UNIVAC

DATA PROCESSING DIVISION

1050 SYSTEMS

REGENT

REFERENCE MANUAL

This manual is published by the UNIVAC Division of Sperry Rand Corporation in loose leaf format as a rapid and complete means of keeping recipients apprised of UNIVAC ® Systems developments. The UNIVAC Division will issue updating packages, utilizing primarily a page-for-page or unit replacement technique. Such issuance will provide notification of hardware and/or software changes and refinements. The UNIVAC Division reserves the right to make such additions, corrections, and/or deletions as in the judgment of the UNIVAC Division, are required by the development of its respective Systems.

PRINTED IN U.S.A.

I. CONTENTS

APPENDIX

ILLUSTRATION

2. INTRODUCTION

A. GENERAL DESCRIPTION

REGENT, a problem oriented programming system and report program generator, is designed to reduce substantially the time and effort necessary to translate general data processing and reporting requirements into detailed computer instructions. It demands little knowledge of computer coding or instructions other than the basic rules for writing in the simplest form of the PAL assembly language. Essentially, the REGENT report program generator is a program which, on the basis of a series of statements provided to it, produces another program which will produce a report or other output of the desired kind. These statements, written on the standard PAL coding form and then keypunched into cards provide:

- The formats of the input card files – these contain the information from which the report is to be prepared.

- The format of the output to be produced – this may be a printed document, a series of summary cards, or both.

- The operations to be performed – arithmetic operations, data movement and editing, control, input/output operations.

The input and output format descriptions and the processing statements will, in conjunction with REGENT, produce an efficient source program, ready for assembly by the PAL Assembler. Also provided is a listing of source input and, if desired, the object coding generated. Sections of programmer's own code may be included as necessary.

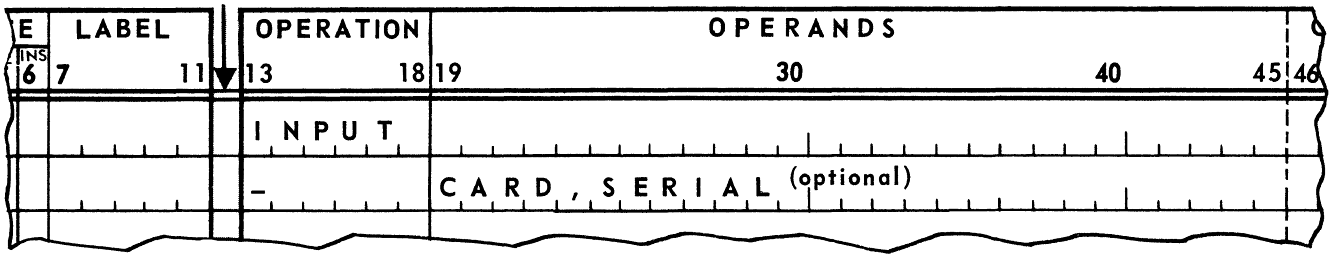
The following pages present the rules for the expression of information for use by the UNIVAC 1050 report program generator.

B. THE EXPRESSION OF INFORMATION

1. I/O Field Description

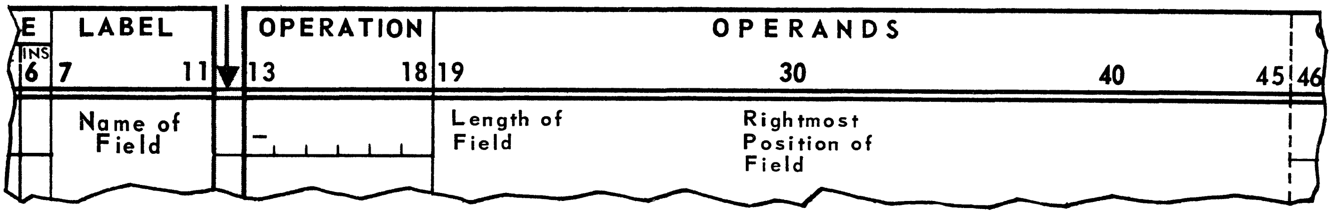
a. Input File Description (Card)

An input card file is described by a series of field definitions which must be supplied to the generator in a group. The first card of the group must contain the entry INPUT in the operation field, with the label field and the operand field left blank. The following card contains a blank label field and minus sign (–) in the operation field, and CARD as a first operand. A second operand, SERIAL, must appear if the object program accepts input from a 90-column Column Reader. The first two cards will appear as



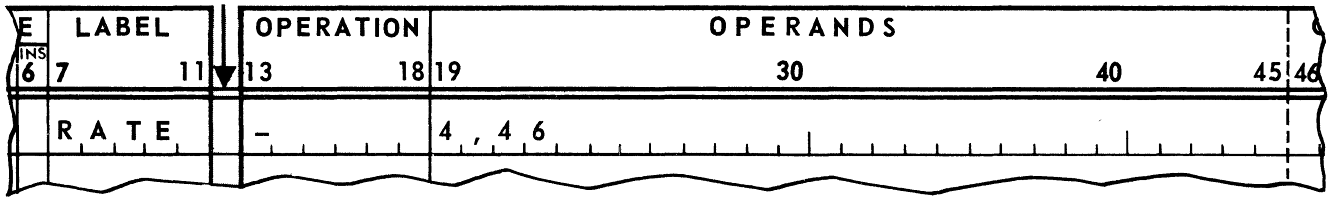
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		INPUT	
		–	CARD, SERIAL (optional)

All fields of the input card file, which are used in arithmetic operations or for control purposes, or which contain information which is to appear in the report, must be given names (labels). The fields are described as follows:



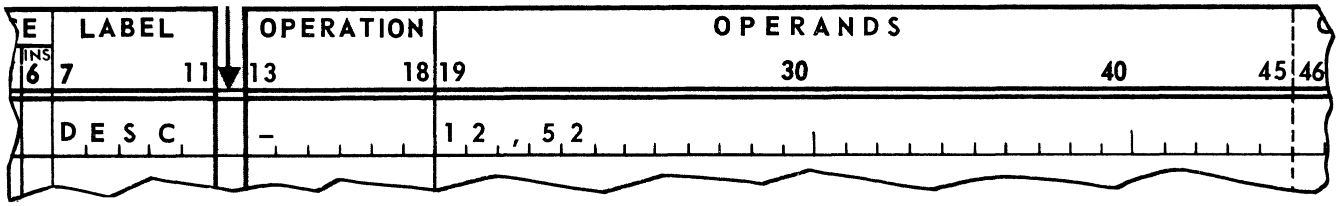
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	Name of Field	–	Length of Field Rightmost Position of Field

For example, a four character (RATE) field in columns 43-46 is described as



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	RATE	–	4,46

Similarly, a twelve character (Description) field in columns 41-52 is described as

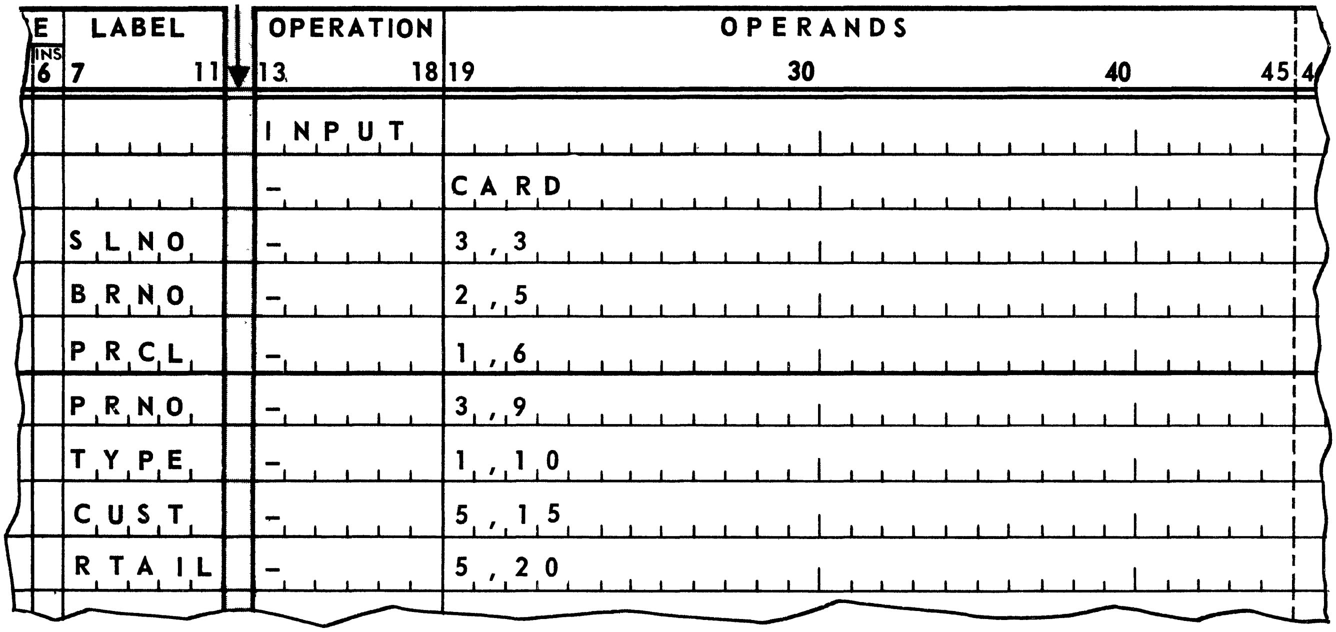


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	DESC	–	12,52

Thus an input card file which contains the following:

COLUMNS	CONTENTS
1-3	Salesman Number
4-5	Branch Number
6	Product Class
7-9	Product Number
10	Type of Sale (N for New and R for Recurring)
11-15	Customer Number
16-20	Retail Amount of Sales

may be described as:



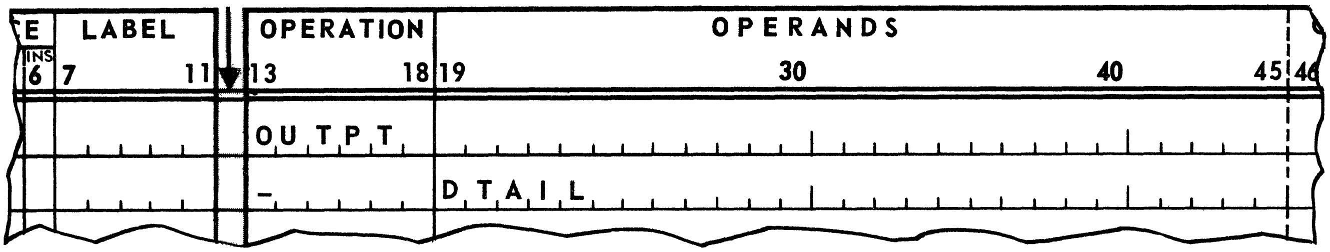
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
		INPUT	
		–	CARD
	SLNO	–	3,3
	BRNO	–	2,5
	PRCL	–	1,6
	PRNO	–	3,9
	TYPE	–	1,10
	CUST	–	5,15
	RTAIL	–	5,20

NOTE: Fields must not exceed 16 characters in length.

b. Output Description

(1) Detail Lines

Detail lines are the type of lines most often produced by the program, usually consisting of information from the input file. Here, as with the input description, each field in the output detail line is given a name (label). The fields are described in the same fashion as the input file giving the name of the field. All field descriptors for the detail line must be fed to the generator in a group. The first two cards of this group must indicate to the generator that the following field descriptors apply to the detail line:

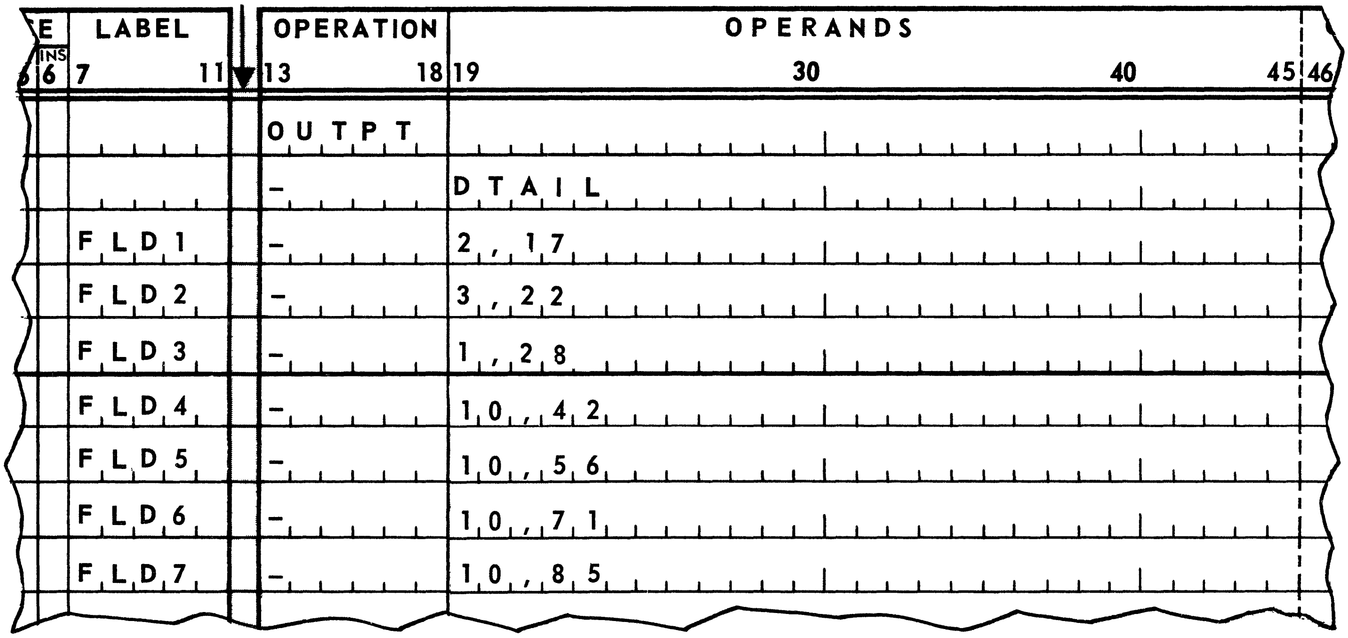


```
            OUTPT
            -     DTAIL
```

Thus, a detail line appearing as

XX XXX X XXX,XXX.XX XXX,XXX.XX XXX,XXX.XX XXX,XXX.XX

can be described as

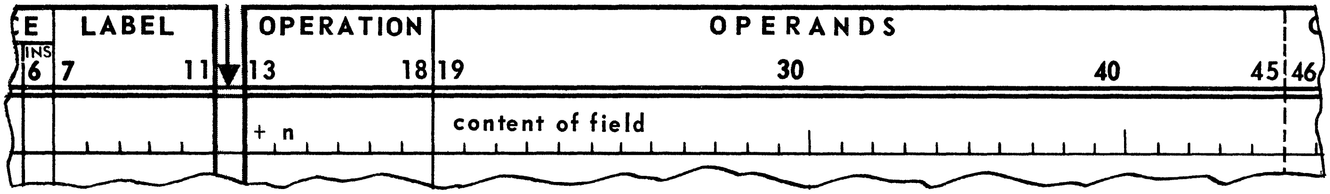


```
            OUTPT
            -     DTAIL
      FLD1  -     2,17
      FLD2  -     3,22
      FLD3  -     1,28
      FLD4  -     10,42
      FLD5  -     10,56
      FLD6  -     10,71
      FLD7  -     10,85
```

In this instance, FLD1 is two characters in length, and its rightmost character will appear in print position 17. FLD2 is three characters in length, and its rightmost character will appear in print position 22. FLD7 is 10 characters long, and its rightmost character will appear in print position 85.

(2) Nondetail Lines

The format of nondetail lines is given to the report generator by indicating the fields which contain constant information, those which contain variable information, and those which are blank. The format is as follows:



```
            + n   content of field
```

where n is the number of characters in the field.

Each nondetail line description is preceded by the following instruction:

E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
	Name of Line		OUTPT	NONDT	

For example, to describe a line of the following format:

OCTOBERƀ1962ƀPAGEƀXX[1]

the following may be written:

E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
	LINE1		OUTPT	NONDT	
			+47	0	
			+12	'OCTOBERƀ1962'	
			+17	0	
			+5	'PAGEƀ'	
	PGNO		+2	0	
			+45	0	

This description specifies that print positions 1 through 47 will be blank; that the next 12 print positions will contain OCTOBER 1962;[2] that the next 17 positions will be blank; the next 5 positions will contain PAGE and a space; the next two positions are reserved for page number (to be provided in the program by means of the label PGNO); and the last 45 positions will be blank.

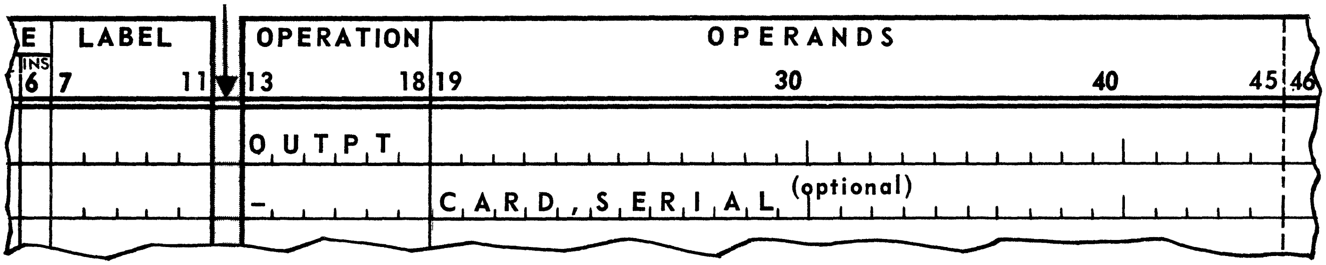
Note that the variable information field (the page number field) has a name to allow for assignment of a value. The sum of the fields described for each print line must be 128. Blank areas such as the 1st, 3rd, and last entries in the example above may be as large as 128 characters. Fields containing information may not exceed 16 characters.

