CHAPTER 8
MEMORY ORGANIZATION
BIM 3rd Semester

Contents

Memory Organization LH 6

8.1 Memory Hierarchy

8.2 Main Memory

8.3 Associative Memory

8.4 Cache Memory

8.5 Virtual Memory

8.6 Memory Management Hardware

Memory Hierarchy

- Memory unit is essential component of digital computer since it is needed for storing programs and data.
- Memory unit that communicates directly with CPU is called Main memory.
- Devices that provide backup storage is called auxiliary memory.
- Only programs and data currently needed by processor reside in the main memory.
- All other information is stored in auxiliary memory and transferred to main memory when needed.

Table 4.1 Key Characteristics of Computer Memory Systems

Location

- Internal (e.g. processor registers, main memory, cache)
- External (e.g. optical disks, magnetic disks, tapes)

Capacity

- Number of words
- Number of bytes

Unit of Transfer

- Word
- Block

Access Method

- Sequential
- Direct
- Random
- Associative

Performance

- Access time
- Cycle time
- Transfer rate

Physical Type

- Semiconductor
- Magnetic
- Optical
- Magneto-optical

Physical Characteristics

- Volatile/nonvolatile
- Erasable/nonerasable

Organization

- Memory modules

- Memory hierarchy system consist of all storage devices from auxiliary memory to main memory to cache memory
- As one goes down the hierarchy :
  - Cost per bit decreases.
  - Capacity increases.
  - Access time increases.
  - Frequency of access by the processor decreases.

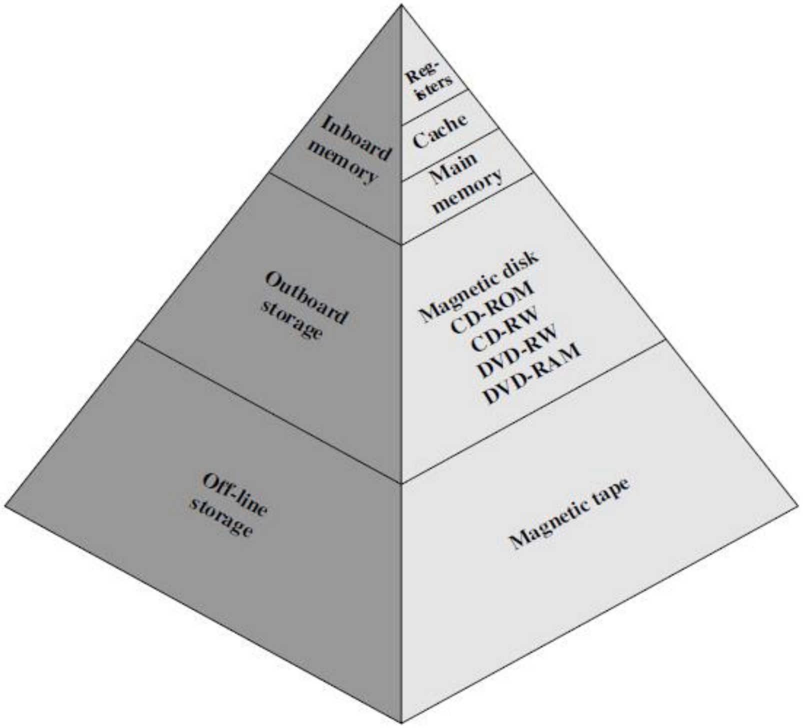


Figure 4.1 The Memory Hierarchy

Primary and Secondary Memory

To identify the behavior of various memories certain characteristics are considered. These are as follows-

Memory types : On the basis of their location inside the computer, memory can be placed in four groups :

CPU Registers: these high speed registers in the CPU work as memory for temporary storage of instruction and data. The data can be read from or written into a register within a single clock cycle.

Main Memory or Primary Memory: Main memory size is large and fast accessing external memory stores programs and data. This memory is slower compared to CPU registers because of main memory has large storage capacity is typically 1 and 2^{10} megabyte.

