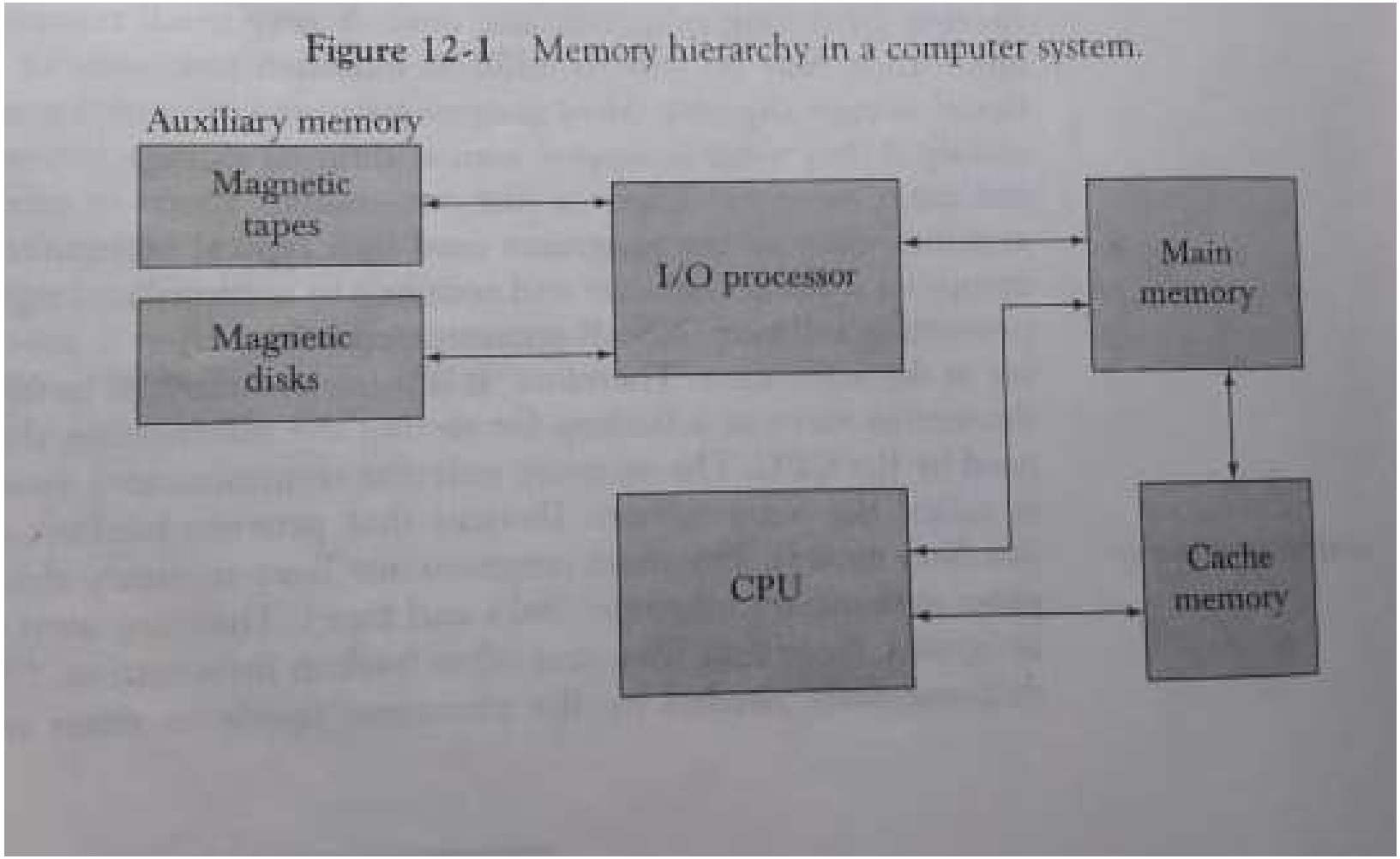
UNIT 4

Memory Organization: Memory Hierarchy, Main Memory, Auxiliary Memory, Associative Memory, Cache Memory - Mapping functions, Replacement algorithms, Write policies.

Memory Hierarchy



- The reason for having two or three levels of memory hierarchy is economics.
- As the storage capacity of the memory increases, the cost per bit for storing binary information decreases and the access time of the memory becomes longer.
- The auxiliary memory has a large storage capacity, is relatively inexpensive, but has low access speed compared to main memory.

- The cache memory is very small, relatively expensive, and has very high access speed.
- Thus, as the memory access speed increases, so does its relative cost.
- The overall goal of using a memory hierarchy is to obtain the highest-possible average access speed while minimizing the total cost of the entire memory system.

- Auxiliary and cache memories are used for different purposes.

- The cache holds those parts of the program and data that are most heavily used, while the auxiliary memory holds those parts that are not presently used by the CPU.

- Moreover, the CPU has direct access to both cache and main memory but not to auxiliary memory.

- The transfer from auxiliary to main memory is usually done by means of direct memory access of large blocks of data.

- The typical access time ratio between cache and main memory is about 1 to 7.
- For example, a typical cache memory may have an access time of 100 ns, while main memory access time may be 700 ns.
- Auxiliary memory average access time is usually 1000 times that of main memory.
- Block size in auxiliary memory typically ranges from 256 to 2048 words, while cache block size is typically from 1 to 16 words.

Main Memory

- The principal technology used for the main memory is based on semiconductor integrated circuits.
- Integrated circuit RAM chips are available in two possible operating modes, static and dynamic.
- The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip.
- The static RAM is easier to use and has shorter read and write cycles.

- One of the major applications of the static RAM is in implementing the cache memories.
- The dynamic RAMs are used for implementing the main memory.
- The ROM portion of main memory is needed for storing an initial program called a bootstrap loader.
- The bootstrap loader is a program whose function is to start the computer software operating when power is turned on.

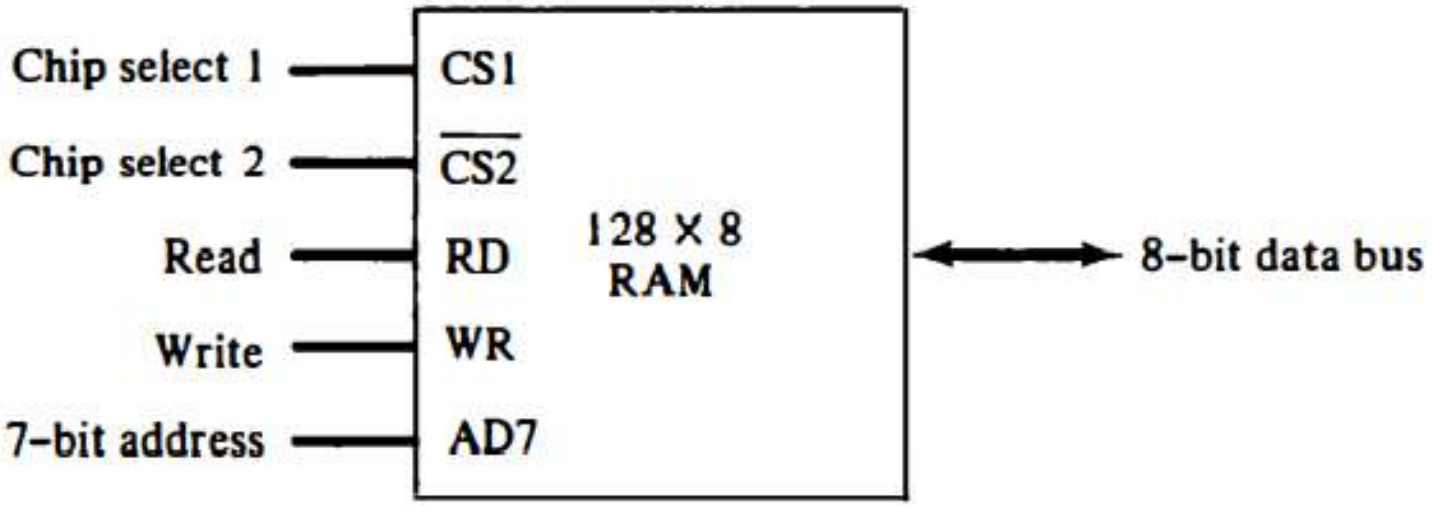
- when power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader.

