Copy No. _____

REVISED MANUAL

HARVEST SYSTEM

November 13, 1957

Company Confidential

IBM Research Laboratory
Poughkeepsie, N. Y.

TABLE OF CONTENTS

H1. Stream Indexing
- H1.1 Control Parameters and Control Bits
- H1.2 Stream Indexing Setup
- H1.3 Operation
  - H1.3.1 Nested Indexing
    - H1.3.1.1 Control Bits
  - H1.3.2 Triangular Indexing
    - H1.3.2.1 Control Bits
  - H1.3.3 Sequential Indexing
    - H1.3.3.1 Control Bits
- H1.4 Special Values of Indexing Parameters
- H1.5 Automatic Adjustment of Index Level
  - H1.5.1 Repeat Byte in P (Q)
  - H1.5.2 Repeat Byte in R
  - H1.5.3 Skip Byte in P (Q)
  - H1.5.4 Runout First Level in P (Q)
  - H1.5.5 Runout Second Level in P (Q)
  - H1.5.6 Runout Higher Level in P (Q)
  - H1.5.7 Runout Field in P (Q)
  - H1.5.8 Runout Sequence in P (Q)
  - H1.5.9 Reset First Level in P (Q, R)
  - H1.5.10 Reset Second Level in P (Q, R)
  - H1.5.11 Reset Higher Level in P (Q, R)
  - H1.5.12 Advance Second Level in P (Q, R)
  - H1.5.13 Advance Higher Level in P (Q, R)
- H1.6 Examples
  - H1.6.1 Nested Mode
  - H1.6.2 Sequential Mode
  - H1.6.3 Triangular Mode

H2. The Streaming Mode
- H2.1 General Operation
  - H2.1.1 Stream Units P and Q
  - H2.1.2 Stream Unit R
  - H2.1.3 Stream Unit T, the Table Address Assembler (TAA)
  - H2.1.4 Stream Unit U, the Table Extract Unit (TEU)
  - H2.1.5 Statistical Accumulator (SACC) and Threshold Register (THR)
  - H2.1.6 Statistical Counter (SCTR)
  - H2.1.7 Logic Unit (LU)
  - H2.1.8 F
  - H2.1.9 Group Signal (G)
  - H2.1.10 Byte Masks (BM)
  - H2.1.11 Match Units (W, X, Y, and Z)

TABLE OF CONTENTS

H2.2 Harvest Setup
H2.2.1 The Setup Words
H2.2.2 Initial and Developed Values of Parameters
H2.2.3 SKIP and SKAM
H2.3 SBBB - The Stream Byte-by-Byte Instruction
H2.3.1 Data Gating
H2.3.1.1 Gate #0
H2.3.1.2 Gates #1, #2, #3, #4, #5, and #6
H2.3.1.3 Gate #7
H2.3.1.4 Gate #8
H2.3.1.5 Gates #9, #10, and #11
H2.3.1.6 Gates #12 and #13
H2.3.2 GS - Group Size
H2.3.3 The STOP Code
H2.3.4 RC - R Control
H2.4 The Action of the Logic Unit in Byte-by-Byte Streaming
H2.4.1 The Byte-by-Byte Operations
H2.4.2 E, F, and G
H2.5 The Functioning of T and U in Byte-by-Byte Streaming
H2.5.1 T
H2.5.2 U
H2.6 The Match Units (W, X, Y, Z)
H2.7 The Statistical Accumulator and Statistical Counter (SACC and SCTR)
H2.7.1 SACC
H2.7.2 SCTR
H2.7.3 Other quantities in the SACC and SCTR setup words
H2.8 Automatic Adjustments
H2.9 Stream Indicator Bits
H2.10 The Hybrid Instructions
H2.10.1 SUBR - STREAM UNCONDITIONAL BRANCH
H2.10.2 SCBR - STREAM CONDITIONAL BRANCH
H2.10.3 (Not Yet Specified)
H2.10.4 SMOV - STREAM MOVE
H2.10.5 SCSI - STREAM COUNT AND SET INDICATOR
H2.10.6 SCAD - STREAM CONDITIONAL ADJUST
H2.10.7 SCLM - STREAM CLEAR MEMORY
H2.10.8 SSTC - STREAM STORE AND CLEAR
H2.10.9 SMER - STREAM MERGE
H2.10.9.1 Index Control Parameters
H2.10.9.2 Type I - SIMPLE MERGE UP (DOWN)
H2.10.9.3 Type II - MERGE OFFSET UP (DOWN)
H2.10.9.4 Type III - MERGE SPLIT UP (DOWN)
H2.10.9.5 Merging Files of Unequal Length

TABLE OF CONTENTS

H2.10.10 SSER - STREAM SEARCH
H2.10.10.1 Index Control Parameters
H2.10.10.2 Type 0 - RANDOM SEARCH OFFSET
H2.10.10.3 Type I - RANDOM SEARCH SPLIT
H2.10.10.4 Type II - ORDERED SEARCH OFFSET and
Type III - ORDERED SEARCH SPLIT
H2.10.11 SADD - STREAM ADD
H2.10.12 SMPY - STREAM MULTIPLY
H2.10.13 SMLU - STREAM MULTIPLE LOOKUP
H2.10.14 SILS - STREAM INDIRECT LOAD AND STORE
H2.10.15 SSEL - STREAM SELECT
H2.10.15.1 Index Control Parameters
H2.10.15.2 Type 0 - SELECT OFFSET LEAST (GREATEST)
H2.10.15.3 Type I - SELECT SPLIT LEAST (GREATEST)
H2.10.16 (Not Assigned As Yet)

Harvest Abbreviations

A, B, C, D	BASIC registers
ADJ	Adjustment
ADV	Advance
BAR	Byte address register
BM	Byte mask
BP	Branch point
BRA	Branch address
BRAM	Branch to arith. mode
CL	Clear
CON	Connection
CPU	Central Processing Unit
E	"or" of k and/or l and/or m
EC	End of chain
EU	Equal-unequal
F	One bit memory for E
FL	Field length
G	Group
GS	Group size
H	Higher levels address
HL	Higher level
I	Increment
IC	Instruction Counter
IM	Index mode
IX	Index address
J	Accumulated increment
k	One bit signal from LU
K	Cell size in TAA
KK	Cell size for TAA in SILS instruction
l	One bit signal from LU
L	Byte output from LU
LG	Least-Greater bit
LS	Large-small; load-store
LU	Logic unit
m	One bit signal from LU
M	Partial count
MD	Memory distributor
MDM	Memory distributor mode
MN	Match - non match
MOD	Modulus
MS	Match station
MU	Match unit
N	Total count
OFF	Offset
P, Q, R	Harvest registers or SU's
PAR	Parity
PB	Parity Bit
PRI	Priority
RA	Relative-absolute
RC	R store control
RR	Round robin
RS	Return to stream indicator
RST	Reset
S	Effective address
SACC	Statistical accumulator
SAM	SACC mode
SBC	Six bit counter
SCD	Search condition
SCTR	Statistical counter
SGN	Sign modifier
SKAM	Skip to arithmetic mode
SM	Switch matrix
SP	Span
SU	Stream unit
SUP	Supplemental signal
SW	Swallow - non swallow
T, TAA	Table Address assembler
TAM	TAA mode
TB	Test bit
TEU, U	Table extract unit
TG	One bit memory device
THR	Threshold register
TR	Triangular indexing
U, TEU	Table extract unit
UD	Up - down
UL	Upper - lower
V	Start point for TAA address
W, X, Y, Z	Match units or characters

H1. Stream Indexing

Generally speaking, the Harvest Computer obtains data from one or two places in memory, performs some arithmetic or logical operation on the data, and stores the result in memory. The addresses of the source (or sources) and destination of the data are independent of each other, and are modified by independent indexing units. Thus one or two streams of input data (input streams) are processed and an output stream is stored in memory.

The pattern of address modification associated with any data stream is determined by a number of parameters at each of several levels of index control. The control levels are chained together so that there are a first level, a second level, ..., and a highest level of control. There is no practical limit to the number of control levels that can be chained to one stream unit. The values of the parameters for each control level are arbitrary, and are set by the programmer when he sets up the indexing chain prior to executing a stream instruction.

The manner in which the various levels react with one another also is determined by the programmer and specified in the various control levels. Three modes of indexing are available to the programmer; he may choose nested, sequential, or triangular indexing (these are discussed below) by setting control bits at the various levels. Some interplay between the three modes is possible (under the programmer's control).

H1.1 Control Parameters and Control Bits

Each control level in the index chain consists of two words. For every level, each pair of words contains the following parameters and control bits:

N,	Total Count;
M,	Current Count;
I,	Increment:
J,	Accumulated Increment, or Field Offset (depending on index mode);
IM,	Index Mode Control Bit;
TR,	Triangular Indexing Control Bit;
UL,	Upper-Lower Control Bit;
EC,	End of Chain Control Bit.

In addition, all levels higher than the second contain

HL,	Higher-Level Control Bit

During stream indexing the first level control parameters and bits for each stream unit are stored in fixed, addressable registers associated with the particular stream units. These addresses are summarized in Figures H1.1a and H1.1b. The location of the control bits and parameter fields, and the sizes of these fields are shown in Figure H1.1a.

Two additional parameters are kept in addressable Stream Address Registers associated with each stream unit. They are:

S,	Effective Address;
H,	Higher Levels Address.

Their addresses are indicated in Figure H1.1a.

Finally, the parameter

BM,	Byte Mask,

is kept in the Stream Address Registers associated with each stream unit. This will be discussed in Chapter H2.

H1.2 Stream Indexing Setup

In order to prepare the Harvest Computer to execute a stream instruction, the programmer must "set up" a number of parameters and controls. To accomplish this, he may TRANSMIT a set of 20 consecutive "setup" words from somewhere in memory to fixed registers whose addresses are the 20 consecutive word addresses 13 through 32. In particular, the nine setup words 1 through 9 are transmitted to registers having the nine word addresses 13 through 21 (Figure H1.1a). The contents of those setup word fields that are indicated as not used have no effect on Harvest, as the corresponding register positions do not exist.

The first three of these nine setup words contain the parameters (S, H) (I, N, Control Bits) and (J, M) that are stored in the three registers that are associated with SU-P. The second three words contain the corresponding parameters that are stored in the three registers of SU-Q. The last three contain parameters for SU-R. Parameters for all control levels higher than the first are stored in consecutive pairs of words starting at the arbitrary addresses H_o (P), H_o (Q), and H_o (R) corresponding to the index chains for SU-P, SU-Q, and SU-R. Within each control level, the first word must contain (I, N, Control Bits) while the second must contain (J, M). H_o must be a full word address and all index control words must lie in memory whose address is less than 2^{12}. These words are <u>not</u> transmitted to registers at setup time, but are brought into registers automatically as needed.

At the time of the initial setup for a particular stream operation, S contains the starting address, S_o, for the particular stream unit involved, H contains the second control level address, H_o. Generally speaking, J and M are initially zero in the nested and triangular indexing modes. In these cases, the second word of the appropriate control level is entirely zero.

The entire Harvest setup (including index and all other information) is summarized in Figure H2.2. Any parameter (bit or field) may be loaded into or stored from the Harvest setup register by the use of VFL instructions.

H1.3 Operation

The Harvest stream unit indexing controls have been designed to facilitate the control of nested loops, since the read out pattern of most streams can be described in terms of nested loops. Triangular indexing is a special case of nested indexing in which the number of times a program executes a given loop within a loop of the next higher order changes with every execution of the higher order loop. Sequential indexing is also a special case of nested indexing. Since the nested mode of operation is basic to the triangular and sequential modes, it will be described first.

The description of stream indexing that follows is summed up in the flow chart, Figure H1.3.

H1.3.1 Nested Indexing

In this indexing mode the first indexing level controls the byte read out of the stream unit concerned. N_1 bytes at a distance of I_1 bits from each other, having the addresses S_o, $S_o + I_1$, $S_o + 2I_1$, ..., $S_o + M_1 I_1$, ..., $S_o + (N_1 - 1) I_1$ are read out of memory via the stream register onto some data path specified in the stream instruction (see Chapter H2). As each new address is formed it is stored in the effective address register, S, while the value of the current count, M_1, in the M register is increased by 1 and the partial accumulated increment $J_1 = M_1 I_1$ in the J register is incremented by I_1. This is known as a first level advance. Each time S, M_1, and J_1 are updated, M_1 is compared with N_1. If $M_1 \neq N_1$, then another byte is read out; if $M_1 = N_1$, then $S = S_o + N_1 I_1$ is decremented by $J_1 = M_1 I_1 = N_1 I_1$, and J_1 and M_1 are reset to zero. This is the end of the first level and is also the end of the first iteration at the second level. S and J_2 are now incremented by I_2, while M_2 is incremented by 1 and compared to N_2. This is a second level advance.

Assume $M_2 \neq N_2$. then the N_1 bytes $S_o + I_2$, $S_o + I_1 + I_2$, ..., $S_o + M_1 I_1 + I_2$, ..., $S_o + (N_1 - 1) I_1 + I_2$, are read out. Again, $S + I_1$ replaces S, $J_1 + I_1$ replaces J_1, and $M_1 + 1$ replaces M_1. Now $M_1 = N_1$ again, $S - J_1$ replaces S, and J_1 and M_1 are reset to zero. This is again the end of first level and is also the end of the second iteration at the second level. S and J_2 are again incremented by I_2, while M_2 is incremented by 1 and compared to N_2 (another second level advance). The effective address is now $S_o + 2I_2$.

Eventually, during the N_2-th iteration at the second level, S becomes $S_o + (N_1 - 1) I_1 + (N_2 - 1) I_2$. After $S + I_1$ replaces S, $J_1 + I_1$ replaces J_1, and $M_1 + 1$ replaces M_1, then $M_1 = N_1$. Now $S - J_1 = S_o + (N_2 - 1) I_2$ replaces S, and J_1 and M_1 are again reset to zero. Next S and J_2 are incremented by I_2 and M_2 is incremented by 1. At this point M_2 is compared

to N_2 and $M_2 = N_2$, so the second level must be reset. $S - J_2$ replaces S, and J_2 and M_2 are reset to zero. This is the end of second level and is also the end of the first iteration at the third level

In general, given a k-level chain, an end of (j + 1)-th level occurs only immediately after an end of j-th level (j = 1, 2, ..., k-1). After S is decremented by J_j, and J_j and M_j are reset to zero, the effective address is $S = S_o + M_{j+1} I_{j+1} + M_{j+2} I_{j+2} + \ldots + M_k I_k$. After an end of j-th level, the next byte is read from $S + I_{j+1}$ unless this address results in an end of (j + 1)-th level.

H1.3.1.1 Control Bits

(1) EC
Given a k-level chain, the End of Chain Control Bit, EC_j, must be zero for all levels $j<k$. If EC_k has the value 1, then after S, J_k, and M_k are reset the indexing mechanism will halt. Moreover, the End of Chain P, Q, or R indicator bit (see Chapter H2.9) will turn on according as EC_k is in the P, Q, or R index chain. EC_k also activates a Stop signal which the programmer may use to advance the computer to the next instruction.

(2) HL
The Higher-Level Control Bit, HL, is used to mark a level higher than the second for purposes of Adjustment. This will be described in Chapter H1.5.

(3) IM, TR
The Index Mode Control Bit, IM, and the Triangular Indexing Control Bit, TR, must be zero in all control levels in the nested mode.

(4) UL
The value of the Upper-Lower Control Bit, UL, is immaterial in the nested mode.

H1.3.2 Triangular Indexing

This mode is identical to the nested mode except that on the control level designated as triangular, after S, J, and M are reset, N is decremented or incremented by 1 depending on the value of UL.

The programmer must note that although in the nested mode all parameters are reset to their initial values if indexing is terminated by EC being 1 for some level, this is not true in the triangular mode. In this mode

the final value of N_j is not equal to the initial value of N_j. The program must reset this parameter before using this particular setup for another data pass.

H1.3.2.1 Control Bits

(1) EC, HL
These bits are used as in the nested mode.

(2) IM
This bit must be zero.

(3) TR
The Triangular Indexing Control Bit must have the value 1 on each level at which it is desired to change N automatically.

(4) UL
To increment N, set the Upper-Lower Control bit to 1; to decrement N, set UL to zero.

H1.3.3 Sequential Indexing

In the sequential mode, several consecutive control levels beginning with the first control the byte read out of the stream unit concerned.

After each byte is read on any control level designated as sequential, S is incremented by I, M by I, but J is unchanged. Moreover, at the end of any such level, S is incremented by J, J is unchanged, and only M is reset to zero.

Thus each sequential level defines and causes to be read out a field of N bytes at intervals I from each other while J defines the distance in bits (field Offset) between the end of this field and the beginning of the next. In particular, the value of J for the last level in the sequence may be the algebraic distance in bits between the end of the last field and the beginning of the first field in the sequence. The entire sequence then must be nested in the next higher control level.

H1.3.3.1 Control Bits

(1) HL
This bit is used as in the nested mode.

(2) EC
This bit is immaterial in the sequential levels, but has its regular meaning in all levels of order higher than the sequential levels.

(3) IM
This bit must have the value 1 in the sequential levels and must have the value zero in the first non-sequential level. It is immaterial in all others.

(4) TR, UL
These bits are immaterial in the sequential levels but have their usual meaning in all others.

H1.4 Special Values of Indexing Parameters

The parameters M and N are always unsigned, while I and J may be positive or negative (see Figure H1.1a).

In the nested and triangular modes, if I is zero then J is always zero. In the sequential mode, J is independent of I.

N may be zero. But since M (which is non-negative) is always incremented before being compared with N, if N is zero then indexing cannot advance to a higher level and cannot be terminated by any EC bit until 2^{14} iterations (if this is the first level) or 2^{18} iterations (if this is a higher level) have been performed. (M resides in a 14 or 18 bit counter.)

H may be zero. If it is, all parameter and control bits for the second level will be loaded into the stream indexing mechanism as zero, and H will remain unchanged. Under these circumstances M_2 will always equal 1 and N_2 will always equal zero when they are compared. Thus the second level will continue to iterate indefinitely, and indexing must be terminated by some signal that originates outside of the stream indexing mechanism involved.

H1.5 Automatic Adjustment of Index Level

As part of certain streaming mode instructions a number of Adjustment operations are available. Those that adjust stream indexing controls are the subject of this section; the rest are discussed in Chapter H2.8.

H1.5.1 Repeat Byte in P (Q)

The last byte read out of stream register P (Q) is read out again, after which indexing proceeds normally. This is defined for all indexing modes.

H1.5.2 Repeat Byte in R

The next byte read into stream register R is read in over the last byte after which indexing proceeds normally. This is defined for all indexing modes.

H1.5.3 Skip Byte in P (Q)

Suppress the read out of the next byte in P (Q) and then proceed normally. This is defined for all indexing modes.

H1.5.4 Runout First Level in P (Q)

The remaining bytes in P (Q) are read out and bypass the LU, TAA, TEU, and all MS's. Indexing then proceeds normally. This is defined for the nested and triangular modes only.

H1.5.5 Runout Second Level in P (Q)

All bytes read out of P (Q) bypass the LU, TAA, TEU, and all MS's until the end of second level is reached. Indexing then proceeds normally. This is defined for the nested and triangular modes only.

H1.5.6 Runout Higher Level in P (Q)

All bytes read out of P (Q) bypass the LU, TAA, TEU, and all MS's until the end of the higher level whose HL bit has the value 1 is reached. Indexing then proceeds normally. This is defined for all indexing modes; but in the sequential mode HL may only have the value 1 in a nested level.

H1.5.7 Runout Field in P (Q)

The remaining bytes in the present sequential level in P (Q) are read out and bypass the LU, TAA, TEU, and all MS's. Indexing then proceeds normally. This is defined only for the sequential mode.

H1.5.8 Runout Sequence in P (Q)

All bytes in the remaining sequential levels in P (Q) are read out and bypass the LU, TAA, TEU, and all MS's. Indexing then proceeds normally. This is defined only for the sequential mode.

H1.5.9 Reset First Level in P (Q, R)

The effective address, S, in P (Q, R) is decremented by J_1, and J_1 and M_1 are reset to zero. Indexing then proceeds normally. This is defined for the nested and triangular modes only.

H1.5.10 Reset Second Level in P (Q, R)

S in P (Q, R) is decremented by J_1 and J_2, and J_1, J_2, M_1, and M_2 are reset to zero. Indexing then proceeds normally. This is defined for the nested and triangular modes only.

H1.5.11 Reset Higher Level in P (Q, R)

S in P (Q, R) is decremented by J_1, J_2, ..., J_h, and these J's and M_1, M_2, ..., M_h are reset to zero. Here h is that level whose HL bit has the value 1. Indexing then proceeds normally. This is defined for the nested and triangular modes only.

H1.5.12 Advance Second Level in P (Q, R)

S in P (Q, R) is decremented by J_1; J_1 and M_1 are reset to zero; S and J_2 are incremented by I_2; and M_2 is incremented by 1. Indexing then proceeds normally. This is defined for the nested and triangular modes only.

H1.5.13 Advance Higher Level in P (Q, R)

S in P (Q, R) is decremented by J_1, J_2, ..., J_{h-1}; these J's and M_1, M_2, ..., M_{h-1} are reset to zero; S and J_h are incremented by I_h; and M_h is incremented by 1. Here h is defined as in H1.5.11. Indexing then proceeds normally. This is defined for the nested and triangular modes only.

H1.6 Examples

The flow of bytes selected from memory by each stream indexing mechanism and presented to the Harvest data buses fits a well defined pattern which may be simple or complex. Examples are given here which illustrate patterns one could expect in the three indexing modes.

H1.6.1 Nested Mode

A stream might consist of one thousand consecutive six-bit bytes beginning at a particular address. This is a simple stream that requires only one index control level. The values of the parameters for this level are:

Level	I	N	J	M	IM	TR	UL	EC	HL
1	6	1000	0	0	0	0	0	1	0

A more complex stream may be described as follows (see Figure H1.6.1 for first two levels): read out three overlapping eight-bit bytes (each byte overlaps the last half of the preceding byte) and then skip six bits; repeat this three times, each time moving the effective starting point 22 bits, thus giving a six-bit skip; repeat this entire process five times using the original starting point; finally each time all previously described steps have been completed repeat them at a new starting point one-hundred bits from the original starting point, continuing such repetition until a total of ten cycles have been completed. This complex stream requires four levels of index control, whose parameters have the following initial values:

Level	I	N	J	M	IM	TR	UL	EC	HL
1	4	3	0	0	0	0	0	0	0
2	22	3	0	0	0	0	0	0	0
3	0	5	0	0	0	0	0	0	0
4	100	10	0	0	0	0	0	1	0

H1.6.2 Sequential Mode

A pattern of the simpler sequential type is characterized by the following example (see Figure H1.6.2). A record consists of twelve 4-bit decimal digits. It is desired to read out three digits at the beginning of the record, skip two, read out two, and then go to the next record. Three levels of control are used to produce this pattern. Their parameters have the following initial values:

Level	I	N	J	M	IM	TR	UL	EC	HL
1	4	3	8	0	1	0	0	0	0
2	4	2	-28	0	1	0	0	0	0
3	48	-	0	0	0	0	0	-	0

H1.6.3 Triangular Mode

Consider the following array of data: the numbers a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, and a_{33} are arranged consecutively in memory. These numbers represent the elements of three rows of a third order matrix. It is desired to read all elements of the first row, then the last two elements of the second row and finally the last element of the third row. Such a readout is illustrated in Figure H1.6.3a. Such an array of data is called an Upper Right matrix triangle. In order to read out such a pattern, it is necessary only to be able to change the value of the total count, N, on that level of indexing that reads a row of the matrix. For example, if each element of the matrix is a six-bit byte, then the first level of indexing controls the readout of a row of the matrix. In this fourth indexing example, the total count for the first level is initially 3. After reading the first row, the total count is decremented by one. Thus the total count is now 2. After the elements of the second row are read out the total count is again decremented 1 and the count is now 1.

In addition to this it is necessary to increment at the second level by four times the byte length, that is, a multiple of the byte size that is one more than the number of elements in the first row. The size of the increment on the second level remains constant throughout the pattern of indexing. Since there are three rows to be read, the value of the total count for the second level is 3. The initial setup is as follows:

Level	I	N	J	M	IM	TR	UL	EC	HL
1	6	3	0	0	0	1	0	0	0
2	24	3	0	0	0	0	0	1	0

There are four triangles in every matrix. They are designated Upper Left, Upper Right, Lower Left and Lower Right. If it were desired to read the Upper Left triangle from the matrix described above, the output would be as follows:

a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{31}.

