IBM 650 PUBLICATIONS - GENERAL INFORMATION

This bulletin is the first of a new series in the IBM 650 area. Each bulletin in the series will be classified in accordance with the new scheme for technical publications: "A" for Applications information, "M" for Machine information, and "S" for Systems.

The new series will replace the existing "IBM 650 Bulletins" series. Some of the information presently contained in 650 Bulletins 1 through 20 will be republished as new "S" bulletins. Other items in the existing series will be incorporated in appropriate "M" bulletins now in preparation. Certain other information is considered obsolete and will not be reprinted.

The publication of abstract pages for the manual of "Library Program Abstracts for the IBM 650" will be discontinued in the new bulletin series. This material will be published and distributed separately in a different format.

IBM 650 DATA PROCESSING SYSTEM BULLETIN

SOAP IIA

SOAP IIA, a major modification of SOAP II, is a multiple machine pass assembly program. By means of the multiple pass feature, programs containing any number of symbols can be assembled without sectioning, prior to processing, as is required when programs using more than 400 symbols are assembled by SOAP II.

Two versions of the program are available: The Basic SOAP IIA, which is designed for program assembly with a basic IBM 650 Data Processing System; and the Tape SOAP IIA program, designed for program assembly with a 650 Tape System. Both versions can assemble programs written for any configuration of the 650 system; they utilize the same basic processing methods, and differ mainly in speed, input-output procedures, and system requirements. The Tape SOAP IIA version is the faster of the two, of course, since IAS can be used for storage of frequently used constants and program loops and for temporary storage locations.

In the following discussion it should be understood that unless a particular item is attributed to one or the other version of the program, that item applies equally to the Basic SOAP IIA and the Tape SOAP IIA. The information contained in this bulletin is intended as a supplement to the SOAP II Reference Manual, C28-4000, (formerly 32-7646) and describes only those features which represent changes or additions to the SOAP II program. Any item not mentioned should be assumed to be the same as in the SOAP II program, and relevant information may be found in the above cited manual.

Briefly the new features of SOAP IIA are:

1. The multiple pass feature is completely automatic except for a limited amount of card handling required with the Basic SOAP IIA.

2. Any program which can be assembled with SOAP II is acceptable as input to SOAP IIA, provided that certain punches do not appear in column 80 of the input cards (see page 2).

3. The symbol table can be punched out by means of a new pseudo-operation code and/or the setting of the 650 console Sign switch.

4. Regional definitions with FWA (first word address) greater than 1999 are acceptable.

5. The use of a new card "type" (type 3), which facilitates the mechanical removal of cards containing dummy instructions, is permitted.

6. Minor revisions have been made in the optimization tables and subroutines.

The system specifications for running the Basic SOAP IIA are the same as for SOAP II, except that the 533 Read-Punch Unit must be equipped with twelve co-selectors. In addition to these requirements, the use of the Tape SOAP IIA requires that the system include at least two 727 Magnetic Tape Units. Copies of the condensed (seven instructions per card) program deck for SOAP IIA and of the flow diagrams and program listings are available on the basis of one copy per installation from:

650 Program Librarian
International Business Machines Corporation
590 Madison Avenue
New York 22, New York

Requests for decks and/or flow diagrams and listings should specify which version is desired; e. g., program deck for the Basic SOAP IIA, flow diagrams and listings for the Tape SOAP IIA, etc.

PROGRAM ASSEMBLY WITH SOAP IIA

Basically, translation and assembly of symbolic program instructions by SOAP IIA are accomplished in the same manner as by SOAP II. However, SOAP IIA can be used to assemble long programs without the sectioning formerly required to insure that the number of symbols in any one section did not exceed a set maximum. When the Basic SOAP IIA is used, only the separation of the output into two parts (i. e., completely assembled and partially assembled cards) is required. The partially assembled cards are then used as input to the following machine pass. In the case of the Tape SOAP IIA the assembly process is completely automatic, and no card handling is required because only completely assembled cards are punched.

Specific information concerning the assembly of programs with SOAP IIA is included in the following paragraphs.

Input Cards

The specifications for input cards (i. e., symbolic instruction cards) for the initial pass of SOAP IIA are identical to those of SOAP II, except for one additional restriction: Column 80 of the input cards must not contain a 12, 1, 2, 3, 4, 5, 6, or 7 punch. The presence of one or more of these punches in column 80 will cause improper assembly.

As will be seen, program cards assembled with SOAP IIA will contain only a 9 punch in column 80. Therefore, symbolic programs can be reassembled with SOAP IIA.

Assembly

The functioning of SOAP IIA is the same as that of SOAP II up to the point at which the maximum number of symbols has been entered into the symbol table. At this point assembly is suspended and a partial assembly phase is begun. During this phase no entries are made in the availability or symbol tables and no pseudo-operation code (except HED) is executed. Partial assembly consists of reading all symbols and checking these symbols against the symbol table. If a symbol is found in the table, its equivalent location is inserted into the output field (either of a card or of a tape record) reserved for the assembled instruction, and a digit is inserted into a control word as an indication that the symbol has been translated. At the beginning of the next pass the symbol table is cleared in preparation for the construction of a new table. The symbolic information contained in a partially assembled card is reproduced into the appropriate output field.

During a machine pass, the indication that a given symbol has been previously translated prevents the program from entering that symbol into the symbol table and assigning it a new location thereby destroying program continuity. The retention of the symbolic information in its original form permits subsequent reassembly of a program.

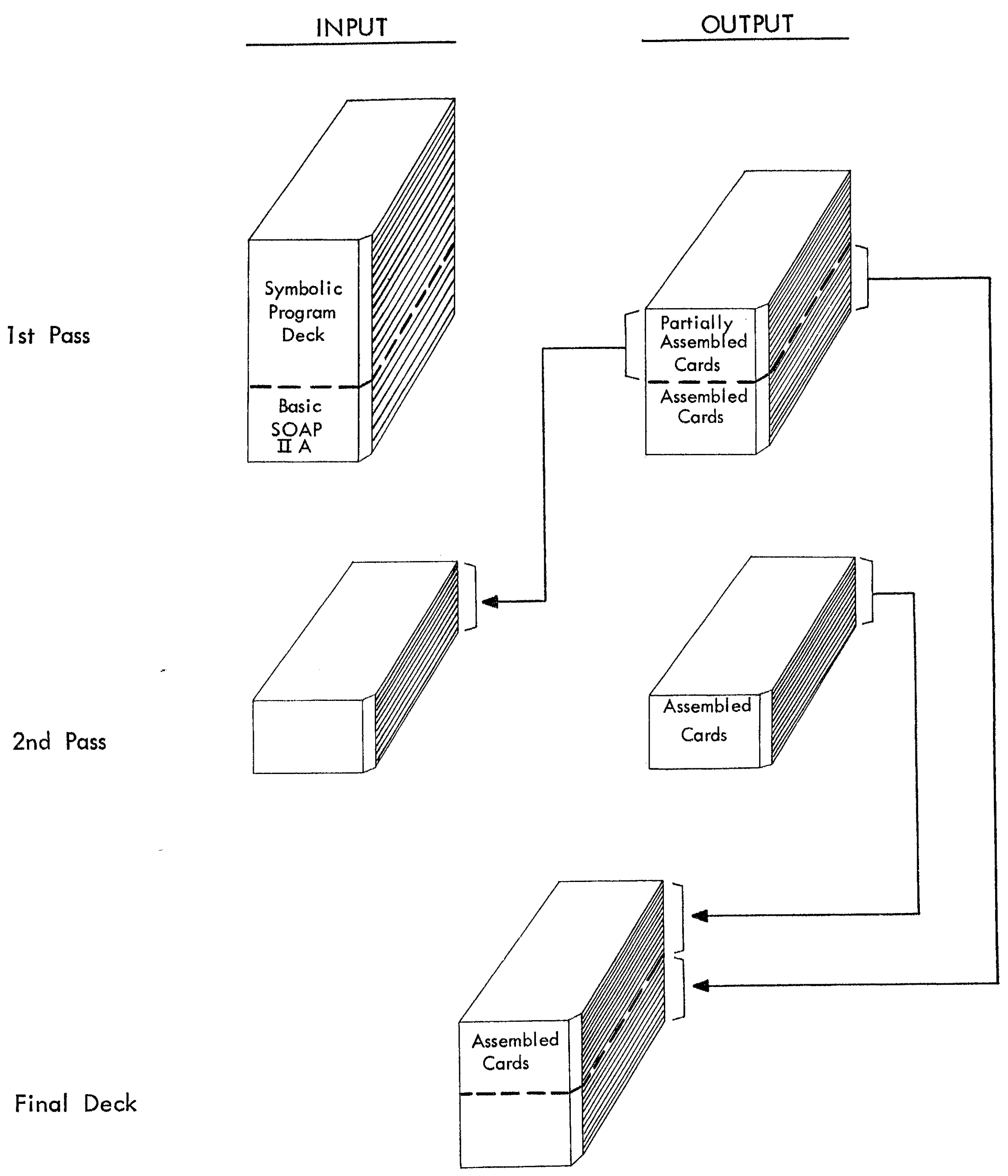
a. Basic SOAP IIA Assembly. When this version is used, the partial assembly phase is begun following the assembly of the card containing the 300th symbol. During this phase partially assembled instructions are punched out as described above, and the control digits are translated into a combination of punches (see Output Cards page 7) which are punched into column 80 of the output cards.

   Following each machine pass the output deck must be checked to determine whether the last output card contains a 9 punch in column 80 or is an availability table card. If either of these conditions is found, assembly is complete; otherwise, all cards containing a 9 punch in column 80 must be removed from the output and held pending completion of assembly. (Note: All completely assembled cards will be together at the front of the output deck.) The remainder of the output deck is used as input for the next pass.