- The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the operating system, which prepares the computer for general use.

RAM and ROM chips

- RAM and ROM chips are available in a variety of sizes.
- To demonstrate the chip interconnection, an example of a 1024 x 8 memory constructed with 128 x 8 RAM chips and 512 x 8 ROM chips.

Figure 12-2 Typical RAM chip.



(a) Block diagram

CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

(b) Function table

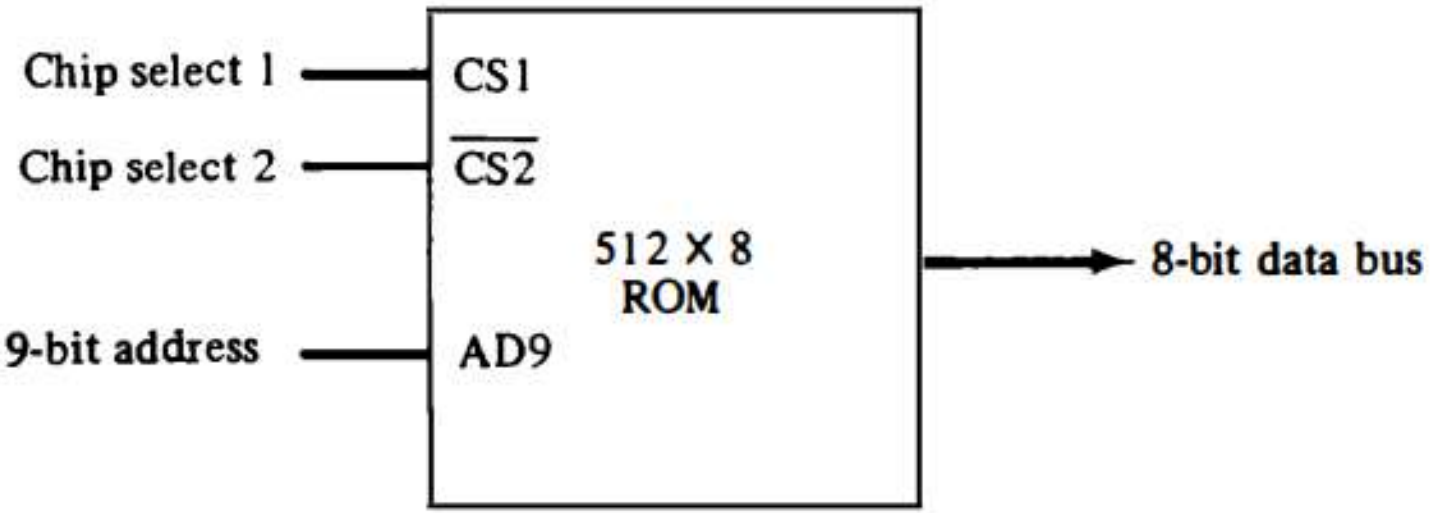


Figure 12-3 Typical ROM chip.

Memory Address Map

TABLE 12-1 Memory Address Map for Microprocomputer

Component	Hexadecimal address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000–007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080–00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100–017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180–01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200–03FF	1	x	x	x	x	x	x	x	x	x

Memory Connection to CPU

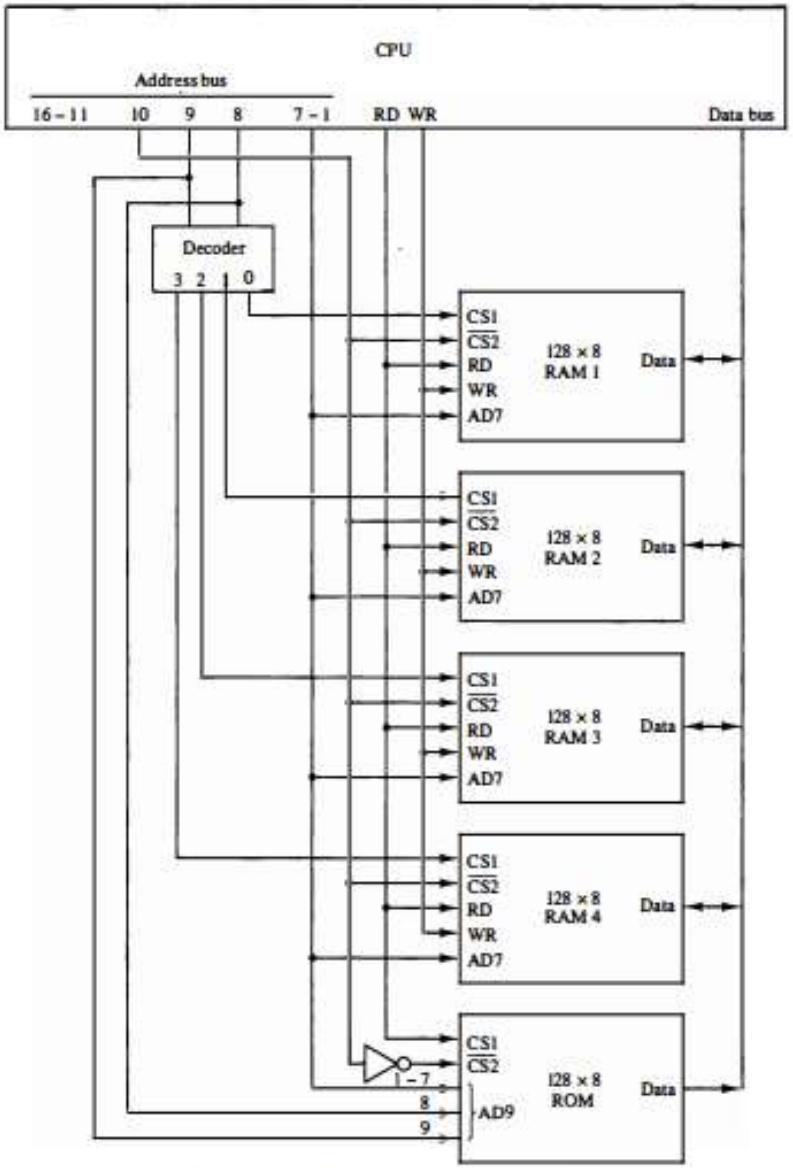


Figure 12-4 Memory connection to the CPU.

