CGTM # 89

IBM 360 - PDP9 ASSEMBLER MANUAL

by

Michael J.C. Hu

March 1970

TABLE OF CONTENTS

I. GENERAL INFORMATION

The assembler accepts source program on IBM card images and produces an object program on magnetic tape, a listing of the assembled program, a cross reference table of all the symbols used in the program, and a formatted tape dump.

The input card format is given on page 4. The assembled program is written without a label at the beginning, and the recording density is 800 bpi.

The instruction consists of all the PDP-9 instructions, OPERATION, EAE, API, TELEPYPE, READER/PUNCH, and the entire set of instructions for controlling the SLAC spiral reader. However, other instructions could be added into the assembler with no difficulties.

The assembler uses 182,982 bytes of core space out of a partition of 300K. Out of 182,982 bytes of core, 102,400 bytes are reserved for symbols and cross-reference tables. The assembler can store over 2000 symbols. This assembler was completed in December 1967, and was extensively used for developing the Spiral Reader control programs.

A word of caution, - - - - at the time when this assembler was being developed, there was no ADVANCED PDP9 ASSEMBLER available. Consequently, some parts of the assembler language are my own and, unfortunately, they are quite incompatible with that of PDP9's.

II. LANGUAGE DEFINITIONS

i) CHARACTER

The following characters set is used:

The alphabet: = letters A through Z

The numerals: = numbers 0 through 9

The special characters: = ' (+ - , * / space ¢

ii) SYMBOL

A symbol is any arrangement of letters and numbers which starts with a letter and contains no more than six total characters.

iii) CONSTANTS

Constants may be integer (decimal or octal), symbolic, or literal.

A. A decimal number is any arrangement of six or less decimal digits from $-(2^{17}-1)$ to $(2^{17}-1)$, appended with the letter D.

Ex. 10D
-87D
100001D

B. An octal number is any arrangement of six or less octal digits.

Ex. 10
777777

C. A symbolic constant meets the specifications for a symbol but is equated to a constant or to the sum or difference of two symbols address locations.

```
Ex.   ZIP  .EQU  70112⁴
      LOOP .EQU  LOOP1+3
      PACK .EQU  Y7-X1
```

D. A literal constant is denoted by special character "(" followed by a number. The number can be either in decimal or octal.

```
Ex.   (7
      (707710
      (-5
      (9D
```

iv) OPERATORS

Operation between two symbols or constants can be either + or -.

v) SEPARATORS

Separators are used to indicate the end of distinct entities of an instruction. The characters used are: , and space.

vi) OPERANDS

Operands are combinations of symbols. The operators + and -, and constants. The acceptable forms are:

Symbol

Symbol ± I

Symbol ± Symbol

± I

where I is an integer

vii) FIELDS

An instruction is a combination of the following fields:

LABEL = Provides a symbol for referencing by other instructions.

OPCODE = Defines the operation portion of the instruction.

ADDRESS= Supplies the instruction with appropriate operands.

COMMENTS = Programmer notes only. This field has no effect on the assembly process.

III. INSTRUCTION FORMAT

i) FORMAT

The instruction format used in this assembler is fixed format.

1 6	7	8 14	15	16 30	31	32 80
LABEL		OPCODE		ADDRESS		COMMENT

The Label field is a fixed length and occupies columns 1 through 6.

The Opcode field is of two types:

A. Opcodes requiring an operand: The Opcode field is fixed in length of 4 characters and occupies columns 8 through 12.

B. Opcodes not requiring an operand: The Opcode field is a variable length starting in Column 8. The terminating character must be a space. The maximum extension of the field is to column 30.

The Address field is variable length and it starts in column 16. The maximum extension of the field is to column 30 and it must be terminated by a blank, so the fields must not contain any imbedded blanks.

The Comment field occupies columns 32 through 80. Any card with special characters * or / in column 1 is considered a comment card. Its contents will be printed in the assembly listing.

ii) FIELDS

A. Label Field

The location field may be blank or contain a symbol starting in column 1. An * or / in column 1 defines a comment card.

a) Any given symbol may appear in the location field only once within any assembly.

B. Opcode Field

The opcode field may be blank or contain any of the following items.

a) The PDP9 mnemonic code as given Appendix 1. Operate instructions may be combined using a , as separator and a blank to terminate the field.

b) Assembler pseudo-ops as given in Section IV.

c) Special operation defined by the programmer. See Appendix 1

C. Address Field

The contents of the address field varies with the instruction.

a) Memory Address Instructions

The address field contains an operand.

```
Ex.  LAC  A
     TAD  B
     DAC  C
     JMP  *-4
```

b) Operation Instructions

No address field. The opcode field may extend beyond column 16.

```
Ex.  SNL,CLL,RAR
     SZA
     HLT
```

c) LAW Instruction

Address field may contain a symbol or constant.

```
Ex.  LAW  -9D
     LAW  LOOP+2
```

d) IOT Instructions

Special I/O instructions. No address field.

e) EAE Instructions

No address field with the exception of shift instructions which require a shift count in the address field.

```
Ex.  CLQ
     LRS  9D
     ALS  1
```

f) .ASC Pseudo Op

The address field contains the text to be converted from IBM EBCDIC to PDP9 ASCII code. All message must be terminated with ¢ character. Maximum length is 20 characters.

```
Ex.  NOTE   .ASC    THIS IS A TELETYPE ¢
            .ASC    MESSAGE #¢
```

g) Blank Opcode Field

The address field may contain an operand as follows:

1. For transmission of subroutine arguments

```
Ex.  JMS  COMP
          X
          Y
```

2. Following an EAE multiply or divide instruction.

```
Ex.   IDIV
          10D
```

A blank opcode field with an operand in the address field is also a convenient method for defining constants.

```
ex.  TEN          10D
     NEG1         777777
     LOCT         T
     INST         042000
```

iii) Indirect Addressing

All memory address instructions may be indirectly addressed. Indirect addressing of an instruction is affected by placing an * immediately following the opcode for the instruction. A blank terminates the opcode field so there must be no imbedded blanks within the field.

```
Ex.    LAC*      10
       DAC*      L+1
       JMP*      *-3
```

IV. ASSEMBLER PSEUDO OPERATIONS

i) Assembler Pseudo Op List

The following are the pseudo ops for the assembler:

.ORG
.BLK
.EXT
.ENT
.EQU
.EJT
.SPC
.FIX
.REL
.END
.ASC

Note all pseudo op instruction begins with character " . " followed by a three character function name.

ii) Pseudo Ops Functional Descriptions

A. Pseudo Ops Affecting the Location Counter

a) Location Counter Initialization Pseudo Op: .ORG

.ORG sets the location counter absolutely to the contents of the address field. If no address field is present, .ORG Ø is assumed.

b) Memory Reservation Pseudo Op: .BLK

.BLK reserves a block of memory whose length is equal to the contents of the address field.

