INPUT - OUTPUT

LOADING PROGRAM

1. When the programmer has completed the code for his problem and checked the code sheets and variable definition sheets, the program is then ready to be key-punched. It is advisable to have listings made on a tabulator of both the code and variable definition sheets; the remarks or notes on the code sheet can be printed. Frequently, errors can be detected at this time; corrections can be made more easily and without waste of machine time at this point in the progress of the program.

2. The program is then ready to be compiled by the Pact I routine; that is, the Pact I steps, or pseudocode will be translated into relative machine language. This involves many things, including the following items:

A. Expand the Pact I operations.

B. Insert shift instructions when necessary.

C. Insert the required subroutines and library programs, and expand the calling sequences for these subroutines or library programs.

D. Duplicate regions when necessary.

E. Add Pact I steps.

F. Assign relative locations for variables, numbers and temporary storage.

G. Make variable list and list of program. Explanation of these lists are given below in paragraph 3.

H. Punch relative binary cards which are of three types:

(1) Load program

(2) Control Card

(3) Instructions for this program.

3. The listing of the program contains both the pseudocode written as Pact I steps and the relative machine language code. The pseudocode is printed as the programmer wrote it with some additional information, one step per line on the left side of the page. The corresponding relative machine language instructions are printed on the right side of the page not more than four instructions per line. All locations of instructions and references to instructions are printed in the octonary system; everything else is in the decimal system. Frequently, there are more than 4 machine language instructions for each step and consequently, more than one line is required on the right side for the compiled program. In such a case blank spaces will appear on the left side until

a new step translated by the compiling routine is listed. The headings for the left side of the listing, the pseudocode, are exactly the same as the headings on the code sheet; however, the notes are omitted. On the right side which represents the relative machine language instructions, the first column is the relative location, the second column is the operation, the third column is the type and the fourth column is the relative address. The three last named columns are repeated three times. The illustrative problem, beginning on page 04·31·90 includes a sample listing.

4. The variable list contains the relative location of each variable in addition to information that was originally written on the variable definition sheet. For a one dimensional or two dimensional array both the first and last relative locations are given. The first column contains the Tag which indicates the type of variable that is being described on this line.

Tag	Variable
S	Scalar
V	One dimensional array, or vector
M	Two dimensional array, or matrix

5. Although it is not necessary for the programmer to know all the details of the relative binary card, it may be interesting and also, at times, helpful for him to be acquainted at least with the form.

A. A relative instruction has the following form:

Operation	Relative address	Index

The sign, plus or minus, has the usual meaning, referring to either a half-word or full-word. The operation will be any one of the usual IBM 701 operations. The relative address together with the index represent the address. The indices are:

Indices	Meaning	Symbol on Listing (type)
0	Actual or absolute	A
1	Instruction	I
2	Variable	V
3	Temporary	T
4	Number	N
5	Temporary for Library Program	
6	(Not assigned)	
7	Symbol	S

B. The relative instructions and information necessary for reading the card are punched in each card. As many as 36 relative instructions can be punched on one card. Space can be conserved on the binary card by punching compactly each instruction without its index in order and also each corresponding index in the same order but in a different part of the card. The instructions containing the sign, the operation and relative address without index are punched in order starting in the 9-row left and continueing through the 1-row right for a full card. The three bits representing the index for each instruction are punched in the 0-row and the 11-row left. These indices contain only three bits and are punched in the same order as the instructions. The index corresponding to the first instruction on the card is punched in card columns 9 through 11 of 0-row, index for second instruction is punched in card columns 12 through 14, etc. The remainder of the card contains information for reading the card. The instruction count is punched in the first half of the 11-row right. The second half of the 11-row right is blank. The code for the card itself is given in the operation part of the first half of the 12-row left (card columns 9 through 14). -0 represents a relative instruction card and -1 represents a relative control card. The address part of the first half of the 12-row left (card columns 15 thru 26) gives the relative card origin. The second half of the 12-row left contains the symbol which is the region or library program symbol in base 48 binary equivalent. The card check sum always appears in 12-row right in card columns 56 through 79. This check sum is formed by adding all the information on the card except itself, essentially as half-words but including the sign bit. An illustration of a binary card appears on page 04·32·5.

6. After the listings are completed and the cards punched, the program can be run on the machine, using the necessary decimal data cards. The information on the control card is used by the loading program; as the program is loaded into electrostatic storage, absolute locations are assigned to all relative locations. Execution of the instructions of the program commences immediately after loading with the first instruction of the program.

READ

1. The decimal data to be used with a program is written on the Pact I data sheets from which the key-punching is done. The data is properly stored by means of the Pact I operation, Read (READ). As many as four full-words of data can be punched in one card. Each word of information must have its location, decimal and binary magnitudes written in addition to the number itself.

2. There are a few restrictions on the use of these various fields of the decimal data card. The magnitudes must lie within prescribed ranges:

$$-5 \leq P \leq 15$$

$$-18 \leq Q \leq 52$$

where P is the number of decimal positions to the left of the decimal point of the number as written. The location of a variable can be taken from the variable list. The location of the first element of a 2 dimensional array is always indicated on the variable list. The location of any element of a 2 dimensional array is ordinarily computed from the variable list by the formula:

$$LOC(A_{IJ}) = LOC(A_{11}) + \Delta_1(I - 1) + 2(J-1)$$

Note $\Delta_1 = 2D_2$ and prints on the variable list under S_1. This would seem to indicate that decimal data can never be punched in advance of compilation of the program. However, this is not always true, as explained in paragraph 4, below.

