S

IBM

Reference Manual

IBM 7750 Assembly Program Using the IBM 1401

IBM®

Reference Manual
IBM 7750 Assembly Program Using the IBM 1401

This publication supersedes the previously published IBM bulletin, "IBM 7750 Assembly Program Using the IBM 1401," form J28-0254.

Address comments regarding this publication to
Programming Systems Publications, IBM Corporation, P.O. Box 390, Poughkeepsie, N.Y.

CONTENTS

INTRODUCTION

This manual contains a description of the 7750 Assembly Program and has been written with a view to facilitate the preparation of programs for the IBM 7750 Programmed Transmission Control. The assembly medium is the IBM 1401. That is, the Assembly Program runs on the 1401 and produces output suitable for loading into the 7750 through the IBM 1410 or 7000 Series Data Processing Systems.

It is assumed that the reader is familiar with the general details and operational techniques of the 7750 Programmed Transmission Control which are described in the IBM General Information Manual "IBM 7750 Programmed Transmission Control," Form D22-6627.

MACHINE REQUIREMENTS

The 1401 system must have the following minimum configuration:

1. 4,000 positions of Core Storage.
2. Three IBM 729 II, 729 IV, or 7330 Magnetic Tape Units.
3. Advanced Programming Features.
4. High-Low-Equal Compare Feature.
5. IBM 1403 Printer, Model 2.
6. IBM 1402 Card Read-Punch.

ASSEMBLY FEATURES

The 7750 Assembly Program will provide the following:

1. mnemonic operation codes for all 7750 imperative operations;
2. symbolic referencing of instructions, data, and areas of 7750 storage;
3. pseudo-operations that are made up of two classes:
   a. declarative operations
   b. processor control operations.

An imperative statement is a symbolic representation of a 7750 machine instruction. The Assembly Program will convert each statement of this type to a single 7750 instruction word in the object program.

Declarative statements may be of two types: data definition, and storage reservation. A declarative statement of the data definition type is used to define constant information for inclusion in the object program. One or more data characters, one or more octal words, or a specially formed word such as a Limit Word, a Channel Word, or a Process Word may be defined. Declarative statements of the storage reservation type cause the Assembly Program to reserve areas that will be used by the 7750 program, but these statements do not cause any data to be loaded with the object program.

Processor control statements do not give rise to any assembled output; instead, they enable the programmer to communicate necessary information to the Assembly Program and to exercise control over various aspects of the assembly process.

STORAGE ASSIGNMENT

Although the 7750 is a fixed-word machine, a word may be interpreted in several ways, depending upon its use in the program. While Instruction Words, Limit Words, Process Words, and Channel Words are unique entities, a Data Word contains within it four characters, each uniquely addressable. Thus, it is important to note that the basic unit of storage in the 7750 is a character.

All actual 7750 addresses must be specified to the Assembly Program as decimal numbers. In a 7750 with 4,096 words of storage, the maximum actual address is 16383. In a 7750 with 8,192 words of storage, the maximum actual address is 32767. In a 7750 with 16,384 words of storage, the maximum actual address is 65535. A word is defined as a string of four characters, the first of which must be located at an address which is exactly divisible by four.

The Assembly Program utilizes two counters to control the assignment of storage. The first of these, the Current Assignment Counter, contains the address of the next available character location in the current sequence of storage assignment. Each time a character is assigned, as in certain data definition operations, the value of the counter is increased by one. When assembling an instruction or a word definition, the program must assign a full word of storage. The Assembly Program first determines whether the next character available is the first character of a word, i.e., is the address of this character exactly divisible by four. If it is not, the value in the Current Assignment Counter is increased by one, two, or three until it has reached the address of the next available word. The word is assigned and the value of the Current Assignment Counter is increased by four. Thus, after a word is assigned, the address in the counter is that of the next sequential word in 7750 storage.

There are two process control operations which enable the programmer to affect the value of the Current Assignment Counter. These are the ORG and LOC statements, which are discussed in detail below. It is important here to establish only one fact. During the assembly, the highest value attained by the Current Assignment Counter is retained by the program when storage assignments are made under ORG control. This value is stored in the High Assignment Counter. The final value of the High Assignment Counter is used by the Assembly Program to determine the first available block for chaining purposes.

CODING SHEET

The 7750 Assembly Program Coding Sheet (Form X28-1625) is used for writing source program entries for input to the Assembly Program. Each line on the coding sheet is punched in a separate card. The fields comprising each line are discussed below.

HEADING LINE

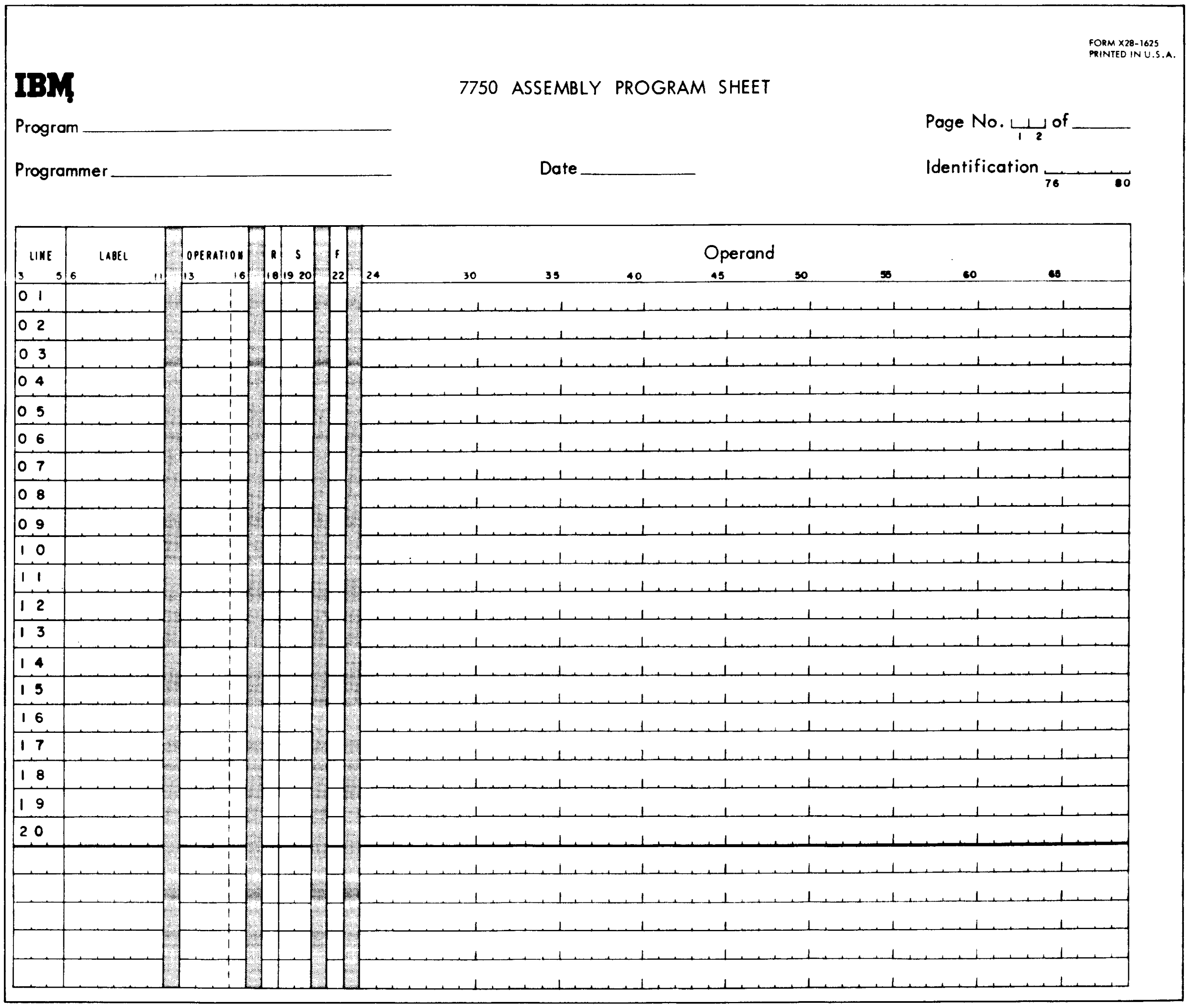
The space provided at the top of the 7750 Assembly Program Coding Sheet is meant to identify and date the program. The information entered in areas labeled "Program," "Programmer," and "Date," is not a part of the source program language and is therefore not punched in cards.

IDENTIFICATION (Columns 76-80)

This entry enables the programmer to identify his program. This field is punched in every card of the source program. The identification field that appears in the control card will appear in every record of the object program.