- **Secondary Memory**: This memory has larger in capacity but slower than main memory. Secondary memory stores system programs, large data files and like the data are not continually required by the CPU. It also acts as an overflow memory when the capacity of the main memory is exceeded. Information in secondary storage is accessed indirectly via input output processor that transfer information between main and secondary memory.

Auxiliary Memory

- Also called as Secondary Memory, used to store large chunks of data at a lesser cost per byte than a primary memory for backup.
- It does not lose the data when the device is powered down—it is non-volatile.
- It is not directly accessible by the CPU, they are accessed via the input/output channels.
- The most common form of auxiliary memory devices used in consumer systems is flash memory, optical discs, and magnetic disks, magnetic tapes.

Types of Auxiliary Memory

- *Flash memory*: An electronic non-volatile computer storage device that can be electrically erased and reprogrammed, and works without any moving parts. Examples of this are **USB flash drives** and **solid state drives**.

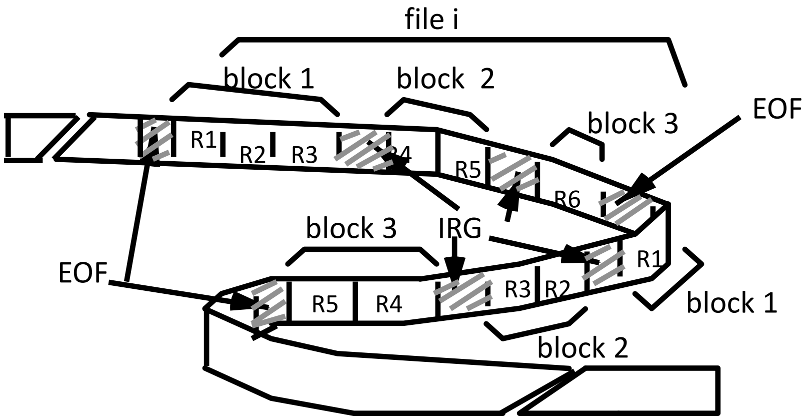
- *Optical disc*: Its a storage medium from which data is read and to which it is written by lasers. There are three basic types of optical disks: CD-ROM (read-only), WORM (write-once read-many) & EO (erasable optical disks).

Types of Auxiliary Memory

- *Magnetic tapes*: A magnetic tape consists of electric, 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 tape along several tracks called **RECORDS.**

- Each record on tape has an identification bit pattern at the beg. and the end.



- R/W heads are mounted in each track so that data can be recorded and read as a sequence of characters.
- Can be stopped , started to move forward, or in reverse, or can be rewound, but cannot be stopped fast enough between individual characters.

Types of Auxiliary Memory

Magnetic Disk:

- A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.
- 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 magnetized surface in spots along concentric circles called tracks. Tracks are commonly divided into sections called sectors.
- Disk that are permanently attached and cannot removed by occasional user are called hard disks.

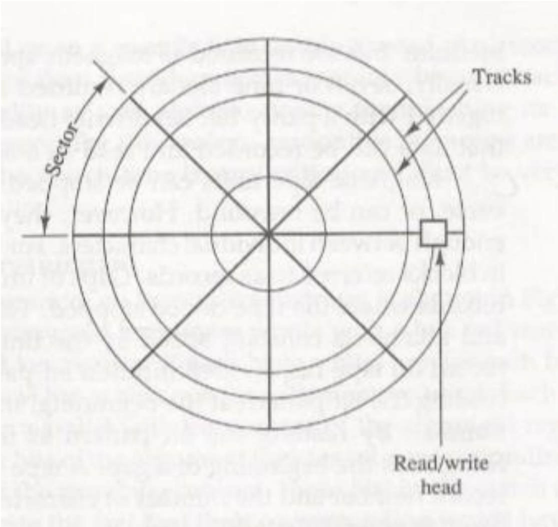
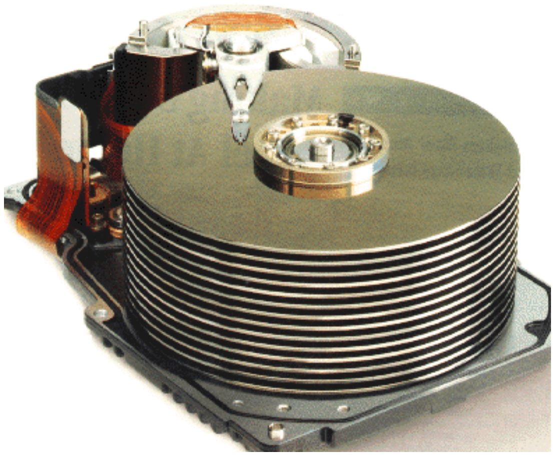