[1] *ƀ ASCII code for space.*

[2] *Alphabetic, numeric, or alphanumeric constants are written enclosed in apostrophes.*

(3) Summary Cards

All fields of the output summary card are given names. The fields are described by giving the name of the field, the length of the field, and the rightmost position of the field. All field descriptions for the summary card are fed to the generator as a group. The first two cards of this group indicate to the generator that the following are field descriptions for the summary card; if a 90-column Column Punch is utilized, a second operand, SERIAL, must appear on the second card. The first two cards appear as follows:

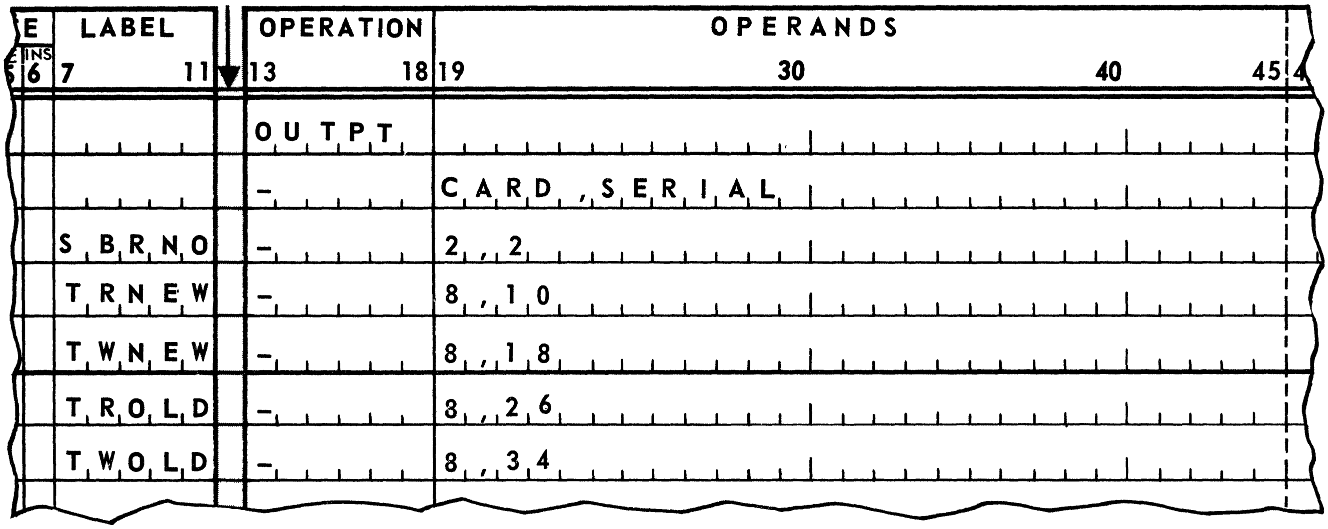


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		OUTPT	
		-	CARD,SERIAL (optional)

Thus a summary card which contains the following:

COLUMNS	CONTENTS
1-2	Branch Number
3-10	Total Retail New Business
11-18	Total Wholesale New Business
19-26	Total Retail Old Business
27-34	Total Wholesale Old Businesss

can be described as:

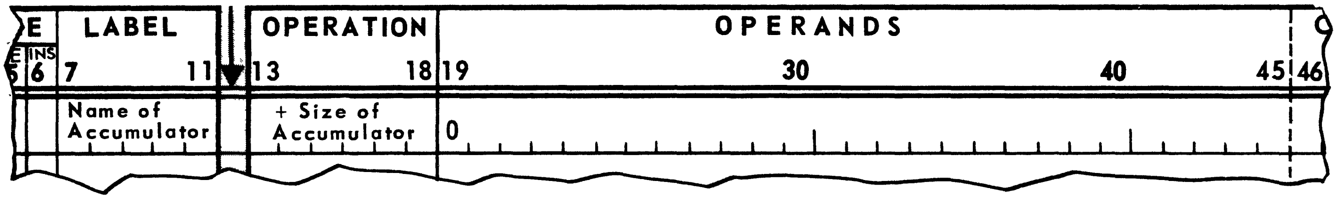


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 4
		OUTPT	
		-	CARD,SERIAL
	SBRNO	-	2,2
	TRNEW	-	8,10
	TWNEW	-	8,18
	TROLD	-	8,26
	TWOLD	-	8,34

NOTE: Fields must not exceed 16 characters in length.

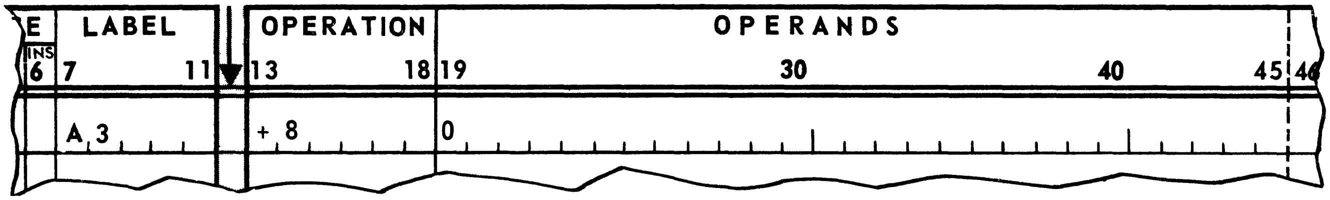
2. Accumulators

The size and name of each Accumulator is provided to the generator as follows:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
	Name of Accumulator	+ Size of Accumulator	0	

To define Accumulator Number 3 (A3) as 8 digits in length and with an initial value of zero, the following line is written:

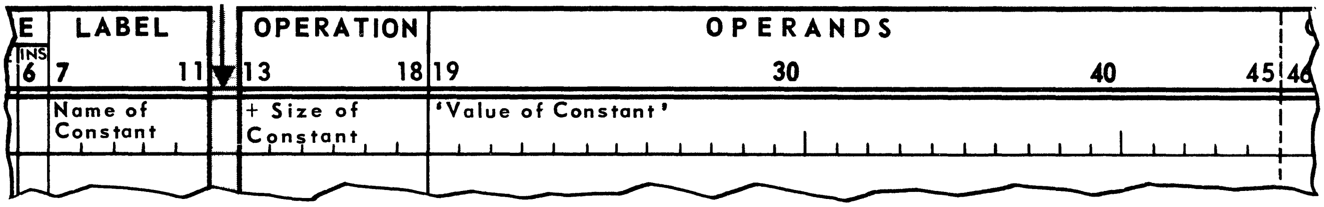


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
	A3	+8	0	

NOTE: Fields must not exceed 16 characters in length.

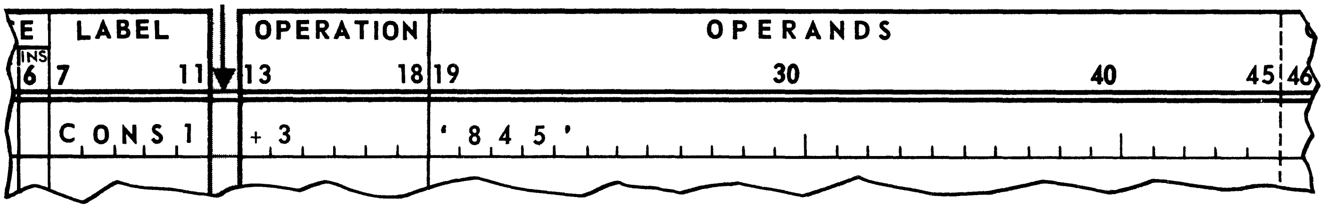
3. Constants

The size and value of each constant required is provided to the generator as



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
	Name of Constant	+ Size of Constant	'Value of Constant'	

To define the decimal constant *845*, the following line is written:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
	CONS1	+3	'845'	

To define the constant TOTAL COST, the following line is written:

E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
	CONS3	+10	'TOTAL COST'

NOTE: Fields must not exceed 16 characters in length.

4. Edit Masks

The mask described for editing is a picture of the resultant edited field. An '@' appears in the mask in each position of the edited field which is to receive a character from the unedited field. Thus the mask

@@,@@@.@@

is used to edit a field of dollars and cents for a value less than $100,000.00. The mask is defined as follows:

E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
	Name of Mask	+ Number of Characters In Mask	'Value of mask'

The above mask can be defined as

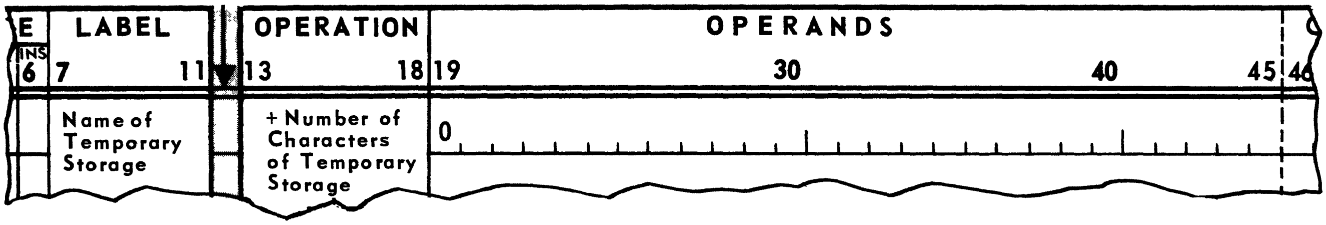
E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
	MASK1	+9	'@@,@@@.@@'

If the least significant character of the mask is a hyphen (–), the rightmost position of the edited field will be a blank if the value of the unedited field is positive, and a – if the value of the unedited field is negative.

NOTE: Fields must not exceed 16 characters in length.

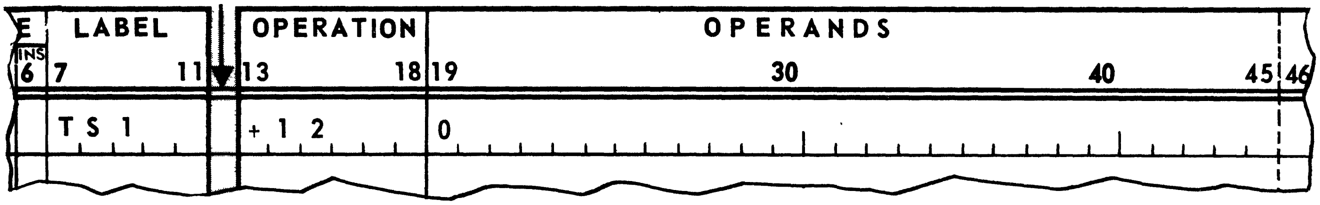
5. Temporary Storage Registers

On occasion, temporary storage is required for maintaining intermediate results. Temporary storage is assigned by writing:



E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45	46
	Name of Temporary Storage		+ Number of Characters of Temporary Storage	0	

For example,



E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45	46
	TS1		+12	0	

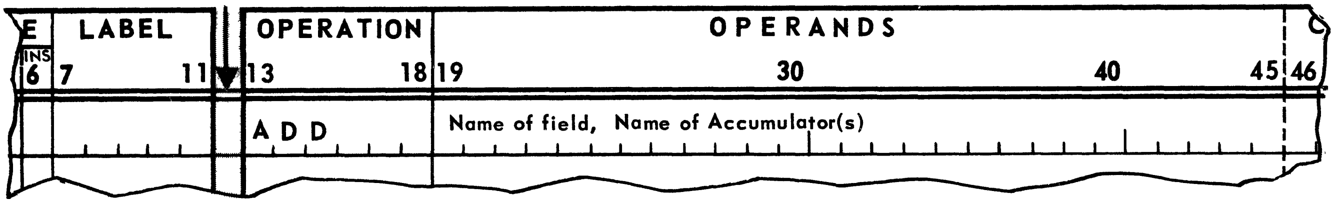
reserves a 12 character storage area for temporary results.

NOTE: Fields must not exceed 16 characters in length.

3. ARITHMETIC OPERATIONS

Processing information is supplied to the generator in the form of statements. The following section describes the arithmetic operations available. All arithmetic performed by the object code is decimal.

A. ADD A FIELD TO AN ACCUMULATOR

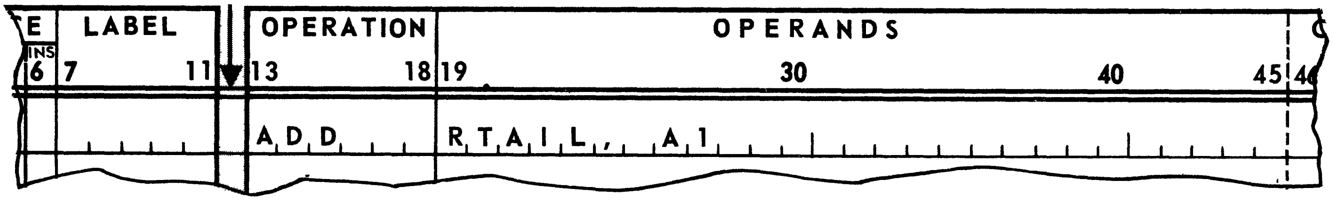


E	LABEL	OPERATION	OPERANDS	
INS 6	7 11	13 18	19 30 40 45	46
		ADD	Name of field, Name of Accumulator(s)	

The contents of the field named in the first expression are added algebraically to the accumulator(s) named in the second expression. If more than one accumulator is named, the addition is performed to each one named. A maximum of 5 accumulators may be named in each ADD operation. The contents of the first named field remain unchanged.

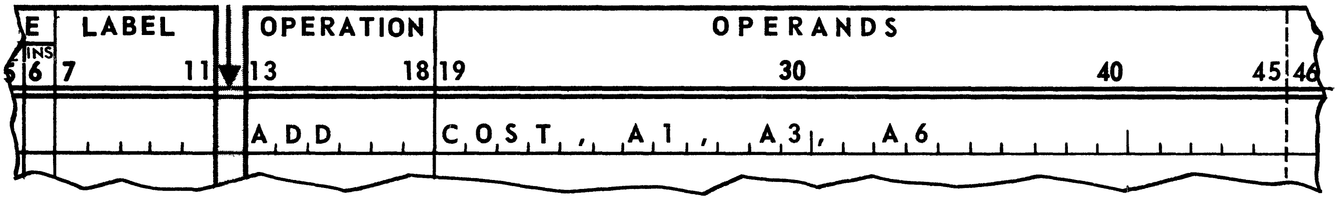
Examples:

Add retail amount (RTAIL) to accumulator 1 (A1):



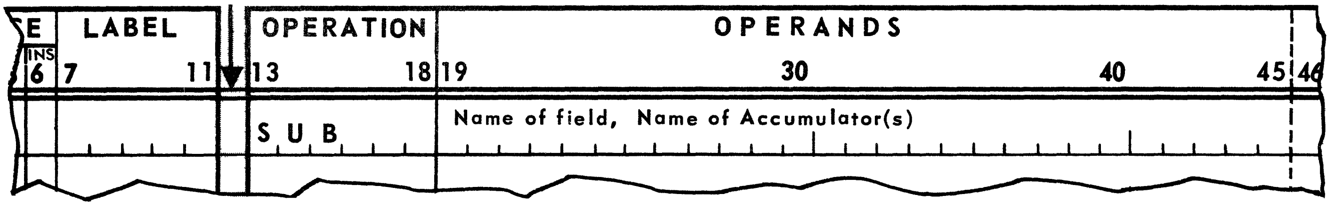
E	LABEL	OPERATION	OPERANDS	
INS 6	7 11	13 18	19 30 40 45	46
		ADD	RTAIL, A1	

Add cost (COST) to accumulators 1, 3, and 6 (A1, A3, and A6):



E	LABEL	OPERATION	OPERANDS	
INS 6	7 11	13 18	19 30 40 45	46
		ADD	COST, A1, A3, A6	

B. SUBTRACT A FIELD FROM AN ACCUMULATOR

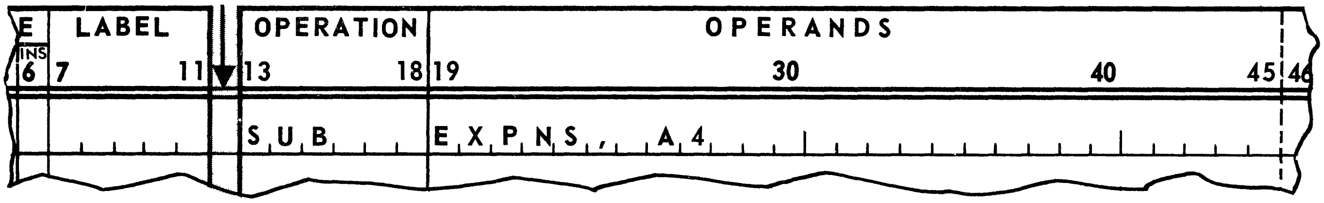


E INS 6	LABEL 7–11	OPERATION 13–18	OPERANDS 19–45
		SUB	Name of field, Name of Accumulator(s)

The contents of the field named in the first expression are subtracted from the accumulator(s) named in the second and subsequent expressions. A maximum of 5 accumulators may be named in a single subtract operation.The contents of the first named field remain unchanged.

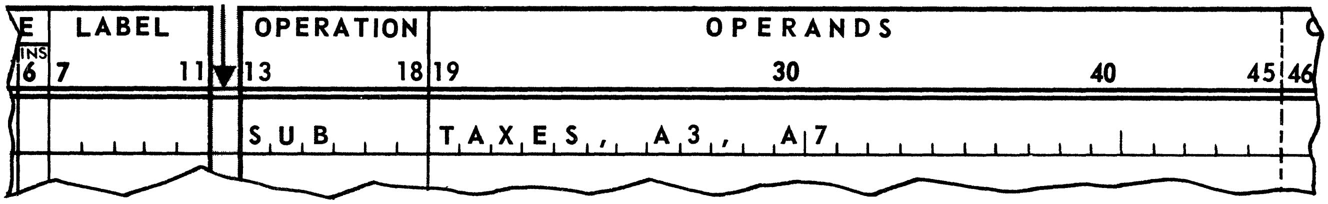
Examples:

Subtract expenses (EXPNS) from accumulator 4 (A4):



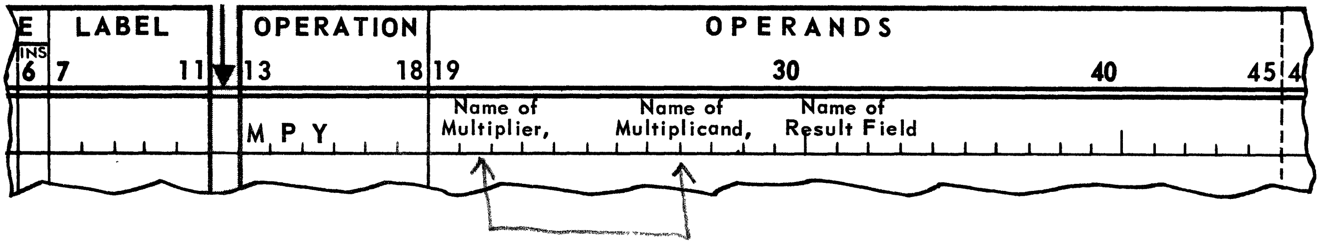
E INS 6	LABEL 7–11	OPERATION 13–18	OPERANDS 19–45
		SUB	EXPNS, A4

Subtract taxes (TAXES) from accumulators 3 and 7 (A3 and A7):



E INS 6	LABEL 7–11	OPERATION 13–18	OPERANDS 19–45
		SUB	TAXES, A3, A7

C. MULTIPLY TWO FIELDS

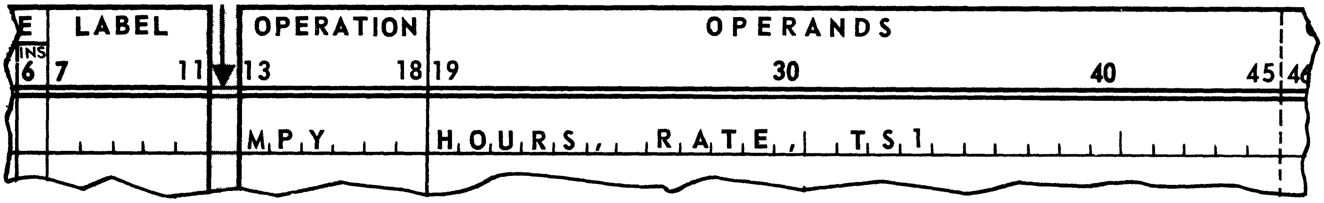


E INS 6	LABEL 7–11	OPERATION 13–18	OPERANDS 19–45
		MPY	Name of Multiplier, Name of Multiplicand, Name of Result Field

The contents of the field named in the first expression are multiplied by the contents of the field named in the second. The result is stored in the third named field. The length of the multiplier may not exceed 8 characters, and the combined length of the multiplier and multiplicand may not exceed 16.

