Octonary Programming System

F. E. Johnston, Jr.

While you are here in Poughkeepsie we would like each of you to participate in writing a program for the 701 and to have a chance to try this program on the machine.

In designing this machine we planned that no one would ordinarily operate it without the help of one or more utility programs. For this reason, you will need certain of these programs to run your problem on the 701. In order that you may get a problem going on the machine in the short time available, we will provide you with some thoroughly tested programs.

Today, I will explain a system which has been found very satisfactory here at the laboratory in training new personnel to operate the machine. This system is just one of the many techniques possible to enter and execute programs. Today, just as an introduction to the operation of the machine, you will write a program to be used with this set of utility programs. Later this week you will have an opportunity to try on the machine any systems or programs you may have.

In the folder on your desks you will find a set of the utility programs you will be using today. You will notice that the first card identifies the program in each section. This identification refers to the actual program plus its loading card. This probably is not clear at this moment, so let us examine the contents of one of the sections. FEJ 019, binary punching program, is a good example. You will notice that the first card of the program is identified with FEJ 015, and the other cards are FEJ 019. 15 in this case is the card that stores 19 in Memory and checks to see that it was loaded correctly in Memory. FEJ 015 is used as the loading card for many programs and thus carries its own number, but when I refer to the punching program, I am including the binary loading program.

Next, let us see the type of cards on which your program will first appear. You will find such cards in the section labeled "Octonary Cards." The first number on the card, 1002, is the location in Memory at which the instruction is to be stored. The sign plus the next six numbers is the instruction. The +11 is the operation part, and the 1030 is the address portion. This indicates that the instruction, add half-word located at octal address 1030, is to appear in electrostatic memory at octal address 1002. All of these numbers are in octonary. Your program will be punched on octonary cards such as these with one instruction per card. The cards in this section are a program ready to load.

Next, let us look at the cards in the section marked "MMA 011," octonary loading program. This is the first of the utility programs I will describe. If you place this deck of cards on top of the octonary program cards and place the complete deck in the card reader, and press the "Load" button, your program will be loaded in the specified addresses. The octal loading program brings in one card at a time, translates the octonary number to a binary number and stores it in Electrostatic Memory at the specified address.

This program is one of the many programs that depend on the convention of assigning the first 200 half-words in Memory as a working location for utility programs. Systems have been devised for using the machine without having a section of Memory set aside, but the system I shall explain will use the concept of a utility program section. This means that when using these programs the available addresses are from 200 to 2047, since the engineering model has only one Electrostatic Memory frame. These addresses are 310 to 3777 in octonary numbers.

note

I will now discuss the procedures which you might wish to use, and at the same time give a brief explanation of the function of each utility program. The first procedure is on Page 3.

Purpose:

To load a program which is punched in octonary cards and to execute the program immediately after loading.

Arrange programs in this order:

Octonary loading program: MMA 011
Octonary program cards.
Control card.
Three blank cards.

Operation:

Place the complete deck of cards in the card reader and press the "Load" button.

Description:

The octonary loading program will read in the octonary cards one at a time, translate the octonary numbers into binary numbers and store in Memory at the location punched in the card. The control card is another octonary card with the following information, 0021+01----. The blank spaces are to be filled in with the location of the first instruction to be executed after the octonary cards have been loaded.

Purpose:

A program punched in octonary is quite bulky and requires quite a bit of time to read in. For this reason, it is generally desirable to punch the program on binary cards with 44 instructions on a card instead of just one. This will mean that instructions will be read at the rate of 6600 per minute instead of 150. The purpose of this procedure is to punch these binary cards from octonary cards.

Arrange programs in this order:

Octonary loading program: MMA 011
Octonary program cards.
Transition program: NR9 008
Binary punching programs: FEJ 019
Control card.
Three blank cards.

Operation:

Place the complete deck of cards in the card reader and press the "Load" button.

Description:

The octonary loading program will store the instructions from the octonary cards. After all of the octonary cards have been read, the punching program will then punch the binary cards. These two programs could be fed separately into the card reader by pressing the "Load" button again. The pressing of the "Load" button in this case is effectively done by the transition program. The 701 is 60,000 times faster than a person in doing arithmetic and could, undoubtedly, bring in the next program much, much faster than an operator.

The control card which follows the punching program is a binary card which contains the location of the first instruction to be punched and the location of the last instruction to be punched. The location of the first instruction to be punched is in the 9 row in columns 15 through 26, and the location of the last instruction to be punched is in the 9 row in columns 33 through 44. The location of the first instruction must be even and the location of the last instruction must be odd.

Purpose:

This procedure is used to load a program which is in the form of binary cards and to execute the program.

Arrange the programs in this order:

Binary loading program: FEJ 015
Binary program cards.
Control cards.
Three blank cards.

Operation:

Place the complete deck of cards in the card reader and press the "Load" button.

Description:

The binary loading program will load into memory the program following it. Each card is checked as it is loaded to see that each instruction is stored at the correct location and that it has not been altered during loading. The control card will then indicate where the first instruction to be executed is located. This is another binary card with the location in the 9 row in columns 69 through 80. There must also be a punch in the 9 row in column 63.

Purpose:

The purpose of this procedure is to load a program which is in binary form and then execute one instruction at a time, and after each instruction print the contents of the different registers. This is one form of tracing and could be used to find faults in the program being traced.

Arrange programs in this order:

Binary loading program: FEJ 015
Binary program cards.
Transition program: NR9 007
Tracing program: JHH 015
Control card.
Three blank cards.

Operation:

Place the complete deck of cards in the card reader and press the "Load" button.

Description:

The binary loading program will load and check the binary program cards. The transition program will then cause the tracing program to load itself and execute. The control card following the tracing program is a binary card which tells the tracing program where to strart tracing. The location of the first instruction to be executed and traced is punched in the 9 row in columns 15 through 26 of the control card.

Purpose:

You might wish to change a few instructions in a program and then trace it. For this procedure you will punch the new instructions in octonary cards and use these as correction cards.

Arrange programs in this order:

Binary loading program: FEJ 015
Binary program cards.
Transition program: NR9 007
Octonary loading program: MMA 011
Octonary correction cards.
Transition program: NR9 008
Tracing program: JHH 015
Control card.
Three blank cards.

Operation:

Place the complete deck of cards in the card reader and press the "Load" button.

Description:

The binary loading program will load and check the binary cards. The transition program will then cause the next program to proceed. After the octonary loading program has loaded all of the octonary correction cards the next transition program will cause the tracing program to be loaded. The tracing program will read its control card and start tracing at the address indicated.

Purpose:

After finding the errors in a program you would next wish to obtain corrected binary cards. It would first be necessary to punch these corrections on octonary cards.

Arrange programs in this order:

Binary loading program: FEJ 015
Binary program cards.
Transition program: NR9 007
Octonary loading program: MMA 011
Octonary correction cards.
Transition program: NR9 008
Binary punching program: FEJ 019
Control card.
Three blank cards.

Operation:

Place the complete deck of cards in the card reader and press the "Load" button.

Description:

The binary loading program will load and check the binary program cards. The transition program will automatically transfer control to the octonary loading program. The correction cards will then be loaded in the proper place. Directly following the correction cards is another transition program which will cause the binary punching program to be loaded. The punching program will read the control card and punch the required instructions.

Another useful utility program which has not been mentioned thus far, and one which you will probably want to use today, is WHT016, octonary printing program. This program will print out any portion of memory above the utility section. The printing program is followed by a binary control card which tells the program where to start printing, and where to stop. This program may be used in sequence with other programs by the same method used to operate the punching program. The same type of control card is used for both the printing and the punching programs.

I have described all of these programs in terms of octonary cards and octonary numbers. If decimal cards are to be used, MMA011 and NR9008 are replaced by MMA000 and NR9010. The tracing program prints all of its information in octonary numbers. It is possible to write a tracing program to print in decimal but the octonary system has proved to be more convenient for the engineers and operators here at the laboratory. You will find that it will be very useful to have an octonary listing of your program, although it might originally have been written in decimal.

Utility Programs

Binary loading program:	FEJ015
Binary loading program:	FEJ018
Binary punching program:	FEJ019
Binary transition program:	NR9007
Octonary loading program:	MMA011
Octonary printing program:	WHT016
Octonary transition program:	NR9008
Octonary tracing program:	JHH015
Decimal loading program:	MMA000
Decimal transition program:	NR9010

FEJ015
Binary Loading

Function

Loading binary program cards containing a check sum and loading address.

Will load any binary card punched by FEJ020, FEJ019 or NR9003 at locations specified on each card, and check that the loading was successful.

Will load any number of instructions at any designated addresses above $(0133)_8$.

Storage

Electrostatic Memory $(0000\text{-}0133)_8$
$(0000\text{-}0091)_{10}$.

Program Stops

a) $(0003)_8$ - with copy check light on (providing no control card was used), indicates program cards successfully loaded.

b) $(0042)_8$ - indicates check sum error in card just read; reload, and if still not successful, check for errors in card, or machine errors.

Program Control

Program Loading:

1) Place following deck in hopper:
   a) FEJ015 (1 self loading binary card).
   b) Binary Program cards.
   c) Control Card (binary)
   d) 2 blank cards.

2) Depress Load button.

Control Card

The control card indicates the location to which control is to be transferred after successful loading. It is punched in the 9 row as follows:

a) A punch in column 63,
b) The location to which control is to be transferred is punched in binary in columns 69-80.

Comments

If the control card and blank cards are omitted, the End of File signal will terminate loading as indicated in Stop (a) above.

FEJ017
Binary Punching

Function

Punching Binary Cards of the type to be loaded by FEJ015 from octonary or decimal program cards. Also prints, in octonary, the instructions being punched.

Program Stops

a) $(1310 \text{ or } 1326)_8$ - final stop, program executed correctly.
b) $(1136)_8$ - error in Sign rows, 11 and 12, in card just read.
c) $(1144)_8$ - error in digit rows, 0-9, in card just read.

Control Panels

Standard Printer Board.
Standard Punch Board.

Program Loading

1) Place the following deck in hopper:

   a) FEJ015
   b) FEJ017 (11 binary cards)
   c) Octonary or decimal program cards

2) Depress Load button

Output

Printed Results - Same form as WHT016.

Comments

FEJ017 is the same as FEJ020 (see comments under FEJ020), except that there are no restrictions as to minimum number of instructions per binary card or as to starting address of each binary card. Thus, the binary cards may not be loadable with FEJ018.

FEJ018
Binary Loading

Function

Loading Binary Cards (see General Information) - of the type punched by FEJ020 and FEJ019. It is a special case of FEJ015, for it will only load an even number of instructions per card, and the first instruction of each card must go to an even address.

Storage

Electrostatic Memory - $(0000\text{-}0064)_8$
$(0000\text{-}0052)_{10}$

Program Stops

a) $(0014)_8$ - with copy check light on (providing no control card was used) indicates program cards successfully loaded.

b) $(0033)_8$ - indicates check sum error in card just read; reload, and if still not successful check for error in card.

Program Control

Program Loading

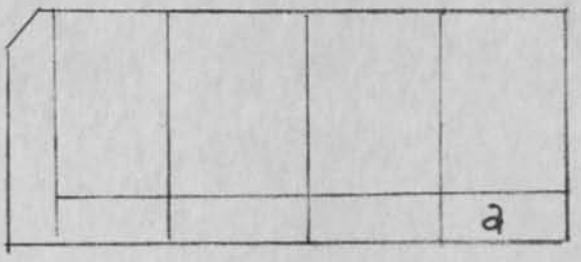
1) Place the following deck in hopper:
   a) FEJ018 (1 self loading binary card)
   b) Binary program cards
   c) Control Card
   d) 2 blank cards

2) Depress Load button

Control Card

The control card indicates the location to which control is to be transferred after successful loading by the following punching in the 9 row:

a) A punch in column 63.
Columns 69-80 - location, in binary, to which control is to be transferred.



Comments

If the control card and 2 blank cards are omitted, the loading will terminate as in Stop (a) above. The card reader START button will have to be depressed when all but the last 2 cards have been read.

FEJ019
Punch Memory

Function

Punching out half-words in binary from any region of memory above $(0107)_8$ and load a following self-loading binary card. The binary cards are of the type to be loaded by FEJ015 or FEJ018.

Storage

Electrostatic Memory $(0000\text{-}0115)_8$
$(0000\text{-}0071)_{10}$

Control Panels

Standard punch board.

Program Control

Program Loading

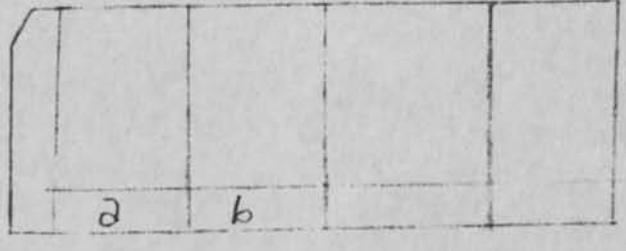
1) Place the following deck in hopper:
   a) FEJ019 (2 self-loading binary cards).
   b) Control Card
   c) Self-loading program (like MMA011 or FEJ015) or 3 blank cards for stop.

2) Depress load button.

Control Card

The following data is punched in binary in the 9 row:

a) Columns 15-26 - memory location (must be even) of first instruction to be punched.
b) Columns 33-44 - memory location (must be odd) of last instruction to be punched.



FEJ020
Binary Punching

Function

Punching Binary Program Cards of the type to be loaded by FEJ015 or FEJ018 from Octonary or Decimal Instruction Cards. Also to print in Octonary, the instructions being read.

Program Stops

a) $(1310 \text{ or } 1326)_8$ - final stop, program executed correctly.
b) $(1136)_8$ - Error in Sign rows 11-12 from card just read.
c) $(1144)_8$ - Error in digit rows, 0-9, from card just read.

Control Panels

Standard Printer Board.
Standard Punch Board.

Program Control

Program Loading

1) Place following deck in hopper:
   a) FEJ015
   b) FEJ020 (12 binary cards)
   c) Octonary or decimal instruction cards.
2) Depress Load button.
3) When card reader stops with SELECT light on, press card reader START button to cause last 2 cards to be read.

Printed Results

The instructions and their locations from each binary card punched are also printed in the same form as used by WHT016, the Print Memory program.

Comments

The instruction cards read by FEJ020 are <u>almost</u> identical to those read by MMA011 and MMA000, Instruction Loading programs:

Columns 10-13; Location of Instruction
Column 14; Sign of Instruction (11 punch for –, 12 for +)
Columns 15-16 Operation Part of Instruction
Columns 17-20 Address Part of Instruction
Columns 21-26 Blank.

