OUTLINE OF 701 ELEMENTARY

CODING CLASS INSTRUCTION

Prerequisite: Grade school arithmetic.

July 7, 1953

General discussion of machine.

Textbooks:

"Principles of Operation, Type 701"

"T-1 701 Section Utility Manual"

Binary to octal to decimal conversion.

Decimal to octal to binary conversion.

Binary and octal arithmetic.

Base "n" notation.

July 9, 1953

Addition, subtraction, multiplication, division within the 701.

Address arithmetic.

Brief summary of 701 operations.

July 14, 1953

Regional programming and S02.

July 16, 1953

Reading and interpreting binary cards so that control cards may be prepared and checked.

July 21, 1953

Card read-in programs from the T-1 Utility Manual. Check sums. V. R.

July 23, 1953

Print and punch programs from the T-1 Utility Manual. Calling sequence.

July 28, 1953

Read-write tape and read-write drum, set E.S., drums, tapes to zero programs from the T-1 Utility Manual.

July 30, 1953

Print operators on given instruction and also tracing program from the T-1 Utility Manual. In addition, dump-load using tape.

Los Alamos Debugging Programs and Techniques
As Used on the IBM 701

by Edward A. Voorhees, Jr.

Introduction

If the experience at other IBM 701 installations coincides with our experience at Los Alamos, I believe we may agree that the main bottleneck in the course of a problem is the period beginning after the coding of the problem has been assembled on cards and ending with the successful calculation of the first results, i.e., the debugging period. Also, in some problems, the code will never take on a fixed form, for with the entry of new parameters it is often necessary to modify the code and, in some cases, to actually recode portions of the problem. Frequently, this situation will require the use of debugging programs and techniques.

At present, there seem to be two main general debugging methods: (a) memory print-out and (b) tracing. Memory print-out may be defined as the listing of a stored program (or a selected part thereof) whose instructions are not being executed concurrently with the execution of the listing program. A tracing program is one that lists the instructions and certain additional information as the instructions of the stored program are actually being executed. It would seem that, in general, the memory print-out method makes for more efficient use of the machine, whereas, the tracing method makes the detection of coding errors easier for the individual at the expense of the machine. There are quite a few exceptions to this statement which arise because of the particular nature of the error being sought.

At Los Alamos a large majority of the coding (and debugging) is performed by persons who are not full-time coders. Many of these people are very capable coders, but for them coding is only a tool of their profession - a tool not unlike operating a slide rule or hand calculator before the advent of large scale computers. As a result, our debugging programs and techniques have been developed with these people in mind, and the trend has been to somewhat favor the person instead of the machine in the developing of new debugging programs. It has been our experience that the beginning coder will rely almost exclusively on tracing programs and that, as he gains experience, he will make increasing use of memory print-outs. The experienced individual will use either tracing or memory print-out, making his choice on the basis of the nature of the suspected error and the difficulty he anticipates in finding it.

Our method of time scheduling serves, however, as a check against idle or excessive wasted time during the debugging phase of the problem. This is accomplished by scheduling short periods of time, of the order of 10 minutes, in general, for debugging during the daytime and longer periods of time during the evening and night for "production" running.

The particular programs described below are used for debugging programs coded in machine language. Our two interpretive coding systems, the single-address Dual system and the three-address Shaco system, each has its own peculiar debugging program and technique. Since 80% of current problems are coded in machine language, (and the figure is gradually increasing) no further development of debugging routines for these systems is anticipated.

All memory print-out programs print information in octal, and all tracing programs print information in octal and decimal. Electrostatic storage will be referred to as memory in the remainder of the paper.

Memory Print-out Programs

a. 186 is a program to print in octal the contents of electrostatic storage except for the 151 half-words which it, itself, occupies. The program will search memory, beginning at the first half-word following itself, for the first half-word neither plus zero nor minus all ones. It will then print the location of that half-word, the half-word, and the following ten half-words, regardless of the composition of these ten words. It then continues the search and prints whenever the above condition is satisfied. 186 may be located anywhere in electrostatic storage.

186 is commonly employed when the problem stops unexpectedly and 151 consecutive half-words of storage are available. After noting the location of the stop and inserting the print board, the operator loads a properly located 186 deck of cards and the listing is issued automatically. Occasionally, 186 is used at an intentional stop to obtain memory listing. A memory malfunction is occasionally detected through 186, in which case the particular memory drawer at fault can be determined. At present, we are revising 186 to print n ($8 \leq n \leq 11$) instructions to a line to accommodate those people who find an octal print-out with 8 instructions to the line easier to read.

b. 982 is essentially a 186 program that will print all of memory except for the two full-words with location -0000 and -0002. This is accomplished through a self-loading program that transfers all of memory except for these two full-words to a drum. This program is intended for use with those programs which are so large that 151 half-words are not available for storing 186. After the listing is complete, memory is restored to its original form with the exception of the two full-words destroyed by the self-loading program. The listing is identical with that obtained from 186.

c. 784 is a program which lists all references made by a specified consecutive range of instructions to a specified consecutive range of half-words. The program operates in the following manner. The first half-word of the specified range of half-words, (viewed as an instruction) and its location are printed and marked with an asterisk. Then the address of each instruction of the set of instructions is examined, and, whenever the address is equal to the location of the half-word being considered, the instruction is printed. When the set of instructions has been completely searched in this manner, the next consecutive half-word is printed and the search is begun again. This continues until the set of half-words is exhausted. A control card specifies the location of the first and last instructions and the location of the first and last half-words. When the search indicated by a control card is complete, the program stops and pressing the START button causes the program to read the next control card. The ranges indicated may be of unit length.

This program is quite useful after having the machine COPY CHECK with control being where it was never intended to be. The offending transfer order can readily be detected by searching the code for references to this half-word.

d. 785 is a program to compare original ordinary binary program cards with the program stored in the machine. 786 is identical, except that it compares regional binary cards with the stored program. The program first transfers all of memory, except for the contents of -0000 and -0002, to a drum. It then reads the first card of the coders program, reads the corresponding set of words from the drum, and then prints all half-words that do not agree. The print-out consists of the location of the half-word, the half-word from the program card, and the half-word from the drum. This process continues until all program cards have been checked. The listing is double-spaced after the printing for each card.

This program has been very widely used, not only in debugging codes, but also in the detection of machine errors. If the code has been mutated by a memory malfunction, this will appear in the listing along with the instructions with variable addresses. The incorrect instruction can be spotted almost immediately by a person familiar with the program. Usually, the appropriate action can then be undertaken without delay. Occasionally 785 is used to check a corrected binary deck against an old binary deck where the listed discrepancies are assumed to be known.

Tracing Programs

a. 794 is a tracing program that can be used in one of two ways: (a) To list, all instructions with a specified operation, and (b) to list all instructions whose addresses lie within a specified range. All listing is done during the execution of the program. A sense switch determines whether the program should or should not stop on negative transfer orders. A second sense switch is used to inform the program as to whether operation or address tracing is desired. It is also possible, by means of a third switch, to have the instructions READ, WRITE, and READ BACKWARD "dummy" executed. If these orders are not "dummy" executed, control is relinquished by the tracing program and given to the input-output program involved. Information printed includes the status of the overflow indicator, the overflow bits, the location of the instruction (octal), the instruction (octal), the sign and first 17 bits of the accumulator (octal), the sign and complete contents of accumulator, MQ, and storage referred to in the address part of the instruction (all as decimal fractions).

b. 796 is a tracing program that lists all transfer and sense orders as they are being executed, i.e. 796 traces logic. A control card determines the location of the first instruction to be traced. A conditional transfer or sense that is not executed, i.e., has no effect on control, is not printed. No sense switches are used by 796. The program has been coded for a two electrostatic frame 701. The application of the program is readily apparent.

c. 795 is a high-speed tracing program which can be used with either a single or double electrostatic frame machine. Provision is made to accommodate any number of "traps", a "trap" being defined to be a portion of the coder's program which is to be traced. If a portion of the program is not being traced it is executed at full speed. After the coder's program has been loaded, 795 is loaded with n+1 control cards. The first n cards designate the range of the n desired traps and the n+1st card contains the location of the first instruction (R) of the coder's program. On each of the n trap cards is the location of the first instruction of the trap (M_i) and the location of the last instruction of the trap (N_i), $M_i \leq N_i$. As 795 reads the ith trap control card, it replaces the instruction M_i with a transfer to a portion, D_i, of the tracing program. Then 795 stores M_i, N_i and the contents of M_i in the D_i block. It then reads the i+1st control card and repeats the procedure in the D_i+1 block, continuing to read control cards until an "R" card is reached, whereupon control is transferred to R. Each D_i block is 6 half-words in length, hence the number of traps which may be specified is limited to the amount of space which is available in the machine for the trap table (D_i) block.

When the coder's program reaches the instruction M_i, control is transferred to 795 and tracing begun, with or without printing depending on the position of a sense switch. When instruction N_i is reached, it is traced and afterward control is relinquished by 795 and given to the coder's program at instruction N_i+1.

When a trap is encountered, the paper is spaced, and the contents of the accumulator, MQ, and overflow bits before the execution of the first instruction are printed, provided the "print" switch is on. The paper is spaced whether printing occurs or does not occur. If the instruction READ, WRITE, or READ BACKWARD is encountered while tracing, it is listed but not executed. All COPY orders are "dummy" executed by loading the MQ with the contents of the memory location referred to in the COPY instruction. All other instructions, including $\pm$ SENSE 40_8, and $\pm$ 00 and $\pm$ 01 transfers between the first and second banks of memory, are executed.

If, while a particular trap is being traced, the coder no longer wishes to trace this trap, he may "erase" this trap by depressing a sense switch which will replace the "transfer to tracing" order in M with the original instruction.

The information printed consists of the location of the instruction; the instruction; the status of the overflow indicator; the overflow bits; and the sign and contents of the accumulator, MQ, and storage location being referred to in the address part of the instruction in octal and decimal.

Conclusion

These programs constitute the major portion of our library of programs used in debugging problem programs. Any of these programs are available to IBM 701 installations. We at Los Alamos would be interested in any suggestions for the improvement of these programs. We would also be interested in any ideas for debugging programs that you have utilized. We feel that only through the exchange of ideas can each installation benefit from the experience gained at other installations.

July 31, 1953

TO: ALL 701 USERS

FROM: 701 PROGRAMMING SECTION, GROUP T-1

SUBJECT: ASSIGNMENT OF 701 TIME

Occasionally a 701 user finds that he has lost his time on the machine because of machine error. If he will write his troubles into the 701 log, he will be given special consideration in the assignment of the next day's time, since the logs of the previous day are consulted in the morning when time is assigned. If the 701 user has lost his time because the machine was turned over to the 701 engineers for maintenance, he will be given time the next day if it is at all possible.

Allan Benson

Allan Benson

Willard Bouricius

Willard Bouricius

TO: 701 Users July 9, 1956

The 701 EDPM is scheduled to be returned to IBM during September, 1956. If any user of 701 feels that he cannot have his problems completed or recoded for the 704 by that time, please contact Jack Mann, E-101A, by memo or telephone 2-5913.

Jack Mann

TO : Operators of 701 and 704 Computers December 20, 1955

FROM : T-1

SUBJECT: Important Changes in Logging Machine Time.

The record keeping for 701-704 machine usage has been extended in two points because of budgetary reasons:

1. All current problems are classified as to "problem category". This can be changed from run to run by telling dispatcher at time of particular run; otherwise the problem will be logged as originally entered in log book.
2. The group designation will now show "for whom" the problem is being run. This can be changed also by telling dispatcher at time of any particular run.

Subroutine for Calculating Clebsch-Gordan

Coefficients in Fixed Point, $\left[C^{abc}_{\alpha\beta\gamma}\right]$.*

Program occupies storage $400_{(10)}$ to $999_{(10)}$.

$$(a + b + c) \leqq 25$$

Load full word constants times 2^{-17} as follows:

$a \longrightarrow$	$766_{(10)}$	$1376_{(8)}$
$b \longrightarrow$	$768_{(10)}$	$1400_{(8)}$
$c \longrightarrow$	$770_{(10)}$	$1402_{(8)}$
$\alpha \longrightarrow$	$772_{(10)}$	$1404_{(8)}$
$\beta \longrightarrow$	$774_{(10)}$	$1406_{(8)}$
$\gamma \longrightarrow$	$776_{(10)}$	$1410_{(8)}$

Enter program by basic linkage as follows:

α	R ADD	α
$\alpha+1$	TR	$396_{(10)}$
$\alpha+2$	Control returns here with answer in accumulator.	

STOPS: Indicate scaling difficulty, see Bertha Fagan or Max Goldstein.

$677_{(8)}$	overflow in constant term
$1153_{(8)}$	Kth term in sum ≥ 4

Coded by: Bertha Fagan & Max Goldstein

---

*See Simon, A. "Numerical Table of Clebsch-Gordan Coefficients", ORNL-1718 Special, for formulas.

TO: H. G. Kolsky (T-5) July 22, 1954

FROM: Edward A. Voorhees

SUBJECT: 701 Coding Course Information

SYMBOL: T-1-03

The 701 Coding Course will be held daily except Tuesday for a period of three weeks, August 9 through August 27. On Monday, Wednesday and Thursday the class will meet from 8:10 A.M. to 9:15 A.M. in the Rhines Raum (E-215). On Friday the class will meet from 8:10 A.M. to 10:00 A.M. in the W-Division Conference Room (ϒ-251). The extra time on Friday will be spent in coding for the 701.

It is our hope that at the conclusion of the course, each participant will be prepared to code intelligently for the 701. To this end we anticipate assigning selected reading and the coding of small problems to be done outside class. Your cooperation in encouraging your representatives to the class in this matter will be appreciated.

An outline of the course and other information regarding the course will be distributed to the members before the first meeting. The following members of your group are enrolled in the course:

Robert O. Bardwell

It would be appreciated if personnel changes were reported promptly.

The class is composed of 35 individuals from 18 different groups. We are anticipating holding a similar class late in the fall for those who were unable to attend this class. An announcement will be made in the future regarding the second class.

Edward A. Voorhees

Edward A. Voorhees

Edward A. Voorhees (T-1)
701 Programming Section
(2-3901)

HARWOOD G. KOLSKY
T-5
G-141

TO : Group and Division Leaders July 6, 1954

FROM : Edward A. Voorhees

SUBJECT: 701 Coding Course

SYMBOL : T-1-03

A short course in the operation and coding of problems for the 701 will be given during the summer, if a sufficient number of people indicate that they will attend such a course. Projected plans are to meet for an hour a day for a period of two or three weeks. Topics proposed for discussion include: Description of the 701, Flow Diagramming, Use of Utility Programs, General Coding, and Coding in Dual.

If the work of your group would benefit now or in the future by having one or several members of your group trained in the use of the 701, would you please submit the names of those individuals who plan to attend the course to

Edward A. Voorhees (2-3901)
Group T-1.

It would be appreciated if the names of those planning to enroll are received by July 16.

Edward A. Voorhees

Edward A. Voorhees
701 Programming Section

EAV:bb

TO: 701 and CPC Users June 22, 1955

In order to clean up for open house and to be ready for moving to the new building, we have to clean up all cards not in files. Also, we would like to limit T-1 files to current problems.

All cards and papers labeled with names will be checked with that person. Otherwise, they will be put in the CPC room on top of files opposite key punchers until July 8, then will be thrown out.

All groups using T-1 files will be asked to remove all cards that do not pertain to current problems.

Please see Jack Mann, T-1, as soon as convenient to determine what can be thrown out or can be stored outside of the current files.

Information as to special CPC or 701 boards which can be released for other uses would also be appreciated.

Sept. 12, 1955.

Tentative T-1 Machine Schedule

1. At present

   2 - 701's

   3 - CPC's

2. Oct. 1, 1955

   2 CPC's discontinued to make room for 704.

3. Nov., 1955

   2 - 701's

   1 - 704 (4096 words)

   1 - CPC

4. April, 1956

   Move to new administration building.

   1 - 701 (2 bank)

   2 - 704's (1 - 4096 words, other 8192 words)

   1 - CPC

5. Aug., 1956

   3 - 704's (2 - 4096 words, other 8192 words)

   1 - CPC

704 Coding Seminars for experienced 701 coders will start early in October.

704 Coding Classes for new coders will start at a later date.

TO: 701 Users

June 29, 1955

We have been cautioned by the Assistant Director for Classification and Security about the use of code words to designate problems run on the 701 computers. Section 5.8, Use of Code Words, in "Primer on Security, 1955", should be followed in choosing names for these problems. Because these names are used only by Computing Groups within the Laboratory, they will not be registered, but the policy set up in section 5.8 should be followed in naming problems for the 701.

Do not use amateur codes.

Do not use names in any way descriptive of problem subject.

Index to Current Utility Programs

011 Read decimal instructions into specified locations of ES-1 or ES-2.

012 Load blocks of either full or half-word decimal data into ES-1 or ES-2.

014 Twelve-digit, decimal input with decimal and binary scale factors. Input is a half-word or full word per card into the ES-1 (or ES-2) location specified.

015 Double precision decimal input with decimal and binary scale factors. Input is two full words per card into ES-1 or ES-2. The location of the first full word is specified on the card.

017 Load full or half-word decimal data into ES-1 or ES-2 with specified address increment to obtain equally spaced words in storage.

025 Reads regional binary cards into specified locations, 43 half-words per card.

026 Load itself, read binary half-words into consecutive ES locations, read binary half-words back from ES locations and from check sum. Load to end of memory.

027 Reads regional binary cards into specified locations, 43 half-words per card, into ES-1 or ES-2.

028 Load itself into ES-1, read binary half-words into consecutive ES locations in ES-1 or ES-2, read binary half-words back from ES locations and form check sum. Load to end of memory of either ES-1 or ES-2.

081 Read octal instructions into specified locations in ES-1 or ES-2.

110 Print floating decimal data.

111 Print half-word floating decimal data.

112 Print half-word floating decimal data from ES-1 or ES-2.

186 Print contents of electrostatic memory in octal.

188 Searches memory (ES-1 or ES-2 or both) for all references to a given address and prints them in octal. This program destroys the first two full words in ES-1, but otherwise leaves ES-1 and ES-2 unchanged.

189 Prints all transfer orders in octal, from one or two banks of memory. Destroys the first two words in ES-1, but otherwise leaves ES-1 and ES-2 unchanged.

210 Label punched cards with decimal integer in columns 1-8.

223 Punch in binary consecutive half-words of ES.

224 Punch in binary consecutive half-words from ES-1 or ES-2.

321 {Read / Write / Read Backward} full words {from / on} any tape without check sum.

322 Dump memory on alternate tapes; read back a selected dump.

323 Two-bank tape dump program.

400 Sin x.

401 Storage check sum.

409 Fixed point tan^{-1}.

410 Integer Root.

411 Sinh x.

413 Cube Root.

417 Cos x.

426 Cosh x.

432 Double Precision Fixed Point e^x.

450 Loan an nth order symmetric matrix, check the matrix for symmetry and then load 451.

451 Eigenvectors and eigenvalues of a real symmetric matrix of nth order $(2 \leq n \leq 31)$.

520 {Read / Write} full words {into / from} consecutive locations of ES-1 or ES-2 {from / onto} any drum.

526 Write all of ES on drum #1 with the exception of full words -0000 and -0002 (not regional).

527 {Read / Write} full words {into / from} consecutive ES locations {from / on} any drum.

607 Regional assembly

a) Assign absolute locations and addresses to a regional program.
b) Expand or contract a regional program, and, if expansion, insert new orders consecutively in the program.
c) Change regional indices.
d) Convert a twelve-digit fractional number in columns 45-57, scale according to the decimal and binary factors specified in columns 58-61, enter as either half-word or full word and assemble.

e) Print the original regional information and comments on the card, the final regional indices, location, operation and address in octal.
f) Punch binary cards for loading with 026, 028 or allied programs.
g) Punch regional binary cards for loading with 025 or allied programs.
h) Punch decimal regional cards, with the changed regional information, and the original comments (only one of the three punch programs may be selected during an assembly, but any or all of the other functions may be performed).

608 Same operations as 607 except the regional decimal punching is not allowed, and two new control cards have been added.

620 Regional binary assembly program.

703 Set drums, ES to zero and rewind tapes.

704 Set drums, ES to zero.

706 Clear ES to zero.

707 Clear ES-1 and ES-2 to zero.

720 Loads itself into ES-1, reads control cards which specify blocks of memory in ES-1 and/or ES-2 to be compared to corresponding contents of drums. Discrepancies are punched out in binary full words.

781 Search memory for transfers to M.

782 Search memory for stores to M.

785 Compares original program cards with program stored in electrostatic memory and prints out all half-words that do not agree.

786 Compares original regional binary cards with program stored in electrostatic memory and print out all half-words that do not agree.

787 Compares original program cards with program stored in ES-1 and ES-2 and prints out all half-words that do not agree.

788 Compares original regional binary program cards with program stored in ES-1 and ES-2 and prints out all half-words that do not agree.

790 Tracing.

791 Determine the cause of an overflow.

794 Tracing with optional operation or address-range selection.

795 Tracing with traps for a one-bank or two-bank memory.

796 Trace logic (one- or two-bank machine).

797 Tracing with traps for a one bank 701.

798 Tracing with traps a one bank program with 798 in the second bank.

820 Check binary cards for proper check sum without destroying memory.

924 Dump-Load Using Tape.

925 Reproduce binary cards with correct check sum.

926 Reproduce regional cards with correct check sum.

982 Prints contents of electrostatic memory in octal. Destroys only the first two full words, leaves the rest of ES unchanged.

983 Print sections of electrostatic memory by means of control cards or MQ entry buttons.

May 10, 1955

404 ln x

INPUT: Enter with x in MQ scaled at t.

Calling Sequence:

α	R Add	α
$\alpha + 1$	Tr	OFO
$\alpha + 2$	$\pm$ 00	t (Sign must be the sign of t)
$\alpha + 3$	Control returns here with ln x in MQ scaled at t = 5.	

DESCRIPTION: ln x is computed by the Rand approximation Sheet 55. For the computation, the approximation formula is converted to

$$\ln x = \sum_{i=1}^{i=8} a_i\left(\frac{x}{2^K} - 1\right)^i + K \ln 2, \quad 1 \leq \frac{x}{2^K} \leq 2$$

and the program finds the appropriate value of K. x must be such that $|\ln x| < 32$. t can be positive or negative.

STOPS: $\alpha + 2$ (coder's program): If $|\ln x| \geq 32$ stop occurs here with overflow part of ln x in Acc. and the rest of ln x in MQ. This stop can fail if $t > 183$ or if $t < -148$.

$\alpha + 2$ (coder's program): If $x \leq 0$ stop occurs here with zero in MQ and x in Acc.

STORAGE: OAO thru OA2

OBO thru OB18 (OBO must be even)

OEO thru OE2 (OEO must be even)

OFO thru OF39

Coded: J. K. Everton, checked out & written, J. K. Everton

Kolsky

Re-Sorted out July '54
Sorted again Apr '55

T-1 701 SECTION
UTILITY MANUAL

The utility programs described in this manual were coded, checked out, and the explanations of them written by the 701 programming section of T-1. The manual should be kept in loose leaf form, as additions to it will be distributed whenever other utility programs are checked out. Any comments or suggestions regarding the programs or the manual will be appreciated.

Dorothy T. Monk
Dorothy T. Monk

Distribution: 701 List

GENERAL PURPOSE UTILITY PROGRAMS

EMPLOYED BY 701 PROGRAMMING SECTION

Catalog and library reference numbers:

Each general purpose utility routine is named for library and cross reference with a three digit decimal number. The first digit of this number specifies the purposes of the routine, and is assigned according to the following definitions:

0 = read cards

1 = print

2 = punch cards

3 = {read from / write on} tapes

4 = special purpose sub-routines

5 = {read from / write on} drums

6 = unspecified

7 = debugging routines

8 = diagnostic test programs

9 = combination codes

The second digit specifies the base or number system primarily involved in the input or output of the routine, assigned as follows.

0 = base not relevant

1 = decimal

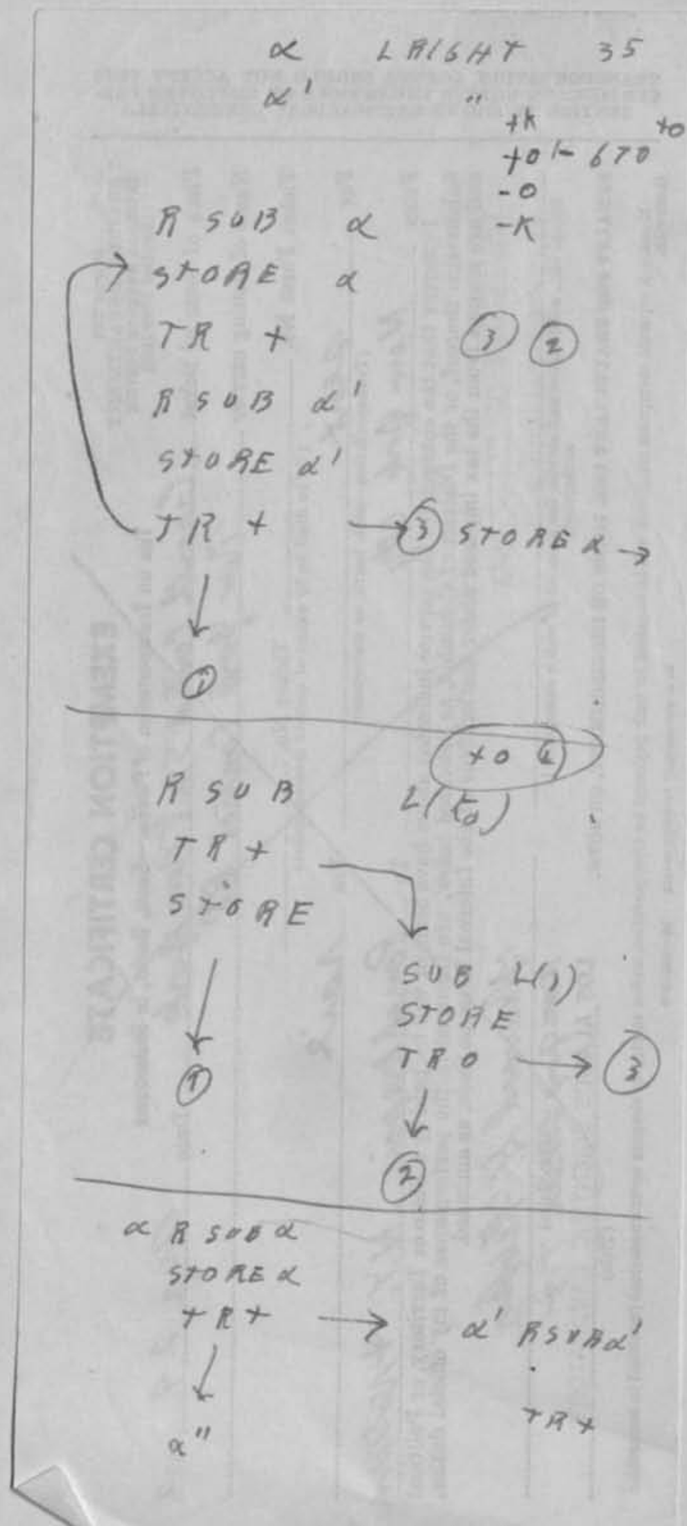
2 = binary

8 = octal

9 = combination of bases

The third digit is the number of the particular routine of the type specified by the first and second digits. For example, routine

022 is routine #2 to read binary cards (0 means read cards, 2 means binary, and the last 2 is just a label to differentiate 022 from other routines of the 02 block). Routine 783 traces (7 = debugging), prints out in octal (8), and is debugging octal routine #3.



T-1's utility programs are being located in three absolute utility regions, as follows:

Region	Begins At	Begins At
A	0	0
B	2048_{10}	4000_8
C	3584_{10}	7000_8

The decks, descriptions in the utility manual, and listings will be labeled with the proper absolute region (A, B or C) if they are absolute, and will be labeled R if they are regional. The letter occurs after the number of the program. These are not to be confused with IBM utility programs where the letters are before the number of the program and do not refer to absolute utility regions.

For a given program there is one page in the manual for each program's absolute decks, following a general and complete regional explanation of the program. The number of a program is the same for all its locations. A regional (unlocated) deck and listing are available to coders who wish to locate the program in some particular part of E.S. other than the several locations chosen for the program by T-1. For instructions for locating programs from regional cards, see the descriptions of regional programming and relocation available from T-1.

For each program there are three absolute decks (one for each of the absolute utility regions A, B and C) ready for immediate use. All the decks in the A region will be filed together, etc.

The self loading programs will begin at 0, 4000_8, and 7000_8. Included in these are two programs which will load binary cards, 021 and 024. They occupy, respectively, 52 and 50 half-words of storage.

IBM's FEJ035 (54 half-words) may be used for loading into the A absolute utility region only. 021, 024, and FEJ035 are each one binary card ; each takes a positive transition card (There are minor coding differences among them, but usually they can be used interchangably.).

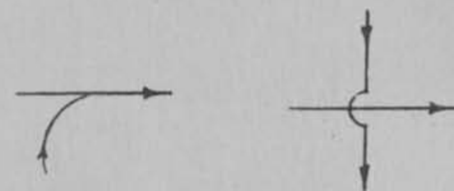
Most of the remaining absolute binary decks will begin immediately after 021. These binary decks are compact if possible; the programs are compact except for the erasable block E, which never has to be loaded. The E block is located in the same place as the binary loading program 021, 024 or FEJ035. Therefore the binary loading program is destroyed when the utility program it loads is run, so loading should, in general, be done with a loading card rather than by transfer to the loading program.

There are a few absolute decks checked out which are not located in A, B or C. These are filed under MISC and the heading cards give the storage occupied.

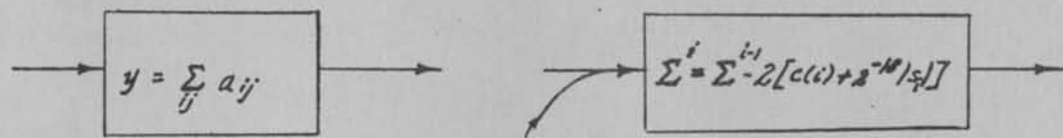
Suggested Symbols for Flow Diagrams

T - 1 701 Section

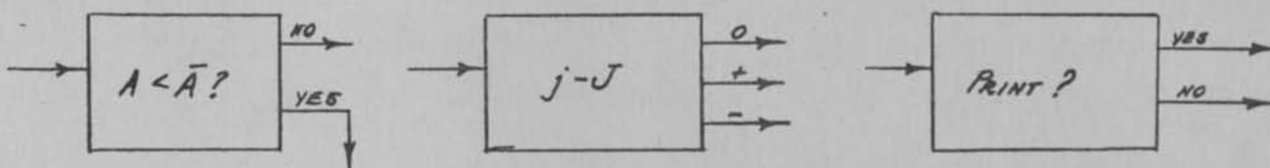
1. Course of control indicated by: ⟶



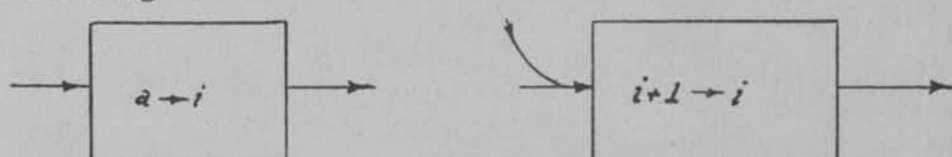
2. Operation or multiple operations box in linear sequence of control or in an induction loop; if in loop notation within the box should be general.



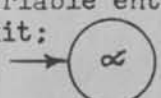
3. Decision or multiple decisions box (alternative or conditional transfer box); if two-choice decision , it is preferable to make the decision so that the two branches can be labeled "yes" and "no".



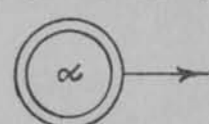
4. Substitution box. a ⟶ i is read "The variable or index i takes on the value a (until the next substitution involving i)," or sometimes, "a to i". i + 1 ⟶ i is read,"The variable i takes on the value: 1 plus the value of i before control reached this substitution." i + 1 ⟶ i can be interpreted as follows: Operate on the i on the left with all predeeding substitutions involving i; from this value determine the new value of i(in this case by adding 1). Substitute this new value of i everywhere following this substitution box until the next substitution box involving i.



5. Variable entrance: control reaches this point from "start" or from the exit:

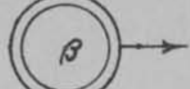


V. exit

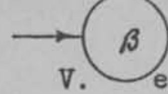


V. entrance

6. Variable exit: control goes from this point to "stop" or to the entrance:

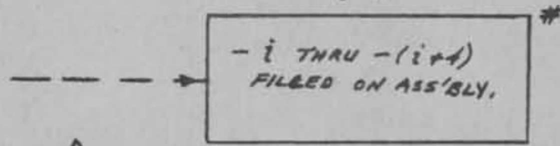


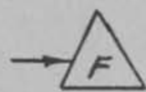
V. entrance



V. exit

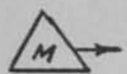
7. Explanation box. Broken line goes to explanations, notational changes, storage content notes, statements of validity, etc. <u>not</u> affecting course of control. Explanation box is denoted by #.

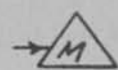


8. Page change boxes. 

means control goes to 

on some other page.



means control came from 

on some other page.

SO_2 ASSEMBLY OF SELF LOADING PROGRAMS

Because SO_2 does not punch self-loading binary cards (SO_2 punches binary cards with check sums, V, and R in the 9 row) it is necessary to use the following or some similar procedure for assembling a self-loading program.

1. Instruction cards. List the instructions on the coding sheet for the keypuncher in the order in which they will be read in by the 701 off the final binary self loading card(s). Assign consecutive dummy locations to the instructions with the dummy index, say H, different from the index used in the address parts of the instructions, which is the "true" location index, say F. Then H0 contains the instruction which will become the first half word in the binary self loading card, i.e., the half word which should be in the 9 row, columns 9 thru 26, H1 contains the second instruction on the SL card, etc.

Example:

Keypunchers Sheet	Loc on Self Loading Card To Be Produced	True Location	Instruction	
Col 9 12 22				
000H0000-3100F0002	9 row col's 9 thru 26	F0	-31	F2
000H0001-3100F0030	9 row col's 27 thru 44	F1	-31	F30
000H0002+0100F0017	9 row col's 45 thru 62	F30	01	F17
000H0003+0000R0010	9 row col's 63 thru 80	F31		R10

2. Origin cards. Assign for F0 the true location of the first instruction in the self loading program. This is the location which must be entered on the instruction entry keys before pressing the load button to run the self loading program (after it has been assembled).

FO must be even.

Assign for HO some even location such that the H block will not interfere with K04, 222, or 223 (one of which is used to punch the binary self loading card(s)) or with the F block.

3. Assembly. Assemble as usual with SO_2. Punch in binary and print. The listing will have only the dummy H locations, no true locations. Prepare a control card for K04, 223 or 222; R equals HO and V equals the number of half words in the self loading program. (Note that R & V must be even for K04 and 223.) Reset E.S. to O's with ∠HO1. Load the binary cards produced by SO_2, the punch program, and punch out the H block. The resulting card(s), punched by K04, 223 or 222, will be self loading into FO if the program was coded correctly.

MEMORY SUMS

It is always desirable to sum information being transferred from E.S. to any of the input-output components. Usually summing takes relatively little time, and it insures that the information gets into or out of E.S. correctly, as well as telling the operator when he is getting occasional machine errors.

The notation, form, and card layout for check sums and the location and number of consecutive words involved in the transfer of binary information has been standardized to agree with that of IBM for all the utility programs described in this manual, unless specifically stated that the sums or notation are not standard. The following definitions may be taken as valid wherever they appear in this manual. If the notation is not standard, different symbols will be used.

R: The initial address, R, usually positive, is the E.S. location of the first of the consecutive half-words (or if R is negative, the first full word; if R is -, it must be even) which are to be loaded into or dumped from E.S. Therefore,

$$(0000 \leq |R| \leq 7777)_8 = (0000 \leq |R| \leq 4095)_{10}.$$

Ordinarily R will be more restricted, since one cannot usually load or dump the part of E.S. which is occupied by the program which is doing the loading or dumping.* If the memory sum is actually kept in E.S. (instead of being discarded after the 701 has verified that it has loaded or dumped the information correctly), the locations +R and +(R+1) are usually reserved for this memory sum, in which case R must be even.

V: The half-word count, V, is the number of half-words which the coder may load into or dump from E.S. with the utility program. V must be

* Exception: 023

positive; if the memory sum is actually being kept, V does not include the two half-words required to store the memory sum. V and R are always subject to the following restrictions in the two cases, and usually are more limited by the specific load or dump program being used. L is the location of the last half-word loaded or dumped.

1. Memory sum actually stored: $(0000 \leq |R| \leq 7777)_8 = (0000 \leq |R| \leq 4095)_{10}$

$$(0 \leq V \leq 7776)_8 = (0 \leq V \leq 4094)_{10}$$

$$L = V + |R| + 1$$

2. Memory sum discarded: $(0000 \leq |R| \leq 7777)_8 = (0000 \leq |R| \leq 4095)_{10}$

$$(0 \leq V \leq 10000)_8 = (0 \leq V \leq 4096)_{10}$$

$$L = V + |R| - 1$$

Memory sums are of the following general type: $- 2 \sum_i \left[(\text{1st 18 bits})_i + (\text{2nd 18 bits})_i\right]$. Consider the half words and sum as integers.

S: The card check sum, S, is defined to be minus twice the sum of "all other half-words" with the sign bit figured as just another bit. "All other half-words" means all other half-words of a card or block except the card check sum itself or σ (see below), i.e., V and R and all half-words to be loaded except S or σ. More precisely,

$$S = - 2 \left[\sum |w| + 2^{-17} N(w)\right] ,$$

where w ranges over all half-words of the card or block to be loaded or dumped (except S itself or σ) and V and R, N(w) is the number of negative half-words, and $\sum$ means "the sum of". S is always negative.

σ: The storage check sum, σ, is defined to be minus twice the sum of "all other half-words" with the sign bit figured as just another bit.

"All other half-words" means here all other half-words to be loaded or dumped except σ itself or S. V and R are not included.

$$\sigma = -2 \left[\sum |u| + 2^{-17} N(u) \right],$$

where u ranges over all half-words of the card or block to be loaded (except σ itself or S), and does not include V and R, N(u) is the number of negative half-words, and $\sum$ means "the sum of". σ is always negative. Note that

$$-2 (|R| + V) = S.$$

For reading cards R, V, and S or σ are punched in binary in the 9 row as follows:

column 9, sign of S or σ = -

columns 10 thru 44, S or σ

columns 51 thru 62, V

column 63, sign of R

columns 69 thru 80, R.

The correct R, V, and S or σ (whichever is required) may be given for each card, or for an entire block of cards, whichever is called for by the loading program. If the R, V and S or σ are for the entire block, they are punched in the 9 row of the first card of that block.

A AND E BLOCKS

The regional index 00A has been used wherever possible in the T-1 utility programs for the universal constants block. This block consists of:

$$00A0000 + 00R0000 = L(0),$$

$$00A0001 + 00R0001 = L(1),$$

and

$$00A0002 + 00R0002 = L(2)$$

Also - 00A0000 = - L(1) = $L(2^{-35})$; therefore in some programs it is required that the origin given for 00A0000 be even. The A block is never destroyed during a run of the utility program, and these constants may be used by the coder at any time. Each utility program has in it only the constants of the A block which that program uses.

The regional index 00E is used for the erasable or temporary storage block. It is assumed on entry to the utility program that this block may contain anything; if it is necessary that the E block be cleared, the utility program clears it. The coder may use the E block as temporary storage at any time except when the utility program is being run. The utility program does not clear its E block after use; therefore the coder must clear it before using if it is necessary that his E block be initially 0. It is usually required that the origin given for E0 be even.

The index 00R means invariant. No origin is given for 00R; locations or addresses with this index are in absolute decimal.

MEMORY SUMS

It is always desirable to sum information being transferred from E.S. to any of the input-output components. Usually summing takes relatively little time, and it insures that the information gets into or out of E.S. correctly, as well as telling the operator when he is getting occasional machine errors.

The notation, form, and card layout for check sums and the location and number of consecutive words involved in the transfer of binary information has been standardized to agree with that of IBM for all the utility programs described in this manual, unless specifically stated that the sums or notation are <u>not</u> standard. The following definitions may be taken as valid wherever they appear in this manual. If the notation is not standard, different symbols will be used.

R: The initial address, R, usually positive, is the E.S. location of the first of the consecutive half-words (or if R is negative, the first full word; if R is -, it must be even) which are to be loaded into or dumped from E.S. Therefore,

$$(0000 \leq |R| \leq 7777)_8 = (0000 \leq |R| \leq 4095)_{10}.$$

Ordinarily R will be more restricted, since one cannot usually load or dump the part of E.S. which is occupied by the program which is doing the loading or dumping.* If the memory sum is actually kept in E.S. (instead of being discarded after the 701 has verified that it has loaded or dumped the information correctly), the locations +R and +(R+1) are usually reserved for this memory sum, in which case R must be even.

V: The half-word count, V, is the number of half-words which the coder may load into or dump from E.S. with the utility program. V must be

* Exception: 023

positive; if the memory sum is actually being kept, V does not include the two half-words required to store the memory sum. V and R are always subject to the following restrictions in the two cases, and usually are more limited by the specific load or dump program being used. L is the location of the last half-word loaded or dumped.

1. Memory sum actually stored: $(0000 \leq |R| \leq 7777)_8 = (0000 \leq |R| \leq 4095)_{10}$

$(0 \leq V \leq 7776)_8 = (0 \leq V \leq 4094)_{10}$

$L = V + |R| + 1$

2. Memory sum discarded: $(0000 \leq |R| \leq 7777)_8 = (0000 \leq |R| \leq 4095)_{10}$

$(0 \leq V \leq 10000)_8 = (0 \leq V \leq 4096)_{10}$

$L = V + |R| - 1$

Memory sums are of the following general type: $-2 \sum_i \left[(\text{1st 18 bits})_i + (\text{2nd 18 bits})_i\right]$. Consider the half words and sum as integers.

S: The card check sum, S, is defined to be minus twice the sum of "all other half-words" with the sign bit figured as just another bit. "All other half-words" means all other half-words of a card or block except the card check sum itself or σ (see below), i.e., V and R and all half-words to be loaded except S or σ. More precisely,

$$S = -2\left[\sum |w| + 2^{+17} N(w)\right],$$

where w ranges over all half-words of the card or block to be loaded or dumped (except S itself or σ) and V and R, N(w) is the number of negative half-words, and $\sum$ means "the sum of". S is always negative.

σ: The storage check sum, σ, is defined to be minus twice the sum of "all other half-words" with the sign bit figured as just another bit.

"All other half-words" means here all other half-words to be loaded or dumped except σ itself or S. V and R are not included.

$$\sigma = -2\left[\sum |u| + 2^{+17} N(u)\right],$$

where u ranges over all half-words of the card or block to be loaded (except σ itself or S), and does not include V and R, N(u) is the number of negative half-words, and $\sum$ means "the sum of". σ is always negative. Note that

$$\sigma - 2(|R| + V) = S.$$

For reading cards R, V, and S or σ are punched in binary in the 9 row as follows:

column 9, sign of S or σ = -

columns 10 thru 44, S or σ

columns 51 thru 62, V

column 63, sign of R

columns 69 thru 80, R.

The correct R, V, and S or σ (whichever is required) may be given for each card, or for an entire block of cards, whichever is called for by the loading program. If the R, V and S or σ are for the entire block, they are punched in the 9 row of the first card of that block.

SO_2 ASSEMBLY OF SELF LOADING PROGRAMS

Because SO_2 does not punch self-loading binary cards (SO_2 punches binary cards with check sums, V, and R in the 9 row) it is necessary to use the following or some similar procedure for assembling a self-loading program.

1. Instruction cards. List the instructions on the coding sheet for the keypuncher in the order in which they will be read in by the 701 off the final binary self loading card(s). Assign consecutive dummy locations to the instructions with the dummy index, say H, different from the index used in the address parts of the instructions, which is the "true" location index, say F. Then H0 contains the instruction which will become the first half word in the binary self loading card, i.e., the half word which should be in the 9 row, columns 9 thru 26, H1 contains the second instruction on the SL card, etc.

Example:

Keypunchers Sheet	Loc on Self Loading Card To Be Produced	True Location	Instruction	
Col 9 12 22				
000H0000-3100F0002	9 row col's 9 thru 26	F0	-31	F2
000H0001-3100F0030	9 row col's 27 thru 44	F1	-31	F30
000H0002+0100F0017	9 row col's 45 thru 62	F30	01	F17
000H0003+0000R0010	9 row col's 63 thru 80	F31		R10

2. Origin cards. Assign for F0 the true location of the first instruction in the self loading program. This is the location which must be entered on the instruction entry keys before pressing the load button to run the self loading program (after it has been assembled).