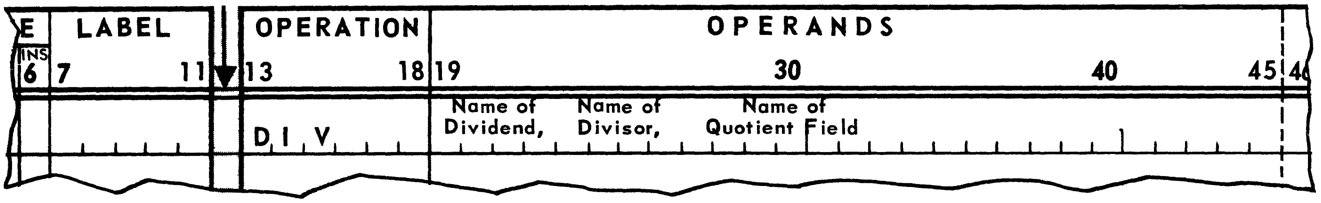
Example:

Multiply hours (HOURS) by a rate (RATE) and place result in a temporary storage (TS1):



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		MPY	HOURS, RATE, TS1	

D. DIVIDE ONE FIELD BY ANOTHER



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		DIV	Name of Dividend, Name of Divisor, Name of Quotient Field	

The contents of the first named expression are divided by the contents of the second; the result is stored in the third.

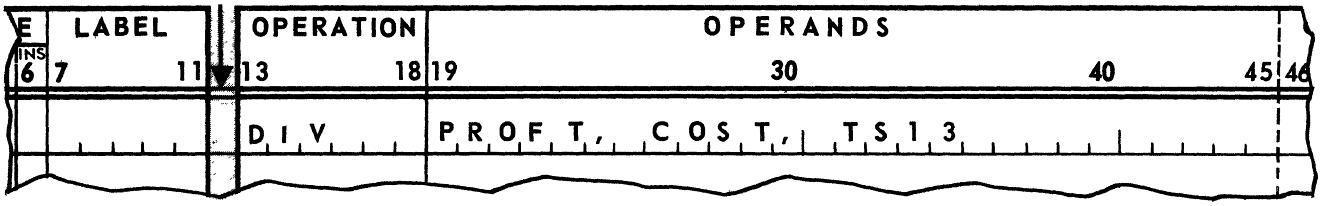
The object coding produced by REGENT prevents improper division from being attempted. UP-3912, the UNIVAC 1050 Central Processor Manual provides a description of the requirements which must be met if a valid divide operation is to be executed. The object program will produce the following results if any of these are violated.

- If the defined length of the quotient field is greater than 8 characters, an expression error associated with the divide instruction will occur at assembly time.

- If the length of the quotient plus the length of the divisor is less than dividend length, or exceeds 16 characters, the quotient field is filled with exclamation marks at the time the divide is executed.

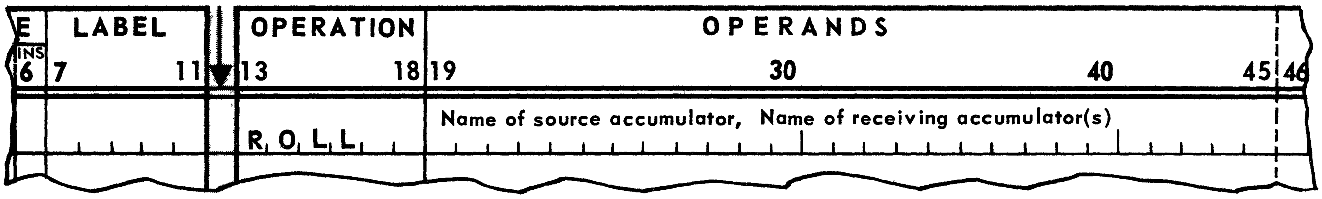
Example:

Divide profit (PROFT) by cost (COST) and place result in a temporary storage (TS13):



E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
		DIV	PROFT, COST, TS13

E. ADD TO ACCUMULATOR(S) AND RESET SOURCE

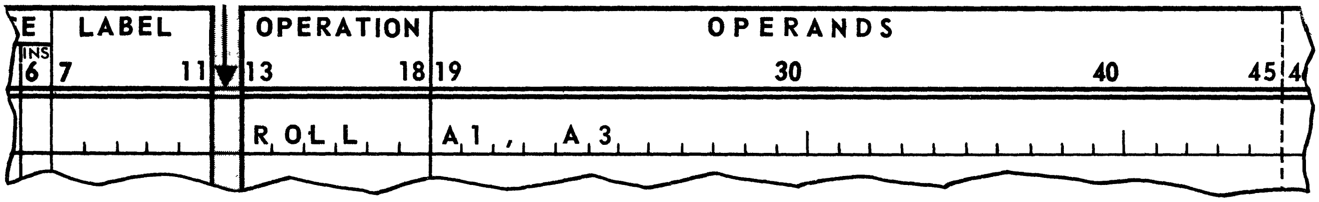


E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
		ROLL	Name of source accumulator, Name of receiving accumulator(s)

The contents of the first named accumulator are added to contents of the second and subsequent named accumulators. A maximum of 5 receiving accumulators may be named. The source accumulator is reset to zero.

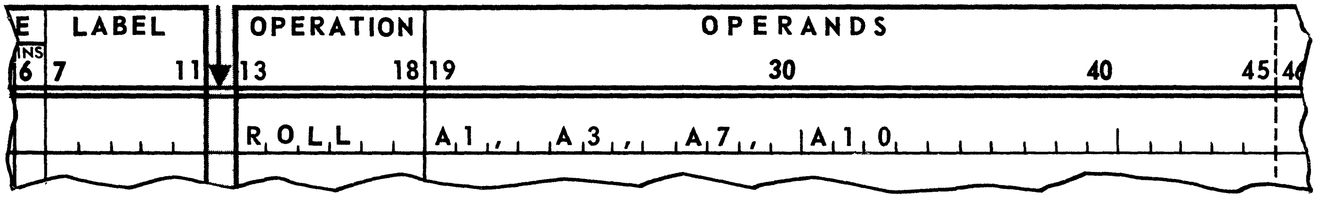
Examples:

Add accumulator 1 (A1) to accumulator 3 (A3) then reset accumulator 1:



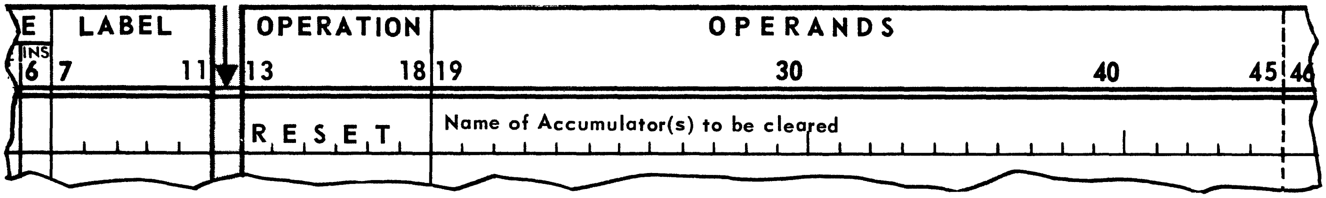
E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
		ROLL	A1, A3

Add accumulator 1 (A1) to accumulators 3, 7, and 10 (A3, A7, and A10) then reset accumulator 1:



E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
		ROLL	A1, A3, A7, A10

F. RESET ACCUMULATOR

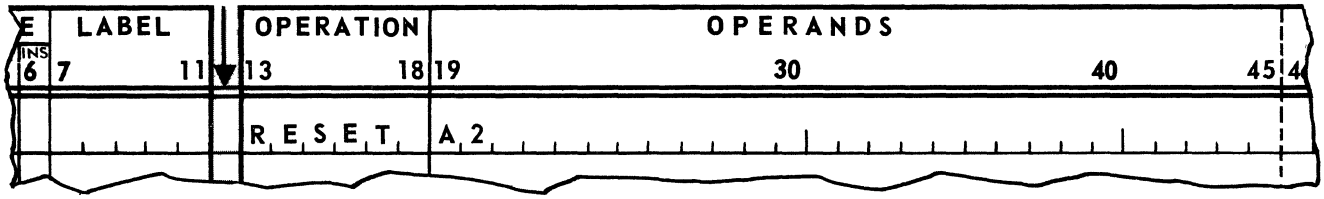


E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45	46
			RESET	Name of Accumulator(s) to be cleared	

Each accumulator named is cleared to zero. A maximum of 5 may be named.

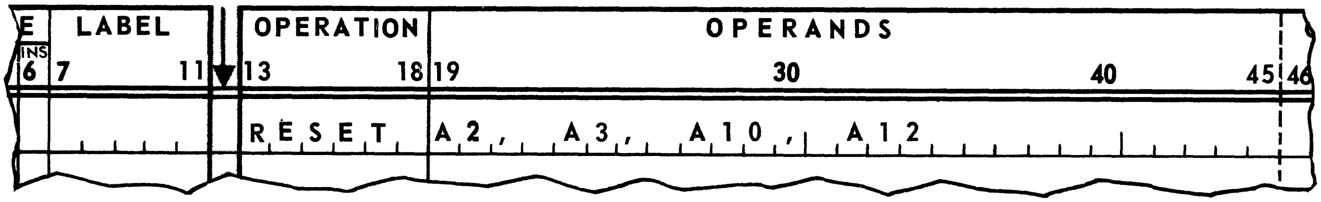
Examples:

Reset accumulator 2 (A2):



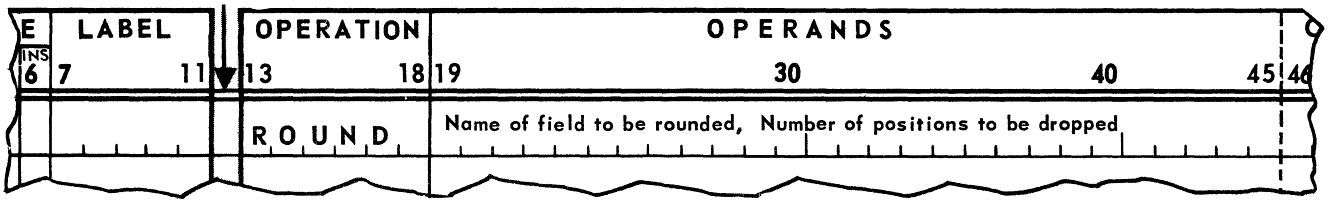
E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45	46
			RESET	A2	

Reset accumulators 2, 3, 10, and 12 (A2, A3, A10, and A12):



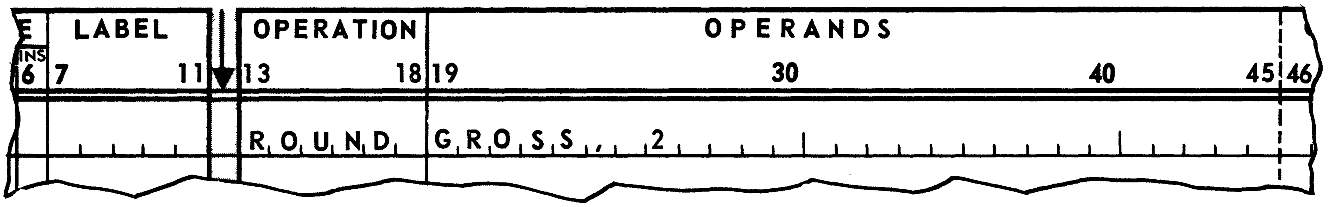
E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45	46
			RESET	A2, A3, A10, A12	

G. HALF ADJUST A FIELD (ROUND)



E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45	46
			ROUND	Name of field to be rounded, Number of positions to be dropped	

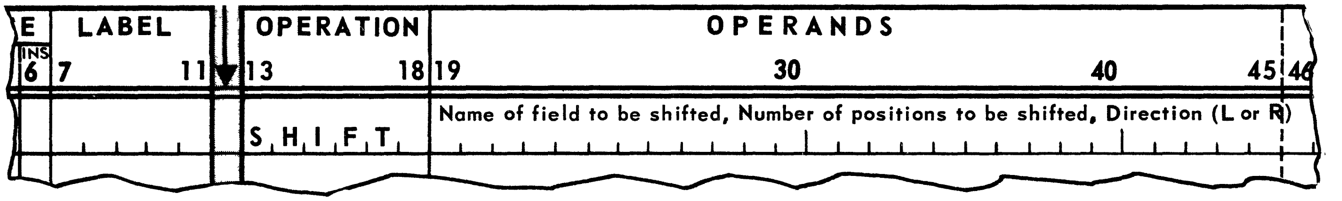
The field named in the first expression will be half adjusted and right justified with zeros inserted to the left. The sign of the field remains unchanged.



E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
		ROUND	GROSS, 2

	GROSS
Before	1 2 3 4 5 6
After	0 0 1 2 3 5

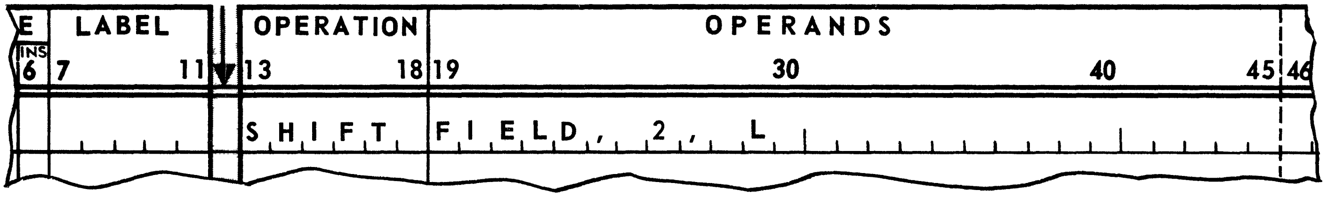
H. POSITION ADJUST A FIELD (SHIFT)



E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
		SHIFT	Name of field to be shifted, Number of positions to be shifted, Direction (L or R)

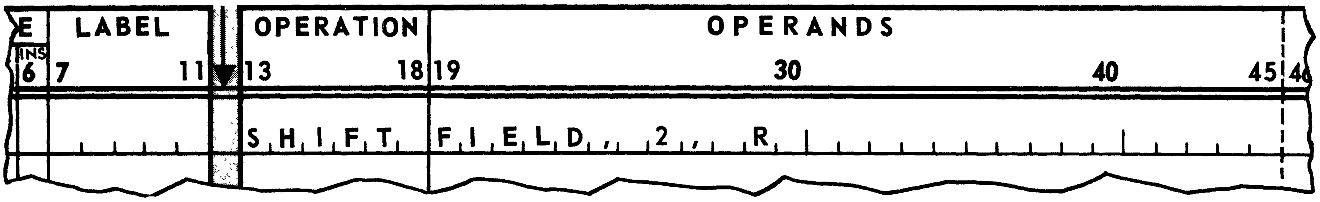
The contents of the named field are shifted destructively, that is, characters shifted beyond the limits of the field are lost. If the third expression is L, the field contents are shifted left with decimal zeros inserted at the right. If the third expression is R, the field is shifted right with zeros inserted at the left. The sign of the field remains unchanged.

Examples:



E	LABEL	OPERATION	OPERANDS
INS 6	7 11	13 18	19 30 40 45 46
		SHIFT	FIELD, 2, L

	FIELD
Before	1 2 3 4 5
After	3 4 5 0 0


E
LABEL
OPERATION
OPERANDS
INS
6
7
11
13
18
19
30
40
45
46

```
SHIFT FIELD,2,R
```

Before	1 2 3 4 5
After	0 0 1 2 3

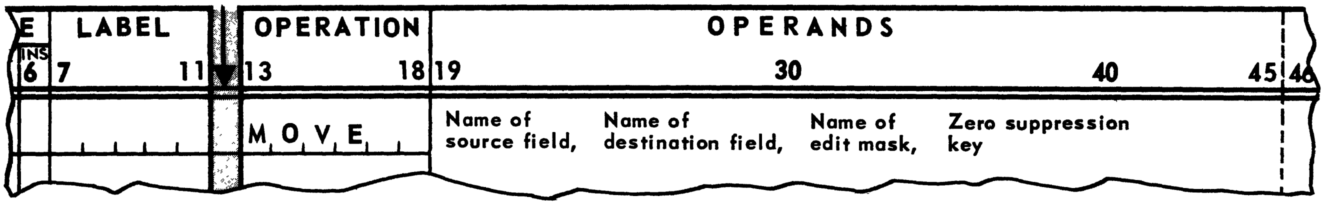
I. DECIMAL OVERFLOW

If decimal overflow occurs on any of the arithmetic operations, the object program will stop (display 11117). If the program is continued following the error stop, question marks (?) will be placed in the result field.

NOTE: In decimal operations, question marks and exclamation points are treated as decimal zeros.