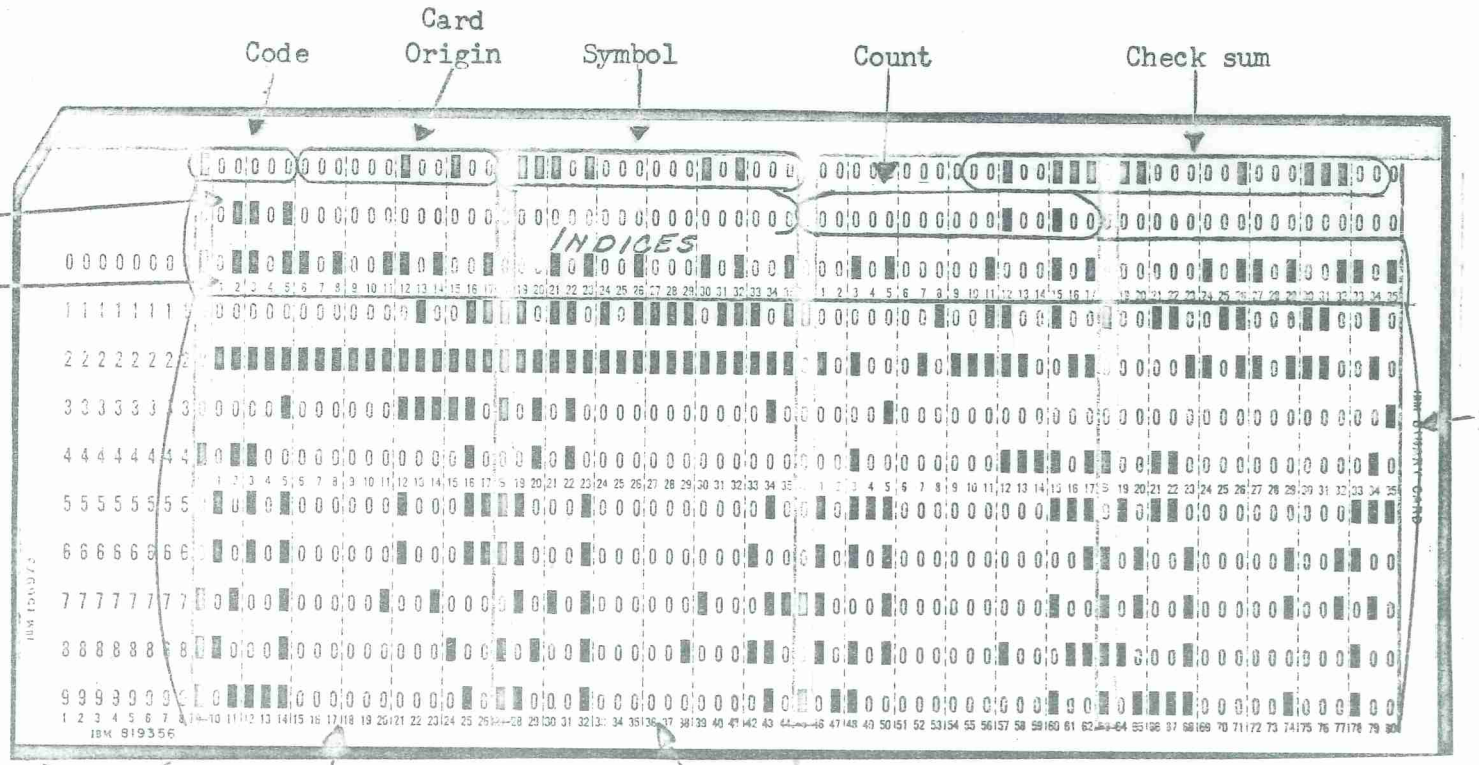
3. There are few clerical details to keep in mind when the data sheet is being completed. Locations specified must be even. The number itself may be placed in any position in the field, if the binary magnitudes will accomodate the number. A card need not contain 4 pieces of information. Any of the 4 lines on the data sheet representing a decimal data card can be left blank; the corresponding fields on the decimal data card will be blank and no information will be stored from these fields. Each field, if punched, must have 1 and only 1 punch per column in the 0 through 9 rows. If, by mistake, the same location is indicated for two pieces of data on one card, the data that is farthest to the right, or in the higher card columns, will be in electrostatic storage to be used in the program. An illustration of a Pact I decimal data card is displayed on page 04·32·5. The last card in a set of data must have a 12 punch in card column 80.

4. It is possible to compile a program and make a run at once if a few precautions are observed. All variables must be listed on the variable definition sheet with no constraints. All variables will appear on the variable list in the order in which the variable cards are read. The first one listed will have relative location 0. The location of the first element of any array can be obtained by counting the number of elements contained in all variables preceding it and multiplying by 2 (The number of elements are essentially specified by the product $D_1 \cdot D_2, (D_I \geq 1)$, on the variable definition card). Then the location of an element A_{IJ} can be computed from the location of A_{11} by the formula given in paragraph 2, above, noting that D_2 is the maximum number of columns as specified on the variable definition sheet.

LIST

1. Printing can be accomplished by giving the List (LIST) operation in Pact I followed by 1 ID instruction for each piece of information to be printed. The factor part of the ID instructions may call for either a variable, or the results of a previous step. Six 11 digit numbers can be printed on a line; the decimal point and the sign, if minus, are printed. A positive sign is omitted. Numbers which have absolute values within the range $0 \leq |N| \leq (2^{35} - 1)$ can be printed. If the number is larger in absolute value than $2^{35} - 1$, a series of dots will be printed, (....). It will not cause an error stop and all other proper information will be printed. In general, the magnitude, Q, of the data to be printed should be less than or equal to 35; otherwise the number may be too large to be printed,

