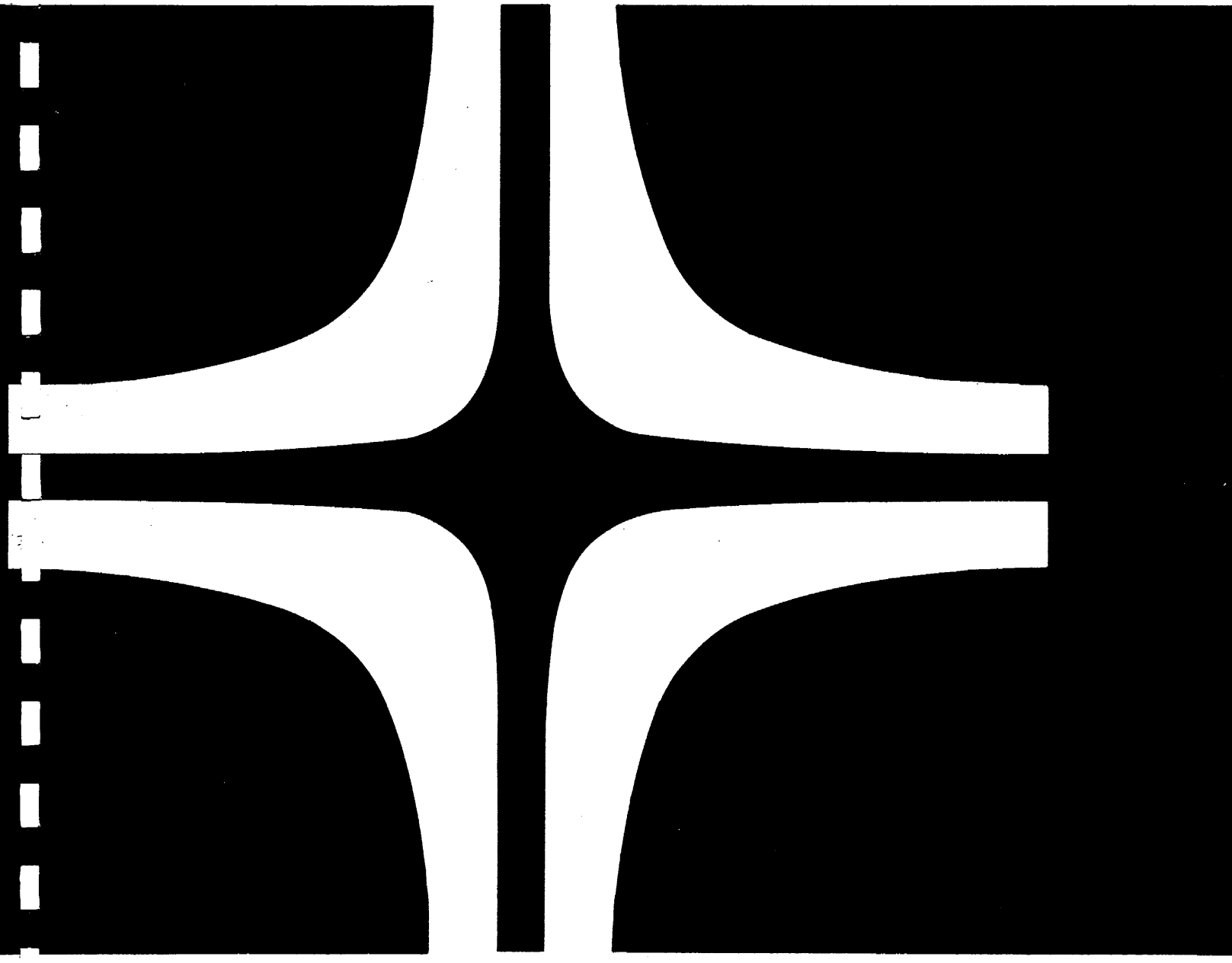
UNIVAC 9000
CARD ASSEMBLER

Programmed Instruction Course

Book 3 - BAL Application



SPERRY UNIVAC
COMPUTER SYSTEMS

EDUCATION CENTER

UE-686.2B

UNIVAC 9000
CARD ASSEMBLER
PROGRAMMED INSTRUCTION COURSE

BAL APPLICATION

Book-3

UE-686.2B

CONTENTS

REFERENCE SUPPLEMENT

CONTENTS (Continued)

INTRODUCTION

This text is Book 3 of a series of programmed instruction manuals designed to teach 9000 Series Card Assembler programming. Successful completion of Book 1 (UE-868.1), Book 2 (UE-868.2) and the self-test evaluation covering Assembler Language programming are prerequisites for starting Book 3.

In this text, the concepts and techniques taught in the course are implemented by a "talk-thru" exercise in which the solution of a typical data processing problem is presented. Each stage in the solution is explained including the problem statement, formatting, flowcharting, and coding. A diagnostic examination covering the flowcharting and coding is included in the exercise.

In the final section of the course, the novice programmer, given a problem statement, is responsible for the solution of the problem. The reference supplement is designed to provide a convenient means of reviewing instructions.

TALK-THRU PROGRAM

This exercise includes all of the documentation required to produce a complete program. The material provided includes the process flowchart, source coding, printer format chart, and the assembler listing. You will be instructed to refer to this material as you follow the analysis of the problem.

The exercise begins with a statement of the problem and a description of the input and output parameters. The next steps are the flowcharting and coding of a solution. One solution is illustrated on the flowchart and coding sheets found on pages 3–18 through 3–23. The several intervening pages of this text give a detailed analysis of this solution by showing each block of the flowchart, the coding associated with each block, and a description of the rationale and implications of each step.

MARKETING SALES REPORT PROBLEM STATEMENT

The programmer's objective is to prepare a report to management reflecting the total annual sales for each salesman, and also, a total of all sales for the year.

There is one input file and one output file as shown below.

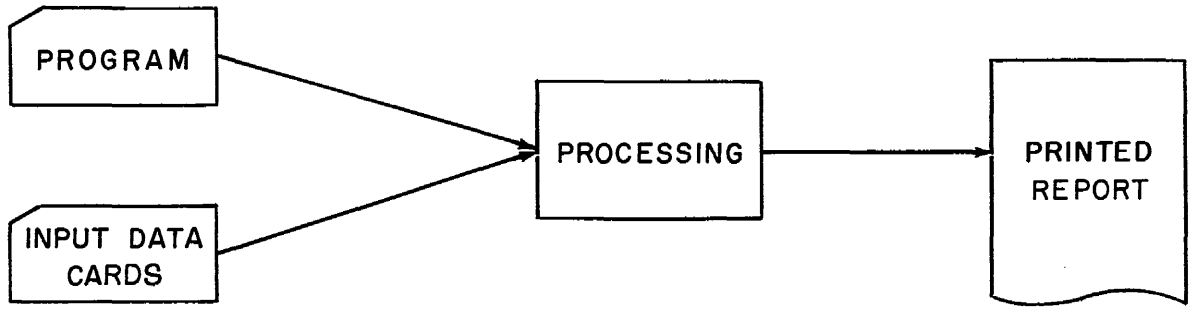


Figure 3–1 System Flowchart

Input Data - Sales cards for each salesman. Sales cards contain: Employee number, name and sales information.

Data cards are in employee number sequence.

Processing - Print headings at top of page (EMPLOYEE NUMBER, SALESMAN and SALE).

Number all pages and print a final heading.

Accumulate total annual sales for each salesman.

Final total of all sales for the year.

Output - A printed report showing total annual sales for each salesman and a final total of all sales for the year. Maximum number of pages is nine.

NOTE: To simplify this exercise, title of report, date, and other items normally found in the heading have been omitted.

INPUT FILE

The input file is constructed of 80-character (80-byte) records in the format described below.

Item	Columns	Bytes
Employee Number	1-6	6
Salesman Name	7-21	15
Sales per Month	22-27	6

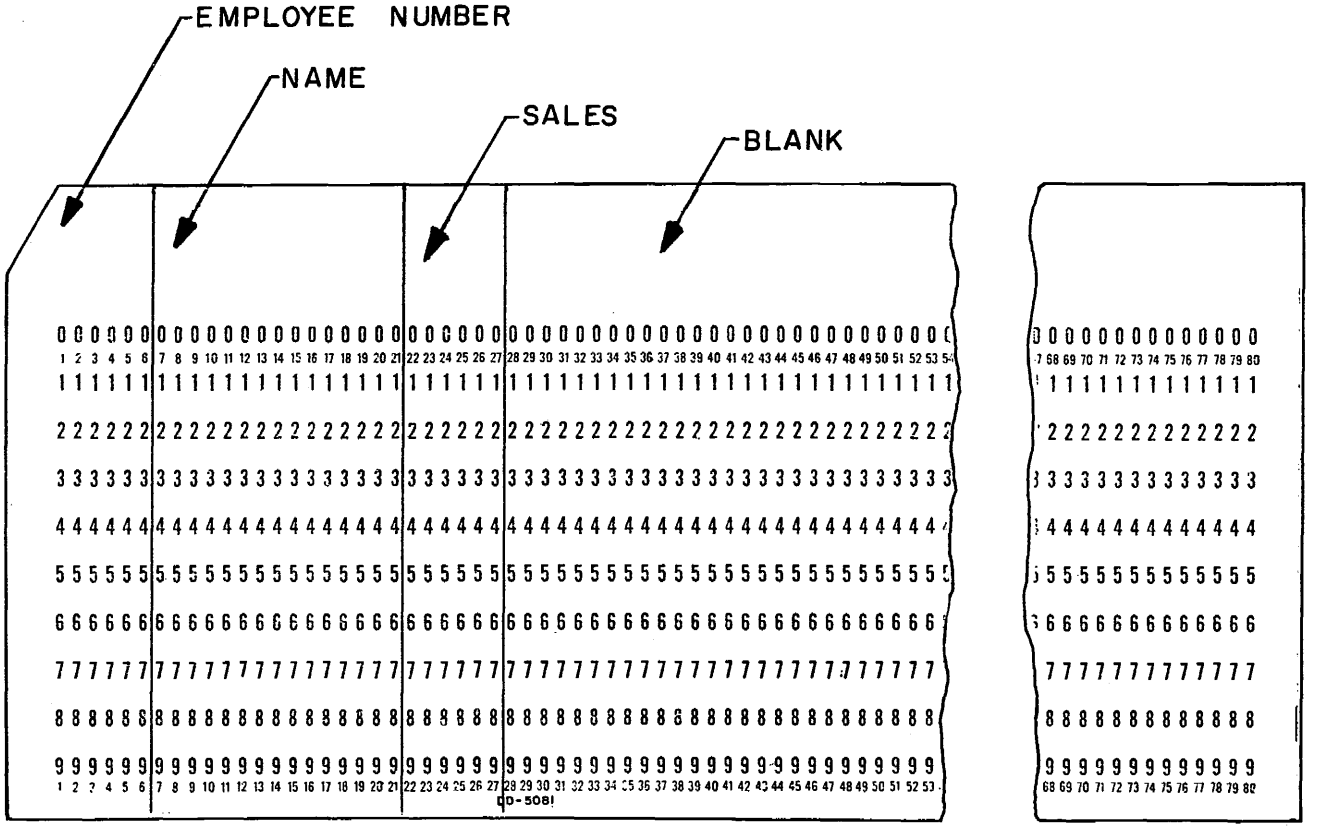


Figure 3–2 Input Card Format

OUTPUT REPORT

The output is a printed report in the following format:

Column headings are printed at the top of each page.

Page numbers are printed at the top of each page.

Lines of print are double spaced.

Total sales is printed at end of report.

The Printer Format Chart on the next page illustrates a typical report to be printed in the format specified above.

PRINTER FORMAT CHART

FORM NUMBER _____ APPLICATION _____ DATE _____

FORM PARTS _____ RUN NAME _____ RUN NUMBER _____ PREPARED BY _____

TYPE OF PRINTOUT _____ RECORD NAME _____ RECORD NUMBER _____ APPROVED BY _____

DATE APPROVED _____

EMPLOYEE NUMBER	SALESMAN	SALES	1
000001	BURNS, THOMAS	651,321.04	
000002	REED, DONALD	123,321.50	
000003	MCKNIGHT, JAMES	798,453.21	
	TOTAL SALES	01,573,095.75	
	END OF REPORT FOR FISCAL YEAR		

START

```
LABEL     OPERATION     OPERAND
1         10        16
          START     Ø
```

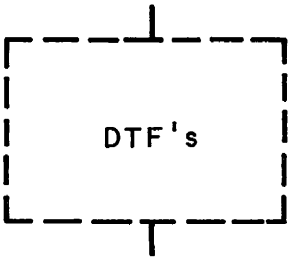
100

See note below.

The START Assembler-directing instruction defines the starting location in memory of the first statement in the program. When the program is linked to other subroutines the START address may be changed.

DEFINE INPUT/OUTPUT DEVICES TO BE USED BY PROGRAM

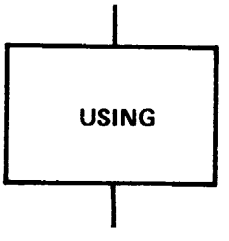
```
LABEL     OPERATION     OPERAND                                  72
1         10        16
READ      DTFCR     EOFA=EOJ,                                    X
                    IOA1=RBUF,                                   X
                    ITBL=TBRD,                                   X
                    MODE=TRANS
PRNT      DTFPR     BKSZ=132,                                    X
                    CNTL=YES,                                    X
                    PRAD=2,                                      X
                    PROV=FOF,                                    X
                    FONT=63
          END
```



DTF's are not handled by the user program. Handling of DTF's will be explained at the end of this talk-thru. The DTF's are positioned here for reference only.

ALLOCATE MEMORY

```
LABEL     OPERATION     OPERAND
1         10        16
          USING     *,Ø
          USING     *,1
```



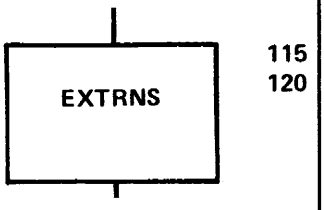
105
110

Each USING statement allocates 4096 bytes of memory. (It is assumed that the memory storage requirement for this program will not exceed 8000 bytes.)

NOTE: The numbers listed near the right margin are referenced to the coding line numbers used by the programmer on the coding sheets, pages 3–19 to 3–22.

SUPPLY PROGRAM WITH LABELS FOR DTF'S

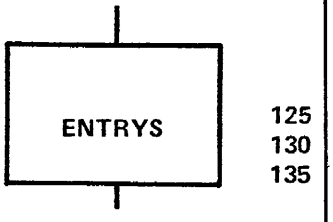
```
LABEL     OPERATION     OPERAND
1         10            16
          EXTRN         READ
          EXTRN         PRNT
```



READ and PRNT will be defined in another program.

SUPPLY SYSTEM WITH LABELS OF SUBPROGRAMS

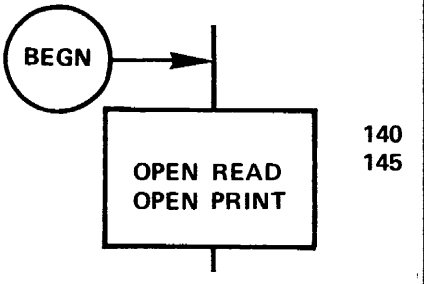
```
LABEL     OPERATION     OPERAND
1         10            16
          ENTRY         RBUF
          ENTRY         EOJ
          ENTRY         FOF
```



RBUF, EOJ, and FOF are the labels of subprograms within the user program.

ACTIVATE CARD READER, PRINTER

```
LABEL     OPERATION     OPERAND
1         10            16
BEGN      OPEN          READ
          OPEN          PRNT
```

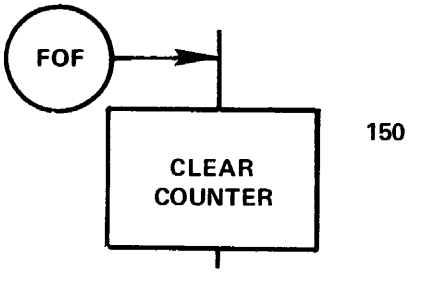


OPEN READ makes the file named READ available for sending input.

OPEN PRNT makes the file named PRNT ready to receive output.

CLEAR PRINTER LINE COUNTER TO ZERO

```
LABEL     OPERATION     OPERAND
1         10            16
FOF       MVC           CNTR,TZER+4
```



Moves zeros from storage area TZER+4 to the two-byte area defined as CNTR.

CLEAR PRINTER WORK AREA WITH SPACES

```
LABEL    OPERATION    OPERAND
1        10           16
         MVI          PRWK,X'40'
         MVC          PRWK+1(131),PRWK
```

MVI moves a space (blank) into the first byte of the PRWK area.

MVC moves a space from the first byte of PRWK into the next 131 positions.

CLEAR PRINTER AREA

155
160

POSITION PAPER ON PRINTER

```
LABEL    OPERATION    OPERAND
1        10           16
         CNTRL        PRNT,SK,7
```

Positions printer paper to the top of page.

ADVANCE TO HOME PAPER POSITION

165

SETUP PAGE NUMBER

```
LABEL    OPERATION    OPERAND
1        10           16
         AP           PAGE(1),ONE(1)
```

Adds one packed byte from the contents of an area defined as ONE to a one-byte area defined as PAGE and stores the result in the area defined as PAGE.

ADD 1 TO PAGE

170

INSERT SPACES BEFORE FIRST SIGNIFICANT DIGIT

```
LABEL    OPERATION    OPERAND
1        10           16
         MVC          PSAL+22(2),MSK3
```

Moves the mask from MSK3 into an area defined as PSAL + 22(2).

Note: (2) = 2 bytes.

MOVE PAGE TO PRINTER WORK AREA

175
180

PREPARE PAGE NUMBER FOR PRINTING

```
LABEL      OPERATION      OPERAND
1          10             16
           ED             PSAL+22(2),PAGE
```

Unpacks PAGE (page number) and places it in PSAL + 22 for a length of 2 bytes.

PREPARE COLUMN HEADERS FOR PRINTING

```
LABEL      OPERATION      OPERAND
1          10             16
           MVC            PEMP(15),HDR1
           MVC            PSMN(8),HDR2
           MVC            PSAL(5),HDR3
```

Moves first header (HDR1-Employee Number) to PEMP (15 positions)

Moves second header (HDR2-Salesman) to PSMN (8 positions)

Moves third header (HDR3-Sales) to PSAL (5 positions)

All headers defined above are subdivisions of PRWK.

ADVANCE SINGLE LINE BEFORE PRINTING

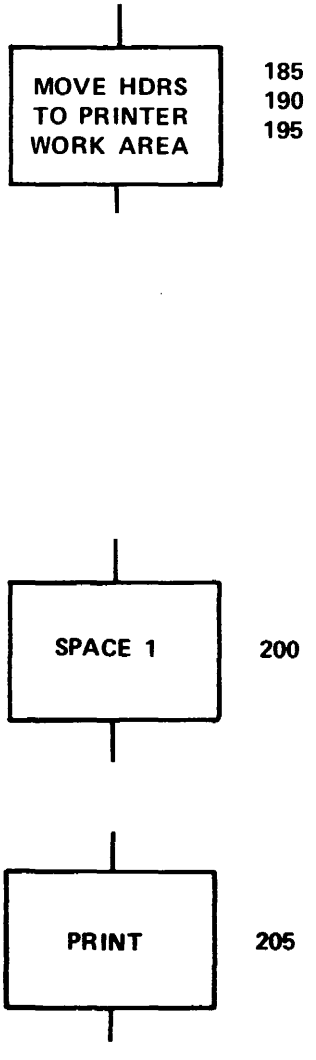
```
LABEL      OPERATION      OPERAND
1          10             16
           CNTRL          PRNT,SP,1
```

Advances single line before printing the header.

PRINT COLUMN HEADINGS

```
LABEL      OPERATION      OPERAND
1          10             16
           PUT            PRNT,PRWK
```

All data that has been placed in the PRWK reserved area HDR1, HDR2, HDR3, PAGE will be sent from PRWK to printer.



READ ANOTHER CARD

```
LABEL     OPERATION     OPERAND
1         10            16
          GET           READ,CARD
```

Reads a card into an area defined as CARD.

CLEAR PRINTER WORK AREA

```
LABEL     OPERATION     OPERAND
1         10            16
MAN       MVI           PRWK,X'40'
          MVC           PRWK+1(131),PRWK
```

MVI places a space into first location of an area labelled PRWK.

MVC moves the space from PRWK + 1 into the next 131 consecutive locations.

SET UP PRINT AREA WITH DATA

```
LABEL     OPERATION     OPERAND
1         10            16
          MVC           PRWK+35(6),EMPN
          MVC           PRWK+56(15),SMAN
```

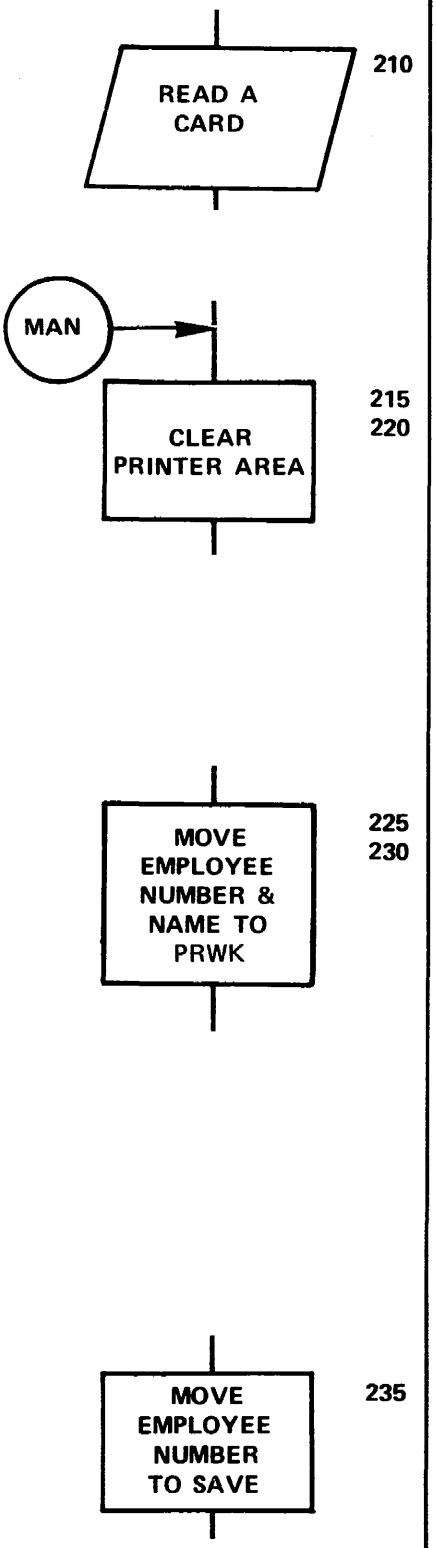
Moves EMPN to PRWK area (Employee number)

Moves SMAN to PRWK area (Salesman's Name)

STORE EMPLOYEE NUMBER FOR LATER COMPARISON

```
LABEL     OPERATION     OPERAND
1         10            16
          MVC           SAVE(6),EMPN
```

Moves EMPN (employee number) to a storage area called SAVE.