   When all output cards of a pass are found to be assembled, the sections previously removed from output decks are placed together to form one program deck (see Figure 1). The final deck, when correctly placed together, will be numbered sequentially throughout the entire deck.

   Example: Assume that the symbolic card shown in Figure 2 is read after assembly is suspended, and the symbols SOME and NEXT are found in the symbol table to correspond to locations 0542 and 1349, respectively. The corresponding output card would be punched as shown in Figure 3.

b. Tape SOAP IIA. When assembling a program with this version, assembly is suspended following the card (or tape record) containing the 280th symbol. However, cards are punched out during the assembly phase only. At the beginning of the first partial assembly phase all remaining cards are read, partially assembled, and written on tape. Upon completion of this phase the next assembly phase is automatically begun, and the input to the program is taken from the tape previously written. Again during this pass assembled cards are punched, assembly is suspended following the processing of the record containing the 280th symbol, and incompletely assembled cards are written on tape. The process continues until the program is completely assembled. (See Figure 4 for schematic diagram.)



Schematic diagram of card handling during Basic SOAP IIA assembly of a symbolic program containing 301 - 600 symbols

Figure 1

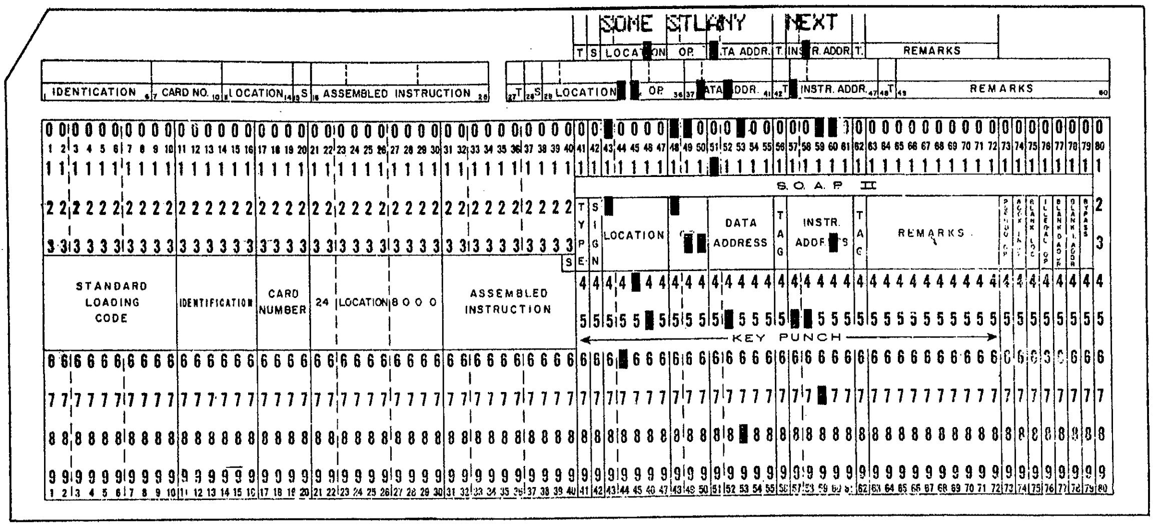


Figure 2

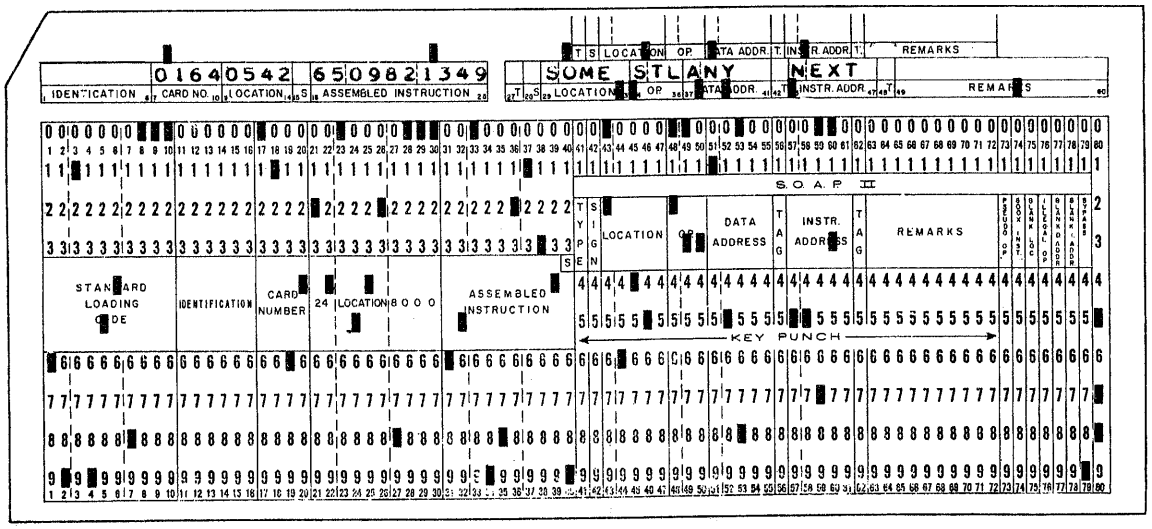
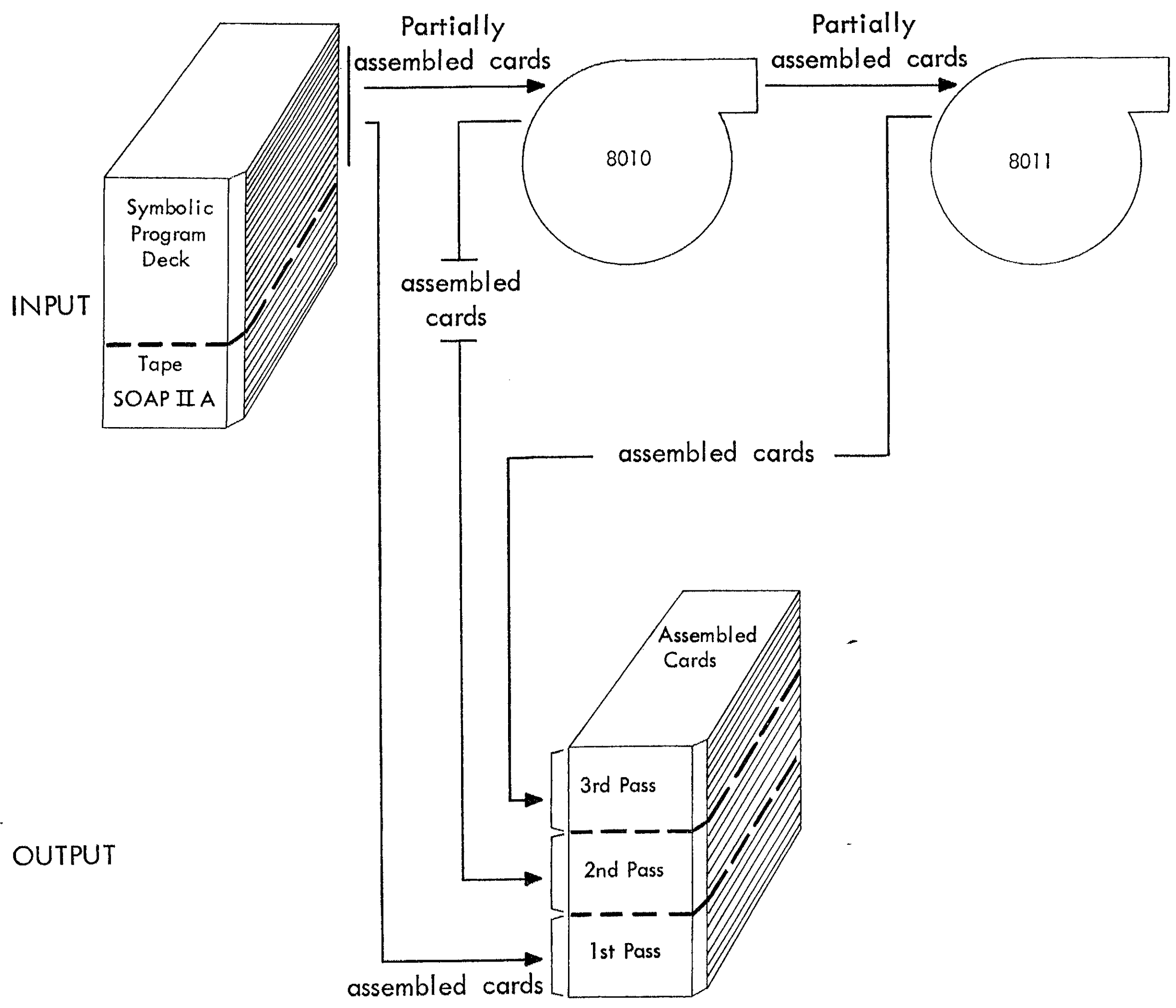


Figure 3

Note: In partially assembled cards, the operation code punched into columns 31 and 32 has no significance, nor do any addresses punched into columns 23-26 and 33-40 which are not indicated by the appropriate punch in column 80 as having been translated. The operation code will be identical with the operation code in the last completely assembled card. Any addresses not translated will be the same as the last translated address of that field. For information pertaining to the punching in column 80, see Output Cards, page 7.



Schematic diagram of Tape SOAP IIA assembly of program containing 560 - 840 symbols.

Figure 4

Symbol Table

With the Basic SOAP IIA the number of symbols contained in the symbol table at the start of partial assembly phase will be 300, 301, or 302 (or 280, 281, or 282 with Tape SOAP IIA) because this phase is begun after assembly of the card containing the 300th (or 280th) symbol, and this card may have one or two new symbols in addition to the 300th (or 280th). Although no more than 302 (or 280) symbols are ever loaded onto the table, an allowance is made for a few additional locations in the table. This allowance permits more rapid partial assembly than would be possible if the entire table were filled.

