A SEARCH MEMORY SUBSYSTEM FOR A GENERAL-PURPOSE COMPUTER

Albert Kaplan
UNIVAC Division
Sperry Rand Corporation
St. Paul, Minnesota

I. INTRODUCTION

The search memory provides an extremely powerful tool for data-processing applications in which files of data are manipulated—applications such as data correlation, information retrieval and data analysis. For many such problems, a moderate size search memory can increase the performance of a general-purpose computer by several orders of magnitude. This paper describes the system design of a search memory subsystem integral to a general-purpose computer. The unit will operate in much the same manner as a conventional arithmetic unit, receiving data directly from the main memory, and placing the results of a search in the accumulator. This generalized design is independent of the particular magnetic memory element used and is not constrained by any particular computer organization. However, values have been assigned to the various parameters in the system so that the concepts can be concretely described. The techniques and concepts presented in this paper are the result of a company-sponsored program. The following sections present an assumed general-purpose computer structure, the design of a search memory subsystem, and possible implementation. Appendix A presents the basic principles of the search memory organization.

II. SEARCH MEMORY SUBSYSTEM

To establish a sound basis for the design of a search memory subsystem, a number of typical applications were analyzed to determine the functional requirements which such applications impose upon the search memory. The primary application was for an information retrieval process suitable for a library, or an inventory control system. Major requirements for such an application are equally search, variable "AND"—"OR"—"NOT" logic between field searches, and the ability to analyze the results of partial searches so that search criteria can be "loosened" or "tightened" according to the number of matches. Data correlation applications require quantitative searches such as *greater than*, *less than*, and *between limits*. Finally, statistical analysis of data sets requires *next higher* and *next lower* as well as the other quantitative criteria. In addition, the *number of matches* is required. The composite of these requirements provided the basis for this subsystem.

In order to keep the design of the search memory subsystem as generally applicable as possible, the less assumed about the computer structure, the better. Thus, the structure shown in Figure 1 is postulated, consisting of a main computer memory which communicates via a memory register with the search memory subsystem, an accumulator, the arithmetic, control, and I/O sections. In concrete terms, a word length of 36 bits and a main memory cycle time of 4 microseconds is also postulated.

The search memory subsystem contains 4096 words of 72 data bits each. Searches are carried

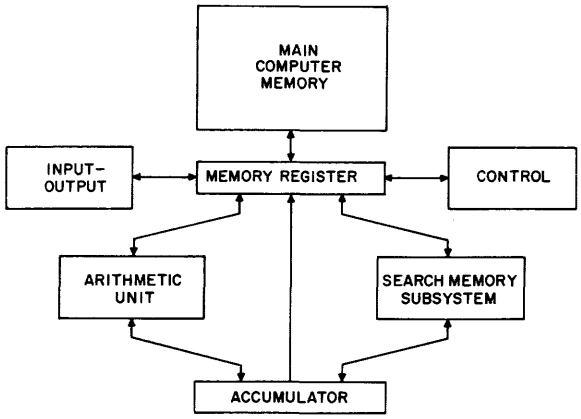

Figure 1. Postulated Computer Structure.

out on 36 bits, and the remaining 36 are entered into the A register on a match condition. The following search criteria are provided:

Equal
Greater than or equal
Less than or equal
Between limits
Next higher
Next lower
Not equal
Not greater than or equal
Not less than or equal
Not between limits

A completely variable field structure is provided so that multiple fields of variable lengths and positions within the search word can be specified under program control. A different search criterion can be applied to each field specified and the results of each field interrogation "AND"ed or "OR"ed with the cumulative result of previous interrogations. Similarly, the results of several full word searches can be "AND"ed or "OR"ed. Execution time of the search operation varies with the number of bits searched, the number of fields, and the search criteria. For a simple operation, such as a 36-bit equality search, the execution time is 8.0 microseconds.* A very complex six-field, six-bits-per-field quantitative search will require 16.0 microseconds. An average search will require 12 microseconds. An interrupted search mode is provided for interrupting the search after a specified field has been interrogated and for entering the approximate number of matches into the A register. Under program control, the search may be either resumed or terminated, in which case the match data is retrieved as in the normal mode. The exact number of matches may be requested under program control. In multiple matches, the first match data word is automatically placed in the A register at the completion of the search. Remaining match data words are requested sequentially under program control. Three modes of loading the search memory are provided. Data may be written into either or both halves of the 72-bit word at a location specified by the program, or at the location of the last match. The third mode facilitates sequential loading of the entire search memory as would be required at the initiation of the program. Figure 2 shows a block diagram of the search memory subsystem.

