THE HARVEST SYSTEM

Paul S. Herwitz and James H. Pomerene

Product Development Laboratory, Data Systems Division
International Business Machines Corporation
Poughkeepsie, New York

Summary

The Harvest System is a large-scale data processor designed for maximum performance in handling extremely large amounts of data in primarily non-arithmetic operations. It is being built by IBM under contract with the Government and incorporates both a new system organization and a stored program concept of macro-instructions which directly implement many useful data manipulating subroutines. Design features include a very high processing rate and an on-line table lookup facility for effecting very general transformations.

Introduction

Most of the information generated and handled in our society is either non-numeric or developed within areas supported by little or no theoretical structure. A general-purpose system for processing this type of information must be organized very differently from more conventional scientific computers. Many of these areas are in the early stages of scientific development where ordering and classification are predominant activities and where the major problem is to uncover patterns and trends upon which theory can be built. There are few rules for determining the relevancy of information so that often enormous amounts must be examined to achieve significant results. Comparatively simple operations and data transformations are the most useful, and the output desired may frequently be some statistical characterization of the input.

The Harvest System is a large scale computer intended for maximum performance in this broad area of information processing. It was defined and is being built under study and development contracts between IBM and the Government. Harvest comprises a major data processing system reflecting the above considerations and includes an IBM Stretch computer for high performance on conventional operations (Figure 1). Stretch has been described elsewhere; only the special data processing portion is discussed here.

Harvest Organization

The Streaming Mode

Processing in Harvest is parallel by character, represented by a quantity of 8 bits or less. This quantity is called a byte and is the basic information unit of the system. The streaming mode is primarily a design attitude whose aim is to select bytes from memory according to some preassigned pattern and to keep a steady stream of such selected bytes flowing through a designated process or transformation and thence back to memory. Emphasis is on maximum flow rate so that the typical large volumes of information can be processed in minimum times. Processing time per byte is held to a minimum by prespecifying byte selection rules, processing paths, and even methods for handling exception cases, so that decision delays are suffered but once for a long sequence of bytes rather than being compounded for each byte. There is a functional analogy to plugboard machines except that the plugging can be changed at high electronic speeds and much of it on the basis of data encountered.

Stream Formation

The bytes which are selected to form the stream are taken from memory according to either simple or complicated patterns which are chosen by the programmer. For technical reasons memory is organized into 64-bit words, but this artificial grouping is suppressed in Harvest so that memory is handled as a long string of bits any one of which can be addressed for selection. Up to 2^{18} words of memory can be directly selected, and since the word size is exactly 2^6 a Harvest address consists of 24 bits: 18 to select the word and 6 to select the bit within the word.

Data are transferred to and from memory as 64 parallel bits; selection down to the bit is accomplished by generalized operand registers called stream units (Figure 2). Each stream unit contains also a switching matrix which allows a byte to be selected out with minimum delay, starting at any bit position within the register. To handle cases where a byte overlaps from one memory word into the next and to minimize waiting time for the next needed word from memory, each stream unit is actually 2 words (i.e., 128 bits) long. The byte output of the stream unit is fed into the processing area through a bit-for-bit mask which enables the programmer to pass any subset of the 8 bits, including non-consecutive combinations. The selection of these bytes is controlled by the low order 6 bits of a stream of 24-bit addresses generated by the pattern selection units. There are two source stream units which feed operands into the processing area and one sink stream unit which accepts results from the processing area.

Pattern Selection

The data input to Harvest may be highly redundant to any particular problem, and so a powerful mechanism is provided for imposing selection patterns on the data in memory. It is assumed that the very effective input-output control in the basic Stretch system has grossly organized the contents of memory. For example, various characteristics may be recorded for a population and recorded in uniform subdivisions of a file. A particular problem may be concerned with only a certain characteristic drawn from each record in the file. Again, for example, data may be stored in memory in matrix form and the particular problem may require the transpose of the matrix.

Pattern selection in Harvest resembles indexing in other computers, except that in Harvest the programmer determines the algorithm which generates the pattern rather than listing the pattern itself. Each stream unit has its independent pattern generating mechanism which is actually an arithmetic unit capable of performing addition, subtraction, and counting operations on the 24-bit addresses. The programmer specifies patterns in terms of indexing levels. Each level consists of an address incrementing value I which is successively added to the starting address value S until N such increments have been applied, after which the next indexing level is consulted to apply a different increment. The programmer may then choose that incrementing continue on this level or that the previous level be resumed for another cycle of incrementing.

Many other indexing modes are provided to permit almost any pattern of data selection. Particular attention has been given to direct implementation of triangular matrix selection and to the iterative chains of any formal inductive process, however complex.

In general, the pattern selection facilities completely divorce the function of operand designation from that of operand processing, except that pre-designated special characteristics of the operands may be permitted to change the selection pattern in some fashion.

Processing Facilities

The pattern selection units determine the movement of data between the stream units and memory and, together with the stream units, determine the byte flow in the processing area. The processing facilities, together with the selection facilities, have been designed to give a flow rate of approximately 4 million bytes per second. Source stream units to the processing area are stream units P and Q while the sink stream unit is stream unit R.