PACK SALE FOR SUBSEQUENT ADDITION

LABEL	OPERATION	OPERAND
MIN	PACK	PAKS(4),SALE(6)

Packs the six bytes of information located at SALE into the four bytes reserved for an area labelled PAKS.

ACCUMULATE TOTALS FOR EACH SALESMAN

LABEL	OPERATION	OPERAND
	AP	TSAL(6),PAKS(4)

Adds the packed information found in PAKS (four bytes) to the six bytes located at TSAL.

READ ANOTHER CARD

LABEL	OPERATION	OPERAND
	GET	READ,CARD

Reads a card and places data into area defined as CARD.

COMPARE EMPLOYEE NUMBERS

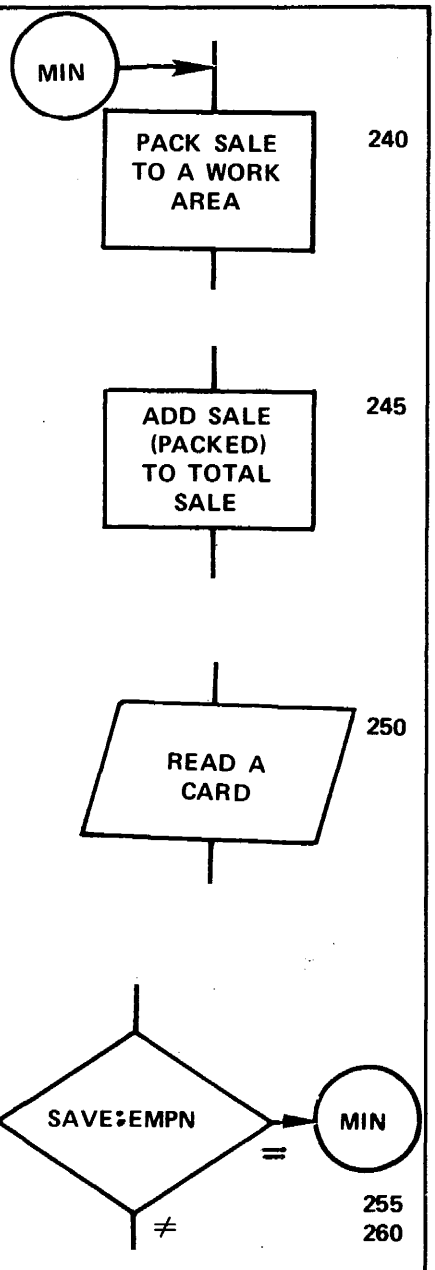
LABEL	OPERATION	OPERAND
	CLC	SAVE,EPMN

Compares information stored in EMPN with the information stored in SAVE.

BRANCH BACK TO MIN IF SAME EMPLOYEE; ADVANCE TO PRINT TOTAL OTHERWISE

LABEL	OPERATION	OPERAND
	BC	8,MIN

Branches if equal to the subroutine labelled MIN. If not equal, processes next instruction.



ACCUMULATE TOTAL SALES

```
LABEL      OPERATION      OPERAND
1          10             16
           AP             FTOT,TSAL
```

Adds (packed) information stored in TSAL to information stored in FTOT and places result back in FTOT.

ADD TOTAL SALES TO FINAL TOTAL — 265

STRUCTURE PRINTER WORK AREA FOR TSAL (SALESMAN'S TOTAL SALES)

```
LABEL      OPERATION      OPERAND
1          10             16
           MVC            PRWK+80(16),MSK1
```

Moves the mask located at MSK1 to PRWK.

MOVE MASK, THEN TOTAL SALES TO PRINTER WORK AREA — 270, 275

PREPARE TOTAL SALES FOR ONE EMPLOYEE FOR PRINTING

```
LABEL      OPERATION      OPERAND
1          10             16
           ED             PRWK+80(16),TSAL
```

Unpacks information found in location labelled TSAL and places it in PRWK + 80 (16 bytes).

PRINT

```
LABEL      OPERATION      OPERAND
1          10             16
           PUT            PRNT,PRWK
```

All data placed in printer work area (PRWK) will be sent to PRNT and printed.

PRINT — 280

ADD A ONE TO LINE COUNTER

```
LABEL      OPERATION      OPERAND
1          10             16
           AP             CNTR,ONE
```

Adds (packed) information found in location labelled ONE to information found at location labelled CNTR.

ADD 1 TO COUNTER — 285

COMPARE FOR 25 LINES (END OF PAGE)

LABEL	OPERATION	OPERAND
	CP	CNTR,FIVE

Compares (packed) data located in area labelled FIVE with the data labelled CNTR.

BRANCH TO FILE OVERFLOW ROUTINE IF EQUAL (SKIP TO NEXT PAGE); IF NOT EQUAL, FALL THRU

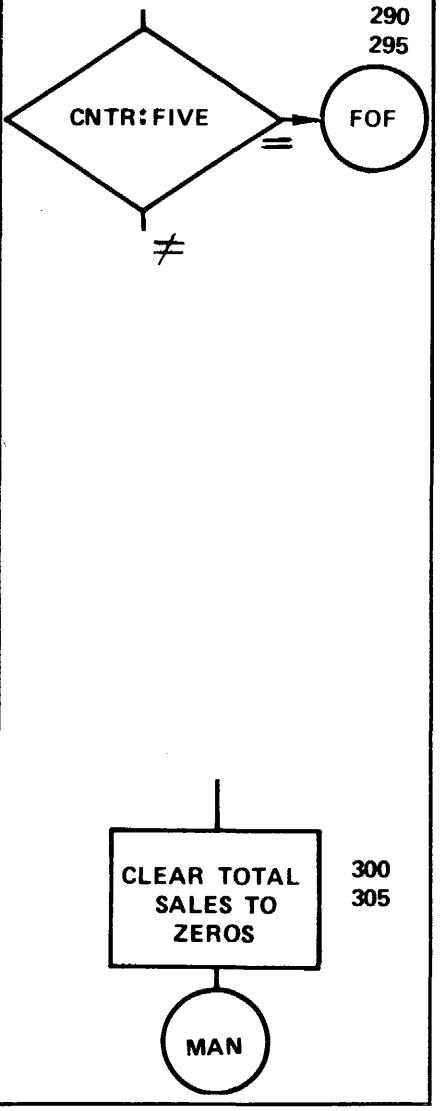
LABEL	OPERATION	OPERAND
	BC	8,FOF

If equal (25 lines have been printed), branches to a subroutine labelled FOF.

CLEAR OUT OLD DATA, PREPARE NEW DATA FOR PRINTING

LABEL	OPERATION	OPERAND
	MVC	TSAL(6),TZER
	BC	15,MAN

Moves characters found in a location labelled TZER to a location labelled TSAL (six positions) then branches unconditionally to subroutine labelled MAN.

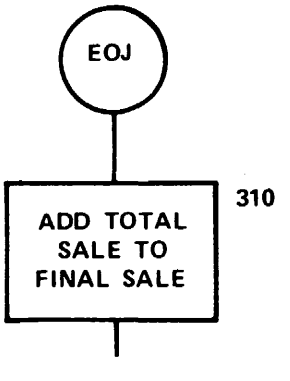


Generally, at this point a routine testing for a /* card would be written. However, the System's IOCS does this for us and the step was therefore omitted.

ADD LAST SALESMAN'S TOTAL TO FINAL TOTAL

LABEL	OPERATION	OPERAND
EOJ	AP	FTOT(8),TSAL(6)

Adds (packed) Total Sales to Final Total and places the result back in FTOT.



SET FORMAT OF PRINTER WORK AREA

```
LABEL    OPERATION    OPERAND
1        10           16
         MVC          PRWK+80(16),MSK1
```

Move MSK1 to printer work area (PRWK + 80, 16 positions).

PLACE TOTAL SALE FOR LAST SALESMAN IN PRWK AREA

```
LABEL    OPERATION    OPERAND
1        10           16
         ED           PRWK+80(16),TSAL
```

Unpacks TSAL and places its contents in PRWK + 80.

PRINT TOTAL SALES LINE

```
LABEL    OPERATION    OPERAND
1        10           16
         PUT          PRNT,PRWK
```

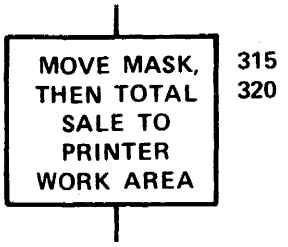
All data placed in PRWK is sent to PRNT and printed.

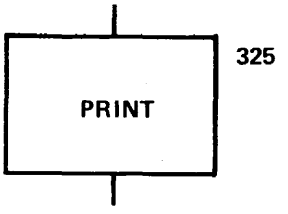
CLEAR PRINTER WORK AREA WITH SPACES

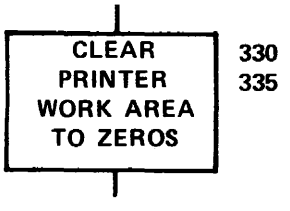
```
LABEL    OPERATION    OPERAND
1        10           16
         MVI          PRWK,X'40'
         MVC          PRWK+1(131),PRWK
```

Moves a space to the first position of Printer Work Area (PRWK).

Moves a space from first position of Printer Work area to the next 131 consecutive locations.







FORMAT PRINTER WORK AREA AND PLACE FINAL TOTAL IN PRWK

```
LABEL    OPERATION    OPERAND
1        10           16
         MVC          PRWK+75(21),MSK2
         ED           PRWK+75(21),FTOT
```

Moves information from MSK2 to PRWK + 75. Unpacks FTOT and places information in PRWK + 75 (21 positions). Moves in final total to occupy positions specified by the mask.

MOVE HEADER FOR FINAL TOTAL INTO PRINTER WORK AREA

```
LABEL    OPERATION    OPERAND
1        10           16
         MVC          PRWK+56(11),HDR4
```

Moves HDR4 (final total) into PRWK + 56.

ADVANCE SINGLE LINE

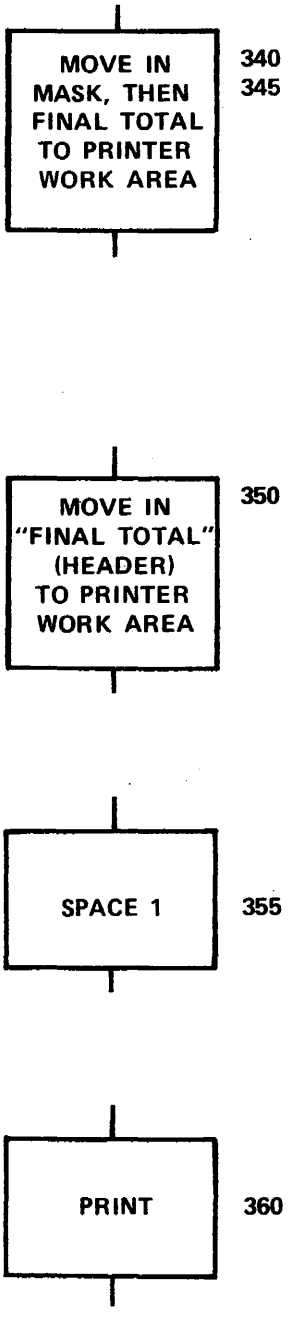
```
LABEL    OPERATION    OPERAND
1        10           16
         CNTRL        PRNT,SP,1
```

Causes the printer to advance one line before printing next line.

PRINT

```
LABEL    OPERATION    OPERAND
1        10           16
         PUT          PRNT,PRWK
```

All data placed in PRWK sent to the printer.



CLEAR PRINTER WORK AREA WITH SPACES

LABEL	OPERATION	OPERAND
	MVI	PRWK,X'40'
	MVC	PRWK+1(131),PRWK

Moves a blank into first position of the printer work area.

Moves blanks into the next 131 positions.

MOVE HEADER TO PRINTER WORK AREA

LABEL	OPERATION	OPERAND
	MVC	PRWK+51(29),HDR5

Moves HDR5 (End of report for fiscal year) into PRWK + 51 (29 positions).

PRINT

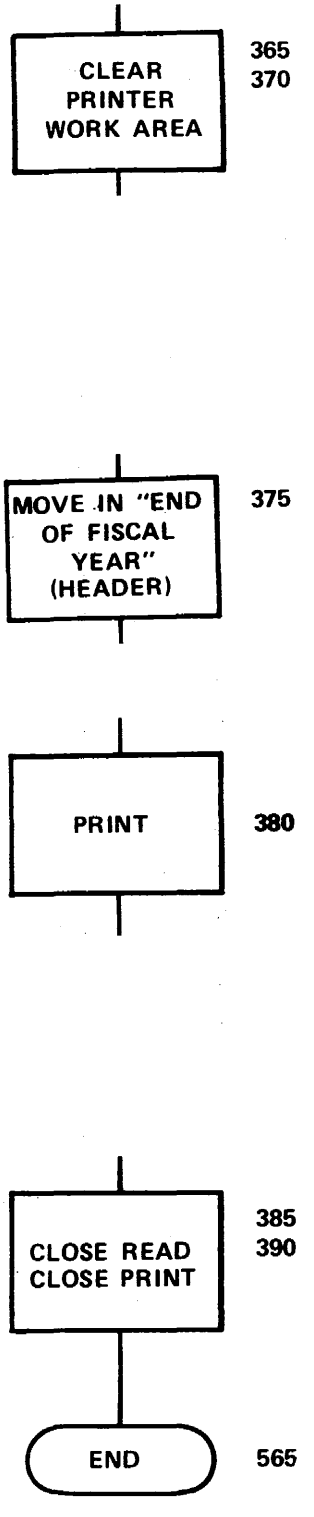
LABEL	OPERATION	OPERAND
	PUT	PRNT,PRWK

All data placed in PRWK sent to PRNT and printed.

DEALLOCATE READER AND PRINTER

LABEL	OPERATION	OPERAND
	CLOSE	READ
	CLOSE	PRNT
	HPR	X'1FFF'

Informs the system that it no longer needs reader and printer, closes READ and PRNT file and displays 1FFF on the console to let operator know that program has terminated normally.



Review the coding forms and flowchart found on the next few pages then complete the Diagnostic Exercise found on the page following the flowchart.

SPERRY UNIVAC **ASSEMBLER CODING FORM** UNIVAC 9000 SERIES

PROGRAM MACRO CALL CARDS PROGRAMMER _____ DATE _____ PAGE ___ OF ___ PAGES

LABEL (1)	OPERATION (10)	OPERAND (16)	COMMENTS	(72)	(80)
READ	DTFCR	EOFA=EOJ,		X	
		IOA1=RBUF,		X	
		ITBL=TBRD,		X	
		MODE=TRANS			
PRNT	DTFPR	BKSZ=132,		X	
		CNTL=YES,		X	
		PRAD=2,		X	
		PROV=FOF,		X	
		FONT=63			
	END				

UD1-1548

SPERRY UNIVAC

ASSEMBLER CODING FORM

UNIVAC 9000 SERIES

PROGRAM USER PROGRAM PROGRAMMER _____ DATE _____ PAGE 1 OF 4 PAGES

LABEL	OPERATION	OPERAND	COMMENTS	
	START	Ø		1ØØ
	USING	*,Ø		1Ø5
	USING	*,1		11Ø
	EXTRN	READ		115
	EXTRN	PRNT		12Ø
	ENTRY	RBUF		125
	ENTRY	EOJ		13Ø
	ENTRY	EOF		135
BEGN	OPEN	READ	GET READ CARD	14Ø
	OPEN	PRNT		145
EOF	MVC	CNTR,TZER+4		15Ø
	MVI	PRWK,X'4Ø'		155
	MVC	PRWK+1(131),PRWK		16Ø
	CNTRL	PRNT,SK,7		165
	AP	PAGE(1),ONE(1)		17Ø
	MVC	PSAL+22(2),MSK3		175
	ED	PSAL+22(2),PAGE		18Ø
	MVC	PEMP(15),HDR1		185
	MVC	PSMN(8),HDR2		19Ø
	MVC	PSAL(5),HDR3		195
	CNTRL	PRNT,SP,1		2ØØ
	PUT	PRNT,PRWK		2Ø5
	GET	READ,CARD		21Ø
MAN	MVI	PRWK,X'4Ø'		215
	MVC	PRWK+1(131),PRWK		22Ø

UD1-1548

SPERRY UNIVAC **ASSEMBLER CODING FORM** UNIVAC 9000 SERIES

PROGRAM USER PROGRAM PROGRAMMER DATE PAGE 24 OF PAGES

LABEL (1)	OPERATION (10)	OPERAND (16)	COMMENTS (72)	(80)
	MVC	PRWK+35(6),EMPN		225
	MVC	PRWK+56(15),SMAN		230
	MVC	SAVE(6),EMPN		235
MIN	PACK	PAKS(4),SALE(6)		240
	AP	TSAL(6),PAKS(4)		245
	GET	READ,CARD		250
	CLC	SAVE,EMPN		255
	BC	8,MIN		260
	AP	FTOT,TSAL		265
	MVC	PRWK+80(16),MSK1		270
	ED	PRWK+80(16),TSAL		275
	PUT	PRNT,PRWK		280
	AP	CNTR,ONE		285
	CP	CNTR,FIVE		290
	BC	8,FOF		295
	MVC	TSAL(6)TZER		300
	BC	15,MAN		305
EOJ	AP	FTOT(8),TSAL(6)		310
	MVC	PRWK+80(16),MSK1		315
	ED	PRWK+80(16),TSAL		320
	PUT	PRNT,PRWK		325
	MVI	PRWK,X'40'		330
	MVC	PRWK+1(131),PRWK		335
	MVC	PRWK+75(21),MSK2		340
	ED	PRWK+75(21),FTOT		345

UD1-1548

SPERRY UNIVAC

ASSEMBLER CODING FORM

UNIVAC 9000 SERIES

PROGRAM USER PROGRAM PROGRAMMER DATE PAGE 3 OF 4 PAGES

LABEL (1)	OPERATION (10)	OPERAND (16)	COMMENTS (72)	(80)
	MVC	PRWK+56(11),HDR4		350
	CNTRL	PRNT,SP,1		355
	PUT	PRNT,PRWK		360
	MVI	PRWK,X'40'		365
	MVC	PRWK+1(131),PRWK		370
	MVC	PRWK+51(29),HDR5		375
	PUT	PRNT,PRWK		380
	CLOSE	READ		385
	CLOSE	PRNT		390
	HPR	X'1FFF'		395
RBUF	DS	CL80	RESERVED FOR READER BUFFER	400
CARD	DS	ØCL80	Ø ALLOWS CARD AREA TO BE SUBDIVIDED	405
EMPN	DS	CL6	RESERVED FOR EMPLOYEE NUMBER	410
SMAN	DS	CL15	RESERVED FOR NAME OF SALESMAN	415
SALE	DS	CL6	RESERVED FOR $ AMOUNT OF SALE	420
	DS	CL53	UNUSED CARD FIELD	425
PRWK	DS	ØCL132	Ø ALLOWS PRINT WORK AREA SUBDIVISION	430
	DS	CL30	BLANK SPACES	435
PEMP	DS	CL15	RESERVED FOR "EMPLOYEE NUMBER" HEADER	440
	DS	CL14	BLANK SPACES	445
PSMN	DS	CL8	RESERVED FOR "SALESMAN" HEADER	450
	DS	CL21	BLANK SPACES	455
PSAL	DS	CL5	RESERVED FOR "SALES" HEADER	460
	DS	CL39	BLANK SPACES	465
HDR1	DC	C'EMPLOYEE NUMBER'		470

UD1-1548

SPERRY UNIVAC ASSEMBLER CODING FORM UNIVAC 9000 SERIES

PROGRAM USER PROGRAM PROGRAMMER DATE PAGE 4 OF 4 PAGES

LABEL	OPERATION	OPERAND	COMMENTS	
HDR2	DC	C'SALESMAN'		475
HDR3	DC	C'SALES'		480
HDR4	DC	C'TOTAL SALES'		485
HDR5	DC	C'END OF REPORT FO'		490
	DC	C'R FISCAL YEAR'		495
TZER	DC	X'ØØØØØØØØØØØØC'		500
CNTR	DC	X'ØØØC'	COUNTER FOR LINES	505
PAGE	DC	X'ØC'	PAGE NUMBER	510
SAVE	DS	CL6	RESERVED FOR EMPLOYEE NO. COMPARISON	515
ONE	DC	X'1F'	STEP UP PAGE NUMBER	520
TSAL	DC	XL6'ØC'	SALES TOTAL PER INDIVIDUAL	525
FTOT	DC	XL8'ØC'	FINAL TOTAL ALL SALES	530
FIVE	DC	X'025C'		535
PAKS	DS	CL4	RESERVED FOR PACKED $ SALES	540
MSK1	DC	X'406B2020206B2020206B2020214B2020'		545
MSK2	DC	X'40206B2020206B2020206B2020206B20'		550
	DC	X'20214B2020'		555
MSK3	DC	X'4020'		560
	END	BEGN	OPERAND IS ADDRESS OF FIRST EXECUTABLE INSTRUCTION	565

