UNIT – III

MEMORY ORGANIZATION

Objective:

- To familiarize with organizational aspects of memory.

Syllabus:

MEMORY ORGANIZATION: Memory Hierarchy, main memory, auxiliary memory, associative memory, cache memory, virtual memory.

Learning Outcomes:

At the end of the unit student will be able to:

- Differentiate different types of memories.
- Analyze the performance of the hierarchical organization of memory.

Learning Material

3.1 MEMORY HIERARCHY

- The memory hierarchy system is shown in the following figure 1.

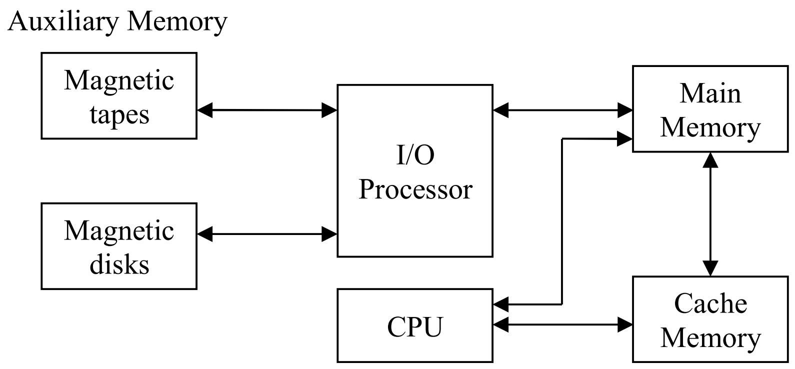


Figure 1: Memory Hierarchy in computer system

- The memory unit that communicates directly with the CPU is called the *main memory*.
- Devices that provide backup storage are called *auxiliary memory*. The most common auxiliary memory devices used in computer systems are magnetic disks and tapes. They are used for storing system programs, large data files, and other backup information.
- Only programs and data currently needed by the processor reside in main memory. All other information is stored in auxiliary memory and transferred to main memory when needed.
- A special very high speed memory called a *Cache* is used to increase the speed of processing by making current programs and data available to the CPU.

- The part of the computer system that supervises the flow of information between auxiliary memory and main memory is called the *memory management system*.

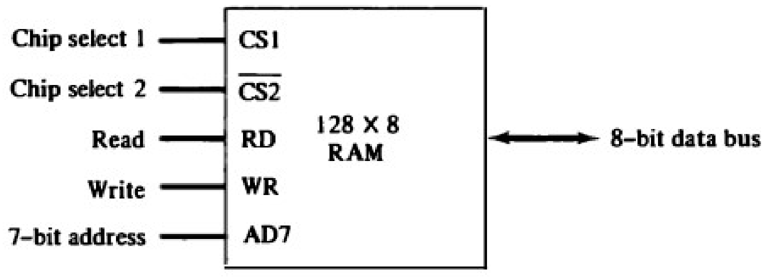
3.2 MAIN MEMORY

- The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs and data during the computer operation.
- The principal technology used for the main memory is based on semiconductor integrated circuits such as RAM and ROM chips.
- RAM chips are available in two possible operating modes, static and dynamic.
- The static RAM consists of internal flip-flops that store the binary information. The stored information remains valid as long as power is applied to the unit.
- The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.
- 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.
- Most of the main memory in a general-purpose computer is made up of RAM integrated circuit chips, but a portion of the memory may be constructed with ROM chips.
- RAM is used for storing the bulk of the programs and data that are subject to change.
- ROM is used for storing programs that are permanently resident in the computer.
- 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.
- Since RAM is *volatile*, its contents are destroyed when power is turned off. The contents of ROM remain unchanged after power is turned off and on again.
- RAM and ROM chips are available in a variety of sizes. If the memory needed for the computer is larger than the capacity of one chip, it is necessary to combine a number of chips to form the required memory size.

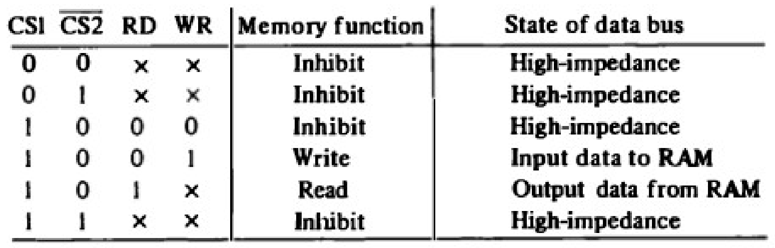
3.2.1 RAM Chip

- A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip only when needed.
- Another common feature is a bidirectional data bus that allows the transfer of data either from memory to CPU during a *read* operation or from CPU to memory during a *write* operation.