Output Cards

All output cards will be punched with one or more identifying digits in column 80. These are as follows:

Identification	Meaning
9	Completely assembled cards (no other punches will appear in such cards).
8	Incompletely assembled cards (may appear alone or in combination with 4, 5, 6, or 7).
7	I-address translated.
6	D-address translated.
5	Location translated.
4	First incompletely assembled card of the machine pass.

Note: Since only completely assembled cards are punched by Tape SOAP IIA, only a 9 punch will appear in column 80 of the output cards from that version.

Miscellaneous: Basic SOAP IIA

When used as program cards, incompletely assembled cards will be passed and not loaded into any drum location. When listed on the 407 Accounting Machine, such cards will cause BYPAS to be printed to the right of the remarks (except HED and comments cards). BYPAS will also be printed for EQU or SYN cards of the final deck which were bypassed because of undefined symbolic or regional instruction addresses.

If an availability table is to be reloaded during a multiple pass assembly in order to restore previous availability conditions, the table must be loaded before the 300th symbol is encountered.

The multifile assembly procedure (i. e. , the stacking of two or more symbolic programs with intervening BOP cards) described on page 15 of the SOAP II manual should not be used unless it is known that each program contains fewer than 300 distinct symbols.

Miscellaneous: Tape SOAP IIA

Programs assembled by Tape SOAP IIA must have, as the last program card, a card punched "XYZ" in columns 48-50 (symbolic op code field). This card, however, will not appear in the output of the program.

If an availability table is to be reloaded during a multiple pass assembly in order to restore previous availability conditions, the table must be loaded before the 280th symbol is encountered.

PROGRAMMING FEATURES OF SOAP IIA

Regional Addresses

SOAP IIA will accept regional definitions with FWA > 1999 and will not make incorrect entries in the availability table, thereby permitting regionalization of tape unit, indexing register, and IAS addresses.

Example:

T P	S N	LOCATION	OPER. CODE	DATA ADDRESS	TAG	INSTR. ADDRESS	TAG	REMARKS
			REG	X 8005		8007		
			REG	T 8010		8015		
			REG	I 9000		9059		

When the above regional definitions are made, the addresses 8006, 8013, and 9050 would be coded in the symbolic program as X0002, T0004, and I0051, respectively.

SOAP IIA will accept 9000-9199 inclusive, as valid IAS addresses. The addresses 9060-9199 can, therefore, be used in a program to indicate IAS addresses which are modified by program steps before execution of the instructions involved (e. g. , timing ring settings for variable length records).

Punch Symbol Table (PST)*

In addition to the pseudo-operation codes described in the SOAP II manual, SOAP IIA includes the pseudo-op PST. This code permits the punching of the contents of the symbol table together with the equivalent absolute addresses. The symbols and their equivalents are punched into the Equivalence (EQU) card format.

*PST pseudo-operation code feature was contributed by Mr. T. F. Perkins, California-Texas Oil Company, New York, N. Y.

Example: Assume the symbol table and equivalence table contain the following entries when a PST card is encountered:

Symbol Table	Equivalence Table
EDPM	00 0100 0462
TAX	
.	.
.	.
WAGES	
FICA	00 1807 0025

cards would be punched as follows:

OP	D-Address	I-Address
PST	(blank)	(blank)
EQU	EDPM	0100
EQU	TAX	0462
.	.	.
.	.	.
.	.	.
EQU	WAGES	1807
EQU	FICA	0025

Note: 1. Symbols which are headed when entered into the symbol table will be headed when punched.

2. All EQU cards will be numbered 0000.

3. If a PST card is encountered during the partial assembly phase no symbol table punch-out will occur.

The SOAP IIA program also includes a test of the 650 console Sign switch which is made at the beginning of the second and each succeeding machine pass of a program assembly. If, when the test is made, the Sign switch is set to minus (-) a punch-out of the contents of the symbol table will occur. That is, all symbols used in the preceding pass will be punched into EQU cards.

Since the above described features are included in the program, and because the PST pseudo-op code is not executed during a partial assembly phase, a punch-out of all symbols used in a program may be obtained by setting the Sign switch to minus (-) and placing a PST card at the end of the input deck. This will cause a punch-out of the contents of the table at the beginning of each machine pass (except the first) and upon completion of assembly.

If the only PST card included in the program is at the end of the input deck, no symbol will be punched more than once unless redefined by an EQU or SYN card.

The EQU cards obtained from a symbol table punch-out have two principal uses:

a. The cards may be sorted on the symbols or equivalents and listed for reference. Such listings are useful for detecting incorrect absolute addresses and erroneous symbols not discovered during pre-assembly desk checking.

b. Some or all of the EQU cards, together with the availability table, may be reloaded to establish assembly conditions if it becomes necessary to incorporate changes or new symbolic sub-programs into an assembled program.

Type 3 Cards

Any type "blank" card, i.e., any card other than a comments (type 1) or relocatable (type 2) card, may be specified as a type 3 card by a 3 punch in column 41. This punch does not affect the assembly but will appear in the output card. Type 3 cards may be used to simplify the procedure for optimization of multiple exit instructions, that is, instructions which are modified by program steps as a result of a test or decision. When using SOAP II, it is desirable to code an instruction for each exit, and after assembly to manually remove the extra cards required by this procedure (see paragraph II, page 3, IBM 650 Bulletin 13, for further discussion). When SOAP IIA is used, the same programming considerations are involved; however, the removal of the extra cards is greatly facilitated by making them type 3 cards and using a sorter to extract them after assembly.

The example shown in paragraph II, page 3, of IBM 650 Bulletin 13 would be coded for SOAP IIA as follows:

TP	SN	LOCATION	OPER. CODE	DATA ADDRESS	TAG	INSTR. ADDRESS	TAG	REMARKS
		END 1	STL	ANY		BLK02		
3		END 1	STL	ANY		BLK03		
3		END 1	STL	ANY		BLK04		

533 Control Panel for SOAP IIA

The SOAP II control panel for the 533 cannot be used with SOAP IIA, because assembly will stop with a branch distributor operation code in the program register as soon as the first symbol is encountered. On the other hand, the SOAP IIA control panel described below <u>can</u> be used with the SOAP II program; when this procedure is used, all output cards except availability table cards will contain a zero punch in column 80. This punch, however, will not hinder subsequent reassembly of the output deck and may be disregarded.

A 533 control panel for SOAP IIA may be created from a SOAP II 533 panel by the following steps:

a. Remove from the SOAP II panel the following:

   1. All wires from Read Card B to words 7, 8, and 9 of Storage Entry C.

   2. The wire from the Word Size Emitter 3 to word 10 of Word Size Entry C.

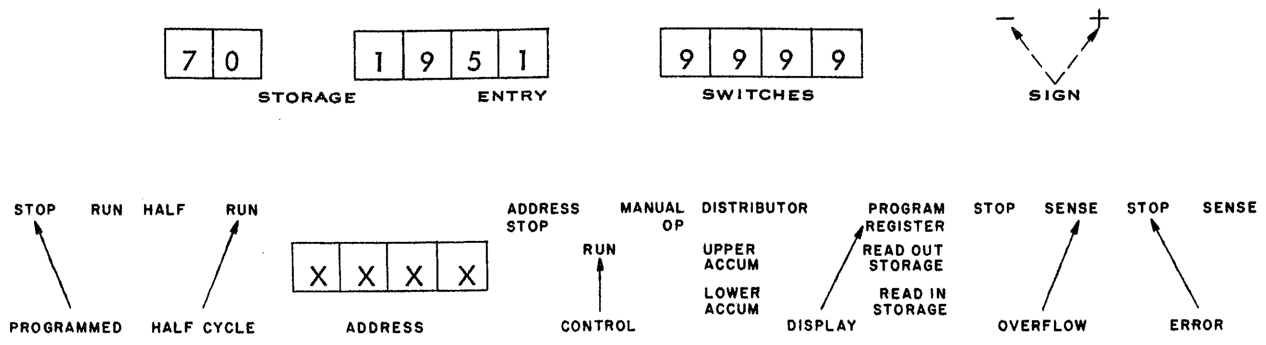
   3. The wire from column 41 of First Reading to the left Read Digit selector and to D PU of Pilot Selector 1.

   4. The wire from the Transfer of Pilot Selector 7 to column 79 of Punch Card A.

b. Add to the remaining wiring of the SOAP II panel the wiring shown in Figure 5.

If a new panel is to be wired for SOAP IIA, the panel should be wired as shown on page 32 of the SOAP II manual, omitting the wiring enumerated above. The wiring shown in Figure 5 should then be added.

BASIC SOAP IIA OPERATING INSTRUCTIONS



Initial Console Setting as shown above.

A. Normal Starting Procedure: Computer Reset; Program Start.

B. Special Instructions

If SOAP IIA is already on the drum do one of the following:

1. Set 00 0000 1000 in the Storage Entry switches.
2. Precede input with a BOP card and set 00 0000 1950 in the Storage Entry switches.

Set Sign switch to minus (-) if automatic symbol table punch-out is desired.