UD1-1548

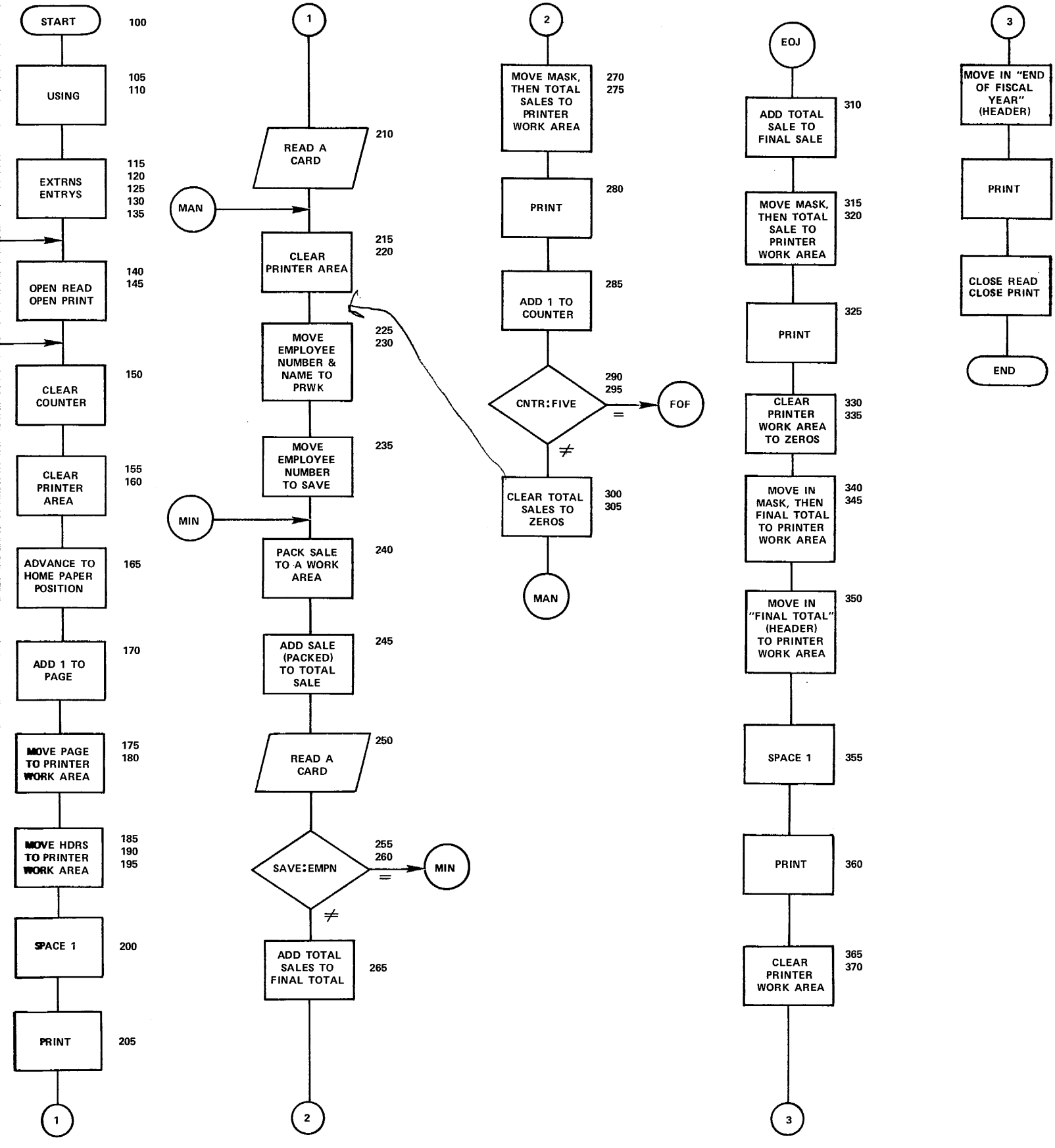


Figure 3-3 Marketing Sales Report Flowchart

DIAGNOSTIC EXERCISE

WORKSESSION – MARKETING SALES REPORT PROBLEM

Circle answers below, then check with the correct answer on the next page.

The following questions refer to the flowchart and the following constant values are assumed:

PAGE = Ø

COUNTER = Ø

1. When is the counter incremented?

   a. After each line of data is printed
   b. After each page of data is printed
   c. Before the first line of data is printed
   d. Before each heading line is printed

2. When is the printer work area cleared? (Select two correct responses.)

   a. Before each line of data is moved into PRWK
   b. Before each line of data is printed
   c. Before each heading line is printed
   d. Before each heading line is moved into PRWK

3. When does page change occur?

   a. After 5 lines of printing
   b. After 25 lines of printing
   c. After 65 lines of printing
   d. After 26 lines of printing

4. When is the FOF routine bypassed?

   a. When CNTR = FIVE(25)
   b. When CNTR ≠ FIVE(25)

5. When is the EOJ routine executed?

   a. After /* is READ
   b. After the last card is READ
   c. After a blank card is READ

6. What is the value of counter after 30 lines of data has been printed?

   a. 2
   b. 5
   c. 29
   d. 0
   e. 31

7. Which block in the flowchart represents the function that starts the header subroutine?

   a. Block 150
   b. Block 165
   c. Block 185
   d. Block 215

DIAGNOSTIC EXERCISE ANSWERS

1. a
2. a, d
3. b
4. b
5. a
6. b
7. a

OPERATING PROCEDURES

INTRODUCTION

Now that you have completed the talk-thru and the diagnostic exercise for the Marketing Sales Report Program, the operating procedures presented in this book will simulate the processing of that Marketing Sales Report program. The Univac-supplied programs used by the programmer at assembly time will be described. How to build an input "job stream" will be illustrated and the operating procedure required to obtain the desired output from the computer will be simulated. To present the material in its simplest form, this section will include only the basic operating procedures. The back of this section contains a printout listing the complete coding for the Marketing Sales Report Program, Linker Map and the output from a sample production run.

When the program coding is completed, the information on the coding forms is punched on cards thereby producing two decks of cards: the DTF statement cards and the main source program cards. The DTF cards are processed by a Preassembly Macro Pass program provided by Univac. The output of the Preassembly Macro Pass program is combined with the user Source Code program and processed by the Assembler Program. The output of the Assembler program is processed by the Linker Program.

Preassembly Macro Pass Program

The Preassembly Macro Pass Program generates the source code for the DTF statements which define the input/output devices the user accesses at program run time. The output of the Preassembly Macro Pass, is combined with the user's source code program and assembled.

Assembler Program

The Two-Pass Assembler converts source code programs (user programs) to machine code (object programs). The assembler produces an object card deck and a printout that lists the source code and the object code generated by the source deck. The output of the assembler is the input to the Linker Program.

Linker Program

The purpose of the Linker Program is to combine the object programs (card reader, printer, user program) into a single object program. The output is an executable object program.

PREASSEMBLY MACRO PASS GENERATOR PROGRAM DATA PREPARATION

The Preassembly Macro Pass program generates the IOCS source code for the peripheral devices accessed by the user program. The input flow is set up as follows:

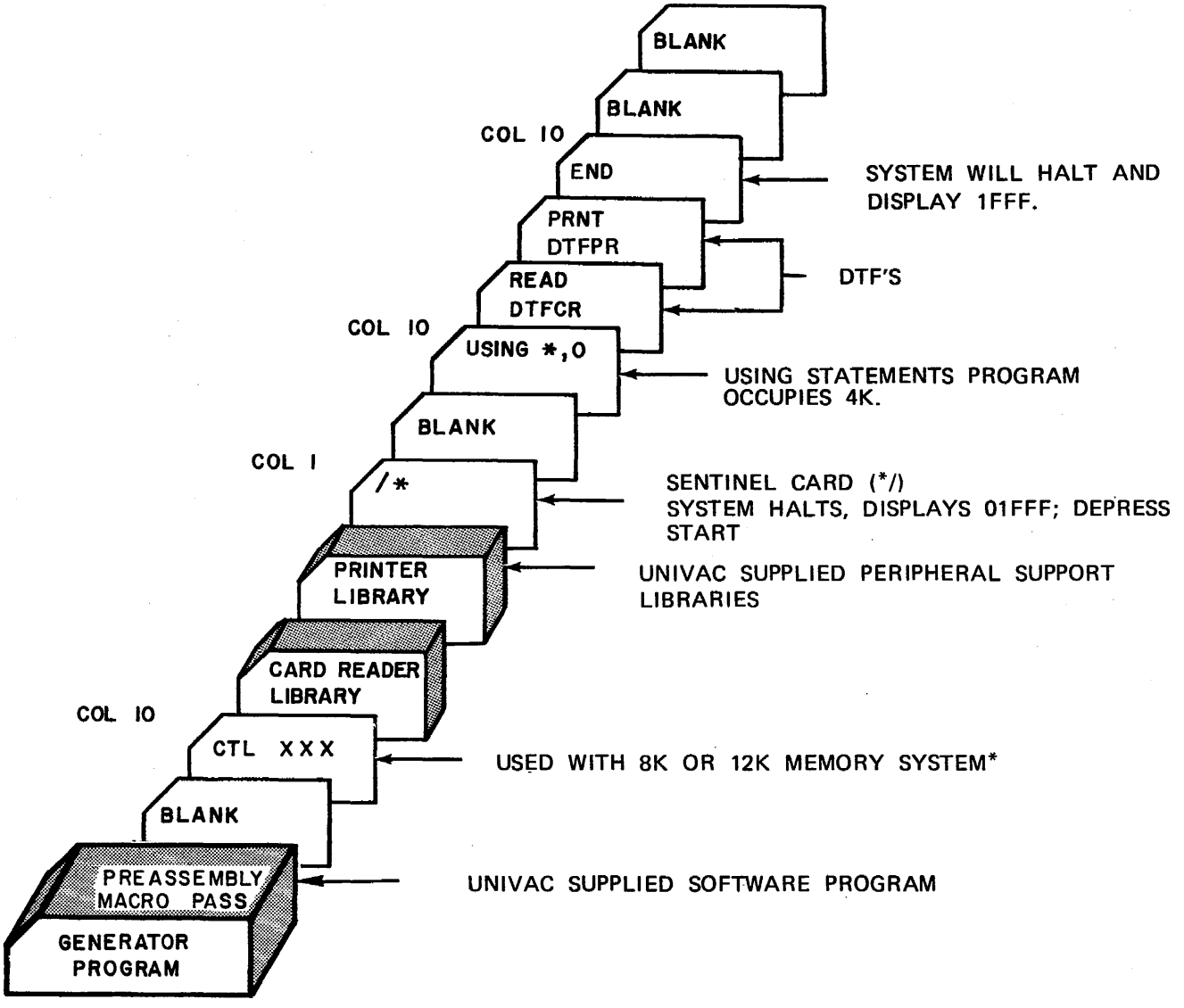


Figure 3–4 Preassembly Input Stream

* Contains the decimal number equal to the highest available memory address beginning in column 16 (8191 for 8K - 12,287 for 12K system). If "CTL" is omitted 16,383 (16K) will be assumed.

PREASSEMBLY MACRO PASS GENERATOR PROGRAM OPERATING PROCEDURES

Unfold the control panel illustration on page 3-39. The buttons on the control panel used in operating the Preassembly Macro Pass Generator Program are numbered on the control panel illustration. As the operating procedure is outlined, simulate the operation by locating the appropriate buttons on the control panel illustration.

1. Load cards (see figure 3–4) in card reader, row 9 edge leading, face down.
2. On the control panel, depress PROC CLEAR button (8).
3. Depress CHANNEL CLEAR button (7).
4. Depress CLEAR PRINTER button (1).
5. Depress CLEAR READER button (2).
6. Depress FEED READER button (3).
7. Depress LOAD button ON (4).
8. Depress RUN/START button (6).
9. Depress LOAD button OFF (4).
10. Depress RUN/START button (8).
11. After LIBRARY is read, machine will HALT and display X' 01FF' on NEXT INSTRUCTION/HALT INDICATOR LAMPS (a lighted lamp indicates a binary 1.)
12. Depress RUN/START button (7) on control panel.
13. Final HALT display is X'1FFF'.

The Punch output stacker should now contain DTF source code cards ready for assembly.

TWO–PASS CARD ASSEMBLY DATA PREPARATION

Remove the END card from the DTF source code decks. Place user program START card in front of the deck. Place the user source code deck behind the DTF source code deck, make sure the last card is an END card.

The "control stream" is constructed as follows:

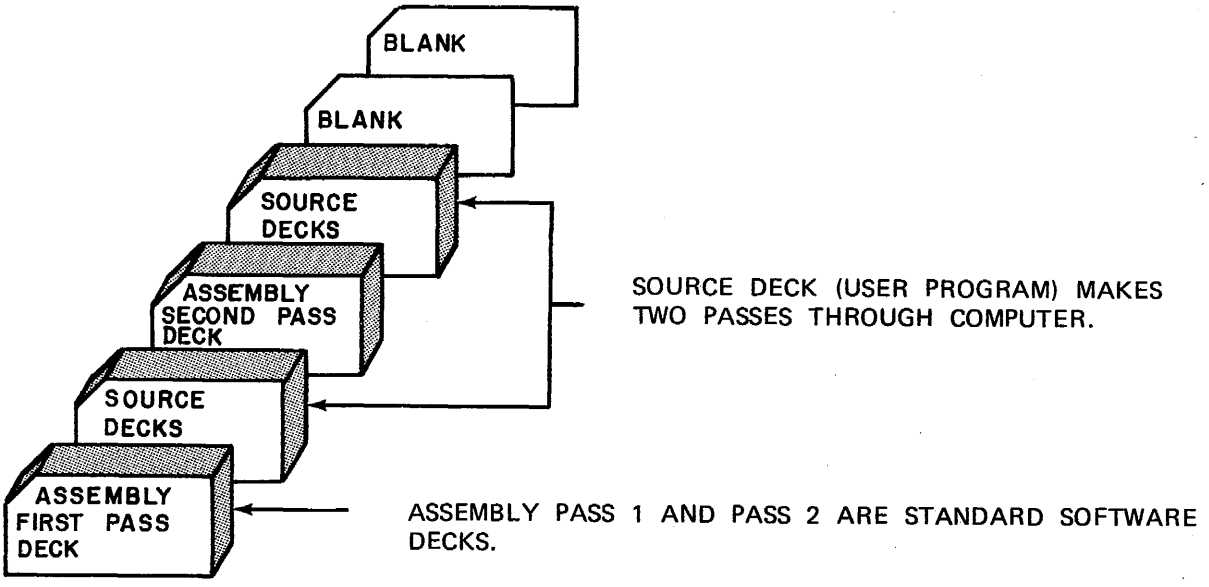


Figure 3–5 Two-Pass Card Assembly Control Stream

In the above example, the DTF source code and user program are assembled together. The user may choose to assemble the DTF source code and the user program separately.

CARD ASSEMBLER OPERATING PROCEDURE

Unfold the control panal illustration on page 3-39. The buttons on on the control panel used in operating the assembly are numbered on the illustration. As the operating procedure is outlined, simulate the operation by locating the appropriate buttons on the control panel illustration.

1. Load cards (see figure 3–5) in the card reader row 9 edge leading, face down.
2. On the control panel, depress PROC CLEAR button (8).
3. Depress CHANNEL CLEAR button (7).
4. Depress CLEAR PRINTER button (1).
5. Depress CLEAR READER button (2).
6. Depress FEED CARD button (3).
7. Depress LOAD button ON (4).
8. Depress RUN/START button (6).
9. Depress LOAD button OFF (4).
10. Depress RUN/START button (6).
11. After the first few cards of Assembler Pass 2 have been read stop the processor by depressing INST button (5) on control panel.
12. Take USING, SOURCE and END CARDS from reader output stacker and place them on top of remaining cards in reader input hopper; follow by 2 blank cards.
13. Depress INST button (5) on control panel.
14. Depress START.
15. Assembler Listing will be printed.
16. Punch output stacker will contain Object code cards for the Linker pass.

TWO–PASS LINKER PROGRAM DATA PREPARATION

The Linker combines the output of the DTF and user program assembly. The input "control stream" for the Linker is as follows:

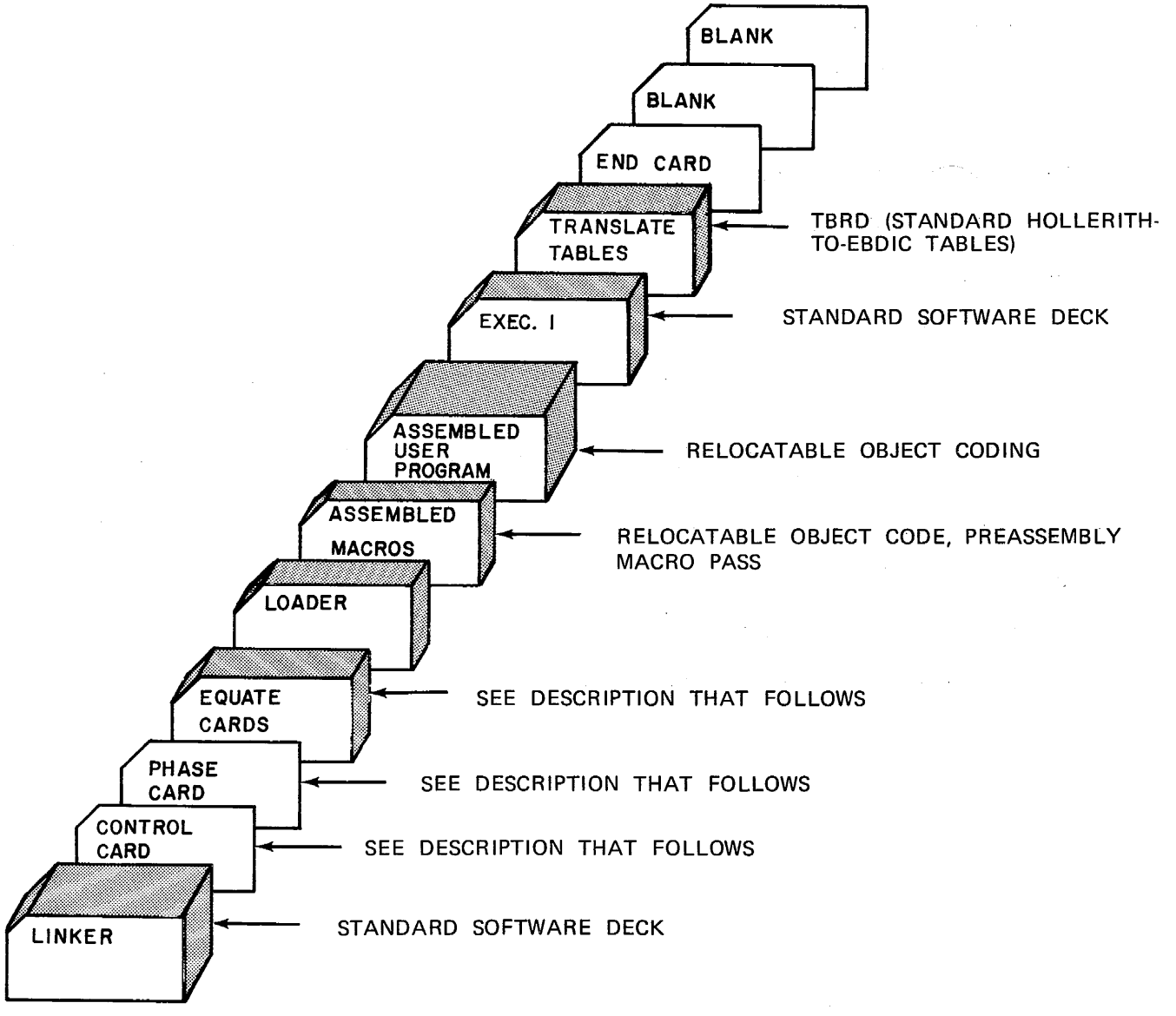


Figure 3–6 Two-Pass Linker Control Stream

CONTROL CARD

	10	16
Format:	CTL	n, p, q

n = 1 (one-pass)
n = 2 (two-pass). If n is blank, 1 pass will be assumed.
p = decimal number of the largest address available on computer doing the linking. If p remains blank, 16,383 bytes will be assumed.
q = decimal number of the highest memory address available for use in program execution. If q remains blank, 16,383 bytes will be assumed. Commas must be punched as specified.

	10	16
Example:	CTL	2,8191,8191

The following illustrates a control card prepared for a two-pass run on a 16K memory system.

