Associative Memory

Associative Memory

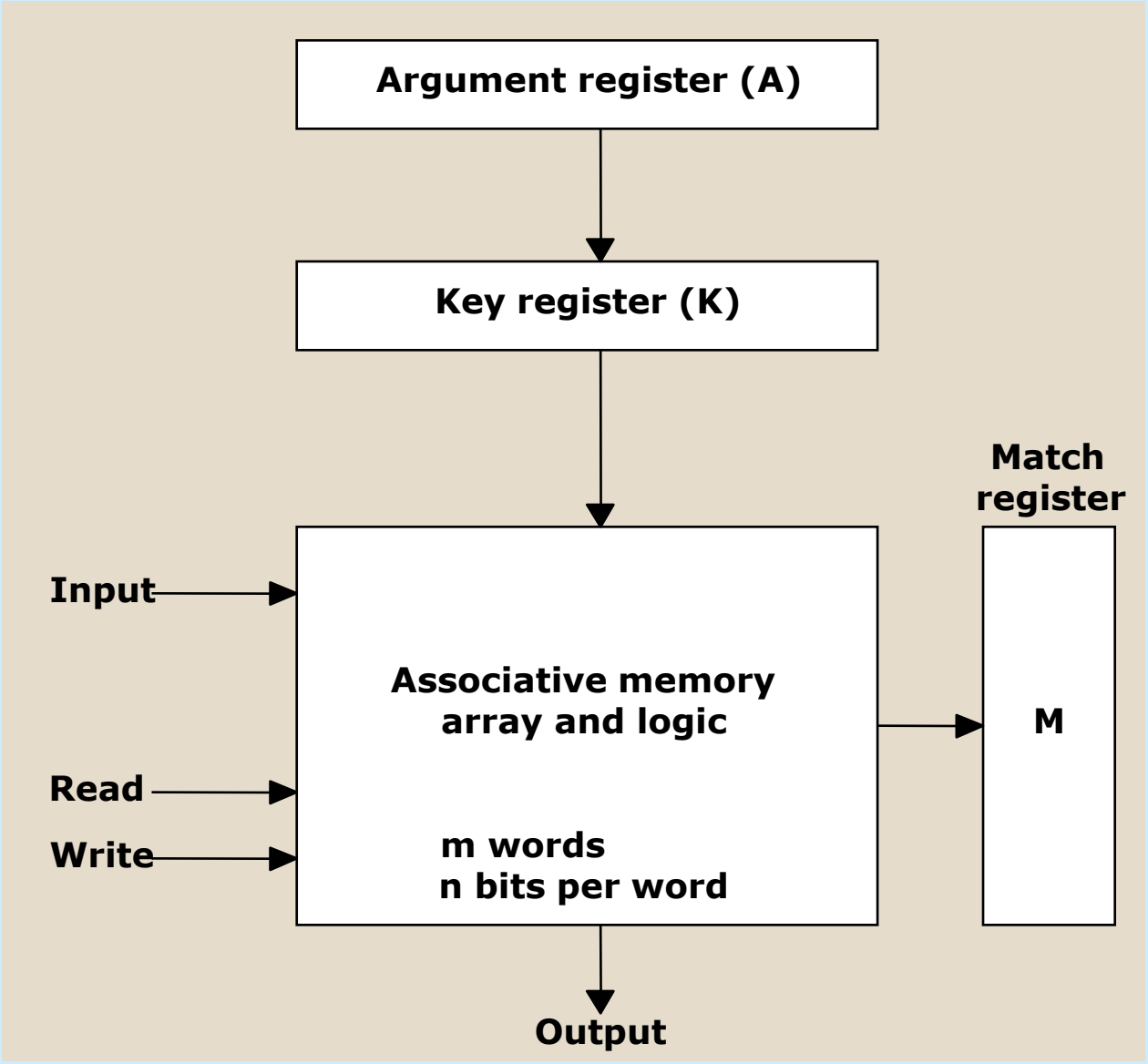
The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.

A memory unit access by content is called an associative memory or Content Addressable Memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than specific address or location.