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 accessed 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 by 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 memory locates all words which match the specified content and marks them for reading.

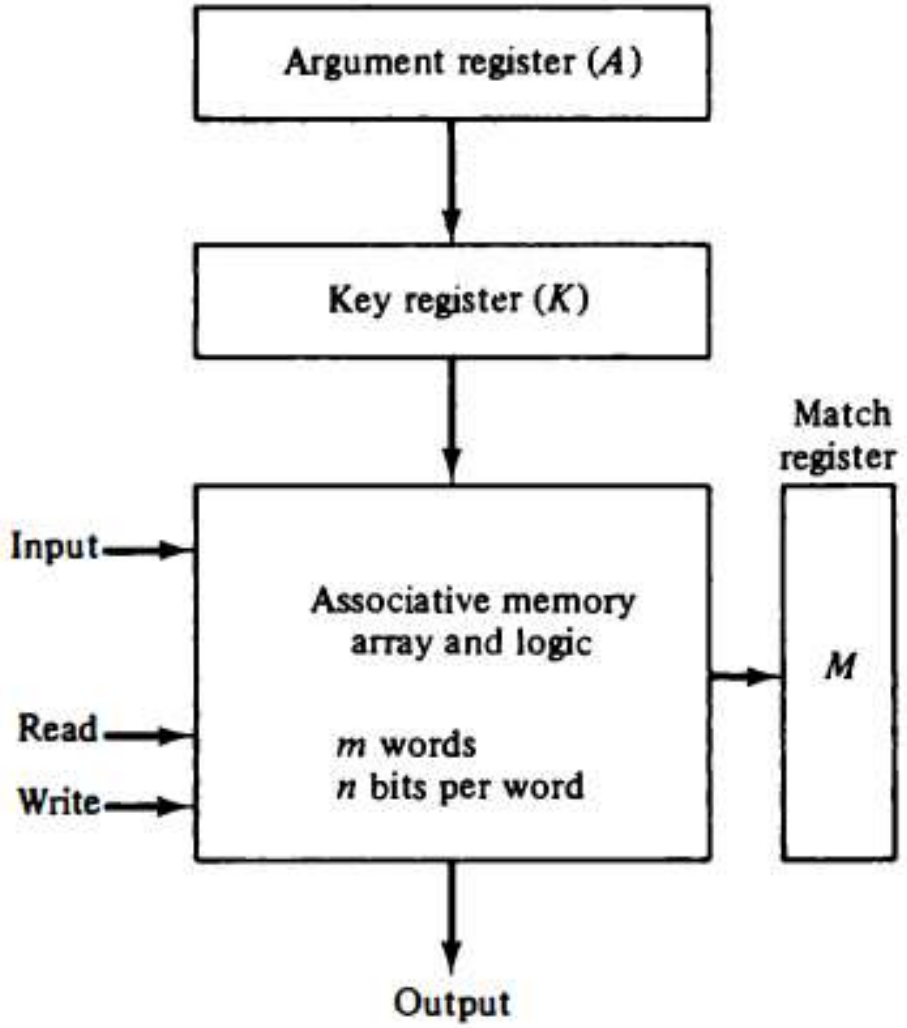
- Because of its organization, the associative memory is uniquely suited to do parallel searches by data association.

- An associative memory is more expensive than a random access memory because each cell must have storage capability as well as logic circuits for matching its content with an external argument.

- For this reason, associative memories are used in applications where the search time is very critical and must be very short.

Hardware Organization

Figure 12-6 Block diagram of associative memory.

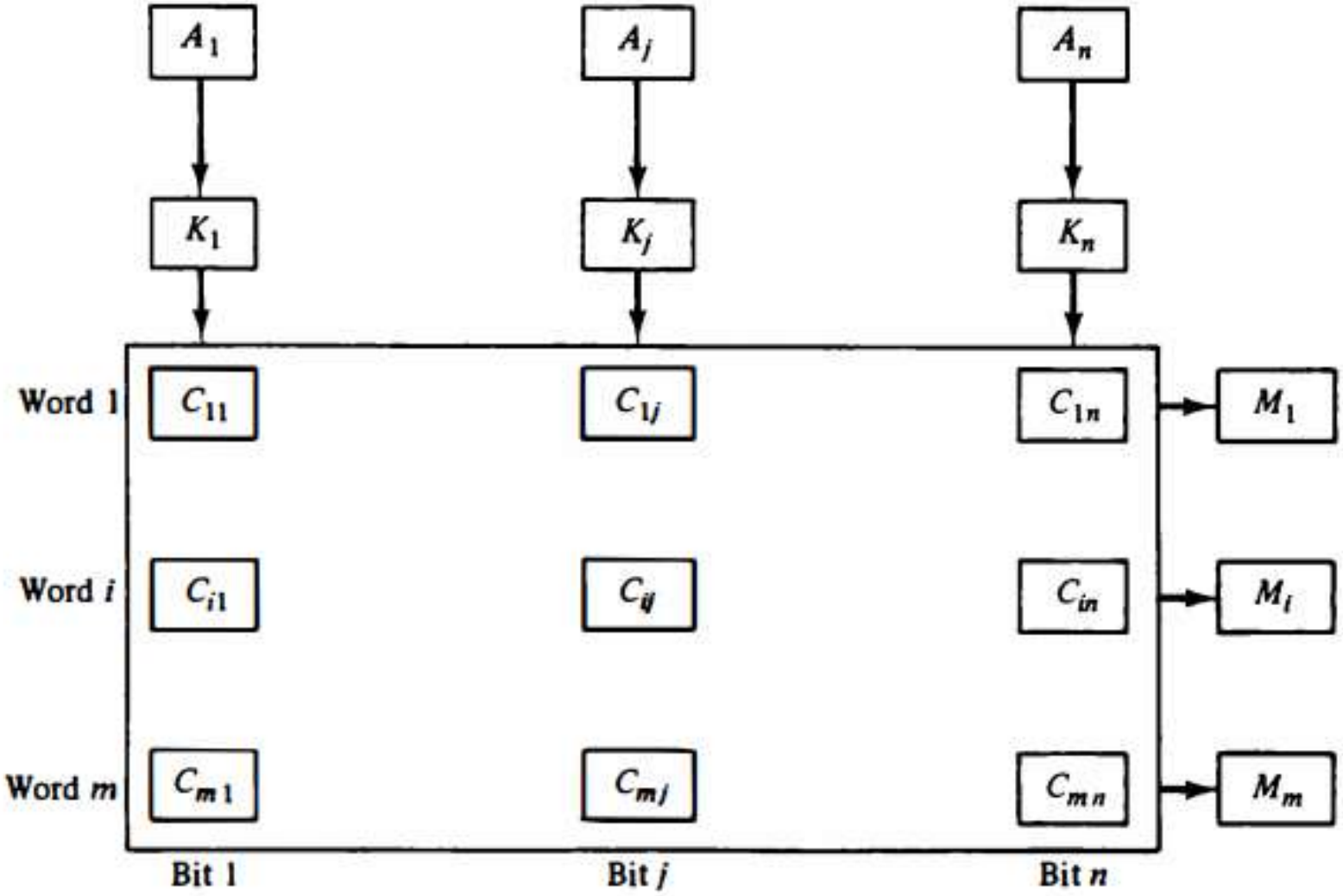


- The key register provides a mask for choosing a particular field or key in the argument word.
- The entire argument is compared with each memory word if the key register contains all l' s.
- Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared.
- Thus the key provides a mask or identifying piece of information which specifies how the reference to memory is made.