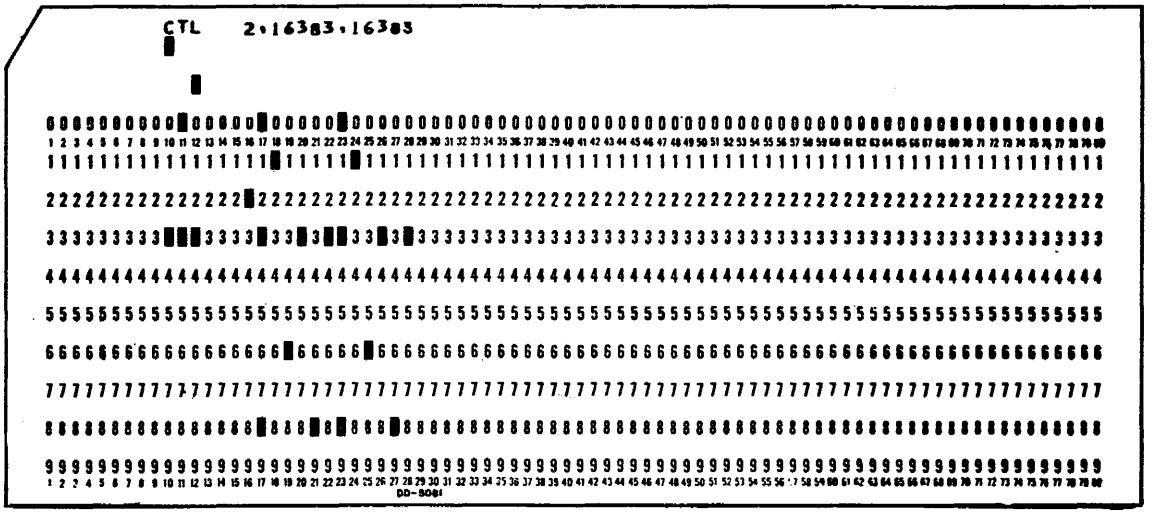


Figure 3–7 Control Card Format

Phase Card

The phase card indicates the name of the object program, so that it will be punched into each card of the final object program deck. Phase name must be the same as operand in the START card of the user program.

Format:

The following illustrates a phase card prepared for a program named PGM.

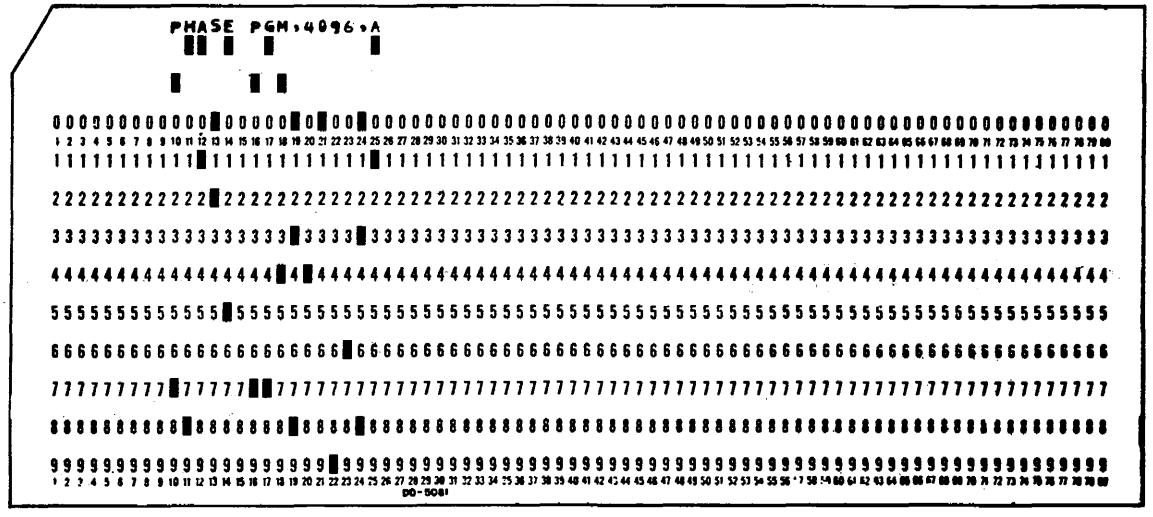


Figure 3–8 Phase Card Format

Standard Loader Equate Card Statements

L?CH	EQU	X'40'	Fill character to be inserted in the area cleared by the card loader.
L?AM	EQU	4	Permits alteration of memory specified by the memory address switches.
L?AR	EQU	128	Start of the read area for the card loader.
L?PG	EQU	208	Start of the object code for the card loader.
L?LO	EQU	128	First memory location to be cleared by the card loader.
L?HI	EQU	16383	Last memory location to be cleared by the card loader.

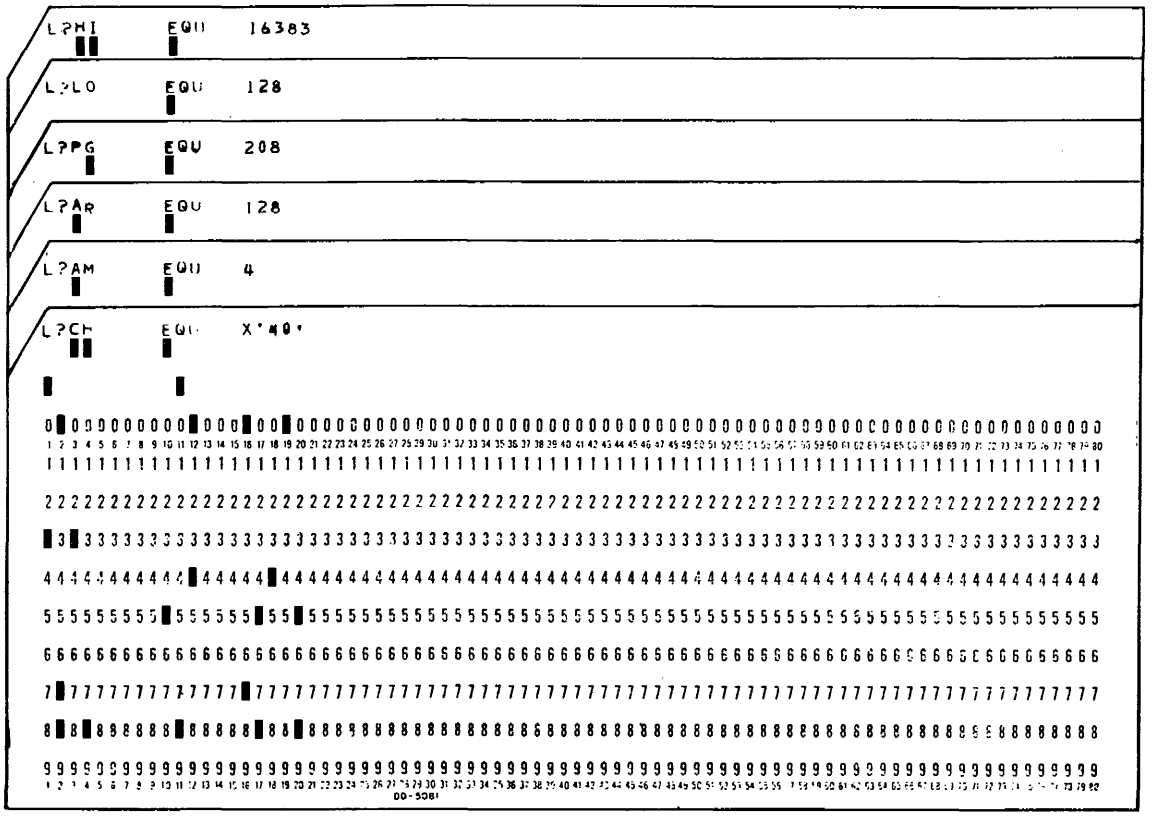


Figure 3–9 Equate Cards

LINKER OPERATING PROCEDURES

Unfold control panel illustration on page 3-39. All the buttons on the control panel used in the Linker run are numbered on the illustration. As the operating procedure is outlined, simulate the operation by locating the appropriate buttons on the control panel.

1. Load cards (see figure 3–6) in the card reader, row 9 edge leading, face down.
2. On the control panel, depress PROC CLEAR button (8).
3. Depress CHANNEL CLEAR button (7).
4. Depress CLEAR PRINTER button (1).
5. Depress CLEAR READER button (2).
6. Depress FEED CARD button (3).
7. Depress LOAD button on (4).
8. Depress RUN/START button (6).
9. Depress LOAD button OFF (4).
10. Depress RUN/START button (6).
11. After first pass, remove Linker object deck from file and place remaining cards in input hopper a second time (last pass).
12. Depress FEED READER button (3) on control panel.
13. Depress RUN/START (6) on control panel.

The Linker listing will be printed during the last pass.

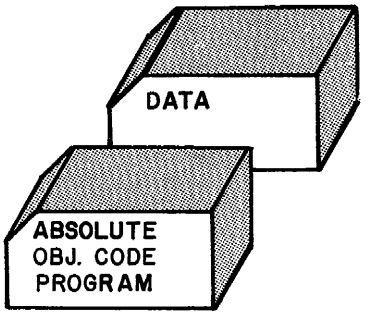
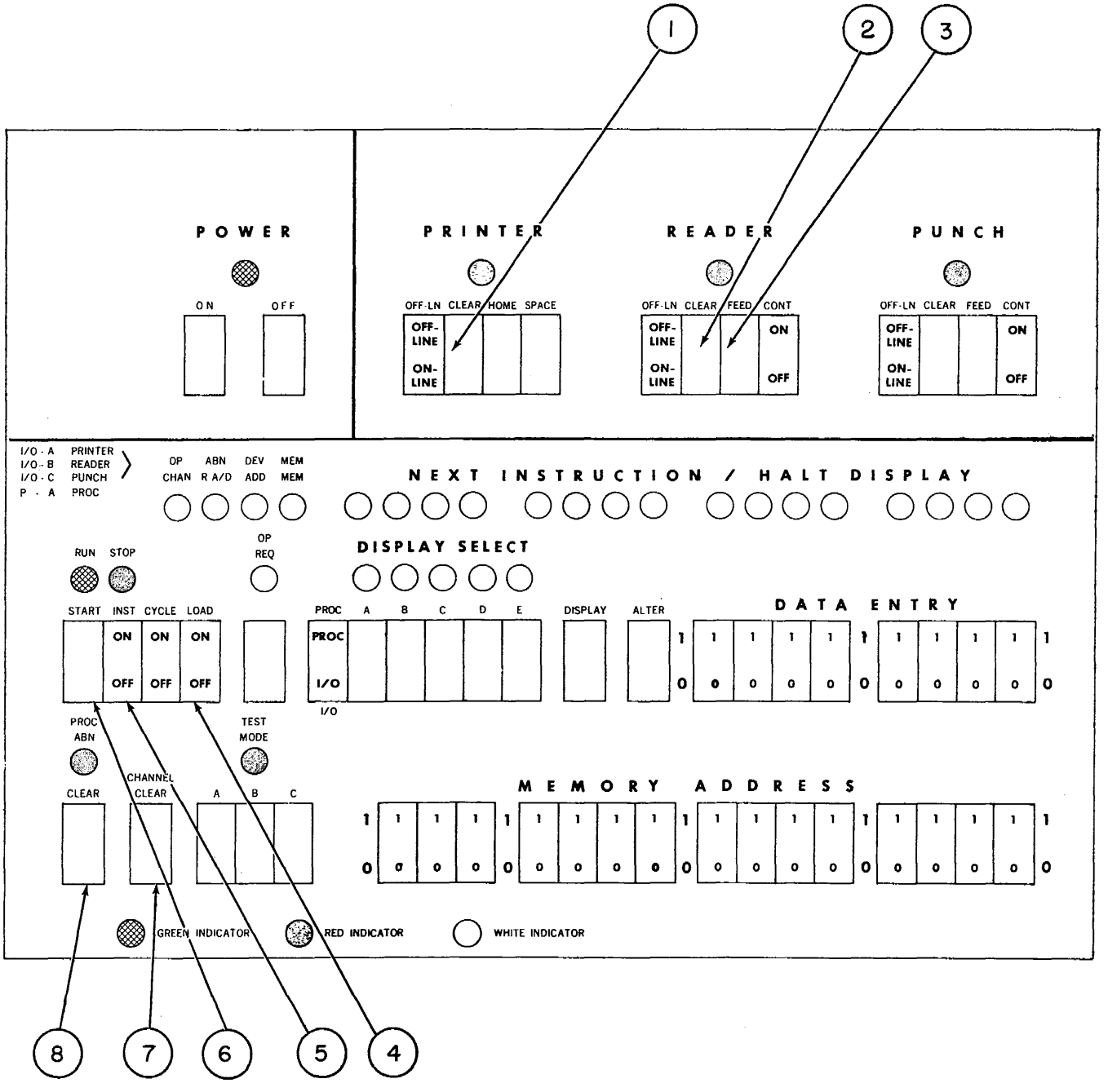


Figure 3–10 Production Run Card Input

1. Place above cards in Reader, row 9 edge leading, face down.
2. On the control panel, depress PROC CLEAR button (8).
3. Depress CHANNEL CLEAR button (7).
4. Depress CLEAR PRINTER button (1).
5. Depress CLEAR READER button (2).
6. Depress FEED READER button (3).
7. Depress LOAD button (4) ON.
8. Depress RUN/START button (6).
9. Depress LOAD button (4) OFF.
10. Depress RUN/START button (6).

Final display is X'1FFF'.

The output is printed data.



The circled numbers identify the buttons used by the operator.

Figure 3–11 Control Panel

```
                9200 CARD ASSEMBLY
          PRINTED ASSEMBLY CODES:
          C COVER ERROR, NO USING COVERING RELOCATABLE OPERAND ADDRESS
          D DOUBLY DEFINED LABEL OR REFERENCE TO DOUBLY DEFINED LABEL
          E EXPRESSION TOO LARGE OR IMPROPER SYNTAX
          H HALF WORD BOUNDARY ERROR ON RX OR AI OPERAND
          I INSTRUCTION ERROR
          L LOCATION COUNTER TOO LARGE
          O ORG ERROR, 2ND DEFINITION OF A LABEL
          R RELOCATABLE TERMS IN THE EXPRESSION ARE IMPROPER OR TOO MANY
          S SEQUENCE BREAK IN COLUMS 76 TO 80
          T TRUNCATION OF OVERSIZE TERM
          U UNDEFINED LABEL REFERENCED IN THIS LINE
          X CONTINUATION,(NONBLANK COL 72 ON NONCOMMENT CARD)NOT PERMITTED

 0001                                        START 0
 0002                                        USING *,0
 0003                                        USING *,1
 0004                                *READ   DTFCR EOFA=EOJ,
 0005                                *         IOA1=RBUF,
 0006                                *         ITBL=TBRD,
 0007                                *         MODE=TRANS
 0008                                * RDR IOCS (CARD)                              001 *
 0009                                        ENTRY READ                             002 *
 0010                                        EXTRN EOJ                              003 *
 0011                                        EXTRN RBUF                             004 *
 0012                                        EXTRN TBRD                             005 *
 0013   0000 47F0009C                READ   BC    15,A?OP                           006 *
 0014   0004 47F00094                       BC    15,A?CL                           007 *
 0015   0008 45F00038                A?GT   BAL   15,A?WT                           008 *
 0016   000C 48F0E000                       LH    15,0(,14)                         009 *
 0017   0010 D24FF000030C                    MVC   0(80,15),RBUF                    010 *
```

```
0018  0016 DC4FF0000000          TR    0(80,15),TBRD                    011 *
0019  001C 9534030C              CLI   RBUF+0+0-0,52                    012 *
0020  0020 4780002C              BC    8,**+12                          013 *
0021  0024 45F00072              BAL   15,A?XF                          014 *
0022  0028 47F0E002              BC    15,2(,14)                        015 *
0023  002C 45F00072              BAL   15,A?XF                          016 *
0024  0030 45F00038              BAL   15,A?WT                          017 *
0025  0034 47F002A8              BC    15,EOJ                           018 *
0026  0038 A501005F      A?WT    TIO   A?DS+5,1                         019 *
0027  003C 47300038              BC    3,A?WT                           020 *
0028  0040 91F3005F              TM    A?DS+5,243                       021 *
0029  0044 4740005A              BC    4,A?DS                           022 *
0030  0048 9108005F              TM    A?DS+5,8                         023 *
0031  004C 4780F000              BC    8,0(,15)                         024 *
0032  0050 D501004600A8          CLC   70(2),A?ST                       025 *
0033  0056 4720F000              BC    2,0(,15)                         026 *
0034  005A A108000061004000 A?DS MSG   24832                            027 *
0035  0062 40F000A6              STH   15,A?CO+2                        028 *
0036  0066 45F0CD72              BAL   15,A?XF                          029 *
0037  006A 48F000A6              LH    15,A?CO+2                        030 *
0038  006E 47F00038              BC    15,A?WT                          031 *
0039  0072 9250C045      A?XF    MVI   69,80                            032 *
0040  0076 D201004600A4          MVC   70(2),A?CO                       033 *
0041  007C A4010012              XIOF  18+0,1                           034 *
0042  0080 4780F000              BC    8,0(,15)                         035 *
0043  0084 A501008D              TIO   A?DI+5,1                         036 *
0044  0088 A108000061004000 A?DI MSG   24832                            037 *
0045  0090 47F0007C              BC    15,A?XF+10                       038 *
0046  0094 45F00038      A?CL    BAL   15,A?WT                          039 *
0047  0098 47F0E000              BC    15,0(,14)                        040 *
0048  009C 45F00072      A?OP    BAL   15,A?XF                          041 *
0049  00AC 47F0E000              BC    15,0(,14)                        042 *
0050  00A4 030C          A?CO    DC    Y(RBUF)                          043 *
```

```
0051  00A6                          DS    CL2                                  044 *
0052  00A8 0358              A?ST   DC    Y(RBUF+80-1)                         045 *
0053  00AA                          ORG   *                                    046 *
0054  00AA F4                      DC    C'4'                                 047 *
0055                         *PRNT  DTFPR BKSZ=132,
0056                         *             CNTL=YES,
0057                         *             PRAD=2,
0058                         *             PROV=FOF,
0059                         *             FONT=63
0060                         * PRINT IOCS (CARD)                              001 *
0061                              ENTRY PRNT                                   002 *
0062                              ENTRY B?SH                                   003 *
0063                              EXTRN FOF                                    004 *
0064  00AC 47F0013A          PRNT   BC    15,B?OP                              005 *
0065  00B0 47F00154                 BC    15,B?CL                              006 *
0066  00B4 47F00084                 BC    15,*                                 007 *
0067  00B8 47F000FE                 BC    15,B?CN                              008 *
0068                         * PUT                                           009 *
0069  00BC 45F00188                 BAL   15,B?VE                              010 *
0070  00C0 48F0E000                 LH    15,0(,14)                            011 *
0071  00C4 D2830080F000             MVC   128(132),0(15)                       012 *
0072  00CA 920001E1          B?SH   MVI   B?SE+1,0                             013 *
0073  00CE 920200C8                 MVI   B?SH+1,2                             014 *
0074  00D2 D20001CF01E4             MVC   B?K+3(1),B?X1                        015 *
0075  00D8 D2000005001E1     B?IS   MVC   80(1),B?SE+1                         016 *
0076  00DE 45F001C8                 BAL   15,B?XF                              017 *
0077  00E2 47F0E002          B?D    BC    15,2(,14)                            018 *
0078  00E6 92F000E3                 MVI   B?D+1,240                            019 *
0079  00EA AAE001E2                 AH    14,B?H                               020 *
0080  00EE 47F001EE                 BC    15,FOF                               021 *
0081  00F2 45F00188          B?F    BAL   15,B?VE                              022 *
0082  00F6 921301CF                 MVI   B?K+3,19                             023 *
0083  00FA 47F00008                 BC    15,B?IS                              024 *
```

```
0084                                  * CONTROL                                    025 *
0085  00FE                            B?CN EQU   *                                 026 *
0086  00FE 9540E003                         CLI   3(14),64                        027 *
0087  0102 47800118                         BC    8,B?D6                          028 *
0088  0106 D20000C8E003                      MVC   B?SH+1(1),3(14)                 029 *
0089  010C 95D7E001                         CLI   1(14),C'P'                      030 *
0090  0110 47800118                         BC    8,B?D6                          031 *
0091  0114 960800C8                         OI    B?SH+1,8                        032 *
0092  0118 913FE002                   B?D6  TM    2(14),63                        033 *
0093  011C 4780E004                         BC    8,4(,14)                        034 *
0094  0120 D20001E1E002                      MVC   B?SE+1(1),2(14)                 035 *
0095  0126 95D7E001                         CLI   1(14),C'P'                      036 *
0096  012A 47800132                         BC    8,B?G                           037 *
0097  012E 960801E1                         OI    B?SE+1,8                        038 *
0098  0132 AAE001F2                   B?G   AH    14,B?H                          039 *
0099  0136 47F000F2                         BC    15,B?F                          040 *
0100                                  * OPEN                                       041 *
0101  013A 92400080                   B?OP  MVI   128,64                          042 *
0102  013E D28200810080                      MVC   129(131),128                    043 *
0103  0144 920200C8                         MVI   B?SH+1,2                        044 *
0104  0148 92F0019D                         MVI   B?C+1,240                       045 *
0105  014C 92F000E3                         MVI   B?D+1,240                       046 *
0106  0150 47F0E000                         BC    15,0(,14)                       047 *
0107                                  * CLOSE                                      048 *
0108  0154 45F00188                   B?CL  BAL   15,B?VE                         049 *
0109  0158 47F0E000                         BC    15,0(,14)                       050 *
0110                                  * ERROR                                      051 *
0111  015C A5030195                   B?E   TIO   B?DI+5,3                        052 *
0112  0160 4730015C                         BC    3,B?E                           053 *
0113  0164 91FB0195                         TM    B?DI+5,251                      054 *
0114  0168 4780019C                         BC    8,B?C                           055 *
0115  016C 91010195                   B?ER  TM    B?DI+5,1                        056 *
0116  0170 47800178                         BC    8,*+8                           057 *
```

```
0117  0174 92000190          MVI   B?C+1,0                    058 *
0118  0178 91020195  B?M     TM    B?DI+5,2                   059 *
0119  017C 47800184          BC    8,B?I                      060 *
0120  0180 920000E3          MVI   B?D+1,0                    061 *
0121  0184 91F80195  B?I     TM    B?DI+5,248                 062 *
0122  0188 4780F000          BC    8,D(,15)                   063 *
0123  018C 94FC0195          NI    B?DI+5,252                 064 *
0124  0190 A10800006300400D  B?DI  MSG   25344                 065 *
0125  0198 47F0F000          BC    15,D(,15)                  066 *
0126  019C 47F00184  B?C     BC    15,B?VX                    067 *
0127  01A0 950F0050          CLI   80,15                      068 *
0128  01A4 47600184          BC    6,B?VX                     069 *
0129  01A8 92F00190          MVI   B?C+1,240                  070 *
0130  01AC A10800006301400D  MSG   25345                       071 *
0131  01B4 47F00000  B?VX    BC    15,0                       072 *
0132                  * VERIFY                                073 *
0133  01B8 40F001B6  B?VE    STH   15,B?VX+2                  074 *
0134  01BC 45F0015C          BAL   15,B?E                     075 *
0135  01C0 45F001C8          BAL   15,B?XF                    076 *
0136  01C4 47F001BC          BC    15,B?VE+4                  077 *
0137  01C8 40F001D2  B?XF    STH   15,B?N+2                   078 *
0138  01CC A4030000  B?K     XIOF  0,3                        079 *
0139  01D0 47800000  B?N     BC    8,0                        080 *
0140  01D4 A5030195          TIO   B?DI+5,3                   081 *
0141  01D8 45F0016C  B?L     BAL   15,B?ER                    082 *
0142  01DC 47F001CC          BC    15,B?K                     083 *
0143  01E0 00        B?SE    DC    X'00'                      084 *
0144  01E1 C0                DC    X'00'                      085 *
0145  01E2 0002      B?H     DC    Y(2)                       086 *
0146  01E4 11        B?X1    DC    YL1(D+17)                  087 *
0147  01E5                   ORG   *                          088 *
0148  01E5 F4                DC    C'4'                       089 *
0149                          USING *,0
```

```
0150                                          USING *,1
0151                                          EXTRN READ
0152                                          EXTRN PRNT
0153                                          ENTRY RBUF
0154                                          ENTRY EOJ
0155                                          ENTRY FOF
0156      01E6 45E00000               BEGN    OPEN  READ
0157      01EA 45E000AC                       OPEN  PRNT
0158      01EE D201047A0478           FOF     MVC   CNTR,TZER+4
0159      01F4 924003AC                       MVI   PRWK,X'40'
0160      01F8 D28203AD03AC                   MVC   PRWK+1(131),PRWK
0161      01FE 45E00088E2D20740               CNTRL PRNT,SK,7
0162      0206 FA00047C0483                   AP    PAGE(1),ONE(1)
0163      020C D201041A04BD                   MVC   PSAL+22(2),MSK3
0164      0212 DE01041A047C                   ED    PSAL+22(2),PAGE
0165      0218 D20E03CA0430                   MVC   PEMP(15),HDR1
0166      021E D20703E7043F                   MVC   PSMN(8),HDR2
0167      0224 D204C4040447                   MVC   PSAL(5),HDR3
0168      022A 45E00088E2070140               CNTRL PRNT,SP,1
0169      0232 45E0008C03AC                   PUT   PRNT,PRWK
0170      0238 45E00008035C                   GET   READ,CARD
0171      023E 924003AC               MAN     MVI   PRWK,X'40'
0172      0242 D28203AD03AC                   MVC   PRWK+1(131),PRWK
0173      0248 D20503CF035C                   MVC   PRWK+35(6),EMPN
0174      024E D20E03E40362                   MVC   PRWK+56(15),SMAN
0175      0254 D2050470035C                   MVC   SAVE(6),EMPN
0176      025A F23504940371           MIN     PACK  PAKS(4),SALE(6)
0177      0260 FA5304840494                   AP    TSAL(6),PAKS(4)
0178      0266 45E00008035C                   GET   READ,CARD
0179      026C D5050470035C                   CLC   SAVE,EMPN
0180      0272 4780025A                       BC    8,MIN
0181      0276 FA75048A0484                   AP    FTOT,TSAL
0182      027C D20F03FC0498                   MVC   PRWK+80(16),MSK1
```

```
0183   0282 DE0F03FC0484                    ED    PRWK+80(16),TSAL
0184   0288 45E0008C03AC                    PUT   PRNT,PRWK
0185   028E FA10047A0483                    AP    CNTR,ONE
0186   0294 F911047A0492                    CP    CNTR,FIVE
0187   029A 478001EE                        BC    8,FOF
0188   029E D20504840474                    MVC   TSAL(6),TZER
0189   02A4 47F0023E                        BC    15,MAN
0190   02A8 FA75048A0484            EOJ     AP    FTOT(8),TSAL(6)
0191   02AE D20F03FC0498                    MVC   PRWK+80(16),MSK1
0192   02B4 DE0F03FC0484                    ED    PRWK+80(16),TSAL
0193   02BA 45E0008C03AC                    PUT   PRNT,PRWK
0194   02C0 924003AC                        MVI   PRWK,X'40'
0195   02C4 D28203AD03AC                    MVC   PRWK+1(131),PRWK
0196   02CA D21403F704A8                    MVC   PRWK+75(21),MSK2
0197   02D0 DE1403F704BA                    ED    PRWK+75(21),FTOT
0198   02D6 D20A03E404AC                    MVC   PRWK+56(11),HDR4
0199   02DC 45E00088E2D70140                CNTRL PRNT,SP,1
0200   02E4 45E0008C03AC                    PUT   PRNT,PRWK
0201   02EA 924003AC                        MVI   PRWK,X'40'
0202   02EE D28203AD03AC                    MVC   PRWK+1(131),PRWK
0203   02F4 D21C030F0457                    MVC   PRWK+51(29),HDR5
0204   02FA 45E0008C03AC                    PUT   PRNT,PRWK
0205   0300 45E00004                        CLOSE READ
0206   0304 45E00080                        CLOSE PRNT
0207   0308 A9001FFF                        HPR   X'1FFF'
0208   030C                         RBUF    DS    CL80
0209   035C                         CARD    DS    OCL80
0210   035C                         EMPN    DS    CL6
0211   0362                         SMAN    DS    CL15
0212   0371                         SALE    DS    CL6
0213   0377                                 DS    CL53
0214   03AC                         PRWK    DS    OCL132
0215   03AC                                 DS    CL30
```

```
0216  03CA                                  PEMP   DS   CL15
0217  03D9                                         DS   CL14
0218  03E7                                  PSMN   DS   CL8
0219  03EF                                         DS   CL21
0220  0404                                  PSAL   DS   CL5
0221  0409                                         DS   CL39
0222  0430 C5D4D7D3D6E8C5C540D5E4D4C2C5D9  HDR1   DC   C'EMPLOYEE NUMBER'
0223  043F E2C1D3C5E2D4C1D5                HDR2   DC   C'SALESMAN'
0224  0447 E2C1D3C5E2                      HDR3   DC   C'SALES'
0225  044C E3D6E3C1D340E2C1D3C5E2          HDR4   DC   C'TOTAL SALES'
0226  0457 C5D5C440D6C640D9C5D7D6D9E340C6D6 HDR5   DC   C'END OF REPORT FO'
0227  0467 D940C6C9E2C3C1D340E8C5C1D9             DC   C'R FISCAL YEAR'
0228  0474 000000000000C                   TZER   DC   X'000000000000C'
0229  047A 000C                            CNTR   DC   X'000C'
0230  047C 0C                              PAGE   DC   X'0C'
0231  047D                                 SAVE   DS   CL6
0232  0483 1F                              ONE    DC   X'1F'
0233  0484 000000000000C                   TSAL   DC   XL6'0C'
0234  048A 0000000000000000C               FTOT   DC   XL8'0C'
0235  0492 025C                            FIVE   DC   X'025C'
0236  0494                                 PAKS   DS   CL4
0237  0498 4068202020682020206820202148202O MSK1   DC   X'4068202020682020206820202148202O'
0238  04A8 402068202020682020206820202068 20 MSK2   DC   X'402068202020682020206820202068 20'
0239  04B8 20214B2020                            DC   X'20214B2020'
0240  04BD 4020                            MSK3   DC   X'4020'
0241                                       END    BEGN
```

```
LINKER MAP

0000                CTL   2,16383,16383
1000                PHASE PGM,4096,A
0040        L?CH    EQU   X'40'
0004        L?AM    EQU   4
0080        L?AR    EQU   128
00D0        L?PG    EQU   208
0080        L?LO    EQU   128
3FFF        L?HI    EQU   16383
0008         A  0001   0008 LD       0000 001
0088         H         0088 L?R3
00C8         H         00C8 L?HE
014C         H         014C L?CD
0008         J              LD            001
0080         K  0002        L?AR
00D0         K  0003        L?PG
0080         K  0004        L?LO
3FFF         K  0005        L?HI
0040         K  0006        L?CH
0004         K  0007        L?AM
0000         Y
1000         A  0001   0000          048F 001
1000         H         0000 READ
10AC         H         00AC PRNT
10CA         H         00CA B?SH
130C         H         030C RBUF
12A8         H         02A8 EOJ
11EE         H         01EE FOF
1000         J
12A8         K  0002        EOJ
```

```
130C            K  0003        RBUF
1570            K  0004        TBRD
11EE            K  0005        FOF
1000            K  0006        READ
10AC            K  0007        PRNT
11E6            Y
14C0            A  0001   0000 EX1      00AF 001
14D4            H         0014 E?IN
14D6            H         0016 E?SC
1500            H         0040 E?RE
152E            H         006E E?DS
14C0            J              EX1
0000            Y
1570            A  0001   0000 TBRD     0100 001
1570            J              TBRD
0000            Y
11E6                 END
```