Figure 12-5 Magnetic disk.



From Computer Desktop Encyclopedia
Reproduced with permission.
© 1997 Singapore Technologies

Continued…

- **Cache Memory**: Most computers have another level of IC memory called cache memory. It is placed between CPU registers and main memory. A cache memory capacity is less than that of main memory but it is faster than that of main memory because some or all of it can reside on the same IC as the CPU. Cache memories are essential components of high performance computers.
- **Location**: The memory which is inside the processor called the internal memory. The memory which is external to the processor is known as external memory.
- **Access Method**: Each memory is a collection of various memory location. Accessing the memory means finding and reaching desired location and than reading information from memory location. The information from locations can be accessed as follows:

Continued…

1. Random access
2. Sequential access
3. Direct access.

- **Random Access**: It is the access mode where each memory location has a unique address. Using these unique addresses each memory location can be addressed independently in any order in equal amount of time. Generally, main memories are random access memories.
- **Sequential Access**: If storage locations can be accessed only in a certain predetermined sequence, the access method is known as serial or sequential access.
- **Direct Access**: In this access information is stored on tracks and each track has a separate read/write head. This features makes it a semi random mode which is generally used in magnetic disks.

Continued…

- **Volatile Memories**: The memories that looses their contents when the power is turned off called volatile memories.
- **Non-volatile Memories**: The memories that do not loose their contents when the power is removed called Non-volatile memories.

MEMORY CLASSIFICATION

In general the memory is classified in two types based on their mode of access of a memory system.

1. Random access memory
2. Sequential access memory

- **Random Access Memory**: The world of data reading or writing from or to the memory requires same time. We can access the data randomly.

  Example: hard disk.

- **Sequential Access Memory**: The information stored in some medium is not immediately accessible but is available as certain intervals of time. The access time is variable.

  Example: magnetic tape.

TYPES OF RAM

- **Static RAM**: It consist of internal latches that store the binary information. The stored information remains valid as long as power is applied to the unit.
- **Dynamic RAM**: It stores the binary information in the form of electric charges on capacitors. The capacitors are provided inside the chip by MOS transistors. The stored charge on the capacitor tends to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory.

ROM & THEIR TYPES

- **Read only memory**: It is non-volatile memory, which retains the data even when power is removed from this memory. Programs and data that can not be altered are stored in ROM. The required paths in a ROM may be programmed in four different ways:

1. **Mask Programming**: It is done by the company during the fabrication process of the unit. The procedure for fabricating a ROM requires that the customer fills out the truth table he wishes the ROM to satisfy.

2.**Programmable Read only memory(PROM):** PROM contain all the fuses intact giving all 1's in the bits of the stored words. A blown fuse defines binary 0 state and an intact fuse give a binary 1 state. This allows the user to program the PROM by using a special instruments called PROM programmer.

Continued…

3. **Erasable PROM(EPROM):** In a PROM once fixed pattern is permanent and can not be altered. The EPROM can be restructured to the initial state even through it has been programmed previously. When EPROM is placed under a special ultra-violet light for a given period of time all the data are erased. After erase, the EPROM returns to it initial state and can be programmed to a new set of values.