PAGE NUMBER (Columns 1 and 2)

This two-character field provides sequencing for the coding sheets. When page number assignments are made, any alphameric characters may be used, but the standard collating sequence of the IBM 1401



FORM X28-1625
PRINTED IN U.S.A.

IBM

7750 ASSEMBLY PROGRAM SHEET

Program ____________________ Page No. ⌊1⌋⌊2⌋ of ______

Programmer ____________________ Date __________ Identification ⌊76 … 80⌋

LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	Operand 24 30 35 40 45 50 55 60 65
0 1						
0 2						
0 3						
0 4						
0 5						
0 6						
0 7						
0 8						
0 9						
1 0						
1 1						
1 2						
1 3						
1 4						
1 5						
1 6						
1 7						
1 8						
1 9						
2 0						

Figure 1.

should be followed. This procedure is necessary to take advantage of the Assembly Program's automatic facility for performing sequence checking on the input program.

LINE (Columns 3-5)

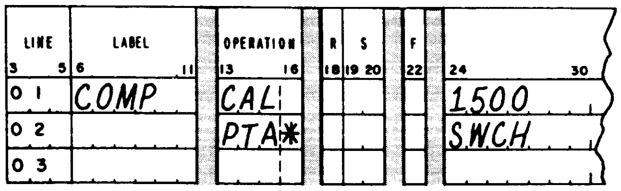
This three-character field provides sequencing for the entries on each coding sheet. Columns 3 and 4 of the first twenty lines are prenumbered. The last five lines are unnumbered and may be used for subsequent insertions. The units position of the field is used to indicate the location of the insertion. For example, should it become necessary to insert a line between 05 and 06, the insertion might be numbered 051. Any alphameric character may be used, but again, the standard 1401 collating sequence should be observed. This procedure is necessary because when the Assembly Program performs sequence checking it does so upon both the page number and the line number (columns 1-5) of each card.

LABEL (Columns 6-11)

The label field is provided to enable the programmer to assign a symbolic designation to an instruction, a data character, or an area of core storage. This symbol may consist of a maximum of six alphameric characters. It must be written left-justified in the label field. The first character must be alphabetic. To avoid assembly errors the programmer must ensure that every symbol used in his program is uniquely defined, i.e., a particular symbol should appear in the label field only once. If the programmer wants to take advantage of the header facility provided in the Assembly Program, he should limit the label to a maximum of five alphameric characters.

OPERATION (Columns 13-16)

The three-character mnemonic operation code is positioned in columns 13-15. (For a complete list of 7750 mnemonic codes see appendix A.) If an instruction is to be executed using an indirect address, an asterisk (*) is placed in column 16. Otherwise this position is left blank. See Figure 2.



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30
0 1	COMP	CAL				1500
0 2		PTA*				SWCH
0 3						

Figure 2.

R (Column 18)

This single character field identifies the register to be used in the execution of an instruction. All imperative operations of the character manipulative type and some of the control type require that a register be specified. The register may be specified by either its numerical or alphabetic designation. Figure 3 lists all of the addressable 7750 registers, their alphabetic and numerical designations, and their maximum size.

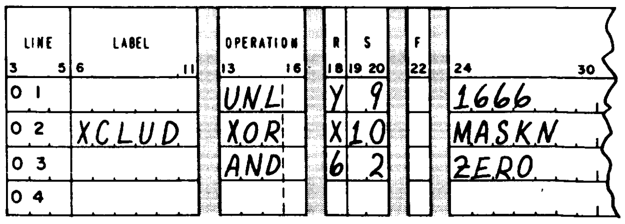
7750 Program Addressable Registers			
REGISTER	Alphabetic Designation	Numerical Designation	Maximum Size
NULL REGISTER[1]	N	0	11
X REGISTER	X	1	11
Y REGISTER	Y	2	11
Z REGISTER	Z	3	11
CHANNEL SERVICE REGISTER	C	4	11
INTERFACE DATA REGISTER	D	5	9
INTERFACE CONTROL REGISTER	I	6	8
MODE REQUEST REGISTER	M	7	5

[1] This is a register which always contains eleven zeros. It may be used only for address modification, never in the R field.

Figure 3.

S (Columns 19-20)

This two-character field indicates the size or number of bits of the register to be involved in a particular operation. S is a decimal number and may range in value from 0 through 11. S should never exceed the maximum size of the register specified by R. If S is a single digit, it should be placed in column 20. It is not necessary to insert a high-order zero. Figure 4 illustrates some uses of the R and S fields:



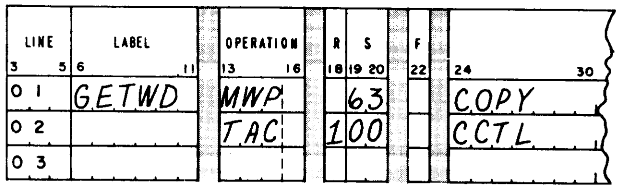
LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30
0 1		UNL	Y	9		1666
0 2	XCLUD	XOR	X	10		MASKN
0 3		AND	6	2		ZERO
0 4						

Figure 4.

RS (Columns 18-20)

Imperative operations of the storage-to-storage type require that the address of a word in Control Storage be specified. For instructions of this type, the R and S fields are joined to form a single three-character field which contains the Control Storage address. The address is specified as a decimal number ranging in value from 0 through 127. Any address that is less than three characters must be placed right-justified in the field. High-order zeros may be omitted.

When storage-to-storage instructions are executed in Channel Service mode, the Control Storage word address is obtained from the Channel Service Register, not from the instruction. However, the programmer must insert a value in this field. A single zero will suffice; otherwise, the Assembly Program will indicate an error. The illustration in Figure 5 contains some examples of the use of the RS field:



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30
0 1	GETWD	MWP		63		COPY
0 2		TAC	1	00		CCTL
0 3						

Figure 5.

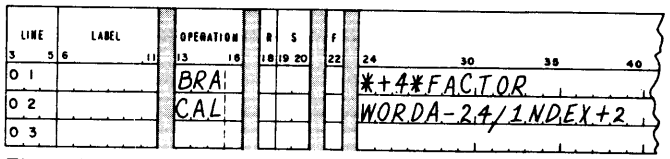
F (Column 22)

This single character field enables the programmer to communicate to the Assembly Program that he wants to flag the operation. A flag may be associated with any imperative operation. Its presence is effective at object time. The flag field may contain digits 1, 2, or 3, or it may be left blank. If the character is 1, the contents of the Z register will be decremented by one before the execution of the instruction. If the character is 2, no change of mode will be permitted to occur between the execution of this instruction and the instruction succeeding it in the same mode. If the character is 3, both of the above actions will be performed: i.e., the Z register will be decremented and mode change will be inhibited. If the field is left blank, none of these actions will take place.

OPERAND (Columns 24-69)

The operand field beginning in column 24 is free-form. It contains the address of the data to be acted upon by the imperative command in the operation field. The address may be a decimal number, a symbol, or an asterisk (*). If an asterisk is used, the contents of the Current Assignment Counter will be substituted. This field must not be left blank in an imperative statement. It must contain at least a zero in column 24, or the Assembly Program will indicate an error.

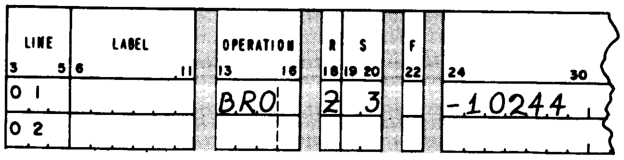
Character adjustment for symbolic or asterisk addresses may be specified. All forms of arithmetic manipulation, i.e., addition (+), subtraction (-), multiplication (*), and division (/), may be used. The adjustment factors may be decimal numbers or symbols. A maximum of nine adjustments may be applied to an address. The usual rules of algebra apply; that is, multiplication and division are performed before addition and subtraction. Any remainder resulting from a divide operation is discarded. The actual address specified or the address generated after character adjustment must not exceed the maximum character size of the object 7750. Figure 6 illustrates an example of address adjustment:



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30 35 40
0 1		BRA				*+4*FACTOR
0 2		CAL				WORDA-24/INDEX+2
0 3						

Figure 6.

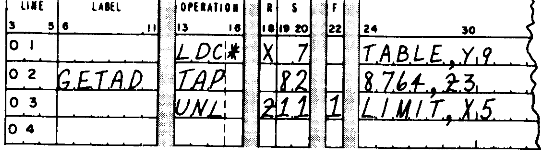
The address specified in an imperative operation may be prefixed by a plus or minus sign. A plus sign is equivalent to no sign at all; that is, the Assembly Program will generate the address as indicated. However, if the address is prefixed by a minus sign, the Assembly Program will substitute for the generated address its complement, modulo the indicated 7750 machine size. For example, if the 7750 has 4,096 words of storage (i.e., equal to 16,384 characters), the address field generated in the example illustrated in Figure 7 will be 6140_{10}.