4. DATA MOVEMENT AND EDITING

A. MOVE



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			MOVE	Name of source field, Name of destination field, Name of edit mask, Zero suppression key	

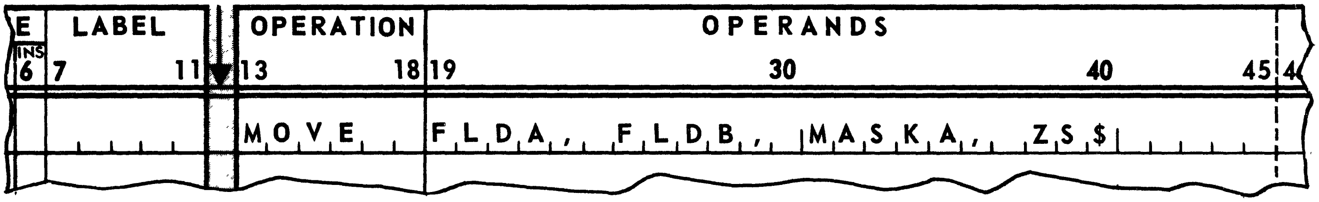
This operation is used to transfer the contents of one field to another with or without editing and/or zero suppression.

- To transfer information without editing or zero suppression only the first two expressions are written. Alphabetic or numeric information may be transferred in this manner.
- To edit decimal data while it is being transferred, the name of a previously defined edit mask is written as the third expression.
- If suppression of leading zeros is desired, one of the following keys is written as the fourth expression:

ZS	Leading zeros and commas are changed to blanks.
ZS*	Leading zeros and commas are changed to asterisks.
ZS$	Leading zeros and commas are changed to blanks, and a $ is inserted to the left of the most significant digit.

Zero suppression may be accomplished without editing if a comma is inserted in place of the third expression.

Examples:



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			MOVE	FLDA,FLDB,MASKA,ZS$	

FLDA	before	0012345
MASKA		@@,@@@.@@
FLDB	after	$123.45

```
E    LABEL      OPERATION  OPERANDS
INS
6  7      11   13      18  19          30                40        45 46
               MOVE        FLDA, FLDB, , ZS *
```

FLDA before 0012345

FLDB after **12345

Note:

If it is desired to preserve the remainder from a Divide, the MOVE operation may be used by writing only one expression, the name of the field into which the remainder of the Divide is to be stored. This special use of the MOVE may be written only if it immediately follows the Divide operation.

B. CLEAR

To clear an area to blanks, the programmer writes:

```
E    LABEL      OPERATION  OPERANDS
INS
6  7      11   13      18  19          30                40        45 46
               CLEAR       Name of area
```

Examples:

```
E    LABEL      OPERATION  OPERANDS
INS
6  7      11   13      18  19          30                40        45 46
               CLEAR       DTAIL
```

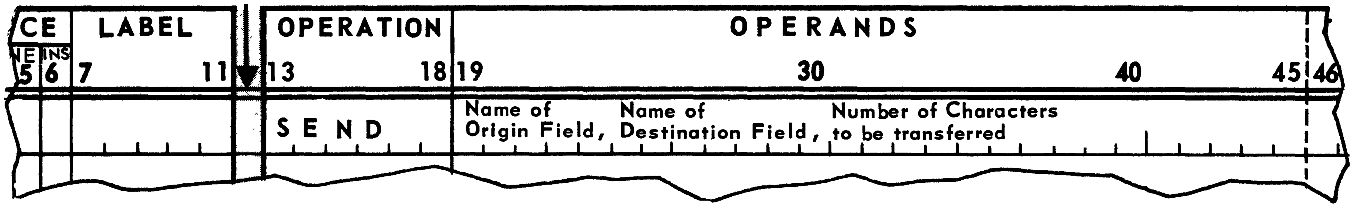
```
               CLEAR       CARD
```

```
               CLEAR       LABEL
```

This causes blanks to be inserted in the entire area of the operand. If the operand is DTAIL, the detail print area (128 characters) will be filled with blanks. If the operand is CARD, the card punch output area (first 90 characters) will be filled with blanks. If the operand is the label of a field (16 characters or less) the field defined by that label will be filled with blanks.

The CLEAR statement should be written *prior* to the creation of any DTAIL or CARD output information.

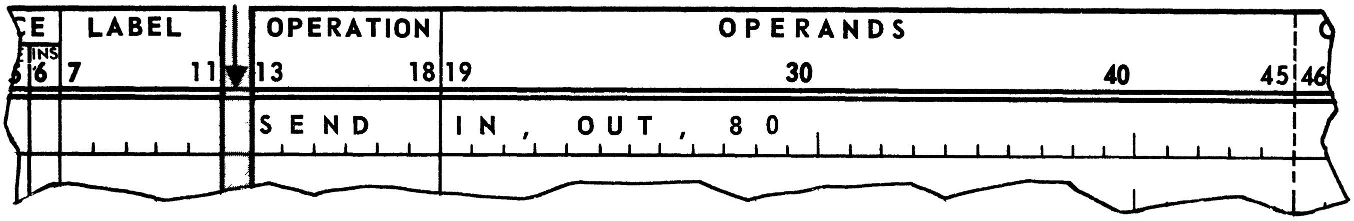
C. SEND



CE		LABEL		OPERATION	OPERANDS
NE 5	INS 6	7 11	▼	13 18	19 30 40 45 46
				SEND	Name of Origin Field, Name of Destination Field, Number of Characters to be transferred

This operation causes a block transfer of up to 1024 characters.

Example:



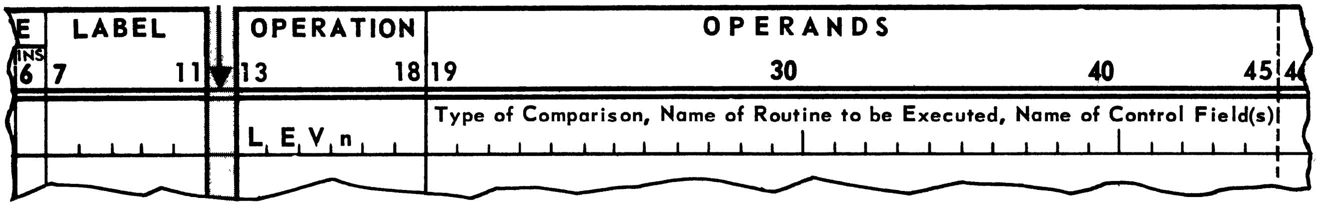
CE		LABEL		OPERATION	OPERANDS
5	INS 6	7 11	▼	13 18	19 30 40 45 46
				SEND	IN, OUT, 80

The 80 characters of information beginning at IN are transferred to OUT and successively higher positions.

5. PROGRAM SEQUENCE CONTROL

The sequence of steps executed by the generated program is determined by the order of input to the report generator. The following are methods whereby the sequence of execution can be altered, based on control breaks or the existence of certain conditions.

A. TOTAL LEVELS (CONTROL BREAKS)



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45 46
			LEVn	Type of Comparison, Name of Routine to be Executed, Name of Control Field(s)

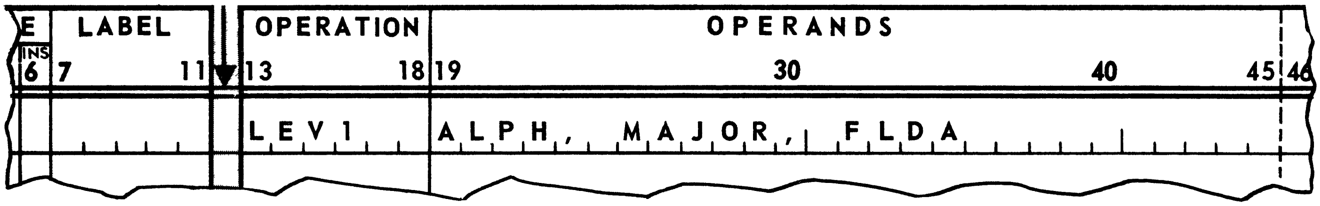
Testing for the presence of a control break is accomplished by writing a LEVn statement. Such a statement consists of the entry LEVn in the operation field where n is a number from 1 through 9, and three, four, or five expressions in the operand field. The first expression will be ALPH, DEC, or ZONE denoting the type of comparison desired. Where the first expression is ALPH or DEC, the second expression is the label of the routine (see Routines, 5–D) to be executed if a change in the control field(s) is detected; the third, fourth, and fifth expressions are labels of control fields which are to be tested. Only one control field is required, but up to three may be used if desired. Where more than one is written, the control break will occur if *any* of the fields change.

If the first expression is ZONE, the second expression is the label of the routine to be executed if the control condition is met, the third expression is the label of a single character field in a card input file, and the fourth expression is 11, 12, or 0 to denote the zone punch to be tested. If the label in the third expression describes a multicharacter field, the least significant character will be tested.

The highest level of control is given the lowest level number, and the lowest level of control is given the highest level number, LEVn statements are written in the order of highest level to lowest level.

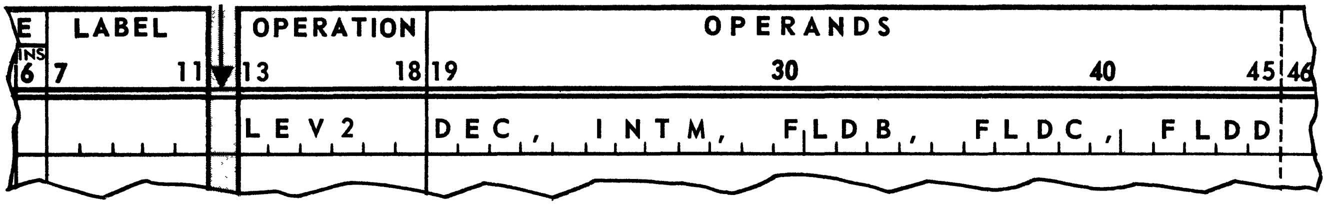
In the generated program, when a total level (control break) occurs, the occurrence of all higher numbered total levels are inhibited. Therefore, the routine executed when a total level occurs should include the execution of the next lower level.

Examples:



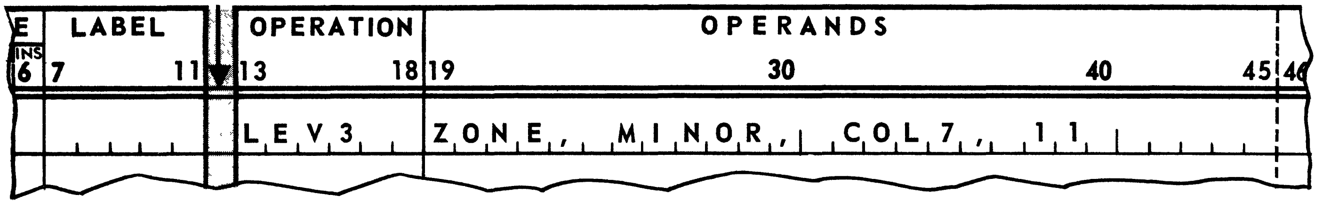
E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45 46
			LEV1	ALPH, MAJOR, FLDA

In this case, an alphabetic comparison is made and if there is a change in the field labeled FLDA, the routine labeled MAJOR is executed.



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
		LEV2	DEC, INTM, FLDB, FLDC, FLDD

A decimal comparison is made, with zones ignored except the sign zone, and the routine labeled INTM is executed when there is a change in the fields labeled FLDB, FLDC, or FLDD.



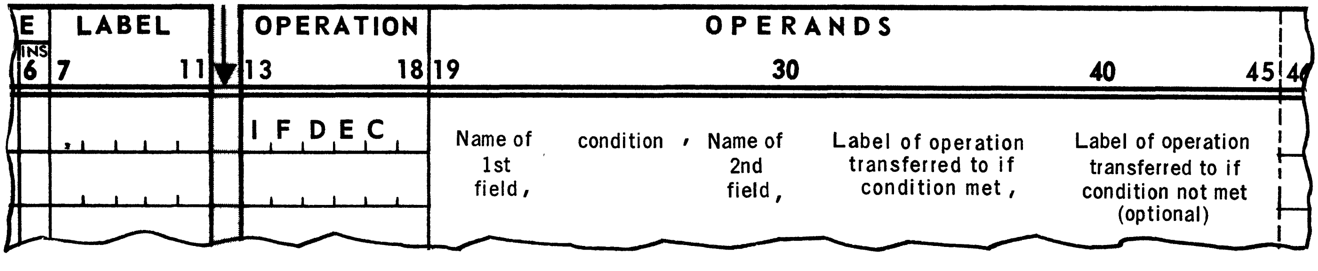
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
		LEV3	ZONE, MINOR, COL7, 11

In this case, when the single character field labeled COL7 contains an 11 punch, the routine labeled MINOR will be executed.

In the particular sequence of LEV statements above, the occurrence of a change in FLDA will cause the execution of the routine labeled MAJOR, the LEV2 and LEV3 statements will have no effect, and, on the completion of MAJOR, the next operation to be performed will be that which follows the LEV3 statement. If FLDA has not changed, the LEV2 statement will cause a test for a change in FLDB, FLDC, or FLDD. If a change has occurred, INTM will be executed, the LEV3 statement will have no effect, and the next operation will be that specified on the line following LEV3. If neither FLDA, FLDB, FLDC, nor FLDD have changed, the LEV3 statement will cause a test for an 11 punch in the single character field label COL7. If the condition is met, MINOR will be executed.

B. COMPARISON OF FIELDS

Two fields may be compared, and program sequence changed based on their relationship, using the IFDEC or IFALP directives.



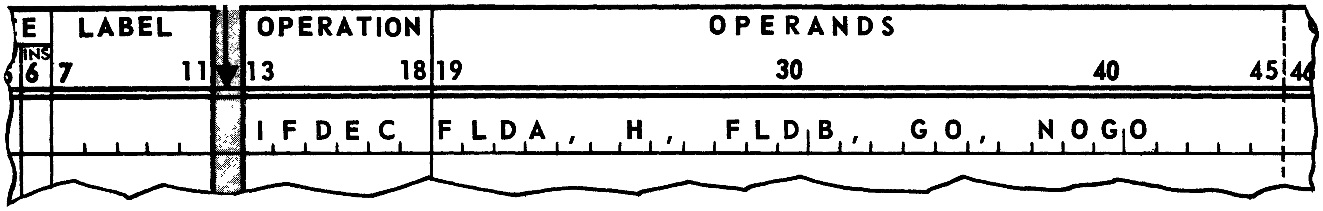
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
	,	IFDEC	Name of 1st field, condition , Name of 2nd field, Label of operation transferred to if condition met , Label of operation transferred to if condition not met (optional)

IFDEC causes a decimal algebraic comparison.

If the condition (2nd operand) is	*The condition is met when*
E	1st field = 2nd field
U	1st field ≠ 2nd field
H	1st field > 2nd field
L	1st field < 2nd field

The last expression in the operands field in all compare operations is optional. If it is blank, the next operation in sequence is performed when the condition is not met.

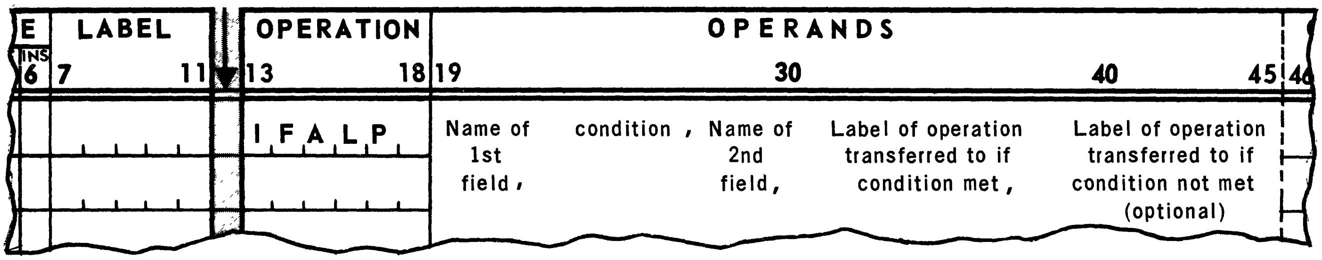
Example:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		IFDEC	FLDA, H, FLDB, GO, NOGO	

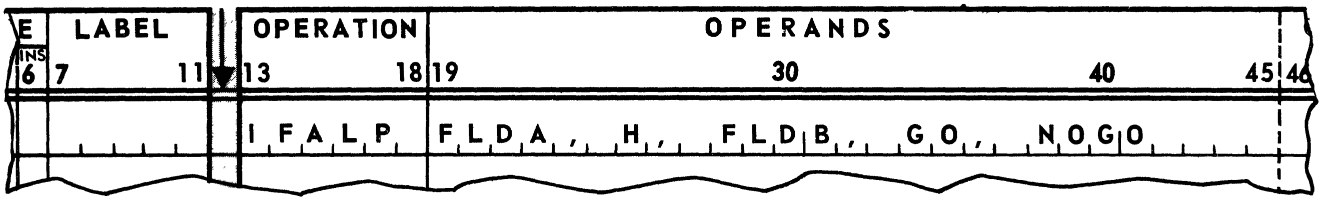
If the value of FLDA is 101 and the value of FLDB is 100, the condition H is met and control will be transferred to the operation labeled GO. If the condition is not met, control will, in this case,be transferred to the operation labeled NOGO. If the last expression were blank, the next instruction in sequence would be executed.

IFALP causes an alphabetic (binary) comparison.



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		IFALP	Name of 1st field, condition , Name of 2nd field, Label of operation transferred to if condition met , Label of operation transferred to if condition not met (optional)	

Example:

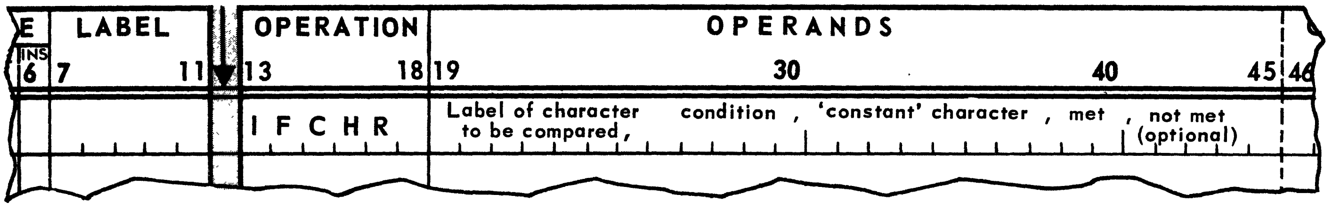


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		IFALP	FLDA, H, FLDB, GO, NOGO	

If the value of FLDA is 99 and the value of FLDB is 100, the condition is not met and control will be transferred to the operation labeled NOGO. If the fifth expression is blank, the next instruction in sequence will be executed.