Card Input - Output (533 or 537)

READ FEED

NO. OF CARDS	FILE DESCRIPTION
192	Basic SOAP IIA (including loader)
xxx	Symbolic program deck
1	PST card (if desired)

PUNCH FEED

CARD FORM
SOAP II

CONTROL PANELS

SOAP IIA 533 control panel
(see Figure 5)

TAPE UNITS

ADDRESS	INPUT, OUTPUT OR OTHER	FILE PROTECTION RING IN	FILE PROTECTION RING OUT	LABEL CHARACTERISTICS	FILE DESCRIPTION
8010					
8011					
8012					
8013					
8014					
8015					

Other Instructions and Remarks:

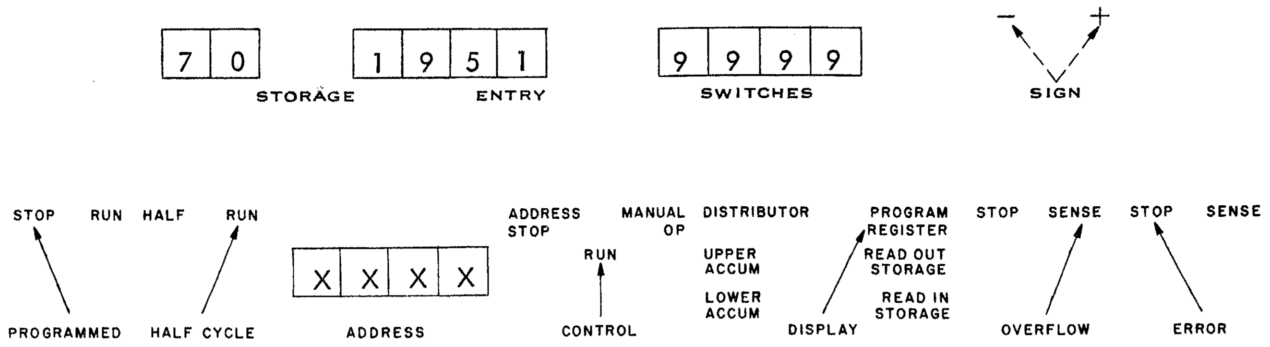
After each machine pass discard the last card out of the Punch Feed. If the preceding card contains a 9 punch in column 80, or if it is an availability table card, assembly is complete. If the card does not meet either of these conditions remove all cards which contain a 9 punch in column 80 from the deck. Use the remainder of the deck for input to the next pass. The first card which does not contain a 9 punch in column 80 (i.e., the first card of the input to the next pass) will contain a 4 punch in that column.

An availability table punch-out may be initiated manually by transferring to location 1900, and a symbol table punch-out by transferring to location 1800.

Program Stops and Required Action:

STOP ADDRESS	MESSAGE – EXPLANATION – ACTION
0222	No locations available for the remaining portion of the program being assembled. Depression of the Program Start key will continue assembly, and addresses not assigned will be left blank in the output cards.

TAPE SOAP IIA OPERATING INSTRUCTIONS



Initial Console Setting as shown above.

A. Normal Starting Procedure: Computer Reset; Program Start.

B. Special Instructions

If SOAP IIA is already on the drum do one of the following:

1. Set 00 0000 1000 in the Storage Entry switches.
2. Precede input with a BOP card and set 00 0000 1950 in the Storage Entry switches.

Set Sign switch to minus (-) if automatic table punch-out is desired.

Card Input - Output (533 or 537)

READ FEED

NO. OF CARDS	FILE DESCRIPTION
203	Tape SOAP IIA (including loader)
xxx	Symbolic program deck
1	PST card (if desired)
1	XYZ card

PUNCH FEED

CARD FORM
SOAP II

CONTROL PANELS

SOAP IIA 533 control panel
(see Figure 5)

TAPE UNITS

ADDRESS	INPUT, OUTPUT OR OTHER	FILE PROTECTION RING		LABEL CHARACTERISTICS	FILE DESCRIPTION
		IN	OUT		
8010	Input-output	X			Scratch tape
8011	Input-output	X			Scratch tape
8012					
8013					
8014					
8015					

OPERATING INSTRUCTIONS, cont'd

Other Instructions and Remarks:

Discard last output card.

An availability table punch-out may be initiated manually by transferring to location 1900, and a symbol table punch-out by transferring to location 1800.

Program Stops and Required Action:

STOP ADDRESS	MESSAGE – EXPLANATION – ACTION
0222	No locations available for the remaining portion of the program. Depression of the Program Start key will continue assembly. Addresses not assigned will be left blank in the output cards.
0444	End of file detected while writing a tape record.* Replace tape with a longer one and begin assembly again.
0555	Error detected while writing a tape record.* Depression of the Program Start key will cause one attempt to rewrite the error record. Do not attempt to rewrite more than three times.
0666	Error detected while writing a tape mark.* Depression of the Program Start key will cause one attempt to rewrite the tape mark. Do not attempt to rewrite more than three times.
0777	Error detected while reading a tape record.* Depression of the Program Start key will cause up to four attempts to reread. If the error condition persists the record may be examined by a console read-out**, or may be printed out, if a 407 Accounting Machine is coupled to Synchronizer 2, by transferring to location 1550. If the record can be corrected manually from the console, transfer to location 1650 to resume assembly.
0888	Error print-out completed.
0999	End of job. Re-initialize before beginning assembly by preceding the program with a BOP card and depressing the Program Start key, or by transferring to location 1000.

* Tape unit address can be determined by displaying the contents of the lower accumulator.
** Records read from tape occupy locations 9050-9059. Format is similar to that shown on page 40 of the SOAP II manual.

Form No. 22-6207-1
Printed in U.S.A.

INTERNATIONAL BUSINESS MACHINES CORPORATION
READ-PUNCH UNIT, 533 CONTROL PANEL
(USED WITH 650 MAGNETIC-DRUM DATA-PROCESSING MACHINE)

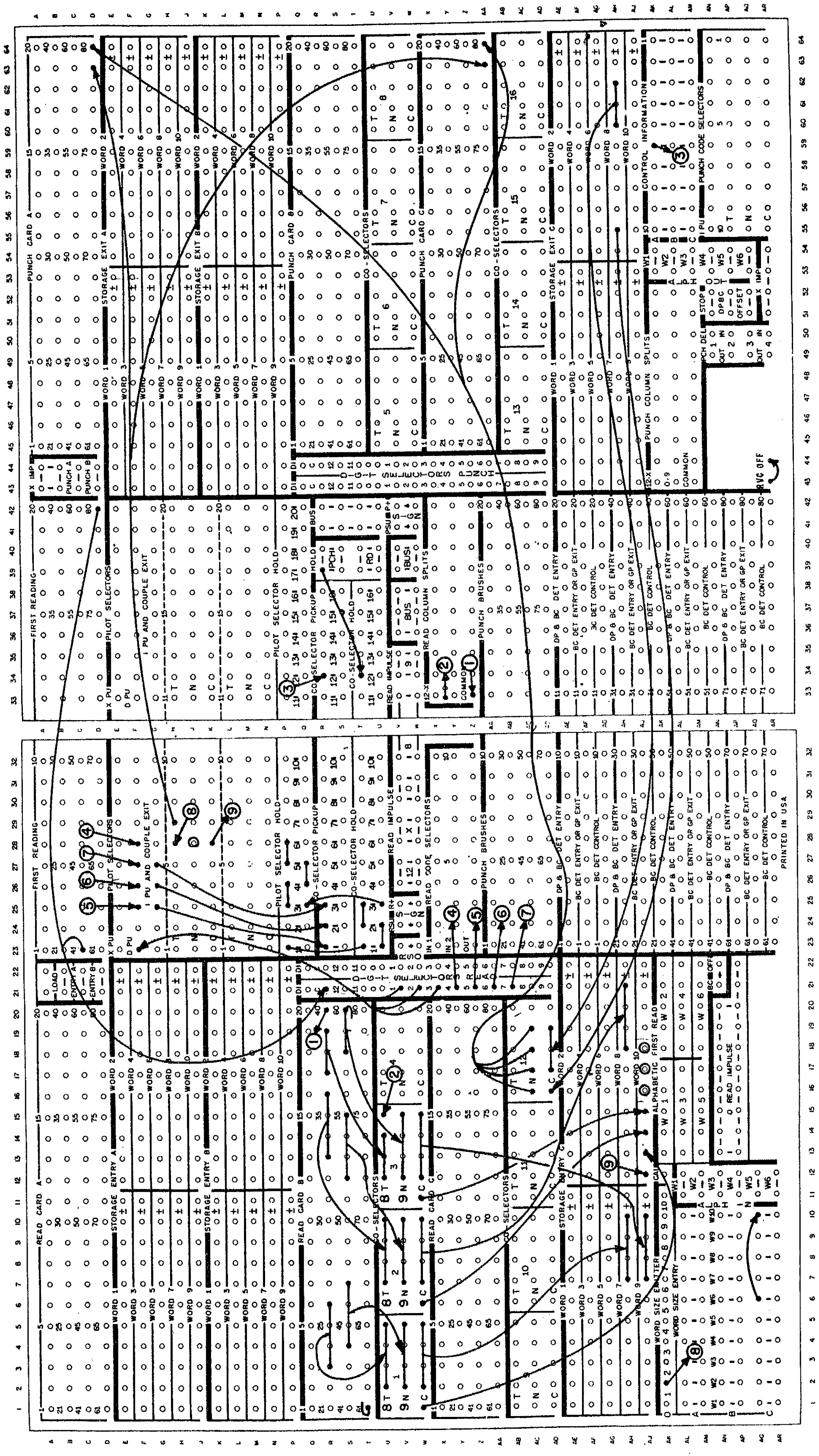


Figure 5

IBM
International Business Machines Corporation
590 Madison Avenue, New York 22, N. Y.

Printed in U.S.A. J28-4001 7/58

650 DATA PROCESSING SYSTEM BULLETIN

THE CHAINING METHOD FOR THE 650 RAMAC® SYSTEM

Considerable time and effort have been spent on the problem of effective utilization of the large storage capacity of the IBM 650 RAMAC System. Several different methods of disk file addressing have been developed. In general, each of these methods has certain advantages and disadvantages which vary in importance depending on the nature of the information to be stored and on the system of file organization employed. Though not a new technique, the chaining method of disk file addressing is one which has recently been adopted for a number of applications and which offers an efficient solution to the addressing problem. Chaining is more effective when a file has high activity with respect to a small percentage of the items in the file and when there is not a rapid turnover of the records stored in the file. While it is recognized, of course, that chaining will not provide a solution to all file addressing problems, the method is considered an efficient over-all system applicable in a wide variety of cases.

