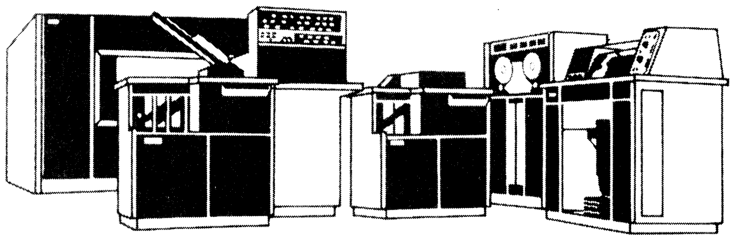


GENERAL REFERENCE MANUAL

UNIVAC 1050 SYSTEMS

REGENT

REPORT PROGRAM GENERATOR

UP-3932

This manual is published by the UNIVAC® Division 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. REGENT (REPORT PROGRAM GENERATOR)

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 ready to run object program. 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.

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

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

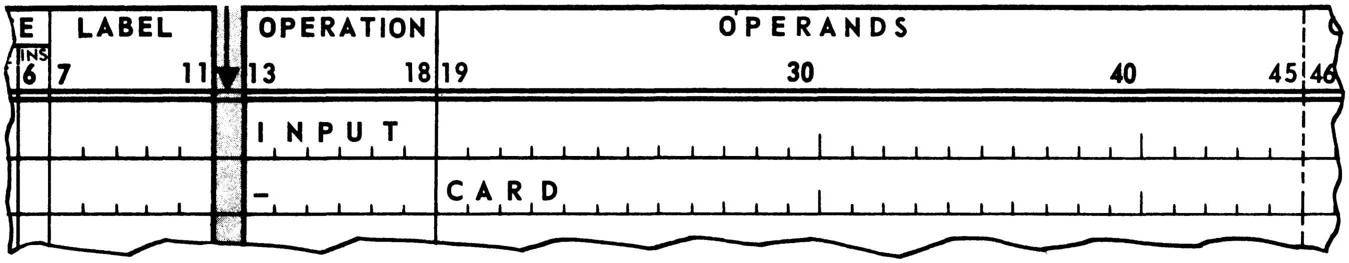
E	LABEL		OPERATION	OPERANDS	
INS 6	7 11	▼	13 18	19 30 40 45	46
	DESC		–	12, 52	

B. THE EXPRESSION OF INFORMATION

1. I/O Field Description

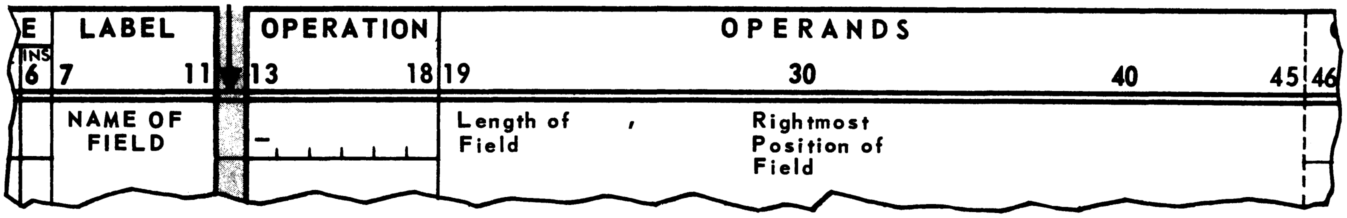
a. Input File Description (Card)

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 in the operand field. The first two cards will appear as



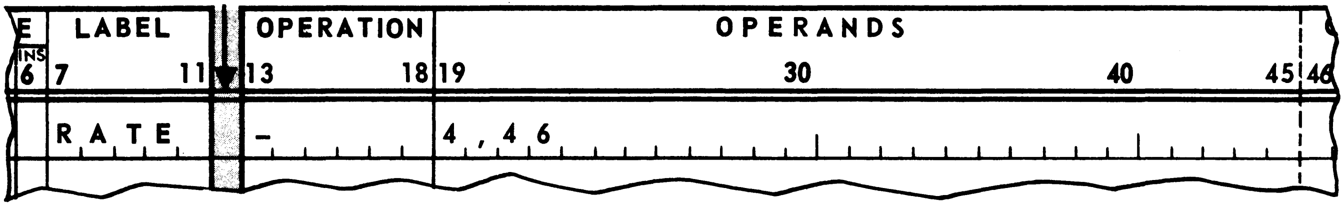
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		INPUT	
		–	CARD

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:



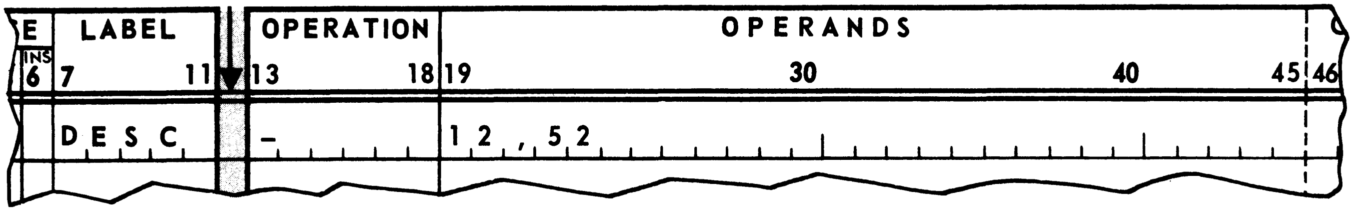
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	NAME OF FIELD	–	Length of Field , Rightmost Position of Field

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

A twelve character (Description) field in columns 41-52 is described as

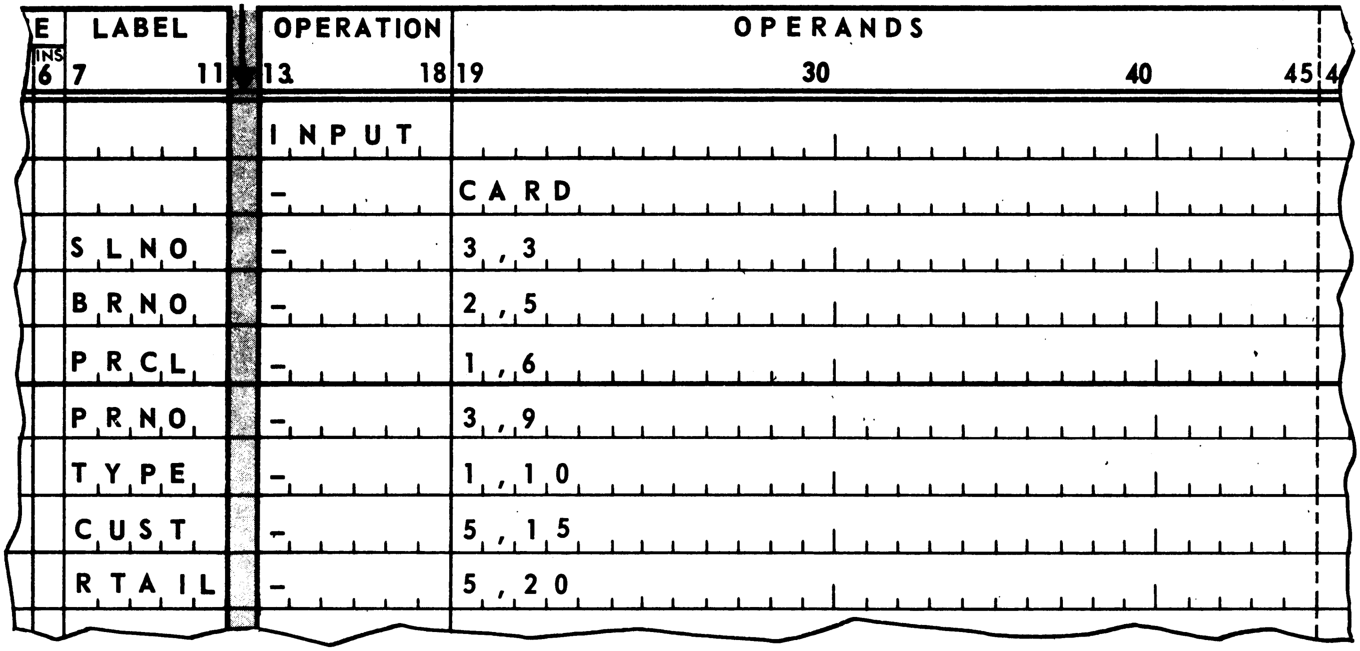


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	DESC	–	12,52

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:



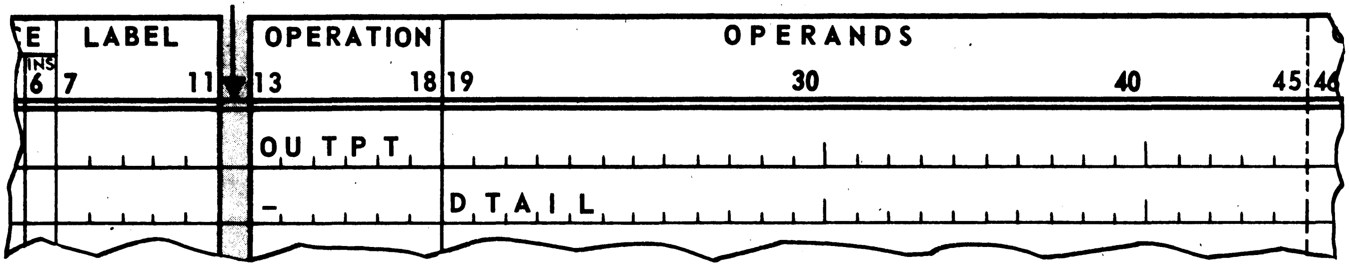
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
		INPUT	
		–	CARD
	SLNO	–	3,3
	BRNO	–	2,5
	PRCL	–	1,6
	PRNO	–	3,9
	TYPE	–	1,10
	CUST	–	5,15
	RTAIL	–	5,20

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:



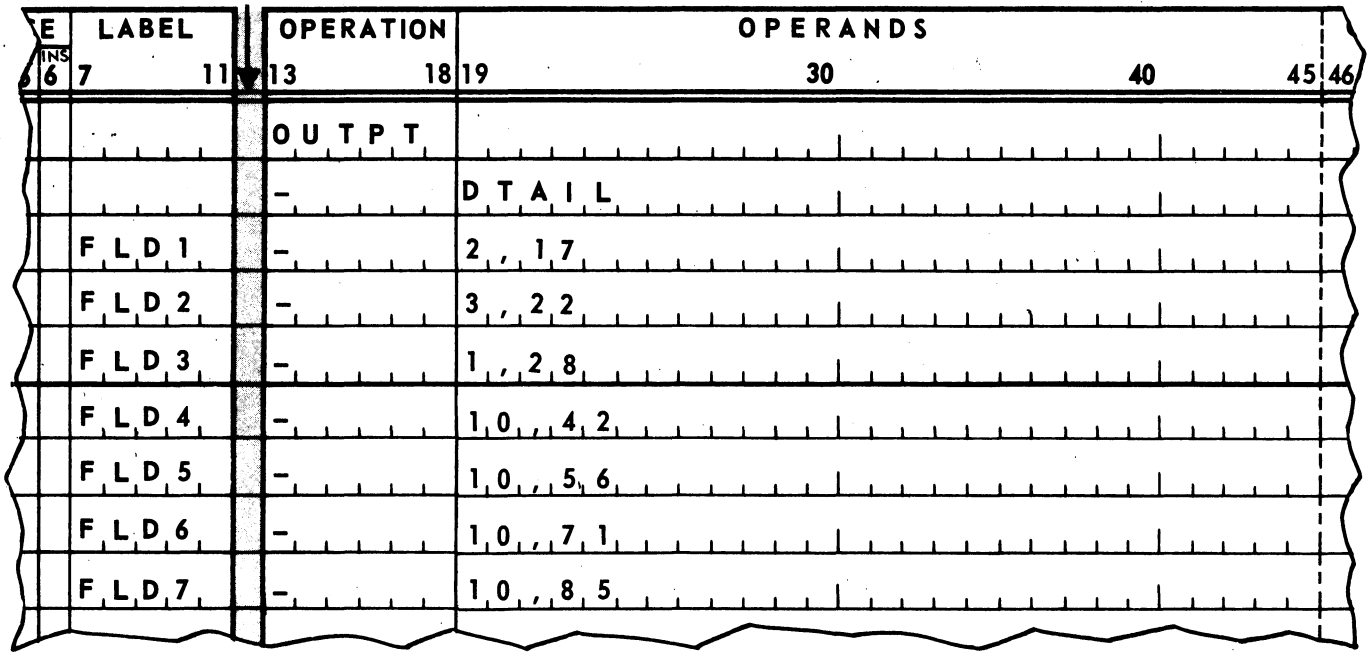
E
LABEL
OPERATION
OPERANDS
INS
6
7
11
13
18
19
30
40
45
46

```
      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



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

