UNIVAC
DATA PROCESSING DIVISION

1005

REPORT PROGRAM GENERATOR EXTRACT UP-4072.2

Programming Information Exchange

RELEASE

This UNIVAC 1005 System Programming Information Exchange Bulletin 2, UP-4072.2, announces the release and availability of "UNIVAC 1005 REPORT PROGRAM GENERATOR EXTRACT," covers and 76 pages.

The UNIVAC 1005 Report Program Generator is a problem oriented programming system designed to reduce substantially the time and effort necessary to translate general data processing and reporting requirements into detailed computer instructions. The 1005 Report Program Generator, on the basis of a series of statements provided by the user, produces a computer program which will prepare the desired reports. The UNIVAC 1005 Report Program Generator provides a printed listing of both the user's input statements and the generated assembly language code. After the assembly phase, this generated code is an efficient ready-to-run object program. "UNIVAC 1005 REPORT PROGRAM GENERATOR EXTRACT" is a provisional document and will be replaced by a more permanent one.

This P.I.E. bulletin is the second of a series to be issued concerning the UNIVAC 1005 System. All P.I.E. bulletins have form numbers and may be ordered with accompanying attachment by their "UP" number, as this one, UP-4072.2.

Automatic distribution of UP-4072.2 has been made in quantity to Area and Territory locations and to internal lists as indicated below. Additional copies of "UNIVAC 1005 REPORT PROGRAM GENERATOR EXTRACT" may be requisitioned from Holyoke, Massachusetts, via a Sales Help Requisition through your local UNIVAC Manager.

MANAGER,
Systems Programming Library Services

TO LISTS	ATTACHMENTS	THIS SHEET IS
211 (less 217), 692 and 153, P.I.E. bulletin only.	"UNIVAC 1005 REPORT PROGRAM GENERATOR EXTRACT," UP-4072.2, plus P.I.E. bulletin to 10 U, 217, 630 and 650.	1005 System P.I.E. Bulletin 2, UP-4072.2 DATE February 11, 1966

UNIVAC 1005
REPORT PROGRAM GENERATOR
EXTRACT

This document is provisional in nature and is intended as a vehicle for meeting immediate needs with regard to system familiarization and orientation. UNIVAC ® Division of Sperry Rand Corporation reserves the right to change and/or modify such information contained herein as may be required by subsequent system developments.

PRINTED IN U.S.A.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

UNIVAC 1005

REPORT PROGRAM GENERATOR EXTRACT

A. INTRODUCTION

The UNIVAC 1005 Report Program Generator is a problem oriented programming system designed to reduce substantially the time and effort necessary to translate general data processing and reporting requirements into detailed computer instructions. No knowledge of computer programming is required other than the basic rules for writing programs in the UNIVAC 1005 Assembly Language. The 1005 Report Program Generator, on the basis of a series of statements provided by the user, produces a computer program which will prepare the desired reports. The user's statements, punched into cards, provide:

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

(2) The formats of the desired output-reports--printed documents, punched summary cards, or both.

(3) The sequence of operations to be performed on the input files--arithmetic operations, input/output, data movement, controls.

The UNIVAC 1005 Report Program Generator provides a printed listing of both the user's input statements and the generated assembly language code. After the assembly phase, this generated code is an efficient ready-to-run object program.

B. GENERAL DESCRIPTION

The UNIVAC 1005 Report Program Generator translates a user's source input statements into 1005 assembly language. Each input statement consists of one operation mnemonic, one or more operands, optionally one label, and optional comments. One or more assembly language statements are generated for each source input line, and the printed output of the Report Program Generator alternates between the source code and its generated code. In addition, assembly language instructions may be included with Report Program Generator statements. These instructions will be copies into the assembly deck generated; a "no macro" message will be printed.

Source input code for the Report Program Generator is prepared using 1005 Assembly Language coding forms. The information on the forms is then punched into cards.

The operation mnemonic (referred to as "the macro" below) is coded in columns 6 thru 10 of the source input; the first character of the macro is coded in column 6, and the remaining characters must follow with no intervening blanks. As an example, of the eight configurations shown below, only the first and fifth are correct.

LABEL	OPER-ATION	OPERAND ONE	
12345	6789$^{1}_{0}$		
	READ		correct
	READ		incorrect
	REA D		incorrect
	R EAD		incorrect
	SET		correct
	SET		incorrect
	SET		incorrect
	S E T		incorrect

When a source input statement is found to contain an invalid operation mnemonic, the statement is punched without alteration into the output deck and is printed with the message "NO MACRO" appended at the right of the printed line (in columns 82 through 89).

Labels A, B (when required), and C (when required) are coded in columns 11 thru 20, 21 thru 30, and 31 thru 40, respectively, of the source input code. Columns 11, 21, and 31 are reserved for indirect addressing designations (excepting comments and constants) and are otherwise unused. Indirect addressing is permitted in only those operands where specifically so stated in the macro descriptions of sections D and F. Except in certain obvious cases, it is expected that labels will be coded in each operand field of the source input code.

If a label is present on a source input statement, it will be present in the label field of the first assembly language statement generated by that macro; its value is then determined in the normal fashion by the Assembler. This ensures that when transferring program control within a report program, a user need only specify (as the operand of his "jump") the label of the desired transfer point.

The label field is five characters, of which only the first three are used--the fourth and fifth are ignored. Thus AGE and AGENT are both interpreted as AGE; COL 7 and COL 8 are both interpreted as COL. Rules for construction of labels are the same as those for the 1005 Assembly System.

Comments are normally specified by using a comment source input card, but alternatively may be coded in columns 62 thru 80 of any source input card. Comment source cards are retained throughout the assembly process; comments "beyond" column 61 are lost during Pass I of the Assembly. If a label is present on a comment source card, its value will be the address of the next available location of memory, as determined by the 1005 Assembly Language processor. This feature allows the definition of more than one label at any processing step.

The increment fields (columns 17-20, 27-30, and 37-40) should be coded with great care. Incrementation is always counted with respect to the "left-hand" value(MSL) of a label, and is not normally required in an operand, except for the TEST CHARACTER and MOVE CHARACTER macros. In the macro descriptions of sections D, E, and F, whether or not an operand may be incremented is indicated for each operand.

System references, as used in this manual, are source input operands in any of the following six forms:

(1)	¤nnn	decimal address
(2)	#aabb	octal representation of any pair of characters
(3)	$RRCCBk	row/column/bank (decimal)
(4)	RC	row/column/bank (internal machine format)
(5)	#yy	system switch
(6)	+LABEL	"right-hand" value of LABEL (the LSL)

Each of these is fully described in the UNIVAC 1005 Assembly Language manual. Listings of System Labels and Switches appear in Appendices I and II to this manual. System References are permitted as operands only where specifically so stated in the macro descriptions of sections D, E, and F. When System References are not permitted, an operand must be either a Programmer-defined label or a System Label.

With each macro description in sections D, E, and F, is a summary table of operand characteristics for each required operand. The three columns in the table refer respectively to:

(1) IA: indirect addressing to define the operand

(2) SR: system references in addition to labels coded as the operand

(3) INC: YES means the increment feature is permitted and NO means it is not.

The normal application of a macro requires neither the use nor the knowledge of the table's contents; the information is useful for advanced applications.

C. SPECIFICATION OF FIELDS AND DATA

1. Internal

a. Constants

Constants are specified by furnishing the name, the length, and the content of each, on a source statement with operation mnemonic "DC."

The name of the constant must satisfy the rules for constructing labels, and is coded in the line label field. The length of the constant does not include the character, if any, used to furnish a sign for the constant. If the sign character is not furnished, it is assumed to be a plus. The length of a negative constant must not exceed 25 characters.

If a constant of more than 44 characters is desired, the excess of 44 is coded on the next sequential source statement with an operation mnemonic of "comma" (,). Additional characters beyond 88 are coded on additional "comma-cards" to a maximum of 961 characters (22 cards including the DC). The "comma-cards" may have a name of their own coded in their label fields, whether or

not the entire constant has a name; the name on a "comma-card" refers to the characters on that card only, but the name on a "DC-card" refers to the entire constant.

Constants may be defined anywhere within a program without interfering with program sequence control; they are not loaded into the instruction area. Each constant may consist of any characters in the character set including blanks and algebraic signs (the algebraic sign of the constant is not considered as one of its characters).

DEFINE CONSTANT (DC)

Function:

Enter a constant into U1005 storage.

Where:

Operation = a two character mnemonic operation code (DC)

Operand 1

Label A = the number of characters in the constant, excluding sign.

IA = the sign of the constant

INC = the characters of the constant

Operand 2 - additional constant characters extending to column 61. Negative constants extend only to column 42.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Enter the constant + 7
(2) Enter the constant - 10
(3) Enter an alphabetic constant.

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	±	18 20		OPERAND 2 IA	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38 40	
S,E,V,E,N	D,C			1	+	7										
M,I,N,U,S	D,C			2	-	1,0										
M,E,S,S,A,G	D,C			2,9		M,O,V	E	,T,H,I,S		T,O		$	P,R,	A,N	D	,P,R,I,N,T,

CONTINUE CONSTANT (comma - ,)

Function:

Continue a constant that overflows from a previous Define Constant or Continue Constant instruction.

Where:

Operation = a one character operation code (,)

Operands 1 and 2 = consecutive character positions, beginning at column 18, and ending at column 61.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Enter the first 44 characters of a 132 character constant.
(2) Enter the next 44 characters of the constant.
(3) Enter the last 44 characters of the constant.

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	+ −	18 20	OPERAND 2 IA *	LABEL B 22 24	+ −	28 30	LABEL C 32 34	+ −	38 40	
L,O,N,G,	D,C	, ,		1,3,2	,	T,H,I	S	, C, O	N, S	T	A, N, T		I, N, C	L, U	D	E, S	T, H, E
, , ,	,	, ,		, , ,	,	C, O, L	U	M, N, S	, A	T	, T, H	E	, R, I	G, H	T	, U, P	, T, O
T,H,I,R,D	,	, ,		, , ,	,	A, N, D		I, N, C	L, U	D	I, N, G		C, O, L	U, M	N	, 6, 1	, ,

b. Work Areas

Work areas are specified by furnishing the name and length of each, on a source statement with operation mnemonic "DA."