If the information is in octonary, columns 9 must be left blank; if in decimal, a 9 punch is put in col. 9. This 9 punch will make these cards unreadable by MMA000.

Forty-four instructions may be placed on one binary card. The Instruction cards must be ordered in an ascending sequence of adjacent instruction locations. Whenever 44 cards are read or the sequence is broken, a binary card is punched and the information printed.

The number of instructions per binary card must be even, and the location of the first instruction on each card must be an even number.

JHH015
Tracing

Function

Printing (in octonary-see attached example) the step-by-step execution of each instruction of any program.

Storage

Electrostatic Memory $(0000\text{-}0199)_{10}$
$(0000\text{-}0307)_8$

Program Stops

a) COPY CHECK- should indicate that a COPY instruction is being traced.

If the COPY instruction is meant to copy information into the machine, the operator may load this information manually at the Operator's Panel.

Depress Start button to continue.

b) Program Stop indicated at $(0146)_8$

This will occur if Stop instruction is given in Traced program. Depress Start button to continue.

Control Panel

Tracing Board in Printer. Alteration switches off.

Program Control:

Program Loading:

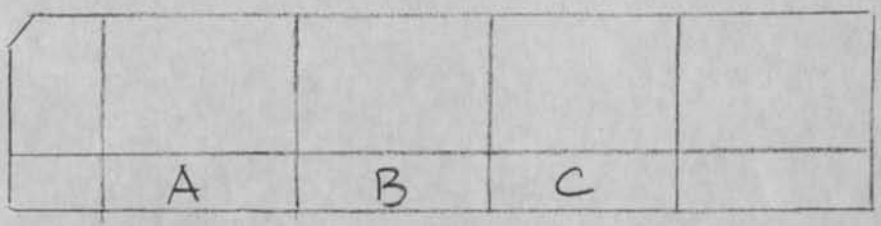
1) Place the following in card feed hopper:
   a) FEJ015 Binary Loading Program
   b) Binary Program to be traced
   c) NR9008 Transition
   d) JHH015 Tracing (6 binary cards)
   e) Control Card (binary)
   f) 3 Blank cards.

2) Depress Load button.

Control Card

1) To begin tracing at once; punch in binary, in columns 15-26, row 9, the address at which to begin tracing (A).

or 2) To begin tracing anywhere in the program after part of the program has been executed, punch in binary in columns 33-44, row 9, the address at which to begin tracing (B); Columns 51-62, row 9, the address at which to begin the program being traced (C).



Example of one line of Printed Report

```
1105  -  12 1004  1  -0000370,000000   000000,000000   -000370,000000
 1    2  3  4     5  6 7       8           9           10          11          12
```

1. Location of Instruction
2. Sign of Instruction
3. Operation part of Instruction
4. Address part of Instruction
5. Overflow indicator (1=on, 0=off).
6. Sign of Accumulator
7. Contents of Accumulator positions p and q.
8. Contents of Accumulator positions 1 to 35.
9. Sign of MQ.
10. Contents of MQ.
11. Sign of contents of memory location corresponding to 4.
12. Contents of memory location corresponding to 4.

KYS003
Relocation

Function

Relocating the memory location of a program either downward to lower address regions (not occupied by KYS003) or upward to higher address regions, provided that the relocated program does not overlap the original program.

Storage

Electrostatic Memory $(0066\text{-}0155)_8$
$(0054\text{-}0109)_{10}$

Program Control

Program Loading

1) Place following deck in hopper
   a) FEJ015
   b) KYS003 (2 binary cards)
   c) Control Cards
   d) 2 blank cards

2) Press LOAD button

Control Cards

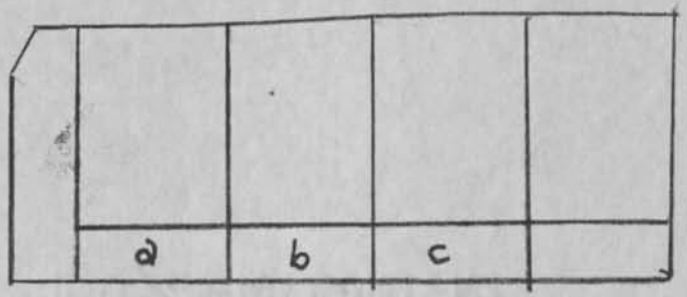
Type 1) For regions containing only instructions and constants which relocate like instructions (constants whose address parts refer to memory locations), punch in binary in the 9's row:

a) Columns 15-26 - location of first instruction of region.
b) Columns 33-44 - location of last instruction of region.
c) Column 45 a punch indicates relocation downward; no punch, upward.
Column 51-62 - distance to move.

Type 2) For regions containing only constants which must not be changed when relocated, punch in binary in the 9's row:

Same as for type 1, except add a punch in column 27.

Type 3) Final transfer card: Punch in binary in a—9's row in columns 15-26 - the location to which control should be transferred, and add a punch in column 9.



Comments

Instructions whose address parts refer to memory locations and are also used a constants cannot be relocated. KYS003 can be used to relocate a program downward, even if the relocated program overlaps the original, if the control cards are read in ascending order of location of the first instruction. The program being reloaded is assumed to be already in memory.

MMA011
Octonary Loading

Function

Loading a deck of octonary cards. Each card contains one half-word and its location, punched in the octonary system. MMA011 translates the half-word and stores it in memory at the indicated location.

Storage

Electrostatic Memory $(0000\text{-}0155)_8$
$(0000\text{-}0109)_{10}$.

Program Input

Octonary cards are punched in the following manner.

not 0 → Column 9 Blank
10-13 Location of Instruction
14 Sign of instruction (11 punch for −, 12 for +)
15-16 Operation of part of instruction
17-20 Address part of instruction
21-26 Blank

Program Stops

a) $(0102)_8$ - error in punched card just read, columns 9-26 rows 0-9 (digit rows).

b) $(0122)_8$ - error in punched card just read, columns 9-26, rows 11 and 12 (sign rows).

If either one of the above program stops occur:

Check card just read for error. If there is an error, correct and continue loading by replacing remaining unread deck in card reader and depress start button on calculator.

Program Control

Program Loading
1) Place the following deck in hopper:
   a) MMA011 (3 self loading binary cards).

b) Octonary Programs cards.
c) Control Card (octonary).
d) 3 Blank Cards

2) Depress Load button

Control card 0021+01 (xxxx)-starting location in loaded program. Punched in columns 10-20 like octonary card.

Comments

The control card indicated above is a regular octonary card which provides a transfer instruction to be placed at (0021)$_8$. This interrupts MMA011 and transfers control to the address punched as the address part of the transfer instruction.

NR9003
Assembly

Function

The purpose of NR9003 is to accept one or more programs in symbolic form and to combine them into one program in actual form. The symbolic program is presented with one instruction punched to a card, and the calculator produces the program on binary cards ready to be loaded with FEJ015 (binary loading program which can load an odd or even number of instructions from a single card). The calculator also produces a printed program which contains the original symbolic program, the new actual program (in octonary) and the comments which were associated with the original program.

Storage

Occupies all of electrostatic memory except addresses 0000 to $(0307)_8$.

Notes and Precautions

Locations must be given in symbolic form. Address parts may be given in symbolic form or as actual 4-digit decimal numbers. Operation parts are to be given in octonary.

A symbolic address is a 6-digit number. The only two important precautions are:

1) Do not use 000 000 as a symbolic address with 00 as a heading.

2) Do not use the same symbolic address with the same heading for two different purposes.

Programs must be broken down into batches of instructions so that the number of synonyms plus half the number of drum assignments plus half the maximum number of undetermined addresses plus the number of instructions in the batch does not exceed 128.

The layout of cards is given in Figure 1.

The cards must be placed in the following order:

1) Synonyms
2) Drum assignments
3) First electrostatic assignment
4) First heading card
5) Instructions of first batch in the correct order for assignment of locations.

6) Other heading cards and their instructions
7) Last heading card.

Comments may be interspersed at will.

Subsequent electrostatic assignment cards may be used at will and each gives the location of the next instruction. Each electrostatic assignment card must be accompanied by a heading card.

When one is combining several different programs, a different 2-digit prefix must be used with each different program to avoid homonyms. The various heading cards which break up a single program into manageable batches may all have the same prefix but this is not necessary. The last card must be a heading but this is a dummy. The value of the prefix on this last card is irrelevant.

Synonym cards are used to establish the connections between independent programs which are being combined by the assembly program.

Program Stops

a) $(1161)_8$ or $(1306)_8$ During card reading the calculator discovered a batch which was too large. Relocate heading cards and start.

b) $(0706)_8$ During printing, same as a.

c) (0712_8) or (1506_8) with tape check. Depress Start button.

d) (0570_8) card error, check card just read. Reload

Control Panel

NR9003 Print Board.
72-72 Punch Board

Program Loading

1) Place the assembly program NR9003 in the card hopper.
2) Place the program to be assembled in the hopper on top of NR9003.
3) Press the card reader START until it is READY.
4) Printer READY with NR9003 plugboard.
5) Card Punch READY with binary cards.
6) Tape 0400 READY
7) Tape 0401 READY

8) Preset RESET AND CLEAR MEMORY.
9) Press LOAD.
10) When the Card Reader stops with SELECT, press the Card Reader START.

The calculator will automatically figure out the actual program, print it, and punch it on binary cards.

Example of Printed Report

Location	Address Part	Location	Instruction	Comment
02.03.00	04.04.00	7742	+ 11 7777	
02.04.00	02.10.00	7743	+ 15 7751	Store Location of Number of Words
02.05.00	03.10.00	7744	+ 15 7773	Store Loc of End of File Procedure

Fig. 2

	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
symbolic instruction	+	0	1	0	2	0	0	location						s	o. p.		address part									
actual instruction	+	0	1	0	2	0	1	location						s	o. p.		addr. part									
	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
synonym	+	0	1	0	2	0	2	use this addr. 2									do not use this 1						p2		p1	
drum assignment	+	0	1	0	2	0	3	symbolic address									actual address						p			
	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
comment	+	0	1	0	2	0	4																			
heading	+	0	1	0	2	0	5																p			
electrostatic assign.	+	0	1	0	2	0	9										address									
	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

These are prefixes.

note: columns 45 - 80 are for comment in any card

note: columns 1 - 8 are for identification in any card

note: the directly executed instructions 01.0206, 01.0207, and 01.0208 are spares

note: actual addresses are decimal while operation parts are octonary

Figure 1

CARD LAYOUT

NR9007
Transition

Function

To terminate the loading of binary cards by FEJ015 or FEJ018 and to load following non-self-loading binary cards.

Storage

Electrostatic Memory: $(0304\text{-}0307)_8$
$(0196\text{-}0199)_{10}$

Program Loading (hypothetical case)

1) Place the following deck in hopper.

   a) FEJ015 or FEJ018
   b) Binary program Cards.
   c) NR9007 (2 binary cards)
   d) (MMA011 followed by Octonary program cards), or some other self-loading program.

2) Depress Load button

Comments

NR9007 is a transition program to facilitate automatic operation.

NR9008
Transition

Function

To terminate loading of Instruction cards by MMA011 and load a following self-loading binary. card.

Storage

Electrostatic Memory $(0305\text{-}0307)_8$
$(0197\text{-}0199)_{10}$

Program Loading (hypothetical case)

1) Place following deck in hopper:

   a) MMA011
   b) Octonary Instruction cards
   c) NR9008 (4 Octonary cards).
   d) FEJ015 (or other binary self-loading program)

2) Depress Load button

Comments

NR9008 is a transition program to facilitate automatic operation.

WHT011
Clear Memory

Function

Storing zeros at all electrostatic memory locations.

Program Loading

1) Place following deck in hopper:
   a) WHT011 (1 self loading binary card).
   b) Self-loading binary card, or 3 blank cards.

2) Depress the LOAD button

Comments

WHT011 terminates with the programmed equivalent of pressing the LOAD button.

WHT014
Print Tape

Function

Printing (in octonary) any desired unit record (not more than 910 full words long) from any tape unit.

Storage

Electrostatic Memory $(3434\text{-}3776)_8$
$(1820\text{-}2046)_{10}$

Program Stop

a) $(3717)_8$ - last control card read, Program is complete.
b) $(3742)_8$ - end of fill on tape has been reached. Depressing start button will read next control card and continue. This next card can apply to the same tape, or to another one as desired.
c) $(3741)_8$ - Tape check; depress start button to read record again and continue.

Control Panel

Standard print board

Program Control

Program Loading

1) Place following deck in hopper
   a) FEJ015
   b) WHT014 (6 binary cards)
   c) Control cards

2) Depress Load Button.
   When card reader stops with SELECT light on, depress card reader START to allow last 2 cards to be read.

Control Card

Punch in binary in the 9 row:
a) Columns 10-26 - number of first record to be printed, considering records to be serially numbered from start of tape.

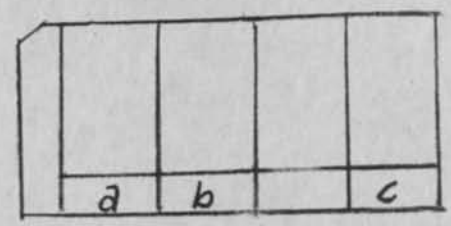
b) Columns 28-44 - number of consecutive records to be printed.

c) Columns 69- 80 - Tape identification

If entire tape is to be printed:

Columns 10-26;$(000001)_8$

Columns 28-44;$(777777)_8$ - or some number known to be greater than or equal to the number of records.



Output

Printed Results : (4 full words per line)

+351246 741002 -156207 376012 +001001 716003 -0156371 123737

Comments

More than one control card may be used - program will continue to read control cards after depressing Start button when Program Stop $(3742)_8$ is indicated.

WHT015
Print Drum

Function

Printing (in octonary) the contents of any desired sequence of locations from a magnetic drum.

Storage

Electrostatic Memory $(3465\text{-}3777)_8$

$(1845\text{-}2047)_{10}$

Program Stops

$(3744)_8$ - end of file (last card read).

Control Panels

Standard printer board.

Program Control

Program Loading

1) Place the following deck in hopper:
   a) FEJ015
   b) WHT015
   c) One or more Control Cards

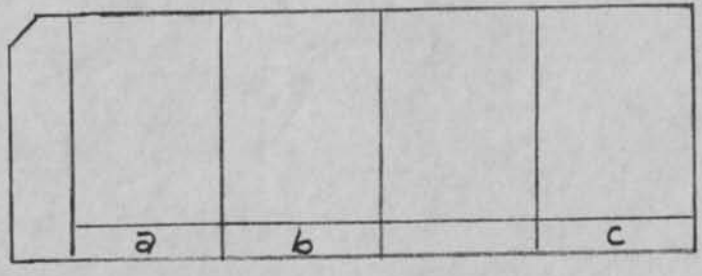
2) Depress Load button.