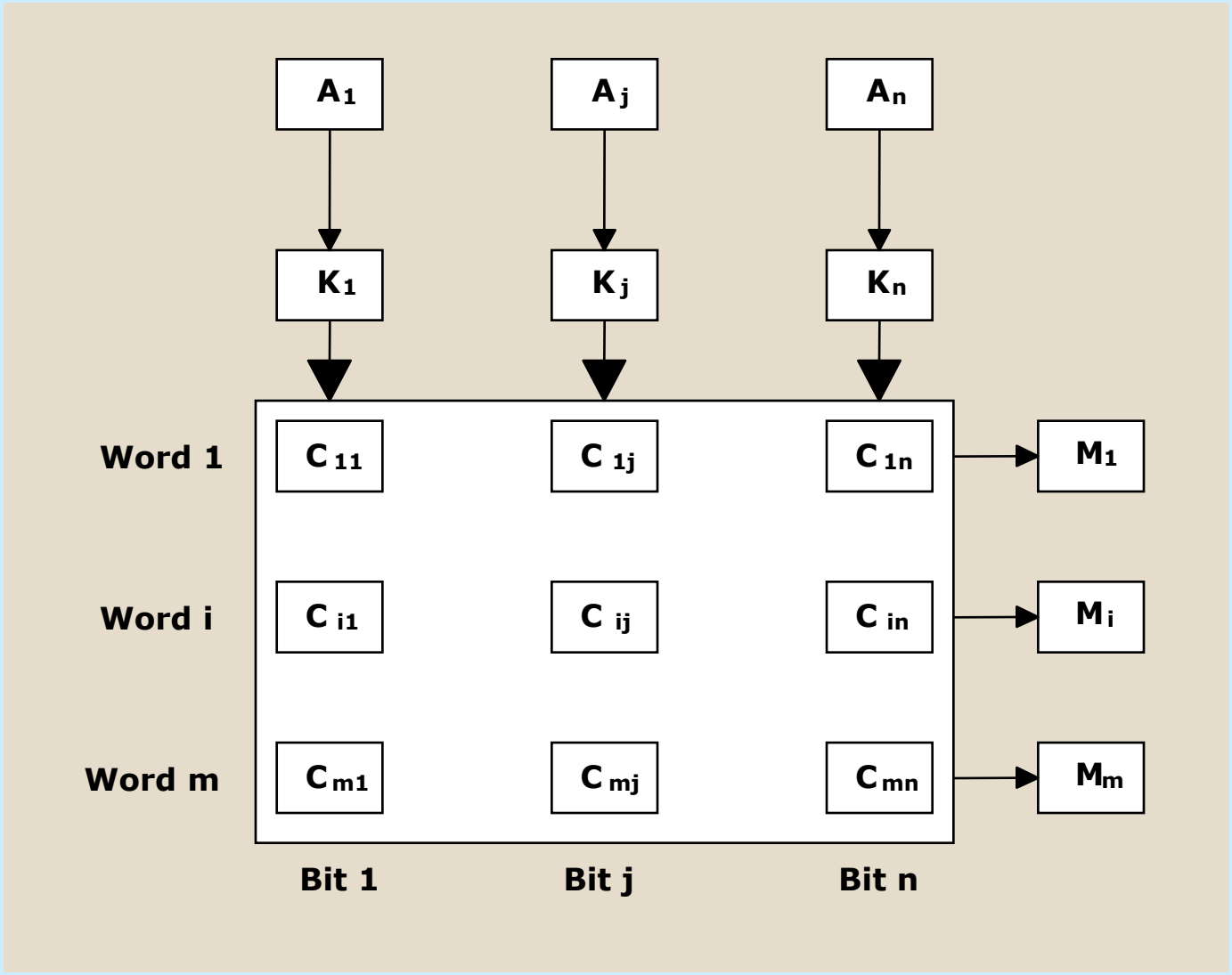
When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word. When a word is to be read from an associative memory, the content of the word or part of the word is specified.

The associative memory is uniquely suited to do parallel searches by data association. Moreover, searches can be done on an entire word or on a specific field within a word. Associative memories are used in applications where the search time is very critical and must be very short.