Transformation Facilities

Two facilities are provided for the transformation of data (Figure 3). Extremely general operations on one or two input variables can be accomplished with the on-line table lookup facility. Simpler operations can be done directly by the logic unit without involving memory lookup. The logic unit also provides a choice of several one-bit characterizations of the input bytes (such as Byte from P $>$ byte from Q). These one-bit signals can be used to alter the stream process through the adjustment mechanism.

The table lookup facility consists of two units. The more important logically is the Table Address Assembler which accepts bytes from one or two sources and from them forms the lookup addresses which are sent to memory (Figure 4). The other is the Table Extract Unit which permits selection of a particular field within the looked-up word. Both units have their own indexing mechanisms and together they permit the programmer to address a table entry ranging in size from one bit up to a full word and starting at any bit position in memory. This freedom is abridged only by considerations of the table structure chosen by the programmer.

The table lookup facility also provides access to the memory features of Count and Existence. Under instruction from the Table Address Assembler the medium speed memory can use the assembled address to logically OR a one into the referenced bit position. The referenced word as it was just before the ORing can be sent to the table extract unit. In the high-speed memory a one may be either ORed or added into the referenced bit position with the same provision for sending the word before alteration to the table extract unit. The ability to add ones into high-speed memory words permits use of these words as individual counters. Several counter sizes can be specified.

Statistical Aids

The table lookup facility may be used to associate statistical weights with the occurrence of particular sets of bytes. For example, the occurrence of a byte P_i in the P stream together with a byte Q_j in the Q stream may be assigned a weight W_{ij}, which would be stored in a table and referenced by an address formed from both P_i and Q_j. Alternatively a memory counter may be associated with each pair P_i, Q_i and stepped on the occurrence of each such pair.

A Statistical Accumulator (SACC) is provided (Figure 5) either to sum the weights W over a succession of sets of bytes or to provide a key statistical measure of the counting results. In addition, SACC can be used for many other accumulating purposes.

A Statistical Counter (SCTR) provides a way of counting the occurrences of any of a large number of events during a stream. In particular, SCTR can be designated to count the number of weights W which have been added into SACC.

The Stream Byte-by-Byte Instruction

The table lookup unit, the logic unit, and the statistical units can be connected into the processing stream in various ways by the programmer. Like a class of analog computers, these connections reflect the structure of a problem and are the electronic equivalents of a plugboard. The hookup chosen by the programmer then applies the same processing to each byte or pair of bytes sent through it; this very general processing mode is called the Stream Byte-by-Byte instruction. The connections, indexing patterns, and special conditions described below all form part of a pre-specified setup which can be considered as a macro-instruction putting the computer into a specific posture for a specific problem.

Monitoring for Special Conditions

The concept of a streaming process determined by a flexible and extensive setup specification is most meaningful when applied to a large batch of data which is all to be treated the same way. In any particular streaming process, special conditions may arise within the data being processed which call for either momentary or permanent changes in the process. For example, the transformation being performed may be undefined for certain characters so that these must be deleted at the input; or a special character may be reserved to mark the end of a related succession of bytes after which the process or the pattern of data selection must be altered.

Special conditions can be monitored in several ways (Figure 5). Special characters can be detected by Match Units, each of which can be assigned a special 8-bit byte to match. There are four Match Units; W, X, Y, and Z, which can be connected to monitor the stream at several different points. When a match occurs, the Match Unit can directly do one of several operations on the stream and can also emit a one-bit signal indicating the match.

A large number of other one-bit signals are generated by the various stream facilities to mark key points in their respective processes. These one-bit signals, collectively called stimuli, can be monitored to accomplish specific operations, such as stepping SCTR or marking the end of an indexing pattern. They can also be used to accomplish a much wider range of operations through the adjustment mechanism.

Up to 64 stimuli are generated by the various processing, indexing, and monitoring functions in Harvest. For any particular problem those stimuli which represent the key or significant properties of the data being streamed can bė chosen. To each stimulus or coincident combination of stimuli the programmer may associate one or more of a large number of reactions on the stream; its data, its process, or its indexing. These stimulus-reaction pairs are called adjustments. The adjustment mechanism gives the programmer a direct way of picking out those elements of the stream which are different from the general run. These different elements may provide the key to the pattern being sought, either because they are particularly relevant or because they are distinctly irrelevant.

Programming Harvest

The Instruction Set

Conventional arithmetic and scientific computational processes and all input-output operations for Harvest are performed in the Stretch computer which is part of the system. When Stretch instructions are used, the system operates in the arithmetic mode; when streaming mode control is imposed, the unusual instructions unique to Harvest are available. There are about 85 instruction families (i.e., instruction types plus associated operation modifiers) in Stretch alone. The Harvest stream instructions add a variety of extremely powerful data processing tools to the several thousand basic Stretch operations. Throughout the system the instruction formats vary in length: single-address instructions are 32 bits long, two-address instructions and instructions that operate on variable length fields are 64 bits long, and stream instructions have an effective length of 192 bits.