3) When card reader stops with SELECT light on, depress card reader START button to cause last 2 cards to be read.

Control Card

Punch in binary in row 9:

a) Columns 15-26, drum location of first word to be printed.
b) Columns 33-44, drum location of last word to be printed.
c) Columns 69-80, Identification of drum to read from.



Comments

More than $(922)_{10}$ full words must not be printed with one control card. In case the record desired exceeds this length, more than one control card must be used.

As an example, suppose that the entire contents of drum 1 are to be printed out. The 3 control cards could be as shown below:

Columns	15-26	33-44	69-80
1.	$(0000)_8$	$(3462)_8$	$(0200)_8$
2.	$(3464)_8$	$(7144)_8$	$(0200)_8$
3.	$(7146)_8$	$(7776)_8$	$(0200)_8$

WHT016
Print Memory in Octonary

Function

Printing all or any part of Electrostatic Memory contents above $(0255)_8$ as octonary half-words with their locations.

Storage

Electrostatic Memory: $(0000\text{-}0255)_8$
$(0000\text{-}0173)_{10}$

Control Panels

Standard print board.

Program Control

Program Loading:

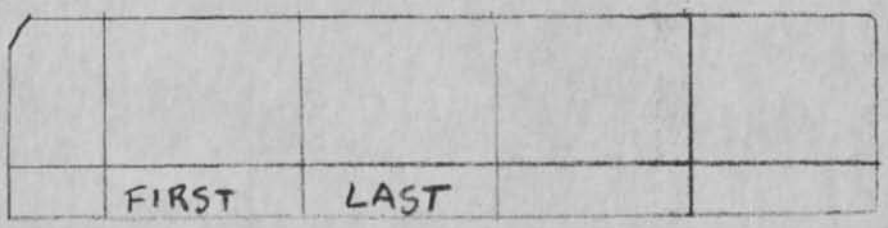
1) Place following deck in hopper:
   a) FEJ015
   b) WHT016 (4 binary cards)
   c) Control Card (binary)
   d) Next self-loading program - or 3 blank cards.

2) Depress Load button

Control Card

Columns 15-26, row 9 - memory location, in binary, of first half-word to be printed.

Columns 33-44, row 9 - memory location in binary of last half-word.



Printed Output

4 half-words with locations in octonary on each line as follows:

Loc.	Contents	Loc.	Contents	Loc.	Contents	Loc.	Contents
0060	+304000	0061	-370206	0062	+120206	0063	+150101

Comments

The program terminates by executing the programmed equivalent of pressing the LOAD button, so that a following self-loading card will be loaded. If 3 blank cards follow the control card, the machine will read the first blank card and stop at 0000.

Binary Loading Program

FEJ 015

This program will store instructions appearing on binary cards of the type shown. Instructions are from left to right starting in the 8 row directly above block 'A.' The instructions are checked during loading to see that each is stored in the proper location and has not been changed during loading. If a card does not load correctly the program will stop at octal address 0042. If the start button is pressed after this stop the next card will be read, thus one may reload the card that was in error.

F. E. Johnston

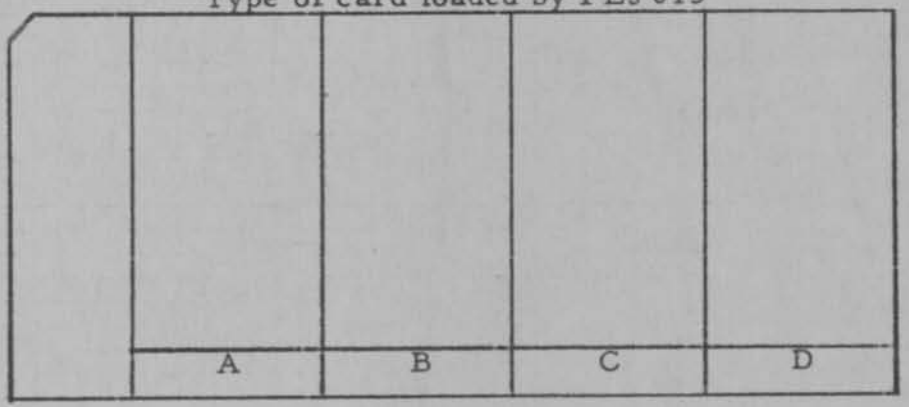
August 20, 1952

IBM

FEJ015

DATE 8-20-52
DRAWN B.
CHECKED

Type of card loaded by FEJ015



A & B = Check Sum = -(Twice the sum of all other half words with the sign bit figured as just another binary bit.)

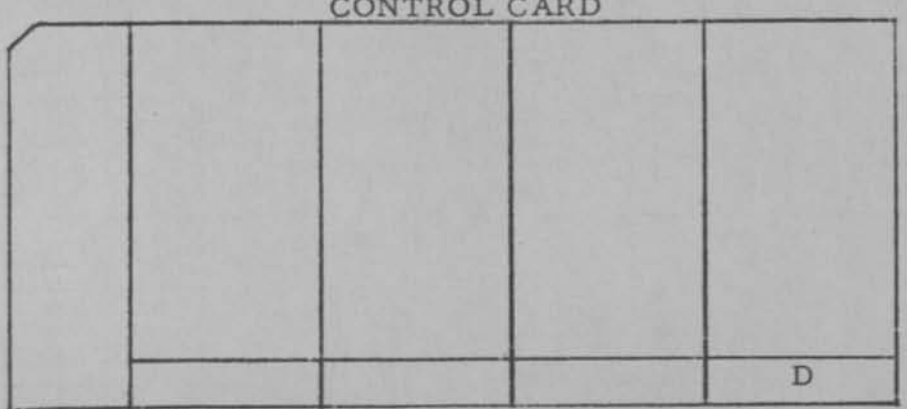
D = First address of this card. Even or odd.

C = Number of instructions on this card. Even or odd.

If no control card is used, program will stop with copy check after last card read in. If a transition program is used the control card is omitted.

If control card is placed at end of deck being loaded, program will transfer to address indicated in space D as a minus address. All punchings except those in block D are ignored.

CONTROL CARD



IBM

FEJ 015

DATE 8-14-52
WRITTEN 79.
CHECKED

Binary Loading Program

LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART				
0000	-	Copy	0002		0000	-37	0002
0001		R Add	0003	read in	0001	12	0003
0002		Add	0000	next card	0002	11	0000
0003	-	Copy	(0004)		0003	-37	0004
0004		Store A	0003		0004	15	0003
0005		TR	0002		0005	01	0002
1.0		R Add	0057	Add First Adr	0006	12	0057
.1		TR +	1.3	+ No. of Instrs	0007	03	0011
.2		Store A	1.4	into check	0010	15	0012
.3		Add	0056	Sum.	0011	11	0056
.4		TR	(2.10)		0012	01	0025
2.0		R Add	0056	All Instrs ?	0013	12	0056
.1		TR 0	3.0	Yes →	0014	04	0040
.2		Sub	0051	← No	0015	05	0051
.3		Store	0056		0016	14	0056
.4		R Add	(0000)	Store	0017	12	0000
.5		Store	(0000)	instruction	0020	14	0000
.6		R Add	2.4	Advance	0021	12	0017
.7		Add	0051	Adr part of	0022	11	0051
.8		Store	2.4	2.4 by 1	0023	14	0017
.9		R Add	(0000)		0024	12	0000
.10		Store	0053	Add	0025	14	0053
.11	-	R Add	0052	Instruction	0026	-12	0052
.12		Acc Lt.	0001	into	0027	26	0001
.13	-	Add	0054	check	0030	-11	0054
.14	-	Store	0054	Sum	0031	-14	0054
.15		R Add	0057		0032	12	0057
.16		Store A	2.5	Set up	0033	15	0020
.17		Store A	2.9	new location	0034	15	0024
.18		Add	0051		0035	11	0051
.19		Store	0057		0036	14	0057
.20		TR	2.0		0037	01	0013
3.0	-	R Add	0054	Did card	0040	-12	0054
.1		TR 0	4.0	Load Correctly	0041	04	0043
.2		Stop	4.0	← No ↓ Yes	0042	00	0043
4.0		R Add	2.15	Reset this	0043	12	0032
.1		Add	0051	program	0044	11	0051
.2		Store A	2.4		0045	15	0017
.3		R Add	2.14		0046	12	0031
.4		Read	4000	Select CR	0047	30	4000
.5		TR	0004		0050	01	0004
0051		00	0001		0051	00	0001
0052		00	0000		0052	00	0000
0053		00	0000		0053	00	0000
0054	-	00	0023	check Sum	0054	-00	0023
0055		33	4242		0055	33	4242
0056		00	0054	No of Instrs	0056	00	0054
0057		00	0000	First Adr.	0057	00	0000

Binary Loading Program

FEJ 018

This program will store instructions appearing on binary cards of the type shown. Instructions are from left to right starting in the 8 row directly above block "A." The instructions are checked during loading to see that each is stored in the proper location and has not been changed during loading. If a card does not load correctly the program will stop at octal address 0033. If the start button is pressed after this stop the next card will be read, thus one may reload the card that was in error.

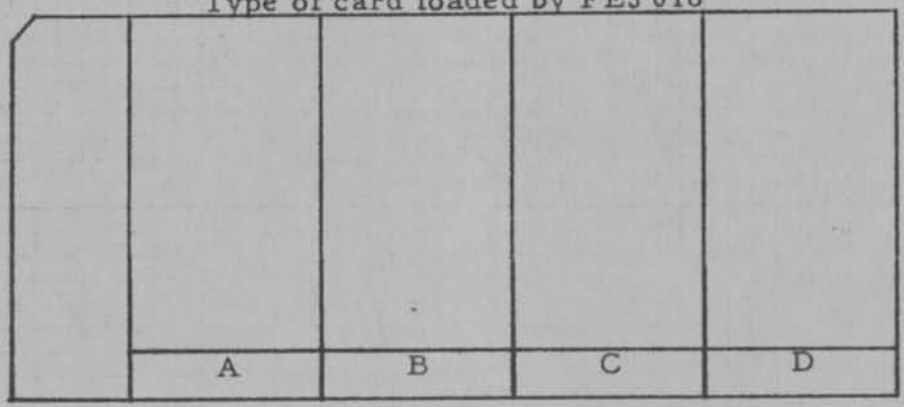
F. E. Johnston

August 20, 1952

FEJ 018

DATE 8-20-52
DRAWN
CHECKED

Type of card loaded by FEJ 018



A & B = Check Sum = -(Twice the sum of all other half words with the sign bit figured as just another binary bit.)

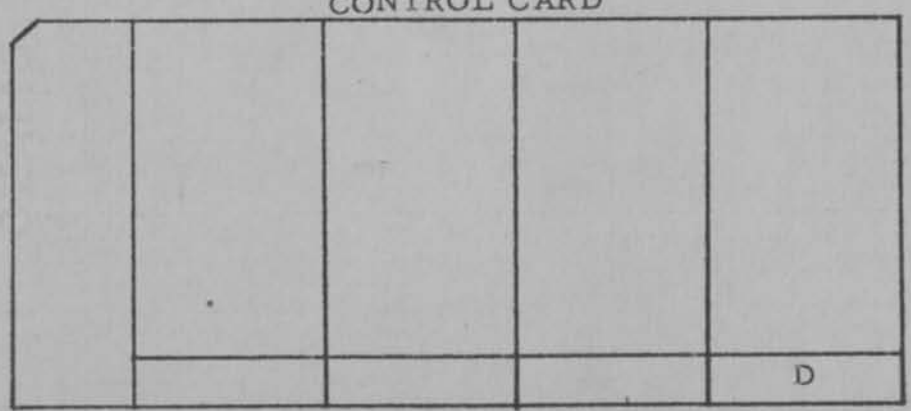
D = First address of this card. Must be even.

C = Number of instructions on this card. Must be even.

If no control card is used, program will stop with copy check after last card read in. If a transition program is used the control card is omitted.

If control card is placed at end of deck being loaded, program will transfer to address indicated in space D as a minus address. All punchings except those in block D are ignored.

CONTROL CARD



IBM

FEJ 018

DATE 8-14-52
WRITTEN FG.
CHECKED

Program to load Binary Cards type 19,20 FEJ

LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART		octonary.		
0000	-	Copy	(0002)	use for comparing	0000	-37	0002
0001		RAdd	0003		0001	12	0003
0002		Add	0000	Bootstrap	0002	11	0000
0003	-	Copy	0004		0003	-37	0004
0004		Store A	0003		0004	15	0003
0005		TR	0001		0005	01	0001
0006		R Add	0013		0006	12	0013
0007		Sub	0016	read in	0007	05	0016
0010		Store A	0013	card after	0010	15	0013
0011		Sub	0000	checking	0011	05	0000
0012		TR 0	2.0	loading	0012	04	0017
0013	-	Copy	(0000)	Program	0013	-37	0000
0014		TR	0006		0014	01	0006
0015		Add Ab.	(0056)	use to compare	0015	13	0056
0016		Stop	0002	← constant	0016	00	0002
2.0		R Add	(0000)	Add up instrs.	0017	12	0000
.1		Store	0055	including sign	0020	14	0055
.2	-	R Add	0054	bit + add	0021	-12	0054
.3	-	Add	0056	to ck sum +	0022	-11	0056
.4	-	Store	0056	store.	0023	-14	0056
.5		R Sub	2.0	Advance Adr.	0024	06	0017
.6		Sub	0005	of 2.0 by 1	0025	05	0005
.7		Store A	2.0	+ compare.	0026	15	0017
.8		Add	0015	← with 0015	0027	11	0015
.9		TR +	2.0	End of card?	0030	03	0017
.10	-	R Add	0056	Does card	0031	-12	0056
.11		TR 0	3.0	check ?	0032	04	0034
.12		Stop	3.0	← No ↓Yes	0033	00	0034
3.0		Read	4000	Read in	0034	30	4000
.1	-	Copy	0060	ck Sum.	0035	-37	0060
.2	-	Copy	0062	No of Instr-First Adr.	0036	-37	0062
.3		R Add	0063	Set to transfer	0037	12	0063
.4		TR +	3.6	out if First Adr.	0040	03	0042
.5		Store A	3.15	is minus.	0041	15	0053
.6		Store A	0013	Set up copy	0042	15	0013
.7		Store A	2.0	starting	0043	15	0017
.8		Add	0062	Adr. + R Add	0044	11	0062
.9		Store A	0000	starting Adr.	0045	15	0000
.10		Store A	0015	+ end adr.	0046	15	0015
.11		Acc Rt.	0021	Add First Adr	0047	27	0021
.12	-	Add	0060	+ No of Instr	0050	-11	0060
.13		Acc Rt.	0001	into ck Sum	0051	27	0001
.14	-	Store	0056	+ store in	0052	-14	0056
.15		TR	0013	0056	0053	01	0013
4.0		00	0000		0054	00	0000
.1		00	0000		0055	00	0000
5.0		check			0056	-00	0015
.1		Sum			0057	-07	5703

Binary Punching Program

FEJ 019

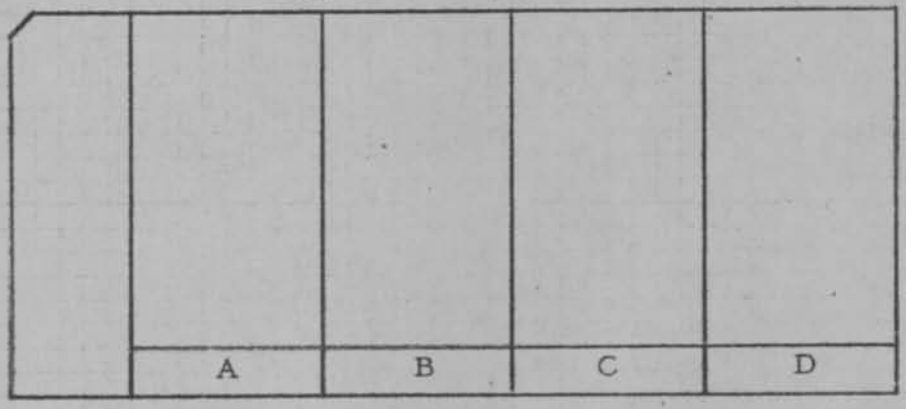
This program will punch the information from electrostatic memory, indicated by a control card, into binary cards of the type shown. Information will appear from left to right starting in the 8 row directly above block "A." The program will punch as many cards as necessary to punch all instructions indicated.

