IN
END OF "ONE TIME" WORK INITIALIZE FOR OUTER LOOP
MC 7 IN
MC 7 2N
GET NEXT CARD & EDIT GNC - UIE
MC 7 IN
IS IT A SPECS CARD? (TYPE 6)
YES
FURTHER EDIT PRINT FIE - PRN
GNE IN
GET NEXT ENTRY FROM SPECS CARD
IS IT A SENTINEL?
NO
HANDLE A SPECS ENTRY PSE
GNE IN
NO
MC 8 IN
NO MORE ENTRIES
MC 7 2N
YES
MC 7 2N
MC 8 IN
IS IT A HEADER CARD? (TYPE 7)
YES
EDIT & PRINT INITIALIZE FOR INNER LOOP
MC 9 5N
NO
MCK IN
X-6 BLOCK CHART
MC 9 5N
GET NEXT CARD & EDIT GNC - UIE
IS IT A DETAIL CARD? (TYPE 8)
YES
HANDLE A DETAIL CARD PDC
MC 9 5N
THIS IS INNER LOOP
NO
IS IT AN END OP. CARD? (TYPE 9)
YES
DO ALL END OF OPERATION WORK
MC 7 2N
THIS IS OUTER LOOP
NO
ERROR STOP 0007
MCK IN
DO ALL END OF RUN WORK
FINAL STOP CODE 8888
HSB
MC1 IN
HIT START TO DO A SECOND ASSEMBLY

5. How X-6 Works

INPUT PROCESSING

Programmers will be able to make more effective use of X-6 if they understand how it performs its function.

Each input card type follows a different path. A brief statement of the steps performed on each card type follows:

1. The fields in the label card (type 1) are carried over to the output without modification.

2. The entries in restrict cards (type 2) are used to mark off locations in the storage availability table. X-6 must not allocate any restricted addresses.

3. The entries in tag-equals cards (type 3) are filed in internal tables equated to the given absolute addresses. The absolute addresses are marked off in the storage availability table.

4. The entries in interlace cards (type 4) are used to mark off interlace positions in the storage availability table, and the origins are filed for future use.

5. The entries in table cards (type 5) are handled in a similar fashion. The only change is that increments as well as origins are saved for future use.

Card types 1-5 must have been received in order, and after the first type 6 or 7 card, no additional 1-5 cards will be accepted. At this point the initial phase of X-6 is complete, and from this point on the routine expects card types 6-9 or 7-9 on a per-operation basis.

6. The entries in specifications cards (type 6) are filed in tables for direct substitution later.

7. The header card (type 7) is used to initialize for the detail cards which follow.

8. Detail cards (type 8) encompass all of the lines of coding and constants which make up a routine. Only detail cards cause output punching. Processing these cards is the primary function of X-6.

9. The end-operation card (type 9) sentinels the end of a group of detail cards.

10. The end-problem card (type 10) sentinels the end of a run and contains the instruction which will be used by the loading routine to start the execution of the assembled program.

DETAIL CARD PROCESSING

The processing of detail cards involves four basic steps:

1. Handle the *a* address.

2. Analyze the instruction code, separating instructions from constants. For instructions, a code word is obtained which controls further processing by indicating the increment needed between various addresses as well as the computer code equivalent for the mnemonic code. It also indicates which addresses are significant.

3. If required, handle the *m* address.

4. If required, handle the *c* address.

MINIMIZATION

The table of Instruction Code Information Words (Table 2) indicates how X-6 minimizes if it is free to choose the addresses. The *a*, *m*, and *c* addresses, if all three are important, go through the same analysis. As a result of this analysis an absolute address is assigned and never altered. It is the input order which controls the allocation rather than the sequence of execution of these instructions. For straight-line sections of coding, X-6 will do as well as the careful programmer. The first reference to any address will be minimum. A simple example will serve to clarify the procedure.

a	*INST*	*m*	*c*
IN	LDA	H1U10	
	STA	W 6	2N

Is IN a new temporary tag? If yes, use clock* to establish tentative best address. The clock reads 058. Get an address. 0258 is free, so use it and mark it off in the storage availability table. File it opposite IN in the temporary tag table. Look up the information word for LDA. Add two word times to establish 060 as the tentative best address. Send H1U10 through address analysis. 4003 must be assigned. Adjust the clock to 103. Add two word times to establish 105 as the tentative best band address. Send the blanks in *c* through address analysis.

**The clock is a counter whose value lies between 000 and 199. The value increases as each instruction is assigned to storage. The function of the clock is to indicate the current band level.*

Get an address. 305 is free, so use it and mark it off. Set a switch so that 305 will be used for the next *a* address. If a tag is found there, an error code will be printed. Print and punch the output and go to the next input instruction.