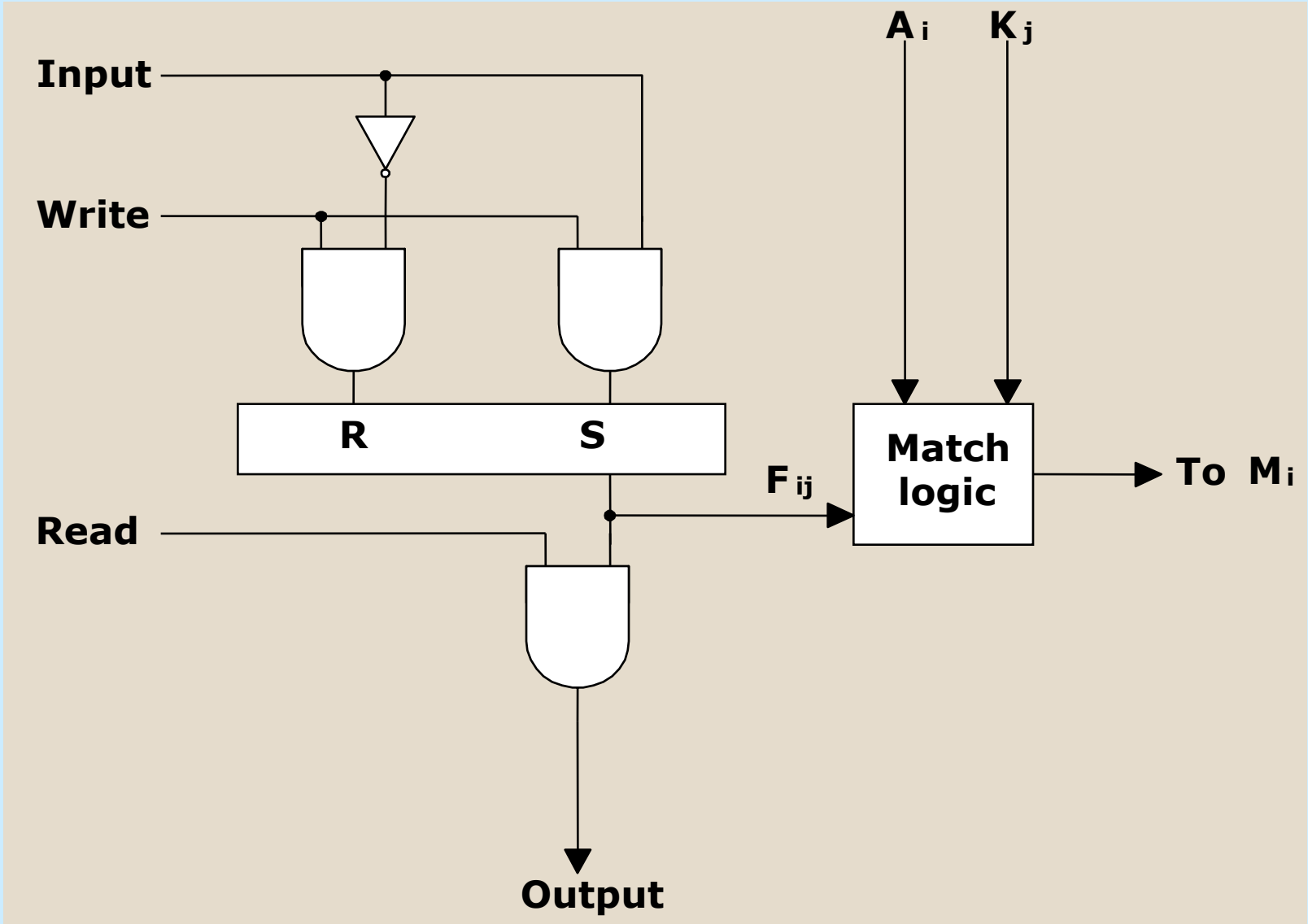
Hardware Organization



Associative memory of an m word, n cells per word



One Cell of Associative Memory



The average time required to reach a storage location in memory and obtain its contents is called the …….

A) Seek Time
B) Transfer Time
C) Access Time
D) Turnaround Time

Match logic

Neglect the K bits and compare the argument in A with the bits stored in the cells of the words.

Word i is equal to the argument in A if

$$A_j = F_{ij} \text{ for } j=1,2\ldots n$$

Two bits are equal if they are both 1 or 0

$$X_j = A_j F_{ij} + A_j' F_{ij}'$$

For a word i to be equal to the argument in A we must have all x_j variables equal to 1.

This is the condition for setting the corresponding match bit M_i to 1.

$$M_i = x_1 x_2 x_3 \ldots x_n$$

Cont.

Now include the key bit K_j in the comparison logic

The requirement is that if K_j=0, the corresponding bits of A_j and Cj need no comparison. Only when K_j=1 must be compared. This requirement is achieved by OR ing each term with K_j

$$X_j + K_j\text{'} = \begin{cases} x_j \text{ if } k_j = 1 \\ 1 \text{ if } k_j = 0 \end{cases}$$

When K_j =1, we have K_j'=0 and $x_j + 0 = x_j$.
When K_j=0, then k_j '=1 and x_j+1=1

The match logic for word i in an associative memory can now be expressed by the following Boolean function.