RELATIVE BINARY CARD

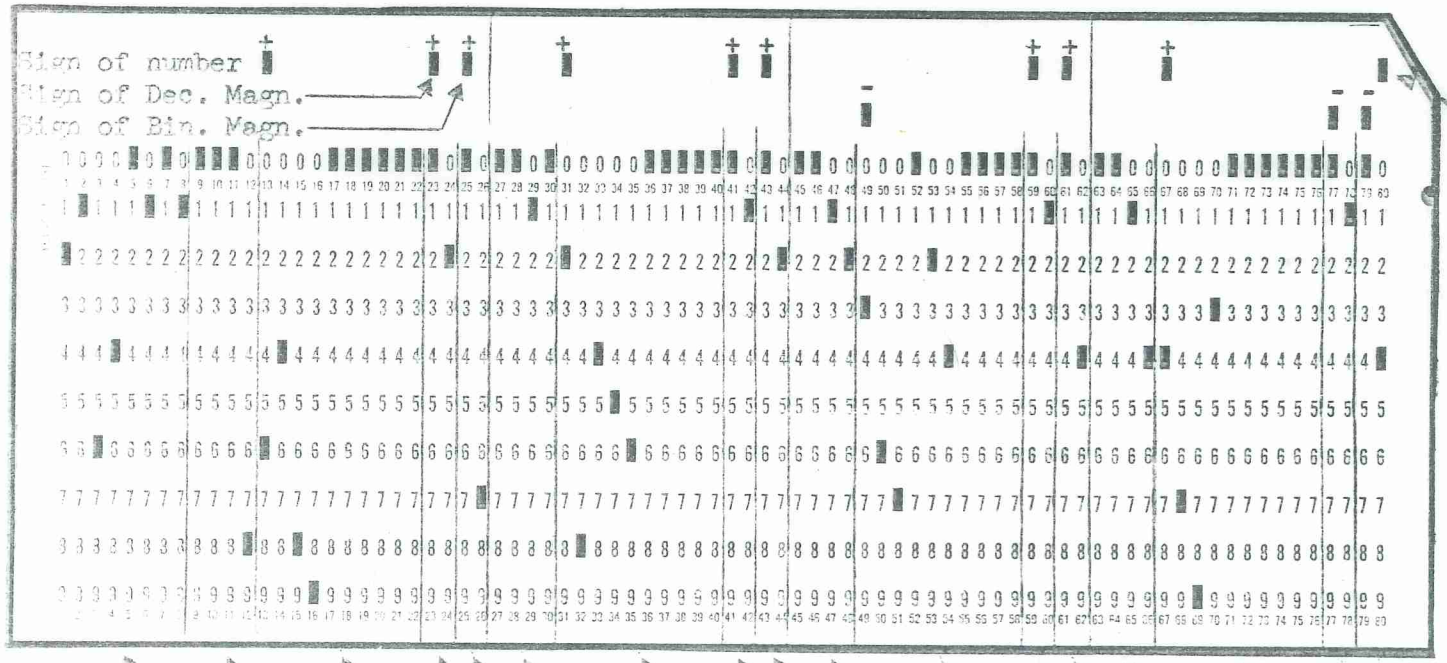


or significance may be lost. If a number is less in absolute value than 10^{-11} approximately, zero will be printed. There is no need to specify the magnitude, Q, of the information to be printed.

2. Information can be spaced and spread across the page in whatever fashion the programmer desires by giving appropriate Pact I steps. By giving a step +Q>0 whose operation is Identification (ID) and whose factor is blank, the corresponding field on a line of printing will be blank. In this way information may be spread across the page. Extra spaces after any can be obtained by giving various sense steps immediately after the last ID step. The step, SE 519, produces one extra space after printing; the step, SE 520, produces two extra spaces after after printing. Sheet ejection is effected by the step, SE 521, after printing. (The number 519, 520, or 521 must be written in the S_1 field) A Halt (HALT) step may not occur immediately after the last ID step.

PACT I

DECIMAL DATA CARD



1-8 Identification 21640101

9-12 First Location 8

13-22 First Number +6489

23-24 Decimal Magnitude +2

25-26 Binary Magnitude +7

27-30 Second Location 10

31-40 Second Number +28456

41-42 Decimal Magnitude +1

43-44 Binary Magnitude +2

45-48 Third Location 12

49-58 Third Number -367024

59-60 Decimal Magnitude +1

61-62 Binary Magnitude +4

63-66 Fourth Location 14

67-76 Fourth Number +4793

77-78 Decimal Magnitude -1

79-80 Binary Magnitude -4

OK New 7-18-55

TABLE 1

PACT I OPERATIONS

OPERATION	SYMBOL	OPERATION	SYMBOL
Take	(No symbol)	Clear	CL
Add	+	Set	SET
Subtract	-	Test	TEST
Multiply	x	Use	USE
Divide	/	Sine	SIN
Residue	RES	Cosine	COS
Equals	EQ	Arctangent	ARCT
Absolute	ABS	Square Root	SQRT
Add absolute	+ABS	Logarithm	LOG
Subtract absolute	-ABS	Exponential	EXP
Transfer	T	Duplicate	DUP
Transfer on Zero	TZ	Exit	EXIT
Transfer on Positive	TP	Library Program	LIB
Transfer on Negative	TN	Identification	ID
Transfer on Overflow	TF	For	FOR
Halt	HALT	Call	CALL
Do Region and Return	DO	Read	READ
Sense	SE	List	LIST

TABLE 2A

Restrictions on Use of Operations

Operations that may have blank factor and number field:

OPERATION	SYMBOL	OPERATION	SYMBOL
Add	+	Sine	SIN
Subtract	-	Cosine	COS
Add absolute	+ABS	Square root	SQRT
Subtract absolute	-ABS	Logarithm	LOG
Multiply	x	Exponential	EXP
Divide	/	Arctangent	ARCT
Residue	RES	Test	TEST

Operations having the following restrictions: (1)

a) The step having this operation should not specify Q because Q is not recognized.
b) No reference should be made to the results of a step having this operation.

Do region and return	DO	Call	CALL
Library Program	LIB	Exit	EXIT
Identification (2)	ID	Use	USE
For	FOR	Clear	CL
Sense	SE	Read	READ
Set	SET	List	LIST
Test	TEST	Transfer	T

(1) The conditional transfers are not included in this table; However, if the condition of the transfer is satisfied, it is not permissible to refer to the results of the step.