FO must be even.

Assign for HO some even location such that the H block will not interfere with K04, 222, or 223 (one of which is used to punch the binary self loading card(s)) or with the F block.

3. Assembly. Assemble as usual with SO_2. Punch in binary and print. The listing will have only the dummy H locations, no true locations. Prepare a control card for K04, 223 or 222; R equals HO and V equals the number of half words in the self loading program. (Note that R & V must be even for K04 and 223.) Reset E.S. to 0's with ⱠHO1. Load the binary cards produced by SO_2, the punch program, and punch out the H block. The resulting card(s), punched by K04, 223 or 222, will be self loading into FO if the program was coded correctly.

How To Reproduce Binary Cards

Use reproducer just outside 701 Room. Use 80-80 punch and compare board. Make sure no wires have been pulled. Do not use 80-80 switch board -- it is unreliable. Do not use reproducer in keypunch room -- it jams on binary cards.

1. Reproduce deck with punch and compare switch on.
2. Switch the two decks to opposite hoppers, face up, 12 edge first, and compare only.
3. Interpret with board that has 1st 8 reading brushes wired to 1st 8 type bars, and no other wires. (for utility decks).

FROM: IBM Corporation

TO: 701 Programmers and Coders

ATTENTION

DATE: July 7, 1953

cc:

SUBJECT: Supplemental Information To "Principles of Operation, Type 701 and Associated Equipment"

1) Additional Information - "Extract" Order

Please refer to our memorandum dated May 28, 1953 on the subject of "Changes to 701". The enclosure to this memorandum listed several short examples on the use of the new "Extract" order. In these examples, the terminology "-STORE A A" was erroneously used. This terminology should be replaced with "EXTRACT A". In addition, the values of x and y are assumed to be positive.

The "Extract" order affects all 36 bit positions of a word, i. e., the bit is treated in the same manner as any of the other bit positions.

The time for obtaining and executing the "Extract" operation is the same as that for any "STORE" type instruction, i. e., 16ms., unless one of the previous 12 instructions was a multiplication in which case the "Extract" order will require 24ms.

2) A Method of Decreasing the Time Necessary to Run Programs Which Contain Drum Reading or Writing Routines Followed by Another Input-Output Routine of Some Type.

The usual method of terminating the reading or writing of a drum unit record is to simply stop giving COPY instructions. However, a period of 1.28 milliseconds must elapse after the last COPY of the unit record before the drum disconnects from the calculator. If the next instruction of the program is an Input-Output Select instruction of some type, the execution of this instruction will be held up 1.28 milliseconds until the drum has disconnected. However, if the last Copy instruction of the drum unit record is followed by the instruction SET DRUM 0000, the drum will disconnect upon receipt of the SET DRUM instruction effecting a time saving of approximately 1.2 milliseconds. Note: The address part of the SET DRUM instruction must be 0000.

If no Input-Output routine follows a drum routine for at least 1.28 milliseconds, there is no benefit to be gained by the use of the SET DRUM 0000 instruction to obtain a quick disconnect. Rather, program time is increased by the amount of time (48 microseconds) necessary for the execution of the SET DRUM instruction.

3) Additional Information Relative to the Use of the Instructions Read Drum and Write Drum

The "Principles of Operation, Type 701 and Associated Equipment" contains the statement (on page 45); "Any number of instructions (except input-output instructions) may intervene between Read or Write and Set Drum and between Set Drum and the first Copy instruction."

It should be emphasized that if the drum has been selected to read or write and it is not desired to use the drum, at least one Copy instruction with an irrelevant address must be supplied so that the drum will disconnect from the calculator.

This type of situation arises frequently. For example, in order to offset the initial access time to the drum, one will normally select the drum well in advance of the time he intends to use it. However, between this original selection and the actual time of drum use, a fork in the program may have caused the calculator to follow a programming path which does not use the drum. In addition, this path would eventually make use of another Input-Output device. This situation would cause the calculator to wait until the drum has disconnected because only one Input-Output component may be selected at any one time.

Elizabeth A. Stewart

EAS:ld.

July 27, 1953

The following Utility Programs are obsolete. The same function can be performed by the Utility Program whose number is given on the right.

Obsolete	Use Instead
020	021 or 024
220	221
185	JTA 7

The pages pertaining to the obsolete programs in the Utility Manual should be removed. T-1 will not keep up the regional or binary decks to these programs in the 701 Room.

The following utility programs are additions to the utility manual:

402R	706R
403R	702 absolute table
705R-1, 2	705 absolute table

The following Utility Programs are now considered obsolete. The same function can be performed by other Utility Programs.

Obsolete	Use Instead
020	026
021	026
023	026
024	026
185	187
220	221
222	223
606	607
793	790
991	992
LCH 10 & 11	992
LCHO	706

The write-ups in the Utility Manuals referring to these programs should be removed. T-1 will not keep up the binary cards in the 701 room, but the binary cards will be available upon request from T-1.

Dura W. Sweeney
3/22/54.

H. Kolsky
T-5

The following Utility Programs are now considered obsolete. The same function can be performed by other Utility Programs.

Obsolete

010

013

020

021

023

024

080

220

221

222

606

992

LCH 0

LCH 2

LCH 10

LCH 11

The write-ups in the Utility Manuals referring to these programs should be removed. T-1 will not keep up the binary cards in the regular 701 Utility Files, but the binary cards will be available from a special drawer in Ready Room Files labeled "Obsolete Programs". Questions on programs to be used should be directed to T-1.

Dura W. Sweeney
8/9/54

Subroutine Library Conventions

The 701 programming section of group T-1 is developing a library of commonly used sub-routines, such as e^x, sin x, floating point addition, memory summing, etc. To make the sub-routines consistent and more useful, the following conventions have been adopted.

(1) fixed point functions. Argument in MQ, result in MQ.

Example: sin x = y. x must be prestored in the MQ, and the routine leaves y in the MQ.

(2) floating point.

E0 x (preserved)

E1 y (z)

-E2 a (preserved) → for functions

-MQ b (c)

Example 1: $a \cdot 2^x + b \cdot 2^y = c \cdot 2^z$; $a < 1$, $b < 1$, $c < 1$.

x, y, z are integers scaled by 2^{-17}. a, x, b, and y must be prestored in the above locations. The result c, z replaces b, y in the locations MQ and E1. a,z and x are preserved in -E2 and E0.

Example 2: $e^{b \cdot 2^y}$. The floating point argument b,y must be prestored in the MQ and E1. The result $c \cdot 2^z$ replaces b, y.

(3) double precision fixed point.

-EZ, -E4 a (preserved)

-E6, -E8 b (c) → for functions

Each double precision number is stored in two full words. The sign of the number must be the same as the signs of its two components.

Example 1: a + b = c. The arguments a and b must be prestored in the above locations. The result, c, replaces b in -E6, -E8. a is preserved in -E2, -E4.

Example 2: $e^b = c$. b must be prestored in -E6, -E8. c replaces b.

(4) double precision floating point.

E0	x	(preserved)	
E1	y	(z)	
-E2, -E4	a	(preserved)	→ for functions
-E6, -E8	b	(c)	

Example 1 $a \cdot 2^x \times b \cdot 2^y = c \cdot 2^z$. a, x, b, y must be prestored. The result, c, z, is put in E1; -E6, -E8. a and x are preserved. a, b, $c < 1$. x, y, and z are integers, scaled by 2^{-17} and put in half-words.

Example 2 $\tan^{-1} b \cdot 2^y = c \cdot 2^z$. b and y must be prestored. The result, c, z, replaces b, y.

Linkage entry should be used for all routines, as follows:

A	RADD	A
A + 1	TR	(to subroutine)
A + 2	Return of control from subroutine.	

Universal constants should be put in the OA block:

OA0	0
OA1	1
OA2	2

There should be nothing else in the OA block.

Any recommendations for library subroutines will be considered. Coders who code a subroutine which is not in the library would do others a service by using the above conventions and donating their routine to the library.

Don Monk

H. Kolsky
T-5

MODIFICATIONS TO "PRINCIPLES OF OPERATION, TYPE 701 AND ASSOCIATED EQUIPMENT" FOR TWO ELECTROSTATIC UNIT OPERATION

Pg. 13 (General Correction)

ELECTROSTATIC

The heart of the machine is the electrostatic storage unit, through which all information to and from all other components of the machine must pass. Electrostatic storage consists of a bank of cathode-ray tubes. Information is stored on the screen of each tube through the presence or absence of charged spots at certain locations on the screen. In this way, a certain number of binary digits (or "bits") may be stored on each tube. One electrostatic storage unit can accommodate 2048 full words or 4096 half words. However, two such units may be used to provide a maximum storage of 4096 full words or 8192 half words. Instructions for both one electrostatic storage unit and two electrostatic storage units will follow.

Principal advantages of electrostatic storage over other types is the very small time necessary to extract information from any given location and send it to the computing unit and the fact that the programmer has random access to any electrostatic storage location. Information is lost when the power is turned off.

Pg. 15 ADDRESS SYSTEM

MEMORY LOCATIONS

Full and half word locations in electrostatic storage, together with tapes, drums, printer, card reader and punch, are identified by a system of numerical addresses. In the case of two electrostatic frame operation, the same numerical addresses exist in each frame. A special method of inter-frame transfer is therefore required, and is described in the following paragraphs.

By means of a number, then, and proper control in the case of two electrostatic storage operation, we may tell the machine to refer to any information contained in electrostatic storage or to any component of the machine, provided only that we use the system to be described.

ELECTROSTATIC

The 4096 different locations for full words in double electrostatic storage are identified by the negative integers from -0000 to -4095. The 8192 possible locations for half-words in double electrostatic storage are distinguished by the positive integers from +0000 to +4095 and the status of a program-controlled TRIGGER which lights the ES2 light on the Operator's Panel. When this TRIGGER and light are ON, the half word location is in Electrostatic Storage Unit No. 2. When this TRIGGER is OFF, the half-word location is in Electrostatic Storage Unit No. 1. The relation between full and half word addresses is as follows: if - 2n refers to a full word location in Electrostatic Storage Unit No. 1, then + 2n identifies the left half-word, and + (2n + 1) the right half-word, into which the full-word location may be split; if - (2n + 1) refers to a full-word location in Electrostatic Storage Unit No. 2, then, also, +2n identifies the left half-word, and + (2n + 1) the right half-word, into which the full word location may be split. Thus, another bit must be programmed and remembered in order to fully identify the 8192 half-word locations in Double-Electrostatic Memory.

For example, if the full-word address is -1962, then the left half-word address is +1962 (ES1) and refers to the sign position and positions 1 to 17 of the full word. The right half word address is +1963 (ES1) and refers to positions 18 to 35 of the full-word location, position 18 being the sign position of the right half-word (Figure 1). If a full word is to be obtained from or supplied to electrostatic storage and, through design, a negative odd address is given (e.g., -1963), the result will concern the physical address (-1962) in Electrostatic Storage Frame No. 2.

The following instructions refer to memory for information during their execution:

SUBTRACT	LOAD MQ REGISTER
RESET AND SUBTRACT	MULTIPLY
SUBTRACT ABSOLUTE VALUE	MULTIPLY AND ROUND
ADD	DIVIDE
RESET AND ADD	COPY AND SKIP (WRITE)
ADD ABSOLUTE VALUE	

The following instructions store information in memory during their execution:

STORE	STORE CONTENTS OF MQ REGISTER
EXTRACT	COPY AND SKIP (READ)
STORE ADDRESS	

I. Single Electrostatic Storage Operation

1. If a full word is to be obtained from or supplied to electrostatic storage and, through error, a negative odd address is given (e.g., -1963), the result will be the same as if the next lower (in absolute value) negative even address (-1962) were given.

II. Two Electrostatic Storage Operation

1. If any instruction which requires a reference to electrostatic memory during its execution is received, and has a negative even address, the execution reference to electrostatic storage will be to ES-1 and in the form of a full word. For example:

   "-RADD 0100" will introduce the contents of ES-1, addresses 0100 and 0101 into the accumulator.

2. If any instruction which requires a reference to electrostatic memory during its execution is received, and has a negative odd address, the execution reference to electrostatic storage will be to ES-2 and in the form of a full word. For example:

   "-RADD 0101" will introduce the contents of ES-2, addresses 0100 and 0101 into the accumulator.

3. If the instruction, EXTRACT is received with a negative even address, the "EXTRACT" function will be performed between the contents of the accumulator and the full word address of ES-1. For example:

   "EXTRACT 0100" will perform the "EXTRACT" function between the accumulator and the contents of ES-1, addresses 0100 and 0101.

4. If the instruction, EXTRACT is received with a negative odd address, the "EXTRACT" function will be performed between the contents of the accumulator and the full word address of ES-2. For example:

"EXTRACT 0101" will perform the "EXTRACT" function between the accumulator and the contents of ES-2, addresses 0100 and 0101.

5. If the instruction "+SENSE 0040" is given, all future instructions with positive addresses which require a memory reference for execution will perform execution references to ES-1 in the form of a half word. For example:

0000	+SENSE	0040	
0001	+RADD	0100	(The contents of ES-1, address 0100)
0002	+ADD	0101	(The contents of ES-1, address 0101)

6. If the instruction "-SENSE 0040" is given, all future instructions with positive addresses which require a memory reference for execution will perform execution references to ES-2. For example:

0002	+ ADD	0101	(The contents of ES-1, address 0101)
0003	- SENSE	0040	
0004	+ RADD	0100	(The contents of ES-2, address 0100)
0005	+ ADD	0101	(The contents of ES-2, address 0101)

7. The execution of either a "STOP AND TRANSFER" or "TRANSFER" instruction with a positive address will cause all future instructions to be introduced from ES-1. For example:

INST. LOC. ES1	ES2	OP CODE	ADDRESS	REMARKS
--	0000	-ADD	0100	Assume the location of ADD to be address 0000, ES-2.
--	0001	+TR	0050	Instruction received from address 0001 in ES-2.
0050	--	-RADD	0100	Instruction received from address 0050, ES-1.

8. The execution of either a "STOP AND TRANSFER" or "TRANSFER" instruction with a negative address will cause all future instructions to be introduced from ES-2. For example:

0000	-ADD	0100	(Assume the location of ADD to be address 0000, ES-1)
0001	-TR	0050	(Instruction received from address 0001, ES-1)
0050	-RADD	0100	(Instruction received from address 0050, ES-2)

9. Normal instruction sequence is within a given electrostatic unit as follows:

(+4095) ES-1 is followed sequentially by (+0000) ES-1.
(+4095) ES-2 is followed sequentially by (+0000) ES-2.

10. The conditional transfer instructions, ± TRANSFER ON OVERFLOW, ± TRANSFER ON PLUS, ± TRANSFER ON ZERO, refer to the same electrostatic storage unit. For example:

    a. ± 2346 TR PLUS 4094 (If the address of the instruction "TR PLUS" is located in ES-1, if the conditional transfer is effected, the next instruction address will be 4094 of ES-1.)

11. Normal "Reset and Clear Memory," "Reset," "Operator's Reset" or "Load" will preselect operation from ES-1.

12. The "MANUAL" light used with single electrostatic storage operation is utilized as the select light for ES-2 on two electrostatic storage operation, and is labeled "ES 2."

13. A rotary switch available to the Customer Engineers--Calculator is included for the purpose of the following:

    Position 1 - Logical ES-1 operation directed to physical ES-1
    Logical ES-2 operation directed to physical ES-2

    Position 2 - Logical ES-1 operation directed to physical ES-2
    Logical ES-2 operation directed to physical ES-1

    Position 3 - Single electrostatic unit operation utilizing physical ES-1 only.

    Position 4 - Single electrostatic unit operation utilizing physical ES-2 only.

H. Kolsky
T-5

Utility Programs for a 2-bank 701.

In order to facilitate the modification of existing utility programs for a two-memory bank 701, the following conventions have been adopted by the T-1 programming section:

I. All utility programs will be located in the first memory bank.

II. All calling sequences for the utility programs will also be located in the first memory bank, and will be

β	+ Sense 40_8	(may be omitted if h.w. status is already in ES-1)
β +1	+ Store (β +j+1)	
β +2		The calling sequence for the utility program, as indicated in the program description for two-bank programs
⋮		
β +j		
β +j+1	[Intermediate exit]	

III. The coder's program will contain the following basic linkage to the utility program calling sequence:

α	+ R add α + 2	
α +1	+ Tr	
α +2	+ Tr α + 3	(if coder's program is in bank 1.)
	- Tr α + 3	(if coder's program is in bank 2.)

Control will then return to α + 3. The half-word status, however, is in ES-1, and hence α + 3 would normally contain a "Sense 40_8" instruction with the proper sign attached.

IV. EXAMPLE:

Coder's program in ES-2, utility program 110 to print out 7 words per line, 10 lines per block, 2 blocks per page, getting the data from ES-2 locations -1 thru -139.

Bank 2			Bank 1
α	+ R add α + 2	β	+ Sense 40_8
α +1	+ Tr β	β +1	Store β + $_8$14
α +2	- Tr α + 3	β +2	+ R add β + 2
α +3	- Sense 40_8	β +3	+ Tr 1F0
		β +4	+ 7, 10
		β +5	+ 0, 1
		β +6	+ 0, 139
		β +7	+ 0, t_1
		β +8	+ 0, t_2
		β +9	+ 0, t_3
		β +10	+ 0, t_4
		β +11	+ 0, t_5
		β +12	+ 0, t_6
		β +13	+ 0, t_7
		β +14	[exit]

V. Binary and regional binary cards will have the same form except that R (first word location) may be positive or negative. A positive R indicates that the half-words are located in ES-1; a negative R indicates that the half-words are located in ES-2.

VI. To provide for assembling and reading into, or punching from, either ES frame, the following programs will be provided:

A. 608: An assembly program of the same form as 607, with provisions for type #5 and type #6 control cards.

Type #5	All following type #0 or type #4 cards are located in ES-1.
Type #6	All following type #0 or type #4 cards are located in ES-2.

When 608 is originally loaded, or after the "Completed Assembly Stop" is reached, 608 will act as if it had received a type #5 control card.

B. 028: A two-card, self-loading program which will read binary half-words from the cards following and locate them in ES-1 or ES-2, according to the sign of R. The transition card from this program should contain 0 in columns 45-62 of the 9 row, and columns 9-44 may contain a "± Sense 40_8" instruction as well as a ± 00 or ± 01 transfer.

C. 224: A binary punch program of the same form as 221, with +R in the calling sequence if the words are to be punched from ES-1, -R in the calling sequence if the words are to be punched from ES-2.

D. 621: A program similar to 620, with provisions for R to be in either ES-1 or ES-2.

E. 029: A program which will load regional binary cards into ES-1 or ES-2.

In order to keep the size of the Console "Bibles" as small as possible for ease in using, T-1 has put programs in three classifications:

1. Current Programs
2. Math Subroutines
3. Seldom Used Programs

Current programs will be kept in the Console "Bibles". Math Subroutines and Seldom Used Programs will be kept in a "Bible" on the dispatcher's desk.

Binary cards for Current Programs will be kept in the same place, (Console and Ready Room files). Decimal regional cards for the Math Subroutines will be kept in the same place in the Ready Room files. Cards for Seldom Used Programs will be kept in a drawer in the Ready Room file marked Seldom Used Programs.

A list of the programs put in these classifications follows:

Current Programs - Console "Bibles"

011	526	925
012	527	926
014	607	982
015	608	983
017	620	Dual Stops
025	703	Dual Modif (trace 1)
026	704	Dual Trace Mod #2 (Dual 784)
027	706	Dual Trace Mod #3
028	707	Dual Trace Mod #4
081	781	Dual Punch
110	782	Dual for ES-1 or ES-2
111	785	Dual 795
112	786	Octal-Dec Table
186	787	RC Series (cards will be found in T-5)
188	788	
189	790	
210	791	
223	794	
224	795	
321	796	
322	797	
520	924	

"Bible" on Dispatcher's Desk

Seldom Used Programs		Math Subroutines
010	JTA 7	400
013	LCH 2	401
016	LCH 10	402
020	LCH 11	403
021	SO2	409
023	NEW SO2	410
024	IBM PROG.	411
080	SHACO	413
086	Shaco Stops	417
185	Octal-Dec Table	426
221	Dual Manual	432
222	Dual Additions	450
320	701 Manual	451
525		
606		
702		
705		
784		
793		
991		
992		

Programs issued after this date (Jan. 28, 1955) will be considered as Current until notified.

Jan. 28, 1955

H. Kolsky
T-5

Memo to 701 Users Regarding 322

As you know, the 704 will be equipped with improved tape units. The successful operation of these new units and the "acceptance" of them by us can lead to vastly faster input and output operation. Such a mode of operation would be very efficient compared to our present usage of cards.

Most of the causes of tape failure during the early days of 701 operation, which led programmers to choose drums over tapes for dumping, have now been corrected. In anticipation of more tape usage during the coming years, we in T-1 are, at this time, making an effort to encourage 701 users to introduce tape usage into their problems whenever it is feasible, especially in dumping. For this reason, the tape dumping program 322 was written.

A two-bank version of this program is expected in the near future. It is requested that any comments you might have with regard to the performance of this program and tape usage, in general, be relayed to me.

Edward A. Voorhees

Type 701 Electronic Data Processing Machines

January 24, 1955

SUBJECT: Multi-File Tape Operation-701

PURPOSE: To offer two methods for multi-file tape operation on the 701.

INFORMATION:

1. This is the faster of the two methods but requires extra programming:

   a. More than one file may be placed upon the tape by utilizing a series of "PREPARE TO WRITE TAPE" instructions following the last "COPY" instruction of the previous file.

   b. The nominal number of "PREPARE TO WRITE" instructions necessary to distinguish the Nth file from the (N + 1) file has been determined to be a minimum of twenty.

   c. It has been expressed by the Engineering Department that the number of "PREPARE TO WRITE TAPE" instructions should not become a specification for the Model 701 EDPM.

   d. A nominal value of 17 milliseconds is required for the execution of each "PREPARE TO WRITE TAPE" instruction, thus approximately 340 milliseconds are required for the preparation for a new file.

   e. A comprehensive test program is available upon request for field testing this feature and provides an immediate answer as to the number of "PREPARE TO WRITE TAPE" instructions required by any particular tape at any time.

   f. Reading multi-file tape is quite straight-forward with recognition of the end of the file as one method of advancing to the next file.

   g. It is necessary that the programmer count the files and constantly maintain his position within the tape.

2. This method is slower, but requires less programming.

   a. Give WEOF just before giving WR for the second file. This prevents R#10 from dropping out and does not allow the heads to be turned off.

NO.	NAME	OLD NAME
010 R	Read decimal instructions into specified locations.	BIDDIR

INPUT: Decimal instructions to be loaded by 010 are punched, one instruction per card, as follows:

Column	Punches
13 thru 16	location of the instruction (must be +)
17	sign of the instruction, 11 for -, 12 for +
18 & 19	operation part of instruction
23 thru 26	address " " "

All of the above information must be punched in decimal. There must be one, and only one, punch per column in columns 13 thru 19 and 23 thru 26. All other columns will be ignored by 010; they may be used in any manner desired for identification, comments, or other information not to be loaded by 010. 010 will load decimal instructions punched by IBM SO_2. 010 will load any portion of E.S. except the 126 half-words occupied by itself. A transition card from 010 may be used if desired. Punch the instruction in decimal as above with location F6.

LOADING: Load 010 binary cards with 021

Loading Deck	# Cards
021	1
010	3
Transition to 010	1
Decimal instructions	n

Loading Deck	# Cards
010 Transition (if desired)	1 or 0
Total	n + 6 or n + 5

STARTING: a. Automatic entry: Put the loading deck in hopper and have card-reader ready. Set load selector to cards, instruction entry keys for 021 $(0, 4000, \text{or } 7000)_8$, automatic-manual switch to automatic, and press load. When 701 stops on the last card, press card-reader start. Feed out cards when select light goes out.

b. Manual entry (when 010 is already in E.S.): Put decimal instruction deck in hopper and have card-reader ready. Start 701 manually at F0. Press card-reader start when cards stop feeding and when select light goes out, feed out cards in reader.

c. Entry by unconditional transfer: Have instruction deck ready in the card-reader. Transfer to F0.

DESCRIPTION: The 701 will read in each decimal instruction, convert it to binary, and store it in the specified half-word location, checking for omitted and double punches. 010 always loads <u>all</u> the cards in the hopper. The transition card from 010 may be at any place in the decimal instruction deck, but it will not be executed until <u>all</u> the cards have been loaded. If there is no transition card in the instruction deck, 010 after loading will execute the instruction stored in F6; if no transition card has been read since loading of 010, F6 will contain a stop.

PROGRAM STOPS:

Regional Location	Meaning
F6	End of file; all cards in the hopper have been read, i.e., all instructions are loaded. To load another deck, have card-reader and press start.
F66	The card being read contains a double punch or lacks a punch in some column 13 thru 19 or 23 thru 26. Take the remaining cards out of the hopper and feed out those in the card-reader. Look at the third card back; correct the card, put these three cards and the remaining deck back in the hopper, have card-reader ready and press start. If there is no punching error, the 701 has made an error in summing. Put the three cards and the remaining deck in hopper and proceed as with stop F6 above. If error keeps repeating, reload or start over or call 701 dispatcher.

OUTPUT: Binary instructions stored in specified half-word locations of E.S.

RESTARTING: Start as before (see STARTING b or c).

STORAGE: Regional F0 thru F119

E0 thru E5

Total 126 half-words, 120 regional cards. For SO_2 assembly E0 and F0 must be specified; E0 must be even.

CODED: AIB, ch'd - dtm, written - dtm

	010	010	010	010
010 Read decimal instructions into specified locations	R	A	B	C
INPUT: Punch transition card with decimal location.	F6	58_{10}	2106_{10}	3642_{10}
LOADING CARD		021A	021B	021C
STARTING: For loading deck, set instruction entry keys to		0	4000_8	7000_8
For manual entry, start at. .	F0	64_8	4064_8	7064_8
For entry by unconditional transfer, transfer to . .	F0	52_{10}	2100_{10}	3636_{10}
PROGRAM STOPS: end of file (if no transition card). . .	F6	72_8	4072_8	7072_8
punch error on third card back. . .	F66	166_8	4166_8	7166_8
STORAGE: decimal. . .	F0-	52-	2100-	3636-
thru	F119	171	2219	3755
	E0-	2-	2048-	3584-
thru	E5	7	2053	3589
octal. . .	F0-	(64-	(4064-	(7064-
thru	F119	$253)_8$	$4253)_8$	$7253)_8$
	E0-	(2-	(4000-	(7000
thru	E5	$7)_8$	$4005)_8$	$7005)_8$

H. Kolsky
T-5

NO.	NAME
011 R	Read decimal instructions into specified locations in ES-1 or ES-2.

INPUT: Decimal instructions to be loaded by 011 are punched one per card as follows:

Column	Contents
9	Sign of operation: x-punch if negative, blank if positive.
10 - 13	Location of the instruction in decimal.
14 - 15	Operation part of the instruction in decimal.
16 - 19	Address part of the instruction in decimal.
20	Blank if location is in ES-1; x-punch if location is in ES-2.

LOADING:

Loading Deck	# Cards
026 (or 028)	1 (or 2)
Coder's binary deck	n
011	2
Decimal instructions	p
Blank Card	1
Coder's binary transition card (if desired)	1
Total	4+n+p (or 5+n+p)

STARTING:

a. Automatic entry: Put loading deck in Card Reader, set instruction entry keys for 026 (or 028), press load. When 701 stops, press card-reader start.

b. Manual entry: (011 already in ES): Put decimal instruction cards followed by a blank card and binary transition card if desired in card-reader. Start 701 manually at $F43_8$. (Have overflow indicator off.)

c. Entry by unconditional transfer: Have decimal instructions followed by a blank card and binary transition card in card-reader. Transfer to $F43_8$. (Have overflow indicator off.)

DESCRIPTION: 011 acts as its own transition card, then self-loads itself over 026 (or 028), then reads the decimal instruction cards following, converts them to binary, and stores the instruction in the location specified.
011 does not check for double-punching or blank columns.
011 turns off the overflow indicator.
If 011 reads a blank card or a decimal instruction card ± 0000 00 0000, it will not store the instruction, and it will consider the next card as a transition card.
011 is designed so that the coder can insert 011 and the decimal instructions to be loaded and a blank card between his binary cards and his transition card. It self-loads itself over the original binary loading card (026 or 028) so that the only extra space required is 10 more half-words than 026, or 2 more half-words than 028.
011 will load instructions into any location in ES-1 or ES-2 except the 60 half-words occupied by itself in ES-1. The coder must use an 011 in the same region as his 026 (or 028) card.

PROGRAM STOPS: F43: Copy check: End of file condition indicating that no transition card was read.

OUTPUT: Binary half-words stored in specified location in ES-1 or ES-2.

STORAGE: F0 to F59, 60 half-words.

CODED: Dura W. Sweeney, 5/20/54.

		Region
011:	Read decimal instructions into ES-1 or ES-2	0000
STARTING:	Manual Entry: Start at	0043_8
	Unconditional TR: TR to	0006_8
STOP:	End of File: Copy check	0053_8
STORAGE:	Decimal	0000-0059
	Octal	0000-0073

011 is available in all octal regions 0000, 1000, 2000, 3000, 4000, 5000, 6000, and 7000. Add the high order digit of the octal region to the above stop to get the proper stop address.

The coder must use 011 in the same region as 026 or 028.

This write-up replaces the previous 011 write-up. October 29, 1954

Ha Kolsky
T-5

NO.	NAME
012	To load blocks of either full or half-word decimal data into ES-1 or ES-2.
INPUT:	Any number of constants with signs, up to 7 full words or 14 half-words per card, may be read by 012, converted to binary, scaled, and stored in specified locations of ES-1 or ES-2. Each block of constants must have the same scaling and must be preceded by a control card punched in decimal as follows:

Control Card

Columns	Punch
9	11 punch
10	0 punch 11 punch for full words
11 - 14	Initial loading address (zeroes must be punched)
14	11 punch for half-words to ES-2
15 - 17	000
18 - 19	The decimal scale factor, u, (position of the decimal point from the left) in the range $00 \leq u \leq 10$. Zeroes must be punched.
20 - 22	000
23 - 24	The binary scale factor, t, (position of the binary point from the left) in the range $00 \leq t \leq 35$. Zeroes must be punched.

Note: "t" must be large enough to accommodate "u".

The 12 row is not read from any type card.

DATA CARD FOR FULL WORDS

Card Columns	Punch
9	BLANK
10 - 19	1st Full word
20 - 29	2nd " "
30 - 39	3rd " "
40 - 44	4th " " (1st five digits)
45	BLANK
46 - 50	4th Full word (last five digits)
51 - 60	5th " "
61 - 70	6th " "
71 - 80	7th " "

DATA CARD FOR HALF-WORDS

Card Columns	Punch
9	BLANK
10 - 14	1st Half-word
15 - 19	2nd "
20 - 24	3rd "
25 - 29	4th "
30 - 34	5th "
35 - 39	6th "
40 - 44	7th "
45	BLANK
46 - 50	8th Half-word
51 - 55	9th "
56 - 60	10th "
61 - 65	11th "
66 - 70	12th "
71 - 75	13th "
76 - 80	14th "

Signs are punched over last digit of each word, an 11 for minus, a 12 for plus is arbitrary.

If less than 7 full words or 14 half-words are to be loaded, the rest of the input card should be left blank. <u>If zeroes are punched in, they will be loaded</u>. All zeroes in numbers to be loaded must be punched. When a blank card is read by the card reader, the next card is treated as a binary transition card unless entry to 012 is made by basic linkage in which control returns to coder's program following the reading of the blank card.

Calling sequence for entry by basic linkage from ES-1 or ES-2 is as follows:

Coder's Program

α	+ R Add	$\alpha + 2$
$\alpha + 1$	+ Tr	F0
$\alpha + 2$	+ Tr	$\alpha + 3$ (if return is to bank 1)
	or	
	- Tr	$\alpha + 3$ (if return is to bank 2)
$\alpha + 3$	Control returns here	

LOADING: Load 012 with 026 or 028

Input Deck	# Cards
026 or 028	1
012	5
012 Transition	1
Control card	1
Block of constants	n
Control card	1
Block of constants	n
	etc.
Blank card	1
Transition card if desired	1 (0)

STARTING: a. Automatic entry: Put loading deck in hopper and have card reader "ready". Set instruction keys to F0 (for 026 or 028), press load button.

b. Manual entry (012 already in E.S.): Put input deck in hopper and have card reader "ready". Transfer to F2 (for 012).

c. Start by linkage occurs automatically.

EXIT: In all cases a blank card must follow the last data card to accomplish an automatic exit. If automatic or manual entry is made to 012, the card following the blank card is treated as an ordinary binary transition card. If entry is made by basic linkage, control returns to the coder's program only after reading the blank card following the last data card. If exit is made 012 turns off the ov ind. If no blank card follows the last data card, end-of-file skip causes a Copy-Check at F18 (for 012). If automatic or manual entry is made and no transition card is read following the blank card, a program stop occurs at E2 (for 012). The entire nine row of the binary transition card is copied into E2 and E4 (for 012).

DESCRIPTION: 012 reads the decimal information, converts it to binary, checks for double punching and blank columns in the nine thru zero rows, scales the numbers according to the u and t given and stores the numbers in consecutive locations starting with the initial loading address punched in the control card. All numbers are considered as completely fractional. u specifies how many places the decimal point is to be shifted to the right. The converted number is also considered as all fractional. t specifies how many places the binary point is to be shifted to the right. If t is not large enough to accommodate u, a divide check occurs at F118 (for 012) for full word loading or F138 (for 012) for half-word loading. If a first control card is not read one of the divide checks just mentioned will occur. If 012 reads a blank or double punched column a program stop occurs at F83 (for 012).

PROGRAM STOPS:

Location	Meaning
E 2	No transition card following the blank card.
F 18	Copy check caused by end-of-file skip.
F 83	Double punches or blank columns in 9 to 0 row of control or data card. Correct, reload, push start.
F 118	Divide check: The binary scale, t, is not large enough to accommodate the decimal scale, u, in loading full words. (Or no first control card read).
F 138	Divide check: The binary scale, t, is not large enough to accommodate the decimal scale, u, in loading half-words. (Or no first control card read).

STORAGE:

E0 thru E35

A0 thru A5

N0 thru N53

F0 thru F157

E0 thru E35 occupies the F0 thru F35 part of 026 or 028. A0 thru A5, N0 thru N53 and F0 thru F157 follow 026 or 028. E0 and N0 must be even.

CODED: Paul E. Harper, July 14, 1954

(This write-up replaces the write-up dated June 30, 1954.)

		012R	012A	012B	012C
012 R	Full or half-word decimal input into ES-1 or ES-2.				
STARTING:	By linkage Tr to	F 0	$(166)_8$	$(4166)_8$	$(7166)_8$
	Other Tr to	F 2	$(170)_8$	$(4170)_8$	$(7170)_8$
STORAGE:	Decimal	E 0-	0	2048	3584
		E 35	35	2083	3619
		A 0-	58	2106	3642
		A 5	63	2111	3647
		N 0-	64	2112	3648
		N 53	117	2165	3701
		F 0-	118	2166	3702
		F 157	275	2323	3859
	Octal	E 0-	0	4000	7000
		E 43	43	4043	7043
		A 0	72	4072	7072
		A 5	77	4077	7077
		N 0	100	4100	7100
		N 65	165	4165	7165
		F 0	166	4166	7166
		F 235	423	4423	7423
STOPS:					
No transition card		E 2	$(0002)_8$	$(4002)_8$	$(7002)_8$
End-of-file		F 18	$(0210)_8$	$(4210)_8$	$(7210)_8$
DPBC detect		F 83	$(0311)_8$	$(4311)_8$	$(7311)_8$
"t" too small for full words		F 118	$(0354)_8$	$(4354)_8$	$(7354)_8$
"t" too small for half-words		F 138	$(0400)_8$	$(4400)_8$	$(7400)_8$

NO.	**NAME**
013 R	Read 5 or 10 digit decimal fractions into specified locations.

INPUT: Decimally punched cards. Each card contains one constant and its location; card layout is as follows:

Columns	Punched
9	11 for 10 digit fraction, no punch for 5 digit fraction.
10 thru 19	10 decimal digit fraction, decimal point taken between columns 9 and 10.
15 thru 19	5 decimal digit fraction, decimal point taken between columns 14 and 15. columns 10 thru 14 must be 0's for 5 digit fraction.
19	sign of the 5 or 10 digit fraction, 11 for minus, 12 for plus. Column 45 must be blank.
20 thru 23	|full word or half-word location|

LOADING: 013 will load any portion of E.S. except the 140 half-words occupied by itself. Load 013 binary cards with 021 . See 021 for complete loading instructions.

Loading deck	# cards
021	1
013	3 or 4
Transition to 013	1
Decimal constants	n
Total	n + 5 or n + 6

STARTING: a. Automatic start: Put the loading deck in the hopper and have the card-reader ready. Set load selector to cards, instruction entry keys for 021, and press load. When 701 stops on the last card, press card-reader start. When select light goes out feed out cards by pressing stop then feed.

b. Manual entry (when 013 is already in E.S.): Put decimal deck in hopper, have card reader ready, and start 701 manually at F0. Feed out cards when select light goes out.

c. Entry by unconditional transfer: Have decimal constants deck in the card-reader and ready. Transfer to F0.

DESCRIPTION: Either a 10 digit fraction is read in, converted to binary, and stored at the full word location indicated, or a 5 digit fraction is read in, converted and stored at the half-word location indicated. Checks are made on numerical punches and sign punches for double punches or omissions.

PROGRAM STOPS:

Location	Meaning
F8	End of file. To load another deck, have card-reader ready and press start.
F49	Punch error in columns 10 thru 23, third card back. Put corrected card in card-reader, have it ready and press start.
F84	Sign error in column 19, third card back. Proceed as with stop F49.

OUTPUT: Binary constants stored in specified half or full word E.S. locations.

RESTARTING: See STARTING, b or c.

STORAGE:	F0 thru F89	No. Cards:	131 regional
	B0 thru B37, B0 odd		3 or 4 binary
	A0 thru A2		
	E0 thru E8, E0 even		
	Total 140 half-words		
CODED:	JDM, ch'd - dlt, written - jdm.		

		013	013	013	013
013	Read 5 or 10 digit decimal fractions into specified locations	R	A	B	C
LOADING CARD:			021A	021B	021C
STARTING:	For loading deck, set instruction entry keys to		0	4000_8	7000_8
	For manual entry, start 701 at . . .	F0	135_8	4135_8	7135_8
	For entry by unconditional transfer, transfer to . . .	F0	93_{10}	2141_{10}	3677_{10}
PROGRAM STOPS:	end of file. . .	F8	145_8	4145_8	7145_8
	Punch error on third card back, columns 10 thru 23 . .	F49	216_8	4216_8	7216_8
	Sign error on third card back, column 19. . .	F84	261_8	4261_8	7261_8
STORAGE:	decimal	E0-	2-	2048-	3584-
	thru	E8	10	2056	3592
		A0-	52-	2100-	3636-
	thru	A2	54	2102	3638
		B0-	55-	2103-	3639-
	thru	B37	92	2140	3676
		F0-	93-	2141-	3677-
	thru	F89	182	2230	3766
	octal	E0-	(2-	(4000-	(7000-
	thru	E8	$12)_8$	$4010)_8$	$7010)_8$
		A0-	(64-	(4064-	(7064-
	thru	A2	$66)_8$	$4066)_8$	$7066)_8$
		B0-	(67-	(4067-	(7067-
	thru	B37	$134)_8$	$4134)_8$	$7134)_8$
		F0-	(135-	(4135-	(7135-
	thru	F89	$266)_8$	$4266)_8$	$7266)_8$

NO.	NAME
014	Twelve-digit, decimal input with decimal and binary scale factors. Input is a half-word or a full-word per card into the ES-1 (or ES-2) location specified.

INPUT: Each card contains the following information in decimal.

Half-word input:

Columns	Content
9 - 44	Must be blank in the 9 to 0 row. If any digital punch is encountered, the card is treated as a binary transition card.
17	A y punch.
45	Sign of the constant: y for plus, x for minus.
46 - 57	The constant to twelve decimals. Zeros must be punched.
58 - 59	The decimal scale factor, u, in the range $00 \leq u \leq 11$. Zeros must be punched.
60 - 61	The binary scale factor, t, in the range $00 \leq t \leq 35$. Zeros must be punched.
65 - 68	The location of the half-word in decimal. Zeros must be punched.
69	x if location is in ES-2, y or blank if location is in ES-1.

Full-word input:

Same as half-word input except column 17 contains an x punch, and the location is even if in ES-1 and odd if in ES-2. Column 69 is ignored.

LOADING: Load 014 with 026 or 028

026 or 028	1 (2)
014	4
014 Transition	1
Input cards	n
Binary transition if desired	1 (0)

This page replaces the previous 014 R - 1. October 29, 1954

STARTING:

a. Automatic entry: Put cards in card reader, set instruction entry keys to F0 (for 026 or 028), press Load.

b. Manual entry, (014 already in ES): Put input deck in card reader. Transfer to F0 (for 014).

c. Entry by transfer: Put input deck in card reader. Transfer to F0.

EXIT:

Automatic exit can only be accomplished by an ordinary binary transition card following the last input card. Note that 014 turns off the overflow indicator. End-of-file skip causes a Copy-Check at F13.

DESCRIPTION:

014 reads the decimal information in columns 46-61 and 65-68, converts it to binary, checks for double punching and blank columns in the nine thru zero rows, scales the number according to the *u* and *t* given, rounds to the binary accuracy specified by column 17 and stores the number in the location specified. *u* and *t* are checked to insure that they are in the proper range, and that *t* is large enough to accommodate the given *u* and that rounding does not cause an overflow.

The twelve-digit number is considered as completely fractional, i.e. the decimal point is between columns 45 and 46. *u* specifies how many places the decimal point is to be shifted to the right. The converted number is also considered as all fractional. *t* specifies how many places the binary point is to be shifted to the right.

PROGRAM STOPS:

Location	Meaning
F13	Copy-Check caused by End-of-File Skip.
F59	Double punch or blank column detection in 9 to 0 row in columns 46-61 or 65-68. Correct, reload, push Start.
F121	The binary scale, t, is not large enough to accommodate the decimal scale, u, or rounding caused an overflow. Correct, reload, push Start.
F149	u or t is out of range. Correct, reload, push Start.

RESTARTING: If a valid program stop above; F59, F121 or F149; push Start, or transfer manually to F0.

STORAGE: E0-E16, B0-B6, F0-F149. E0-E16 occupies the F0-F16 part of 026 or 028. B0-B6 and F0-F149 follow 026 or 028. E0 and B0 must be at an even address.

Total 174 half-words.

CODED: Dura W. Sweeney, June 1, 1954.

This page replaces the previous 014 R - 3. October 29, 1954

014 R	Twelve digit decimal input as half-word or full-word into ES-1 or ES-2.	014 R	014 A	014 B	014 C
	Starting: TR to	F0	$(0101)_8$	$(4101)_8$	$(7101)_8$
	Storage: Decimal	E0-	0	2048	3584
		E16	16	2064	3600
		B0-	58	2106	3642
		B6	64	2112	3648
		F0-	65	2113	3649
		F149	214	2262	3798
	Octal	E0-	0	4000	7000
		E20	20	4020	7020
		B0-	72	4072	7072
		B6	100	4100	7100
		F0-	101	4101	7101
		F225	326	4326	7326
	Stops:				
	End-of-File	F13	$(0116)_8$	$(4116)_8$	$(7116)_8$
	DPBC detect	F59	$(0174)_8$	$(4174)_8$	$(7174)_8$
	Improper Scaling	F121	$(0272)_8$	$(4272)_8$	$(7272)_8$
	Scale factors out of range	F149	$(0326)_8$	$(4326)_8$	$(7326)_8$

H. Kolsky
T-5

NO.	NAME
015	Double precision, decimal input with decimal and binary scale factors. Input is two full words per card into ES-1 (or ES-2). The location of the first full word is specified on the card.

INPUT: Each card contains the following information in decimal.

Columns	Content
9 - 44	Must be blank in the 9 to 0 row. If any digital punch is encountered, the card is treated as a binary transition card.
45	Sign of the constant: y for plus, x for minus.
46 - 67	The constant to twenty-two decimals. Zeros must be punched.
69 - 70	The decimal scale factor, u, in the range $00 \leq u \leq 22$. Zeros must be punched.
71 - 72	The binary scale factor, t, in the range $00 \leq t \leq 70$. Zeros must be punched.
73 - 76	The location in decimal. Zeros must be punched. The location is even for ES-1 and odd for ES-2.

LOADING: Load 015 with 026 or 028

026 or 028	1 (2)
015	5
015 Transition	1
Input cards	n
Binary transition if desired	1 (0)

STARTING:

a. Automatic entry: Put cards in card reader, set instruction entry keys to F0 (for 026 or 028), press Load.

b. Manual entry, (015 already in ES): Put input deck in card reader. Transfer to F0 (for 015).

c. Entry by transfer: Put input deck in card reader. Transfer to F0.