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30
0 1		BRO	2	3		-10244
0 2						

Figure 7.

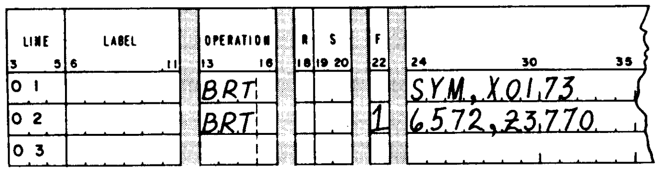
Address modification, where applicable, is indicated immediately after the address and adjustment field, and is separated from that field by a comma. The alphabetic designation of the register to be used in the modification and a decimal number specifying the length or number of bits of the register that is to be effective must be indicated. Figure 8 illustrates how address modification is specified:



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30
0 1		LDC*	X	7		TABLE,Y9
0 2	GETAD	TAP		82		8764,23
0 3		UNL	2	11	1	LIMIT,X5
0 4						

Figure 8.

The Branch on Test operation (BRT) presents a single exception to the use of the operand field as discussed above. The location to which control is to be transferred if the test is successful is indicated in the usual way. However, immediately following this address is a comma and a five character field. The first character is the alphabetic designation of the register whose content is to be tested. The succeeding four characters are the octal representation of the mask to be used in the test. The mask must always be defined as four octal digits. High-order zeros may not be omitted. The mask may range in value from 0000 through 3777. Figure 9



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30 35
01		BRT				SYM,X0173
02		BRT			1	6572,Z3770
03						

Figure 9.

illustrates how the BRT operation is specified. No R and S fields are specified for this operation. Note that in line 02 the Z register will be decremented by one before the test is performed.

COMMENTS

At least one blank must separate the operands from any comments the programmer may want to include with the statement. No blanks may appear within the operands themselves, because the Assembly Program assumes that all characters following the first blank encountered after column 24 are comments. Comments appear on the assembly listing but do not affect the assembled program in any way. If the programmer wishes to devote an entire card to comments, an asterisk must be punched in column 6. The remainder of the card (column 7-69) will then be treated as comments.

DECLARATIVE OPERATIONS

Declarative statements are source language statements which present information about a program, e.g., data description.

There are two types of declarative operations that are used in the 7750 Assembly Program. The first type is used to make storage reservations for the characters, words, and blocks, and the second type is used to define the data. A detailed description of declarative operations that are used to store constants and define work areas is presented below. A complete list of declarative operation codes is illustrated in Appendix A.

STORAGE RESERVATION

CSS: Reserve Character(s)

This operation enables the programmer to reserve a single character location or a series of sequential character locations in core storage.

Label: This field may contain a symbolic address which will reference the first character location reserved. If more than one character location is being reserved by a single statement, the succeeding characters may be referenced by using this label with appropriate character adjustment.

Operation: CSS

Operand: This field is used to specify the number of characters to be reserved and must contain either a decimal number or a symbol.

Decimal Number: The actual number, in decimal, of characters to be reserved must be specified.

Symbol: A symbol equivalent to the number of characters to be reserved must be used. This symbol must not be character adjusted and must have been previously defined in the label field of a preceding statement.

WSS: Reserve Word(s)

This operation is used to reserve one word or a series of sequential words in core storage.

Label: This field may contain a symbolic address which will reference the first word in the sequence. Character adjustment must be provided by the programmer to reference any other words in the sequence. For example, if a statement reserves three words, the second word would be addressed by the label +4.

Operation: WSS

Operand: This field is used to specify the number of words to be reserved in the sequence and must contain either a decimal number or a symbol.

Decimal Number: The actual number of words to be reserved is specified in decimal form.

Symbol: A symbol equivalent to the number of words to be reserved may be used. This symbol must not be character adjusted and must have been previously defined in the label field of a preceding statement.

BSS: Reserve Block(s)

This operation enables the programmer to reserve one block or a series of sequential blocks in core storage. A block is defined as eight sequential words, the first of which has a decimal address which is exactly divisible by 32. When this statement is encountered, and the value in the Current Assignment Counter is not a block address, the Assembly Program will automatically increase the value to the next available block address.

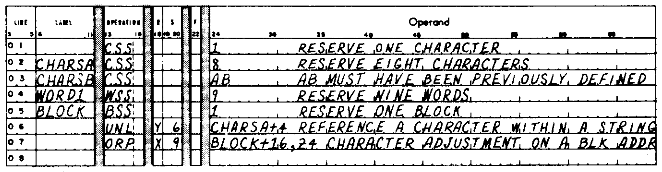
Label: This field may contain a symbolic address, which will reference the first word of the first block reserved. The programmer must use character adjustment to reference other words within the area reserved. For example, if a statement is used to reserve two blocks, the first word of the second block would be addressed by the label +32.

Operation: This field is used to specify the number of blocks to be reserved and must contain either a decimal number or a symbol.

Decimal Number: The actual number of blocks to be reserved is specified in decimal form.

Symbol: A symbol equivalent to the number of blocks to be reserved may be used. This symbol must not be character adjusted and must have been previously defined in the label field of a preceding statement.

The format of the CSS, WSS, and BSS operation codes is illustrated in Figure 10.



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	Operand 24 30 35 40 45 50 55 60 65
01		CSS				1 RESERVE ONE CHARACTER
02	CHARSA	CSS				8 RESERVE EIGHT CHARACTERS
03	CHARSB	CSS				AB AB MUST HAVE BEEN PREVIOUSLY DEFINED
04	WORD1	WSS				9 RESERVE NINE WORDS
05	BLOCK	BSS				1 RESERVE ONE BLOCK
06		UNL	Y	6		CHARSA+4 REFERENCE A CHARACTER WITHIN A STRING
07		ORP	X	9		BLOCK+16,24 CHARACTER ADJUSTMENT ON A BLK ADDR
08						

Figure 10.

DATA DEFINITION

DAC: Define Octal Character(s)

This operation is used to specify a string of up to ten constant characters as either octal numbers or symbols.

Label:	This field may contain a symbolic address which will be associated with the first character of the string. If more than one character is defined by the statement, succeeding characters may be referenced by using this label with appropriate character adjustment.
Operation:	DAC
Operand:	This field is used to specify the constant characters and may contain a combination of up to ten octal numbers or symbols. Each adjacent constant is separated by a comma, and no blank spaces are permitted between constants.

Octal Number: The maximum size of each octal number is 3777 (eleven binary bits). High-order zeros may be omitted.

Symbol: When the constant character is a symbolic address, the Assembly Program will truncate the five low-order bits of the symbol's equivalent sixteen-bit binary address. The remaining eleven high-order bits will be converted to an octal constant which is in effect a block address. Symbolic addresses must not be character adjusted.

DEC: Define Decimal Character(s)

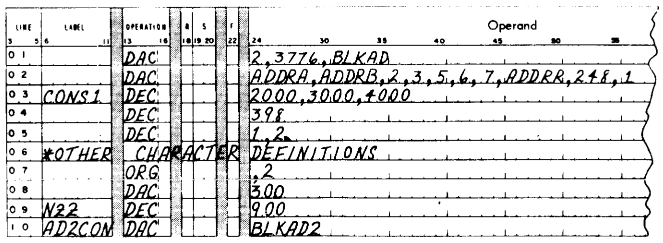
This operation enables the programmer to specify a string of up to ten constant characters as decimal numbers.

Label:	This field may contain a symbolic address which will be associated with the first character of the string. If more than one character is defined by the statement, succeeding characters may be referenced by using this label with appropriate character adjustment.
Operation:	DEC
Operand:	This field is used to specify up to ten decimal constant characters. The maximum size of each decimal number is 2047 (eleven binary bits). Each adjacent constant is separated by a comma, and no blank spaces are permitted between constants.

Restriction on the Use of CSS, DAC, and DEC Codes

If comments or any of the operations HED, SEQ, USQ, SPC, LST, or ULS are placed in the midst of the above statements, such an action must be followed by an "ORG ,2" statement, or previously defined data may be overlayed in 7750 core storage.

Figure 11 illustrates the format of the DAC and DEC operation codes:



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	Operand 24 30 35 40 45 50 55
01		DAC				2,3776,BLKAD
02		DAC				ADDRA,ADDRB,2,3,5,6,7,ADDRR,248,1
03	CONS1	DEC				2000,3000,4000
04		DEC				398
05		DEC				1,2
06	*OTHER	CHARACTER				DEFINITIONS
07		ORG				,2
08		DAC				300
09	N22	DEC				900
10	AD2CON	DAC				BLKAD2

Figure 11.