EMPLOYEE NUMBER	SALESMAN	SALES	1
		.00	
00 C0 01	ANDERSON, MAMIE	3,564.10	
00 C0 02	BROWN, JOHN	10,192.10	
00 C0 03	CUNNINGHAM, TOM	4,727.70	
00 C0 04	DOBBINS, PAUL	13,532.30	
00 00 05	GETTY, PAUL	13,260.50	
00 C0 06	HUGHES, HOWARD	24,459.03	
00 00 07	WASHINGTON, ED	4,444.20	
00 C0 08	ELLSWORTH, AL	18,403.82	
00 C0 09	CASTOR, FRANK	2,104.52	
00 00 10	PETERSON, JOHN	9,542.03	
00 00 11	JOHNSON, PAUL	2,014.53	
00 00 12	JAMES, DIANE	14,963.64	
00 C0 13	EAST, MICHAEL	7,542.10	
00 C0 14	TISHMAN, BOB	17,568.99	
00 C0 15	ROBERTSON, ROD	9,765.84	
00 00 16	ISAACMAN, BOB	1,111.22	
00 00 17	CHRISLER, PETER	10,947.62	
00 00 18	DELNERO, MIKE	6,528.30	
00 C0 19	FOWLER, FRANK	2,035.20	
00 C0 20	HUDSON, STEVE	12,975.57	
00 C0 21	HILL, BEVERLY	5,243.06	
00 C0 22	JOHNSON, FRANK	2,530.25	
00 00 23	THOMAS, LANCE	5,623.02	
00 C0 24	MILLER, GLADYS	21,784.01	

EMPLOYEE NUMBER	SALESMAN	SALES	2
000026	TODD, BARBARA	23,029.13	
000027	WHITMAN, TOM	18,273.10	
000028	POTTS, RONALD	7,852.41	
000029	SCOTT, GEORGE	5,623.54	
000030	LUCAS, MARY	7,920.21	
000031	BROTHERS, PAT	5,602.42	
000032	GOLDSTEIN, MARK	7,582.02	
000033	FREEMAN, PETER	3,052.05	
000034	ROSS, BETSY	9,766.43	
000035	WHITMAN, WALT	6,025.41	
000036	ADAMS, JOHN	7,025.74	
000037	WEBB, JOAN	8,963.02	
000038	DAVIS, BEATRICE	5,202.41	
000039	MENTO, JOAN	1,020.14	
000040	HUNTER, THELMA	8,520.35	
000041	WELLS, BEATRICE	9,657.20	
000042	HEART, JOHN	2,050.30	
000043	WARHOL, ANDY	1,056.23	
000044	DELANEY, JOHN	3,025.14	
000045	DOE, JOHN	9.85	
000046	GARDNER, TRUDY	9.85	
000047	MANNING, BEN	7,348.10	
000048	ABBOTT, ANN	65.20	
000049	HAUSMANN, TOM	520.35	
000050	MANN, BUDD	3,203.56	

EMPLOYEE NUMBER	SALESMAN	SALES	3
000051	GRANT, EDWARD	11,100.86	
000052	FAGAN, EDWARD	4,210.35	
000053	PETERS, DANIEL	2,054.12	
000056	LA ROCCA, JOE	8,532.04	
	TOTAL SALES	403,065.18	

END OF REPORT FOR FISCAL YEAR

SAMPLE PRODUCTION RUN OUTPUT (CONT.)

TERMINAL PROBLEM

INTRODUCTION

The terminal problem is a monthly payroll reconciliation report program that will provide a practical exercise in applying the knowledge and coding skill you have acquired in this course. Use the Reference Supplement Section starting on page 3–59 to review specific instructions.

Given the following problem statement and the input/output requirements for the program produce the flowchart and coding for the solution of the problem.

PAYROLL RECONCILIATION REPORT PROGRAM OBJECTIVE

The Programmer's objective is to prepare a monthly reconciliation report that will provide management with salary status information and the accounting department with a monthly payroll total for all salaried employees.

Below is a diagram followed by a description of the problem in terms of input, processing, and output.

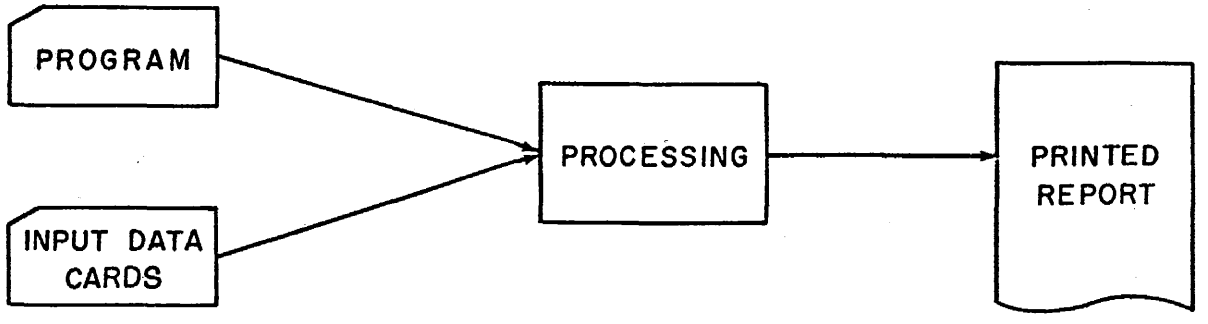


Figure 3–12 System Diagram

Input Data

The punched card deck contains a date card (the first card in the deck), and detail cards. Detail cards have a 1 punched in column 1, and employee number social security number, check amount and check number. Detail cards are in alphabetical order by employee name.

Processing

Print report title and page number at the top of every page, print column headings and detail information on each employee. Accumulate a final total; edit the total; then, print the total.

Output

Printed report as shown on next page.

INPUT DATA CARD FORMAT

COLUMN	DATA
1	Identifying Detail Card
27-31	Employee Number
33-41	Social Security Number
43-48	Check Amount
50-54	Check Number

PRINTER FORMAT CHART

FORM NUMBER PAYROLL REPORT APPLICATION DATE

FORM PARTS RUN NAME RUN NUMBER PREPARED BY

TYPE OF PRINTOUT RECORD NAME RECORD NUMBER APPROVED BY

DATE APPROVED

HAA HDA HDB HDC HDD

X——————X PAYROLL CHECK RECONCILIATION PAGE ZZZ

(DATE)

EMP. NO. SOC. SEC. NO. CHECK AMT. CHECK NO.

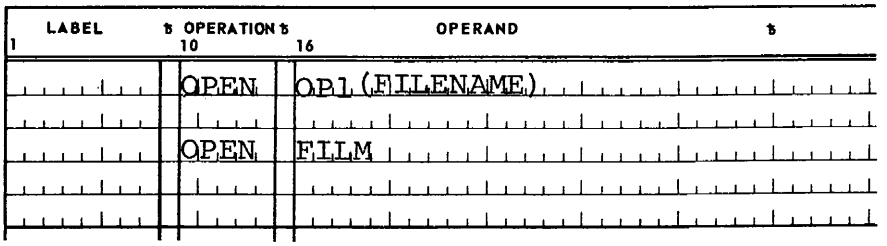
ZZZZZ XXX-XX-XXXX X,XXX.XX ZZZZZ

TOTAL AMOUNT XXX,XXX.XX

OPEN

Format

Example



A. FUNCTION

The OPEN macro makes a file available for input or output; each OPEN statement activates a specified file that is to be utilized in the user's program.

B. RULES

1. A Label field entry is optional.
2. The Operation field entry is OPEN.
3. The Operand (OP1) entry is the filename specified in the DTF statements.
4. A file must be opened before any input/output macros can be issued for that file e.g., GET, PUT.
5. Suggested order for starting program:

```
START
DTF's
BAL
USING
EXTRN
ENTRY
OPEN
GET
```

C. APPLICATION

One OPEN statement must be used for each file to be accessed.

CLOSE

Format

Example

LABEL	OPERATION	OPERAND
	CLOSE	OP1,(FILENAME)
	CLOSE	FILE

A. FUNCTION

To deactivate, or close, any file that was previously opened.

B. RULES

1. The Label field is not used.
2. The Operation field entry is CLOSE.
3. The Operand (OP1) entry specifies the filename of the file to be accessed.
4. A file may be closed after it is determined that the file has been processed; e.g., after the PUT macro has been issued for output file. It is convenient to close all files following the last PUT statement.

C. APPLICATION

One CLOSE statement must be used for each file referenced.

D. EXAMPLE

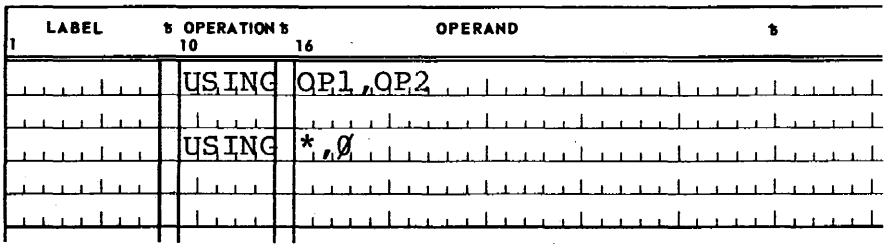
If the filenames specified in DTF are FILM, FILT, and FILO, the appropriate CLOSE Macros are as shown below.

LABEL	OPERATION	OPERAND
	CLOSE	FILM
	CLOSE	FILT
	CLOSE	FILO

USING

Format

Example



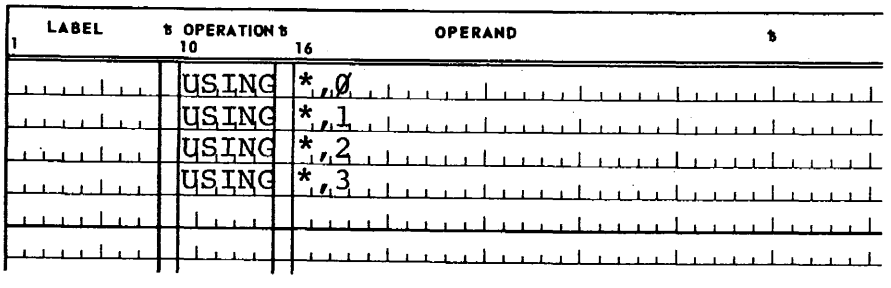
A. FUNCTION

The USING directive provides for direct addressing by listing the number of modules of 4096 bytes that are available for instructions including the Input/Output Control System (IOCS) and the tables.

B. RULES

1. The Label field is not used.
2. The Operation field entry is USING.
3. The example above makes 4K bytes available (*,0).
4. The statements must be assigned in sequence starting with USING *,0.
5. One USING statement for each block of 4K bytes.
6. The available memory can be designated for up to 32K bytes.

The following lines of coding would be included for a computer whose memory is rated at 16K bytes.



For 8K bytes of memory only the first two lines would be used.

EXTRN

Format

Example

LABEL	OPERATION	OPERAND
	EXTRN	OP1(FILENAME FOR READER)
	EXTRN	OP1(FILENAME FOR PRINTER)
	EXTRN	READ

A. FUNCTION

Allows for the labels used in one program to be defined in another.

B. RULES

1. The Label field not used.
2. The Operation field entry is EXTRN.
3. The Operand (OP1) is the filename for the reader or the printer. (The name given in a DTF statement)
4. If the EXTRN statement is not supplied the assembler will flag the filenames as undefined (unreferenced).

The suggested placement in the program is shown below.

LABEL	OPERATION	OPERAND
	START	
	DTFxx	
	BAL	
	USING	*,Ø
	EXTRN	READ
	ENTRY	EOJ
	OPEN	

ENTRY

Format

Example

LABEL	OPERATION	OPERAND
	ENTRY	OP1(FILENAME-NAME OF SUBPROGRAM)
	ENTRY	EOJ

A. FUNCTION

Supplies a name of a subprogram within the user program.

(Supplies reference to a keyword parameter in the DTF statement; e.g., DTFCR EOFA=EOJ)

B. RULES

1. The Label field not used.
2. The Operation entry is ENTRY.
3. The Operand (OP1) specifies the name of the user's subprogram.
4. ENTRY supplies reference to a keyword parameter in the DTF statement.

LABEL	OPERATION	OPERAND	COMMENTS
	ENTRY	EOJ (END OF JOB ROUTINE)	
	ENTRY	FOF (FORMS OVERFLOW ROUTINE)	

GET

Format

Example

LABEL	OPERATION	OPERAND
	GET	OP1(FILENAME),OP2(WORKNAME)
	GET	FILM,AREA

A. FUNCTION

The GET macro makes the next consecutive logical record available for processing in either an input area or a work area.