4. **Electrically Erasable PROM(EEPROM):** It is similar to EPROM except that the previously programmed connections can be erased with an electrical signal instead of ultra violet light. The advantage is that device can be erased without removing it from it's socket.

Main Memory

- It is the memory used to store programs and data during the computer operation.
- The principal technology is based on semiconductor integrated circuits.
- It consists of RAM and ROM chips.
- RAM chips are available in two form static and dynamic.

SRAM	DRAM
Uses capacitor for storing information	Uses Flip flop
More cells per unit area due to smaller cell size.	Needs more space for same capacity
Cheap and smaller in size	Expensive and bigger in size
Slower and analog device	Faster and digital device
Requires refresh circuit	No need
Used in main memory	Used in cache

- ROM is uses random access method.
- It is used for storing programs that are permanent and the tables of constants that do not change.
- ROM store program called bootstrap loader whose function is to start the computer software when the power is turned on.
- When the power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader.

Bootstrap Loader:

- Bootstrap loader is a program that is stored in ROM and is used to start the loading of OS from hard disk to RAM when power is turned on in a computer. When a computer is turned on hardware of the computer sets PC (program counter) to first instruction of bootstrap loader so that execution of bootstrap loader begins when power is turned on.

RAM and ROM chips

- RAM and ROM chips are available in a variety of sizes. If we need larger memory for the system, it is necessary to combine a number of chips to form the required memory size.

RAM Chips

- A RAM chip is better suited to communicate with CPU if it has one or more control inputs that select the chip only when needed. The block diagram of a RAM chip is shown below:

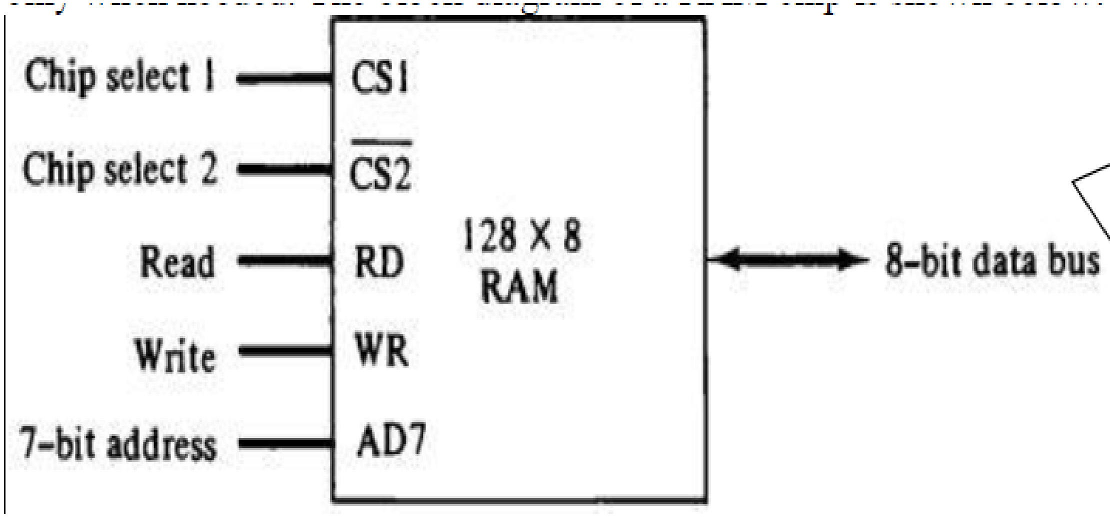
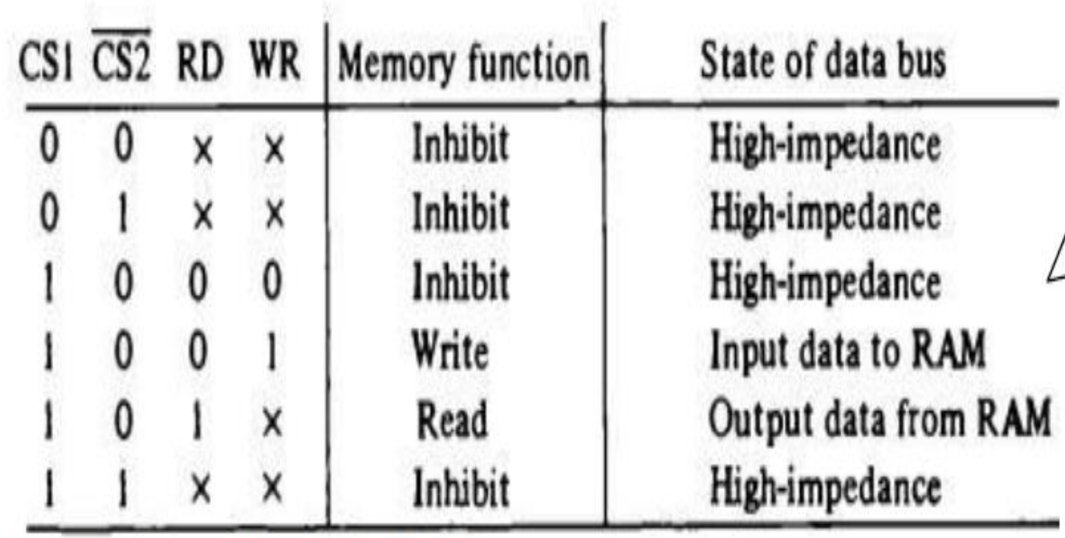