EXIT: Automatic exit can only be accomplished by an ordinary binary transition card following the last input card. Note that 015 turns off the overflow indicator. End-of-file skip causes a Copy-Check at F15.

DESCRIPTION: 015 reads the decimal information in columns 46-67 and 69-76, converts it to binary, checks for double punching and blank columns in the nine thru zero rows, scales the number according to the u and t given, rounds and stores the number in the locations specified. u and t are checked to insure that they are in the proper range, and that t is large enough to accommodate the given u and that rounding does not cause an overflow.

The double precision number is considered as completely fractional, i.e. the decimal point is between columns 45 and 46. u specifies how many places the decimal point is to be shifted to the right. The converted number is also considered as all fractional. t specifies how many places the binary point is to be shifted to the right.

PROGRAM STOPS:

Location	Meaning
F15	Copy-Check caused by End-of-File Skip.
F62	Double punch or blank column detection in 9 to 0 row in columns 46-61 or 65-68. Correct, reload, push Start.
F179	The binary scale, t, is not large enough to accommodate the decimal scale, u, or rounding caused an overflow. Correct, reload, push Start.
F190	u or t is out of range. Correct, reload, push Start.

RESTARTING: If a valid program stop above; F62, F179 or F190; push Start, or transfer manually to F0.

STORAGE: E0-E20, B0-B12, F0-F190. E0-E20 occupies the F0-F20 part of 026 or 028. B0-B12 and F0-F190 follow 026 or 028. E0 and B0 must be at an even address.

Total 225 half-words.

CODED: Dura W. Sweeney, June 23, 1954.

		015 R	015 A	015 B	015 C
015 R	Double precision decimal input as two full-words into ES-1 or ES-2.				
	Starting: TR to	F0	$(0101)_8$	$(4101)_8$	$(7101)_8$
	Storage: Decimal	E0-	0	2048	3584
		E20	20	2068	3604
		B0-	58	2106	3642
		B12	70	2118	3654
		F0-	71	2119	3655
		F190	261	2309	3845
	Octal	E0-	0	4000	7000
		E24	24	4024	7024
		B0-	72	4072	7072
		B12	106	4106	7106
		F0-	107	4107	7107
		F276	405	4405	7405
	Stops:				
	End-of-File	F15	$(0126)_8$	$(4126)_8$	$(7126)_8$
	DPBC detect	F62	$(0205)_8$	$(4205)_8$	$(7205)_8$
	Improper Scaling	F179	$(0372)_8$	$(4372)_8$	$(7372)_8$
	Scale factors out of range	F190	$(0405)_8$	$(4405)_8$	$(7405)_8$

NO.	NAME
016R	Read decimal absolute instructions, up to 12/card, into blocks of E.S.

DESCRIPTION: Blocks of decimal absolute instructions are converted to binary and stored in blocks of electrostatic storage by 016. The initial storage location of each block is specified by a heading card. 016 checks to see that no columns of the control card or the data cards are blank or have double punches. 016 also checks to make sure the first card it reads is a control card.

INPUT: The control card is punched as follows:

Columns	9	11 punch
	9-10	0
	11-14	Initial loading address of block of E.S. may be even or odd

The instruction cards are punched as follows:

Columns	9-14	1st instruction
	15-20	2nd instruction
	21-26	3rd instruction
	27-32	4th instruction
	33-38	5th instruction
	39-44	6th instruction
	45-50	7th instruction
	51-56	8th instruction
	57-62	9th instruction
	63-68	10th instruction
	69-74	11th instruction
	75-80	12th instruction

In addition,

If 1st	instruction	is	negative,	there	must	be	an	11	punch	in	col	14
2nd	"	"	"	"	"	"	"	"	"	"	"	20
3rd	"	"	"	"	"	"	"	"	"	"	"	26
4th	"	"	"	"	"	"	"	"	"	"	"	32
5th	"	"	"	"	"	"	"	"	"	"	"	38
6th	"	"	"	"	"	"	"	"	"	"	"	44
7th	"	"	"	"	"	"	"	"	"	"	"	50
8th	"	"	"	"	"	"	"	"	"	"	"	56
9th	"	"	"	"	"	"	"	"	"	"	"	62
10th	"	"	"	"	"	"	"	"	"	"	"	68
11th	"	"	"	"	"	"	"	"	"	"	"	74
12th	"	"	"	"	"	"	"	"	"	"	"	80

If $n < 12$ words are to be loaded, the last 6(12-n) columns should' be blank.

LOADING: 016 is loaded with 021. See 021 for complete loading instructions.

Loading Deck	# Cards
021	1
016	5
Transition to 016	1
Control Card	1
Instruction Cards	n
Control Card	1
Instruction Cards	n

etc.

STORAGE:	E0 thru E28, E0 even	
	A0 thru A3, A0 even	
	F0 thru F201	
	205 regional cards, 5 binary cards.	
STOPS:	F 14	End of file, all instructions loaded. Push start to read more cards. There will be no check for a leading control card.
	F110	Control card has a blank column or is double punched. Correct card, place it in the reader, have the card reader ready. Push Start to continue.
	F140	Instruction card has a blank column or a double punch. Correct card, place it in the reader, have the card reader ready. Push Start to continue.
CODED:	Scully 6/53	

016R	Read blocks of decimal absolute instructions into E.S.	016R	016A 1	016B	016C
START: Transition card punched (octal)		F0	$(67)_8$	$(4067)_8$	$(7067)_8$
STORAGE:	decimal	E0-	0-	2048-	3584-
		E28	28	2076	3612
		A0-	52-	2100-	3636-
		A2	54	2102	3638
		F0-	55-	2103-	3639-
		F201	256	2304	3840
	octal	E0-	(0-	(4000-	(7000-
		E28	$34)_8$	$4034)_8$	$7034)_8$
		A0-	(64-	(4064-	(7064-
		A2	$66)_8$	$4066)_8$	$7066)_8$
		F0-	(67-	(4067-	(7067-
		F201	$400)_8$	$4400)_8$	$7400)_8$
STOPS: All instructions stored	decimal	F14	69	2117	3653
	octal		$(0105)_8$	$(4105)_8$	$(7105)_8$
Control card BCDP	decimal	F110	165	2213	3749
	octal		$(0245)_8$	$(4245)_8$	$(7245)_8$
Instruction card BCDP	decimal	F140	195	2243	3779
	octal		$(0303)_8$	$(4303)_8$	$(7303)_8$

NO.	NAME
017	To load full or half word decimal data into ES-1 or ES-2 with specified address increment to obtain equally spaced words in storage.
INPUT:	Any number of constants with signs, up to 7 full words or 14 half words per card, may be read by 017, converted to binary, scaled, and stored. The first word is stored in the initial loading address with each succeeding word of that block being stored in the location obtained by adding the increment to the address of the last word stored. All words within a block must have the same scaling and must be preceded by a control card punched in decimal as follows:

Control Card

Columns	Punch
9	11 punch
10	Blank for half words 11 punch for full words
11 - 14	Initial loading address. (odd or even for half words; even if ES-1 FW, odd if ES-2 FW store)
14	11 punch for half words to ES-2
15 - 17	000
18 - 19	The decimal scale factor, u, (position of the decimal point from the left) in the range $00 \leq u \leq 10$.
20 - 22	000
23 - 24	The binary scale factor, t, (position of the binary point from the left) in the range $00 \leq t \leq 35$.
25	0
26 - 29	Address increment.

Note: "t" must be large enough to accommodate "u".

The 12 row is not read from any type card.

DATA CARD FOR FULL WORDS

Card Columns	Punch
9	BLANK
10 - 19	1st Full word
20 - 29	2nd " "
30 - 39	3rd " "
40 - 44	4th " " (1st five digits)
45	BLANK
46 - 50	4th Full word (last five digits)
51 - 60	5th " "
61 - 70	6th " "
71 - 80	7th " "

DATA CARD FOR HALF-WORDS

Card Columns	Punch
9	BLANK
10 - 14	1st Half-word
15 - 19	2nd "
20 - 24	3rd "
25 - 29	4th "
30 - 34	5th "
35 - 39	6th "
40 - 44	7th "
45	BLANK
46 - 50	8th Half-word
51 - 55	9th "
56 - 60	10th "
61 - 65	11th "
66 - 70	12th "
71 - 75	13th "
76 - 80	14th "

Signs are punched over last digit of each word, an 11 for minus, a 12 for plus is arbitrary.

If less than 7 full words or 14 half-words are to be loaded, the rest of the input card should be left blank. If zeroes are punched in, they will be loaded. All zeroes in numbers to be loaded must be punched. When a blank card is ready by the card reader, the next card is treated as a binary transition card unless entry to 017 is made by basic linkage in which control returns to coder's program following the reading of the blank card.

Calling sequence for entry by basic linkage from ES-1 or ES-2 is as follows:

Coder's Program

α	+ R Add	$\alpha + 2$
$\alpha + 1$	+ Tr	FO
$\alpha + 2$	+ Tr	$\alpha + 3$ (if return is to bank 1)
	or	
	- Tr	$\alpha + 3$ (if return is to bank 2)
$\alpha + 3$	Control returns here	

LOADING: Load 017 with 026 or 028

Input Deck	# Cards
026 or 028	1
017	5
017 Transition	1
Control Card	1
Block of constants	n
Control card	1
Block of constants	n
	etc.
Blank card	1
Transition card if desired	1(0)

STARTING: a. Automatic entry: Put loading deck in hopper and have card reader "ready". Set instruction keys to FO (for 026 or 028), press load button.

b. Manual entry (017 already in E.S.): Put input deck in hopper and have card reader "ready". Transfer to F2 (for 017).

c. Start by linkage occurs automatically.

EXIT: In all cases a blank card must follow the last data card to accomplish an automatic exit. If automatic or manual entry is made to 017, the card following the blank card is treated as an ordinary binary transition card. If entry is made by basic linkage, control returns to the coder's program only after reading the blank card following the last data card. If exit is made 017 turns off the ov ind. If no blank card follows the last data card, end-of-file skip causes a Copy-Check at F18 (for 017). If automatic or manual entry is made and no transition card is read following the blank card, a program stop occurs at E2 (for 017). The entire nine row of the binary transition card is copied into E2 and E4 (for 017).

DESCRIPTION: 017 reads the decimal information, converts it to binary, checks for double punching and blank columns in the nine thru zero rows, scales the numbers according to the u and t given and stores the numbers in equally spaced locations starting with the initial loading address punched in the control card. The first number is stored in the initial loading address with each succeeding word being stored in the location obtained by adding the address increment to the location of the last word stored. This process goes on until five consecutive blank columns are read at which time

017 reads the next card and tests to see if that card is a control card, and if it is the process is repeated. If the card is a blank card, exit is made from 017 under the conditions explained in the paragraph above. All numbers are considered as completely fractional. <u>u</u> specifies how many places the decimal point is to be shifted to the right. The converted number is also considered as all fractional. <u>t</u> specifies how many places the binary point is to be shifted to the right. If <u>t</u> is not large enough to accommodate <u>u</u>, a divide check occurs at F120 (for 017) for full word loading or F140 (for 017) for half-word loading. If a <u>first</u> control card is <u>not</u> read one of the divide checks just mentioned will occur. If 012 reads a blank or **double punched** column a program stop occurs at F85 (for 017).

PROGRAM STOPS:

Location	Meaning
E 2	No transition card following the blank card.
F 18	Copy check caused by end-of-file skip.
F 85	Double punches or blank columns in 9 to 0 row of data card. Correct, reload, push start.
F 120	Divide check: The binary scale, t, is not large enough to accommodate the decimal scale, u, in loading full words. (Or no <u>first</u> control card read).
F 140	Divide check: The binary scale, t, is not large enough to accommodate the decimal scale, u, in loading half-words. (Or no <u>first</u> control card read).

STORAGE: E0 thru E36

A0 thru A5

No thru N53

F0 thru F159

E0 thru E36 occupies the F0 thru F35 part of 026 or 028.

A0 thru A5, N0 thru N53 and F0 thru F159 follow 026 or 028. E0 and N0 must be even.

CODED: Paul E. Harper, July 20, 1954

017 R Full or half-word decimal input into ES-1 or ES-2.	017R	017A	017B	017C
STARTING: By linkage Tr to	F 0	$(166)_8$	$(4166)_8$	$(7166)_8$
Other Tr to	F 2	$(170)_8$	$(4170)_8$	$(7170)_8$
STORAGE: Decimal	E 0-	0	2048	3584
	E 36	36	2084	3620
	A 0-	58	2106	3642
	A 5	63	2111	3647
	N 0-	64	2112	3648
	N 53	117	2165	3701
	F 0-	118	2166	3702
	F 159	277	2325	3861
Octal	E 0-	0	4000	7000
	E 44	44	4044	7044
	A 0	72	4072	7072
	A 5	77	4077	7077
	N 0	100	4100	7100
	N 65	165	4165	7165
	F 0	166	4166	7166
	F 237	425	4425	7425
STOPS:				
No transition card	E 2	$(0002)_8$	$(4002)_8$	$(7002)_8$
End-of-file	F 18	$(0210)_8$	$(4210)_8$	$(7210)_8$
DPBC detect	F 85	$(0313)_8$	$(4313)_8$	$(7313)_8$
"t" too small for full words	F 120	$(0356)_8$	$(4356)_8$	$(7356)_8$
"t" too small for half-words	F 140	$(0402)_8$	$(4402)_8$	$(7402)_8$

NO.	NAME	OLD NAME
020R	Read binary half-words into consecutive locations	CRBO2S

INPUT: Any number of binary cards may be loaded by 020. Each card must have in the 9 row,

Columns 9 thru 44	S, the card check sum for that card
" 51 thru 62	V, the half-word count for that card. V must be even.
" 69 thru 80	R, the initial loading address for that card, the E.S. location of the first half-word to be loaded from that card. R must be even.

Rows 8 thru 12 contain the half-words to be loaded in binary, preceded by their signs, 4 to a row, in

Columns 9 thru 26,
27 thru 44,
45 thru 62, and
63 thru 80.

As many rows per card may be used as desired; the last row used may contain 2 or 4 half-words.

LOADING: 020 is self loading.

Loading Deck	# Cards
020	1
Binary deck to be loaded	n
Total	n + 1

STARTING: a. Automatic entry: Press reset. Put loading deck in hopper and have card-reader ready. Set instruction entry keys to FO, automatic-manual switch to automatic, load selector to cards, and press load button. Feed out cards when select light on card reader goes out.

b. Manual entry (when 020 is already in E.S.): Press reset. Put binary deck to be loaded in hopper and have card-reader ready. Start 701 manually at F8. Feed out cards when select light on card-reader goes out.

DESCRIPTION: The binary half-words of each card of the deck to be loaded are read and stored in E.S. locations R thru R + V - 1 for that card. Check is made to see that the sum of the information in E.S. agrees with the S read from the card. 020 will load binary half-words into any position of E.S. except the 48 half-words occupied by itself, F0 thru F47. 020 will read cards punched by 220.

There must not be any blank cards in the loading deck, and R and V must be even. If there is a blank card or a card with odd R or V in the deck, 020 will stop on a copy check. In this case 020 must be reloaded before the rest of the deck can be read.

If 020 copies binary information to the end of E.S. and there are still half-words to be loaded on the current card, these remaining half-words will be read into E.S. 0, 1, 2, ... The following is an example of this case:

Let R = 4094

V = 6

then R + V = 4100 = 4 mod 4096

The results of loading with 020 are:

E.S. Location	Contents (from card)
R = 4094	1st half-word of 8 row
4095	2nd " " " " "
0	3rd " " " " "
1	4th " " " " "
2	1st " " " 7 row
R + V - 1 = 3	2nd " " " " "

PROGRAM STOPS:

Regional Location	Meaning
F47	End of file; all half-words have been loaded.
F44	Check sums do not agree; take the remaining cards out of the hopper and feed out those in card-reader. Error is on the third card back. Correct and put these three cards and the remaining deck in the hopper and restart. If stop keeps repeating reload or call 701 Engineer.

RESTARTING: see STARTING b.

STORAGE: 020 occupies F0 thru F47; total, 48 half-words. For SO_2 assembly, see special instructions "SO_2 Assembly of Self-loading Programs". Origins F0 and H0 must be specified and must be even.

CODED: WA

020 Read binary half-words into consecutive locations.

		020R	020M1
STARTING:	For automatic entry, set instruction entry keys to ...	F0	764_8
	For manual entry, start 701 at ...	F8	774_8
PROGRAM STOPS:			
	end of file ...	F47	1043_8
	Check sums do not agree ...	F44	1040_8
STORAGE:	decimal	F0-	500-
	thru	F47	547
	octal	F0-	(764-
	thru	F47	$1043)_8$

NO.	**NAME**
021R	Load itself, check loading of itself, and read binary half-words into consecutive E.S. locations.

INPUT: Each card must contain in the 9 row in binary

columns 9 thru 44	S, the card check sum for that card.
51 thru 62	V, the number of half words on that card.
69 thru 80	R, the location of the first half-word to be loaded from that card.

Rows 8 thru 12 contain the half-words to be loaded in binary, preceded by their signs, four to a row, in

Columns 9 thru 26,
27 thru 44,
45 thru 62, and
63 thru 80.

LOADING: 021 is self loading.

Loading Deck	# Cards
021	1
binary deck to be read	n
transition from 021, if desired	1 (or 0)
Total	n + 2 (or n + 1)

STARTING: a. Automatic entry: Press reset. Put loading deck in hopper and have card reader ready. Set instruction entry keys to F0, load selector to cards, automatic-manual switch to automatic, and press the load button. Press card reader start when 701 stops on the last card.

b. Manual entry (when 021 is already in E.S.): Press reset. Put binary deck to be read in hopper and have card-reader ready. Start 701 manually at F40. Press card reader start when 701 stops on last card.

c. Entry by unconditional transfer: Have binary deck to be read ready in card reader. Transfer to F40. Press card reader start for last card.

DESCRIPTION: The binary half-words of each card of the deck to be loaded are read and stored in E.S. locations R thru R + V - 1 for that card. Check is made to see that the sum of the information in E.S. agrees with the S read from that card. V and R may be even or odd. 021 R will load any part of E.S. except the 52 half-words occupied by itself.* A transition card from 021 may be placed anywhere in the deck. As soon as the transition card is read, control is lost from 021 to the location specified. The transition card is punched in columns 10 thru 26 in the 9 row. It must be plus and may be a tr (01) or tr ov (02) to turn off the overflow indicator.

PROGRAM STOPS:

Regional Location	Meaning
F38	S on card does not agree with computed check sum. Pressing start will force 021 to load the card anyway. To correct card, take card out of hopper, feed out cards and correct the third card back. Put these three and the remaining cards in the hopper and restart.

F 44 (if there was no transition card)	End of file; all binary cards are loaded.

OUTPUT: Binary half-words stored in the specified E.S. locations.

RESTARTING: Start as before, see STARTING b or c.

STORAGE: Regional F0 thru F51, total 52 half-words. For SO_2 assembly, see special instructions "SO_2 Assembly of Self Loading Programs". Origins F0 and H0 must be specified and must be even. 021 is one binary card, 48 regional cards. The check sum for the 021 binary card is computed from the formula $S = (37,7760; -25, 2005)_8 - \left[(F0)_8 \times (47)_8\right]$ and is punched on the 47th and 48th regional cards in the form of decimal orders.

CODED: Scully 5-53

* When V + R is greater than 4095, 021 will load the first word on the card into R and then stop at F38.

021 Self-loading load binary cards	021R	021A	021B	021C	021M1
STARTING: Set instruction entry					
keys to ...	F0	0	4000_8	7000_8	3000_8
For manual entry,					
start 701 at ...	F40	50_8	4050_8	7050_8	3050_8
For unconditional					
transfer entry, transfer to ...	F40	40_{10}	2088_{10}	3624_{10}	1576_{10}
PROGRAM STOPS: Check sum					
computed disagrees with sum					
read ...	F38	46_8	4046_8	7046_8	3046_8
end of file ...	F44	54_8	4054_8	7054_8	3054_8
STORAGE: decimal ...	F0-	0-	2048-	3584-	1536-
thru	F51	51	2099	3635	1587
octal ...	F0-	(0-	(4000-	(7000-	(3000-
thru	F51	$63)_8$	$4063)_8$	$7063)_8$	$3063)_8$
*If V + R > 4095, after loading					
into R, 701 stops at ...	F38	46_8	4046_8	7046_8	3046_8

NO.	NAME
023 R	SL load binary half-words into consecutive locations and load over self.

INPUT: Binary cards directly after the 023 cards. For normal loading any number of binary blocks with as many cards per block as desired may be loaded. (A block may of course be one card.) Each block has in the 9 row of its first card the card check sum S, half-word count V, and the initial location R, for that entire block, punched as in standard binary card layout. The remaining rows of the first card and the rest of the cards in the block contain the half-words to be loaded in binary, four to a row, for as many cards and rows per card as needed. The last row of a block may have one, two, three, or four half words. A transition card, + 01;xxxx in the 9 row, columns 10 thru 26, may be placed between any two blocks of binary cards.

For loading over self with 023*, three cards are required; each must contain the standard S, V, and R in the 9 row, with the following restrictions:

---

*023 when loading over self must load over all itself. However, zeros will be automatically stored in the remaining 023 storage if V < the 44 or 20 required for loading the coders half-words over all of 023. If card(s) two or (and) three are completely blank, zeros will be loaded in the range of the blank card(s).

1st card	V = 44	R = F0
2nd card	V = 44	R = F44
3rd card	V = 20	R = F88

Rows 8 thru 12 of the first two contain the half-words to be loaded into the range specified by their fixed V's and R's. Rows 8 thru 4 of the third contain the half-words to be loaded into F88 thru F107. After loading over self, transition occurs automatically to F107; therefore the coder should store in F107 (the last half-word loaded when 023 loads over itself) a transfer ($\pm$ 01; xxxx) to his program.

LOADING: 023 is self loading.

Loading Deck	# Cards
023	3
Binary blocks to be loaded	n
Transition from 023 (if desired)	1 or 0
Load over self cards (if desired)	3 or 0

STARTING: a. Automatic entry: Put loading deck in hopper and have card reader ready. Set the load selector to cards, automatic manual switch to automatic, instruction entry keys to F0 and press the load button. Press card reader start when 701 stops on last card.

b. Manual entry (when 023 is already in E.S.): Have binary deck to be loaded ready in the card reader and start 701 manually at F12.

c. Entry by unconditional transfer: Have binary deck to be loaded ready in the card reader and transfer to F12.

DESCRIPTION: 023 loads itself, then the binary half words of each block are read and stored in E.S. locations R thru R + V - 1 for that block. Check is made to see that the sum of the information in E.S. agrees with the S read from the card. V and R may be even or odd. Whenever a transition card (see INPUT) is encountered between blocks, control goes immediately to the location specified. If a transition card is erroneously placed within a block, the 701 will stop. No check sums are kept in E.S., L for a block equals R + V - 1. 023 will load over itself if the last three cards of the binary deck being loaded are punched for this purpose as specified under INPUT. When finished loading over self, 023 goes to F107 for a transition instruction, which is the last half word loaded from the last card. 023 will load cards punched by 220, 221, and IBM SO2.

PROGRAM STOPS: (occur on normal loading only)

Regional Location	Contents	Meaning
F15	00; F12	End of file normal; all cards are loaded.
F32	00; F33	One or more cards are missing from the last block. Feed out cards in reader; have missing cards ready in card reader and press console start.

F72	00; F63	S on card does not agree with computed check sum. Transition card is within a block or S is incorrect. If S is in error, pressing console start will force the block to load anyway . Or feed out cards, correct error or remove transition and restart with that block.

A program stop 00; 0000 at F107 will occur if blank card is substituted for the third card used by o23 to load over itself*.

RESTARTING: see STARTING b or c.

OUTPUT: Blocks of binary half-words stored consecutively in E.S.

STORAGE: 023 occupies F0 thru F107 while loading itself and during normal loading of binary blocks; 023 will also load this range with coder's binary half-words. 023 is three binary cards, 116 regional cards. F0 must be even.

Procedure for S02 assembly is as follows:

(1) Divide the 023 regional cards into two decks,

A = H0 thru H95

B = H96 thru H115.

(2) Assemble deck A as for any normal self loading program, see "S02 Assembly of Self Loading Programs". This will produce the first two binary self loading cards of the final 023 absolute deck.

(3) Assemble deck B with same F0 as used in (2). Give for H0 origin the value (F96 - 8). This will produce one binary card (with S, V, and R in the 9 row).

(4) The final 023 absolute deck consists of the two cards from (2) and the one card from (3). The listing from (2) will have dummy locations only; the listing from (3) will have true locations.

CODED: JDM, ck'd IB, DTM, written DTM

023 SL load binary half-words into consecutive E.S. locations and load over self.

	023R	023A	023B	023C
INPUT: For loading over self with 023, use these cards				
1st card R equals	F0	0	4000_8	7000_8
2nd card R equals	F44	54_8	4054_8	7054_8
3rd card R equals	F88	130_8	4130_8	7130_8
After loading over self, 023 goes for a transfer to ...	F107	153_8	4153_8	7153_8
STARTING: For automatic entry, set instruction entry keys to ...	F0	0	4000_8	7000_8
For manual entry, transfer to ...	F12	14_8	4014_8	7014_8
For unconditional transfer entry, transfer to ...	F12	12_{10}	2060_{10}	3596_{10}
PROGRAM STOPS:				
end of file normal ...	F15	17_8	4017_8	7017_8
card(s) missing from last block	F32	40_8	4040_8	7040_8
Check sum error or transition card misplaced ...	F72	110_8	4110_8	7110_8
023 has finished loading over self, loaded part of self with 0's ...	F107	153_8	4153_8	7153_8
STORAGE: Normal loading decimal	F0-	0-	2048-	3584-
thru	F107	107	2155	3691
Normal loading octal	F0-	(0-	(4000-	(7000-
thru	F107	$153)_8$	$4153)_8$	$7153)_8$

NO.	NAME
024R	Load itself and read binary half-words into consecutive E.S. locations.

INPUT: Each card must contain in the 9 row in binary

columns 9 thru 44	S, the card check sum for that card
51 thru 62	V, the number of half words on that card
69 thru 80	R, the location of the first half word to be loaded from that card

Rows 8 thru 12 contain the half words to be loaded in binary, preceded by their signs, four to a row,

in columns 9 thru 26,

27 thru 44,

45 thru 62, and

63 thru 80.

LOADING: 024 is self loading.

Loading deck	# Cards
024	1
Binary deck to be read	n
Transition from 024, if desired	1 (or 0)
Total	n + 2 (or n + 1)

STARTING: a. Automatic entry: Press reset. Put loading deck in hopper and have card reader ready. Set instruction entry keys to F0, load selector to cards, automatic-manual switch to automatic, and press the load button. Press card reader start when 701 stops on the last card.

b. Manual entry (when 024 is already in E.S.): Press reset. Put binary deck to be read in hopper and have card-reader ready. Start 701 manually at F6. Press card reader start when 701 stops on last card.

c. Entry by unconditional transfer: Have binary deck to be read ready in card reader. Transfer to F6. Press card reader start for last card.

DESCRIPTION: The binary half words of each card of the deck to be loaded are read and stored in E.S. locations R thru R + V - 1 for that card. Check is made to see that the sum of the information in E.S. agrees with the S read from that card. V and R may be even or odd. 024R will load any part of E.S. except the 50 half words occupied by itself. A transition card from 024 may be placed anywhere in the deck. As soon as the transition card is read, control is lost from 024 to the location specified. The transition card is punched in columns 10 thru 26 in the 9 row. It must be plus and may be a tr (01) or tr ov (02) to turn off the overflow indicator.

PROGRAM STOPS:

Regional Location	Meaning
F43	S on card does not agree with computed check sum. Pressing start will force 024 to load the card anyway. To correct card, take card out of hopper, feed out cards and correct the third card back. Put these three and the remaining cards in the hopper and restart.

F48 (if there was no transition card)	End of file; all binary cards are loaded.

OUTPUT: Binary half words stored in the specified E.S. locations.

RESTARTING: Start as before, see STARTING b or c.

STORAGE: Regional F0 thru F49, total 50 half words. For SO_2 assembly, see special instructions "SO_2 Assembly of Self Loading Programs". Origins F0 and H0 must be specified and must be even. 024 is one binary card, 46 regional cards.

CODED: Scully 5-53

024 Self-loading load binary cards	024R	024A	024B	024C	024M1
STARTING: Set instruction					
entry keys to ...	F0	0	4000_8	7000_8	3000_8
For manual entry,					
start 701 at ...	F6	6	4006_8	7006_8	3006_8
To enter by unconditional transfer, transfer					
to ...	F6	6	2054_{10}	3590_{10}	1542_{10}
PROGRAM STOPS: Computed check					
sum disagrees with check					
sum read ...	F43	53_8	4053_8	7053_8	3053_8
end of file ...	F48	60_8	4060_8	7060_8	3060_8
STORAGE: decimal ...	F0-	0-	2048-	3584-	1536-
thru	F49	49	2097	3633	1585
octal ...	F0-	(0-	(4000-	(7000-	(3000-
thru	F49	$61)_8$	$4061)_8$	$7061)_8$	$3061)_8$

025 R — Reads regional binary cards into specified locations, 43 half-words per card.

INPUT: Each card must contain in the 9 row in binary:

Cols. 9 thru 35	S, the card check sum for that card.
Cols. 36 thru 44 46 thru 80	V, the number of half-words on that card, variant and invariant information.
Row 8	
Cols. 9 thru 26	R, the location of the first half-word on the card.
Cols. 27 thru 80	The first three half words.
Row 7 - 12	The remainder of the half-words.

INCREMENT CARD:

An increment card containing the following information punched in the 9 row in binary must follow the 025 deck.

Cols. 9 - 26	Blank.
Col. 27	Sign of the increment.
Cols. 28 - 44	Increment.

025 adds the increment on the control card to the FWA and all variant addresses. <u>If the increment is to be zero, it must be minus zero.</u>

LOADING: 025 is self-loading.

Loading Deck	Cards
025	2
025 increment card	1
regional binary cards	n
025 increment card	1

LOADING: (contd)

Loading Deck	Cards
regional binary cards	p

---------------------	

transition from 025, if desired.

Several programs may be entered simultaneously. If they all have the same increment. only one increment card is needed. But a different increment card may precede each program.

STARTING: a. Automatic entry: Set load selector to cards. Put loading deck in hopper and have card-reader ready. Set instruction entry keys for 025, automatic-manual switch to automatic and press load button. When select light goes out, feed cards out of card reader.

b. Manual entry (when 025 is already in E.S.) Press reset. Put regional binary deck preceded by increment card in hopper and have card-reader ready. Start 701 manually at F15.

DESCRIPTION: The 43 binary half-words of each card are read, the increment is added to all the variant addresses and to the FWA, and they are stored in E.S. locations R + increment through R + V + increment - 1. Check is made to see that the sum of the information read off the card agrees with the check sum read from that card. Also, each half-word is read out of its stored location and compared with what was stored there.

PROGRAM STOPS:	Regional Location	Meaning
	F133	S on the card does not agree with the computed check sum. If start is pressed, 025 will load the next card.
	F105	Half-word read from E.S. location does not compare with number stored there. Restart.
	F2	End of file.

OUTPUT: Binary half-words stored in specified E.S. locations.

RESTARTING: Start as before, see starting.

STORAGE: Regional F0 - $F133_8$

$F134 - 147_8$ - erasable storage

Total - 150_8 half-words

CODED: D.W.S., checked and written M.C.F.

025 Self-loading load regional binary cards	025 R	025 A	025 B	025 C
STARTING: Set instruction entry keys to	$F0_8$	0	4000_8	7000_8
For manual entry, start 701 at	$F15_8$	15_8	4015_8	7015_8
PROGRAM STOPS: Check sum computed disagrees with sum read	$F133_8$	133_8	4133_8	7133_8
End of file	$F2_8$	2_8	4002_8	7002_8
Half-word read from E.S. location does not compare with number stored there	F105	105_8	4105_8	7105_8
STORAGE: Octal	$F0_8$	0	4000_8	7000_8
thru	$F133_8$	133_8	4133_8	7133_8
Erasable storage	$F134_8$	134_8	4134_8	7134_8
thru	$F147_8$	147_8	4147_8	7147_8

H. Kolsky
T-5

NO.	NAME
026R	Load itself, read binary half-words into consecutive E.S. locations, read binary half-words back from E.S. locations and form check sum. Load to end of memory.

INPUT: Each card must contain in the 9 row in binary

Columns	Content
9 - 44	S, the card check sum for that card.
51 - 62	V, the number of half-words for that card.
69 - 80	R, the location of the first half-word to be loaded from that card.

Rows 8 thru 12 contain up to 44 half-words, four half-words per row in columns 9-26, 27-44, 45-62, and 63-80.

LOADING: 026 is self-loading.

Loading Deck	# Cards
026	1
Binary cards	n
Transition from 026 (if desired)	1 (of 0)
Total	n + 2 (or n + 1)

STARTING: Automatic Entry: Put loading deck in hopper, and have card reader "Ready". Set instruction key to F0, press load button. Press "Start" on card reader when card reader stops on last card.

Manual Entry: (When 026 is already in E.S.) Press "Reset" on console, put binary deck in hopper and have card reader "Ready". Start 701 manually at F6. Press "Start" when card reader stops on last card.

Transfer Entry: (026 not in E.S.) Give following orders in program: Read Card Reader, - Copy F0, TR F0. (026 already in E.S.) Tr to F6. For transfer entry the low order of the accumulator has to be zero.

DESCRIPTION: The binary half-words on each card of the deck to be loaded are read and stored in consecutive E.S. locations from R to V + R - 1 for that card. After storage, each word is brought back from its location and subtracted from the check sum. Either V or R may be odd or even. 026 will load into any part of E.S. except the 50 half-word occupied by itself. 026 will load to the end of memory.

026 always turns on the overflow indicator. The entire 9 row of the transition card is read into F2 and F4. Column 9 in the 9 row must be blank. 026 transfers to F2 when this blank is sensed.

PROGRAM STOPS:

Location	Meaning
F44	S on the card does not agree with computed check Sum. Press "Start" to continue card reading. 026 will stop on next card if the low order of the accum. is not zero.
F11 (if no transition card)	End of file condition: Copy check. All binary cards are loaded.

RESTARTING: Manual Transfer to F6.

STORAGE: F0 to F49 (50 half-words). F0 to F5 is the self-loading part of 026 and is used during loading of binary cards for erasable. F6 to F45 is occupied by 026. F46 to F49 is used for erasable storage during loading.

CODED: Dura W. Sweeney, 3/11/54.

This write-up replaces the previous 026R - 1 & 2.

October 29, 1954

026R:	Self-loading, Loads binary cards	026	026 - 0000
STARTING:	Automatic Entry	F0	0000_8
	Manual Entry	F6	0006_8
	Transfer Entry	F6	0006_8
STOPS:	Check sum disagrees	F44	0054_8
	End of File	F10	0013_8
STORAGE:	Decimal	F0-	0000
		F49	0049
	Octal	F0-	0000
		F61	0061

026 is available in octal regions 0000, 1000, 2000, 3000, 4000, 5000, 6000, and 7000. Entry, Stop, and Storage locations are easily computed by adding the high order octal digit to the locations specified for 026 - 0000.

This page replaces the previous pages for 026.

027 R	Reads regional binary cards into specified locations, 43 half-words per card, into ES-1 or ES-2.	
INPUT:	Each card must contain in the 9 row in binary:	
	Cols. 9 thru 35	S, the card check sum for that card.
	Cols. 36 thru 44 46 thru 80	V, the number of half-words on that card, variant and invariant information.
	Row 8	
	Cols. 9 thru 26	R, the location of the first half-word on the card. + if ES-1, - if ES-2.
	Cols. 27 thru 80	The first three half-words.
	Row 7 - 12	The remainder of the half-words.
INCREMENT CARD:	An increment card containing the following information punched in the 9 row in binary <u>must</u> follow 027.	
	Cols. 10 - 26	Lower cut address.
	Col. 27	Sign of the increment.
	Cols. 28 - 44	Increment.
	Cols. 46 - 62	Upper cut address.
	027 adds the increment on the control card to the FWA and all variant addresses, if they are greater than or equal to the lower cut address and less than the upper cut address. If the increment is to be added to all locations and variant addresses, the lower cut address is zero and the upper cut address is $(1\ 0000)_8$.	
LOADING:	027 is self-loading.	

Loading Deck	Cards
027	3
027 increment card	1
regional binary cards	n
027 increment card	1

LOADING: (contd)	Loading Deck	Cards
	regional binary cards	p

	---------------------	
	transition from 027, if desired.	1

Several programs may be entered simultaneously. If they all have the same increment, only one increment card is needed. But a different increment card may precede each program.

TRANSITION: A card is considered a binary transition card if the 9 row left is positive and columns 46-62 are zero.

STARTING: a. Automatic entry: Set load selector to cards. Put loading deck in hopper and have card-reader ready. Set instruction entry keys for 027, automatic-manual switch to automatic and press load button. When select light goes out, feed cards out of card reader.

b. Manual entry (when 027 is already in E.S.) Press reset. Put regional binary deck preceded by increment card in hopper and have card-reader ready. Start 701 manually at $F(0026)_8$.

DESCRIPTION: The 43 binary half-words of each card are read, the increment is added to all the variant addresses and to the FWA, if they are greater than or equal to the lower cut address and less than the upper cut address, and they are stored in E.S. locations R + (increment) through R + V + (increment) - 1. Check is made to see that the sum of the information read off the card agrees with the check sum read from that card. Also, each half-word is read out of its stored location and compared with what was stored there.

Note: There is an ambiguity as to whether an address refers to ES-1 or ES-2. 027 will add the increment to <u>all</u> addresses in the range of the lower and upper cut address since it is unable to determine whether they refer to ES-1 or ES-2. (027 will load 081, therefore the user may restore incorrectly changed addresses back to the original).

PROGRAM STOPS:

Regional Location	Meaning
$F(0176)_8$	S on the card does not agree with the computed check sum. If start is pressed, 027 will load the next card.
$F(0151)_8$	Half-word read from E.S. location does not compare with number stored there. Restart.
$F(0040)_8$	End of file. (Copy check).

OUTPUT: Binary half-words stored in specified E.S. locations.

RESTARTING: Start as before, see starting.

STORAGE: Regional $F(0000)_8$ - $F(0177)_8$

Total - 128 half-words

CODED: D. W. Sweeney, June 23, 1954

	027 R	027 A	027 B	027 C
027 Self-loading, load regional binary cards into ES-1 or ES-2.				
STARTING: Set instruction entry keys to	$F0_8$	0	4000_8	7000_8
For manual entry, start 701 at	$F26_8$	26_8	4026_8	7026_8
PROGRAM STOPS: Check sum computed disagrees with sum read	$F176_8$	176_8	4176_8	7176_8
End of file	$F40_8$	40_8	4040_8	7040_8
Half-word read from E.S. location does not compare with number stored there	F151	151_8	4151_8	7151_8
STORAGE: Octal	$F0_8$	0	4000_8	7000_8
thru	$F177_8$	177_8	4177_8	7177_8

Note: 027 is available in all regions, 0000, 1000, 2000, 3000, 4000, 5000, 6000, 7000, octal.

NO.	NAME
028	Load itself into ES-1, read binary half-words into consecutive E.S. locations in ES-1 or ES-2, read binary half-words back from E.S. locations and form check sum. Load to end of memory of either ES-1 or ES-2.

INPUT: Each card must contain in the 9 row in binary

Columns	Content
9 - 44	S, the card check sum for that card.
51 - 62	V, the number of half-words for that card.
69 - 80	± R, the location of the first half-word to be loaded from that card. + R indicates that the half-words are loaded into ES-1, -R indicates that the half-words are loaded into ES-2.

Rows 8 thru 12 contain up to 44 half-words, four half-words per row in columns 9-26, 27-44, 45-62, and 63-80.

LOADING: 028 is self-loading.

Loading Deck	# Cards
028	2
Binary cards	n
Transition from 028 (if desired)	1 (or 0)
Total	n + 3 (or n + 2)

STARTING: Automatic Entry: Put loading deck in hopper, and have card reader "Ready". Set instruction key to F0, press load button. Press "Start" on console when card reader stops on last card.

Manual Entry: (When 028 is already in E.S.) Press "Reset" on console, put binary deck in hopper and have card reader "Ready". Start 701 manually at F6. Press "Start" when card reader stops on last card.

Transfer Entry: (028 not in E.S.) Give following orders in program: Read Card Reader, - Copy F0, TR F0. (028 already in E.S.) Tr to F6.

DESCRIPTION: The binary half-words on each card of the deck to be loaded are read and stored in consecutive E.S. locations in either ES-1 or ES-2 from R to R + V - 1 for that card. After storage, each word is brought back from its location in either ES-1 or ES-2 from the check sum. Either V or R may be odd or even. 028 will load into any part of ES-1 or ES-2 except the 58 half-word occupied by itself in ES-1. 028 will load to the end of either ES-1 or ES-2. <u>028 turns off the overflow indicator.</u> The entire 9 row of the transition card is read into F2 and F4. Columns 45-62 in the 9 row must be blank. 028 transfers to F2 when this blank is sensed. The following orders may be given on the transition card:

a. $\pm$ TR(x)
b. $\pm$ STOP(x)
c. $\pm$ Sense 40_8, $\pm$ TR(x)
d. $\pm$ Sense 40_8, $\pm$ STOP(x)

PROGRAM STOPS:

Location	Meaning
F52	S on the card does not agree with computed check Sum. <u>Press "Start" to continue card reading.</u>
F11 (if no transition card)	End of file condition: Copy check. All binary cards are loaded.

RESTARTING: Manual Transfer to F6.

STORAGE: F0 to F57 (58 half-words). F0 to F5 is the self-loading part of 028 and is used during loading of binary cards for erasable. F6 to F53 is occupied by 028. F54 to F57 is used for erasable storage during loading.

CODED: Dura W. Sweeney, 3/11/54.

028:	Self-loading, Loads binary cards	028	028 - 0000
STARTING:	Automatic Entry	F0	0000_8
	Manual Entry	F6	0006_8
	Transfer Entry	F6	0006_8
STOPS:	Check sum disagrees	F52	0064_8
	End of File	F10	0013_8
STORAGE:	Decimal	F0-	0000
		F57	0057
	Octal	F0-	0000
		F71	0071

028 is available in octal regions 0000, 1000, 2000, 3000, 4000, 5000, 6000, and 7000. Entry, Stop, and Storage locations are easily computed by adding the high order octal digit to the locations specified for 028 - 0000.

NO.	NAME	CROSS REFERENCES OLD NAMES
080 R	Read octal instructions into specified locations	BIDOIR

INPUT: Octal instructions to be loaded by 080 are punched, one instruction per card, as follows:

column 17: sign of the instruction, 12 for +, 11 for -.

columns 35 thru 38: location of the instruction (must be +)

columns 39 and 40: operation part of instruction.

columns 41 thru 44: address part of instruction

All of the above information must be punched in octal. There must be one, and only one, punch per column in columns 17 and 35 thru 44. All the other columns of the card will be ignored by 080; they may be used in any manner desired for identification, comments, or other information not to be loaded by 080. 080 will load octal cards punched by IBM SO_2. 080 will load any portion of E.S. except the 112 half-words occupied by itself.

LOADING: Load 080 binary cards with 021 . See 021 for complete loading instructions

Loading Deck	# Cards
021	1
080	3
Transition to 080	1

Loading Deck	# Cards
Octal instructions	n
080 transition (if desired)	1 (or 0)
Total	n + 6 (or n + 5)

If a transition card from 080 is used, the transfer will not be executed until the last card in hopper has been read.

STARTING: a. Automatic entry: Put the loading deck in hopper and have card reader ready. Set load selector to cards, instruction entry keys for 021, automatic-manual switch to automatic, and press load. When 701 stops on the last card, press card-reader start. Feed out cards when select light on card reader goes out.

b. Manual entry (when 080 is already in E.S.): Put octal instruction cards (and transition from 080 if desired) in the card-reader and have it ready. Start 701 manually at F0. Feed out cards when card-reader select light goes out.

c. Entry by unconditional transfer: Have instruction deck (followed by transition if desired) in card-reader and ready. Transfer to F0.

DESCRIPTION: The 701 will read in each instruction, convert it to binary, and store it in the specified half-word location, checking for omitted and double punches. When finished loading, the 701 will stop. 080 always loads all the cards in the hopper. If transition from 080 is required, punch the instruction in octal with location F6.

If no transition card follows the instruction deck to be loaded, 080 after loading will execute the instruction stored in F6; if no transition card has been read since loading of 080, F6 will contain a stop.