Example

A	101	111100	
K	111	000000	
Word 1	100	111100	no match
Word 2	101	000001	match

Figure 12-7 Associative memory of m word, n cells per word.

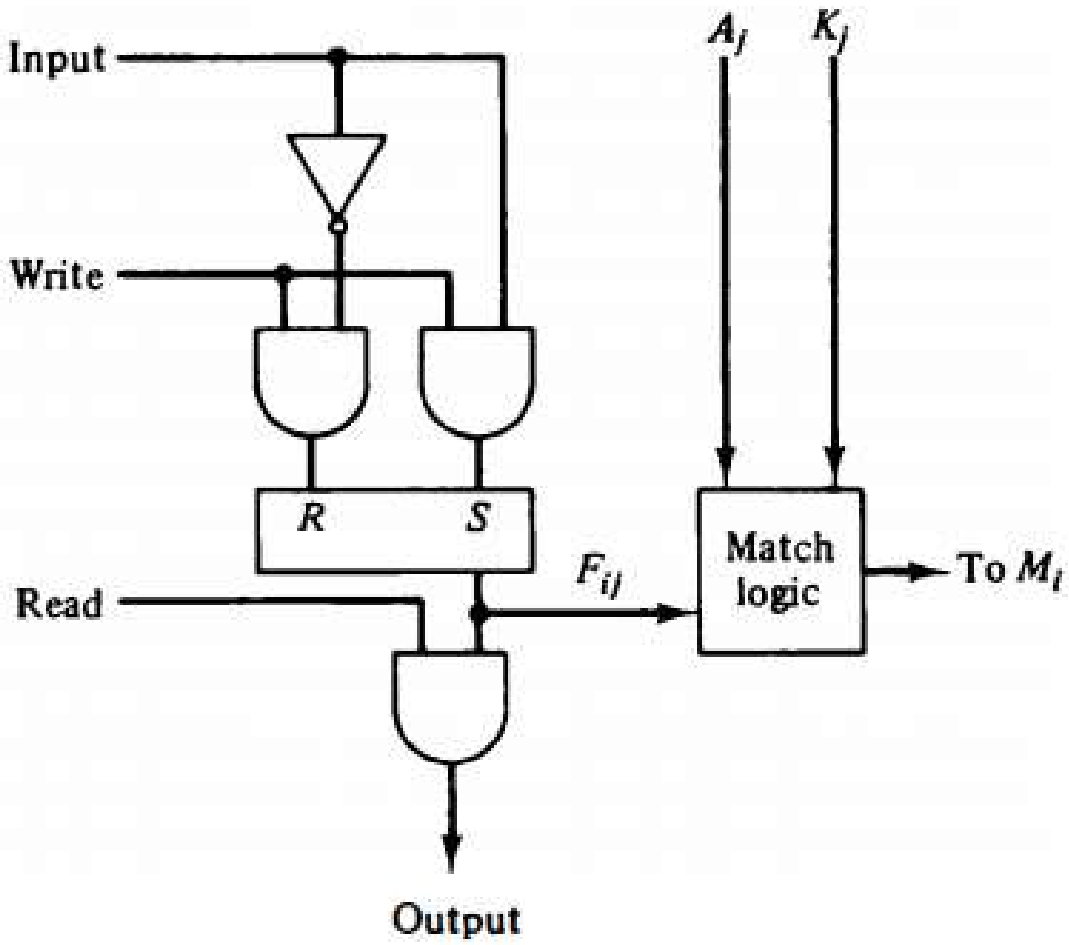


- The cells in the array are marked by the letter C with two subscripts.
- The first subscript gives the word number and the second specifies the bit position in the word. Thus cell C_{ij} is the cell for bit j in word i.
- A bit A_i in the argument register is compared with all the bits in column j of the array provided that $K_i = 1$.

- This is done for all columns j = 1, 2, . . . , n. If a match occurs between all the unmasked bits of the argument and the bits in word i, the corresponding bit M_i in the match register is set to 1.

- If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

Figure 12-8 One cell of associative memory.



- It consists of a flip-flop storage element F_{ij} and the circuits for reading, writing, and matching the cell.
- The input bit is transferred into the storage cell during a write operation.
- The bit stored is read out during a read operation.
- The match logic compares the content of the storage cell with the corresponding unmasked bit of the argument and provides an output for the decision logic that sets the bit in M_i.

Match Logic

- First, we neglect the key 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 $A_j = F_{ij}$ for $j = 1, 2, \ldots, n$.

- Two bits are equal if they are both 1 or both 0. The equality of two bits can be expressed logically by the Boolean function

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

where $x_j = 1$ if the pair of bits in position j are equal; otherwise, $x_j = 0$.

- 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. The Boolean function for this condition is

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

and constitutes the AND operation of all pairs of matched bits in a word.

- We 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 F_{ij}; need no comparison.

- Only when $K_j = 1$ must they be compared.

- This requirement is achieved by ORing each term with K'_j, thus:

$$x_j + K'_j = \begin{cases} x_j & \text{if } K_j = 1 \\ 1 & \text{if } K_j = 0 \end{cases}$$

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

$$M_i = (x_1 + K_1')(x_2 + K_2')(x_3 + K_3') \cdots (x_n + K_n')$$

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

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

where $\prod$ is a product symbol designating the AND operation of all n terms. We need m such functions, one for each word i = 1, 2, 3, ... , m.