Streaming mode instructions are very much like built-in subroutines or macro-instructions. Just as it is necessary to initialize a programmed subroutine, it is also necessary to initialize or set up the Harvest processor. Roughly about 150 parameters and control bits may influence streaming. Harvest is set up by loading values of some of these parameters into and setting the desired control bits in specific, addressable setup registers prior to the execution of a stream instruction. Certain changes in the parameter values or control bit settings generate stimuli which may be used to terminate or cause automatic adjustments to be made to the stream, or to cause a change to the arithmetic mode of operation. The adjustment operations essentially constitute a second level of stored program and are used most generally to handle the exception cases that occur during processing operations. Thus the programmer sets up Harvest to execute a stream instruction; execution begins and is automatically modified as changing data or setup dictates; much routine bookkeeping is done automatically by the several independent pattern generating (indexing) mechanisms; changing values of parameters are always available for programmed inspection if automatic inspection is not sufficient for the particular operation being performed.

While most of the programming in the streaming mode of operation is centered around the Stream-Byte-by-Byte instruction, a number of other instructions derive from the unique organization of Harvest. The arrangement of the Harvest data paths and processing units facilitates the inclusion of operations that perform within one instruction each many of the routine collating functions such as merging, sorting, and file searching and maintenance that are so common to commercial data processing. Facilities of the table lookup unit are used extensively in these as well as in several other instructions designed primarily for the logical manipulation of data.

Since such extensive use is made of tables of parameters, table transformations, and other data arrays in the Harvest approach to data processing, a special Clear Memory instruction is available for clearing large blocks of memory in minimum time and with minimum programming effort. A single execution of this instruction will clear as few as 64 consecutive words or as many as 2048, depending on the programmer's wishes. The execution time in the latter case is less than 3-1/2 μsec, and only one instruction access from memory is made. A full memory complement of 2^{18} words can be cleared in less than one millisecond.

Collating Operations

Generally speaking, in order to perform merging, file searching, and other such collating operations, it is necessary to specify a number of parameters such as record length file length, control field length and position, etc. For Harvest, these parameters need only be tabulated in proper order by the programmer. They are then utilized by the indexing mechanisms to cause data to be picked from and later be stored into memory according to the patterns that naturally occur in such data.

The Merge instruction family actually contains eight independent control sequences that may be used to merge files or even to completely sort blocks of records with but a single instruction access from memory. The particular option chosen by the programmer depends upon whether files are to be arranged in ascending or descending order, whether or not the record block can be contained in at most half the available memory, and whether the control field heads the record or is offset.

As an indication of the processing speed of Harvest, in the most favorable case (one-word records with control fields at the beginning of records) a block of 30,000 records already in memory can be completely sorted in 1-1/4 seconds or less.

The Search instruction complex consists of twelve control sequences, each of which facilitates abstracting from a master file all records whose control fields bear any specific one of six possible relationships to the control field of each record of a detail file. The possible relationships are the six standard comparison conditions $<, \leqq, >, \geqq, =, \neq$. If it is not desired to move the records that meet the search condition, it is possible to tabulate their addresses automatically.

Another complex called Select permits the programmer to select from a file that record having the least or greatest control field.

For the purpose of facilitating file maintenance operations, Harvest includes a collating instruction complex called Take-Insert-Replace. When the operation is executed under "instruction control," then whenever master and detail record control fields match, either the master record is taken out of (deleted from) the master file or is replaced by the detail record. Under "data control" the action taken whenever control fields match is indicated by the contents of a special control byte in the detail record. The masters can be deleted or replaced, or the detail record can be inserted in the master file; or under certain circumstances the maintenance procedure can be interrupted when master records with special characteristics are located, and resumed with a minimum of programming effort when desired.

Instructions such as the collating operations described above lead to a considerable reduction in the length of the generalized report generators, file maintenance routines, sorting and merging programs, etc., that might be expected to become associated with such a computer system.

Table Lookup Operations

It is often desired to be able to obtain data from or store data at an address that is not directly dependent on the data itself. The Indirect Load-Store instruction complex permits wide latitude in the formation of such addresses and in the subsequent manipulation of the original data. In essence the operation is as follows: parameters from one of the source stream units are used in the formation of an address in the table lookup unit; either this primary address itself or either of the addresses found in the word at the memory location specified by the primary address becomes either the origin of a field of data to be entered via the other source stream unit or the location at which the data field is to be stored by the sink stream unit; the data is moved from source to sink and the entire cycle is repeated. The counting and ORing features of the table lookup unit are available to the programmer as modifications of the basic instruction control sequence.

The second instruction complex built around the table lookup unit is called Sequential Table Lookup. An extremely powerful but conceptually simple instruction, it permits a class of transformations to be performed that may best be described as data dependent. The instruction causes a series of table references to be made, each successive reference after the first being made in a table whose address is extracted automatically from the previously referenced table entry. Also, as each reference is completed, a variable amount of data may be extracted from the table entry. Moreover, the indexing of the input or output data may be adjusted according to the contents of the table entry (similar to the operation of the Turing machine). The applications of Sequential Table Lookup are manifold; the editing for printing of numerical data, the transliteration of Roman numerals to Arabic, and the scanning of symbolic computer instructions for assembly and compilation purposes are but a few.

The extensive use of tables in problem solution typifies the different approach the programmer will take with Harvest. The problem of transliteration of Roman numerals to Arabic illustrates the power of the method. Several simplifying assumptions have been made so that the flow chart is easier to follow. First, the data - a set of numbers expressed in Roman numerals, each number separated from the next by a blank (B) - is assumed to be perfect, and only the characters I, V, X, L, C, D, and M are used. Second, the set of numbers is terminated by two blanks, Third, the use of four successive identical characters (as Roman IIII for Arabic 4) is outlawed. Finally, the numbers to be transformed are all assumed to lie on the range 1 to 1000, inclusive.