The corresponding Lower Right triangle consists of the elements

$$a_{13},\ a_{22},\ a_{23},\ a_{31},\ a_{32},\ a_{33};$$

the Lower Left triangle is

$$a_{11},\ a_{21},\ a_{22},\ a_{31},\ a_{32},\ a_{33}.$$

An analogous pattern of indexing may be applied to vectors. Thus, if a vector consists of the three elements a_1, a_2, a_3, and it is desired to read these elements out in the order a_3, a_2, a_3, a_1, a_2, a_3, then we can say that we are applying Triangular Indexing to the vector. Figure H1.6.3b illustrates precisely this Lower Right vector. Once again there are four triangles, Upper Left, Upper Right, Lower Left and Lower Right. The analogous Upper Left vector triangle produces an array as follows:

$$a_1,\ a_2,\ a_3,\ a_1,\ a_2,\ a_1;$$

the Upper Right vector triangle is

$$a_1,\ a_2,\ a_3,\ a_2,\ a_3,\ a_3;$$

finally the Lower Left vector triangle is

$$a_1,\ a_1,\ a_2,\ a_1,\ a_2,\ a_3.$$

The following table indicates the values of the control bits and the parameters for indexing in this fashion:

Triangle	TR_x	UL_x	Initial Value of N_x	Value of I_{x+1}		Value of N_{x+1}	
				Vector	Matrix	Vector	Matrix
Upper Left	1	0	n	0	nI_x	n	n
Upper Right	1	0	n	I_x	$(n+1)I_x$	n	n
Lower Left	1	1	1	0	nI_x	n	n
Lower Right	1	1	1	$-I_x$	$(n-1)I_x$	n	n

n is the order of the matrix, or the dimension of the vector. In all cases $I_x > 0$.

H2. The Streaming Mode

Harvest is designed to operate upon a sequence of bytes in an extremely rapid and automatic manner. This speed is attained by the simultaneous functioning of several independent indexing and addressing mechanisms (described in Chapter H1) and by the sequential processing of bytes by units working in an assembly-line fashion. In addition to the various processing units a counter and an accumulator have been provided for calculating pertinent statistics during the course of byte processing.

In one respect the functioning of Harvest in the streaming mode may be compared to that of a plug board computer. That is, the operation to be performed and all the relevant parameters are specified before any action takes place; then the execution of a single streaming instruction may cause the logical or arithmetic manipulation of many thousands of data bytes. Of course the big difference is that the "plugging" in Harvest is all done under program control and can be changed completely in a number of microseconds. Moreover the arithmetic and logical facilities of a general-purpose computer are always at hand to resolve difficulties that cannot easily be resolved in the streaming mode.

To a large extent the streaming mode is conceptually separate from the arithmetic mode. The arithmetic section contains the instruction "BRANCH TO Y AND ENTER STREAMING MODE". Upon this command the complexion of Harvest changes; the manifold processing units are activated, and the philosophy of information handling and processing changes. One of the basic differences between the arithmetic mode and the streaming mode of operation is that in the streaming mode the programmer has the responsibility of designating how the Instruction Counter must change from instruction to instruction, while in the arithmetic mode this is, in general, automatic. When a streaming instruction indicates a shift to the arithmetic mode, the normal arithmetic functioning of the Basic computer is resumed.

In order to diminish the necessity for frequent alternations between the streaming and arithmetic modes and to gain speed and flexibility in some vital manipulative operations, the streaming mode is equipped to perform automatically many kinds of "adjustments" upon the stream of data and the operation applied to it. In addition to the automatic adjustments the streaming mode has a small repertoire of "hybrid" instructions which perform basic arithmetic and logical functions but which utilize the various independent indexing and addressing facilities of Harvest to gain speed and generality. Lastly, a few streaming conditions are represented in the indicator register and may stimulate action there upon completion of the streaming instruction.

H2.1 General Operation

Figure H2.1 shows the various functional units of the Harvest CPU and the data paths and gates that connect them. The general operation of these units is described in the sections that follow.

H2.1.1 Stream Units P and Q

These units control the input data streams. Each consists of a double length (128-bit) register (P_0 - P_1 or Q_0 - Q_1), an associated switch matrix (SM_P or SM_Q), and a built-in indexing unit (see Chapter H1). The SU built-in controls permit automatic loading of the registers with 64-bit words from memory, normally without any interruption in the stream of bytes being generated. The streams consist of 8-bit bytes that are read from the registers and switched to the byte buses for processing.

H2.1.2 Stream Unit R

This unit (consisting of a double length register, R_0-R_1, the switch matrix SM_R, and a built-in indexing unit) is used primarily to store a data stream in memory. It switches 8-bit bytes from the byte bus to the proper positions in its double length register. The contents of these registers are then normally stored automatically as full 64-bit words in memory.

H2.1.3 Stream Unit T, the Table Address Assembler (TAA)

The Table Address Assembler provides a means of assembling bytes from one or more SU's together with the contents of a register and a program-controlled counter, the Memory Distributor (MD), to form addresses for table lookup, counting in memory, and other similar memory reference functions.

The word referenced in memory may have a 1 added or or-ed to it in a designated position. Moreover, the original, unchanged word referenced may be sent to the Table Extract Unit (TEU).

The TAA consists of a 24-bit register, T, an associated switch matrix, SM_T, a base address register, the memory distributor, MD, and a simplified built-in indexing unit.

H2.1.4 Stream Unit U, the Table Extract Unit (TEU)

The Table Extract Unit is a data storage register similar in design to the Stream Units. Its primary function is to receive data words from memory as addressed by the Table Address Assembler and to select from them the appropriate bytes for transfer to other units in the computer.

A single length register, U, a switch matrix, SM_U, and a single-level built-in indexing unit make up the TEU.

H2.1.5 Statistical Accumulator (SACC) and Threshold Register (THR)

The SACC accumulates input bytes that come from the Logic Unit, LU, or from the TEU. It may also be stepped by any one of several 1-bit signals from various parts of the CPU. Associated with the SACC is a special Threshold register, THR, which may be loaded as a part of the setup operation. Whenever the value in SACC reaches or exceeds the value in THR, a special 1-bit signal is produced. This signal may cause special action to take place in the CPU.

H2.1.6 Statistical Counter (SCTR)

The SCTR may be stepped by practically any 1-bit signal from the Harvest CPU. Its contents may be gated onto the byte bus to SU-R by Adjustment operations. The SCTR may be reset automatically. It also may be connected to SMP and have its current contents continuously available for logical operations.

H2.1.7 Logic Unit (LU)

The LU performs a variety of arithmetic and logical operations on the successive bytes or pairs of bytes presented to it. It provides an 8-bit output, L, and three 1-bit outputs, k, l, and m. L may be gated onto the byte bus. A selected set of the 1-bit signals, k, l, and m are or-ed together to produce a 1-bit signal, E. This signal may also be put on the byte bus, or be used to cause Adjustments.

H2.1.8 F

F is a small memory device used primarily to record the changes in E. It may be gated to the byte bus or may automatically adjust the flow of data in the CPU.

H2.1.9 Group Signal (G)

G is a memory device used to record the status of groups of bytes that pass through the LU. The size of the group is specified in the particular stream instruction being used. The status recorded is a function of the three 1-bit signals, k, l, and m. The output of G is an 8-bit byte that may be gated to R, or be used to cause Adjustment.

H2.1.10 Byte Masks (BM)

In order to provide a ready means of operating upon any size byte from 1 through 8 bits and on any subset of bits within a byte the four byte masks BM_P, BM_Q, BM_R, and BM_U are provided in association with the corresponding SU's. In particular, BM_R indicates which bits are to be stored in a memory word and which are not. The mask format is specified as a part of the streaming setup (see Section H1.1 and Figure H1.1a).

H2.1.11 Match Units (W, X, Y, and Z)

During the streaming mode of operation it is possible to monitor input and output streams for the occurrance of prespecified bytes. The four units W, X, Y, and Z provide a capacity for specifying four different special bytes at any one time. Automatic adjustment of the data flow may be initiated upon recognition of a prespecified byte.

H2.2 Harvest Setup

In order to execute a Harvest stream instruction, the programmer must initialize, or set up, the Harvest CPU. He accomplishes this by loading various memory positions – i.e., set up registers – with the proper values of a number of parameters and control bits while in the arithmetic mode of operation.

Figure H2.2 shows the array of twenty consecutive words (forty consecutive half words) that comprise the complete Harvest setup. The setup registers are loaded, just as any memory position is loaded, by STORE, TRANSMIT, SWAP, etc., instructions. For example, if the twenty word setup array is stored in memory beginning at location x + 1, then the parameters contained in the third setup word (righthand column of numbers, Figure H2.2) may be loaded in to the appropriate setup register by TRANSMIT two half words from x + 3 to 15 (lefthand column of numbers, Figure H2.2). By TRANSMIT forty half words from x + 1 to 13, all the setup parameters and control bits would be loaded in the appropriate setup registers. VFL instructions may be used to load or store any particular parameter or control bit. In Figure H2.2, the number directly under a field is its bit address; the number in the field is its field length (one-bit field lengths are not designated).

H2.2.1 The Setup Words

Setup words 1 through 9 and 15 through 20 have been described in Section H1.2. They contain the stream index control parameters and control bits.

Setup word 10 contains the parameters for the TAA and the MD. Setup word 11 contains the parameters for the TEU. The use of these parameters and control bits is described in Section H2.5.

Setup word 12 contains the parameters for all four MU's. Their use is discussed in Section H2.6.

Word 13 contains the parameters for the SACC and SCTR, while word 14 contains the THR. These parameters are described in Section H2.7.

Setup word 14 also contains E, F, G (discussed in Section H2.4.2), MOD (Section H2.4.1), and RS (Section H2.9).

H2.2.2 Initial and Developed Values of Parameters

The values of most of the parameters and control bits must be designated by the programmer in the setup before streaming begins. However, certain parameters and control bits have values that are developed during the streaming mode of operation of Harvest, S_U, BA_T, E, F, G, RS. The programmer is concerned with the values of these parameters only if streaming has been interrupted; he is not concerned with their initial values. The abbreviations for these fields are in parentheses in Figure H2.2.

H2.2.3 SKIP and SKAM

As was indicated in Section H2, while Harvest is in the streaming mode the programmer has the responsibility of designating how the Instruction Counter, IC, must change from instruction to instruction. In order to accomplish this, the programmer uses a four bit (three bits plus sign) SKIP field (Figure H2.10) in every stream instruction (whether byte-by-byte or hybrid). The value of this field is added algebraically to the current value of the IC at the end of execution of the stream instruction, giving IC + SKIP as the location of the next instruction to be executed.

The programmer also must indicate whether or not the next instruction is in the arithmetic mode. If the control bit SKAM (skip to arithmetic mode) has the value 1, then the next instruction is in the arithmetic mode; if the value is zero, then the computer remains in the streaming mode.

H2.3 SBBB - The Stream Byte-by-Byte Instruction

Byte-by-byte operations require additional parameters not specified in the setup words. These parameters are presented in the Stream Byte-by-Byte Instruction itself. They include:

a) Data Gating - the interconnection of the various processing units (H2.3.1)
b) Op Code - the operation to be performed in the Logic Unit (H2.4.1)
c) MOD - the modulus in modular operations; an extra available 8-bit byte in others (H2.4.1)
d) F Mode - the behavior of F (H2.4.2)
e) GS - the group size (H2.3.2)
f) STOP - the criterion for passing to the next instruction (H2.3.3)
g) RC - the control of R storage (H2.3.4)
h) SKIP and SKAM - the progression to the next instruction (H2.2.3)

H2.3.1 Data Gating

The data paths that are to be used in a byte-by-byte stream operation are specified by the first fourteen bits of the instruction. Each of these fourteen bits refers to a particular gate in the network of data paths (see Figure H2.1). Some of the gates act merely as switches: i.e., they select between two possible sources. Most act as true gates: i.e., they permit or prevent the passage of data. The specific actions of and restrictions on each gate are described in paragraphs H2.3.1.1 - H2.3.1.6. These restrictions insure that the expected result of an operation is actually obtained. Other combinations are at the user's risk.

H2.3.1.1 Gate #0

0 = Data from Register P enters SM_P
1 = Data from SCTR enters SM_P

This gate is used to put the current contents of SCTR on the byte buses for further processing.

In addition to specifying that Gate #0 is set at 1 one must also set up Register P indexing to specify what exactly is read out of SCTR. For this purpose the SCTR acts as a register whose bits 34-49 are the current value of the SCTR, the other 48 bits all being 0. Only the bit portion of the addresses generated by P indexing are used when Gate #0 is set at 1.

H2. 3. 1. 2 Gates #1, 2, 3, 4, 5, and 6

These gates control the paths:

1: P → LU
2: Q → LU
3: U → LU
4: P → T
5: Q → T
6: LU → T

For all these gates:

0 = closed; no data passes
1 = open; data passes

Gates #2 and #3 cannot be open simultaneously. They both refer to one input to the LU.

Gates #5 and #6 cannot be open simultaneously. They both refer to one input to T.

Gates #3 and #6 cannot be open simultaneously. This "Figure 8" path has been dis-allowed because of the extreme difficulty in controlling it satisfactorily.

Gate #6 cannot be open if Gate #2 is closed. The second level of indexing of T is controlled by Q when Gate #6 is used.

At least one of Gates #1, #2, #4, and #5 must be open. Some input is needed.

Gate #3 cannot be open if both Gates #4 and #5 are closed. There must be an input to T if something is to emerge from U.

Thus there are altogether 25 allowable combinations for these six gates:

any xx0xx0 except 000000
x011x0
x01010
x10x01

For combinations 101100, 101110, and 110101: N_1P must be 1.
For combination 110110: N_1P and N_1Q must both be 1.

H2.3.1.3 Gate #7

0 = E → Gate #8
1 = F → Gate #8

H2.3.1.4 Gate #8

0 = L → MS_L
1 = Output of Gate #7 → MS_L

If the output of Gate #7 is put on the byte bus to MS_L, the significant bit is put on the leftmost of the eight lines. The righthand seven bits are made 0.

H2.3.1.5 Gates #9, #10, and #11

000 = no readin to R
001 = U → R
010 = output of MS_L → R
011 = not allowed
100 = G → R at end of group
101 = not allowed
110 = not allowed
111 = G → R at every byte

H2.3.1.6 Gates #12 and #13

00 = no input to SACC
01 = U → SACC
10 = output of MS_L → SACC
11 = not allowed

H2.3.2 GS - Group Size

The GS field defines a group length in terms of the indexing parameters of the Stream Units. This length is used for control of G and assorted adjustment operations. The field is 4 bits long. The first two bits identify the level of indexing:

00 = end of byte
01 = end of first level
10 = end of second level
11 = end of higher level (defined by HL control bit – see H1.1)

The second two bits identify the controlling unit:

00 = SU-P
01 = SU-Q
10 = SU-R
11 = SU-U

Thus 0110 defines the group size as the second level of R. Codes 1011 and 1111 are defined to say that no group size is specified.

H2.3.3 The STOP Code

The STOP Code specifies when to terminate this byte-by-byte instruction and progress to the next instruction. If the instruction is stopped in some other way before this point is reached, this field naturally does not apply. The first two bits specify the end-of-level signal which controls the STOP:

00 = end of byte
01 = end of first level
10 = end of second level
11 = end of chain (defined by EC control bit – see H1.1)

The second two bits identify the controlling unit:

00 = SU-P
01 = SU-Q
10 = SU-R
11 = SU-U

Thus 0100 specifies STOP at the end of the first level in P. Codes 1011 and 1111 are defined as NO STOP. Some other means of getting to the next instruction must then be arranged.

H2. 3. 4 RC – R Control

The first bit of this two-bit field designates how R is to store information during byte-by-byte operations:

0 = The data formed in R is to be sent to memory, totally obliterating any previous data in that memory word. Any bits not specified will be made 0's.
1 = The data formed in R will replace the corresponding bits in memory but not affect adjacent bits. This is generally slower than option 0, but more selective.

In option 0 the R register is then initially all 0's. In option 1 the R register initially contains the contents of the memory word in which the data will be stored. As each byte enters, the bits opposite 1's in BM_R will replace the bits already in R; the bits opposite 0's in BM_R will have no effect.

The second bit designates how R is to behave at the conclusion of a byte-by-byte operation:

0 = R is to be stored before proceeding. (This insures that results which do not occupy a full word will still be stored.)
1 = R is not to be stored. (Storage is to be avoided in some types of instruction loops.)

H2.4 The Action of the Logic Unit in Byte-by-Byte Streaming

During streaming one may obtain any possible function of a set of bytes by means of the table lookup facility. Nevertheless a separate Logic Unit is provided. There are several reasons: many functions are extremely simple and regular; table lookup slows down when references to memory occur in a random order; large tables can often be reduced in size by first performing a simple function on the operands; if a table is to be used, it must first be read into memory while built-in functions in the LU are always immediately available.

The LU accepts an input byte from either or both of two sources and produces one output byte. In a streaming operation the LU performs the same function upon each set of sequentially presented inputs. The function, routing of the bytes, and other details of the stream operation are specified in advance by means of a series of setup instructions and by the stream instruction itself. The setup remains fixed until specifically altered.

The eight lines from registers P and Q or U lead directly to the eight positions of the LU. The lowest-numbered bit (the leftmost or high-order bit) of an input register goes to the leftmost position of the LU.

The 8-bit outputs of the LU correspond bit by bit to the inputs. All eight bits may travel to each of three possible recipients: R, T, and SACC.

In addition to the byte output the LU produces three primary 1-bit signals – k, l, and m – which are combined in units E, F, and G. These are described in Section H2.4.2.

H2.4.1 The Byte-by-Byte Operations

In the definitions of the operations P represents the input from P. Q represents the input from Q or U, depending on the data gating. If a function requires two operands – and most do – and only one gate to the LU is open, the missing operand is automatically taken to be all 0's.

For logical operations bytes are regarded merely as an ordered set of bits. For comparison and modular arithmetic bytes are regarded as unsigned positive numbers in binary form.

To obtain the expected answer in modular arithmetic P, Q, and MOD must all have the same byte size – just sufficient for representation of MOD – and be positioned as far to the left as possible. P and Q must already have been reduced to a value between 0 and MOD-1, inclusive. When MOD is specified in the setup (see Figure H2.4) the pertinent bits are to the left; empties on the right are filled with 0's. BM_P and BM_Q must have 1's in those positions corresponding to the proper byte size with 0's in the other (righthand) positions. For example; proper use of a modulus of 22 requires:

$BM_P = BM_Q$ = 11111000

MOD = 10110000

bytes from P, Q = xxxxx---

(A MOD of 10110000 is used not only for 22, but also for 44, 88, and 176. Which it acts as is determined by the placement and size of P and Q.)

The 32 Byte-by-Byte logical operations are specified by five bits of the op code of the stream instruction. See Figure H2.4 (the sixth bit says that the instruction is a byte-by-byte instruction rather than a hybrid instruction). They are defined as follows:

OP Code	L = Byte Output		k = 1	l = 1	m = 1
0	Logical Connection 0		Connection all 0	Connection of even parity but not all 0	Connection of odd parity
1	"	1	"	"	"
2	"	2	"	"	"
3	"	3	"	"	"
4	"	4	"	"	"
5	"	5	"	"	"
6	"	6	"	"	"
7	"	7	"	"	"
8	"	8	"	"	"
9	"	9	"	"	"
10	"	10	"	"	"
11	"	11	"	"	"
12	"	12	"	"	"
13	"	13	"	"	"
14	"	14	"	"	"
15	"	15	"	"	"

OP Code	L = Byte Output	k = 1	l = 1	m = 1
16	Max (P, Q)	P > Q	P = Q	P < Q
17	Min (P, Q)	"	"	"
18	P if P = Q, otherwise all 0	"	"	"
19	P if P = Q, otherwise no byte output	"	"	"
20	P if P ≠ Q, otherwise all 0	"	"	"
21	P if P ≠ Q, otherwise no byte output	"	"	"
22	Q if P ≠ Q, otherwise all 0	"	"	"
23	Q if P ≠ Q, otherwise no byte output	"	"	"
24	P - Q if P≥Q, otherwise all 0	"	"	"
25	P - Q if P≥ Q, otherwise no byte output	"	"	"
26	Q - P if Q≥P, otherwise all 0	"	"	"
27	Q - P if Q≥P, otherwise no byte output	"	"	"
28	P - Q modulo MOD (MOD≤255)	"	"	"
29	Q - P modulo MOD "	"	"	"
30	P + Q modulo MOD "	"	"	"
31	P + Q modulo MOD "	P + Q > M	P + Q = M	P + Q < M

The 16 Logical Connections referred to above are:

Code	Logical Connection	PQ 00	01	10	11
0	0	0	0	0	0
1	$P \cdot Q$	0	0	0	1
2	$P \cdot \overline{Q}$	0	0	1	0
3	P	0	0	1	1
4	$\overline{P} \cdot Q$	0	1	0	0
5	Q	0	1	0	1
6	$P \neq Q$	0	1	1	0
7	$P \vee Q$	0	1	1	1
8	$\overline{P} \cdot \overline{Q}$	1	0	0	0
9	$P \equiv Q$	1	0	0	1
10	$\overline{Q}$	1	0	1	0
11	$P \vee \overline{Q}$	1	0	1	1
12	$\overline{P}$	1	1	0	0
13	$\overline{P} \vee Q$	1	1	0	1
14	$\overline{P} \vee \overline{Q}$	1	1	1	0
15	1	1	1	1	1

H2.4.2 E, F, and G

The three primary 1-bit signals of the LU – k, l, and m – are manipulated in the units E, F, and G to produce other signals which enter into the streaming operations. These signals are all calculated at the same time as the byte L. Since they occur in the first segment of the byte path – from input register to LU – they are available for controlling the indexing of the registers feeding the LU.

E is the final one-bit output of the Logic Unit. It may be put on the byte bus, cause SCTR to step, or cause an adjustment. It is calculated by specifying in a 3-bit field in the byte-by-byte instruction which of the bits k, l, and m are to be or-ed together. The three bits correspond to k, l, and m in that order. Thus an E field of 101 in the instruction means that E consists of the "or" of k and m.

F is a small memory device used to record changes in E. It takes its input from E and changes state according to a 3-bit F field in the byte-by-byte instruction. The coding there means:

000	Stay on 1 after the first one has been received
001	Change state on 1
010	Set to 1 if the previous bit was 0, the current bit 0
011	" 0, " 1
100	" 1, " 0
101	" 1, " 1
110	Set to 1 if the previous bit and the current bit are the same
111	" are different

F is set to 0 at the beginning of this byte-by-byte instruction and at every end-of-group signal occurring during the instruction. The output of F may be put on the byte bus, cause SCTR to stop, or cause an adjustment.

G is a memory device used to infer and record the status of groups from analysis of the bytes within the groups. It is fed directly by k, l, and m from the Logic Unit. G may stimulate the SCTR or cause an adjustment. Its output may go to R through Gate 9.

This output is an 8-bit byte:

a) For logic operations 0-15:

Bits 0 and 1: 00 = (does not arise)
01 = Group connection all 0
10 = Group connection of even parity but not all 0
11 = Group connection of odd parity

Bit 2 = 1 if group connection all 0, 0 if not all 0

Bit 3 = 1 if group connection of even parity, 0 if of odd parity

Bit 4 = 1 if group connection of even parity but not all 0, 0 if of odd parity or all 0

Bit 5 = 1 if group connection not all 0, 1 if all 0

Bit 6 = 1 if group connection of odd parity, 0 if of even parity

Bit 7 = 1 if group connection of odd parity or all 0, 0 if of even parity but not all 0

b) For operations 16-31:

Bits 0 and 1: 00 = (does not arise)
01 = Group P $>$ Group Q
10 = Group P $=$ Group Q
11 = Group P $<$ Group Q

Bit 2 = 1 if Group P $>$ Group Q, 0 if Group P $\leq$ Group Q
Bit 3 = 1 " $\geq$ " , 0 if " $<$ "
Bit 4 = 1 " $=$ " , 0 if " $\neq$ "
Bit 5 = 1 " $<$ " , 0 if " $>$ "
Bit 6 = 1 " $\leq$ " , 0 if " $\geq$ "
Bit 7 = 1 " $\neq$ " , 0 if " $=$ "

c) The output outlined in (b) above does not really have a straightforward interpretation for operation 31. However it is put in this category because of the similarity of its k, l, and m outputs to those of operations 16-30.

d) Selection of the bits for storage thru R is made by BM_R and the R indexing.

H2.5 The Functioning of T and U in Byte-by-Byte Streaming

The Table Address Assembler (T) and the Table Extract Unit (U) are provided to facilitate the referencing and altering of tabular quantities stored in memory. T forms table addresses from a Base Address and one or more bytes, while U converts the extracted word into bytes for further processing.