- The block diagram of a RAM chip is shown in figure2. The capacity of the memory is 128 words of eight bits (one byte) per word.



(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 2: Typical RAM chip

- It requires a 7-bit address bus and an 8-bit bidirectional data bus.
- The read and write inputs specifies the memory operation and the two chips select (CS) control inputs are for enabling the chip only when it is selected by the microprocessor.
- The availability of more than one control input to select the chip facilitates the decoding the address lines when multiple chips are used in the microcomputer.
- The function table listed in figure 2(b) specifies the operation of the RAM chip.
- The unit in operation only when CS1 = 1 and $\overline{CS2}$ = 0.
- If the chip select inputs are not enabled, or if chip select inputs are enabled but the READ or WRITE inputs are not enabled, then the memory in Inhibit and its data bus is in high-impedance state.

3.2.2 ROM Chip

- A ROM chip is organized externally in a similar manner of RAM chip.
- The block diagram of a ROM chip is shown in Figure3.
- The nine address lines in the ROM chip specify any one of the 512 bytes stored in it.
- The two chip select inputs must be CS1 = 1 and $\overline{CS2}$ = 0 for the unit to operate. Otherwise, the data bus is in a high-impedance state.
- There is no need for a read or write control because the unit is read only.

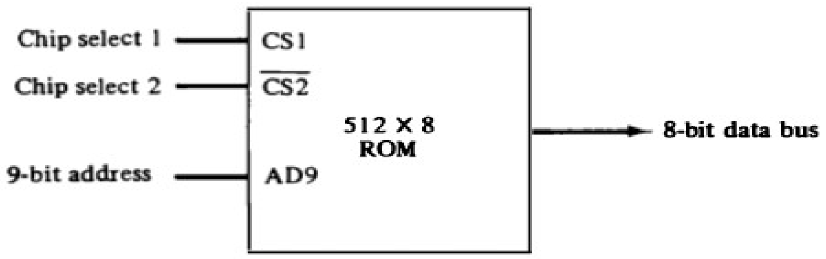


Figure 3: Typical ROM chip

3.2.3 Memory Address Map

- The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM.
- The interconnection between memory and processor is then established from knowledge of the size of memory needed and the type of RAM and ROM chips available.
- The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip. The table, called a memory address map, is a pictorial representation of assigned address space for each chip in the system.
- The memory address map for this configuration is shown in Table 1. The component column specifies whether a RAM or a ROM chip is used. The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip. The address bus lines are listed in the third column.

Table 1: Memory Address Map for Microcomputer

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

- Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero.

3.2.4 Memory Connection to CPU

- RAM and ROM chips are connected to a CPU through the data and address buses. The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs. The connection of memory chips to the CPU is shown in Figure 4.
- This configuration gives a memory capacity of 512 bytes of RAM and 512 bytes of ROM.
- Each RAM receives the seven low-order bits of the address bus to select one of 128 possible bytes. The particular RAM chip selected is determined from lines 8 and 9 in the address bus. This is done through a 2 x 4 decoder whose outputs go to the *CS1* inputs in each RAM chip.
- Thus, when address lines 8 and 9 are equal to 00, the first RAM chip is selected. When 01, the second RAM chip is selected, and so on.
- The RD and WR outputs from the microprocessor are applied to the inputs of each RAM chip.
- The selection between RAM and ROM is achieved through bus line 10. The RAMs are selected when the bit in this line is 0, and the ROM when the bit is 1.