OCT: Define Octal Word(s)

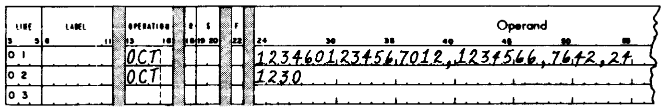
This operation is used to specify up to four sequential constant octal words.

Label:	This field may contain a symbolic address which will be associated with the first word in the sequence. Character adjustment must be provided by the programmer to reference any other words in the sequence.
Operation:	OCT

Operand: This field is used to specify up to four octal constant words. Each octal word may have a maximum of 16 digits.

The low-order digit must be a 0, 2, 4, or 6 because the Assembly Program determines the correct parity bit for the word. Each constant word must be separated from the next by a comma. High-order zeros may be omitted. No blank spaces are permitted.

Figure 12 illustrates the basic format of the OCT operation code:



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	Operand 24 30 38 40 48 50
0 1		OCT				1234601234567012,1234566,7642,24
0 2		OCT				1230
0 3						

Figure 12.

LWD: Define Limit Word

This operation is used to define a limit word.

Label: This field may contain a symbolic address which will be associated with the defined word.

Operation: LWD

Operand: This field is used to define the address, the limit, and a decimal constant, in exactly the same order and separated by commas. No blank spaces are permitted. The address or the limit may be either an actual decimal address or a symbolic address. Character adjustment is not permitted. The maximum size of the decimal constant is 2047.

CWD: Define Channel Word

Using this operation the programmer is able to direct the processor to assemble a Channel Word. This operation defines the address field and those portions of the control field that may logically be specified at assembly time.

Label: This field may be used to assign a symbolic designation to the Channel

Operation: CWD

Operand: This field may contain up to six operands. They must be specified in the following order and be separated by commas. Blanks must not be inserted within the operands.

Address: This may be an actual decimal address or a symbol. Character adjustment of the address is not permitted.

Character Length: This must be specified as a decimal number of one or two digits. In a regular Channel Word this value may range up to 11. In an Error Channel Word this value must range from 12 through 15.

Start-Stop/Synchronous: This is either a zero or a one. A zero indicates that transmission over the channel is Start-Stop. A one indicates that transmission is Synchronous.

Send/Receive: This is either a zero or a one. A zero indicates that the line is to be set to Receive status. A one indicates the line is to be set to Send status.

Hold/Not Hold: This bit is either a zero or a one. A zero indicates the channel is to be set in Hold status. A one indicates the channel is to be set in Not Hold status. When defining Channel Words that are to be loaded directly into Control Storage by the 7750 Load Program, this bit must be specified as zero.

Fractional Sampling Bit: If the characters transmitted over the channel controlled by this word have non-integer Stop bits, this bit must be specified as one. Otherwise it should be omitted.

Restriction on the Use of LWD and CWD Codes

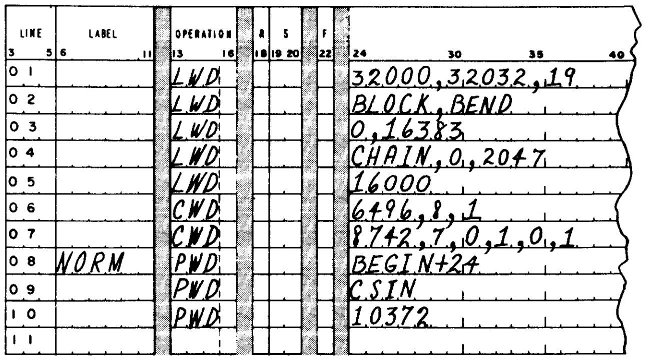
When interpreting the operands specified in these two statements, the Assembly Program is "position sensitive." That is, the order in which an operand is encountered determines how it is handled. Therefore, an operand within a group must not be omitted. It must be indicated with a zero. The only exception to this rule is that an operand or operands at the end of a group may be omitted if the programmer does not need to specify them.

PWD: Define Process Word

This operation enables the programmer to set the instruction counter portion of a Process Word. The Process Words that are defined by this operation are those associated with Normal Mode, Out Mode, Channel Service Mode, and Service Mode. The address specified in the operand field is the address of the first instruction that the programmer desires to execute in the specified mode.

Label:	This field may be used to assign a symbolic designation to the Process Word.
Operation:	PWD
Operand:	This field may contain an actual decimal address or a symbol. The symbolic address may be character adjusted. In either case, if the value specified is not a word address, the Assembly Program, in truncating the two low-order binary bits of the address, will generate a full word reference.

Figure 13 illustrates the basic format of the LWD, CWD, and PWD operation codes:



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	24 30 35 40
0 1		LWD				32000,32032,19
0 2		LWD				BLOCK,BEND
0 3		LWD				0,16383
0 4		LWD				CHAIN,0,2047
0 5		LWD				16000
0 6		CWD				6496,8,1
0 7		CWD				8742,7,0,1,0,1
0 8	NORM	PWD				BEGIN+24
0 9		PWD				CSIN
1 0		PWD				10372
1 1						

Figure 13

PROCESSOR CONTROL OPERATIONS

Control statements are, in effect, orders to the 1401 processor which give the programmer control over the assembly process. The 7750 Assembly Program using the 1401 has twelve control statements, and a detailed description of the use of these control statements is presented below. A complete list of Processor Control operations is illustrated in Appendix A.

CTL: CONTROL

A control statement must be the first entry (card) in a source program deck. The user must prepare this card to specify the storage size of the processing 1401, the storage size of the 7750, the type of output desired, and the presence or absence of the Punch Release Feature on the processing 1401.

Label: This field is ignored by the Assembly Program.

Operation: CTL

Operand: Column 24 - Size of 1401

Digit	Core Storage Size
1	4,000 positions
2	8,000 positions
3	12,000 positions
4	16,000 positions

Column 25 - Size of 7750 Process Storage

Digit	Core Storage Size
1	4,096 Words
2	8,192 Words
3	16,384 Words

Column 26 - Card or Tape Output

Digit	Type of output
1	Object program card deck.
2	Object program on tape.
3	Object program on cards and on tape.

Column 27 - Punch Release on 1401

blank	= Punch Release Feature is not present.
1	= Punch Release Feature is present.

Columns 29-69 - Comments, if desired.

Columns 76-80 - Program identification to be punched into every card of the object program deck.

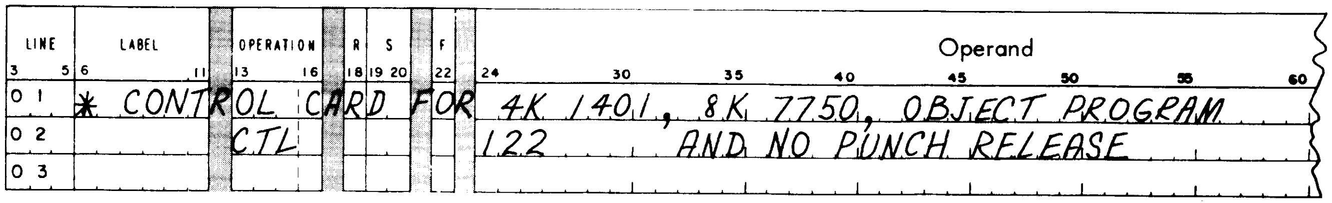
The basic format of the CTL statement is illustrated in Figure 14.

NOTE: If the CTL card is missing from the source deck, the processor will indicate an error condition and continue the assembly by assuming the minimum machine configuration for both the 1401 and the 7750. In such a case the processor will provide only card output.

CONTROL OF STORAGE ASSIGNMENT

An ORG or LOC statement is used whenever it is desired to alter the sequence in which storage assignments are made by the processor. At the beginning of a source program assembly, both the Current Assignment Counter and the High Assignment Counter are set at the first available word location in Process Storage following the area required by the 7750 Load Program. The actual address value used is 600_{10} which is equivalent to word address 226_8. If the first statement of the source program is not an ORG or a LOC statement, storage assignments will be made, in order, from this reference point with both assignment counters being updated until an ORG or LOC statement is encountered.

It should be noted that if the programmer uses an ORG or LOC statement that will result in the loading of information into Process Storage in the area occupied by the Load Program, an error will



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	Operand 24 30 35 40 45 50 55 60
0 1	* CONTROL CARD FOR					4K 1401, 8K 7750, OBJECT PROGRAM
0 2		CTL				122 AND NO PUNCH RELEASE
0 3						

Figure 14

be indicated on the assembly listing. However, if the programmer desires to load information into Control Storage using the pseudo-operations CWD or PWD, such specifications must be preceded by an ORG or LOC statement which will reset the Current Assignment Counter to a valid Control Storage address. For example, if the programmer wants to specify the Channel Service Mode Process Word, he must set the Current Assignment Counter to 252_{10}, the character address of the sixty-third word in Control Storage.