The purpose of this bulletin is to present a general description of the chaining method of disk file addressing and of a related set of evaluation and maintenance programs currently under development. Various general aspects of disk file addressing are discussed in the first section. Chaining is then described and compared with other methods. A short discussion of address conversion methods and loading is included; and, finally, the several programs designed for implementing the chaining are briefly described.

GENERAL CONSIDERATIONS IN DISK FILE ADDRESSING

Manually maintained files permit direct access to any desired item. Commercial files cannot be readily organized to accept inquiries on more than one basis, e.g., customer name, location, or product employed. For this purpose, duplicate files are often used, each organized on some different principle. Although duplicate disk files are not suggested, and are not necessary, some of the principles of manual file organization may be employed to advantage.

Information in a file is generally in units, with each unit having a designating identification, such as a part number, a man number, or an account number. This identification is conventionally called the control data. If the control data of the records in a particular file are consecutive numbers without gaps, they may be converted to addresses in the disk file by simple arithmetic. For example, if each file location can hold only one item and the item numbers run from 15,000 to 18,499, then by subtracting 15,000 from the item numbers, disk file addresses from 00000 to 03499 could be assigned. The machine operation in this respect can be very efficient.

It is rare that files are organized in such a way that the control data can be used either directly as disk file addresses or converted simply as in the above illustration. In most applications, the control data of each record must be converted into an address by a somewhat more complex procedure. For example, if information on 3,000 stock items has to be stored and each stock number is ten digits long, the preceding method would utilize only 3,000 out of ten billion possible locations. This is but one example of many in which a filing system is used to identify only a few thousand parts while the numbering system has a potential range in the billions.

Each IBM 355 Disk File unit has a capacity of 10,000 tracks. As many as four units are available for use with the 650, permitting a total capacity of 40,000 separate locations. The addresses range as follows:

Unit	From	To
0	00000	09999
1	10000	19999
2	20000	29999
3	30000	39999

By some method of conversion the control data must be translated into numbers falling within the above range, depending upon the number of units being used. The problem is to store the information so that it may be recovered in the least amount of time while keeping unused (but assigned) storage space to a minimum. These two requirements often conflict so that in practice compromises must be sought.

First, the user must specify the range within which the addresses are to fall. Then, the control data must be converted into addresses within this range. It is desirable to minimize the number of duplicated addresses resulting from the conversion. Thus, if 7,000 items are to be stored within 7,000 tracks, ideally the control data of each record should convert to a unique address. There would be no duplication and no unused addresses.

In practice this is never the case. There is always some duplication and, as a result, some unused space. For example, assume that a record has been stored in a given track and that each track has room for only one record. Then, if the control data of a second record converts to the same address, the second record will have to be stored in a different, unused location. This displaced record is termed an "overflow." Overflows occur when there are more items than can be stored in a location. Thus, if records are 200 digits long, three records could be stored in a track. The "track capacity" would be three. A fourth item would have to be stored as overflow.

With multiple track records, the excess beyond the first track is treated as overflow and may be linked in some manner to the first track of the record.

CHAINING

As indicated in the introduction, several systems have been developed for addressing the file and keeping track of the overflow addresses. The method known as chaining offers an efficient, over-all solution to the problems of organization, maintenance and retrieval of information in a 650 RAMAC system. The merits of the chaining method were established by mathematical analysis and have been substantiated by testing on actual files.

In recovering information from the file, the ideal system would require only one seek. The disk table method * requires a minimum of two seeks, one to obtain the address and a second seek to obtain the item in question. Certain indirect addressing methods* other than chaining brought this average seek below two, provided that the file was less than 85% full. However, since unused storage space is wasteful, it is desirable to run the file as close as possible to 100% full. With such packing, the average seek resulting from these addressing methods goes up well above two. At 90% full, with a track capacity of one record, the average number of seeks is about 5.53. It jumps to 16.91 at 100% full.

Though the disk table method with its average of about two seeks could be useful in this situation (i.e., above 85% full), it suffers certain disadvantages in the maintenance of a file, particularly in handling additions and deletions.

Chaining lowers the average number of seeks to approximately 1.5 when the file is 100% full. However, this assumes that a good conversion method is used to assign disk addresses. The value of any conversion scheme can be determined by the "Address Conversion Evaluation Program for Chaining" to be described later in this bulletin. The graph on page 4 shows the expected average number of seeks assuming that the addressing formula produces a normal distribution with respect to address usage and that all items in the file have equal activity. If the addressing formula produces results which are poorly distributed, then the average number of seeks will be greater than the expected. Conversely if the distribution is very good, the average will be slightly lower.

The average number of seeks will be further reduced when the track capacity (in records) is increased. Thus, with the file 100% full, the average seek is 1.4 for three items per track and 1.37 for five items per track. It will drop still lower as the track capacity is further increased.

An additional refinement called "frequency loading" will bring the average seek to 1.18 for one item per track at 100% full. In frequency loading, items which experience the greatest activity are loaded first and are followed by the less active items. By this means, the number of seeks for overflow items, and thus the average number of seeks, is reduced.

Suppose that a file has a 70-20 activity, i.e., 70% of the activity on 20% of the files. If this 20% is loaded first, then for 70% of the cases, the average number of seeks will be very close to one (less than 1.18). For the other 30%, the average will be greater. For a file close to 100% full and with one data record per track, the average number of seeks will be about 1.18. It will be even less than this if the track capacity is increased.

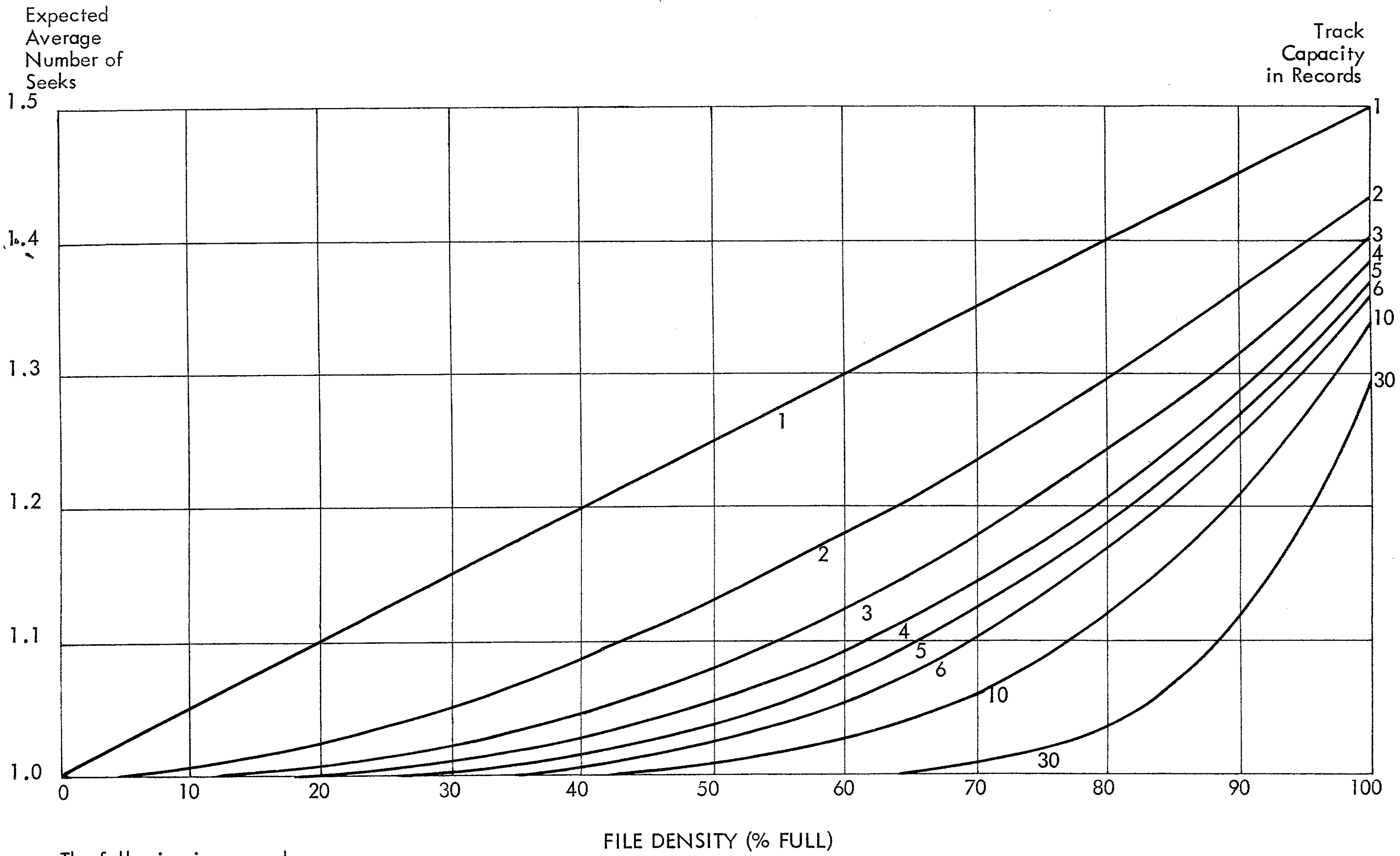
ADDRESS CONVERSION

In chaining, as in some of the other methods of indirect addressing, a formula is applied to the control data of each record to develop a numerical result. Five digits are chosen

*Described in the IBM 650 RAMAC Manual of Operation, Form 224-6270.