```
      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.

| +n | value of field

Each line description is preceded by an instruction to the generator that this is a description of a nondetail line, as follows:

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

To describe a line of the following format:

OCTOBER 1962 PAGE XX

one can write

E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
	LINE1	OUTPT	NONDT
		+47	0
		+12	'OCTOBERΔ1962'
		+17	0
		+5	'PAGEΔ'
	PGNO	+2	0
		+45	0

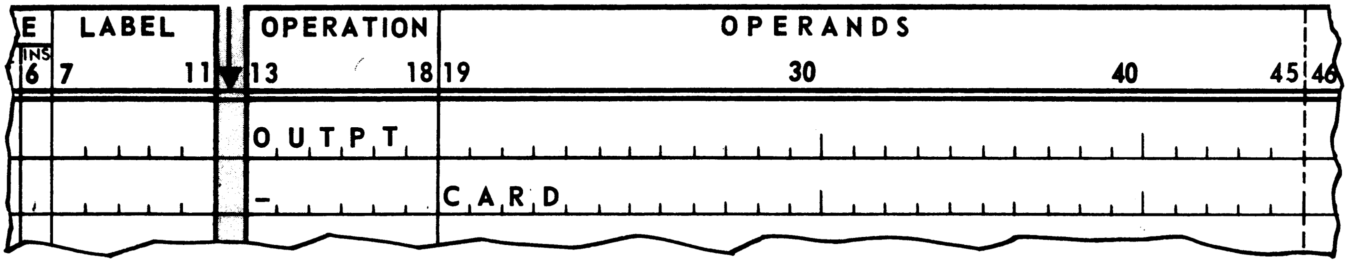
This description specifies that print positions 1 through 47 will be blank; that the next 12 print positions will contain OCTOBER 1962;* that the next 17 positions will be blank; the next 5 positions will contain PAGE and a space; the next two positions are reserved for page number (to be provided in 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. 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.

* *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 and have the following format:

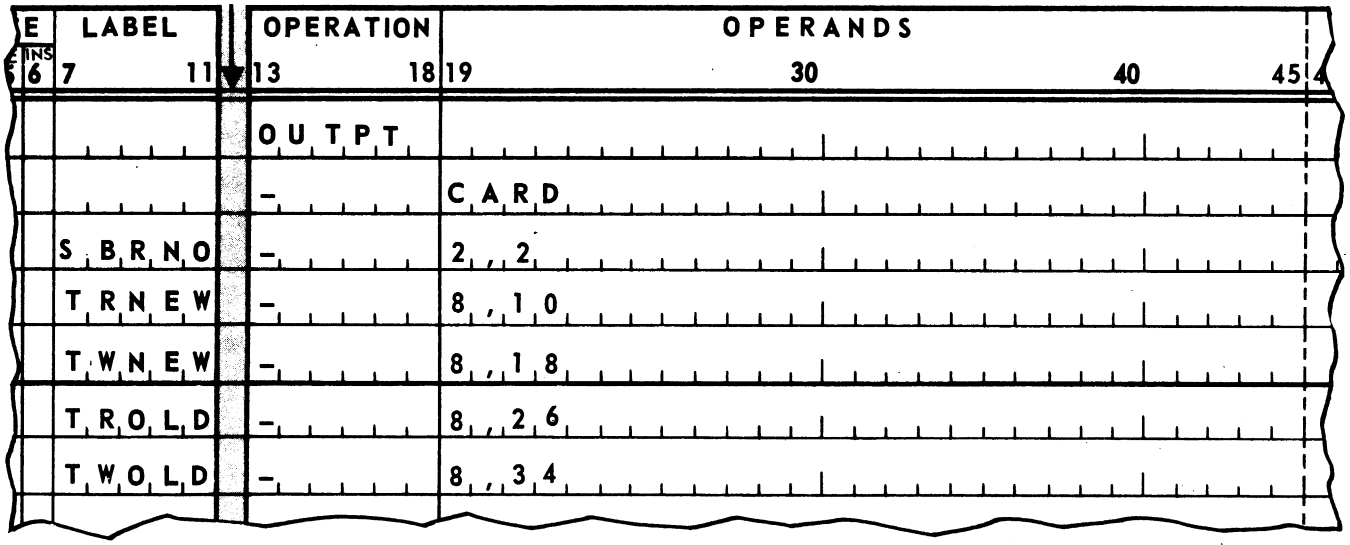


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		OUTPT	
		–	CARD

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 Business

can be described as:

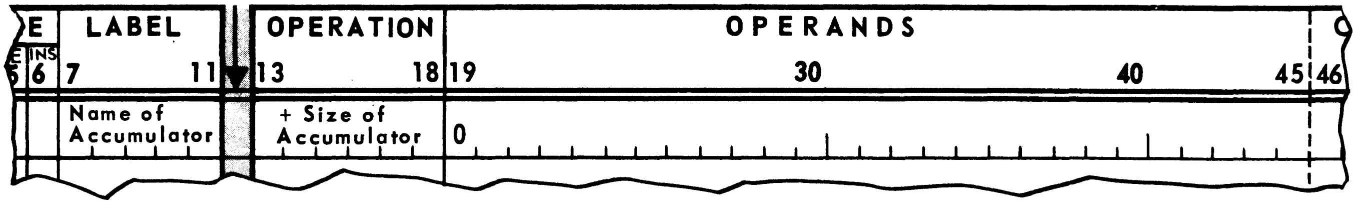


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
		OUTPT	
		–	CARD
	SBRNO	–	2,2
	TRNEW	–	8,10
	TWNEW	–	8,18
	TROLD	–	8,26
	TWOLD	–	8,34

NOTE: Fields must not exceed 16 characters in length.

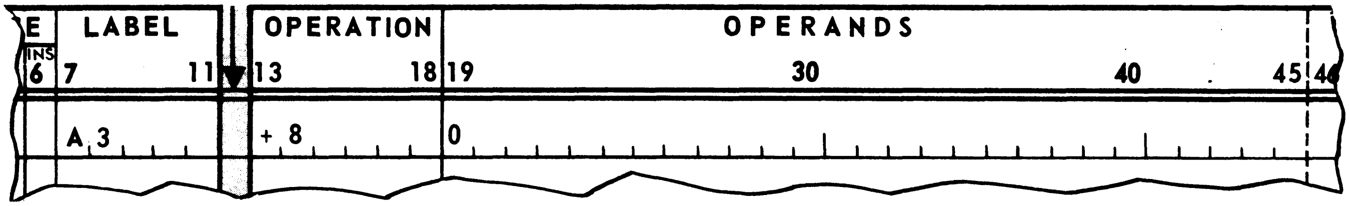
2. Accumulators

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:

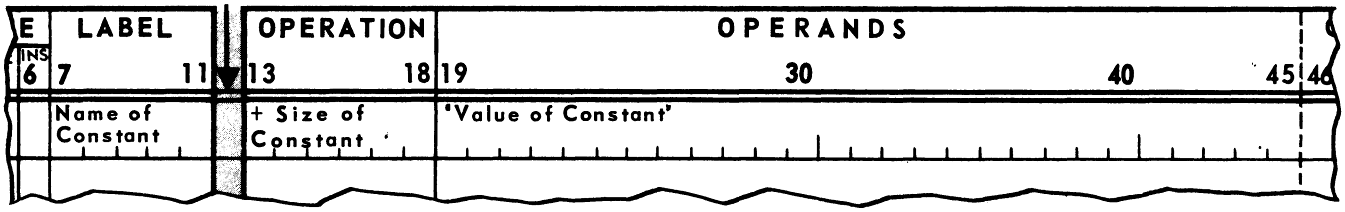


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
	A3	+8	0	

NOTE: Fields must not exceed 16 characters in length.

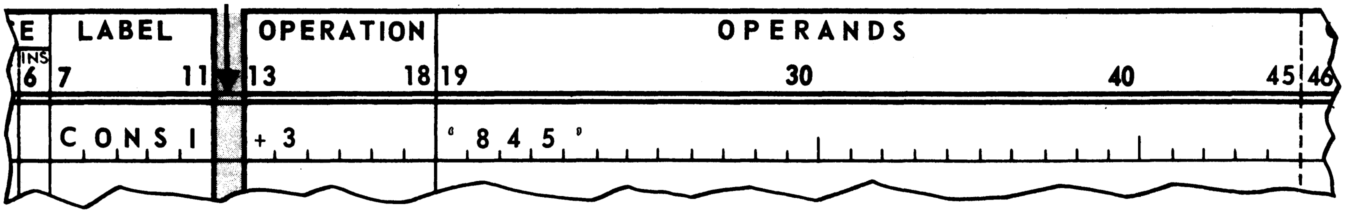
3. Constants

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



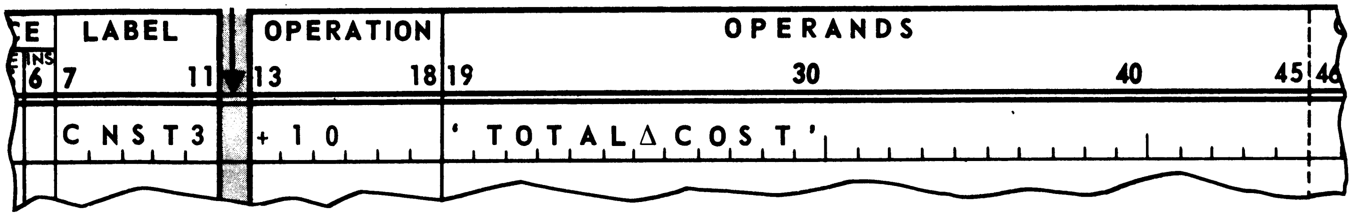
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
	Name of Constant	+ Size of Constant	'Value of Constant'	

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 TOTALΔCOST, the following line is written:



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
	CNST3	+10	'TOTALΔCOST'

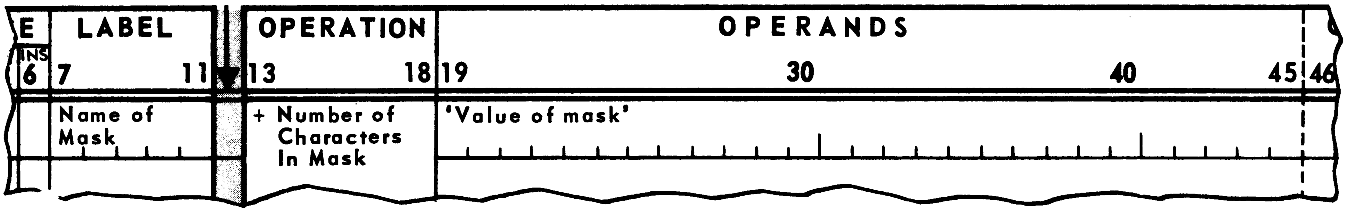
NOTE: Fields must not exceed 16 characters in length.