The flow chart (Figure 6) shows the eighteen tables (consisting of a total of eighty-two memory words) used. Under each table heading a two-part entry is shown, the parts separated by a colon. On the left of the colon is the argument being looked up, followed in parentheses by an indication of the range on which the number or digit that will eventually result must lie. On the right of the colon the parameters of the table word corresponding to the argument are indicated symbolically: e.g., RO-1B (meaning "read out the integer 1 followed by the character for blank") or NRO (meaning "no readout"). This is followed by an integer in parentheses indicating what data byte is the next argument (0 means same byte, 1 means next byte, etc.); the arrow indicates the table in which the next argument is looked up.

As an illustration, consider the transliteration of DCLXXVIII:

1. D is looked up in the First Table; the number must be on the range 500 through 899. No digit is read out. The next argument is the next data byte.

2. C is looked up in the D_1 Table; the range must be 600 through 899. No readout. The next argument is the next data byte.

3. L is looked up in the DC_1 Table; the range is 650 through 689. Read out 6. The next argument is the next data byte.

4. X is looked up in the L_1 Table; the range of the unknown part of the number is 60 through 89. No readout. The next argument is the next data byte.

5. X is looked up in the LX_1 Table; the range is reduced to 70 through 89. No readout. The next argument is the next data byte.

6. V is looked up in the LX_2 Table; the range is now 75 through 79. Read out 7. The next argument is the next byte.

7. I is looked up in the V_1 Table; the range of the final digit is 6 through 8. No readout. The next argument is the next data byte.

8. I is looked up in the V_2 Table; the final digit is 7 or 8. No readout. The next argument is the next byte.

9. I is looked up in the V_3 Table; the final digit is 8. Read out 8 B. The next argument is the second following byte (the next byte is a B), i.e., the first byte of the next number to be transliterated, and is looked up in the First Table.

The process just described yielded the number 678 for DCLXXVIII. Only one instruction was accessed from memory (the Sequential Table Lookup) and, in fact, this single access served to transform the entire set of numbers. The process terminated when the character B was looked up in the First Table.

Clearly the decision logic for the problem is incorporated in the structure of the tables. However, in constructing these tables the programmer must concentrate on precisely this logic; most of the bookkeeping and other peripheral programming considerations are automatically taken care of. Wherever it was possible, this philosophy guided the systems planning of Harvest.

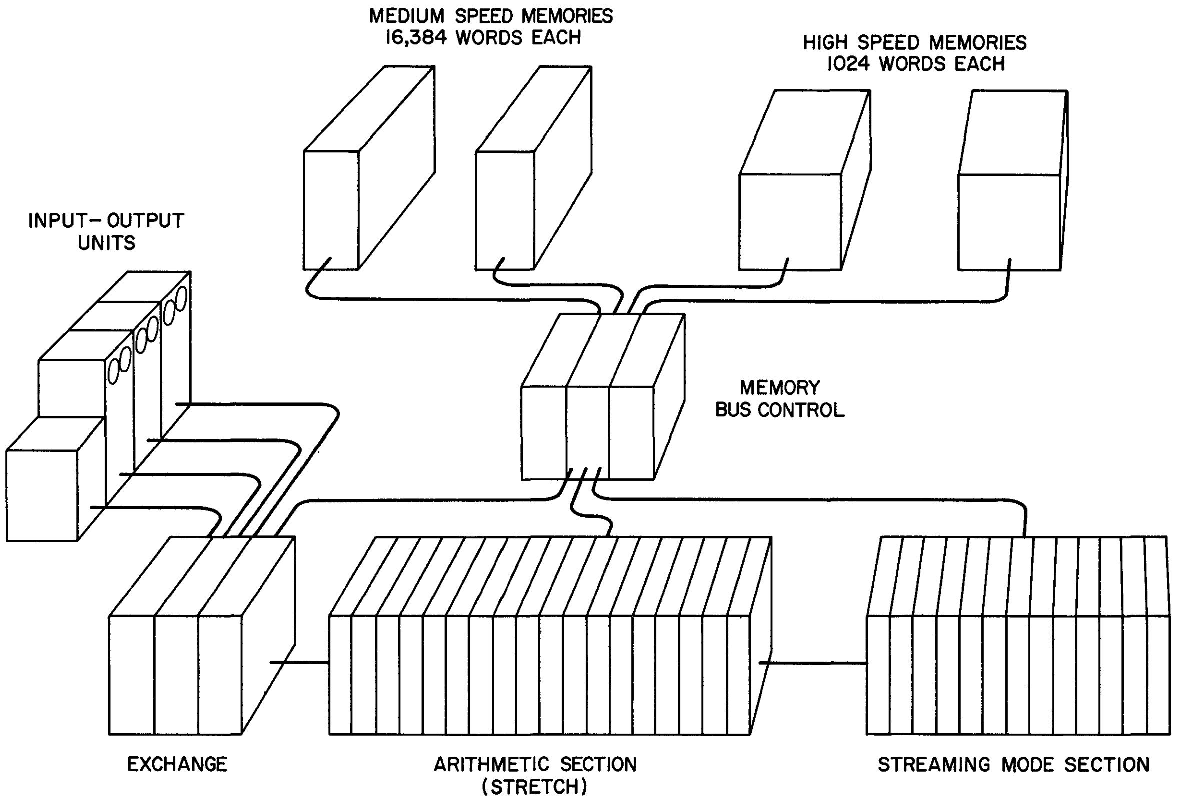


Fig. 1. The Harvest System.