The word referenced by T may be extracted. If the word lies in the 2-microsecond memory it may also have a 1 or-ed in the addressed bit after extraction takes place. If the word lies in the 1/2-microsecond memory the same or-ing can take place. Alternatively, in the 1/2-microsecond memory only, the 1 may be added in a selected position after the extraction. These three functions are referred to as "extraction", "or-ing in memory", and "counting in memory".

For extraction, the address formed by the TAA refers to the left-hand bit of the first byte that is extracted by the TEU. For or-ing in memory, the address formed by the TAA refers to the bit in memory to which a 1 is or-ed. For counting in memory, the address formed by the TAA – after a modification – refers to the bit in memory to which a 1 is added. For a combined action e.g., extract and count – the formed address may serve two different functions.

H2.5.1 T

The TAA may accept bytes from two sources. The first is the output of MS_P through Gate #4, that is, either P or SCTR. The second is either the output of MS_Q or the output of MS_L, depending on whether Gate #5 or Gate #6 is open. How the bytes enter and are positioned in the address is determined both by the parameters given in the T setup word (#10) and by end-of-level signals coming from P and Q.

The sequence of steps performed in forming the table address are:

1) TBA (the Base Address, specified in the setup word) enters the cleared address accumulator.

2) If Gate #4 is set at 1, a set of bytes from the first source enters the address accumulator via the T Switch Matrix and is added to the Base Address. The number of bytes entered is N_{1P}. The first byte is positioned so that its leftmost bit is at the start point V (specified in the setup). (The positions in the table address are labelled 0-23, as are the other addresses.) The next byte is positioned so its leftmost bit is at the point $V + I_{1T}$. The next, at $V + 2I_{1T}$. At the

end-of-first-level-in-P signal, readin from the first source is discontinued. The position of entry of the last byte from this source now has another I_{1T} added to it.

If Gate #4 is set at 0, there is no read-in from the first source. V is not incremented.

3) a) If Gate #4 was set at 1, the start point for the second source is $V + N_{1P} I_{1T}$.
If Gate #4 was set at 0, the start point for the second source is V.

b) If either Gate #5 or #6 is set at 1 but not both, a set of bytes from the second source enters the address accumulator via the T Switch Matrix and is added to the current contents. The number of bytes entered is N_{1Q}. If Gate #5 is open the bytes come directly from Q. If Gate #6 is open the bytes come from the Logic Unit. However in this latter case Gate #2 must also be open – the actual entry can be 0, if necessary. Then the end-of-level signal from Q will flow along with the bytes to the TAA and govern the end of the readin. The first byte is positioned as in (3a) above. The next byte is at that position plus I_2, etc., until all have been read in.

c) If neither Gate #5 or Gate #6 was set at 1, nothing enters from source two, and the assembling of bytes is complete at the end of the readin from source one.

4) If during the readin of bytes from either source the input byte "hangs over" the right edge of the address mechanism, the excess bits are disregarded. If the positioning address of the input byte is negative the entire byte is disregarded.

5) If the Memory Distributor Mode Bit (MDM) is 0 the assembling of the table address is now complete. If MDM is 1, this further operation takes place: Bits 0-17 of the assembled address are each shifted two places left. (This destroys Bits 0 and 1.) The two bits from the Memory Distributor field (MD) are put in the vacated positions 16 and 17. The MD then is increased by 1 modulo 4 and will insert this increased value into the next assembled table address. It keeps counting modulo 4 on every subsequent address and thus cycles the addresses through the four interleaved memory units.

6) If extraction is to be performed, the bit portion of the assembled address is sent to the TEU. If or-ing in memory is to take place, the bit portion of the address designates where in the memory word the 1 is to be or-ed. If counting in memory is to take place, the bit portion of the address is further modified to determine where in the memory word the 1 is to be added. See below.

A 3-bit field in the T setup word specifies the TAA mode (TAM). The bits refer, respectively, to counting in memory, or-ing in memory, and extraction, and a 1 means that the function takes place. More precisely:

TAM:
- 000 = No memory reference is made. The 24-bit table address is sent directly to the first 24 bits (#0-23) of U.
- 001 = The referenced word is extracted from memory and sent to U.
- 010 = A 1 is or-ed to the referenced word in memory at the position designated by the bit portion of the address.
- 011 = The successive performance of operations 001 and 010.
- 100 = (1/2-microsecond memory only) A 1 is added to the last position of the cell in memory containing the addressed bit. (See below for explanation of cell size.)
- 101 = (1/2-microsecond memory only) The successive performance of operations 001 and 100.
- 110, 111 = Not defined.

Counting in memory may take place in cells of size 8, 16, or 32. A counting cell resides entirely within a single memory word and a word is completely filled with cells of one size. For the three possible sizes of cells, one memory word contains either 8, 4, or 2 cells respectively. The one is added to the righthand bit of the cell. If a cell should attempt to overflow, the entry in the cell will become all 0's, but the attempted carry will not propagate to the next cell. Moreover the Memory Counter Overflow Indicator (Bit #43) will go on so that one may check at the end of the counting operation to see if any cell did overflow. (A more current check may be managed for 8-bit cells by monitoring the output of U in MS_U. However the necessary Match Unit is not always available.)

The cell size is determined by the 2-bit K field in the setup word:

- 00 = 8
- 01 = 16
- 10, 11 = 32

The coding sets up the overflow control. To determine where the 1 is to be added to the memory word, the K field is or-ed to bits 19 and 20 of the assembled table address. In addition, the number 111 is or-ed to bits 21, 22, and 23 of the assembled address. Thus the 1 can be added to the memory word only in every 8th position – i.e., at bit positions 7, 15, ..., 63.

Bit #50 of the T setup word is the Replace Bit (RPL). If it is 0, everything procedes as described above. If it is 1, everything procedes as described for the formation of the first table address. However then the initial base address is replaced by the just-formed first address. And on every subsequent address formation the base address used is the table address formed just previously.

H2.5.2 U

The address of the word entering the Table Extract Unit is placed in the word portion of S_U. The bit address has already come from the TAA and has been put both in the bit portion of S_U and in the BA_T field. The left bit of the first byte to be read out is specified by this bit address. After the first byte has been read out, the bit portion of S_U is increased by I_U; this gives the location of the second byte, etc. The number of bytes read out is N_U. The only index reset available in U is to the beginning of the reference. In this case BA_T replaces the bit portion of S_U, and M_U is set to 0.

If the bit address of the byte to be extracted is so close to the right edge of the word that fewer than eight bits can be obtained from U, the overhanging ones are automatically made 0. If the bit address goes higher than 63 the carry to the word portion of the address is suppressed. Thus there is cyclic readout in a slightly degraded sense.

S_U and BA_T are not set up. They always get their information from T. If the operation stops at any time S_U indicates the address of the last byte read out. BM_U specifies Byte Mask U.

H2.6 The Match Units (W, X, Y, Z)

Four Match Units are provided to monitor the information passing on the byte buses. The Match Units may be connected to the buses at any of four Match Stations (MS's) within the computer. MS_P monitors the output of P; MS_Q, the output of Q; MS_L, the output of the Logic Unit and its associated signals; and MS_U, the output of U. Several Match Units may be connected to the same Match Station, but no unit may be connected to more than one Match Station.

The Match Units normally look at every byte. Some automatic adjustments may specifically disable them for one or more bytes. The RUNOUT indexing level adjustments also disable them.

The operation of the Match Units is specified in one setup word. For each unit there are 14 bits, and the same format applies to all:

First 8 bits: The Match Character (W, X, Y, Z)

Next 3 bits: The point of connection (CON)
- 0xx: No connection
- 100: MS_P
- 101: MS_Q
- 110: MS_L
- 111: MS_U

Next bit: The Span Bit (SP)
- 0: The Match Unit compares all 8 bits of the passing bytes with all 8 bits of the Match Character.
- 1: The Match Unit compares only the rightmost bit of the passing bytes with the rightmost bit of the Match Character.

Next bit: The Swallow Bit (SW)
- 0: Regardless of whether or not a match occurs, the bytes pass the Match Station without alteration.
- 1: If a byte is matched (regardless of SP or MN setting) the entire byte is suppressed – taken off the bus – and the next byte from that sending unit is automatically called for.

Last bit: The "Match or No-match" Bit (MN)
- 0: An adjustment is performed on a match.
- 1: An adjustment is performed on a no-match.

Note: Swallowing takes place only on matches; adjustment, on either matches or no-matches.

The Match Units may send counting pulses to the SCTR or the SACC – this is determined by the SCTR and SACC setup in the byte-by-byte stream instruction. However their main function is to signal automatic adjustments. The word following the byte-by-byte instruction specifies what these adjustments are to be. See H2.8.

In the specification of adjustments the "simultaneous observation of W and X or of Y and Z" is referred to. The pair W and X are defined to signal adjustment simultaneously only when their specified recognition conditions have arisen and they are connected to the two byte buses entering the Logic Unit (one to each bus). If P and Q supply the entries, one of W or X must be connected to MS_P, the other to MS_Q. If P and U supply the two entries, one of W or X must be connected to MS_P, the other to MS_U.

If W or X signal simultaneously in this sense, the W-X adjustment field is first consulted. If this says "sum of actions", the W adjustment is first performed, then the X adjustment. If it says anything else, this W-X adjustment is performed and the individual W and X adjustments are disregarded.

Y and Z simultaneity is defined similarly.

Because of the timing involved in an assembly-line processing of data, the Match Units are not free to perform unrestricted adjustments. The restrictions that insure meaningful results are detailed in H2.8.

H2.7 The Statistical Accumulator and Statistical Counter (SACC and SCTR)

H2.7.1 SACC

The basic function of SACC is to accumulate bytes from either the Logic Unit or the Table Extract Unit. The input from the Logic Unit is usually either E or F signals or the data itself from P or Q. The input from U is generally more important in that two frequently-used statistical functions can be calculated during a table addressing operation. a) During a count in memory the function

$$\frac{1}{2} \sum_i f_i (f_i - 1) = \sum_i \sum_{x=0}^{f_i - 1} x$$

can automatically be accumulated. b) By using ordinary extraction the function $\sum_i f_i W_i$ can be accumulated. An initial value of SACC may be set up, and everything is then added to this.

Bytes entering SACC may be unsigned or signed. The bits in SACC are numbered 0-24, with #24 being its sign. Unsigned bytes enter bits 16-23. With signed bytes the lefthand 7 bits enter bits 16-22 while the rightmost bit of the byte interacts with the sign bit of SACC. In addition there is the option of having SACC automatically reset to 0 when its value becomes negative. The first bit of the SACC mode field (SAM) says:

0 = unsigned bytes
1 = signed bytes.

The second bit of SAM says:

0 = normal accumulation
1 = reset negative values.

Associated with SACC is a Threshold Register (THR) which is set up in advance. When the SACC value equals or exceeds the THR value a signal is generated which sets a bit in the Indicator Register and which may also cause a count or automatic adjustment.

Whether or not the SACC is being used as an accumulator it may also have one-bit counts entered into it or be automatically reset to zero, as governed by the SACC STEP and SACC RST fields in the stream byte-by-byte instruction. The counts are entered into Bit 22; this is equivalent to counting by 2's. The reset is always to 0, not to the initial value.

The coding for SACC STEP gives the source of the counting pulse:

0000 = no count
0001 = E = 0
0010 = E = 1
0011 = F = 0
0100 = F = 1
0101 = G = 01 at end-of-group
0110 = G = 10 at end-of-group
0111 = G = 11 at end-of-group
1000 = match in W
1001 = match in X
1010 = match in Y
1011 = match in Z
1100 = operation in LU
1101 = byte enters R
1110 = end-of-group
1111 = STOP.

The coding for SACC RST gives the source of the resetting pulse:

00 = no reset
01 = match in Z
10 = end-of-group
11 = STOP.

Care must be used in specifying SACC counting and resetting as some asynchrony with the byte stream may be involved. Counts are entered in addition to any bytes being entered. If the same stimulus should be specified for both counting and resetting, the reset will take place and not the count.

SACC overflow and underflow are also represented in the Indicator Register.

H2.7.2 SCTR

SCTR is a counter which can count practically any kind of signal arising in the Harvest processing units. Its current contents may also be put onto the byte buses by means of Gate #0. (See H2.3.1.)

The pulses which step or reset the counter are designated in the setup word. The SCTR STEP field indicates the source of the count:

000000	No count	
000001	Byte from P	If Gate #0 is set at 1 these mean the readout of the SCTR controls its stepping.
000010	EOL_1P	
000011	EOL_2P	
000100	EOL_HP	
000101	Byte from Q	
000110	EOL_1Q	
000111	EOL_2Q	
001000	EOL_HQ	
001001	Byte into R	
001010	EOL_1R	
001011	EOL_2R	
001100	EOL_HR	
001101	Operation in LU	
001110	Address formed in TAA	
001111	Byte into SACC	
010000	+ (or 0) Byte into SACC	These equal option 001111 if bytes are unsigned.
010001	− Byte into SACC	
010010	SACC $\geq$ THR (once for every upward passage)	
010011	SACC becomes negative	
010100	Match in W	
010101	Match in X	
010110	Match in Y	
010111	Match in Z	
011000	No-match in W	
011001	No-match in X	
011010	No-match in Y	
011011	No-match in Z	
011100	Matches in W and X simultaneously (only one count)	
011101	Matches in Y and Z simultaneously (only one count)	
011110	No-matches in W, X, Y, and Z simultaneously (only one count)	
011111	k = 0	
100000	k = 1	
100001	l = 0	
100010	l = 1	
100011	m = 0	
100100	m = 1	
100101	E = 0	
100110	E = 1	
100111	F = 0	
101000	F = 1	

101001	Bit 2 of G is 1 at end of group
101010	" 3 "
101011	" 4 "
101100	" 5 "
101101	" 6 "
101110	" 7 "
101111	E = 0 & F = 0 (only one count)
110000	E = 0 & F = 1 "
110001	E = 1 & F = 0 "
110010	E = 1 & F = 1 "
110011	E = 0 v F = 0 (only one count even if both occur)
110100	E = 0 v F = 1 "
110101	E = 1 v F = 0 "
110110	E = 1 v F = 1 "
110111	E = 0 v F = 0 (two counts if both occur)
111000	E = 0 v F = 1 "
111001	E = 1 v F = 0 "
111010	E = 1 v F = 1 "
111011	not specified
111100	"
111101	"
111110	"
111111	"

The coding for SCTR RST is:

000 = no reset
001 = match in W
010 = match in X
011 = match in Y
100 = SACC becomes negative
101 = SACC is reset
110 = end-of-group
111 = STOP

Care must also be used in specifying SCTR stepping and resetting as again asynchrony may be involved. If the same stimulus is specified for both counting and resetting, the reset will take place and not the count.

SCTR overflow is represented in the Indicator Register.

H2. 7. 3 Other quantities in the SACC and SCTR setup words

In the right half of the setup word containing THR are five additional fields. These fields are not set up, but merely represent the addresses with which some particular machine registers may be addressed. They are:

(E) - The value of E
(F) - The value of F
(G) - The value of G
(MOD) - The MOD register
(RS) - The "Return to Stream" bit, which is a 1 if an interruption occurred at the end of a streaming instruction and the next instruction was to have been a streaming mode.

H2.8 Automatic Adjustments

Automatic adjustments are designed to facilitate minor modifications in the presentation of data and in the operations performed on it. They occur only in connection with byte-by-byte streaming operations. See Figure H2.8 for the adjustment word format.

When a byte-by-byte instruction occupies the lefthand 64 bits of the instruction register, the next word from memory contains the associated adjustment data and is placed in the righthand 64 bits of the instruction register. One must be careful not to skip + 1 when leaving a stream byte-by-byte instruction for this would result in the adjustment word being interpreted as an arithmetic or stream instruction.

The adjustment word is divided into nine parts: the first seven parts contain seven adjustment codes; the eighth gives the priority among adjustments should several signals arise simultaneously; the ninth identifies the supplementary signal, if any. The following seven signals are the ones causing automatic adjustment:

1) W
2) X
3) [W-X]: Simultaneous signals from W and X (See H2.6)
4) A specified supplementary signal
5) Y
6) Z
7) [Y-Z]: Simultaneous signals from Y and Z (See H2.6)

(The setup information for each match unit designates whether an adjustment signal is to be emitted from that unit on a match or on a no-match.)

The supplementary signal (SUP) is chosen from the following list:

0	none	16	$P \geq Q$ (at end of group)
1	$E = 0$	17	$P = Q$ "
2	$E = 1$	18	$P \leq Q$ "
3	$F = 0$	19	$P < Q$ "
4	$F = 1$	20	$P \neq Q$ "
5	$E = 0$ and $F = 0$	21	Memory count overflow
6	$E = 0$ and $F = 1$	22	HL_P
7	$E = 1$ and $F = 0$	23	HL_Q
8	$E = 1$ and $F = 1$	24	HL_R
9	$P > Q$ (any byte time)	25	EC_P
10	$P \geq Q$ "	26	EC_Q
11	$P = Q$ "	27	EC_R
12	$P \leq Q$ "	28	STOP (includes 25, 26, and 27)
13	$P < Q$ "	29	End of Group
14	$P \neq Q$ "	30	SACC $\geq$ THR
15	$P > Q$ (at end of group)	31	not yet specified

The priority between W and X is fixed: W always precedes X. The priority between Y and Z is fixed: Y always precedes Z. The priority among (WX), (YZ), and S is specified by a two-bit field (PRI):

00: (WX) (YZ) (S)
01: (WX) (S) (YZ)
10 or 11: (S) (WX) (YZ)

There follows a list of automatic adjustments with comments upon the probable restrictions on their use. The numbering is purely for reference purposes; the actual coding has not yet been assigned. The restrictions indicated are those which insure meaningful application of the adjustments. Any restriction may be ignored at the programmer's risk. Restrictions on the combination adjustments (#128-255) are the union of all the restrictions applying to the individual adjustments making up the combination.

0 No op for single signal; sum of op's for MU combinations–No restrictions
1 No op
2 G and go to I. C. + 2 words
3 " + 3 "
4 " + 4 "
5 " + 5 "
6 " + 6 "
7 " + 7 "
8 " + 8 "
9 " - 1 "
10 " - 2 "
11 " - 3 "
12 " - 4 "
13 " - 5 "
14 " - 6 "
15 " - 7 "
16 " - 8 "
17 Store R and go to I. C. + 2 words
18 " + 3 "
19 " + 4 "
20 " + 5 "
21 " + 6 "
22 " + 7 "
23 " + 8 "
24 " - 1 "
25 " - 2 "
26 " - 3 "
27 " - 4 "
28 " - 5 "
29 " - 6 "
30 " - 7 "
31 " - 8 "

I. C. refers to the address of the byte-by-byte instruction, not the adjustment word. The stream must drain out as it does for STOP, and there consequently are no restrictions. There is no automatic method of return to the stream instruction – it must be programmed.

32	Repeat Byte from P	The indexing of P can be adjusted by: 1) MU's at MS_P if either (or both) Gates #1 or 4 are open 2) MU's at MS_Q if Gates #1 and 2 are both open or if Gates #4 and 5 are both open 3) MU's at MS_L if Gates #4 and 6 are both open 4) MU's at MS_U if Gates #1 and 3 are both open 5) End-of-level signals from P, Q, and U in configurations corresponding to MS_P, MS_Q, and MS_U above. 6) E, F, and G signals if Gate #1 is open.
33	Reset L_1P	
34	Skip Next Byte from P	
35	Runout L_1P	
36	Reset L_2P	
37	Advance L_2P	
38	Runout L_2P	
39	Reset L_HP	
40	Advance L_HP	
41	Runout L_HP	
41a	Runout Field in P	
41b	Runout Sequence in P	

42	Repeat Byte from Q	The indexing of Q can be adjusted by: 1) MU's at MS_P if Gates #1 and 2 are both open or if Gates #4 and 5 are both open 2) MU's at MS_Q if either (or both) Gates #2 and 5 are open 3) MU's at MS_L if both Gates #4 and 6 are open 4) End-of-level signals from P and Q in configurations corresponding to MS_P and MS_Q above 5) E, F, and G signals when Gate #2 is open.
43	Reset L_1Q	
44	Skip Next Byte from Q	
45	Runout L_1Q	
46	Reset L_2Q	
47	Advance L_2Q	
48	Runout L_2Q	
49	Reset L_HQ	
50	Advance L_HQ	
51	Runout L_HQ	
51a	Runout Field in Q	
51b	Runout Sequence in Q	

52	Repeat Byte in R (Read on top of last byte)	The indexing of C may be adjusted by: 1) MU's at MS_P if Gates #1 and 10 are both open 2) MU's at MS_Q if Gates #2 and 10 are both open 3) MU's at MS_L if Gate #10 is open 4) MU's at MS_U if Gates #3 and 10 are both open or if Gate #11 is open 5) End-of-level signals in configurations corresponding to MS_P, MS_Q, and MS_U above 6) E, F, and G signals when Gate #10 is open.
53	Reset L_1R	
54	Skip Next Byte in R (Leave space in R)	
55	Suppress Next Byte in R (Suppress before entering R)	
56	Reset L_2R	
57	Advance L_2R	
58	Reset L_HR	
59	Advance L_HR	

60	Reset T (Cancel Address)	This adjustment may be made by: 1) MU's at MS_P if Gate #4 is open 2) MU's at MS_Q if Gate #5 is open 3) MU's at MS_L if Gate #6 is open.
61	Repeat Byte from U	U indexing may be adjusted by: 1) MU's at MS_P when Gates #1 and 3 are both open 2) MU's at MS_U when at least one of Gates #3, 11, or 13 is open 3) End-of-level signals from P and U in configurations corresponding to MS_P and MS_U above 4) E, F, and G signals when Gate #3 is open.
62	Reset L_1U	
63	Skip next Byte from U	
64	Runout L_1U	
65	Go to next reference in U immediately	
66	Skip next reference in U after finishing this one	
67	Skip all references in U for the duration of group; do not finish this one.	
68	Suppress output of Gate #8	The LU may be adjusted for #68-73 1) MU's at MS_P if Gate #1 is open 2) By MU's at MS_Q if Gate #2 is open 3) By MU's at MS_Q if Gate #3 is open 4) By MU's at MS_L if at least one of Gates #6, 10 or 12 is open.
69	Insert P in place of L } P, Q are the logical operands.	
70	Insert Q in place of L }	
71	Insert MOD in place of L	
72	Reverse P, Q in L definitions for next byte	
73	Reverse P, Q in L definitions for remainder of group	
74	Insert MOD in place of U	Only if Gate #12 had been closed and #13 open. Controlled only by MS_U or end-of-level in U. Opens Gate #12 and closes Gate #13 for this one byte only.
75	Ignore W on this byte	1) Can be given by W, X, Y, or Z only for MU's with lower priority and when attached to same MS or if associated in a sense defined for simultaneous signals 2) Can be given by SUP (#0-20) with similar restrictions 3) A match unit should not disable itself by this adjustment.
76	Ignore X on this byte	
77	Ignore Y on this byte	
78	Ignore Z on this byte	

79	Disable W for next byte	1) Can be given by MU's connected to same MS or when associated in a sense defined for simultaneous signals 2) Can be given by SUP (#0-20) with similar restrictions.
80	Disable X for next byte	
81	Disable Y for next byte	
82	Disable Z for next byte	
83	Disable W for duration of group	
84	Disable X for duration of group	
85	Disable Y for duration of group	
86	Disable Z for duration of group	
87	Step SACC (by 2) (This count is in addition to any specified by SACC setup or coming in as a byte)	1) Anytime if SACC is being used merely as a counter 2) By MU's at MS_L if Gate #12 is open 3) By MU's at MS_U if Gate #13 is open 4) At end-of-group or STOP.
88	Reset SACC	
89	Step SCTR (by 1) (This count is in addition to any specified by the SCTR setup)	1) Anytime if SCTR is being used merely as a counter 2) If Gate #0 is open; a) By MU's at MS_P if Gates #1 or 4 are open b) By MU's at MS_Q if Gates #1 and 2 or Gates #4 and 5 are both open c) By MU's at MS_L if Gates #4 and 6 are both open d) By MU's at MS_U if Gates #1 and 3 are both open.
90	Reset SCTR	
91	Read last 7 bits plus sign of SACC into R	Only at STOP or End-of-Group
92	Read last 15 bits plus sign of SACC into R	
93	Read last 23 bits plus sign of SACC into R	
94	Read last 8 bits of SCTR into R	
95	Read all 16 bits of SCTR into R	

128	17 + 68
129	18 + 68
130	19 + 68
131	20 + 68
132	21 + 68
133	22 + 68
134	23 + 68
135	24 + 68
136	25 + 68
137	26 + 68
138	27 + 68
139	28 + 68
140	29 + 68
141	30 + 68
142	31 + 68
143	17 + 69
144	18 + 69
145	19 + 69
146	20 + 69
147	21 + 69
148	22 + 69
149	23 + 69
150	24 + 69
151	25 + 69
152	26 + 69
153	27 + 69
154	28 + 69
155	29 + 69
156	30 + 69
157	31 + 69
158	17 + 70
159	18 + 70
160	19 + 70
161	20 + 70
162	21 + 70
163	22 + 70
164	23 + 70
165	24 + 70
166	25 + 70
167	26 + 70
168	27 + 70
169	28 + 70
170	29 + 70
171	30 + 70
172	31 + 70
173	75 + 76
174	75 + 77
175	75 + 78
176	76 + 77
177	76 + 78
178	77 + 78
179	75 + 76 + 77
180	75 + 76 + 78
181	75 + 77 + 78
182	76 + 77 + 78
183	75 + 76 + 77 + 78
184	79 + 80
185	79 + 81
186	79 + 82
187	80 + 81
188	80 + 82
189	81 + 82
190	79 + 80 + 81
191	79 + 80 + 82
192	79 + 81 + 82
193	80 + 81 + 82
194	79 + 80 + 81 + 82
195	83 + 84
196	83 + 85
197	83 + 86
198	84 + 85
199	84 + 86
200	85 + 86
201	83 + 84 + 85
202	83 + 84 + 86
203	83 + 85 + 86
204	84 + 85 + 86
205	83 + 84 + 85 + 86

206 37 + 47
207 37 + 56
208 37 + 65
209 37 + 47 + 56
210 37 + 65 + 56
211 37 + 47 + 56 + 68
212 37 + 65 + 56 + 68
213 37 + 47 + 56 + 69
214 37 + 65 + 56 + 69
215 37 + 47 + 56 + 70
216 37 + 65 + 56 + 70
217 37 + 47 + 56 + 71
218 37 + 65 + 56 + 71

219 37 + 60
220 47 + 60
221 37 + 47 + 60

222 72 + 79
223 72 + 80
224 72 + 81
225 72 + 82

226 73 + 83
227 73 + 84
228 73 + 85
229 73 + 86

230 56 + 67

H2.9 Stream Indicator Bits

Although the streaming mode provides within itself for most eventualities a few conditions are nevertheless represented in the general Indicator Register. However these bits are tested only at the end of a streaming instruction and an instruction may take a long time to complete, so too much dependence cannot be placed on them for remedying stream conditions.