Fig: Typical RAM chip (128 words of eight bits each)

→ Requires 7-bit address and an 8-bit bidirectional data bus

→ Chip select (CS) control inputs are for enabling the chip only when it is selected by CPU.



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

Fig: Function table for RAM chip

→ The unit is in operation only when CS1=1 and (CS2)'=0.

→ High impedance state indicates open circuit i.e. output does not carry a signal and has no logic significance.

ROM Chips

- Since a ROM chip can only read, data bus is unidirectional (output mode only).

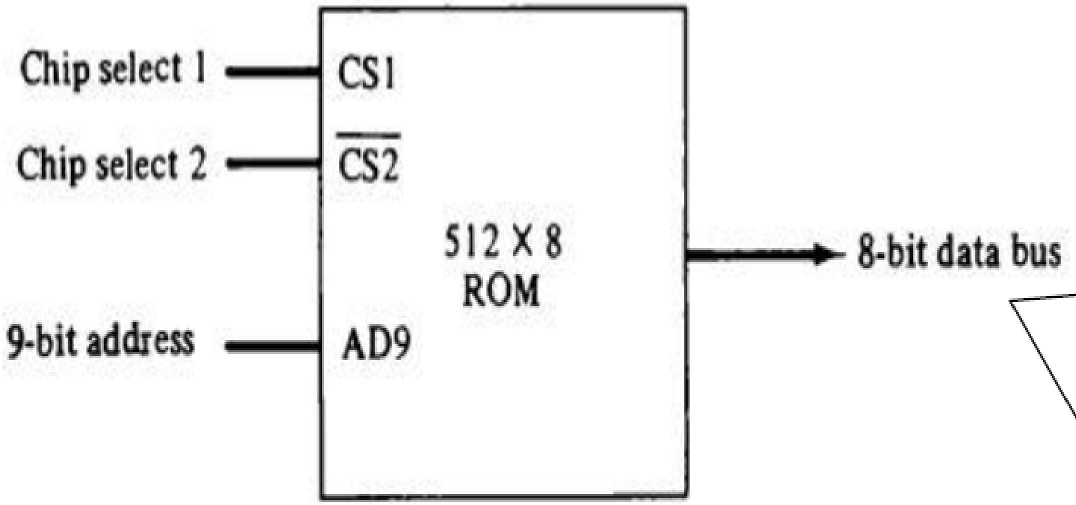


Fig: Typical ROM chip (512 byte ROM)

→ 9 address lines to address 512 bytes

→ Two chip select (CS) inputs CS1=1 and (CS2)’=0 for the unit to operate, otherwise the data bus is in high-impedance state.