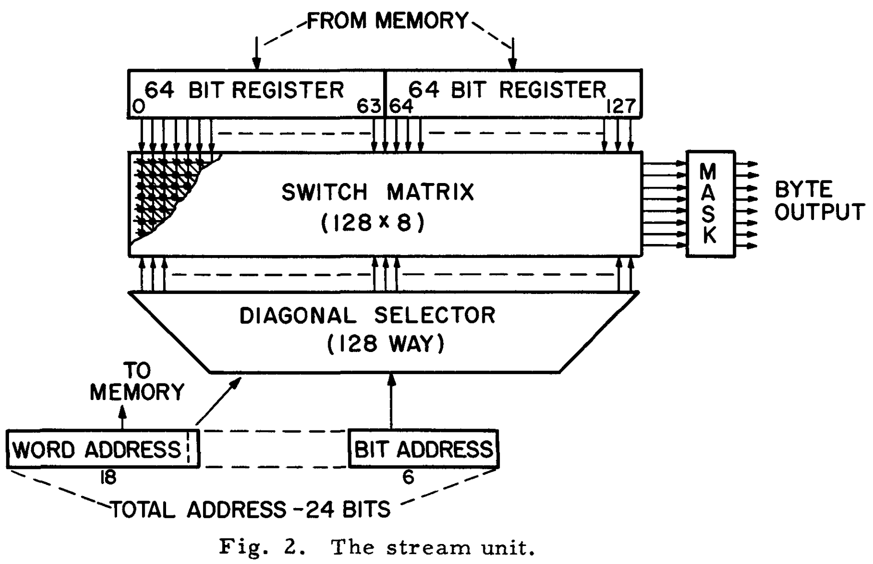


Fig. 2. The stream unit.

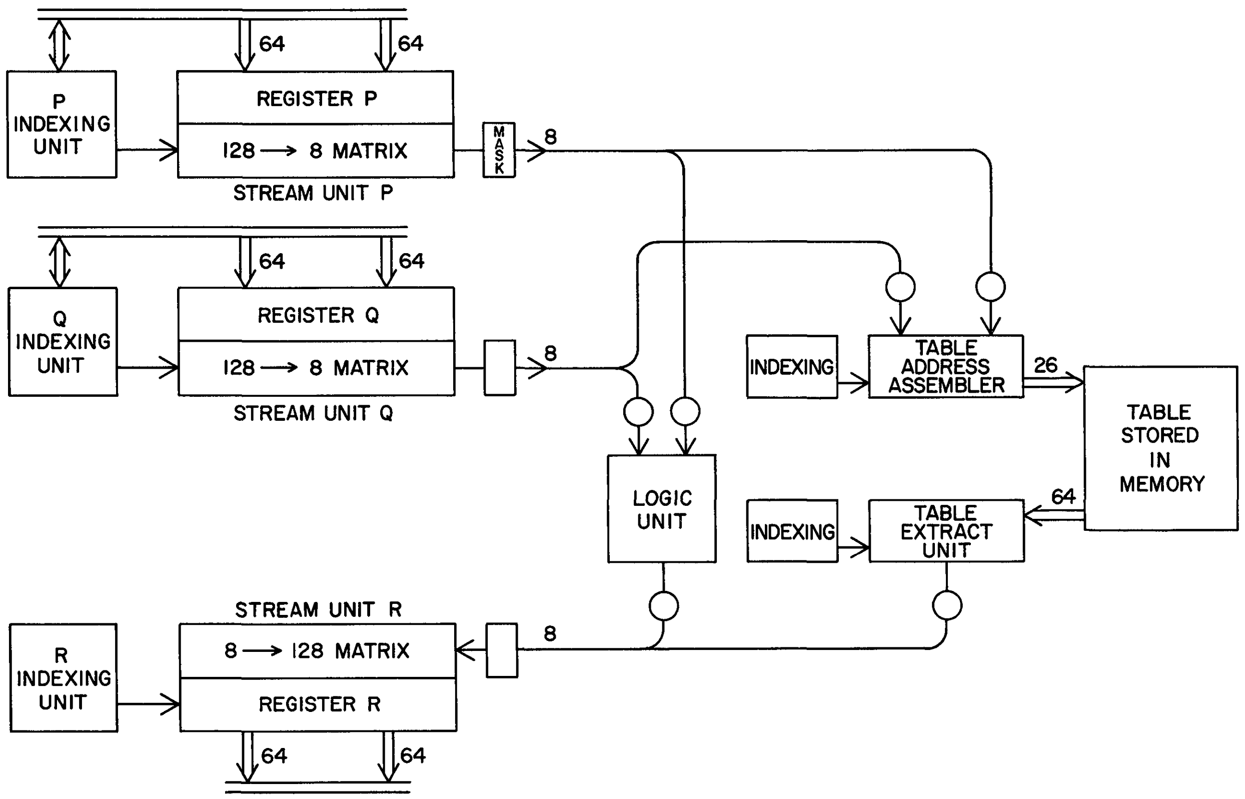


Fig. 3. Transformation facilities.