(2) Q becomes a parameter in a calling sequence.

Rev 7-18-55

TABLE 2B

FIRST OPERATION IN A NEW SEQUENCE OF STEPS

PROPER FIRST OPERATIONS	
OPERATIONS	SYMBOL
Take	TAKE
Transfer	T
Halt	HALT
Do region and return	DO
Sense	SE
Clear	CL
Set	SET
Test	TEST
Use	USE
Absolute	ABS
Sine (2)	SIN
Cosine (2)	COS
Arctangent (2)	ARCT
Square root (2)	SQRT
Logarithm (2)	LOG
Exponential (2)	EXP
Duplicate (1)	DUP
Exit	EXIT
Library Program	LIB
Call	CALL
Read	READ
List	LIST

IMPROPER FIRST OPERATIONS	
OPERATIONS	SYMBOL
Add	+
Subtract	-
Multiply	x
Divide	/
Residue	RES
Equals	EQ
Add absolute	+ABS
Subtract absolute	-ABS
Transfer on zero	TZ
Transfer on positive	TP
Transfer on negative	TN
Transfer on overflow	TF
Identification	ID
For	FOR

(2) These are improper if the factor is blank.

(1) Duplicated region must have proper first operation.

New 7-18-55

8-1-55

Remove vertical lines

TABLE 3

SUBROUTINE OPERATIONS

SUBROUTINE	Q of ARGUMENT	Q of RESULT (1)	CALLING SEQUENCE SUPPLIED (3)		
			LOC.	OP	ADDRESS
Sine	$\|Q\| \leq 35$	Q = 1	a-1 a a+1 a+2	IM RA T H	-L (argument) +a +(SIN) ± Q of argument
Cosine	$\|Q\| \leq 35$	Q = 1	a-1 a a+1 a+2	IM RA T C	-L (argument) +a + (COS) ± Q of argument
Square root	No restriction	Q = integral part of 1/2 (Q_A + 1) (2)	a-1 a a+1 a+2	IM RA T H	-L (argument) +a + (SQRT) ± Q of argument
Exponential	$-34 \leq Q \leq 34$	Coder must specify Q of result	a-1 a a+1 a+2 a+3	IM RA T H H	-L (argument) +a +(EXP) +(34-Q_A) (2) ± Q of result
Logarithm	$-35 \leq Q \leq 35$	Q = 6	a-1 a a+1 a+2	IM RA T H	-L (argument) +a + (LOG) ± Q of argument
Arctangent	$-50 \leq Q \leq 50$	Q = 1	a-1 a a+1 a+2	IM RA T H	-L (argument) +a +(ARCT) ±Q of argument

(1) The Q of the result can be specified by the programmer; however, if the Q field is left blank on the code sheet, the result is stored at the Q indicated in this column. When the coder does specify a Q of the result, he should guard against an overflow caused by a Q that is too small or the loss of significant digits caused by a Q that is too large.

(2) Q_A is the Q of the argument.

(3) The address for the instruction at location a + 1 below is indicated by the name of the subroutine. The correct address will be entered when the program is compiled.

TABLE 4

REMINDERS ABOUT VARIABLES AND NUMBERS

1. Definitive subscripts must begin with S_1.

2. Any 2 variables with the same factor must have the same number of definitive subscripts and either S_1 for the one variable differs from S_1 for the other variable, or the variables must have 2 definitive subscripts and S_2 for the one variable differs from S_2 for the other variable.

3. All vectors, matrices, and constraints must be specified by a variable card.

4. Every variable must appear either as a variable on a variable card or as a factor of a Pact instruction with an equals (EQ) order part.

5. The NUM constraint cannot be used to specify a relative origin of 0000.

6. There must be at least one variable card.

7. No variable can be specified on the same card with a NUM constraint.

8. The last variable card must have a 12 punch in column 80.

9. For matrices the following correspondences always hold true, both for variable cards and for Pact instructions:

D_1 corresponds to S_1 and only to S_1.

D_2 corresponds to S_2 and only to S_2.

10. A maximum of 341 variables in all are allowed; i.e., the number of variables plus the number of different EQ variables must not exeed 341. The total number of variables plus the number of numbers must not exceed 512.

11. The sum consisting of the number of variables assigned plus twice the number of constraining variables not yet assigned cannot exceed 512 at any time of the variable assembly process.

12. A maximum of 10 phase positions are permitted.

13. Let X and Y be any two different variables, and let Z be any constraining variable. Then none of the following situations is permitted:

a. X SUC Z and Y SUC Z

b. X IMS Z and Y IMS Z

c. X SUC Z and Y IMS Z , or vice versa.

14. Every constraining variable must appear elsewhere as a variable.

15. No variable should appear more than once as a variable on a variable card.

16. A maximum of 120 REL variables are permitted in a problem.

17. If a variable appears as a variable and as the factor of one or more instructions with an EQ order part, the Q must be the same in all places.

18. Variables in S_2 must have integer values (Q = 17).

TABLE 5

RULES FOR Q

1. If a particular variable occurs in more than 1 instruction having an EQ order part, then the Q's specified must be the same. Otherwise the Q of one of these instructions will be assigned to all the others (when no variable card is used for this variable).

2. If a variable is specified on the variable definition sheet and its Q column is blank, then the Q specified by an instruction having an EQ order part in which the variable occurs will be used. then the variable will be assigned a Q of zero.

3. If a variable is specified on the variable definition sheet and its Q column is not blank, then this Q will be assigned to all instructions having EQ order parts in which the variable occurs.

4. If the Q of a variable is blank on the variable definition sheet or the variable is not listed on the variable definition sheet, and also if the Q is left blank in the EQ step in which the variable occurs, then a Q equal to zero will be used.

5. The Q must be specified for the result of any step to which reference is made by means of an R or N in the clue column of the factor field.