PROGRAM STOPS:

Regional Location	Meaning
F6	End of file; all the cards in the hopper have been read, i.e., all instructions are loaded. To load another octal deck, have card reader ready and press start.
F64	The card being read contains a double punch or lacks a punch in some column 35 thru 44 or 17. Take the remaining cards out of the hopper and feed out those in the card-reader. Look at the third card back; correct the card, put these three cards and the remaining deck back in the hopper, have card-reader ready and press start. If there is no punching error in columns 35 thru 44 on the third card back, the 701 has made an error in summing. Put the three cards and the remaining deck in the hopper and proceed as with stop F6 above.

OUTPUT: Binary instructions stored in specified half-word locations of E.S.

RESTARTING: Startas before (see STARTING b or c).

STORAGE: Regional

B0	thru	B18
F0	thru	F84
E0	thru	E7

Total 112 half-words, 104 regional cards.

For SO_2 assembly, origins B0, F0, and E0 must be specified. E0 must be even.

CODED: AIB, ch'd - dtm, written - dtm.

		R	A	B	C
080	Read octal instructions into specified locations				
INPUT:	Punch transition from 080 with octal location. . .	F6	72_8	4072_8	7072_8
LOADING CARD:	. . .		021A	021B	021C
STARTING:	For loading deck, set instruction entry keys to. . .		0	4000_8	7000_8
	For manual entry, start at. . .	F0	64_8	4064_8	7064_8
	For unconditional transfer entry, transfer to . . .	F0	52_{10}	2100_{10}	3636_{10}
PROGRAM STOPS:	end of file. . .	F6	72_8	4072_8	7072_8
	Punch error on third card back. . .	F64	164_8	4164_8	7164_8
STORAGE:	decimal. . .	B0-	137-	2185-	3721-
	thru	B18	155	2203	3739
		F0-	52-	2100-	3636-
	thru	F84	136	2184	3720
		E0-	2-	2048-	3584-
	thru	E7	9	2055	3591
	octal. . .	B0-	(211-	(4211-	(7211-
	thru	B18	$233)_8$	$4233)_8$	$7233)_8$
		F0-	(64-	(4064-	(7064-
	thru	F84	$210)_8$	$4210)_8$	$7210)_8$
		E0-	(2-	(4000-	(7000-
	thru	E7	$11)_8$	$4007)_8$	$7007)_8$

H. Kolsky
T-5

NO.	NAME
081 R	Read octal instructions into specified locations in ES-1 or ES-2.

INPUT: Octal instructions to be loaded by 081 are punched one per card as follows:

Column	Contents
9	Sign of operation: x-punch if negative, blank if positive.
10 - 13	Location of the instruction in octal.
14 - 15	Operation part of the instruction in octal.
16 - 19	Address part of the instruction in octal.
20	Blank if location is in ES-1; x-punch if location is in ES-2.

LOADING:

Loading Deck	# Cards
026 (or 028)	1 (or 2)
Coder's binary deck	n
081	1
Octal instructions	p
Coder's binary transition card (if desired)	1
Total	3+n+p (or 4+n+p)

STARTING:

a. Automatic entry: Put loading deck in Card Reader, set instruction entry keys for 026 (or 028), press load. When 701 stops, press card-reader start.

b. Manual entry: (081 already in ES): Put octal instruction cards followed by binary transition card if desired in card-reader. Start 701 manually at F6.

c. Entry by unconditional transfer: Have octal instructions followed by binary transition card in card-reader. Transfer to F6.

DESCRIPTION: 081 self-loads itself over 026 (or 028), then reads the following octal instruction cards, converts them to binary, and stores the instruction in the location specified. 081 does not check for double-punching or blank columns. 081 turns off the overflow indicator.

081 is designed so that the coder can insert 081 and the octal instructions to be loaded between his binary cards and his transition card. It self-loads itself over the original binary loading card (026 or 028) so that no extra space is occupied except that required for the original binary loading card.

081 will load instructions into any location in ES-1 or ES-2 except the 42 half-words occupied by itself in ES-1. The coder must use 081 in the same region as 026 or 028.

PROGRAM STOPS: F11: Copy check: End of file condition indicating that no transition card was read.

OUTPUT: Binary half-words stored in specified location in ES-1 or ES-2.

STORAGE: F0 to F41, 42 half-words.

CODED: Dura W. Sweeney, 4/20/54.

		Region
081:	Read octal instructions into ES-1 or ES-2	0000
STARTING:	Manual Entry: Start at	0006_8
	Unconditional TR: TR to	0006_8
STOP:	End of File: Copy check	0013_8
STORAGE:	Decimal	0000-0041
	Octal	0000-0051

081 is available in all octal regions 0000, 1000, 2000, 3000, 4000, 5000, 6000, and 7000. Add the high order digit of the octal region to the above stop to get the proper stop address.

The coder must use 081 in the same region as 026 or 028.

H. Kolsky
T-5

DESCRIPTION: 081 acts as its own transition card, then self-loads itself over 026 (or 028), then reads the octal instruction cards following, converts them to binary, and stores the instruction in the location specified. 081 does not check for double-punching or blank columns. 081 turns off the overflow indicator.

The entire nine row of an octal instruction card must be blank.

081 is designed so that the coder can insert 081 and the octal instructions to be loaded between his binary cards and his transition card. It self-loads itself over the original binary loading card (026 or 028) so that no extra space is occupied except that required for the original binary loading card.

081 will load instructions into any location in ES-1 or ES-2 except the 42 half-words occupied by itself in ES-1. The coder must use an 081 in the same region as his 026 (or 028) card.

PROGRAM STOPS: F11: Copy check: End of file condition indicating that no transition card was read.

OUTPUT: Binary half-words stored in specified location in ES-1 or ES-2.

STORAGE: F0 to F41, 42 half-words.

CODED: Dura W. Sweeney, 4/20/54.

This page replaces the previous page 081 R - 2.

NO.	NAME
086R	Read octal absolute instructions, up to 12/card, into blocks of E.S.

DESCRIPTION: Blocks of octal absolute instructions are converted to binary and stored in blocks of electrostatic storage by 086. The initial storage location of each block is specified by a heading card. 086 checks to see that no columns of the control card or the data cards are blank or have double punches, and ignores any punches in the 8 or 9 rows. 086 also checks to make sure the first card it reads is a control card.

INPUT: The control card is punched as follows:

Columns 9	11 punch
9-10	0
11-14	Initial loading address of block of E.S. in octal; may be even or odd

The instruction cards are punched in octal as follows:

Columns 9-14	1st instruction
15-20	2nd instruction
21-26	3rd instruction
27-32	4th instruction
33-38	5th instruction
39-44	6th instruction
45-50	7th instruction
51-56	8th instruction
57-62	9th instruction
63-68	10th instruction
69-74	11th instruction
75-80	12th instruction

In addition,

If 1st instruction is negative, there must be an 11 punch in col 14

2nd " " " " " " " " " " " 20

3rd " " " " " " " " " " " 26

4th " " " " " " " " " " " 32

5th " " " " " " " " " " " 38

6th " " " " " " " " " " " 44

7th " " " " " " " " " " " 50

8th " " " " " " " " " " " 56

9th " " " " " " " " " " " 62

10th " " " " " " " " " " " 68

11th " " " " " " " " " " " 74

12th " " " " " " " " " " " 80

If n<12 words are to be loaded, the last 6(12-n) columns should be blank.

LOADING: 086 is loaded with 021. See 021 for complete loading instructions.

Loading Deck	# Cards
021	1
086	5
Transition to 086	1
Control Card	1
Instruction Cards	n
Control Card	1
Instruction Cards	n

etc.

STORAGE:	E0 thru E28, E0 even	
	A0 thru A2, A0 even	
	F0 thru F205	
	209 regional cards, 5 binary cards.	
STOPS:	F 18	End of file, all instructions loaded. Push start to read more cards. There will be no check for a leading control card.
	F 114	Control card has a blank column or is double punched. Correct card, place it in the reader, have the card reader ready. Push Start to continue.
	F 144	Instruction card has a blank column or a double punch. Correct card, place it in the reader, have the card reader ready. Push Start to continue.
CODED:	Scully 6/53	

086 Read blocks of decimal absolute instructions into E.S.

		086R	086A	086B	086C
START: Transition card punched (octal)		F0	$(67)_8$	$(4067)_8$	$(7067)_8$
STORAGE:	decimal	E0-	0-	2048-	3584-
		E28	28	2076	3612
		A0-	52-	2100-	3636-
		A2	54	2102	3638
		F0-	55-	2103-	3639-
		F205	260	2308	3844
	octal	E0-	(0-	(4000-	(7000-
		E28	$34)_8$	$4034)_8$	$7034)_8$
		A0-	(64-	(4064-	(7064-
		A2	$66)_8$	$4066)_8$	$7066)_8$
		F0-	(67-	(4067-	(7067-
		F205	$404)_8$	$4404)_8$	$7404)_8$
STOPS: All instructions stored	decimal	F18	73	2121	3657
	octal		$(0111)_8$	$(4111)_8$	$(7111)_8$
Control card BCDP	decimal	F114	169	2217	3753
	octal		$(0251)_8$	$(4251)_8$	$(7251)_8$
Instruction card BCDP	decimal	F144	199	2247	3783
	octal		$(0307)_8$	$(4307)_8$	$(7307)_8$

110 R Print floating decimal data

INPUT: The following calling sequence is required for the linkage entry

α	$\pm$	R Add	α
$\alpha + 1$	$\pm$	Tr	1FO
$\alpha + 2$	+	n	ℓ
$\alpha + 3$	$\pm$	0	FWA (must be even)
$\alpha + 4$	$\pm$	0	LWA (must be even and > FWA)
$\alpha + 5$	+	0	t_1
$\alpha + 6$	+	0	t_2
$\alpha + 7$	+	0	t_3 $\quad 0 \leq t_i \leq 33$
$\alpha + 8$	+	0	t_4
$\alpha + 9$	+	0	t_5
$\alpha + 10$	+	0	t_6
$\alpha + 11$	+	0	t_7
$\alpha + 12$	Control automatically returns to here		

where n = the number of words printed per line

ℓ = the number of data lines per block; if $\ell \leq 28$ there are two blocks per page, and if $\ell > 28$, there is one block per page.

FWA = first word address = the location of the first full word of data to be printed

LWA = last word address = the location of the last full word of data to be printed.

t_i = the number of binary places before the binary point, counted from left to right.

For example, n = 7, ℓ = 10, FWA = 0, LWA = 140, t_i = i - 1 for i = 1, 2, . . ., 7. 110 will print out 7 words per line, 10 lines per block, two blocks per page, getting the data from E.S. locations - 0 thru - 140. 110 will assume the binary points to be located as follows:

	bit 1	2	3	4	5	6	7	...	34	35
1st word of line	.x	x	x	x	x	x	x	...	x	x
2nd word of line	x	.x	x	x	x	x	x	...	x	x
3rd word of line	x	x	.x	x	x	x	x	...	x	x
4th word of line	x	x	x	.x	x	x	x	...	x	x
5th word of line	x	x	x	x	.x	x	x	...	x	x
6th word of line	x	x	x	x	x	.x	x	...	x	x
7th word of line	x	x	x	x	x	x	.x	...	x	x

LOADING: Load 110 binary cards with 021. See 021 for loading details.

STARTING: Put 110 print board in printer and have printer ready. Entry is by linkage only, see INPUT. Do <u>not</u> restore paper. Paper should be positioned so that over half a page, but not an entire page, is out (beyond type bars). 110 will then restore the paper properly for one or two block print outs.

DESCRIPTION: 110 will print out the full word data in floating decimal form, n words per line, ℓ data lines per block plus the exponent line; ℓ does <u>not</u> include the line required for printing this power of 10 which multiplies each column. This first line of each block is the number of places the decimal point should be shifted to the right for the column of fractions below it. Each data word is a ten digit decimal fraction. If $n < 7$, there will be $7 - n$ columns of zeros to the right of the n data columns. Also if the number of full words called for, $\frac{LWA - FWA + 2}{2}$, is not an integral multiple of n, zeros will be printed to fill out the last line. The zeros do not need to be supplied to 110 by the coder. For example, if FWA = 0, LWA = 8, then the number of full words = 5. If n = 4 (4 words per line), then the print out will be:

exp line	0	1	2	3	0	0	0
data line	.xxx...	.xxx...	.xxx...	.xxx...	.000...	.000...	.000...
data line	.xxx...	.000...	.000...	.000...	.000...	.000...	.000...

The error mark (-) on the extreme left means that for the previous line the echo impulses from the printer did not agree with the impulses originally sent; therefore the printer made an error (either printed incorrectly, or sent incorrect echoes). The line with the error mark is correct if the line below it has no error mark. The error mark indicates only that echoes did not agree and there may be a printing error on the line just above. Whenever there is a printer error, 110 will try again to print that line, and continue trying until the echo impulses agree and the line is printed correctly.

OUTPUT: Floating decimal data (see DESCRIPTION).

STORAGE: Regional A0 thru A3, A0 even

1B0 thru 1B14, 1B0 even

1F0 thru 1F275

E0 thru E75, E0 even

Total 371 halfwords.

CODED: Voorhees, written DTM

11/2/53 - This is the replacement for page 4 of the 110 writeup.

110 Print floating decimal data		110R	110A	110B	110C
+ 1 contains $\pm$ tr to ...		1F0	78_{10}	2124_{10}	3660_{10}
LOADING card; when loading, set			021A	021B	021C
instruction entry keys to ...			0	4000_8	7000_8
STORAGE:	decimal	A0-	352-	2400-	3936-
	thru	A3	355	2403	3939
		1B0	356-	2404-	3940-
	thru	1B14	370	2418	3954
		1F0-	76-	2124	3660-
	thru	1F275	351	2399	3935
		E0-	0	2048	3584-
	thru	E75	75	2123	3659
	octal	A0-	(540-	(4540	(7540-
	thru	A3	543)$_8$	4543)$_8$	7543)$_8$
		1B0-	(544-	(4544-	(7544-
	thru	1B14	562)$_8$	4562)$_8$	7562)$_8$
		1F0-	(114-	(4114-	(7114-
	thru	1F275	537)$_8$	4537)$_8$	7537)$_8$
		E0-	0	(4000-	(7000-
	thru	E75	113_8	4113)$_8$	7113)$_8$

11/2/53 - This is the replacement for the absolute location sheet of the 110 writeup.

NO.	NAME
111R	Print half-word floating decimal data

INPUT: The following calling sequence is required for the linkage entry:

α	+	R Add, α	
$\alpha + 1$	$\pm$	Tr, FO	
$\alpha + 2$	+	n, ℓ	
$\alpha + 3$	$\pm$	0, FWA	
$\alpha + 4$	$\pm$	0, LWA	(LWA > FWA)
$\alpha + 5$	+	t_1, t_2	
$\alpha + 6$	+	t_3, t_4	
$\alpha + 7$	+	t_5, t_6	$0 \leq t_i \leq 16$
$\alpha + 8$	+	t_7, t_8	
$\alpha + 9$	+	t_9, t_{10}	
$\alpha + 10$	+	t_{11}, t_{12}	
$\alpha + 11$	+	t_{13}, t_{14}	
$\alpha + 12$		Control automatically returns here	

Where: n = the number of half words printed per line $(0 < n \leq 14)$

ℓ = the number of lines per block; if $\ell \leq 28$ there are two blocks per page; if $\ell > 28$ there is one block per page.

FWA = the location of the first half word of data to be printed.

LWA = the location of the last half word of data to be printed.

t_i = the number of binary places before the binary point, counted from left to right.

LOADING: Load 111 binary cards with 021, see 021 for loading details.

STARTING: Put 111 board in printer and have printer ready. Entry is by linkage only. Do not restore paper. Paper should be positioned so that over half a page, but not an entire page, is beyond type bars. 111 will then restore the paper properly for one or two block print outs.

DESCRIPTION: 111 will print out the half word data in floating decimal form, n words per line, ℓ data lines per block plus the exponent line; ℓ does not include the line required for printing this power of 10 which multiplies each column. This first line of each block is the number of places the decimal point should be shifted to the right for the column of fractions below it. Each data word is a five digit decimal fraction. If $n < 14$, there will be 14 - n columns of zeros to the right of the n data columns. Also if the number of words called for, FWA - LWA + 1, is not an integral multiple of n, zeros will be printed to fill out the last line. Zeros do not need to be supplied to 111 by the coder.

The error mark (-) on the extreme left means that, for the previous line, the echo impulses from the printer did not agree with the impulses originally sent; therefore, the printer made an error. The line with the error mark is correct if the line below it has no error mark. The error mark indicates only that echos did not agree and there may be a printing

H. Kolsky
T-5

error on the line above. Whenever there is a printing error, 111 will try again to print that line, and continue trying until the echo impulses agree and the line is printed correctly.

111 rounds the numbers before printing.

OUTPUT: Floating decimal half-word data.

STORAGE: Regional: E0 thru E83, E0 even
A0 thru A11, A0 even
F0 thru F352
Total 365 half-words. For 607 assembly specify E0, A0 and F0.

CODED: Scully, checked: Voorhees.

12/21/53 This page replaces page 3 of the 111R writeup.

111 Print half-word floating decimal data.

		111 R	111 A	111 B	111 C
α + 1 contains $\pm$ Tr to		F0	$(96)_{10}$	$(2144)_{10}$	$(3680)_{10}$
When loading set			021A	021B	021C
instruction entry keys to			0	4000_8	7000_8
STORAGE:	decimal	E0-	0	2048-	3584-
	thru	E83	83	2131	3667
		A0-	84-	2132-	3668-
	thru	A11	95	2143	3679
		F0-	96-	2144-	3680-
	thru	F352	448	2496	4032
	octal	E0-	(0-	(4000-	(7000-
	thru	E83	$123)_8$	$4123)_8$	$7123)_8$
		A0-	(124-	(4124-	(7124-
	thru	A11	$137)_8$	$4137)_8$	$7137)_8$
		F0-	(140-	(4140-	(7140-
	thru	F352	$700)_8$	$4700)_8$	$7700)_8$

12/21/53 This page replaces the old absolute location page of the 111 writeup.

H. Kolsky
T-5

NO.	NAME
112 R	Print half-word floating decimal data from ES-1 or ES-2.

INPUT: The following sequence is required for linkage entry:

To print from ES-1, see 111 R.

To print from ES-2,

Coder's Program

α	+ R Add	$\alpha + 2$
$\alpha + 1$	+ Tr	β
$\alpha + 2$	+ Tr	$\alpha + 3$ (if return is to bank 1)
		or
	- Tr	$\alpha + 3$ (if return is to bank 2)
$\alpha + 3$	Control returns here	

Frame I

β	+ Sense 32			
$\beta + 1$	Store $\beta + 14$			
$\beta + 2$	R Add $\beta + 2$			
$\beta + 3$	Tr	FO		
$\beta + 4$	n,	ℓ		
$\beta + 5$	0,	$\pm$ FWA	{ + if in ES-1; - if in ES-2 }	LWA > FWA
$\beta + 6$	0,	$\pm$ LWA	{ + if in ES-1; - if in ES-2 }	LWA > FWA
$\beta + 7$	t_1,	t_2		
$\beta + 8$	t_3,	t_4	$0 \leq t_i \leq 16$	
$\beta + 9$	t_5,	t_6		
$\beta + 10$	t_7,	t_8		
$\beta + 11$	t_9,	t_{10}		
$\beta + 12$	t_{11},	t_{12}		
$\beta + 13$	t_{13},	t_{14}		
$\beta + 14$	[exit]			

For definitions of n, ℓ, FWA, LWA and t_i, see 111 R.

LOADING: See 111 R.

STARTING: See 111 R.

DESCRIPTION: See 111 R.

OUTPUT: Floating decimal half-word data.

STORAGE: Regional E0 - E83, E0 even

A0 - A11, A0 even

F0 - F359

Total 372 half-words. For 607 assembly, specify E0, A0 and F0.

NOTE: When possible, use 111, since less storage space is occupied by that program.

CODED: Freshour, Checked & written, Korell

112 Print half-word floating decimal data.

	112 R	112 A	112 B	112 C
α + 1 or β + 3 contains Tr to	F0	$(96)_{10}$	$(2144)_{10}$	$(3680)_{10}$
When loading set		026 A	026 B	026 C
instruction entry keys to		0	4000_8	7000_8
STORAGE: decimal	E0	0-	2048-	3584-
thru	E83	83	2131	3667
	A0-	84-	2132-	3668-
thru	A11	95	2143	3679
	F0	96-	2144-	3680-
thru	F359	455	2503	4039
octal	E0	(0-	(4000-	(7000-
thru	E83	$123)_8$	$4123)_8$	$7123)_8$
	A0	(124-	(4124-	(7124-
thru	A11	$137)_8$	$4137)_8$	$7137)_8$
	F0	(140-	(4140-	(7140-
thru	F359	$707)_8$	$4707)_8$	$7707)_8$

NO.	NAME
185 R	Print Octal Instructions

INPUT: Control Card: Punch in binary in the nine row the following information in the specified columns:

	Columns
First Location	15 - 26
Last Location	33 - 44

LOAD: Load 185 binary cards with 021 . See 021 for complete loading instructions.

Loading Deck	# Cards
021	1
185	2
Transition to 185 (TR FO)	1
Control Cards	n
Total	n + 4

STARTING: a. Automatic entry: Put the loading deck in hopper and have card reader ready. Set load selector to cards, instruction entry keys for 021, and press load. When 701 stops on last card, press card reader start.

b. Manual entry (when 185 is already in E.S.): Place control cards in card reader and have it ready. Start 701 manually at FO.

c. Entry by unconditional transfer: Have control cards in card reader. Transfer to FO.

DESCRIPTION: The 701 will print locations in octal and the contents of those locations as octal instructions, starting with the first location and ending with the last (see INPUT).

Note: The last location must be greater than or equal to the first location.

Note: Use 793 tracing board in printer.

PROGRAM STOP:

F59 All information requested by control card has been printed. To read next control card press start button on console.

OUTPUT: Printed sheets, one instruction per line.

STORAGE: Regional A0 thru A2

F0 thru F61

E0 (even) thru E11

Total - 77 half-words, 65 regional cards.

For SO_2 assembly, origins A0, F0, E0 must be specified.

CODED: BW, Ch'd - BW, written BW

List Octal Instructions

	R	A	B	C	M1
STORAGE: Decimal	A0 -	52 -	2100 -	3636 -	100 -
	A2	54	2102	3638	102
	F0 -	55 -	2103 -	3639 -	103 -
	F61	116	2164	3700	164
	E0 -	2 -	2048 -	3584 -	168 -
	E12	14	2060	3596	180
Octal	A0 -	(64 -	(4064 -	(7064 -	(144 -
	A2	$66)_8$	$4066)_8$	$7066)_8$	$146)_8$
	F0 -	(67 -	(4067 -	(7067 -	(147 -
	F61	$164)_8$	$4164)_8$	$7164)_8$	$244)_8$
	E0 -	(2 -	(4000 -	(7000 -	(250 -
	E12	$16)_8$	$4014)_8$	$7014)_8$	$264)_8$
START AT:					
Decimal	F0	0055	2103	3639	0103
Octal	F0	$(0067)_8$	$(4067)_8$	$(7067)_8$	$(0147)_8$
STOP, ALL THROUGH					
Decimal	F59	0114	2162	3698	0162
Octal	F59	$(0162)_8$	$(4162)_8$	$(7162)_8$	$(0242)_8$

H. Kolsky
T-5

NO.	NAME
186 R	Print contents of electrostatic memory in octal.

LOADING: Load 186 binary cards with 021. See 021 for loading instructions.

Loading Deck	# Cards
021	1
186	3
Transition to 186	1
Total	5

STARTING: a. Automatic entry: Put loading deck in hopper of card-reader; have card-reader ready; set instruction keys for 021, press load. When card-reader stops, press start on card-reader.

b. Manual entry (when 186 is already in electrostatic). Set instruction keys to 0F0; enter instruction; press start on console.

c. Entry by transfer: Transfer to 0F0.

DESCRIPTION: 186 will search the electrostatic memory starting at the first half word following itself (0F105) for the first half word not plus zero or not all minus ones. It will then print the location of that half word, the half word, and the following ten half words (whether zero, minus ones, or otherwise). It will then continue the search. 186 searches from the half word following itself (0F105) to the half word preceding its erasable storage (0E0 minus 1). 186 will always print at least one line. Since the erasable storage of 186 is located in the last 46 half words of the space occupied by 021, it will always print the first six half words of 021.

PROGRAM STOPS:

Regional Location	Meaning
0F92	Search is complete.

OUTPUT: Printed sheets, eleven octal instructions and the location of the first instruction per line.

STORAGE: 0E0 - 0E45 (the last 46 half words of the space occupied by 021)

0F0 - 0F104 (follows 021)

Total 151 half words. 105 regional cards.

CODED: DWS, ch'd DWS, written DWS, 9-3-53

186: Print contents of electrostatic memory in octal.

		Reg	A	B	C
INPUT:	α + 1 contains Tr to	OF0	64_8	4064_8	7064_8
STARTING:	For automatic entry set instruction keys to	---	0	4000_8	7000_8
	For manual entry (186 already in electrostatic) set instruction entry keys to	OF0	64_8	4064_8	7064_8
	For entry by transfer, Tr to	OF0	64_8	4064_8	7064_8
PROGRAM STOP:	Search is complete	OF92	220_8	4220_8	7220_8
STORAGE:	Decimal	OE0	6	2054	3590
	thru	OE45	51	2099	3635
		OF0	52	2100	3636
	thru	OF104	157	2205	3741
	Octal	OE0	6	4006	7006
		OE45	63	4063	7063
		OF0	64	4064	7064
		OF104	235	4235	7235

NO.	NAME
188	Searches memory (ES-1 or ES-2 or both) for all references to a given address and prints them in octal. This program destroys the first two full words in ES-1, but otherwise leaves both ES-1 and ES-2 unchanged.

INPUT:

Loading deck	# cards
526	4
026A	1
188	8
transition to 188	1
Total	14

STARTING: Automatic entry: Put loading deck in the hopper of the card reader and have card reader ready. Set instruction keys to zero, and press the load button. Press card reader start when 701 stops on last card. Put 186 printer board in the printer. When program stops at 66_8, put on manual, set MQ entry keys for the search address desired, enter MQ, put on automatic, and then push "Start". Do not "reset". If reset does get pushed, transfer manually to 67_8, enter MQ, and push start.

SWITCHES:

#1 down: 188 searches ES-1

#2 down: 188 searches ES-2

#1 and #2 down: 188 searches both ES-1 and ES-2

#3 up: 188 restores ES-1 after search

#3 down: After search, 188 transfers to 66_8 ready for another search address to be entered into the MQ

DESCRIPTION: 526 writes ES-1 on drum #1 with the exception of the first two full words. 026A loads 188. First 188 stores for printing the contents of the search address in ES-1 if ES-1 is to be searched and the contents of the search address in ES-2 if ES-2 is to be searched. If ES-2 is to be searched, 188 reads a full word at a time from ES-2 and checks each half word for the given address. At the end of searching ES-2, if there is a partial line to be printed, it will be printed with zeros for the rest of the line.

If ES-1 is to be searched, the first part is read in from the drum and 188 searches for the given address. The second part of the drum is then read in and 188 searches it for the given address.

If sense switch 3 is up after 188 finishes searching, ES-1 will be restored except for the first two full words. If #3 is down, 188 will transfer to 66_8 and will be ready to have another search address entered in the MQ.

PROGRAM STOPS:

$(0066)_8$	Enter the search address in the MQ and push "start".
$(0001)_8$	Search is complete and ES-1 has been restored.

OUTPUT: Printed sheets; the first line contains the number of the bank being searched (2 if ES-2, 1 if ES-1), the search address and its contents in ES-1 if ES-1 is to be searched, the search address and its contents in ES-2 if ES-2 is to be searched, and three references to the search address giving location, operation, and address. The rest of the lines contain the number of the bank being searched and five references to the search address (zeros are printed if there are not enough references to complete a line).

Coded, written & checked: D. Solbrig

Harwood Kolsky
T-5

NO.	NAME
189	Prints all transfer orders in octal, from one or two banks of memory. Destroys the first two words in ES-1, but otherwise leaves both ES-1 and ES-2 unchanged.

INPUT:

Loading deck	# Cards
526	4
026A	1
189	8
Transition to 189	1
Total	14

STARTING: Automatic entry: put loading deck in the hopper of the card reader. Have card reader ready. Put 186 board in the printer and have printer ready. Set instruction keys to zero, and press load button. Press card reader start when 701 stops on last card. There is no manual entry. There is no entry by transfer.

DESCRIPTION: 526 writes all of ES-1 on drum #1, with the exception of the first two words, -0000 and -0002. 026A loads 189. If the search is to be conducted in ES-1 only, 189 reads the first half of the drum into ES-1 and then searches for the orders 01, 02, 03, 04. These are printed, five to a line, with their locations and addresses. The second half of the drum is then read in and searched. Upon completion of the search, 189 reads itself out of ES-1 and leaves ES-1 just as it was, with the exception of the first two full words.

If the search is to be conducted in ES-2 only, the contents of ES-1 are read onto drum #1 and the first half of ES-2 is read into ES-1. Searching and printing then continue until the first half of ES-2 is searched; then the second half is read in and searched. Upon completion of the second half search, 189 reads itself out of ES-1, thus leaving both ES-2 and ES-1 unchanged except for the first two full words of ES-1.

If both banks are to be searched, ES-2 is searched first, then ES-1, and finally the contents of drum #1 are read back into ES-1.

SWITCHES:
#1 down: 189 searches ES-1 only
#2 down: 189 searches ES-2 only
#1 and #2 down: 189 searches both ES-1 and ES-2.

PROG. STOP:

Instruction counter	meaning
$(0001)_8$	Search is complete

OUTPUT: Printed sheets, each line containing five locations, their transfer instructions and their addresses. (A two prints on the left if TR from ES-2; one if from ES-1)

CODED: William J. Worlton. Checked, E. A. Voorhees, written, WJW.

NO.	NAME
210	Label punched cards with decimal integer in columns 1-8.

INPUT: By basic linkage.

The full word (I) to be punched is placed in the MQ, with binary point at position 35, by the coder's program before the calling sequence.

The calling sequence is as follows:

α	+	R Add α
$\alpha + 1$	+	Tr. F0 (or F1 - see description below)
$\alpha + 2$		Return

If I is negative, columns 1-8 will be left blank.

If $I > 10^8$ the least significant 8 digits will be punched.

LOADING: Load 210 with 026.

STARTING: Starting by basic linkage occurs automatically.

DESCRIPTION: 210 will punch a card with the identification (I) in columns 1-8. This identification or decimal integer is under the control of the coder's program. The card punched by 210 is then used to gangpunch successive binary cards, punched by 224, 607, or other programs, in columns 1-8. The same integer will be gangpunched until it is stopped in one of the following ways:

1. Reentry to 210 with new number.
2. Clearing out the punch.
3. Entry to 210 with a negative number in which case a blank card will be punched.

Timing difficulties necessitate that in the general case 210 produces two cards punched in cols. 1-8. The first one will be punched, possibly incompletely, with whatever was current just before entry to 210. The second one will be punched with the new identification in cols. 1-8 (blank if $I < 0$). If the programmer is certain that the card punch will be disconnected when he enters 210, he may enter at F1 and avoid obtaining the first of the above two cards.

An example of using 210 follows:

The problem number (4 digits) is to be punched in cols. 1-4.

The run number (4 digits) is to be punched in cols. 5-8.

First identification -

Problem number = 2346

Run number = 5781

23465781 is entered as a constant with a binary scaling of 35, by 607, 012 etc.

This number is placed into the MQ and calling sequence to 210 follows.

To change 23465781 to 23475781, a 1 times 10^4 is added to the first number and we get the new identification:

Problem number - 2347
Run number - 5781

Therefore identification can be changed by doing arithmetic on an initial number.

OUTPUT: 2 (or 1 if entry is at F1) cards with identification or I in cols. 1-8. The standard punch board is used.

STORAGE: Regional

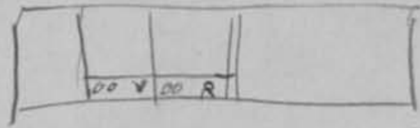
A0 - A2
E0 - E10 (even)
F0 - F42

CODED: Ewing, Wood

CHECKED & WRITTEN: M. Anderson

210 Label punched cards with decimal integer in columns 1-8.

		210 R	210 A	210 B	210 C
INPUT:	α + 1 contains tr to	OFO	146_{10}	2194_{10}	3730_{10}
			A Block	Same as 224	
STORAGE:	Decimal	AO-	142_{10}-	2190_{10}-	3726_{10}-
		A2	144_{10}	2192_{10}	3728_{10}
		FO-	146_{10}-	2194_{10}-	3730_{10}-
		F42	188_{10}	2236_{10}	3772_{10}
		EO-	190_{10}-	2238_{10}-	3774_{10}-
		E10	200_{10}	2248_{10}	3784_{10}
	Octal	AO-	216_8-	4216_8-	7216_8-
		A2	220_8	4220_8	7220_8
		FO-	222_8-	4222_8-	7222_8-
		F42	274_8	4274_8	7274_8
		EO-	276_8-	4276_8-	7276_8-
		E10	310_8	4310_8	7310_8



Punch in Binary

NO. **NAME**

221R Punch in binary consecutive half-words of E.S.

INPUT: Punch control card in binary as follows, 9 row:

Columns	Punched
15 thru 26	V = the number of half-words to be punched
33 thru 44	R = the location of the first half-word to be punched
45 thru 62	Exit instruction to be executed immediately after completing punching

Calling sequence for entry by linkage is as follows:

α	R ADD	α
$\alpha + 1$	TR	F8
$\alpha + 2$		V
$\alpha + 3$		R
$\alpha + 4$	Control automatically returns here	

LOADING: Load 221 with 021 or 024.

Loading Deck	# Cards
021	1
221	3
Transition to 221	1
221 Control Card	1
Total	6

STARTING: a. Automatic entry with control card: Set load selector to cards. Set instruction entry keys to 0 for deck in A utility region, to 4000_8 for B region deck, or to 7000_8 for C region deck. Set automatic-manual switch to automatic, put loading deck in hopper and press card reader start, then load button. When 701 stops on the last card, press card reader start.

b. Manual entry with control card (when 221 is already in E.S.): Press reset. Put control card in hopper and have card-reader ready. Start 701 manually at F0. Feed out control card when card-reader select light goes out.

c. Start by linkage occurs automatically (see INPUT).

DESCRIPTION: 221 will punch in binary the half-words in E.S. locations R thru R + V - 1 with card check sum S, V and R in the 9 row. R and V may be even or odd. If $R + V > 4096_{10}$, 221 after punching the last half-word in E.S. will punch the half-words starting with 0 and continue consecutively until it has punched V half-words. When punching is finished control returns to $\alpha + 4$ if entry was made by linkage or executes the exit instruction punched in the control card. Cards punched by 221 can be loaded with FEJ035, 021 or 024.

PROGRAM STOP: (if no exit instruction was punched in the control card):

Regional Location	Meaning
F7	Punching completed

OUTPUT: Binary cards with S, V and R in the 9 row; rows 8 thru 12 contain the half words of E.S. locations R thru R + V - 1, 44 per card except possibly on the last card where only as many are punched as is necessary to complete the specified range.

RESTARTING: Start as before, see STARTING b or c.

STORAGE: Regional

A0 thru A2

F0 thru F96

E0 thru E7

Total 108 half words

221 is 100 regional cards, 3 binary cards.

For SO_2 assembly origins A0, E0 and F0 must be specified, E0 must be even and F0 must be odd.

CODED: DTM 5-53

221 Punch in binary consecutive half-words of E.S.

		221R	221A	221B	221C
INPUT: α + 1 contains tr to ...		F8	63_{10}	2111_{10}	3647_{10}
STARTING: Set instruction entry keys to ...			0	4000_8	7000_8
To start manually, transfer to ...		F0	67_8	4067_8	7067_8
PROGRAM STOP: punching completed ...		F7	76_8	4076_8	7076_8
STORAGE:	decimal	A0-	52-	2100-	3636-
	thru	A2	54	2102	3638
		F0-	55-	2103-	3639-
	thru	F96	151	2199	3735
		E0-	2-	2048-	3584-
	thru	E7	9	2053	3591
	octal	A0-	(64-	(4064-	(7064-
	thru	A2	$66)_8$	$4066)_8$	$7066)_8$
		F0-	(67-	(4067-	(7067-
	thru	F96	$227)_8$	$4227)_8$	$7227)_8$
		E0-	(2-	(4000-	(7000-
	thru	E7	$11)_8$	$4007)_8$	$7007)_8$

NO.	NAME
222 R	Punch in binary consecutive half-words of E.S.

INPUT: Punch control card in binary as follows, 9 row:

Columns	Punched
15 thru 26	V = the number of half-words to be punched
33 thru 44	R = the location of the first half-word to be punched
45 thru 62	Exit instruction to be executed immediately after completing punching

Leave the rest of the card blank. Calling sequence for entry by linkage is as follows:

α	R ADD	α
$\alpha + 1$	TR	F8
$\alpha + 2$		V
$\alpha + 3$		R
$\alpha + 4$	Control automatically returns here	

LOADING: Load 222 binary cards with 021

Loading Deck	# Cards
021	1
222	2
Transition to 222	1
222 Control Card	1
Total	5

STARTING: a. Automatic start with control card. Set load selector to cards, instruction entry keys for 021, automatic-manual-switch to automatic, put loading deck in hopper and press card reader start, then load button. When select light goes out, feed cards out of card-reader.

b. Manual entry with control card (when 222 is already in E.S.). Press reset. Put control card in hopper and have card reader ready. Start 701 manually at F0. Feed out control card when select light on card-reader goes out.

c. Start by linkage occurs automatically (see INPUT).

DESCRIPTION: Half-words in E.S. locations R thru R + V - 1 are punched without check sums in binary. R and V may be even or odd. When punching is finished control returns to $\alpha + 4$ if entry was made by linkage or executes the exit instruction punched in the control card. Note that both operation and address parts must be punched for the exit instruction, which may be + or -. No check sums are taken or punched, and no locations, initial address or half-word count is punched. 222 is intended primarily to punch self-loading cards.

PROGRAM STOP: (if no exit instruction was punched in the control card)

Regional Location	Meaning
F7	Punching completed

OUTPUT: Binary cards, with consecutive half-words in E.S. locations R thru R + V - 1 punched in rows 9 thru 12, 4 half-words per row, in columns 9 thru 26

27 thru 44

45 thru 62, and

63 thru 80.

There are 48 half words per card except possibly on the last card, where only as many are punched as is necessary to complete punching of the indicated range.

RESTARTING: Start as before (see STARTING b or c).

STORAGE: Regional A1 thru A2

F0 thru F55

E0 thru E4

Total 63 half-words

E0 must be even and F0 must be odd. Origins A0, F0 and E0 must be specified for SO_2 assembly.

CODED: JDM

222 Punch in binary consecutive half-words of E.S.

		222R	222A	222B	222C
INPUT: α + 2 contains tr to ...		F8	63_{10}	2111_{10}	3647_{10}
STARTING: For automatic entry, set instruction entry keys to ...			0	4000_8	7000_8
For manual entry, start 701 at ...		F0	67_8	4067_8	7067_8
PROGRAM STOP: Punching completed		F7	76_8	4076_8	7076_8
STORAGE:	decimal	A1-	53-	2101-	3637-
	thru	A2	54	2102	3638
		F0-	55-	2103-	3639-
	thru	F55	110	2158	3694
		E0-	2-	2048	3584-
	thru	E4	6	2052	3588
	octal	A1-	(65-	(4065-	(7065-
	thru	A2	$66)_8$	$4066)_8$	$7066)_8$
		F0-	(67-	(4067-	(7067-
	thru	F55	$156)_8$	$4156)_8$	$7156)_8$
		E0-	2-	(4000-	(7000-
	thru	E4	6	$4004)_8$	$7004)_8$

NO.	NAME
223R	Punch in binary consecutive half-words of E.S.

INPUT: Punch control card as follows: 9 row left contains in binary

Columns 15 thru 26	R (must be even)
Columns 33 thru 44	V (must be even)
Columns 63 thru 80	Exit instruction,

where R is the location of the first half-word to be punched (initial punch address) and V is the number of half-words to be punched. Both operation and address parts of the exit instruction must be punched. Leave the rest of the card blank.

LOADING: 223 is self loading.

Loading Deck	# Cards
223	1
223 control card	1
Blank card	1
Total	3

STARTING: a. Automatic entry: Press reset. Put loading deck in hopper and have card-reader ready. Set instruction entry keys to F0, automatic-manual switch to automatic, load selector to cards, and press load button. Feed out cards when select light on card-reader goes out.

b. Manual entry (when 223 is already in E.S.): Press reset. Put control card followed by two blank cards in hopper and have card-reader ready. Start 701 manually at F6. Feed out cards when select light on card-reader goes out.

DESCRIPTION: Half-words in E.S. locations R thru R + V - 1 are punched in binary without check sums. When punching is finished the exit instruction punched in the control card will be executed. No check sums are taken or punched; and no locations, initial address, or half-word count is punched. 223 is intended primarily to punch self-loading cards.

PROGRAM STOP (if no exit instruction was punched in the control card):

Regional Location	Meaning
F29	Punching completed

OUTPUT: Binary cards, with consecutive half-words in E.S. locations R thru R + V - 1 punched in rows 9 thru 12, 4 half-words per row, in

Columns 9 thru 26,
27 thru 44,
45 thru 62, and
63 thru 80.

There are 48 half-words per card except possibly on the last card, where only as many are punched as is necessary to complete punching of the indicated range.

RESTARTING: Start as before (see STARTING b).

STORAGE: 223 occupies F0 thru F47 while loading; after loading 223 occupies F0 thru F30 (F31 thru F47 are set to 0's during loading).

For SO_2 assembly, see special instructions, "SO_2 Assembly of Self-loading Programs". Origins F0 and H0 must be specified; F0 and H0 must be even.

CODED: DTM

223 Punch in binary consecutive half-words of E.S.

			223R	223A	223B	223C
STARTING:	For automatic entry, set instruction entry keys to ...			0	4000_8	7000_8
	For manual entry, start 701 at ...		F6	6	4006_8	7006_8
PROGRAM STOP:	Punching completed		F29	35_8	4035_8	7035_8
STORAGE:	while loading	decimal	F0-	0-	2048-	3584-
		thru	F47	47	2095	3631
	while loading	octal	F0-	(0-	(4000-	(7000-
		thru	F47	$57)_8$	$4057)_8$	$7057)_8$
	after loading	decimal	F0-	0-	2048-	3584
		thru	F30	30	2078	3614
	after loading	octal	F0-	(0-	(4000-	(7000-
		thru	F30	$36)_8$	$4036)_8$	$7036)_8$

H. Kolsky
T-5

NO.	NAME
224	Punch in binary, consecutive half-words from ES-1 or ES-2.
INPUT:	Punch control card in binary, as follows; 9 row:

Columns	Punch
9	Sign of R - blank if R is in ES-1 9 punch if R is in ES-2
15 - 26	R = the location of the first half-word to be punched.
33 - 44	L = the location of the last half-word to be punched.
45 - 62	Exit instruction to be executed immediately after completing punching. (Transfer to FO(224) if more than one control card is used.)

Calling sequence for entry by basic linkage is as follows:

Using only one frame -

α	R add
α + 1	Tr OF4
α + 2	$\pm$ stop R (FWA)
α + 3	+ stop L (LWA)
α + 4	control returns here

Using two frames -

Coder's program

α	+ R add α + 2
α + 1	+ Tr β
α + 2	+ Tr α + 3 - or - Tr α + 3 -

Frame 1

β	+ Sense 32
β + 1	Store β + 6
β + 2	R add β + 2
β + 3	Tr OF4
β + 4	$\pm$ stop R (FWA)
β + 5	+ stop L (LWA)
β + 6	control returns here

This page replaces page one of 224 R, May 24, 1954.

LOADING: Load 224 with 026 or 028

Loading Deck	# Cards
026 or 028	1 or 2
224	2
Tr to 224	1
224 Control cards	n

STARTING: a. Automatic entry with control card: Put loading deck in hopper and have card reader "Ready". Set instruction keys to F0, (for 026 or 028) press load button.

b. Manual entry with control card (when 224 is already in E.S.): Press reset. Put control card in hopper and have card reader "Ready". Start 701 manually at F0 (for 224)

c. Start by linkage occurs automatically.

DESCRIPTION: 224 will punch in binary the half-words in ES-1 or ES-2, specified by the first word and last word addresses on the control card or in the basic linkage. It punches the card with check sum, V and R in the 9 row, and 44 half-words or less in rows eight thru twelve. When punching is finished, control returns to $\beta + 6$, if entry was made by basic linkage, or executes the instruction punched in the control card cols. 45-62. Cards punched by 224 can be loaded with 026 or 028.

PROGRAM STOP: (if no exit instruction was punched in the control card)

Regional Location	Meaning
E6	punching completed

OUTPUT: Binary cards with S, V and $\pm$ R in the 9 row; rows 8 thru 12 contain the half-words specified by the first and last word addresses.

STORAGE: Regional

A0 - A2

F0 - F83

E0 - E7

Total - 95 half-words.