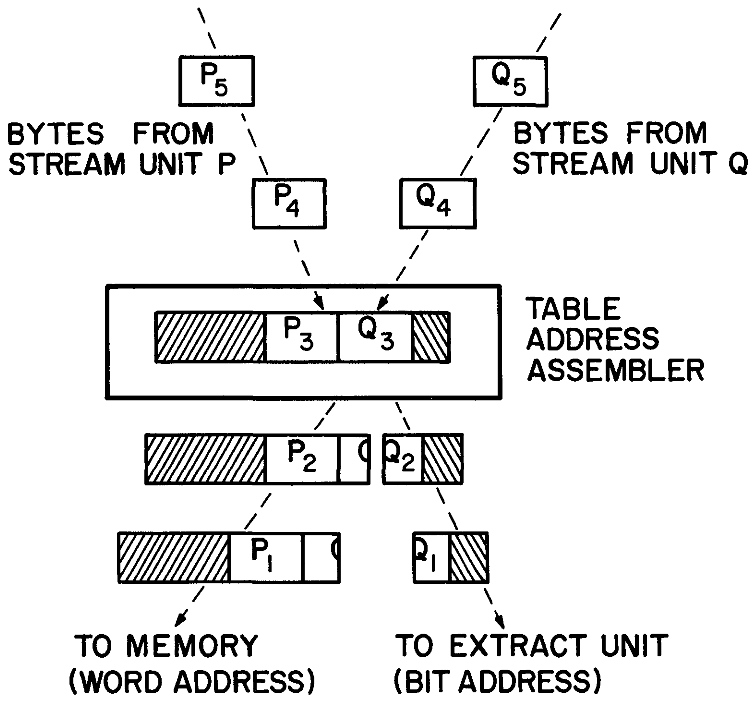


Fig. 4. Formation of lookup address.