After completing the punching this program effectively presses the "Load" button. If this is the final program of a sequence it should be followed by three blank cards; the program will then stop at octal address 0000. If this is not the final program of a sequence it should be followed by a self-loading card to bring in the next program.

F. E. Johnston

August 12, 1952

Type of card punched

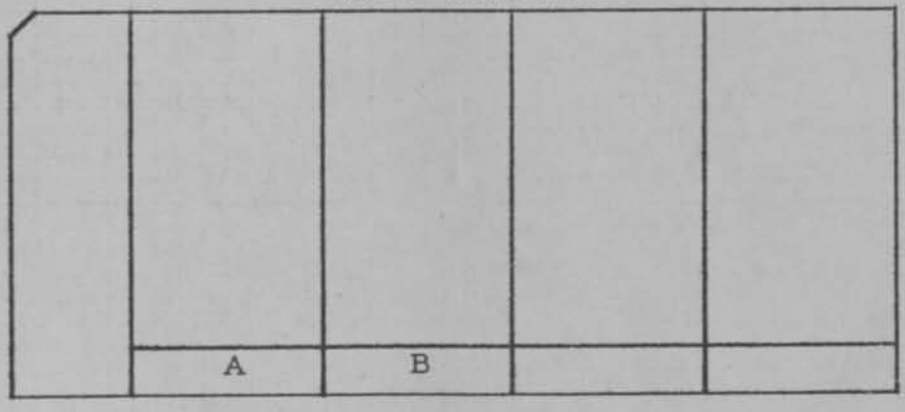


A & B = Check Sum = -(Twice the sum of all other half words with the sign bit figured as just another binary bit.)

D = First address of this card.

C = Number of instructions on this card.

CONTROL CARD



A = Location of first instruction to be punched. Must be even.

B = Location of last instruction to be punched. Must be odd.

IBM

FEJ 019

DATE 8-12-52
WRITTEN
CHECKED

FEJ 018 or

Punch from E.M. binary cards to be loaded with FEJ 015

	LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART				
	0000	-	Copy	0002		0000	-37	0002
	0001		R Add	0003		0001	12	0003
	0002		Add	0000	stor. for	0002	11	0000
	0003	-	Copy	0004	ck sum	0003	-37	0004
	0004		Store A	0003	No of Instr.	0004	15	0003
	0005		TR	0002	First Adr.	0005	01	0002
	0006		Read	4000		0006	30	4000
	0007		R Add	0011	↓ From c.c.	0007	12	0011
	0010		Add	0000	First Adr.	0010	11	0000
	0011	-	Copy	0060	No of Instr.	0011	-37	0060
	0012		Store A	0011		0012	15	0011
	0013		R Add	0011		0013	12	0011
	0014		Add	0017		0014	11	0017
	0015		TR +	0007		0015	03	0007
	0016		TR	1.0		0016	01	0020
	0017		Copy	0122		0017	37	0122
	1.0		Read	4000	read in	0020	30	4000
	.1	-	Copy	0010	controls	0021	-37	0010
	.2		R Add	0011	figure No.	0022	12	0011
	.3		Add	4.1	of instrs	0023	11	0110
	.4		Sub	0010	from Control	0024	05	0010
	.5		Store	0011	Card.	0025	14	0011
	.6		Acc L	0050	clear	0026	26	0050
	.7	-	Store	0002	mem.	0027	-14	0002
	.8	-	Store	0004		0030	-14	0004
	.9		R Add	0010	Get First Adr	0031	12	0010
	.10		Store	0005	& store in	0032	14	0005
	.11		Store A	2.1	working	0033	15	0041
	.12		Store A	3.4	locations	0034	15	0072
	.13		Add	4.2	Advance Adr	0035	11	0111
	.14		Store A	0010	by 44 dec.	0036	15	0010
	.15		Store A	4.3	& detect end of card.	0037	15	0112
	2.0		Write	2000		0040	32	2000
→	.1		R Add	(0000)		0041	12	0000
	.2		Store	4.6	Add instrs	0042	14	0115
	.3	-	R Sub	4.5	& to check	0043	-06	0114
	.4		Acc L	0001	sum.	0044	26	0001
	.5	-	Add	0002		0045	-11	0002
	.6	-	Store	0002		0046	-14	0002
	.7		R Add	0004	count instrs	0047	12	0004
	.8		Add	4.1	& store	0050	11	0110
	.9		Store	0004		0051	14	0004
	.10		R Add	0011	Count down	0052	12	0011
	.11		Sub	4.1	from No	0053	05	0110
	.12		Store	0011	of instructions	0054	14	0011
	.13		TR 0	3.12	All? → Yes	0055	04	0102
	.14		R Sub	2.1	↓ No	0056	06	0041
	.15		Sub	4.1	test if	0057	05	0110
	.16		Store A	2.1	Full card	0060	15	0041
	.17		Add	4.3		0061	11	0112
	.18		TR +	2.1	Yes No → 2.1 ↓	0062	03	0041

IBM

FEJ 019

DATE 8-12-52
WRITTEN
CHECKED

LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART				
2.19		R Sub	0004	Add First	0063	06	0004
.20		Sub	0005	Adr +	0064	05	0005
.21		Acc Rt	0021	No of Instrs	0065	27	0021
.22	-	Add	0002	into check	0066	-11	0002
.23	-	Store	0002	Sum	0067	-14	0002
3.2	-	Copy	0002	ck Sum	0070	-37	0002
.3	-	Copy	0004	No Instrs-First Adr.	0071	-37	0004
.4	-	Copy	(0000)		0072	-37	0000
.5		R Add	3.4		0073	12	0072
.6		Add	0000	Punch	0074	11	0000
.7		Store A	3.4	Next	0075	15	0072
.8		Add	2.1	card.	0076	11	0041
.9		Acc L	0007		0077	26	0007
.10		TR 0	1.6		0100	04	0026
.11		TR	3.4		0101	01	0072
.12		R Add	4.4		0102	12	0113
.13		Store	1.6	set to	0103	14	0026
.14		R Add	2.1	discontinue	0104	12	0041
.15		Add	4.1	punching.	0105	11	0110
.16		Store A	2.1		0106	15	0041
.17		TR	2.19		0107	01	0063
4.1		00	0001		0110	00	0001
.2		00	0054		0111	00	0054
.3		R Add	0000		0112	12	0000
.4		TR	5.0		0113	00	0116
.5		00	0000		0114	00	0000
.6		00	0000		0115	00	0000
5.0		READ	4000		0116	30	4000
.1	-	Copy	0000		0117	-37	0000
.2		TR	0000		0120	01	0000
.3		00	0000		0121	00	0000

Program being punched must start at even address and end on odd address.

Octonary to Binary Conversion and Storage of Instruction Cards MMA 011

Decimal to Binary Conversion and Storage of Instruction Cards MMA 000

These two programs are identical except for the table of translation constants located at $(0130)_8$ to $(0147)_8$.

M. M. Astrahan

August 18, 1952

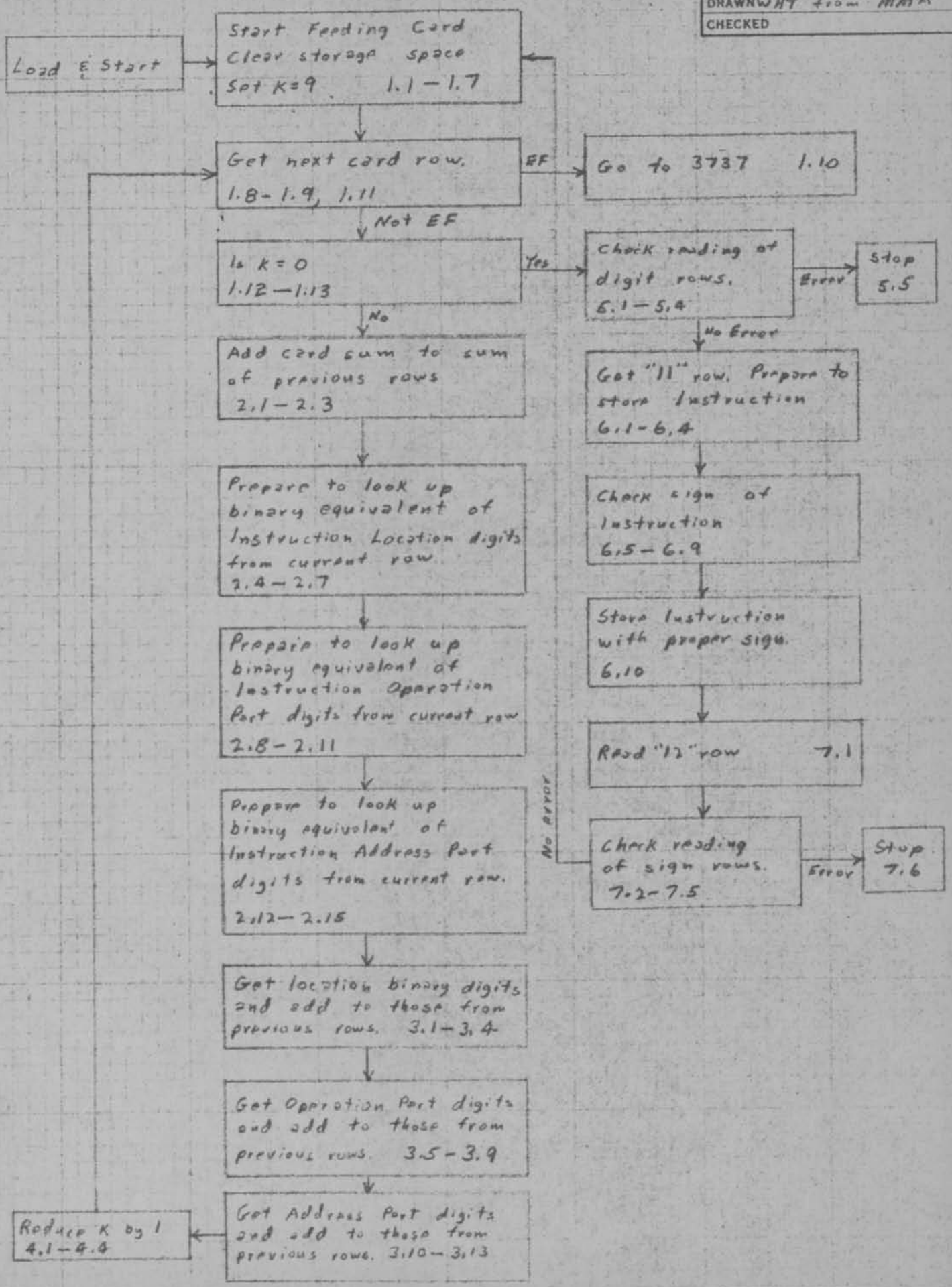
IBM

MMA-011 OR 000

DATE 3-24-52

DRAWN WHT from MMA

CHECKED



OCTONARY TO BINARY CONVERSION OF INSTRUCTION CARDS

MMA 011 & 000

DATE 3-24-52

WRITTEN WHT from MMA

CHECKED

	LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART		Octonary		
	0.1	-	COPY	0002		0000	77	0002
	0.2	-	COPY	0004		0001	77	0004
	0.3	-	COPY	0006		0002	77	0006
	0.4	-	COPY	0010		0003	77	0010
0.9 →	0.5		R ADD	0.8		0004	12	0007
	0.6		SUB	0.10		0005	05	0011
	0.7		STORE	0.8		0006	14	0007
0.16 →	0.8	-	COPY	(0010)	Split loading	0007	77	0010
	0.9		TR	0.5	sub program	0010	01	0004
	0.10		STOP	0002		0011	00	0002
	0.11		R ADD	0.17		0012	12	0020
	0.12		SUB	0.10		0013	05	0011
	0.13		STORE	0.17		0014	14	0020
	0.14		TR 0	1.1		0015	04	0021
	0.15		READ	4000		0016	30	4000
	0.16		TR	0.8		0017	01	0007
	0.17		STOP	(0006)		0020	00	0006
0.14 7.5 →	1.1		READ	4000	Card reader	0021	30	4000
	1.2		R ADD	12.1		0022	12	0130
	1.3		STORE	11.3	Clear storage	0023	14	0154
	1.4		STORE	11.4	space	0024	14	0155
	1.5		STORE	10.3		0025	14	0152
	1.6		R ADD	11.5	Set K=9	0026	12	0124
	1.7		STORE	11.2		0027	14	0153
4.4 →	1.8		COPY	10.1	Read left hand cd. row	0030	37	0150
	1.9		TR	1.11		0031	01	0033
	1.10		TR	3737	E.F.	0032	01	7737
	1.11		COPY	10.2	Read right hand card row	0033	37	0151
	1.12		R ADD	11.2	Is K=0	0034	12	0153
	1.13		TR 0	5.1		0035	04	0076
↓								
	2.1		R ADD	10.1	Accum. rows for	0036	12	0150
	2.2		ADD	10.3	check (K≠0)	0037	11	0152
	2.3		STORE	10.3		0040	14	0152
	2.4		R ADD	10.1		0041	12	0150
	2.5		A RIGHT	15 (13)	Cd. col. 5 is in 2^{-17}	0042	27	0015
	2.6		ADD	11.1	pos. of accum.	0043	11	0123
	2.7		STORE A	3.1	Addr. part of 3.1 = (N+12.1)	0044	15	0055
	2.8		A LEFT	23 (19)	Eliminate instruction	0045	26	0023
	2.9		A RIGHT	20 (16)	Location digits	0046	27	0020
	2.10		ADD	11.1		0047	11	0123
	2.11		STORE A	3.5	Addr. part of 3.5 = (M+12.1)	0050	15	0061
	2.12		A LEFT	23 (19)	Eliminate instruction	0051	26	0023
	2.13		A RIGHT	17 (15)	op. part digits	0052	27	0017
	2.14		ADD	11.1		0053	11	0123
	2.15		STORE A	3.10	Addr. part of 3.10 = (L+12.1)	0054	15	0066
↓								
	3.1	N = Instr. loc. octonary digits from current row						
		M = Instr. op. part octonary digits from current row						
		L = Instr. address part octonary digits from current row						