224 is 2 binary cards.

CODED: DWS, checked & written - MCF.

H. Kolsky
T-5

224 Punch in binary consecutive half-words from ES-1 or ES-2.

		224R	224A	224B	224C
INPUT:	α + 1 or β + 3 contains	OF4	62_{10}	2110_{10}	3646_{10}
	tr to		76_8	4076_8	7076_8
STARTING:	Set instruction keys	OFO	0	2048_{10}	3584_{10}
			0	4000_8	7000_8
	To start manually transfer to	OFO	58_{10}	2106_{10}	3642_{10}
			72_8	4072_8	7072_8
PROGRAM STOP:	Punching completed	E6	56_{10}	2104	3640
			70_8	4070_8	7070_8
STORAGE:	Decimal	AO-	142_{10}-	2190_{10}-	3726_{10}-
		A2	144_{10}	2192_{10}	3728_{10}
		EO-	50_{10}-	2098_{10}-	3634_{10}-
		E6	56_{10}	2104_{10}	3640_{10}
		FO	58_{10}-	2106_{10}-	3642_{10}-
		F86	141_{10}	2189_{10}	3725_{10}
	Octal	AO-	216-	4216-	7216
		A2	220_8	4220_8	7220
		EO-	62-	4062-	7062
		E6	70_8	4070_8	7070
		FO-	72-	4072_8	7072_8
		F83	215_8	4215_8	7215_8

7-28-54 - This page replaces the previous page for this write-up.

NO.	NAME
320 R	Read / Write full words from / on any tape with check sums.

INPUT: Control card: Punch in binary in the 9 row the following information in the specified columns

	Columns
First Word Address (must be even)	15 - 26
Last Word Address (must be even)	33 - 44
Read = 11,000 / Write = 11,010	46 - 50
Tape No. 100,000,000 = 256, 100,000,001 = 257, 100,000,010 = 258, or 100,000,011 = 259.	51 - 62
Exit Address	69 - 80

Program may be entered by calling sequence:

α	Radd α
$\alpha + 1$	TR FO
$\alpha + 2$	00 First word address
$\alpha + 3$	00 Last word address
$\alpha + 4$	Read / Write Tape No.
$\alpha + 5$	Control returns here

LOADING: Load 320 binary cards with 021

Loading Deck	# Cards
021	1
320	2
Transition card (TR F14) If using control card	1

STARTING: The following methods of starting may be used if 320 is being used with a control card.

a. Automatic entry: Put the loading deck in hopper and have card-reader ready. Set load selector to cards, instruction entry keys for 021, automatic-manual switch to automatic, and press load. When 701 stops on last card, press card-reader start.

b. Manual entry (when 320 is already in E.S.): Place control card in card-reader and have it ready. Start 701 manually at F14.

c. Entry by transfer: Have control card in reader, transfer to F14.

DESCRIPTION: 320 will read or write on any tape with check sums, 320 only reads records that have been written by this program.

Tape must be positioned in correct status when used (See IBM 701 manual for information about tape status)

STOPS: F39: Error in record being read; start over.

STORAGE: A0 - A2

F0 - F48

E0 - E9, E0 even

CODED: JDM, written BW

{Read / Write} on any tape with check sum

		R	A	B	C	M1
STORAGE:	Decimal	A0 -	52 -	2100 -	3636 -	614 -
		A2	54	2102	3638	616
		F0 -	55 -	2103 -	3639 -	617 -
		F48	103	2151	3687	665
		E0 -	2 -	2048 -	3584 -	666 -
		E9	11	2057	3593	675
	Octal	A0 -	(64 -	(4064 -	(7064 -	(1146 -
		A2	66)$_8$	4066)$_8$	7066)$_8$	1150)
		F0 -	(67 -	(4067 -	(7067 -	(1151 -
		F48	147)$_8$	4147)$_8$	7147)$_8$	1231)$_8$
		E0 -	(2 -	(4000 -	(7000 -	(1232 -
		E9	13)$_8$	4011)$_8$	7011)$_8$	1243)$_8$
START WITH CALLING SEQUENCE:	Decimal	F0	0055	2103	3639	0617
	Octal	F0	(0067)$_8$	(4067)$_8$	(7067)$_8$	(1151)$_8$
START WITH CONTROL CARD:	Decimal	F14	0069	2117	3653	0631
	Octal	F14	(0105)$_8$	(4105)$_8$	(7105)$_8$	(1167)$_8$
STOP, CHECK SUMS DISAGREE:	Decimal	F39	0094	2142	3678	0656
	Octal	F39	(0136)$_8$	(4136)$_8$	(7136)$_8$	(1220)$_8$

NO.	NAME
321 R	Read / Write / Read Backward full words from / on any tape without check sums.

INPUT: Control card: Punch in binary in the 9 row the following information in the specified columns.

	Columns
First Word Address (even)	15 - 26
Last Word Address (even)	33 - 44
Read = 11,000 Write = 11,010 Read Backward = 11,001	46 - 50
Tape No. 100,000,000 = 256 100,000,001 = 257 100,000,010 = 258, or 100,000,011 = 259	51 - 60
Exit Address	69 - 80

321 may be entered by the following calling sequence:

α	RADD	α
$\alpha + 1$	TR	FO
$\alpha + 2$	00	First Word Address
$\alpha + 3$	00	Last Word Address
$\alpha + 4$	Read / Write / Read Backward	Tape No.
$\alpha + 5$	Control return here.	

LOADING:

	No. Cards
021	1
321	2
Transition card (TR F14) if using control card	1

STARTING: a. Automatic entry with control card: Set load selector to cards. Set instruction entry keys to 0 for deck in A region, to 4000_8 for B region, to 7000_8 for C region. Set automatic-manual switch to automatic, put loading deck in hopper and press card reader start, then load button. When 701 stops on last card press card reader start.

b. Manual entry with control card (when 321 is already in E.S.): Press reset. Put control card in hopper and have card reader ready. Start 701 manually at F14.

c. Start by linkage occurs automatically.

DESCRIPTION: Will read, write or read backwards from any tape. Does not take a check sum. Tape must be properly positioned in correct status(this program does not remember the number of times it has been used).

STORAGE: A1 - A2

E0 (even) - E3

F0 - F31

CODED: JDM, written BW

321

{Read / Write} on any tape without check sum

	R	A	B	C	M1
STORAGE: Decimal	A1 -	52 -	2100 -	3636 -	614 -
	A2	53	2101	3637	615
	F0 -	54	2102 -	3638 -	616 -
	F31	- 85	2133	3669	647
	E0 -	2 -	2048 -	3584	648
	E3	5	2051	3587	651
Octal	A1 -	(64	(4064 -	(7064 -	(1146 -
	A2	-65)$_8$	4065)$_8$	7065)$_8$	1147)$_8$
	F0 -	(66 -	(4066 -	(7066 -	(1150 -
	F31	147)$_8$	4147)$_8$	7147)$_8$	1207)$_8$
	E0 -	(2 -	(4000 -	(7000 -	(1210 -
	E3	5)$_8$	4011)$_8$	7011)$_8$	1213)$_8$
START WITH CALLING					
SEQUENCE: Decimal	F0	0054	2102	3638	0616
Octal	F0	(0066)$_8$	(4066)$_8$	(7066)$_8$	(1150)$_8$
START WITH CONTROL					
CARD: Decimal	F14	0068	2116	3652	0630
Octal	F14	(0104)$_8$	(4104)$_8$	(7104)$_8$	(1166)$_8$

NO.	NAME
322R	Dump memory on alternate tapes; read back a selected dump.

ENTRY: The following calling sequence is required:

α	+ R add	α
α + 1	+ Tr	OF3
α + 2	+ R add	α
α + 3	+ Tr	OFO
α + 4	+ 00	FWA
α + 5	+ 31	LWA + 1
α + 6	Control automatically returns here if reading or writing occurs correctly.	

Writing takes place if entry is at α ; reading takes place if entry is at α + 2.

STARTING: Starting occurs automatically by entrance into the calling sequence.

DESCRIPTION: Writing: 322R will write a specified block of memory on logical tape #256, forming a check sum as it does so, and write this check sum on the tape. The tape is then read backward and a second check sum is formed, which is then compared with the one on the tape. If the check sums agree, no stop occurs, and control automatically goes to the coder's program. If the check sums disagree, a stop occurs, and the operator can try writing on the same tape by pushing the START button. If a second dump is taken, it will be written on logical tape #257, a third on tape #256 again, etc., thus giving the operator access to two dumps at all times. After writing on one tape, the other tape is rewound, thus leaving the tapes in visibly different status as follows:

DUMP	READ Lite	REWOUND Lite
n	ON	OFF
n-1	OFF	ON

After reading in a dump, writing will occur on the tape not read.

Reading: 322R has been revised for the greater convenience of the operator when memory is lost. Assume such a situation occurs; to read back a dump from tape, proceed as follows. Reload 322. If the last dump is desired, look at the tape units being used and note the number of the unit with the READ light on; set SENSE Switch #6 accordingly, and enter the calling sequence at $\alpha + 2$. If Sense Switch #6 is up, logical unit #256 will be read; if Sense Switch #6 is down, logical unit #257 will be read.

During the reading, a check sum is formed, and compared with the check sum on the tape. If they disagree, a stop occurs, and the operator can try reading from the same tape again by pushing the START button; the operator also has the option of changing the position of Sense Switch #6 and reading in a different dump. If the check sums agree, no stop occurs, and control automatically returns to the coder's program.

Check sum: The check sum used in this program is not the standard check sum; timing conditions preclude the possibility of using this. The check sum is formed by adding full words, and letting the overflow bits drop off.

WARNING: Do NOT write or read the section of memory containing 322 itself!

STOPS:

Location	Meaning
OF69	Check sums disagree; push START to try reading or writing again.

SWITCHES: Sense Switch #6:

Up:	Read back logical Tape #256.
Down:	Read back logical Tape #257.

STORAGE:
A0 - A2
B0 - B6
E0 - E5
F0 - F91

CODED: WJW. Checked & written, WJW.
February 17, 1955

322R:	Dump memory on alternate tapes; read back a selected dump.	322R	322A	322B	322C
STORAGE:	Decimal:	A0-	0150	2198	3734
		A2	0152	2200	3736
		B0-	0153	2201	3737
		B6	0159	2207	3743
		E0-	0000	2048	3584
		E5	0005	2053	3589
		F0-	0058	2106	3642
		F91	0149	2197	3733
	Octal:	A0-	0226	4226	7226
		A2	0230	4230	7230
		B0-	0231	4231	7231
		B6	0237	4237	7237
		E0-	0000	4000	7000
		E5	0005	4005	7005
		F0-	0072	4072	7072
		F91	0225	4225	7225
STOPS:	Check sums disagree. Push Start to try again.	F69	$(0177)_8$	$(4177)_8$	$(7177)_8$
SENSE ORDERS:		F74	$(0204)_8$	$(4204)_8$	$(7204)_8$
		F80	$(0212)_8$	$(4212)_8$	$(7212)_8$

NO.	NAME
323R	Two-bank tape dump program.

ENTRY: The following calling sequence is required:

α	+ R add	α
α + 1	+ Tr	OF8
α + 2	+ R add	α
α + 3	+ Tr	OFO
α + 4	+ 00	FWA
α + 5	+ 31	LWA + 1
α + 6	+ 00	FWA
α + 7	+ 31	LWA + 1
α + 8	Control automatically returns here if reading or writing occurs correctly.	

Writing takes place if entry is at α ; reading takes place if entry is at α + 2.

STARTING: Starting occurs automatically by entrance into the calling sequence.

DESCRIPTION: Writing: 323 will write two blocks of memory on logical tape unit #256, forming a check sum over both records as it does so, and write this check sum on the tape. The tape is then read backward and a second check sum formed for comparison. If the check sums agree, no stop occurs, and control goes to the coder's program. If the check sums disagree, a stop occurs, and the operator can try writing on the same unit again by simply pushing the START button. If a second dump is to be taken, it will be written on logical tape unit #257, a third on logical tape unit #256 again, etc., thus giving the operator access to two dumps. While one tape unit is being read backward to form the comparison check sum, the other tape unit is given a Rewind, thus leaving the tape units in visibly different status, as follows:

DUMP	READ LITE	REWOUND LITE
n	ON	OFF
n-1	OFF	ON

After reading a dump into memory, writing will occur on the tape unit not read.

Reading: To read back a particular dump, proceed as follows: (If memory is lost be sure 322 and calling sequence are read into the machine.) Select the tape number of the dump desired from the visible status of the tapes (see above), and set Sense Switch #6 Up for logical tape unit #256, Down for logical tape unit #257; enter the calling sequence at $\alpha + 2$. During the reading, a check sum is formed, and compared with the check sum written on the tape. If the check sums agree, no stop occurs and control goes to the coder's program. If the check sums disagree, a stop occurs, and the operator can try reading the same tape unit again by pushing the START button, or, by changing the position of Sense Switch #6 and pushing the START button, he can try reading the other tape unit.

Check sum: The check sum used in 323 is formed by simply adding full words and letting the overflow bits drop off. Timing conditions preclude the possiblity of using the standard check sum.

Warning: Do not write the section of memory containing 323 itself!

STOPS:

Location	Meaning
OF108	Check sums disagree; push START to try reading or writing again.

SWITCHES: Sense Switch #6:

Up:	Read back logical tape #256.
Down:	Read back logical tape #257.

STORAGE:
A0-A2
B0-B8
E0-E5
F0-F111

CODED: WJW. Checked & written, WJW.
February 17, 1955

323R: Two-bank tape dump program		323R	323A	323B	323C
STORAGE:	Decimal:	A0-	0170	2218	3754
		A2	0172	2220	3756
		B0-	0173	2221	3757
		B8	0181	2229	3765
		E0-	0000	2048	3584
		E5	0005	2053	3589
		F0-	0058	2106	3642
		F111	0169	2217	3753
	Octal:	A0-	0252	4252	7252
		A2	0254	4254	7254
		B0-	0255	4255	7255
		B8	0265	4265	7265
		E0-	0000	4000	7000
		E5	0005	4005	7005
		F0-	0072	4072	7072
		F111	0251	4251	7251
STOPS:	Check sums disagree. Push Start to try again.	F108	$(0246)_8$	$(4246)_8$	$(7246)_8$
SENSE ORDERS:		F68	$(0176)_8$	$(4176)_8$	$(7176)_8$
		F74	$(0204)_8$	$(4204)_8$	$(7204)_8$

400 Sin X

INPUT: The argument X, unscaled, must be put into the MQ.

Calling sequence:

A	RADD	A
A + 1	TR	OFO
A + 2	Return of Control	

LOADING: 400 may be loaded with 021, 023, or 024

DESCRIPTION: Sin X is taken by a series expansion and the result put in the MQ, unscaled. Only restriction is 1 x 1< 1. Accuracy to 34 bits has been checked.

STORAGE: OFO thru OF24

OAO thru OA2

OBO thru OB1

OEO thru OE4 OEO even

CODED: Don Monk, checked and written, Don Monk

401 Storage Check Sum

INPUT: Routine entered by calling sequence:

A	RADD	A
A + 1	TR	OFO
A + 2		FWA
A + 3		LWA

FWA = first full word to be summed,

FWA even.

LWA = last full word to be summed,

LWA even and $\geq$ FWA

-OEO = L(resultant sum σ)

LOADING: 401 may be loaded with 021, 023, or 024

DESCRIPTION: A storage check sum is taken from the first full word specified through the last full word specified. The check sum is the standard "super check sum". Every half word component, including sign, is considered as the right half word of a full word whose left half word component is -0, and two times such full words are added. Hence the sum, σ, is as follows:

$$\sigma = -2\left[\Sigma |u| + 2^{17} N(u)\right]$$

where u ranges over all half words from the first full word through the last full word, and N(u) is the number of negative half-words in that range.

STORAGE: OFO thru OF30

OAO thru OA2

OBO thru OB2 OBO even

OEO thru OE3 OEO even

37 regional cards

CODED: Don Monk, checked and written, Don Monk

402 Fixed Point e^x

INPUT: The argument must be put in the MQ. The calling sequence is

A	RADD	A
A + 1	TR	FO
A + 2	RETURN OF CONTROL	

LOADING: 402 may be loaded by 021, 023 or 024.

DESCRIPTION: The argument, x, must be less than $\log_e 2$ = .6931471806, and greater than -1. The result e^x is put in the MQ, scaled by 1/2.

STORAGE: A0 thru A2

B0 thru B1

F0 thru F19

E0 thru E5

25 regional cards

CODED: Don Monk

CHECKED-OUT: Don Monk

WRITTEN: Don Monk

403 Floating Point Addition

INPUT: The following must be prestored:

a	:	-E2	(preserved)
x	:	E0	"
b	:	-MQ	(c)
y	:	E1	(z)

Calling Sequence:

A	R ADD	A
A + 1	TR	F0
A + 2	(Return of control)	

LOADING: 403 may be loaded with 021, 024, or 023.

DESCRIPTION: Two floating binary numbers $A = a \cdot 2^x$ and $B = b \cdot 2^y$ are added to form $C = c \cdot 2^z$. a, b and c are less than 1 and are stored in full words. x, y and z are integral and are stored in half words with a scale factor 2^{-17}. Leading zeros are not shifted off. Overflow must be off.

OUTPUT: The result of the addition, $C = c \cdot 2^z$, is stored as follows:

c	:	-MQ
z	:	E1

STORAGE: 0F0 thru 0F28

0A0 thru 0A2 E0 thru E5

32 regional cards, 1 binary card

CODED: Ruth Scully

CHECKED-OUT: Don Monk

WRITTEN: Don Monk

H. Kolsky
T-5

409 R fixed point $\tan^{-1}$

INPUT: Calling sequence:

A	RADD	A
A + 1	TR	OFO

The argument must be put in the MQ, and must be $\leq .5$ for 35 bit accuracy.

LOADING: 409 may be loaded by 021, 023, or 024

DESCRIPTION: $\tan^{-1}x$ is evaluated by a series expansion. The result is found in the MQ, scaled by $\frac{1}{2}$.

STORAGE:

OFO thru OF29

OAO thru OA2

OBO thru OB1

OEO thru OE7   OEO even

35 regional cards, 1 binary card

CODED: Don Monk, checked out and written, Don Monk

H. Kolsky
T-5

410 R Integer Root

INPUT: The argument x, unscaled, must be put in the MQ.

Calling sequence:

α	+ RADD	α
$\alpha + 1$	TR	OFO
$\alpha + 2$	+ 00	n
$\alpha + 3$	Control returns here.	

LOADING: 410 R is in regional cards for 607.

DESCRIPTION: $y = \sqrt[n]{x}$, unscaled, is taken by iteration of the formula $y' = \frac{(n-1)y + x/y^{n-1}}{n}$ and put in the MQ and in - OEO. n must be an integer greater than zero, stored in location $\alpha + 2$ with a scale of 2^{-17}. x must be greater than zero and less than one.

Since the machine's accuracy is limited to 35 bits, it is advisable that n not be exceedingly large and that x not be very close to either 0 or 1. For very large values of n, better and faster results might be obtained by factoring n and applying this program with each of the factors.

STORAGE: OAO thru OA2
OFO thru OF43
OEO thru OE9 OEO even
47 regional cards

CODED: John Holladay, checked & written, John Holladay

411 R Sinh x

INPUT: The argument x, unscaled, must be put into the MQ.

Calling sequence:

A	R Add	A
A + 1	TR	OFO
A + 2	Control returns to here	

LOADING: Load 411 with 021, 023, or 024.

DESCRIPTION: The restriction on x is $|x|<1$. 1/2 sinh x is computed by a series expansion and the result stored in the MQ (also in E2). Note that sinh x is scaled by 1/2. The exit instruction is a break-point (-01; A + 2).

STORAGE:

OAO	thru	OA2	
OBO	thru	OB1	
OFO	thru	OF23	
OEO	thru	OE4,	EO even

29 Regional cards

CODED: Don Monk, checked out & written, Dot Monk

Correction on 413

UNDER STORAGE:

OFO Block should read

OFO thru OF48

413 Cube Root

INPUT: The argument must be put in the MQ.

Linkage entry:

A	RADD	A
A + 1	TR	FO
A + 2	Return of Control	

LOADING: 413 may be loaded by 021, 023, or 024

DESCRIPTION: $\sqrt[3]{X}$ is taken, the result being put in the MQ. Accuracy to 34 bits is obtained for up to five leading zeros in the argument. X must be positive, and less than 1.

STORAGE: 0A0 thru 0A2

0B0 thru 0B7 0B0 even

0F0 thru 0F47

0E0 thru 0E5 0E0 even

CODED: Don Monk, checked, Dot Monk, written, Don Monk

417 R Cos x

INPUT: The argument x, unscaled, must be put into the MQ.

Calling sequence:

A	R Add	A
A + 1	TR	OFO
A + 2	Return to control	

LOADING: 417 may be loaded with 021, 023, or 024.

DESCRIPTION: Cos x is computed by a series expansion and the result put in the MQ, unscaled. Only restriction is $0<|x|<1$. The exit instruction is a break-point (-01; A + 2). Accuracy has been checked to 10 decimal places.

STORAGE:

OA0	thru	OA2	
OB0	thru	OB1	
OE0	thru	OE4	OE0 even
OF0	thru	OF24	

CODED: Doris White, checked out & written, Doris White

A. Kolsky
T-5

426 R   Cosh x

INPUT: The argument x, unscaled, must be put into the MQ.

Calling sequence:

A	R Add	A
A + 1	TR	OFO
A + 2	Return of control	

LOADING: 426 may be loaded with 021, 023, or 024.

DESCRIPTION: Cosh x is computed by a series expansion and the result put in the MQ, scaled by 1/2. Only restriction is $|x|<1$. The exit instruction is a break-point (-01; A + 2). Accuracy to 10 decimal places has been checked.

STORAGE:

OAO	thru	OA2	
OBO	thru	OB1	
OEO	thru	OE4	OEO even
OFO	thru	OF23	

CODED: Doris White, checked out & written, Doris White

432 Double Precision Fixed Point e^x

INPUT: Calling Sequence:

A	RADD	A
A + 1	TR	F0

The argument must be prestored in full words -E6, -E8.

These full words must have the same sign. $-1 < x < .69315\ldots = \log_e 2$

LOADING: 432 may be loaded with 021, 023, or 024.

DESCRIPTION: e^x is evaluated by a series expansion. The result is stored in full words -E0, -E2, both with the sign of the result, and scaled by $\frac{1}{2}$. Accuracy has been checked to 18 decimal figures for four arguments, and to 19 figures for one argument.

STORAGE:

0F0	thru	0F59
0A0	thru	0A2
0B0	thru	0B1
0E0	thru	0E15

65 regional cards, 2 binary cards

CODED, CHECKED OUT, WRITTEN: Don Monk

H. Kolsky
T-5

NO.	NAME
450	Load an nth order symmetric matrix, check the matrix for symmetry, and load 451, $(2 \leq n \leq 31)$.

INPUT:

HEADING CARD

Card Col.	Punch in decimal
9	12
10 - 11	0
12 - 14	638
15 - 28	0
29	1

DATA CARD

Card Col.	Punch
9	Blank
10 - 14	2n as a 5 digit integer
15 - 44	0
46 - 80	0

add a 12 punch in cols. 14, 19, 24, 29, 34, 39, 44, 50, 55, 60, 65, 70, 75, 80.

HEADING CARD

Card Col.	Punch
9	11
10 - 12	009
13 - 19	0000008
20 - 24	00028
25 - 29	2n(n+1) as a 5 digit integer

DATA CARD

Card Col.	Punch
9	Blank
10 - 19	1st full word
20 - 29	2nd full word
30 - 39	3rd full word
40 - 44	4th full word (1st 5 digits)
45	Blank
46 - 50	4th full word (last 5 digits)
51 - 60	5th full word
61 - 70	6th full word
71 - 80	7th full word

Signs are punched over last digit of each word, an 11 for minus and a 12 for plus.

The matrix elements are in sort by row-major, column-minor, with an additional full word equal to zero as the last element of every row.

LOADING: 450 is self-loading into zero.

Loading Deck	# Cards
450	12 (1st card clears E.S. to zero and may be omitted)
Header	1
Data Card	1
Header	1
Data	least integer $\geq \frac{n(n+1)}{7}$
451	46

STOPS:

1. $(257)_8$ DPBC error L.H. sign, correct card.
2. $(265)_8$ DPBC error R.H. sign, correct card.
3. $(271)_8$ DPBC error L.H. digits, correct card.
4. $(275)_8$ DPBC error R.H. digits, correct card.
5. $(300)_8$ Header error, no header for 1st block, too many cards in a block, correct card.
6. $(426)_8$ DPBC error, Heading Card, correct card.
7. $(566)_8$ Input matrix is not symmetric as indicated by:
   L.H. Acc = $(\text{row index})_8$
   R.H. Acc = $(\text{col. index})_8$
   Push start to continue checking. Each error will then show up twice. When card reader starts feeding, reset E.S., correct cards, and reload.

STORAGE: $(0000)_8$ - $(0636)_8$

CODED: RvH, checked - RvH, 6/54.

NO.	NAME
451	Eigenvectors and eigenvalues of a real symmetric matrix of nth order ($2 \leq n \leq 31$).
INPUT:	See 450
LOADING:	451 deck follows cards being loaded by 450.
UNITS USED:	Card Reader; Printer; Drums 1, 2, and 3; Single Electrostatic Memory.
DESCRIPTION:	See output.
OUTPUT:	(Use Dual II board in printer with all alteration switches off.)

1) Components of eigenvectors printed out 6 per line.
2) Corresponding eigenvalue, one word per line.
3) Space.
4) Repeat 1), 2), 3) until n answers have been obtained.

STOP:

Location	Meaning
$(1256)_8$	Problem completed.

CODED: RvH, checked - RvH, 6/54.

H. Kobsky
T-5

NO. **NAME**

520 {Read / Write} full words {into / from} consecutive locations of ES-1 or ES-2 {from / onto} any drum.

INPUT: By basic linkage only: Basic linkage is described below.

Case 1. For a machine operating in frame #1 and in + Sense 32_{10} mode, the calling sequence from $\beta + 2$ through $\beta + 8$ is sufficient. When finished, 520 will relinquish control to $\beta + 8$.

Case 2. For a machine operating in frame #2, use the entire linkage.

In frame #2

α	+ R add	$\alpha + 2$	
$\alpha + 1$	+ Tr	β	
$\alpha + 2$	+ Tr	$\alpha + 3$	(+ if $\alpha + 3$ is in frame #1)
	- Tr	$\alpha + 3$	(- if $\alpha + 3$ is in frame #2)
$\alpha + 3$	control returns here		($\alpha + 3$ will normally contain a - Sense 32_{10} instruction.)

In frame #1

β	+ Sense	32_{10}	
$\beta + 1$	+ Store	$\beta + 8$	
$\beta + 2$	+ R add	$\beta + 2$	
$\beta + 3$	+ Read/Write	Drum #	decimal: 0128, 0129, 0130, 0131; octal: 0200, 0201, 0202, 0203
$\beta + 4$	+ Set drum	S.D.A. (even)	
$\beta + 5$	+ Tr	FO	
$\beta + 6$	- Copy	F.W.A.	even, if data is being written from/read to ES-1 . odd, if data is being written from/read to ES-2 .
$\beta + 7$	- Copy	L.W.A.	even, if data is being written from/read to ES-1 . odd, if data is being written from/read to ES-2 .
$\beta + 8$	[Case 1, control returns here / Case 2, leave open]		

S.D.A. is the drum location of the first full word to be {read into / written from} E.S.

F.W.A. is the location in E.S. of the first full word {read into / written from} E.S.

L.W.A. is the location in E.S. of the last full word {read into / written from} E.S., and must be numerically equal to or greater than F.W.A.

LOADING: Load 520 binary cards with 026 or 028

Loading deck	# Cards
026 or 028	1
520	2
Transition card for 026 or 028	(1 or 0)

STARTING: Starting by basic linkage occurs automatically

DESCRIPTION: Writing: Consecutive full words of ES-1 or ES-2, beginning at F.W.A. and ending with L.W.A., are written on the specified drum in full word locations starting at S.D.A. A check sum, the standard check sum, of the information written on the drum is formed and written on the drum in the location following the location of L.W.A. When this is done, 520 reforms a check sum using the data on the drum and compares it with the check sum that is on the drum.

At most, 2047 full words can be written on a drum, since 520 places a check sum in the position following the last word of information written on the drum. Also, all the information written on the drum at this time must be read off the drum later to get check sum agreements. The coder should not allow 520 to write itself on the drum, (complicates check sums.)

Reading: Information on the drum beginning at the full word location S.D.A. is read from the specified drum into ES-1 or ES-2 in locations F.W.A. through L.W.A. The standard check sum is formed and compared with the check sum that is on the drum. The check sum is not stored in memory.

PROGRAM STOPS:

Location	Meaning
F67	Check sums disagree. Press the start button to try to write or read again.

STORAGE: E.S. Storage

A0 through A2
E0 through E9, E0 is even
F0 through F68

Drum Storage

S.D.A. through L.W.A. - F.W.A. + 2 full words

CODED: D. E. Harris, checked & written, D. E. Harris

520 (Read / Write) full words (into / from) consecutive E.S. locations in bank 1 or bank 2 (from / onto) any drum.

		R	A	B	C
INPUT: β + 5 contains transfer to		F0	61_{10} or 75_8	2109_{10} or 4075_8	3645_{10} or 7075_8
STARTING: Entry by unconditional transfer		F0	61_{10} or 75_8	2109_{10} or 4075_8	3645_{10} or 7075_8
PROGRAM STOPS: Check sums do not agree. Press start button to try reading or writing again.		F67	128_{10} or 200_8	2176_{10} or 4200_8	3712_{10} or 7200_8
STORAGE: decimal		A0-	58_{10}	2106_{10}	3642_{10}
	thru	A2	60_{10}	2108_{10}	3644_{10}
		E0	0_{10}	2048_{10}	3584_{10}
	thru	E9	9_{10}	2057_{10}	3593_{10}
		F0	61_{10}	2109_{10}	3645_{10}
	thru	F68	129_{10}	2177_{10}	3713_{10}
octal		A0-	0072_8	4072_8	7072_8
	thru	A2	0074_8	4074_8	7074_8
		E0-	0000_8	4000_8	7000_8
	thru	E9	0011_8	4011_8	7011_8
		F0-	0075_8	4075_8	7075_8
	thru	F68	0201_8	4201_8	7201_8

NO. **NAME**

525R {Read / Write}* full words {into / from} consecutive E.S. locations {from / on} any drum.

INPUT: Punch control card in binary as follows, 9 row:

columns 10 thru 14:	11,000 (read) or
columns 10 thru 14:	11,010 (write)
columns 19 thru 26:	drum no. 10,000,000 = 128, 10,000,001 129, 10,000,010 130, or 10,000,011 131.
columns 33 thru 44:	SDA = set drum address = the location of the first full word to be {read into / written from} E.S. If the SDA is odd, the 701 will interpret this as being the next lowest even integer; therefore only full words can be written on a drum.
columns 51 thru 62:	FWA = first word address = the location in E.S. of the first full word to be {read into / written from} E.S. The FWA must be even.
columns 69 thru 80:	LWA = last word address = the location in E.S. of the last full word to be {read into / written from} E.S. The LWA must be even.
8 row, columns 10 thru 26:	Exit instruction to be executed immediately after {reading / writing}. Note that both operation and address parts must be specified (the instruction may be + or -).

* Read upper line in brackets for directions for reading from drum with 525, read lower line for directions for writing.

Leave the rest of the card blank.

Calling sequence for entry by linkage is as follows:

α	RADD	α
$\alpha + 1$	TR	F11
$\alpha + 2$	{Read / Write}	DRUM NO.
$\alpha + 3$	0	SDA (See restrictions above)
$\alpha + 4$	0	FWA (See restrictions above)
$\alpha + 5$	0	LWA (See restrictions above)
$\alpha + 6$	Control automatically returns to here.	

When 525 is being used for reading from a drum into E.S., a check sum, σ, defined as usual, must be stored on the drum in the full word location following the last full word to be stored in E.S. The full word drum locations SDA thru SDA + (LWA-FWA) contain the full words to be stored in E.S. The location of σ is SDA + (LWA-FWA) + 2.

LOADING: Load 525 binary cards with 021. See 021 for complete loading instructions.

Loading Deck	# cards
021	1
525	4
Transition to 525	1
525 control card	1

STARTING:

a. Automatic start with control card. Reset console by pressing reset button. Set instruction entry keys for 021, automatic-manual switch to automatic, put loading deck in hopper and press card-reader start, then the load button. Feed out cards when card-reader select light goes out.

b. Manual entry with control card. (When 525 is already in E.S.) Press reset. Put control card in hopper and have card-reader ready. Start 701 manually at FO. Feed out control card after the select light on the card-reader goes out.

c. Start by linkage occurs automatically (see INPUT).

DESCRIPTION:

Reading: Full words starting at drum location SDA are read into E.S. full word locations FWA thru LWA. This information is summed in E.S. and check made to see that this E.S. sum agrees with the drum sum, σ, which is stored in drum location SDA + LWA - FWA + 2. No check sum is kept in E.S.

Writing: Full words in E.S. locations FWA thru LWA are written on the specified drum in full word locations starting with SDA. A memory sum, σ, of the information in FWA thru LWA of E.S., is written on the drum in full word location SDA + LWA - FWA + 2. When writing is finished a check sum of the information just written is taken and compared with the check sum in drum location SDA + LWA - FWA + 2.

PROGRAM STOPS:

Regional Location	Meaning
F62	Check sums do not agree. If reading, check sum on drum may be in error. Press start to read or write again. If error keeps repeating, reload and start over or call 701 dispatcher.
F10 (occurs if no exit instruction is punched on control card)	{Reading / Writing} is finished and check sums agree, and 701 is prepared to read another control card.

OUTPUT: When writing, full words on specified drum locations starting with SDA and σ on drum location SDA + LWA - FWA + 2. When reading, full words in E.S. FWA thru LWA from the specified drum, the first full word from SDA, second from SDA + 2, etc.

RESTARTING: If 525 is already in E.S., start as before (see STARTING b or c).

STORAGE: Regional A0 thru A2

E0 thru E11, E0 even

F0 thru F91, F0 even

total = 107 half-words

For SO_2 assembly, origins A0, E0, and F0 must be specified.

Drum storage occupied

SDA thru SDA + LWA - FWA + 2
525 is 95 regional cards
3 or 4 binary cards

CODED: WGB, ch'd - dtm, written - dtm

525 {Read / Write} full words {into / from} consecutive E.S. locations {from / on} any drum.

		525R	525A	525B	525C
INPUT: α + 1 contains transfer to ...		F11	67_{10}	2115_{10}	3651_{10}
STARTING: For automatic entry, set instruction entry keys to ...			0	4000_8	7000_8
For manual entry, start 701 at ...		F0	70_8	4070_8	7070_8
PROGRAM STOPS: Check sums do not agree ...		F62	166_8	4166_8	7166_8
{Reading / Writing} finished.		F10	102_8	4102_8	7102_8
STORAGE:	decimal	A0-	53-	2101-	3637-
	thru	A2	55	2103	3639
		F0-	56-	2104-	3640-
	thru	F91	147	2195	3731
		E0-	2-	2048-	3584-
	thru	E11	13	2059	3595
	octal	A0-	(65-	(4065-	(7065-
	thru	A2	$67)_8$	$4067)_8$	$7067)_8$
		F0-	(70-	(4070-	(7070-
	thru	F91	$223)_8$	$4223)_8$	$7223)_8$
		E0-	(2-	(4000-	(7000-
	thru	E11	$15)_8$	$4013)_8$	$7013)_8$

NO.	NAME
526	Write all of e.s. on drum #1 with the exception of full words -0000 and -0002. (Not Regional)

INPUT: Four self-loading binary cards. Program will read 021A, 024A, FEJ35, etc. after transfer of e.s. to drum is complete if desired.

LOADING DECK	# CARDS
526	4
Any A region self-loading card or deck	1 (or n)
Binary, octal or dec. deck	m

Note: Although all of e.s. is now available and A region self-loading should be used after 526, one may use 526 by itself and proceed as usual with a self-loading program not in A region, provided the 526 cards are fed out of card reader after 526 turns on the Copy Check Light.

STARTING: Put loading deck in card reader and depress start button on card reader until Ready Light comes on. Set instruction keys to 0000, automatic-manual switch to automatic, and press load button. Press card reader start when 701 stops on last card.

DESCRIPTION: 526 writes all of e.s. on drum #1 with the exception of the first two full words in e.s. The drum location for a given word is 4 less than that of its e.s. location. For example, the e.s. full word located in -0004 has the drum location of -0000, etc. No check sums are taken.

STORAGE: 0000 - 0003 DESTROYED BY 526.

0004 - 0019 USED BY 526 AFTER DRUM WRITING.

CODED: E. A. Voorhees, checked and written, E. A. Voorhees

H. Kolsky
T-5

NO.	NAME
527 R	{Read / Write} full words {into / from} consecutive E.S. locations {from / on} any drum.

527 replaces 525 if you want to enter the drum program by basic linkage as it is a shorter program. However, if you want to enter the drum program by using a control card, you have to use 525.

INPUT: By basic linkage only: Calling sequence is as follows:

α	+	R add	α	
$\alpha + 1$	+	Read/Write	Dr#	
$\alpha + 2$	$\pm$	Set Drum	S.D.A.	(even)
$\alpha + 3$	$\pm$	Tr	FO	
$\alpha + 4$	-	Copy	F.W.A.	(even)
$\alpha + 5$	-	Copy	L.W.A. + 1	(odd)

$\alpha + 6$ Control returns here automatically; S.D.A. is the drum location of the first full word to be {read into / written from} E.S., F.W.A. is the location in E.S. of the first full word {read into / written from} E.S. and LWA is the location in E.S. of the last full word {read into / written from} E.S.

Since the program does not {write / read} cyclically {from / into} E.S.,

L.W.A. $\geq$ F.W.A.

LOADING: Load 527 binary cards with 026

Loading Deck	# Cards
026, etc.	1
527	2

STARTING: Starting by basic linkage occurs automatically.

DESCRIPTION: Writing: Consecutive full words of E.S., beginning at F.W.A. and ending with L.W.A., are written on the specified drum in the full word locations starting with S.D.A. A check sum, the standard check sum, of the information written on the drum is formed and written on the drum in the location following the location of L.W.A. After the writing on the drum is completed, the information is read back into E.S. where a check sum is again formed and compared with the check sum read off the drum.

Reading: Information on the drum beginning at the full word location S.D.A. is read from the specified drum into E.S. locations F.W.A. to L.W.A. inclusive. The standard check sum is formed and the data read into E.S. and compared with the check sum written on the drum. This program will not read itself from the drum over itself in E.S.

PROGRAM STOPS:

Location	Meaning
F62	Check sums disagree. Press the start button to try to read or write again.

STORAGE: E.S. Storage:

A0 through A2

E0 through E9, E0 is even.

F0 through F63.

Drum Storage:

S.D.A. and L.W.A. - F.W.A. + 2.

full words starting at S.D.A.

CODED: T.L.J., ch'd T.L.J., written T.L.J.

527 {Read / Write} full words {into / from} consecutive E.S. locations {from / onto} any drum.

		R	A	B	C
INPUT:	α + 3 contains transfer to ---	F0	55_{10}	2103_{10}	3639_{10}
STARTING:	Entry by unconditional transfer	F0	55_{10}	2103_{10}	3639
PROGRAM STOPS:	Check sums do not agree. Press start button to try reading or writing again.	F62	117_{10}	2165_{10}	3701_{10}
STORAGE:	decimal	A0-	52	2100	3636
	thru	A2	54	2102	3638
		E0-	0	2048	3584
	thru	E9	9	2057	3593
		F0	55	2103	3639
	thru	F63	118	2166	3702
	octal	A0-	(64	4064_8	7064_8
	thru	A2	$66)_8$	4066_8	7066_8
		E0-	(0-	4000_8	7000_8
	thru	E9	$11)_8$	4011_8	7011_8
		F0-	$(67_8$	4067_8	7067_8
	thru	F63	$166)_8$	4166_8	7166_8

H. Kolsky
T-5

REGIONAL ASSEMBLY PROGRAM, 607

FUNCTIONS OF 607

Regional Assembly Program, 607, will perform the following functions:

1. Assign absolute locations and addresses to a regional program.
2. Expand or contract a regional program, and if expansion, insert new orders consecutively in the program.
3. Change regional indices.
4. Convert a twelve digit fractional number in columns 45-57, scale according to the decimal and binary factors specified in columns 58-61, enter as either half-word or full word, and assemble.
5. Print the original regional information and comments on the card, the final regional indices, location, operation, and address in decimal, and the final location, operation, and address in octal.
6. Punch binary cards for loading with 021, FEJ035, LCH21 or allied programs.
7. Punch regional binary cards for loading with 025 or allied programs.
8. Punch decimal regional cards, with the changed regional information, and the original comments, (only one of the three punch programs may be selected during an assembly, but any or all of the other functions may be performed).

CARD LAYOUT

Regional information and control punches are punched in columns 9-26, comments or constants are punched in columns 45-64, alphabetic abbreviation of the operation or further comments are punched in columns 65-72.

CARD LAYOUT (contd.)

The control punches in column 9 are the digits 0, 1, 2, 3, or 4 only.

The location index in columns 10-12 consists of a two digit number in the range 00-99 followed by an alphabetic punch, A to R.

The relative location in columns 13-16 consists of a four digit number in the range 0000-4095.

Column 17 contains only an x punch or a y punch.

The operation in columns 18-19 consists of a two digit number in the range 00-31.

The address index in columns 20-22 is of the same form as the location index.

The relative address in columns 23-26 is of the same form as the relative location.

There must be one and only one digital punch in columns 9-16 and 18-26. There must be either an x or a y punch in columns 12, 17, and 22. Columns 45-72, if used for comments may contain any punching desired, or they may be blank. If columns 45-61 are used for entry of constants, there must be an x or y punch in column 45 and a digital punch only in columns 46-61. The remaining columns 62-72 may contain any punching desired.

INPUT

There are three types of input to 607. The first consists of information for the control of assembly of the regional program, (digital punches 1, 2, or 3 in column 9). The second type of input consists of the program itself, either instructions (digital punch 0 in column 9) or constants (digital punch 4 in column 9). The third type of input consists of the six Sense Switches on the console. These switches control the printing and punching of the output information.

ASSEMBLY CONTROL

There are three types of control cards for the input of information for assembly of a program. They are distinguished by the control punch in column 9 as follows:

Column 9	
1	Absolute Location. Type #1 card
2	Expansion or Contraction. Type #2 card
3	Index Change. Type #3 card

The cards are punched as follows:

Column	Content
9	#1, 2, or 3 control punch.
10-12	Cut index.
13-16	Cut address.
17	Sign of increment, plus : y, minus : x.
18-21	Zeros.
22	Alphabetic punch, R.
23-26	Increment.

In the case of type #1 or #2 cards, the increment in columns 17, 23-26 is added to all relative locations or addresses which have the cut index specified in columns 10-12 if the relative location or address is greater than or equal to the cut address specified in columns 13-16. In the case of type #3 cards, the increment is added to all location or address indices which have the cut index specified if the relative location or address is greater than or equal to the cut address.

The original index, location, or address is replaced by the sum of the original plus the increment, if the increment is added. The original index, location, or address is then no longer available.

ASSEMBLY CONTROL (contd.)

The control cards are entered into a block of electrostatic storage, which can contain up to 200 control cards. They are entered into successive positions in this block of storage in the order in which they are placed in the hopper of the card reader. During assembly this block of storage is searched from the first to the last control card entered. Any assembly operations are therefore performed in the order in which the control cards were entered into the machine.

Ordinarily all type #1, #2, and #3 cards are entered before any program cards, but they may be entered at any time before they are needed, i.e. in front of a particular program card or cards which need these controls for proper assembly.

There must be a #1 control card for each index used in the program, except for the indices containing the letter I or R, and except when decimal punching is selected, in which case the #1 cards are optional.

INSTRUCTION INPUT

Instruction cards are punched as follows:

Column	Content
9	#0 control punch.
10-12	Location index.
13-16	Relative location.
17	Sign of instruction, plus : y, minus : x.
18-19	Operation.
20-22	Address index.
23-26	Relative address.
45-64	Comments about the instruction.
65-72	Operation word.

CONSTANT INPUT

There are two types of constant input; half-word constants and full word constants. They are distinguished by the punch in column 17. The card is punched as follows:

Column	Content
9	#4 control punch
10-12	Location index.
13-16	Relative location.
17	Half-word : y, Full word : x.
18-21	Zeros.
22	Letter R.
23-26	Zeros.
45	Sign of constant, plus : y, minus : x.
46-57	Constant.
58-59	Decimal scaling factor.
60-61	Binary scaling factor.
62-72	Any punching desired.