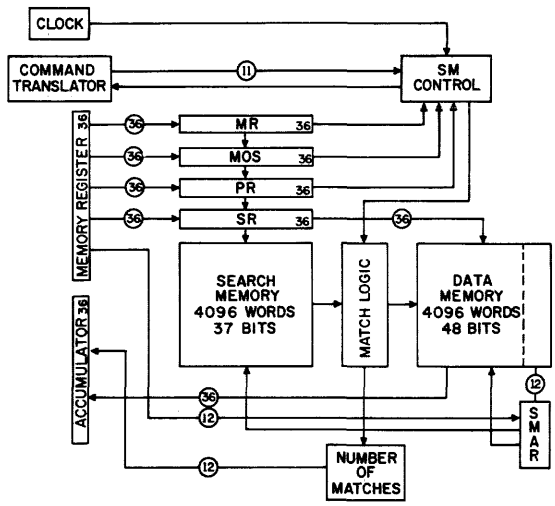

Figure 2. Search Memory Subsystem.

Major components of the search memory subsystem are the search memory, with a postulated size of 4096 words, 37 bits per word (36 bits hold search data, and the remaining bit is used as a dummy bit for the "NOT" searches); a 4096-word, 48 bits per word, word-organized data memory (36 bits hold match data, and the remaining 12 hold the address of that word); a search register to hold the search word; three control registers, PR, MOS, and MR; the match logic which interprets the signals emanating

* The operation times quoted are for the UNIVAC Bicore Non-Destructive Readout thin film memory element. They will change according to the memory element used.

from the search memory; an address register used for writing information in memory; a number-of-matches unit which produces both a high-speed approximate count and a slow-speed exact count; and control and timing circuitry.

Four data registers are used with the search memory. Search register (SR) holds the 36-bit search word. Parameter register (PR) is used for variable length field designation. Coding of PR is done according to the following rules. The first bit of each field is set to a "1", and all successive bits within the field are set to "0".

The sample PR code below would indicate eight search fields of various lengths.

10000000	1000000	100000	100	10	1	100	100000
Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8

A 36-bit mask register (MR) enables masking within specified fields. A "1" in MR enables the search of the corresponding bit in SR, a "0" masks that bit.

The mode-of-search register (MOS) is used to designate the search criteria for the various fields specified in the parameter register. This register holds up to six mode statements, of six bits each. Each mode statement specifies the search criteria for one of the search fields and is composed of four quantities. The first bit designates the logical connective to the previous searches, "AND" or "OR". The next three bits specify the search criteria: *equality, greater than* or *equal, less than* or *equal, between limits, next higher,* or *next lower.* The next bit specifies the negative of the search criteria. The last bit designates that the search is to be interrupted after that field has been interrogated and the approximate number of matches entered in the accumulator. This feature cannot be applied to the last field. The first bit of MOS is used to inhibit the clearing of the match logic so that another search can be performed and their results combined. If more than six fields have been specified in the parameter register, the last mode statement is applied to all successive fields.