The second instruction will take four word times unless W6 or 2N have been encountered previously.

As experience is gained, programmers will be able to code operations and order input to achieve better minimization from an X-6 assembly process.

TABLE 2. INSTRUCTION CODE INFORMATION WORDS

If control column indicates Index register modification, add one more word time before m.

	Digits 1-2	Digit 3 Action Code	Digits 5-7 Before m	Digits 8-10 Before c	
ADD	70	0	002	003	
BUF	20	0	002	002	
DIV	55	0	002	113	
ERS	35	0	002	002	
LDA	25	0	002	002	
LDL	30	0	002	002	
LDX	05	0	002	002	
MUL	85	0	002	103	
STA	60	0	002	002	
STL	50	0	002	002	
STX	65	0	002	002	
SUB	75	0	002	003	
LIR	02	0	000	003	
IIR	07	0	000	004	
ATL	77	1	000	003	
CTM	12	1	000	003	
MTC	17	1	000	003	
ZUP	62	1	000	004	
CLA	26	2	003	000	
CLL	31	2	003	000	
CLX	06	2	003	000	
JMP	00	2	002	000	
CAA	36	2	003	000	
CAX	86	2	014	000	
CTA	23	2	002	000	
HSS	47	3	000	003	
PFD	16	3	222	003	222 is a code not affecting timing
RSS	57	3	000	003	
SHL	37	3	111	003	111 means use amount of shift
SHR	32	3	111	003	
HBU	96	4	198	203	
PRN	11	4	197	592	
RBU	46	4	198	203	
RCC	81	4	198	203	
HBT	42	5	004	003	
HCC	72	5	004	003	
PBT	27	5	004	003	
RBT	22	5	004	003	
STP	67	5	003	003	
TEQ	82	5	003	003	
TGR	87	5	003	003	

6. Programming Procedure

FLOW-CHARTING

The only suggested modifications in standard flow-charting practices are as follows:

1. Define your operations as you flow-chart -- keep them short.

2. Use *large* circles for communication links *between* operations; assign permanent tags to these circles.

3. Use *smaller* circles for communication links *within* operations; assign *temporary* tags to these circles.

4. Use X-6 symbology in the flow chart. Assign table and interlace symbols, and working storage addresses at this time.

CODING

1. Start *each* operation with a header, card type 7, on a new piece of coding paper.

2. Code the main chain first and then the lesser used branch paths. Since each address is allocated the first time it is encountered, this technique will produce better minimization.

3. Use the comments columns liberally. The X-6 edited listing will be more valuable if full comments are appended. Limit your comments to numbers and the alphabetics.

4. Use the card number as a cross reference to the box on the flow chart.

5. End each operation with an end-operation sentinel, card type 9.

6. Be sure all working storages are filled properly initially. Main storage is often filled with stop orders rather than zeroes.

7. Buffer tests must be inserted by the programmer when required. Accurate estimates can be made by consulting the section on 'Minimization'.

PREPARATION FOR ASSEMBLY

1. Have all operations keypunched and verified.

2. Obtain any needed library routines and prepare specification cards.

3. Prepare card types 1, 2, 3, 4, 5, and 10 if this has not already been done. Be sure to restrict the area used by the standard loading routine.

4. Arrange the input deck in the desired order. If the program is very large, place the most important operations first; they will get better minimization.

5. Sight check the separate operations to make certain that card types 7, 8, and 9 within each operation are identically punched in columns 3-5, (operation number).

6. Either manually or by machine, check that card numbers are ascending within operations with *no omissions*.

ASSEMBLY

1. Follow the X-6 operating instructions.

2. Check the edited listing carefully, all detected input data errors are coded and tabulated in print word 01 on the listing. These errors must be corrected before debugging can commence.

3. The output program deck is complete in stacker zero of the Read-Punch Unit. Any cards in stacker 1 should be destroyed.

7. Operating Instructions

LOADING AND ASSEMBLING

1. Load X-6 program deck. Successful stop is 67TTTT000T.

2. After X-6 is loaded, or earlier:

   a. Feed blank cards through to all stations of the RPU.

   b. Advance paper in HSP so six free holes show above the paper holding clamps.

   c. Put X-6 input program deck in the HSR.

3. To assemble a program:

   a. Go on continuous, general clear, and run.

   b. Successful stop is 678888cccc.

   c. Error stops are listed on the following pages along with error codes which do not stop the computer.

4. Get a memory dump to preserve the information accumulated during the assembly which will be useful for debugging.

   The X-6 Memory Layout in Appendix II can be used to interpret the memory dump.