ORG: Origin

An Origin statement is used to specify the storage address at which the programmer desires the processor to begin assigning locations to instructions, constants, and work areas. An Origin statement also directs the processor to retain the highest value attained by the Current Assignment Counter in the High Assignment Counter.

Label:	This field may be used to assign a symbolic designation to the address placed in the Current Assignment Counter.
Operation:	ORG
Operand:	This field specifies the address which is to be placed in the Current Assignment Counter. The address must be in one of the following forms:

Decimal number: An actual storage location.

Symbol: The actual storage location equivalent of the symbol will be used. The symbol must have been previously defined and must not be character adjusted. The symbolic address may, however, be immediately followed by a comma and a decimal number from 1 through 15, which specifies the number of low-order zeros desired in the binary address. In this case, the value placed in the Current Assignment Counter will be the first address equal to or higher than the symbol's binary equivalent which has the specified number of zeros (1-15).

Comma and number: If column 24 contains a comma followed immediately by a decimal number in the range 1-15, the value in the Current Assignment Counter will be examined to determine whether or not its binary equivalent contains the specified number of low-order zeros. If it does not, the value in the Current Assignment Counter will be increased to the next higher address whose binary equivalent does contain the desired number of zeros.

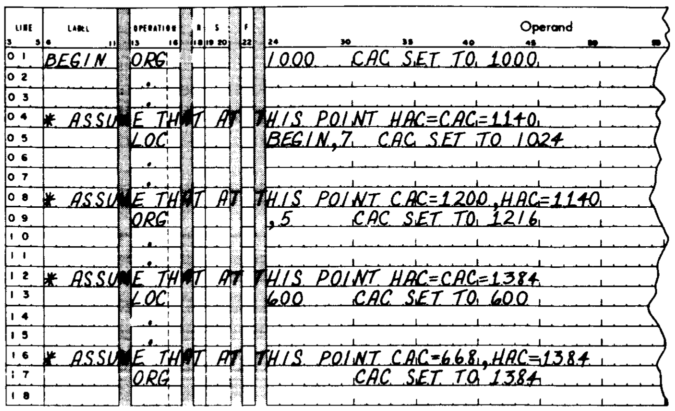
Blank: If column 24 is left blank, the present value of the High Assignment Counter will be inserted in the Current Assignment Counter.

LOC: Locate

The LOC operation is analogous to the ORG operation in that it enables the programmer to establish the setting of the Current Assignment Counter and to reference the setting of the High Assignment Counter. Unlike the ORG operation, however, storage assignments made subsequent to a LOC operation have no effect upon the High Assignment Counter. The LOC operation is particularly useful when writing programs or segments of programs that are to be subsequently overlayed.

Label:	This field may be used to assign a symbolic designation to the address placed in the Current Assignment Counter.
Operation:	LOC
Operand:	The entries that may be made in this field are exactly the same as those for an ORG statement.

The basic format of ORG and LOC statements and the effects of these statements upon the Current Assignment Counter (CAC) and the High Assignment Counter (HAC) are illustrated in Figure 15.



```
LINE  LABEL   OPERATION  R  S  F  Operand
3   5 6    11 13    16  18 19 20 22 24   30   35   40   45   50   55
01    BEGIN   ORG                  1000   CAC SET TO 1000
02              .
03              .
04    * ASSUME THAT AT THIS POINT HAC=CAC=1140
05            LOC                  BEGIN,7  CAC SET TO 1024
06              .
07              .
08    * ASSUME THAT AT THIS POINT CAC=1200,HAC=1140
09            ORG                  ,5       CAC SET TO 1216
10              .
11              .
12    * ASSUME THAT AT THIS POINT HAC=CAC=1384
13            LOC                  600      CAC SET TO 600
14              .
15              .
16    * ASSUME THAT AT THIS POINT CAC=668,HAC=1384
17            ORG                           CAC SET TO 1384
18
```

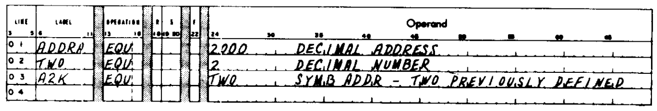
Figure 15

EQU: Equate

This operation enables the programmer to equate two symbolic addresses or a symbolic address and an actual storage address. This statement is used when it is necessary or desirable to reference a storage location by more than one label or when the programmer desires to assign an actual storage address to a label.

Label:	This field must contain the symbol whose equivalence is being defined. It must not be blank.
Operation:	EQU
Operand:	This field must contain either a symbolic address or an actual decimal address. It should not be blank. If a symbolic address is used, it must not be character adjusted and it must have been previously defined, i.e., it must have appeared in the label field of a preceding statement. An actual address may be any decimal number which is a valid machine address.

The basic format of the EQU statement is illustrated in Figure 16.



LINE	LABEL	OPERATION	R	S	F	Operand
01	ADDRA	EQU				2000 DECIMAL ADDRESS
02	TWO	EQU				2 DECIMAL NUMBER
03	A2K	EQU				TWO SYMB ADDR - TWO PREVIOUSLY DEFINED
04						

Figure 16

HED: Prefix Header

This operation directs the processor to associate a prefix code with symbolic addresses in the subsequent section of the program. The prefix is associated with all symbolic addresses in both the label and the operand fields that have five or fewer characters. Any six-character symbolic addresses will remain unaltered. It is possible to start, stop, and interrupt prefixing. Prefixes do not appear in the listing.

Start Prefixing

Label:	This field is ignored by the processor.
Operation:	HED
Operand:	A single alphabetic character, the prefix code, must be placed in column 24.

Stop Prefixing

Label:	This field is ignored.
Operation:	HED
Operand:	A blank character must be present in column 24. Symbolic addresses in the subsequent statements will remain unaltered.

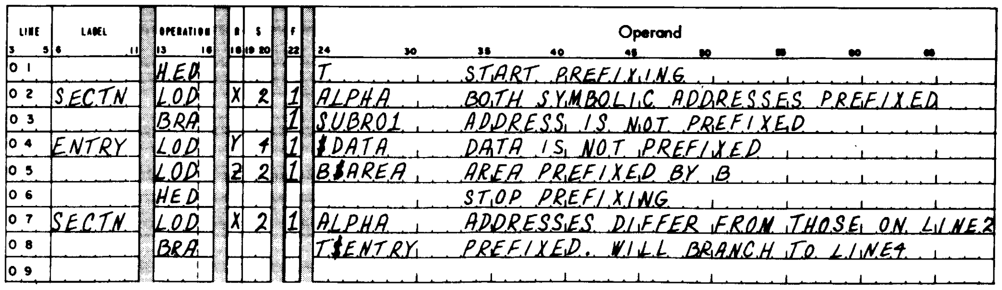
Interrupt Prefixing

If a section of a program is under HED control, it is possible to change the prefixing of a single program statement within the section. This option applies to symbolic address in the operand field only. To suspend prefixing of a symbolic address, a dollar sign ($) is placed immediately to the left of the symbolic address in the operand field. In order to alter the prefix character to be associated with a symbolic address, first the desired prefix code- a single alphabetic character- and then a dollar sign are placed immediately to the left of the symbolic address.

See the example for using prefix codes in Figure 17.

END: End

The last entry of the source program must be an END statement. This statement signals the processor that all the source program entries have



LINE	LABEL	OPERATION	R	S	F	Operand
01		HED				T START PREFIXING
02	SECTN	LOD	X	2	1	ALPHA BOTH SYMBOLIC ADDRESSES PREFIXED
03		BRA			1	SUBR01 ADDRESS IS NOT PREFIXED
04	ENTRY	LOD	Y	4	1	$DATA DATA IS NOT PREFIXED
05		LOD	2	2	1	B$AREA AREA PREFIXED BY B
06		HED				STOP PREFIXING
07	SECTN	LOD	X	2	1	ALPHA ADDRESSES DIFFER FROM THOSE ON LINE2
08		BRA				T$ENTRY PREFIXED. WILL BRANCH TO LINE4
09						

Figure 17

been read. It also causes the processor to generate a special transfer card. The transfer card is used by the 7750 Load Program to begin object program execution automatically after the entire program has been loaded into core storage. The Load Program accomplishes this by branching to the address specified in the operand field of the END statement.

Since the Load Program is executed entirely in In Mode, the transfer address specified by the programmer should be the address of the instruction in the object program which causes the 7750 to leave In Mode. The execution of this instruction will accomplish two things. First, it will cause the In Mode Process Word Instruction Counter to be reset to the first address of the In Mode program. Second, it will cause the 7750 to revert to Normal Mode. The Normal Mode Process Word Instruction Counter should be specified to contain the address of the first instruction the programmer wants executed in Normal Mode.