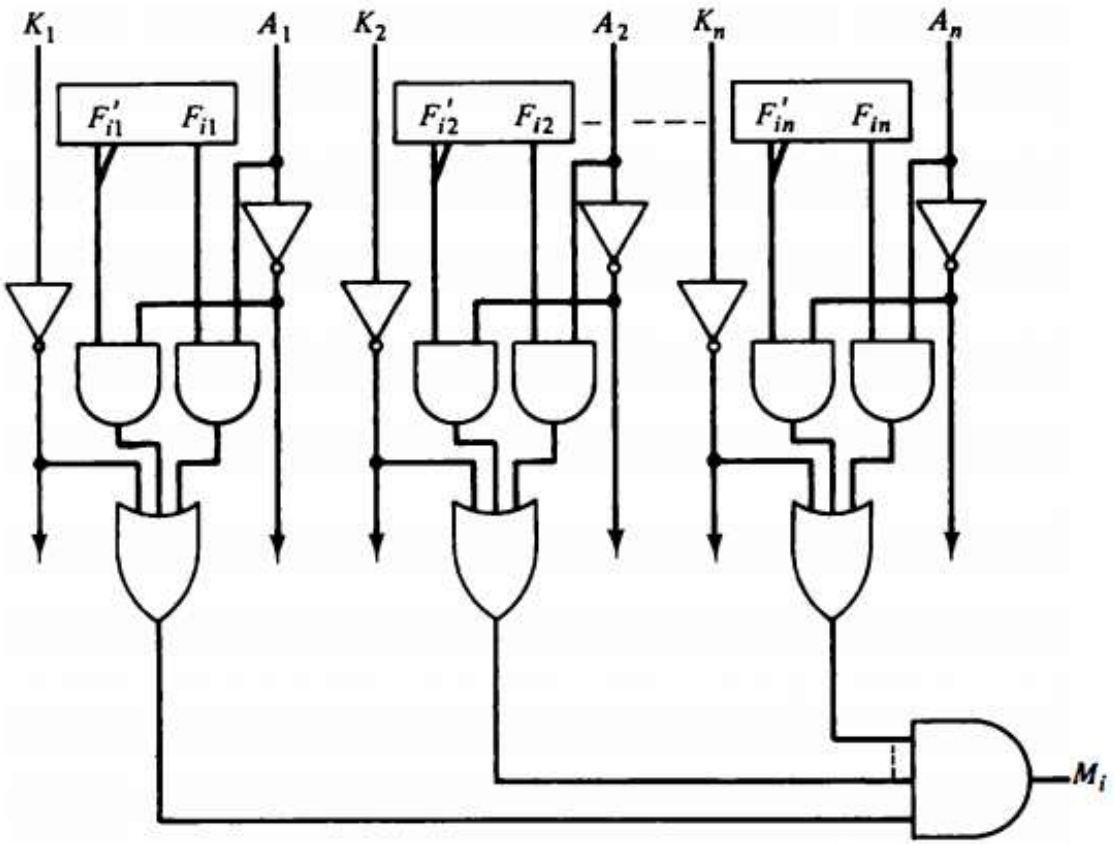


Figure 12-9 Match logic for one word of associative memory.

Read Operation

- In most applications, the associative memory stores a table with no two identical items under a given key.

- In this case, only one word may match the unmasked argument field.

- By connecting output M_i directly to the read line in the same word position (instead of the M register), the content of the matched word will be presented automatically at the output lines and no special read command signal is needed.

Write Operation

- Writing in an associative memory can take different forms, depending on the application.
- If the entire memory is loaded with new information at once prior to a search operation then the writing can be done by addressing each location in sequence.
- This will make the device a random access memory for writing and a content addressable memory for reading.

- 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 and inactive words.

- This register, sometimes called a tag register, would have as many bits as there are words in the memory.

- For every active word stored in memory, the corresponding bit in the tag register is set to 1. A word is deleted from memory by clearing its tag bit to 0.

- Words are stored in memory by scanning the tag register until the first 0 bit is encountered.
- This gives the first available inactive word and a position for writing a new word.
- After the new word is stored in memory it is made active by setting its tag bit to 1.
- An unwanted word when deleted from memory can be cleared to all 0' s if this value is used to specify an empty location.

Cache Memory

- Analysis of a large number of typical programs has shown that the references to memory at any given interval of time tend to be confined within a few localized areas in memory.

- This phenomenon is known as the property of locality of reference.

- If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a cache memory.

- The cache memory access time is less than the access time of main memory by a factor of 5 to 10.

- The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

Basic operation of the Cache

- When the CPU needs to access memory, the cache is examined.
- If the word is found in the cache, it is read from the fast memory.
- If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.
- A block of words containing the one just accessed is then transferred from main memory to cache memory.

- The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed.

- In this manner, some data are transferred to cache so that future references to memory find the required words in the fast cache memory.

Hit ratio

- The performance of cache memory is frequently measured in terms of a quantity called hit ratio.
- When the CPU refers to memory and finds the word in cache, it is said to produce a hit.
- If the word is not found in cache, it is in main memory and it counts as a miss.
- The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio.

- The average memory access time of a computer system can be improved considerably by use of a cache.

- If the hit ratio is high enough so that most of the time the CPU accesses the cache instead of main memory, the average access time is closer to the access time of the fast cache memory.

Mapping functions

- The transformation of data from main memory to cache memory is referred to as a mapping process.

- Three types of mapping procedures are of practical interest when considering the organization of cache memory:

- 1. Associative mapping
- 2. Direct mapping
- 3. Set-associative mapping

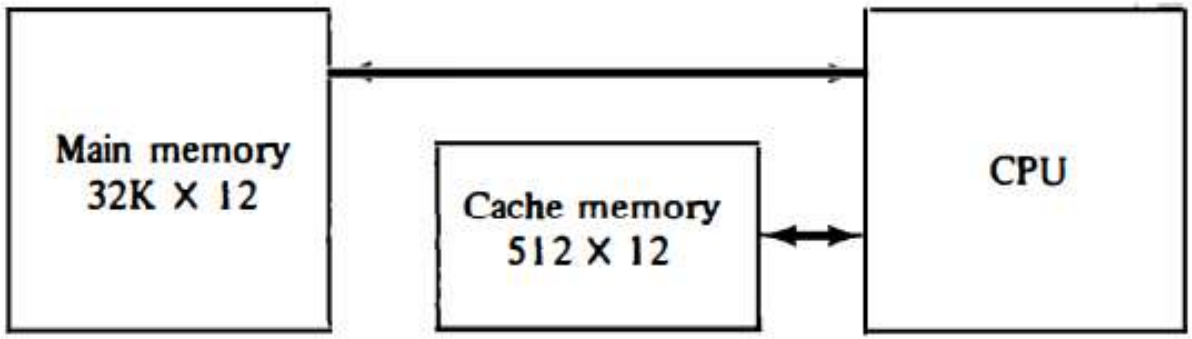
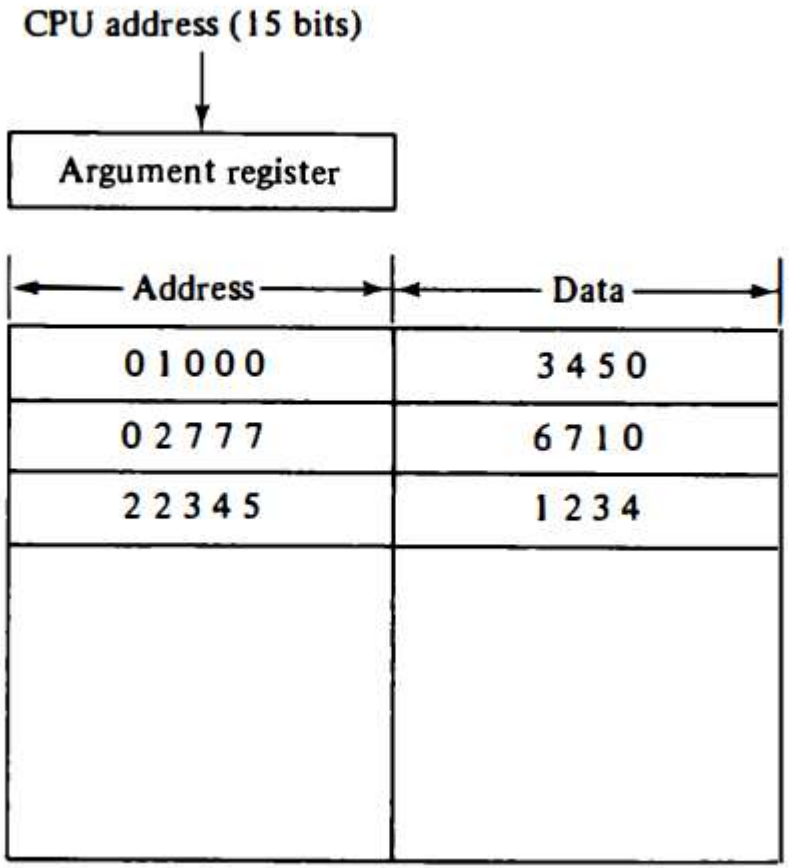


Figure 12-10 Example of cache memory.