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

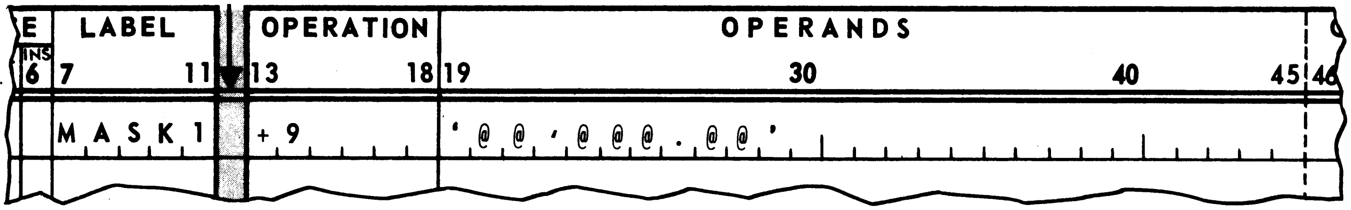
@@,@@@.@@

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

The above mask can be defined as



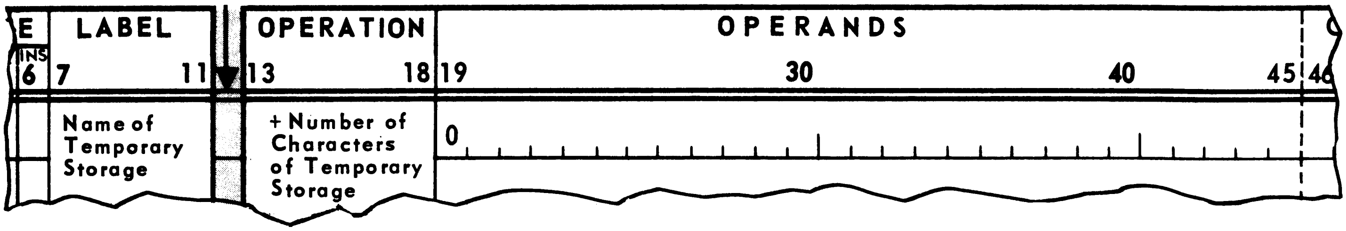
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
	MASK1	+9	'@@,@@@.@@'

If the least significant character of the mask is a '–' 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.

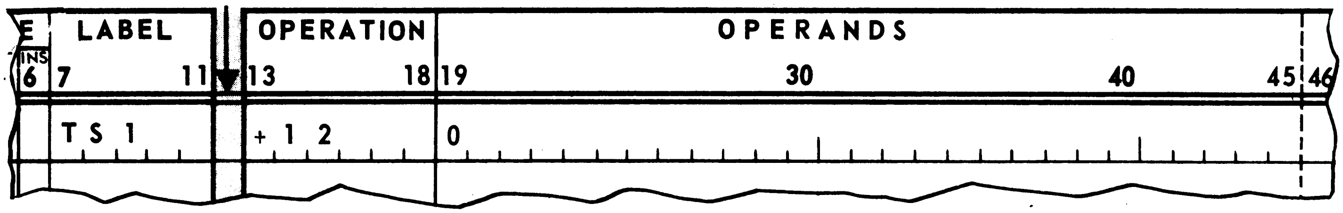
5. Temporary Storage Registers

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



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

For example,



E	LABEL		OPERATION	OPERANDS
INS 6	7 11		13 18	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.

C. 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.

1. Add a Field to an Accumulator

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

The contents of the field named in the first expression a. e 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:

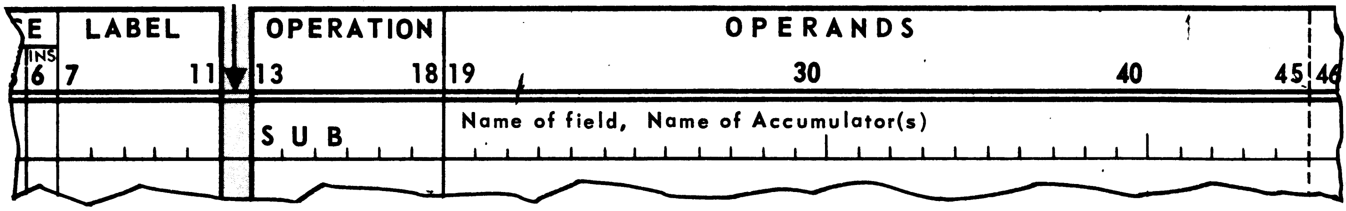
Add retail amount (RTAIL) to accumulator 1 (A1):

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

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

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

2. Subtract a Field from an Accumulator

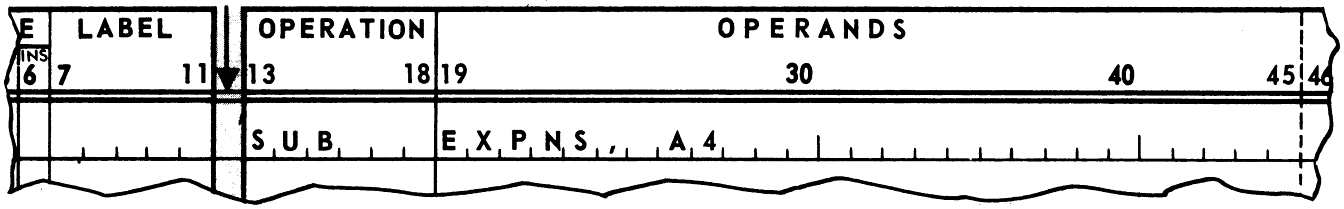


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		SUB	Name of field, Name of Accumulator(s)

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.

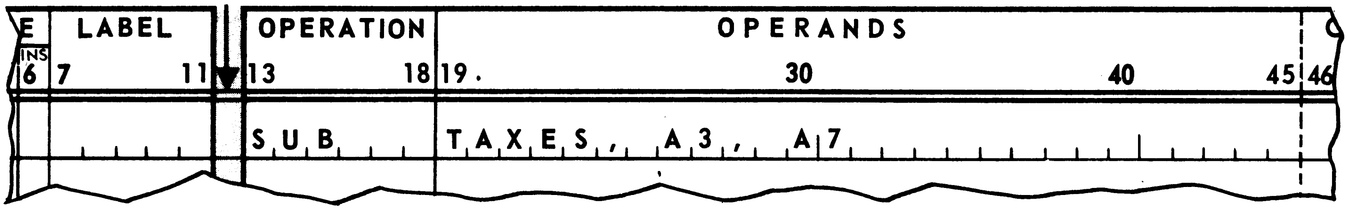
Examples:

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



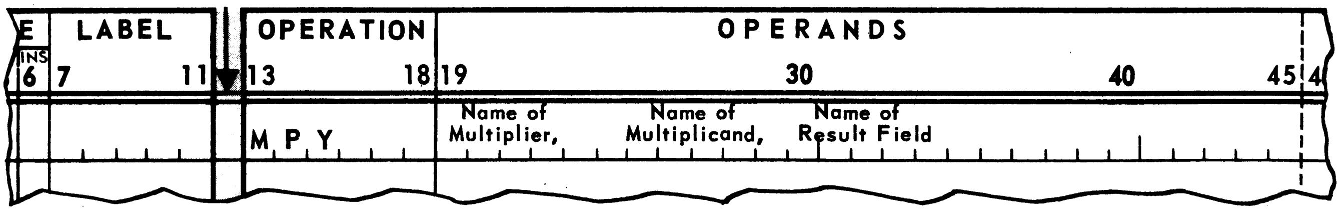
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		SUB	EXPNS, A4

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



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		SUB	TAXES, A3, A7

3. Multiply Two Fields

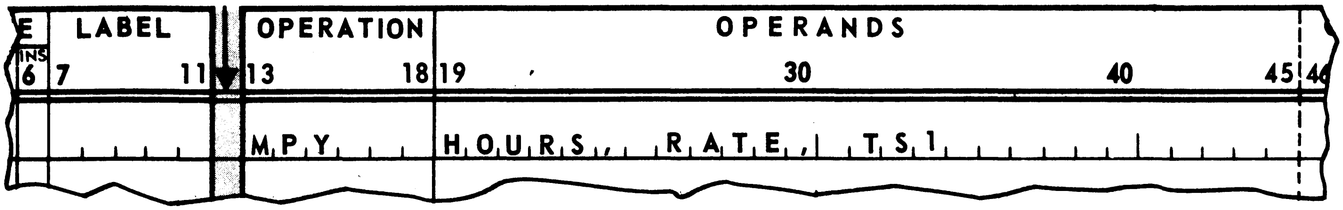


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 4
		MPY	Name of Multiplier, Name of Multiplicand, Name of Result Field

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.

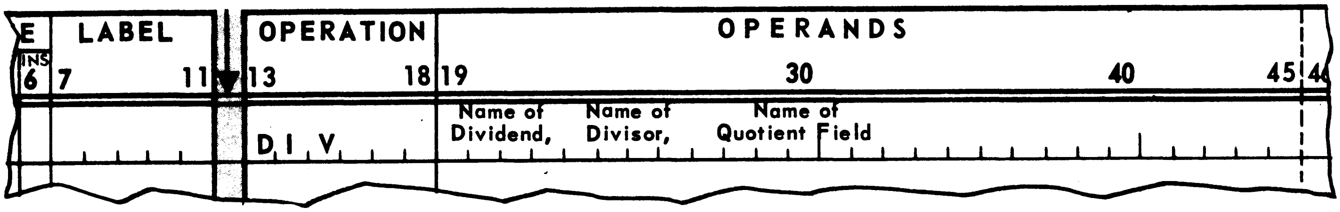
Example:

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	

4. 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.

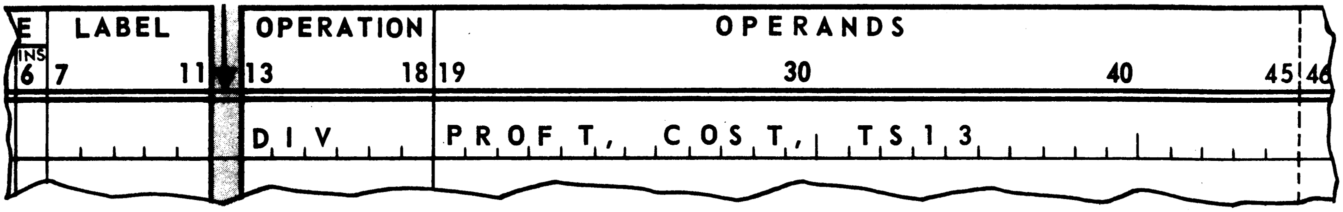
The object coding produced by REGENT prevents improper division from being attempted, including a division which would produce decimal overflow. UP-3912, the UNIVAC 1050 Central Processor Manual provides a description of the requirements which must be met if a valid divide operation is to be executed. The object program will stop if any of these are violated, the display indicating the nature of the violation. If the program is continued following the error stop, the operands will be arbitrarily modified in the following manner:

> If the defined length of the quotient field is greater than 8 characters, it will be shortened to 8, the most significant digit(s) being omitted.
>
> If the length of the quotient (as modified above if necessary) plus the length of the divisor is less than dividend length, or exceeds 16 characters, the divisor is altered to the required length.

If the absolute value of the divisor is such that decimal overflow will occur if a division is attempted, the divide instruction will not be executed.

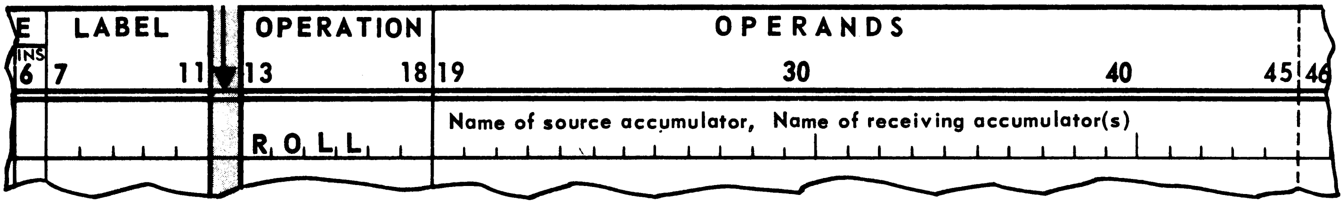
Example:

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



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

5. Add to Accumulator(s) and Reset Source

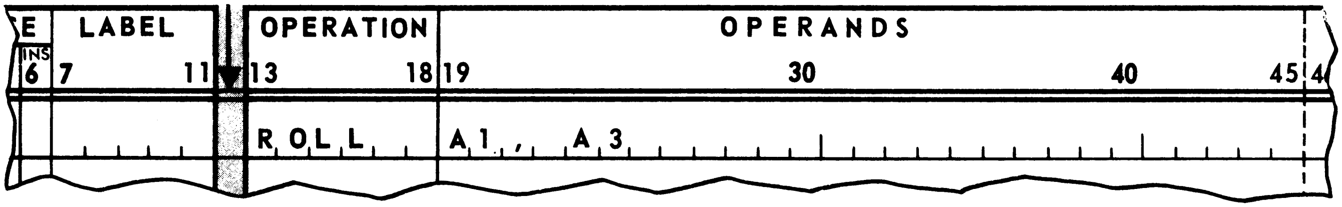


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		ROLL	Name of source accumulator, Name of receiving accumulator(s)

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.