C. COMPARISON OF CHARACTER, ZONE, AND SIGNS

1. Character Comparison

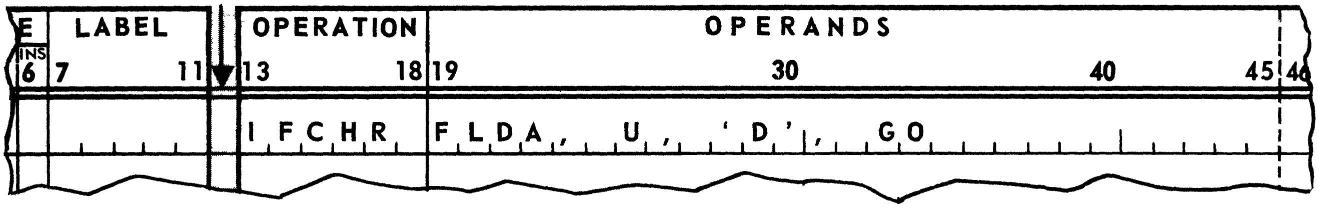


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		IFCHR	Label of character to be compared, condition , 'constant' character , met , not met (optional)	

This operation is the same as IFDEC and IFALP except that only a single character is compared to the actual character designated by the 3rd operand. If the first named expression is longer than 1 character, only the least significant character is compared.

NOTE: If it is desired to test for a blank, the third operand is a zero (no apostrophes).

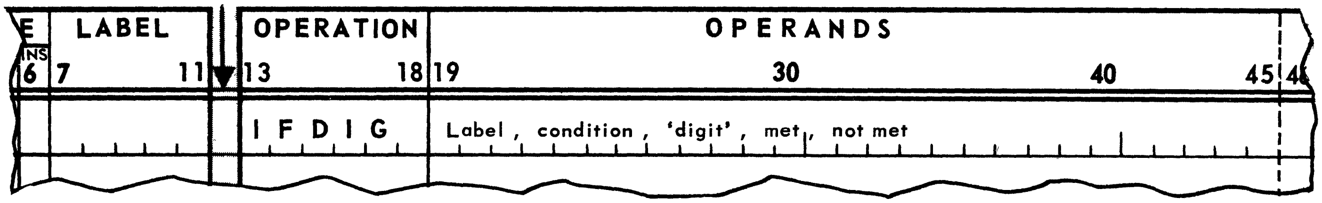
Example:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		IFCHR	FLDA, U, 'D', GO	

If the value of FLDA is ABCD, since 'D' is equal to the least significant character of FLDA, the unequal (U) condition has not been met. There is no fifth expression, therefore the operation following next in sequence will be performed next.

2. Digit Comparison



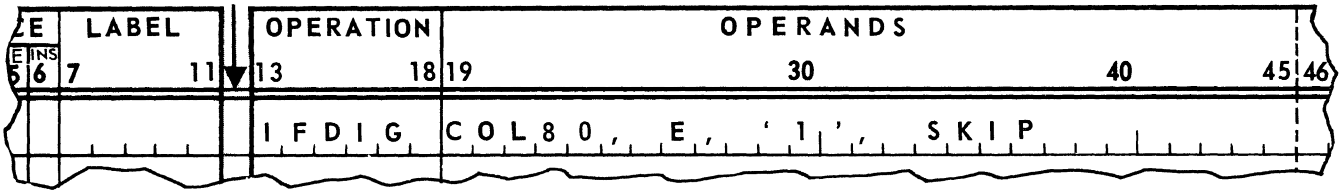
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		IFDIG	Label , condition , 'digit' , met , not met	

This operation is the same as IFCHR with two exceptions:

(a) The third operand will be a decimal number from 0 through 9.

(b) Zone bits are ignored in the comparison .

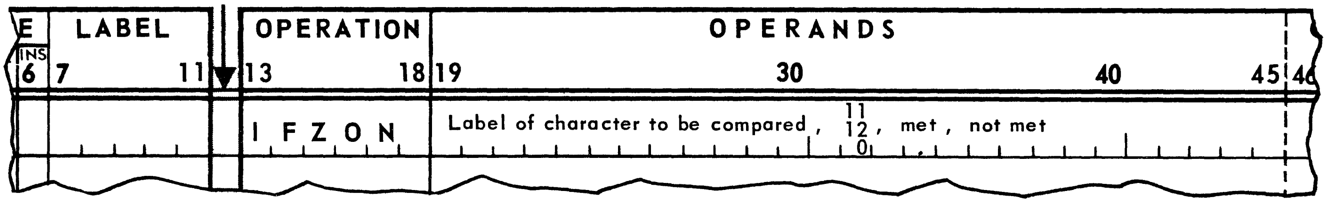
Example:



LABEL	OPERATION	OPERANDS
	IFDIG	COL80, E, '1', SKIP

If the character labeled COL80 contains a 1 punch, the equal condition is met, regardless of the zone portion, and control will be transferred to the operation labeled SKIP.

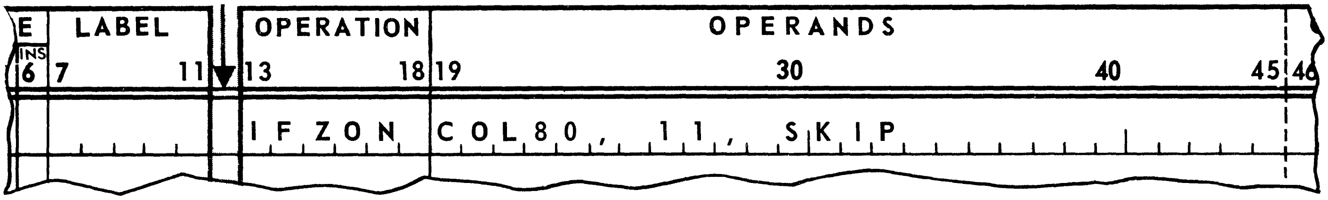
3. Zone Comparison



LABEL	OPERATION	OPERANDS
	IFZON	Label of character to be compared, 11 12 0, met, not met

The IFZON operation always tests for the presence of the zone punch designated by the second operand which must be 11, 12, or 0. The operation will normally be used to test for the existence of a zone punch in an 80 column card where the remaining portion of the column (digit) has a separate meaning.

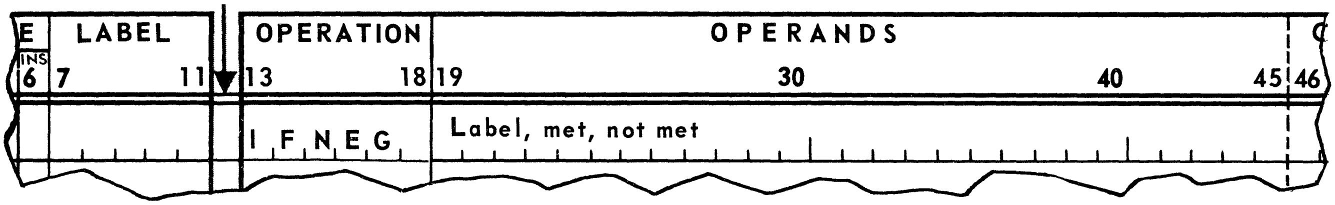
Example:



LABEL	OPERATION	OPERANDS
	IFZON	COL80, 11, SKIP

If the character labeled COL80 contains an 11 punch, the condition is met, regardless of the digit (1-9) portion.

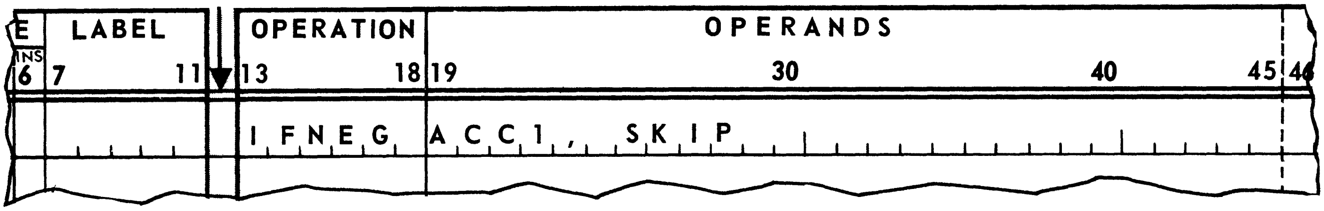
4. Sign Comparison



LABEL	OPERATION	OPERANDS
	IFNEG	Label, met, not met

The field whose label is designated by the 1st operand is tested for negative value. If it is negative, the condition is met.

Example:



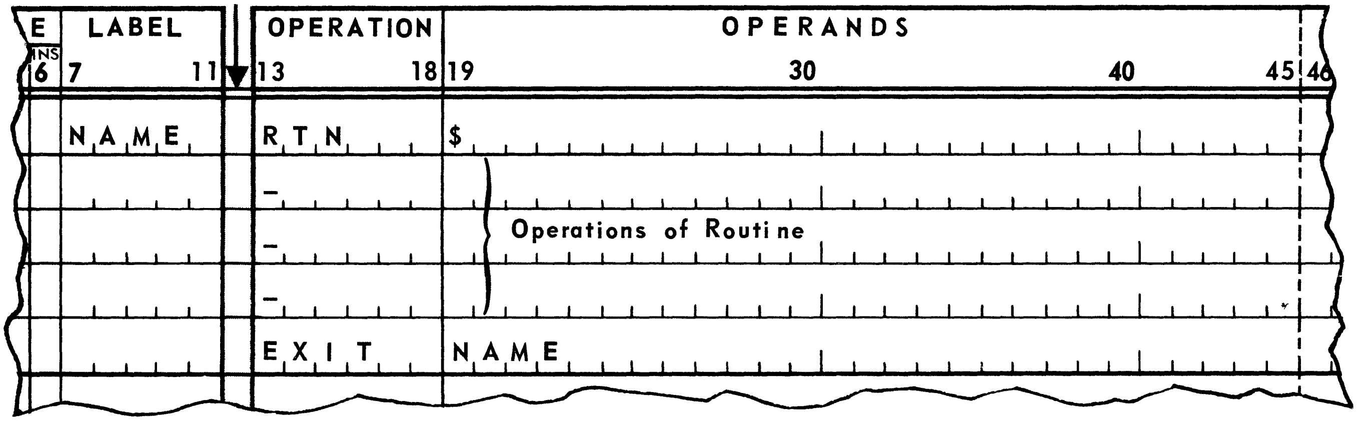
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		IFNEG	ACC1, SKIP

If the accumulator named ACC1 contains 12345–, control is transferred to SKIP.

D. ROUTINES

Routines are a series of operations which are performed when called for.

These operations are designated as follows:



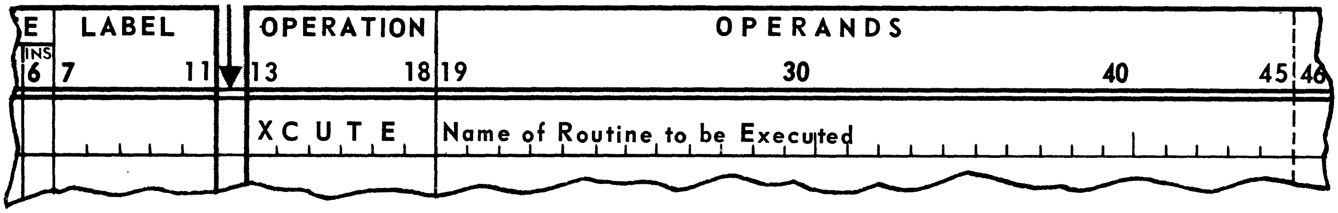
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	NAME	RTN	$
		–	Operations of Routine
		–	
		–	
		EXIT	NAME

The lines between RTN and EXIT will be executed as a closed subroutine. NAME for these two lines must be the same, the name of the routine.

E. EXECUTION OF ROUTINES

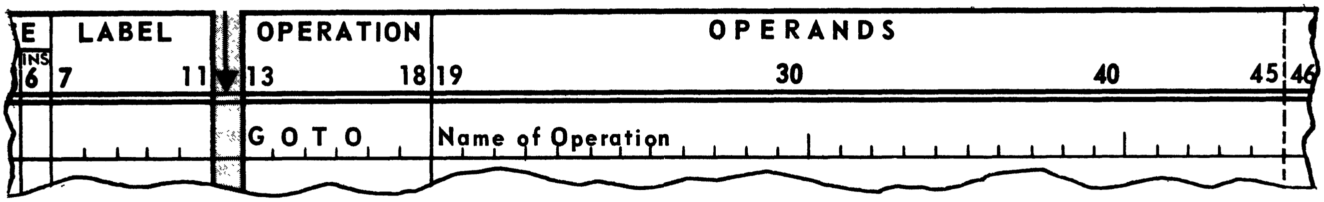
Routines are normally executed as a result of the occurrence of either a LEV break or page overflow.

Execution of a routine may also occur through use of the XCUTE command:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		XCUTE	Name of Routine to be Executed

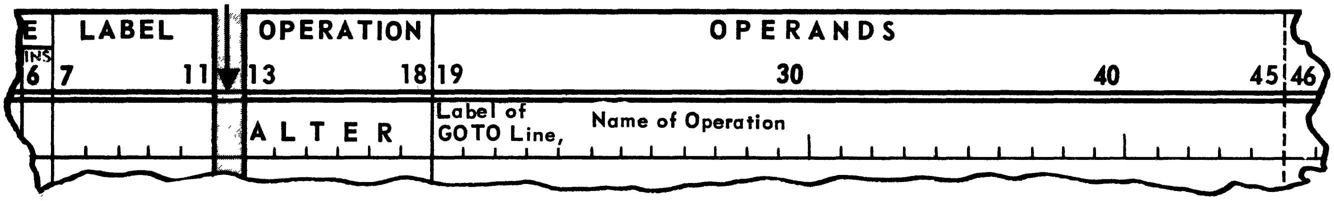
F. UNCONDITIONAL TRANSFER OF CONTROL



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			GOTO	Name of Operation	

This causes an unconditional transfer of control to the operation named in the operand field.

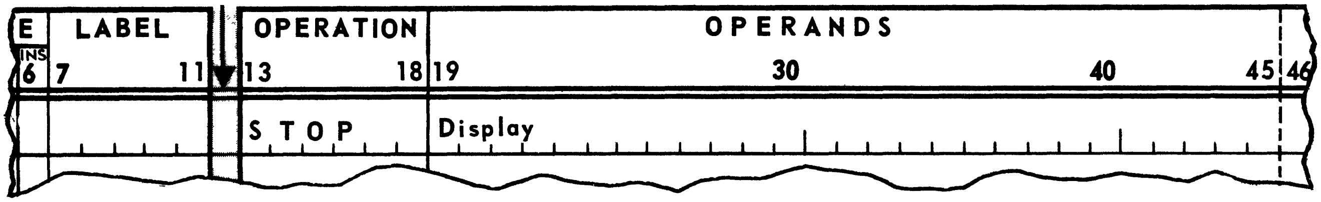
G. ALTER OPERATION NAME



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			ALTER	Label of GOTO Line, Name of Operation	

This operation permanently replaces the operand specified in the GOTO line with the name of a new operation.

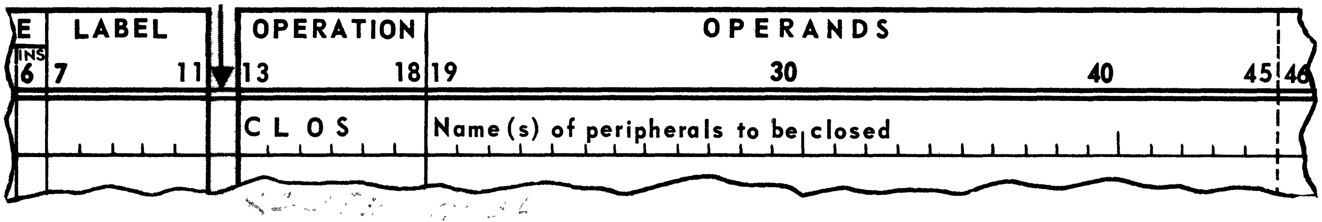
H. DISPLAY STOP



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			STOP	Display	

This causes a display stop to be generated containing the display written in the operand field. If the Reader is being used it may be closed before stopping (see CLOS below). This will not impair continued use of the Reader after the stop.

I. CLOSING RUN



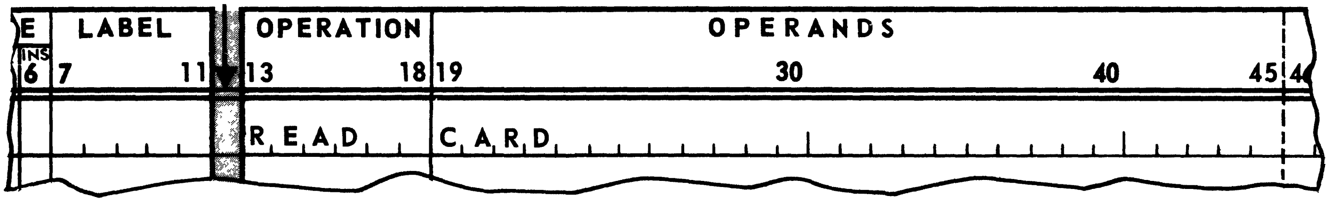
E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			CLOS	Name(s) of peripherals to be closed	

This operation causes the I/O devices named to complete current operations and stop running. It must be executed at the end of processing.

6. INPUT/OUTPUT CONTROL

A. READING

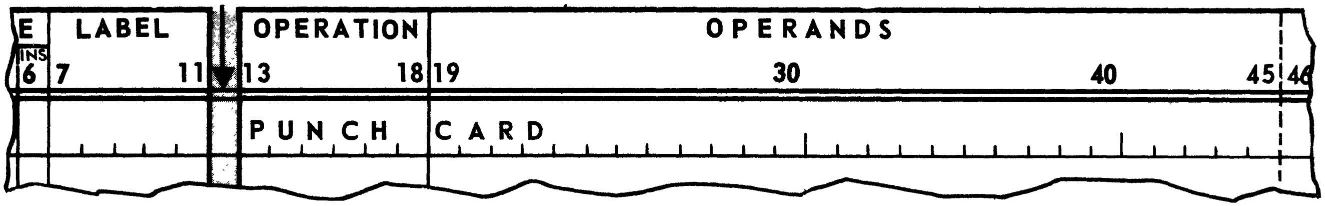
To read an input record, the following is written:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		READ	CARD	

B. PUNCHING

To punch a summary card, the following is written:

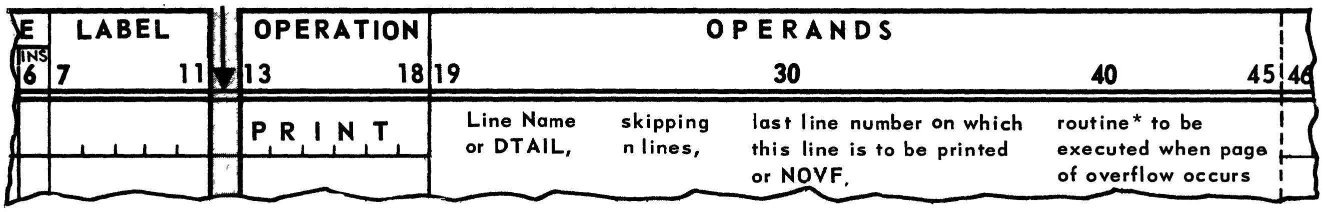


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		PUNCH	CARD	

C. PRINTING

Two forms of print command are available. Form 1 is used when it is desired to skip a certain number of lines before printing; form 2 is used when it is desired to print a line on a particular position of the output page.