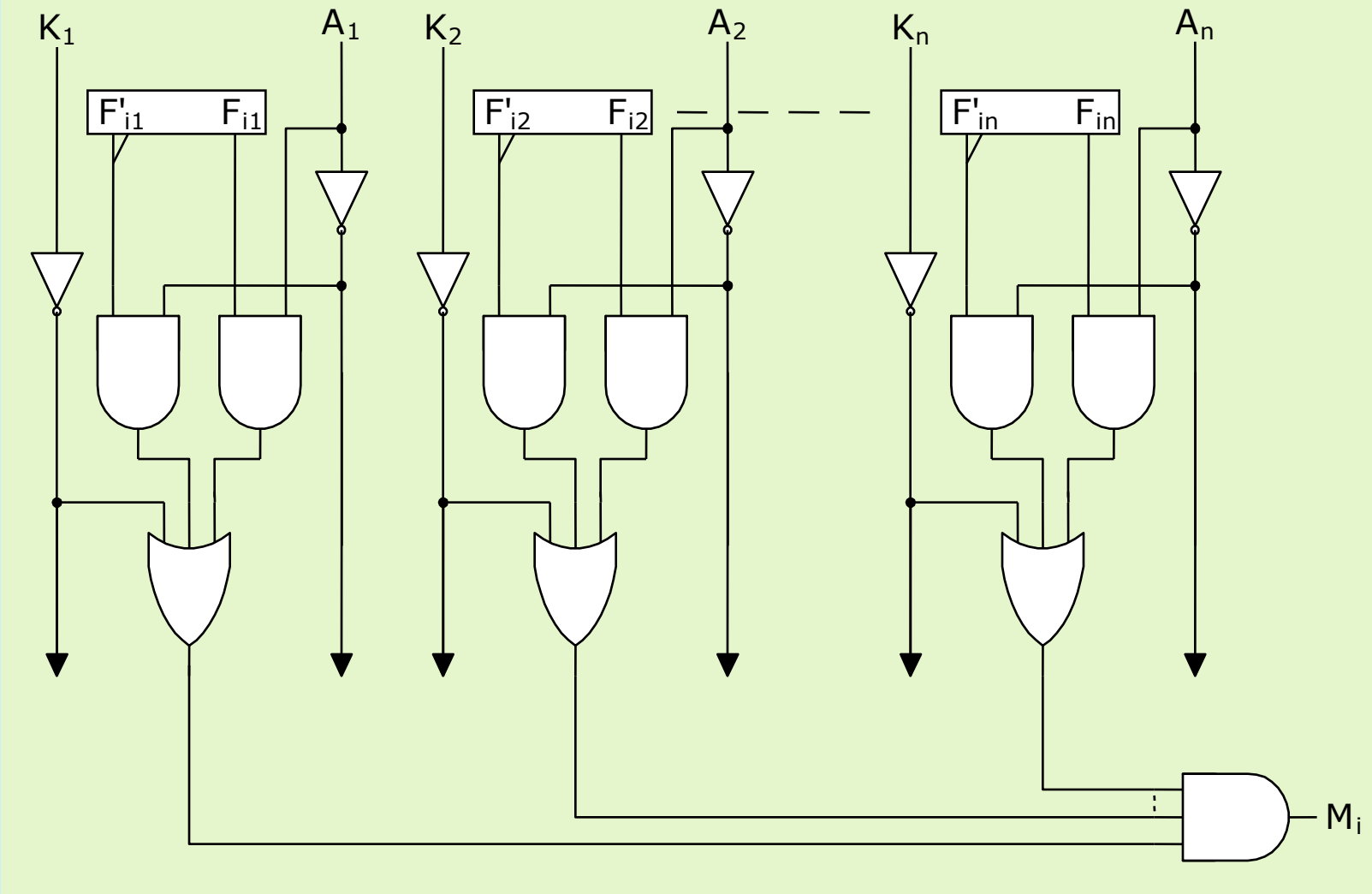
$$M_i = (x_1 + k_1\text{'})\,(x_2 + k_2\text{'})\,(x_3 + k_3\text{'})\,\ldots\,(x_n + k_n\text{'})$$

If we substitute the original definition of x_j, the above Boolean function can be expressed as follows:

$$M_i = \prod_{j=1}^{n} (A_j F_{ij} + A_j\text{'} F_{ij}\text{'} + k_j\text{'})$$

Where is a product symbol designating the AND operation of all n terms.

Match Logic cct.



In a vectored interrupt

(A) the branch address is assigned to a fixed location in memory.

(B) the interrupting source supplies the branch information to the processor through an interrupt vector.

(C) the branch address is obtained from a register in the processor

(D) none of the above

Read Operation

If more than one word in memory matches the unmasked argument field, all the matched words will have 1's in the corresponding bit position of the match register.

It is then necessary to scan the bits of the match register one at a time. The matched words are read in sequence by applying a read signal to each word line whose corresponding M_i bit is a 1.

If only one word may match the unmasked argument field, then connect output M_i directly to the read line in the same word position,

The content of the matched word will be presented automatically at the output lines and no special read command signal is needed.

If we exclude words having zero content, then all zero output will indicate that no match occurred and that the searched item is not available in memory.

Write Operation

If the entire memory is loaded with new information at once, then the writing can be done by addressing each location in sequence.

The information is loaded prior to a search operation.

If unwanted words have to be deleted and new words inserted one at a time, there is a need for a special register to distinguish between active an inactive words.

This register is called “Tag Register”.

A word is deleted from memory by clearing its tag bit to 0.