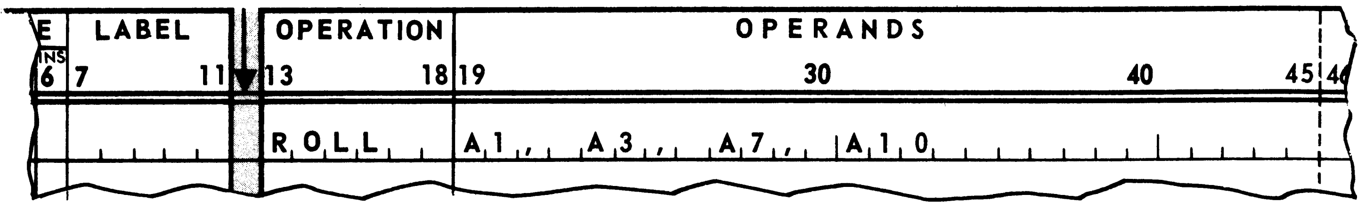
Examples:

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



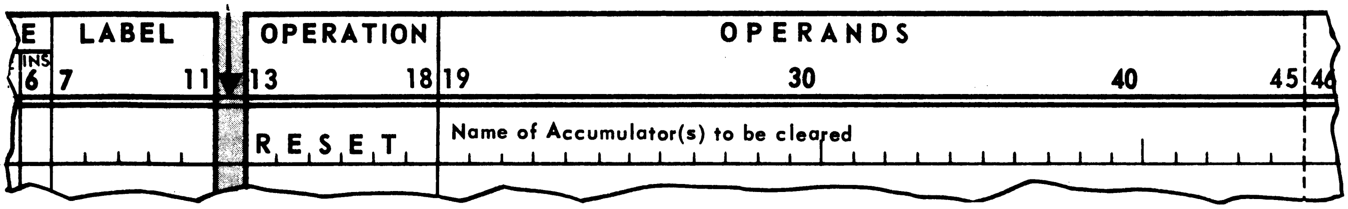
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		ROLL	A1, A3

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



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

6. Reset Accumulator

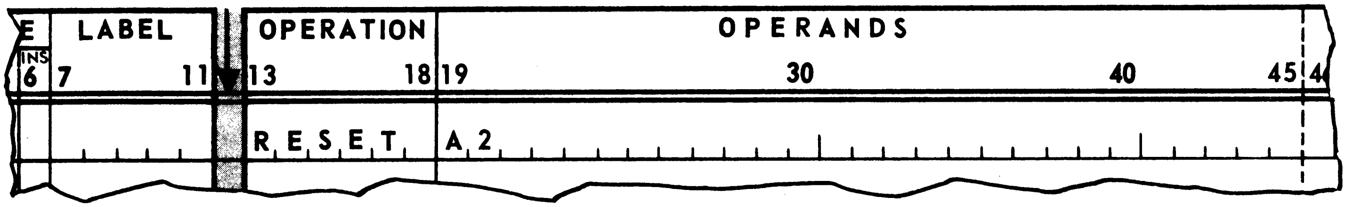


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		RESET	Name of Accumulator(s) to be cleared

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

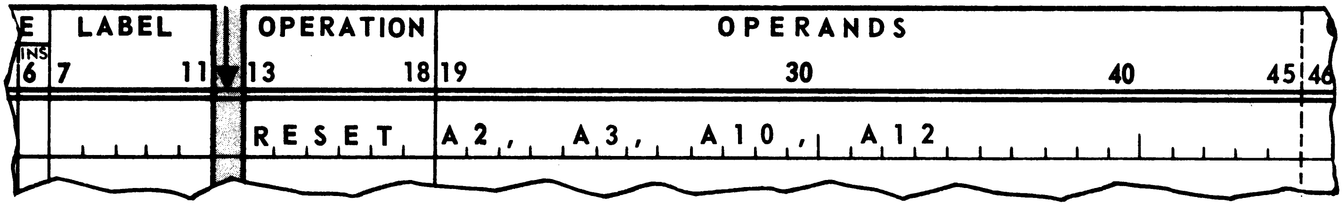
Examples:

Reset accumulator 2 (A2):



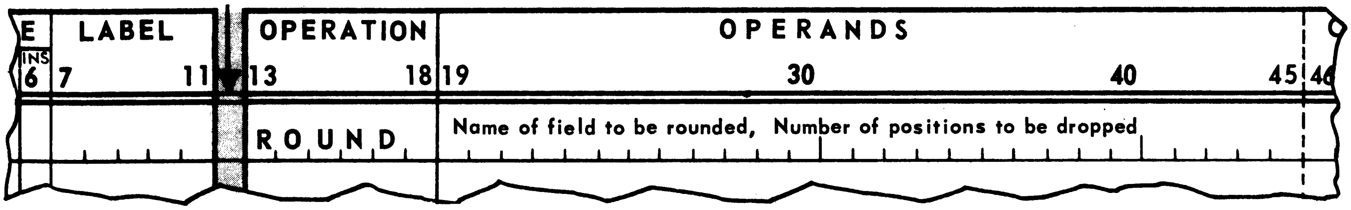
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		RESET	A2

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



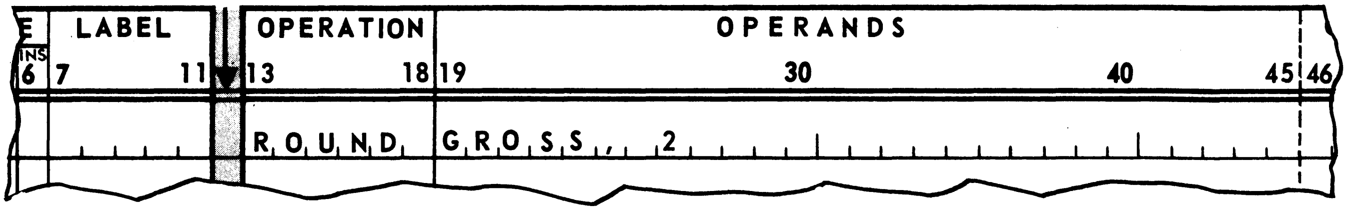
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		RESET	A2, A3, A10, A12

7. 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

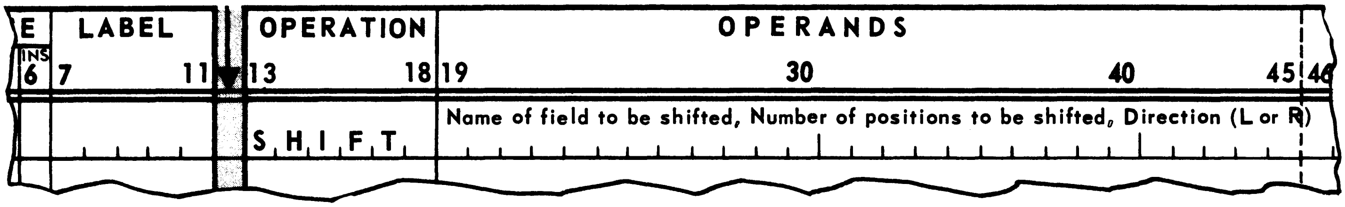
The field named in the first expression will be half adjusted and right justified with zeros inserted to the left. The sign of the field remains unchanged.



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

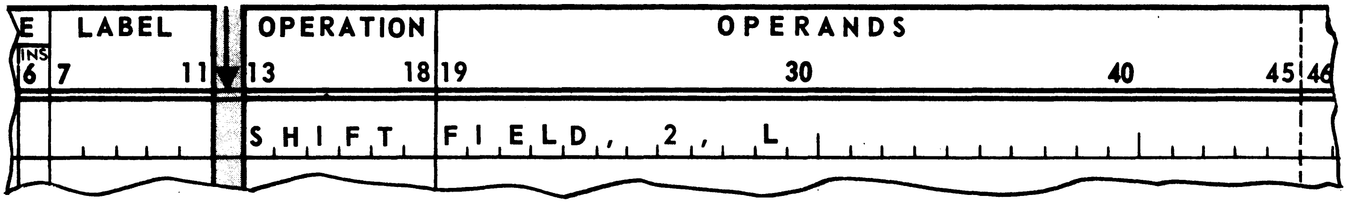
8. Position Adjust a Field (Shift)



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)	

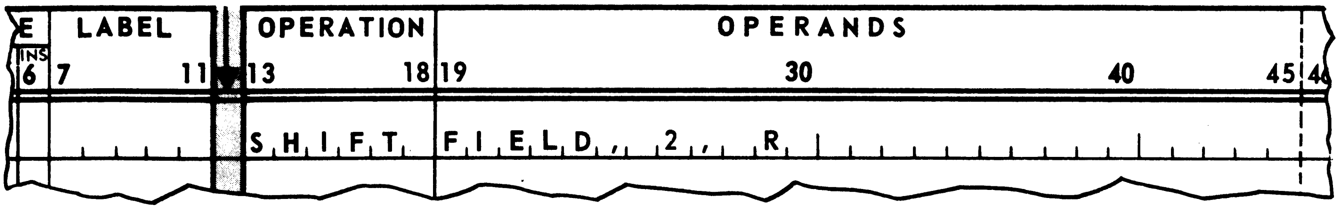
The contents of the named field are shifted destructively, that is, characters shifted beyond the limits of the field are lost. If the third expression is L, the field contents are shifted left with decimal zeros inserted at the right. If the third expression is R, the field is shifted right with zeros 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 INS 6	LABEL 7 11		OPERATION 13 18	OPERANDS 19 30 40 45	46
			SHIFT	FIELD,2,R	

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

D. DATA MOVEMENT AND EDITING

1. 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:

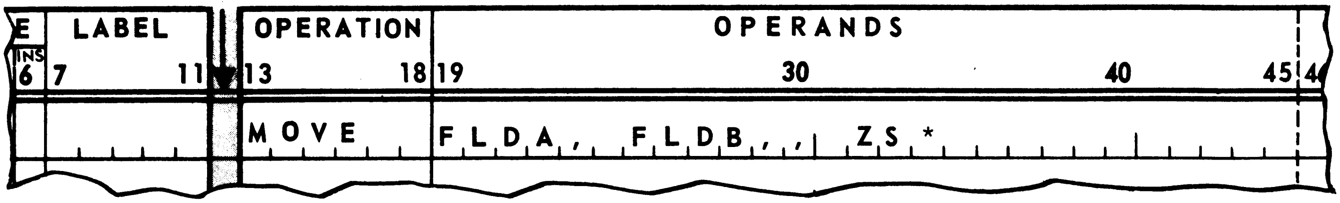
ZS	Leading zeros and commas are changed to blanks.
ZS*	Leading zeros and commas are changed to asterisks.
ZS$	Leading zeros and commas are changed to blanks, and a $ is inserted to the left of the most significant digit.

Zero suppression may be accomplished without editing if the third expression is left blank, and a comma is inserted in its place.