4. If the Q is left blank in ~~the~~ EQ steps in which a given variable occurs, then a Q = 0 will be used - if not specified or var. by. shut. If Q is specified ~~more than once~~ in one EQ step, then this Q will be automatically assigned to the variable in all other EQ steps where the Q is blank

TABLE 6

ABBREVIATIONS FOR

IBM 701 OPERATIONS USED IN LISTING

Abbreviation	Operation	Abbreviation	Operation
H	Halt and transfer	M	Multiply
T	Transfer	MR	Multiply and Round
TF	Transfer on overflow	D	Divide
TP	Transfer on Plus	R	Round
TZ	Transfer on Zero	LL	Long Left shift
S	Subtract	LR	Long Right shift
RS	Reset and Subtract	AL	Accumulator Left shift
SV	Subtract absolute Value	AR	Accumulator Right shift
N	No operation	RD	Prepare to ReaD
A	Add	RB	Prepare to Read Backward
RA	Reset and Add	W	Prepare to Write
AV	Add absolute Value	WE	Write End of file
ST	STore	RW	ReWind tape
SA	Store Address	SD	Set Drum address
SM	Store MQ register	SE	SEnse and skip or control
LM	Load MQ register	C	Copy and skip

ERROR STOPS IN PACT I ROUTINE

CARD READING SENSE LIGHT 1

LOCATION (Octonary)	INSTRUCTION	CONDITION	REMARKS
3076		Light 1 on; sequence error in Pact cards	"Start" will continue reading
3427		Copy checks; No 12-last in column 80 of last Pact instruction card	
4250		Light 2 on; operation not a legitimate Pact operation	"Start" will continue with false operation.

INITIAL ASSEMBLY SENSE LIGHT

7076	00 5613	Tape record exceeds expected maximum length.	
7077	00 6454	Blank Q referred to by R or N	
7100	00 6356	Too many duplicate operations	
7102	00 6470	Improper Transfer (non-existent step number).	
7103	00 5756	R or N factor below any step number (blank?)	
7104	00 5761	Transfer below any step. May be caused by blank factor on transfer.	
7105	00 5505	Tape Check (if machine has tape check sense feature).	
7106	00 5535	Tape Check (if machine has tape check sense feature).	
7107	00 5764	Too many duplicate operations.	
7110	00 5620	Tape check (if machine has tape check sense feature.)	

INTERMEDIATE COMPUTATIONS SENSE LIGHT

LOCATION (Octonary)	INSTRUCTION	CONDITION	REMARKS
6313	00 6257	Duplication Loop	
6314	00 6236	Duplication of non-existent region	
7214	00 6354	Region too large	
7215	00 6643	No Region to be duplicated	
7216	00 6716	No Transfer factor	
7220	00 7173	Improper dup search	
7221	00 7200	Dup tape search error	
7223	00 6321	Tape check (if machine has tape check sense feature)	
7224	00 6354	Tape check (if machine has tape check sense feature)	

VARIABLE STORAGE SENSE LIGHT

(Sense Switch 6 must be off)

Part I			
6271		Non-standard constraint	Check variable cards
6345		Two cards with same variable	Check variable cards
6526		Check-Sum error reading tapes on drums	Place switch #6 on, hit start, and then place #6 off. If halt occurs again, the record was probably written incorrectly.
6537		D1 is a non-blank symbol	Check variable cards
6545		D2 is a non-blank symbol	Check variable cards
6554		Definitive subscript begins in S2	Check variable cards
6615		D2 is a non-blank symbol	Check variable cards
6654		D2 is a non-blank symbol	Check variable cards

VARIABLE STORAGE SENSE LIGHT (cont'd)

LOCATION (Octonary)	INSTRUCTION	CONDITION	REMARKS
6664		Definitive subscript begins in S2	Check variable cards
7010		Non-standard AVS indicator	Machine inconsistency
7053		Variable table overflow	Check variable cards
7321		REL LOC is non-blank symbol	Check variable cards
7354		REL LOC is non-blank symbol	Check variable cards
7500		Too many REL's -- 120 maximum	Check variable cards
Part 2			
5712		Check-sum error reading tapes or drums	Place switch #6 on, hit start, and then place #6 off. If halt occurs again, the record was probably written incorrectly.
5725		REL LOC is non-blank symbol	
5760		REL LOC is non-blank symbol	
6167		Switch #6 is on	
6173		REL LOC is non-blank symbol	
Part 3			
5712		Check sum error reading tapes or drums (See Part I)	
6034		Non-standard AVS indicator	Probable machine inconsistency
6077		Variable table overflow	Check variable cards

VARIABLE STORAGE SENSE LIGHT (cont'd)

LOCATION (Octonary)	INSTRUCTION	CONDITION	REMARKS
6375		REL LOC is non-blank symbol	Probable machine inconsistency
6554		Variable print overflow - max. 341	Check variable cards
6601		Two cards have same variable	Check variable cards
6662		Variables overflow high-speed storage	Check variable cards
7157		Variable print overflow - max. 341	Check variable cards
7531		Print variable not in variable table	Probable machine inconsistency
7563		Variable print overflow - max. 341	Check variable cards
Part 4			
6747		There are too many variables to be listed. The remainder of the variables (up to the allowable maximum) will be assigned correctly but will not be listed.	
7032		The Q of this instruction is not equal to the Q specified on the variable card (in this case the Q of the variable card will be assigned) or it is not equal to the Q of some other instruction with an EQ order part and the same factor. (In this case the Q of the other instruction will be assigned).	Region and step in the accumulator in the notation used by Part I.