The seven bits referring directly to streaming are:

#43 = Memory Count overflow
#44 = SCTR overflow or underflow
#45 = SACC overflow
#46 = SACC $\geq$ THR
#47 = End of Chain in P
#48 = " Q
#49 = " R

In addition, Bit #18 is on whenever a streaming mode instruction is being executed.

These bits are tested only at the end of an instruction. Once they go on, they stay on until they are turned off by

a) causing an interruption, or
b) a conditional instruction, or
c) a VFL instruction.

All interruption instructions are automatically in arithmetic mode. Therefore there is a special trigger – the RS bit – which goes on whenever an interruption occurs at the end of a streaming operation and the next instruction was to have been in a streaming mode. It stays on until it is reset. It is addressed as Bit #50 of the setup word containing THR.

H2.10 The Hybrid Instructions

In addition to the byte-by-byte streaming operations Harvest provides a set of sixteen instructions with a format very similar to the format of the basic VFL instructions. We call these sixteen "hybrid instructions" since they are not byte-by-byte but yet utilize or apply to the Harvest concept of streaming information.

The first eight of them are primarily one-performance instructions, often occurring at the completion of a byte-by-byte operation. They are used either to specify some special operation not present in the basic machine or to generalize some basic instruction in a logically satisfying manner. STREAM CLEAR MEMORY is representative of the first aim; STREAM CONDITIONAL BRANCH, of the second.

The second set of eight are true "hybrid" instructions. They are closely analogous to some of the basic VFL instructions, but they utilize the tremendous power, flexibility, and speed that the multiple indexing and control mechanisms of Harvest provide. They operate on fields and govern a large number of executions with but one instruction and the accompanying stream setup.

The instruction formats are given in Figure H2.10.

H2.10.1 SUBR - STREAM UNCONDITIONAL BRANCH

This instruction is used to store the present contents of the instruction counter and to perform an unconditional branch. The branch address is either relative or absolute.

1. The Branch Address is modified by the designated index word.

2. If the indexed branch address is 0 no branch is made, and the next instruction is designated by the SKIP field.

3. If the indexed branch address is not 0, the 2-bit Branch Code designates the location of the next instruction:

   00 = Relative branch up: next instruction is taken from location [Address of present instruction + indexed Branch Address]
   01 = Relative branch down: next instruction is taken from location [Address of present instruction — indexed Branch Address]
   10, 11 = Absolute branch: next instruction is taken from location [Indexed Branch Address]

4. If a branch is taken, the present contents of the instruction counter are stored in the location specified in the instruction (address portion of designated 1/2-word). If the storage location is 0, no store takes place.

5. The BRAM bit applies if a branch is taken; the SKAM bit applies if a SKIP is made. Each means:

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

H2.10.2 SCBR - STREAM CONDITIONAL BRANCH

The instruction produces a branch or not depending on the parity of any selected bit in memory. The address of the test bit may be automatically augmented by 1 in preparation for the next test. The parity of the test bit may be altered or set as desired.

1. The bit address is modified by the designated index word to derive the location of the test bit. If the indexed bit address is in word 0, the test bit is defined to be 0.

2. If the test bit matches bit 24 of the instruction, a branch is taken. If it does not match, the instruction designated by the SKIP field is secured.

3. The 2-bit Branch Code applies when the branch is taken:

   00 = Relative branch up: next instruction is taken from location [Address of present instruction + Branch Address]
   01 = Relative branch down: next instruction is taken from location [Address of present instruction - Branch Address]
   10, 11 = Absolute branch: next instruction is taken from location [Branch Address]

4. a) If the Branch Address is 0 and the ADV bit is 0, then no branch is taken, and the normal SKIP applies.

   b) If the Branch Address is 0 and the ADV bit is 1, then a 6-bit counter (which is set to 0 at the beginning of the operation) is stepped by 1 before this instruction repeats. If this 6-bit counter steps to 64, the resultant overflow prevents further repetition of the instruction, and the next instruction is taken from SKIP + 1.

5. The BRAM bit applies when a branch is taken; the SKAM bit applies when the SKIP is taken. Each means:

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

6. Whether there is a branch or not, the RST bits modify the test bit after testing:

   00 = leave unchanged
   01 = invert
   10 = set to 0
   11 = set to 1

This field naturally has no effect if the indexed test bit address was in word 0.

7. Whether there is a branch or not, the ADV bit modifies the bit address given in the instruction, after testing:

   0 = leave unchanged
   1 = increase by 1 bit.

H2.10.3 Not yet specified.

H2.10.4 SMOV - STREAM MOVE

The instruction is used to move a field from any register or memory word to Stream Unit R during a halt in the streaming operations. It is particularly useful in adjoining identifications to processed fields.

1. Bits 25 and 26 designate the source of the field to be moved:

   00 = fixed address
   01 = P
   10 = Q
   11 = U

2. With option 00 above, the Address (bits 0-23) is modified by the specified index word to give the location of the first bit of the field to be moved. With the other three options, bits 0-24 of the instruction are modified by the index word. For P and Q this entire quantity is added to the present effective address to give the location of the first bit of the field to be moved. For U, only the last seven bits (6 + sign) are added to the present effective address to give the location of the first bit of the field to be moved.

3. The 13-bit field (12 + sign) labelled "Offset in R" is added to the present effective address in R to give the location of the first bit of the destination of the field.

4. Now the field of length 1-64 (FL = 0 if field length is 64 bits) is moved to R.

5. After the movement the effective addresses of P, Q, R, and U are set again to the addresses they possessed before the MOVE instruction was given.

6. The fields read from a fixed address or from U cannot cross a word boundary. If such a movement is attempted, the portion of the field read into R which would have come from the other side of the word boundary is automatically made all 0's.

7. At the conclusion of the operation, the next instruction is taken from the location specified by the SKIP field. The SKAM bit means as follows:

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

H2.10.5 SCSI - STREAM COUNT AND SET INDICATOR

This instruction is used to count the number of 0's or 1's in a designated field and to set an indicator bit depending on the relative sizes of the count and a comparison field in the instruction.

1. The Bit Address is modified by the designated index word. If the word part of the address is 0, no operation is performed, and the next instruction is immediately secured from the location designated by the SKIP field.

2. If the bit address does not refer to word 0, the field starting at that bit and of length ≤ 64, as specified by FL, is examined and either 0's or 1's counted according to whether the parity bit is 0 or 1.

3. The number of 0's or 1's counted is then compared with the comparison value given in the instruction. Indicator bit 61, 62, or 63 is set according to whether the count was less than, equal to, or greater than the comparison value. The actual count is available in the counter at the end of the operation. If the count is 64, the attempted carry is used to set bit #63. However the counter contents will indicate 000000.

4. If the count is ≤ 63, the location of the next instruction is taken from SKIP. If the count is 64, SKIP + 1 is used. In either case the SKAM bit means:

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

H2.10.6 SCAD - STREAM CONDITIONAL ADJUST

This instruction allows one to perform any two of the adjustment operations depending on the parity of any selected bit in memory. The address of the test bit may be automatically augmented by 1 in preparation for the next test. The parity of the test bit may be altered or set as desired.

1. The bit address is modified by the designated index word to derive the location of the test bit. If the indexed bit address is in word 0, the test bit is defined to be 0.

2. If the test bit matches bit 24 of the instruction, adjustments are made. The two 8-bit adjustment fields in the instruction refer to the same 8-bit adjustment codes used for automatic adjustments. Normally Adjustment 1 will be made first, then Adjustment 2. Then the instruction designated by the SKIP field is secured.

3. If the first adjustment is a NO-OP the second adjustment is disregarded. If the first adjustment is a BRANCH, the next instruction is taken from the designated location, and the second adjustment field and SKIP field of the present instruction are disregarded. If the first instruction is neither NO-OP nor BRANCH it is performed; if then the second instruction is a BRANCH, the next instruction is taken from the designated location, and the SKIP field of the present instruction is disregarded.

4. The BRAM bit applies when an adjustment BRANCH is taken; the SKAM bit applies when the SKIP is taken. Each means:

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

   Any branch here to arithmetic mode must refer to an instruction starting at the beginning of a word.

5. Whether there are adjustments made or not, the RST bits modify the test bit, after testing:

   00 = leave unchanged
   01 = invert
   10 = set to 0
   11 = set to 1

   This field does not apply if the indexed test bit address was in word 0.

6. Whether there are adjustments made or not, the ADV bit modifies the bit address given in the instruction, after testing:

   0 = leave unchanged
   1 = increase by 1 bit.

H2.10.7 SCLM - STREAM CLEAR MEMORY

This instruction enables one to clear blocks of words in memory. Two sizes of blocks are available both for the 1/2-microsecond memory and the 2-microsecond memory.

CLEAR MEMORY

1. The word address is modified by the specified index word. The bit portion of the resultant address is disregarded.

2. A block of memory is now cleared, the block sizes for the two speeds of memory are given by the LS bit:

		1/2 μsec	2 μsec
0:	small	8 words	1024 words
1:	large	128 words	16384 words

   The block which is cleared consists of the set of consecutive words which starts at a multiple of the block size and which contains the indexed word address.

3. After clearing, the next instruction is taken from the location designated by the SKIP field. The SKAM bit means as follows:

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

H2.10.8 SSTC - STREAM STORE AND CLEAR

This instruction transmits a half-word from one specified location to another. The source may optionally be cleared.

1. The store address is modified by the designated index word. The last 5 bits of the bit portion of the address do not apply.

2. The half-word at the source address is stored in the half-word at the indexed store address. If the source half-word was part of a setup word and some of the bits were unassigned, the unassigned bits are mapped as zeros.

3. After the transfer the source half-word may be cleared:

   CL: 0 = do not clear
   1 = clear

4. After the transfer the index value may be incremented by one half-word.

   ADV: 0 = do not advance
   1 = advance

5. After the operation the next instruction is secured from the location designated by the SKIP field. SKAM means

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

H2.10.9 SMER - STREAM MERGE

For relatively short records, sorting by moving the records is an efficient and orderly process. Since the Stream Units provide an automatic flow of data from memory, a large part of the bookkeeping associated with binary merge sorting can be relegated to the SU control mechanism.

It is assumed that records to be merged are in two blocks. It is desired to merge the two blocks of records into one block. The ordering of the records depends upon the relative values of their control fields.

The MERGE instructions used in Harvest permit handling a general record format. A control field or subfield may be no longer than 64 bits, and may be offset not more than 4095 bits from the beginning of the record. If, however, there is only one control field and it occurs at the beginning of the record, the entire record may be treated as if it were the control field and its length is virtually unlimited.

The SIMPLE MERGE UP and SIMPLE MERGE DOWN instructions may be used for merging records for which it is assumed that the control field is the entire record. Four more MERGE instructions are provided for more complicated operations. (See Figure H2.10.9a for format.) The MERGE OFFSET UP and MERGE OFFSET DOWN instructions are used for merging records having a single control field offset from the beginning of the records. The MERGE SPLIT UP and MERGE SPLIT DOWN instructions are used in conjunction with MERGE OFFSET UP and MERGE OFFSET DOWN for merging records with split control fields. The UP and DOWN instructions produce ascending and descending sequences, respectively.

Referring to the instruction format shown in Figure H2.10.9a, the TYPE field is coded as follows:

TYPE	Instruction
00 or 01	SIMPLE MERGE (Type I)
10	MERGE OFFSET (Type II)
11	MERGE SPLIT (Type III)

Figure H2.10.9b shows the general plan of the Merge loop. The various types of Merge instructions differ in the manner in which they determine what record is to be moved. After this has been determined, the same general control sequence is executed by all six merge instructions. The control sequence is diagrammed in Figure H2.10.9c.

H2.10.9.1 Index Control Parameters

The first level index control parameters for SU-P and SU-Q are used to define the record: I_1 is equal to the byte size and N_1 is equal to the number of bytes in a record. The second level parameters define the input block: I_2 is equal to the record length in bits ($I_2 = N_1 I_1$) and N_2 is equal to the number of records in an input block. During internal sorting, initially $N_2 = 1$. It doubles after each internal merge pass and has a final value equal to half the total number of input records that are to be accomodated in memory. The third level is used to locate the next input block: I_3 is equal to the input block length (i.e., $I_3 = N_2 I_2$), and as such doubles after each internal merge pass. I_3 has the initial value I_2 and the final value equal to

(half the total number of input records that are to be accomodated in memory) x (the length of a record in bits).

N_3 may be set equal to zero during internal merging.

Only the first level of indexing need be used in SU-R for an internal or external merge pass. Its parameters define the total output block: I_1 is equal to the byte size and N_1 is equal to the total number of bytes in the entire output block.

During an external merge pass N_{2P} and N_{2Q} remain constant at half the total number of input records that are to be accomodated in memory. I_{3P} and I_{3Q} may now be set at zero, while N_{3P} and N_{3Q} keep track of the total input block size. They have an initial value of 1 and double after every complete external pass.

For internal passes the Merge instructions may be set to branch after the output block is filled by setting the Branch Point Control Bit, BP, to the value 1. Filling the output block terminates an internal pass, and necessary bookkeeping may be taken care of by ordinary programming.

For external passes the BP bit is set to the value zero and the Merge instructions branch after an input sub-block (second level in P or Q) is emptied. The external pass terminates when two input blocks (third level in P and Q) have been merged.

For internal and external passes, the Merge instructions skip whenever a record has been moved and neither an input nor output block has been emptied or filled.

If BP = 1 then whenever moving a record from P (Q) results in an end of second level in P (Q) a second level runout is executed in Q (P), just as is wanted in internal merging.

H2. 10. 9. 2 Type I - SIMPLE MERGE UP (DOWN)

These instructions assume that the control field is the entire record. FL and OFF are not used. Three situations can arise:

i) the records are identical;
ii) control fields are identical, but data fields are not;
iii) control fields differ.

In the first case, it is immaterial which record is moved to SU-R. In the second case, it is assumed that it is also immaterial since the sort is used to arrange control fields. If this assumption is unsatisfactory, a MERGE OFFSET must be used. In the third case, the decision as to which record to move is made on the basis of the relative sizes of the control fields which begin the record; and only that portion of the records common to both control fields has already been moved at the time the decision is made.

1. Corresponding bytes from two records are compared, and

   a. if the compared bytes are equal, the common byte goes to R;

   b. if the byte from P is less (greater) than the byte from Q, the remainder of the record from P is moved to R without further comparison and index level 1 of Q is reset;

   c. if the byte from P is greater (less) than the byte from Q, the remainder of the record from Q is moved to R without further comparison and index level 1 of P is reset;

   d. if the records are identical, the bytes are considered as having passed from P to R, and the first index level of Q is reset.

2. For both the UP and DOWN instructions, if the record is moved from P (Q), and

   a. there results an end of second level in P (Q), and
      (1) BP = 0, then branch;
      (2) BP = 1, then runout the second index level in Q (P); now if

         (a) there results an end of first level in R, then branch;
         (b) there does not result an end of first level in R, then skip;

   b. there does not result an end of second level in P (Q), then skip.

3. If a skip is taken, the next instruction is obtained from location IC + SKIP. If SKIP = 0, then the current instruction is repeated (without a new memory reference).

4. For SIMPLE MERGE, SKIP = 0.

5. If a branch is taken, then if

   a. BRA = 0, a skip occurs instead of a branch;

   b. BRA ≠ 0, the usual BRANCH CODES apply (see, for example, Section H2.10.1, Paragraph 3).

6. The BRA is indexed by the contents of the index register specified by IX.

7. BRAM applies if a branch is taken:
   BRAM = 0 next instruction in streaming mode;
   BRAM = 1 next instruction in arithmetic mode.
   SKAM does not apply, as a skip in merging is always to a stream instruction.

H2.10.9.3 Type II - MERGE OFFSET UP (DOWN)

This instruction is used to merge records that have a single control field offset from the beginning of the record. It is also used in conjunction with MERGE SPLIT (Section H2.10.9.4) instructions to merge records whose control field is split into several subfields. The following control sequence is for non-split control fields.

1. The two control fields of length specified by FL (FL = 0 for fields of length 64 bits) offset a number of bits specified by OFF from the beginning of the records are compared byte-by-byte, from high order (lefthand) bytes to low order (righthand) bytes; and if

   a. the control field from P is less (greater) than or equal to the control field from Q, the record from P is moved to R, and index level 1 of Q is reset;

   b. the control field from P is greater (less) than the control field from Q, the record from Q is moved to R, and index level 1 of P is reset.

2 and 3. Same as Section H2.10.9.2, Paragraphs 2 and 3.

4. For MERGE OFFSET not used in conjunction with MERGE SPLIT, SKIP = 0.

5, 6, and 7. Same as Section H2.10.9.2, Paragraphs 5, 6, and 7.

For split control fields, assume there are n subfields ($n \leq 8$). One MERGE SPLIT instruction is used for each of the first n -1 subfields, and a MERGE OFFSET instruction is used for the n -th subfield. In this case, in Paragraph 1 above the OFF specifies the offset (positive or negative) in bits from the end of the (n -1)-th subfield; and in Paragraph 4 above, SKIP = 1 -n.

H2.10.9.4 Type III - MERGE SPLIT UP (DOWN)

This instruction is used in conjunction with the MERGE OFFSET (Section H2.10.9.3) instruction to merge records whose control field is split into several subfields. Assume there are n such subfields where $n \leq 8$; then n -1 MERGE SPLIT instructions in sequence followed by one MERGE OFFSET instruction form an instruction loop that will determine which of a pair of records is to be moved. (The use of the MERGE OFFSET in this loop is described in Section H2.10.9.3.)

1. The k -th pair (k = 1, ..., n -1) of control subfields of lengths specified by FL (FL = 0 for fields of length 64 bits) offset a number of bits (positive or negative) specified by OFF from the ends of the (k -1)th pair of subfields if $k > 1$ or the beginnings of the records if k = 1 are compared (by the k -th MERGE SPLIT instruction) byte-by-byte, from high order (lefthand) bytes to low order (righthand) bytes; and if

   a. the control subfield from P is equal to the control subfield from Q, the instruction at IC + 1 is executed [the (k + 1)-th MERGE SPLIT if $k < n - 1$, otherwise the MERGE OFFSET];

   b. the control subfield from P is less (greater) than the control subfield from Q, the record from P is moved to R, and index level 1 of Q is reset;

   c. the control subfield from P is greater (less) than the control subfield from Q, the record from Q is moved to R, and index level 1 of P is reset.

2 and 3. Same as Section H2.10.9.2, Paragraphs 2 and 3.

4. For the k -th MERGE SPLIT instruction (k = 1, ..., n -1), SKIP = 1 -k.

5, 6, and 7. Same as Section H2.10.9.2, Paragraphs 5, 6, and 7.

H2.10.9.5 Merging Files of Unequal Length

The Merge instructions can be used readily to merge two ordered files of unequal length. The first level index control parameters for SU-P and SU-Q are used (as in Section H2.10.9.1) to define the record, and I_2 is equal to the record length in bits ($I_2 = N_1 I_1$). N_{2P} is equal to the number of records in the P-file, while N_{2Q} is equal to the number in Q-file. Since this merge is a one pass operation, the N_2's remain fixed.

Once again only the first level of indexing is used in SU-R and it defines the total output block. I_1 is equal to the byte size and N_1 is equal to the total number of bytes in the entire output block $\left[N_{1R} = N_{1P} (N_{2P} + N_{2Q})\right]$.

Finally, the Branch Point Control Bit, BP, should be set to 1 so that whenever one complete file has been moved to R, the other will be runout to R also.

H2.10.10 SSER - STREAM SEARCH

It is often necessary to search through a file of records to find all records whose control fields are, say, equal to that of another record. Alternatively, it might be necessary to search for all records whose control fields are greater than each of the control fields of records in a second file. Harvest contains a family of SEARCH instructions which permit searching a file under very general conditions.

As in the MERGE instructions, a general record format may be handled. A control field or subfield may be no longer than 64 bits, and may be offset not more than 4095 bits from the beginning of the record. Only SEARCH OFFSET and SEARCH SPLIT instructions are defined, and UP and DOWN ordering of records does not apply. The SEARCH OFFSET is used to search records having a single control field offset from the beginning of the record, while SEARCH SPLIT instructions are used in conjunction with SEARCH OFFSET to handle records with split control fields. (See Figure H2.10.10a for format.)

The SEARCH instructions assume that the file being search is indexed by SU-P and that the master file (one or more records) is indexed by SU-Q. If the "search condition" is met, the record is moved from P to R. There are six search conditions that may be chosen from; they are designated by the three bit SCD field (Figure H2.10.10a) according to the following table:

SCD	Move record if
000	$P \leq Q$
001	$P \geq Q$
010	$P < Q$
011	$P > Q$
100 or 101	$P = Q$
110 or 111	$P \neq Q$

If the file to be searched is in sort with respect to the control fields, the search operation may be speeded up (provided the search condition is not $P = Q$ or $P \neq Q$) by using an ORDERED SEARCH. This is accomplished by immediately moving the remainder of the file from P without further searching, once the search condition is met. The control sequence shown in Figure H2.10.10c shows three entry points A, B, and C. If the programmer chooses ORDERED SEARCH (operation Types II or III), the control sequence is entered at B or C; if he chooses RANDOM SEARCH (operation Types 0 or I), the sequence is entered at A or C (see also Figure H2.10.10b). The programmer must use RANDOM SEARCH if the search condition is $P = Q$ or $P \neq Q$. Moreover, the ordering of the control fields must be ascending if the search condition is $P \geq Q$ or $P > Q$, but descending if the conditon is $P \leq Q$ or $P < Q$.

If the master file contains more than one record, it may be desirable to perform a round robin search (each control field from the P-file is compared with each control field from the Q-file). The Round Robin Control Bit, RR, determines whether or not a new record will be fetched by SU-Q after each complete examination of the P-file (see Figure H2. 10. 10c). It is assumed that the round robin search will be performed only with RANDOM SEARCH and search conditions $P = Q$ or $P \neq Q$.

The TYPE field (Figure H2. 10. 10a) is as follows:

TYPE	Instruction
00	RANDOM SEARCH OFFSET (Type 0)
01	RANDOM SEARCH SPLIT (Type I)
10	ORDERED SEARCH OFFSET (Type II)
11	ORDERED SEARCH SPLIT (Type III)

H2. 10. 10. 1 Index Control Parameters

The first level index control parameters for SU-P and SU-Q are used to define the record: I_1 is equal to the byte size and N_1 is equal to the number of bytes in a record. The second level parameters define the input file: I_2 is equal to the record length in bits ($I_2 = N_1 I_1$), N_{2P} is equal to the number of records in the P-file, and N_{2Q} the number of records in the Q-file.