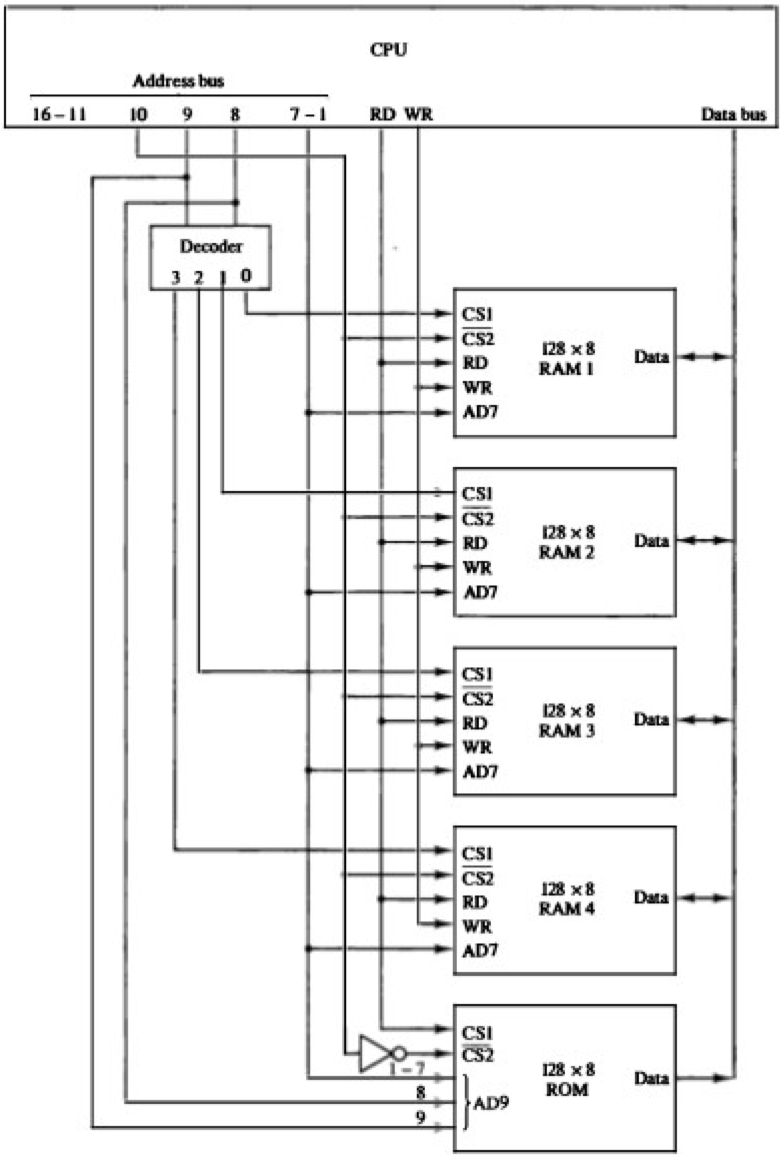


Figure 4: Memory connection to the CPU

3.3 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.

- Bits are recorded as magnetic spots on the surface as it passes a stationary mechanism called a *write head*. Stored bits are detected by a change in magnetic field produced by a recorded spot on the surface as it passes through a *read head*.

3.3.1 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.
- All disks rotate together at high speed and are not stopped or started for access purposes. 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. The subdivision of one disk surface into tracks and sectors is shown in Figure 5.

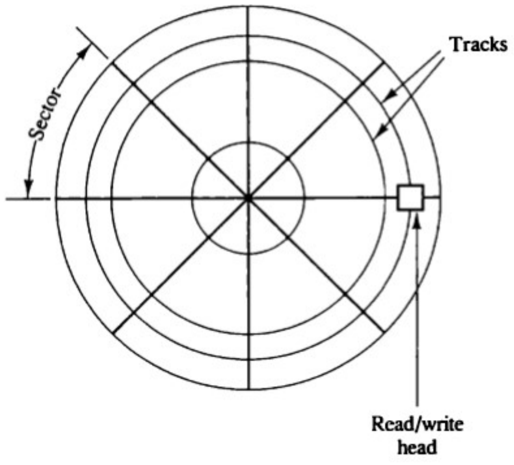


Figure 5: Magnetic disk

- 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*.

3.3.2 Magnetic Tape

- A magnetic tape transport consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic-tape unit.
- 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.
- Magnetic tape units can be stopped, started to move forward or in reverse, or can be rewind.

3.4 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 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. So 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.

3.4.1 Hardware Organization

- The block diagram of an associative memory is shown in Figure 6. It consists of a memory array and logic for *m* words with *n* bits per word.