VARIABLE STORAGE SENSE LIGHT (cont'd)

LOCATION (Octonary)	INSTRUCTION	CONDITION	REMARKS
Part 5			
6501		The Q of the variable in S_2 is not 17.	Region and step in the accumulator in the notation used by Pact I.
6635		The use of one subscript field is not allowed in this instruction.	Region and step in the accumulator in the notation used by Pact I.
6746		The variable in S_2 has not been specified either by an EQ operation or by a variable card.	Region and step in the accumulator in the notation used by Pact I.
6751		The use of two subscript fields is not allowed in this instruction.	Region and step in the accumulator in the notation used by Pact I.
6753		The variable in the factor has not been specified either by an EQ operation or by a variable card. A Q of 0 will be assigned to the variable.	Region and step in the accumulator in the notation used by Pact I.
6763		No Q has been assigned to the variable in the factor field. It will be assigned a Q of 0.	Region and step in the accumulator in the notation used by Pact I.
7044		There are too many variables to be listed. The remainder of the variables (up to the allowable maximum) will be assigned correctly but will not be listed.	

OPERATION EXPANSION SENSE LIGHT

1017		No unit records on tape 400	
1025		Only one (1) unit record on tape 400	
1064		But it panics	

CALLING SEQUENCE SENSE LIGHT

LOCATION (Octonary)	INSTRUCTION	CONDITION	REMARKS
3443	00 3403	The first step of a region is not a CALL.	
3507	00 3510	Improper end of file condition.	
3561	00 4160	Improper end of file condition.	
4042	00 4043	Improper end of file condition.	
4164	00 5425	Improper end of file condition.	
5450	00 5456	Tape record too large.	
6047	00 6001	Improper end of file condition.	
7523	00 7525	Improper end of file condition.	
7574	00 6001	Improper end of file condition.	
7626	00 7634	Tape record too large.	

STOPS IN LOOP GENERATION SENSE LIGHT

1266		Subscript found which is not set or used	
1270		Either one or a double subscript of a variable found which is not set or used.	
1434		Factor on FOR is a symbol	
1435		S1 on FOR is a symbol	
1436		S2 on FOR is a symbol	
1437		S1 on FOR is a symbol	
1440		S2 on FOR is a symbol	

FINAL ASSEMBLY SENSE LIGHT (cont'd)

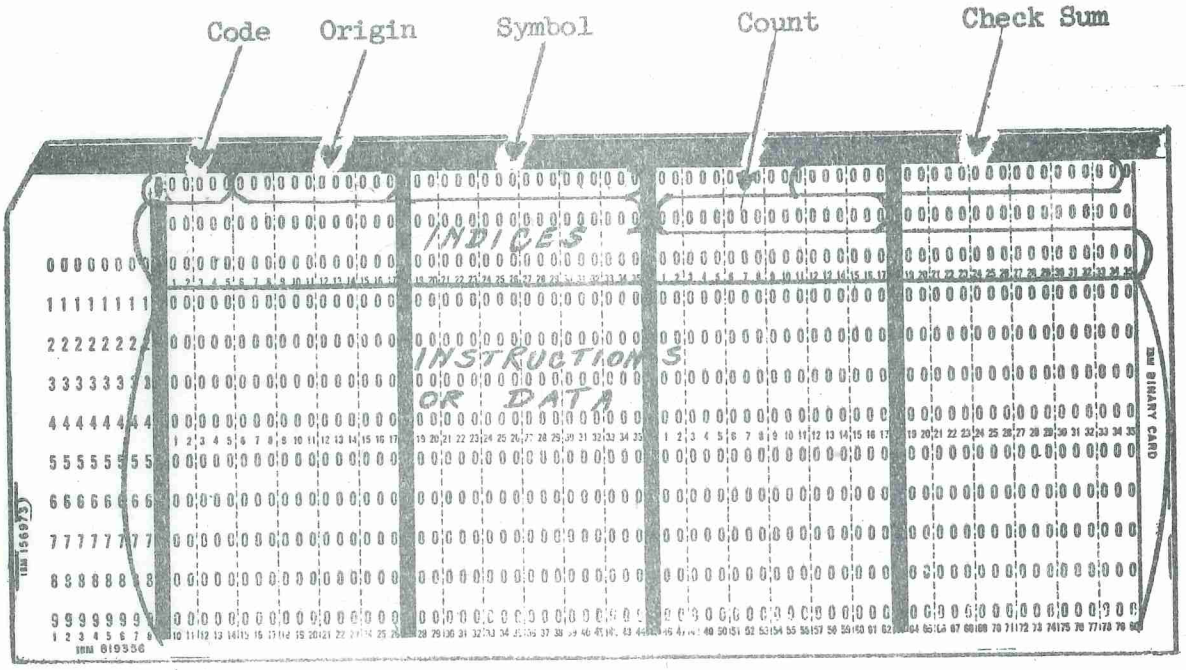
LOCATION (Octonary)	INSTRUCTION	CONDITION	REMARKS
Pass 1			
6270	00 6247	Instruction Count Compiling Error	
6271	00 6126	Improper Tape Format	
6272	00 6045	Tape Check (if machine has tape check sense feature)	
6273	00 6126	Oversized Tape Record	
6274	00 6126	Tape Check (if machine has tape check sense feature)	
Pass 2			
6320	00 6126	Non-existent symbol	
6321	00 6106	Bad symbol due to bad subscript expansion or bad call expansion	
6322	00 5607	Too many symbols	
6323	00 5615	Improper tape format	
6324	00 5701	Tape check (if machine has tape check sense feature)	
6325	00 5734	Oversized tape record	
6326	00 5734	Tape Check (if machine has tape check sense feature)	
6014 6124		Proper end. Machine error	

CHECKOUT

LOADING AND EXECUTING A PACT I COMPILED PROGRAM

1. A special relative binary card form has been designed for Pact I. Although it is not necessary to understand how binary card loading is accomplished, this information is useful at check out time and will be necessary if the programmer wishes to overlap instruction region storage.