Associative mapping

- The fastest and most flexible cache organization uses an associative memory.
- The associative memory stores both the address and content (data) of the memory word.
- This permits any location in cache to store any word from main memory.

Figure 12-11 Associative mapping cache (all numbers in octal).



- If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache.

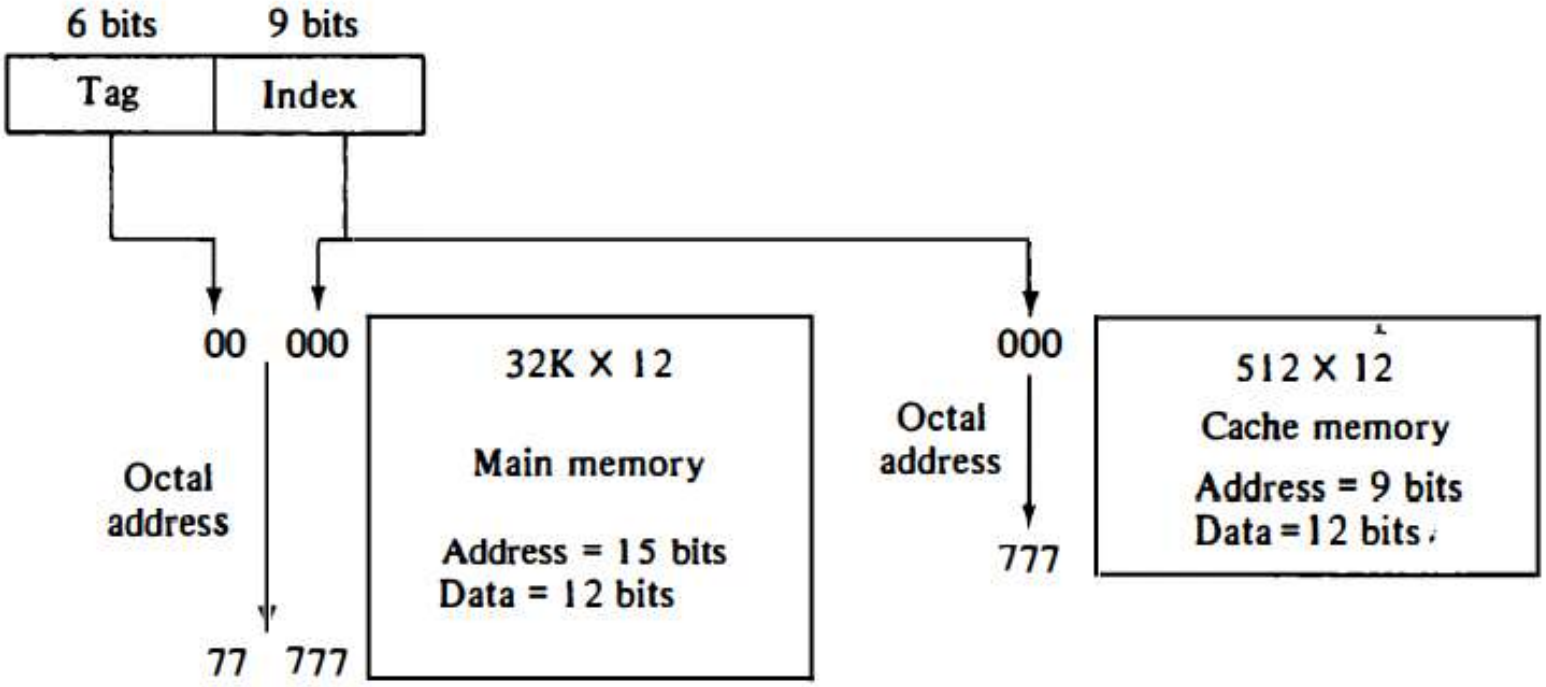
- The decision as to what pair is replaced is determined from the replacement algorithm that the designer chooses for the cache.

- A simple procedure is to replace cells of the cache in round-robin order whenever a new word is requested from main memory. This constitutes a first-in first-out (FIFO) replacement policy.

Direct mapping

- Associative memories are expensive compared to random-access memories because of the added logic associated with each cell.
- In the general case, there are 2^k words in cache memory and 2^n words in main memory.
- The n-bit memory address is divided into two fields: k bits for the index field and n - k bits for the tag field.
- The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.

Figure 12-12 Addressing relationships between main and cache memories.