Instructions to be used with the search memory are listed below; (M) represents the contents of a main memory location specified in the instruction:

Mnemonic	*Function*	*Time*
LPR	(M) → PR	4 μsec
LMS	(M) → MOS	4 μsec
LMR	(M) → MR	4 μsec
SRC	(M) → SR, search, first match data → A; clear load designator; if no match, skip next instruction; SMAR holds match address	(See below)
NMD	Next match data → A; if no more, skip next instruction; SMAR holds match address; if interrupted search, terminate search and first match data → A	4 μsec
NMA	Exact number of matches → A	4 μsec
SMA	(M) → SMAR	4 μsec
SLD	Set load designator, clear SMAR	4 μsec
WSM	(M) → SR (SR) → SM(SMAR)	16-32 μsec (See below)
WDM	(M) → SR (SR) → DM(SMAR), (SMAR) + 1 → SMAR if load designator is set	4 μsec
RIS	Resume interrupted search	(See below)

The first three instructions, LPR, LMS, and LMR, are used to enter words from main memory into the search memory control registers. Once they have been set up, an SCR instruction loads the search register and initiates the search according to the instructions in PR, MR, and MOS. Upon completion of the search, a flip-flop associated with each word will be set either to match or mismatch. These flip-flops are scanned, and the first match is located. The corresponding location in the word-organized data memory is driven to obtain the 36-bit match data and the 12-bit address of the match. The address is placed in SMAR, and the match data placed in the accumulator. If there are no matches, the next instruction is skipped. The search memory address register (SMAR) holds the match address so that new information can be written into the search memory or the data memory at this location. Next, the NMD instruction is applied repeatedly to retrieve all the matches for this search. Normally, this would be done in a loop with the skip next instruction feature used to exit from the loop when all the matches have been processed. If an interrupted search has been specified, NMD is used to terminate the search and retrieve the match data. If the program elects to continue the interrupted search, the RIS instruction is used.

The time required to execute a search operation is a function of the number of bits searched, the mode of search, and the number of fields. Total search time is 0.1 microsecond per bit searched for *equality, greater than* or *less than,* plus 1.1 microseconds per bit searched for *next higher* or *next lower,* plus 1.0 microsecond per field searched (except first field), plus 4.0 microseconds to get the next instruction. The resulting time must then be rounded to the next higher multiple of 0.4 microseconds to regain synchronization with the control section. Thus, for a 36-bit equality search, the time is 8.0 microseconds. For a fairly complex six-field, six-bit-per-field, greater or less than search, the search time is 16.0 microseconds. When an interrupted search is specified, the above formula will apply for the portion of the word searched plus 24 microseconds to generate the "approximate" number of matches. A search can be initiated 16 microseconds after a write. However, 32 microseconds are required before another write. These times apply only to the Bicore thin film memory element.

In addition to the approximate number of matches generated in an interrupted search, an exact match counter is provided for statistical data analysis. The approximate number (10%) is generated in 24 microseconds with an analog summer and A/D converter. Every word in the search memory must be scanned for an exact count which requires 410 microseconds. This time can be reduced by scanning several words at a time. The NMA instruction is used to place the exact number of matches in the accumulator. The remaining instructions are used for writing in the search memory and data memory. The SMA instruction places an address in SMAR. The WSM and WDM copy a word from main memory into the search memory or the data memory via the search register at a location specified by SMAR. A block loading feature provides for rapid loading of the search memory and data memory at the start of the program. The SLD instruction clears SMAR to location zero and sets a special load designator. A word is then written into location zero of the search memory. Then a WDM instruction loads location zero in the data memory. However, since the load designator is set, SMAR is automatically incremented and the process can be repeated to load location one, two, three, etc.. On the next search instruction the load designator is automatically cleared.

For each word in the search memory, final determination whether that word is a match or not is performed in the match logic. Match determination is performed in the following manner.

The memory is driven bit-serially, starting from the highest order unmasked bit location. Each bit-line is driven to either a "1" or a "0" corresponding to the contents of the search register at that particular bit location. If the bit in a particular word is already in the state to which it is being driven, there will be no output. Consequently, that bit will be known to be equal to the corresponding bit in the search register. If that bit is in opposition to the state to which it is being driven, there will be an output on the word line, the polarity of which depends upon the stored bit (see Appendix A). The outputs on the word line indicate

that the bit in memory does not compare with the corresponding bit in the search register. Two inputs can be provided to the match logic by controlling time and duration of bit drive on each bit line, and by inverting and time-gating the word line outputs onto separate lines from the sense amplifier according to the mode of search indicated in the MOS register. An output on one line indicates that search criterion ($=$, $\geq$, $\leq$) is satisfied for the particular bit. An output on the other line indicates that the mismatch was such that the search criterion failed at that bit location. Absence of an output from the sense amplifier indicates that the bit memory is the same as the bit in the search register, a condition which indicates neither passing nor failing of the search criterion.