2. Pact Relative Binary Card Form:



a. Code: (-00) Relative Binary Instruction cards have code -00.

b. Origin: Specifies the relative location of the first half-word represented by the card.

c. Symbol: Represents the base 48 (in binary) equivalent of the region or library program symbol. (The variable region is here represented by +2, temporary by +3, number by +4, etc.)

d. Check Sum: Consists of the positive sum of the first 46 half-words on the card including the sign bit as the 18th bit of each half-word.

e. Count: Indicates the number of half-words represented by the card.

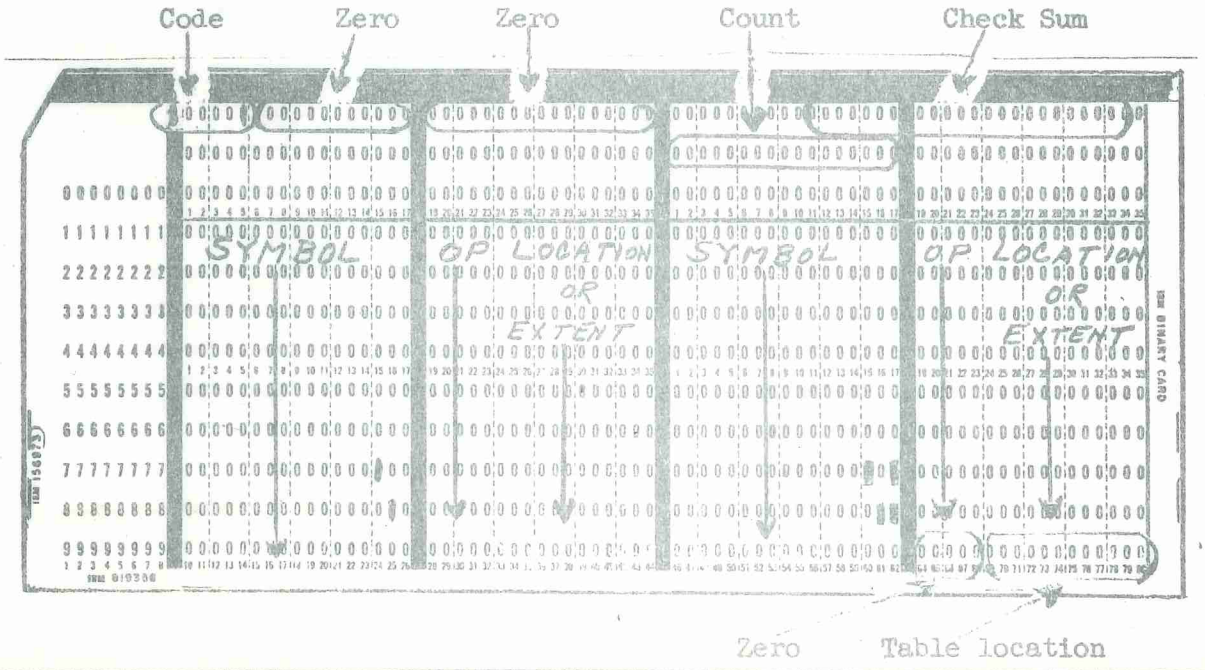
f. Instructions or Data and Indices: Corresponding to each half-word of instructions of data is a 3 bit index. These indices are as follows:

Index	Code on Listing	Region
0	A	Absolute
1	I	Instruction
2	V	Variable
3	T	Temporary
4	N	Number
5	...	Library Programs' Temporary
6	...	not used
7	S	Symbolic

Instruction references correspond to the region indicated by the symbol field into which the card is being loaded.

3. The program to load these binary cards and control cards is a special self loading program occupying 6 cards. These 6 cards are identical for all programs compiled by Pact I. This program is located at the high order end of electrostatic and loads cards by first reading their card image into the first 48 half-words of memory and then interpreting them. The loader will continue reading relative binary cards or control cards until it encounters an error stop or a card with a card code other than -00 (Relative Binary Instruction Card) or -01 (Pact Region Origin Control Card). If the start button is pressed after an error stop then the loader will continue to interpret the current card and then continue loading. When a card is read which does not have a code of -00 or -01 then control is transferred to zero immediately after the card has been read into the first 48 half-words.

4. Pact Region Origin Control Card (Initial Control Card)



a. Code: (-01) Pact region origin cards have code (-01).

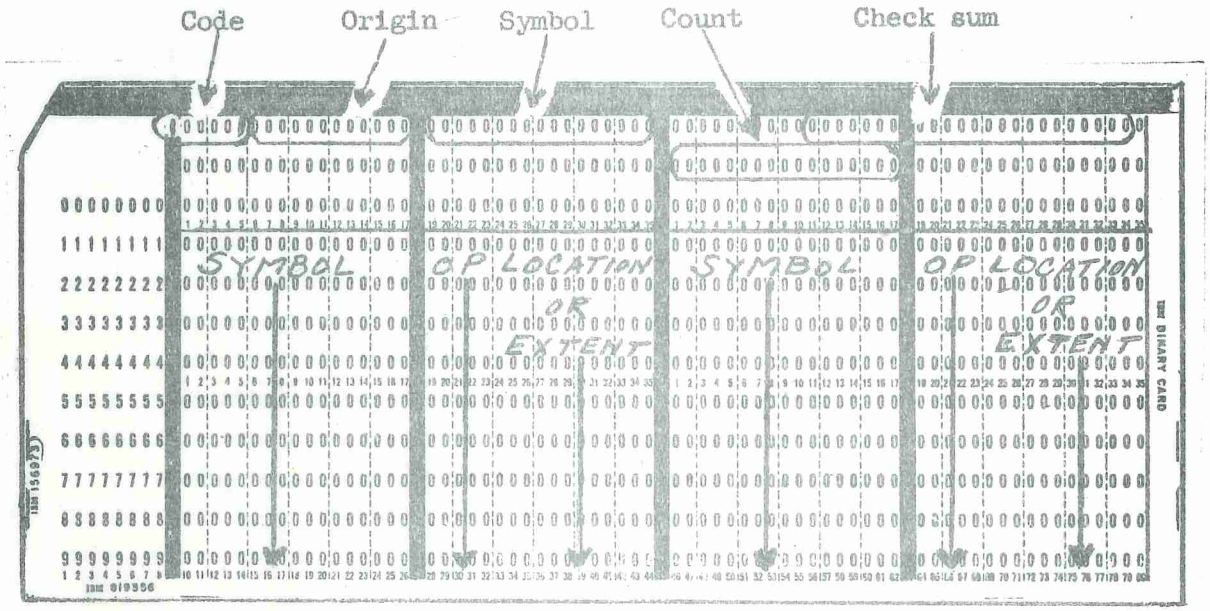
b. Check Sum: The check sum is the positive sum of the first 46 words on the card including the sign of each half-word as the 18th bit.