Direct mapping cache with block size of one word

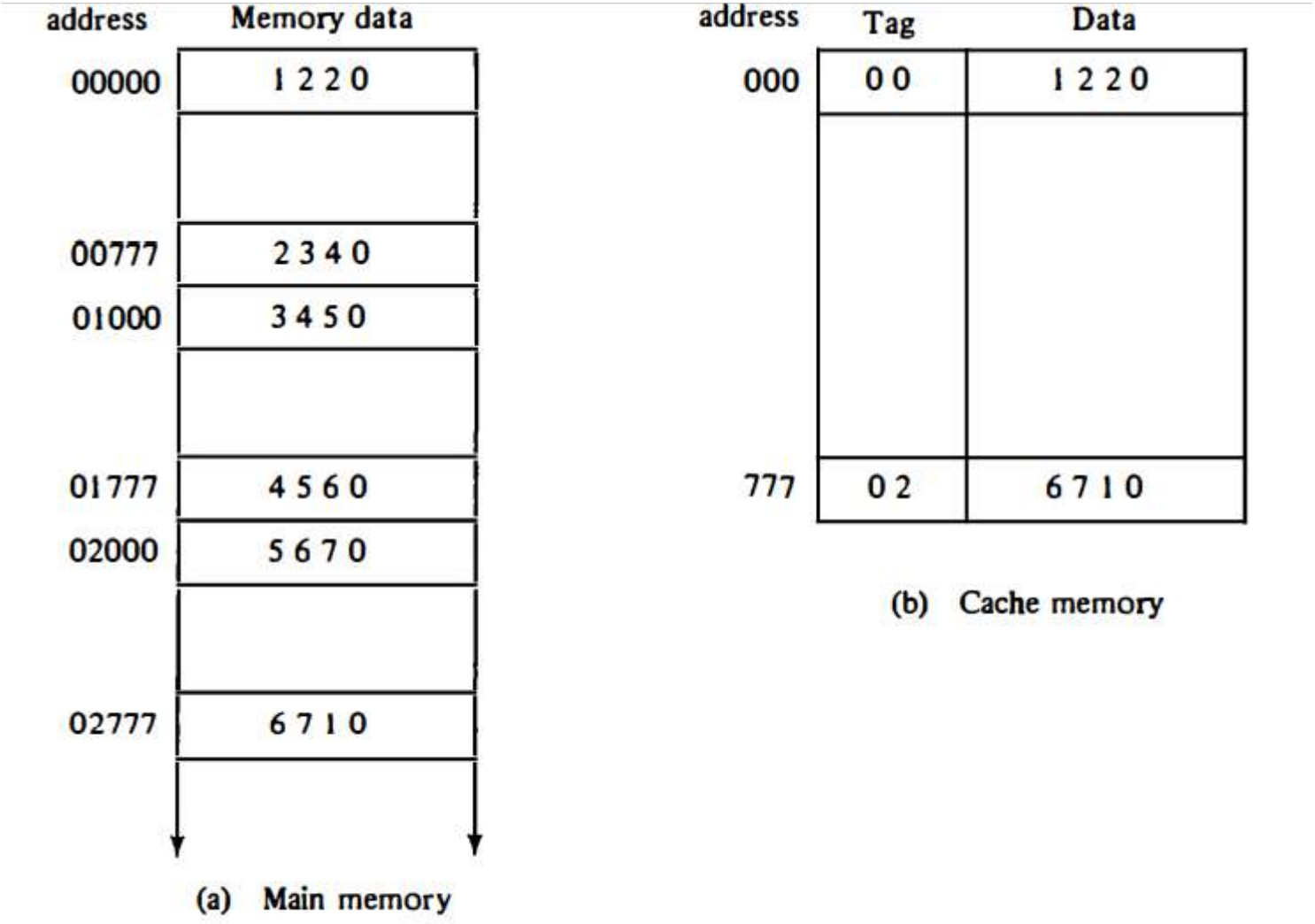


Figure 12-13 Direct mapping cache organization.

- The tag field of the CPU address is compared with the tag in the word read from the cache.
- If the two tags match, there is a hit and the desired data word is in cache.
- If there is no match, there is a miss and the required word is read from main memory.
- It is then stored in the cache together with the new tag, replacing the previous value.

- The disadvantage of direct mapping is that the hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly.

- This possibility is minimized by the fact that such words are relatively far apart in the address range.

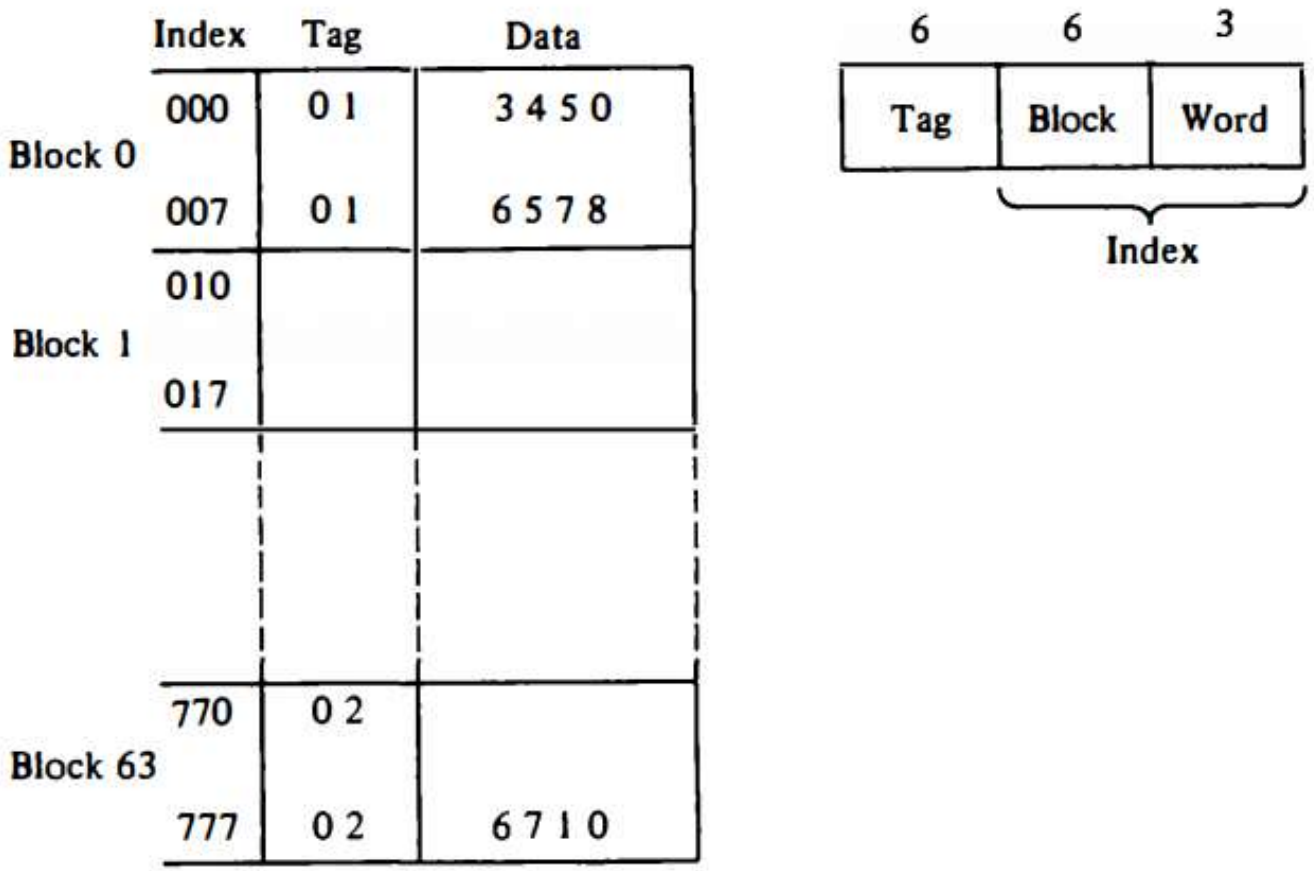


Figure 12-14 Direct mapping cache with block size of 8 words.

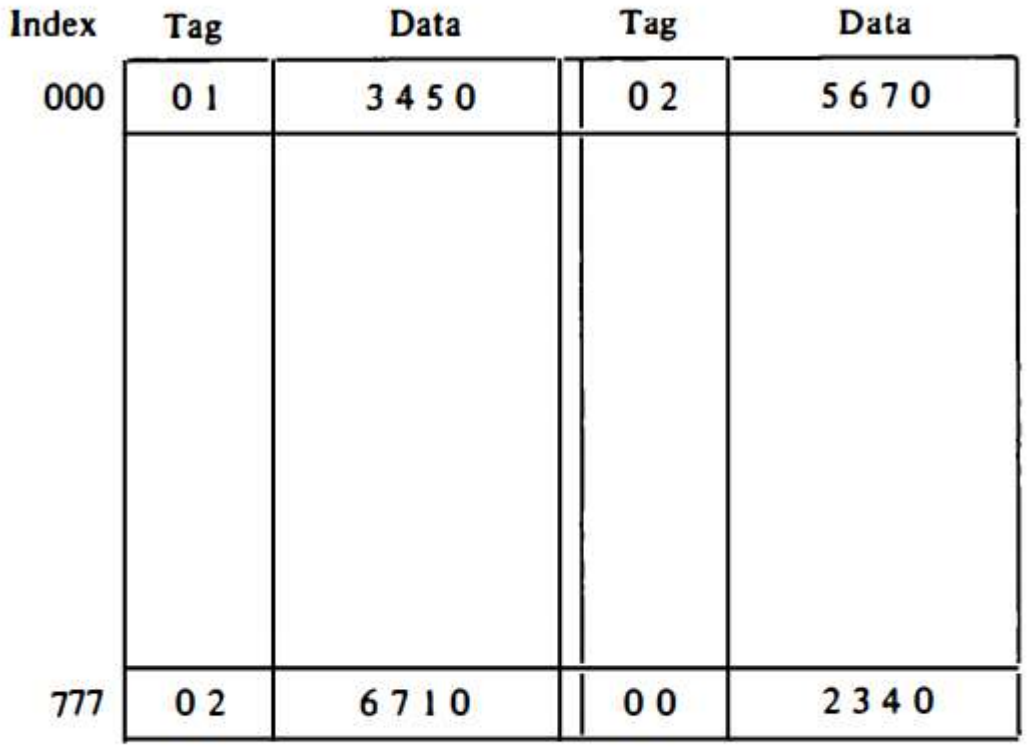
- The tag field stored within the cache is common to all eight words of the same block.
- Every time a miss occurs, an entire block of eight words must be transferred from main memory to cache memory.
- Although this takes extra time, the hit ratio will most likely improve with a larger block size because of the sequential nature of computer programs.