Ex. SPACE .BLK 25D

.BLK psuedo op will reserve 25 locations of memory space starting at location SPACE.

c) Text Generation Pseudo Op: .ASC

.ASC converts IBM EBCDIC to PDP9 ASCII code. Every five characters will be converted and packed into 2 PDP-9 words (36 bits). The maximum character per text statement is 20, and each text statement must be terminated

with a ¢ character.

Ex. TEXT .ASC THIS IS A TELETYPE M¢
.ASC ESSAGE #¢

d) External Symbol Definition Pseudo Op: .EXT

.EXT is used to provide the assembler with the information that the symbol in the address field of the operation, defined as being external, is an entry point in some other assembly that is to be referenced from the assembly containing the .EXT operation. In order to reference symbols defined as being external indirect addressing must be used. The assembler allocates a location to each external symbol in each assembly in which it occurs and it is this location which is referenced indirectly. The loader will place the actual address of the symbol in all locations assigned to it as an external symbol.

Ex. .EXT BEGIN
JMS* BEGIN

B. Pseudo Ops Not Affecting the Location Counter

a) Entry Point Definition Pseudo Op: .ENT

.ENT defines the contents of the address field as an entry point to the program to provide a means of reference from another assembly.

Ex. .ENT BEGIN

Note the address field must contain a symbol.

b) Equivalence Pseudo Op: .EQU

.EQU gives the label field symbol the same value as the address field expression.

Ex. 1. To define an operation to clear an external statue.register:

CXSR .EQU 7Ø524Ø

That is whenever operation CXSR is referenced in the program, the operation code 7Ø524Ø will be inserted into the instruction.

2. To equate symbols:

```
P   .EQU   Q
```

References to P or Q will refer to the same memory address.

3. To equate a symbol to a particular memory address:

```
LOOP   .EQU   1ØØØ
```

References to LOOP will refer to the memory location 1ØØØ.

c) Page Eject Pseudo Op: .EJT

.EJT results a page ejection during printing of the assembly listing, causing the next instruction following the .EJT operation to be printed at the top of a new page. The .EJT card will not be printed in the listing.

d) Space Pseudo Op: .SPC

.SPC results a line space during printing of the assembly listing, causing the next instruction following the .SPC operation to be printed on the next line after the line space operation. The .SPC card will not be printed in the listing.

e) End Pseudo Op: .END

.END is used to signal the end of an assembly. It must be the last card in the deck.

C. Pseudo Ops Determing the Mode of the Assembler Output

a) Fixed Mode Assembly Output: .FIX

.FIX produces non-relocatable assembly output. The .FIX operation, if present, must be the first card in the symbolic deck. The location counter is set to 200 but may be changed by an .ORG pseudo op.

b) Relocatable Mode Assember Output: .REL

.REL produces relocatable assembly output. The location counter is set to ∅ but may be reset with an ORG pseudo op. The .REL operation, if present, must be the first card in the symbolic deck.

If neither a .FIX nor .REL instruction is read, the output will be in relocatable mode.

V. ASSEMBLER ERROR CODES

The assembler will print out any error it found during the assembly of a program. The following is a list of the codes used:

S = ILLEGAL SYMBOL occurs whenever the first character in the label field is a number, or a blank

M = MULTIPLE DEFINITION occurs whenever the same symbol occurs twice in the label field.

I = ILLEGAL INSTRUCTION occurs whenever the instruction in the opcode field is not defined in the assembler's instruction set.

P = ILLEGAL PSEUDO OP occurs whenever the operation in the opcode field is not defined in the assembler's pseudo-opcode set.

O = ILLEGAL OPERAND occurs whenever the expression in the address field is incorrectly expressed.

U = UNDEFINED SYMBOL occurs whenever the symbol in the address field is never defined anywhere in the label field.

VI. SPECIAL SYMBOLS AND THEIR MEANINGS

The assembler uses a certain number of characters to perform internal operations:

* = When used as the first character in the label field, the assembler will treat the rest of the information as comment.
When used as the last character after an instruction code in the opcode field, the assembler will modify the instruction code to be indirect addressing.
When used as the first character in the address field, the assembler will use the content of the program counter as a reference to compute the address.
When used elsewhere, the assembler will treat it like any other characters.

/ = When used as the first character in the label field, the assembler will treat the rest of the information as comment.
When used elsewhere, the assembler will treat it like any other characters.

. = When used as the first character in the opcode field, the assembler will treat the rest of the information in the opcode field as pseudo op. Elsewhere this character will be treated by the assembler like any other characters.

, = When used after an instruction in the opcode field, the assembler will OR the instruction code to the next instruction code of the instruction following the ,. The assembler will keep on OR'ing instructions together in the opcode field, as long as those instructions are separated by ,. Elsewhere this character will be treated by the assembler like any other character.

(= When used as the first character in the address field, the assembler will treat numbers following it as a literal. Elsewhere this character will be treated by the assembler like any other character.

\+ = When used in the address field, the assembler will perform an ADD operation. Elsewhere this character has no effect on the assembler's operation.

- = When used in the address field, the assembler will perform an SUBTRACT operation. Elsewhere this character has no effect on the assembler's operation.

D = When used as the last character in a numerical string in the address field, the assembler will treat the numerical string as a decimal quantity, and will proceed to convert the decimal quanity into an octal quantity. Elsewhere this character will be treated by the assembler like any other character.

¢ = When used in the address field of a .ASC pseudo op, the assembler will stop looking for text information to convert. Otherwise, this character will be treated like any other characters by the assembler.

= The blank character when used in the label, opcode, or address field will be interpreted by assembler as a termination character. Blank characters in comment field has no effect on the assembler.