B. RULES

1. Records are processed in work areas. In this case, the GET macro moves each record from the DTFCR specified input area (IOA1) to a work area. In this format, the Label field is not used, and the Operation field entry is GET. The Operand field has two parts, OP1 and OP2, which are separated by commas.

   a. OP1 (Filename) - the name specified in the DTFCR. Name of file from which record is to be retrieved.

   b. OP2 (Workname) - the name specified in the DS instruction that reserves the work area in memory.

   NOTE: DTFCR's must include an IOA1=RBUF entry (name of the buffer area for card reader).

2. If the data transferred to Workname (AREA) by the GET instruction is a standard end-of-file card (the data contains the characters /* in the first two locations) control is transferred to the EOJ subprogram.

EXAMPLE

LABEL	OPERATION	OPERAND	COMMENTS
FILU	DTFCR	IOA1=RBUF,(READER BUFFER AREA)	
	.		
	.		
AREA	DS	CL125 (READER WORK AREA)	
	.		
	.		
	.		
	GET	FILU,AREA	

PUT

	LABEL	OPERATION	OPERAND
	1	10	16
Format		PUT	OP1(FILENAME),OP2(WORKNAME)
Example		PUT	FILU
		PUT	FILU,AREA

A. FUNCTION

The PUT macro causes the writing, punching, or displaying of logical records that have been assembled directly in the input area or a specified work area.

B. RULES

The PUT macro moves a record from a specified work area for output to a printer, punch, or other device and immediately frees the area for other program use. The Label field is not used, and the Operation field entry is PUT. The Operand field has two parts that are separated by commas:

1. OP1 - the filename specified in the DTF statement.

2. OP2 - the work area specified in the DS instruction that reserves a work area in memory.

   NOTE: DTFPR's must include PROV = FOF (Forms Overflow Routine) entry.

C. APPLICATION

All data transferred from the printer work area to the buffer area causes a line to be printed, advances the paper, and checks for the end of page condition. If the end of page is detected, control is transferred to the FOF program before further execution of the program is accomplished. If the forms overflow condition is not detected, processing continues after the printing of the line of data is completed.

CNTRL

	LABEL	OPERATION	OPERAND
	1	10	16
Format		CNTRL	FILENAME,CODE,M,N
Example		CNTRL	FILP,SP,2

A. FUNCTION

To permit the programmer to specify the format of a printout.

B. RULES

1. The Label field is not used.
2. The Operation field entry is CNTRL.
3. The Operand field entries are the following:
   a. OP1 - Filename specified in DTFPR

      SK,M,N = channel to skip to on carriage control tape
      M = skip before printing
      N = skip after printing

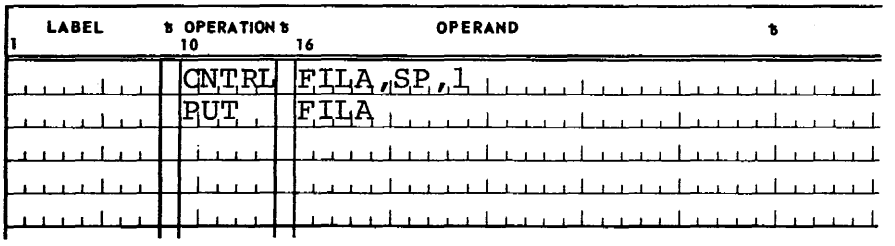
   b. Code - 0, 1 or 2

      If SP is used, the 2 signifies "space two lines".

      SP,M,N (M=number of spaces before printing)
      (N=number of space after printing)
4. M or N may be omitted, commas may NOT be omitted.
5. Normally, single spacing is automatically provided. If any other type of spacing is required, SP is used.
6. If CNTRL is omitted, normal spacing as specified by the PRAD Detail Entry card will be executed.
7. Throughput will be faster if both spacing and skipping are specified after printing, rather than before.

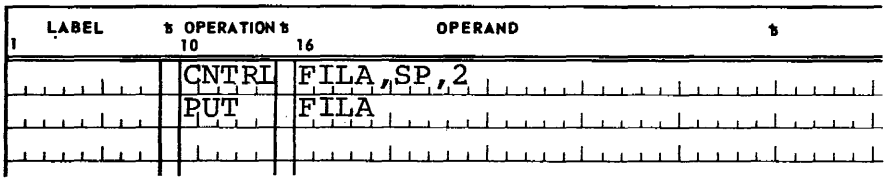
C. EXAMPLES

1.



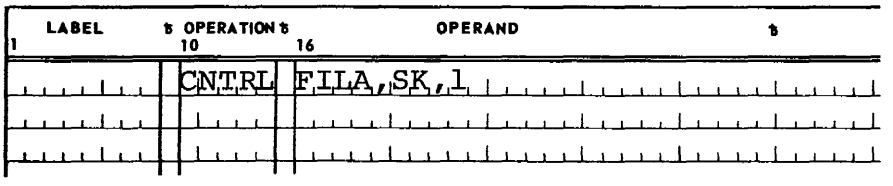
Advances one line before printing.

2.



Advances two lines before printing.

3.



Specifies an immediate skip to bottom of page.

The CNTRL macro instruction should be used only in conjunction with a 48 or 16-character print bar.

INTRODUCTION TO PACK AND UNPACK

PACKED FORMAT:

One byte (8 bits) represents two decimal digits. The rightmost half-byte represents the sign.

BYTE		BYTE		BYTE		BYTE	
Digit	Digit	Digit	Digit	Digit	Digit	Digit	Sign

EXAMPLE: 37246C Pos. sign = C
Neg. sign = D

UNPACKED FORMAT (ZONED FORMAT):

The low-order four bits of each 8-bit contain the decimal digit, and the high-order four bits define the EBCDIC zone. The high-order four bits of the rightmost byte of the field contain the sign of the field.

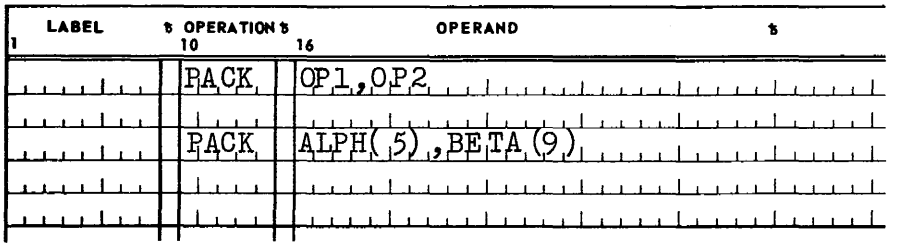
BYTE		BYTE		BYTE		BYTE	
Zone	Digit	Zone	Digit	Zone	Digit	Sign	Digit

EXAMPLE: F1F3F4C5 Pos. signs - F,A,C,E
Neg. signs - B and D

PACK

Format

Example



A. FUNCTION

The operand specified by the OP2 is converted from zoned format to packed format, and the result is placed in the location specified by OP1.

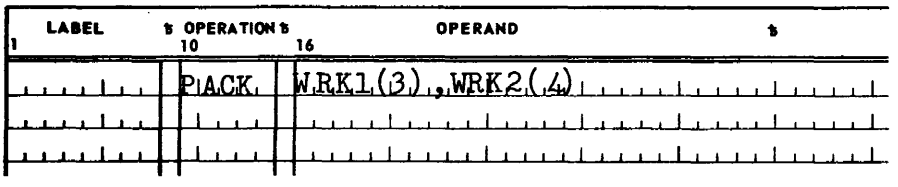
B. RULES

1. The operand specified by the second address must be in zoned format.
2. Packing may be done in place, or in another area. Also operands may overlap.
3. If packing is done in another area, the area does not have to be cleared or initialized.
4. The maximum size of the second operand is 16 bytes.
5. To determine the minimum size of the first operand, divide the number of bytes in the zoned field by two and add one to the result. If the zoned field contains an odd number of bytes, ignore the fraction when you divide by two.
6. The PACK instruction must be used when data to be processed must be operated on by any of the decimal instructions, e.g., AP,SP, CP.

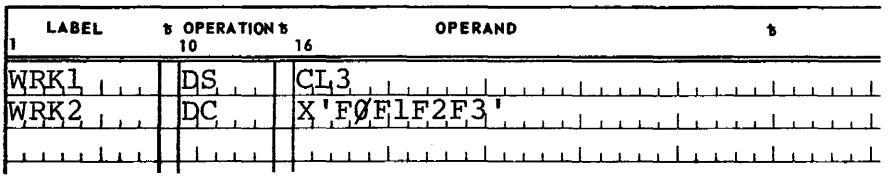
C. APPLICATION

1. The fields are processed one byte at a time from right to left. Signs and digits are not checked for validity.
2. The two portions of the low-order byte are reversed, leaving the sign in the low-order position. Then, the zone portion is stripped from each successive byte, and two decimal digits of the second operand and are combined to produce each byte of the packed field.
3. If the first operand is too long, it will be filled with high-order zeros.
4. If the first operand is too short, any remaining high-order digits in the second operand will be ignored.
5. The second operand is not changed except when operands overlap.
6. No condition code is generated.

D. EXAMPLES



Storage Definitions:



Operands Before:

Operand 1	WRK1	F0F0F0
Operand 2	WRK2	F0F1F2F3

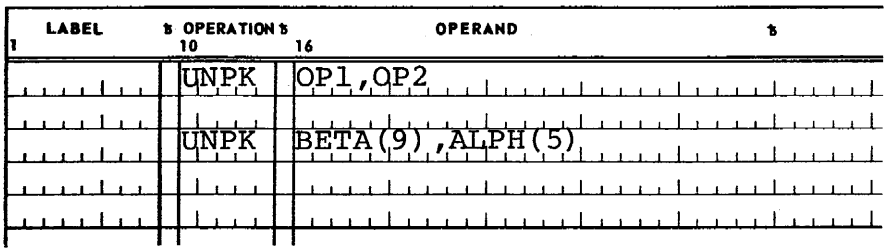
Operands After:

Operand 1	WRK1	00123F
Operand 2	WRK2	F0F1F2F3

UNPACK

Format

Example



A. FUNCTION

The operand specified by OP2 is converted from packed format to zoned format and the result is placed in the location specified by OP1.

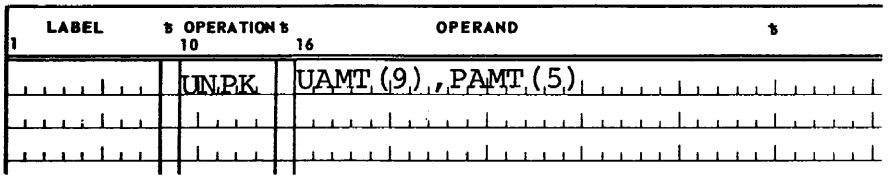
B. RULES

1. The second operand should be in packed format.
2. The maximum size of the first operand (zoned field) is 16 bytes.
3. A field should not be unpacked into itself.
4. To determine the minimum size of the first operand, double the number of bytes in the packed field and subtract one from the result.
5. Packed data cannot be printed in meaningful form. Therefore, this instruction is used to enable data to be punched in standard code, or printed in readily readable form.
6. The first operand does not have to be cleared or initialized by the user.

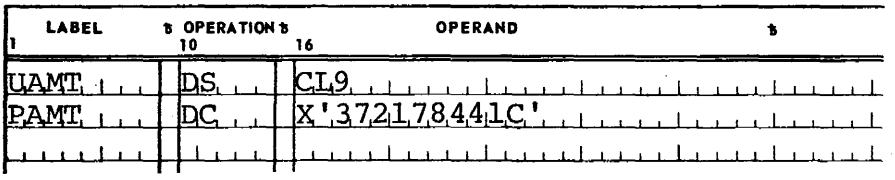
C. APPLICATION

1. The fields are processed one byte at a time from right to left. Signs and digits are not checked for validity.
2. The two portions of the low-order byte are reversed, leaving the sign in the high-order position. The sign is a standard plus (1100) or minus (1101) sign, depending upon the sign of the packed field. Each digit is preceded by a hexadecimal F (this occupies the zone portion of each byte).
3. If the first operand is too short, any remaining high-order digits are ignored.
4. If the first operand is too long, it will be filled with high-order zeros.
5. The condition code remains unchanged.
6. If two different operand fields are specified, the second operand field is unchanged.

D. EXAMPLE



Storage Definitions



LABEL	OPERATION	OPERAND
UAMT	DS	CL9
PAMT	DC	X'372178441C'

Operands Before

UAMT 000000000000000000

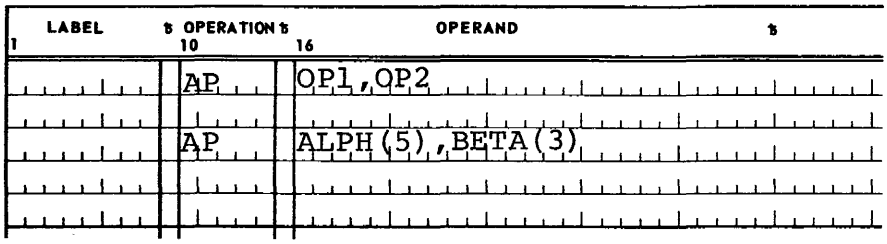
PAMT 372178441C

Operands After

UAMT F3F7F2F1F7F8F4F4C1

PAMT 372178441C

ADD PACKED DECIMAL



A. FUNCTION

The operand specified by OP2 is added algebraically to the operand specified by OP1. The result is stored in the field specified by OP1. The sign and magnitude of the sum determine the condition code.

B. RULES

1. Both operands must be in packed format.
2. The first operand must be long enough to contain all the significant digits of the sum.
3. If the second operand is shorter than the first, the addition will be normal.
4. The maximum length of either operand is 16 bytes.
5. Overflow occurs if:
   a. There is a carry out of the high-order position of the result.
   b. The second operand is larger than the first operand and significant result positions are lost.
6. If operands overlap, their rightmost byte location must coincide.
7. A field may be added to itself.

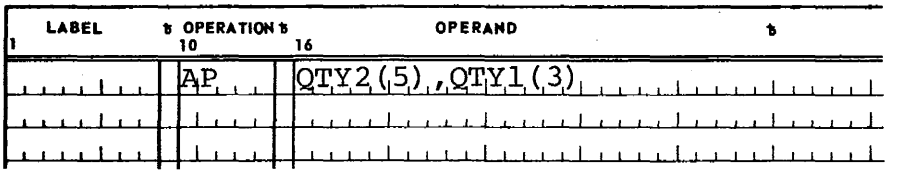
C. APPLICATION

1. Processing is from right to left. Signs are checked first before the arithmetic is performed.
2. All signs and digits are checked for validity.
3. High-order zeros are supplied for either operand during instruction execution.
4. The operand specified by the second address is unaltered.
5. Algebra rules are used for determining signs.
6. Zero result is always positive, except when high-order digits are lost because of overflow.
7. In overflow, a zero result has the sign of the correct result.
8. The sum is in packed format.

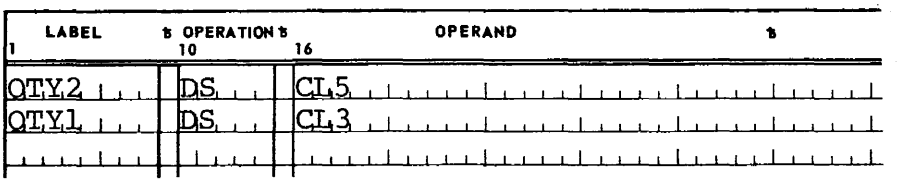
9. The condition code settings are as follows:

CONDITION	SETTING
Sum = 0	0
Sum is < zero	1
Sum is > zero	2
Overflow	3

D. EXAMPLE



Storage Allocations



Operands Before

First Operand:	QTY2	000000123C
Second Operand:	QTY1	08900C

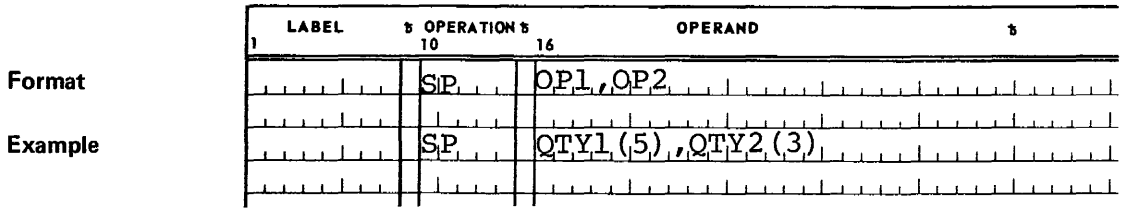
Operands After

First Operand:	QTY2	000009023C
Second Operand:	QTY1	08900C

Condition Code Setting

2 (result is positive.)

SUBTRACT PACKED DECIMAL



A. FUNCTION

The operand specified by OP2 is subtracted algebraically from the operand specified by OP1. The result is stored in the field specified by OP1. The sign and magnitude of the difference determine the condition code.

B. RULES

1. Both operands must be in packed form.
2. The first operand must be long enough to contain all the significant digits of the difference. Otherwise, overflow occurs.
3. A field may be subtracted from itself.
4. If the second operand is shorter than the first, subtraction will take place normally.
5. The maximum length of either operand is 16 bytes.
6. If operands overlap, their rightmost byte locations must coincide.

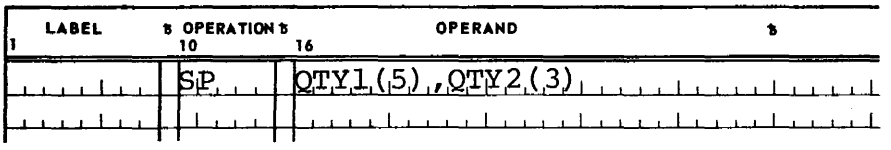
C. **APPLICATION**

1. Processing is from right to left. The signs are checked first and then arithmetic is performed.
2. All signs and digits are checked for validity.
3. High-order zeros are supplied for either operand during instruction execution.
4. The operand specified by the second address is unaltered.
5. Algebra rules are used for determining signs.
6. Zero result is always positive except when high-order digits are lost because of overflow.
7. In overflow, a zero result has the sign of the correct difference.
8. The difference is in packed format.

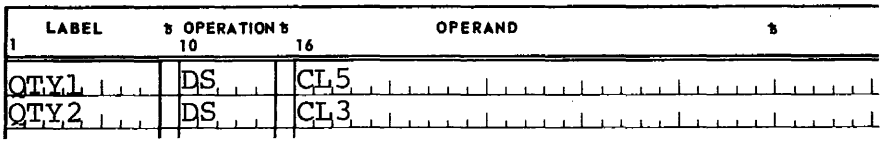
9. The condition code settings are as follows:

CONDITION	SETTING
Difference = 0	0
Difference < 0	1
Difference > 0	2
Overflow	3

D. EXAMPLE



Storage Allocations



Operands Before

QTY1	000000122C
QTY2	00123C

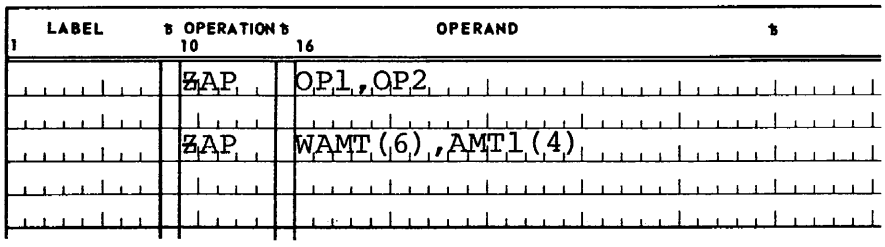
Operands After

QTY1	0000001D
QTY2	00123C

Condition Code Setting

1 (result is negative)

ZERO AND ADD PACKED DECIMAL



A. FUNCTION

The storage location specified by OP1 is cleared to zero and then the OP2 data (packed format) is added to OP1. The result of addition determines the condition code.

B. RULES

1. The operands may have different lengths. However, the first operand should be longer than the second operand.
2. The maximum length of operands is 16 bytes.
3. The second operand must be in packed format.
4. Operands may overlap if their rightmost byte locations coincide or if the rightmost byte of the first operand is to the right of the rightmost byte of the second operand.
5. ZAP is used when OP1 in a decimal instruction (e.g., AP, MP) is too small to hold the result of the operations. The operand is placed into a larger field through the use of a ZAP instruction. Then the new larger field is used as the first operand.

C. APPLICATION

1. Processing is from right to left.
2. The second operand is unaltered.
3. Only the second operand is checked for valid sign and digit codes.
4. A second operand that is longer than the first causes overflow.
5. When high-order digits are lost due to overflow, a zero result has a positive sign.
6. The condition codes are set as follows:

   0 - Result is zero
   1 - Result is less than zero
   2 - Result is greater than zero
   3 - Overflow

D. EXAMPLE

LABEL	OPERATION	OPERAND
	ZAP	WAMT(6),AMT1(4)

Storage Allocations

LABEL	OPERATION	OPERAND
WAMT	DS	CL6
AMT1	DS	CL4

Operands Before

WAMT	35792485354D
AMT1	1233663C

Operands After

WAMT	00001233663C
AMT1	1233663C

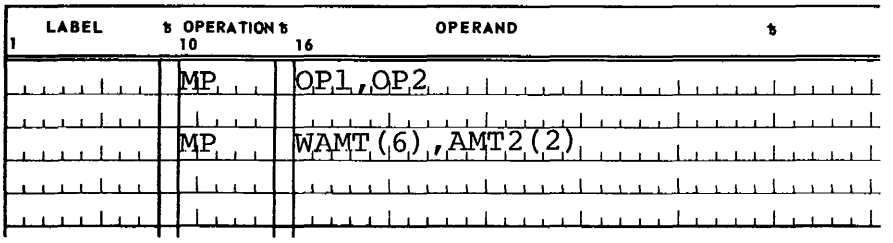
Condition Code Setting

2 (result is positive)

MULTIPLY PACKED DECIMAL

Format

Example



A. FUNCTION

The operand specified by OP1 (multiplicand) is multiplied by the operand specified by OP2 (multiplier). The signed product is placed in the OP1 location.