Only the first level of indexing need be used in SU-R. I_1 is equal to the byte size. If it is desired to limit the size of the output file, then N_1 should be set equal to the total number of bytes in the entire output file. If no limitation is desired, then N_1 may be set to zero.

H2. 10. 10. 2 Type 0 - RANDOM SEARCH OFFSET

This instruction is used to search a file of randomly ordered records that have single control fields offset from the beginnings of the records. It is also used in conjunction with RANDOM SEARCH SPLIT (Section H2. 10. 10. 3) instructions to search records whose control fields are split into several subfields. The following control sequence is for non-split control fields.

1. The two control fields of length specified by FL (FL = 0 for fields of length 64 bits) offset a number of bits specified by OFF from the beginnings of the records are compared byte-by-byte, from high order (lefthand) bytes to low order (righthand) bytes; and if

a. the search condition as specified by SCD (see Section H2.10.10) is satisfied, the record from P is moved to R, and index level 1 of Q is reset;

b. the search condition is not satisfied, the record in P is bypassed (index level 2 of P is advanced), and index level 1 of Q is reset.

2. Regardless of whether the record in P is moved or bypassed, if

a. there results an end of second level in P or an end of first level in Q, and

(1) RR = 0, then branch;
(2) RR = 1, then advance the second index level in Q; now if

(a) there results an end of second level in Q, then branch;
(b) there does not result an end of second level in Q, then skip;

b. there does not result an end of second level in P or an end of first level in Q, then skip.

3. If a skip is taken, the next instruction is obtained from location IC + SKIP. If SKIP = 0, then the current instruction is repeated (without a new memory reference).

4. For RANDOM SEARCH OFFSET, SKIP = 0.

5. If a branch is taken, then if

a. BRA = 0, a skip occurs instead of a branch;

b. BRA $\neq$ 0, the usual BRANCH CODES apply (see, for example, Section H2.10.1, Paragraph 3).

6. The BRA is indexed by the contents of the index register specified by IX.

7. BRAM applies if a branch is taken:

BRAM = 0, next instruction in streaming mode;

BRAM = 1, next instruction in arithmetic mode.

SKAM does not apply, as a skip in searching is always to a stream instruction.

For split control fields, assume there are n subfields ($n \leq 8$). One RANDOM SEARCH SPLIT instruction is used for each of the first n-1 subfields, and a RANDOM SEARCH OFFSET instruction is used for the n-th subfield. In this case, in Paragraph 1 above, the OFF specifies the offset (positive or negative) in bits from the end of the (n-1)-th subfield; and in Paragraph 4 above, SKIP = 1-n.

H2.10.10.3 Type I - RANDOM SEARCH SPLIT

This instruction is used in conjunction with the RANDOM SEARCH OFFSET instruction (Section H2.10.10.2) to search a file of randomly ordered records whose control fields are split into several subfields. Assume there are n such subfields where $n \leq 8$; then n-1 RANDOM SEARCH SPLIT instructions in sequence followed by one RANDOM SEARCH OFFSET instruction form an instruction loop that will determine whether a record is to be moved or bypassed. (The use of the RANDOM SEARCH OFFSET in this loop is described in Section H2.10.10.2.)

1. The k-th pair ($k = 1, \ldots, n-1$) of control subfields of lengths specified by FL (FL = 0 for fields of length 64 bits) offset a number of bits (positive or negative) specified by OFF from the ends of the (k-1)-th pair of subfields if $k > 1$ or the beginnings of the records if $k = 1$ are compared (by the k-th RANDOM SEARCH SPLIT instruction) byte-by-byte, from high order (lefthand) bytes to low order (righthand) bytes; and if

   a. it is not determined whether or not the search condition as specified by SCD (see Section H2.10.10) is satisfied (that is, if the control subfield from P is equal to the control subfield from Q), the instruction at IC + 1 is executed [the (k + 1)-th RANDOM SEARCH SPLIT if $k < n-1$, otherwise the RANDOM SEARCH OFFSET];

   b. it is determined that the search condition is satisfied, the record from P is moved to R, and index level 1 of Q is reset;

   c. it is determined that the search condition is not satisfied, the record in P is bypassed (index level 2 of P is advanced), and index level of 1 of Q is reset.

2 and 3. Same as Section H2.10.10.2, Paragraphs 2 and 3.

4. For the k-th RANDOM SEARCH SPLIT instruction ($k = 1, \ldots, n-1$), SKIP = 1-k.

5, 6, and 7. Same as Section H2.10.10.2, Paragraphs 5, 6, and 7.

H2. 10. 10. 4 Type II - ORDERED SEARCH OFFSET and Type III - ORDERED SEARCH SPLIT

The use of these instructions is almost identical to the use of the Type 0 - RANDOM SEARCH OFFSET and Type I - RANDOM SEARCH SPLIT. They are used when the file to be searched (P-file) has its records ordered by control fields.

The control sequence in Section H2. 10. 10. 2 applies to ORDERED SEARCH OFFSET if Paragraph 1. a. is changed to

> "the search condition as specified by SCD (see Section H2. 10. 10) is satisfied, the record from P and all remaining records from the P-file are moved to R, and index level 1 of Q is reset",

and Paragraph 4 is changed to

> "For ORDERED SEARCH OFFSET, SKIP = 0".

Similarly, the control sequence in Section H2. 10. 10. 3 applies to ORDERED SEARCH SPLIT if throughout Paragraph 1 ORDERED replaces RANDOM, if Paragraph 1. b. is changed to

> "it is determined that the search condition is satisfied, the record from P and all remaining records from the P-file are moved to R, and index level 1 of Q is reset",

and Paragraph 4 is changed to

> "For the k -th ORDERED SEARCH SPLIT instruction (k = 1, . . . , n -1), SKIP = 1 -k".

H2.10.11 SADD - STREAM ADD

This instruction produces in R the sum of an operand from P and an operand from Q. The addition may be accumulative. The number of operands involved in the intermediate sums as well as the general progression of operands through memory are defined by the stream indexing. The field lengths and signs of the operands are defined in the instruction.

1. P and Q supply the two operands, and R forms and stores the results. All numbers are integral, in binary notation, and, if signed, have a 1-bit sign at the right hand end.

2. SGN_P, SGN_Q, and SGN_R specify the signs of the operands and results. Each means:

   00 = unsigned number
   01 = unsigned number
   10 = signed number, taken with this sign
   11 = signed number, but taken with opposite sign

3. FL_P, FL_Q, and FL_R give the field lengths. FL_P and FL_Q may be from 1 to 64 (000000 ≡ 64), and FL_R may be from 1 to 128 (0000000 ≡ 128). The FL includes the sign, if present.

4. If a result exceeds FL_R, the right hand end of the field remains anchored and the overflow occurs to the left. These overflow bits are lost. Indicator Bit #30 is turned on. Moreover, the result may not occupy space in more than two adjacent words. If it extends to the left of two consecutive words, those bits are permanently lost. (This applies to intermediate results, also.) Again #30 is turned on.

5. The indexing mechanisms of P and Q define the start points of fields only, and are entirely unrestricted. The actual movement of the fields is done by the basic arithmetic field length mechansims.

6. The R indexing mechanism both positions the results in memory and determines the number of results accumulated in one place before storing. It again defines the start points of fields.

   a. If nested indexing is used, accumulation is for the duration of the first level. If $N_1 = 1$, for example, there is no accumulation of results. Here I_1 must equal 0, and is made 0 (for this instruction only) regardless of the set up specifications.

b. If sequential indexing is used, the end of each sequential level indicates the end of accumulation. All the I's on the sequential levels must equal 0, and are made 0 (for this instruction only) regardless of the set up specifications.

7. The whole operation stops at the end of highest level in R. At the conclusion of the operation, the next instruction is secured from the location specified by SKIP. The SKAM bit acts as follows:

0 = next instruction is in streaming mode
1 = next instruction is in arithmetic mode.

H2.10.12 SMPY - STREAM MULTIPLY

This instruction produces in R the product of an operand from P and and operand from Q. The multiplication may be accumulative. The number of operands involved in the intermediate products as well as the general progression of operands through memory are defined by the stream indexing. The field lengths and signs of the operands are defined in the instruction.

1. P and Q supply the two operands, and R forms and stores the results. All numbers are intergral, in binary notation, and, if signed, have a 1-bit sign at the right hand end.

2. SGN_P, SGN_Q, and SGN_R specify the signs of the operands and results. Each means:

   00 = unsigned number
   01 = unsigned number
   10 = signed number, taken with this sign
   11 = signed number, but taken with opposite sign

3. FL_P, FL_Q, and FL_R give the field lengths. FL_P and FL_Q may be from 1 to 64 (000000 ≡ 64), and FL_R may be from 1 to 128 (0000000 ≡ 128). The FL includes the sign, if present.

4. If a result exceeds FL_R, the right hand end of the field remains anchored and the overflow occurs to the left. These overflow bits are lost. Indicator Bit #30 is turned on. Moreover, the result may not occupy space in more than two adjacent words. If it extends to the left of two consecutive words, those bits are permanently lost. (This applies to intermediate results, also.) Again #30 is turned on.

5. The indexing mechanisms of P and Q define the start points of fields only, and are entirely unrestricted. The actual movement of the fields is done by the basic arithmetic field length mechanisms.

6. The R indexing mechanism both positions the results in memory and determines the number of results accumulated in one place before storing. It again defines the start points of fields.

   a. If nested indexing is used, accumulation is for the duration of the first level. If $N_1 = 1$, for example, there is no accumulation of results. Here I_1 must equal 0, and is made 0 (for this instruction only) regardless of the set up specifications.

b. If sequential indexing is used, the end of each sequential level indicates the end of accumulation. All the I's on the sequential levels must equal 0, and are made 0 (for this instruction only) regardless of the set up specifications.

7. The whole operation stops at the end of highest level in R. At the conclusion of the operation, the next instruction is secured from the location specified by SKIP. The SKAM bit acts as follows:

0 = next instruction is in streaming mode
1 = next instruction is in arithmetic mode.

H2.10.13 SMLU - STREAM MULTIPLE LOOKUP

This instruction takes a given entry through a succession of tables. There are two types differing in the generality of the placement of results and the progression through and location of tables.

1. If the TYPE bit is 0:

   a. P's indexing mechanism determines a sequence of 19-bit (1/2-word) addresses, each of which is the start point address of a 24 bit base address.

      Q's indexing mechanism determines a series of parameters.

      R's indexing mechanism determines a series of initial arguments.

   b. The general operation will be:

      The first argument and the first parameter will look up an entry in the first table.

      The result from the first table and the second parameter will look up an entry in the second table.

      Finally, the result from the last table will replace the original argument in the memory word.

   c. P determines the sequence of base addresses for the successive tables. If P uses sequential indexing, the end of each sequential level defines the end of the table lookup process on one argument. If P uses nested indexing the end of the 1st level defines the end of the table lookup process on one argument. The 24-bit base addresses go in parallel to the TAA mechanism and are added to the base address specified in the TAA setup.

      The entries from Q are added to the combined base address just mentioned. They are positioned by the second level of TAA indexing. The number of bytes coming from Q is determined by the 1st level of indexing of Q. All levels must be nested.

      R supplies the initial argument. It, and intermediate arguments, are added to the partially formed address in TAA. They are positioned by the first level of TAA indexing. The number of bytes coming from R is determined by the 1st level of indexing of R. All levels of indexing must be nested. The initial arguments must be of length ≤ 24 and cannot cross word boundaries. The intermediate arguments must

be of the same length as the initial ones and also cannot cross word boundaries. The first levels of R and U must be set up identically.

d. BM_Q, BM_R, and BM_U must be specified.

e. The whole operation stops at end of highest level in R.

2. If the TYPE bit is 1:

a. P's indexing mechanism determines a series of initial arguments.

Q's indexing mechanism determines a series of parameters.

R's indexing mechanism determines the storage location of the results.

The intermediate values themselves contain information about the base address of the next table.

b. The general operation will be:

The first argument and the first parameter look up an entry in the first table.

The result from the first table and the second parameter look up an entry in the second table.

Finally, the result from the last table is stored in a designated location.

c. The first level of indexing of P defines the initial argument. All levels must be nested. The bytes from P act as the first source for T for the first look up. Intermediate results come from U and must be of the same size as the initial arguments. These intermediate results act as the first source for second and later look ups.

The parameters from Q act as the second source for T. All levels must be nested. The end of the successive lookups on one argument is determined by the end-of-second-level signal in Q.

R stores the final results.

The intermediate results cannot cross word boundaries in U.

Either the intermediate results or the parameters must contain the locations of the base addresses of successive tables relative to the original base address. A few bits often suffice.

d. BM_P, BM_Q, BM_R, and BM_U must be specified.

e. The operation stops at end of highest level in P.

3. When the operation stops the next instruction is secured from the location specified by the SKIP field. The SKAM bit says:

0 = next instruction is in streaming mode
1 = next instruction is in arithmetic mode.

H2.10.14 SILS - STREAM INDIRECT LOAD AND STORE

This instruction transmits fields from Q to R. Either the address of Q or the address of R is an indirect address obtained by table lookup with data from P as the argument.

1. Bit LS specifies LOAD or STORE:

   0 = LOAD (indirect address for Q)
   1 = STORE (indirect address for R)

2. The data specified by the first level of P reads into T as the first source. There is no second source. P indexing must be nested.

3. The TAA behavior is as follows:

   TAM: 000 = The assembled address is sent to S_Q or S_R for LS = 0 or 1, respectively.
   001 = The reference is sent to S_Q or S_R for LS = 0 or 1, respectively.
   010, 011 = The address portion of the half-word referenced by the first 19 bits of the assembled table address is sent to S_Q or S_R for LS = 0 or 1, respectively. Then the bit in memory addressed by the whole 24-bit assembled table address has a 1 or-ed to it.
   100, 101, 110, 111 = (1/2-microsecond memory only) The address portion of the half-word referenced by the first 19 bits of the assembled table address is sent to S_Q or S_R for LS = 0 or 1, respectively. Then this same address portion of the memory word has a 1 added to it in a position determined by the KK field. (The K field of the T setup is not effective.)

   KK: 000 = 1 added in position 20 of address (count by 8 bits)
   001 = " 19 " (count by 16 bits)
   010 = " 18 " (count by 32 bits)
   011 = " 17 " (count by 1 word)
   100 = " 16 " (count by 2 words)
   101 = " 15 " (count by 4 words)
   110 = " 14 " (count by 8 words)
   111 = " 13 " (count by 16 words)

4. After S_Q or S_R is supplied with an indirect address (LS = 0 or 1, respectively) the data specified by the first level of Q is transmitted serially to R.

5. a. For LS = 0 (load) all stream indexing for P, Q, R, and T must be specified except S_Q, second and higher levels of Q, and I_2T.

   b. For LS = 1 (store) all stream indexing for P, Q, R, and T must be specified except S_R, second and higher levels of R, and I_2T.

6. The operation stops at the end of highest level in P. The next instruction is located by the SKIP field. The SKAM bit means:

   0 = next instruction is in streaming mode
   1 = next instruction is in arithmetic mode.

H2.10.15 SSEL - STREAM SELECT

The SELECT instructions provide an efficient method for selecting from a file of records that record with the least or greatest control field.

As in the SEARCH instructions, a general record format may be handled. The maximum permissible control field or subfield is 64 bits, and the maximum permissible offset is 4095 bits. SELECT OFFSET and SELECT SPLIT are defined, and UP and DOWN ordering does not apply. The SELECT OFFSET is used with records having single control fields, while SELECT SPLIT is used in conjunction with SELECT OFFSET for records with split control fields.

Figure H2.10.15a gives the format for the SELECT instruction. The Least-Greater Control Bit, LG is coded as follows:

LG	Action
0	Select least
1	Select greatest

The TYPE field is coded as follows:

TYPE	Instruction
0	SELECT OFFSET (Type 0)
1	SELECT SPLIT (Type I)

Figure H2.10.15b shows the SELECT control sequence.

H2.10.15.1 Index Control Parameters

Only one record file is involved. The address of the first record is setup initially in S_Q the address of the second record is setup in S_P.

The first level index control parameters for all three stream units define the record: I_1 is equal to the byte size and N_1 is equal to the number of bytes in a record.

The second level parameters are needed in P only, and define the input file: I_2 is equal to the record length in bits ($I_2 = N_1 I_1$) and N_{2P} is equal to the total-number-of-records-in-the-file minus one.

H2.10.15.2 Type 0 - SELECT OFFSET LEAST (GREATEST)

The instruction is used with records having a single control field. It is also used in conjunction with SELECT SPLIT (Section H2.10.15.3) instructions to select from records whose control fields are split into several subfields. The following control sequence (see Figure H2.10.15b) is for non-split control fields.

1. The two control fields of length specified by FL (FL = 0 for fields of length 64 bits) offset a number of bits specified by OFF from the beginnings of the records are compared byte-by-byte, from high order (lefthand) bytes to low order (righthand) bytes; and if

   a. The control field from Q is less (greater) than or equal to the control field from P, the first index level of P is advanced and S_Q is reset;

   b. the control field from Q is greater (less) than the control field from P, then S_P is reset and replaces S_Q, and the first index level of P is advanced.

2. Regardless of whether S_Q is reset or replaced, if

   a. there results an end of second level in P, the first index level of Q is run out (i.e., the record addressed by S_Q is moved to R) and a branch is taken;

   b. There does not result an end of second level in P, then SKIP.

3. If a skip is taken, the next instruction is obtained from location IC + SKIP. If SKIP = 0, then the current instruction is repeated (without a new memory reference).

4. For SELECT OFFSET, SKIP = 0

5. If a branch is taken, then if

   a. BRA = 0, a skip occurs instead of a branch;

   b. BRA $\neq$ 0, the usual BRANCH CODES apply (see, for example, Section H2.10.1, Paragraph 3).

6. The BRA is indexed by the contents of the index register specified by IX.

7. BRAM applies if a branch is taken:

   BRAM = 0, next instruction in streaming mode;

   BRAM = 1, next instruction in arithmetic mode.

   SKAM does not apply, as a skip in selecting is always to a stream instruction.

For split control fields, assume there are n subfields ($n \leq 8$). One SELECT SPLIT instruction is used for each of the first n-1 subfields, and a SELECT OFFSET instruction is used for the n-th subfield. In this case, in Paragraph 1 above, the OFF specifies the offset (positive or negative) in bits from the end of the (n-1)-th subfield; and in Paragraph 4 above, SKIP = 1-n.

H2.10.15.3 Type I - SELECT SPLIT LEAST (GREATEST)

The instruction is used in conjunction with the SEARCH OFFSET instruction (Section H2.10.15.2) for records whose control fields are split into several subfields. Assume there are n such subfields where $n \leq 8$; then n-1 SELECT SPLIT instructions in sequence followed by one SELECT OFFSET instruction form an instruction loop that will determine whether the Q record address is to be reset or replaced. (The use of the SELECT OFFSET in this loop is described in Section H2.10.15.2.)

1. The k-th pair ($k = 1, \ldots, n-1$) of control subfields of lengths specified by FL (FL = 0 for fields of length 64 bits) offset a number of bits (positive or negative) specified by OFF from the ends of the (k-1)-th pair of subfields if $k > 1$ or the beginnings of the records if $k = 1$ are compared (by the k-th SELECT SPLIT instruction) byte-by-byte, from the high order (lefthand) bytes to the low order (righthand) bytes; and if

   a. the control subfield from Q is equal to the control subfield from P, the instruction IC + 1 is executed [the (k+1)-th SELECT SPLIT if $k < n-1$, otherwise the SELECT OFFSET];

   b. the control subfield from Q is less (greater) than the control subfield from P, the first index level of P is advanced and S_Q is reset;

   c. the control subfield from Q is greater (less) than the control subfield from P, the S_P is reset and replaces S_Q, and the first index level of P is advanced.

2 and 3. Same as Section H2.10.15.2, Paragraphs 2 and 3.

4. For the k-th SELECT SPLIT instruction (k = 1, . . . , n-1), SKIP = 1 -k.

5, 6, and 7. Same as Section H2.10.15.2, Paragraphs 5, 6, and 7.

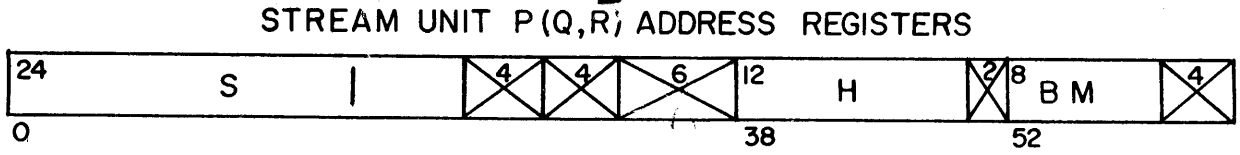
H2.10.16 Not assigned as yet.