The name of the area is coded in the label field and must satisfy the rules for constructing labels. Maximum area size is 961 characters.

Work areas may be defined anywhere within a program without interfering with program sequence control; they are not reserved in the instruction area.

Work areas are not automatically cleared when the object program is loaded.

DEFINE WORK AREA (DA)

Function:

Define a work area.

Where:

Operation = a two character mnemonic operation code (DA).

Operand 1

Label A = the number of character positions required in the work area.

Operand 2 = not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Define an eight character work area named TEMP.
(2) Define a 12-character work area named T2.
(3) Define an 80 character work area named CARD.

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	+ −	18 20	OPERAND 2 IA *	LABEL B 22 24	+ −	28 30	LABEL C 32 34	+ −	38 40
T,E,M,P,	D,A			8									
T,2,	D,A			1,2									
C,A,R,D,	D,A			8,0									

c. Accumulators.

Accumulators are specified by furnishing the name and length of each, on a source statement with operation mnemonic "DA."

The name of the accumulator is coded in the label field and must satisfy the rules for constructing labels. The accumulator may not exceed 31 characters in length. Accumulators may be defined anywhere within a program without interfering with program sequence control; they are not reserved in the instruction area.

Accumulators are not automatically set to zero when the object program is loaded.

DEFINE ACCUMULATOR (DA)

Function:

Define an accumulator.

Where:

Operation = a two character mnemonic operation code (DA)

Operand 1

Label A = the number of positions required in the accumulator.

Operand 2 = not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Define a 19 digit accumulator named A19.
(2) Define a 6 digit accumulator named A2.

LABEL 1 3	OP 6 7	OPERAND 1 IA *	LABEL A 12 14	+ −	18 20	OPERAND 2 IA *	LABEL B 22 24	+ −	28 30	LABEL C 32 34	+ −	38 40
A,1,9	D,A		1,9									
A,2,	D,A		6									

d. Edit Masks

Edit masks are specified by furnishing the name, the length, and the content of each, on a source statement with operation mnemonic "DC".

The name of the mask is coded in the label field and must satisfy the rules for constructing labels. A mask must not exceed 31 characters in length. If a field to be edited is larger than 31 characters; it must be edited in segments not exceeding 31 characters.

Edit masks may be defined anywhere within a program without interfering with program sequence control; they are not loaded into the instruction area.

For rules governing the use and definition of edit masks, see Appendix IV.

DEFINE EDIT MASK (DC)

Function:

Define an Edit Maks.

Where:

Operation = a two character mnemonic operation code (DC)

Operation 1

Label A = number of positions in the mask.

Operand 2

Label B = The characters of the mask beginning in column 18.

Label C = Consecutive positions containing the overflow characters from Label B.

Examples:

(1) Edit a 7-digit field into a 16-character dollars and cents field suppressing leading zeros and commas.

(2) Same as above but insert asterisks for suppressed characters.

LABEL	OP		OPERAND 1				OPERAND 2									
1 3	6 7		IA	LABEL A 12 14	±	18 20	IA	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38		40
T,H,O,U,	D,C			1 6		T,O,T	A	L, ,$	,		•					
T,H,O,U,	D,C			1,6,		T,O,T	A	L, ,$	,		•					

2. Input (Card)

An input card file is described by a set of source statements which must be supplied to the Report Program Generator in a group. Each distinct card file requires its own group, and any number of card files is permitted.

The first card of each group is a "DA" whose first (and only) operand is "$R1." (See next example.) A line label is not permitted; references to the entire card input area are made through use of the label "$R1" which is the system label of the card input area.

The remaining cards of the group are field definitions. A field definition is a source statement with (a) operation mnemonic "dash" (-), (b) the name of the field coded in the label field, and (c) decimal numbers coded in Labels A and B. Operand two is the length (number of characters) of the field being defined; Operand one is the column number of the "right-most" character of the field as it appears in the card. Every field of the input card must have a name. A field name may appear in the descriptions of more than one input file if the respective fields agree in position and length.

Every column of the input card file need not appear in a defined field.

DEFINE INPUT FIELD (-)

Function:

Define a field in the input file.

Where:

Operation = a one character operation code (-).

Operand 1

Label A = The number of the last card column in the field. The number entered must not exceed 80.

Operand 2

Label B = The number of card columns in the field. This number must not be higher than the number entered in Label A.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) TYPE in column 1 (length 1)
(2) DATE in columns 2 to 6 (length 5)
(3) AGENT in columns 7 to 10 (length 4)
(4) AMT in columns 11 to 18 (length 8)
(5) INFO in columns 1 to 18 (length 18)
(6) ITEM in columns 51 to 80 (length 30)
(7) XYZ in columns 1 to 80 (length 80)

NOTE: "XYZ" may be used as the "name" of the card image area, as an alternative to using "$R1".

LABEL 1 3	OP 6 7	OPERAND 1 IA *	LABEL A 12 14	+/-	18 20	OPERAND 2 IA *	LABEL B 22 24	+/-	28 30	LABEL C 32 34	+/-	38 40
	DA		$R1									
TYPE	-		1				1					
DATE	-		6				5					
AGENT	-		10				4					
AMT	-		18				8					
INFO	-		18				18					
ITEM	-		80				30					
XYZ	-		80				80					

3. Output

a. Printing

(1) Detail Lines

Each card of the input file(s) may be printed as a "detail line" by transferring the contents of the card input area, "$R1," to the leftmost 80 positions of the print output area, "$8Ø," and specifying a print operation. The MVALF and PRINT macros, which accomplish this action, are described in sections D, 2, a and E, 2. To print a detail line requires the following coding:

LABEL	OP		OPERAND 1				OPERAND 2					
1 3	6 7		IA	LABEL A 12 14	+ −	18 20	IA	LABEL B 22 24	+ −	28 30	LABEL C 32 34	+ − 38 40
(optional)	M,V	A,L,F		$,R,1				$,8,0				
(optional)	P,R	I,N,T										

In general, programmed clearing of the print output area prior to using is not necessary, the PRINT operation automatically clears all 132 positions to spaces.

(2) Nondetail Lines

A nondetail line is described by a set of source statements which must be supplied to the Report Program Generator in a group. Each distinct nondetail line requires its own group, and any number of nondetail lines is permitted.

The first card of each group is a "DA" whose first (and only) operand is "$PR." (See next example.) A line label is not permitted; references to the entire print area are made through use of the label "$PR," which is the system label of the print output area.

The remaining cards of the group are field definitions. A field definition is a source statement with: (a) operation mnemonic "dash" (-_, (b) the name of the field coded in the label field, and (c) decimal numbers coded in operands one and two. Operand two is the length (number of characters) of the field being defined; operand one is the print position number (from 1 to 132) of the "rightmost" character of the field. Every field must have a name. A field name may appear in the descriptions of more than one nondetail line if the respective fields agree in position and length.

When a line has been printed, the entire contents of the print output area, "$PR," will automatically be

cleared to blanks; information required following the printing must be specifically "saved" by moving it to other areas.

Constants that are to appear in the printed line must be transferred to the appropriate field each instance of printing, and must be defined as constants through use of the "DC" macro.

The sum of the lengths of the fields specified in the field definitions of any one group (excluding field overlapping) must not exceed 132, but may be any smaller number. In general, only the first line printed during execution need specify the contents of all 132 positions; automatic clearing of the print output area forces all otherwise unspecified print positions to be blank thereafter.

DEFINE PRINT FIELD (-)

Function:

Define a field in the print area.

Where:

Operation = a one character operation code (-).

Operand 1

Label A = The "rightmost" position of the field. The number entered must not exceed 132.

Operand 2

Label B = The number of characters in the field. This number must not be higher than the number entered in Label A.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) 15 positions for HEADR constant in 1 to 15
(2) 5 positions for DATE field in 16 to 20
(3) 20 positions for blanks (automatic if printing occurred previously)
(4) 25 positions for PROD field in 41 to 65
(5) 20 positions for blanks (automatic if printing occurred previously)
(6) 6 positions for CODER field in 86 to 91
(7) 41 positions for blanks (automatic if printing occurred previously)

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	D,A			$ R R									
H,E,A,D,R	-			1,5				1,5					
D,A,T,E	-			2,0				5					
P,R,O,J				6,5				2,5					
C,O,D,E,R				9,1				6					

(3) Printing Summary Cards

Each summary card may be printed by transferring the contents of the punch output area, "$P1,! to the left-most 80 positions of the print output area, "$8Ø," and specifying a print operation. Printing must occur before punching for this case, due to the automatic clearing of the punch output prea following the actual punching. To print a summary card requires the following code:

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	±	18 20	OPERAND 2 IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
(optional)	M,V,A,L,F		$,P,1				$,8,Ø					
(optional)	P,R,I,N,T											

b. Punching

(1) Detail Reproducing

Each card of the input file(s) may be punched as part of the output file by transferring the contents of the card input area, "$R1," to the card output area, "$P1," and specifying a punch operation. The MVALF and PUNCH macros, which accomplish this action, are described in sections (D2a) and (E5). To punch a detail line requires the following coding:

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	±	18 20	OPERAND 2 IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
(optional)	M,V,A,L,F		$,R,1				$,P,1					
(optional)	P,U,N,C,H											

In general, programmed clearing of the punch output area prior to using is not necessary, as the PUNCH operation automatically clears all 80 positions to spaces.

(2) Nondetail (Summary) Punching

A nondetail (summary) card is described by a set of source statements which must be supplied to the Report Program Generator in a group. Each distinct nondetail card requires its own group, and any number of nondetail cards is permitted.

The first card of each group is a "DA" whose first (and only) operand is "$P1." (See next example.) A label is not permitted; references to the entire card output area are made through use of the label "$P1," which is the system label of the card output area.

The remaining cards of the group are field definitions. A field definition is a source statement with (a) operation mnemonic "dash" (-), (b) the name of the field coded in the label field, and (c) decimal numbers coded in Labels A and B. Label B is the length (number of characters) of the field being defined; Label A is the column number of the "rightmost" character of the field as it will appear in the punched card. Every field must have a name, but not every column of the output card need appear in a defined field. A field name may appear in the descriptions of more than one output file if the respective fields agree in position and length.