APPENDIX 1

INSTRUCTION SUMMARY

MEMORY REFERENCE INSTRUCTIONS

Mnemonic Symbol	Octal Code	Machine Cycles	Operation Executed
CAL	00	2	Call subroutine. The address portion of this instruction is ignored. The action is identical to JMS 20.
DAC Y	04	2	Deposit AC. The content of the AC is deposited in the memory cell at location Y.
JMS Y	10	2	Jump to subroutine. The content of the PC and the content of the L are deposited in memory cell Y. The next instruction is taken from cell Y + 1.
DZM Y	14	2	Deposit zero in memory. Zero is deposited in memory cell Y.
LAC Y	20	2	Load AC. The content of Y is loaded into the AC.
XOR Y	24	2	Exclusive OR. The exclusive OR is performed between the content of Y and the content of the AC, with the result left in the AC.
ADD Y	30	2	Add (1's complement). The content of Y is added to the content of the AC in 1's complement arithmetic and the result is left in the AC.
TAD Y	34	2	Two's complement add. The content of Y is added to the content of the AC in 2's complement arithmetic and the result is left in the AC.
XCT Y	40	1+	Execute. The instruction in memory cell Y is executed.
ISZ Y	44	2	Increment and skip if zero. The content of Y is incremented by one in 2's complement arithmetic. If the result is zero, the next instruction is skipped.
AND Y	50	2	AND. The logical operation AND is performed between the content of Y and the content of the AC with the result left in the AC.
SAD Y	54	2	Skip if AC is different from Y. The content of Y is compared with the content of the AC. If the numbers are different, the next instruction is skipped.
JMP Y	60	1	Jump to Y. The next instruction to be executed is taken from memory cell Y.

OPERATE INSTRUCTIONS

Mnemonic Symbol	Octal Code	Event Time	Operation Executed
~~OPR 0~~ NOP	740000	---	Operate group or no operation. Causes a 1-cycle program delay.
CMA	740001	3	Complement accumulator. Each bit of the AC is complemented.
CML	740002	3	Complement link.
OAS	740004	3	Inclusive OR ACCUMULATOR switches. The word set into the ACCUMULATOR switches is OR combined with the content of the AC, the result remains in the AC.
RAL	740010	3	Rotate accumulator left. The content of the AC and L are rotated one position to the left.
RAR	740020	2	Rotate accumulator right. The content of the AC and L are rotated one position to the right.
HLT	740040	---	Halt. The program is stopped at the conclusion of the cycle.
SMA	740100	1	Skip on minus accumulator. If the content of the AC is negative (2's complement) number the next instruction is skipped.
SZA	740200	1	Skip on zero accumulator. If the content of the AC equals zero (2's complement), the next instruction is skipped.
SNL	740400	1	Skip on non-zero link. If the L contains a 1, the next instruction is skipped.
SKP	741000	1	Skip. The next instruction is unconditionally skipped.
SPA	741100	1	Skip on positive accumulator. If the content of the AC is zero (2's complement) or a positive number, the next instruction is skipped.
SNA	741200	1	Skip on non-zero accumulator. If the content of the AC is not zero (2's complement), the next instruction is skipped.
SZL	741400	1	Skip on zero link. If the L contains a 0, the next instruction is skipped.
RTL	742010	2,3	Rotate two left. The content of the AC and the L are rotated two positions to the left.
RTR	742020	2,3	Rotate two right. The content of the AC and the L are rotated two positions to the right.

OPERATE INSTRUCTIONS (continued)

Mnemonic Symbol	Octal Code	Event Time	Operation Executed
CLL	744000	2	Clear link. The L is cleared.
STL	744002	2,3	Set link. The L is set to 1.
RCL	744010	2,3	Clear link, then rotate left. The L is cleared, then the L and AC are rotated one position left.
RCR	744020	2,3	Clear link, then rotate right. The L is cleared, then the L and AC are rotated one position right.
CLA	750000	2	Clear accumulator. Each bit of the AC is cleared.
CLC	750001	2,3	Clear and complement accumulator. Each bit of the AC is set to contain a 1.
LAS	750004	2,3	Load accumulator from switches. The word set into the ACCUMULATOR switches is loaded into the AC.
GLK	750010	2,3	Get link. The content of L is set into AC17.
LAW N	76XXXX	---	Load the AC with 76XXXX.

EAE INSTRUCTION LIST

Mnemonic Symbol	Octal Code	Operation Executed
EAE	640000	Basic EAE command. No operation.
LRS	640500	Long right shift.
LRSS	660500	Long right shift, signed (AC sign = link).
LLS	640600	Long left shift.
LLSS	660600	Long left shift, signed (AC sign = L).
ALS	640700	Accumulator left shift.
ALSS	660700	Accumulator left shift, signed (AC sign = L).
NRM ~~NORM~~	640444	Normalize, unsigned. Maximum shift is 44_8.
NRMS ~~NORMS~~	660444	Normalize, signed (AC sign = L).
MUL	653122	Multiply, unsigned. The number in the AC is multiplied by the number in the next core memory address.
MULS	657122	Multiply, signed. The number in the AC is multiplied by the number in the next core memory address.
DIV	640323	Divide, unsigned. The 36-bit content of both the AC and MQ is divided by the number in the next core memory location.
DIVS	644323	Divide, signed. The content of both the AC and MQ as a 1's complement signed number is divided by the number in the next core memory location.
IDIV	653323	Integer divide, unsigned. Divide the number in the AC as an 18-bit unsigned integer by the number in the next core memory location.
IDIVS	657323	Integer divide, signed. Same as IDIV but the content of the AC is a 17-bit signed number.
FDV ~~FRDIV~~	650323	Fraction divide, unsigned. Divide the 18-bit fraction in the AC by the 18-bit fraction in the number in the next core memory location.
FDVS ~~FRDIVS~~	654323	Fraction divide, signed. Same as FRDIV, but the content of the AC is a 17-bit signed number.
LACQ	641002	Replace the content of the AC with the content of the MQ.
LACS	641001	Replace the content of the AC with the content of the SC.

EAE INSTRUCTION LIST (continued)

Mnemonic Symbol	Octal Code	Operation Executed
CLQ	650000	Clear MQ.
ABS	644000	Place absolute value of AC in the AC.
GSM	664000	Get sign and magnitude. Places AC sign in the link and takes the absolute value of AC.
OSC	640001	Inclusive OR the SC into the AC.
OMQ	640002	Inclusive OR AC with MQ and place results in AC.
CMQ	640004	Complement the MQ.
LMQ	652000	Load MQ.