Label:	This field is ignored by the Assembly Program.
Operation:	END
Operand:	This field must contain either a symbolic address or an actual address. It must not be blank. A symbolic address must not be character adjusted and must have been previously defined. An actual address may be any decimal machine location.

EXE: Execute

During the loading of the object program it may be desirable to discontinue the loading process temporarily to execute the portion of the program just loaded. An EXE statement performs this function.

An EXE statement in the symbolic program causes the processor to generate a special transfer card. This card, when recognized by the 7750 Load Program, will cause it to halt the loading process and execute a branch instruction. The branch is made to the address that is in the operand portion of the EXE statement.

To continue the loading process after the desired portion of the program has been executed, the programmer must provide, as the last instruction of the portion executed, a branch back to the load routine.

Label:	This field is ignored by the Assembly Program.
Operation:	EXE
Operand:	This field must contain either a symbolic address or an actual address. It must not be blank. A symbolic address must not be character adjusted and must have been previously defined. An actual address may be any decimal machine location.

Figure 18 illustrates the basic format of the EXE and END statements .

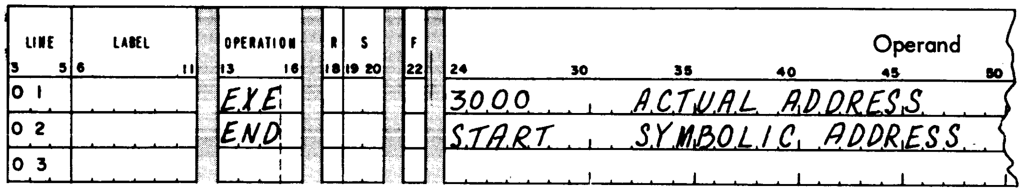
SEQUENCE CHECKING

Unless instructed otherwise, the Assembly Program will perform a sequence check on the page number and the line number fields (columns 1-5) of each input card. The 1401 collating sequence is used. While performing this operation, if a card is discovered to be out of sequence, an error will be indicated on the program listing. This error is meant to alert the programmer to the possibility that a card may have been misplaced in the source deck; however, a sequence error has no effect upon the processing of the input statement.

USQ: Suspend Sequence Checking

The processor automatically checks page and line numbers to determine whether the source cards are in sequential order. If the programmer desires the Assembly Program to suspend this checking operation, he should use the USQ operation.

Label:	This field is disregarded by the processor.



LINE 3 5	LABEL 6 11	OPERATION 13 16	R 18	S 19 20	F 22	Operand 24 30 35 40 45 50
0 1		EXE				3000 ACTUAL ADDRESS
0 2		END				START SYMBOLIC ADDRESS
0 3						

Figure 18

Operation: USQ

Operand: This field is ignored by the processor when processing an USQ entry. It may be used for comments.

SEQ: Resume Sequence Checking

If the Assembly Program had been instructed to suspend sequence checking of the page and line numbers, the operations may be resumed by inserting an SEQ entry. Checking will resume on the next card. It is not necessary to insert an SEQ entry at the beginning of the source program deck because the processor automatically begins sequence checking.

Label: This field is disregarded by the Assembly Program.

Operation: SEQ

Operand: The operand field is ignored by the processor. It may be used for comments.

CONTROL OF LISTING FORMAT

The processor automatically produces a combined listing of the source and object programs in single space format, skipping to a new page when the previous page is filled. It is possible to alter or interrupt this listing by using the operation SPC (Spacing Control), ULS (Suspend Listing), and LST (Resume Listing).

SPC: Spacing Control

The normal format of the listing may be altered by including SPC statements in the source program deck.

Label: The processor ignores this field.

Operation: SPC

Operand: One of the following entries must be made in column 24.

Digit	Action
0	Double space the listing from this point forward.
Blank	Resume single spacing from this point forward.
1	Skip to a new page at this point in the listing. Listing will then continue in the previous mode of spacing.

ULS: Suspend Listing

This operation enables the programmer to suspend the listing of any section of a program. When this operation is encountered in the source program the processor will discontinue the listing from that point forward.

Label: This field is ignored.

Operation: ULS

Operand: This entire field may be used for comments.

LST: Resume Listing

With this operation it is possible to resume the listing of a later section of the program following a section which is under the control of a ULS operation.

Label: This field is ignored.

Operation: LST

Operand: This entire field may be used for comments.

ASSEMBLY OUTPUT

The Assembly Program produces as output a program listing and the assembled program punched in cards or written on magnetic tape. The programmer instructs the Assembly Program as to whether the output is required on tape or in cards. If the output is desired on both, this can be obtained.

PROGRAM LISTING

The Assembly Program prints one (and sometimes more than one) line of output for each source statement in the program unless instructed otherwise. Each line of the listing is divided into nine fields. The names of these fields are listed at the top of each page. They are from left to right as follows: Locn, Field Defined Word, Actual Word, Pgln, Label, Op, RS, F, Operand. The first three fields are in octal notation and contain information about the assembled statement. The contents of each of these fields is discussed below. The last six fields duplicate the information on the source statement card and are self-explanatory.

Locn

This field contains the octal address of the location at which the assembled word will be positioned in 7750 storage. Although all actual addresses are specified to the Assembly Program as decimal numbers representing character locations, they are converted by the program to octal word addresses with character positions within the word suffixed if necessary.

For example, the decimal character address 8176 is equivalent to the octal word address 3774. This is also the octal address of the A character in that word. The B character of the word is indicated as octal 3774-1. Its equivalent decimal character address is 8177. The C and D characters of the word are indicated as octal 3774-2 and 3774-3. Their respective decimal addresses are 8178 and 8179.

For all imperative statements and data definitions of full words, the Locn field will contain only the octal word address. When listing character data definition statements, if the first character defined is not located in the A character of a word, the octal word address with the character position suffix will appear in the Locn field.

Field Defined Word

This field contains the contents of the assembled word. The contents of this assembled word are divided into subfields according to the type of word being used, and these subfields are listed in octal form. There are six possible word formats. Figure 19 illustrates them and identifies the various subfields in each format.

```
Instruction Word
    Address   R   S    M    L   F    Operation
    xxxxx-x   x   xx   x    xx  x     xxxx

BRT Instruction Word
    Address   Mask    M    F         Operation
    xxxxx-x   xxxx    x    x         0544

Data Word
    A     B     C     D
    xxxx  xxxx  xxxx  xxxx

Limit Word
    Address     Limit     D
    xxxxx-x     xxxxx-x   xxxx

Channel Word
    Address     Control
    xxxxx-x     xxxxxxx

Process Word
    Instruction Counter
    xxxxx
```

Figure 19

Actual Word

The Actual Word field contains the sixteen octal digits which represent the assembled word. These sixteen digits in cards or on tape are the output for subsequent loading into the 7750. Whereas the Field Defined Word represents only those portions of the word defined by the programmer, the Actual Word indicates the entire contents of the word including the parity bit. Since the Actual Word represents the entire forty eight bits of the binary word divided into sixteen sequential groups of three (each three bits represented by a single octal digit), it usually bears little resemblance to the Field Defined Word whose subfields are based upon the binary word divided into its logical groupings . The following example is offered as an illustration:

Source Statement	LOD* X09 1 22094, Z7
Field Defined Word	12623-2 1 11 3 07 1 3312
Actual Word	2544 7014 4335 6624

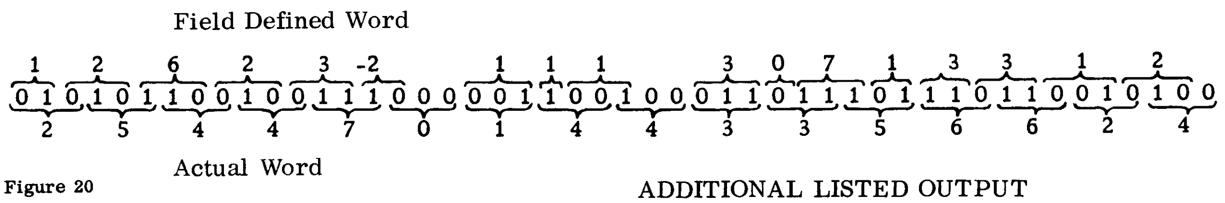


Figure 20

As a further illustration see Figure 20 and note how the two octal forms representing the word are derived from the binary word.

ERROR INDICATIONS IN PROGRAM LISTING

Whenever an error is detected anywhere in a source statement, a pound sign (#) will be printed on the listing to the left of the Pgln field. This is the general error indication. The Assembly Program will further attempt to pinpoint the error by placing a pound sign immediately to the left of the particular field or fields that are in error. The chart below lists the specific error being indicated when a pound sign is associated with a particular field.