The decimal point is considered between columns 45 and 46. The binary point in the machine is considered at the left. The decimal scaling factor in columns 58-59 specifies how many places the decimal point is shifted to the right. The binary scaling factor in columns 60-61 specifies how many places the binary point is shifted to the right in the machine.

It will be noted that the same form in columns 45-61 is used for both half and full words. In every case the machine converts all twelve digits and scales them. Then half-words are rounded to 17 bits and full words are rounded to 35 bits.

After rounding, the location is assembled. Card reading is then resumed in the case of half-words. In the case of full words, the final

CONSTANT INPUT (contd.)

location is increased by one, a new card image is formed and the second half-word entered. Then card reading is resumed.

The range of the decimal scaling is 00-11, and the range of binary scaling is 00-35. Caution must be observed that the binary scaling is great enough to hold the integers specified in the decimal scaling. For example: The integer, 13, would have the decimal scaling factor 02, the binary scaling factor must be 04 or greater.

The following example shows several constants. It will be noted that half-word constants, full word constants, or half-word instructions may be intermixed as the coder chooses. It will also be noted that only one card is entered for full word constants, but that the machine manufactures the second card image to accommodate the second half-word.

Columns 9-26	Columns 45-57	58-61	Constant
415C0010 - 0000R0000	+314159265359	0102	Full word π.
015C0012 + 0000R1000			Half word 1,000.
415C0013 + 0000R0000	+150000000000	0517	Half word 15,000.
415C0014 - 0000R0000	-000200000000	0000	Full word $-2\cdot10^{-4}$.
415C0016 - 0000R0000	+513141592654	0207	Dual full word π.

The coder must make sure that the absolute location of a full word constant starts with an even number.

It will be noted in the above example that it is easier to enter the constant 1,000 on a type #0 card, than to go to the extra trouble of filling out columns 45-61. This is true of all integers less than 4096. In the case of 15,000, it is easier to enter on a type #4 card than to convert it to a decimal instruction for entry on a type #0 card. It will also be noted that the scaling for all "Dual constants" is 0207.

CARD ASSEMBLY

There are two main portions of program assembly. The first is the assembly of individual cards and their storage in print storage. The second is the punching or printing when print storage is full.

The first operation performed is the reading of cards. A card is read and the regional information and constant values are converted to binary between COPY orders. The regional information is then checked for inconsistencies such as relative locations or addresses over 4095 or indices containing letters S to Z. The card is also checked for double digital punching or blank columns in the case of a misread or mispunched card.

If the card is a type #1, #2, or #3, control transfers to enter the information in storage for assembly control. The information entered is added to the sum of the block for checking purposes, then card reading is resumed.

If the card is a type #0 card, control transfers to the assembly program. The block of storage containing the type #1, #2, and #3 cards is then searched and the assembly performed. After assembly, the final indices, location, and address are checked for inconsistencies such as an address which is negative or over 4095. The location is checked to see that it is in consecutive order with the preceding card entered. The card image is then transferred to print storage, and card reading resumed. If non-consecutive locations are encountered, or the block of print storage is full, control is transferred to the print - punch programs. After the completion of the print - punch cycle, the last card assembled is transferred to print storage and card reading resumed.

If the card is a type #4 card, control transfers to check the constant for double punching or blank columns, and to see that the

CARD ASSEMBLY (contd.)

decimal and binary scaling factors are in range. The constant is then scaled and checked for scaling and rounding overflow. The card is then treated as a type #0 card, and control transfers to the assembly program. In the case of full word constants, the second card image is formed and transferred to print storage before card reading is resumed.

All assembly work is done after the 12 Right COPY of one card and before the next "Read Card Reader" instruction is given. The 701 Manual allows 70 milliseconds for the assembly operations. The time involved for the relative location and address to be compared with each value in the control card table is 0.732 ms. Theoretically, therefore, the card reader should start reading at half speed when about 95 control cards have been entered in the table. In actual practice, 150 control cards were entered in the table and the card reader was still operating at full speed.

PUNCHING AND PRINTING

Up to 44 cards will be entered into print storage before the print-punch cycle begins (up to 43 cards if Sense Switch #5 is Down). This corresponds to the number of half-words punched on a binary card, (or a regional binary card). After card reading is stopped, but before the print - punch cycle starts, the block of storage containing the control cards is summed and compared with the original sum. If they disagree, the machine stops.

If Sense Switch #2, #4, and #5 are up, a binary card will be punched in the form for loading with 021 or allied loading programs. After punching the half-words are again summed and compared with the original check sum. If these disagree, the machine stops. Pushing the "Start"

PUNCHING AND PRINTING (contd.)

button will repunch the card. If the card is correct, control is transferred to the print program.

If Sense Switch #4 is up and #5 is down, a regional binary card will be punched in the form for loading with 025 or allied programs. After punching, the half-words are summed and compared with the original check sum. The machine behavior is as described above for binary punching.

If Sense Switch #1 is up, the contents of print storage will be printed. The contents of columns 10-26 and 45-72 of the original card read into the machine have been preserved in print storage and print directly. The final location index, location, instruction sign, operation, address index, and address are converted to decimal and entered in the card image. The final location, operation and address are also converted to octal and entered in the card image. The information prints from left to right on the printed page as follows:

1. The original location index, relative location, instruction sign, operation, address index, and relative address (columns 10-26 of the original card).
2. The comments (columns 45-64 of the original card).
3. The final location index, location, instruction sign and operation in decimal.
4. The operation word (columns 65-72 of the original card).
5. The final address index and address in decimal.
6. The final location, instruction sign, operation, and address in octal.

In the case of type #4 cards specifying full words, the second line printed will contain only the information of 3, 5, and 6 above.

PUNCHING AND PRINTING (contd.)

After printing each line, the card image is transferred back to print storage and the next location index compared with the one printed. If they differ, the paper double spaces before the next line printed.

If Sense Switch #6 is up, the paper will restore to the next sheet of paper after the print cycle is completed. This will give a one to one correspondence between binary punched cards and printed pages. If Sense Switch #6 is down, the paper will only restore on overflow, (after printing 62 lines).

If Sense Switch #4 is down and #5 is up, decimal regional cards will be punched after the print cycle is completed. (If Sense Switch #1 is down, there will be no print cycle, but since the print cycle is used to do the decimal conversion, that cycle takes place except for the "Write Printer" and "Copy" instructions. The time involved is about two seconds.) The decimal punch program rearranges the final regional information and puts it in the card image so that it punches in columns 10-26 of the card. 0 is always punched in column 9. The contents of column 45-72 of the original card read by the card reader are preserved and punched in columns 45-72. The only output of this program is type #0 cards. Original type #4 cards read into the machine will punch as one or two type #0 cards.

INDEX CHANGES

Any location or address index not containing the letter I or R may be changed to any other permissible index, which may include the letter I or R. This part of the assembly work is controlled by a type #3 card. The index is converted to binary and stored in the machine as follows:

The two digital punches plus 256 times the digit in the letter part,

INDEX CHANGES (contd.)

plus 128 if the letter contains an x punch, or plus zero if the letter contains a y punch.

For example:

$27F = 27^{y}_{6} = 27 + 256\cdot 6 + 0 = 1563$

$13N = 13^{x}_{5} = 13 + 256\cdot 5 + 128 = 1421$

To change an index, the increment in columns 17, 23-26 of a type #3 card must be the difference between the cut index and the new index desired. For example:

To change 27F (= 1563) to 14G (= 1806), the increment should be +243.

To change 10H (= 2058) to 15P (= 1935), the increment should be -123.

To change 6F (= 1542) to OR (= 2432), the increment should be +890.

PROGRAM CORRECTION

There are two main types of correction necessary. The first is the correction of errors on one or more cards, not necessitating the insertion or deletion of program cards. The second is expansion or contraction of a program to form gaps or to close up gaps in the coding. Expansion or contraction implies that new instruction cards are entered to fill the gap, or that old instruction cards are removed to form a gap.

PROGRAM CARD CORRECTIONS

607 is designed so that each printed page corresponds to a punched binary card. Correct the necessary cards in the block of program cards corresponding to the proper printed page. Reassemble that block of cards. This will give a corrected listing and binary card. The incorrect listing and card may then be discarded and the correct page and card entered in their place.

INSERTION OR DELETION

Type #2 cards will form gaps (positive increment) or close up gaps (negative increment) in a program. If the coder desires the removal of certain instructions, he removes the cards from his deck, punches the proper type #2 card and reassembles his deck. See example below. If the coder desires the addition of certain instructions, he punches these cards, inserts them in order in his deck, punches the proper type #2 card, and reassembles his deck.

In the following example, the original deck on the left (only the location part is shown) is to be changed so that three cards are inserted after 27F6, two cards are to be deleted (27F10 and 27F11) and another card is to be inserted following 27F12. The left-hand column shows the original program, and the right-hand column shows the order of the changed program. If addresses referring to the new indices are coded with the new index, the changes are properly made.

27F6	27F6
27F7	30F7
27F8	30F8
27F9	30F9
27F10	27F7
27F11	27F8
27F12	27F9
27F13	27F12
	30F13
	27F13

The control cards for this operation are shown below in the order in which they should be entered preceding the corrected program cards.

1. #2 27F13 +1
2. #3 30F13 -3
3. #2 27F10 -2

INSERTION OR DELETION (contd.)

4. #2 27F7 +3

5. #3 30F7 -3

6. #1 27F (coders location for the 27F block)

It will be noted that all assembly work is done in the order in which the control cards are entered in the machine. It will also be noted that a different index is used for the insertion cards. This normally is an index not originally used by the coder. The following table shows the assembly of the cards as they search the control table. To the left is the original, and each succeeding column to the right shows the changes performed as that card searches the control table. The column on the right shows the final assembly, before the absolute location is added.

	1	2	3	4	5	
27F6	---	---	---	---	---	27F6
30F7	---	---	---	---	27F7	27F7
30F8	---	---	---	---	27F8	27F8
30F9	---	---	---	---	27F9	27F9
27F7	---	---	---	27F10	---	27F10
27F8	---	---	---	27F11	---	27F11
27F9	---	---	---	27F12	---	27F12
27F12	---	---	27F10	27F13	---	27F13
30F13	---	27F13	27F11	27F14	---	27F14
27F13	27F14	---	27F12	27F15	---	27F15

This is a suggested method of doing insertions. The coder may design any other scheme he pleases.

SENSE SWITCHES

#1

Up: Print.

Down: Transfer over the print cycle, except if #2 is also down, then print anyway.

#2

Up: Transfer to the binary punch cycle.

Down: Transfer over the binary punch cycle.

#3

Up: No function.

Down: Stop if non-consecutive locations are encountered. This gives the operator an opportunity to run out the cards and arrange them in order if consecutive locations should not have occurred. If Sense Switch #6 is up, take the first card following the last one on the printed page, put the cards in proper order with all #1, 2, or 3 controls in front, have the card reader "Ready", manually enter the address $(0334)_8$ on the instruction entry keys, and press "Start". This will avoid duplication or omission of printing and binary half-words in the binary cards. If the non-consecutive location should have occurred, press "Start" to continue the assembly as if no "Stop" had occurred.

#4

Up: No function.

Down: Punch decimal regional cards. In this case, 44 cards will always be read between printing and punching cycles, non-consecutive locations are ignored. Also the entry of type #1 cards for absolute location of the program is optional.

#5

Up: No function.

Down: Transfer to the regional binary punch program.
Note: This switch must be down before 607 is read into the machine.

#6

Up: Allow paper restoring between printed blocks of information.

Down: Allow paper restoring only on overflow (after printing 62 lines).

STOPS	All stop addresses are given in octal.
0220	The table containing the control cards is full. Start over.
0252	The cut index of this type #1, #2, or #3 card contains the letter I or R. Correct, reload, push "Start".
0253	The decimal scaling factor on this type #4 card is greater than 11. Correct, reload, push "Start".
0254	The binary scaling factor on this type #4 card is greater than 35. Correct, reload, push "Start".
0255	There is no type #1 control card in storage for this card's location. Place proper type #1 card in front of this card, reload, push "Start".
0256	There is no type #1 control card in storage for this card's address. Place proper type #1 card in front of this card, reload, push "Start".
0377	Assembly is complete. If "Start" is pushed, the machine will act as if 607 were just loaded in the machine and the transition card had just been read.
0424	This "Stop" indicates a machine error. It is caused by the "End of Record" skip during card reading. It should never occur.
0654	Mispunched or misread card. The correct card sum is in the MQ, the card sum for this card is in the Acc. Columns 1-17 of the Acc and MQ correspond to the sum of the x and y row in columns 10-26. Columns 18-35 of the Acc and MQ correspond to the sum of the 9 thru 0 rows in columns 9-26. Correct, reload, push "Start".
0660	The control punch in column 9 of this card is not 0, 1, 2, 3, or 4. Correct, reload, push "Start".
0661	Improper original location index. Correct, reload, push "Start".
0662	Improper original relative location. Correct, reload, push "Start".
0663	Improper original operation. Correct, reload, push "Start".
0664	Improper original address index. Correct, reload, push "Start".

STOPS All stop addresses are given in octal.

1065 Improper scaling on this type #4 card. If there are no bits in the overflow positions of the Acc, the binary scaling factor was not large enough to accommodate the decimal integers. If there are bits in the overflow position, they were caused by rounding. Correct, reload, push "Start".

1317 The final location is negative or greater than 4095. The Acc will show a minus sign or the location minus 4096 respectively. If the trouble is with this card, correct, reload, push "Start" If the trouble is with the control cards (type #1, #2, or #3), correct control cards, reload 607, control cards, and proper program cards so that duplication or omission of printing and binary half-words, punched in the binary cards, will be avoided.

1324 The final address is negative or greater than 4095. See "Stop" 1317 for correction procedure.

1334 The final location index is improper. See "Stop" 1317 for correction procedure.

1361 The final address index is improper. See "Stop" 1317 for correction procedure.

1431 This stop only occurs if Sense Switch #3 is Down. See Sense Switch #3.

1500 The sum of the block of storage containing the control cards does not agree with the original sum. A memory dump with 186 should show the trouble. The block of storage is located from 2542_8 to 3672_8. In any case reload 607, all control cards, and the proper program cards to insure no duplication or omission of printing or binary half-words in the punched cards.

1617 The check sum following the punching of a binary card does not agree with the check sum punched in the card. The original check sum is in the Acc; the second check sum is in the MQ. Push "Start" to repunch the card.

2002 The check sum following the punching of a regional binary card does not agree with the check sum punched in the card. The original check sum is in the Acc; the second check sum is in the MQ. Push "Start" to repunch the card.

2534 The decimal regional card just punched has a blank column or double punch. Push "Start" to repunch card. See "Stop" 0654 for explanation of contents of Acc and MQ.

Regional Assembly Program, 608

Regional Assembly Program, 608, performs the same operations as 607 except the regional decimal punching is not allowed, and two new control cards have been added.

The two new control cards are designated by a 5 or a 6 punch in column 9. Both the type 5 and type 6 control cards have 00R0000 + 00 00R0000 punched in columns 10-26.

A type 5 control card will perform the following functions:

1. Punch and print any type 0 or type 4 cards which have been assembled, but not punched or printed preceding the type 5 control card.

2. Modify addresses such that all succeeding binary or regional binary cards are punched with a +R.

3. Cause the printing of ES-1 in the upper right-hand corner of each succeeding listed page.

4. Continue assembly on succeeding cards.

A type 6 control card will perform the following functions:

1. Punch and print any type 0 or type 4 cards which have been assembled, but not punched or printed preceding the type 6 control card.

2. Modify addresses such that all succeeding binary or regional binary cards are punched with a -R.

3. Cause the printing of ES-2 in the upper right-hand corner of each succeeding listed page.

4. Continue assembly on succeeding cards.

After the transition card is read following the loading of 608, or if "Start" is pressed following the Completed Assembly Stop, 0377, 608 will act as if a type 5 control card has just been read.

Written: Dura W. Sweeney, 3/22/54.

SENSE SWITCHES

#1 Up: Print.

Down: Transfer over the print cycle, except if #2 is also down, then print anyway.

#2 Up: Transfer to the binary punch cycle.

Down: Transfer over the binary punch cycle.

#3 Up: No function.

Down: Stop if non-consecutive locations are encountered. This gives the operator an opportunity to run out the cards and arrange them in order if consecutive locations should not have occurred. If Sense Switch #6 is up, take the first card following the last one on the printed page, put the cards in proper order with all #1, 2, or 3 controls in front, have the card reader "Ready", manually enter the address $(0334)_8$ on the instruction entry keys, and press "Start". This will avoid duplication or omission of printing and binary half-words in the binary cards. If the non-consecutive location should have occurred, press "Start" to continue the assembly as if no "Stop" had occurred.

#4 Up: No function.

Down: Causes a "Stop" at $(1503)_8$. Use 607 for regional decimal punching. Press "Start" to continue as if no "Stop" had occurred.

#5 Up: No function.

Down: Transfer to the regional binary punch program.
Note: <u>This switch must be down before 607 is read into the machine, or before "Start" is pressed following Completed Assembly Stop, 0377.</u>

#6 Up: Allow paper restoring between printed blocks of information.

Down: Allow paper restoring only on overflow (after printing 62 lines).

STOPS	All stop addresses are given in octal.
0220	The table containing the control cards is full. Start over.
0252	The cut index of this type #1, #2, or #3 card contains the letter I or R. Correct, reload, push "Start".
0253	The decimal scaling factor on this type #4 card is greater than 11. Correct, reload, push "Start".
0254	The binary scaling factor on this type #4 card is greater than 35. Correct, reload, push "Start".
0377	Assembly is complete. If "Start" is pushed, the machine will act as if 607 were just loaded in the machine and the transition card had just been read.
0424	This "Stop" indicates a machine error. It is caused by the "End of Record" skip during card reading. It should never occur.
0654	Mispunched or misread card. The correct card sum is in the MQ, the card sum for this card is in the Acc. Columns 1-17 of the Acc and MQ correspond to the sum of the x and y row in columns 10-26. Columns 18-35 of the Acc and MQ correspond to the sum of the 9 thru 0 rows in columns 9-26. Correct, reload, push "Start".
0660	The control punch in column 9 of this card is not 0, 1, 2, 3, or 4. Correct, reload, push "Start".
0661	Improper original location index. Correct, reload, push "Start".
0662	Improper original relative location. Correct, reload, push "Start".
0663	Improper original operation. Correct, reload, push "Start".
0664	Improper original address index. Correct, reload, push "Start".
0665	Improper original relative address. Correct, reload, push "Start".
0744	Mispunched or misread constant. The correct card sum is in the MQ. The card sum for this card is in the Acc. Columns 19-35 of the Acc and MQ correspond to the sum of the 9 thru the y row of columns 45-61 of the card. Correct, reload, push "Start".
1065	Improper scaling on this type #4 card. If there are no bits in the overflow positions of the Acc, the binary scaling factor was not large enough to accommodate the decimal integers. If there are bits in the overflow position, they were caused by rounding. Correct, reload, push "Start".
1317	The final location is negative or greater than 4095. The Acc will show a minus sign or the location minus 4096 respectively. If the trouble is with this card, correct, reload, push "Start". If the trouble is with the control cards (type #1, #2, or #3), correct control cards, reload 607, control cards, and proper program cards so that duplication or omission of printing and binary half-words, punched in the binary cards, will be avoided.

STOPS — All stop addresses are given in octal.

1324 The final address is negative or greater than 4095. See "Stop" 1317 for correction procedure.

1334 The final location index is improper. See "Stop" 1317 for correction procedure.

1361 The final address index is improper. See "Stop" 1317 for correction procedure.

1431 This stop only occurs if Sense Switch #3 is Down. See Sense Switch #3.

1500 The sum of the block of storage containing the control cards does not agree with the original sum. A memory dump with 186 should show the trouble. The block of storage is located from 2542_8 to 3672_8. In any case reload 607, all control cards, and the proper program cards to insure no duplication or omission of printing or binary half-words in the punched cards.

1503 See Sense Switch #4.

1617 The check sum following the punching of a binary card does not agree with the check sum punched in the card. The original check sum is in the Acc; the second check sum is in the MQ. Push "Start" to repunch the card.

2445 There is no type #1 control card in storage for this card's location. Place proper type #1 card in front of this card, reload, push "Start".

2446 There is no type #1 control card in storage for this card's address. Place proper type #1 card in front of this card, reload, push "Start".

620

H. Kolsky
T-5

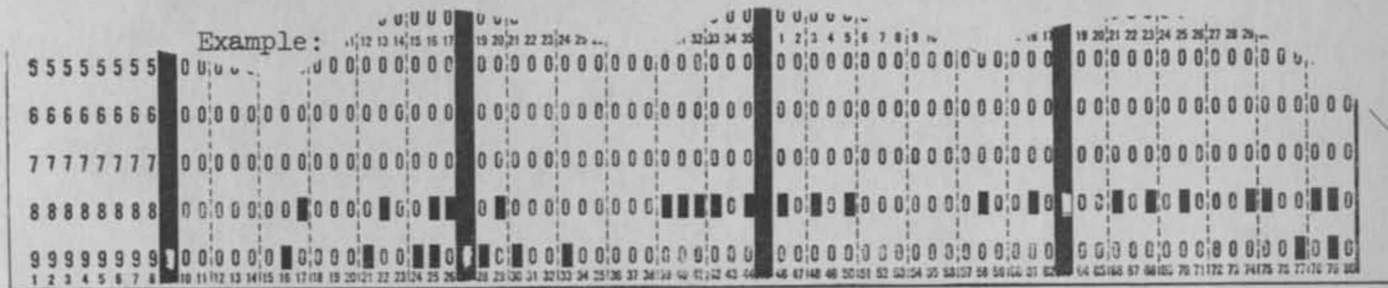
REGIONAL BINARY CARDS

A regional binary card is a card with the check sum, S, in the 9 row in columns 9-35, the half-word count, (H.W.C.) variant and invariant information in columns 36-44 and 46-80, and the first word address, (F.W.A.) in the 8 row in columns 9-26. The remainder of the card can contain up to 43 half-words.

The H.W.C. is designated by the first punch found in the 9 row in columns 36-80 excluding column 45. If the first punch is in column 36, the H.W.C. is 43. If the first punch is in column 46, the H.W.C. is 34, etc.

The first position in the 9 row following the punch for the H.W.C. is the variant or invariant information about the address of the half-word in columns 27-44 of the 8 row, the second position contains the information about the address of the half-word in columns 45-62 of the 8 row etc. A punch in a particular position of the 9 row following the H.W.C. punch indicates that the address of the corresponding half-word is invariant. No punch in that position indicates that the address is variant.

Example:



The above card is a regional binary card with three half-words.

The check sum (columns 9-35) is equal to $-2(001023322)_8 = -(002046644)_8$.

The H.W.C. is 3 indicated by a punch in column 77. The absence of punches in columns 78 and 80, indicates that the addresses of the first and third half-words are variant. The punch in column 79 indicates that the address of the second half-word is invariant.

The F.W.A. is 1023 indicated by the punches in the 8 row in columns 9-26.

Regional binary cards are punched by 607 if Sense Switch #5 is down. While assembling with 607, any regional address with the letter index I or R is considered an invariant address, all other addresses are variant. In other words, if any regional address requires a #1 control card for assembly, 607 considers that address as a variant address.

If the address part of a half-word refers to an electrostatic location, it is defined as being a variant address. All other address parts are defined as being invariant addresses.

1/14/54 This is a replacement page for the previous write-up on Regional Binary Cards.

T-1 UTILITY PROGRAMS IN REGIONAL BINARY FORM

T-1 has assembled all non-self-loading utility programs in regional binary form. These programs are located absolutely in the A region. The coder can use these programs, and the locations listed in the write-ups for the A region, to prepare the necessary cards for relocating these programs with 025 or 620.

These utility programs can be incremented from the A region to any other position in the 701 by using 025 to relocate them. Caution should be observed in preparing proper increment cards if these utility programs are to be loaded with 025 in the A region. 025A uses electrostatic storage at $(0000\text{-}0147)_8$, and the increment card must relocate the utility program (except the erasable storage) beyond $(0147)_8$.

The disadvantage of using 025 is that the programs must be so relocated, each time that they are read into the 701. The coder must also mentally add the increment to the addresses given in the write-up to obtain the "Stop" locations and other pertinent information about the program. Another disadvantage is that the entire program including the erasable storage is relocated with the program.

620 allows the coder to enter a cut-address as well as an increment so that the various parts of the utility program can be relocated where desired. 620 punches and prints the regional binary cards with the program's new locations.

REGIONAL BINARY ASSEMBLY PROGRAM, 620

620 has been designed to allow the 701 operator to correct his original code more quickly than it can be done by using 607. It also enables the coder to enter any desired utility program as an integral part of his code without reassembling that utility program with 607.

The advantage of 620 over 607 is mainly in speed. 620 reads regional binary cards. This allows information to be read into the 701 up to forty-three times as fast as 607. 620 also prints an octal listing similar to the one printed by 607, but 620 prints seven of these per page rather than the one printed by 607.

FUNCTIONS OF 620

Regional Binary Assembly program, 620, can be used to perform the following functions upon programs in regional binary form:

1. Relocate an entire program, or parts of a program, punched in regional binary form.
2. Correct one or more orders punched in one or more regional binary cards.
3. Expand a program, and, if desired, insert new orders in the gap formed by the expansion.
4. Contract a program and delete the unwanted orders (if there are such, in the gap closed by the contraction).
5. Punch a new set of correct regional binary cards.
6. Print the contents of the new regional binary cards in octal in the form of seven columns per page.

620 will only operate upon regional binary cards. Regional binary cards are obtained from an original 607 assembly, if Sense Switch #5 on the console is in the "Down" position.

CUT-ADDRESSES AND INCREMENTS

620 allows the usual cut-address and increment, i.e. all locations and all variant addresses greater than or equal to the cut-address have the

increment added to give new locations and increments. 620 also allows an upper cut-address in addition to the usual cut-address and increment. In this usage, the increment is added only to those locations and variant addresses which are greater than or equal to the usual cut-address and less than the upper cut-address.

620 has been written so that before the program begins to read the coder's cards, the usual cut-address and increment are reset to zero, and the upper cut-address is reset to 4096. Also, if the coder punches only the usual cut-address and increment in a card to be read by 620, the upper cut-address will be reset to 4096. This means that the coder can ignore the feature of the upper cut-address, if he so desires, since all locations and addresses are smaller than 4096.

INPUT

The major input to 620 is binary cards. The only other input to 620 is Sense Switch #1 which controls printing.* 620 will read three types of binary cards. They are identified by the punches or lack of punches in columns 9 and 45 of the nine row of the card as follows:

In the Nine Row		Type of Card
Col 9	Col 45	
punch	blank	A regional binary card. (See the write-up for regional binary cards.)
blank	punch	A new-order card. This card contains the correction(s) or new order(s) to be entered in the program. This card has the same form as a regional binary card except the columns 9-35 of the nine row must be blank. (See the write-up for regional binary cards.)

* If the switch is down, no printing takes place.

blank blank A control card which contains the usual cut-address and increment, and upper cut-address, if any. These are punched in binary in the nine row as follows:

Columns 15-26: The usual cut-address.

Column 27: The sign of the increment: Blank if positive; punched if negative.

Columns 33-44: The increment.

Columns 51-62: The upper cut-address. If columns 45-62 of the control card are blank, 620 enters 4096 or $(1\ 0000)_8$ as the upper cut-address.

RELOCATION OF PROGRAMS

There are three types of relocation involved. The first is the simple relocation of the entire block of orders. The second is the relocation of the latter part of the block of orders without changing the location of the first part of the block. The third type is the relocation of the first part of a block of orders or some orders in the middle of a block without changing the location of the remaining portion(s).

In the three following examples, the three types of relocation are shown.

Assume the coder desired to relocate the entire program, 527, from $(0000\text{-}0166)_8$ to $(1000\text{-}1166)_8$. The cards would be prepared and read into the 701 after 620 as follows:

1. A control card punched with the usual cut-address and upper cut-address blank and an increment of $(1000)_8$.

2. The two regional binary cards of program, 527.

Assume the coder desires to relocate everything except the erasable storage, $(0000\text{-}0011)_8$, which is to remain unchanged. $(0064\text{-}0166)_8$ is to be relocated to $(1000\text{-}1102)_8$. The cards would be prepared and read into the 701 after 620 as follows:

1. A control card punched with a usual cut-address of $(0064)_8$, and an increment of $(0714)_8$. The upper cut-address is blank.
2. The two regional binary cards of program, 527.

Assume that the coder desires to relocate the program, 527, so that the erasable storage located at $(0000\text{-}0011)_8$ remains unchanged, the "A" block located at $(0064\text{-}0066)_8$ is relocated at $(2000\text{-}2002)_8$, and the remainder of the program located at $(0067\text{-}0166)_8$ is relocated to $(1000\text{-}1077)_8$. This would require two runs with 620. The cards should be prepared and read into the 701 after 620 as follows:

First run:
1. A control card punched with a usual cut-address of $(0064)_8$, an increment of $(1714)_8$, and an upper cut-address of $(0067)_8$.
2. The two regional binary cards of program, 527.

Second run:
1. A control card punched with the usual cut-address of $(0067)_8$, an increment of $(0711)_8$ and an upper cut-address of $(2000)_8$.
2. The three regional binary cards resulting from the first run above.

In the second run it would not be necessary to read the third regional binary card into the 701 as that one is not being relocated, but, for a consistent and complete listing, it would be desirable.

CORRECTION OF REGIONAL BINARY CARDS

Rather than reassembling a block of orders with 607 to get a correct binary card, or rather than deleting an order and punching the correct order in a binary card with the attendant difficulty of getting the proper check sum, or rather than using 010 or 016 to read in decimal instructions following binary loading, the coder may use 620 to obtain a correct card. For example: Suppose card number one of program 110 (normally loaded at $(0114\text{-}0166)_8$) has the incorrect order, $(+11\ 0521)_8$, with variant address, located at $(0116)_8$. This order will be corrected to $(+11\ 0542)_8$ by preparing the binary cards and reading them into the 701 after 620 as follows:

1. The incorrect regional binary card.
2. A new-order card containing

   In the nine row: A 9 punch in columns 45 and 79

   In the eight row: The location of the order, $(0116)_8$ in columns 15-26 and the new-order, $(+11\ 0542)_8$, in columns 27-44.

620 will read the first card and locate it; then it will read the second card and store the correct order in place of the incorrect order, punch the corrected card, and print an octal listing if desired.

It will be noted that a new-order card can contain up to forty-three orders. Therefore, the coder may correct more than one order of a regional binary card by punching only one new-order card, if the orders to be corrected are close together. If they are far apart, it is easier to punch two or more correction cards to follow the original. For example: If two orders are to be corrected which have one correct order between them, the new order card can be easily punched to contain all three of the orders, but if two orders are to be corrected which have ten intervening orders, the time necessary to punch two cards, with separate locations, is shorter than punching one card with twelve orders on it.

Several different, original cards may be corrected by only one run with 620, as long as there is no overlap of their locations. (If two orders with the same location are entered by 620, the last one will replace the first one.)

EXPANSION AND INSERTION

The expansion of a program and the insertion of new orders in the gap formed by the expansion is where 620 has its main advantage over 607. The coder using 620 can enter his entire program and be sure that all cross reference addresses are properly changed at much greater speeds than 607 can operate.

It is desired to add two new orders $(-16\ 0010 \text{ and } -17\ 0012)_8$, with variant addresses, between the locations $(0514)_8$ and $(0515)_8$ in a program located from $(0100)_8$ to $(0772)_8$. There is a gap in the program at $(0640-0650)_8$, and it is desired not to relocate the programs after that. The cards are prepared and read into the 701 after 620 as follows:

1. A new-order card containing:

   In the nine row: A 9 punch in columns 45 and 78.

   In the eight row: Columns 15-26 the location $(0515)_8$,

   Columns 27-44 the order $(-16\ 0010)_8$,

   Columns 45-62 the order $(-17\ 0012)_8$.

2. A card containing the cut-address $(0515)_8$ in columns 15-26 of the nine row, the increment $(0002)_8$ in columns 33-44 of the nine row, and the upper cut-address $(0650)_8$ in columns 51-62 of the nine row.

3. The eleven regional binary cards of the original program.

620 will read and store the two new orders. It will then enter the cut-addresses and increment. 620 then reads the eleven regional binary cards and adds the increment, $(0002)_8$, to all locations and variant addresses which are greater than or equal to the cut-address, $(0515)_8$ but less than $(0650)_8$. The numbers are stored away according to their new locations, therefore, the locations $(0100)_8$ - $(0514)_8$ are stored as received, but the locations $(0515)_8$ - $(0640)_8$ are changed to $(0517)_8$ - $(0642)_8$ and stored after the two new orders entered on the first card. The locations from $(0650)_8$ to $(0772)_8$ are stored as received. 620 then punches out the correct regional binary cards, and prints an octal listing if desired.

CONTRACTION AND DELETION

Contraction is similar to expansion except that the increment is negative so that a gap in the program is closed up rather than formed. There may be orders which are to be deleted in the gap which is being closed. The operation is the same in any case. For example: Assume that in a program located $(0100)_8$ to $(0772)_8$ there is a gap or four unnecessary orders located at $(0413)_8$ to $(0416)_8$. The cards are prepared and read into the 701 after 620 as follows:

1. A card containing the cut-address $(0417)_8$ in columns 15-26 of the nine row and an increment of $-(0004)_8$ in columns 27-44 of the nine row.

2. The eleven regional binary cards of the original program.

620 enters the cut-address and increment and then reads the regional binary cards. The locations from $(0100)_8$ to $(0412)_8$ remain unchanged and are stored as received; the locations from $(0413)_8$ to $(0416)_8$ (if there are any such orders) remain unchanged and are stored as received, but the locations $(0417)_8$ to $(0772)_8$ are incremented to $(0413)_8$ to $(0766)_8$ and stored. (Note that the old locations $(0417)_8$ to $(0422)_8$ become $(0413)_8$ to $(0416)_8$ and are stored in the gap or in place of unwanted orders.) 620 then punches the correct regional binary cards and prints an octal listing if desired.

OUTPUT

620 contains a punch and a print program. Punching will take place any time that new order cards or regional binary cards are read into the 701 after 620. Printing is optional and under the control of Sense Switch #1. If the switch is down, no printing will take place. The punch output is new regional binary cards. The print output is an octal listing of the punched cards arranged in vertical columns on the printed page. There may be up to seven such columns per page, depending upon how many cards were punched.

Any column contains the following information in octal:

1. A four digit number indicating the location of the half-word.
2. The symbols + or - for the sign of the half-word.
3. A two digit number indicating the operation.
4. A four digit number indicating the address.
5. The symbol, ⌑, following any address, indicates that the address is invariant.

OPERATION OF 620

620 uses all of electrostatic for itself and for erasable storage. The storage is used temporarily to store the half-words read off cards before they are stored on the drums, and for storage after the drums are read to prepare the card images for punching and printing. Drums 128 and 129 are used as a sorter to get the half-words read into consecutive order. Each half-word read into the 701 is stored as the lower half or a full word. The upper half of that full word contains the location of the half-word in the address part and the operation "Stop" or "TR" in the operation part if the address of the half-word is variant or invariant respectively. The drum position where the half-word with its location is stored may be computed by doubling the location and using only the twelve low order bits to determine the drum location. The drum number is the thirteenth bit; 0 for Drum 128, and 1 for Drum 129.

620 can read up to 38 regional binary cards or new order cards, before erasable storage is full. At this time, control transfers to the Drum-Write program, and all half-words read are stored on the two drums. 620 will continue to read cards in groups of 38 until an End-of-File condition is set up on the Card Reader. Control then transfers following the Drum-Write program to the Drum-Search program.

The first operation in 620 is to copy into all positions of Drums 128 and 129 the full words, minus zero. The Drum-Search program starts at $(0000)_8$

of Drum 128 and copies off consecutive full words into a fixed position in electrostatic storage until the first positive full word is sensed. That full word and the following positive full words (up to forty-two such words) are transferred to the first of seven punch storages. The search is then continued for the next positive full word until all seven punch storages are full or both drums have been completely searched. At this time, control transfers to the Punch program.

The Punch program sets up the card image, computes the check sum, and punches the first card from information contained in the first punch storage. It recomputes the check sum after punching. If the two check sums agree, the second card is set up and punched. This continues until the punch storages, filled by the Drum-Search program, have all been punched. Control then transfers to the Print program. If the check sums disagree, the 701 stops. Pressing the "Start" button repunches the card.

The Print program reads the first word from each of the punch storages, forms them into a card image and prints a line. It then reads the second word from each of the punch storages, and prints a line. (Any word read from a punch storage which is negative forces the card image to remain blank, so that the corresponding column, on that line of the page, contains nothing.) Printing continues until all seven punch storages contain negative words or until forty-three lines have been printed. Control at this time transfers back to the Drum-Search program, if the search has not been completed.

If the search is complete, 620 stops after printing the last line (or punching the last card, if Sense Switch #1 is down). If the "Start" button on the console is now pushed, 620 will start over again as if 620 had just been read into the machine and the Transition card had just been read. This allows the operator to make several runs with 620 without having to load 620 into the machine before each run.

H. Kolsky
T.5

STOPS	All Stop addresses are given in octal
0447	No cards followed 620 transition card. Press "Start" to read cards.
0562	The final location is negative or greater than 4095. If the accumulator is negative, the location is negative; if it is positive, add $(1\ 0000)_8$ to get the location. Pressing the Start button allows 620 to read the next card.
0624	The final address is negative or greater than 4095 (see the above Stop 0562).
0646	The check sum computed for this regional binary card does not agree with the check sum read. The difference (the sum read minus the sum computed) is in the Accumulator. Pressing the "Start" button allows 620 to read the next card.
1016	No valid orders were found during Drum Search. Press "Start" to research Drums.
1256	The check sum of the regional binary card just punched does not agree with the recomputed check sum. The recomputed check sum is in the Accumulator. The original check sum is in the MQ. Press "Start" to repunch card.
1526	Final Assembly Stop. Assembly is complete. Press "Start" to begin 620 over again.

CRITICAL LOCATIONS

0302	Usual cut-address.
0303	Increment.
0304	Upper cut-address.
0003	Original location.
0008	Original address.
0004	Final location in the address part, $(00)_8$ or $(01)_8$ in the operation part to indicate variant or invariant address respectively.
0005	Final sign, operation, and address.

Written: Dura W. Sweeney, 3/11/54.
This page replaces the previous page 10 in the 620 write-up. Please replace all 620 decks with new ones from the files.

702 R S.L. Rewind Specified Tapes

INPUT: a. Sense Switches: #1. Controls rewinding of tape

256. Depress to rewind.

#2. Controls rewinding of tape

257. Depress to rewind

#3. Controls rewinding of tape

258. Depress to rewind

#4. Controls rewinding of tape

259. Depress to rewind

b. Basic linkage: Entry occurs automatically

Calling sequence is as follows

α	R Add	α
$\alpha + 1$	Tr	F26
$\alpha + 2$	Control returns here	

LOADING: 702 is self loading. If 702 is not to be used for rewinding tapes immediately after it loads itself, but is to be entered by linkage later then all sense switches must be up or off while loading 702. <u>Warning</u>: If 702 is to be used by linkage do not follow 702 with another self loading card from the same region as this will destroy 702.

STARTING: Automatic entry. Put the 702 card in the hopper and push the start button on the card reader. When card reader stops set load selector on console to cards, instruction entry keys to F0, automatic-manual switch to automatic, depress sense switches corresponding to the tapes to be rewound on loading, and press load button. Feed out card when tapes have been rewound or when card reader select light goes out.

DESCRIPTION: Immediately upon loading 702 will rewind any or all of the tapes depending on which sense switches are down. After rewinding or if no switches are down or on, 702 will stop at F22 and when console start is pressed will read a self loading card into F0, i.e., read one full word into F0 and transfer to F0. When entry is made by linkage 702 will stop at F28 prior to rewinding. This allows time for selection of switches controlling tapes to be rewound. After rewinding 702 will stop at F22 for switches to be restored. Upon restarting 702 will return control to original program.

PROGRAM STOPS:

Regional Location	Meaning
F22	Rewinding is finished. Permits restoration of sense switches prior to reading next card, or if basic linkage is used, prior to returning to original program. Reset switches and press console start button.
F28	Occurs on basic linkage only. Permits selection of tapes to be rewound via sense switches. Depress proper switch(es) and press console start button.

OUTPUT: Rewinding of tapes specified by selection of proper sense switches.

STORAGE: Regional: F0 thru F28

Total 29 half-words, 29 regional cards, one binary card.

CODED: EMW, ch'd EMW, written EMW

702 SL	Rewind tapes via sense	702R	702A	702B	702C
STARTING:	Set instruction entry keys to . . .	F0	0	4000_8	7000_8
INPUT:	α + 1 contains tr to	F26	32_8	4032_8	7032_8
PROGRAM STOPS:	Select sense switches and push start.	F22	26_8	4026_8	7026_8
	Restore switches and push start	F28	34_8	4034_8	7034_8
DESCRIPTION:	After rewinding 702 loads one full word into -F0 and transfers to . .	F0	0	4000_8	7000_8
	or stops at F28 and returns to original program on pushing start.				

NO.	NAME
703 R	Set drums, E.S. to 0 and rewind tapes.
INPUT:	703 card must be followed by self-loading card(s) or three blank cards.
LOADING:	703 is self-loading.
STARTING:	Put 703 card then self-loading or 3 blank cards in hopper and press card-reader start. Set load selector to cards, instructions entry keys to 0, and press the load button.
DESCRIPTION:	703 writes zeros on all drum locations; sets acc, mq, and all of E.S. to zero, rewinds all tapes, and turns off the overflow indicator, then loads one full word from the first card following the 703 card into F0 and transfers to F0 and executes that instruction. Therefore, if 703 card is followed by self-loading (into F0) cards they will be loaded, or if followed by blank cards, the 701 will stop at F0.
PROGRAM STOP:	(If 703 card is followed by blank cards)

Location	Meaning
F 0	Clearing and rewinding is finished.

OUTPUT:	All registers, E.S., drum storages are set to 0, tapes rewound, and ov turned off.
STORAGE:	703 occupies (0 thru 3 and 7724 thru $7777)_8$ when loading, and clears itself out. 703 is one binary card.
CODED:	JDM, ch'd-dtm, written -dtm. C.O. JDM Apr 53

703 Set drums, E.S. to 0 and rewind tapes

	704R	704A	704B	704C
DESCRIPTION:				
After clearing and rewinding 703 loads one full word into -F0 and transfers to F0.	F0	0	4000_8	7000_8
PROGRAM STOP:				
Clearing and rewinding is finished.	F0	0	4000_8	7000_8

NO.	NAME
704 R	Set drums, E.S. to 0
INPUT:	704 must be followed by self loading card(s) or three blank cards.
LOADING:	704 is self-loading
STARTING:	Put 704 card, then self-loading or 3 blank cards in hopper and press card-reader start. Set load selector to cards, instruction entry keys to 0, automatic-manual switch to automatic, and press the load button.
DESCRIPTION:	704 writes zeros on all drum locations; sets acc, mq and all of E.S. to zero, turns off the ov indicator, then loads one full word from the first card following the 704 card into F0 transfer to F0 and executes that instruction. Therefore, if 704 card is followed by self-loading (into F0) cards, they will be loaded, or if followed by blank cards, the 701 will stop at F0.
PROGRAM STOP:	(if 704 card is followed by blank cards)

Location	Meaning
F 0	Clearing is finished

OUTPUT:	All registers, E.S., drum storages are set to 0, and ov indicator turned off.
STORAGE:	704 occupies (0 thru 3 and 7724 thru $7777)_8$ when loading, and clears itself out. 704 is one binary card.
A & CO	WGB 5-53

		704R	704A	704B	704C
704	Set drums, E.S. to 0	704R	704A	704B	704C
DESCRIPTION:	After clearing 704 loads one full word into -FO and transfers to FO	FO	0	4000_8	7000_8
PROGRAM STOP:					
	Clearing is finished.	FO	0	4000_8	7000_8

705R S.L. Clear Specified Drums

INPUT: a. Sense Switches: #1. Controls clearing of drum 128. Depress to clear.

#2. Controls clearing of drum 129. Depress to clear.

#3. Controls clearing of drum 130. Depress to clear.

#4. Controls clearing of drum 131. Depress to clear.

b. Basic Linkage: Entry occurs automatically. Calling sequence is as follows

α	R ADD	α
$\alpha + 1$	TR	F29
$\alpha + 2$	Control returns here	

LOADING: 705 is self loading. If 705 is not to be used for clearing drums immediately after it loads itself, but is to be entered by linkage later, then all sense switches must be up, or off, while loading 705. Warning: if 705 is to be used by linkage do not follow 705 with another self loading card from the same region as this will destroy 705.

STARTING: Automatic entry. Put the 705 card in the hopper and push the start button on the card reader. When card reader stops set load selector on console to cards, instruction entry keys to F0, automatic-manual switch to automatic, depress sense switches corresponding to the drums to be cleared on loading, and press load button. Feed out card when drums have been cleared or when card reader select light goes out.