INPUT/OUTPUT TRANSFER INSTRUCTIONS

Mnemonic Symbol	Octal Code	Operation Executed
		Program Interrupt
IOF	700002	Interrupt off. Disable the PIC.
ION	700042	Interrupt on. Enable the PIC.
		Real Time Clock
CLSF	700001	Skip the next instruction if the clock flag is set to 1.
CLOF	700004	Clear the clock flag and disable the clock.
CLON	700044	Clear the clock flag and enable the clock.
		Perforated Tape Reader
RSF	700101	Skip if reader is a 1.
RCF	700102	Clear reader flag, then inclusively OR the content of reader buffer into the AC.
RRB	700112	Read reader buffer. Clear reader flag and AC, and then transfer content of reader buffer into AC.
RSA	700104	Select reader in alphanumeric mode. One 8-bit character is read into the reader buffer.
RSB	700144	Select reader in binary mode. Three 6-bit characters are read into the reader buffer.
		Perforated Tape Punch
PSF	700201	Skip if the punch flag is set to 1.
PCF	700202	Clear the punch flag.
PSA or ~~PLS~~	700204 ~~700206~~	Punch a line of tape in alphanumeric mode.
PSB	700244	Punch a line of tape in binary mode.
		I/O Equipment
IORS	700314	Input/output read status. The content of given flags replace the content of the assigned AC bits.
TTS	703301	Test Teletype and skip if KSR 33 is connected to computer.
CAF	703302	Clear all flags.
~~SKP7~~	~~703341~~	~~Skip if processor is a PDP-7 or PDP-9~~

INPUT/OUTPUT TRANSFER INSTRUCTIONS (continued)

Mnemonic Symbol	Octal Code	Operation Executed
		Teletype Keyboard
KSF	700301	Skip if the keyboard flag is set to 1.
KRB	700312	Read the keyboard buffer. The content of the buffer is placed in AC10-17 and the keyboard flag is cleared.
		Teletype Teleprinter
TSF	700401	Skip if the teleprinter flag is set.
TCF	700402	Clear the teleprinter flag.
TLS	700406	Load teleprinter buffer. The content of AC10-17 is placed in the buffer and printed. The flag is cleared before transmission takes place and is set when the character has been printed.
		Type KF09A Automatic Priority Interrupt
SPI	705501	Skip on priorities inactive.
ISA	705504	Initiate selected activity.
DBK	703304	Debreak.
DBR	703344	Debreak and restore.
RPL	705512	Read API status.

SPECIAL I/O INSTRUCTIONS

Radius-angle high speed data channel

DWRA	702021	WRITE RADIUS AND ANGLE
DWSC	702022	WRITE SCOPE POINT
DESC	702024	ERASE SCOPE
DRDP	702041	READ DATA POINT (4 WORDS)
DWST	702044	WRITE STATUS
DRST	702052	READ STATUS

Periscope

PSKP	701021	SKIP ON $\overline{\text{INTERRUPT ENABLED}} + \overline{\text{COUNT ENABLED}}$
PRST	701032	READ STATUS
PØR1	701024	OR 1's TO STATUS
PSAS	701041	SKIP ON AC•STATUS = 0
PWVE	701042	WRITE VELOCITY
PØRO	701044	OR 0's TO STATUS
PWST	701064	WRITE STATUS

X stage

XSKP	701121	SKIP ON $\overline{\text{INTERRUPT ENABLED}} + \overline{\text{COUNT ENABLED}}$
XRST	701132	READ STATUS
XØR1	701124	OR 1's TO STATUS
XSAS	701141	SKIP ON AC•STATUS = 0
XWVE	701142	WRITE VELOCITY
XØRO	701144	OR 0's TO STATUS
XWST	701164	WRITE STATUS

Y stage

YSKP	701221	SKIP ON $\overline{\text{INTERRUPT ENABLED}} + \overline{\text{COUNT ENABLED}}$
YRST	701232	READ STATUS
YØR1	701224	OR 1's STATUS
YSAS	701241	SKIP ON AC•STATUS = 0
YWVE	701242	WRITE VELOCITY
YØRO	701244	OR 0's STATUS
YWST	701264	WRITE STATUS

Film drive 1

FSKP	701321	SKIP ON $\overline{\text{INTERRUPT ENABLED}}$ + $\overline{\text{COUNT ENABLED}}$
FRST	701332	READ STATUS
FØR1	701324	OR 1's TO STATUS
FSAS	701341	SKIP ON AC•STATUS = 0
FWVE	701342	WRITE VELOCITY
FØRO	701344	OR 0's TO STATUS
FWST	701364	WRITE STATUS

Film drive 2

GSKP	701421	SKIP ON $\overline{\text{INTERRUPT ENABLED}}$ + $\overline{\text{COUNT ENABLED}}$
GRST	701432	READ STATUS
GØR1	701424	OR 1's TO STATUS
GSAS	701441	SKIP ON AC•STATUS = 0
GWVE	701442	WRITE VELOCITY
GØRO	701444	OR 0's TO STATUS
GWST	701464	WRITE STATUS

Film drive 3

HSKP	701521	SKIP ON $\overline{\text{INTERRUPT ENABLED}}$ + $\overline{\text{COUNT ENABLED}}$
HRST	701532	READ STATUS
HØR1	701524	OR 1's STATUS
HSAS	701541	SKIP ON AC•STATUS = 0
HWVE	701542	WRITE VELOCITY
HØRO	701544	OR 0's TO STATUS
HWST	701564	WRITE STATUS

Magnetic Tape Controller

MSK2	703621	SKIP IF TAPE DRIVE #2 READY
MSK1	703641	SKIP IF TAPE DRIVE #1 READY
MWST	703624	WRITE STATUS
MRST	703632	READ STATUS
MWDB	703644	WRITE DATA BUFFER
MRDB	703652	READ DATA BUFFER

Console buttons

CPS	702101	SKIP ON CP ENABLED
CPW	702104	WRITE CP STATUS
CPR	702112	READ CP STATUS

Image Plane Digitizer (IPD)

MSS	702331	SKIP IF IPD NOT HOMED
MSLW	702324	WRITE LEFT REGISTER
MSLR	702332	READ LEFT REGISTER
MSRW	702344	WRITE RIGHT REGISTER
MSRR	702352	READ RIGHT REGISTER