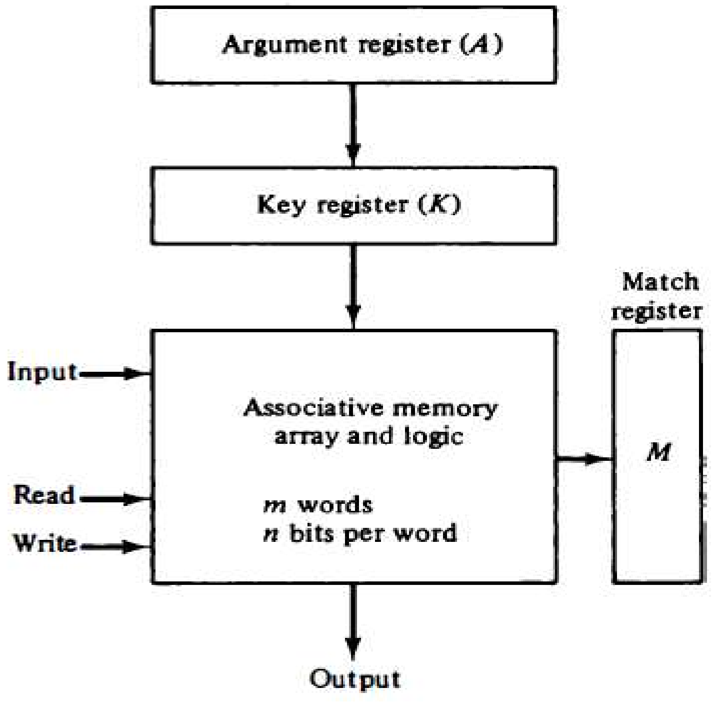


Figure 6: Block diagram of associative memory

- The argument register A and key register K each have *n* bits, one for each bit of a word. The match register M has *m* bits, one for each memory word.
- Each word in memory is compared in parallel with the content of the argument register. The words that match the bits of the argument register set a corresponding bit in the match register.
- After the matching process, those bits in the match register that have been set indicate that their corresponding words have been matched.
- Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.
- 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 l'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.
- For example the argument registers A and the key register K have the bit configuration shown below. Only the three leftmost bits of A are compared with memory words because K has 1's in these positions.

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

- Word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.
- The relation between the memory array and external registers in an associative memory is shown in Figure 7.
- 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.
- Bit A_j in the argument register is compared with all the bits in column j of the array provided that $K_j = 1$.
- 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.

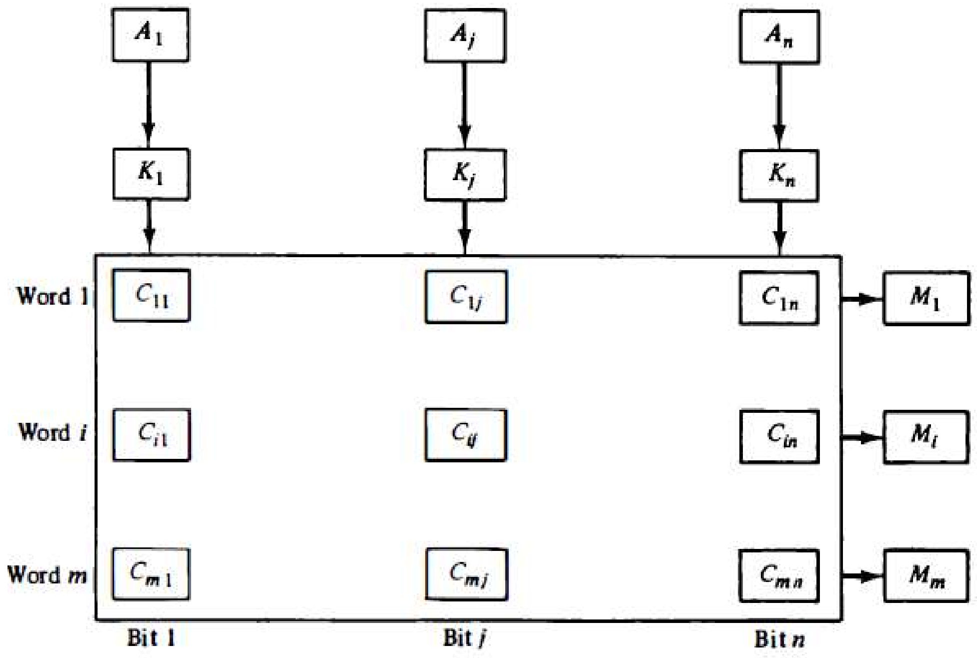


Figure 7: Associative memory of m word, n cells per word

- The internal organization of a typical cell C_{ij} is shown in Figure 8. It consists of a flip-flop storage element F_{ij} and the circuits for reading, writing, and matching the cell.