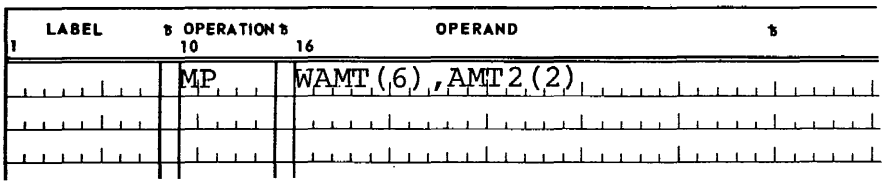
B. RULES

1. Both the multiplier and multiplicand must be in packed form.
2. The second operand (multiplier) must be shorter than the first operand (multiplicand) and must not exceed eight bytes in length.
3. The maximum length is 16 bytes (one length is specified for each operand.
4. The multiplicand must have high-order zero bytes equal to the number of bytes in the multiplier field.
5. Operands may overlap if their rightmost bytes coincide.

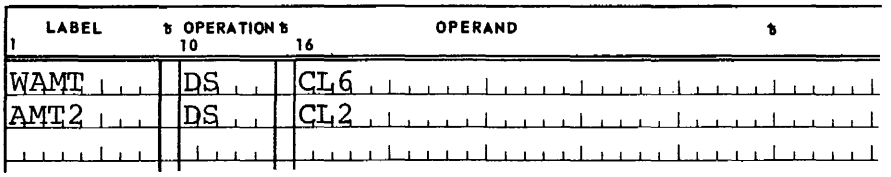
C. APPLICATION

1. Instruction operates right to left.
2. All signs and digits are checked for validity.
3. The second operand is unaltered unless operands overlap.
4. Overflow cannot occur.
5. The condition code remains unchanged.
6. The sign of the product is determined by the rules of algebra, even if one or both operands are zero; i.e., minus zero is a possible result.
7. The product is in packed format.

EXAMPLE



Storage Allocations



LABEL	OPERATION	OPERAND
WAMT	DS	CL6
AMT2	DS	CL2

Operands Before

WAMT	00001233665C
AMT2	012C

Operands After

WAMT	00014803980C
AMT2	012C

MOVE CHARACTER

Format

LABEL	OPERATION	OPERAND
	MVC	OP1,OP2

Example

LABEL	OPERATION	OPERAND
	MVC	BAKR(7),ABLE

A. FUNCTION

The source field specified by OP2 is moved into the destination field specified by OP1.

B. RULES

1. One length indicator is specified for both operands.
2. A maximum of 256 bytes may be moved with one instruction.
3. One character (e.g., a space) may be used to clear an entire field, if the first operand field starts one character to the right of the second operand field.
4. Overlapping of fields is permitted.

C. APPLICATION

1. Bytes are moved one at a time in each field.
2. Movement is from left to right.
3. The number of bytes moved is determined by the implicit or explicit length of the first operand.
4. The bytes being moved are not inspected or changed.
5. The second operand is not altered, unless operands overlap.
6. The condition code is unchanged.

D. EXAMPLE

LABEL	OPERATION	OPERAND
	MVC	DOG(3),CAT

Storage Definitions

LABEL	OPERATION	OPERAND
DOG	DS	CL3
CAT	DC	X'ØØØØ2C'

Operands Before

DOG	00001C
CAT	00002C

Operands After

DOG	00002C
CAT	00002C

MOVE IMMEDIATE

Format

LABEL	OPERATION	OPERAND
	MVI	OP1,OP2

Example

LABEL	OPERATION	OPERAND
	MVI	SPOT,C'A'

A. FUNCTION

One byte of immediate data is stored in the main memory location specified by the first address (OP1). The immediate data is specified by the second operand of the instruction (OP2).

B. RULES

1. The second operand, called a self-defining value, may be written as a single character in quotes preceded by a C (e.g., C'A') or as two hexadecimal digits in quotes preceded by the letter X (e.g., X'C1').

2. The first operand field is a one-byte receiving field.

3. The length indicator is never specified since this instruction only operates on one byte.

4. The condition code setting is not affected.

D. EXAMPLE

LABEL	OPERATION	OPERAND
	MVI	SPOT,C'A'

Storage Definition

LABEL	OPERATION	OPERAND
SPOT	DS	CL3

SPOT (1st operand) Before SPOT After

00C2C3 C1C2C3

COMPARE PACKED DECIMAL

	LABEL	OPERATION	OPERAND
Format		CP	OP1,OP2
Example		CP	BAL(4),CHK(3)

A. FUNCTION

The operand specified by the first address is algebraically compared with the operand specified by the second address. The results of the comparison determine the condition code.

B. RULES

1. Both operands must be in packed decimal format.
2. Operands may be of different lengths.
3. The comparison is albegraic.
4. If operands overlap, their rightmost byte locations must coincide.
5. The maximum length for either operand is 16 bytes.

C. APPLICATION

1. Comparison is from right to left, taking into account the sign as well as all the digits of each field.
2. If fields of unequal length are compared, the shorter field is extended with high-order zeros.
3. Plus zero and minus zero compare equally (no distinction is made).
4. The condition code settings are as follows:

   0 - Operands are equal numerically

   1 - First operand algebraically less than 2nd operand

   2 - First operand algebraically greater than 2nd operand

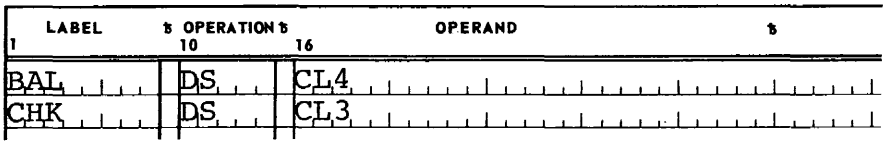
   3 - Not used. Overflow cannot occur.

5. Neither operand is altered.

D. EXAMPLE

LABEL	OPERATION	OPERAND
	CP	BAL(4),CHK(3)

Storage Definitions



Operands Before

1st OP.	BAL	0912394C
2nd OP.	CHK	12394C

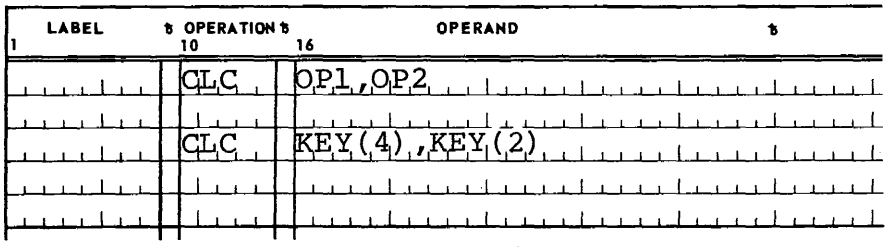
Operands After

1st OP.	BAL	0912394C
2nd OP.	CHK	12394C

Condition Code Setting

2 (1st operand is greater than second operand).

COMPARE LOGICAL



A. FUNCTION

The operand specified by OP1 is logically compared with the operand specified by OP2. The result of the comparison determines the condition code. All bits are processed as part of an unsigned binary quantity.

B. RULES

1. Only 1 length field is used (OP1).
2. An operand of up to 256 bytes may be compared with another operand of the same length unpacked (EBCDIC) characters.
3. Operands may be in any format.
4. The operation may be used for alphanumeric comparisons.

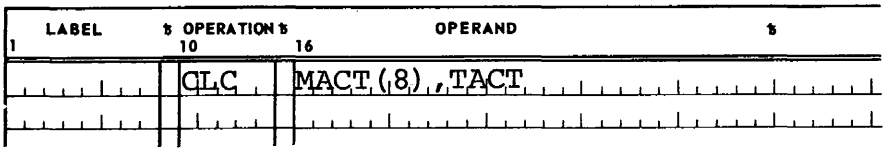
C. APPLICATION

1. Processing is from left to right.
2. Instruction terminated on inequality, or when the operands are exhausted.
3. Both operands are unaltered.
4. The condition code is set as a result of the comparison.

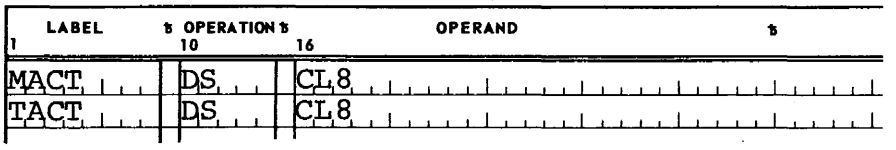
Condition Code Settings

CONDITION	CONDITION CODE
Operands equal	0
1st operand less than second operand	1
1st operand greater than 2nd operand	2
Not used	3

D. EXAMPLE



Storage Definitions



Operands Before

1st OP.	MACT	F0F0F8F4F3F1F2C4
2nd OP.	TACT	F7F5F8F4F3F1F2C4

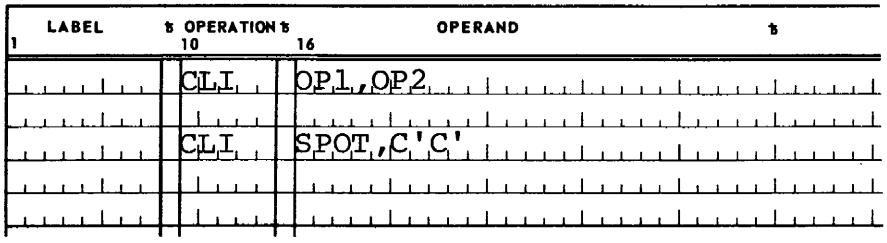
Operands After

1st OP.	MACT	F0F0F8F4F3F1F2C4
2nd OP.	TACT	F7F5F8F4F3F1F2C4

Condition Code Setting

1 (first operand less than second operand).

COMPARE LOGICAL IMMEDIATE



A. FUNCTION

One byte of immediate data (OP2) is logically compared with one byte in memory. The address of the byte in memory is specified by OP1. The immediate data is specified by OP1. The immediate data is specified by OP2 of the instruction. The result of the comparison determines the condition code. The byte comparison is according to absolute EBCDIC coded values and an unsigned binary quantity.

B. RULES

1. OP2, called a self-defining value, may be written as a single character in quotes preceded by a C (e.g., C'A') or two hexadecimal digits in quotes preceded by the letter X (e.g., X 'C3).

2. The first operand field is a 1-byte field.

3. The length indicator is never specified, since this instruction only operates on one byte.

4. The first operand field does not have to be at an even location.

5. The first operand may be in any format.

NOTES:

1. Condition code settings are the same as those for CLC.

2. Both operands are unaltered.

C. EXAMPLE

```
LABEL    OPERATION    OPERAND
1        10           16
         CLI          SPOT,C'C'
```

Storage Definition

```
LABEL    OPERATION    OPERAND
1        10           16
SPOT     DS           CLI
```

SPOT Before	SPOT After
C5	C5

Condition Code Setting

2 (first operand (SPOT) is > immediate field C'C').

BRANCH ON CONDITION

	LABEL	OPERATION	OPERAND
Format		BC	OP1,OP2
Example		BC	15,TAG1

A. FUNCTION

Branching instructions test the setting of the condition code indicator and branch to another location in the program based on the particular setting being tested.

B. RULES

1. These instructions may be used after arithmetic instructions; e.g., if result is positive, branch to specified location.

2. They may be used after compare instructions, e.g., if first operand is greater than second operand, branch to specified location.

C. APPLICATION

15 - Branch on all conditions

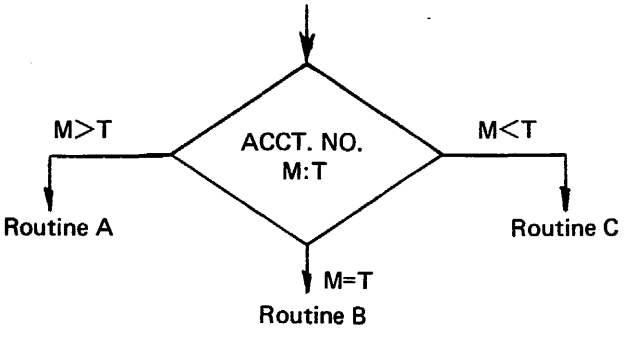
8 - Branch if both operands are equal (CC - 0).

2 - Branch if result is positive (CC - 2).

4 - Branch on minus (CC - 1).

D. EXAMPLE

Flowchart



Coding

LABEL	OPERATION	OPERAND
	CLC	MACT,(7),TACT
	BC	2,RTEA
	BC	8,RTEB
	BC	4,RTEC

Note: 8 can be omitted if it is desirable to "fall through" on an equal condition and continue straight-line processing.

BRANCH AND LINK

	LABEL	OPERATION	OPERAND
	1	10	16
Format		BAL	OP1,OP2
Example		BAL	8,SUBR

A. FUNCTION

To permit a series of instructions (called a subroutine) to be written once and executed several times (subroutines and user program may be tied together).

B. RULES

1. Operand (OP1) specifies a register number (8 through 15) that stores the address of the instruction to be performed after returning from the branch subroutine.

2. OP2 is the label of the branch subroutine.

A BAL instruction may be used to branch to a subroutine that will:

clear an area

read data

pack and add

check tables

STORE HALFWORD

	LABEL	OPERATION	OPERAND
	1	10	16
Format		STH	OP1,OP2
Example		STH	14,PCOT+2

A. FUNCTION

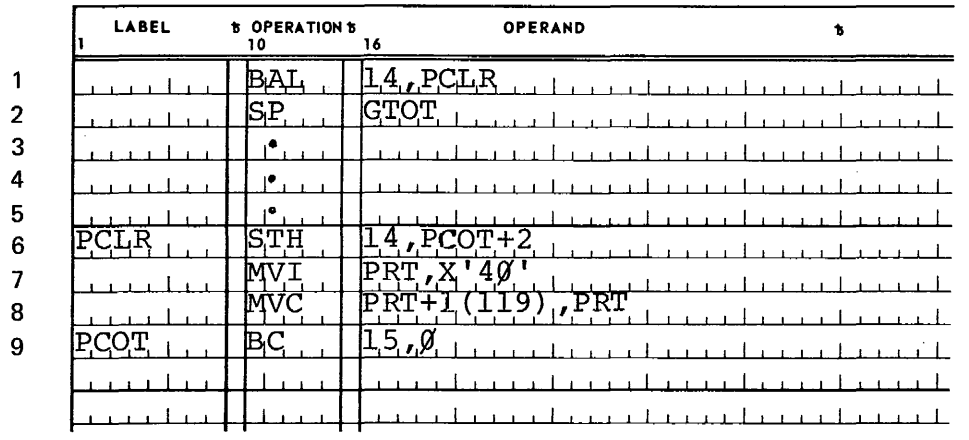
STH places the contents of the register specified by the OP1 address into the halfword specified by the OP2 address.

B. RULES

1. Data is stored on an even numbered address.
2. Goes from main storage into a register.
3. The receiving address is a register (8 through 15).
4. Boundary alignment is required. The sending field must be a memory location.
5. Data is in binary; may be defined by an address constant or as instruction address.
6. The contents of a halfword will be loaded into the register.
7. Does not alter CCI (condition code indication).

D. EXAMPLE

To write the instructions necessary to interrupt a program flow use a subprogram (labelled PCLR) to clear print buffer. After clearing buffer the routine will refer to interrupt.



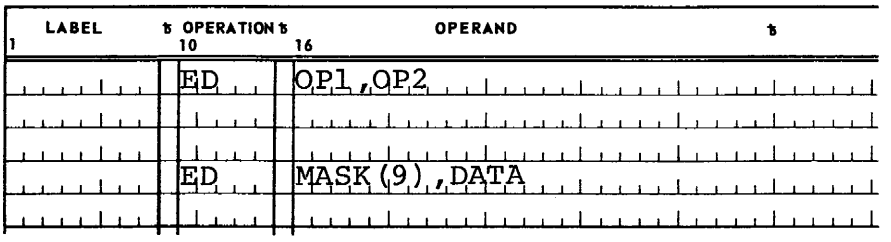
	LABEL	OPERATION	OPERAND
1		BAL	14,PCLR
2		SP	GTOT
3		.	
4		.	
5		.	
6	PCLR	STH	14,PCOT+2
7		MVI	PRT,X'40'
8		MVC	PRT+1(119),PRT
9	PCOT	BC	15,0

(Line 1) Store address of next instruction (line 2) in register 14, Branch to PCLR (line 6).

(Line 6) Store return address (line 2) in register 14;process next instructions in line.

(Line 9) Branch unconditionally to line 2 .

EDIT INSTRUCTION



A. FUNCTION

The purpose of the Edit instruction is to produce easy-to-read printed documents by inserting the required punctuation and graphic symbols.

B. RULES

1. Data to be edited must be packed in decimal form.

2. Operation will change packed decimal field called source field to EBCDIC (zoned format) and insert the necessary punctuation characters, i.e., dollar signs, commas, decimal points, asterisks.

3. Pattern for editing is called the edit mask and is set up as a hexadecimal constant by a DC statement. If an edit mask is to be used more than once, it must be moved to a working storage area before each use otherwise the editing function will destroy the mask.

4. Result of edit replaces 1st operand (OP1) which is the mask field.

5. Construction of mask pattern is as follows:

   a. Number of bytes in mask must be at least the number of significant digits which will print when the format is converted from packed to zoned; e.g., for four packed bytes, the corresponding mask must contain at least seven digit select characters.

   b. First byte of mask in hexadecimal configuration is a fill character.

      Examples:

      blank (X'40')
      dollar sign (X'5B')
      asterisk (X'5C')

   c. Commas and decimal points are inserted as specified in mask:

      comma (X'6B')
      decimal point (X'4B')

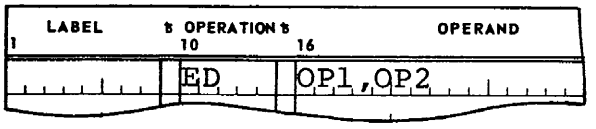
d. Following control characters are used:

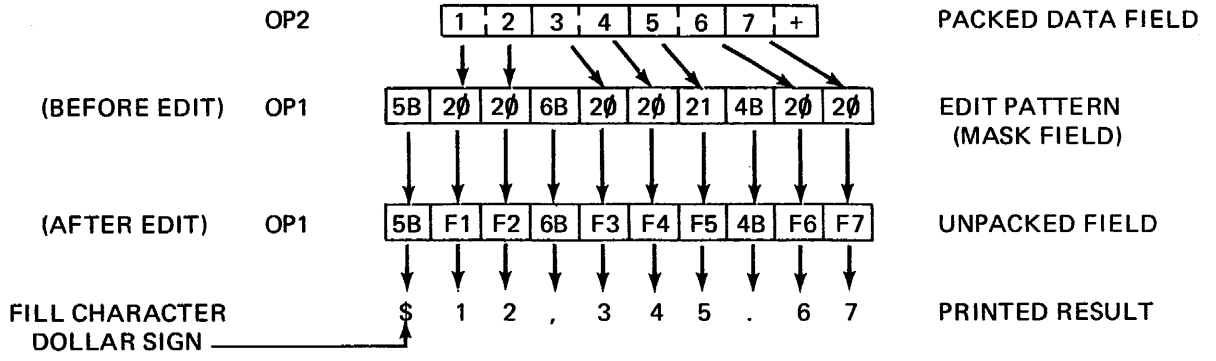
(1) Digit Select character (X'20') is placed in mask where it is desired to insert a digit from the packed field. Digit is inserted unless it is a leading insignificant zero and a Significance Start character has not been encountered previously.

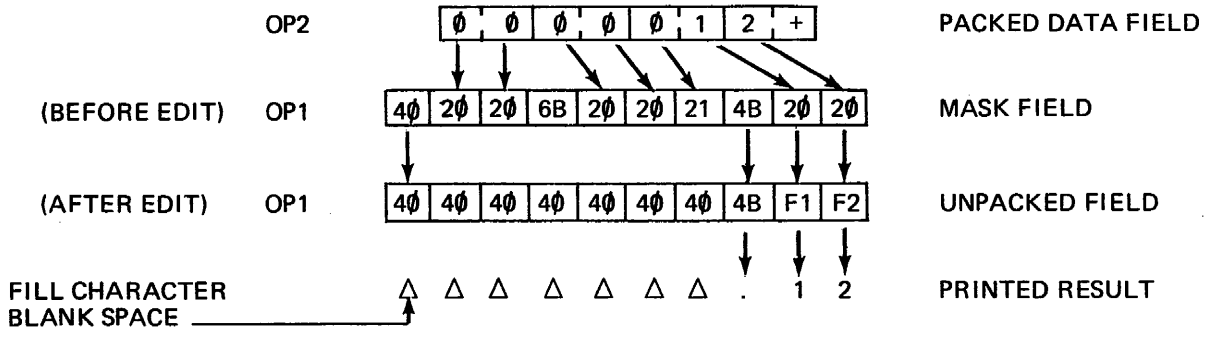
(2) Significance Start character (X'21') serves same function as Digit Select character but has one additional function; it specifies that all of the following digits are to be inserted from the packed field even if one or more leading zeros are still present.