MMA-011 or 000

DATE 3-24-52
WRITTEN w HT from MMA
CHECKED

	LOCATION ±	INSTRUCTIONS OR DATA ± OPERATION PART	ADDRESS PART				
	2.15						
	↓						
	3.1	LOAD MQ	(12.i)	i=1, or 2, ... or 16	0055	17	0000
	3.2	MPY	11.2	Correct for current row	0056	20	0153
	3.3	ADD	11.3	Accum. parts of	0057	11	0154
	3.4	STORE	11.3	Instr. loc.	0060	14	0154
	3.5	LOAD MQ	(12.i)	i=1, or 2, or 3, or 4	0061	17	0000
	3.6	MPY	11.2		0062	20	0153
	3.7	A LEFT	14(12)		0063	26	0014
	3.8	ADD	11.4	Accum. parts of	0064	11	0155
	3.9	STORE	11.4	Instr. op. part	0065	14	0155
	3.10	Load MQ	(12.i)	i=1, or 2, ... or 16	0066	17	0000
	3.11	MPY	11.2		0067	20	0153
	3.12	ADD	11.4	Accum. parts of	0070	11	0155
	3.13	STORE	11.4	Instr. Addr. Part	0071	14	0155
	↓						
	4.1	R ADD	11.2		0072	12	0153
	4.2	SUB	11.6	Reduce K by 1	0073	05	0125
	4.3	STORE	11.2		0074	14	0153
	4.4	TR	1.8		0075	01	0030
1.14 →	5.1	R ADD	10.1	K=0	0076	12	0150
	5.2	ADD	10.3	check reading of	0077	11	0152
	5.3	SUB	11.8	digit rows	0100	05	0127
	5.4	TR 0	6.9		0101	04	0103
	5.5	STOP	1.1	Error	0102	00	0021
5.4 →	6.1	COPY	10.1	Read "11" row	0103	37	0150
	6.2	COPY	10.2		0104	37	0151
	6.3	R ADD	11.3	Prepare to store	0105	12	0154
	6.4	STORE A	6.10	instruction	0106	15	0114
	6.5	R ADD	10.1		0107	12	0150
	6.6	TR 0	6.9		0110	04	0113
	6.7	R SUB	11.4	Instruction is −	0111	06	0155
	6.8	Tr	6.10		0112	01	0114
6.6 →	6.9	R ADD	11.4	Instruction is +	0113	12	0155
6.8 →	6.10	STORE	(Instr. Loc.)	Store Instruction	0114	14	0000
	↓						
	7.1	COPY	10.3	Read left hand "12" row	0115	37	0152
	7.2	R ADD	10.3		0116	12	0152
	7.3	ADD	10.1	check reading of	0117	11	0150
	7.4	SUB	11.7	sign rows	0120	05	0126
	7.5	TR 0	1.1		0121	04	0021
	7.6	STOP	1.1	Error	0122	00	0021

MMA-011 OR 000

DATE 3-24-52
WRITTEN WHT from MMA
CHECKED

LOCATION ±	INSTRUCTIONS OR DATA ± OPERATION PART	ADDRESS PART					
10.1	(left hand card words)		} Temporary Storage		0150	—	———
10.2	(right hand card words)				0151	—	———
10.3	(Acc. of rows for check)				0152	—	———
11.1	Stop	12.1			0123	00	0130
11.2	(row index = K)		$K=(9,8,\ldots 1,0)\times 2^{-4}$		0153	—	———
11.3	(Acc. of parts converted Inst. Loc.)				0154	—	———
11.4	(Acc. of parts of converted Instr.)				0155	—	———
11.5	9×2^{-4}				0124	22	0000
11.6	1×2^{-4}				0125	02	0000
11.7	1×2^{-5}				0126	00	0100
11.8	$15\times 2^{-4}+63\times 2^{-11}=367700$ (octonary)				0127	36	7700
FOLLOWING TABLE FOR MMA-011 (OCT→BIN)							
12.1	00	0000			0130	00	0000
12.2	00	0020	$(1)_8\times 2^{-13}$	binary pt	0131	00	0020
12.3	00	0200	$(10)_8\times 2^{-13}$	to left	0132	00	0200
12.4	00	0220	$(11)_8\times 2^{-13}$	of pos.	0133	00	0220
12.5	00	2000	⋮	1.	0134	00	2000
12.6	00	2020	⋮		0135	00	2020
12.7	00	2200			0136	00	2200
12.8	00	2220			0137	00	2220
12.9	02	0000			0140	02	0000
12.10	02	0020			0141	02	0020
12.11	02	0200			0142	02	0200
12.12	02	0220			0143	02	0220
12.13	02	2000			0144	02	2000
12.14	02	2020			0145	02	2020
12.15	02	2200	$(1110)_8\times 2^{-13}$		0146	02	2200
12.16	02	2220	$(1111)_8\times 2^{-13}$		0147	02	2220
FOR MMA-000, (DEC→BIN), USE FOLLOWING TABLE							
12.1	0				0130	00	0000
.2	$(1)_{10}\times 2^{-13}$				0131	00	0016
.3	$(10)_{10}\times 2^{-13}$				0132	00	0160
.4	$(11)_{10}\times 2^{-13}$				0133	00	0176
.5					0134	00	1600
.6					0135	00	1616
.7					0136	00	1760
.8					0137	00	1776
.9					0140	03	3712
.10					0141	03	3728
.11					0142	03	3872
.12					0143	03	3888
.13					0144	04	1216
.14					0145	04	1232
.15	$(1110)_{10}\times 2^{-13}$				0146	04	1376
12.16	$(1111)_{10}\times 2^{-13}$				0147	04	1392

NR9007
Transition from binary loading

After loading a program on binary cards, one may want to load a program such as Tracing or Print Memory. These programs are self loading. It is convenient to have a program to effectively push the load button. NR 9007 does this.

This program usually follows a binary deck. It consists of the following instructions on one binary card, and a control card for FEJ015 to transfer control to 0305.

```
0304       STOP     0305
0305       READ     4000
0306   —   COPY     0000
0307       TR       0000
```

NR 9008
Transition from Octonary Loading

After loading a deck of octonary cards one may want to load a program on binary cards such as Binary Punching, Tracing, or Printing. These are self loading programs. It is convenient to have a program to effectively push the load button. NR9008 does this.

This program usually follows an octonary deck being loaded by MMA011; it consists of the following octonary cards.

```
0021     TR    0305
0305     READ  4000
0306  —  COPY  0000
0307  +  TR    0000
```

NR9010

Transition from Decimal to Binary Loading

After loading a deck of decimal cards, one may desire to load a program on binary cards, such as Tracing or Print Memory. These programs are self loading. It is convenient to have a program which pushes the load button. NR9010 does this. NR9010 consists of the following decimal cards:

```
0017 TR     0197
0197 READ   2048
0198 COPY   0000
0199 TR     0000
```

IBM

WHT 016
DATE 8-18-52
DRAWN
CHECKED

PRINT OUT MEMORY IN OCTONARY

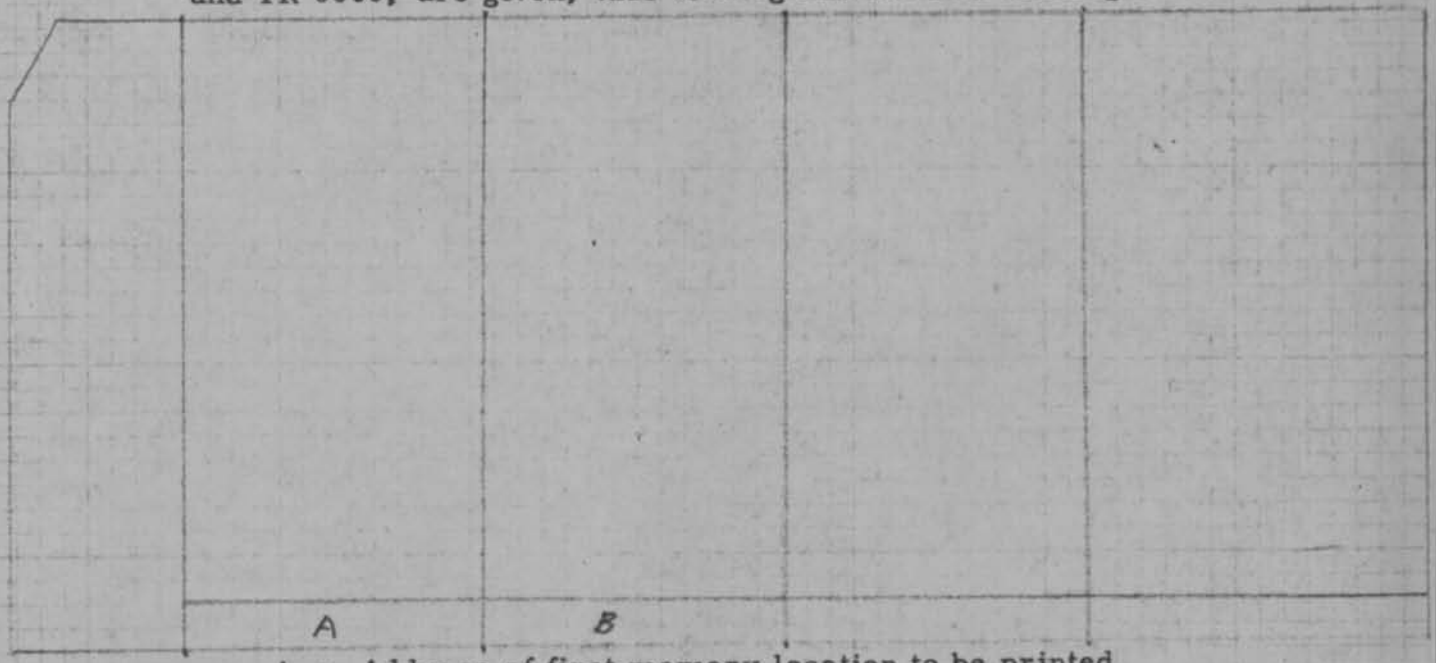
The attached program will print out in octonary all or any part of memory above 0274, four instructions with their locations to the line of printing. If the straight-across plugboard is used, printing will be in the following form:

0060+304000	0061-370206	0062+120206	0063+150101
0064-151234	0065+130022	...	...
...	...		

etc.

A control card is used to determine first and last addresses to be printed. It is loaded immediately following WHT00R04. Its form is as follows:

Upon completion of the printing of the desired portion of memory, the sequence of instructions, READ 4000, -COPY 0000, and TR 0000, are given, thus loading another self-loading card.

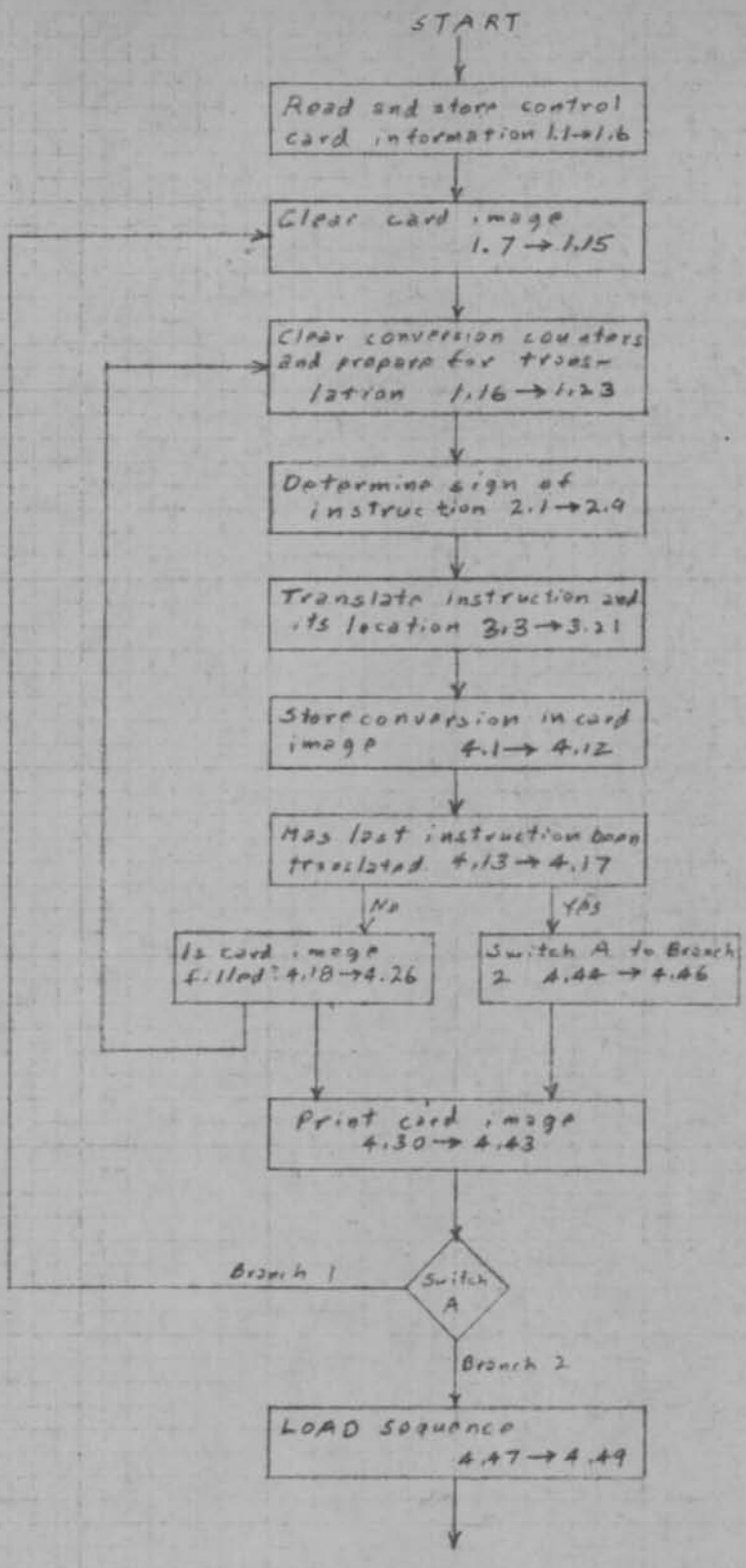


A = Address of first memory location to be printed.