Auto-fiducial

AFS	702201	SKIP ON AF ENABLED
AFW	702204	WRITE AF STATUS
AFR	702212	READ AF STATUS

APPENDIX 2

PDP-9 I/O CODES

MODEL 33, 35 ASR/KSR TELETYPE CODE (ASCII) IN OCTAL FORM

Character	IBM HEX CODE	8-Bit Code (in Octal)	Character	IBM HEX CODE	8-Bit Code (in Octal)
A	C1	301	!	4 2	241
B	C 2	302	"	4 4	242
C	C 3	303	#	4 6	243
D	C 4	304	$	4 8	244
E	C 5	305	%	4 A	245
F	C 6	306	&	4 C	246
G	C 7	307	'	4 E	247
H	C 8	310	(	5 0	250
I	C 9	311	)	5 2	251
J	D 1	312	*	5 4	252
K	D 2	313	+	5 6	253
L	D 3	314	,	5 8	254
M	D 4	315	-	5 A	255
N	D 5	316	.	5 C	256
O	D 6	317	/	5 E	257
P	D 7	320	:	7 A	272
Q	D 8	321	;	7 C	273
R	D 9	322	<	7 E	274
S	E 2	323	=	7 E	275
T	E 3	324	>	6 E	276
U	E 4	325	?	6 F	277
V	E 5	326	@	7 C	300
W	E 6	327	[	B B	333
X	E 7	330	/	B C	334
Y	E 8	331	]	B D	335
Z	E 9	332	↑	5 F	336
0	F 0	260	←	5 F	337
1	F 1	261	Leader/Trailer	0 0	200*
2	F 2	262	Line-Feed	2 5	212*
3	F 3	263	Carriage-Return	0 D	215
4	F 4	264	Space	4 0	240
5	F 5	265	Rub-out	F F	377*
6	F 6	266	Blank	0 0	000*
7	F 7	267	ALT Mode	7 B	375
8	F 8	270	* Ignored by the operating system		
9	F 9	271			

Appendix 3

IBM 360 - PDP9 ASSEMBLER ORGANIZATION

A. ASSEMBLER LAYOUT AND SIZE

The assembler consists of five parts; and their names are: SORT, BINSEAR, BEGIN, RWTAB, and SYMTAB. The function of each is described below.

SORT is used for sorting symbols into their alphabetical order.

BINSEAR is used for searching symbols. To identidy a specific symbol it performs a binary search on an alphabetically ordered set of symbols.

BEGIN is the main program which performs format and syntax checking on the input data, builds symbol and cross reference tables, translates instructions and symbols into numbers and assembles them into machine codes, and finally generates an output listing and a magnetic output tape.

RWTAB is a reserved word table used for storing PDP9 instructions and their equivalent machine codes.

SYMTAB is a table used for storing symbols that are used in the program. The symbol cross reference table begins right at the end of the SYMTAB.

The physical size of these programs in bytes are:

Name	Hexidecimal Size	Decimal Size in Bytes
SORT	84	132
BINSEAR	84	132
BEGIN	1F94	8086
RWTAB	588	1416
SYMTAB	19,000	102,400
		112,166

Maximum Input and Output data buffer length:

1.	INPUT DATA BUFFER	=	64,000 bytes
2.	OUTPUT DATA BUFFER	=	6,600 bytes
3.	MAGTAGE OUTPUT DATA BUFFER	=	216 bytes
			70,816 bytes

Hence, the maximum core space required by assembler in IBM 360/91 is (112,166 + 70,816) = 182,982 bytes.

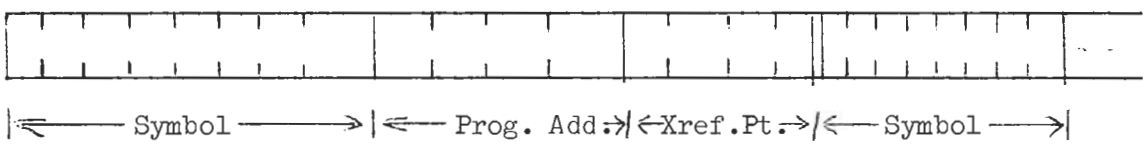
B. ASSEMBLER TABLE BUILDING FORMATS

The assembler is concerned with three tables: Reserved word table (RWTAB), Symbol table (SYMTAB), and Symbol cross reference table (XREF).

a) Reserved Word Table Format

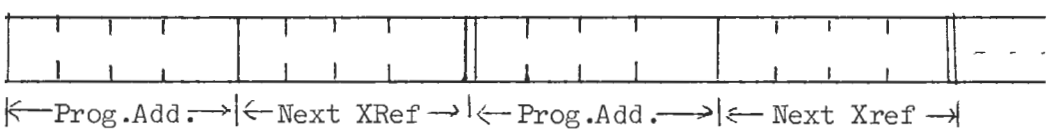
Each square represents one byte. The first 4 bytes represent the reserved word or the mnemonic instruction, and the second 4 bytes represent its equivalent machine code. Hence, each reserved word in reserved word table uses 8 bytes.

b) Symbol Table Format



The first 8 bytes are for symbol name. The maximum allowable symbol length is six characters, so there are two bytes left unused. The second 4 bytes represent the program address, or the value of the program counter when the particular symbol was first encountered in the label field. The third 4 bytes represent the pointer pointing to the address of the first reference in the cross reference table. Hence, each symbol in the symbol table uses 16 bytes. The symbol table (SYMTAB) begins right at the end of reserve word table (RWTAB).

c) Cross Reference Table Format



The first 4 bytes represent the program address where the symbol name was referred in the operand field. The second 4 bytes represent the pointer pointing to the address of the next reference in the

cross reference table. Hence, each XREF in the cross reference uses 8 bytes. It is necessary to clear the entire cross reference table with blanks; this way, the last reference made on a symbol will always have its NEXT XREF space blank. In the cross reference printout routine, a blank in the NEXT XREF space will terminate the cross reference printout for that particular symbol.

C. ASSEMBLER OUTPUT CODE CONVERSIONS

a) Assembler Output Listing Code Conversion

Since the numerical portion of the PDP9 assembler has to be in octal, the IBM 360 printout has to be in EBCDIC, and assembler computations are done in binary, hence, the following code converstion procedure must be followed:

1. 13 bit binary address is converted to octal for printout:

   XXXXXXXXXXXXX ⟶ 000X0XXX0XXX0XXX0XXX

   where X represents a useful bit.