EDIT INSTRUCTION EXAMPLES

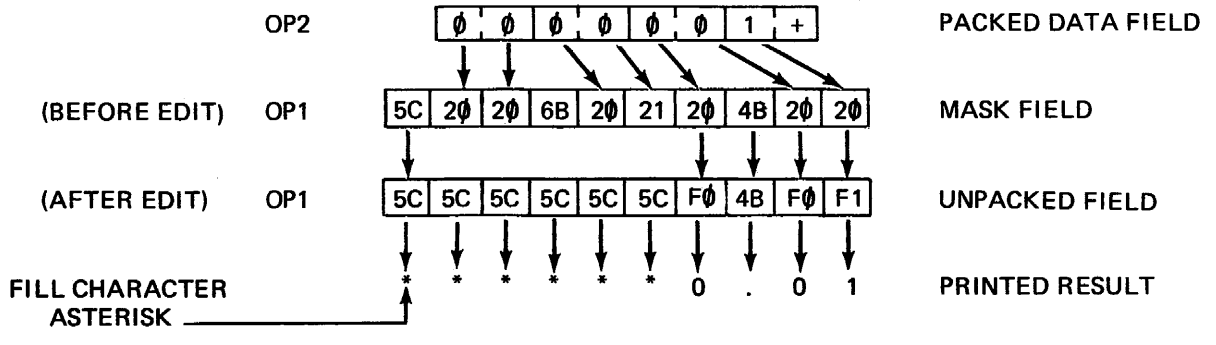
SEE NOTE BELOW







(The symbols Δ or b are used by programmers to indicate blank spaces when checking number of spaces in the result.)



NOTE

Edit control hexadecimal characters:

- Fill character (40, 5B, 5C, etc.)
- Digit Select Character (20)
- Significance Start Character (21)
- Insert characters (6B, 4B)

HALT AND PROCEED

Format

Example

LABEL	OPERATION	OPERAND
	HPR	OP1
	HPR	X'ØFFF'

A. FUNCTION

Stops the processor and displays the OP1 address in the Halt/Display indicators on the control panel.

B. RULES

1. The label field not used.
2. The operation is HPR.
3. OP1 is usually expressed in 1 to 4 hexadecimal digits.
4. OP2 is Ø in all cases.
5. Base Displacement is assumed if OP1 content exceeds X'7FFF'.
6. Forms:

   HPR X'7FFF'

   HPR C'??'

   HPR 2075

Suggested place in coding

LABEL	OPERATION	OPERAND
	CLOSE	READ
	CLOSE	AREA
	HPR	X'ØFFF'
	END	BEGN

ERRATA

9000 CARD ASSEMBLER
PROGRAMMED INSTRUCTION COURSE

Book 3 - BAL Application

Prepared by:

Systems Education Department
Univac Education Center
P.O. Box 1110
Princeton, N.J. 08540

SPERRY UNIVAC COMPUTER SYSTEMS

February 1974
UE-686.2B

ERRATA

UNIVAC 9000 CARD ASSEMBLER

PROGRAMMED INSTRUCTION COURSE

Book 3 - BAL Application (UE-686.2B)

NOTE: CORRECTIONS, DELETIONS, AND CHANGES ARE TO BE PERFORMED AS AN EXERCISE UPON COMPLETION OF BOOK 3.

The Marketing Sales Report Problem which begins on page 3-3 has coding and flowchart errors:

1. When SAVE = EPMN and CNTR = FIVE, a branch to FOF is executed. As a result, the card that was read before the end of page condition (FOF) was sensed, is lost.

2. Refer to figure 3-3, pg. 3-23/24, Marketing Sales Report Flowchart. Find the error in the flowchart and make the necessary corrections. Then check your corrections against those on page E-3.

3. After corrections have been made in figure 3-3, make corrections in the coding to reflect the flowchart corrections.

4. In the Marketing Sales Report Program, two procedures were used for Printer Overflow. The Programmer coded for an end-of-page Condition and also defined PROV=FOF in the DTF statements. Only one procedure should be used.

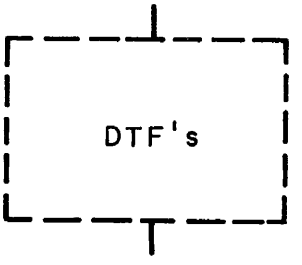
5. Page 3-7: Define Input/Output Devices to be used by program.

   Delete line 9:

   Column 16 PROV=FOF
   Column 72 X

DEFINE INPUT/OUTPUT DEVICES TO BE USED BY PROGRAM

LABEL (1)	OPERATION (10)	OPERAND (16)	72
READ	DTFCR	EOFA=EOJ,	X
		IOA1=RBUF,	X
		ITBL=TBRD,	X
		MODE=TRANS	
PRNT	DTFPR	BKSZ=132,	X
		CNTL=YES,	X
		PRAD=2,	X
		PROV=FOF,	X
		FONT=63	
	END		



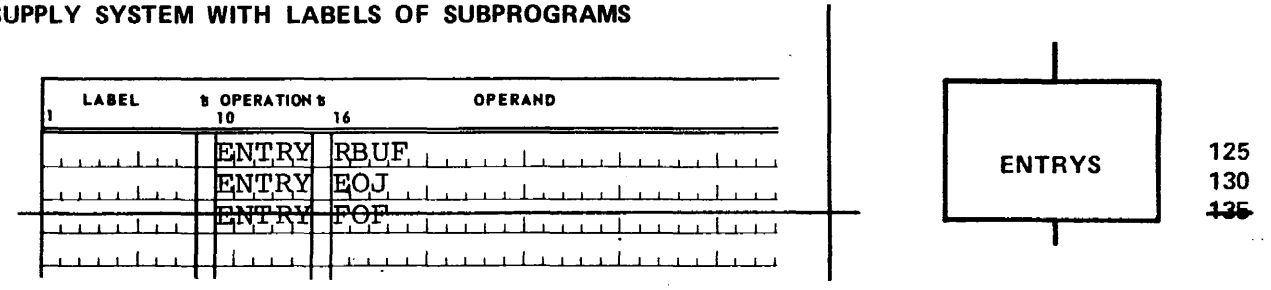
6. Page 3-8:

Delete coding line 135:

Column 10 ENTRY

Column 16 FOF

SUPPLY SYSTEM WITH LABELS OF SUBPROGRAMS



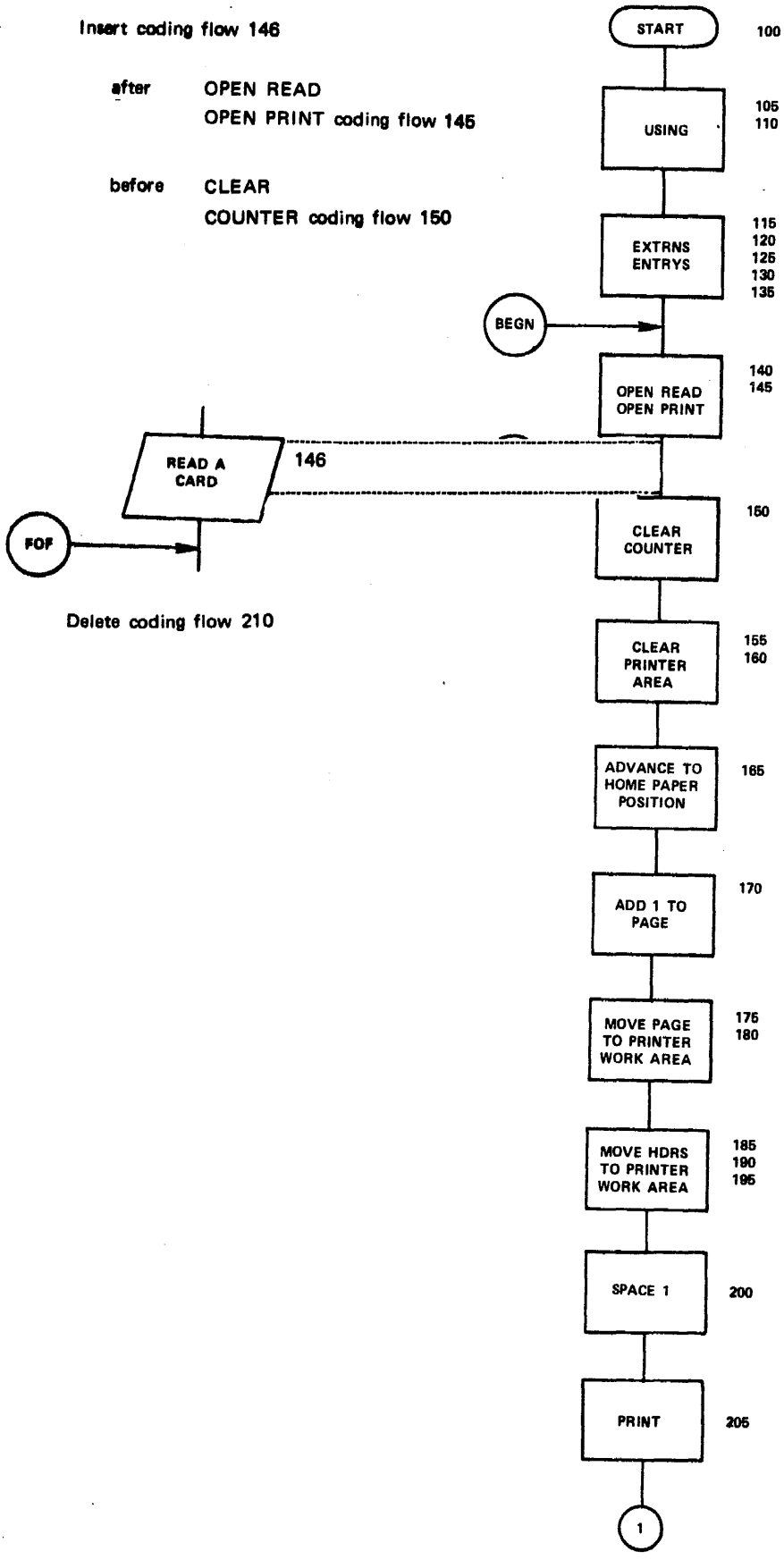
Change explanation of coding to read:

RBUF and EOJ are the labels of subprograms within the user program.

Insert coding flow 146

after OPEN READ
OPEN PRINT coding flow 145

before CLEAR
COUNTER coding flow 150



Delete coding flow 210

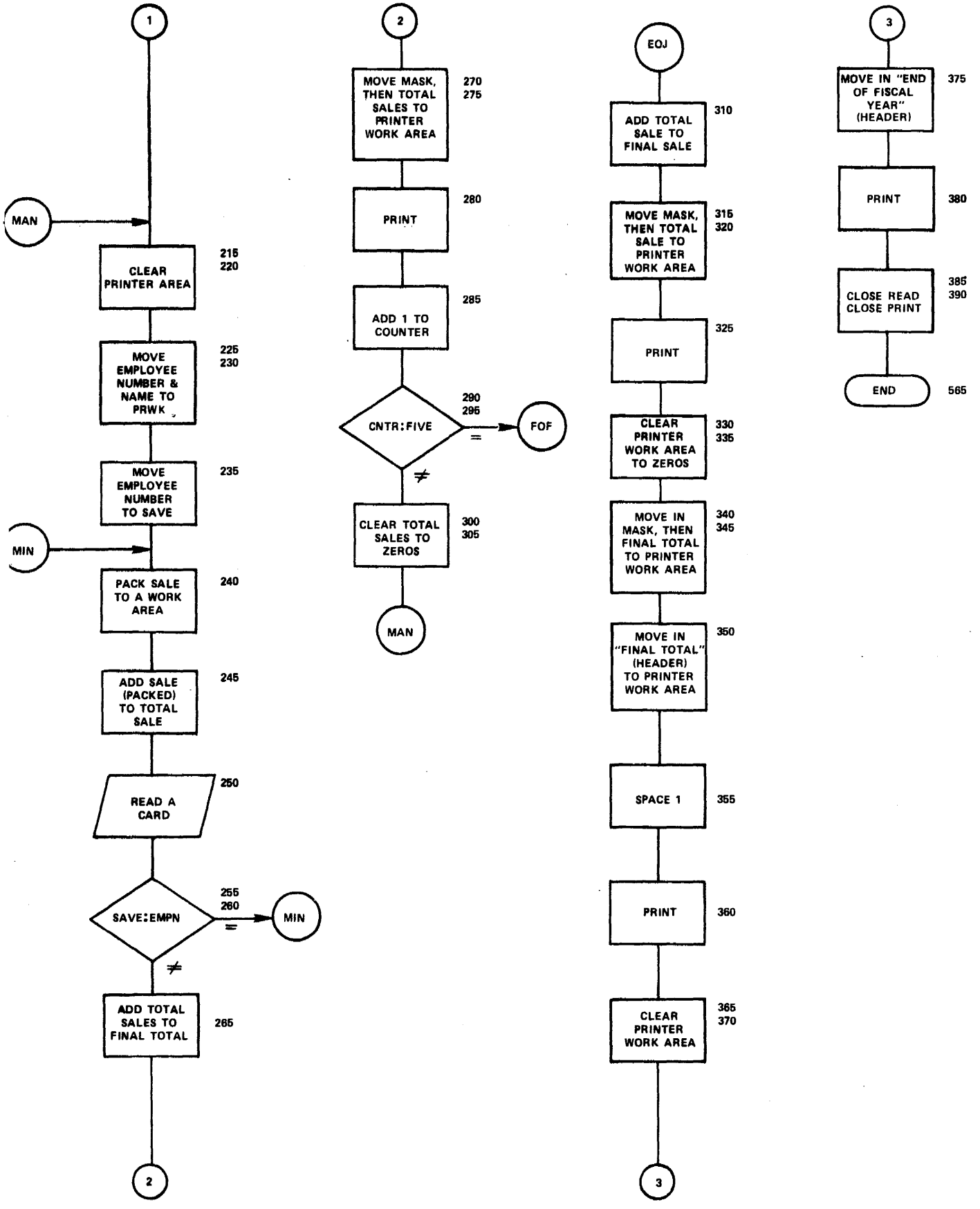


Figure 3-3 Marketing Sales Report Flowchart

7. Page 3-8:

Insert coding line 146 after coding line 145:

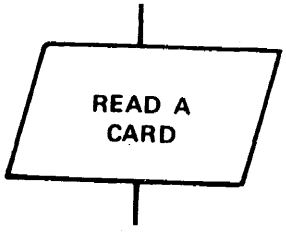
Column 10 GET

Column 16 READ,CARD

READ A CARD

1 LABEL	10 OPERATION 16	OPERAND
	GET	READ,CARD

Reads a card and places data into area defined as CARD.



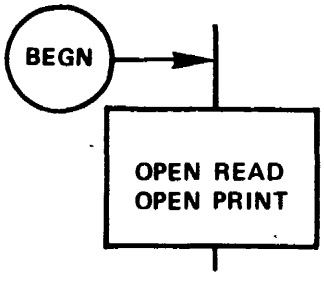
146

ACTIVATE CARD READER, PRINTER

1 LABEL	10 OPERATION 16	OPERAND
BEGN	OPEN	READ
	OPEN	PRNT

OPEN READ makes the file named READ available for sending input.

OPEN PRNT makes the file named PRNT ready to receive output.

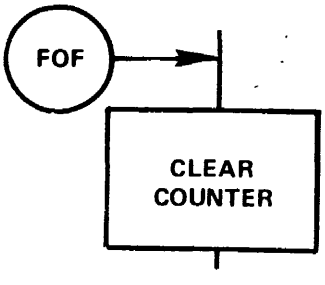


140
145

CLEAR PRINTER LINE COUNTER TO ZERO

1 LABEL	10 OPERATION 16	OPERAND
FOF	MVC	CNTR,TZER+4

Moves zeros from storage area TZER+4 to the two-byte area defined as CNTR.



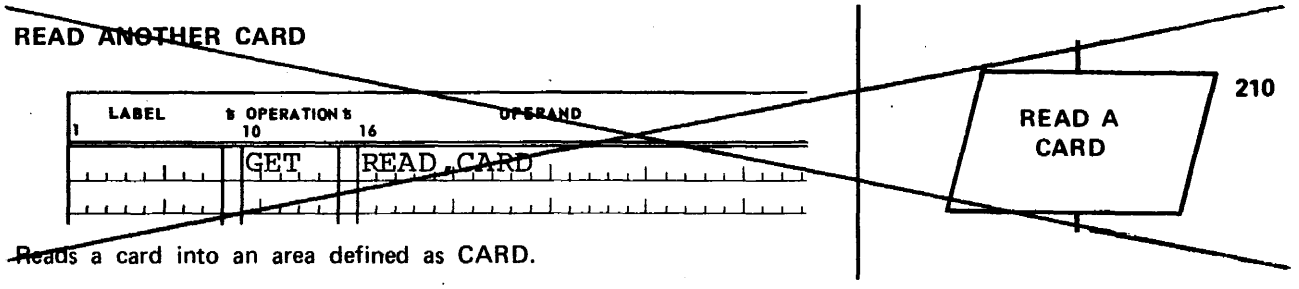
150

8. Page 3-11:

Delete coding line 210:

Column 10 GET

Column 16 READ,CARD



9. Page 3-18:

Delete coding line number 9:

Column 16 PROV = FOF,

Column 72 X

ASSEMBLER CODING FORM

PROGRAM MACRO CALL CARDS PROGRAMMER DATE PAGE OF PAGES

1 LABEL	10 OPERATION	16 OPERAND	COMMENTS	72	80
READ	DTFCR	EOFA=EOJ,		X	
		IOA1=RBUF,		X	
		ITBL=TBRD,		X	
		MODE=TRANS			
PRNT	DTFPR	BKSZ=132,		X	
		CNTL=YES,		X	
		PRAD=2,		X	
		PROV=FOF,		X	
		FONT=63			
	END				

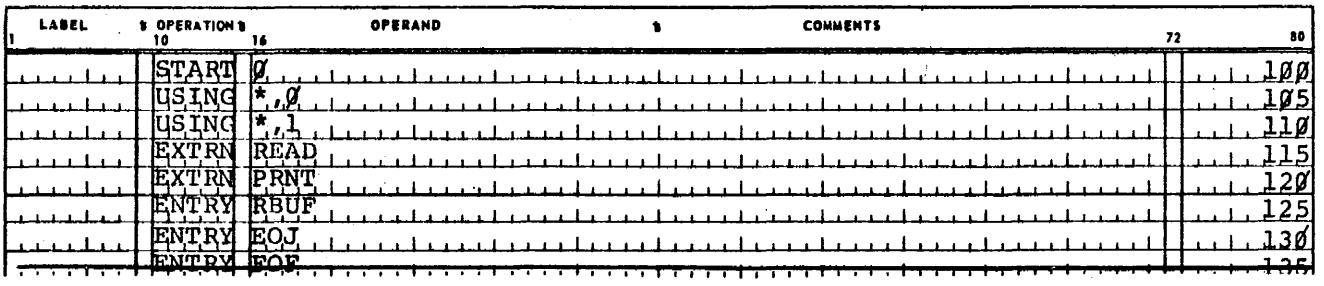
10. Page 3-19:

Delete coding line 135:

Column 10 ENTRY

Column 16 FOF

Column 78 135



LABEL	OPERATION	OPERAND	COMMENTS	
	START	Ø		1ØØ
	USING	*,Ø		1Ø5
	USING	*,1		11Ø
	EXTRN	READ		115
	EXTRN	PRNT		12Ø
	ENTRY	RBUF		125
	ENTRY	EOJ		13Ø
	ENTRY	FOF		135

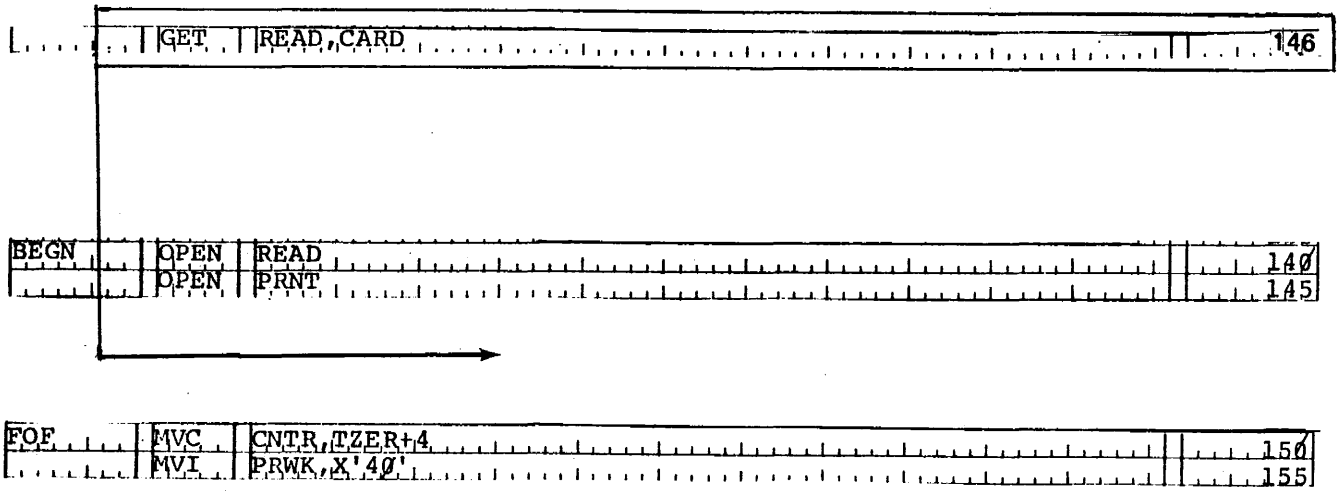
11. Page 3-19:

Insert coding line 146:

Column 10 GET

Column 16 READ,CARD

Column 78 146



	GET	READ,CARD	146
BEGN	OPEN	READ	14Ø
	OPEN	PRNT	145
FOF	MVC	CNTR,TZER+4	15Ø
	MVI	PRWK,X'4Ø'	155

12. Page 3-19:

Correct coding line 205:

Column 10 PUT

Column 16 PRNT, PRWK

```
         MVC     PSMN(8),HDR2                                          190
         MVC     PSAL(5),HDR3                                          195
         CNTRL   PRNT,SP,1                                             200
         PUT     PRNT,PRWK                                             205
```

13. Page 3-19:

Delete coding line 210:

Column 10 GET

Column 16 READ, CARD

Column 78 210

```
         GET     READ,CARD                                             210
MAN      MVI     PRWK,X'40'                                            215
         MVC     PRWK+1(131),PRWK                                      220
```