EXPECTED AVERAGE NUMBER OF SEEKS FOR

VARYING FILE DENSITY AND TRACK CAPACITY



The following is assumed:
1. Normal distribution with respect to address usage.
2. Equal activity of all items in the file.

from the result to provide the disk file address. A good formula should pack a file well and yet minimize the number of duplicate locations which cause overflows. Various techniques are used for this purpose such as multiplying the control data by a constant or by squaring it. 650 Bulletin 11 gives several such conversion techniques.

Described below is a simple but efficient method of address conversion, i.e., conversion of control data to a disk file address, which warrants serious consideration in any application involving indirect addressing. This method has been used very effectively for several chaining applications in which other conversion formulas did not produce satisfactory results.

1. If the control data consists of more than ten digits, fold the number to ten digits or fewer. Folding shrinks a number by partitioning it and adding the pieces to form a new number.

2. Modify the range of addresses (sometimes called the "compression factor" or "scaling factor") so that the units position is a 1, 3, 7 or 9.

3. Divide the range into the control data.

4. Add the remainder to the starting disk location to obtain the disk file address.

   Thus:

Control data:	9675889993	
Range:	5000 tracks	(00101 to 05100)

   Modify the range to 5001.

   Divide the control data by the modified range:
   $9675889993 \div 5001 = 1934791$

Remainder:	00202
Plus:	00101
Disk Address:	00303

The advantage of the above method lies in the fact that every digit in the control data contributes to the remainder and the remainder is properly scaled. In cases where this method has been employed the resulting addresses have been randomly distributed.

Overflow Records

To illustrate the handling of overflow records, assume that a conversion procedure has developed the same disk file address from the control data of three different records.

Control Data	Disk Address

1 329 401
2 739 902
5 782 133
	05772

If the track has a capacity of one record, then there will be two overflow records. The first record, (control data 1 329 401) goes into disk file location 05772, which is called the "home" position. The other two records are overflow records and must go into two unused locations.

Assume that the second record (control data 2 739 902) goes into 05779. Then the link, 05779, of the chain is specified in one of the words of the record in location 05772. The record in 05772 might appear as follows:

Control Data	Information	Link Address	Information
1329401	XXX...XXX	05779	XXX...XXX

If the third record (control data 5 782 133) is stored in 05785, then 05785 will appear as the link in location 05779. Thus, to find the third record, three seeks would be required: a seek of home (05772), a seek in the first overflow address (05779) and a seek in the second overflow address (05785). If the records were not linked and if the record were sought sequentially from 05772 to 05785, then 14 seeks would have been required to locate the third record.

It is true that by the disk table method, the third record could be located in two seeks. However, it should be remembered that every record would require at least two seeks: one or more for the address and one for the record. With chaining, the greatest average number of seeks is 1.5. It is less when the track capacity is greater than one.

CHAIN LOADING

Efficient loading of a chained file requires two passes. In the first pass only those records are stored for which home locations are available. All other records are overflows and will be rejected. During the second pass, unused space is found for each overflow record. The address of this location is stored as the last word in the preceding record of the chain.

Overflow records should be stored as close to home as possible so that they can be retrieved in the shortest time. The minimum seek time is about 65 milliseconds when the overflow record is stored either directly above or below the preceding record of the chain. If space is not available over or under the preceding record, then the availability of space on the adjacent track of the same disk is investigated. This "over/under" search continues until space for the overflow is found. If necessary, the search is continued on an adjacent disk. In this fashion, a chain of records is built.

PROGRAMS FOR CHAINING

Five programs to implement the chaining method of file organization are briefly described in this section. The machine requirements vary from one program to another, but the chaining method as a whole requires the following configuration of equipment:

533 Read - Punch Unit:	1
355 Disk File Unit:	1 to 4
727 Magnetic Tape Unit:	2 or more

The first of the five programs, the Address Conversion Evaluation Program, has been completed. This routine is designed to run on a basic 650 and is being made available in advance of the other components of chaining to permit early testing of addressing formulas. The other programs are in various stages of development, and are presently scheduled for completion in early August 1958.

Note: For efficient functioning of these and other programs, the first 150 tracks of the first disk file unit are reserved for utility program use. Fifty additional tracks in the first unit are reserved for each additional file unit in the system. Thus if only one file is to be used, 150 tracks are reserved: 00000 - 00149. If four units are to be used, 300 tracks are required: 00000 - 00299. These tracks representing less than 2% of disk storage will be utilized for such functions as housekeeping, maintenance, subroutines, availability tables, etc.

Address Conversion Evaluation Program for Chaining

This program is designed to test address conversion formulas which are to be used in filing systems based on chaining. It tests the formulas on the control data of the actual file to be stored in order to determine the average number of seeks. The testing is entirely automatic and, as indicated previously, is performed on a basic 650.

Address conversion will vary because of differences in:

1. The distribution of the control data in the original files: Commercial files are generally broken down into groupings. Between these groups there may be large gaps of unused numbers. Within any group the control data of the individual records probably will differ from each other in only one or two positions. There will be areas of concentrated numbers and areas of open space. These distribution characteristics vary from file to file. Hence, a conversion method which works well with one file may not work well with another. Each file has to be examined and tested individually to find its best conversion formula.

2. The conversion formulas employed: Formulas for conversion may be varied by changing the factors used or the sequence of operations to be followed. Each will produce a different distribution of the resulting addresses. How these distributions relate to the economy factors of time and space must be examined.

The program analyzes the control data and then punches out the average number of seeks on a single card. No further tabulations or computations are necessary. In this manner, several conversion methods may be tested in rapid succession and those which are found to be inadequate quickly eliminated.

In order to distinguish between two or three methods developing approximately equal average seeks, the average seek card for each method may be re-fed. The machine will then punch out the frequency summaries. These are the counts of the unused addresses and of those addresses used once, twice, three times and on up to 99 times. Thus, the user can determine the degree of uniformity with which the range of tracks is employed. That method developing a low average seek and a uniform distribution will probably function with greatest efficiency.

The program also includes a feature which permits the user to inspect the actual frequency distribution of track usage, i. e. , the number of times each track address was used.

The Address Conversion Evaluation Program was specifically designed to test conversion formulas to be used in conjunction with chaining. The evaluation program described in 650 Bulletin 11 retains its general utility but is not suggested for use with chaining programs.

The program writeup and the card deck in five-per-card form are available from:

650 Program Librarian
International Business Machines Corporation
590 Madison Avenue
New York 22, New York

Character Distribution Analysis Program

This program aids in analyzing the actual control data for possible use of selected positions as direct addresses.

Each position of the control data is individually examined. A tally is made of the digits or characters which are used in every position. Those positions having a suitable distribution may possibly be used as disk addresses. The selected columns should then be tested with the Address Conversion Evaluation Program to determine the nature of the address distribution in actual operation.

The tallying is performed in a single pass on the basic 650 and is entirely automatic. The results may be listed for analysis on an IBM 407 Accounting Machine.

Chain Loading Program

The distinguishing feature of chaining, as pointed out earlier in this bulletin, is the method of handling overflows. By booking or linking all of these overflow records to the home address or to a preceding record, which is in turn linked to the home address, the average seek time is reduced below the average required by other methods.

Input to the chain loading program will be in the form of magnetic tape records prepared from card records by the user's card-to-tape program. As indicated previously, the loading program involves two passes. The first pass loads the home records. The second pass loads the overflow records. To load the home records, a table of disk file addresses is maintained on the drum. As the control data of each record is converted to an address during the first pass, the loading program will check the drum table to determine whether or not the location is full. If space is available, the record will be stored. As each track is filled, the corresponding address in the drum table is checked off. Once a disk storage location has been filled, any additional records for this location are stored on tape for the second pass.

During the second pass the overflow records, i. e. , the records which were rejected during the first pass, are read from the tape. The control data of each record is converted, and its chain is searched until the last record is found. The program then consults the drum table for available space close to the last record. When space is found, the new record is stored and its location recorded in the link word of the preceding record.

Additions and Deletions

This program is designed to handle alterations to files in such a manner that control is maintained over the retrieval of information and the efficiency of the organization of the files kept at a high level.

An addition is a new entry to the file. The program will handle additions in the following manner:

1. The control data of the addition is converted to an address which is the home address.

2. If home is unoccupied, the addition will be stored there.

3. If home is occupied, the record in home is tested. If the record in home belongs there, the addition is stored at the end of that chain and linked to the preceding record.

4. If the record in home does not belong there, i. e. , it is an overflow from some other home location, it is removed to make room for the addition. The record which was removed is relocated, and its link in the preceding record is changed.

A deletion is an item to be removed from the file because of non-activity or obsolescence. Since references may yet be made to such an item and there is usually no urgent need to remove it, the record is retained and flagged. This is done by placing a "9" in the link word. Thus, when a reference is made to the item, the processing program can take suitable action such as punching out a card or printing a comment. In this manner, a non-available item will be highlighted in the minimum time and with the least handling. The actual removal of obsolete items is accomplished either by one of the user's programs or by the Optimal Sequencing Program described below.

Optimal Sequencing Program for Chaining

The Optimal Sequencing Program is designed to rearrange each chain in the disk file in frequency order, i. e. , high activity items first, when information has become available about the relative activity of the various items in the file. The file could be sorted periodically to increase its efficiency.

The program will perform three functions:

1. The chains will be arranged in order of activity.

2. The frequency counter of each sorted record will be set to zero.

3. Flagged deletions will be placed at the end of each chain or removed if desired.

Since any deletion should be copied on a card or on tape, it is desirable to postpone the removal of individual items until a file maintenance operation is required. Periodic removal of deleted items in a batch, such as at the end of an accounting period, usually improves control and maintenance of the system.

The program uses a maximum of 20 tracks, each containing 60 words, or a total of 1,200 words. It will sort from 20 to 50 records in a single pass depending upon the track capacity. If there are more than 50 records in a chain, the first 50 will be sorted and stored back in the disk file. The remainder will be sorted and then merged with the first group of 50.