2. The converted address is then OR'ed with the octal instruction code:

   0XXX0XXX0000000000000000

   ØR

   0000000X0XXX0XXX0XXX0XXX

   ↓

   0XXX0XXX0XXX0XXX0XXX0XXX

3. For printout the each converted octal must be in hexidecimal:

   1111 0000

   OR

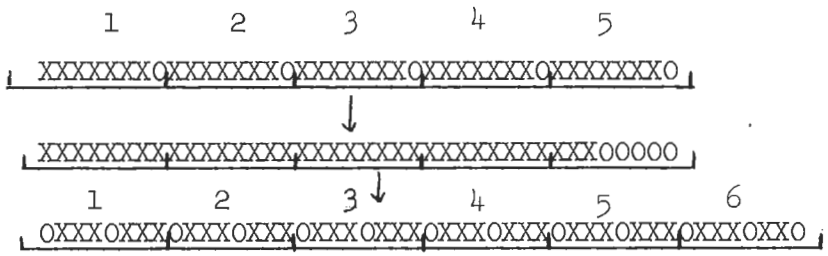
   0000 0XXX

   ↓

   1111 0XXX

b) Assembler Text Conversion

Text is written in IBM EBCDIC, and this is then translated to PDP9 ASCII by table lookup method. See Appendix 2. Each ASCII character is in this format: XXXXXXX0. Similarly, the text has to be converted to octal for printout. To do this, .ASC pseudo-op takes five ASCII characters a time and does shifting and packing on them:



Note that 5 ASCII characters after conversion will exactly occupy 2 PDP9 words. Once in octal form, to get hexidecimal printout, just follow the method described in C-a-3.

c) Assembler Magtape Output Format

The octal form is useful for hexidecimal printout; however, it is not very practical for magtape output. Rather for magtape output, it is useful to have the zeros removed and pack the useful bits in a string. That is, change 0XXX0XXX0XXX to XXXXXXXXX. This part of the operation is done at the time when the hexidecimal printout is being done.

Assembler produces a fixed record magtape. Each record is 108 bytes long, and the first two bytes of the first record on the magtape gives the starting address of the assembled program.

Appendix 4

SAMPLE PROGRAMS

```
                        .FIX
                *       THIS IS A SAMPLE PROGRAM
                *       IT ROTATES A BIT THROUGH THE AC AT A RATE
                *       DETERMINED BY THE AC SWITCHES

00200   750004          GO      LAS
00201   741101                  SPA,CMA                 EXAMINE AC SWITCHES
00202   600200                  JMP     GO              WAIT UNTIL ACSO=0
00203   040224                  DAC     CNTSET
00204   200225                  LAC     ONE             1 IS A CONSTANT
00205   040223                  DAC     BIT
00206   744000                  CLL                     CLEAR THE LINK
00207   200224          LOOP    LAC     CNTSET
00210   040222                  DAC     CNT
00211   200223                  LAC     BIT
00212   440222          LOOP1   ISZ     CNT             LOOP UNTIL CNT GOES TO ZERO
00213   600212                  JMP     LOOP1           JUMP TO PRECEDING LOCATION
00214   740010                  RAL
00215   040223                  DAC     BIT             ROTATE BIT
00216   750004                  LAS
00217   740100                  SMA                     IF ACSO=1, RESET TIME CONSTANT
00220   600207                  JMP     LOOP
00221   600200                  JMP     GO
                *       STORAGE FOR PROGRAM DATA
00222   000000          CNT     0
00223   000000          BIT     0
00224   000000          CNTSET  0
00225   000001          ONE     1
```

```
                        *               THIS IS AN ASSEMBLER TEST PROGRAM

                                .FIX
                                .CRG    5000
                                .EXT    LOOKUP,COUNT,SHOWUP
05000   000000          JACK    .BLK    1000
06000   306C13                  ADD     A
06001   306103                  ADD     A+70
06002   300C13                  ADD     (1000
06003   300C14                  ADD     (256            400
06004   300C10                  ADD     (-256
06005   300C07                  ADD     (+256
06006   75CC00                  CLA
06007   75CC01                  CLC
06010   744C01                  CLL,CMA                 744001
06011   754C01                  CLL,CMA,CLC             754001
06012   74CC02                  CML
06013   046C75          A       DAC     X
06014   066C13                  DAC*    A
06015   146C13                  DZM     A
06016   146C76                  DZM     Y
06017   740C40                  HLT
06020   446C77                  ISZ     Z
06021   606C13                  JMP     A
06022   606C44                  JMP     SUB+20
06023   106C24                  JMS     SUB
06024   206C73          SUB     LAC     P
06025   226C11                  LAC*    A-2
06026   75CC04                  LAS
06027   760C00                  LAW     0
06030   777777                  LAW     -1
06031   760C10                  LAW     8D
06032   760C20                  LAW     16D
06033   777772                  LAW     -6
06034   766C03                  LAW     A-10
06035   766C13                  LAW     A
06036   767C13                  LAW     A+1000
06037   606C41                  JMP     *+2
06040   606C34                  JMP     *-4
06041   626C12                  JMP*    A-1
06042   740C00                  NOP
06043   746C74                  OAS     ACCUM
06044   74CC30                  RAL,RAR                 740030
06045   744C30                  RCL,RCR                 744030
06046   742C30                  RTL,RTR                 742030
06047   546C50                  SAD     B
06050   741C00          B       SKP
06051   740100                  SMA
06052   741600                  SNA,SNL                 741600
06053   741100                  SPA
06054   744CC2                  STL
06055   741600                  SZL,SZA                 741600
06056   346C71                  TAD     C
06057   346C06                  TAD     A-5
06060   777743                  TAD     A-B
06061   340C35                  TAD     B-A
06062   340C15                  TAD     (48D
06063   4C6C7C                  XCT     M
```

```
06064   246C72                  XCR     D
06065   246C13                  XCR     A+80
06066   117322          AB      .EQU    117322
06067   000543          BC      .EQU    000543
06070   740C10          M       .EQU    740010
06071   006C72          C       .EQU    D
06072   000C05          D               5
06073   047420          P       DAC     7420
06074   000C00          ACCUM   0
06075   006C13          X       .EQU    A
06076   006C13          Y       .EQU    A
06077   006C13          Z       .EQU    A
06100   201347          TYPE1   .ASC     .<(+&$*);-/,=>?:#@%
06101   424126
06102   231105
06103   224566
06104   265365
06105   436574
06106   375644
06107   340C00
06110   406C50          TYPE2   .ASC    ABCDEFGHIJKLMNOPQRST
06111   342212
06112   432171
06113   044624
06114   456311
06115   547236
06116   502432
06117   251650
06120   526552          TYPE3   .ASC    UVWXYZ0123456789
06121   754262
06122   551406
06123   131146
06124   321526
06125   633560
06126   344C00
06127   000C00
```

```
                        *               THIS PROGRAM CONTAINS ILLEGAL COMMANDS
                        *
                        *
                        *
                                .REL
C0000   000C11  SI      3TYPE   LCC     (A
00001   440C12                  ISZ     (A+B
00002   006C13  MI      M         LAC   A
00003   00CC00  P               .PCS    B
00004   040C03                  DAC     +C
00005   00CC00  IO                JMP     P
00006   006C71  I               CAC     C
                                .END
00007   000256                          (+256
00010   777522                          (-256
00011   000C00  0                       (A
00012   000C00  0                       (A+B
00013   001C00                          (1000
00014   00C256                          (256
00015   000C60                          (48D
```