Form 1



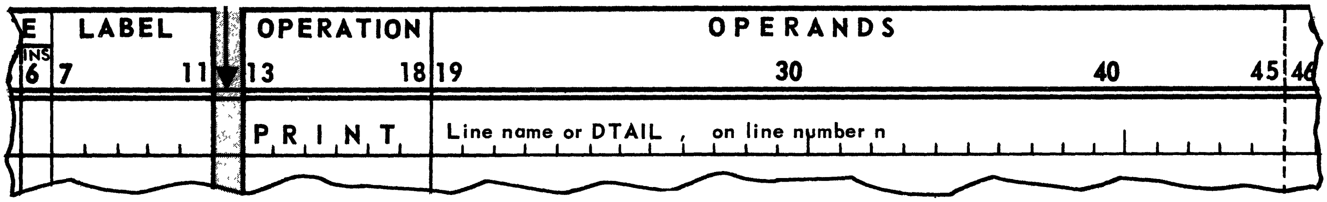
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		PRINT	Line Name or DTAIL, skipping n lines, last line number on which this line is to be printed or NOVF, routine* to be executed when page of overflow occurs	

* *The following requirements apply to the page overflow routine:*

- *must be properly defined (see Routines, 5–D);*
- *must include at least one, but can include any number of form 2 print commands;*
- *must not contain a form 1 print.*

The above command will cause the report generator to produce coding to print the line named, or a detail line, after skipping n lines. If an attempt to print this line beyond the last line parameter is made, page overflow will occur and the routine named is executed. If no page overflow is desired, the 3rd operand is written as NOVF and the 4th operand is omitted.

Form 2



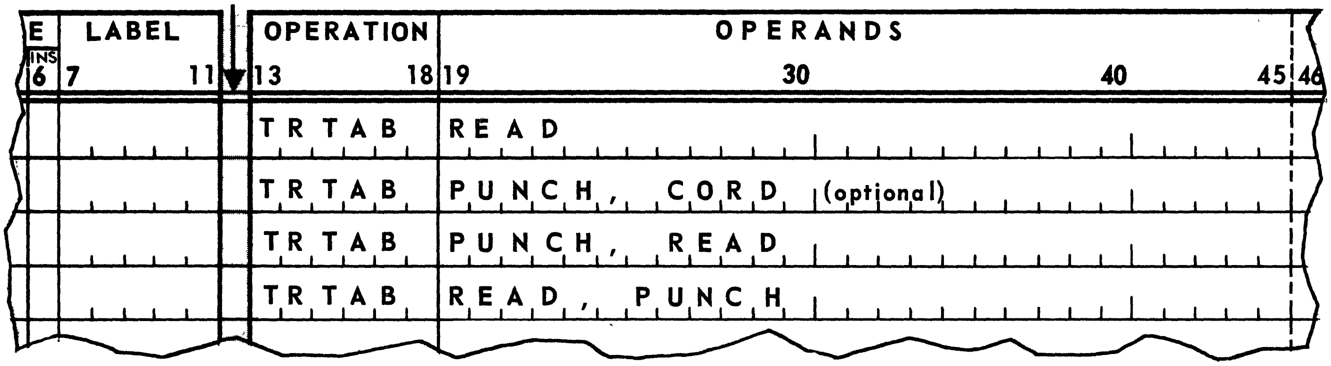
E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			PRINT	Line name or DTAIL , on line number n	

This command will cause the printing of the line named or of a detail line on the line number designated.

D. 90-COLUMN INPUT/OUTPUT TRANSLATION

1. Translate Table

The TRTAB directive is used to generate a 90-column input and/or output translate table(s). The resulting tables are employed by the REGENT translate command (see TRNSL). The operand(s) appearing on the TRTAB line may be in any order as follows:



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			TRTAB	READ	
			TRTAB	PUNCH, CORD (optional)	
			TRTAB	PUNCH, READ	
			TRTAB	READ, PUNCH	

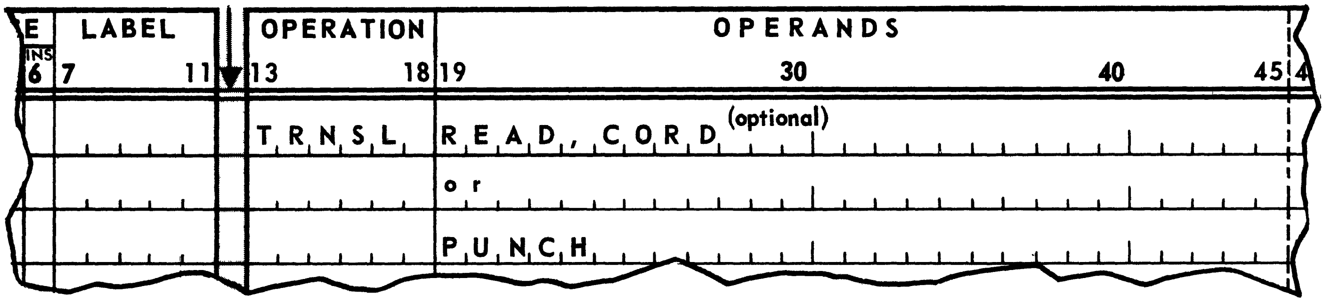
a. If the object program is to run under control of the Executive Routine, CORD must appear as one of the operands. In this case, the only other operand is PUNCH, thereby resulting in the generation of a 90-column output translate table; READ should not appear in the operands field, since the translate coding generated by REGENT (TRNSL READ, CORD) assumes that an input translate table has been provided by the Executive Routine in memory locations 01500–01577 (octal).

b. In the event the object program is to run without the Executive Routine, the operands field may contain READ and/or PUNCH only. The presence of the operand(s) READ and/or PUNCH results in the generation of a 90 column input and/or output translate table.

NOTE: The UNIVAC 1050 system requires that the table used at the time of translation lie within the first memory module (locations 0–4096).

The TRTAB READ and/or PUNCH command must be positioned in the source program such that the resulting table(s) satisfy the above requirement. It is recommended that the TRTAB directive immediately follow the BEGIN statement and precede the USE statement.

2. Translation Command



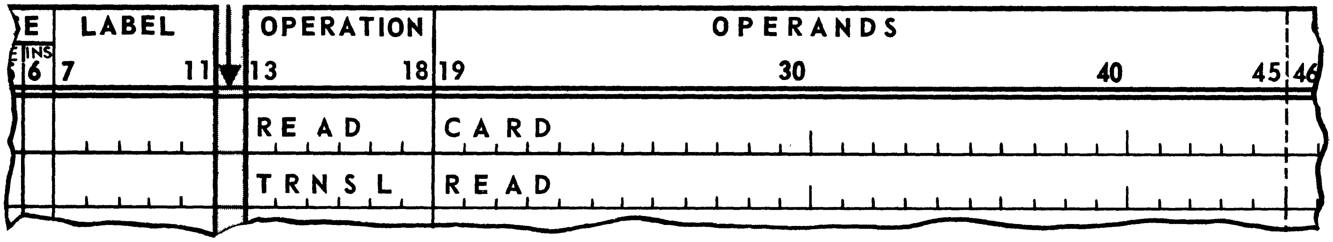
```
E  LABEL      OPERATION  OPERANDS
6  7      11  13      18 19          30           40         45 4
              TRNSL      READ,CORD (optional)
                         or
                         PUNCH
```

The TRNSL command is used to translate 90-column card code to internal code or vice versa.

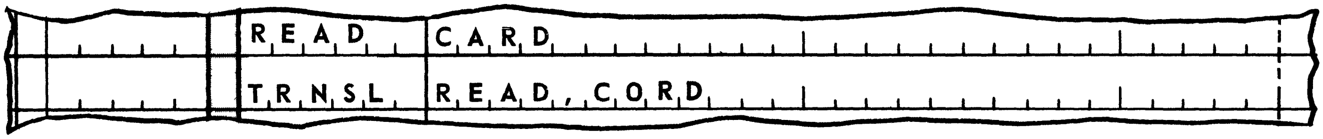
The first operand is READ or PUNCH specifying input or output translation, respectively. A second operand, CORD, is present if and only if the object program is to run under control of the Executive Routine (either OPS *or* OPR).

Input translation will normally immediately follow a read operation and output translation immediately precede a punch command.

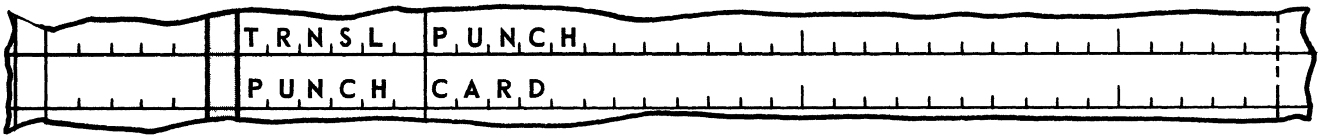
Examples:



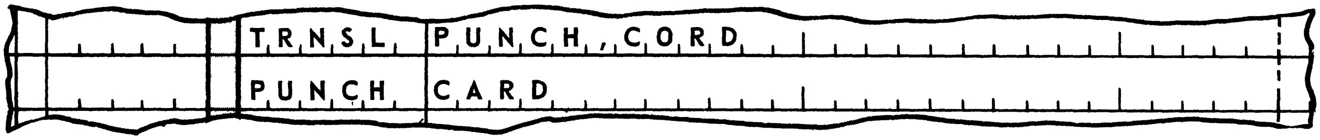
```
E  LABEL      OPERATION  OPERANDS
6  7      11  13      18 19          30           40         45 46
              READ       CARD
              TRNSL      READ
```



```
              READ       CARD
              TRNSL      READ,CORD
```



```
              TRNSL      PUNCH
              PUNCH      CARD
```



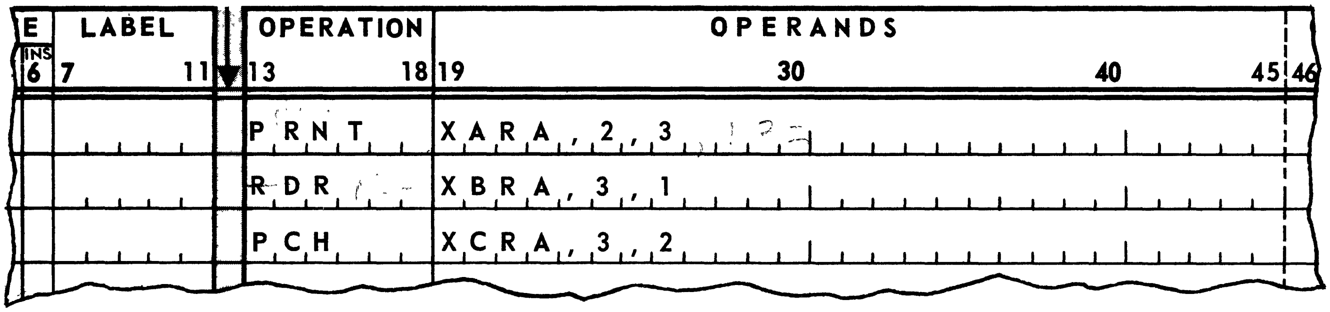
```
              TRNSL      PUNCH,CORD
              PUNCH      CARD
```

7. SELECTION OF INPUT/OUTPUT CONTROL ROUTINES

A. MAGNETIC TAPE SYSTEM

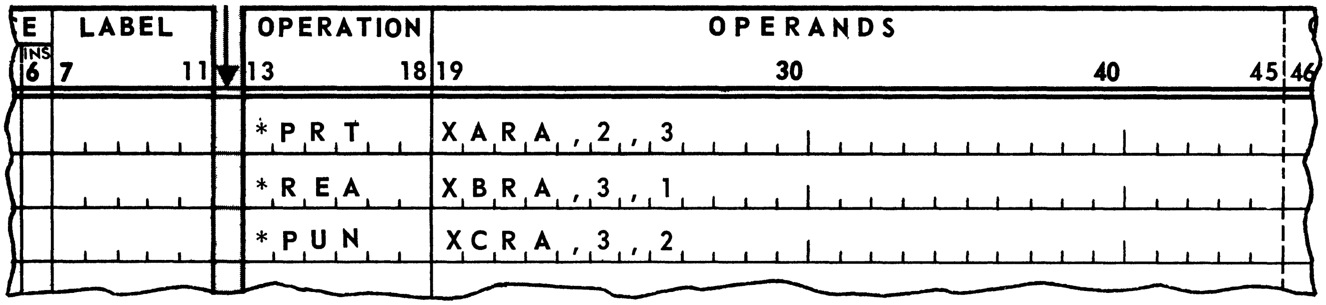
If the PAL Tape Assembler is being used to generate the REGENT object program, all of the I/O control routines being used by the installation are available to the REGENT user. A call line must be written for each device being used in the program. The operation field of the call line will contain the name of the control routine desired. The operands are standard (for REGENT) and will always be written as shown below:

1. Call Lines (with Executive)



E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			PRNT	XARA,2,3	
			RDR	XBRA,3,1	
			PCH	XCRA,3,2	

2. Call Lines (without Executive)

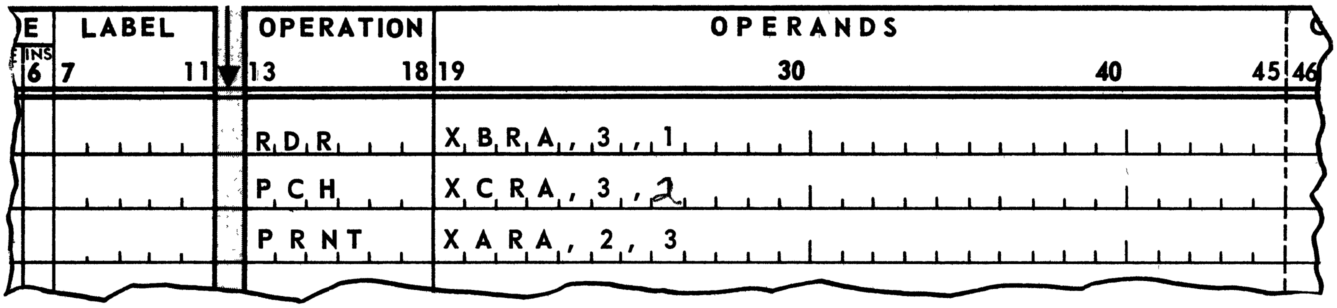


E INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			*PRT	XARA,2,3	
			*REA	XBRA,3,1	
			*PUN	XCRA,3,2	

B. CARD SYSTEM

If the PAL Card Assembler is used, the call lines for I/O control routines are omitted. The required I/O routines supplied by Univac must be inserted in the Card REGENT output deck prior to final assembly. They should be inserted immediately before the last statement (END). These routines contain certain parameters that are not explicitly defined.

The definitions must be supplied as input to Card REGENT and be consistent with the input/output definitions appearing in the REGENT program. This can be accomplished through the use of the special directives that appear as follows:


E
LABEL
OPERATION
OPERANDS
INS
6
7
11
13
18
19
30
40
45
46

```
            RDR   XBRA,3,1
            PCH   XCRA,3,2
            PRNT  XARA,2,3
```

The appropriate directives must be inserted in the REGENT source deck following the I/O definitions.

The directive(s) cause Card REGENT to generate EQU statements that properly specify the values of undefined parameters in the corresponding I/O control routine(s).

8. PROGRAM CONSIDERATIONS

A. REQUIRED OPERATION STATEMENTS

The following statements must appear in all REGENT programs in the location and manner prescribed.

1. BEGIN

E	LABEL		OPERATION	OPERANDS	
INS 6	7 11	▼	13 18	19 30 40 45	46
			BEGIN		

This must be the first statement. The operands field in the BEGIN statement for REGENT is the same as for the PAL assembly system.

2. USE

E	LABEL		OPERATION	OPERANDS	
INS 6	7 11	▼	13 18	19 30 40 45	46
			USE	Names of I/O peripherals to be used, CORD or OPR (optional)	

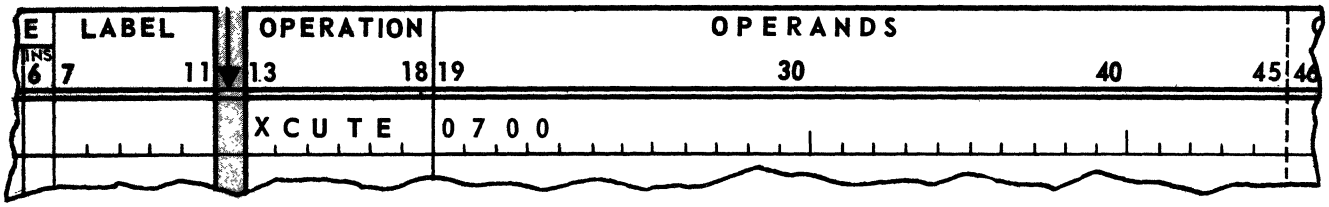
The BEGIN statement must be followed by the USE statement. The operands will be READ, and/or PRINT, and/or PUNCH, in any order. This operation enables REGENT to generate the instructions necessary to initialize the peripheral devices to be used. If the program is to run under control of the Executive Routine, either CORD (for OPS) or OPR (for OPR) must appear as an operand.

3. PAGE

E	LABEL		OPERATION	OPERANDS	
INS 6	7 11	▼	13 18	19 30 40 45	46
			PAGE	Number of lines on page of printed output	

If one of the operands of the USE statement is PRINT, the PAGE statement must follow the USE statement. This enables REGENT to control the page overflow, and line counting operations as required by the PRINT statements.