DESCRIPTION: Immediately upon loading 705 will clear any or all of the drums depending on which sense switches are down. After clearing, or if no sense switches are down 705 will stop at F 25 and when console start is pressed will read a self loading card into FO, i.e., read one full word into -FO and transfer to FO. When entry is made by linkage 705 will stop at F 31 prior to clearing. This allows time for selection of switches controlling drums to be cleared. After clearing 705 will stop at F 25 for switches to be restored. Upon restarting 705 will return control to original program.

PROGRAM STOPS:

Regional Location	Meaning
F25	Clearing is finished, permits restoration of sense switches prior to reading next card, or if basic linkage is used, prior to returning to original program. Reset switches and press console start button.
F31	Occurs on basic linkage only. Permits selection of drums to be cleared via sense switches. Depress proper switches and press console start button.

OUTPUT: Clearing of drums specified by selection of proper sense switches.

STORAGE: Regional FO thru F36

Total 37 half-words, 37 regional cards, one binary card.

CODED: EMW, ch'd EMW, written EMW.

705 SL Clear Drums via sense	702R	702A	702B	702C
STARTING: Set instruction entry keys to . . .	F0	0	4000_8	7000_8
INPUT: α+ 1 contains transfer to	F29	35_8	4035_8	7035_8
PROGRAM STOPS: Select switches and push start	F25	31_8	4031_8	7031_8
Restore switches and push start	F31	37_8	4037_8	7037_8
DESCRIPTION: After clearing 705 stops at F 31 and either returns to original program via linkage upon pressing start button or loads one full word into -F0 and transfers to . .	F0	0	4000_8	7000_8

NO.	NAME
706R	Clear E.S. to 0

INPUT: 706 must be followed by a blank card or by self loading card(s) into F0.

LOADING: 706 is self loading.

STARTING: Put 706 card then self loading or blank card(s) in hopper and press card reader start. Set load selector to cards, instruction entry keys to F0, and press the load button.

DESCRIPTION: 706 sets acc, mq and all of E.S. to 0, turns off the ov indicator, then loads one full word from the first card following the 706 card into F0 and transfers to F0 and executes that instruction. Therefore, if 706 card is followed by self-loading cards into F0, they will be loaded, or if followed by a blank card, the 701 will stop at F0.

PROGRAM STOP: (If 706 card is followed by blank card)

Location	Contents	Meaning
F0	00; 0000	Registers and E.S. are cleared.

OUTPUT: All registers and E.S. are set to 0 and ov indicator turned off.

STORAGE: (while loading)

F0 thru F5 — F0 must be even.

F 4064 thru F 4095

706 is one binary card and clears itself out. 706 can <u>not</u> be assembled by S02 or 606. 706 is 38 regional cards.

CODED: DTM 7-21-53

706	706R	706A	706B	706C
Set instruction entry keys to	FO	0	4000_8	7000_8
After clearing, 706 loads SL card(s) into	FO	0	4000_8	7000_8
PROGRAM STOP: Clearing finished	FO	0	4000_8	7000_8

H. Kolsky
T-5

NO.	NAME
707 R	Clear ES-1 and ES-2 to 0.
INPUT:	See 706 R
LOADING:	See 706 R
STARTING:	Put 707 card, then self-loading or blank card(s) in hopper and press card reader start. Set load selector to cards, ES selector to ES-1-2 or 2-1, instruction entry keys to F0, and press load button.
DESCRIPTION:	See 706 R
PROGRAM STOP:	See 706 R
OUTPUT:	See 706 R
STORAGE:	See 706 R
CAUTION:	707 cannot be used for single bank operation. Use 706 for clearing memory for single bank operation.

Coded, written, chkd., Willbanks, 5/22/54

NO.	NAME
720R	Loads itself into E.S. 1, reads control cards which specify blocks of memory in E.S. 1 and/or E.S. 2 to be compared to corresponding contents of drums. Discrepancies are punched out in binary full words.

INPUT: Input is by control cards. The eight-row left must be blank. The nine-row is punched to contain the following information in binary.

Columns	
15-26	the drum number: 0200_8 is drum #1; 0201_8 is drum #2; 0202_8 is drum #3; 0203_8 is drum #4
33-44	the S.D.A. (even)
51-62	F.W.A. (the location of the first word in memory block to be compared to the drum: even for E.S. 1; odd for E.S. 2
69-80	L.W.A. (location of the last word of the memory block): even for E.S. 1; odd for E.S. 2

LOADING: 720 is self loading

Loading Deck	# Cards
720	1
control cards	n
Total	1 + n cards

STARTING: Load card punch unit with cards.

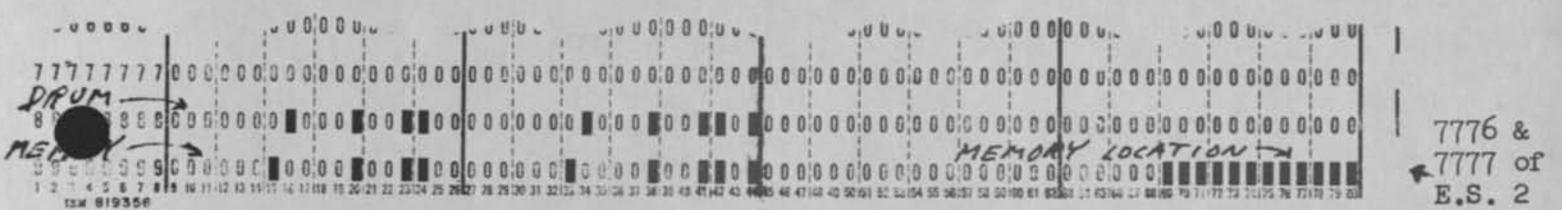
a. Automatic entry: Put loading deck in hopper, have card reader "Ready". Set instruction keys to F0, press "load" button. Press start when card reader stops on last card.

b. Manual entry: (When 720 is already in E.S.) Press "reset" button on console, put control card deck in hopper and have card reader "ready". Start manually at F7. Press card reader start button to read in last control card.

c. Transfer entry: (720 not in E.S.) Give following orders in program: Read card reader, - copy F0, tr F0. (720 already in E.S.) transfer to F7. Press card reader start to read in last control card.

DESCRIPTION: 720 loads itself into and operates in a space smaller than current card loading programs. 720 reads the first control card and compares the specified block of memory to the specified drum data. While comparing, discrepancies are punched in the following manner:

8 Row left contains full word from drum
9 Row left contains full word from memory
9 Row right gives the location in memory of the quantity punched in 9Row left (which is viewed as full word, even for E.S. 1, odd for E.S. 2)



Having finished a block of memory-drum comparison, 720 then reads succeeding control cards. Having treated the last control card, the end of the program is shown by program stop at $F12_8$. Pushing start button causes 720 to attempt to read in more control cards.

720 has no effect on overflow.

PROGRAM STOPS:

Location	Meaning
$F12_8$	Comparison has been completed. Push start to read more control cards

RESTARTING: With control cards in the hopper, manually transfer to F7.

STORAGE: F0 through F47 (48 half words)(includes erasable storage).

CODED: D. E. Harris, checked and written, D. E. Harris, 1/24/55

720 Self loading, compares blocks of memory to drums.

INPUT: By control cards.

		720	720-0000
STARTING:	Automatic entry	F0	0000_8
	Manual entry	F7	0007_8
	Transfer entry	F7	0007_8
STOPS:	Comparison has been completed	F12	0012_8
STORAGE:	Decimal	F0-	0000_{10}
		F47	0047_{10}
	Octal	F0-	0000_8
		F57	0057_8

720 is available in octal regions 0000, 1000, 2000, 3000, 4000, 5000, 6000, 7000. Octal entry, stop, and storage locations are easily computed by adding the high order octal digit to the locations specified for 720-0000.

H. Kolsky
T-51

781 Search Memory for Transfers to M

INPUT: Only 781. One binary card.

LOADING: Set instruction keys to F.0, press load button.

Stop at F.6. At this time manually enter M into the MQ and start.

DESCRIPTION: 781 will search memory starting at 0000 until it finds a transfer instruction to M. If it does not find one by the time it reaches 7777, it starts at 0000 looking for transfers to M-1 etc. Upon finding a transfer to M-K (K=0,1,2,...), the location is punched in the 9 left row and the instruction in the 9 right row. The program stops at F.40 with:

The transfer to (M-K) in the MQ, and

the location of the transfer in the accumulator.

Pressing the start button initiates 781 looking for more transfers to (M-K).

STORAGE: 781 occupies 46 half words of memory.

octal							
	OF.0	0000	1000	2000	3000		7000
	OF.6	0006	1006	2006	3006	...	7006
	OF.40	0050	1050	2050	3050		7050

Coded and checked by: L. Gatt, 12-10-54.

782 Search Memory for Stores to M

INPUT: Only 782. One binary card.

LOADING: Set instruction keys to F.0, press load button.

Stop at F.6. At this time manually enter M into the MQ and start.

DESCRIPTION: 782 will search memory starting at 0000 until it finds a store, store address or store MQ with address equal to M Upon making a complete search of memory and not finding a store to M, 782 will stop at F.6 with sense light 4. At this point, the operator may enter a new M in the MQ and start.

Upon finding a ST, SA, or SM inst to M, 782 will punch the location in the 9 left row and the instruction in the 9 right row. Then 782 will stop with:

The store inst in the MQ, and

the location of the store in the accumulator.

Pressing the start button initiates 782 looking for more stores to M.

STORAGE: 782 occupies 46 half words of memory.

octal							
F.0	0000	1000	2000	3000		7000
F.6	0006	1006	2006	3006	'''	7006
F.40	0050	1050	2050	3050		7050

Coded and checked by: L. Gatt, 12-10-54.

NO.	**NAME**
784 R	Print operators on given instructions.

INPUT: Control card: Punch in binary in the 9 row the following information in the specified columns:

	Columns
First Location	15 - 26
Last Location	33 - 44
First Address	51 - 62
Last Address	69 - 80

Leave rest of card blank.

LOADING: Load 784 binary cards with 021

Loading Deck	# Cards
021	1
784	3
Transition to 784 (TR F0)	1
Control Cards	n
Total	n + 5

STARTING: a. Automatic entry: Put the loading deck in hopper and have card reader ready. Set load selector to cards, instruction entry keys for 021, and press load. When 701 stops on last card, press card reader start.

b. Manual entry (when 784 is already in E.S.): Place control cards in card reader and have it ready. Start 701 manually at F0.

c. Entry by unconditional transfer: Have control cards in card reader. Transfer to F0.

DESCRIPTION: The 701 will look at those consecutive half-words starting with the first location and ending with the last location (see INPUT). Those half-words with address parts equal to the first address (see INPUT) will be printed in octal along with their locations. This process will be repeated for those consecutive addresses beginning with the first address and ending with the last address (see INPUT). If non-consecutive addresses are desired, individual control cards must be used. In addition the 701 will print the operands, i.e. the contents of the first address will be printed along with the address itself, etc. This information is distinguished from the other information by an asterisk. Note that the last location must be greater than or equal to the first location, and the last address must be greater than or equal to the first address.

NOTE: Use 793 tracing board in printer.

PROGRAM STOP:

Regional	Meaning
F86	701 has obtained all information and is ready to read next control card.

STORAGE: Regional

A0 thru A2

F0 thru F86

E0 thru E15 , E0 even

CODED: BW, ch'd - BW , written BW

CROSS REFERENCES

		R	A	B	C	M1
STORAGE:	Decimal	A0 -	52 -	2100 -	3636 -	800 -
		A2	54	2103	3638	802
		F0 -	55 -	2103 -	3639 -	803 -
		F86	141	2189	3725	889
		E0 -	2 -	2048 -	3584 -	890 -
		E15	17	2063	3599	905
	Octal	A0 -	(64 -	(4064 -	(7064 -	(1440 -
		A2	66)$_8$	4066)$_8$	7066)$_8$	1442)$_8$
		F0 -	(67 -	(4067 -	(7067 -	(1443 -
		F86	215)$_8$	4215)$_8$	7215)$_8$	1571)$_8$
		E0 -	(2 -	(4000 -	(7000 -	(1572 -
		E15	21)$_8$	4017)$_8$	7017)$_8$	1611)$_8$
START:						
	Decimal	F0	0055	2103	3689	0803
	Octal	F0	$(0067)_8$	$(4067)_8$	$(7067)_8$	$(1443)_8$
STOP, ALL THROUGH						
	Decimal	F86	0141	2189	3725	0889
	Octal	F86	$(0215)_8$	$(4215)_8$	$(7215)_8$	$(1571)_8$

memory printout

NO.	NAME
785	Compares original program cards with program stored in electrostatic memory and prints out all half-words that do not agree. Use with 526.

INPUT:

Loading deck	# Cards
526	4
706A	1
021A	1
785	5
Transition to 785	1
Original program cards	n
Total	n + 12

STARTING: Automatic entry - Put loading deck in the hopper of the card reader. Have card reader ready. Set instruction keys to zero, and press the load button. Press card reader start when 701 stops on the last card. There is no manual entry. There is no entry by transfer.

DESCRIPTION: 526 writes all of electrostatic memory on drum #1 with the exception of the first two full words -0000 and -0002. 706A clears E.S. to zero. 021A loads 785. 785 reads each program card into memory, reads the corresponding half-words from the drum, compares them and prints out only half-words that do not agree. The printout consists of the location of the half-word, the half-word from the program card, and the half-word from the drum. The listing is double spaced after the printing for each program card.

PROGRAM STOPS:	Location	Meaning
	73_8	End of file; all binary cards have been read and checked.
	122_8	S on the card does not agree with computed check sum. If start is pressed, 785 will load the next card.
	46_8	See 021 stops.

OUTPUT: Printed sheets with the location of the half-word, the half-word read from the card, and the half-word from the drum on each line when there is disagreement. Use 186 print board with alteration switch #4 up.

CODED: M.C.F., checked and written, M.C.F.

H. Kolsky
T-5

785 and 786 Revision

785 and 786 will now restore electrostatic storage to its original form after making the comparison, except that full words -0000 and -0002 have been destroyed.

New 785 and 786 decks have been put in the files.

May 17, 1954

H. Kolsky
T-5

786 Compares original regional binary program cards with program stored in electrostatic memory and prints out all half-words that do not agree. Use with 526.

INPUT:

Loading deck	# cards
526	4
706A	1
026A	1
786	6
transition to 786	1
increment card	1
regional binary prog.	n
increment card	
regional binary prog.	
- - - - - - - - - -	
- - - - - - - - - -	

STARTING: Put loading deck in the hopper of the card reader. Have card reader ready. Set instruction keys to zero, and press the load button. There is no manual entry. There is no entry by transfer.

DESCRIPTION: 526 writes all of electrostatic memory on drum #1 with the exception of the first two full words -0000 and -0002. 706A clears E.S. to zero. 026A loads 786. 786 reads each program card into memory, adds the increment to all variant addresses, reads the corresponding half-words from the drum, compares them and prints out only half-words that do not agree. Several programs may be compared simultaneously. If they all have the same increment, only one increment card is needed. But a different increment card may precede each program. <u>If the increment is to be zero, it must be minus zero</u>. The printout consists of the location of the half-word, the half-word from the program card, and the half-word from the drum. The listing is double spaced after the printing for each program card.

PROGRAM STOPS:

Location	Meaning
67_8	End of file; all binary cards have been read and checked.
126_8	S on the card does not agree with computed check sum. If start is pressed, 785 will load the next card.
45_8	See 026 stops.

OUTPUT: Printed sheets with the location of the half-word, the half-word read from the card, and the half-word from the drum on each line where there is a disagreement. Use 186 print board with alteration switch #4 up.

CODED: MCF, checked & written MCF.

Kolsky, Harwood
T-5

787 Compares original program cards with program stored in ES-1 and ES-2 and prints out all half-words that do not agree. Use with 526.

INPUT: Loading deck

526	4
706A	1
026A	1
787	6
transition to 787	1
original program cards	n

STARTING: Automatic entry - Put loading deck in the hopper of the card reader. Have card reader ready. Set instruction keys to zero, and press the load button. Press card reader start when 701 stops on the last card. There is no manual entry. There is no entry by transfer.

DESCRIPTION: 526 writes all of ES-1 on drum #1 with the exception of the first two full words -000 and -0002. 706A clears ES #1 to zero. 026 loads 787. 787 reads each program card into memory, reads the corresponding half-words from drum or ES-2, compares them and prints out only half words that do not agree. If the half word is from ES-2, ES-2 is printed on the sheet to the far right. The listing is double spaced after the printing for each program card. Information on drum #1 is read back into ES-1 after 787 has finished the comparison.

PROGRAM STOPS: Location

Location	
1	All cards have been read and ES-1 has been restored to its original form except that -0000 and -0002 have been destroyed.
126_8	S on the card does not agree with the computed check sum. If start is pressed, 787 will load the next card.
54_8	See 026 stops.

OUTPUT: Printed sheets with the location of the half-word, the half-word read from the card, and the half-word from the drum or ES-2 and ES-2 if the half-word is from ES-2 on each line when there is disagreement. Use 186 printboard.

CODED: MFA, written & checked MFA.

788 Compares original regional binary program cards with program stored in ES-1 and ES-2 and prints out all half-words that do not agree. Use with 526.

INPUT: Loading deck

526	4
706A	1
026A	1
788	8
transition to 788	1
increment card	n
regional binary prog.	
increment card	
regional binary prog.	

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

STARTING: Put loading deck in the hopper of the card reader. Have card reader ready. Set instruction keys to zero and press the load button. There is no manual entry or entry by transfer.

DESCRIPTION: 526 writes all of electrostatic memory on drum #1 with the exception of the first two full words -0000 and -0002. 706 clears ES-1 to zero. 026 loads 788. 788 reads each program card into memory, adds the increment to all variant addresses, reads the corresponding half-words from the drum or ES-2, compares them and prints out only half-words that do not agree. Several programs may be compared simultaneously. If they all have the same increment, only one increment card is needed, but a different increment card may precede each program. The printout consists of the location of the half word, the half word from the program card, half-word from the drum or ES-2, and ES-2 if the half word is from there. The listing is double spaced after the printing for each program card.

PROGRAM STOPS: Location

Location	
1_8	All cards have been read and ES-1 has been restored to its original form except that -0000 and -0002 have been destroyed.
126_8	S on the card does not agree with the computed check sum. If start is pressed 788 will load the next card.

OUTPUT: Printed sheets with the location of the half word, the half word read from the card, the half word from the drum or ES-2, and ES-2 if the half word is from there. Use 186 printboard.

CODED: MFA, written & checked MFA.

Harwood Kolsky
T-5

Delayed Tracing

NO.	NAME
790 R	Tracing

INPUT: Control card is punched in binary in the 9 row as follows:

col. 9	no punch = plus for ordinary tracing (without "trap") 9 punch = minus for tracing with execution of a trap.
col's 15 thru 26	R = location of the first instruction to be traced (except when the trap is to be executed immediately, in which case 15 thru 26 contain M).
col's 33 thru 44	M = the location of the first instruction of the trap.
col's 51 thru 62	N = the location where tracing is to be resumed after execution of the trap; the last instruction executed in the trap must bring control to N.

SWITCHES: See DESCRIPTION below for a more detailed explanation of 790 switches. Any combination of settings of the three switches is permissible.

2 Breakpoint switch

on (down) 701 stops at 73 on breakpoints (when a negative transfer instruction is executed in the code being traced). Push start to continue tracing as usual.

off (up): 701 does not stop on breakpoints (ignores signs of transfer instructions).

#3 Print switch

on (down): 701 prints the listing described under OUTPUT below as it traces.

off (up): 790 traces at full speed without printing.

#4 I-O switch

on (down): "Dummy" execution of read, write, and read backward (operations 24 thru 26) instructions occurs when these operations are encountered in the code being traced.

off (up): Whenever a read, write, or read backward instruction is encountered in the code being traced, control leaves the tracing program and is transferred to that I-O instruction immediately after putting the proper "contents" in the various registers. Control does not return to tracing unless this is provided in the code following the I-O instruction. Control can be made to return to tracing after execution of an I-O loop by use of a trap (see below).

LOADING: Load 790 with 021.

Loading deck	# cards
021	1
790	6
Transition to 790: 02 or 01; 7F0	1
790 control card	1
Total	9

STARTING: Put tracing board (790 - 793) in printer and have printer ready. The contents of all the registers and condition of the overflow indicator are preserved on all types of entry.

a. Automatic entry with control card: Put the loading deck in hopper and have card reader ready. Set the instruction entry keys for 021. Press card reader start, then load button. Feed out cards when select light on card reader goes out.

b. Manual entry with control card: (When 790 is already stored in E.S.) Put control card in card reader, press card reader start, start 701 manually at 7F0.

c. Entry by unconditional transfer: Load the following,

E 46: + 00; R for ordinary tracing or
- 00; R for tracing with trap

E 47: + 00; M

E 48: + 00; N

then transfer to 7F9.

DESCRIPTION: Trap. A trap is a portion of the coder's program which is to be executed full speed without tracing before tracing begins or at some time after tracing has begun. M is the location of the first instruction of the trap.

The trap is executed as follows: After reading of the control card tracing begins as usual with the instruction in R (unless R = M). When 790 reaches the point where it is about to trace the instruction in M, it replaces (temporarily destroys) the instruction in N with a transfer back to 790. Therefore the contents of N are destroyed only during the execution of the trap. Control is then given to M (after filling registers with proper contents) and goes full speed until coming to N. The last instruction in the trap must either be located in N-1 or control must reach N by transfer or logical skip within the trap. Control then goes back to tracing, registers are preserved, the contents of N are replaced, registers are filled, and tracing begins again starting with the instruction in N.

Each time thereafter (until a new control card is read) when 790 is about to trace the instruction in M, it instead executes the trap, then begins tracing again with N. Contents of all the registers are preserved on entering a trap or re-entering tracing from a trap. Only one trap can be executed at a time, that is, one consecutive portion of the coder's program can be a trap, although this same trap will be executed over and over if the coder's program goes through that section of his code repeatedly. Note that there are no tracing print out and breakpoint stops during execution of the trap since control is not in 790.

Tracing. The 701 will trace the instructions beginning with the instruction in R (unless R = M and R is preceded by a minus sign) keeping the contents of the acc, mq and status of the overflow indicator after the execution of each instruction, printing out this and additional information (see OUTPUT) if the print switch is on. Breakpoints, or intermediate stops, are indicated in the code being traced by minus signs on the transfer instructions. When 790 encounters such a transfer, the 701 will stop at 7F73 if the breakpoint switch is on and control actually transfers (the transfer is unconditional or, if conditional, the condition for transfer is satisfied) before tracing of the next instruction. This allows for manual corrections, changing board, etc. If console lights are not disturbed after a breakpoint stop, when the start button is pressed tracing continues starting with the next instruction (whose location is the address part of the breakpoint, i.e., negative transfer instruction). Contents of all the registers and the status of the overflow indicator are preserved.

If the stop and transfer instruction being executed has been disturbed (by execution of manual entries and corrections on console) start 701 manually at 7F74 to continue tracing. If memory display is pushed after a breakpoint stop, start must be pressed twice to continue. The location of the instruction currently being traced may be read from E46 when control is at a breakpoint or program stop. Print out of the breakpoint or stop instruction (if printing) occurs after the stop.

Print out occurs after the tracing and execution of each instruction if switch #3 is down. If the print switch is off, tracing proceeds at full speed without printing. Tracing with printing occurs at the rate of 150 instructions per minute.

A "dummy" execution of read, write, and read backward instructions may be substituted for actual execution by having switch #4 on. This dummy execution is simply an unconditional transfer to the next instruction; no I-O unit is selected, no information passes between E.S. and any I-O unit, no end of file skips etc. will occur. The original I-O instruction remains unchanged in the code being traced and appears on the print out. It is not executed; however, and the contents of all the registers remain exactly the same after dummy execution as they were before the R, W, or RB was encountered. The alternatives to dummy execution are an unconditional transfer to the first R, W, or RB encountered, or to make the I-O loop into a trap (see above). It is impossible to trace and execute I-O instructions simultaneously because only one I-O unit can be selected at a time, and tracing would always exceed the timing limitations even if the tracing was not printing. When switch #4 is off and 790 encounters a R, W, or RB instruction in the code being traced, control is taken away from the tracing program and given to this I-O instruction and does not return to tracing unless a trap has been inserted or special provisions are made in the I-O code which follows.

Dummy execution of copy instructions consists of a transfer to the next instruction and loading of the mq with the full or half word called for by the copy instruction. If the copy order was plus, the half word will still appear in the left of the mq. Dummy execution of copies always occurs while tracing, i.e., whether switch #4 is on or off. Forced dummy execution of copies avoids copy check which might occur because of lack of end of record, end of file skips, etc. when switch #4 is down.

PROGRAM STOPS: 7F 73 C (00; 7F74) Breakpoint; push start to continue

7F 60 C (00; 7F73) Program stop in code being traced; push start to continue.

OUTPUT: Print out consists of the following information, nine quantities per line, from left to right:

Information	Converted To
(1) location of instruction	an octal integer
(2) instruction	
sign	- for minus, blank for plus
operation part	an octal integer
address part	an octal integer
(3) overflow bits	0, 1, 2, or 3
(4) sign and contents of acc	blank or -; an octal integer
(5) sign and contents of mq	blank or -; an octal integer
(6) sign and contents of the storage location referred to in the address part of the instruction	blank or -; an octal integer; if a half word, the 6 right octal digits will be zeros.

Commas are printed between the two half words of (4), (5), and (6).

(7) status of the ov indicator	ON for on, blank for off
(3) ov bits	0, 1, 2, or 3
(8) sign and contents of the acc	blank or minus; a decimal fraction
(9) sign and contents of the mq	blank or minus; a decimal fraction

RESTARTING: Start as before, see STARTING b and c.

STORAGE: A0 thru A2

E0 thru E55

7B0 thru 7B23

7F0 thru 7F222

Origins A0 and E0 must be even.

250 regional cards, 6 binary cards

CODED: D. T. Monk

Harwood Kolsky
T-5

790: TRACING

		790R	790A	790B	790C
STARTING:	For automatic entry set instruction entry keys to		0	4000_8	7000_8
	For manual entry, start at ...	7F0	123_8	4123_8	7123_8
	Entry by unconditional tr				
Load decimal	± 00; R in ...	E46	46_{10}	2094	3630_{10}
	+ 00; M in ...	E47	47_{10}	2095	3631_{10}
	+ 00; N in ...	E48	48_{10}	2096	3632_{10}
octal	± 00; R in ...	E46	56_8	4056_8	7056_8
	+ 00; M in ...	E47	57_8	4057_8	7057_8
	+ 00; N in ...	E48	60_8	4060_8	7060_8
tr to ...		7F9	92_{10}	2140_8	3676_{10}
...		7F9	134_8	4134_8	7134_8
DESCRIPTION:	If instruction counter has been disturbed during a breakpoint or program stop, to continue tracing tr to ...	7F74	235_8	4235_8	7235_8
	The location of the instruction currently being traced may be read from the address part of ...	E46	56_8	4056_8	7056_8
PROGRAM STOPS:	Breakpoint, push start to continue ...	7F73	234_8	4234_8	7234_8
	Program stop in code being traced; push start to continue ...	7F60	217_8	4217_8	7217_8

STORAGE:	decimal	790R	790A	790B	790C
		A0-	56-	2104-	3640-
		A2	58	2106	3642
		7B0-	59-	2107-	3643-
		7B23	82	2130	3666
		7F0-	83-	2131-	3667-
		7F222	305	2353	3889
		E0-	0-	2048-	3584-
		E55	55	2103	3639
	octal	A0-	(70-	(4070-	(7070-
		A2	$72)_8$	$4072)_8$	$7072)_8$
		7B0-	(73-	(4073-	(7073-
		7B23	$122)_8$	$4122)_8$	$7122)_8$
		7F0-	(123-	(4123-	(7123-
		7F222	$461)_8$	$4461)_8$	$7461)_8$
		E0-	(0-	(4000-	(7000-
		E55	$67)_8$	$4067)_8$	$7067)_8$
A & CO:	DTM 8-5-53				

H. Kolsky
T-5

791 R Determine the cause of an overflow.

INPUT: The location of the first operation to be performed must be stored in the address part of OB1. The contents of the accumulator and MQ must be as the coder's program would require for performing this operation. The sense switches on the console should be adjusted according to the needs of the coder's program. Control must then be transferred to either OF0 or OF1. Transferring to OF0 turns off the overflow indicator before tracing, if it was on. Transferring to OF1 with the overflow indicator on results in the program stop at OF27.

LOADING: The coder may either wish to start tracing at the beginning of his program or at some point in it. To start tracing at the beginning, the following loading deck is suggested:

Loading deck	# cards
026 (or 028)	1 (or 2)
Coder's binary deck	n
791 R	1
081	1
Octal instruction OB1 0 address of first instruction	1
Binary transition card to OF0	1
Total	n+5 (or n+6)

To start tracing in the middle of a program, it is easier to start after a transfer has been executed. However, such a transfer must be one that is never executed again after tracing has begun. Let such an instruction be L T X where L is its location, T the transfer operation and X the address for where control would be transferred. Then the following loading deck is suggested:

Loading deck	# cards
026 (or 028)	1 (or 2)
Coder's binary deck	n
791 R	1
081	1
Octal instruction OB1 0 X	1
Octal instruction L T OF0 or Octal instruction L T OF1	1
Coder's binary transition card	1
Total	n+6 (or n+7)

STARTING: Adjust sense switches on console according to the needs of the coder's program. Put loading deck in card reader; set instruction entry keys for 026 (or 028); press load. When 701 stops, press card reader start.

DESCRIPTION: 791 R is a high speed, non-printing, tracing program whose sole purpose is to determine the cause of an overflow. It takes from 82 to 96 more machine cycles to perform an order with 791 R than without, except for multiplications or divisions, in which case, less extra time is usually used.

791 R will perform all the operations of a coder's program, unless there is an end of record skip in the program, until an overflow has been produced. However, one should not attempt to use 791 R to trace input-output operations requiring timing.

Before performing each operation of the coder's program, 791 R tests the overflow indicator and, if it is on, turns it off and stops at OF27 leaving the address of the next instruction to be performed in the accumulator. Pressing the start button causes 791 R to continue performing.

Instruction OF0 of 791 R is "transfer on overflow to OF1".

A program stop or divide check or copy check at OF20 is due to either the coder's program or to timing. The address of the offending operation plus 2^{-17} will be found in OB1. Pressing the start button after such a program stop or copy check causes 791 R to continue tracing the coder's program.

The operation and sign of the half-word in OB1 is immaterial with the following exception: If the operation part of OB1 is ± copy, the overflow light will go on just before operation 4095 is performed. For the program stop at OF27, the operation part of the accumulator will be the absolute value of the operation part of OB1.

STORAGE:

OF0 thru OF27
OA0 thru OA1
OB0 thru OB3   OB0 even
One binary card.
31 regional cards since none are required for OB1 thru OB3.
One binary card.

For tracing with a two-bank memory, 791 R must be in the first bank. Also, the part of the coder's program being traced and all half-word storage to which it refers must be in the first bank.

CODED: John Holladay. Checked & written: John Holladay

791 Determine the cause of an overflow.

		791 R	791 A	791 B	791 C
INPUT:	Store address of first order to be executed in	OB1	131_8	4131_8	7131_8
	To turn off overflow and enter program, transfer to	OF0	72_8	4072_8	7072_8
	To leave the overflow indicator alone and enter program, transfer to	OF1	73_8	4073_8	7073_8
PROGRAM STOPS:	Overflow. For this stop, the address of the next instruction will be in the address part of the accumulator.	OF27	125_8	4125_8	7125_8
	Coder's stop, divide check or copy check	OF20	116_8	4116_8	7116_8
	Address of coder's offending instruction plus 2^{-17}	OB1	131_8	4131_8	7131_8
STORAGE:	Decimal	OF0	58	2106	3642
	thru	OF27	85	2133	3669
		OA0	86	2134	3670
	thru	OA1	87	2135	3671
		OB0	88	2136	3672
	thru	OB3	91	2139	3675
	Octal	OF0	72	4072	7072
	thru	OF27	125	4125	7125
		OA0	126	4126	7126
	thru	OA1	127	4127	7127
		OB0	130	4130	7130
	thru	OB3	133	4133	7133

NO.	NAME
793 R	Tracing, octal and decimal print-out

INPUT: Control card for STARTING is punched as follows: 9 row, columns 15 thru 26, contains in binary the location of the first instruction to be traced.

LOADING: Load 793 with 021. See 021 for complete loading instructions.

Loading deck	# cards
021	1
793	6
Transition to 793	1
793 Control Card	1

SWITCHES: See DESCRIPTION below for a more detailed explanation of 793 switches. Any combination of settings of the three switches is permissible.

#2 Breakpoint switch

on (down): 701 stops at F55 on breakpoints (when a negative transfer instruction is executed in the code being traced). Pushing start causes tracing to continue as usual.

off (up): 701 does not stop on breakpoints (ignores signs of transfer instructions).

#3 Print switch

on (down): 701 prints the listing described under OUTPUT below as it traces.

off (up): 701 traces at full speed without printing.

#4 Input-Output Switch

on (down): "Dummy" execution of read, write and read backward instructions occurs when these operations are encountered in the code being traced.

off (up): Whenever a read, read backward or write instruction is encountered in the code being traced, control leaves the tracing program and is transferred to that I-O instruction immediately after putting the proper "contents" in the various registers. Control does not return to tracing unless this is provided in the code following the I-O instruction.

STARTING: Put 793 tracing board in printer and have printer ready.

a. Automatic entry with control card: Put the loading deck in hopper and have card-reader ready. Set instruction entry keys to zero. Press card-reader start, then load button*. Feed out cards when select light on card reader goes out.

b. Manual entry with control card: (When 793 is already stored in E.S.) Put control card in card-reader, press card-reader start. Start 701 manually at FO. Contents of all the registers will be preserved for tracing of the first instruction.

c. Entry by unconditional transfer: Load MQ address bits with the location of the first instruction to be traced, and transfer to F 10*.

*see page 793 R - 7, bottom

DESCRIPTION: The 701 will trace the instructions beginning with the location given in the control card or stored, keeping the contents of the acc, mq, ov bits and status of the ov indicator after the execution of each instruction, printing out this and additional information (see OUTPUT) if the print switch is on. Breakpoints, or intermediate stopping places, are indicated in the code being traced by negative signs on the transfer instructions. When the 701 encounters such a transfer while tracing, it will first see if it should stop on breakpoints (switch #2). If switch #2 is on, and control actually transfers (the transfer is unconditional or, if conditional, the conditions for transfer are satisfied), then the 701 will stop before tracing the next instruction. This allows for changing of switches, tapes, filling hoppers, etc. or manual changing of some of the contents of E.S. (usually corrections), and tells the coder where control is tracing in his code. If console lights are not disturbed after a breakpoint stop, when the start button is pressed tracing continues starting with the next instruction (whose location is the address part of the breakpoint, i.e., negative transfer instruction). Contents of all the registers and the status of the overflow indicator are preserved.

If the stop and transfer instruction being executed has been disturbed, start 701 manually at F56 to continue tracing; pressing memory display takes stop out of the instruction register - push start twice to continue. The location of the instruction currently being traced may be read from the address part of F12 when control is at a breakpoint or program stop. Print out of the breakpoint or stop instruction (if printing) occurs <u>after</u> the breakpoint or program stop. Printing may be begun or discontinued at breakpoint or program stops only.

Print out occurs <u>after</u> the tracing and execution of each instruction if switch #3 is on. If the print switch is off, tracing proceeds at full speed without printing. Tracing with printing occurs at the rate of 150 instructions per minute.

A "dummy" execution of read, read backward and write instructions may be substituted for actual execution by having switch #4 on. This dummy execution is simply an unconditional transfer to the next instruction; no I-O unit is selected, no information passes between E.S. and any I-O unit, no end of file skips, etc. will occur. The original I-O instruction remains unchanged in the code being traced and appears on the print out. It is not executed, however, and the contents of all the registers remain exactly the same after dummy execution as they were before the R, W or RB was encountered. The alternative to dummy execution is an unconditional transfer to the first R, W or RB encountered. It is impossible to trace and execute I-O instructions simultaneously because only one I-O unit can be selected at a time, and tracing would always exceed the timing limitations even if the tracing was not printing. When switch #4 is off and a R, RB or W is encountered in the code being traced, control is taken away from the tracing program and given to this I-O instruction and does not return to tracing unless special provisions are made in the I-O code which follows.

Dummy execution of copy instructions always occurs while tracing, i.e., whether switch #4 is on or off. Forced dummy execution of copies avoids copy check which might occur because of lack of end of record, end of file skips, etc. when switch #4 is on.

PROGRAM STOPS:

Regional Location	Meaning
F 55	Breakpoint. Push start to continue tracing.
F 51	Program stop in the code being traced. Push start to continue.

OUTPUT: Print out consists of the following information, nine quantities per line, from left to right:

information	converted to
(1) location of the instruction	an octal integer
(2) instruction	
sign	- for minus, blank for plus
operation part	an octal integer
address part	an octal integer
(3) overflow bits	0, 1, 2, or 3
(4) sign and contents of the acc	blank or -; an octal integer
(5) sign and contents of the mq	blank or -; an octal integer
(6) sign and contents of the storage location referred to in the address part of the instruction	blank or -; an octal integer if a half-word, the 6 right octal digits will be zeroes.

Commas are printed between the two half-words of (4), (5), and (6).

(7) status of the overflow indicator	ON for on, blank for off
(3) overflow bits	0, 1, 2, or 3
(8) sign and contents of the acc	- for minus, blank for plus; a decimal fraction
(9) sign and contents of the mq	- for minus, blank for plus; a decimal fraction

RESTARTING: Start as before, see STARTING b and c

STORAGE, REGIONAL

A0 thru A2

E2 thru E50

F0 thru F224

Total: 277 half words

For regional assembly by IBM SO_2, origins A0, E0, and F0 must be specified:

A0, E0, must be even.

CODED: DLT, eh'd-jdm, written dtm.

*When tracing is started with a control card and manual start, the proper status of the overflow indicator is preserved for tracing of the first instruction. However, if the loading deck is used, tracing will begin with the ov indicator off. If started by transfer, it is assumed that the overflow indicator is off before tracing and execution of the first instruction, whether it is on or off. If, however, it was on, the proper status will be preserved for tracing of the second and all following instructions.

793 Tracing, decimal and octal print out.

		793R	793A	793B	793C
STARTING:	For automatic entry, set instruction entry keys to ...		0	4000_8	7000_8
	For manual entry, start 701 at ...	F0	67_8	4067_8	7067_8
	For entry by unconditional transfer, transfer to ...	F10	65_{10}	2113_{10}	3649_{10}
DESCRIPTION:	If instruction counter has been disturbed during a breakpoint or program stop, to continue tracing, transfer to ...	F56	157_8	4157_8	7157_8
	The location of the instruction currently being traced may be read from the address part of ...	F12	103_8	4103_8	7103_8
PROGRAM STOPS:	Breakpoint, push start to continue ...	F55	156_8	4156_8	7156_8
	Program stop in code being traced; push start to continue	F51	152_8	4152_8	7152_8
STORAGE:	decimal	A0-	52-	2100-	3636-
	thru	A2	54	2102	3638
		F0-	55-	2103-	3639-
	thru	F224	279	2327	3863
		E2-	2-	2050-	3586-
	thru	E50	50	2098	3634
	octal	A0-	(64-	(4064-	(7064-
	thru	A2	$66)_8$	$4066)_8$	$7066)_8$
		F0-	(67-	(4067-	(7067-
	thru	F224	$427)_8$	$4427)_8$	$7427)_8$

793R E2- E50	793A (2- $62)_8$	793B (4002- $4062)_8$	793C (7002- $7062)_8$

H. Kolsky
T-5

NO.	NAME
794 R	Tracing with optional operation or address-range selection.

INPUT: Control card for starting is punched as follows: 9 row, columns 15-26, contains in binary the location of the first instruction to be traced.

LOADING: Load 794 with 021 or 026.

Loading deck	# cards
021 or 026	1
794	6
Transition to 794	1
794 Control Card	1

STARTING: Put 794 tracing board in printer and have printer ready. Place loading deck in hopper and have card reader ready. Set the Instruction Entry keys for 021 or 026. Press card reader start and then Load button. See description below of Sense Switches for explanation of programmed stops.

SWITCHES: #2 ON (down) OFF (up)

Stop on minus transfers. No stop minus transfers.

ON (down): 701 stops at F67 on breakpoints (when a negative transfer instruction is executed in the code being traced. Push start to continue tracing.

OFF (up): 701 ignores signs of transfer instructions.

#3 ON (down) OFF (up)

Operation trace. Address-range trace.

ON (down): Operation tracing. 701 stops at F 14. Set automatic-manual switch to manual. Enter operation in octal into the operation part of the MQ register via MQ-Entry keys. Take switch off manual and push start. See description of OUTPUT for further explanation.

OFF (up): Address-range tracing. 701 stops at two places, F17 and F20. Load manually first octal address of the address-range into the address part of the MQ register on first stop. Take switch off manual and push start. Load last address of the address-range in the same manner on the second stop. See description of OUTPUT for more details.

#4	ON (down)	OFF (up)
	"Dummy" execution of R, W, & RB.	No "dummy" execution.

ON (down): "Dummy" execution of read, write and read backward instructions occurs when these operations are encountered in the code being traced.

OFF (up): Whenever a read, read backward or write instruction is encountered in the code being traced, control leaves the tracing program and is transferred to that I-O instruction immediately after putting the proper "contents" in the various registers. Control does not return to tracing unless this is provided in the code following the I-O instruction.

DESCRIPTION: The 701 will trace the instructions beginning with the location given in the control card, keeping the contents of the Acc, MQ, OV bits and status of the OV indicator after the execution of each instruction. In the operation trace (Sense Switch #3 down) only those instructions with operation equal to the preset operation are printed out (See OUTPUT). In the address-range trace, only those instructions with address equal to either, or lying within the range, of the first and last address are printed out (See OUTPUT). Break-points are indicated in the code being traced by negative signs on the transfer instructions. If the transfer is executed, the 701 will stop before tracing the next instruction. If the console lights are not disturbed after a breakpoint stop, when the start button is pressed, tracing continues starting with the next instruction whose location is the address part of the negative transfer instruction. Contents of the registers and the status of the overflow indicator are preserved.

A "dummy" execution of read, read backward and write instructions may be substituted for actual execution by having switch #4 on. This dummy execution is simply an unconditional transfer to the next instruction. "Dummy" execution of copy instructions always occurs while tracing.

PROGRAM STOPS:

Regional Location	Meaning
F67	Breakpoint. Push start to continue tracing.
F14	Load pre-set operation for operation tracing. Push start.
F17	Load <u>first</u> address of the address range for address tracing. Push start.
F20	Load <u>last</u> address of the address range for address tracing. Push start.

OUTPUT: Print out consists of the following information, eight quantities per line, from left to right:

	Information	Converted to
(1)	Status of OV indicator	+1 for ON; +2 for OFF
(2)	Overflow bits	0, 1, 2, or 3
(3)	Location of the instruction	an octal integer

(4) Instruction	
sign	- for minus, + for plus;
operation part	an octal integer
address part	an octal integer
(5) Sign and Contents of Acc as an instruction (17 high order bits)	- for minus, + for plus; an octal integer
(6) Sign and Contents of Acc	- for minus, + for plus; a decimal fraction
(7) Sign and Contents of MQ	- for minus, + for plus; a decimal fraction
(8) Sign and Contents of the storage referred to in the address part of the instruction	- for minus, + for plus; a decimal fraction

Storage, Regional

A0 thru A3

E0 thru E67

F0 thru F115

1F0 thru 1F21

2F0 thru 2F102

TOTAL 313 half words. A0 and E0 must be even.

CODED & CHECKED: P.E.H.

H. Kolsky
T-5

794 TRACING WITH OPTIONAL OPERATION OR ADDRESS-RANGE SELECTION

		R	A	B	C
STORAGE:	decimal	A (z)-	68-	2116-	3652-
		A 3	71	2119	3655
		E (z)-	(z)-	2048-	3584-
		E 67	67	2115	3651
		F (z)-	72-	2120-	3656-
		F 115	187	2235	3771
		1F (z)-	188-	2236-	3772-
		1F 21	209	2257	3793
		2F (z)-	210-	2258-	3794-
		2F 102	312	2360	3896
	octal	A (z)-	(104-	(4104-	(7104-
		A 3	107)$_8$	4107)$_8$	7107)$_8$
		E (z)-	(0-	(4000-	(7000-
		E 67	103)$_8$	4103)$_8$	7103)$_8$
		F (z)-	(110-	(4110-	(7110-
		F 115	273)$_8$	4273)$_8$	7273)$_8$
		1F (z)-	(274-	(4274-	(7274-
		1F 21	321)$_8$	4321)$_8$	7321)$_8$
		2F (z)-	(322-	(4322-	(7322-
		2F 102	470)$_8$	4470)$_8$	7470)$_8$
STOPS:	BREAKPOINT	F 67	$139_{10} = 213_8$	$2187_{10} = 4213_8$	$3723_{10} = 7213_8$
	OPERATION TRACE	F 14	$86_{10} = 126_8$	$2134_{10} = 4126_8$	$3670_{10} = 7126_8$
	ADDR TRACE { LOAD FIRST ADDR	F 17	$89_{10} = 131_8$	$2137_{10} = 4131_8$	$3673_{10} = 7131_8$
	ADDR TRACE { LOAD SECOND ADDR	F 20	$92_{10} = 134_8$	$2140_{10} = 4134_8$	$3676_{10} = 7134_8$

H. Kolsky
T-5

NO.	NAME
795 R	Tracing with traps for a one-bank or two-bank memory.