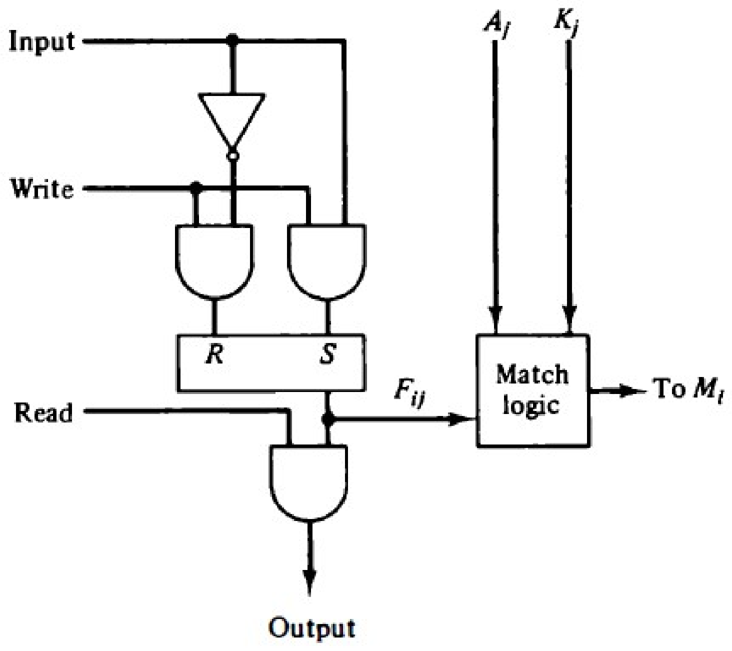


Figure 8: One cell of associative memory

- 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.

3.4.2 Match Logic

- The match logic for each word can be derived from the comparison algorithm for two binary numbers. 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 $A_i = F_{ij}$ 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 $x_i = 1$ if the pair of bits in position j are equal; otherwise, $x_i = 0$.
- 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

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

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

- 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

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

- The circuit for matching one word is shown in figure 9. Each cell requires two AND gates and one OR gate.

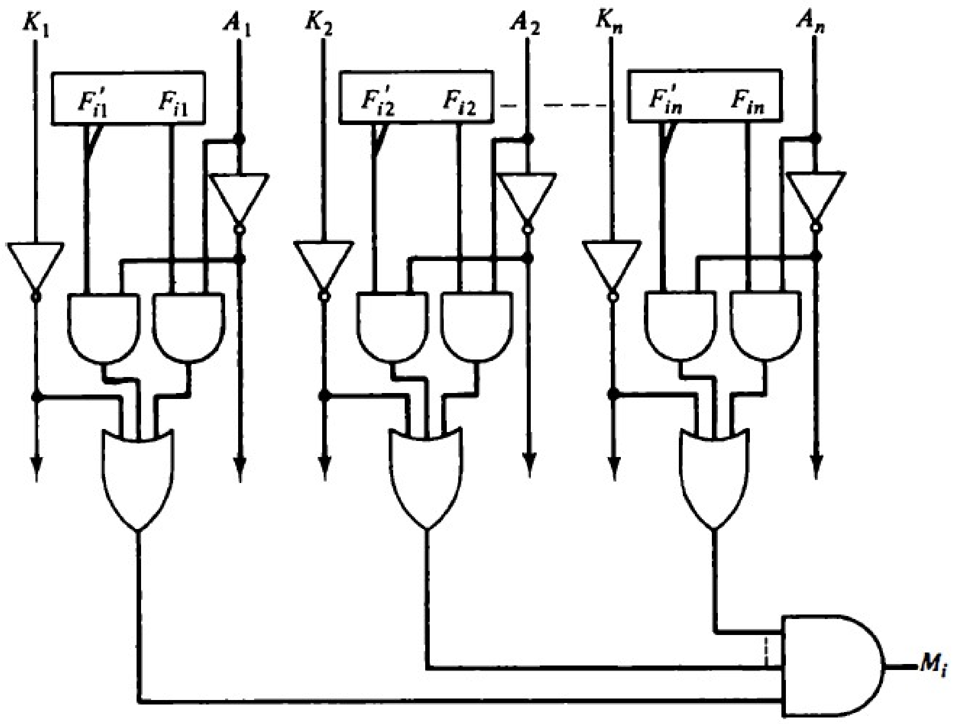


Figure 9: Match logic for one word of associative memory

- The inverters for A_j and K_j are needed once for each column and are used for all bits in the column. The output of all OR gates in the cells of the same word go to the input of a common AND gate to generate the match signal for M_i.
- M_i will be logic 1 if a match occurs and 0 if no match occurs.

3.4.3 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.
- 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.

3.4.4 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.

- The advantage here is that the address for input can be decoded as in a random access memory. Thus instead of having m address lines, one for each word in memory, the number of address lines can be reduced by the decoder to d lines, where $m = 2^d$.