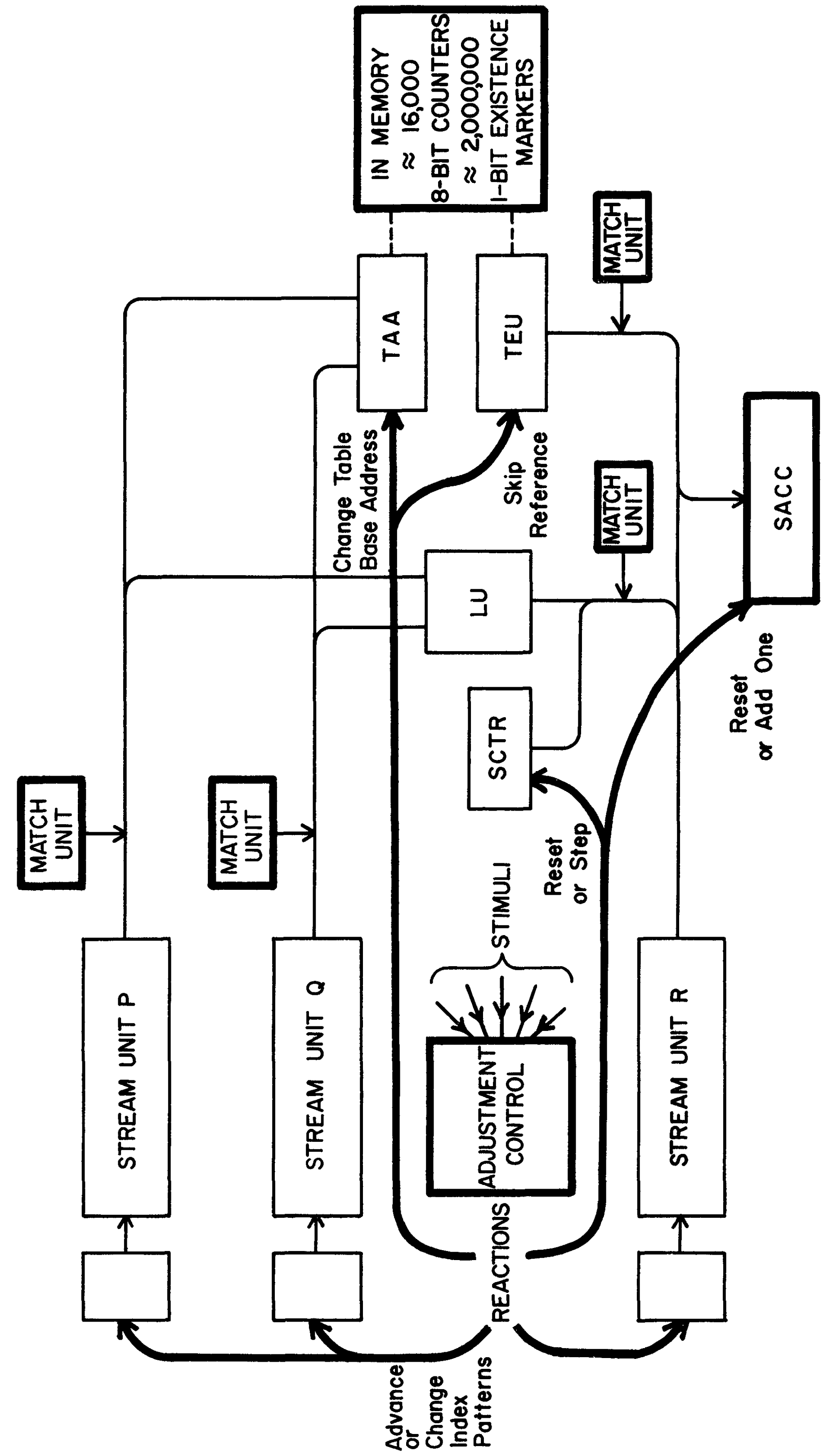
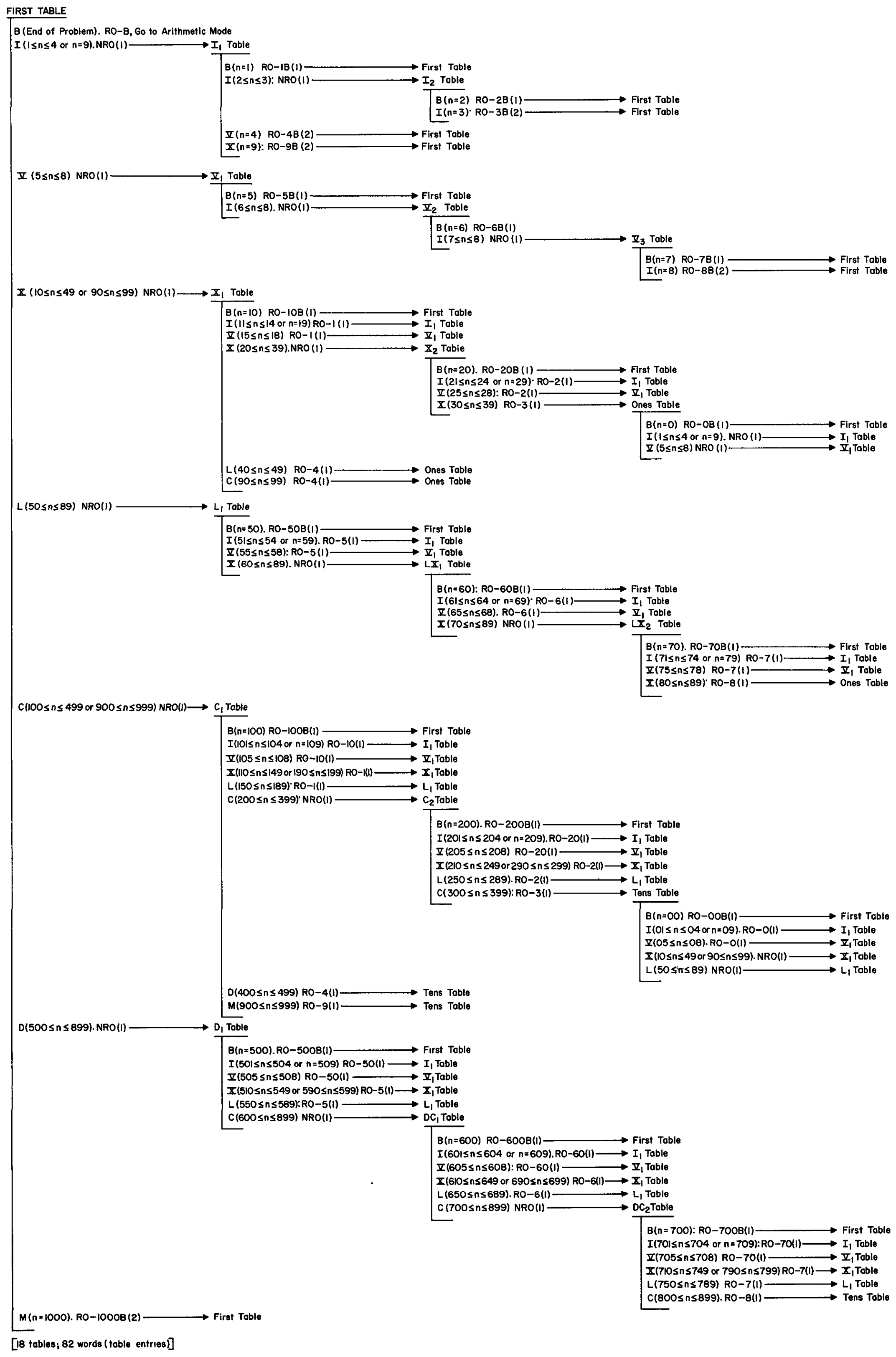


Fig. 5. Monitoring and statistical features with typical adjustment reactions.