REGISTER ADDRESS

STREAM UNIT P(Q,R) ADDRESS REGISTERS

SETUP WORD

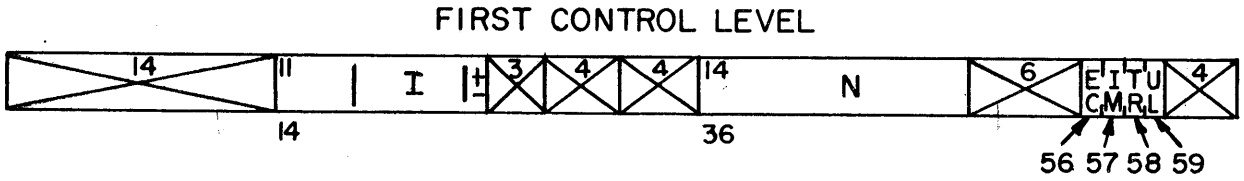
13 (16,19)



1 (4,7)

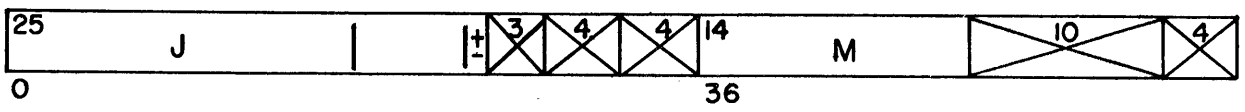
FIRST CONTROL LEVEL

14 (17,20)



2 (5,8)

15 (18,21)



3 (6,9)

AS NEEDED, CONTROL LEVELS OTHER THAN THE FIRST ARE STORED IN REGISTERS

27 (29,31)

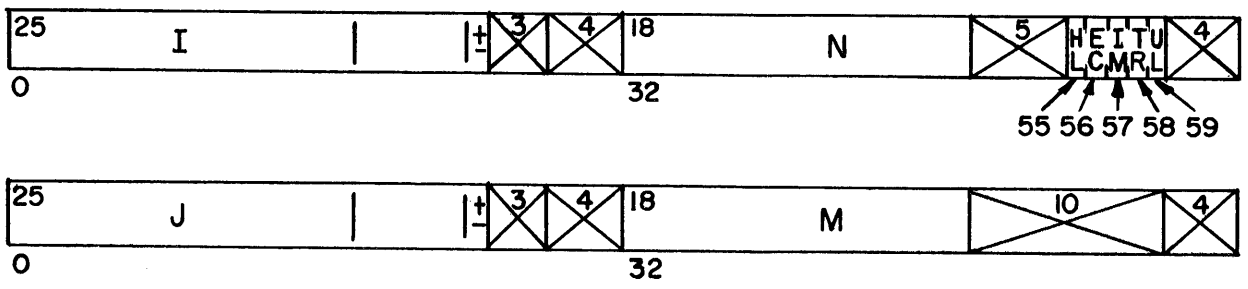
AND

28 (30,32)

SECOND CONTROL LEVEL IDENTICAL TO FIRST BUT STORED AT Ho & Ho+1

THIRD CONTROL LEVEL

STORED AT



Ho+2

Ho+3

ALL OTHER CONTROL LEVELS ARE IDENTICAL TO THIRD

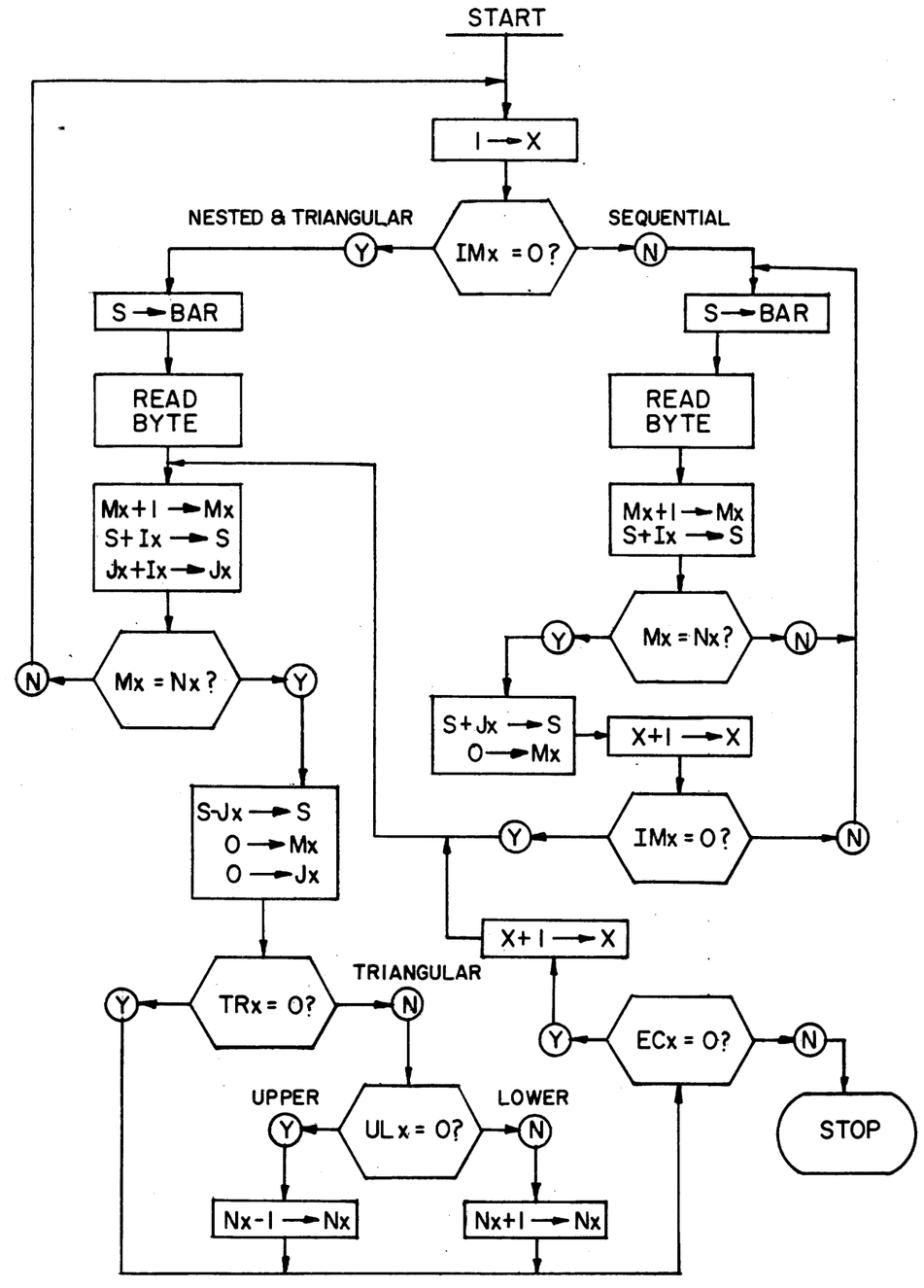
STREAM INDEX CONTROL PARAMETERS

FIGURE H1.1a

STREAM UNIT	PARAMETER	ADDRESS	FIELD LENGTH	REGISTER
P	S_P	13.0	24	13
	H_P	13.38	12	
	BM_P	13.52	8	
	I_{IP}	14.14	11	14
	N_{IP}	14.36	14	
	$(\text{CONTROL BITS})_{IP}$	14.56	4	
	J_{IP}	15.0	25	15
	M_{IP}	15.36	14	
Q	S_Q	16.0	24	16
	H_Q	16.38	12	
	BM_Q	16.52	8	
	I_{IQ}	17.14	11	17
	N_{IQ}	17.36	14	
	$(\text{CONTROL BITS})_{IQ}$	17.56	4	
	J_{IQ}	18.0	25	18
	M_{IQ}	18.36	14	
R	S_R	19.0	24	19
	H_R	19.38	12	
	BM_R	19.52	8	
	I_{IR}	20.14	11	20
	N_{IR}	20.36	14	
	$(\text{CONTROL BITS})_{IR}$	20.56	4	
	J_{IR}	21.0	25	21
	M_{IR}	21.36	14	

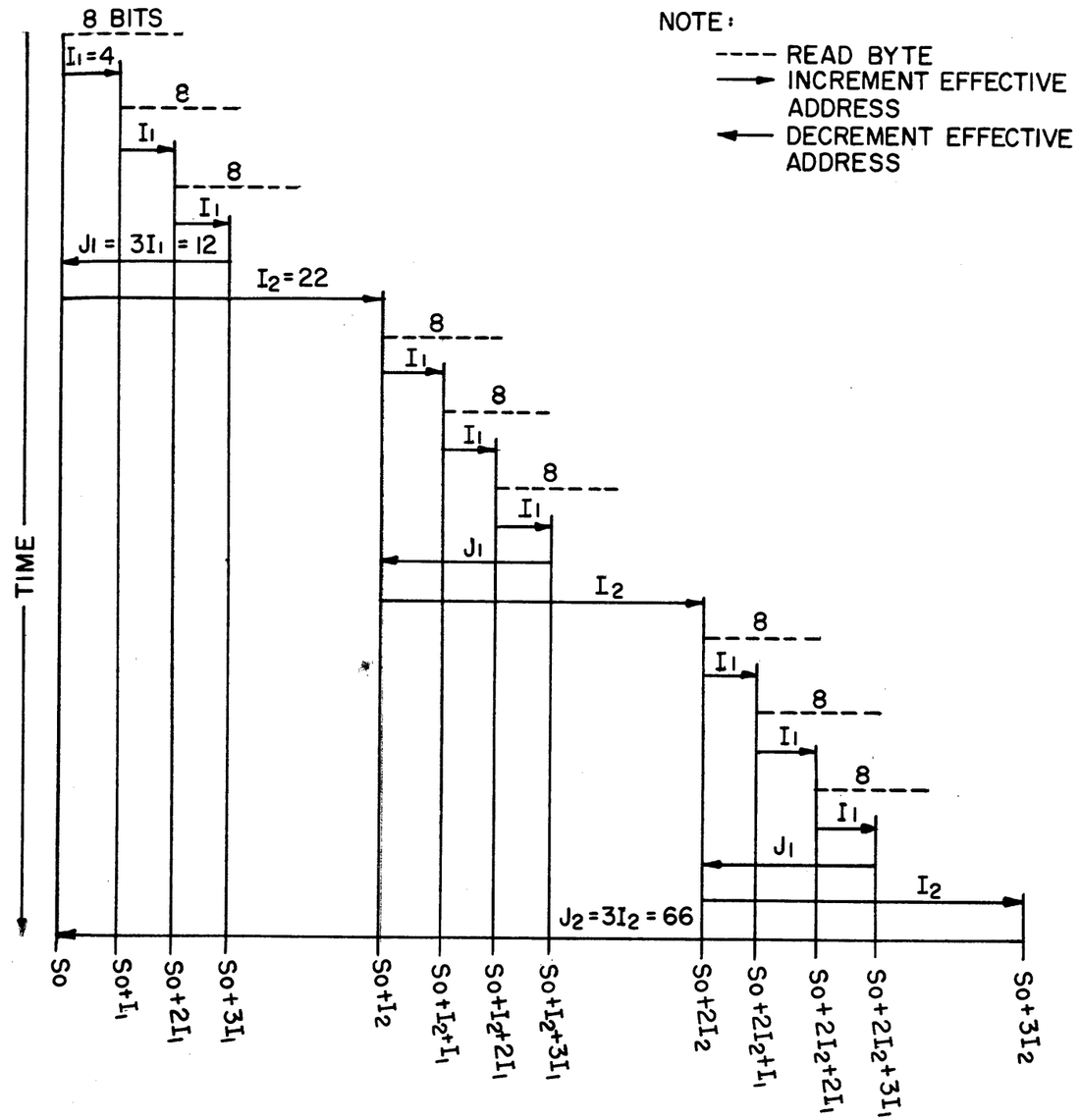
STREAM INDEX ADDRESSES

FIGURE H1.1b



STREAM INDEXING CONTROL SEQUENCE

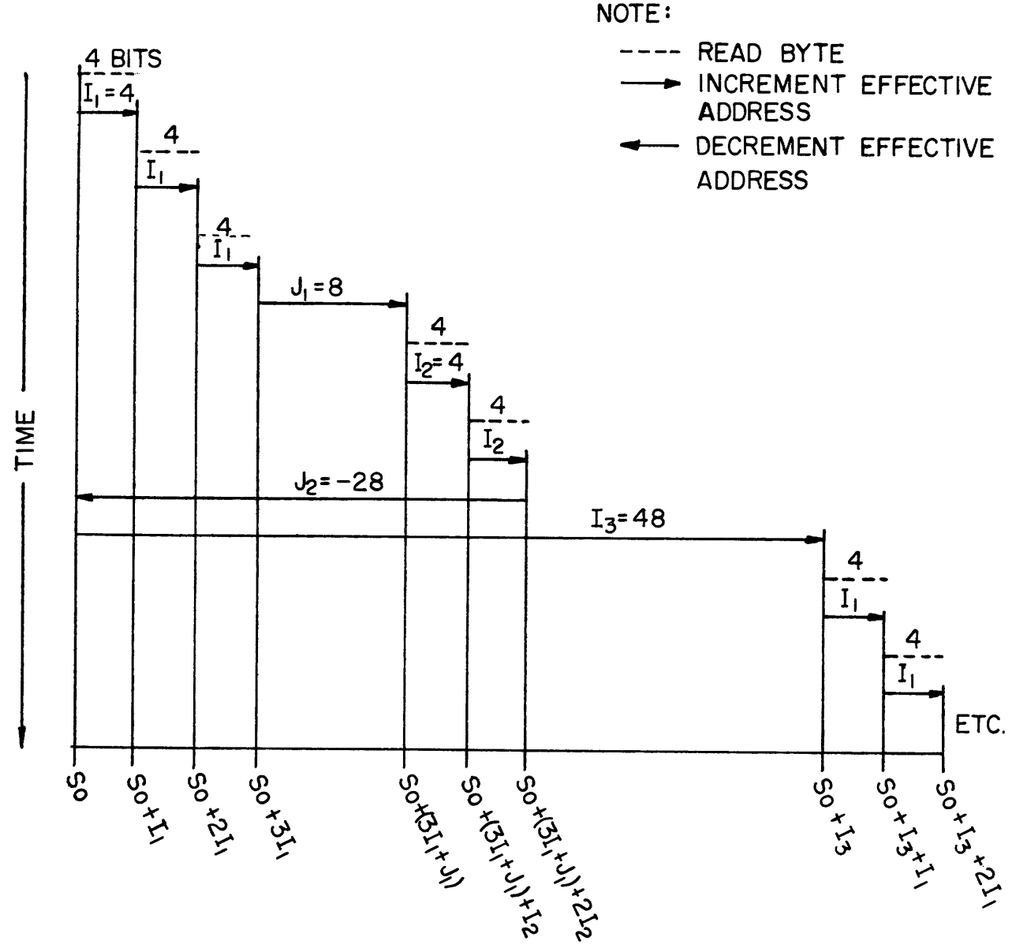
FIGURE H1.3



EFFECTIVE ADDRESSES AT CORRESPONDING TIMES

NESTED INDEXING (TWO LEVELS)

FIGURE H1.6.1

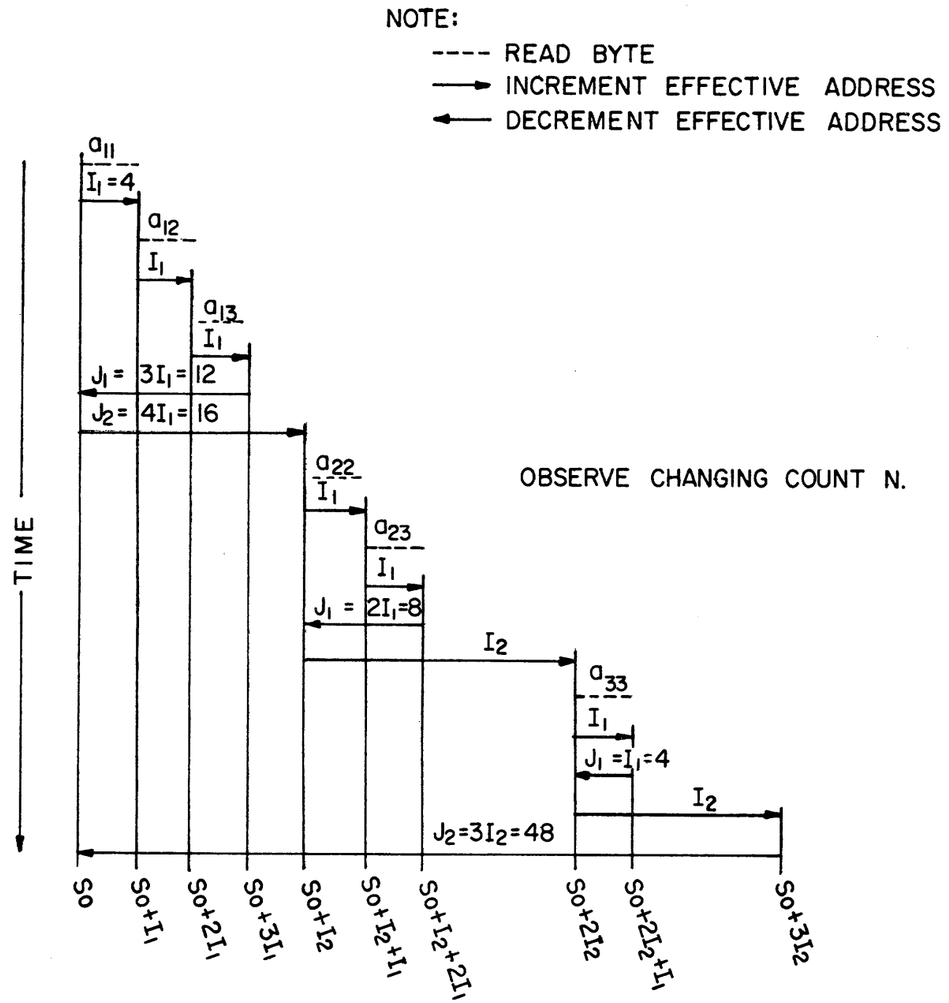


EFFECTIVE ADDRESSES AT CORRESPONDING TIMES

SEQUENTIAL INDEXING

(SEQUENCE OF TWO LEVELS NESTED IN A THIRD)

FIGURE H1.6.2

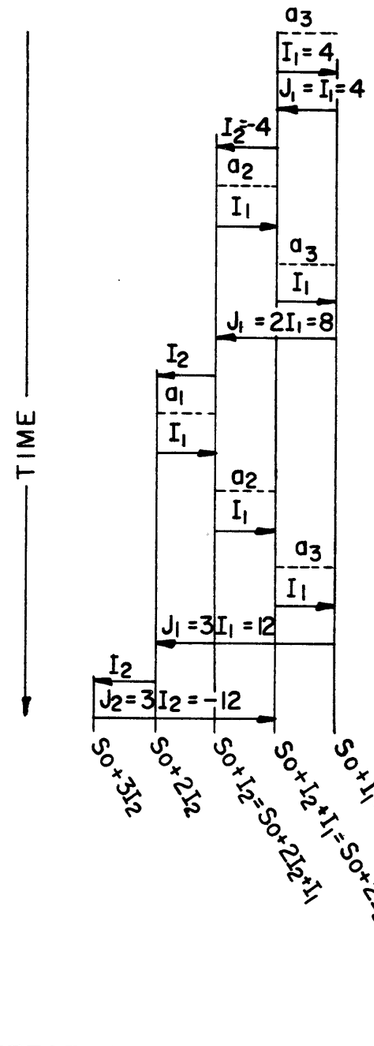


EFFECTIVE ADDRESSES AT CORRESPONDING TIMES

TRIANGULAR INDEXING

(UPPER RIGHT MATRIX TRIANGLE)

FIGURE H1.6.3a



NOTE:

---- READ BYTE

→ INCREMENT EFFECTIVE ADDRESS

← DECREMENT EFFECTIVE ADDRESS

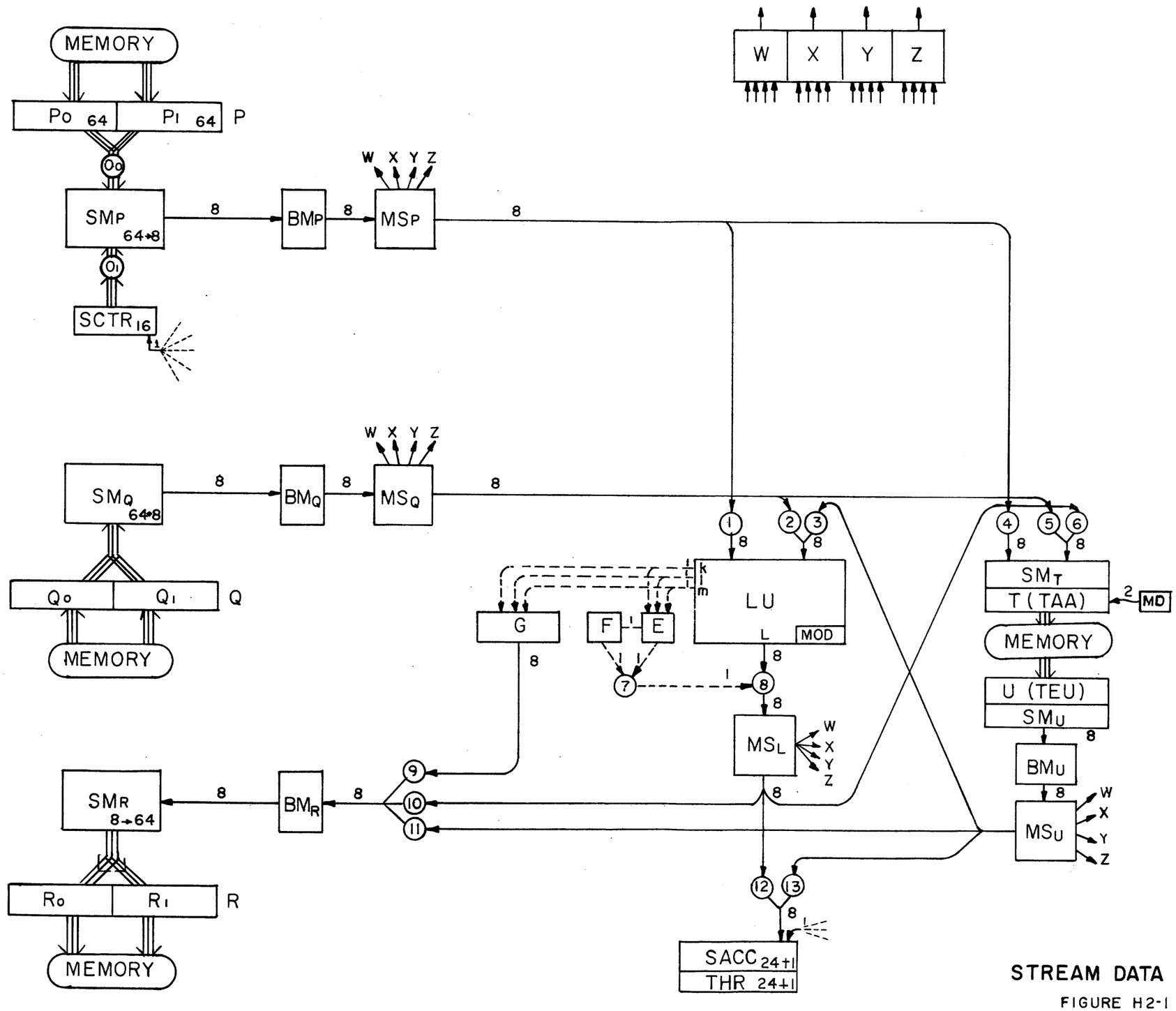
OBSERVE CHANGING COUNT N_1 AND NEGATIVE INCREMENT I_2

EFFECTIVE ADDRESSES AT CORRESPONDING TIMES

TRIANGULAR INDEXING

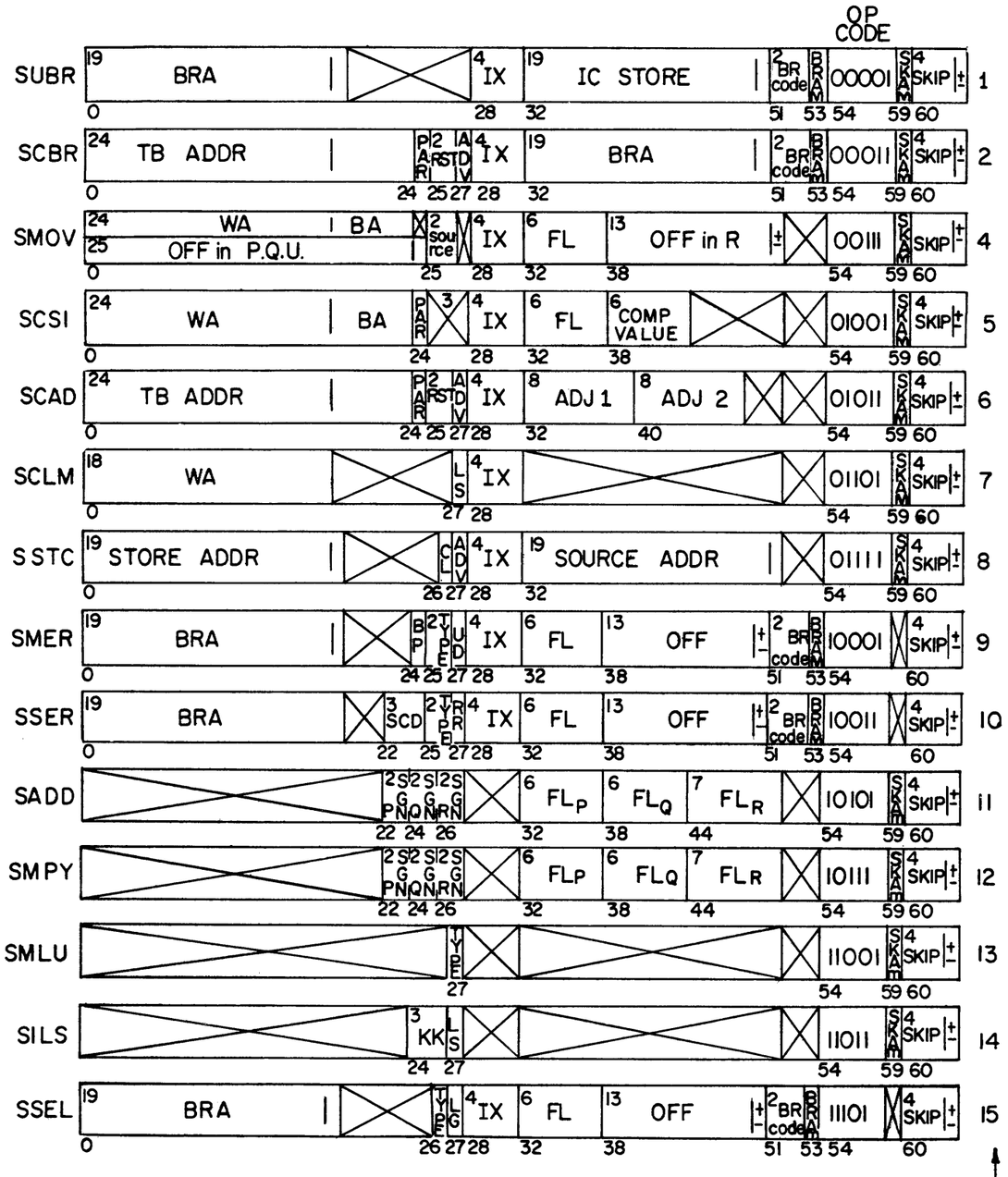
(LOWER RIGHT VECTOR TRIANGLE)