Field	Error Indicated
Locn	An ORG or LOC statement has directed the processor to assemble words which when loaded will overlay the Load Program.
Pgln	A sequence error has been detected.
Label	Label contains an invalid character. Label is multi-defined. The CAC setting has exceeded maximum specified 7750 size. Storage assignments are made modulo the specified size.
Oper	Invalid operation code.
RS	RS specification necessary, but omitted. RS specified, but not valid for this operation. Invalid R specified. S specified exceeds maximum size of R.
F	Flag invalid in this operation.
Operand	Operand omitted. Invalid character(s) in operand. Multi-defined symbolic operand. Undefined symbolic operand. Too many terms in operand. Address adjustment specified, but invalid in this operation. ML specified, but invalid in this operation. Invalid M specified. L specified exceeds maximum size of M.

ADDITIONAL LISTED OUTPUT

Several different types of messages and error indications may be printed out prior to or during the actual program listing. The following is a list of messages and a brief description of their meaning. Messages 1,3,4,10, and 11 (indicated with an * sign) will always appear in the listing. The rest of the messages, i.e., 2,5,6,7,8, and 9 will appear only when the indicated error occurs.

*1. One of the following messages must appear in the listing:

4K 1401, 8K 7750, OUTPUT ON CARDS AND TAPE
<u>Explanation</u>: A message of this type lists the information obtained from the CTL card which is to be applied during this assembly run.

CORRECTED CONTROL CARD 4K 1401, 8K 7750, OUTPUT ON CARDS AND TAPE
<u>Explanation</u>: A message of this type indicates that one or more of the controls specified was invalid. The processor assumes the minimum case for such a specification and lists it accordingly.

NO CONTROL CARD ASSUME 4K 1401, 4K 7750, OUTPUT ON CARDS ONLY
<u>Explanation</u>: This message will appear if the first card of the source program deck is not the CTL card.

2. NO END CARD
<u>Explanation</u>: This message is printed if no END card is found in the source program deck. The program creates an END card by assuming a transfer address of 00000.

*3. INSTRUCTION COUNT xxxx
<u>Explanation</u>: This message indicates the total number of imperative statements in the source program.

*4. LABEL COUNT xxxx
<u>Explanation</u>: This message indicates the total number of labels that appear in the source program.

5. MULTI-DEFINED SYMBOL IN PPO (Image of statement in error)

This message is printed for either of these reasons.
Explanation: (1) This is produced if the processor discovers that a symbolic operand used in a principal pseudo-operation, i.e., CSS, WSS, BSS, ORG, LOC, EQU, EXE, END, has been defined more than once. All subsequent definitions of the symbol after the first are considered errors.
(2) If a multi-defined symbol is discovered in the PPO table, any principal pseudo-operation which uses this symbol as an operand will cause the printing of this message.

6. UNDEFINED SYMBOL IN PPO (Image of Statement in error)
Explanation: If a principal pseudo-operation with a symbolic operand is encountered and the operand's actual equivalent is not found in the PPO table, this error will be indicated.

7. 7750 OVERFLOW (Image of statement currently being processed)
Explanation: If the Current Assignment Counter should attain a value greater than the size of the object 7750, this message will be printed.

8. MULTI-DEFINED SYMBOL (Image of statement in error)
Explanation: Should more than one definition of a single symbol be encountered during processing, all but the first will be treated as errors and this message will be printed.

9. UNDEFINED SYMBOL (Image of statement in error)
Explanation: Should a statement contain a symbolic operand which has not been defined in the label field of another statement, this error will be indicated.

*10. LOWEST UNUSED 7750 Location Is xxxxx-x
Explanation: This message indicates the octal address of the first available character of storage in the object 7750 after the assembled program is loaded.

*11. AVAILABLE BLOCK STORAGE LIMITS xxxxx-xxxxx
Explanation: This message indicates the limits of block storage available for chaining. The first address is that of the first available block. The second address is that of the last block in the object 7750. This block will not be included in the chain set up by the Load Program since it may be needed in conjunction with the Action Delay Feature.

PUNCHED CARD OUTPUT

The Assembly Program will produce a deck of cards containing the assembled instructions unless the programmer specifies only tape output. This deck may be loaded into the 7750 through the computer. The eighty-column card is divided into five sixteen-character fields. Field I contains control information needed by the 7750 Load Program. Fields II, III, and IV contain the assembled words. Field V contains the program identification and a sequence number. Because a zero is transferred to the 7750 as an 8-bit and a 2-bit, all zeros in Fields I-IV are converted to blanks before a card is punched. A blank, when transferred to the 7750, produces the desired configuration - three binary zero bits.

The Control Field (Field I) contains in its first five positions the address of the location in 7750 storage into which the word in Field II is to be loaded. This address is based upon the octal word address listed in the Locn field for this statement. However, where the octal word address represents fourteen binary bits grouped 2,3,3,3,3, the address punched in the card assumes the addition of a fifteenth low-order binary bit, always zero, and a grouping 3,3, 3,3,3. For example, if the assembled word in Field II is to be loaded into octal location 05374, the address in the Control Field will be 12770. This adjustment is necessary because of the way the 7750 accepts information when it is being loaded.

Other characters in the Control Field indicate how many assembled words are punched in the card, whether they are to be loaded into Process Storage or Control Storage, or whether this is an END or EXE card.

Column 16 of the Control Field contains:

(a) the parity for the control word so that it may be loaded into the 7750 (bit 1 of the octal character); and

(b) the count (0-3) of the number of words in the card to be loaded into Process Storage (bits 2 and 4 of the octal character).

If the word count indicated in column 16 is zero, no words on the card are to be loaded into Process Storage. Column 12 should be examined next. This contains the count (0-3) of the number of words in the card to be loaded into Control Storage.

If both the Process Storage and the Control Storage word counts are zero, the card is either an END or EXE card. This distinction is indicated in column 10. If column 10 contains a zero, this is an END card. If column 10 contains a one, this is an EXE card.

Each card may contain up to three assembled words that are to be loaded into sequential words in either Process Storage or Control Storage. These words occupy Fields II, III, and IV. If a card contains less than three assembled words, each of the unused fields is filled with a blank word. A blank word consists of fifteen blank characters and a one in the sixteenth character. This enables the first four fields of every card to be loaded into the 7750 with no parity errors. An EXE card has three blank words in fields II, III, and IV. In an END card, field II is a limit word which contains the available block storage limits. This word is used by the 7750 Load Program to establish the chain of empty storage. Fields III and IV of an END card are blank words.

Field V is provided as a convenience for the programmer. The identification enables him to ascertain quickly the program's identity. The sequence numbers are provided for checking that the cards are in order and that none is missing.

MAGNETIC TAPE OUTPUT

Each eighty-character record written on tape contains exactly the same information as is punched in one card. The programmer may obtain card output or tape output, or both.

CHECKING THE SOURCE PROGRAM DECK

The programmer should make the following checks of the source program deck prior to the start of the assembly operations in order to ensure that no gross errors will hamper the assembly procedure:

1. The first card of the source program deck must be the CTL card containing the proper control information for this run. (See description of CTL card on p. 11.) If CTL card is not the first card of the source program deck, the processor will assume the minimum machine configuration for both the 1401 and the 7750 and produce only the card output.
2. If the programmer wants the processor to begin making storage assignments at a location other than 600_{10} (word address 226_8), the statement immediately following the CTL card must be an ORG or LOC specifying the desired beginning address. (See pp. 11-12.)
3. If the assembling 1401 contains no more than 4,000 positions of storage, the total number of unique symbolic operands used in the following operations - CSS, WSS, BSS, ORG, LOC, EQU, EXE, END - must not exceed twenty.
4. Care must be taken to ensure that no symbolic operand is undefined or multi-defined.
5. The last card of the source program deck must be an END statement card. The address in the END statement must be the address of the instruction in the object program which causes the 7750 to leave In mode. (See pp. 13-14.)

OPERATING PROCEDURE

When all of the above listed checks have been made, the following steps may be taken to start the actual assembly operation.

1. Mount the 7750/1401 Assembly Program system tape on tape unit 1 and place the unit in ready status. This tape must be rewound before starting.
2. Mount the two work tapes on tape units 2 and 3 and place the units in ready status. These tapes will be rewound by the program, if necessary.
3. Place the source program deck in the card reader.
4. Put Sense Switch A and the I/O Check Stop Switch on (up).
5. Reset the machine by pressing the Start Reset button.
6. Press the Tape Load button.
7. When the assembly process is completed, the machine will stop at a programmed halt (operation code - BA821) with the value 0770 in the B Address Register.