The match logic accepts the above outputs from the sense amplifier, mode of search information from the MOS register, timing information from the memory driver control circuits, and performs two basic functions:

1) Inhibiting the word line noise brought about by activation and deactivation of the digit drivers.
2) Determining field comparison or non-comparison according to the following rules:

   Determination of field comparison (i.e., compliance with the search criterion) can be made upon detection of the first bit found to differ from the corresponding SR bit. (Each bit of a field bears greater numerical significance than the sum of the succeeding lower order bits.)

   If no bits in the field are found to differ from their corresponding bits in SR, the field has complied with the search criterion.

A schematic of the match logic is shown in Figure 3. To simplify description, the diagram shows noninverting logic. The sense amplifier accepts signals from the sense line and, on the basis of control information from the MOS register, issues a pulse on either the pass or the fail lines. If, for example, the MOS register specifies greater than or equal (GT) and the polarity of the signal on the sense line indicates that the SR bit being interrogated is a "0", a

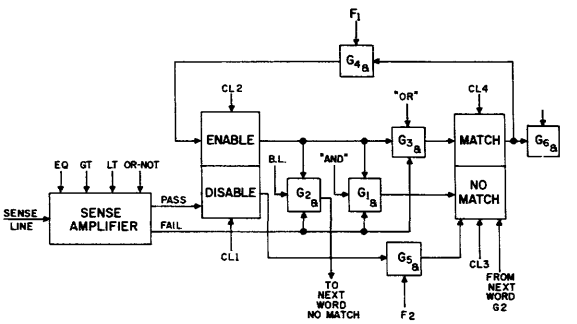

Figure 3. Match Logic.

mismatch signal on the sense line means that the corresponding bit in the search memory word is a "1" and a pulse is issued on the pass line. The match logic must then accept the first signal from the sense amplifier for each field and block all others. To trace the operation of the match logic, consider an equality, greater than or equal, or less than or equal mode of search and an "AND" coupling to the previous interrogations. The match flip-flop contains the cumulative results of the previous interrogations, and the enable flip-flop is set to enable at the start of the field. The field is interrogated bit-serially, starting with the highest order unmasked bit. As long as the bits of the field in this word are identical to those in SR, there will be no signal from the sense amplifier and the match logic will remain quiescent. When the first mismatch is found, a signal will be issued on either the pass or the fail line. If the signal was on the fail line, it will pass through G_1, which is enabled, and set the match flip-flop to no match. Since any successive signals from this field cannot change this no-match state, they are automatically ignored. If, however, the signal was on the pass line, it sets the disable state, and G_1 is blocked. Thus, any successive signals on the fail line cannot get past G_1, and the state of the match flip-flop remains as it was before this field was interrogated. This satisfies the "AND" condition. To accomplish the between-limits, the odd numbered words in the search memory are driven for greater than or equal and the even numbered words for less than or equal. The no-match signals in each pair of words are cross-gated via G_2 so that a no-match on either will result in no-match for both. Only a match on both words represents a match on between-limits.

To apply the "NOT" condition to these three search criteria, the roles of the pass and the fail lines are interchanged by reversing the direction of the bit drives and thus the polarities of the signals on the sense line. A pass signal will emerge on the fail line and vice versa. Thus, with the same logic as above, a pass signal from the memory will result in setting the match flip-flop to no-match, and a fail signal will leave the match flip-flop unchanged.

This process breaks down, however, when the search memory word is identical to the search register and no signal appears on either the pass or the fail line. To overcome this, one extra bit is added to each word which is so set that when interrogated, a signal always results on the fail line.

The "OR" function is accomplished by the equivalence of "OR" to "AND-NOT". As in "NOT", the signals on the pass and fail lines are interchanged. G_3 is activated instead of G_1 so that a pass signal, if it is the first signal in the field, will set match and fail will leave the match flip-flop unchanged.

