ASSOCIATIVE SELF-SORTING MEMORY

Robert R. Seeber, Jr.
Product Development Laboratory, Data Systems Division
International Business Machines Corporation
Poughkeepsie, New York

Summary

A cryogenic associative memory is proposed in which the status of a memory word is determined, with respect to an interrogating word, on a high, low or equal basis. Thus the bracketing pair of words is determined, allowing the interrogating word to be inserted between them, a "dummy" register holding it temporarily. A double-shuffle operation moves the other words to make room for the new word in proper sequence.

Introduction

One of the major problems that occurs in the use of a computer, particularly in business uses, lies in the sorting of data. The importance of this problem can be judged on the basis of the hundreds of pages of description of sorting routines that have been written. Programs are in some cases many thousands of instructions in length. Heretofore, the attack on the sorting problem has been almost entirely on the programming level. In this paper is proposed a memory system organization to achieve sorting within the memory. By the use of associative memory principles, extended to facilitate sorting, a new approach to sorting is developed. Words to be sorted enter memory in random order but each is placed within memory in its proper relationship to previously sorted words.

Associative Memory

Associative memory may be defined as memory in which a word of data is retrieved on the basis of part or all of the data content of the word. Words are stored in vacant registers and subsequently are recovered not by naming the location of a register but rather by naming a portion of the word as an identifier. This identifier or tag may in some cases be a portion added to the data word for the identification purpose. However, in the more general case, which we call a fully associative memory, the data word can be retrieved by using selected portions of the word itself as its identifier; in this case, the remaining portions are masked out. Thus the knowledge of the exact storage location in the physical sense may be completely immaterial to the operation. A more detailed explanation of a simple associative memory was given in an earlier paper.1

Sorting with an Associative Memory

The use of an associative memory reduces somewhat the need for sorting, particularly in the case where sorting is used to provide means for locating particular words by their ordered identifier or argument, as is usual in table look-up operations. Since an associative memory allows direct access by these identifiers, sorting for these purposes is not necessary. But for other purposes, particularly for sorting output lists, sorting is required. This may be done by an associative memory by arranging to retrieve in order all possible combinations of the given identifier. For example, if we have a 3-decimal digit identifier and want to retrieve 965 different data words identified by this 3-digit code, we can set up a counter to provide us with all combinations running from 000 through 999 to act as identifiers for the 965 words. The efficiency will be quite high, requiring 1000 retrieval tries to produce the actual 965 retrievals in order. However, this system breaks down in the more frequent case of the identifier not so densely coded. For example, a 10-decimal digit part number code may have only a few thousand different parts which would require 1010 retrieval tries to recover in order. Since sparsely populated codes seem to be the more general rule, particularly in business problems, a means for actually sorting the words would be very desirable.

Proposed System

In a general associative memory a simultaneous comparison is made between an entry word and all of the word registers in the memory to determine the match or non-match status of each word. This is done by providing comparison circuits between the entry register and each of the word registers such that an equal or unequal status is determined. This has previously been done for the purpose of retrieving a matched word from memory. For
writing into memory in order, we now add the requirement that these comparison circuits be extended to provide a comparison indication on high, low or equal for each of the word registers. This then will supply us with enough information so that we can determine where within a previously sorted sequence a new word should be inserted. Additional registers called dummy registers are supplied between each of the word registers. This allows an incoming word to be placed between the proper two words in memory; then, by a double-shuffle transfer cycle, the words in memory are shuffled to enter the new word into its proper place.

Figure 1 is a block diagram showing this arrangement. At the top there is an entry register where data words are coming in from some other portion of the computing system. Comparison circuits from this entry register extend through all of the word registers and each of the word registers supplies indication as to whether it is low, high or equal to the word in the entry register. From this information, the dummy register lying between the two word registers which bracket the word in the entry register can be selected and the word then can be transferred from the entry register to that dummy register. On this same half cycle, all the words in word registers above the selected dummy register move up to the respective dummy registers immediately above each of those word registers. On the next half cycle, while a new word is entering the entry register, the words in dummy registers move up to word registers immediately above those dummy registers, thus leaving the words in memory again in word registers in proper order including that word just entered.