PROCESSING ERROR INDICATIONS AND HALTS

Control Card Errors

An error in control card specification.
<u>Explanation</u>: If the CTL card specifies in column 24 a processor machine size which is greater than that of the machine being used, the program will hang up when trying to use the additional storage.
<u>Action</u>: Correct the CTL card and restart the program.

Card Reader Errors

READER ERROR. READY DECK AND PUSH START
<u>Explanation</u>: If an error in Card Reader is sensed, this message will be printed, and the program will halt with a value of 0110 in the B Address Register.
<u>Action</u>: Set I/O Check Stop Switch on, re-ready the source program deck, and press START.

NO CARDS IN READER
<u>Explanation</u>: If the last card is sensed at the beginning of the run, this message will be printed. The program will halt and the B Address Register will have the value 0115.
<u>Action</u>: Put the cards in proper order, ready the deck, and press START.

Tape Errors

TAPE READ REDUNDANCY
<u>Explanation</u>: Read Redundancy on System Tape - If a read redundancy occurs during the loading of any pass of the processor from tape unit 1, the load program will halt after back-spacing the tape. Pressing the I Address Register key will display the value 0007 in the Storage Address Lights.
<u>Action</u>: Press the Start key so that the loader may reread the record.

PERMANENT READ ERROR (Image of the record in error)
<u>Explanation</u>: Read Redundancy on Work Tape - In case the read redundancy occurs from tape unit 2 or 3, the processor will attempt to read the tape an additional ten times. If the read is still unsuccessful, the program will backspace and reread three records. If the error still persists, this message will be printed, and the processor will then continue further operations as if the information had been correctly read.

Tape Write Redundancy

WRITE ERROR - 10 ERASES UNSUCCESSFUL

SHORT WRITE ERROR - 10 ERASES UNSUCCESSFUL

Explanation: The first message will be written when invalid information has been written. The second message will be written when a short length record has been written. In the event of a write redundancy on tape unit 2 or 3, the processor will attempt the following procedure 10 times - backspace and rewrite; if the error persists, backspace, skip and erase tape, and rewrite. If the record is still unwritten, one of the above messages will be printed and the program will halt. Pressing the B Address Register key will display one of the following values in the Storage Address Lights:

Value	Tape Unit
0160	2 or 3
0662	2
0763	3

Action: Press the Start key at this point, in order to cause the processor to make 10 more attempts to write the record. If the error still persists, change the tape reel, and begin the operation again.

OUTPUT TAPE END OF FILE

Explanation: End of File on Work Tapes - If end of file is reached while the processor is writing tape, this message will be printed. The program will halt and the B Address Register will display one of the following values:

Value	Tape Unit
0130	2 or 3
0632	2
0733	3

Action: Replace the short tape reel and begin again.

APPENDIX A

LIST OF 7750 MNEMONIC CODES

Following is a list of the imperative operation codes (with their machine-language equivalents), declarative operation codes, and processor control operation statements that are used in the 7750 Assembly Program.

IMPERATIVE OPERATIONS

TYPE	MNEMONIC OP CODE	DESCRIPTION	MACHINE CODE
Character	LOD	Load Character	1302
	LOD*	Load Character Indirect	1312
	LOI	Load Character and Increment	1332
	LOI*	Load Character Indirect and Increment	1332
	LDC	Load Complemented Character	1342
	LDC*	Load Complemented Character Indirect	1352
	LCI	Load Complemented Character and Increment	1362
	LCI*	Load Complemented Character Indirect and Increment	1372
		NOTE: Any of the above listed Load operations referencing the Channel Service Register (C) will always reset the four high-order bits.	
	UNL	Unload Character	1202
	UNL*	Unload Character Indirect	1212
	ULI*	Unload Character Indirect and Increment	1232
	ULC	Unload Complemented Character	1242
	ULC*	Unload Complemented Character Indirect	1252
	UCI*	Unload Complemented Character Indirect and Increment	1272
	XOR	Exclusive Or	1101
	XOR*	Exclusive Or Indirect	1111
	XOI	Exclusive Or and Increment	1121
	XOI*	Exclusive Or Indirect and Increment	1131
	XOC	Exclusive Or Complement	1141
	XOC*	Exclusive Or Complement Indirect	1151
	XCI	Exclusive Or Complement and Increment	1161
	XCI*	Exclusive Or Complement Indirect and Increment	1171
		NOTE: All of the above listed Exclusive Or operations must reference the X Register.	
	IOR	Inclusive Or	1102
	IOR*	Inclusive Or Indirect	1112
	IOI	Inclusive Or and Increment	1122
	IOI*	Inclusive Or Indirect and Increment	1132
	IOC	Inclusive Or Complement	1142
	IOC*	Inclusive Or Complement Indirect	1152
	ICI	Inclusive Or Complement and Increment	1162
	ICI*	Inclusive Or Complement Indirect and Increment	1172
		NOTE: Inclusive Or Operations listed above must not reference the Channel Service Register (C).	

IMPERATIVE OPERATIONS

TYPE	MNEMONIC OP CODE	DESCRIPTION	MACHINE CODE
	ORP	Or to Process Storage	1002
	ORP*	Or to Process Storage Indirect	1012
	ORI*	Or to Process Storage Indirect and Increment	1032
	OCP	Or Complement to Process Storage	1042
	OCP*	Or Complement to Process Storage Indirect	1052
	OCI*	Or Complement to Process Storage Indirect and Increment	1072
	AND	And	1145
	AND*	And Indirect	1155
	ANI	And and Increment	1165
	ANI*	And Indirect and Increment	1175
	ANC	And Complement	1105
	ANC*	And Complement Indirect	1115
	ACI	And Complement and Increment	1125
	ACI*	And Complement Indirect and Increment	1135
		NOTE: And operations listed above must not reference the Channel Service Register (C) or the Interface Data Register (D).	
Address Limit	GTA	Get Address	0702
	GTA*	Get Address Indirect	0712
	GTL	Get Limit	0302
	GTL*	Get Limit Indirect	0312
	GOA	Get and Or Address	0502
	GOA*	Get and Or Address Indirect	0512
	GOL	Get and Or Limit	0102
	GOL*	Get and Or Limit Indirect	0112
	PTA	Put Address	0602
	PTA*	Put Address Indirect	0612
	PTL	Put Limit	0202
	PTL*	Put Limit Indirect	0212
Storage-to-Storage	TAP	Transmit Address to Process Storage	0606
	TAP*	Transmit Address to Process Storage Indirect	0616
	TAC	Transmit Address to Control Storage	0706
	TAC*	Transmit Address to Control Storage Indirect	0716
	TOC	Transmit and Or Address to Control Storage	0506
	TOC*	Transmit and Or Address to Control Storage Indirect	0516
	MWP	Move Word to Process Storage	0206
	MWP*	Move Word to Process Storage Indirect	0216
	MWC	Move Word to Control Storage	0306
	MWC*	Move Word to Control Storage Indirect	0316
	MOC	Move Word and Or to Control Storage	0106
	MOC*	Move Word and Or to Control Storage Indirect	0116
Control	BRA	Branch	0707
	BRA*	Branch Indirect	0717
	BRZ	Branch on Zero	0704
	BRZ*	Branch on Zero Indirect	0714
	BRO	Branch on Ones	0744
	BRO*	Branch on Ones Indirect	0754
	BRT	Branch on Test	0544
	CAL	Compare Address to Limit	0003
	CAI	Compare Address to Limit and Increment	0023
		NOTE: With the single exception of BRA*, none of the control type imperative operations listed above may be address modified.	

DECLARATIVE OPERATIONS

TYPE	MNEMONIC OP CODE	DESCRIPTION
Storage Reservation	CSS	Reserve Character(s)
	WSS	Reserve Word(s)
	BSS	Reserve Block(s)
Data Definition	DAC	Define Octal Character(s)
	DEC	Define Decimal Character(s)
	OCT	Define Octal Word(s)
	LWD	Define Limit Word
	CWD	Define Channel Word
	PWD	Define Process Word

PROCESSOR CONTROL OPERATIONS

MNEMONIC OP CODE	DESCRIPTION
CTL	Control
ORG	Origin
LOC	Locate
EQU	Equate
HED	Prefix Header
EXE	Execute
END	End
USQ	Suspend Sequence Checking
SEQ	Resume Sequence Checking
ULS	Suspend Listing
LST	Resume Listing
SPC	Spacing Control

NOTE: When assembling on a 1401 that has 4,000 positions of core storage, a restriction must be observed in the use of symbolic operands in the following pseudo-operations: CSS, WSS, BSS, ORG, LOC, EQU, EXE, END. For these operations, the total number of unique symbolic operands used must not exceed twenty.

C28-6259

Printed in U.S.A. C28-6259

IBM

International Business Machines Corporation

Data Processing Division

112 East Post Road, White Plains, New York