DEFINE PUNCH FIELD (-)

Function:

Define a field in an output card.

Where:

Operation = A one character operation code (-).

Operand 1

Label A = The "rightmost" column of the field. The number entered must not exceed 80.

Operand 2

Label B = The number of characters in the field. This number must not be higher than the number entered in Label A.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) 18 positions for SLSMN field in positions 1 to 18
(2) 3 positions for BRNCH field in positions 20 to 22
(3) 11 positions for YRVLM field in positions 30 to 40
(4) 10 positions for NET field in positions 50 to 59
(5) 7 positions for COMM field in positions 64 to 70
(6) 2 positions for CON constant in positions 79 to 80

LABEL	OP		OPERAND 1					OPERAND 2							
1 3	6 7		IA *	LABEL A 12 14	±	18 20		IA *	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38 40
	DA			$P1											
SLSMN	-			18					18						
BRNCH	-			22					3						
YRVLM	-			40					11						
NET	-			59					10						
COMM	-			70					7						
CON	-			80					2						

D. PROCESSING DATA

1. ARITHMETIC Operations

Five Macro Instructions are provided for arithmetic operations.

ADD (ADD)

Function:

Algebraically add a field or accumulator to a second field or accumulator. Both fields are assumed to be signed.

Notes: (1) The maximum length of each operand is 31 locations. They need not be of the same length.

(2) The contents of Operand 1 are not affected by this instruction. The result is stored in Operand 2.

(3) One of three sign indicators (#AP, #AZ, #AM) will be set to reflect the resulting condition. The indicator set will remain set until the next arithmetic or round instruction is given.

(4) Arithmetic overflow will cause indicator #AF to be set.

Where:

Operation = a mnemonic operation code (ADD).

Operand 1

Label A = The label address of the first field or accumulator.

Operand 2

Label B = The label address of the second field or accumulator.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	YES
Operand 2 - Label B	YES	NO	YES
Label C	NO	NO	NO

Examples:

(1) Add field A to accumulator 1
(2) Add field TAX to field DEDCT
(3) Add accumulator 3 to accumulator 5

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	±	18 20	OPERAND 2 IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	A,D	D		A				A,1					
	A,D	D		T,A,X				D,E,D	C,T				
	A,D	D		A,3				A,5					

SUBTRACT (SUB)

Function:

Algebraically subtract one field or accumulator from a second field or accumulator. Both fields are assumed to be signed.

Notes: (1) The maximum length of each operand is 31 locations. They need not be of the same length.

(2) The contents of Operand 1 are not affected by this instruction. The result is stored in Operand 2.

(3) One of three sign indicators (#AP, #A2, #AM) will be set to reflect the resulting condition. The indicator set will remain set until the next arithmetic or round instruction is given.

Where:

Operation = a mnemonic operation code (SUB).

Operand 1

Label A = The label address of the first field or accumulator.

Operand 2

Label B = The label address of the second field or accumulator.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	YES
Operand 2 - Label B	YES	NO	YES
Label C	NO	NO	NO

Examples:

(1) Subtract NET from GROSS.
(2) Subtract accumulator 2 from PAY.
(3) Subtract ABC from accumulator 6.

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	± 18 20	OPERAND 2 IA	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	SUB		NET			GROSS			
	SUB		A2			PAY			
	SUB		ABC			A6			

MULTIPLY (MPY)

Function:

Multiply a field or accumulator by a second field or accumulator by a second field or accumulator, storing the result in a third area.

Notes: (1) The signs of both operands are ignored and assumed to be positive.

(2) The contents of Operands 1 and 2 (label B) will will not be disturbed, unless overlapped by the third Operand (Label C).

Where:

Operation = a mnemonic operation code (MPY).

Operand 1

Label A = The label address of a four (4) digit multiplicand.

Operand 2

Label B = The label address of a six (6) digit multiplier.

Label C = The label address of a ten (10) digit product area.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	YES	YES
Label C	NO	NO	NO

Examples: (1) Multiply A

(1) Multiply A by B and store the result in C.
(2) Multiply PAY by RATE and store the result in accumulator 4.

LABEL	OP		OPERAND 1				OPERAND 2					
		IA	LABEL A	±		IA	LABEL B	±		LABEL C	±	
1 3	6 7	*	12 14		18 20	*	22 24		28 30	32 34		38 40
	MPY		A				B			C		
	MPY		PAY				RATE			A4		

MULTIPLY LONG (MPYL)

Function:

Multiply a field or accumulator by a second field or accumulator, storing the result in a third area.

Notes: (1) The signs of both operands are ignored and assumed to be positive.

(2) The contents of Operands 1 and 2 (label B) will not be disturbed, unless overlapped by the third Operand (Label C).

Where:

Operation = a mnemonic operation code (MPYL).

Operand 1

Label A = The label address of a nine (9) digit multiplicand.

Operand 2

Label B = The label address of an eleven (11) digit multiplier.

Label C = The label address of a twenty (20) digit product area.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	YES	YES
Label C	NO	NO	NO

Examples:

(1) Multiply (long) CENTS by RATIO and store the result in LIRA.
(2) Multiply (long) accumulator 2 by AMT and store the result in COST.

LABEL	OP		OPERAND 1					OPERAND 2					
1 3	6 7		IA *	LABEL A 12 14	+ −	18 20		IA *	LABEL B 22 24	+ −	28 30	LABEL C 32 34	+ − 38 40
	MPYL			CENTS					RATIO			LIRA	
	MPYL			A1					AMT			COST	

DIVIDE (DIV)

Function:

Divide a field or accumulator by a second field or accumulator, storing the result in a third area.

Notes: (1) The signs of both operands are ignored and assumed to be positive.

(2) The contents of Operands 1 and 2 (label B) will not be disturbed, unless overlapped by the third Operand (Label C).

Where:

Operation = a mnemonic operation code (DIV).

Operand 1

Label A = The label address of a six (6) digit divisor.

Operand 2

Label B = The label address of an eight (8) digit dividend.

Label C = The label address of an eight digit area, to contain the eight (8) digit quotient.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	YES	YES
Label C	NO	NO	NO

Examples:

(1) Divide TOTAL by WEEKS and store the result in TEMP.

LABEL	OP		OPERAND 1					OPERAND 2							
1 3	6 7		IA	LABEL A 12 14	±	18 20		IA	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38 40
	DIV			WEEKS					TOTAL				TEMP		

2. Internal Data Transfers and Editing

Ten Macro instructions are provided to transfer, edit, and modify data.

a. Data Transfers (alphanumeric and numeric)

MOVE ALPHANUMERIC (MVALF)

Function:

Move an alphanumeric field or accumulator into a second field or accumulator.

Where:

Operation = a mnemonic operation code (MVALF).

Operand 1

Label A = The label-address of the field or accumulator to be moved. The data stored in Operand 1 will not be altered by the instruction.

Operand 2

Label B = The label address of the field or accumulator to receive the data moved. Operand 2 must not include more than 961 storage locations, or more locations than are specified by Operand 1.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	YES	NO	YES
Label C	NO	NO	NO

Examples:

(1) Move accumulator 1 to GROSS.
(2) Move SALES to INCOM.
(3) Move DAY to Accumulator 3.

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	± 18 20	OPERAND 2 IA	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	M,V	A,L,F		A,1			G,R,O,S,S			
	M,V	A,L,F		S,A,L,E	S		I,N,C,O,M			
	M,V	A,L,F		D,A,Y			A,3			

MOVE NUMERIC (MVNUM)

Function:

Move a field or accumulator into a second field or accumulator, deleting all zone and sign bits.

Where:

Operation = a mnemonic operation code (MVNUM).

Operand 1

Label A = The label address of the field or accumulator to be moved. The data stored in Operand 1 will not be altered by the instruction.

Operand 2

Label B = The label address of the field or accumulator to receive the data moved. Operand 2 must not include more than 961 storage locations or more locations than are specified by Operand 1.

Label C = Not used

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	YES	NO	YES
Label C	NO	NO	NO

Examples:

(1) Move the decimal field VALUE to accumulator 1.
(2) Replace NET by the absolute value of NET.
(3) Move the field INPUT to accumulator 2, removing overpunches.

LABEL	OP		OPERAND 1					OPERAND 2					
			IA	LABEL A	±			IA	LABEL B	±		LABEL C	±
1 3	6 7		*	12 14		18	20	*	22 24		28 30	32 34		38 40
	MVNUM			VALUE					A,1				
	MVNUM			NET					NET				
	MVNUM			INPUT					A,2				

b. Data Transfer with Edit Feature

MOVE WITH EDIT (MVEDT)

Function:

Move an alphanumeric field or accumulator to a second field or accumulator, modifying the data transferred by a specified mask.

Where:

Operation = a mnemonic operation code (MVEDT).

Operand 1

Label A = The label address of the field or accumulator to be moved. The data stored in Operand 1 will not be altered by the instruction.

Operand 2

Label B = The label address of the field or accumulator to receive the data moved.

Label C = The label address of the edit mask.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	YES	YES	YES
Operand 2 - Label B	YES	NO	YES
Label C	YES	YES	YES

NOTE: A description of edit masks and editing features is presented in Appendix IV.

Examples:

(1) Move TAX to OUTPT, editing with DOLLR.
(2) Move accumulator 5 to SAVE, editing with ZERO.
(3) Move IN to TEMP, editing with accumulator 4.

LABEL	OP		OPERAND 1						OPERAND 2								
1 3	6 7		IA	LABEL A 12 14		±	18 20		IA	LABEL B 22 24		±	28 30		LABEL C 32 34		±	38 40
	MVE	DT		TAX						OUTPT					DOLLR			
	MV	EDT		A 5						SAVE					ZERO			
	MV	EDT		IN						TEMP					A 4			

c. Filling - Work Areas

FILL AREA (FILL)

Function:

Fill a field or accumulator with a specified character.

Where:

Operation = a mnemonic operation code (FILL)

Operand 1

Label A = The label address of the area to be filled. Operand 1 must not exceed 961 characters in length. Fill begins in the "leftmost" position specified.

Operand 2

Label B = The character with which the area specified by Operand 1 is to be filled. This character is always entered in both columns 22 and 23.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	YES	NO	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Fill TOTAL with zeroes
(2) Fill accumulator 6 with asterisks
(3) Fill all but the two leftmost characters of HEADR with dashes.