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	ΔΔ$123.45


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

```
            MOVE  FLDA,FLDB,,ZS*
```

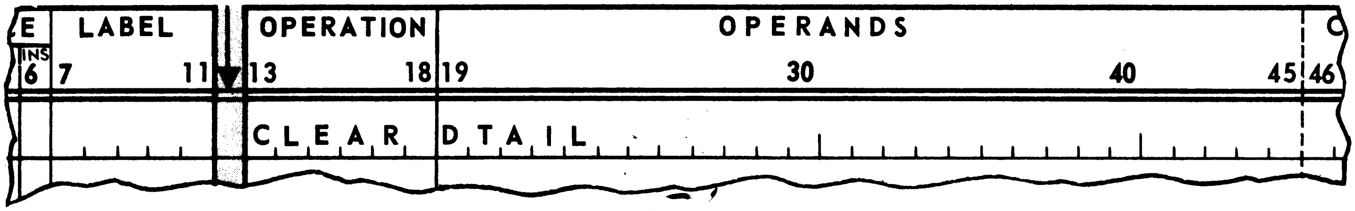
FLDA	before	0012345
FLDB	after	**12345

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.

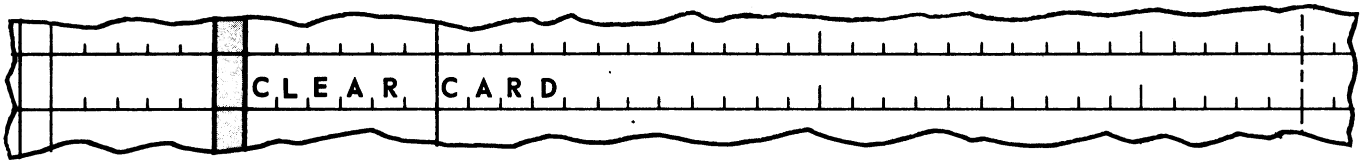
2. Clear

To clear an area to blanks, the programmer writes:


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

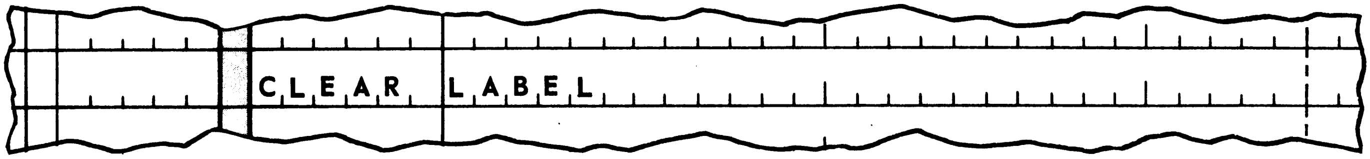
```
            CLEAR DTAIL
```

or



```
            CLEAR CARD
```

or



```
            CLEAR LABEL
```

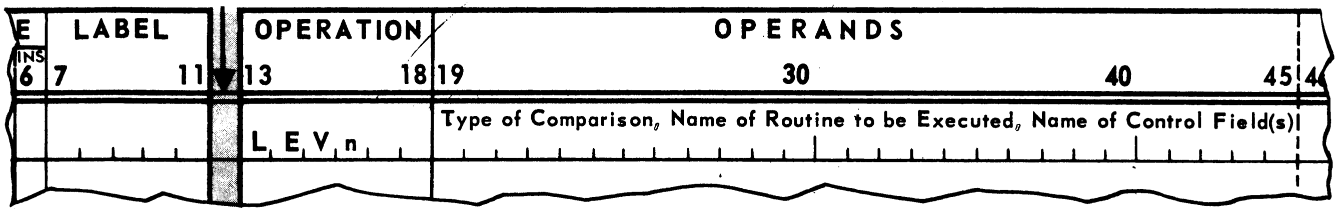
This causes blanks to be inserted in the entire area of the operand. If the operand is DTAIL, the detail print area (128 characters) will be filled with blanks. If the operand is CARD, the card punch output area (80 characters) will be filled with blanks. If the operand is the label of a field (16 characters 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.

E. 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.

1. 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)

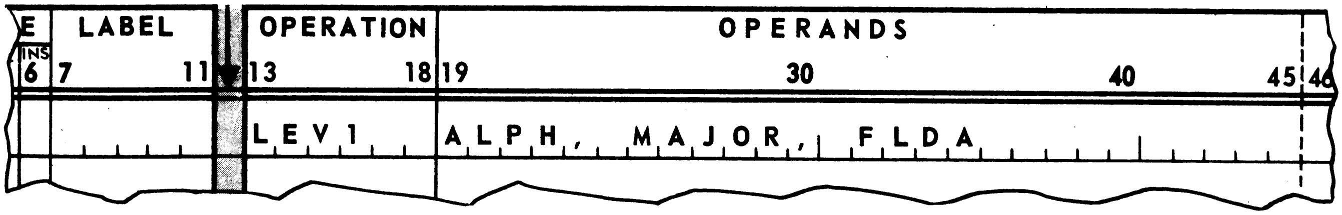
Testing for the presence of a control break is accomplished by writing a LEVn statement. Such a statement consists of the entry LEVn in the operation field where n is a number from 1 through 9, and three, four, or five expressions in the operand field. The first expression will be ALPH, DEC, or ZONE denoting the type of comparison desired. Where the first expression is ALPH or DEC, the second expression is the label of the routine 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.

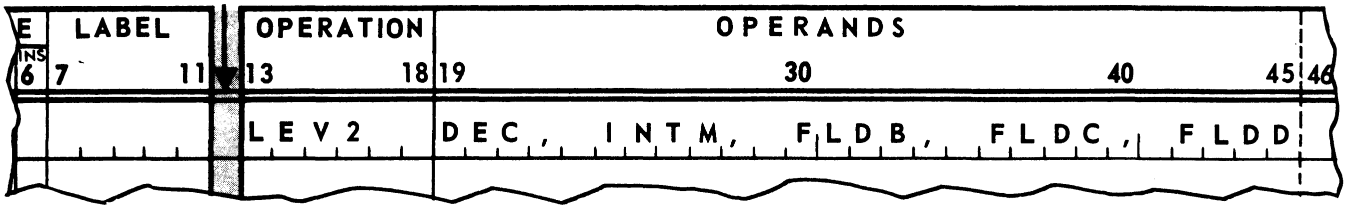
In the generated program, when a total level (control break) occurs, the occurrence of all higher numbered total levels are inhibited. Therefore, the routine executed when a total level occurs should include the execution of the next lower level.

Examples:



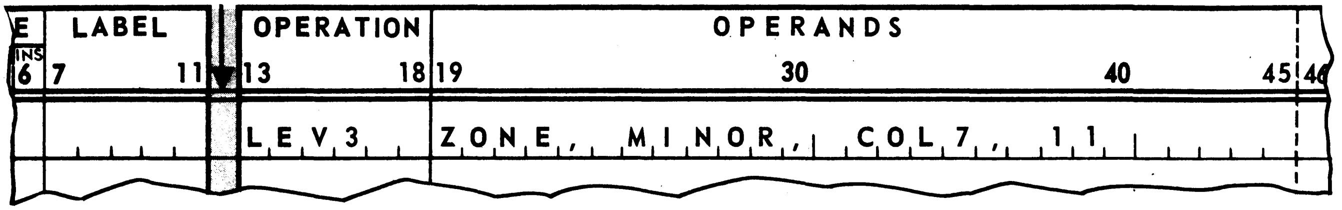
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		LEV1	ALPH, MAJOR, FLDA

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



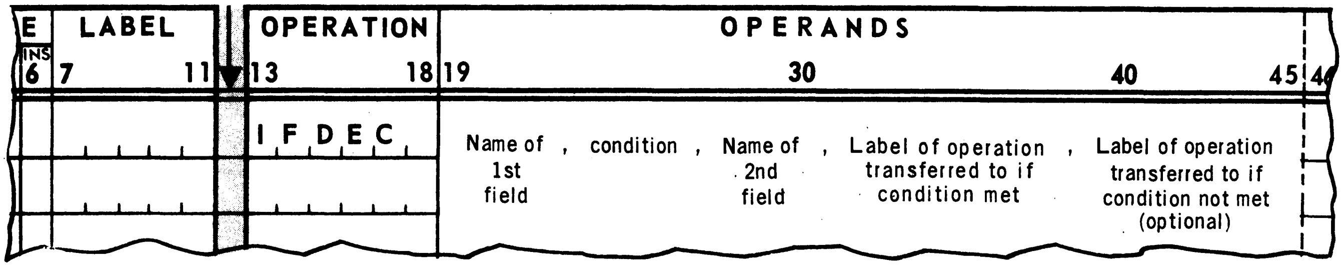
```
E    LABEL        OPERATION   OPERANDS
INS
6  7        11    13       18 19              30                  40      45 46
                  LEV3        ZONE, MINOR, COL7, 11
```

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 character field labeled COL7. If the condition is met, MINOR will be executed.