A	06C13	06000	06001	06014	06015	06021	06025	06034	06035
		06075	06076	06077	00002				
AB	06C66								
ACCUM	06C74	06043							
B	06C50	06047	06060	06061					
BC	06C67								
BIT	00223	00205	00211	00215					
C	06C71	06056	00006						
CNT	00222	00210	00212						
CNTSET	00224	00203	00207						
COUNT	00C00								
D	06C72	06064	06071						
GO	00200	00202	00221						
JACK	05C00								
LOOKUP	00C00								
LOOP	00207	00220							
LOOP1	00212	00213							
M	06C70	06063							
ONE	00225	00204							
P	06C73	06024							
SHOWUP	00C00								
SUB	06C24	06022	06023						
TYPE1	06100								
TYPE2	06110								
TYPE3	06120								
X	06C75	06013							
Y	06C76	06016							
Z	06C77	06020							
(+256	00C07	06005							
(-256	00C10	06004							
(A	00C11	00000							
(A+B	00C12	00001							
(1000	00C13	06002							
(256	00C14	06003							
(48D	00C15	06062							

Appendix 5

JOB CONTROL CARDS

To assemble a program on the IBM 360, one needs a set of Job Control Cards at the beginning of the deck. The Job Control Cards are given on the next page and they are identified by // at the beginning of the card.

```
//MJH2PDP9 JOB 'MJH$CG,(M.J.HU X2289)',54,CLASS=B,PRTY=5,CPU=30S,      X JOB 316
//             LINES=45000,CO=3
//STEP0    EXEC  PGM=IEFBR14
//D1       DD    DSNAME=PROGRAM,VOLUME=SER=SCFEV4,DISP=(OLD,DELETE), X
//             UNIT=2314
//D2       DD    DSNAME=PASSI,VOLUME=SER=SCFEV4,DISP=(OLD,DELETE),   X
//             UNIT=2314
//D3       DD    DSNAME=PASSII,VOLUME=SER=SCFEV4,DISP=(OLD,DELETE),  X
//             UNIT=2314
ACCOUNT(INITIATION)                MJH2PDP9 STEP0                          DATE=
IEF236I ALLOC. FOR MJH2PDP9 STEP0
IEF237I D1       ON 231
IEF237I D2       ON 231
IEF237I D3       ON 231
IEF284I   PROGRAM                                        NOT DELETED 8
IEF284I   VOL SER NOS= SCFEV4 1.
IEF284I   PASSI                                          NOT DELETED 8
IEF284I   VOL SER NOS= SCFEV4 1.
IEF284I   PASSII                                         NOT DELETED 8
IEF284I   VOL SER NOS= SCFEV4 1.
ACCOUNT(STEP) 54                   MJH2PDP9 STEP0    RUN TIME=(    ,00.01) DATE=
//PDP9 EXEC LINKGO,PARM.LKED='LET,XREF'
XX         PROC EDITOR=IEWLF88O                                          SYS00100
XXLKED     EXEC PGM=&EDITOR,PARM=(LET,MAP,LIST),REGION=150K              SYS00200
***                                                                      SYS00300
***   STEP LKED OF PROCEDURE LINKGO                                      SYS00400
***                                                                      SYS00500
XXSYSLIB   DD DSNAME=SYS4.FORTLIB,DISP=SHR                               SYS00600
XX         DD DSNAME=SYS1.FORTLIB,DISP=SHR                               SYS00700
XX         DD DSNAME=SYS3.FORTLIB,DISP=SHR                               SYS00800
//LKED.SYSUT1  DD SPACE=(CYL,(2,1))
X/SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))                               SYS00900
XXSYSPRINT DD SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=3509)            SYS01000
//LKED.SYSLIN  DD  DSNAME=PUB.MJH.PDP9A1,VOLUME=SER=PUB002,DISP=OLD,    X
//             UNIT=2314
X/SYSLIN   DD DDNAME=SYSIN                                               SYS01100
//           DD DSNAME=PUB.MJH.PDP9A2,VOLUME=SER=PUB002,DISP=OLD,       X
//             UNIT=2314
//           DD DSNAME=PUB.MJH.PDP9A3,VOLUME=SER=PUB002,DISP=OLD,       X
//             UNIT=2314
//           DD DSNAME=PUB.MJH.PDP9A4,VOLUME=SER=PUB002,DISP=OLD,       X
//             UNIT=2314
XXSYSLMOD  DD DSNAME=&&GOSET(MAIN),UNIT=(SYSDA,SEP=SYSUT1),             *SYS01200
XX           SPACE=(CYL,(1,1,1)),DISP=(NEW,PASS)                         SYS01300
ACCOUNT(INITIATION)                MJH2PDP9 LKED                           DATE=
IEF236I ALLOC. FOR MJH2PDP9 LKED     PDP9
IEF237I SYSLIB   ON 537
IEF237I          ON 437
IEF237I          ON 537
IEF237I SYSUT1   ON 430
IEF237I SYSPRINT ON 661
IEF237I SYSLIN   ON 234
IEF237I          ON 234
IEF237I          ON 234
IEF237I          ON 234
IEF237I SYSLMOD  ON 537
IEF285I   SYS4.FORTLIB                                   KEPT
IEF285I   VOL SER NOS= SCF171.
IEF285I   SYS1.FORTLIB                                   KEPT
IEF285I   VOL SER NOS= SCF170.
```

```
IEF285I   SYS3.FORTLIB                                    KEPT
IEF285I   VOL SER NOS= SCF171.
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.R0000699        DELETED
IEF285I   VOL SER NOS= SCR001.
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.R0000700        DELETED
IEF285I   VOL SER NOS=      .
IEF285I   PUB.MJH.PDP9A1                                  KEPT
IEF285I   VOL SER NOS= PUB002.
IEF285I   PUB.MJH.PDP9A2                                  KEPT
IEF285I   VOL SER NOS= PUB002.