B = Address of last memory location to be printed.

IBM

WHT-016
DATE 8-18-52
DRAWN WHT
CHECKED



Print Out Memory in Octonary

WHT-016

DATE 8-15-52
WRITTEN WHT
CHECKED

	LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART				
	1.1		READ	4000	Copy and store	0066	30	4000
	1.2	-	COPY	5.0	control card information	0067	-37	0232
	1.3		R ADD	5.0		0070	12	0232
	1.4		STORE A	2.1	Store address of	0071	15	0115
	1.5		STORE A	2.5	first instruction in	0072	15	0121
	1.6		STORE A	2.9	2.1, 2.5, and 2.9	0073	15	0125
4.42 →	1.7		LOAD MQ	5.12		0074	17	0246
	1.8		R ADD	5.12		0075	12	0246
	1.9		STORE A	1.10		0076	15	0077
1.15 →	1.10	-	STORE MQ	( )	Clear card image	0077	-16	0000
	1.11		R ADD	1.10	to zero.	0100	12	0077
	1.12		SUB	5.10		0101	05	0244
	1.13		STORE A	1.10		0102	15	0077
	1.14		ADD	5.15		0103	11	0251
	1.15		TR +	1.10		0104	03	0077
4.28 →	1.16		R ADD	5.12		0105	12	0246
	1.17	-	STORE	6.0		0106	-14	0262
	1.18	-	STORE	6.2	Clear conversion	0107	-14	0264
	1.19	-	STORE	6.4	counters to zero	0110	-14	0266
	1.20	-	STORE	6.6		0111	-14	0270
	1.21	-	STORE	6.8		0112	-14	0272
	1.22		R ADD	5.14	Set 3.12 to store	0113	12	0250
	1.23		STORE A	3.12	first 6.4	0114	15	0137
	2.1		R ADD	(current inst.)	Determine sign of instruct.	0115	12	0000
	2.2		TR +	2.7	If -, put bit in 11	0116	03	0123
	2.3		R ADD	5.13	row of conversion ctr.	0117	12	0247
	2.4		STORE	6.1	and put + instruction	0120	14	0263
	2.5		R SUB	(current inst.)	in accumulator	0121	06	0000
	2.6		TR	3.3		0122	01	0126
2.2 →	2.7		R ADD	5.13	If positive, put bit	0123	12	0247
	2.8		STORE	6.0	in 12 row of	0124	14	0262
	2.9		R ADD	(current inst.)	conversion ctr.	0125	12	0000
	↓							
2.6 →	3.3		L RIGHT	44 $(36)_D$	Put inst. in MQ	0126	25	0044
	3.4		R ADD	2.1	Get location of current inst.	0127	12	0115
	3.5		A LEFT	7	Eliminate op. part of 2.1	0130	26	0007
	3.6		A RIGHT	31 $(25)_D$	Line up location with inst.	0131	27	0031
	3.7		L RIGHT	11 $(9)_D$	Get digit in acc.	0132	25	0011
3.21 →	3.8		A LEFT	22 $(18)_D$	Move to left of acc.	0133	26	0022
	3.9		ADD	5.16	Determine conversion	0134	11	0252
	3.10		STORE A	3.13	counter in which to	0135	15	0140
	3.11		STORE A	3.14	store bit	0136	15	0141
	3.12		R ADD	( )	Add bit to	0137	12	0000
	3.13		ADD	( )	conversion counter	0140	11	0000
	3.14		STORE	( )		0141	14	0000
	3.15		R ADD	3.12	Increase address of	0142	12	0137
	3.16		ADD	5.11	3.12 by 1	0143	11	0245
	3.17		STORE A	3.12		0144	15	0137
	↓							
	3.18							

IBM

WHT-016

DATE 8-15-52
WRITTEN WHT
CHECKED

	LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART			
	3.17						
	↓						
	3.18		SUB	5.17	Is translation finished?	0145	05 0253
	3.19		TR O	4.1	Yes	0146	04 0151
	3.20		L LEFT	3	No. Get next digit	0147	24 0003
	3.21		TR	3.8		0150	01 0133
3.19 →	4.1		R ADD	5.18	Set 4.3 to store 7 row	0151	12 0254
	4.2		STORE A	4.3	of conversion counter	0152	15 0153
4.12 →	4.3		R ADD	( )	Store conversion counter	0153	12 0000
	4.4		STORE	( )	contents in card image	0154	14 0000
	4.5		R ADD	4.4		0155	12 0154
	4.6		ADD	5.9	Increase address of	0156	11 0243
	4.7		STORE A	4.4	4.4 by 4	0157	15 0154
	4.8		R ADD	4.3		0160	12 0153
	4.9		SUB	5.11	Decrease address of	0161	05 0245
	4.10		STORE A	4.3	4.3 by 1	0162	15 0153
	4.11		SUB	5.19	Is storage completed?	0163	05 0255
	4.12		TR +	4.3	No.	0164	03 0153
	4.13		R ADD	2.1	Yes	0165	12 0115
	4.14		A LEFT	7	Has last instruction	0166	26 0007
	4.15		A RIGHT	7	been translated?	0167	27 0007
	4.16		SUB	5.1		0170	05 0233
	4.17		TR O	4.44	Yes	0171	04 0224
	4.18		R ADD	2.1	No	0172	12 0115
	4.19		ADD	5.11	Increase addresses of	0173	11 0245
	4.20		STORE A	2.1	2.1, 2.5, and 2.9 to	0174	15 0115
	4.21		STORE A	2.5	get next instruction	0175	15 0121
	4.22		STORE A	2.9		0176	15 0125
	4.23		R ADD	4.4	Prepare to store in	0177	12 0154
	4.24		SUB	5.20	next portion of	0200	05 0256
	4.25		STORE A	4.4	C.I.	0201	15 0154
	4.26		SUB	5.21	Is card image filled?	0202	05 0257
	4.27		TR O	4.29	Yes	0203	04 0205
	4.28		TR	1.16	No. Translate next inst.	0204	01 0105
4.27 4.46 →	4.29		WRITE	1000		0205	32 1000
	4.30		R ADD	5.12		0206	12 0246
	4.31		STORE A	4.4	Reset addresses of 4.4	0207	15 0154
	4.32		STORE A	4.37	and 4.37 to zero	0210	15 0215
	4.33		COPY	5.12		0211	37 0246
	4.34		COPY	5.12	Copy 0's for 8 rows	0212	37 0246
	4.35		COPY	5.12		0213	37 0246
	4.36		COPY	5.12		0214	37 0246
4.43 →	4.37	-	COPY	( )	Copy half row	0215	-37 0000
	4.38		R ADD	4.37		0216	12 0215
	4.39		SUB	5.10	Increase address of	0217	05 0244
	4.40		STORE A	4.37	4.37 by 2	0220	15 0215
	4.41		ADD	5.22	Is printing finished?	0221	11 0260
	4.42		TR O	(1.7)	Yes	0222	04 0074
	4.43		TR	4.37	No	0223	01 0215

WHT-016

DATE 8-15-52
WRITTEN WHT
CHECKED

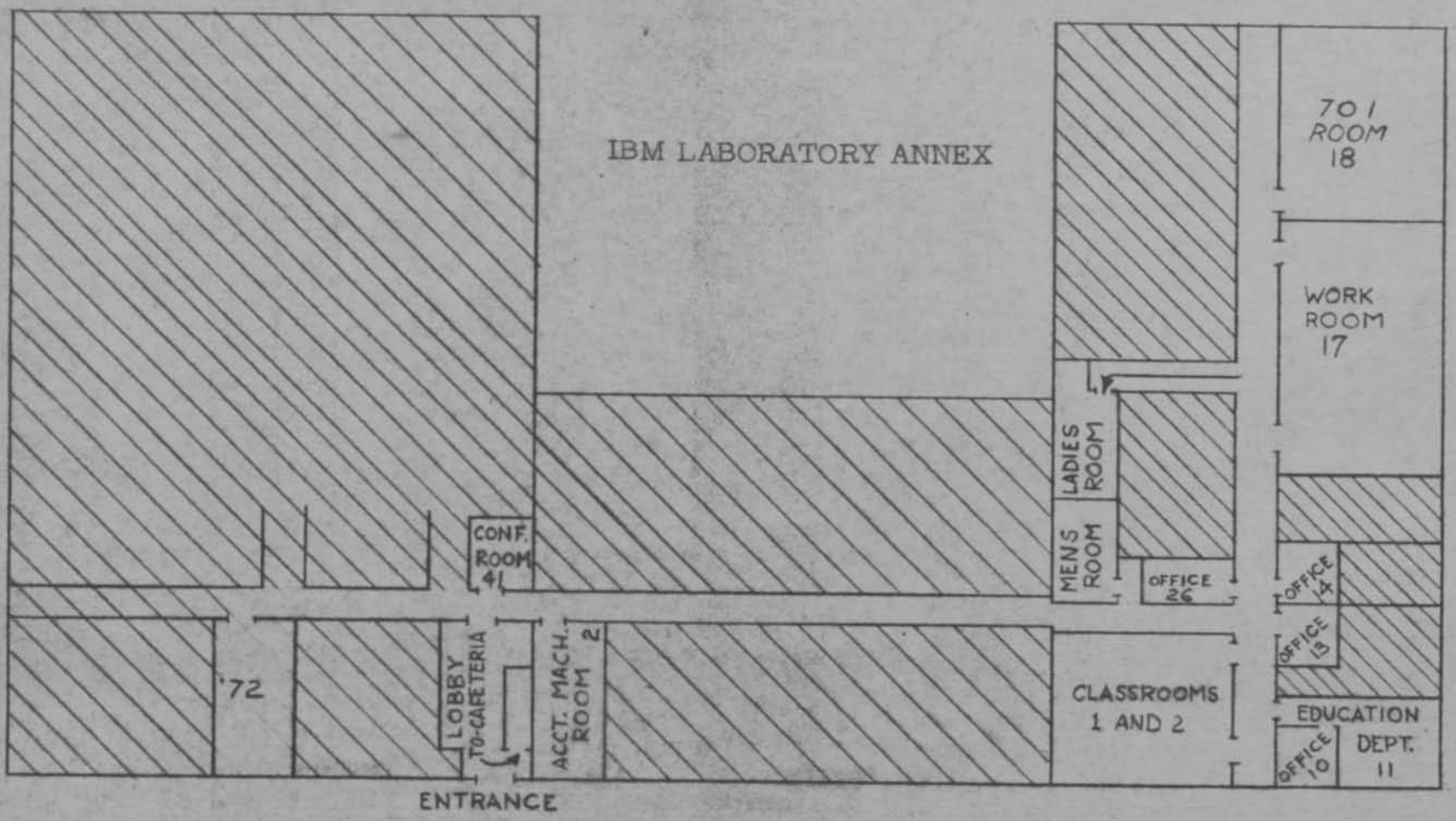
LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART				
4.44		R ADD	5.23	Prepare for giving	0224	12	0261
4.45		STORE A	4.42	LOAD after printing.	0225	15	0222
4.46		TR	4.29		0226	01	0205
4.47		READ	4000		0227	30	4000
4.48	-	COPY	0000	LOAD sequence	0230	-37	0000
4.49		TR	0000		0231	01	0000

WHT-016

DATE 8-15-52
WRITTEN WHT
CHECKED

LOCATION ±	INSTRUCTIONS OR DATA ±	OPERATION PART	ADDRESS PART			
		CONSTANTS AND	STORAGE			
5.0		First address to be printed		Information from	0232	00 0000
5.1		Last address to be printed		control card	0233	00 0000
5.2		2000			0234	00 2000
5.3		1000			0235	00 1000
5.4		400			0236	00 0400
5.5		200			0237	00 0200
5.6		40			0240	00 0040
5.7		20			0241	00 0020
5.8		10			0242	00 0010
5.9		4			0243	00 0004
5.10		2			0244	00 0002
5.11		1			0245	00 0001
5.12		0			0246	00 0000
5.13		100			0247	00 0100
5.14		STOP	5.2		0250	00 0234
5.15		STORE MQ	L(12R)+2		0251	16 0050
5.16		STOP	6.2		0252	00 0264
5.17		R ADD	5.12		0253	12 0246
5.18		STOP	6.9		0254	00 0273
5.19		R ADD	6.0		0255	12 0262
5.20		47			0256	00 0047
5.21		STORE	0004		0257	14 0004
5.22		COPY	0050		0260	37 0050
5.23		STOP	4,47		0264	00 0227
6.0		12 row		Conversion counters	0262	00 0000
6.1		11 row			0263	00 0000
6.2		0 row			0264	00 0000
6.3		1 row			0265	00 0000
6.4		2 row			0266	00 0000
6.5		3 row			0267	00 0000
6.6		4 row			0270	00 0000
6.7		5 row			0271	00 0000
6.8		6 row			0272	00 0000
6.9		7 row			0273	00 0000

7L of C.I. starts at address 0000



Cross hatched areas are restricted to
IBM LABORATORY PERSONNEL
only.

Procedure:

I. To make binary cards:

a. L-05 to load
M-10

b. Load decimal cards
intr. + data decks. Start calculator at $(1005)_8$

→ get out 2 binary cards: 1 instr, 1 data

II. Combine above 2 cards with P-01 (print routine).

load L-05
P-01
binary cards.