The next higher and next lower searches are the most complex and require two passes through the field. The first pass is a conventional greater-than or less-than search. After this pass is completed, all enable flip-flops are disabled and then G_4 is pulsed to enable only those which were set to match. Then the field is searched again. After each bit is interrogated, a 4096-input "OR" gate (not shown) examines the state of all enable flip-flops. If at least one is still enabled, the disable side sets the no-match state via G_5. If all are disabled, the match state again resets the enable via G_4, *and also* the field in the search register is reset to all "0"s for next-higher or to all "1"s for next-lower. To illustrate this process, consider the following next-higher search. Assume that the first pass has been completed. The states of the enable (E) and match (M) flip-flops are shown after each bit is searched and after resetting with G_4 and G_5.

	↗00
Search Register	11011
Word 1	11100
Word 2	11101
Word 3	11110

	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
	EM EM	EM EM	EM EM	EM EM	EM EM
Word 1	11 → 11	11 → 11	G_4 01 → 11	11 → 11	11 → 11
Word 2	11 → 11	11 → 11	G_4 01 → 11	11 → 11	G_5 01 → 00
Word 3	11 → 11	11 → 11	G_4 01 → 11	G_5 01 → 00	00 → 00

E represents the state of the enable flip-flop and M the state of the match flip-flop.

III. IMPLEMENTATION

The system design presented above can be implemented with any of the present magnetic nondestructive readout memory elements with minor variations for the electrical characteristics of the particular element. There is no inherent limitation to the size or word length of the subsystem. If the word length of the search memory is greater than that of the main memory, extra instructions can be used to load portions of the various registers. For example, if the search memory were twice as long as the main memory, two instructions would be required to load each register, one for the left half and one for the right half. The data memory need not be the same size as the search memory; it could contain several words for each search memory word. Similarly, the vari-

ous functions and components within the subsystem can be varied to suit the particular requirements.

Considerable logic circuitry is required for this subsystem (Figure 3). With conventional circuitry, a subsystem of this magnitude would be very unwieldy. However, the fallout technology from aerospace computer developments will enable such a subsystem to be built in a moderate size and at a moderate cost. Microelectronic circuitry, besides having such virtues as high reliability, high environmental tolerance, low power and relatively high speed, is also quite economical when procured in large quantities. A typical package, containing an average of 2.5 logic nodes, measures $1/4'' \times 1/8'' \times 0.035''$ and consumes 12 mw of power. Using such elements with a Bicore thin film search memory, and a state-of-the-art word-organized core data memory, the subsystem described above would occupy a unit 2 by 2 by 2.5 feet for a volume of 10 cubic feet.

IV. CONCLUSION

The system design for a search memory subsystem for a general-purpose computer has been presented. The design is as general as possible so as to be independent of memory element and computer structure. This design encompasses many of the functions required for such applications as information retrieval, data correlation, and statistical data analysis, and can increase the capability of the computer for these types of problems by several orders of magnitude.

APPENDIX A

Search Memory Organization

A search memory is a device which simultaneously compares an input word with all words stored in the memory and indicates whether each word satisfies the search criterion. Among the criteria which can be readily implemented are equality, greater than or equal, less than or equal, between limits, next higher, next lower, or the negative of these criteria. The time for a search varies with memory element and organization and ranges upward from 100 nsec.

Several possible organizations for the search memory include searching all bits in parallel, searching bit serial, and various series-parallel modes. The organization which appears to be optimal for this system is the bit serial mode. The reasons for this choice will be developed below. In this organization (Figure A-1) the circles represent nondestructive readout memory elements whose property is essential to the operation of the search memory since all memory elements are interrogated during the search. The rows are words and the columns bit positions. A bi-directional driver is provided for each bit position which is controlled by the contents of the search register. A "1" in the search register results in a positive current pulse on the bit line, a "0" in a negative pulse. Each word has a sense line, with the sense line and memory elements so coupled that a positive current pulse (a "1" in the search register) and a stored "1" will result in a very small output on the sense line. A positive current pulse and a stored "0" result in a large positive signal on the sense line. Similarly a "0" in the search register produces a negative bit current pulse which results in a very small signal for a stored "0" and a large negative signal for a stored "1".