2. Comparison of Fields

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



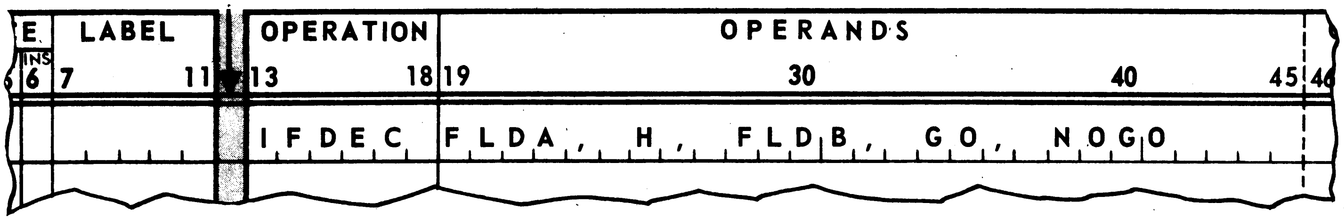
```
E    LABEL        OPERATION   OPERANDS
INS
6  7        11    13       18 19              30                  40      45 46
                  IFDEC       Name of , condition , Name of , Label of operation , Label of operation
                              1st                   2nd       transferred to if    transferred to if
                              field                 field     condition met        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.

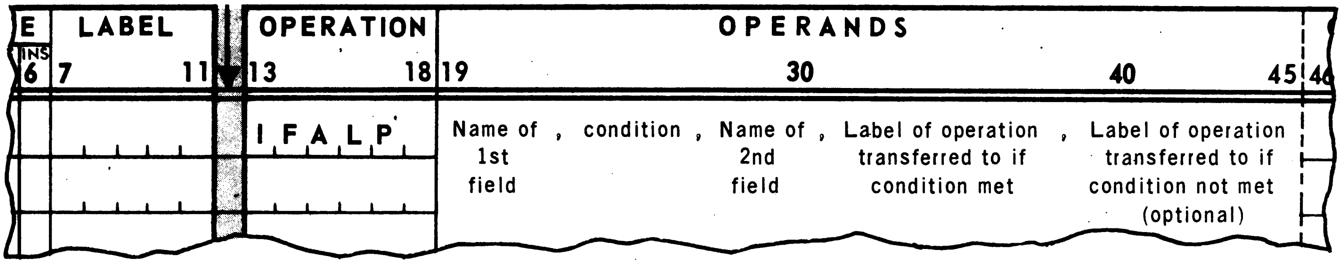
Example:



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

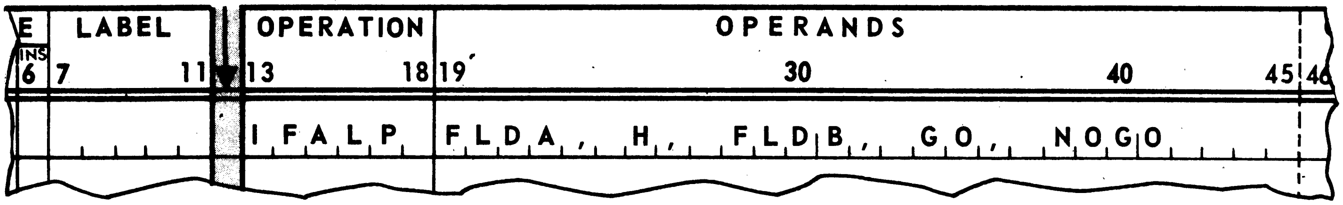
If the value of FLDA is 101 and the value of FLDB is 100, the condition H is met and control will be transferred to the operation labeled GO. If the condition is not met, control will, in this case,be transferred to the operation labeled NOGO. If the last expression were blank, the next instruction in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

IFALP causes an alphabetic (binary) comparison.



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

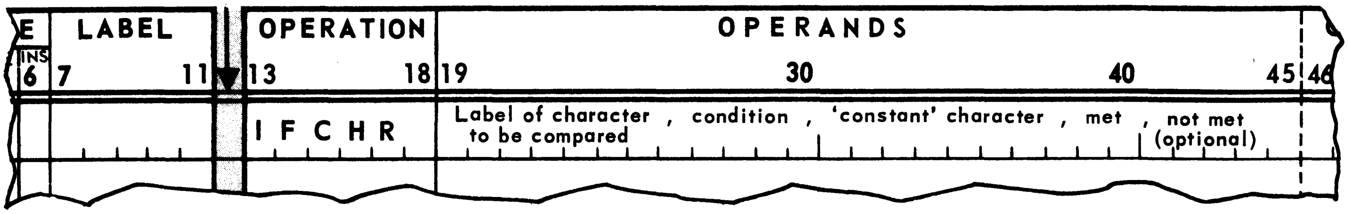


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45
		IFALP	FLDA,H,FLDB,GO,NOGO

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.

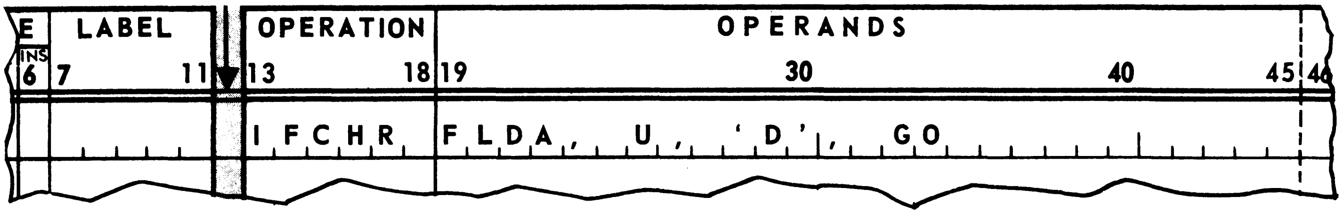
3. Comparison of Character, Zone, and Signs

a. 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)

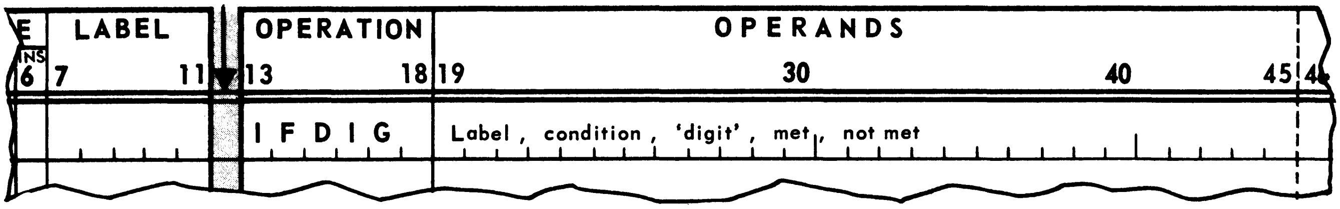
This operation is the same as IFDEC and IFALP except that only a single character is compared to the actual character designated by the 3rd operand. If the first named expression is longer than 1 character, only the least significant character is compared.



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.

b. Digit Comparison



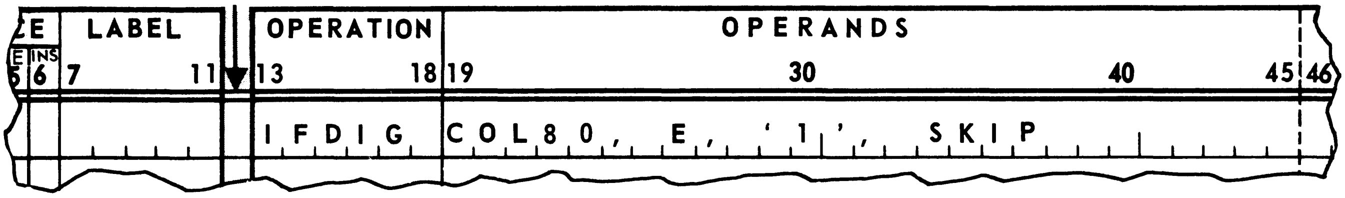
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		IFDIG	Label , condition , 'digit' , met , not met

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 .

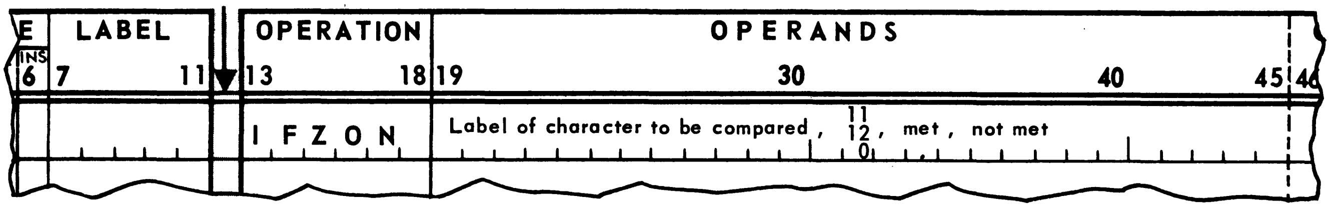
Example:


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

```
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.

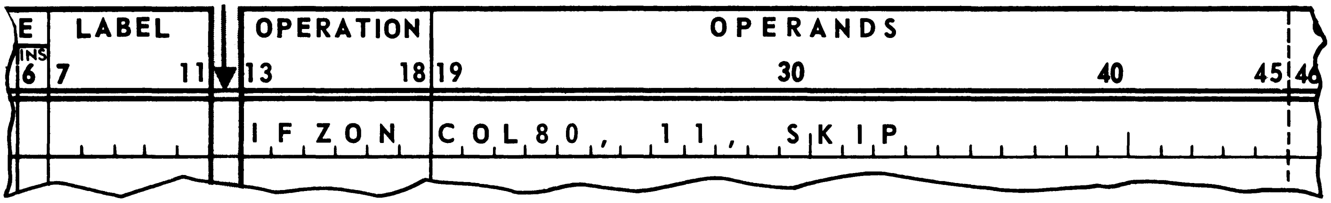
3. Zone Comparison


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

```
                                            11
IFZON  Label of character to be compared ,  12 ,  met ,  not met
                                            0
```

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.

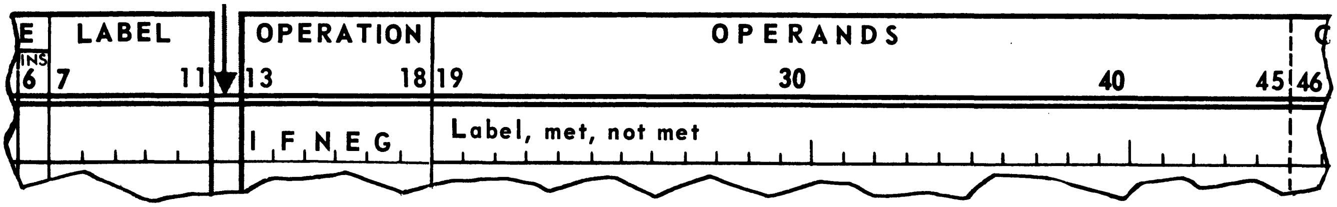
Example:


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

```
IFZON  COL80,11,SKIP
```