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	± 18	20	OPERAND 2 IA	LABEL B 22 24	± 28	30	LABEL C 32 34	± 38	40
	FILL			TOTAL				ØØ					
	FILL			Ab				**					
	FILL			HEADR	+	2		--					

d. Clearing-Work Areas

CLEAR AREA (CLEAR)

Function:

Clear one, two, or three fields or accumulators to spaces.

Where:

Operation = a mnemonic operation code (CLEAR)

Operand 1

Label A = The label address of a field or accumulator to be cleared. The maximum number of characters in this Operand is 961. Clearing begins at the "leftmost" position specified.

Operand 2

Label B = The label address of a second field or accumulator to be cleared. The maximum number of characters in this Operand is 961. Clearing begins at the "leftmost" position specified.

Label C = The label address of a third field or accumulator to be cleared. The maximum number of characters in this Operand is 961. Clearing begins at the "leftmost" position specified.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	YES	NO	YES
Operand 2 - Label B	YES	NO	YES
Label C	YES	NO	YES

Examples:

(1) Clear accumulators 3 and 7 and field MM2
(2) Clear fields SALES, NET, and MONTH
(3) Clear field OUT and all but the four leftmost characters of MASK

LABEL	OP	OPERAND 1				OPERAND 2					
1 3	6 7	IA *	LABEL A 12 14	±	18 20	IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	± 38 40
	CLEAR		A,3				A,7			MM2	
	CLEAR		SALES				NET			MONTH	
	CLEAR		OUT				MASK	+	4		

e. Moving a Single Character

MOVE CHARACTER (MVCHR)

Function:

Move a character, contained in the instruction to a single storage location.

Where:

Operation = A mnemonic operation code (MVCHR)

Operand 1

Label A = The character to be moved. It is coded into columns 12 and 13.

Operand 2

Label B = The label address of the location to receive the specified character. The location specified may be part of a larger field or accumulator.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	YES	YES	YES
Label C	NO	NO	NO

Examples:

(1) Move a 7 to the fourth character of COST
(2) Move a blank to the first character of field B32
(3) Move a - to the last character of accumulator 1.

LABEL	OP		OPERAND 1					OPERAND 2							
			IA	LABEL A		±		IA	LABEL B		±		LABEL C		±	
1 3	6 7		*	12 14			18 20	*	22 24			28 30	32 34			38 40
	MVCHR			7,7					COST		+	3				
	MVCHR								B,3,2							
	MVCHR			-,-					+,A,1							

f. Rounding Arithmetic Results

ROUND (ROUND)

Function:

Round a decimal value by half adjusting in the "rightmost" position of the area specified. This instruction will cause the value five (5) to be added in the rightmost position of the field. The result is then shifted one position to the right, dropping the rounded position and filling the "leftmost" position with a space code.

NOTES: (1) The rounding of negative values should be preceded by the addition of the value - 10.

(2) The ROUND instruction affects the sign indicators.

Where:

Operation = a mnemonic operation code (ROUND)

Operand 1

Label A = The label address of the field or accumulator to be rounded.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Round accumulator 2 one place
(2) Round LEVEL (known to be negative) one place (MTEN contains -10)

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	ROUND			A2									
	ADD			MTEN				LEVEL					
	ROUND			LEVEL									

g. Shifting Arithmetic Results

SHIFT FIELD (SHIFT)

Function:

Shift the contents of a field or accumulator to the right a specified number of positions, filling the opened positions to the left with spaces.

Where:

Operation = a mnemonic operation code (SHIFT)

Operand 1

Label A = The number of positions the field or accumulator is to be shifted. The maximum shift is 961 locations.

Operand 2

Label B = The label address of the field or accumulator to be shifted.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Shift accumulator 5 right 3 places.
(2) Round QOTNT 7 places.

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	± 18	20	OPERAND 2 IA	LABEL B 22 24	± 28	30	LABEL C 32 34	± 38	40
	SHIFT		*	3				A5					
	SHIFT			6				QOTNT					
	ROUND			QOTNT									

h. Transfer of Sign

TRANSFER SIGN (SIGN)

Function:

Transfer the algebraic sign of a field or accumulator to a second field or accumulator.

NOTES: (1) The sign is located in the zone bits of the rightmost character of a field.

(2) Only the sign bits of the receiving field are altered.

Where:

Operation = a mnemonic operation code (SIGN)

Operand 1

Label A = The label address of the field containing the sign to transferred.

Label B = The label address of the field to receive the sign.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Move the sign of YRNET to WKNET
(2) Move the sign of accumulator 2 to SIGN and make accumulator 2 positive.

LABEL	OP		OPERAND 1					OPERAND 2							
1 3	6 7		IA	LABEL A 12 14	±	18 20		IA	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38 40
	SIGN			YRNET					WKNET						
	SIGN			A,2					SIGN						
	MVNUM			A,2					A,2						

i. General Logical Command

EDIT LOGICAL (EL)

Function:

(1) Erase bits of the character specified by Operand 2, Label B, if the corresponding bits of the first (XS3) character in Operand 1, Label A, are zeros. This is an <u>and</u> operation, similar to logical multiplication without carry.

Rules:

$$\begin{aligned} 0 \times 0 &= 0 \\ 0 \times 1 &= 0 \\ 1 \times 0 &= 0 \\ 1 \times 1 &= 1 \end{aligned}$$

(2) Superimpose the bit pattern of the second (XS3) character in Operand 1, Label A, onto the character specified by Operand 2, Label C. This is an <u>or</u> operation, similar to logical addition without carry.

Rules:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + + &= 1 \end{aligned}$$

Where:

Operation = a two character mnemonic operation code (EL).

Operand 1

Label A = The two characters to be used in the edit operation.

NOTE: Label A does not specify a location in this instruction. The two characters represent the bit patterns to be used.

Operand 2

Label B = The label address of the character on which the erasure is to be carried out.

Label C = The label address of the character on which the superimpose is to be carried out.

NOTE: Labels B & C may specify the same storage location.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	NO
Operand 2 - Label B	NO	YES	YES
Label C	NO	YES	YES

Examples:

(1) AND a 4-bit into CODE and OR a Y-bit into PLUS
(2) AND a 4-bit into CODE
(3) OR a Y-bit into PLUS

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	±	18 20		OPERAND 2 IA *	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38 40
	EL			#,Ø,4	2,Ø				C,O,D	E,			P,L,U	S,	
	EL			#,Ø,4	Ø,Ø				C,O,D	E,					
	EL			#,7,7	2,Ø								P,L,U	S,	

E. INPUT/OUTPUT

1. Reading Cards

READ A CARD (READ)

Function:

Read the next card from the input file.

NOTE: Card images are always read into the following storage locations:

80 column - positions 1 - 80
90 column - positions 1 - 45 and 63 - 107

Where:

Operation = a mnemonic operation code (READ).

Operand 1 = Not used.

Operand 2 = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Read the next card from the input file.
(2) Move the 80-col card image to field CARD.
(3) Move columns 41 thru 80 of an 80-col card image to HALF.

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	±	18 20	OPERAND 2 IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	READ											
	MVALF		$R1				CARD					
	MVALF		$R1	+	40		HALF					

2. Printing

PRINT (PRINT)

Function:

Print a line, space the form one line and clear Print Storage. Print Storage is located at positions 161-292.

Where:

Operation = A mnemonic operation code (PRINT).

Operand 1 = Not used.

Operand 2 = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Print the contents of the print area.
(2) Print the contents of the 132-char field HEADR.

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	P,R	I,N,T											
	M,V	A,L,F		H,E,A,D,R				$,P,R					
	P,R	I,N,T											

3. Spacing Forms

SPACE (SPACE)

Function:

Advance the form in the printer one space without printing.

Where:

Operation = A mnemonic operation code (SPACE).

Operand 1 = Not used.

Operand 2 = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Advance the carriage three (blank) lines.

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	S P	A C E											
	S P	A C E											
	S P	A C E											

4. Skipping Forms

SKIP (SKIP)

Function:

Skip the form in the printer to a specified line. The seven (7) code on the Form Control Tape is reserved as the "Home Paper" code.

Where:

Operation = A mnemonic operation code (SKIP).

Operand 1

Label A = The decimal equivalents of the bit configurations on the Form Control Tape. The number is entered into column 12.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Example:

(1) Skip to control tape configuration 5

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	± 18 20	OPERAND 2 IA	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	SKIP		5						

5. Punching Cards

PUNCH (PUNCH)

Function:

Punch the contents of the punch storage area, clearing Punch Storage to spaces.

NOTE: The Punch Storage area is always located in the following locations:

80 column - positions 293-372
90 column - positions 293-337 and 383-427

Where:

Operation = A mnemonic operation code (PUNCH).

Operand 1 = Not used.

Operand 2 = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Example:

(1) Punch a card (Line 1).
(2) Then punch the contents of field SUMRY. (Lines 2 and 3)

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	+ −	18 20	OPERAND 2 IA *	LABEL B 22 24	+ −	28 30	LABEL C 32 34	+ −	38 40
	PUNCH												
	MVALF			SUMRY				$P1					
	PUNCH												

6. General Input/Output Command

GENERAL COMMAND (GC)

Function:

To initiate operation and/or control all input/output devices under the direct control of the UNIVAC 1005.

Where:

Operation = A mnemonic operation code (GC)

Operand 1

Label A = Two of the six (XS3) characters that are used to produce the necessary bit patterns to select and operate the various Input/Output devices.

Operand 2

Label B = Two of the six (XS3) characters that are used to produce the necessary bit patterns to select and operate the various Input/Output devices.

Label C = Two of the six (XS3) characters that are used to produce the necessary bit patterns to select and operate the various Input/Output devices.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	NO
Operand 2 - Label B	NO	YES	NO
Label C	NO	YES	NO

Examples:

(1) Print with double spacing (one blank line following a printed line)
(2) Read and Punch
(3) Read and Print only 90 columns.

LABEL 1 3	OP 6 7	OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	G C		\				-					
	G C		Ø							3		
	G C		[				]					

F. PROGRAM CONTROL

Twenty two macro instructions are available for controlling the flow of processing. They are necessary for starting and halting a program, for setting and testing conditions, for operations with subroutines, and for altering the execution sequence of program.