FIGURE H1.6.3b



STREAM DATA PATHS

FIGURE H2-1

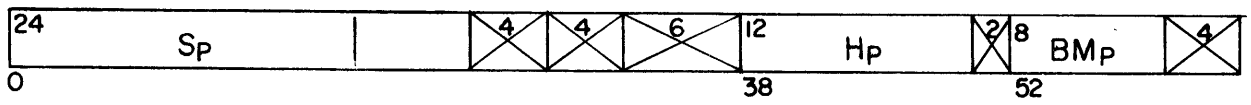
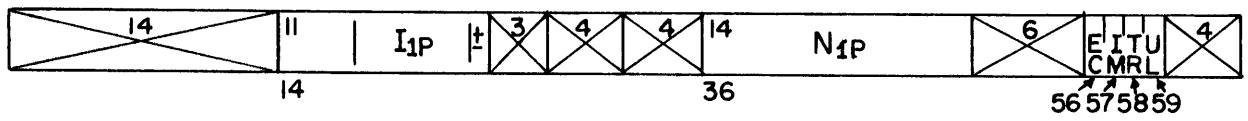
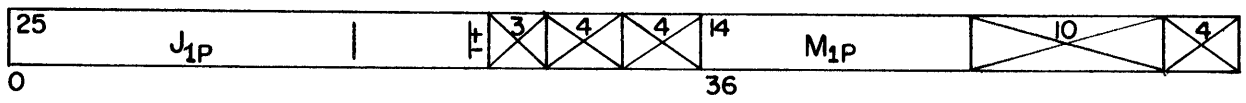
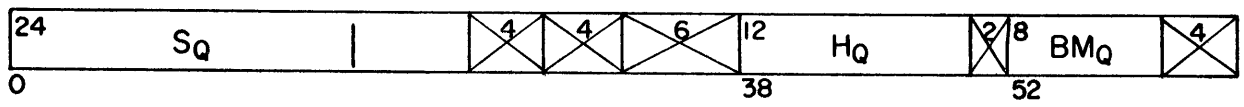
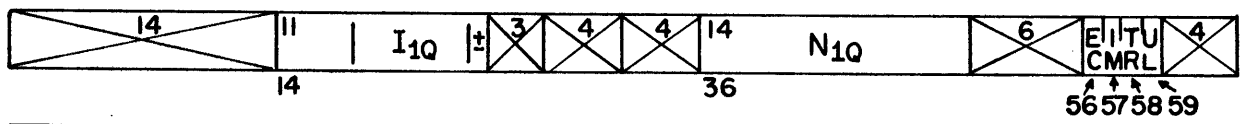
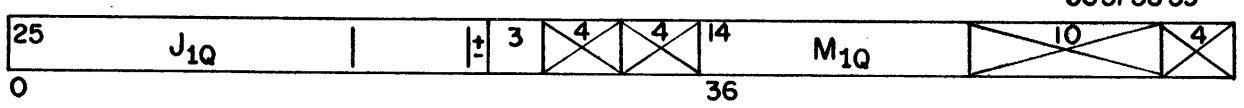
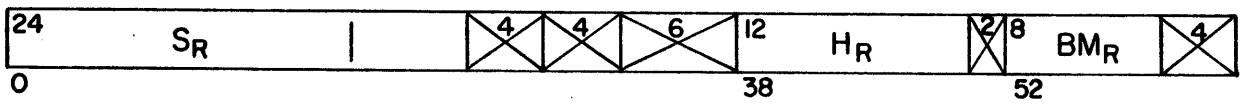
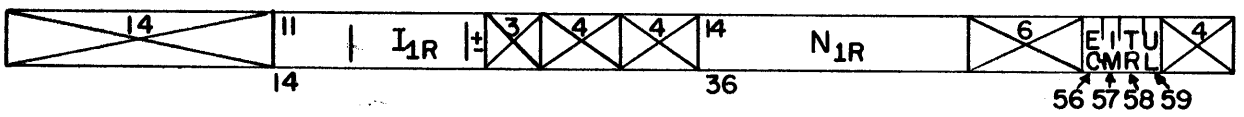
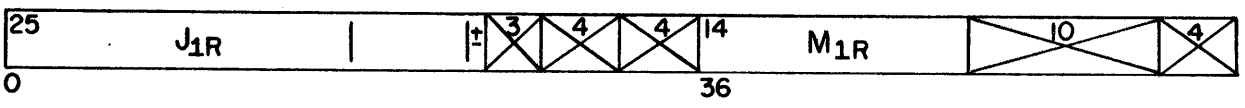
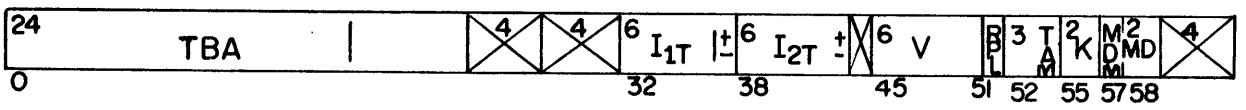
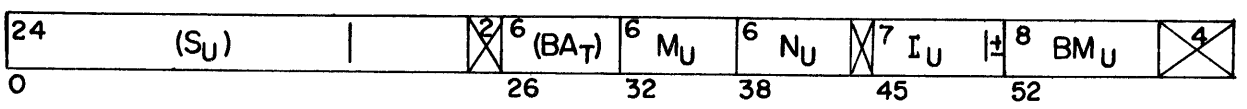
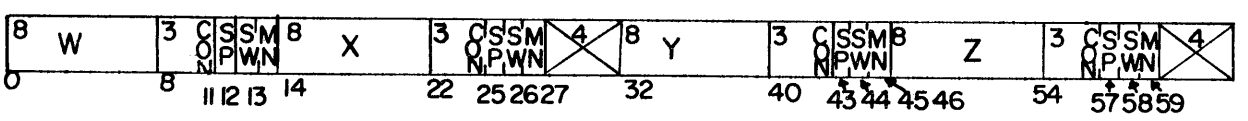
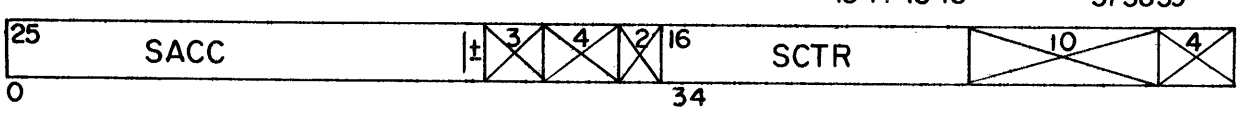
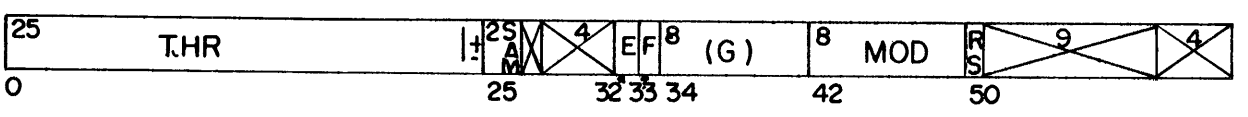
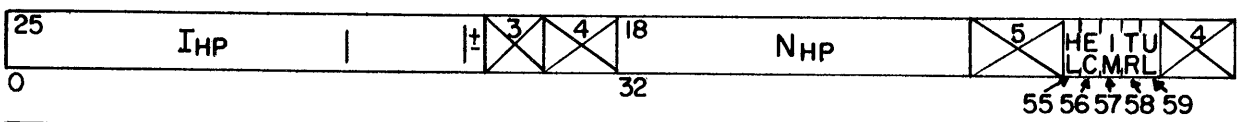
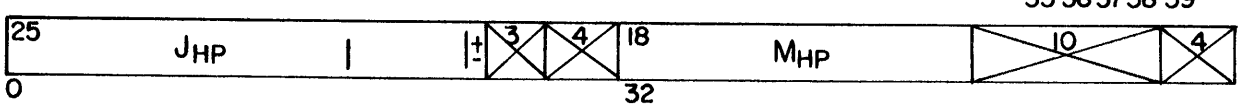
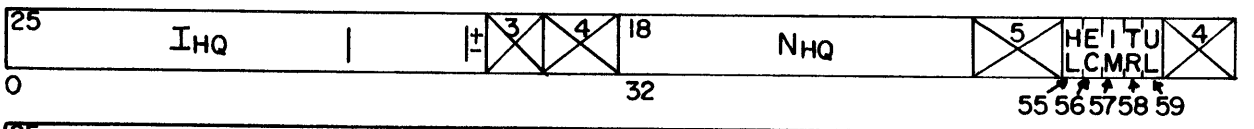
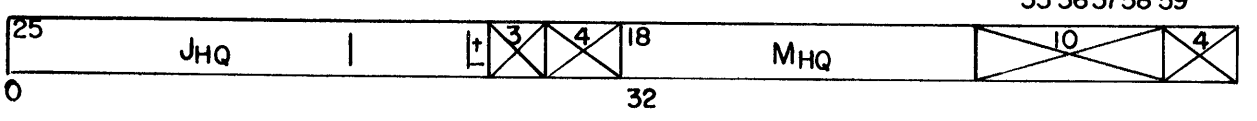
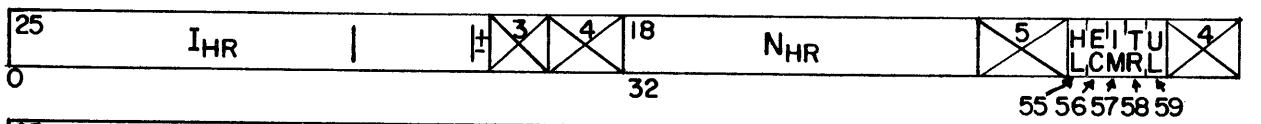
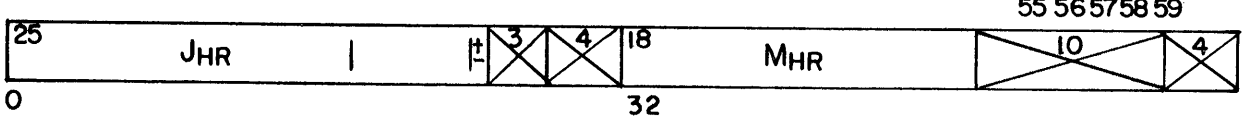


OP CODES 00101 & 11111 NOT AS YET ASSIGNED

H2.10

STREAMING MODE
HYBRID INSTRUCTION FORMATS

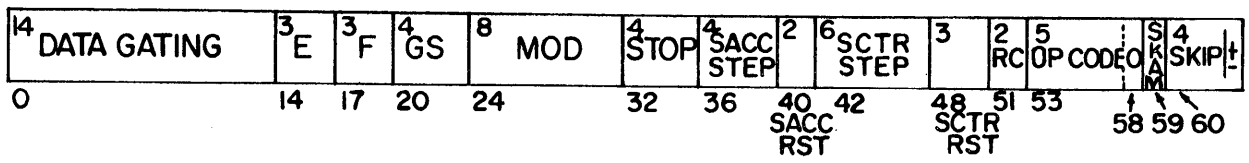
FIGURE H2.10

SETUP REGISTER	FORMAT	SETUP WORD
13		1
14		2
15		3
16		4
17		5
18		6
19		7
20		8
21		9
22		10
23		11
24		12
25		13
26		14
27		15
28		16
29		17
30		18
31		19
32		20