3.5 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.
- Cache memory is placed between the CPU and main memory as illustrated in Figure 1. 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.
- The fundamental idea of cache organization is to reduce the speed mismatch between CPU and main memory.
- The basic operation of the cache is, when the CPU needs to access memory, the cache is examined. If the required data is found in the cache, it is read from it.
- If the required data by the CPU is not found in the cache, then the main memory is accessed to read the required data, just accessed data is then transferred from main memory to cache memory.
- 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 required data in cache, it is said to produce a *hit*. If the required data is not found in cache, it is in main memory and it counts as a *miss*.

$$Hit\ ratio = \frac{number\ of\ hits}{number\ of\ hits + number\ of\ miss}$$

- The transformation of data from main memory to cache memory is referred to as a *mapping* process.
- Three types of *mapping techniques* for cache memory as follows.
  1. Associative mapping
  2. Direct mapping
  3. Set-associative mapping
- For these three mapping procedures one example of a memory organization as shown in following Figure 10.

- The main memory can store 32K words of 12 bits each. i.e. 32K×12 of main memory.
- The cache is capable of storing 512 of these words of 12 bits each. i.e 512×12 of cache memory.

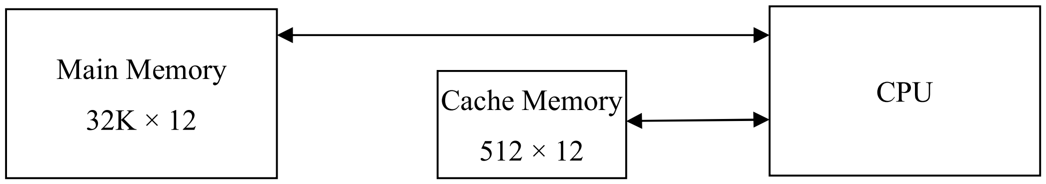
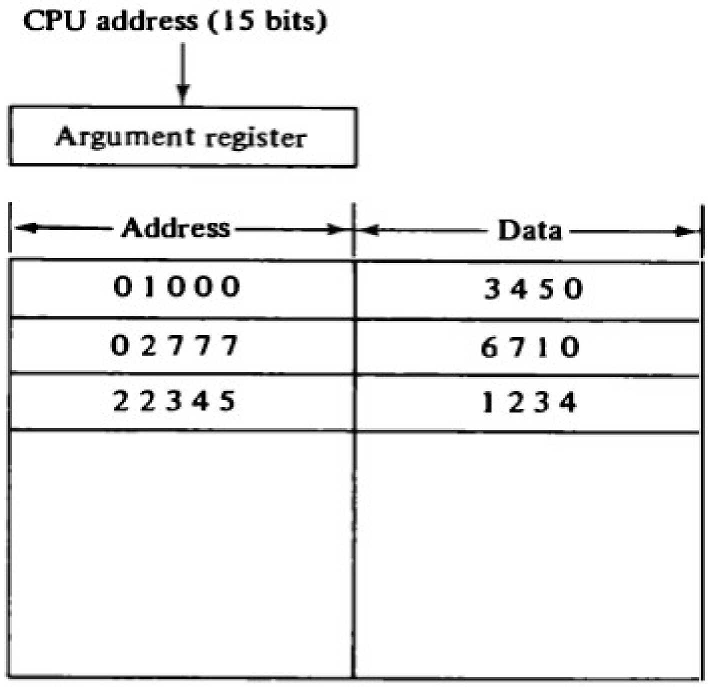


Figure 10: Example of cache memory

- For every word stored in cache, there is duplicate copy in main memory.

3.5.1 Associative Mapping

- The fastest and most flexible cache organization uses an associative memory. This organization is illustrated in Figure 11.
- The associative memory stores both the address and content (data) of the memory word.
- For example Main Memory → 32K×12 → 2^{15}× 12 ➔ 15 Address lines and 12 data lines.
- Cache Memory→ 512 × 12 → 2^{9}× 12 ➔ 9 Address lines and 12 data lines.
- The address value of 15 bits is shown as a five-digit octal number and its corresponding 12 -bit word is shown as a four-digit octal number.



Address	Data
01000	3450
02777	6710
22345	1234

Figure 11: Associative mapping cache (all numbers in octal)

- A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.
- If the address is found, the corresponding 12-bit data is read and sent to the CPU.
- If the address is not matched, then CPU accesses the main memory for the data. The address and data pair is then transferred to the associative cache memory.
- If the cache is full then replacement is occur in cache memory.

3.5.2 Direct Mapping

- Associative memories are expensive compared to random-access memories because of the added logic associated with each cell.
- Here the CPU address bits are divided into two fields named as *tag* field and *index* field.
- The number of bits in the index field is equal to the number of address bits required to access the cache memory.
- The number of bits for tag field is the number of bits required to represent the address of Main Memory - number of bits required to represent the address of Cache Memory.
- For example: MM → 32K × 12 ➔ 15 *address lines* and 12 *data lines*

  CM → 512 × 12 ➔ 9 *address lines* and 12 *data lines*
- So CPU address of 15 bits is divided into two fields. The 9 least significant bits constitute the *index* field and the remaining 6 bits form the *tag* field.
- 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 as *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.