After each rearrangement of the file, it should be immediately unloaded onto tape for audit purposes. The tape would then reflect the condition of the file as of the last sort.

IBM®

International Business Machines Corporation
590 Madison Avenue, New York 22, N. Y.

Printed in U.S.A. J28-4002 7/58

650 DATA PROCESSING SYSTEM BULLETIN

CORRECTIONS IN SORT II

The detection of three latent errors in IBM 650 Tape Sorting Program, Sort II, makes necessary several changes in the Sort II condensed (five-per-card) load deck. The changes are designed to correct the following conditions:

1. Improper sorting in Phase I of records whose first control data word is either +9999999999 or -0000000001.

2. Incorrect output reel assignment in Phase II if the first control data word of any record is negative.

3. Improper merging of the highest record of the file when the merge output consists of two reels.

The condensed deck and the corresponding lines of the listing on pages 74-80 of the Sort II manual, form 328-0415 (formerly 31-0415), should be changed to read as follows:

Card Serial Number	Word Number	New Value
014	2	1190350305
156	3	4400750309
	4	6080010075
	7	1911030503
	8	0919601960
222	5	2500010242
229	5	2500010833
366	6	2400340474
380	2	6104240491
	3	9999999999
	4	2400600887
	7	0001042404
	8	7419601960

The availability tables on pages 72-73 of the Sort II manual should be altered as follows:

page 72	word opposite 0005:	0000000000
	" " 0009:	0000010000
page 73	word opposite 0001:	0000000000
	" " 0024:	0000000000

On page 52 of the Sort II manual, alter line 49, add lines 49A and 49B, and alter line 50:

```
49      2COMP     SUP    9035          0017   11   9035   0305
49A               NZU    STHSH         0305   44   0075   0309
49B               RAU    8001   STHSH  0309   60   8001   0075
50      STHSH     STD    HASH          0075   24   0028   0031
```

On page 64 of the Sort II manual, alter line 1067, and add line 1067A:

```
1067              STD    WTIN          1456   24   0034   0474
1067A             STD    NMWT          0474   24   0060   0887
```

On page 66, alter line 342 and add lines 344A and 344B following line 344:

```
342     2G1       NTS    2H1A          0287   25   0001   0242
344A    2H1A      RSU           2H1    0001   61   0424   0491
344B               99    9999   9999   0424   99   9999   9999
```

On page 71, alter line 510:

```
510     C2H1      NTS    2H1A   8C1    0322   25   0001   0833
```

Two typographical errors should be corrected in the Sort II manual:

1. On page 12, change the last line to read "word) in IAS, and zeros filled in as words 8 and 9."

2. On page 40, the rectangular box in the lower right whose contents presently read "Store Its Sort Words as XW" should read "Store Its Sort Words as XS."

IBM
International Business Machines Corporation
590 Madison Avenue, New York 22, N. Y.

Printed in U. S. A. J28-4005 9/58

IBM 650 DATA PROCESSING SYSTEM BULLETIN

STATUS OF SOAP PROGRAMS

In view of the number of inquiries received, it appears desirable to clarify the status of the S. O. A. P. program, which is frequently referred to as SOAP I.

SOAP II - Symbolic Optimal Assembly Program for the IBM 650 Data Processing System - has replaced SOAP I. The principal advantages of SOAP II over SOAP I are:

1. Provision for the operation codes pertaining to Automatic Floating Decimal Arithmetic, Indexing Registers, additional Input-Output Synchronizers, Immediate Access Storage, Tapes, and RAMAC®.

2. Improvement and extension of the pseudo-operation structure.

These improvements necessitated a reduction of 200 locations in the size of the symbol table: from 600 locations in SOAP I to 400 locations in SOAP II. The consequent problem of assembling long programs was recognized, and this led to the development by the IBM 650 Applied Programming group of SOAP II A, a modification of SOAP II. SOAP II A permits assembly of programs containing any number of symbols, while retaining all features of SOAP II; in addition, improvements have been incorporated with respect to addressing, programmed switches, symbol documentation, and reassembly. The program exists in two forms: Basic SOAP II A, written for the basic (card) 650; and Tape SOAP II A, a high-speed program employing IAS and two 727 Magnetic Tape Units.

Thus SOAP I is considered obsolete because it has no features not available in SOAP II and/or SOAP II A and furthermore cannot be used to assemble programs written for 650 systems equipped with additional features. The manual describing SOAP I is out of print and will not be reprinted. Practically all major 650 utility programs have been written, or rewritten, in SOAP II form; and those programming systems which themselves use an assembly routine, e.g., FOR TRANSIT, utilize SOAP II rather than SOAP I.

The basic publication concerning SOAP II is, of course, the SOAP II Reference Manual, C28-4000 (formerly 32-7646). Both versions of the SOAP II A program are described in a recent IBM 650 Systems Bulletin, titled SOAP II A (J28-4001).

The conversion of programs from SOAP I form to SOAP II form is easily accomplished with the SOAP I to SOAP II Translator which is available from the IBM 650 Program Library under file number 1.6.016. (Note: A SOAP II, rather than SOAP II A, control panel for the 533 Read-Punch unit should be used with the Translator.)

IBM 650 DATA PROCESSING SYSTEM BULLETIN

CORRECTIONS IN SOAP IIA

The detection of a latent error in both the basic version and the tape version of the SOAP IIA program makes necessary a change in one card of each condensed (seven-instructions-per-card) load deck. The change will remedy the failure of the program to blank out invalid location addresses, i.e., addresses other than 0000-1999, 8000-8003, or 9000-9059. This change will also correct the failure of the program to punch the identifying digit needed to cause "BLANK L" to print in the post-assembly 407 listing as an indication of such invalid location addresses.

In each case the change applies to the card in which the first word is 00 0933 0007 and involves re-punching the fifth word of that card, as follows:

Basic SOAP IIA	60 0535 1339
Tape SOAP IIA	60 0985 1339

In the flow charts and listings for both versions of SOAP IIA, instruction L0004 in Subroutine 19 should have the operation code RAU, rather than ALO.

Copies of the Basic SOAP IIA and Tape SOAP IIA load decks supplied by the IBM 650 Program Librarian after October 1, 1958, incorporate the above changes.

Printed in U. S. A. J28-4017 11/58

650 DATA PROCESSING SYSTEM BULLETIN

IBM 650 MODEL 4 (4000 Word Magnetic Drum)

This edition, G24-5009-1, obsoletes Form G24-5009-0. The major change is: " Tagging 80XX or 90XX I-Addresses for D-Address Modification. "

The IBM 650 Model 4 provides all configurations of the IBM 650 System with additional drum storage. The 650 Model 4 increases the flexibility of the 650 system by doubling the number of addressable drum-storage locations, which also increases the number of available optimum locations. The additional 2000 drum-storage locations are assigned addresses 2000 through 3999.

The increase in addressable locations is achieved by increasing the number of 50-word bands from 40 to 80 (Figure 1). Addresses 0000 through 3999 are valid drum addresses for the 650 Model 4.

The Model 4 operates the same as the IBM 650 Model 2. The same operation codes are used, and the same operation execution timings apply. However, the method of using indexing registers in the Model 4 is different from the method used in the Model 2. The major part of this bulletin is used to illustrate the indexing operation in the Model 4.

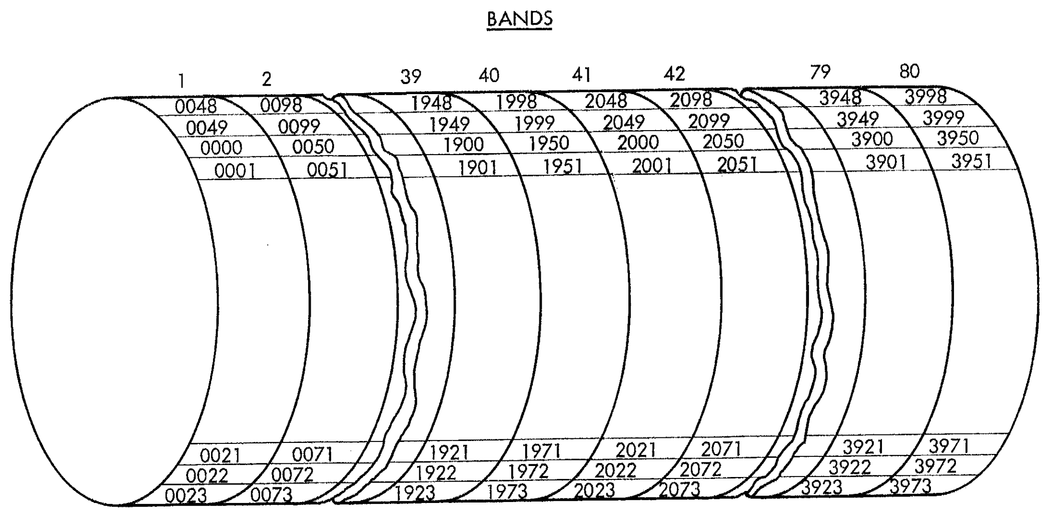


Figure 1. Drum Storage Location Schematic

Indexing Registers

Addresses assigned to the indexing registers are:

Indexing Register	Address
IRA	8005
IRB	8006
IRC	8007

These addresses can be used as the instruction address of any instruction, or as the data address of the following instructions:

00-01	54
10-11	60-61
25-26	64-69
30-49	90-99

Address Modification

The primary use of indexing registers is to modify addresses automatically by adding (algebraically) the contents of an indexing register to an address.

The indexing register can contain either positive or negative values, making it possible to modify addresses, by tagging the instruction.

The basic instruction is not changed by this modification, and it can be remodified any number of times.