ERROR CODES (THESE APPEAR ON LISTING)

CODE	ORIGINATES IN OP.	MEANS
A	PTS	More than 400 perm. tags. Address 9999 has been assigned.
B	TTS	More than 50 temp. tags. Address 9999 has been assigned.
C	KWS	Address higher than K 399 or W 399 has been requested. 9999 has been assigned.
D	MAR	No more storage -- have assigned 9999.
E	MAR	No two consecutive addresses free. Have assigned 9999.
F	STS	Nothing in specs table matches this "X" symbolic address. Absolute 9999 has been assigned.
G	AAR	An incorrect *a* address. Previous instruction had blanks in *m* or *c* part. This *a* should have been blank. This *a* has been processed properly - the previous line must be fixed.
H	PDC (AC2)	Spaces in *m* and *c*. Spaces in *m* will be assumed to be in error.
I	ICA	Invalid instruction code.

STOP CODES (IN M PART OF STP ORDER)

CODE	ORIGINATES IN	MEANS
0001	GN2	The card being diverted to HSR Stacker #2 has failed to pass read check. Reposition cards and hit start to try again.
0002	GNC	Malfunction in HSR has caused overflow. Fix trouble. Hit start to try again.
0003	MCO	No label card (type 1). Prepare label card. Reposition input deck. Hit start to begin again.
0004	PSE	Too many specs for current library routine. Hit start to proceed. Error code F will appear later.
0005	PRN	Malfunction in printer has caused overflow. Fix trouble. Hit start to print current line. *(IT WAS PRN ORDER THAT CAUSED IT)*
0006	PUN	Malfunction in RPU. Fix trouble. Hit start to execute punch order.
0007	MC9	Card type sequence error. Check last card read. If it is a type 7 card, hit start to get to next stop order. Go to *c* to process card. If it is type 8, go to *m* of next stop order.
0008	PDC	Operation number on detail card is incorrect. Hit start and machine will stop on 67 order. Go to *m* to process card. Go to *c* to get next card.
0009	PDC	Card number on detail card incorrect. Same action as 0008 STOP.
8888	MCK	Final successful stop. Follow normal operating instruction before hitting start if new assembly is wanted.

Appendix I

PROBLEM SOLUTIONS

Problem I Solution

a	op	m	c	
ΔΔΔ1N	LDA	Δ3182		
	ADD	Δ3178	ΔΔΔ2O	Tag indicates possible overflow condition
ΔΔΔ2O	STA	Δ3210	ΔΔ271	If overflow does not occur
ΔΔΔ2P	LDX		ΔΔΔ3N	If overflow occurs
		00000	00001	
ΔΔΔ3N	STX	Δ3200	ΔΔΔ2O	

Problem 2 Solution

a	*op*	*m*	*c*	
ΔΔΔ1N	LDL	WΔΔΔ2		
	MUL		ΔΔΔ2N	Total deduction for dependents
		60000	00000	
ΔΔΔ2N	LDX	ΔΔΔrA		
	LDA	WΔΔΔ1		
	SUB	ΔΔΔrX		
	SUB	WΔΔΔ3		
	ATL			
	MUL		ΔΔΔ3N	Derive tax
		00200	00000	
ΔΔΔ3N	STA	Δ4073		Store tax

Problem 3 Solution

a	op	*m*	*c*	
ΔΔΔ1N	LDA	WΔΔΔ1		Quantity⟶rL
	LDL		ΔΔΔ2N	
		00009	^90000	
ΔΔΔ2N	TGR		ΔΔΔ3N	Compare for magnitude
	MUL	WΔΔΔ2		Quantity x Price⟶rA
	ATL			
	MUL	KΔΔΔ1	ΔΔΔ4N	KΔΔΔ1 = ^7000000000
ΔΔΔ4N	STA	Δ4053		
	STP			
ΔΔΔ3N	MUL	WΔΔΔ2		Quantity x Price⟶rA
	ATL			
	MUL	KΔΔΔ2	ΔΔΔ4N	KΔΔΔ2 = ^6000000000

Appendix II

X-6 MEMORY LAYOUT

A memory dump at the end of a successful assembly is desirable for debugging and patching of object programs.