The program is normally executed in the order in which the processing statements are presented to the Report Program Generator. However, numerous macro instructions are provided to allow the programmer to alter the sequence under a variety of conditions.

A control statement is not required to indicate the end of source input code statements. The Report Program Generator halts when the last input card is read.

1. Program Start

END PROGRAM LOAD (END)

Function:

Terminate loading of the program and begin execution. This instruction is identical to the END directive of the 1005 Assembler and is not part of a loaded program. It has no function during compiling or assembling and does not terminate the reading of source input cards.

Where:

Operation = A mnemonic operation code (END).

Operand 1

Label A = The label address of the first processing instruction to be executed.

Operand 2 = Not used.

Operand Characteristics:

		IA	SR	INC
Operand 1 -	Label A	NO	YES	YES
Operand 2 -	Label B	NO	NO	NO
	Label C	NO	NO	NO

Examples:

(1) Terminate loading and start the program at STEP.

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	± 18 20	OPERAND 2 IA	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	END		STEP						

2. Program Halt

HALT PROCESSING (HALT)

Function:

Stop program execution and light a Halt Indicator. This instruction becomes part of the object program. When the RUN key on the U1005 Console is depressed following a halt, processing will continue at the source statement immediately following the source HALT statement.

Where:

Operation = A mnemonic operation code (HALT).

Operand 1

Label A = The code for the desired System Switch. A listing of System Switches is provided in Appendix II.

Operand 2 = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Halt execution and turn on halt indicator 2. Continue to RESTT if RUN button is pressed.

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	± 18 20	OPERAND 2 IA	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	HALT		#H2						
	GOTO		RESTT						

3. Setting Conditions

SET CONDITION (SET)

Function:

Set or reset one, two or three System Switches.

Where:

Operation = A mnemonic operation code (SET).

Operand 1

Label A = The code for the System Switch to be set or reset. A listing of System Switches is provided in Appendix II.

Operand 2

Label B = The code for a second System Switch to be set or reset.

Label C = The code for a third System Switch to be set or reset.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	NO
Operand 2 - Label B	NO	YES	NO
Label C	NO	YES	NO

Examples:

(1) Set even parity for tape operation.
(2) Set sense switches one and two and servo No. 1.
(3) Reset sense switch one and set servo No. 2.

LABEL 1 3	OP 6 7	OPERAND 1 IA *	LABEL A 12 14	+/-	18 20	OPERAND 2 IA *	LABEL B 22 24	+/-	28 30	LABEL C 32 34	+/-	38 40
	SET		#EP									
	SET		#+2				#+1			#S1		
	SET		#-1				#S2					

4. Sequence Control

a. Testing for Conditions

TEST CONDITION (TEST)

Function:

Test a System Switch for a specified setting. Transfer program sequence control if the condition is present.

Where:

Operation = A mnemonic operation code (TEST).

Operand 1

Label A = The letters "COND."

Operand 2

Label B = The code for the desired System Switch. A listing of System Switches is provided in Appendix II.

Label C = The label address of the next instruction to be executed if the condition is present. If the condition tested is not met, control is transferred to the next instruction in sequence.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	YES	NO
Label C	NO	YES	YES

Examples:

(1) Test for arithmetic overflow; if set go to OVER routine.
(2) Test for and reset parity error and if set go to PARER.
(3) Test for alteration switch No. 2 set and if set go to ON2.

LABEL 1 3	OP 6 7	OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	TEST		COND				#AF			OVER		
	TEST		COND				#PE			PARER		
	TEST		COND				#-2			ON2		

TEST NEGATIVE (TEST)

Function:

Test the contents of a single storage location for a negative sign. A field is identified as negative by the presence of an X bit in the rightmost character position.

Where:

Operation = a mnemonic operation code (TEST)

Operand 1

Label A = The letters "NEG".

Operand 2

Label B = The label address of the character location to be tested.

Label C = The label address of the next instruction to be executed if the location tested is negative. If the location is not negative, control is transferred to the next instruction in sequence.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	YES	YES
Label C	NO	YES	YES

Examples:

(1) If field PRFT is negative, transfer control to BOMB.
(2) If the third character of INPUT is positive, go to MAN.

LABEL	OP	OPERAND 1				OPERAND 2				
		IA	LABEL A	±		IA	LABEL B	±	LABEL C	±
1 3	6 7		12 14		18 20		22 24		28 30 32 34	38 40
	TEST		NEG				+PRFT		BOMB	
	TEST		NEG				INPUT	+	2 NEXT	
	GOTO		MAN							
NEXT										

TEST CHARACTER (TEST)

Function:

Test a storage location for the presence of a specific character.

Where:

Operation = a mnemonic operation code (TEST)

Operand 1

Label A = The specific character for which the test is being made, not the address of the test character. This character is entered into column 13.

Operand 2

Label B = The label address of the storage location being tested.

Label C = The label address of the next instruction to be executed if the location tested contains the character specified in Operand 1. If the character is not present, control is transferred to the next instruction in sequence.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	NO
Operand 2 - Label B	YES	YES	YES
Label C	NO	YES	YES

Example:

(1) If the last character of UNIT is not a 7, go to NO7.
(2) If the second character of UNIT is *, go to SPECL.

LABEL	OP	OPERAND 1			OPERAND 2			
1 3	6 7	IA	LABEL A 12 14	± 18 20	IA	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	TEST		77			+UNIT		NEXT	
	GOTO		NO7						
NEXT	TEST		**			UNIT	+ 1	SPECL	

b. Comparing for Conditions

COMPARE ALPHANUMERIC (COMPA)

Function:

Perform an alphanumeric comparison on two fields or accumulators. This comparison is made on a match/non-match basis. Either the "equal" or "not-equal" indicator is set as a result of this instruction. These indicators remain set until the next compare instruction. They may be tested by the IFEQ and IFNE instructions.

Where:

Operation = A mnemonic operation code (COMPA).

Operand 1

Label A = The label address of a field or accumulator.

Operand 2

Label B = The label address of a field or accumulator to be compared with Operand 1.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	YES	NO	YES
Label C	NO	NO	NO

Examples:

(1) Compare INPUT with NAME
(2) Compare accumulator 1 with INPUT
(3) Compare the "rightmost" 6 characters of accumulator 2 (9 characters) with SCALE

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	± 18	20	OPERAND 2 IA *	LABEL B 22 24	± 28	30	LABEL C 32 34	± 38	40
	C O M P A			I N P U T				N A M E					
	C O M P A			A 1				I N P U T					
	C O M P A			S C A L E				A 2	+	3			

TRANSFER IF EQUAL (IFEQ)

Function:

If a previous comparison set the EQUAL indicator, transfer program control.

NOTES: (1) The IFEQ instruction can be used in conjunction with any of the compare instructions.

(2) Indicator settings are not affected by the IFEQ instructions.

Where:

Operation = A mnemonic operation code (IFEQ).

Operand 1

Label A = The label address of the next instruction to be executed, if the EQUAL indicator is set. If the EQUAL indicator is not set, the next instruction in sequence is executed.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) If the previous compare set the EQUAL indicator, transfer program control to step EQU.

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	± 18 20	OPERAND 2 IA *	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	I,F	E,Q,		E,Q,U						

TRANSFER IF NOT EQUAL (IFNE)

Function:

If a previous comparison set the NOT-EQUAL indicator, transfer program control.

NOTE: Indicator settings are not affected by the IFNE instruction.

Where:

Operation = A mnemonic operation code (IFNE).

Operand 1

Label A = The label address of the next instruction to be executed, if the NOT-EQUAL indicator is set. If the indicator is not set, the next instruction in sequence is executed.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) If the previous compare set the NOT-EQUAL indicator, transfer program control to PHASE.

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	± 18 20	OPERAND 2 IA *	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	I F	N E		P H A S E						

COMPARE NUMERIC (COMPN)

Function:

Perform an algebraic comparison on two numeric fields or accumulators. This instruction always results in the setting of one of three indicators. These indicators are; (1) EQUAL, (2) LESS-THAN, and (3) GREATER-THAN. If either the LESS-THAN or GREATER-THAN indicator is set, the NOT-EQUAL indicator will also be set. All of these indicators will remain set until the next compare instruction is given. They may be tested by any "IF" instruction.

Where:

Operation = A mnemonic operation code (COMPN).

Operand 1

Label A = The label address of a numeric field or accumulator.

Operand 2

Label B = The label address of the field or accumulator to be compared with Operand 1.

NOTES: (1) Operands 1 and 2 must be of equal length.

(2) Resulting indicator settings will be made with respect to Operand 1; e.g., if Operand 1 is greater, the GREATER-THAN indicator will be set.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	YES	NO	YES
Label C	NO	NO	NO

Examples:

(1) Compare SUM with TOTAL.
(2) Compare accumulator 3 with SUM.
(3) Compare HYTE with all but the first digit of TOP.

LABEL	OP	OPERAND 1				OPERAND 2				
1 3	6 7	IA	LABEL A 12 14	±	18 20	IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	COMPN	*	SUM			*	TOTAL					
	COMPN		A3				SUM					
	COMPN		HYTE				TOP	+	1			

TRANSFER IF LESS THAN (IFLT)

Function:

If the previous comparison set the LESS-THAN indicator, transfer program control.

NOTES: (1) Indicator settings are not affected by the IFLT instruction.

(2) IFLT instruction may be used only in conjunction with the COMPN and COMPM instructions.

Where:

Operation = A mnemonic operation code (IFLT).

Operand 1

Label A = The label address of the next instruction to be executed, if the LESS THAN indicator is set. If the indicator is not set, the next instruction in sequence is executed.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Example:

(1) If the previous compare set the LESS-THAN indicator, transfer program control to step ENUF.

LABEL	OP		OPERAND 1					OPERAND 2								
			IA	LABEL A		±		IA	LABEL B		±		LABEL C		±	
1 3	6 7		*	12 14			18 20	*	22 24			28 30	32 34			38 40
	I F	L T		E N U F												

TRANSFER IF GREATER THAN (IFGT)

Function:

If the previous comparison set the LESS-THAN indicator, transfer program control.