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

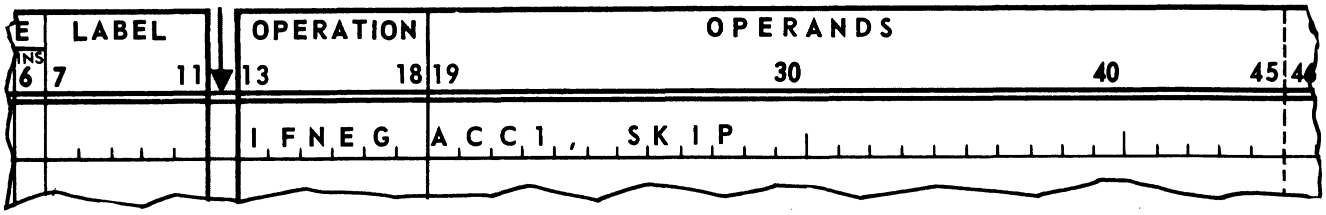
4. Sign Comparison


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

```
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:



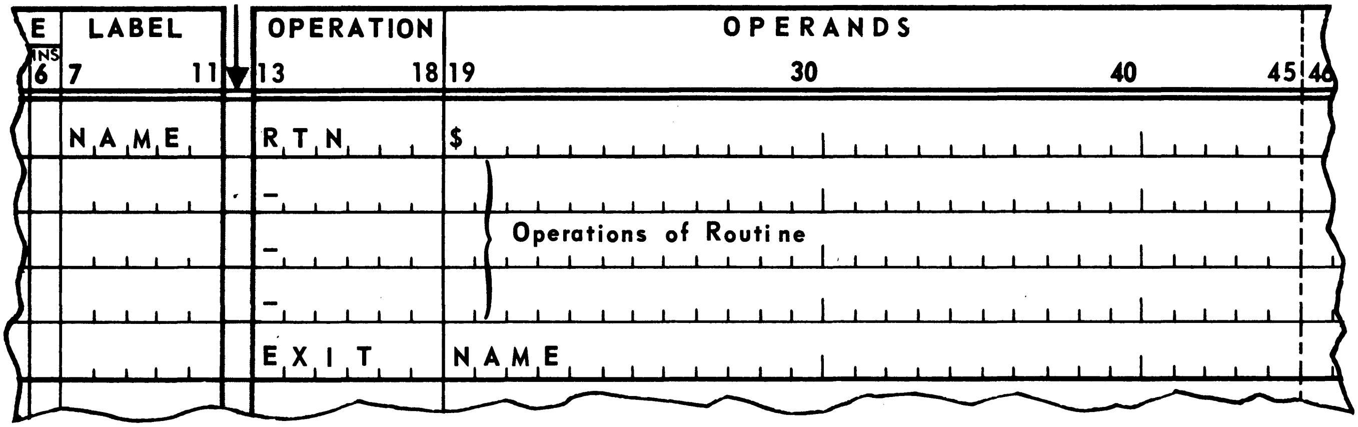
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		IFNEG	ACC1, SKIP

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:



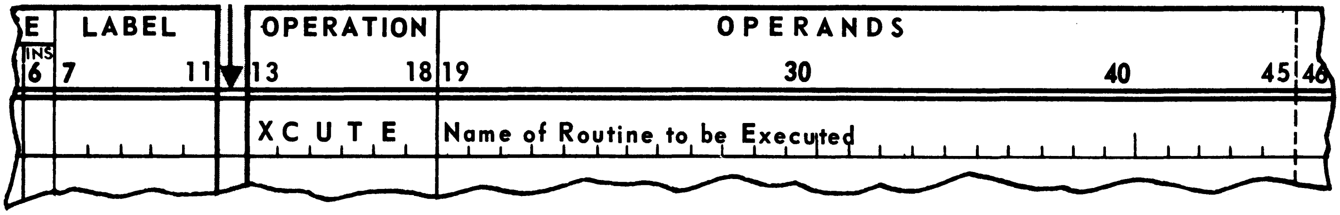
E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	NAME	RTN	$
		–	
		–	Operations of Routine
		–	
		EXIT	NAME

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

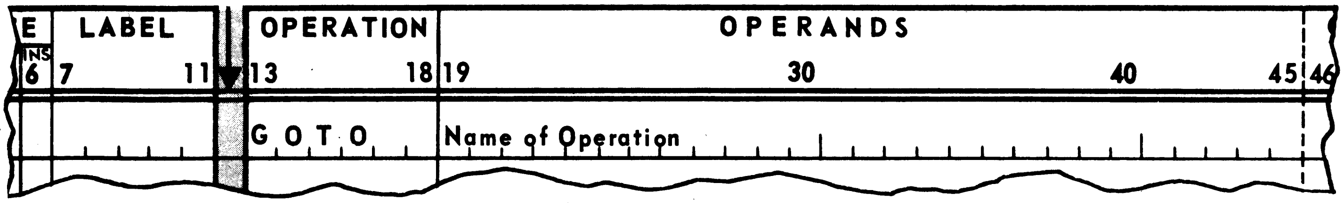
Routines are normally executed as a result of the occurrence of either a LEV break or page overflow.

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

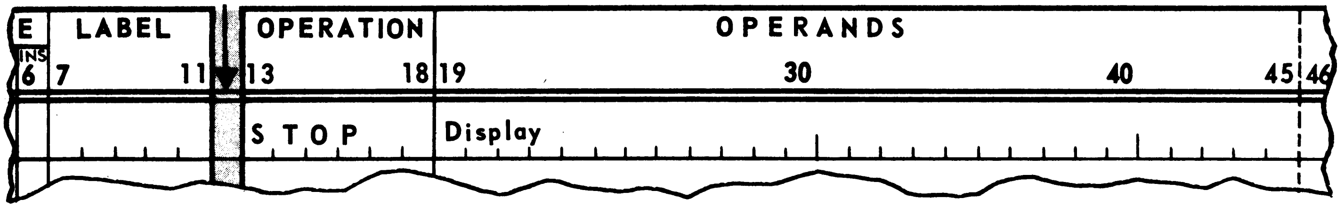
6. Unconditional Transfer of Control



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.

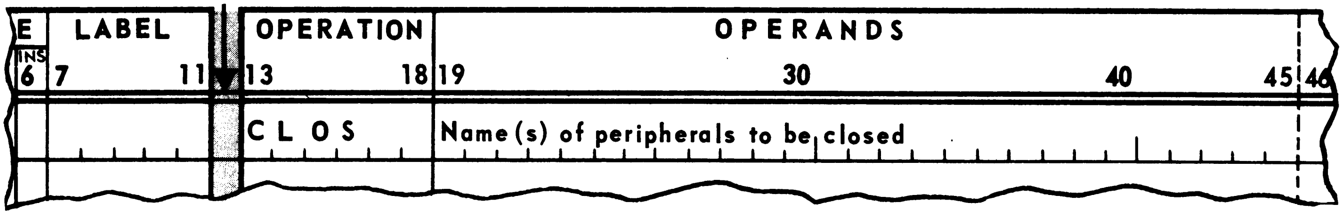
7. Display Stop



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.

8. Closing Run



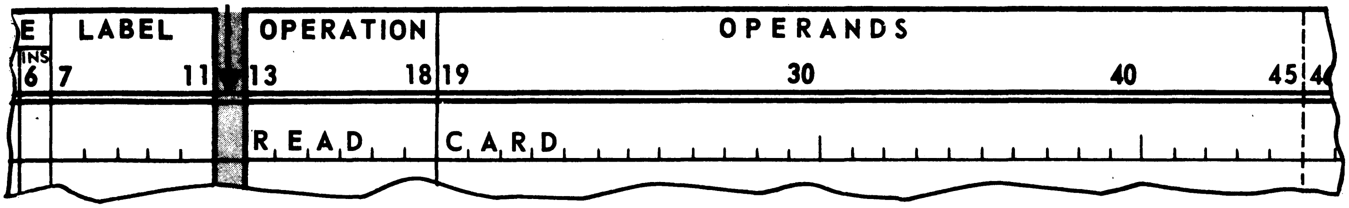
E INS 6	LABEL 7 11	▼	OPERATION 13 18	OPERANDS 19 30 40 45 46
			CLOS	Name (s) of peripherals to be closed

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

F. INPUT/OUTPUT CONTROL

1. Input Reading

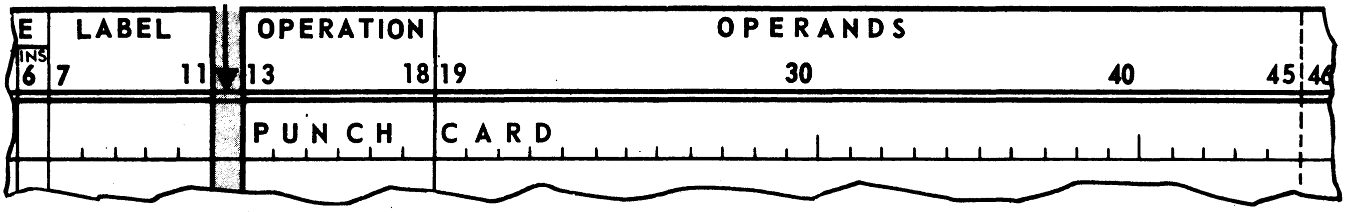
To read an input record, one writes



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

2. Output Punching

To punch a summary card, one writes

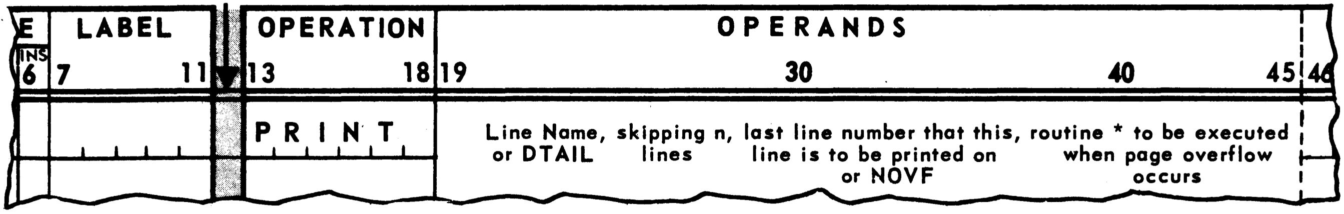


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		PUNCH	CARD

3. Output 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

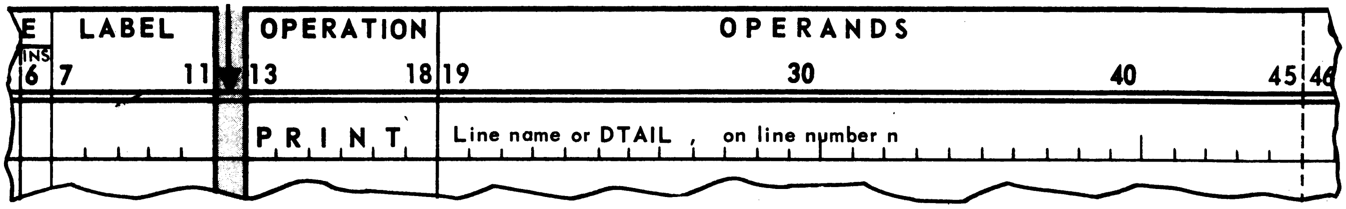


E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
		PRINT	Line Name, or DTAIL skipping n, lines last line number that this, line is to be printed on or NOVF routine * to be executed when page overflow occurs

* *A form 1 print must not be incorporated in the page overflow routine.*

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

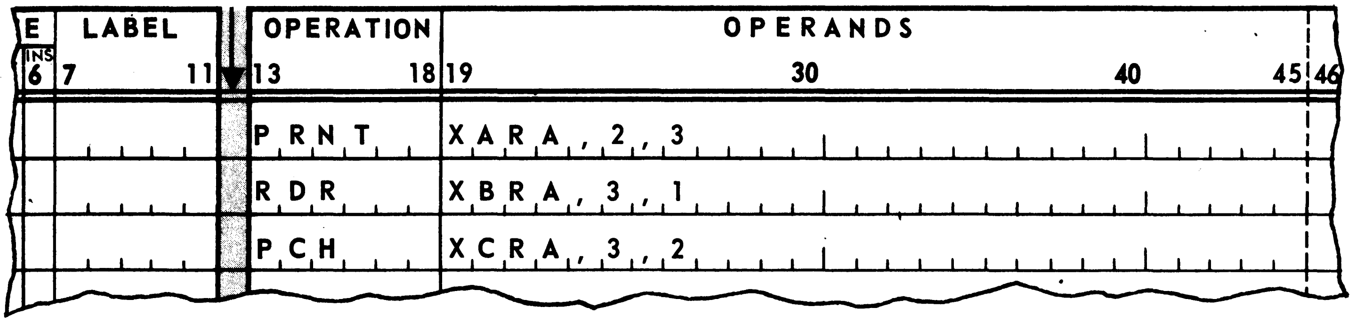
This command will cause the printing of the line named or of a detail line on the line number designated.

G. SELECTION OF INPUT/OUTPUT CONTROL ROUTINES

1. Magnetic Tape System

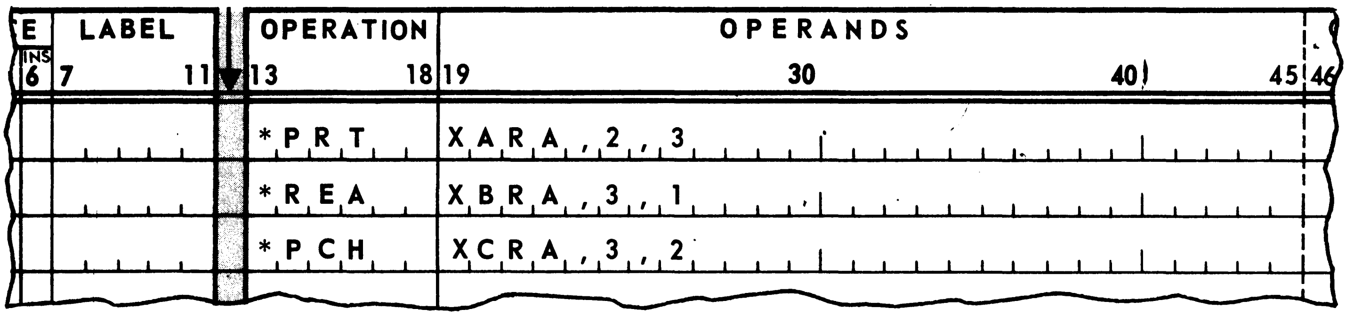
If the PAL Tape Assembler is being used to generate the REGENT program, all of the I/O control routines being used by the installation are available to the REGENT user. A call line must be written for each device being used in the program. The operation field of the call line will contain the name of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

a. Call lines for I/O routines which operate under control of the I/O Coordinator



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	

b. Call lines for I/O routines which operate without the I/O Coordinator



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

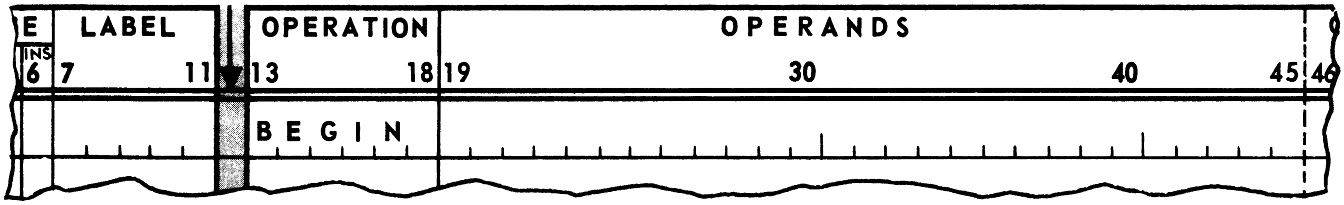
2. Card System

If the Card REGENT program is used, the call lines for I/O controls 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 after the first card (BEGIN).

H. REQUIRED OPERATION STATEMENTS

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

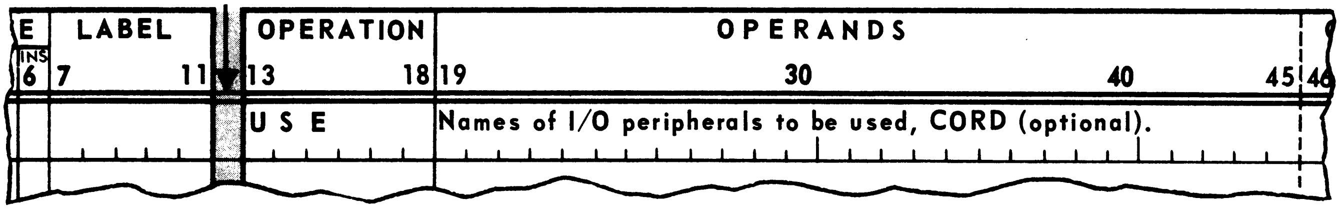
1.



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 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.

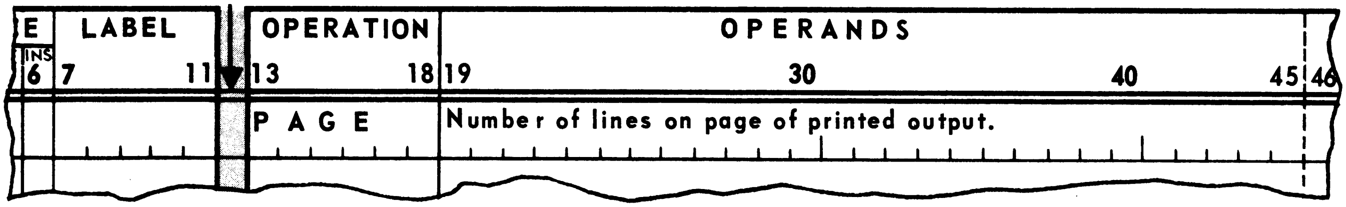
2.



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		USE	Names of I/O peripherals to be used, CORD (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. The operand, CORD, is used to indicate that the program is to run under control of the I/O Coordinator (must not appear in input to Card REGENT)