FIRST TABLE

- B (End of Problem). RO-B, Go to Arithmetic Mode
- I ($1 \le n \le 4$ or n=9). NRO(1) → I_1 Table
  - B(n=1) RO-1B(1) → First Table
  - I($2 \le n \le 3$): NRO(1) → I_2 Table
    - B(n=2) RO-2B(1) → First Table
    - I(n=3)· RO-3B(2) → First Table
  - V(n=4) RO-4B(2) → First Table
  - X(n=9): RO-9B(2) → First Table
- V ($5 \le n \le 8$) NRO(1) → V_1 Table
  - B(n=5) RO-5B(1) → First Table
  - I($6 \le n \le 8$). NRO(1) → V_2 Table
    - B(n=6) RO-6B(1)
    - I($7 \le n \le 8$) NRO(1) → V_3 Table
      - B(n=7) RO-7B(1) → First Table
      - I(n=8) RO-8B(2) → First Table
- X ($10 \le n \le 49$ or $90 \le n \le 99$) NRO(1) → X_1 Table
  - B(n=10) RO-10B(1) → First Table
  - I($11 \le n \le 14$ or n=19) RO-1(1) → I_1 Table
  - V($15 \le n \le 18$) RO-1(1) → V_1 Table
  - X($20 \le n \le 39$). NRO(1) → X_2 Table
    - B(n=20). RO-20B(1) → First Table
    - I($21 \le n \le 24$ or n=29)· RO-2(1) → I_1 Table
    - V($25 \le n \le 28$): RO-2(1) → V_1 Table
    - X($30 \le n \le 39$) RO-3(1) → Ones Table
      - B(n=0) RO-0B(1) → First Table
      - I($1 \le n \le 4$ or n=9). NRO(1) → I_1 Table
      - V($5 \le n \le 8$) NRO(1) → V_1 Table
  - L($40 \le n \le 49$) RO-4(1) → Ones Table
  - C($90 \le n \le 99$) RO-4(1) → Ones Table
- L($50 \le n \le 89$) NRO(1) → L_1 Table
  - B(n=50). RO-50B(1) → First Table
  - I($51 \le n \le 54$ or n=59). RO-5(1) → I_1 Table
  - V($55 \le n \le 58$): RO-5(1) → V_1 Table
  - X($60 \le n \le 89$). NRO(1) → LX_1 Table
    - B(n=60): RO-60B(1) → First Table
    - I($61 \le n \le 64$ or n=69)· RO-6(1) → I_1 Table
    - V($65 \le n \le 68$). RO-6(1) → V_1 Table
    - X($70 \le n \le 89$) NRO(1) → LX_2 Table
      - B(n=70). RO-70B(1) → First Table
      - I($71 \le n \le 74$ or n=79) RO-7(1) → I_1 Table
      - V($75 \le n \le 78$) RO-7(1) → V_1 Table
      - X($80 \le n \le 89$)' RO-8(1) → Ones Table
- C($100 \le n \le 499$ or $900 \le n \le 999$) NRO(1) → C_1 Table
  - B(n=100) RO-100B(1) → First Table
  - I($101 \le n \le 104$ or n=109) RO-10(1) → I_1 Table
  - V($105 \le n \le 108$) RO-10(1) → V_1 Table
  - X($110 \le n \le 149$ or $190 \le n \le 199$) RO-1(1) → X_1 Table
  - L($150 \le n \le 189$)' RO-1(1) → L_1 Table
  - C($200 \le n \le 399$)' NRO(1) → C_2 Table
    - B(n=200). RO-200B(1) → First Table
    - I($201 \le n \le 204$ or n=209). RO-20(1) → I_1 Table
    - V($205 \le n \le 208$) RO-20(1) → V_1 Table
    - X($210 \le n \le 249$ or $290 \le n \le 299$) RO-2(1) → X_1 Table
    - L($250 \le n \le 289$). RO-2(1) → L_1 Table
    - C($300 \le n \le 399$): RO-3(1) → Tens Table
      - B(n=00) RO-00B(1) → First Table
      - I($01 \le n \le 04$ or n=09). RO-0(1) → I_1 Table
      - V($05 \le n \le 08$). RO-0(1) → V_1 Table
      - X($10 \le n \le 49$ or $90 \le n \le 99$). NRO(1) → X_1 Table
      - L($50 \le n \le 89$) NRO(1) → L_1 Table
  - D($400 \le n \le 499$) RO-4(1) → Tens Table
  - M($900 \le n \le 999$) RO-9(1) → Tens Table
- D($500 \le n \le 899$). NRO(1) → D_1 Table
  - B(n=500). RO-500B(1) → First Table
  - I($501 \le n \le 504$ or n=509) RO-50(1) → I_1 Table
  - V($505 \le n \le 508$) RO-50(1) → V_1 Table
  - X($510 \le n \le 549$ or $590 \le n \le 599$) RO-5(1) → X_1 Table
  - L($550 \le n \le 589$): RO-5(1) → L_1 Table
  - C($600 \le n \le 899$) NRO(1) → DC_1 Table
    - B(n=600) RO-600B(1) → First Table
    - I($601 \le n \le 604$ or n=609). RO-60(1) → I_1 Table
    - V($605 \le n \le 608$): RO-60(1) → V_1 Table
    - X($610 \le n \le 649$ or $690 \le n \le 699$) RO-6(1) → X_1 Table
    - L($650 \le n \le 689$). RO-6(1) → L_1 Table
    - C($700 \le n \le 899$) NRO(1) → DC_2 Table
      - B(n=700): RO-700B(1) → First Table
      - I($701 \le n \le 704$ or n=709): RO-70(1) → I_1 Table
      - V($705 \le n \le 708$) RO-70(1) → V_1 Table
      - X($710 \le n \le 749$ or $790 \le n \le 799$) RO-7(1) → X_1 Table
      - L($750 \le n \le 789$) RO-7(1) → L_1 Table
      - C($800 \le n \le 899$). RO-8(1) → Tens Table
- M(n=1000). RO-1000B(2) → First Table

[18 tables; 82 words (table entries)]

Fig. 6.