NOTES: (1) Indicator settings are not affected by the IFGT instruction.

(2) The IFGT instruction may be used only in conjunction with the COMPN and COMPM instructions.

Where:

Operation = A mnemonic operation code (IFGT).

Operand 1

Label A = The label address of the next instruction to be executed, if the GREATER-THAN indicator is set. If the indicator is not set, the next instruction in sequence is executed.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 = Label A	NO	YES	YES
Operand 2 = Label B	NO	NO	NO
Label C	NO	NO	NO

Example:

(1) If the previous compare set the GREATER THAN indicator, transfer control to step PRICE.

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	±	18 20	IA	OPERAND 2 LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	I F G T			P R I C E									

COMPARE MAGNITUDE (COMPM)

Function:

Perform an absolute magnitude compare on two numeric fields or accumulators. This instruction will always result in the setting of one of three indicators. These indicators are; (1) EQUAL, (2) LESS-THAN and (3) GREATER-THAN. If either the LESS-THAN or the GREATER-THAN indicator is set, the NOT-EQUAL indicator will also be set. These indicator settings will not be altered until the next compare instruction is given. All of these indicators may be tested by any "IF" instruction.

Where:

Operation = A mnemonic operation code (COMPM).

Operand 1

Label A = The label address of a field or accumulator.

Operand 2

Label B = The label address of a field or accumulator to be compared to Operand 1.

NOTE: Indicator settings are made with respect to Operand 1; e. g., if Operand 1 is greater than Operand 2, the GREATER-THAN indicator will be set.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	YES	NO	YES
Label C	NO	NO	NO

Examples:

(1) Compare the magnitudes of accumulator 2 and AMT.
(2) Compare the magnitudes of TIME and accumulator 3.
(3) Compare the magnitudes of the field FIFTY and all but the left three characters of AMT.

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	±	18 20	OPERAND 2 IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	COMPM		A2				AMT					
	COMPM		TIME				A3					
	COMPM		FIFTY				AMT	+	3			

c. Explicit Sequence Change

TRANSFER CONTROL (GOTO)

Function:

Unconditionally transfer program control to the instruction specified by Operand 1.

Where:

Operation = A mnemonic operation code (GOTO).

Operand 1

Label A = The label address of the next instruction to be executed.

Operand 2

Label A = Not used.

Label B = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Example:

(1) Transfer program control to program step TAX.

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	± 18	20	OPERAND 2 IA	LABEL B 22 24	± 28	30	LABEL C 32 34	± 38	40
	GO	TO		TAX									

d. Implicit Sequence Change (Level Breaks)

Level break operations (LEVLA, LEVLN, LEVLM) are used to conditionally transfer program control to a specified subroutine when consecutive values of a field in an input file differ. The "condition" is that the ALLOW switch is ON.

The ALLOW switch is an internal switch set to either ON or OFF, and is set to ON automatically when an object program is loaded. Whenever a level break occurs (that is, consecutive values of a field differ), the ALLOW switch is set to OFF before program control is transferred to the appropriate subroutine. The only means of returning the switch to the ON position is the use of the ALLOW BREAK macro instruction.

The first time "thru" the level break operation, no break can occur; processing is limited to "saving" the first value of the designated field. The second time and thereafter, testing for differing consecutive values occurs and a break is possible. In all cases, the current value of the tested field is "saved" for the next comparison.

When a level break occurs and program control transfers to the specified subroutine, the EXIT operation of that subroutine is automatically set to return program control to the operation sequentially following the level break operation (LEVLA, LEVLN, or LEVLM).

Level break operations should be coded in order of decreasing priority of fields tested. Appendix 3 is an example of the normal scheme. An ALLOW BREAK operation precedes the first level break operation, but there may be intervening operations between any two level break operations or between any LEVL operation and ALLOW BREAK. More than one ALLOW BREAK may appear in a program.

LEVEL BREAK ALPHANUMERIC (LEVLA)

Function:

Transfer program control if a control break (level break) occurs on an alphanumeric field. Comparison is based on all six bits of each character.

Where:

Operation = A mnemonic operation code (LEVLA).

Operand 1

Label A = The label address of a field to be tested for a level break.

Operand 2

Label B = The label address of a BEGIN statement of a subroutine.

Label C = The number of characters in the field specified by Operand 1.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Alphanumeric level breaks in FIELD (8 characters) are to be processed by COST.
(2) Alphanumeric level breaks in the seven "rightmost" characters of FIELD are to be processed by TIME.

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	±	18 20	OPERAND 2 IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	LEVLA		FIELD				COST			8		
	LEVLA		FIELD	+	1		TIME			7		

LEVEL BREAK NUMERIC (LEVLN)

Function:

Transfer Program Control if a control break (level break) occurs on a numeric field. The comparison of fields is algebraic. All zone bits are ignored, except the sign bit in the "rightmost" position.

Where:

Operation = A mnemonic operation code (LEVLN).

Operand 1

Label A = The label address of a field to be tested for a level break.

Operand 2

Label B = The label address of a BEGIN statement of a subroutine.

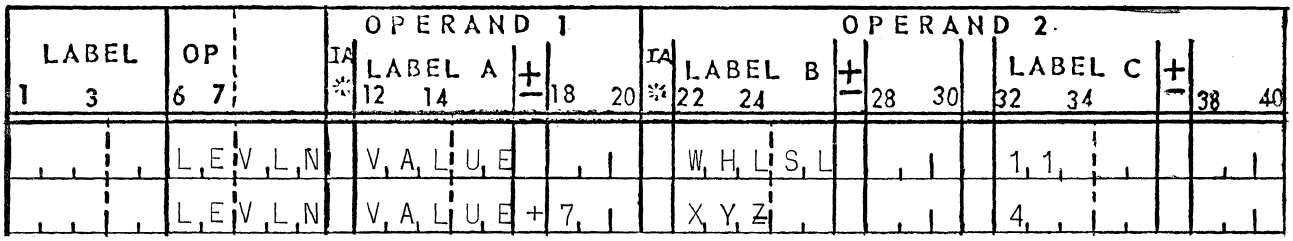
Label C = The number of characters in the field specified by Operand 1.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Numeric level breaks in VALUE (11 characters) are to be processed by WHLSL.
(2) Numeric level breaks in the four "rightmost" characters of VALUE are to be processed by XYZ.



LEVEL BREAK MAGNITUDE (LEVLM)

Function:

Transfer program control if a control break (level break) occurs on a numeric field. The comparison of fields is numeric. All signs and zone bits are ignored.

Where:

Operation = A mnemonic operation code (LEVLM).

Operand 1

Label A = The label address of a field to be tested for a level break.

Operand 2

Label B = The label address of a BEGIN statement of a subroutine.

Label C = The number of characters in the field specified by Operand 1.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Magnitude level breaks in A3 (12 characters) are to be processed by TOTAL.
(2) Magnitude level breaks in the two "rightmost" characters of A3 are to be processed by MAJOR.

LABEL	OP		OPERAND 1					OPERAND 2							
1 3	6 7		IA *	LABEL A 12 14	±	18 20		IA *	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38 40
	LEVLM			A3					TOTAL				12		
	LEVLM			A3	+	10			MAJOR				2		

ALLOW BREAK (ALLOW)

Function:

Turn on the internal switch that permits the occurrence of level breaks.

Where:

Operation = A mnemonic operation code (ALLOW).

Operand 1

Label A = The letters "BREAK".

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Example:

(1) Turn on the internal switch which permits the occurrence of level breaks.

LABEL	OP		OPERAND 1					OPERAND 2					
			IA	LABEL A	+			IA	LABEL B	+		LABEL C	+
1 3	6 7		*	12 14	−	18	20	*	22 24	−	28 30	32 34	− 38 40
	ALLOW			BREAK									

5. Loop Control

LOOP (LOOP)

Function:

Repeat the execution of a group of instructions a specified number of times.

NOTES: (1) LOOP is normally the final operation in the repeated group. If LOOP is the first operation of the group the value specifying the number of times the group is to be executed, must be one more than the number of executions desired.

(2) When the execution has been repeated the desired number of times, control is transferred to the next instruction in sequence, and the loop operation is automatically reset.

Where:

Operation = A mnemonic operation code (LOOP).

Operand 1

Label A = A two digit number (Ø1 to 99) representing the number of times the loop is to be executed.

Operand 2

Label B = The label address of the first instruction of the group.

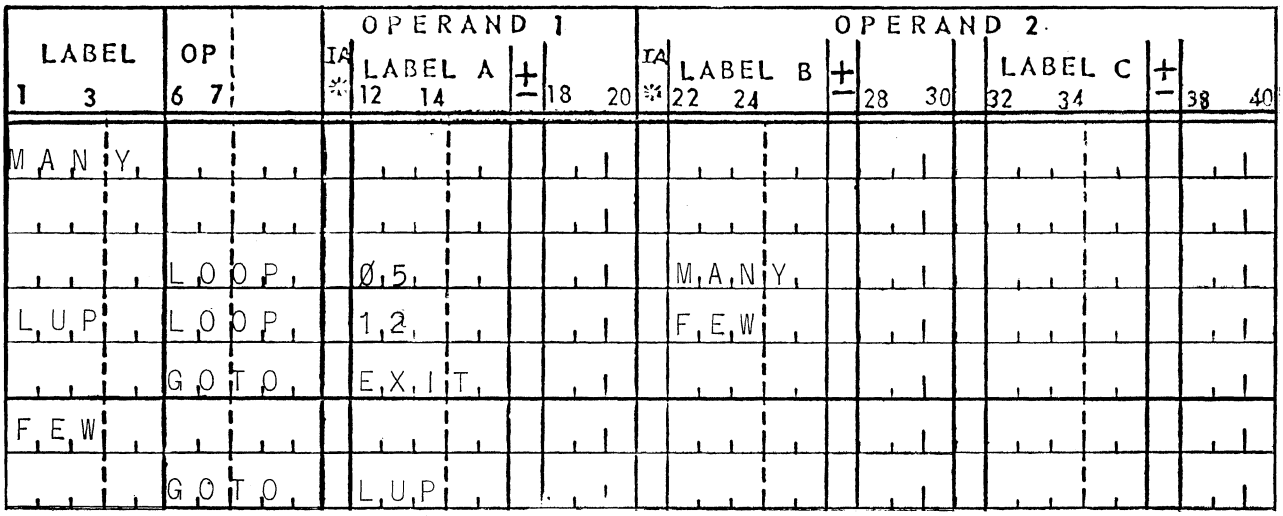
Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	YES	YES
Label C	NO	NO	NO

Examples:

(1) Repeat the group of operations starting with MANY 5 times from below.
(2) Repeat the group of operations starting with FEW 11 times (from above).