→ No need for read or write control since the unit can only read.

Memory Address Map

- The addressing of memory can be established by means of a table that specifies the memory address assigned to each RAM or ROM chip. This table is called memory address map and is a pictorial representation of assigned address space for particular chip.

Example: Suppose computer system needs 512 bytes of RAM and 512 bytes of ROM.

Solution,

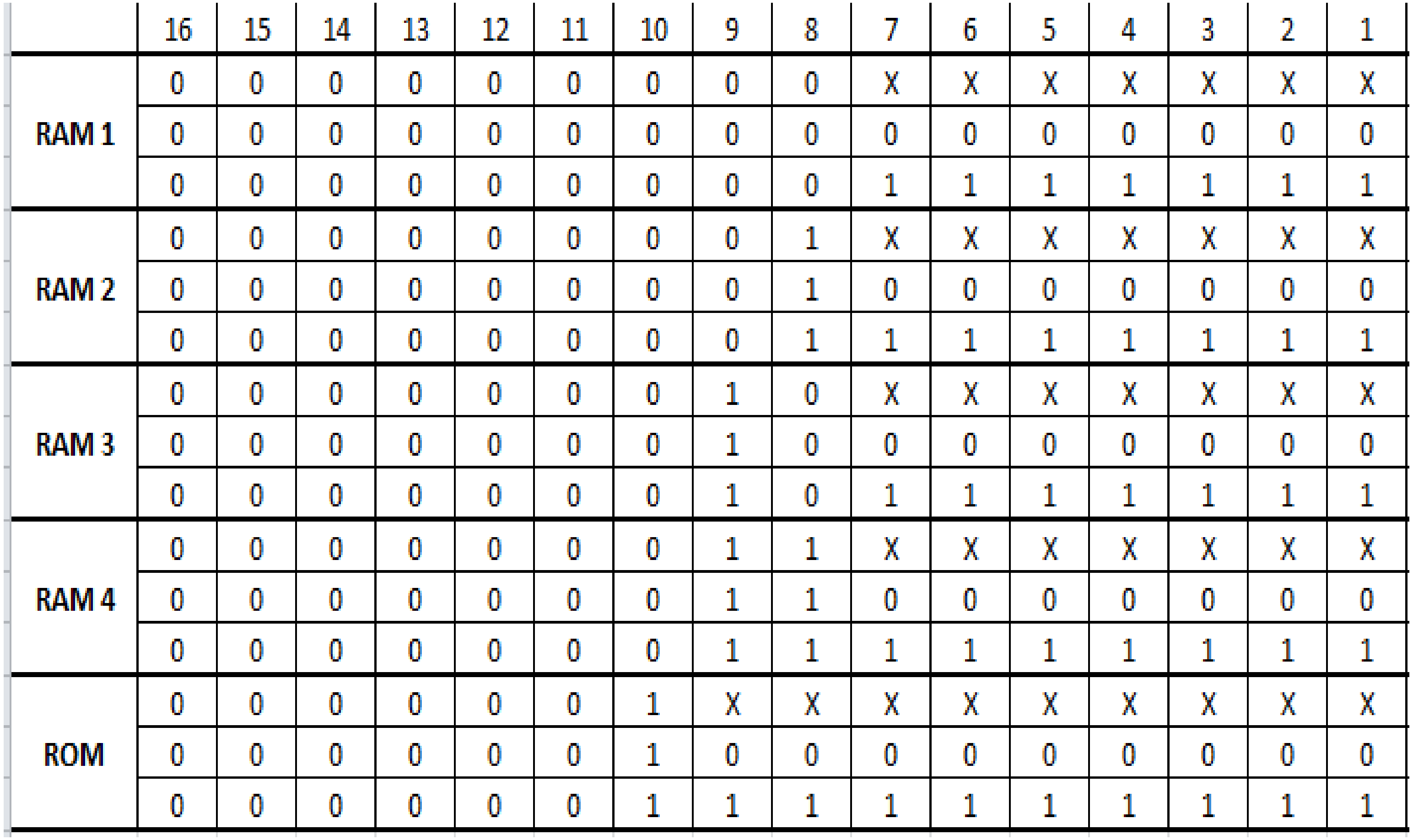
We have a chip size of RAM =128=2^7

We have a chip size of ROM = 512=2^9

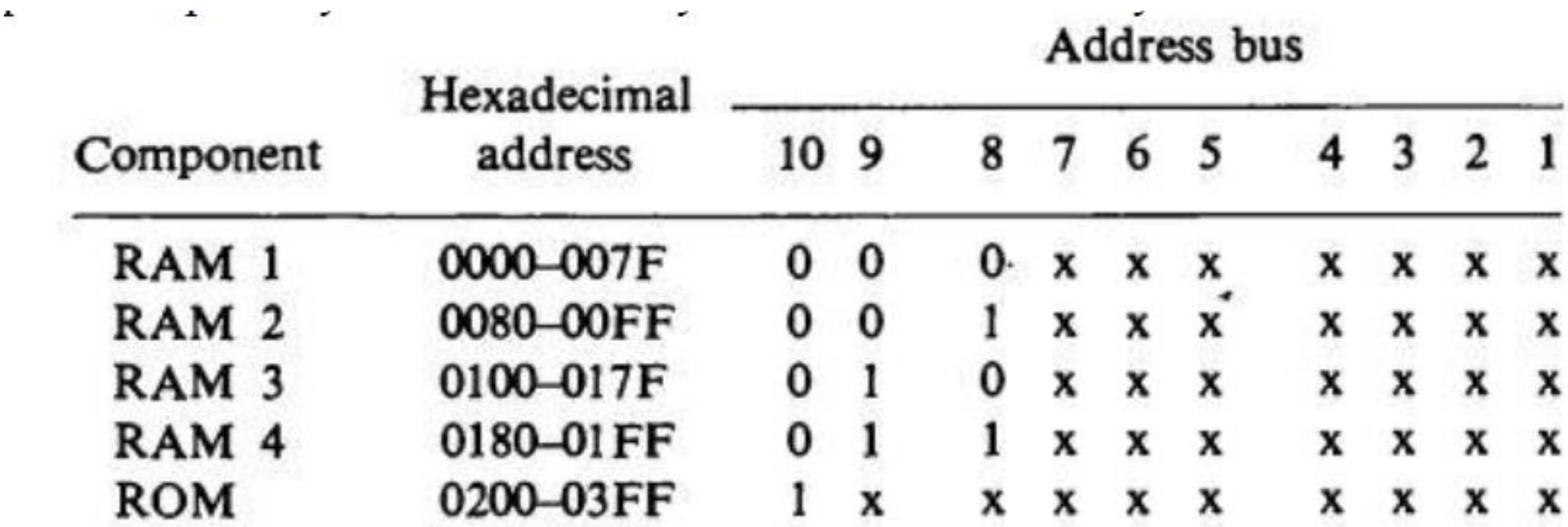
So, total RAM required = 512/128 = 4

So, total ROM required = 512/512 =1

Therefore, 4 RAM and 1 ROM is required.