The relationship of the stored information, bit current polarity, and output signal polarity are tabulated in Table A-I below.

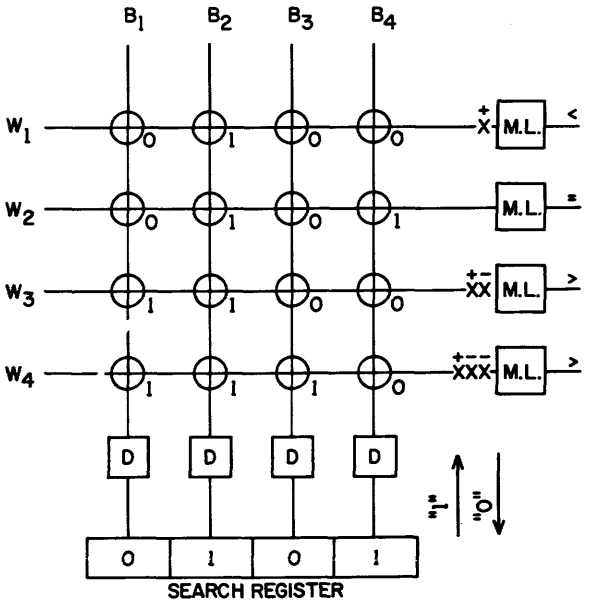

Figure A–1. Search Memory Organization.

TABLE A-I

Stored Information	Search Register	Bit Current Polarity	Output Signal	
			Switch	Reswitch
1	1	+	0	0
1	0	—	— V	+ V
0	1	+	+ V	— V
0	0	—	0	0

In the sample pattern shown in Figure A-1 the search register contains 0101 which matches word two in the memory. Bit one of the search register is a "0" and produces a negative current pulse on the bit one line. Words three and four contain "1"s in that bit position and signals result on their sense lines. These are denoted by X's. Bit two of the search register is a "1" and produces a positive current pulse on the bit line. Since all four words have a "1" in that bit position, no outputs are produced on the sense lines. Bit three of the search register is a "0" and produces a second output on the word four sense line. Finally, bit four of the search register is a "1" and the positive current pulse produces an output on the sense lines of words one, three, and four. Notice that word two, which matches the search register, has had no outputs on its sense line while the others have had at least one signal. Thus, the matching word can be detected.

To perform a greater than or less than search, the bit serial mode of operation is essential. The basic algorithm is that the highest order bit in which the word in the search memory differs from the search register determines whether that word is greater or less than the contents of the search register. If this bit in the search register is a "1", then the word is less. If it is a "0", then the word is greater than the search memory. This algorithm is easily implemented with this organization because of the bipolar output from the memory element as shown in Table A-I. The memory is interrogated bit-serially starting with the highest order bit, and match logic is implemented on each sense line to interpret the first signal and to ignore all successive signals. With the same example as in Figure A-1, bit one of the search register is a "0", and the negative current pulse produces a negative signal on the sense line of words three and four. This signal is interpreted by the match logic to mean greater than, and all further outputs from words three and four are ignored. Bit two produces no outputs. Bit three produces a negative output on the word four sense line, but this is ignored. Finally, bit four produces positive outputs for words one, three, and four. The signals on the words three and four are ignored as before. However, the positive signal on the word one sense line is intepreted as less than, and word one is also set to ignore all successive signals. Thus in one search, all words in the memory have been quantitatively compared with the word in the search memory. A masking function can be very easily implemented by inhibiting the drivers. With no pulse on the bit line, no signals can result on the sense line, and the bit is effectively masked.

Since a bit-serial mode of operation is required for the quantitative comparisons, this same mode is used for equality. The equality search can be performed on a fully bit-parallel basis. However, this would complicate the control and would require a much more sophisticated sensing mechanism because of lower signal-to-noise ratios.