c. Count: Indicates the number of half-words to be loaded into the table by the card. The count must be even.

d. Table Location: Indicates the location of the region origin table in electrostatic storage.

e. Initial Region: The first region to be executed is the region specified by the entry in 6 row right. The method of implementing this execution is described in paragraph 6 and 8.

5. Pact Region Origin Control Card (Subsequent Control Cards).



a. Code (-01): Pact Region origin cards have code (-01)

b. Origin: Specifies the location of the first table entry on the card relative to the location of the card symbol in the table.

c. Symbol: Card symbol.

d. Check Sum: The check sum is the positive sum of the first 46 words on the card including the sign bit of each half-word as the 18th bit.

The origin of subsequent Pact Region Control Cards is determined in two stages First, the card symbol is compared with the symbols listed in the Region Origin Table (cf. para. 8). The card origin is then determined by adding to the location of the table entry with the same symbol the amount in the origin field (b). There are two common expected types of subsequent Pact Region Control Cards. Additional cards may be needed to handle cases wherein the Region Origin Table exceeds 36 half-words and for which the symbol will be zero and the origin a multiple of 36. It may be desirable to load personal library programs simply by inserting binary cards into a compiled binary deck. In this case, the expected procedure will be to accompany the library program with a control card which has an origin 0, a count of 2, the symbol of the library program for the card symbol, and for the first entry the same symbol and the extent or origin of the program.

6. Transfer Card: At the end of each binary deck compiled by Pact I will be punched a transfer card of the following form:

0	RA	4095 +
1	SA	3 +
2	RA	2 +
3	T	0 +
4	H	0 +
~~5~~	~~H~~	~~8 +~~

The function of this card is the initiate a linkage to the first program region to be executed.

7. The absolute address of an instruction is calculated by adding to the relative address indicated in the instruction the origin of the corresponding region as specified in the region origin table unless the instruction has an index of 7. If the instruction has an index of 7 then the absolute address will be computed by looking up the address in the location in the table indicated by the address of the instruction. Variables or numbers loaded on binary cards will ordinarily have zero indices.

8. Region Origin Table: The region table contains the absolute origin of each region (a library program is treated as a region) referred to by a Pact program together with the region symbol. The symbol occupies the first half-word and the origin with an operation part of +00 occupies the second half-word of the table entry. The entries are loaded from Region Origin Control Cards on which are specified the region symbols together with either the absolute origin desired (with operation part +00) or the extent of the region (with the operation Pact +01). Those regions whose extent only are specified are loaded successively at the upper end of electrostatic storage the first one ending at 4093, the second one immediately preceding the first and so on. The corresponding absolute addresses are placed in the region origin table (with operation part +00). The location of the region origin table is specified on the initial region origin

control card. In location 4094 is loaded the location of the Region Origin Table and in 4095 is loaded the location of the first region to be executed. If, on a relative binary card, reference is made to a region which has not as yet been defined by a control card, an error stop occurs.

9. Standard arrangement of Electrostatic Storage and the standard initial control card: During compilation the extents of several regions are compared and the initial control card is punched as follows:

a. Library Temporary Storage: The library temporary storage is assigned an origin of 6 by the compiler.

b. Region Origin Table Location: The maximum library temporary storage extent plus 6 is compared with 48 and the larger is assigned by the compiler as the origin of the Region Origin Table. This is in order to prevent destruction of any of the table by card images.

c. Initial Region Origin: The first instruction region compiled is assigned by the compiler an origin at the terminus of the Region Origin Table.

d. Variable Extent: The variable extent and the temporary extent are examined by the compiler and if necessary the variable extent is increased so that the temporary region and variable region will ordinarily completely overlap the portion of the load program necessary to complete the loading.

It should be pointed out that this is the function of the compiler only and does not affect the normal execution of the loader.

APPENDIX 1

Working Committee

Rand Corporation, Santa Monica, California

W. S. Melahn Chairman, November 15, 1954 to May 17, 1955	Loop expansion
J. I. Schwartz	Loop expansion
G. Hempstead	Loop expansion
I. Greenwald	Pact Primer
J. Derr	Variable storage, array storage and number assignment

Naval Ordnance Test Station, China Lake, California

B. G. Oldfield	Operation expansion
R. C. Miller, Jr. Chairman, May 17, 1955 to date	Operation expansion
R. G. Selfridge	Operation expansion

Douglas Aircraft Company, Inc., Santa Monica, California

C. L. Baker	Card reading, final listing

Douglas Aircraft Company, Inc., El Segundo, California

H. G. Martin	Basic linkage and calling sequence

Douglas Aircraft Company, Inc., Long Beach, California

R. P. Bacon November 15, 1954 to June 1955	Subroutine operations
T. Littlejohn June 1955 to date	Subroutine operations

Lockheed Aircraft Corp., Burbank, California

R. C. Luke	Variable storage, array storage and number assignment

North American Aviation Inc., Los Angeles, California

O. R. Mock	Duplicate expansion, temporary storage, final assembly and punching
F. R. Anderson	Manual