III. Print out memory (P-01 print board)
- put binary calling card in feed
- start at $(3406)_8$.

→ get out: listing of instructions in decimal & octal.

IV. Tracing of routine (Tracing board)

load FEJ 0 180 1
JHH 0 150 1
- binary octal card

→ get out tracing of routine in octal.

ABSTRACT PROCEDURE #L-1

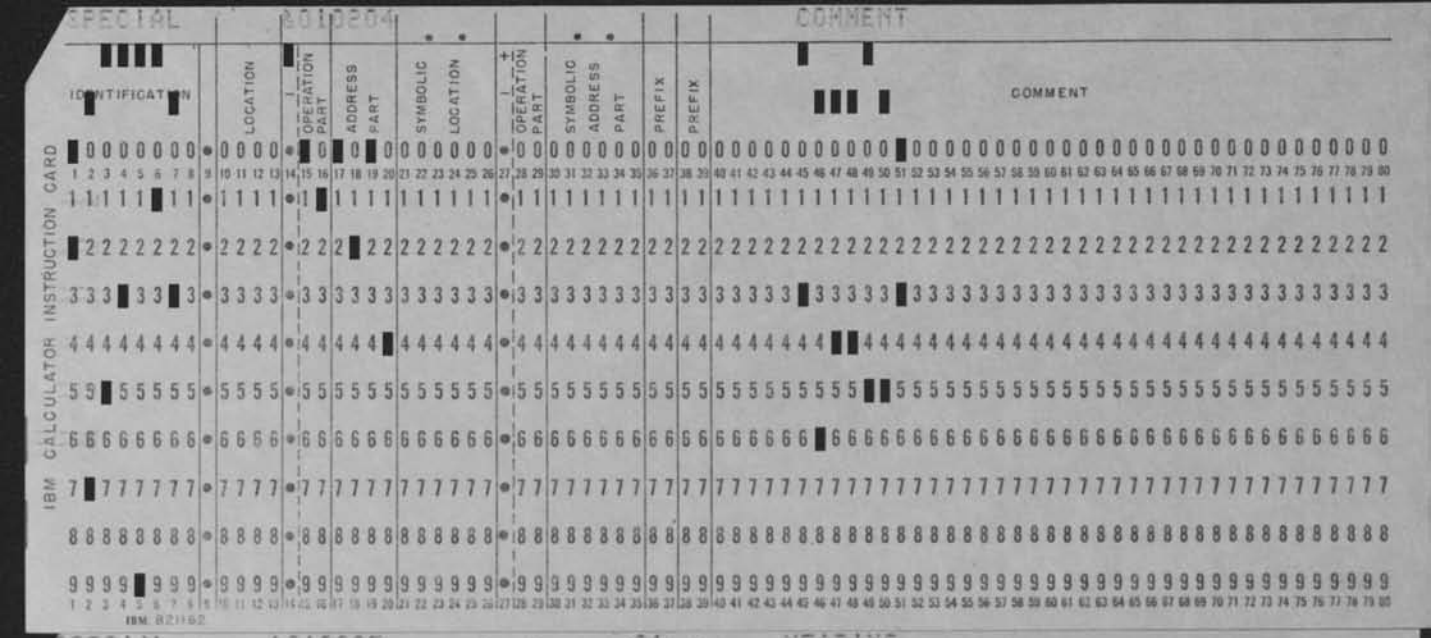
Keller
GE.

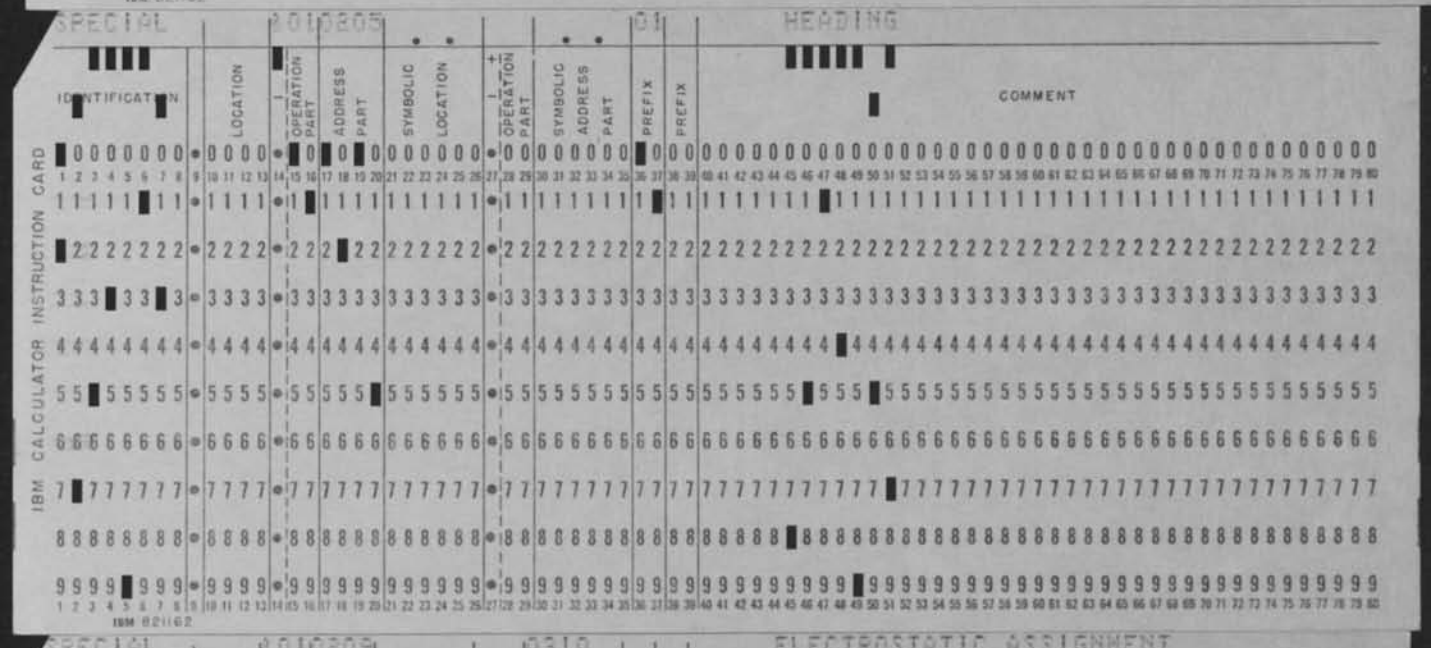
LIST OF ABSTRACT ORDERS

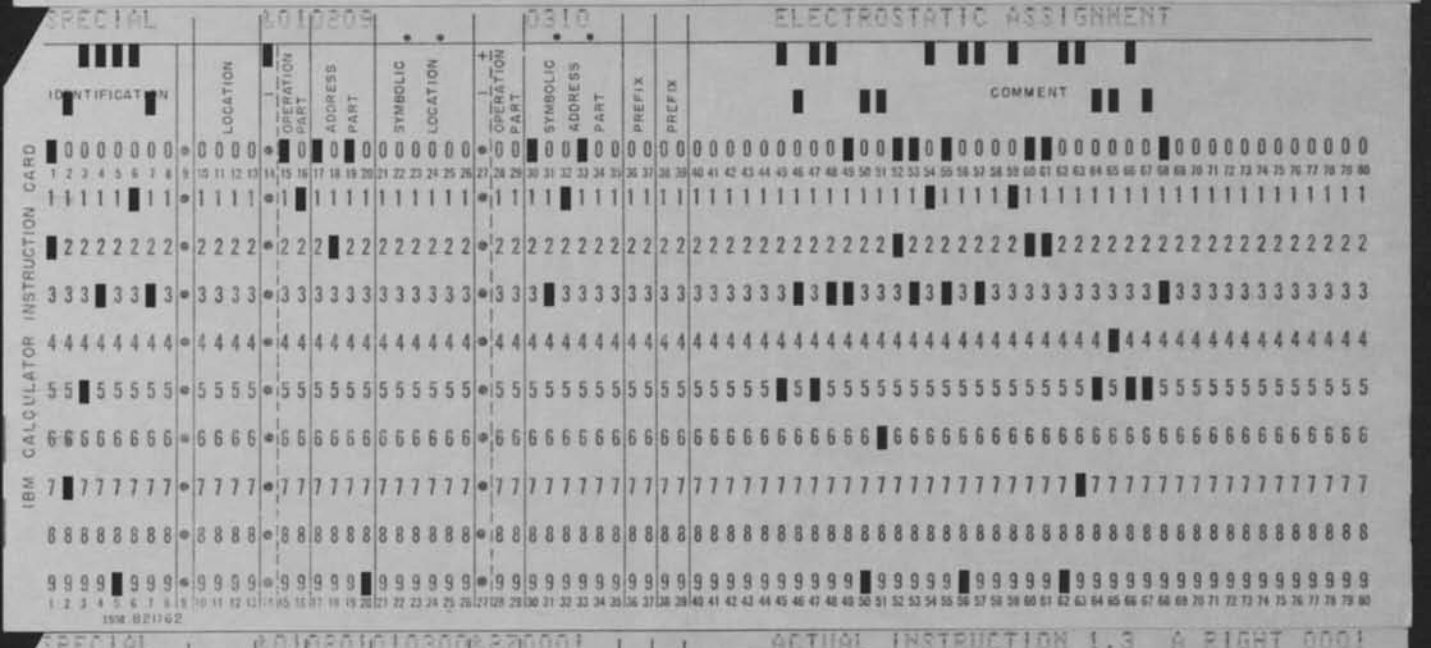
Operation Number	Name of Operation	Type of Point	Size of Word	Location of Word
+00	Transfer to Sub-Sequence or Sub-Program	--	--	E
+01	Transfer on Zero	--	--	E
+02	Transfer on Plus	--	--	E
+03	Transfer	--	--	E
+04	Reset Add Half Word	Fixed	H	E
+05	Address Add	"	H	E
+06	Store Half Word	"	H	E
+07	Store Address	"	H	E
+08	Store Full Word	Float.	F	E
-8 } -9	" " "	"	"	D
+10	Reset Add	"	"	E
-10 } -11	" "	"	"	D
+12	Add	"	"	E
-12 } -13	"	"	"	D
+14	Subtract	"	"	E
-14 } -15	"	"	"	D
+16	Subtract and Change Sign of the Difference	"	"	E
-16 } -17	" " " " " " "	"	"	D
+18	Multiply	"	"	E
-18 } -19	"	"	"	D
+20	Divide	"	"	E
-20 } -21	"	"	"	D
+22	Divide and Take Reciprocal	"	"	E
-22 } -23	" " " "	"	"	D
+24	Reset Add and Take Square Root	"	"	E
-24 } -25	" " " " " "	"	"	D
+26	Reset Add and Square	"	"	E
-26 } -27	" " " "	"	"	D
+28	Reset Add and Take Logarithm to Base 10	"	"	E
-28 } -29	" " " " " " " "	"	"	D
+30	Reset Add and Calc. 10^x	"	"	E
-30 } -31	" " " " "	"	"	D