LABEL 1 3	OP 6 7	OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
MANY												
	LOOP		Ø5				MANY					
LUP	LOOP		12				FEW					
	GOTO		EXIT									
FEW												
	GOTO		LUP									

6. Subroutines

A subroutine, as here defined, is a group of operations coded sequentially, whose first operation is BEGIN (a labelled operation) and whose last operation is EXIT (also labelled). A subroutine is characterized by the following property: When Operand 2 of any "LEVL" macro is the label of a BEGIN, the return exit addesss is automatically inserted into the corresponding EXIT whenever a level break occurs to that BEGIN.

In all other cases of transferring control to a BEGIN (by GOTO, TEST, "IF", SBRT, and LOOP), the address in EXIT can and must be set by the programmer. This is most easily accomplished through use of the SBRT operation, described below. "Nesting" of subroutines is not allowed between any pair of BEGIN's at least one EXIT must appear.

BEGIN A SUBROUTINE (BEGIN)

Function:

Define the beginning of a subroutine. Begin must be labelled. When executed as a result of a LEVL instruction it sets up the return address in the corresponding exit.

Where:

Operation = A mnemonic operation code (BEGIN).

Operand 1

Label A = Not used.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Example:

(1) Begin a subroutine.

LABEL 1 3	OP 6 7	OPERAND 1 IA	LABEL A 12 14	± 18 20	OPERAND 2 IA	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
	B,E,G,I,N								

EXIT FROM A SUBROUTINE (EXIT)

Function:

Provide a variable GOTO of which the operand is set by the program during execution.

NOTE: The exit address can be specified by the programmer in either of two ways, each of which requires that the EXIT operation be labelled. They are; (1) SBRT (described below) and, (2) the use of an alphanumeric move from the operand of a DI to the modified label (see Example 2).

Where:

Operation = A mnemonic operation code (EXIT).

Operand 1

Label A = Not used.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Set up a variable exit.
(2) Move the address of RTRN to the EXIT of Example (1).

LABEL 1 3	OP 6 7		OPERAND 1 IA *	LABEL A 12 14	±	18 20	OPERAND 2 IA *	LABEL B 22 24	±	28 30		LABEL C 32 34	±	38 40
LABEL	EXIT													
	MVALF			NAME				LABEL	+	3				
NAME	DI			RTRN										

EXECUTE A SUBROUTINE (SBRT)

Function:

Transfer program control to a subroutine and set an exit address.

Where:

Operation = A mnemonic operation code (SBRT).

Operand 1

Label A = The label address of the first step of the subroutine.

Label B = The label address of an EXIT or GOTO operation, which is to contain the return address. The address of this operand is not incremented.

Label C = The address to which the subroutine will return when it executes the EXIT or GOTO named in Label B. If no entry is made in Label C, return from the subroutine is made automatically to the next sequential step following the SBRT operation.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	YES	YES
Operand 2 - Label B	NO	NO	NO
Label C	NO	YES	YES

Examples:

(1) Transfer control to TAX and set the exit to normal return.
(2) Transfer control to TAX and set the exit to CMPUT (the EXIT corresponding to BEGIN labelled TAX is labelled EX1).

LABEL 1 3	OP 6 7		OPERAND 1 IA	LABEL A 12 14	±	18 20	OPERAND 2 IA	LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	SBRT			TAX				EX1					
	SBRT			TAX				EX1			CMPUT		

G. COMMENTS

. (PERIOD)

Function:

Comment cards have no function. They will be printed and punched, however.

Comments should not extend beyond column 61.

Example:

LABEL 1 3	OP 6 7	IA	OPERAND 1 LABEL A 12 14	±	18 20	IA	OPERAND 2 LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40	
			THESE CARDS ARE MERELY REPRODUCED										
			AND PRINTED.										

H. COPY SOURCE DECK

COPY (COPY)

Function:

Begin reproducing the source deck as comments in the output deck to be assembled, or terminate reproducing.

Where:

Operation = a mnemonic operation code (COPY)

Operand 1

Label A = The word ON or the word OFF to turn the feature on or off.

Operand 2

Label B = Not used.

Label C = Not used.

Operand Characteristics:

	IA	SR	INC
Operand 1 - Label A	NO	NO	NO
Operand 2 - Label B	NO	NO	NO
Label C	NO	NO	NO

Examples:

(1) Turn COPY on.
(2) Turn COPY off.

LABEL 1 3	OP 6 7	IA	OPERAND 1 LABEL A 12 14	±	18 20	IA	OPERAND 2 LABEL B 22 24	±	28 30	LABEL C 32 34	±	38 40
	COPY		ON									
	COPY		OFF									

I. PROGRAM ORGANIZATION

Before writing a 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 to write field definitions of the input and output areas.

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 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.

(2) Normally, the reading of an input item will be immediately followed by a test for the end of the run.

(3) The LEVL directives normally 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 LEVL operations have been written in their proper sequence, they should be followed by the processing which is to be done when no control break has occurred. This will conserve storage and result in a more efficient program.

(4) Sequence control directives should be preceded by those 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) End of job processing, to which control is transferred as a result of the test mentioned in (2) above, should include execution of the highest priority level break subroutine (assuring execution of all lower ones) and the page overflow routine.

J. OPERATING PROCEDURES

The Report Program Generator produces a 1005 Assembly Language intermediate output deck from the user's source statements; the intermediate deck is then assembled (as it stands) and a final object code deck produced.

The program is run by placing the Report Program Generator in the card read hopper, adding the source statements to the hopper, and pressing START, CLEAR, FEED, and RUN.

Each source statement will be printed and the assembly language code generated, if any, will be printed directly following. The generated code is always punched, and the source code will or will not be punched according to the most recent setting of the COPY switch (ON is set if not otherwise specified).

The assembly phase should be executed as described in the 1005 Assembly Language Manual.

A P P E N D I X 1

SYSTEM LABELS

$R1	80 Col Read Area	1 - 80
$R2	80 Character Area	81 - 160
$RC	80-Col Read Code Image	1 - 160
$PR	Print Area (132 Chars)	161 - 292
$P1	80 Col Punch Area	293 - 372
	(Also 80 Col Read-Punch Read Area)	
$P2	80-Col Read-Punch Punch Area	373 - 452
$PC	80-Col Punch Code Image	293 - 452
$Z1	80-Col Read-Punch Code Image Read	293 - 452
$Z2	80-Col Read-Punch Code Image Punch	453 - 612
$BM	First Character beyond I/O up to 961	613 - 961 (349)
$IR	Instruction register Bank I Row 32 Col 1 - 7	
$XR	Transfer register Bank II Row 32	
$TR	Translate Table Area - Upper Bank II	1859 - 1922 (64)
$AR	Arithmetic register Bank I Row 32 Col 1 - 31	
$CC	Address Counter - Chars 8 and 9 of Row 32 Bank 1	32/8/1 to 32/9/1
$X2	Generator	2/32/1
$8Ø	FIRST (leftmost) 80 Chars of Print Output Area	161 - 240

APPENDIX 2

SYSTEM SWITCHES

Two Characters with a single bit (numbered 1 to 12)

SWITCH	BIT	JUMP ON CONDITION	SET CONDITION
#FF	1 (X)	Form Overflow	
#AF	2 (Y)	Arithmetic Overflow	
#+2	3 (8)	Sense #2 Set	Sense #2
#+1	4 (4)	Sense #1 Set	Sense #1
#-2	5 (2)	Alternate #2 On	Reset Sense #2
#-1	6 (1)	Alternate #1 On	Reset Sense #1
#IN	7 (X)	Interrupt	
#UA	8 (Y)	Unit Alert	
#PE	9 (8)	Parity Error Always (Resets)	
#AP	10 (4)	Arithmetic Plus	
#AZ	11 (2)	Arithmetic Zero	
#AM	12 (1)	Arithmetic Minus	
#SO	1 (X)		Odd Parity
#SE	2 (Y)		Even Parity
	7 (X)		Reserved by hardware
	8 (Y)		Reserved by hardware
#S2	9 (8)		Servo #2
#S1	10 (4)		Servo #1
#H2	11 (2)		Indicator #2 and Halt
#H1	12 (1)		Indicator #1 and Halt
#H3	both 11 (2) & 12 (1)		

APPENDIX 3

LEVEL BREAKS

FLOW of Processing

1. ALLOW BREAK

   This statement is the first instruction of the Level Break series. It sets the ALLOW BREAK SWITCH to the ON position.

2. NO BREAK

   The program will pass through the LEVEL statements in the order given if no change in control fields is encountered. Detail processing will follow.

3. MINOR BREAKS

   When the program recognizes a change in the minor control field only, the following events will occur:

   a. The ALLOW BREAK SWITCH will be set to the OFF position.

   b. The address of the detail statement which sequentially follows the minor LEVEL statement is transferred to the EXIT statement of the minor total subroutine. (Flowchart: Ex 3 is set to D.)

   c. The program executes the minor total subroutine transfers control to detail processing via the EXIT statement of the minor subroutine.

4. INTERMEDIATE BREAKS

   When the program recognizes a change in the intermediate control field, the following events will occur:

   a. The ALLOW BREAK SWITCH will be set to the OFF position.

   b. The address of the minor LEVEL statement is transferred to the EXIT statement of the intermediate subroutine. (Flowchart: Ex 2 is set to C.)

   c. The program executes the SBRT statement of the intermediate total subroutine. This statement transfers the address of the first processing step of the intermediate total subroutine (INT + 7 in the example) to the EXIT statement of the minor total subroutine. (Flowchart: Ex3 is set to INT + 7.)

   d. The program executes the minor total subroutine and transfers control to the first processing step of the intermediate total subroutine via the EXIT statement of the minor total subroutine.

e. The program executes the intermediate total subroutine and transfers control to the minor LEVEL statement via the EXIT statement of the intermediate total subroutine.

f. The program will execute the minor LEVEL statement. No break will occur, because the intermediate break set the ALLOW BREAK SWITCH to the OFF position.