Means are provided for exiting one word from dummy register 1, that is, the forward exit register, as the memory fills up. That is, as the memory fills up, earlier words are pushed out the top. In this case, the echo register retains the information of the word just exited.

Another mode of operation is provided so that words coming in through the entry register may be accepted in inversely sorted order. In this case, as the registers fill up, words are pushed out the bottom through the backward exit register; this inverse function may be useful in handling partly sorted blocks of words read backwards from a tape on which data has first been stored in the normal forward order. This operation may reduce the need for re-winding operations when tapes are used in conjunction with this sorting memory.

**Sorting Example**

The chart of Figure 2 shows the successive cycles of the operation of sorting 21 different words of which the sorting identifiers are shown on the entry line. It is assumed that the memory for this example has 5 word registers, W1 through W5, and 6 dummy registers, D1 through D6. The two-digit identifier has a bar over it when the word has just arrived at the location so indicated. A code with no bar indicates that the word is still present at that location but has been moved elsewhere, i.e., this is a "shadow" word. Where a bar appears under an identifier, this indicates that the word has not been moved on that cycle but has been previously moved to that location.

Thus, it may be seen in Figure 2 that the number 29 is placed in the entry register during the A portion of cycle 1. During the B portion of cycle 1, the number 29 is transferred to dummy register W + 1. During the A portion of cycle 2, the number 1 is placed in the entry register and the number 29 is transferred from dummy register W + 1 into word register W, where in this instance of illustration, W equals 5. During the B part of the second cycle, the 1 is transferred into dummy register W where W is equal to 5 in this instance, and the number 29 is transferred from dummy register 6 into word register 5. Since the number 1 in the entry register was less than the number 29, it will be observed that the number 1 was placed nearer the top of the column of registers. During the A portion of cycle 3, the number 44 is placed in the entry register, the number 1 is transferred from dummy register 5 to word register 4, and the number 29 remains in word register 5, the image being in dummy register 6. During the B portion of cycle 3, the number 44 is moved into dummy register 6 replacing the image 29, the number 29 is moved into dummy register 5 and the number 1 is moved from word register 4 into dummy register 4. The first three cycles have illustrated the cases where the number in the entry register was less than the numbers stored in memory and the other instance where the word in the entry register was greater than those stored in memory. The next example of cycle 4 concerns merging a number among those previously stored.

In the A part of cycle 4, the number 10 is placed in the entry register, 1 is moved to word register 3, 29 is moved to word register 4, and 44 is moved to word register 5, thus
making available the vacated dummy registers. During the B part of cycle 4, the number 10 is inserted in dummy register 4 and the number 1 is moved from word register 3 into dummy register 3. In this manner, a word is merged with those in memory.

Cycle 7 illustrates the overflow of a word to the output bus; cycle 11 illustrates the start of another block of words at the end of memory.

Cryotron Circuits

When cryotron circuits are available, they would appear to have ideal properties for such a memory. It should be noted that in the self-sorting memory, as in other associative memories, there is a great deal of distributed logic. The cryotron is a single element which can be used for both storage and logical purposes. An implementation of the self-sorting memory in single-crossing, thin-film cryotrons is proposed. Cryotron circuits have previously been described by Dudley Buck. 2 Here we use a simplified symbolism, a gate being shown by a semi-circle with its diameter lying along the gate line and the corresponding control wire at right angles to the gate wire and bisecting the semi-circle. Figure 3 is the configuration for a flip-flop with read-in and read-out circuits employing this symbol.

The top current source (denoted by a '1') splits into two paths, only one of which is superconductive at a time. If the left path is conducting, the flip-flop is said to be "on" or contain a "1"; if the right path is conducting, the flip-flop is "off" or contains a "0." The feedback action of the flip-flop is accomplished by the top or bottom cryotron, depending on whether the flip-flop contains a "0" or a "1." If it contains a "1," the left path of the flip-flop is conducting. This current through the lower cryotron control makes the right path resistive and keeps the current flowing in the left path. Similar action takes place at the top cryotron when the right-hand path is conducting.