Set-Associative mapping

- The disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

- A third type of cache organization, called set-associative mapping, is an improvement over the direct mapping organization in that each word of cache can store two or more words of memory under the same index address.

- Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set.



Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

Figure 12-15 Two-way set-associative mapping cache.

- Each index address refers to two data words and their associated tags.
- Each tag requires six bits and each data word has 12 bits, so the word length is 2(6 + 12) = 36 bits.
- An index address of nine bits can accommodate 512 words. Thus the size of cache memory is 512 x 36.
- It can accommodate 1024 words of main memory since each word of cache contains two data words.
- In general, a set-associative cache of set size k will accommodate k words of main memory in each word of cache.

- When the CPU generates a memory request, the index value of the address is used to access the cache.
- The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs.
- The comparison logic is done by an associative search of the tags in the set similar to an associative memory search: thus the name "set-associative."

- The hit ratio will improve as the set size increases because more words with the same index but different tags can reside in cache.

- An increase in the set size increases the number of bits in words of cache and requires more complex comparison logic.

Replacement Algorithms

- When a miss occurs in a set-associative cache and the set is full, it is necessary to replace one of the tag-data items with a new value.

- The most common replacement algorithms used are: random replacement, first-in, first-out (FIFO), and least recently used (LRU).

- With the random replacement policy the control chooses one tag-data item for replacement at random.
- The FIFO procedure selects for replacement the item that has been in the set the longest.
- The LRU algorithm selects for replacement the item that has been least recently used by the CPU.
- Both FIFO and LRU can be implemented by adding a few extra bits in each word of cache.

Write policies

- When the CPU finds a word in cache during a read operation, the main memory is not involved in the transfer.

- However, if the operation is a write, there are two ways that the system can proceed.

- The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache memory being updated in parallel if it contains the word at the specified address. This is called the write-through method.

- The second procedure is called the write-back method. In this method only the cache location is updated during a write operation. The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory.

Cache initialization

- It is customary to include with each word in cache a valid bit to indicate whether or not the word contains valid data.
- The cache is initialized by clearing all the valid bits to 0.
- The valid bit of a particular cache word is set to 1 the first time this word is loaded from main memory and stays set unless the cache has to be initialized again.

- The introduction of the valid bit means that a word in cache is not replaced by another word unless the valid bit is set to 1 and a mismatch of tags occurs.

- If the valid bit happens to be 0, the new word automatically replaces the invalid data.

- Thus the initialization condition has the effect of forcing misses from the cache until it fills with valid data.

Auxiliary Memory

- The most common auxiliary memory devices used in computer systems are magnetic disks and tapes.
- The average time required to reach a storage location in memory and obtain its contents is called the access time.
- In electromechanical devices with moving parts such as disks and tapes, the access time consists of a seek time required to position the read-write head to a location and a transfer time required to transfer data to or from the device.

Magnetic Disks

- A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.
- Often both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface.
- Bits are stored in the magnetized surface in spots along concentric circles called tracks.
- The tracks are commonly divided into sections called sectors.
- In most systems, the minimum quantity of information which can be transferred is a sector.

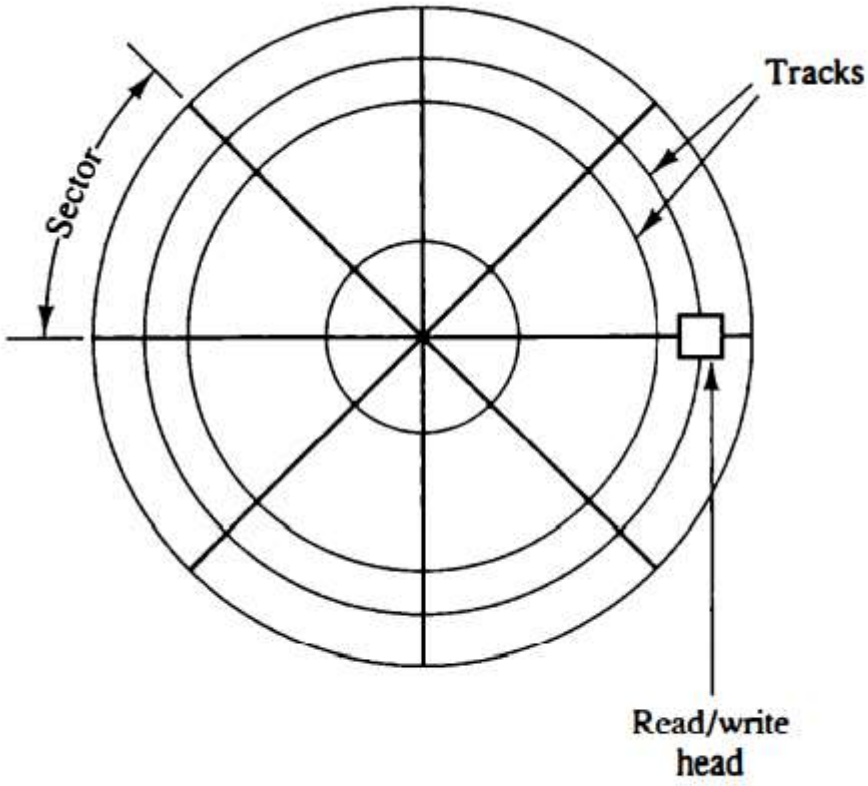


Figure 12-5 Magnetic disk.

- A disk system is addressed by address bits that specify the disk number, the disk surface, the sector number and the track within the sector.
- Information transfer is very fast once the beginning of a sector has been reached.
- Disks may have multiple heads and simultaneous transfer of bits from several tracks at the same time.
- Disks that are permanently attached to the unit assembly and cannot be removed by the occasional user are called hard disks.
- A disk drive with removable disks is called a floppy disk.

Magnetic Tape

The tape itself is a strip of plastic coated with a magnetic recording medium.

Bits are recorded as magnetic spots on the tape along several tracks.

Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.

Read/write heads are mounted one in each track so that data can be recorded and read as a sequence of characters.