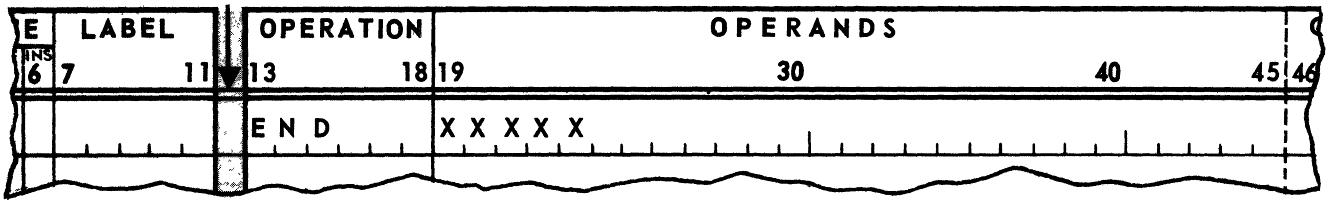
4. XCUTE



E	LABEL		OPERATION	OPERANDS	
INS 6	7 11		13 18	19 30 40 45	46
			XCUTE	0700	

If the I/O Executive Routine is being used, this operation releases the programs storage allocation, and relinquishes control to the Executive. It should be written at the conclusion of processing (after the CLOS).

5. END



E	LABEL		OPERATION	OPERANDS	
INS 6	7 11		13 18	19 30 40 45	46
			END	XXXXX	

The last line (card) of the source program should contain the unique label "XXXXX" as shown.

B. PROGRAM ORGANIZATION

Before writing a REGENT program, it is advisable to prepare a complete description of the problem with particular attention to input and output layout. With this done, it is a simple task to assign names to the various fields and lines and to write definitions of the input and output areas using the INPUT and OUTPT directives.

Having prepared all of the field descriptions required, a list of constants, edit masks, and accumulators should be prepared. The input/output layouts should be consulted to be certain each accumulator has been defined with a sufficient length to handle the maximum possible total size.

The sequence of operations in the object program is determined by the sequence in which they are written and may be altered as directed by program control directives.

Below are some conventions which should be followed to assure correct and efficient object coding.

1. A page overflow routine should be executed at the very beginning to assure proper initial positioning of paper before processing begins. The first operation to be performed by the program *must* contain the label START.

2. Normally, the reading of an input item will be immediately followed by a test for the end of the run. This will probably consist of an IF operation, comparing a previously defined sentinel constant with a field from the input.

3. The LEV directives should occur before any further processing is specified, since a control break indicates that the last card of a control group has already been processed. After the LEV operations have been written in their proper sequence, they should be followed by the processing which is done, if no control break has occurred. This will normally be the computation, movement, and printing of the detail line.

4. Sequence control directives should be preceded by operations which are to be performed regardless of the result of the transfer. This will conserve storage and result in a more efficient program.

5. Closed subroutines which are written using the RTN directive will follow the detail processing.

6. End of job processing, to which control is transferred as a result of the test mentioned in note 2 above, should include execution of the highest control total level (assuring execution of all lower ones) and the page overflow routine.

9. REGENT OPERATING INSTRUCTIONS

A. CARD REGENT

Card REGENT is available in two separate versions for use with either the 80-column (REG8) or 90-column (REG9) card systems.

1. Hardware Requirements

   a. Storage Size – 8K minimum

   b. I/O Configuration

   Input device, for source program – Column or Row Card Reader.

   Output device, for coding produced by REGENT – Column or Row Card Punch.

2. Operating Procedure

   a. Reader

   (1) Depress the POWER-ON button until the light is on.
   (2) Load cards into the input magazine (Card REGENT object program followed by the source program to be processed).
   (3) Depress the CLEAR button on the reader.

   b. Punch

   (1) Depress the POWER-ON button until the light is on.
   (2) Depress the OFFLINE button until the light is on.
   (3) Depress the CLEAR button, then the MANUAL FEED button until card(s) appears in the output stacker. Then remove the card(s) from the output stacker.
   (4) Depress the OFFLINE button until the light is off.

   c. Console

   (1) Depress the CLEAR button.
   (2) Depress the PROGRAM START button to initiate processing.
   (3) Stop display (077777) indicates the end of REGENT processing. Remove the output* produced by REGENT from the punch. Select the required I/O control routines from those supplied with Card REGENT. Insert the routine(s) immediately before the last card (END card) of the output deck. The resulting deck is then ready to be used as direct input to the PAL Card Assembler.
   (4) The operating instructions for the PAL Card Assembler should be followed. After both passes of the assembler have been completed, the I/O control routine cards should be removed and stored for future use.

* The output cards are sequenced by REGENT beginning with the number 05000. REGENT input statements appear in the output as comment cards containing a period in column 7, blanks in the sequence number field, and the original sequence number in the ID field.

B. PRE-PASS REGENT (TAPE REGENT PROCESSOR)

Pre-pass REGENT is available in two separate versions, for use with either the UNISERVO IIIA or IIIC tape systems. Each of these versions appears on a different master instruction tape (IIIA or IIIC MIT); however, both programs possess the same program ID (REGO).

1. General

The following notes apply to both the IIIA and IIIC versions of Pre-pass REGENT.

a. Hardware Requirements

(1) Storage Size – 8K minimum
(2) I/O Configuration

Minimum

- Three UNISERVO tape units

Allowable input devices for source program

- UNISERVO tape unit
- 80-column Column Reader
- 80-column Row Reader
- 90-column Column Reader
- 90-column Row Reader

Output device for coding produced by REGENT

- UNISERVO tape unit

b. Input Specifications

(1) Source Program on Cards

The first card of the program must consist of a valid BEGIN statement having a four character program ID in columns 7-10.

The last card must be an END statement.

(2) Source Program on Tape

The input tape may consist of one or more source programs. The header block for each program should contain a unique four character program ID.

AJAX (tape maintenance system for the UNIVAC 1050 system) should be used to create the tape, to ensure that the input tape will possess the proper format required by the REGENT processor.

c. Output Specifications

(1) The tape output produced by REGENT has the following form:

block 1	A program header (D block) containing the program ID. This ID is identical to the ID specified in the input program header block, or the label field of BEGIN card.
blocks 2 through n – 1	Source code (E blocks).
block n	Program sentinel (F block).
block n + 1	Tape sentinel (Z block).

(2) The output tape format is totally consistent with the approved conventions for UNIVAC 1050 source code library tapes.

2. Introduction to Operating Procedure

a. Run Description

(1) Order of Events

1 – The Executive Routine is loaded.
2 – The Executive Routine transfers the REGENT loader to storage.
3 – Pre-pass REGENT is loaded.
4 – The source program is processed by REGENT.
5 – REGENT transfers the PAL loader to storage.
6 – The PAL assembler is loaded.
7 – REGENT output is assembled.

(2) Program Control

For the duration of each phase mentioned above, only one program is in control.

During phase *1* – The Executive Routine (loader) is in control.
During phase *2* – The Executive Routine is in control.
During phase *3* – Pre-pass REGENT (loader) is in control.
During phase *4* – Pre-pass REGENT is in control.
During phase *5* – Pre-pass REGENT is in control.
During phase *6* – PAL (loader) is in control.
During phase *7* – PAL is in control.

(3) Operator Control

The program may require operator action at various points in the run. This is accomplished by program execution of a jump display instruction (stop) that causes the computer to stop, thus allowing operator intervention.

The display appearing in the first 16 bits of the M portion of the console register indicates an existing condition that may require action on the part of the operator. A list of stop displays appears in Section B-3 (Operating Procedure). The operator should become thoroughly acquainted with this list in order to expedite recovery when a condition arises. (See Figure 9-1.) The following is provided as a guide:

I/O Displays (Bit positions 16 = 1)

		CHANNEL					UNIT				ERROR MESSAGE					
BIT POSITIONS	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
DISPLAY	1	c	c	c	0	0	u	u	u	u	z	z	z	z	z	z

Non-I/O Displays (Bit position 16 = 0, in most cases)

		PASS			ERROR MESSAGE											
BIT POSITIONS	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
DISPLAY	0	x	x	x	y	y	y	z	z	z	z	z	z	z	z	z

In general,

xxx = 001 (1) and yyy = 110 (6) imply a REGENT stop;

xxx = 001 (1) and yyy ≠ 110 (6) imply a PAL stop;

xxx = 111 (7) implies an Executive Routine stop.

b. Communication

(1) Operator to Program

(a) Operator to Executive (see Operating Procedure, Section 9-B-3; refer to one of the Executive Routine (OPS or OPR) documents if a more detailed description is desired.

(b) Operator to REGENT

In the case of tape input, the operator must provide REGENT with a four character program ID (PID). The PID controls the process of locating the source program to be processed. Following are acceptable forms of the program ID:

PID = $ZZZ (with no variations). The source program to be processed is the first run on tape.

PID = rrrr, where rrrr is any ID other than $ZZZ or blank.

The run labeled rrrr, where rrrr equals the ID specified in the source code header (D block) is located and processed.

The operator may supply the PID by utilizing one of the three locate options described in steps (9) and (10) of the Console Operating Instructions (see Section B-3-c).

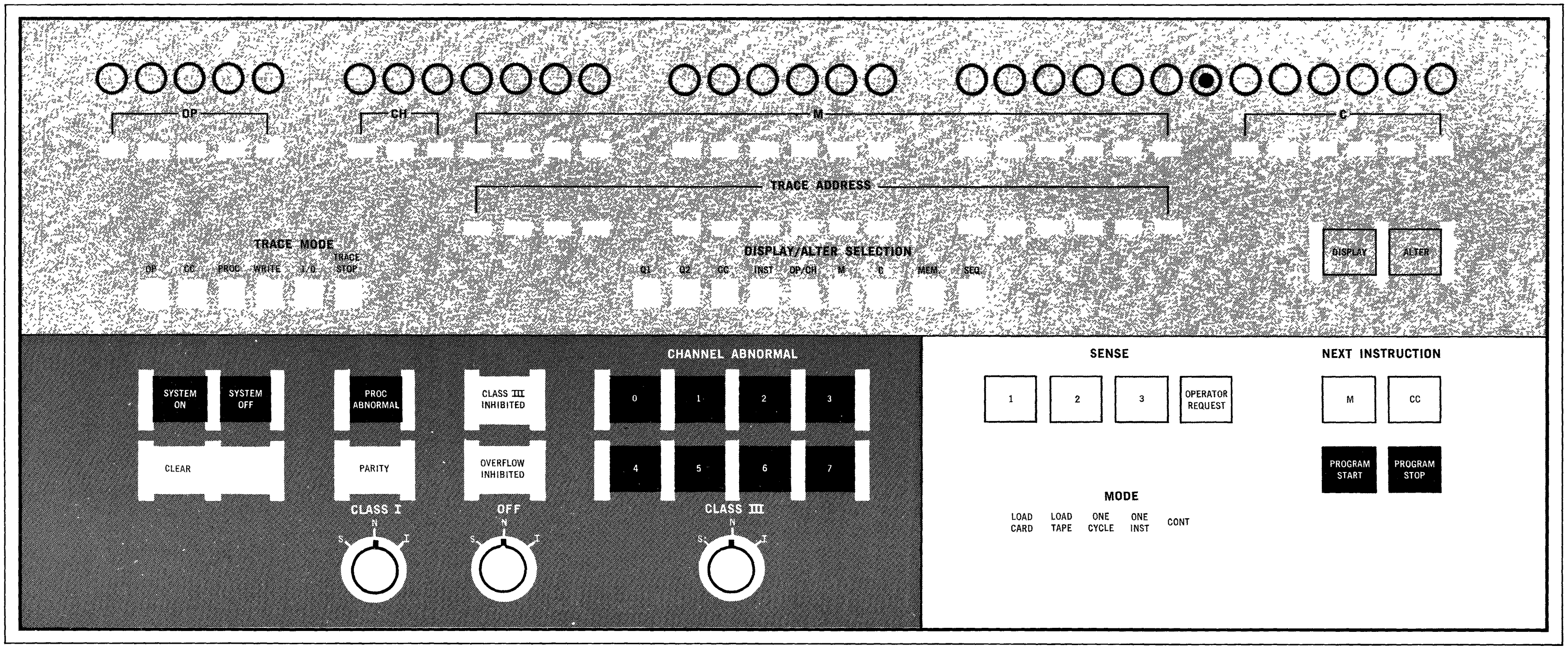


Figure 9-1. Central Processor Console

(2) Program to Program

(a) Executive Routine to REGENT

The Executive communicates information to REGENT through storage locations 0, 1, 2, 3. The information transferred consists of the four characters that appeared in columns 7-10 of the REGENT call card (read by the Executive). The contents of storage locations 0-3 specify to REGENT the location of the source program as follows:

If the contents of storage locations 0-3 are blank, Pre-pass REGENT will check Sense Switch 1 with the following result:

ON – PID accepted via a trace switch key in.
OFF – Call card (PID in columns 1-5) or source program cards are read.

If the contents of storage locations 0-3 are not blank, Pre-pass REGENT will use the four characters of information for the PID and attempt to locate the specified program on tape.

NOTE: The four character ID supplied by the Executive is preserved by REGENT on the first and successive recovery attempts.

(b) REGENT to PAL

The above mentioned storage locations (0-3) are also used by REGENT to inform PAL that the program (REGENT output) to be assembled is the first on tape. REGENT accomplishes this by setting (0-3) = $ZZZ immediately prior to loading PAL.

3. Operating Procedure

a. General

At REGENT processing time,

Sense Switch 1 controls the location of source code, unless the locate option (described in step (9)-(a) of the Console Operating Instructions) is employed – in which case Sense Switch 1 control is nonexistent.

OFF – Pre-pass REGENT expects a source program or call card in reader.

ON – REGENT will stop to allow a trace switch key in of the four character source program ID (PID) that specifies the program to be processed from tape.

At assembly time,

Sense Switch 1 controls the codedit on tape option.

ON – Write codedit on tape (tape unit 2).

OFF – Do not write codedit on tape.

Sense Switch 2 controls the printing of the assembly listing.

OFF – Entire listing will be printed.

ON – Procedure generated lines (including those resulting from REGENT processing) will not be printed.

b. Peripheral Operating Instructions

(1) UNISERVO Units

(a) Mount the master instruction tape (MIT) on tape unit 0 with the write enable ring in.
(b) If a source program is to be processed from tape, mount the source code library on tape unit 1 with the write enable ring removed.

If a source program is on cards, mount a blank tape on tape unit 1 (with the write enable ring in) for use at assembly time.

(c) Mount blank tape on tape unit 2 with the write enable ring in.
(d) Bring all tapes to the load point.

(2) Reader (if used)

(a) Depress the POWER-ON button until the light is on.
(b) Load cards into the input magazine.
(c) Depress the CLEAR button on the reader.

(3) Printer (in preparation for assembly run)

(a) Depress the OFFLINE button until the light is on.
(b) Set the paper five holes above the sprocket.
(c) Depress the OFFLINE button until the light is off.

c. Console Operating Instructions

(1) Depress the CLEAR button.
(2) Depress the LOAD TAPE Mode button.
(3) Depress the PROGRAM START button.
(4) Depress the CONTinuous Mode button.
(5) Depress the PROGRAM START button.

Upon stop 017325, select the desired Executive Routine. To bypass an Executive Routine, depress the M and PROGRAM START buttons. Every time the M and PROGRAM START buttons are depressed, an Executive Routine will be skipped. When the tape is positioned to the desired Executive, go to step (6). Normally each installation will have a master instruction tape (MIT) with the desired Executive as the first Executive on tape and no bypassing will be necessary.

(6) Depress the PROGRAM START button.
(7) Upon stop 070001, the Executive Routine is ready to load Pre-pass REGENT.
(8) If the REGENT call card ($REG1 in columns 1-5, blanks in remaining columns) and source program deck are in the reader, depress the START button to load Pre-pass REGENT and initiate processing of source program.
(9) If the source program to be processed is on tape and the card reader is available, either of the following options may be used.

(a) (Recommended option.) A single call card is placed in the reader. This card performs a dual function, i.e., it calls Pre-pass REGENT and controls the subsequent location of the source program to be processed. The format of the card is as follows:

SEQUENCE			LABEL		OPERATION	OPERANDS
PAGE 1 3	LINE 4 5	INS 6	7 11	▼	13 18	19 30 40
$RE	G1		yyyy			

where yyyy is any acceptable PID.

(b) Two cards are placed in the reader. The first calls the REGENT processor, while the second card controls the location of the source program to be processed. Their format is as follows:

SEQUENCE			LABEL		OPERATION	OPERANDS
PAGE 1 3	LINE 4 5	INS 6	7 11	▼	13 18	19 30 40
$RE	G1					(REGENT call card)
$yy	yy					(Source program call card)

where yyyy is any acceptable PID.

After executing one of the options, 1 or 2, depress the START button to initiate REGENT processing of the source program.

(10) If the card reader is not available,

(a) At display stop 070001 depress the OPERATOR REQUEST button.
(b) Depress the Trace Mode PROC button.
(c) Set the trace switches to octal 005430.
(d) Depress the PROGRAM START button.
(e) Upon stop 077000, key in the next two characters of the Pre-pass REGENT program ID. Set the trace switches to octal 003204.
(f) Set Sense Switch 1.
(g) Depress the PROGRAM START button.
(h) Upon stop 016001, the REGENT processor is ready to accept a trace switch key in of the PID.
(i) Set binary value of the first two characters of the PID in trace switches.
(j) Depress the PROGRAM START button.
(k) Upon stop 016700, set the binary value of the next two characters in the trace switches.
(l) Reset Sense Switch 1.
(m) Depress the PROGRAM START button to continue processing.

(11) Stop 016777, end of Pre-pass REGENT run.

At the conclusion of the run, REGENT locates the PAL Assembler and stops to allow operator intervention. The final positioning of tapes will be as follows:

Tape unit 0 – MIT positioned at PAL Assembler program.

Tape unit 1 – If REGENT input is on tape, the tape is rewound (with interlock).

Tape unit 2 – The REGENT output tape is rewound (without interlock).

The output tape on tape unit 2 contains the source program produced by Pre-pass REGENT. In order to assemble the REGENT output at this time, the operator need only perform the following procedure:

(a) Mount a blank tape (with write enable ring in) on tape unit 1 and position the tape at load point (unless this was done previously).
(b) Depress the PROGRAM START button to assemble. REGENT will transfer control to the PAL Assembler. The input for the assembly is from tape unit 2; the resulting object program output will be on tape unit 1.
(c) Stop 017325, end of assembly.