DEFINITION: A "trap" is a portion of the coder's program which is to be traced.

INPUT: The control cards are punched in binary in the 9 row as follows:

Trap Cards:

Col. 10 — no punch; M_i is in the first bank of the memory.

9 punch; M_i is in the second bank of the memory.

Cols. 15-26 — M_i = The location of the first instruction of the trap.

Cols. 33-44 — N_i = The location of the last instruction of the trap in either bank regardless of M_i.

Start Card:

Col. 9 — 9 punch

Col. 10 — no punch; R is in the first bank of the memory. Status to Frame 1.

9 punch; R is in the second bank of the memory. Status to Frame 2.

Cols. 15-26 — R = The first instruction of the coder's program.
R may be equal to any M.

All other columns must be blank.

SWITCHES: See DESCRIPTION below for a detailed explanation of 795 switches.

#1 ON (Down): If switch #3 is OFF then only those lines with operations 0, 1, 2, 3, and 4 will be printed.

OFF: Ignore

#2 Erase switch

on (down) 701 stops at F184. To erase trap which is being traced, push START. To ignore the erase order, transfer manually to F185.

off (up) 701 traces each trap which has been entered into the trap table.

#3 Print switch

on (down) 701 traces at full speed without printing and does not test switch #1.

off (up) 701 prints the listing described under OUTPUT as it traces after testing switch #1.

LOADING: Load 795 with 021 or 026 into the first bank of the memory.

Loading deck	# cards
021 or 026	1
795	9
Transition to 795 (02,F0)	1
Trap cards	n (1 for each trap)
Start card	1
Total	n + 12

STARTING: Put 795 tracing board in printer and have printer ready. Put the loading deck in hopper and have card reader ready. Set the Instruction Entry keys for 021 or 026. Press Card Reader Start, and then Load button. 795 will transfer control to instruction R in the coder's program.

DESCRIPTION: M_i is the first instruction to be traced in trap i, and N_i is the last instruction to be traced in trap i. As soon as instruction N_i has been traced, control will go to the coder's program and instructions will be executed at full speed without tracing until another trap (or the same trap) is encountered.

Traps are executed as follows: When 795 reads a control card, it replaces the instruction M_i with a transfer to a portion,

D_i, of the tracing program. 795 stores M_i, N_i and the contents of M_i in the D_i block. It then reads the i + 1st control card and repeats the procedure in the $D_i + 1$ block, continuing to read control cards until an R card is reached, whereupon control is transferred to R. Each D_i block is 6 half-words in length, hence the number of traps which may be specified is limited to the amount of space which is available in the machine for the trap table (D_i) block. (Normally, the D block follows the 795 code, but the coder may specify the D block by storing in OBO the address DO, which must be odd and equal to the first word address of the D block minus 1.)

SUGGESTIONS FOR SUCCESSFUL TRACING:

1. After a trap is set, be certain that M_i is not referred to by another instruction for information, or that M_i is not modified by another instruction before the actual tracing begins.
2. Be certain that 795 does not destroy necessary information.
3. Do not have the operating program destroy 795.

TRACING: When the coder's program reaches instruction M_i, control is transferred to 795. The contents of M_i are returned to M_i; the contents of the Accumulator, MQ register, overflow positions, and the status of the overflow indicator are preserved. Tracing continues through N_i, keeping the contents of the Accumulator, MQ, and overflow bits <u>after</u> the execution of each instruction, and the contents of the address <u>before</u> execution, printing out this information (see OUTPUT) if the print swicth (#3) is off. When a trap is first encountered, the paper is spaced and the contents of the Acc, MQ, and overflow bits before the execution of the first instruction of the trap are also printed out regardless of the position of the 1 or 3 switch. Read, Write and Read Backward instructions are "dummy" executed during the tracing program. Copy orders are "dummy" executed by loading the MQ with the contents of the memory location referred to in the copy instruction. All other instructions are executed. When instruction N_i has been traced, 795 loads the Acc, MQ, and overflow bits with their proper contents, sets the overflow indicator to its proper status, and sets the proper half-word status, before transferring to $N_i + 1$.

ERASING: If, while a trap, i, is being traced, the coder no longer needs to trace this trap, he may erase the trap by putting Switch #2 to its ON (down) position. The 701 will stop at F184. Put switch #2 up, and push start to erase the trap. The trap, i, cannot be traced again until a new control card for it is loaded into the machine. If, after putting switch #2 down, the coder does not wish to erase the trap, he should put switch #2 up, and transfer manually to F185.

STOPS:

F 184 Erase trap stop. Push start to erase trap; put switch #2 up.

F 156 Program stop or divide check (indicated by console light) in code being traced. Push start to continue.

OUTPUT: Print out consists of the following information, ten quantities per line, from left to right:

	Information	Converted to
(1)	Frame location of instruction	1 or 2
(2)	location of instruction	an octal integer
(3)	Instruction	
	sign	blank for plus, - for minus
	operation part	an octal integer
	address part	an octal integer
(4)	Half-word status	1 or 2
(5)	status of the ov indicator	ON for on, blank for off
(6)	overflow bits	0, 1, 2, or 3
(7)	sign and contents of the acc	blank or -; an octal integer
(8)	sign and contents of the MQ	blank or -; an octal integer
(9)	sign and contents of the first half-word of the storage location referred to in the address part of the instruction	blank or -; a 6-digit octal integer
(10)	sign and contents of the acc and overflow bits	blank or -; an integer plus a decimal fraction
(11)	sign and contents of the MQ	blank or -; a decimal fraction
(12)	sign and contents of the storage location referred to	blank or -; a decimal fraction; if a hlaf-word, the 5 right decimal digits will be zero.

RESTARTING: Have control cards in reader, and reader ready. Start at F0.

STORAGE: A0 thru A10

B0 thru B18

F0 thru F356

E0 thru E57

D0 thru D(6 · n), n = the number of traps.

Origin E0 must be even, <u>D0 must be odd</u>. 387 regional cards, 9 binary cards.

CODED: L. Gatt

795 Tracing with traps.

	R	A	B	C
STORAGE: decimal	A0-	58-	2106-	3642-
	A10	68	2116	3652
	B0-	69-	2117-	3653-
	B18	87	2135	3671
	F0-	88-	2136-	3672-
	F356	444	2492	4028
	E0-	0-	2048-	3548-
	E57	57	2105	3641
	D0-	445-	2493-	4029-
	D(6·n)	445+6n	2493+6n	4029+6n
octal	A0-	(72-	(4072-	(7072-
	A10	$104)_8$	$4104)_8$	$7104)_8$
	B0-	(105-	(4105-	(7105-
	B18	$127)_8$	$4127)_8$	$7127)_8$
	F0-	(130-	(4130-	(7130-
	F356	$674)_8$	$4674)_8$	$7674)_8$
	E0-	(0-	(4000-	(7000-
	E57	$71)_8$	$4071)_8$	$7071)_8$
	D0-	(675-	(4675-	(7675-
	D(6·n)	$675+6n)_8$	$4675+6n)_8$	$7675+6n)_8$
STOPS: Program stop in code being traced.	F 156	244_{10}	2292_{10}	3828_{10}
		$= 0364_8$	$= 4364_8$	$= 7364_8$
Erase trap	F 184	272_{10}	2320_{10}	3856_{10}
		$= 0420_8$	$= 4420_8$	$= 7420_8$

NO.	NAME
796R	Trace Logic (One- or two-bank machine).

INPUT: Control Card is punched in binary in the 9 row as follows:

Col 9	No punch, R is in first memory bank. 9 punch, R is in second memory bank.
Cols 10-14	blank
Cols 15-26	R = location of the first instruction to be traced.

LOADING:

Loading deck	# Cards
026	1
796	6
Transition to 796	1
796 Control Card	1
Total	9

STARTING: Put 796 board in printer and have printer ready. The contents of all the registers and condition of the overflow indicator are preserved on all types of entry.

a) Automatic entry with control card: Place the loading deck in the hopper and have card reader ready. Set the Instruction Entry keys for 026. Press card reader Start, and then Load button. Press Program Advance when Select Light on reader goes on. 796 will start to trace at R.

b) Manual entry with control card: (When 796 is already stored in E.S.) Put control card in hopper, press card reader Start. Start 701 manually at F0.

c) Entry by unconditional transfer: Load MQ with + 00, R (if R is in first memory bank) or - 00, R (if R is in second memory bank), then transfer to F10.

DESCRIPTION: 796 will trace the instructions beginning with the location given in the control card or MQ, printing out all transfer and "sense and skip" orders which are executed. Read, Write and Read Backward orders are not executed; 796 skips these orders. Copy orders are "dummy" executed by loading the MQ with the contents of the memory location referred to in the copy order. All other instructions, including ± sense 40_8 and ± 00, ± 01 transfers between banks 1 and 2, are executed.

OUTPUT: Location, sign, operation and address, in octal, of all executed transfer orders and "sense and skip" orders; 7 per line, 31 lines (double spaced) per printed page.

STOP: F71, a stop instruction has been encountered in programmer's code.

Push Start to print out last line of 7 orders.

STORAGE: Regional

E0 thru E49

A0 thru A8

B0 thru B10

F0 thru F200

R. Freshour 2/25/54

Corrected 12/12/54

796 Trace Logic		796R	796A	796B	796C
START: Transition card punched	decimal	F0	70	2118	3654
	octal	F0	106	4106	7106
STORAGE:	decimal	E0-	0-	2048-	3584-
		E49	49	2097	3633
		A0-	50-	2098-	3634-
		A8	58	2106	3642
		B0-	59-	2107-	3643-
		B10	69	2117	3653
		F0-	70-	2118-	3654-
		F200	270	2308	3854
	octal	E0-	(0-	(4000-	(7000-
		E49	$61)_8$	$4061)_8$	$7061)_8$
		A0-	(62-	(4062-	(7062-
		A8	$72)_8$	$4072)_8$	$7072)_8$
		B0-	(73-	(4073-	(7073-
		B10	$105)_8$	$4105)_8$	$7105)_8$
		F0-	(106-	(4106-	(7106-
		F200	$416)_8$	$4416)_8$	$7416)_8$
STOP: Stop instruction has been executed in programmer's code		F71	141 = $(215)_8$	= $(4215)_8$	= $(7215)_8$

Corrected 12/12/54

H. Kolsky
T-5

NO.	NAME
797	Tracing with traps for a one bank 701.

Refer to 795-R write-up for explanation of tracing with traps. The same trap and R cards are used as for 795, and the output is identical to that of 795. The switches operate in the same fashion.

In addition, 797 will suppress tracing after executing the two instructions

α	10	α
$\alpha + 1$	3	xxxx

Tracing will then begin automatically at the next non-stop instruction following $\alpha + 1$.

For example, consider the code:

loc.		op.	address
1000	+	10	1000
1001	+	03	3000
1002	-	00	4444
1003	+	00	3333
1004	-	12	2222

After printing the line for 1000, 797 will locate 1004 as the first non-stop instruction. It will then replace 1004 with a transfer to 797 saving {-12 2222} for later replacement and execution. Then control is sent to 3000. The 701 then proceeds with full speed through program (not high-speed tracing). When control is at 1004, tracing will start as though a new trap had been encountered. No extra trap storage is necessary for this device.

Further suggestion for 795 and 797: Do not load the same trap cards more than once without reloading your program.

Coded: L. Gatt 10/14/54

		R	A	B	C
STORAGE:	decimal	A0-	348-	2396-	3932-
		A31	379	2427	3963
		D0-	379-	2427-	3963-
		D0+(6·n)	379+6n	2427+6n	3963+6n
		E0-	0-	2048-	3584-
		E55	55	2103	3639
		F0-	50-	2098-	3634-
		F297	347	2395	3931
	octal	A0-	534-	4534-	7534-
		A31	573	4573	7573
		D0-	573-	4573-	7573-
		D0+(6·n)	573+6n	4573+6n	7573+6n
		E0-	0-	4000-	7000-
		E55	67	4067	7067
		F0-	62-	4062-	7062-
		F297	533	4533	7533
STOPS:	Program stop in code being traced.	F122	172_{10}	2220	3756
			254_8	4254_8	7254_8
	Erase trap	F147	197_{10}	2245	3781
			305_8	4305_8	7305_8

Note: If trap is not to be erased after depressing switch #2, transfer

	R	A	B	C
manually to:	F148	198_{10}	2246	3782
		306_8	4306_8	7306_8

NO.	NAME

798 Tracing with traps a one-bank program with 798 in the second bank. This utility program was written particularly for people who have large one-bank codes and find it difficult to leave room for a tracing program. It will work provided the code being traced does not contain any negative transfers or any references to negative odd addresses. Enough room (58 half-words) must be available in ES-1 for 028 which is used to load 798.

For explanation of tracing with traps, refer to the 795 write-up. 798 uses the same trap and R cards as 795 and the output is the same. The switches operate in the same fashion. For explanation of suppressing tracing after a calling sequence, see the 797 write-up. 798 operates in the same way.

If a person anticipates that he will want to trace after running his code for a ways, he should set the machine for 2-bank operation before he starts so that the second bank will be available for 798.

LOADING:

deck	# cards
028	2
798	9
Tr to 798	1
Trap cards	n (1 for each trap)
Start card	1
Total	13 + n

The machine must be set for two-bank operation.

STOPS:

350_8 Program stop in code being traced. Push "start" to continue.

402_8 Erase trap. Put up switch #2 and press "start".

If trap is <u>not</u> to be erased after depressing switch #2, put up switch #2 and transfer manually to 403_8 in ES-2 by entering in the instruction entry keys -010403.

Coded, checked & written: D. Solbrig, 3/18/55

820 Check binary cards for proper check sum without destroying memory.

LOADING: 820 is self loading:

	no. of cards
820	1
binary cards to be checked	n
transition if wanted	1
	n + 2

DESCRIPTION: 820 will check the check sum and half-word count of any number of binary cards without loading the binary cards into memory. That is, only 46 half-words are used by the entire deck of n + 2 cards mentioned in the loading process. (Not equivalent to loading.)

(This is useful, especially immediately after making a card dump to see if the dump is good before leaving the 701 or before proceeding with further calculation.)

820 will check one card at a time; if it finds the check sum and half-word count correct, it will proceed to read another card etc.

820 will stop at F.42 on a check sum disagreement.

Press the start to continue checking or transfer to F.4 of your 224 program to punch a new card and then continue checking. Note: 820 sets up the calling sequence to 224 for every card read in.

STORAGE: F.0 to F.45

Octal location

F.0	0000	1000	...	7000
F.42	0052	1052	...	7052

Coded, written & checked: L. Gatt, 2-24-55

NO.: 924 R CROSS REFERENCE: 324 R

NAME: Dump-Load Using Tape

DESCRIPTION: 924 causes a specified block of full words to be written on tape 256 and then sends control to a specified location. When desired it brings this same block back into its original space in E.S. by means of a self-loading feature and then sends control to a specified location.

LOADING: In the card reader hopper place the following deck:

(a) 021 (or 024, or any other equivalent self-loading program)

(b) 924

Press card reader start button until ready light is on

Set instruction-entry keys to address required by self loading program used in (a) (this will be zero if 021 A is used)

Set automatic-manual switch to automatic

Set load-selector switch to cards

Press load button

Press start button on card reader when it has stopped with last card halfway in

CONTROL CARDS: A. Dumping control card

This is punched in binary in the 9 row

Columns	
15-26	R
33-44	L
50	0 or 1
51-62	M
68	0 or 1
69-80	N

Where - R = Location of first full word to be dumped

- L = Location of last full word to be dumped

M = Location where machine goes for its next instruction after dumping

N = Location where machine goes for its next instruction after loading

If a zero appears in column 50 (or 68), the machine will stop before going to M (or N) and will go to M (or N) when the start button is pressed. If a one appears in column 50 (or 68) the machine will go to M (or N) without stopping.

B. Transition card

This is punched in binary in the 9 row

Columns	
14	1
15-26	OF1

The actual value of OF1 depends on the location of 924.

OPERATION: A. To dump using control card

1. Assuming 924 is already in the machine

Put control card in card reader hopper and press card reader start button until ready light is on

Start control at OF1 as follows:

Set automatic-manual switch to manual

Set instruction-entry keys to OF1

Press enter-instruction button

Set automatic-manual switch to automatic

Press start button

2. **Assuming 924 is not in the machine and a dump is desired immediately after loading 924**

   In the card reader hopper place the following deck:

   (a) 021 (or 024, or any other equivalent self loading program)

   (b) 924

   (c) Transition card

   (d) Dumping control card

   Then continue as described in "loading"

B. To dump using stored program with calling sequence

The following calling sequence is written in program where dump is desired:

```
        ⋮
a       R ADD a
a + 1   TR OF10
a + 2   [0, R]
a + 3   [0, L]
a + 4   [0 or 1, M]
a + 5   [0 or 1, N]
        ⋮
```

Where R, L, M, and N are as defined under "control cards".

C. To load using control panel

Start control at 2F0 as follows:

Set automatic-manual switch to automatic

Set load-selector switch to tape

Set instruction-entry keys to 2F0

Press load button

D. To load using stored program

The following is written in program where loading is desired:

```
     ⋮
     a      Rewind 256
 a + 1      Read 256
 a + 2      - Copy 2F0
 a + 3      TR 2F0
     ⋮
```

The instruction in a is needed only if tape 256 has been disturbed since the dump

STORAGE:

Regions	Parity	
0F1 - 0F115	0F1	Odd
1F0 - 1F69	1F0	Either
2F0 - 2F71	2F0	Even
0E0 - 0E9	0E0	Even

Storages actually used in operation

When dumping: 0F1 - 0F115
1F0 - 1F69
0E0 - 0E9

When loading: 2F0 - 2F71
0E0 - 0E5

If it is desired to use this program more than once without reloading it each time, then 0F1 - 0F115 and 1F0 - 1F69 must be saved. 0E0 - 0E9 and 2F0 - 2F71 may be used as erasable storage.

STOPS:	Loc	Contents	Meaning
	0F109	00,0F26	Either the 2F program or the block being dumped was not written on tape correctly. Press start button to try again.
	2F23	00,2F28	The 2F program did not load itself correctly. Press start button to try again.
	2F54	00,2F61	The block being loaded was not read in correctly. Press start button to try again.

SPECIAL INSTRUCTION FOR ASSEMBLING WITH SO2:

After assembling in the usual way with SO2 using relocation cards for the 0F, 1F, 2F, and 0E regions, one must punch a card which will store a number, whose magnitude depends on the location of 924, in the full word location which was -0F114 before relocation. This full word may be found by activating a dump with the relocated 924. The machine will stop at the location which was 0F109 before relocation, and the desired full word will appear in the accumulator. It should be copied down and punched on a card in binary as follows:

ROW	COLUMNS	
9	9 - 44	Check sum for this card = - 2 (binary integer punched in 9-row columns 45 - 62 + binary integer punched in 9-row columns 63 - 80 + binary integer punched in 8-row columns 9 - 26 + binary integer punched in 8-row columns 27 - 44) Note that if a punch appears in columns 9 or 27 8-row it is to be considered a binary bit, not a sign.
9	61	Punch
9	69 - 80	That number into which OF114 has been transformed by the relocation
8	9 - 44	The full word, with its sign which was copied out of the accumulator.

This card should be kept permanently as the last card of the 924 binary deck.

CODED: W. A., written W.A.

H. Kolsky
T-5

NO.	NAME
925	Reproduce binary cards with correct check sum.
INPUT:	Binary cards with or without check sum. 925 does not reproduce regional binary cards correctly.
LOADING:	925 is self-loading into 0000

Load deck	# Cards
925	1
binary cards to reproduce	n
Total	n + 1

STARTING:	Load selector to cards. Set instruction entry keys to zero. Set automatic-manual switch to Automatic. Press load button. It is not necessary to reset and clear memory.
DESCRIPTION:	925 will read in one binary card, compute its check sum, and punch a new card with correct check sum.
STOP:	[0015] End of file: Program finished.
OUTPUT:	n binary cards with correct check sum.
STORAGE:	The first 96 half-words of memory.

CODED & CHECKED: Lou Gatt

NO.	NAME
926	Reproduce regional binary cards with correct check sum.
INPUT:	Regional binary cards with or without check sum. 926 does not reproduce binary cards correctly.
LOADING:	926 is self-loading into 0000 after resetting and clearing memory.

Load deck	# cards
926	1
regional binary cards to reproduce	n
Total	n + 1

STARTING: Load selector to cards. Set instruction entry keys to zero. Set automatic-manual switch to Automatic. Reset and clear memory. Press load button.

DESCRIPTION: 926 will read in one regional binary card, compute its check sum, and punch a new card with correct check sum.

STOP: 0014 End of file: Program finished.

OUTPUT: n regional binary cards with correct check sum.

STORAGE: All of memory (reset and cleared).

CODED & CHECKED: Lou Gatt

NO.	NAME
982	Prints contents of electrostatic memory in octal. Destroys only the first two full words, leaves the rest of E.S. unchanged. Use with 526.

INPUT:

Loading deck	# Cards
526	4
021A	1
982	5
Transition to 982	1
TOTAL	11

STARTING: Automatic entry: Put loading deck in the hopper of the card reader. Have card reader ready. Set instruction keys to zero, and press the load button. Press card reader start when 701 stops on last card. There is no manual entry. There is no entry by transfer.

DESCRIPTION: 526 writes all of electrostatic memory on drum #1 with the exception of the first two full words -0000 and -0002. 021A loads 982. 982 reads the first half of the drum into electrostatic and searches for the first half-word not plus zero nor minus all ones. Having found such a half-word, it will print the location of this half-word, the half-word itself and ten consecutive half-words, whatever they may be on one line. Searching and printing continues till the second half of the drum is read into electrostatic and treated like the first half. After 982 has completed the search and printed what was found, the drum is read back into electrostatic, unchanged except that the two full words -0000 and -0002 have been destroyed.

PROGRAM STOP:	Instruction Counter	Meaning
	1	Search is complete

OUTPUT: Printed sheets, eleven octal instructions, and the location of the first instruction per line.

CODED: C.E.M., checked and written, C.E.M.

983 Print sections of electrostatic memory by means of control cards or MQ Entry buttons.

INPUT: Control cards with starting addresses are punched as follows: 9 row, columns 15-26 contains in binary the starting address (1 address per card).

LOADING:

Loading deck	# Cards
526	4
706	1
026	1
983	7
Tr 983	1
Control cards	n
Total	n + 14

STARTING: Put the 186 board in the printer and have printer ready. Place loading deck in hopper and have card reader ready. Set instruction keys for zero and press load button on the console. See description below for entry by control card or MQ Entry keys.

1. Entry by control card.

Punch up the control cards with each starting address. Put these control cards behind the 983 transition card. Put Sense Switch #1 down and start 983. 983 will read the first control card and start printing half words, that are not zero or minus ones, at the address on the first control card. When you have printed all that you want at the first starting address, push the reset button on the console and then the start button. 983 will read the next control and repeat the process of printing, starting with the new address. Continue pushing the reset and start buttons as in the procedure above until all the control cards are read. When 983 reaches the end of memory, the drum is read back into electrostatic storage. Therefore, it is best to arrange your control cards so that the address nearest 4095_{10} is the last one to be read.

2. Entry by MQ button.

983 will stop at 151_8. Load manually the first location into the address part of the MQ. Push the start button on the console. When you have printed what you want with this location, press the reset button. This puts you at zero. Load the next address manually into the address part of the MQ and push the start button. Continue with the above procedure until you have entered all the addresses that you want. 983 will print to the end of memory unless you stop it, and then will read back information from the drum. Therefore, it is best to enter the address nearest 4095_{10} last.

3. If you want to start printing at 4 as 982 does, you do not need to enter anything manually in the MQ, but instead just push the start button when you get a stop at 151_8.

4. You may start 983 with a control card and then continue by using the MQ Entry procedure. Do not put Sense Switch #1 down if you use this method.

DESCRIPTION: 983 does the same as 982 except you have the option of starting to print memory wherever you want to, and you may print several sections without having to print what you don't need. 983 will not print zeroes or minus ones unless they occur between half words that are printed. 983 will read information on the drum back into electrostatic storage after it reaches 4095_{10}. Full words zero and two are destroyed.

PROG. STOP: 151_8 Enter the first word address into the MQ or push the start button if you want to start printing at four.

1 Search is complete and the drum has been read back to electrostatic storage.

OUTPUT: Printed sheets, eleven octal instructions and the location of the first instruction per line.

CODED: M. F. Anderson, checked & written, M. F. Anderson
12-13-54

NO.	NAME
991R	Read 10 digit decimal numbers, convert them to binary, scale them, and punch them out in a form suitable for loading with 021.
INPUT:	Any number of constants, one per card, may be converted by 991R. Each block of constants with the same scaling must be preceded by a control card punched, in decimal, as follows:

Columns 9	9
10-11	0
12-13	p, the position of the decimal point (from the left)
14-15	q, the position of the binary point (from the left)
16-19	R, the initial address into which the first binary output card of the group is to be loaded. R must be even

For a given p, q must not be less than $\bar{q}$ in the following table.

p	$\bar{q}$
0	0
1	4
2	7
3	10
4	14
5	17
6	20
7	24
8	27
9	30
10	34

The constant card must be punched, in decimal, as follows

Columns 9	0
10-19	10 decimal digit constant
19	11 punch for a negative constant no punch for a positive constant

LOADING: Load 991 with 021. See 021 for complete loading instructions.

Loading Deck	# Cards
021	1
991	7
Transition to 991	1
Control card	1
Block of constants	n
Control card	1
Block of constants	n

Etc.

DESCRIPTION: 991 first checks to see that the first card read in is a control card. If it is not a control card, the machine will stop at F159. Pushing the Start button will cause the machine to read the next card and test again to see if it is a control card. The constants following each control card are converted to binary and scaled according to the p and q on the control card. The resulting binary constants are punched out, up to 22 per card with an S, V and R for that card punched in the 9 row. The R for the first output card of the block is the R of the control card. 991 also checks columns 9 - 19 for double punches and blank columns.

OUTPUT: Binary cards suitable for loading with 021, with S, V, and R in the 9 row. Rows 8 thru 12 contain the scaled binary constants, up to 22 per card.

STORAGE: E0 thru E51, E0 even

A0 thru A2

N0 thru N27, N0 even

G0 thru G41

F0 thru F181, F0 even

255 regional cards, 7 binary cards.

STOPS:

F55	Columns 10-19 are double-punched or have a blank column.
F60	Column 9 does not have a 0 or a 9 punch.
F158	Column 9 is double-punched

In each of these cases, remove the card from the reader, correct it, and place it in the card reader. Have the card reader ready. If the card is the first control card, start the machine manually at F0. If it is not the first control card, or is a data card, start the machine at F8.

F147	Q on the control card is less than the legal $\overline{Q}$. Push Start to recompute Q. If the machine stops again at F147, remove and correct the card and start as above.
F157	End of file, all constants scaled and punched.
F159	First card not a control card. Push start to search for control card. Machine will continue to stop here until a control card is found.

CODED: Scully 6/53

		991R	991A	991B	991C
991 Read decimal constants, scale, and punch in binary					
START: Transition card punched:					
	decimal	F0	128	2176	3712
	octal	F0	200	4200	7200
STORAGE:	decimal	E0-	0-	2048-	3584-
		E51	51	2099	3635
		A0-	54-	2102-	3638-
		A2	56	2104	3640
		N0-	100-	2148-	3684-
		N27	127	2175	3711
		G0-	57-	2105-	3641-
		G41	99	2147	3683
		F0-	128-	2176-	3712-
		F181	309	2357	3893
	octal	E0-	(0-	(4000-	(7000-
		E51	$63)_8$	$4063)_8$	$7063)_8$
		A0-	(66-	(4066-	(7066-
		A2	$70)_8$	$4070)_8$	$7070)_8$
		N0-	(144-	(4144-	(7144-
		N27	$177)_8$	$4177)_8$	$7177)_8$
		G0-	(71-	(4071-	(7071-
		G41	$143)_8$	$4143)_8$	$7143)_8$
		F0-	(200-	(4200-	(7200-
		F181	$465)_8$	$4465)_8$	$7465)_8$
STOPS:	Cols. 10-19 BCDP	F55	183 $(0267)_8$	2231 $(4267)_8$	3767 $(7267)_8$
	Col. 9 not 0 or 9	F147	188 $(0274)_8$	2236 $(4274)_8$	3772 $(7274)_8$
	Q less than $\overline{Q}$	F147	275 $(0423)_8$	2323 $(4423)_8$	3859 $(7423)_8$
	End of file	F157	285 $(0435)_8$	2333 $(4435)_8$	3869 $(7435)_8$
	Col. 9 is double punched	F158	286 $(0436)_8$	2334 $(4436)_8$	3870 $(7436)_8$
	First card is not a control card	F159	287 $(0437)_8$	2335 $(4437)_8$	3871 $(7437)_8$

H. Kolsky
T-5

NO.	NAME
992 R	Read 10 digit decimal numbers up to 7 per card, convert them to binary, scale them, and either store them in specified blocks of E.S. or punch them out in a form suitable for loading with 026. (Use for full word data only).
INPUT:	Any number of constants, up to 7 per card, may be converted by 992 R. Each block of constants with the same scaling must be preceded by a control card punched in decimal as follows:

Columns	10	11 punch
	10 - 11	p, the position of the decimal point (from the left).
	12	0
	13 - 14	q, the position of the binary point, (from the left).
	15	0
	16 - 19	R, the initial address of the block. R must be even.

For a given p, q must not be less than $\bar{q}$ in the following table:

p	$\bar{q}$
0	0
1	4
2	7
3	10
4	14
5	17
6	20
7	24
8	27
9	30
10	34

The constant card must be punched in decimal as follows:

Columns	10 - 19	1st constant
	19	11 punch if constant is negative
	20 - 29	2nd constant
	29	11 punch if minus
	30 - 39	3rd constant
	39	11 punch if minus
	40 - 44	1st part of 4th constant
	45	blank
	46 - 50	2nd part of 4th constant
	50	11 punch if minus
	51 - 60	5th constant
	60	11 punch if minus
	61 - 70	6th constant
	70	11 punch if minus
	71 - 80	7th constant
	80	11 punch if minus

If less than 7 constants are to be converted, the rest of the input card should be left blank. If zeros are punched in, they will be loaded.

LOADING: Load 992 with 021 or 026.

Loading Deck	# Cards
026	1
992	9
Transition to 992	1
Control Card	1
Block of constants	n
Control Card	1
Block of constants	n
	etc.

Switch #1 — Up: 992 will store the scaled constants in blocks of E.S. specified by the control card.

Down: 992 will punch out the scaled constants in form suitable for loading with 021 or 026.

DESCRIPTION: 992 first checks to see that the first card read in is a control card. If it is not a control card, the machine will stop at F109. Pushing the start will cause the machine to read the next card and test again to see if it is a control card. The constants following each control card are converted to binary and scaled according to the p and q on the control card. If switch #1 is up, the resulting binary constants are stored in E.S. from R to R + n - 1. If switch #1 is down, the binary constants are punched out, up to 22 per card with an S, V and R for that card punched in the 9 row. The R for the first output card of the block is the R of the control card. 992 checks columns 10 - 44 and 46 - 80 for blank columns and double punches.

STORAGE:

E0 thru E72, E0 even

A0 thru A10, A0 even

N0 thru N68, N0 even

F0 thru F267

STOPS:

F101 — Q on the control card is less than the legal $\bar{Q}$. Push start to recompute Q. If machine stops again, remove and correct card, and start at F2.

F83 Control card is BCDP. Remove and correct card and start at F2.

F109 First card not control card. Press start to read another card. Machine will continue to stop here until a control card is found.

F164 Data card BCDP. Remove and correct card and start at F2.

F242 EF stop.

CODED: Freshour 11/53

		992R	992A	992B	992C
992	Read 10 digit decimal numbers, convert to binary, scale & store in E.S. or punch out.				
LOADING CARD:			026A	026B	026C
PROGRAM STOPS:	Q on control card less than legal $\overline{Q}$	F101	400_8	4400_8	7400_8
	Control card is DPBC	F83	356_8	4356_8	7356_8
	First card not control card	F109	410_8	4410_8	7410_8
	Data card DPBC	F164	477_8	4477_8	7477_8
	EF stop	F242	615_8	4615_8	7615_8
STORAGE:	decimal . . .	A0	74	2122	3658
	thru	A10	84	2132	3668
		N0	86	2134	3670
	thru	N68	154	2202	3738
		F0	155	2203	3739
	thru	F267	422	2470	4006
		E0	0	2048	3584
		E73	73	2121	3657
	octal . . .	A0	112	4112	7112
	thru	A10	124	4124	7124
		N0	126	4126	7126
	thru	N68	232	4232	7232
		F0	233	4233	7233
	thru	F267	646	4646	7646
		E0	0	4000	7000
	thru	E73	111	4111	7111

Kolsky

RC-5

PURPOSE: Load itself, load binary full-words into consecutive E.S. locations from binary cards with card check sum <u>either</u> per card <u>or</u> per block of cards.

INPUT: First card of block or Each card } must contain in binary in the nine row:

Columns	Content
9-44	S, card check sum for { the block. / that card. }
51-62	V (even), half word count of { the block. / that card. }
69-80	R (even), the location of the first full word to be loaded. $R \geqslant 0060_8$

LOADING: RC-5 is self-loading.

STARTING: Place RC-5 followed by cards to be loaded in hopper, and have card reader ready. Set instruction keys to zero; press load button. Press start on card reader when card reader stops on last card. If RC-5 is already in E.S., put binary deck in hopper, have card reader ready, and start the 701 manually at 0006.

DESCRIPTION: The binary full words of each card or block of cards are read and stored in E.S. Then each word is called from its location and summed to check that the check sum just computed agrees with that from the card. RC-5 may be modified to transfer on end of file by inserting a transfer order in 0011_8.

PROGRAM STOPS:

Location	Meaning
0011_8	End of file condition---all cards are loaded.
0053_8	Check sum error. Difference between S on card and the computed check sum is in the accumulator. Press start to continue loading.

STORAGE: 0000 - 0057_8

CODED: Ruth Clark

PURPOSE: Punch in binary consecutive full words from E.S.

INPUT: Entry by basic linkage as follows:

α	+ R ADD	α	
α + 1	+ TR	OFO	
α + 2	+ STOP	V	Half word count (even)
α + 3	+ STOP	R	Unloading address (even)
α + 4	+ STOP	R'	Reloading address (even)

α + 5 Control returns here upon completion of the program.

LOADING: Load with 026 or RC-5.

DESCRIPTION: RC-8 will punch in binary the full words from -R thru -(R + V - 2) in E.S. Each card contains a card check sum S, V, and R in the nine row where R is the reloading address. RC-8 will punch data from locations -R thru -(R + V - 2) in E.S. to be reloaded into locations -R' thru -(R' + V - 2). R = R' if words are to be reloaded into the same location.

PROGRAM STOPS: None

STORAGE: Regional: OFO - OF59

OBO (even) - OB9

OAO - OA2

Total: 73 half words

CODED: Ruth Clark

PURPOSE: Read from tape or drum; or write on tape, drum, or cards. (full words)

INPUT: Entry by basic linkage as follows:

α + R ADD α

α + 1 + TR OFO

α + 2 + Select Input order*

α + 3 + STOP V (even), Half word count

α + 4 + STOP R (even), Location of first full word.

α + 5 + STOP S (even), Set drum address.

α + 6 Control returns here after completion of program.

*Select input order is one of the following:

WRITE {
- any tape identification
- any drum identification
- punch identification

or

READ {
- any tape identification
- any drum identification

LOADING: Load with 026 or RC-5

DESCRIPTION: RC-9 will write a unit record of consecutive full words on tape or drum. A storage check sum is stored on the tape or drum as the first full word of the unit record. RC-9 will read a unit record (of full words) with storage check sum as the first full word from tape or drum. Cards are punched with card check sum S, V, and R in the nine row of each card. RC-9 will not "punch blank cards", i.e. RC-9 tests for the first non-zero full word and punches a full card (or part of a card if the remaining number of half words in the record is less than 44) starting with that location. After punching that card, it again tests for the next non-zero full word, etc.

The coder must make sure that the tape called on is in the proper status and rewound if necessary.

PROGRAM STOPS:

Location	Meaning
0F68	Storage check sum on tape or drum does not agree with that computed. (Only occurs when reading.)

STORAGE: Regional: 0F0 - 0F108

0B0 (even) - 0B9

0A0 - 0A2

Total: 122 half words.

CODED: Ruth Clark

PURPOSE: Print decimal data (7 ten-place decimal numbers per line).

INPUT: Entry by basic linkage as follows:

α	+ R ADD	α	
$\alpha + 1$	+ TR	OFO	
$\alpha + 2$	+ STOP	L	number of lines
$\alpha + 3$	+ STOP	R	location of first word to be printed
$\alpha + 4$	Control returns here after L lines are printed		

LOADING: Load with 026 or RC-5.

STARTING: Put RC-10 board in the printer.

DESCRIPTION: RC-10 prints L lines, seven ten-place decimal numbers per line, from full word data stored consecutively in E.S. Scaling and provision that the converted full word is not an eleven-place decimal number are left for the coder. RC-10 uses integer-type conversion. A number will be printed with p decimal places if the coder multiplies the number by $2^{-q} \cdot 10^{p}$ where $q = 35 - t$ = the number of binary places to the right of the binary point and p is the number of decimal places to right of the decimal point. (t is defined the same as the t_i in 110.) RC-10 does not restore paper. Paper can be restored by giving the following two orders: + WRITE 0512_{10} followed by + SENSE 0521_{10}.

PROGRAM STOPS:

Location	Meaning
3F14	Error in signs
3F31	Error in left-half digits
3F41	Error in right-half digits

Pushing start button will cause program to try the L lines again if any of these stops occur.

STORAGE: Regional:

0F0 - 0F46
1F0 - 1F42
2F0 - 2F43
3F0 - 3F42
4F0 - 4F20
0B0 - 0B16
1B0 - 1B4
0J0 (even) - 0J1
0A0 (even) - 0A2
0N0 (even) - 0N-49

Total: 275 half words.

CODED: Ruth Clark

PURPOSE: Read any unit record of full words with storage check sum (as first full word) from any tape.

INPUT: Entry by basic linkage as follows:

α	+ R ADD	α	
$\alpha + 1$	+ TR	OFO	
$\alpha + 2$	*	R	first word location in E.S.
$\alpha + 3$	Control returns here.		

*The operation part specifies which tape:

00 (STOP) indicates tape 0256_{10}

01 (TR) indicates tape 0257_{10}

02 (TR OV) indicates tape 0258_{10}

03 (TR +) indicates tape 0259_{10}

LOADING: Load with RC-5 or 026.

DESCRIPTION: RC-11 will read any unit record of full words with storage check sum as the first full word of the record. The coder must make sure the tape is in the proper status and position.

PROGRAM STOPS:

Location	Meaning
OF37	Error in storage check sum.

STORAGE: Regional: OFO - OF37

OKO (even) - OK5

OAO - OA2

Total: 47 half words

CODED: Ruth Clark

H. Kolsky
T-5

LCH 10

Nov. 23 - This is the replacement for the old LCH 10 writeup.

7 FULL OR 14 HALF WORD DATA LOADING

PURPOSE: To load blocks of either full or half word data, punched 70 decimal digits per card. The initial loading address, input scaling and block lengths are as specified by heading cards.

STORAGE:

PROGRAM:	$(54)_{10}$	to	$(359)_{10}$
	$(66)_8$		$(547)_8$
ERASABLE:	$(00)_{10}$		$(42)_{10}$
	$(00)_8$		$(52)_8$

USE: FULL WORDS:

HEADING CARD

Card Columns	Punch In Decimal
9	11
10	0
11 - 14	Initial Loading Address
15 - 17	000
18 - 19	P
20 - 22	000
23 - 24	Q
25	0
26 - 29	Halfword Count Of Block

DATA CARD

Card Columns	Punch
9	BLANK
10 - 19	1st Full word

20 - 29	2nd Full word
30 - 39	3rd " "
40 - 44	4th " " (1st five digits)
45	BLANK
46 - 50	4th Full word (last five digits)
51 - 60	5th " "
61 - 70	6th " "
71 - 80	7th " "

Signs are punched over last digit of each word, an 11 for minus, a 12 for plus.

<u>HALF WORDS</u>

HEADING CARD

Card Columns	Punch In Decimal
9	12
10	0
11 - 14	Initial Loading Address
15 - 17	000
18 - 19	P
20 - 22	000
23 - 24	Q
25	0
26 - 29	Half word Count of Block

DATA CARD

Card Columns	Punch
9	BLANK
10 - 14	1st Halfword
15 - 19	2nd "
20 - 24	3rd "
25 - 29	4th "
30 - 34	5th "
35 - 39	6th "
40 - 44	7th "
45	BLANK
46 - 50	8th Halfword
51 - 55	9th "
56 - 60	10th "
61 - 65	11th "
66 - 70	12th "
71 - 75	13th "
76 - 80	14th "

Place FEJ 035, followed by binary cards followed by decimally punched cards in hopper. Press load button.

STOPS:

1. $(116)_8$ End of file.

2. $(255)_8$ DPBC error left half sign. Re-load corrected card and press start.

3. $(263)_8$ DPBC error right half sign. Restart as in 2.

4. $(267)_8$ DPBC error left half digits. Restart as in 2.

5. $(273)_8$ DPBC error right half digits. Restart as in 2.

6. $(276)_8$ Missing heading card. Could be caused by:

    1. No heading card to start first block.

    2. More cards in block than possible for half word count specified.

7. $(426)_8$ DPBC error heading card. Restart as in 2.

NOTES: Program may be used with calling sequence by storing the exit address in $(220)_{10}$- $(332)_8$ for full words or in $(240)_{10}$- $(360)_8$ for half words. To activate card reading, transfer to $(52)_{10}$- $(64)_8$. Exit will be made after specified number of half words have been read.

LCH 11

BINARY PUNCHING

Nov. 23 - This is the replacement for the old LCH 11 writeup.

PURPOSE: To be used in conjunction with LCH 10, in order to punch, in binary, decimal data loaded by LCH 10. The binary cards produced may be loaded with 021, FEJ 035, 026, etc.

STORAGE: $(338)_{10}$- $(410)_{10}$

$(522)_8$- $(632)_8$

USE: Place the five binary cards of LCH 11 between the 7th and 8th cards of LCH 10. Punching will take place after each block of Data is read in.

Starts at $(64)_8$.

H. Kolsky
T-5

Dual for ES-1 and/or 2

The present Dual system may be used on a 2 bank machine if the following restrictions are observed:

1. All orders to be interpreted by Dual must be in ES-1. (This includes orders to be traced and calling sequences.) Full words may be in either ES-1 or 2.

2. Dual must be in ES-1.

3. Half-word status must be in ES-1 upon entry to Dual or the print program.

4. The floating point print calling sequence must be changed as follows:

α	+ R Add	α + 2	
α + 1	+ tr	75F0	
α + 2	+ R Add	α + 2	
α + 3	- tr	75F0	
α + 4	+ n	FWA	(even or odd)
α + 5	+ Stop	LWA + 2	(even or odd)

Two half-words have been added to enable Dual break-point and floating trace to operate. These have been located at 3024_{10} and 3025_{10}, 5720_8 and 5721_8.

I. J. Cherry

H. Kolsky
T-5

Dual for ES-1 and/or 2

CORRECTION:

Six half-words have been added to enable Dual break-point print and floating trace to operate. These are $(69F0 - 69F2)_R$, and $(70F0 - 70F2)_R$, $(3019 - 3025)_{10}$, $(5714 - 5721)_8$.

Dual Trace After Print.

Trace modification #3.

Description:

This modification allows the resumption of fixed point tracing immediately following a block print. It consists of five binary cards to follow the standard Dual trace cards.

Storage in addition to Dual Trace:

$(73F0170\text{-}73F0179)_R$, $(0222\text{-}0231)_{10}$, $(0336\text{-}0347)_8$.

Dual Trace Fixed Transfers.

Trace modification #4.

Description:

The floating point trace is not disturbed. The fixed point trace prints all transfers and only transfers. It consists of five binary cards to follow the normal trace or dual trace modification #3.

Storage in addition to Dual Trace:

$(73F0180\text{-}73F0181)_R$, $(0232\text{-}0233)_{10}$, $(0350\text{-}0351)_8$.