LOCATION	NAME	USE
1200	Table S8	Valid mnemonic codes stored 20 words apart.
1216	Table S9	Information words for each mnemonic code stored 20 words apart.
2110-2117	Table V3	Two or three part interlace word position for O.
2118-2130	Table V4	Two part interlace word position for P.
2100-2109	Table S5	Interlace origins (from card type 4).
2200 Band	02 Interlace	Repunching of output cards which fail read check.
2400-2449	Table S3	Temporary tags with absolute addresses. Cleared after every operation. No value after complete assembly.
2450-2465	Table V2	Two and three part interlace word positions for H and R.
2470-2479	Table S6	Interlace origins (from card type 4).
2480-2509	Table S7	Table origins and increments (from card type 5).
2520-2539	Table V1	X-6 equivalents for last set of specifications.
2540-2559	Table V0	Specifications. Cleared after every operation. No value after complete assembly.
2600-2999	Table S4	K and W addresses and absolute addresses are stored as follows: 2600 K0 and W0 as 0KKKK0WWWW 2601 K1 and W1 as 0KKKK0WWWW
3000-3199	Table S2	Addresses of permanent tags in same order as Table S1, stored as: 0aaaa0aaaa Left half-words used for first 200 tag-addresses, then right half-words are filled.

LOCATION	NAME	USE
3200-3599	Table SI	Permanent tags. The 5 character alpha-numeric tag is stored as zzzzznnnnn. One tag per word.
3600-3799	Table SO	Storage availability. Each word of table represents a band relative address, 0-199. The 20 bits in the left half-word are zero for unused or 1 for used representing the 20 standard access bands. The 20 bits in the right half of words 3600-3649 represent high-speed access storage. Addresses 4000, 4050, 4100 and 4150 are included in first digit of right half-word. Right half of words 3650-3799 are unused.
3800 Band	PO Interlace	Header for X-6 listing.
4000 Band	HO Interlace	High-Speed Reader read-in area.
4200 Band	OI Interlace	Output punching area.
4200 Band	RO Interlace	Read-Punch Unit read-in area.
4400 Band	PI Interlace	Detail lines for X-6 listing.
000-0199	Restricted	Used to load X-6 and later filled with memory dump routine.

Appendix III

SAMPLE LISTING

The following is a sample of the listing produced by X-6 which affords the programmer a detailed correlation of computer and X-6 code.

```
X6B80  OP   CD  LOCA OP MMMM CCCC K   A TAG  C   OP  M TAG C TAG   TP  CD    COMMENTS              071659  PAGE   2
       AAR    1                                                      7        ADDRESS ANALYSIS ROUTINE
       AAR    2 0200 50 4002  204     AAR1N      STL AAR5F AAR2N    8  0001  SET EXIT
       AAR    3 0204 31  207         AAR2N      CLL                8  0002  REENTRY POINT CIRCLE 1
       AAR    4 0207 25 4009  211                LDA W   0 AAR3N    8  0003  ZZZZZNNNNN
       AAR    5 0211 82  414  214     AAR3N      TEQ   19N    1N    8  0004  SWITCH 1 A SETTING  LDX AAR6N
       AAR    6 0414 50 4116  218       19N      STL W  13          8  0005  ZERO TO FN INDICATOR
       AAR    7 0218 30 4002  254                LDL AAR5F MAR1N    8  0006  GO TO STORAGE AVAILABILITY ROUTINE
       AAR    8 0214 05 4009  261        1N      LDX W   0          8  0007  ZZZZZNNNNN
       AAR    9 0261 26  264                     CLA                8  0008
       AAR   10 0264 32 0500  272                SHR     5          8  0009
       AAR   11 0272 20    T  276                BUF    RX          8  0010  NNNNNZZZZZ
       AAR   12 0276 35 4028  280                ERS K  29          8  0011  0000H0000H
       AAR   13 0280 30    K  284                LDL    RA          8  0012  0000N0000Z
       AAR   14 0284 06  287                     CLX                8  0013
       AAR   15 0287 32 0500  295                SHR     5          8  0014
       AAR   16 0295 37 0500  303                SHL     5          8  0015
       AAR   17 0303 12       306                CTM                8  0016  0000D00000
       AAR   18 0306 17       309                MTC       47N      8  0017  RA IS 1111N111111 RX IS 0000Z00000
       AAR   19 0509 32 0500  317       48N      SHR     5          8  0018
       AAR   20 0317 37 0500  325                SHL     5          8  0019
       AAR   21 0325 20    T  329                BUF    RX          8  0020  0000N0000Z
       AAR   22 0329 82  532  332                TEQ       6N       8  0021  IF NOT EQUAL DIGIT 5 IS ALPHA
       AAR   23 0532 25 4009  361                LDA W   0          8  0022  ZZZZZNNNNN
       AAR   24 0361 35 4013  365                ERS K  8           8  0023  HHHHHH0000
       AAR   25 0365 37 0500  373                SHL     5          8  0024  UNPRIMED PART OF DIGIT 1 D00000000000
```

Remington Rand Univac
DIVISION OF SPERRY RAND CORPORATION

PRINTED IN U.S.A. U-1964