DISPLAY STOPS

DISPLAY	PASS	DESCRIPTION	RECOVERY
070001	Executive	Executive is ready to load Pre-pass REGENT.	– – – – –
070005	Executive	Card read is not call card.	Depress the PROGRAM START button to retry.
070707	Executive	Pre-pass REGENT program not found on tape (tape unit 0 rewinding).	Wait for completion of the rewind; then, depress the PROGRAM START button to retry.
077000	Executive	ID key in.	– – – – –
016707	REGENT	Source program to be processed not found on tape (tape unit 1 rewinding).	See Recovery Notes (2) and (3).
016001	REGENT	ID key in.	Set the first two characters of the source program ID, to be processed from tape, in trace switches. Depress the PROGRAM START button.
016700	REGENT	ID key in.	Set the second two characters of source program ID in trace switches. Depress the PROGRAM START button.
016333	REGENT	Source program being processed from tape contains a non-source code block (in general, caused by absence of an END statement).	Depress the PROGRAM START button to complete REGENT run. Input processing terminates with the last source code block (E block) read.
016771	REGENT	No BEGIN card, or improper source program call card.	See Recovery Note (1).
016772	REGENT	No BEGIN statement on tape.	Depress the PROGRAM START button to continue processing without a BEGIN statement.

DISPLAY STOPS

DISPLAY	PASS	DESCRIPTION	RECOVERY
016777	REGENT	End of REGENT run.	See Step (11), Console Operating Instructions, to assemble REGENT output tape.
011111	PAL	Codedit on tape (option).	If codedit on tape (tape unit 2) is not desired, reset Sense Switch 1. For codedit, Sense Switch 1 must remain set and the following operation performed: a. Mount blank tape with write enable ring in on tape unit 2. b. Depress the PROGRAM START button.
013333	PAL	No END statement in source program (REGENT output) on tape.	Unrecoverable. Use AJAX to correct tape, then reassemble (see PAL operating instructions).
017771	PAL	No BEGIN statement in source program (REGENT output) on tape.	Same as above.
110000	REGENT	Reader error.	Replace cards (if any) from error stacker in input magazine. Depress the CLEAR button on the reader. Depress the PROGRAM START button.
040u44	All	Tape error (Block count wrong).	Unrecoverable. Restart run.
1c0u55	All	Tape error.	Unrecoverable. Restart run.
1c0u66	All	Tape parity.	Program in control has rocked tape. Depress the PROGRAM START button to continue. Program will attempt to recover. If program does not recover after several tries, restart run.

NOTE: c = channel, u = unit

d. Recovery Notes

(1) Reload the proper BEGIN card or source program call card* and depress the PROGRAM START button.

(2) If a stop occurred after employing the method described in step (9)-(a) of the Console Operating Instructions, and the inability to locate the source program resulted from,

- ■ A wrong tape on tape unit 1. Mount the correct input tape on tape unit 1, and depress the PROGRAM START button to resume processing.
- ■ An incorrect PID appearing in columns 6-10 of call card. The condition is essentially unrecoverable. Correct the PID and return to step (1) of the Console Operating Instructions.

(3) If a method other than that described in step (9)-(a) of the Console Operating Instructions was used, mount the correct input tape on tape unit 1 and/or correct source program call. The source program call must be reissued. This can be accomplished by either reloading a source program call card or setting Sense Switch 1 to allow key in. Depress the PROGRAM START button to continue.

C. REGENT TAPE LIBRARY

REGENT source cards are used as direct input to the PAL Tape Assembler; the assembler operating instructions should be followed. No further processing of the output object programs is required.

Optional recovery method — in lieu of recorrecting and reloading a source program call card, set Sense Switch 1 and depress the PROGRAM START button to allow a trace switch key in.

10. UNIVAC 1050/1004 REGENT

A. GENERAL DESCRIPTION

The UNIVAC 1050/1004 REGENT directive statements are the same as those of the UNIVAC 1050 REGENT with the exception of the CLEAR directive, which may not be used, the I/O selection directives, and the I/O field description directives.

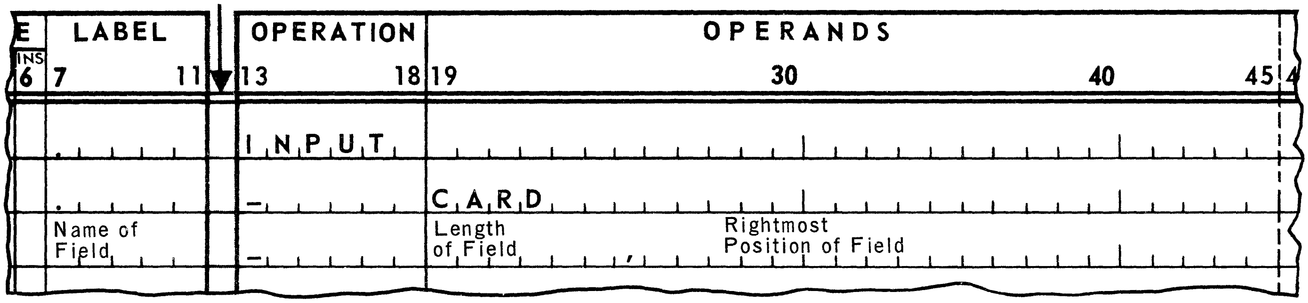
The use of the UNIVAC 1004 as the input/output device for REGENT programs precludes the use of the 1050 standard peripheral devices (card reader, card punch, and printer), and their associated input/output control routines. Whereas in the UNIVAC 1050 REGENT, the I/O selection directive and I/O field description directives are distinct and not positionally related; in the UNIVAC 1050/1004 REGENT, the two directives are combined. The field description directives for card input, card output, and output detail must immediately follow the I/O selection directive for the related I/O device. The output nondetail directive is the exception and may be used as with UNIVAC 1050 REGENT.

B. CARD SYSTEM

If the PAL Card Assembler is used, the call lines for the I/O control routines are omitted. The I/O control routines are supplied by UNIVAC in card deck form. The I/O control routine package must be inserted in the card REGENT output deck prior to final assembly.

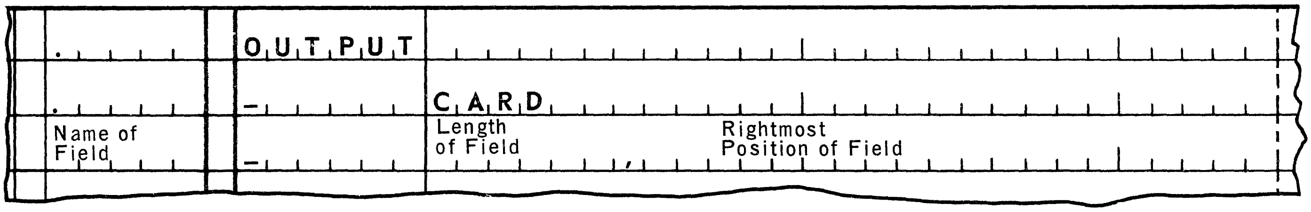
The following I/O field descriptions must appear directly behind the related I/O control routine card deck:

Card reader control routine input area,



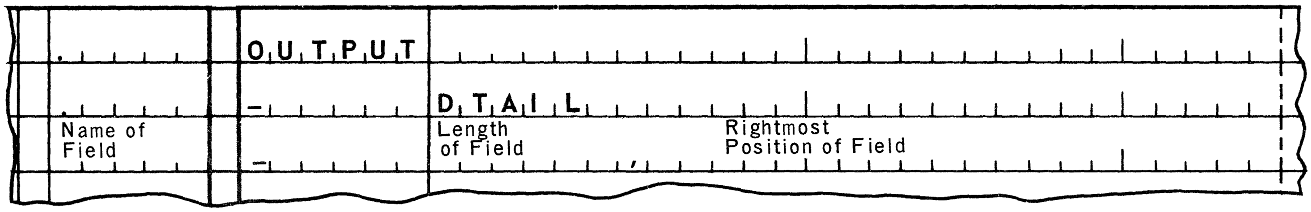
E INS 6	LABEL 7 11	↓	OPERATION 13 18	OPERANDS 19 30 40 45
	.		INPUT	
	.		–	CARD
	Name of Field		–	Length of Field , Rightmost Position of Field

Card punch control routine output area,



	.		OUTPUT	
	.		–	CARD
	Name of Field		–	Length of Field , Rightmost Position of Field

Printer control routine output area,



	.		OUTPUT	
	.		–	DTAIL
	Name of Field		–	Length of Field , Rightmost Position of Field

Note: Column seven must contain a period as illustrated.

The I/O Field descriptions are inserted into the UNIVAC provided 1050/1004 I/O package as follows:

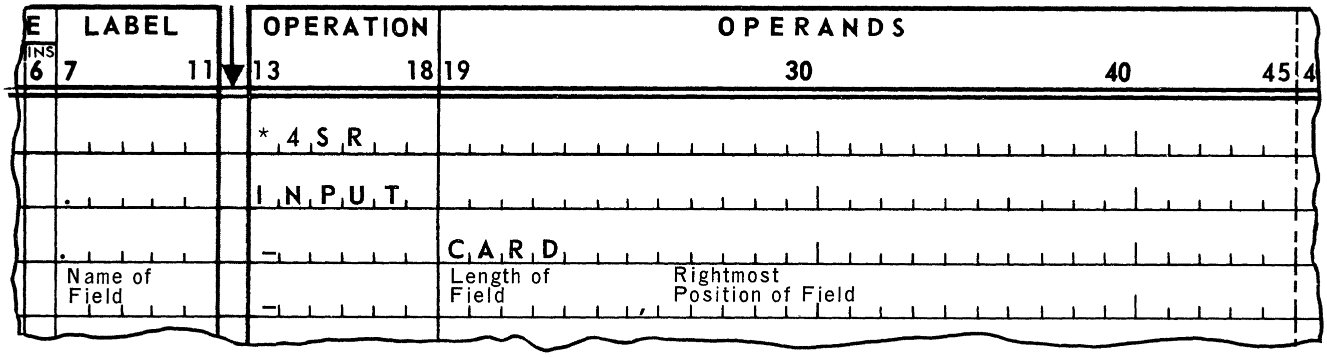
Following card 02610 the card input field.
Following card 02790 the card output field.
Following card 02980 the detail output field.

The END card comes last.

C. TAPE SYSTEM

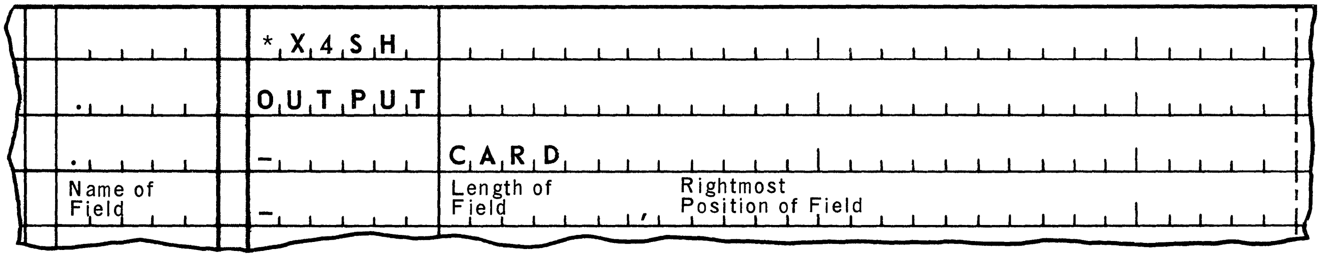
If the PAL Tape Assembler is used, the call lines for UNIVAC 1050 REGENT I/O control routines are omitted. The tape REGENT ROUTINE calls and I/O control routine field descriptions, that must appear immediately behind them, are as follows:

80 column read routine,



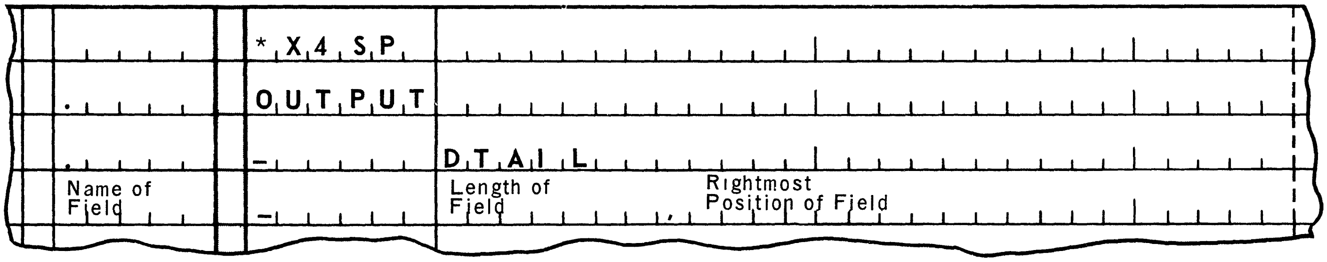
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
		*4SR	
	.	INPUT	
	.	–	CARD
	Name of Field	–	Length of Field , Rightmost Position of Field

80 column punch routine



		*X4SH	
	.	OUTPUT	
	.	–	CARD
	Name of Field	–	Length of Field , Rightmost Position of Field

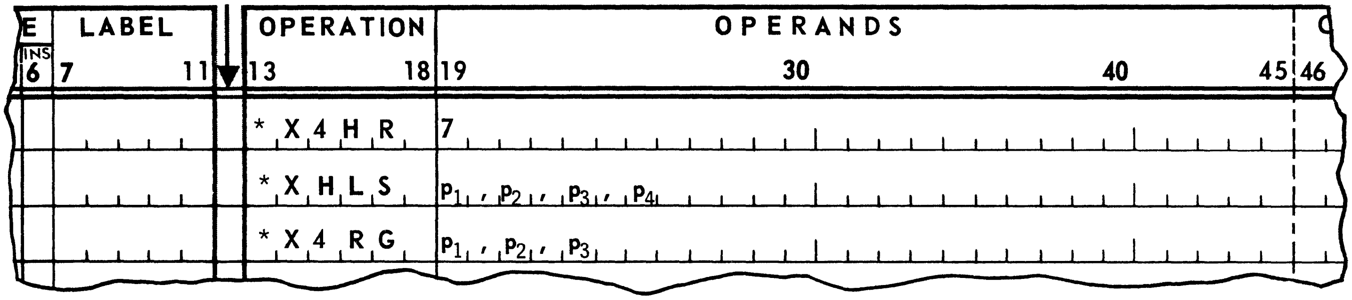
printer routine,



		*X4 SP	
	.	OUTPUT	
	.	–	DTAIL
	Name of Field	–	Length of Field , Rightmost Position of Field

The calls for the 90 column card reader and punch routines would appear as *4SR(9) and *4SH(9) respectively. Column seven must contain a period as illustrated.

The following calls must appear after the I/O control routine calls and I/O field descriptions:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		*X4HR	7	
		*XHLS	p_1, p_2, p_3, p_4	
		*X4RG	p_1, p_2, p_3	

For operating under the OPR Executive Routine the second call is XHLS; if the system is a 90 column system, the second call is followed by a nine in parentheses, i.e., *XHLS(9) or XHLS(9).

p_1	= T	if the reader is to be used.
p_1	= N	if no reading is required.
p_2	= T	if the punch is to be used.
p_2	= N	if no punching is required.
p_3	= P	if the printer is to be used.
p_3	= N	if no printing is required.
p_4	= 7	

APPENDIX A. REGENT DIRECTIVE REFERENCE TABLE

TYPE	OPERATION FIELD DIRECTIVE	DESCRIPTION	SEE PAGE
ARITHMETIC	ADD	Add field to an accumulator.	3–1
	SUB	Subtract field from an accumulator.	3–2
	MPY	Multiply two fields.	3–2
	DIV	Divide one field by another.	3–3
	ROLL	Add to accumulator(s) and reset source.	3–4
	RESET	Reset accumulator(s) to zero.	3–5
	ROUND	Half adjust a field (round).	3–5
	SHIFT	Position adjust field (shift).	3–6
DATA MOVEMENT	MOVE	Transfer contents of field to another field, editing optional.	4–1
	CLEAR	Clear an area to blanks.	4–2
	SEND	Block transfer up to 1024 characters.	4–3
SEQUENCE CONTROL	LEVn	Test for control break, if present transfer control.	5–1
	IFDEC	Decimal algebraic compare, transfer control if condition met.	5–2
	IFALP	Alphabetic (binary) compare, transfer control if condition met.	5–3
	IFCHR	Compare character, transfer control if condition met.	5–4
	IFDIG	Compare digit, transfer control if condition met.	5–4
	IFZON	Compare for designated zone punch, transfer control if condition met.	5–5
	IFNEG	Compare for negative value, transfer control if negative.	5–5
	RTN	Lines between RTN and EXIT are a closed subroutine.	5–6
	XCUTE	Execute routine named.	5–6
	GOTO	Unconditionally transfer control to operation named.	5–7
	ALTER	Replace name of operation in operand of named GOTO line.	5–7
	STOP	Stop and display contents of operand field.	5–7
	CLOS	Complete current operations and stop peripherals named.	5–7
REQUIRED STATEMENTS	BEGIN	First statement. Operand is same as for PAL.	8–1
	USE	Initialize the peripheral devices named.	8–1
	PAGE	Specifies number of lines on page of printout in operand.	8–1
	XCUTE	If an Executive is used XCUTE 0700 to relinquish control.	8–2
	END	Last line of source program, contains operand XXXXX.	8–2
EXPRESSION OF DATA	INPUT	Input file description, precedes detail field descriptions.	2–2
	OUTPT	Output file description, predeces detail field descriptions.	2–3
	–	Detail field description within an I/O file.	2–2
	+n	Nondetail field description, generates data.	2–4
	+n	Constant description.	2–7
	+n	Accumulator description.	2–7
	+n	Edit mask description.	2–8
	+n	Temporary storage register description.	2–9
I/O CONTROL	READ	Read an input card.	6–1
	PUNCH	Punch a summary card.	6–1
	PRINT	Two forms 1 skip n lines, 2 print a line on line n.	6–1
	TRTAB	Generate a 90 column input or output translate table.	6–2
	TRNSL	Translate 90 column code to internal code or vice versa.	6–3
I/O ROUTINE CALLS	RDR	XBRA, 3, 1 Card System card reader routine call.	7–1
	PCH	XCRA, 3, 1 Card System card punch routine call.	7–1
	PRNT	XARA, 2, 3 Card System print routine call.	7–1
	*REA	XBRA, 3, 1 Tape System card reader routine call.	7–1
	*PUN	XCRA, 3, 2 Tape System card punch routine call.	7–1
	*PRT	XARA, 2, 3 Tape System print routine call.	7–1
	RDR	XBRA, 3, 1 Tape System card routine with Executive.	7–1
	PCH	XCRA, 3, 2 Tape System punch routine with Executive.	7–1
	PRNT	XARA, 2, 3 Tape System print routine with Executive.	7–1

UNIVAC

DIVISION OF SPERRY RAND CORPORATION