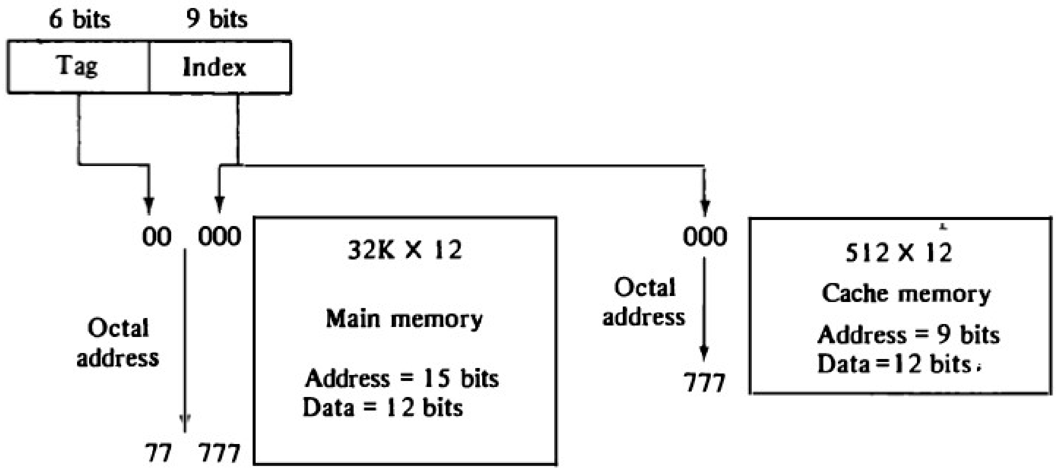


Figure 12: Addressing relationships between main and cache memories

- The internal organization of the words in the cache memory is as shown in figure 13(b).

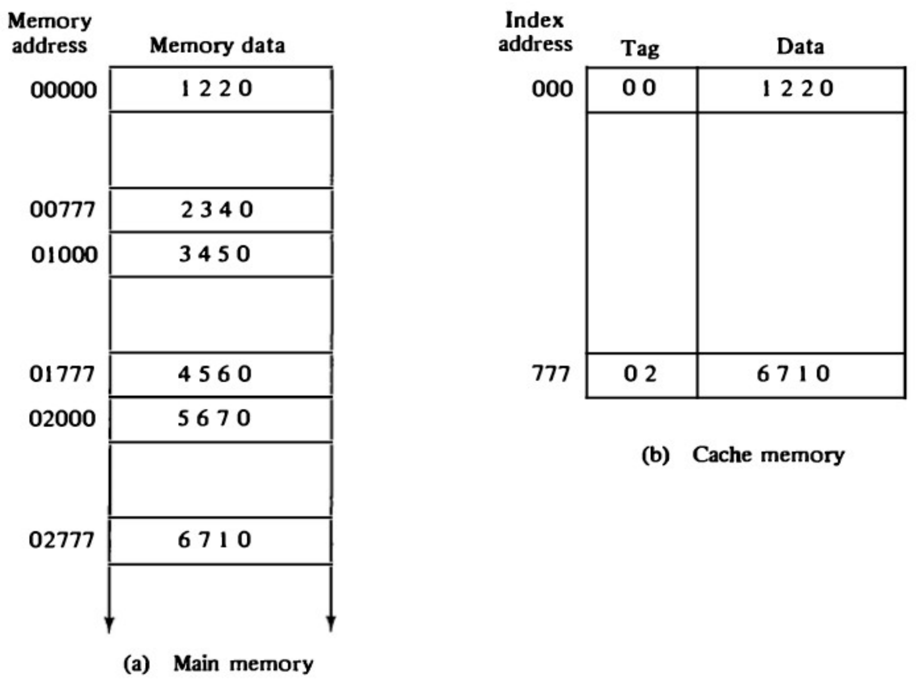


Figure 13: Direct mapping cache organization

- Each word in cache consists of the data word and its associated tag. When a new word is first brought into the cache, the tag bits are stored along the data bits.
- When the CPU generates a memory request, the index field is used for the address to access the cache.
- 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.