HARVEST SETUP

FIGURE H2.2

INSTRUCTION WORD

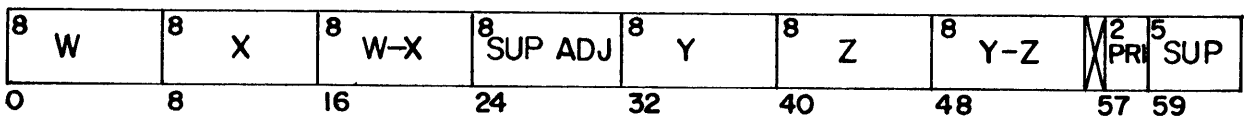


STREAMING MODE
BYTE-BY-BYTE FORMAT

FIGURE H2.4

SBBB

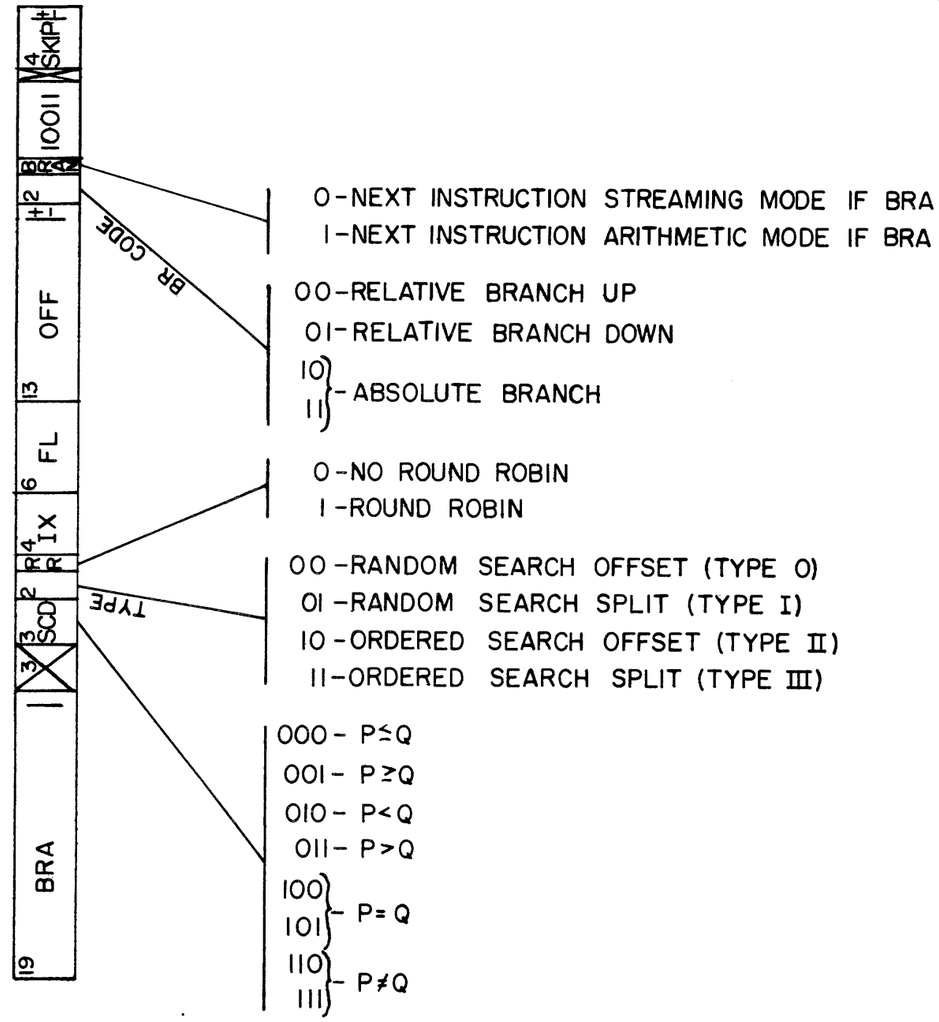
ADJUSTMENT WORD



STREAMING MODE
ADJUSTMENT FORMAT

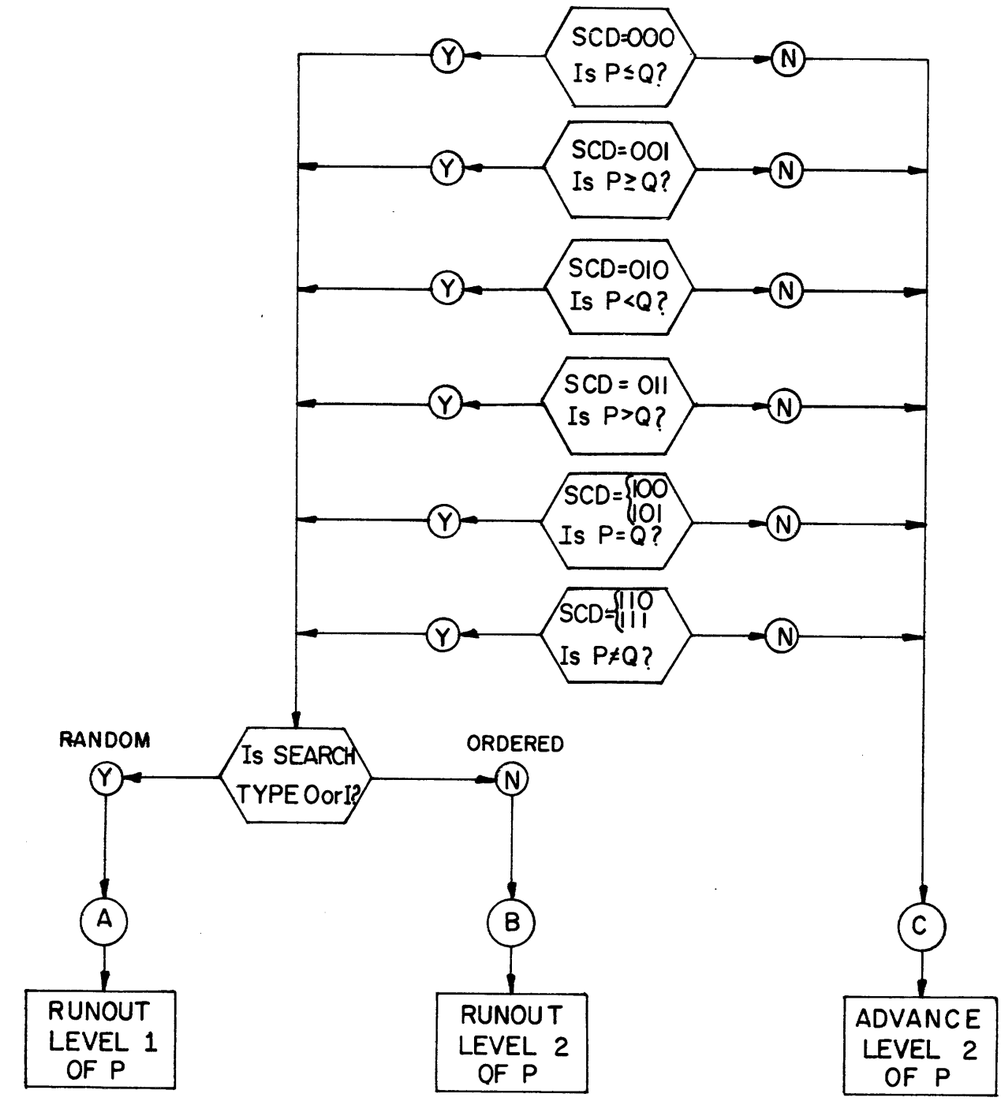
FIGURE H2.8

SSER



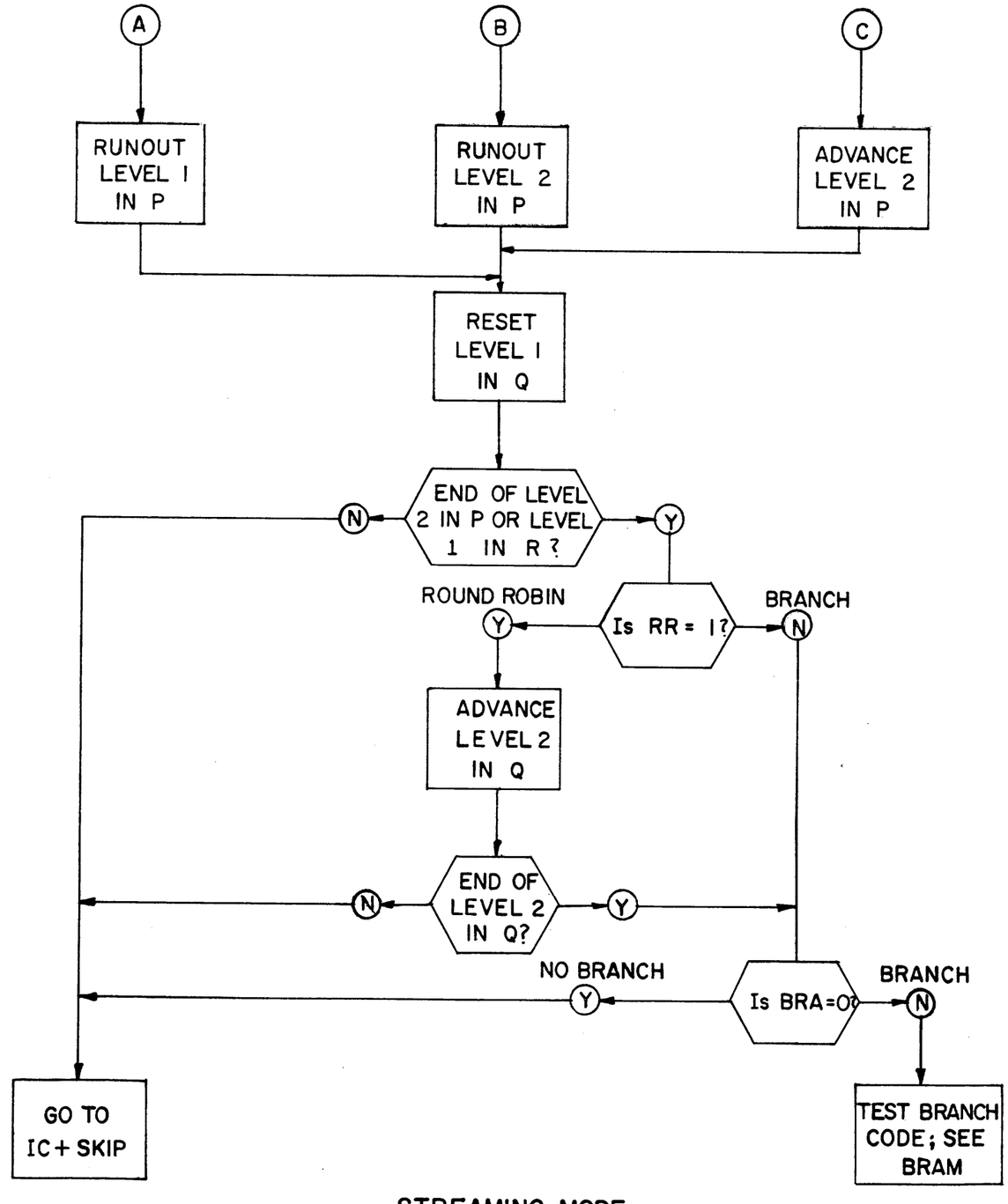
STREAMING MODE SEARCH FORMAT

FIGURE H2.10.10a



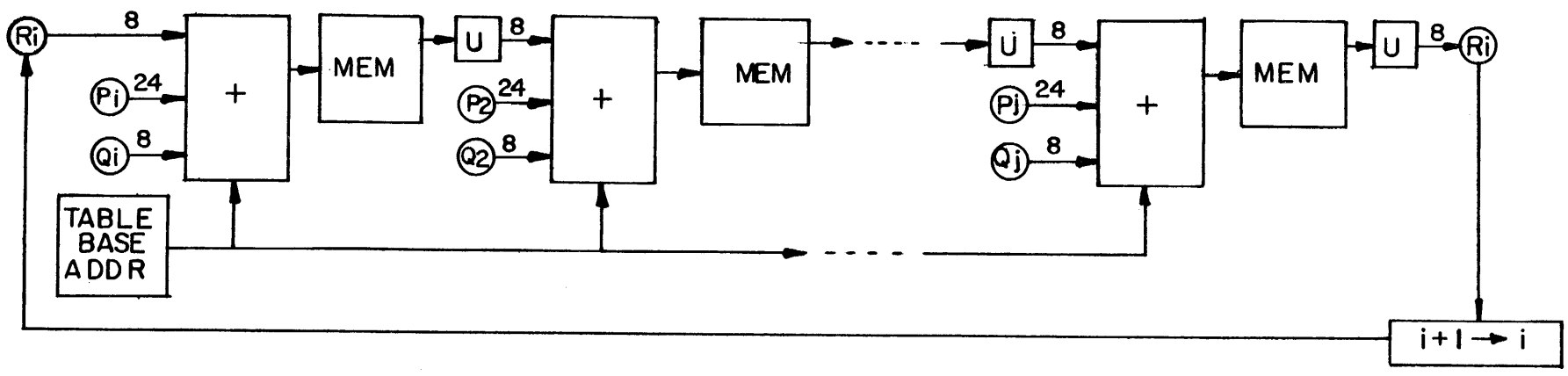
STREAMING MODE
SEARCH CONDITIONS

FIGURE H2.10.10b

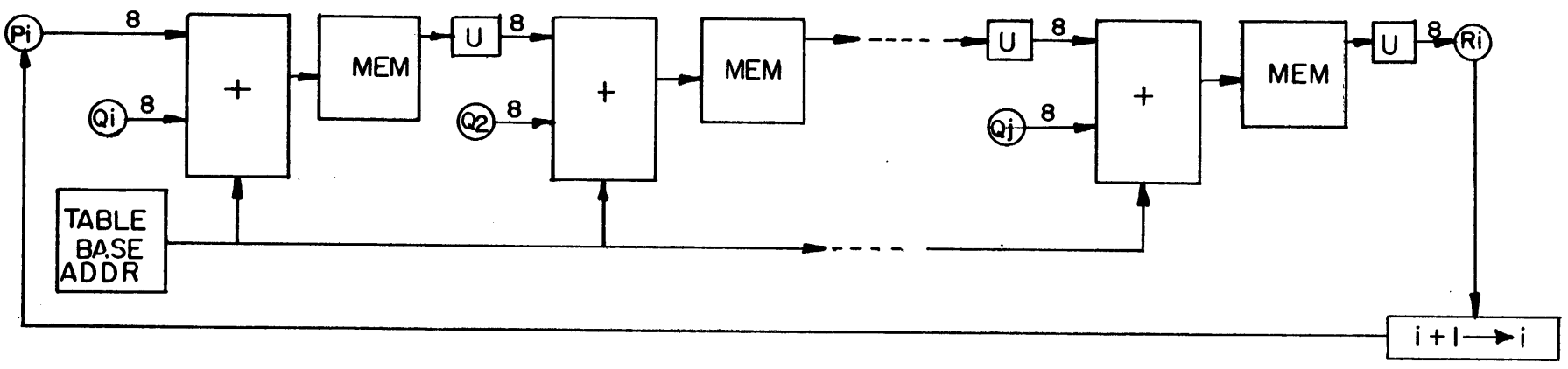


STREAMING MODE
SEARCH CONTROL SEQUENCE

FIGURE H2.10.10c



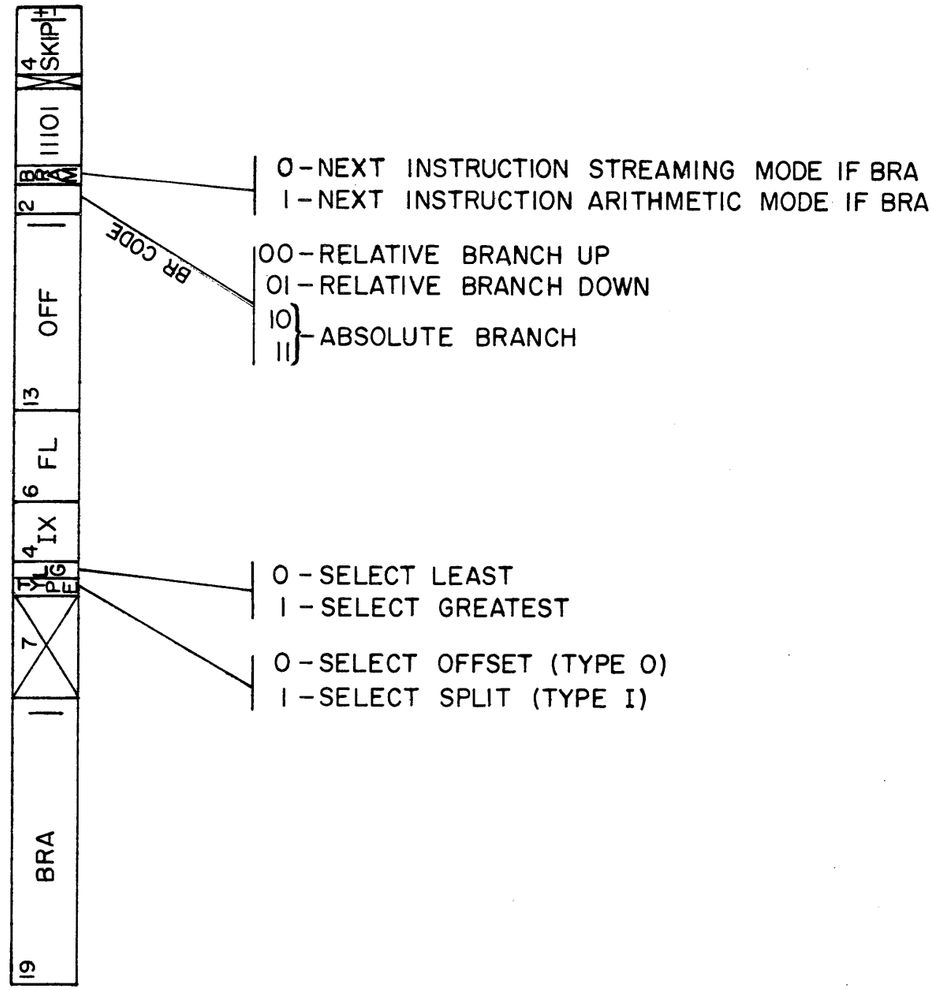
STREAMING MODE
STREAM MULTIPLE LOOK UP DATA FLOW
TYPE 0



STREAMING MODE
STREAM MULTIPLE LOOK UP DATA FLOW
TYPE I

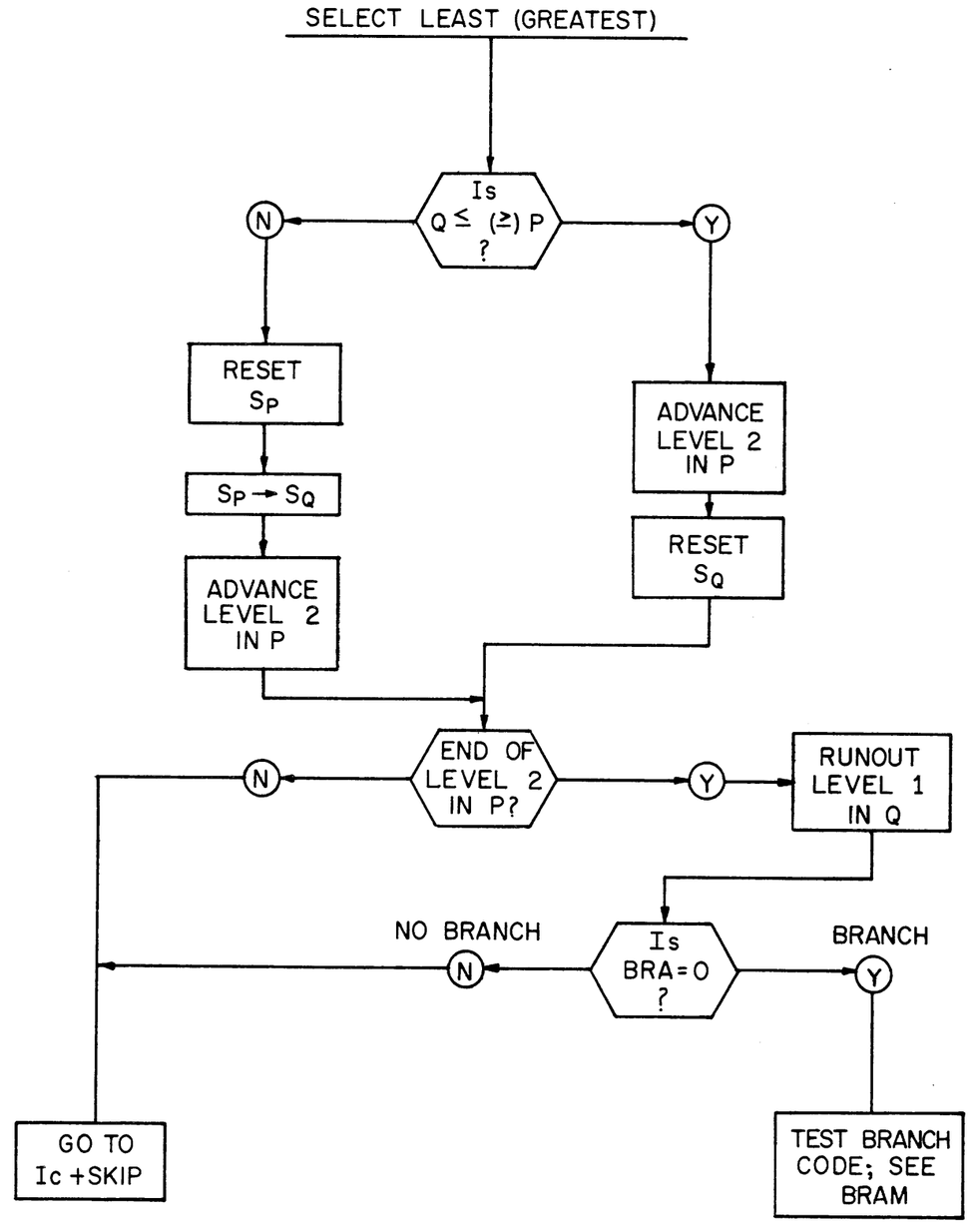
FIGURE H2.10.13

SSEL



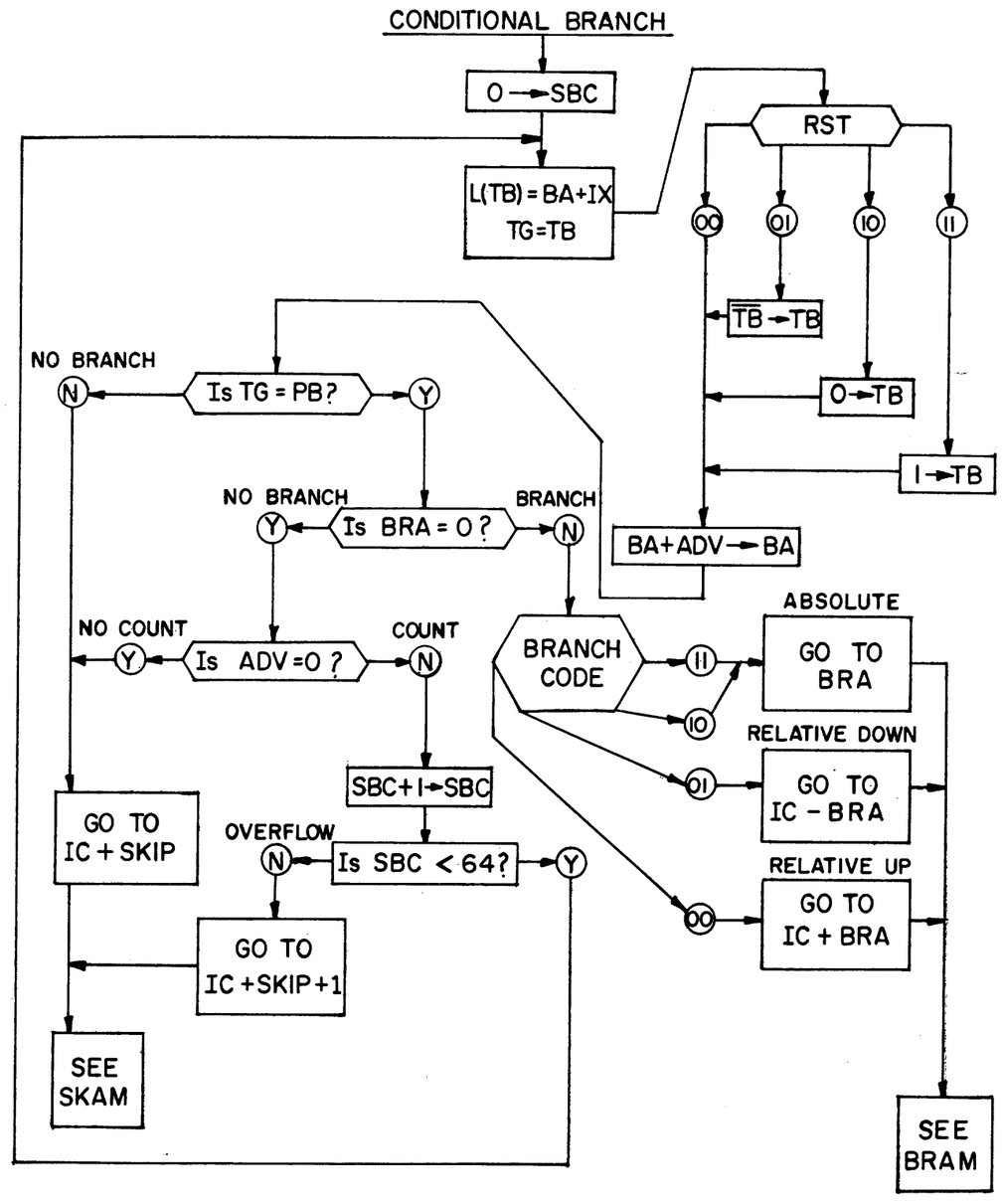
STREAMING MODE
SELECT FORMAT

FIGURE H2.10.15a



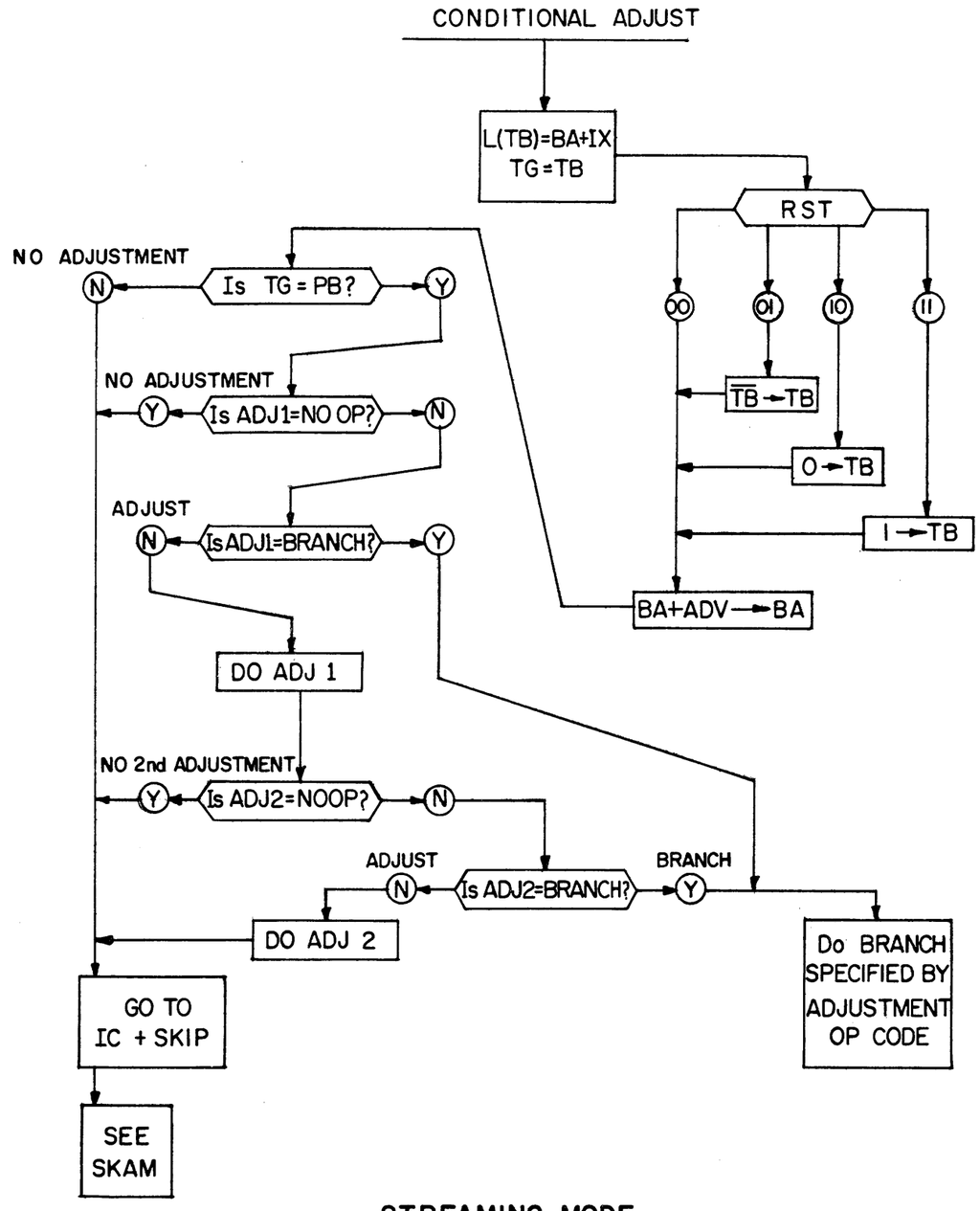
STREAMING MODE
SELECT CONTROL SEQUENCE

FIGURE H2.10.15b



STREAMING MODE
CONTROL SEQUENCE FOR
CONDITIONAL BRANCH

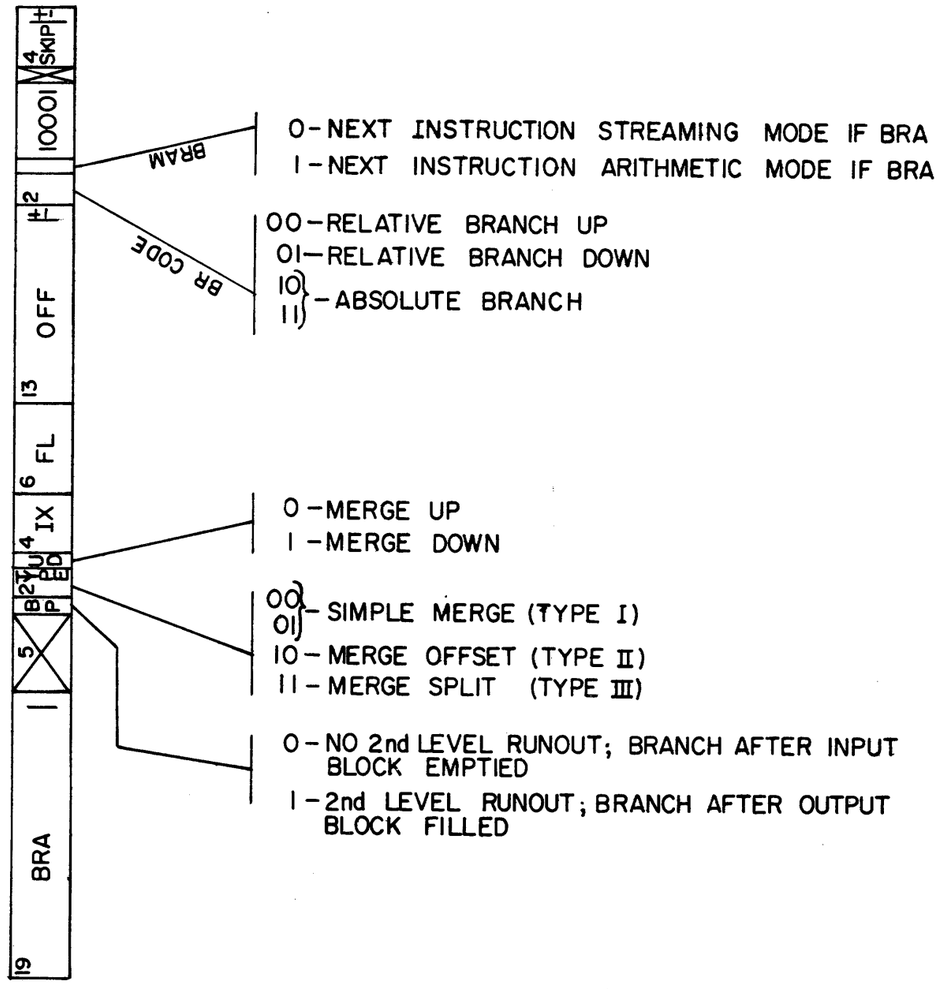
FIGURE H2.10.2



STREAMING MODE CONTROL SEQUENCE FOR CONDITIONAL ADJUST

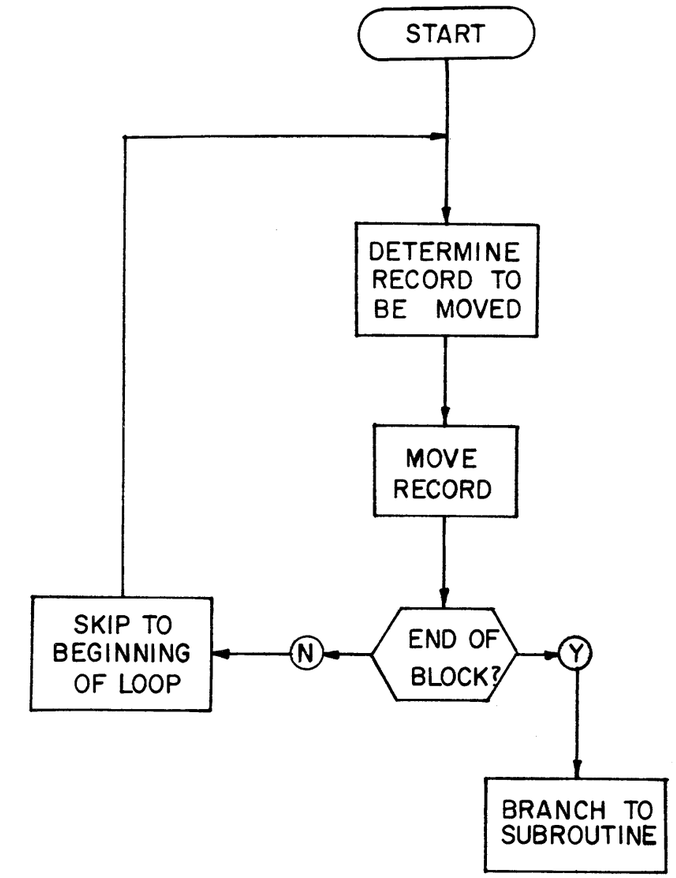
FIGURE H2.10.6

SMER



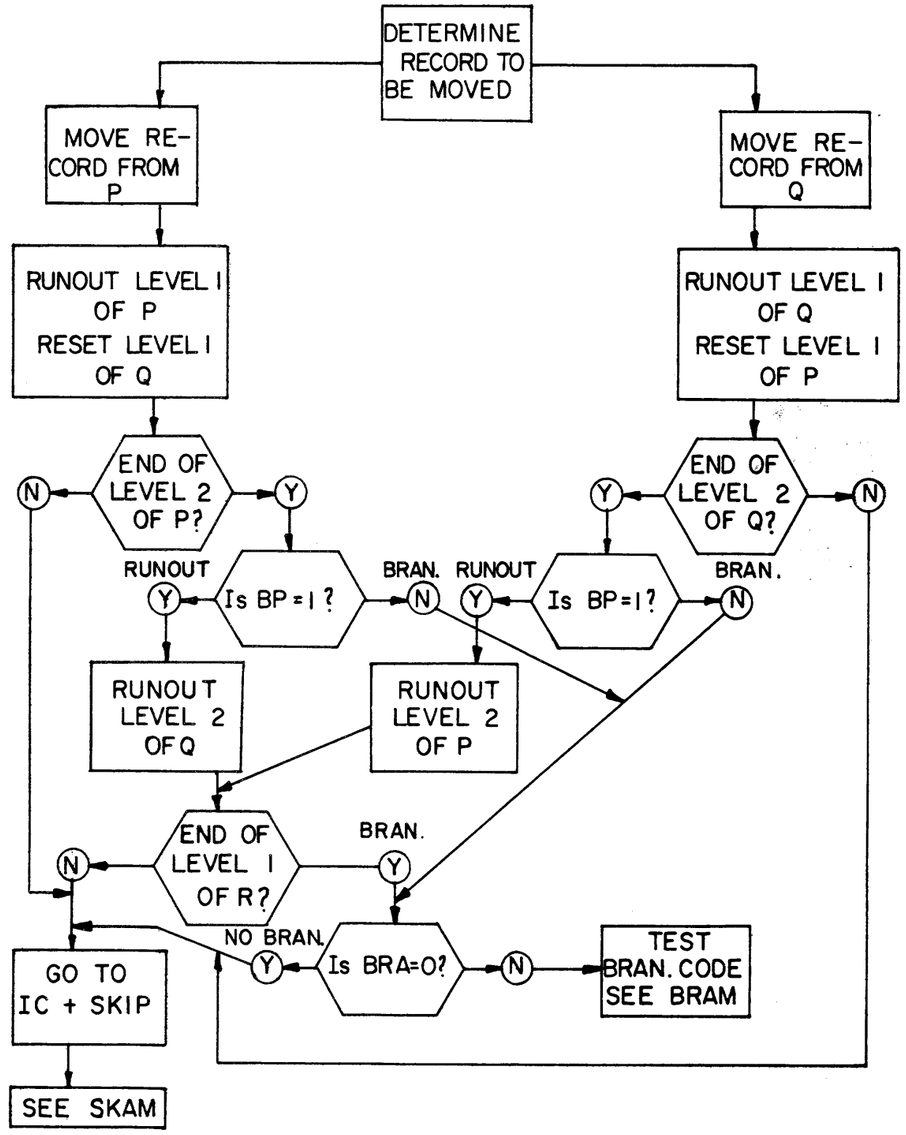
STREAMING MODE MERGE FORMAT

FIGURE H2.10.9a



STREAMING MODE
GENERAL MERGE LOOP

FIGURE H2.10.9b



STREAMING MODE
GENERAL MERGE CONTROL
SEQUENCE

FIGURE H2.10.9c