Assume that we have a "0" in the flip-flop, and we want to change its state, that is, read in a "1." We cause a current flow through the control path of the "read-in 1" cryotron, making that cryotron resistive. Since both the right and left paths of the flip-flop are now resistive, the current divides in half. When the current through each path falls near the half point, neither the upper nor the lower feedback cryotron is resistive. This leaves the "read-in 1" cryotron as the only resistive element, forcing all the current to flow through the left-hand path and making the lower cryotron resistive. The flip-flop has now reached a stable state in the "1" or "on" condition, and the "read-in 1" current can be removed. Similarly, we can change back to an "off" condition by applying a current through the "read-in 0" cryotron.

The read-out is accomplished by completing a circuit from the read-out current source through one of the read-out cryotrons and to the output device. If we assume that the flip-flop is in the "0" state, current will be flowing through the right path and through the control path of the "read-out 1" cryotron, making that cryotron resistive. Current then flows through the superconducting "read-out 0" cryotron gate to the output device, where a "0" will be sensed. The circuit is similar through the "read-out 1" cryotron when the flip-flop contains a "1."

Cryotron Bit Position

The heart of the memory system lies in the data bit position shown in Figure 4. In this figure a portion of a data bit for word register W-1 is shown near the top of the figure and the complementary portion for word register W is shown near the bottom of the figure. Between is the flip-flop for the dummy register lying between these two word registers. At the top of the drawing there are provided read-out and read-in circuits for going from or to that word register to or from the dummy register immediately below. In the dummy register flip-flops, there are shown corresponding read-in and read-out circuits for transfer between the dummy register and the word register above or below it. There is also shown the select circuit which provides for the storing in this dummy register of a data word coming from the entry register.

For word register W there are shown entry and exit transfer circuits for coming from or going to the dummy register above it. Next is a match circuit which controls the exit of data from this word register when it is a matched register for the read-out of data in the normal associative manner. The last lines on the figure show the equal, low and high matching circuit for determining the selection of the proper dummy register.
The vertical lines extending through memory are the entry and exit busses for the "0" (on the left of the bit) and for the "1" (on the right of the bit); also the "0" and "1" compare lines (on either side of the storage loops), carrying the information from the entry register to be compared with the bit status. The no compare line will carry current instead of either the "0" or "1" line if this bit position is to be masked out, thus forcing an equal comparison as far as this bit is concerned.

As an example of operation, assume that the bit shown is the right-hand bit of the interrogating tag in a forward sorting operation. Bits to the left have shown an equal status. This bit is a "1" compared with a "0" in the entry register. Current flowing into the comparing circuit of this bit on the equal line, from the left, will be blocked from the three upper comparing lines and permitted to flow through the bottom line, thus onto the high line into the control section of this word, as shown in Figure 5. Assuming that this word, W, is the first word in memory having the "high" status, the control circuits will operate the dummy out line during the "A" half of cycle, thus moving the word from dummy register W up to word register (W-1) completing the previous sort operation. During the second half cycle, "B," the word from the entry register will be entered in dummy register W preparatory to being moved into word register (W-1), on the "A" half cycle of the next sort operation.

Figure 5 shows the circuits for operating the entry and transfer circuits as controlled by the high, low, and equal signals.

**Conclusions**

By extending the properties of an associative memory, it is possible to secure a self-sorting memory. The work thus far done is of a purely systems nature since the proper components are not yet available. With a self-sorting memory, programming can be materially simplified and the internal sorting procedure improved.

**Acknowledgments**

The assistance of Arthur J. Scriver, Jr. is gratefully acknowledged.

**References**


Fig. 1. Memory System.
Fig. 2. Sorting Example.
Fig. 3. Cryotron Flip-Flop.
Fig. 4. Data Bit.
Fig. 5, Word Register Control.