- Eg: Suppose that the CPU now wants to access the word at address 02777. So the index value is 777 and tag value is 02.
- The index value 777 is used as address to access the cache memory. In the above figure 13, index address 777 is available, and then tag value 02 is compared with tag value in the cache memory associated with index address 777.
- Here tag value of CPU address is match with tag value of cache with index address also. So desired data is available in cache memory. So it is hit operation.

→ 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.

- Eg: Suppose that the CPU now wants to access the word at address 01777. So the index value is 777 and tag value is 01.
- The index value 777 is used as address to access the cache memory. In the above figure 13, index address 777 is available, and then tag value 01 is compared with tag value in the cache memory associated with index address 777.
- Here tag value of CPU address 01 is not match with tag value of cache memory is 02. So desired data was not available in cache memory. So it is *miss* operation.
- Whenever a miss operation was occur, 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.

3.5.3 Set-Associative Mapping

- To overcome the disadvantage of direct mapping, i.e. two words with the same index in their address but with different tag values cannot reside in cache memory at the same time, Set-associative mapping is proposed.
- Here each word of cache can store two or more words stored together with its tag.
- An example of a set-associative cache organization for a set size of two is shown in Figure 14.

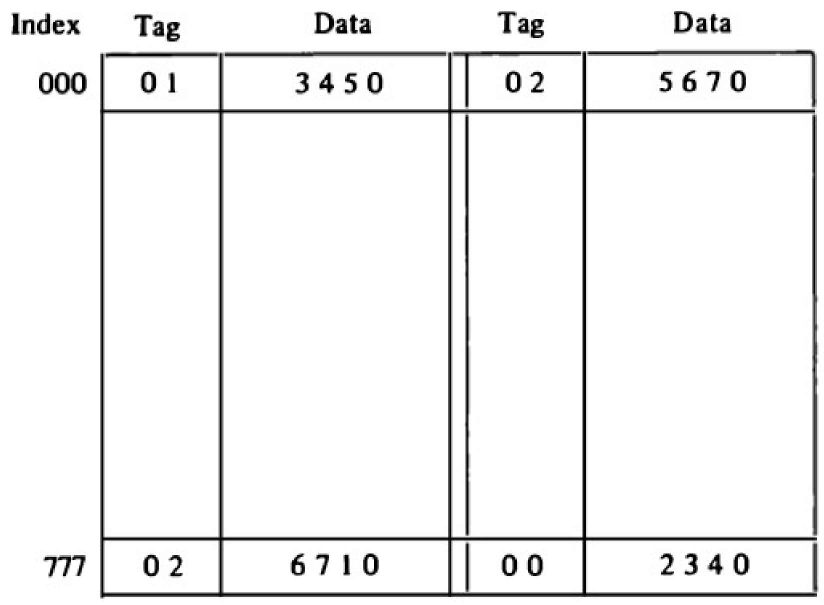


Figure 14: Two way set-associative mapping cache

- In the above figure 14, each index address refers to two data words and their associated tags.
- Here each tag requires 6 bits, i.e. 2 octal digits and 12 bits for data, i.e. 4 octal digits.
- In the above figure 14, the words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000 with different tag values 01 and 02 respectively.
- Similarly, the words at addresses 02777 and 00777 of main memory are stored in cache memory at index address 777 with different tag values 01 and 02 respectively.
- 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. If tags are match then it is a *hit* operation.
- If the tag field of the CPU address and tags in the cache are not matched then it is a *miss* operation.
- 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.

3.5.4 Writing into Cache

- An important aspect of cache organization is concerned with memory write requests.
- There are two ways for writing into Cache.
- 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. This method has the advantage that main memory always contains the same data as the cache.

- 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 updated into main memory.

3.5.5 Cache Initialization

- The cache is initialized when power is applied to the computer or when the main memory is loaded with a complete set of programs from auxiliary memory.
- After initialization the cache is considered to be empty, but in effect it contains some non-valid data. 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.

3.6 VIRTUAL MEMORY

- In memory hierarchy system, programs and data first stored in Auxiliary memory.
- Position of data or programs are brought into main memory as they needed by the CPU.
- Virtual memory is a concept used in some large computers that permit the users to construct programs as though a large memory space were available, equal to the totality of Auxiliary memory.
- Virtual memory is used to give programmers as the illusion that they have large memory even though computer actually has a relatively small main memory.
- A virtual memory system provides a mechanism for translating program generated address into correct main memory location.
- Address space: An address used by a programmer will be called a virtual address, and the set of virtual address called as address space.
- Memory space: An address in main memory (physical memory) is called a location (or) physical address, and set of physical addresses is called as memory space.
- The address space allowed is larger than the memory space in computer with virtual memory.