IEF285I   PUB.MJH.PDP9A3                                  KEPT
IEF285I   VOL SER NOS= PUB002.
IEF285I   PUB.MJH.PDP9A4                                  KEPT
IEF285I   VOL SER NOS= PUB002.
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.GOSET           PASSED
IEF285I   VOL SER NOS= SCF171.
ACCOUNT(STEP) 54                    MJH2PDP9 LKED     RUN TIME=(     ,00.14) DATE=70
XXGO        EXEC PGM=*.LKED.SYSLMOD,COND=(5,LT,LKED)                          SYS01400
***                                                                           SYS01500
***   STEP GO OF PROCEDURE LINKGO                                             SYS01600
***                                                                           SYS01700
XXFT05F001  DD DDNAME=SYSIN                                                   SYS01800
XXFT06F001  DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=3458,BUFNO=2)        SYS01900
//GO.DISC1  DD  DSNAME=PROGRAM,DCB=(BLKSIZE=32000,LRECL=80,RECFM=FBT),  X
//              UNIT=2314,VOLUME=SER=SCFEV4,SPACE=(CYL,(6,1)),          X
//              DISP=(NEW,PASS,DELETE)
//GO.FDISC1 DD  DSNAME=PROGRAM,DCB=(BLKSIZE=32000,LRECL=80,RECFM=FBT),  X
//              UNIT=2314,VOLUME=SER=SCFEV4,DISP=(OLD,DELETE)
//GO.DISC2  DD  DSNAME=PASSI,DCB=(BLKSIZE=31680,LRECL=132,RECFM=FBT),   X
//              UNIT=2314,VOLUME=SER=SCFEV4,SPACE=(CYL,(6,1)),          X
//              DISP=(NEW,PASS,DELETE)
//GO.FDISC2 DD  DSNAME=PASSI,DCB=(BLKSIZE=31680,LRECL=132,RECFM=FBT),   X
//              UNIT=2314,VOLUME=SER=SCFEV4,DISP=(OLD,DELETE)
//GO.DISC3  DD  DSNAME=PASSII,DCB=(BLKSIZE=31680,LRECL=132,RECFM=FBT),  X
//              UNIT=2314,VOLUME=SER=SCFEV4,SPACE=(CYL,(6,1)),          X
//              DISP=(NEW,PASS,DELETE)
//GO.FDISC3 DD  DSNAME=PASSII,DCB=(BLKSIZE=31680,LRECL=132,RECFM=FBT),  X
//              UNIT=2314,VOLUME=SER=SCFEV4,DISP=(OLD,DELETE)
//GO.ONPRINT    DD  SYSOUT=A,DCB=(BLKSIZE=3300,LRECL=132,RECFM=FBA),    X
//              SPACE=(TRK,300)
//GO.ONTAPE   DD    UNIT=(TAPE9,,DEFER),VOLUME=SER=PDP9AP,LABEL=(1,NL),  X
//              DSNAME=TOPDP9,DCB=(BLKSIZE=108,LRECL=108,RECFM=F),      X
//              DISP=(NEW,DELETE)
//GO.FTAPE    DD    UNIT=(TAPE9,,DEFER),VOLUME=SER=PDP9AP,LABEL=(1,NL),  X
//              DSNAME=TOPDP9,DCB=(BLKSIZE=108,LRECL=108,RECFM=F),      X
//              DISP=(NEW,KEEP)
//GO.ONPRINT1   DD  SYSOUT=A,DCB=(BLKSIZE=3300,LRECL=132,RECFM=FBA),    X
//              SPACE=(TRK,300)
//GO.SYSUDUMP DD    SYSOUT=A
//GO.CARDS    DD    *,DCB=BLKSIZE=80
//
ACCOUNT(INITIATION)                 MJH2PDP9 GO                          DATE=700
IEF236I ALLOC. FOR MJH2PDP9 GO        PDP9
IEF237I PGM=*.DD ON 537
IEF237I FT06F001 ON 661
IEF237I DISC1    ON 231
IEF237I FDISC1   ON 231
IEF237I DISC2    ON 231
IEF237I FDISC2   ON 231
IEF237I DISC3    ON 231
```

```
IEF237I FDISC3   ON 231
IEF237I ONPRINT  ON 666
IEF237I ONTAPE   ON 0C1
IEF237I FTAPE    ON 0C1
IEF237I ONPRINT1 ON 667
IEF237I SYSUDUMP ON 668
IEF237I CARDS    ON 685
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.GOSET        PASSED
IEF285I   VOL SER NOS= SCF171.
IEF285I   PROGRAM                                      PASSED
IEF285I   VOL SER NOS= SCFEV4.
IEF285I   PROGRAM                                      DELETED
IEF285I   VOL SER NOS= SCFEV4.
IEF265I   PASSI                                        PASSED
IEF285I   VOL SER NOS= SCFEV4.
IEF285I   PASSI                                        DELETED
IEF285I   VOL SER NOS= SCFEV4.
IEF285I   PASSII                                       PASSED
IEF285I   VOL SER NOS= SCFEV4.
IEF285I   PASSII                                       DELETED
IEF285I   VOL SER NOS= SCFEV4.
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.R0000702     DELETED
IEF285I   VOL SER NOS=      .
IEF285I   TOPDP9                                       DELETED
IEF285I   VOL SER NOS= PDP9AP.
IEF285I   TOPDP9                                       KEPT
IEF285I   VOL SER NOS= PDP9AP.
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.RC000703     DELETED
IEF285I   VOL SER NOS=      .
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.R0000705     DELETED
IEF285I   VOL SER NOS=      .
ACCOUNT(STEP) 54                  MJH2PDP9 GO       RUN TIME=(    ,11.44) DATE=7
IEF285I   SYS70048.T145902.RV000.MJH2PDP9.GOSET        DELETED
IEF285I   VOL SER NOS= SCF171.
IEF284I   PROGRAM                                      NOT DELETED 8
IEF284I   VOL SER NOS= SCFEV4 1.
IEF284I   PASSI                                        NOT DELETED 8
IEF284I   VOL SER NOS= SCFEV4 1.
IEF284I   PASSII                                       NOT DELETED 8
IEF284I   VOL SER NOS= SCFEV4 1.
ACCOUNT(JOB)  54                  MJH2PDP9          RUN TIME=(    ,11.59) DATE=7
```