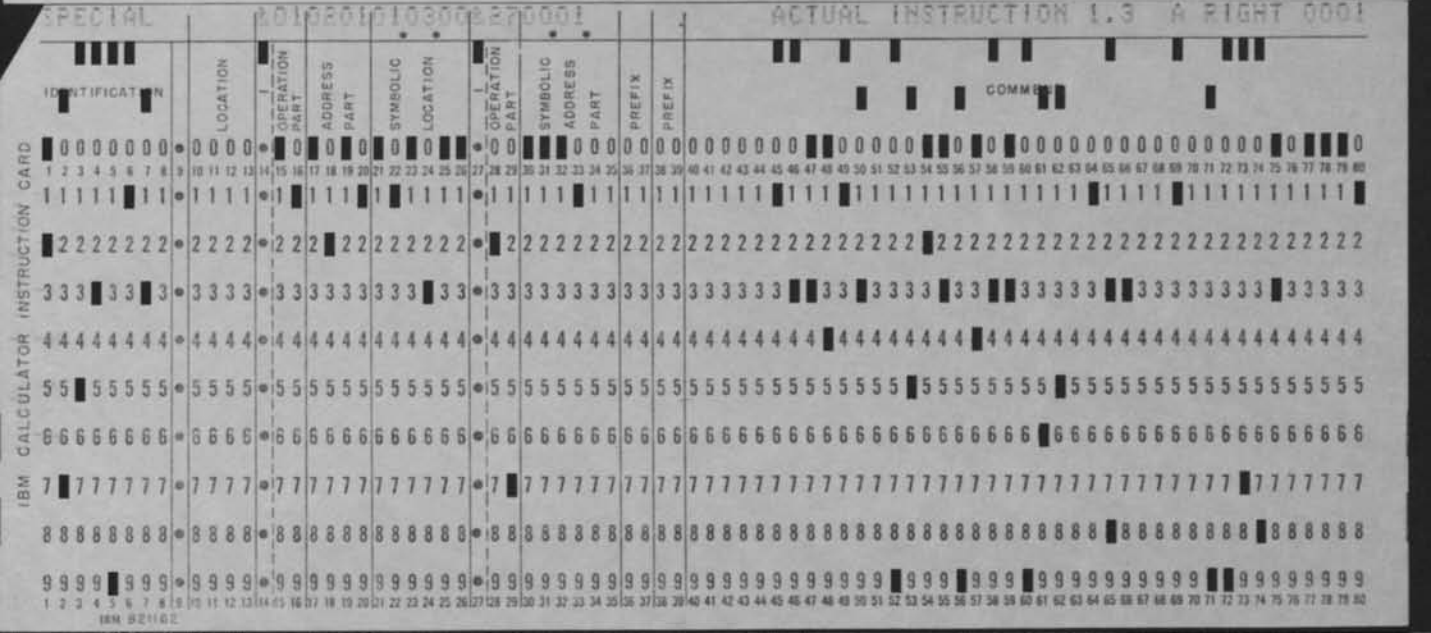
CORRELATION BETWEEN ABSTRACT DRUM ADDRESSES AND REAL DRUM ADDRESSES

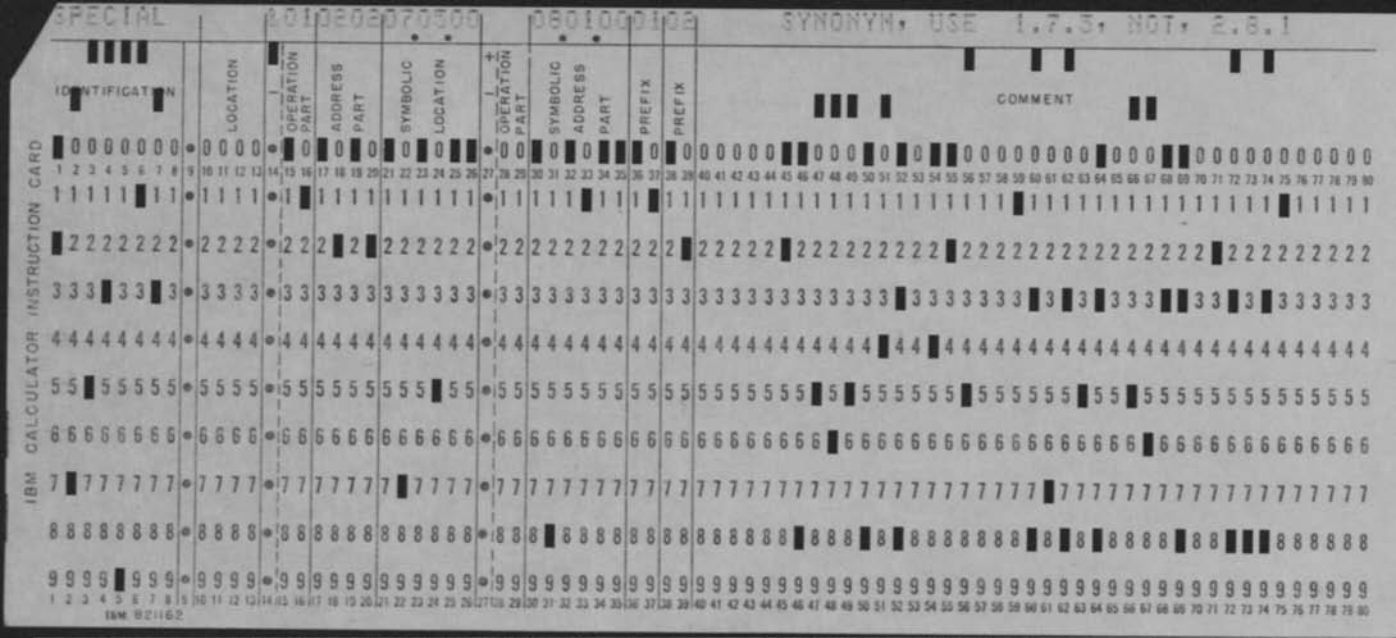
	REAL ADDRESSSES	ABSTRACT ADDRESSES
	Only full words can be read from the drum but counting is by half words. n = the half-word count to a given location on a given drum.	Only full words can be read from the drum and counting is by full words. $\frac{n}{2}$ = the full-word count to a given location on a given drum.
Drum #1	Drum Address = $\pm$ 0128 Address within the specified drum = $\pm$ n in which n is an even number between 0000 and 4094 inclusive.	Address (total) = $-\left(0000 + \frac{n}{2}\right)$ $0 \leqslant -A \leqslant 2047$
Drum #2	Drum Address = $\pm$ 0129 Address within the specified drum is the same as for drum #1.	Address (total) = $-\left(2048 + \frac{n}{2}\right)$ $2048 \leqslant -A \leqslant 4095$
Drum #3	Drum Address = $\pm$ 0130 Address within the specified drum is the same as for drum #1.	Address (total) = $-\left(4096 + \frac{n}{2}\right)$ $4096 \leqslant -A \leqslant 6143$
Drum #4	Drum Address = $\pm$ 0131 Address within the specified drum is the same as for drum #1.	Address (total) = $-\left(6144 + \frac{n}{2}\right)$ $6144 \leqslant -A \leqslant 8191$

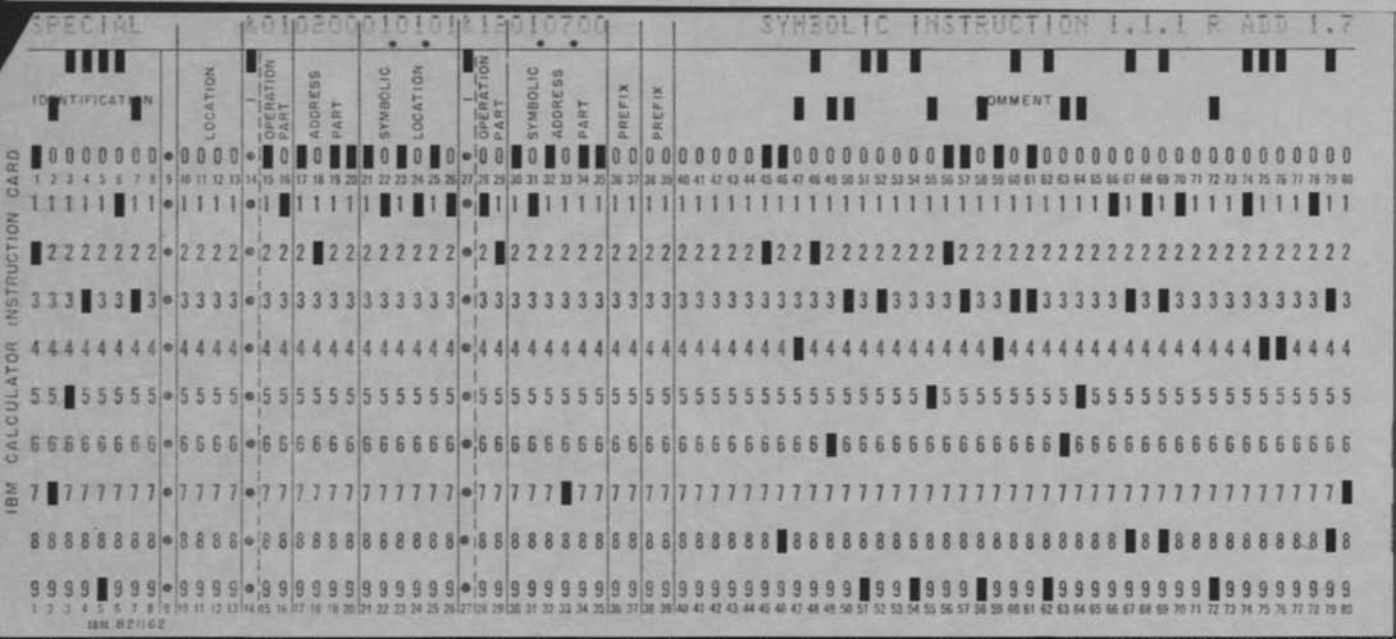

SPECIAL
COMMENT
IDENTIFICATION
LOCATION
OPERATION PART
ADDRESS PART
SYMBOLIC LOCATION
OPERATION PART
SYMBOLIC ADDRESS PART
PREFIX
PREFIX
COMMENT
IBM CALCULATOR INSTRUCTION CARD
IBM 821162

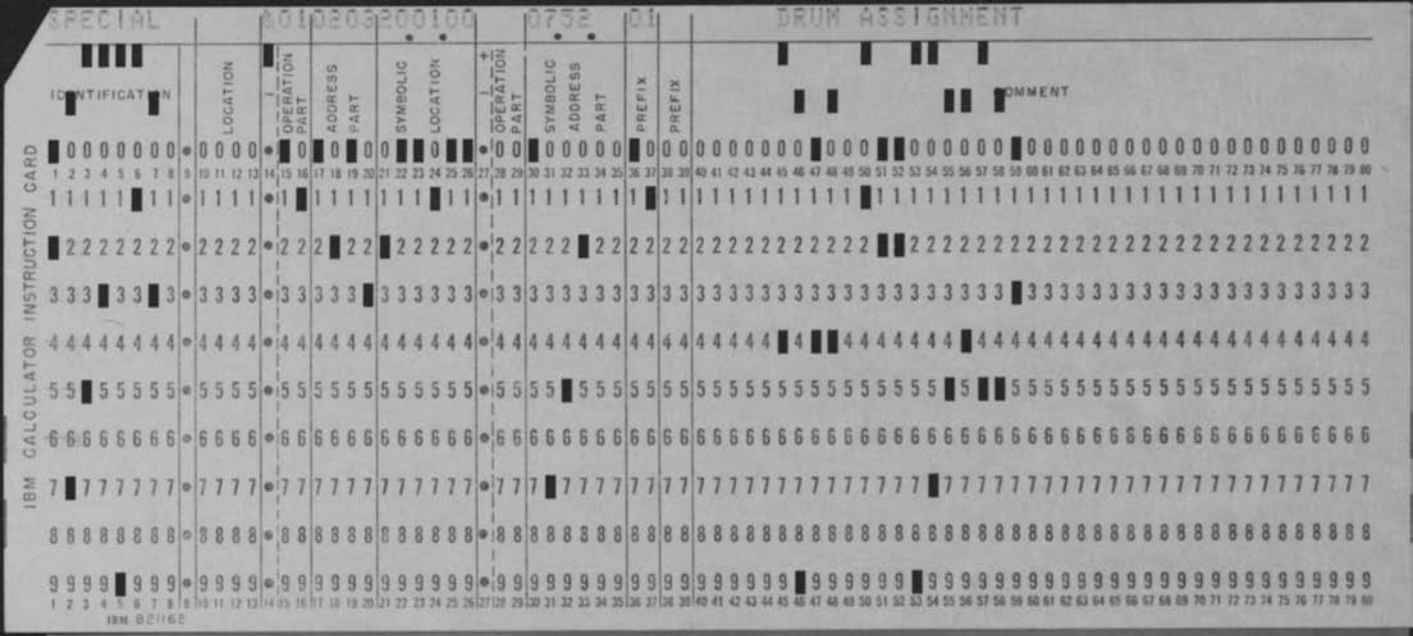

SPECIAL
HEADING
IDENTIFICATION
LOCATION
OPERATION PART
ADDRESS PART
SYMBOLIC LOCATION
OPERATION PART
SYMBOLIC ADDRESS PART
PREFIX
PREFIX
COMMENT
IBM CALCULATOR INSTRUCTION CARD
IBM 821162

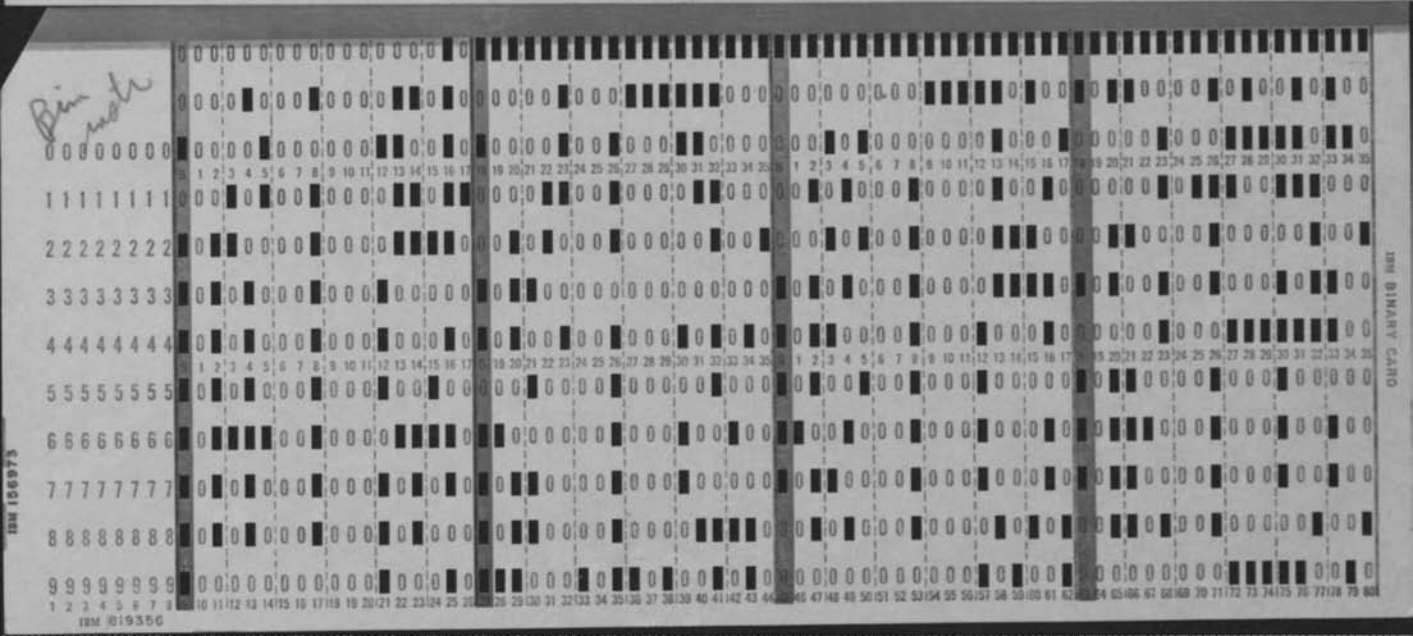

SPECIAL
ELECTROSTATIC ASSIGNMENT
IDENTIFICATION
LOCATION
OPERATION PART
ADDRESS PART
SYMBOLIC LOCATION
OPERATION PART
SYMBOLIC ADDRESS PART
PREFIX
PREFIX
COMMENT
IBM CALCULATOR INSTRUCTION CARD
IBM 821162


SPECIAL
ACTUAL INSTRUCTION 1.3 A RIGHT
IDENTIFICATION
LOCATION
OPERATION PART
ADDRESS PART
SYMBOLIC LOCATION
OPERATION PART
SYMBOLIC ADDRESS PART
PREFIX
PREFIX
COMMENT
IBM CALCULATOR INSTRUCTION CARD
IBM 821162


SPECIAL
SYNONYM, USE 1.7.3, NOT, 2.8.1
IDENTIFICATION
LOCATION
OPERATION PART
ADDRESS PART
SYMBOLIC LOCATION
OPERATION PART
SYMBOLIC ADDRESS PART
PREFIX
PREFIX
COMMENT
IBM CALCULATOR INSTRUCTION CARD
IBM 821162


SPECIAL
SYMBOLIC INSTRUCTION 1.1.1 R ADD 1.7
IDENTIFICATION
LOCATION
OPERATION PART
ADDRESS PART
SYMBOLIC LOCATION
OPERATION PART
SYMBOLIC ADDRESS PART
PREFIX
PREFIX
COMMENT
IBM CALCULATOR INSTRUCTION CARD
IBM 821162


SPECIAL
DRUM ASSIGNMENT
IDENTIFICATION
LOCATION
OPERATION PART
ADDRESS PART
SYMBOLIC
LOCATION
OPERATION PART
SYMBOLIC
ADDRESS PART
PREFIX
PREFIX
COMMENT
IBM CALCULATOR INSTRUCTION CARD
IBM 82166


Binary
IBM BINARY CARD
IBM 156975
IBM 81935G