The program executes the minor LEVEL statement so that the new minor compare field can be stored.

g. The program advances to the next sequential processing step following the minor LEVEL statement.

5. MAJOR BREAKS

When the program recognizes a change in the major control field, the following events will occur:

a. The ALLOW BREAK SWITCH will be set to the OFF position.

b. The address of the intermediate LEVEL statement is transferred to the EXIT statement of the major total subroutine. (Flowchart: Ex 1 is set to B.)

c. The program executes the SBRT statement of the major total subroutine. This statement transfers the address of the first processing step of the major total subroutine (MAJ + 7 in the example) to the EXIT statement of the intermediate total subroutine. (Flowchart: Ex 2 is set to MAJ + 7.)

d. The program executes the SBRT statement of the intermediate total subroutine. This statement transfers the address of the first processing step of the intermediate total subroutine (INT + 7 in the example) to the EXIT statement of the minor total subroutine. (Flowchart: Ex3 is set to INT + 7.)

e. The program executes the minor total subroutine and transfers control to the first processing step of the intermediate total subroutine via the EXIT statement of the minor total subroutine.

f. The program executes the intermediate total subroutine and returns to the first processing step of the major total subroutine.

g. The program executes the major total subroutine and transfers control to the intermediate LEVEL statement via the EXIT statement of the major total subroutine.

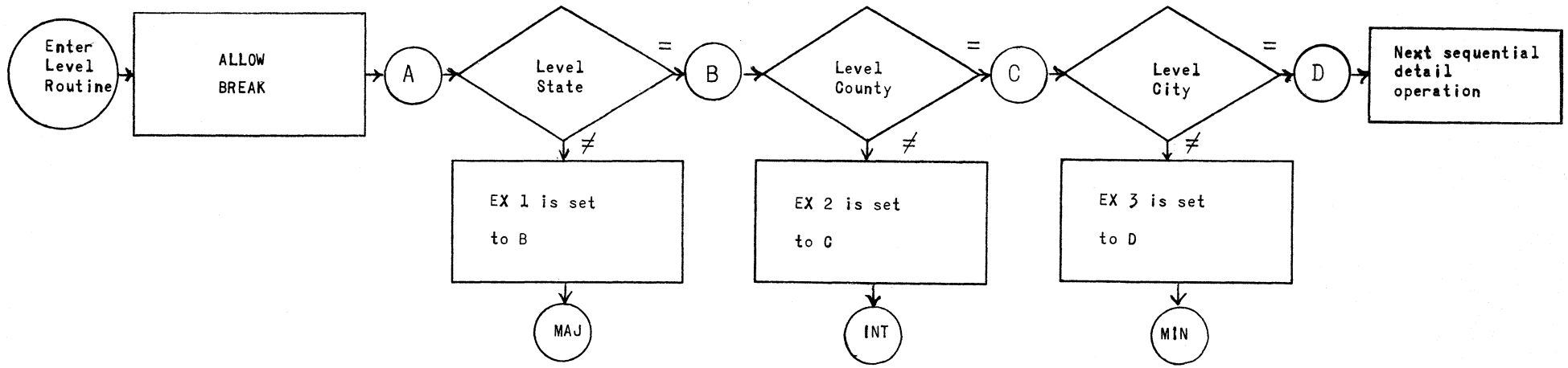
h. The program will execute the intermediate and minor LEVEL statements. No break will occur, because the major break set the ALLOW BREAK SWITCH to the OFF position.

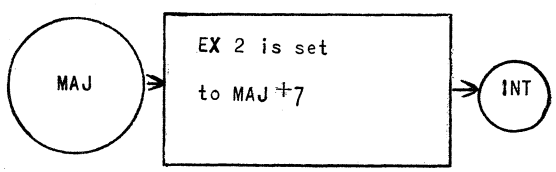
The program executes these LEVEL statements so that the new intermediate and minor compare fields can be stored.

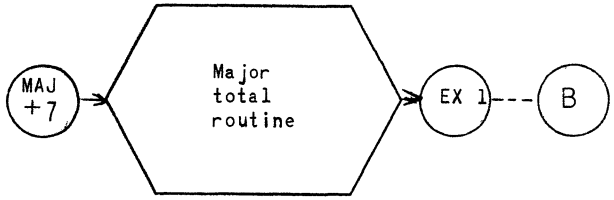
i. The program advances to the next sequential processing step following the minor LEVEL statement.

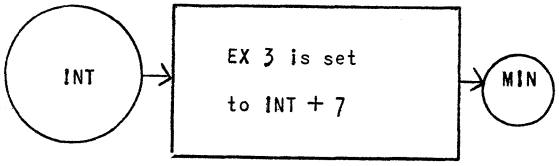
LEVEL BREAKS

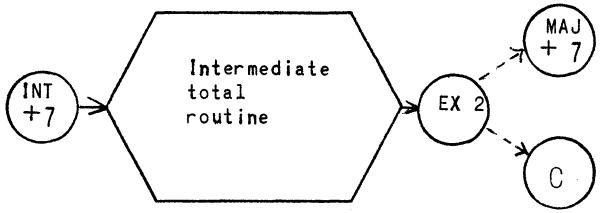
	LABEL 1 3	OP 6 7	OPERAND 1 IA *	LABEL A 12 14	± 18 20	OPERAND 2 IA *	LABEL B 22 24	± 28 30	LABEL C 32 34	± 38 40
DETAIL PROCESSING										
		ALLOW		BREAK						
LEVELING ROUTINE	A	LEVLA		STATE			MAJ		2	
	B	LEVLA		COUNT			INT		3	
	C	LEVLA		CITY			MIN		4	
DETAIL PROCESSING	Next Sequential Detail Instructions									
		Detail Instructions								
	MAJ	BEGIN								
		SBRT		INT			EX2			
MAJOR TOTAL SUBROUTINE	Total Instructions									
	Total Instructions									
	EX1	EXIT								
	INT	BEGIN								
		SBRT		MIN			EX3			
INTERMEDIATE TOTAL SUBROUTINE	Total Instructions									
	Total Instructions									
	EX2	EXIT								
	MIN	BEGIN								
		Total Instructions								
MINOR TOTAL SUBROUTINE		Total Instructions								
		Total Instructions								
	EX3	EXIT								

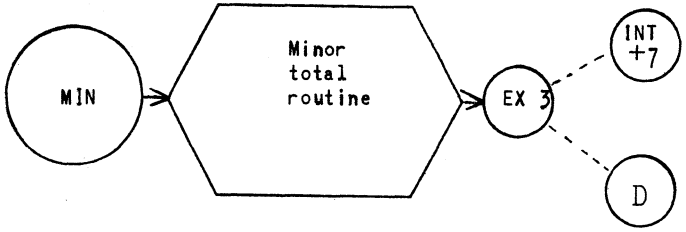












Appendix 3

LEVEL BREAKS

APPENDIX 4

Use and Definition of Edit Masks

Edit masks are normally used to insert specified characters into, and/or delete certain leading characters from, numeric fields. Insertions may also be made into alphanumeric fields. One field is "edited" into a second through use of the Move-with-edit macro, MVEDT. An edit mask may not exceed a length of 31 characters.

Internal counters examine the contents of successive locations of the "sending" field, the "receiving" field, and the edit mask, beginning with the "leftmost" character (the MSL) of each. The editing process is terminated when either the last character of the edit mask has been reached, or case (2) described below occurs. The receiving field is not automatically cleared to blanks by the editing process. During editing, the characters comprising the edit mask have the following meaning:

(1) All characters except unequal, lozenge, left-slash, and delta: Send the current character of the edit mask to the current character of the receiving field and increment by one position the edit mask and receiving field counters.

(2) Unequal (≠): Terminate the editing process and do not send a character to the current character of the receiving field.

(3) Lozenge (⌑): Move the current character of the sending field to the current character of the receiving field, and advance all three counters by one position.

(4) Left-slash (\): Turn on the zero-suppress feature (blank fill), do not increment any counters, and then continue as in case (3). The zero-suppress feature is turned off by the next "current character" of either the sending field or edit mask which is neither a zero nor a comma. While the feature is in effect, zeros and commas sent from either the sending field or edit mask are changed to blanks during transmission to the receiving field.

(5) Delta (Δ): Turn on the zero-suppress feature (asterisk fill), do not increment any counters, and then continue as in case (3). The zero-suppress feature is turned off by the next "current character" of either the sending field or edit mask which is neither a zero nor a comma. While the feature is in effect, zeros and commas sent from either the sending field or edit mask are changed to asterisks during transmission to the receiving field.

Examples:

A	DC	17	TOTAL b̸ b̸ $ ⌶ ⌶, ⌶ ⌶ ⌶. ⌶ ⌶
B	DC	17	TOTAL b̸ b̸ $ \ ⌶, ⌶ ⌶ ⌶. ⌶ ⌶
C	DC	17	TOTAL b̸ b̸ $ Δ ⌶, ⌶ ⌶ ⌶. ⌶ ⌶
D	DC	17	TOTAL b̸ b̸ $ \ ⌶, ⌶ Δ ⌶. ⌶ ⌶
E	DC	7	1234567
F	DC	7	ØØØØØØ2
G	DC	7	ØØ34567
H	DC	7	ØØØØ567

A on E	produces	TOTAL $12,345.67
B on E	"	TOTAL $12,345.67
C on E	"	TOTAL $12,345.67
D on E	"	TOTAL $12,345.67
A on F	produces	TOTAL $ØØ,ØØØ.Ø2
B on F	"	TOTAL $.Ø2
C on F	"	TOTAL $******.Ø2
D on F	"	TOTAL $ **.Ø2
A on G	produces	TOTAL $ØØ,345.67
B on G	"	TOTAL $ 345.67
C on G	"	TOTAL $***345.67
D on G	"	TOTAL $ 345.67
A on H	produces	TOTAL $ØØ,ØØ5.67
B on H	"	TOTAL $ 5.67
C on H	"	TOTAL $*****5.67
D on H	"	TOTAL $ *5.67

UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UP-4072.2