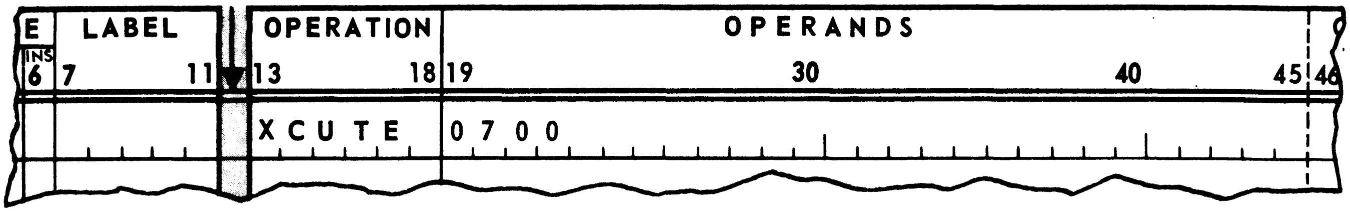
3.



E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45	46
		PAGE	Numbe r 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.

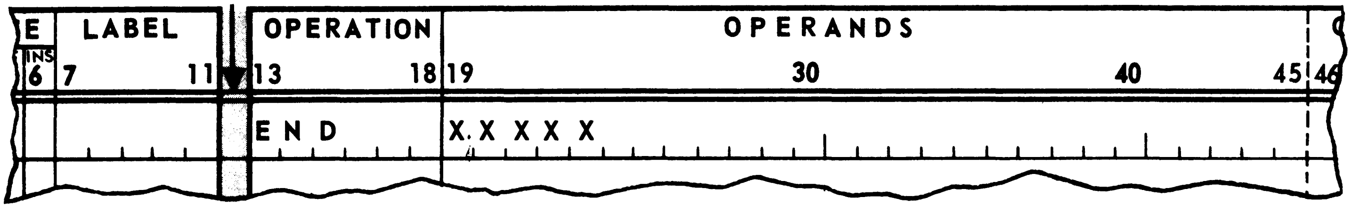
4.



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

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

5.



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

The last line (card) of the source program will be as above with no variations.

I. 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.

a. 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.

b. 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.

c. 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.

d. 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.

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

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

J. OPERATING INSTRUCTIONS

1. Magnetic Tape REGENT

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

2. Card REGENT

The Card REGENT program produces a PAL source deck which is subsequently assembled using the PAL Card Assembler, after the desired I/O control routines are added. The steps required to produce an object program are as follows:

a. The Card REGENT object program is loaded from the card reader, followed by the source cards.

The intermediate source output is punched and the input cards are listed on the printer. (This listing may be eliminated by setting Sense Switch 1.)

b. The intermediate output cards are removed from the punch. The I/O control routines required for the programs are selected from those supplied with Card REGENT. These are inserted after the first (BEGIN) card of the intermediate deck. The deck is then ready to be used as direct input to the PAL Card Assembler.

Output cards are sequenced by Card REGENT beginning with 05000. The I/O routines have sequence numbers lower than 5000. This is done for future reference or for sorting purposes. Input statements will be punched in the output as comment cards containing **** in columns 7-11 and blanks in the sequence number field.

c. Assembly

The operating instructions for the PAL Card Assembler should be followed. After both passes of the assembly have been completed, the I/O control cards should be removed and stored for future use.

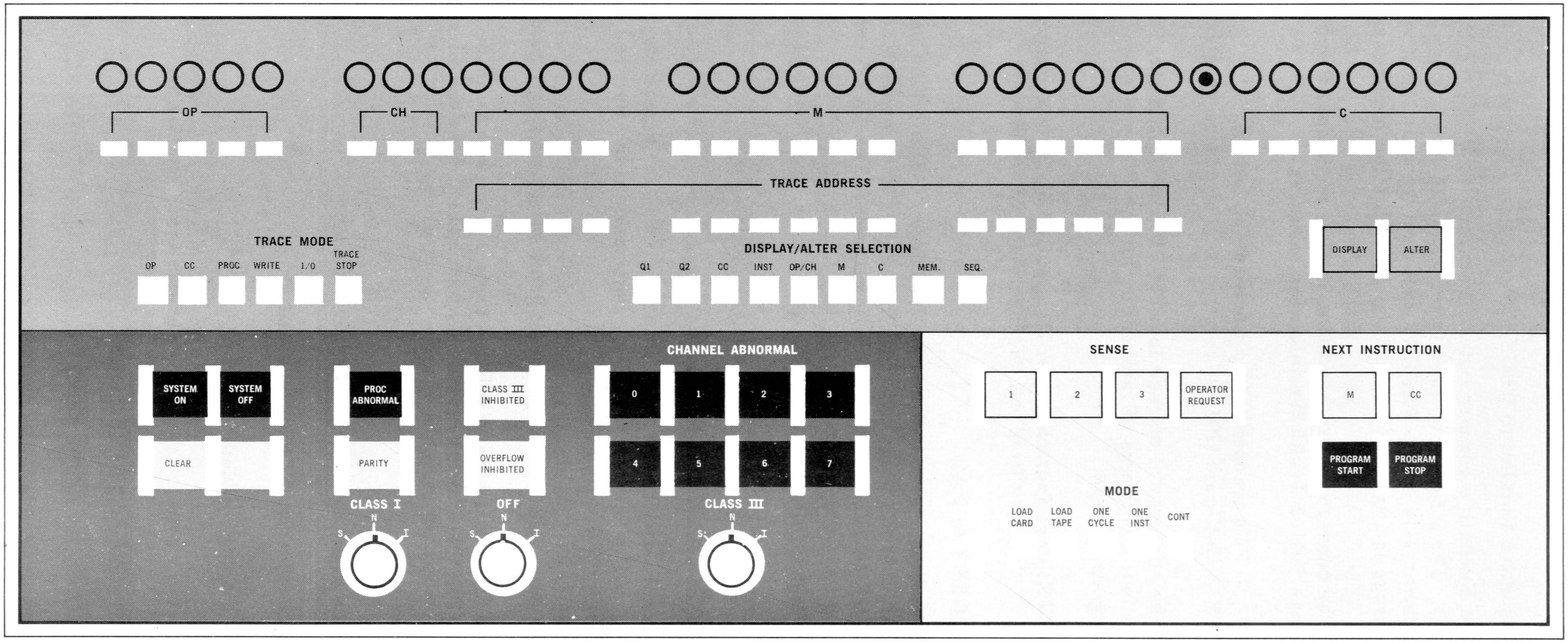


Figure 9-1. Central Processor Console

(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.

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.

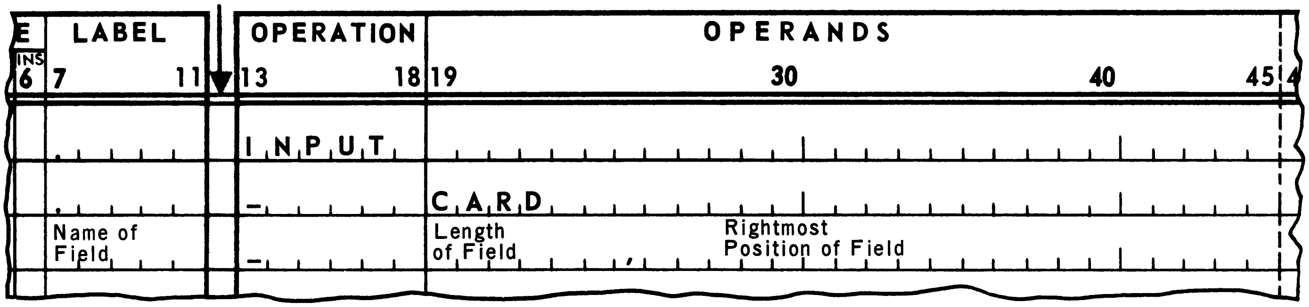
The use of the UNIVAC 1004 as the input/output device for REGENT programs precludes the use of the 1050 standard peripheral devices (card reader, card punch, and printer), and their associated input/output control routines. Whereas in the UNIVAC 1050 REGENT, the I/O selection directive and I/O field 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.

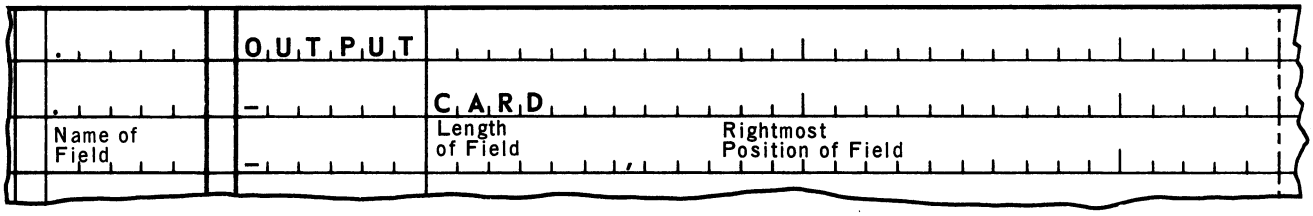
The following I/O field descriptions must appear directly behind the related I/O control routine card deck:

Card reader control routine input area,



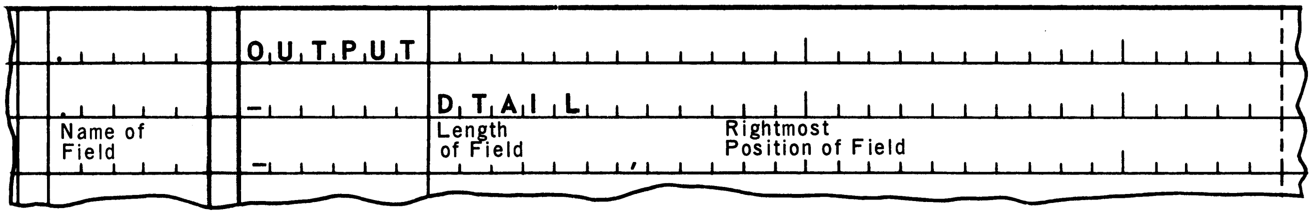
E INS 6	LABEL 7 11	12	OPERATION 13 18	OPERANDS 19 30 40 45
	.		INPUT	
	.		-	CARD
	Name of Field		-	Length of Field , Rightmost Position of Field

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	LABEL		OPERATION	OPERANDS	
INS 6	7 11	▼	13 18	19 30 40 45	46
			*X4HR	7 ← assumes ch. 7 although X4HR also handles 3	
			*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 MOVE-MENT	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