Mode Switch

When the Model 4 modified Indexing Register feature is included in the IBM 650 Model 4 System, the method of addressing drum locations depends on the setting of the mode switch located on the IBM 653. This mode switch is an optional feature available only when the system has the indexing feature. When the mode switch is set to 2000:

1. Both the D- and I-address can be modified.

2. Complete compatibility with present indexing as described in 650 Data Processing Bulletin, G24-5003. Programs already in use can be processed by the Model 4 without any changes.

3. Drum locations 2000-3999 cannot be addressed.

If the mode switch is set to 4000:

1. The indexing structure is changed as follows:

   a. Tagging the D-address with 4000 indexes the D-address with the contents of indexing register A.

   b. Tagging the I-address with 4000 indexes the D-address with the contents of indexing register B.

   c. Tagging both the D- and I-addresses with 4000 indexes the D-address with the contents of indexing register C.

2. Any of the 4000 drum addresses can be directly addressed and can be indexed.

3. The I-address, when a drum address, cannot be modified. However, by the use of a branch code which substitutes D for I, the I-address is effectively modified. An extra program step is needed each time (Figure 2).

4. IAS addresses can be modified in the normal manner by adding 200, 400, or 600 to indicate the use of indexing register A, B, or C, respectively.

Index Register A $\boxed{0025}$

2000 Mode

Loc.	Inst.	D	I
0996	70	1951	[3000]
Next Inst. is located in			1025

4000 Mode

	Loc.	Inst.	D	I
	0996	70	1951	0997
(Extra Step)	0997	NZA	[5000]	1000
	Next Inst. is located in			1025

Figure 2. A Method of Modifying the I-Address

With the mode switch at 4000, address modification is accomplished by adding the contents of an indexing register to a basic drum address (Examples 1, 2, 3; Figure 3). If the contents of the indexing register are positive, and the resulting effective address exceeds 9999, the carry is lost and only the four low-order digits of the sum are retained. The machine does not stop because of this overflow (Example 4; Figure 3).

If the contents of the indexing register are negative, adding the 10's complement forces a carry to occur when the difference is positive (Example 5; Figure 3).

Example	Program Register	IRA		IRB		IRC		Program Register	
	Before	Before	After	Before	After	Before	After	After	Remarks
1.	65 7123 0124	0223	0223					65 3346 0124	D-address indexed by IRA.
2.	65 0123 7288			0075	0075			65 0198 3288	D-address indexed by IRB.
3.	65 4211 4314					1645	1645	65 1856 0314	D-address indexed by IRC.
4.	65 7123 7288	0223	0223	0075-	0075-	8231	8231	65 1354 3288	D-address indexed by IRC; D exceeds 9999; carry is lost.
5.	65 3122 7128	0223	0223	0075-	0075-	8231	8231	65 3047 3128	D-address indexed by IRB; D exceeds 9999, carry is lost.
6.	65 5168 5269	0223	0223	0075-	0075-	8231	8231	65 9399 1269	D-address indexed by IRC; 9399 causes storage selection error.
7.	65 4123 0124	1011-	1011-					65 9112 0124	D-address indexed by IRA; complement 9112 causes storage selection error.
8.	65 4123 0124	1111-	1111-					65 9012 0124	D-address indexed by IRA; complement, however is a meaningful address.

Figure 3. Address Modification on the 650 Model 4 (Mode Switch Setting 4000)

If indexing by subtraction results in a negative address, the complement result is not reconverted. This may result in a storage selection error if the effective address is not a meaningful address (Examples 6 and 7; Figure 3). If the complement is a valid address, no error is indicated (Example 8; Figure 3).

Tagging 80XX or 90XX I-Addresses for D-Address Modification

The D-address can be modified when the next instruction is located in IAS or is a valid 8000 address. To modify a D-address with indexing register B, the I-address (80XX or 90XX) is tagged by adding 800. To modify a D-address with indexing register C, the D-address is tagged by adding 4000 and the I-address (80XX or 90XX) is tagged by adding 800. This method of modification can be done only if the mode switch is set to 4000 and the I-address is either 80XX or 90XX (Figure 4). I-addresses of 9000 through 9059 cannot be modified when the 800 tag is used.

Program Register Before	IRB	IRC	Program Register After	Remarks
Tag ↓ 65 0141 9815	0050	0075	65 0191 9015	D-Address indexed by IRB
Tag ↓ Tag ↓ 65 4182 8800	0050	0075	65 0257 8000	D-Address indexed by IRC

Figure 4. Address Modification When 80XX or 90XX is used as the I-Address

IR Arithmetic Operations

Four operation codes (Figure 5) are associated with each indexing register: add, subtract, reset-add, or reset-subtract data into each register. The D-addresses 8005, 8006, and 8007 cannot be used with the IR arithmetic operation codes.

When index registers are used as accumulators, they are similar to the upper and lower accumulator except:

1. They are smaller (4 positions).
2. They do not indicate when an overflow occurs.
3. They can accept data only from positions 1-4 of another immediate access storage device, or positions 5-8 of the program register.
4. They cannot accept data directly from a drum location.
5. They subtract by the 10's complement method, and a complement result is reconverted.

	IRA	IRB	IRC
Add	50 (AXA)	52 (AXB)	58 (AXC)
Subtract	51 (SXA)	53 (SXB)	59 (SXC)
Reset Add	80 (RAA)	82 (RAB)	88 (RAC)
Reset Subtract	81 (RSA)	83 (RSB)	89 (RSC)

Figure 5. IR Arithmetic Operation Codes

In any immediate-access storage device such as the upper or lower accumulator (8002, 8003), the distributor (8001), the console switches (8000), or IAS (9000 to 9059), the four low-order positions of data can be added into any index register with an IR arithmetic operation code as shown in Figure 6.

Instruction	Data		Contents of IRA		Contents of Distributor	
	Loc.	Contents	Before	After	Before	After
AXA 50 8001 xxxx	8001	139423 8621+	0000+	8621+	1394238621	1394238621+
RAA 80 9021 xxxx	9021	006123 1426-	2713+	1426-	0176548921	0061231426-
SXA 51 8000 xxxx	8000	006123 1426+	2426+	1000+	1643263118	0061231426+
SXA 51 8002 xxxx	8002	152619 8712+	2426+	6286-	0000000718	1526198712+

Figure 6. Using IR as Accumulators

Data stored in drum addresses 0000 to 3999 cannot be added or subtracted directly into index registers by the IR arithmetic codes. When any number between 0000 and 3999 appears as the data address of an IR arithmetic operation code, the number itself is placed in the specified indexing register (Figure 7). The sign of this number is always treated as plus, because the program register carries no sign. This is a convenient way of placing factors in an index register with one instruction, and does not require another storage location to store the factor itself. It is also a simple way to add or subtract 1 for counting a program loop.

Instruction	Contents of IRA	
	Before	After
RAA 80 0126 xxxx	1111+	0126+
AXA 50 0126 xxxx	1111+	1237+
SXA 51 0126 xxxx	1226+	1100+
RSA 81 0126 xxxx	1428+	0126-

Figure 7. Using IR as Accumulators

Arithmetic Operations Between Indexing Registers

If the contents of an indexing register are to be operated on by an IR arithmetic code, the D-address, the I-address, or both the D- and the I-addresses must use the 4000-tag address. The address or addresses tagged determine the indexing register to be used.

The indexing-register arithmetic-operation code determines the indexing register to be operated on. It also specifies whether the effective data address (basic data address plus the tag) is to be added or subtracted (Figure 8). This could result in the sum of three factors:

1. the base number (0000-3999) in the data address

2. the contents of IR called for by the tag

3. the contents of the IR indicated by the operation code.

Program Register	IRA		IRB		IRC		Program Register
Before	Before	After	Before	After	Before	After	After
AXA 50 4156 3122	0500+	1156+					50 0656 3122
AXA 50 0156 7122	0500+	0545+	0111-	0111-			50 0045 3122
AXA 50 4156 4122	0500+	2106+			1450+	1450+	50 1606 0122
SXA 51 0250 6018	0500+	0375+	0125-	0125-			51 0125 2018
RAA 80 0001 1415	0500+	0001+					80 0001 1415
RAA 80 0005 6313	0500+	0155+	0150+	0150			80 0155 2313
RSA 81 4002 4009	0500+	1241-			1239	1239	81 1241 0009

Figure 8. Arithmetic Operations Between Indexing Registers

IR Branching Codes

Testing the indexing register to make logical decisions is done by using branch codes. Two branch codes (Figure 9) are associated with each of the indexing registers to test them individually for a minus condition, and for a non-zero condition.

	IRA	IRB	IRC
Branch on non-zero	40 NZA	42 NZB	48 NZC
Branch on minus	41 BMA	43 BMB	49 BMC

Figure 9. IR Branching Operation Codes

The data address (branch address) or the instruction address of an IR branch code can be any valid machine address (Drum IAS, IR, or arithmetic unit). If the next instruction is taken from an indexing register, it is treated as a no-op code, and the instruction following the no-op code is taken from the address specified in the index register.

Indexing of Immediate Access Storage Addresses

The method of tagging immediate access storage addresses for the IBM 650 Model 4 has not changed. The method is the same whether the mode switch is set to 2000 or 4000. Tagging is accomplished by adding 200, 400, or 600 to the IAS address to indicate the use of indexing register A, B, or C respectively.

Addresses that Cannot be Tagged

With the Model 4 machine, the following addresses cannot be tagged when in the D portion of the instruction word:

8000	Console
8001	Distribution
8002, 8003	Accumulators
8005 through 8007	Indexing Register
8010 through 8015	Magnetic Tape Units

However, the address developed after modification by an indexing register can be any one of the preceding, if it is meaningful to the operation.

SOAP IIA-4000

SOAP IIA-4000 can be used for optimizing programs for the IBM 650 Model 4.

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York

Printed in U. S. A. G24-5009-1 12/59