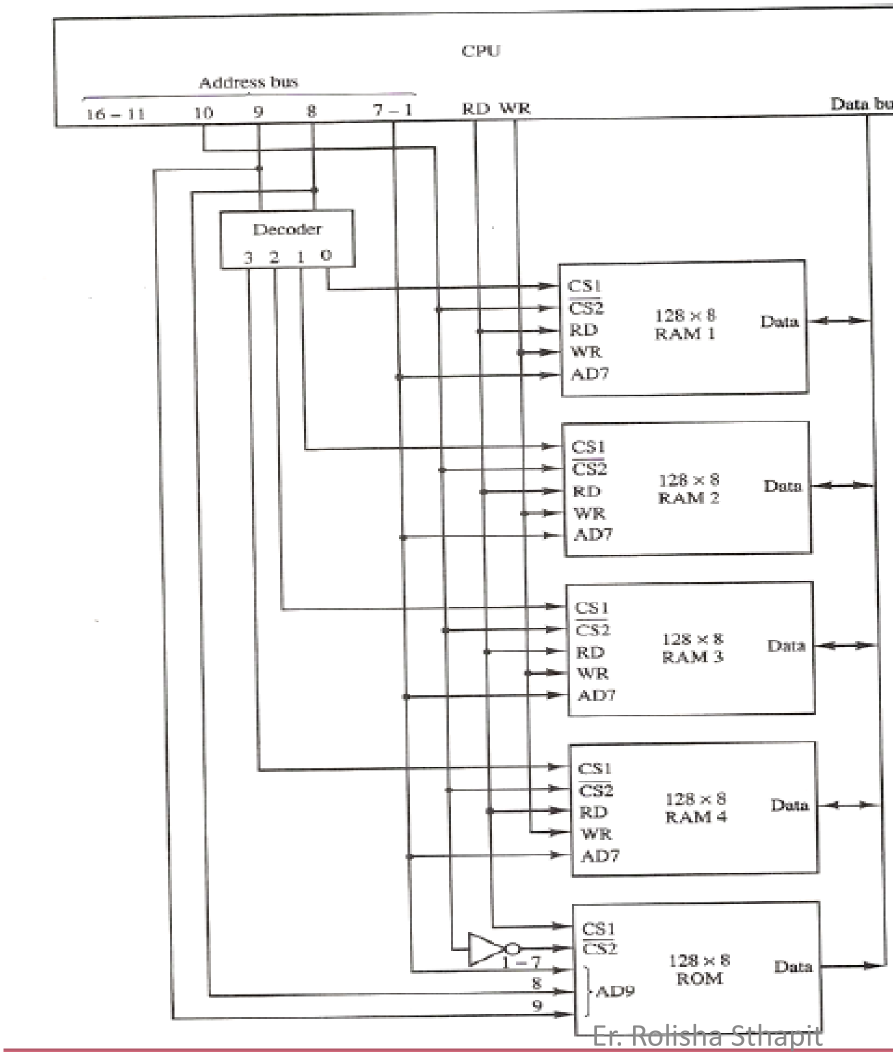
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RAM 1	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
RAM 2	0	0	0	0	0	0	0	0	1	X	X	X	X	X	X	X
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
RAM 3	0	0	0	0	0	0	0	1	0	X	X	X	X	X	X	X
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1
RAM 4	0	0	0	0	0	0	0	1	1	X	X	X	X	X	X	X
	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
ROM	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	X
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1



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-CPU Connection

RAM and ROM chips are connected to CPU through data and address buses.



Example gives an indication of the interconnection complexity that can exist between memory chips and CPU. More the chips, more external decoders are required for selection among the chips.

- ➔ This configuration gives 512 bytes of RAM and 5112 bytes of ROM
- ➔ Each RAM receives 7 low-order bits of the address bus to select a byte.
- ➔ RAM chips are selected with decoder with selection input of line 8 and 9.
- ➔ The selection between RAM and ROM is done by line 10. When 0, RAMs are selected and when 1 ROM get selected.

ROM is 512*8

Q. Design a system memory of 2048 words that includes 1024 words of RAM and 1024 word of ROM. Use RAM chip of 256 words and ROM chip of 1024 words.

Solution,

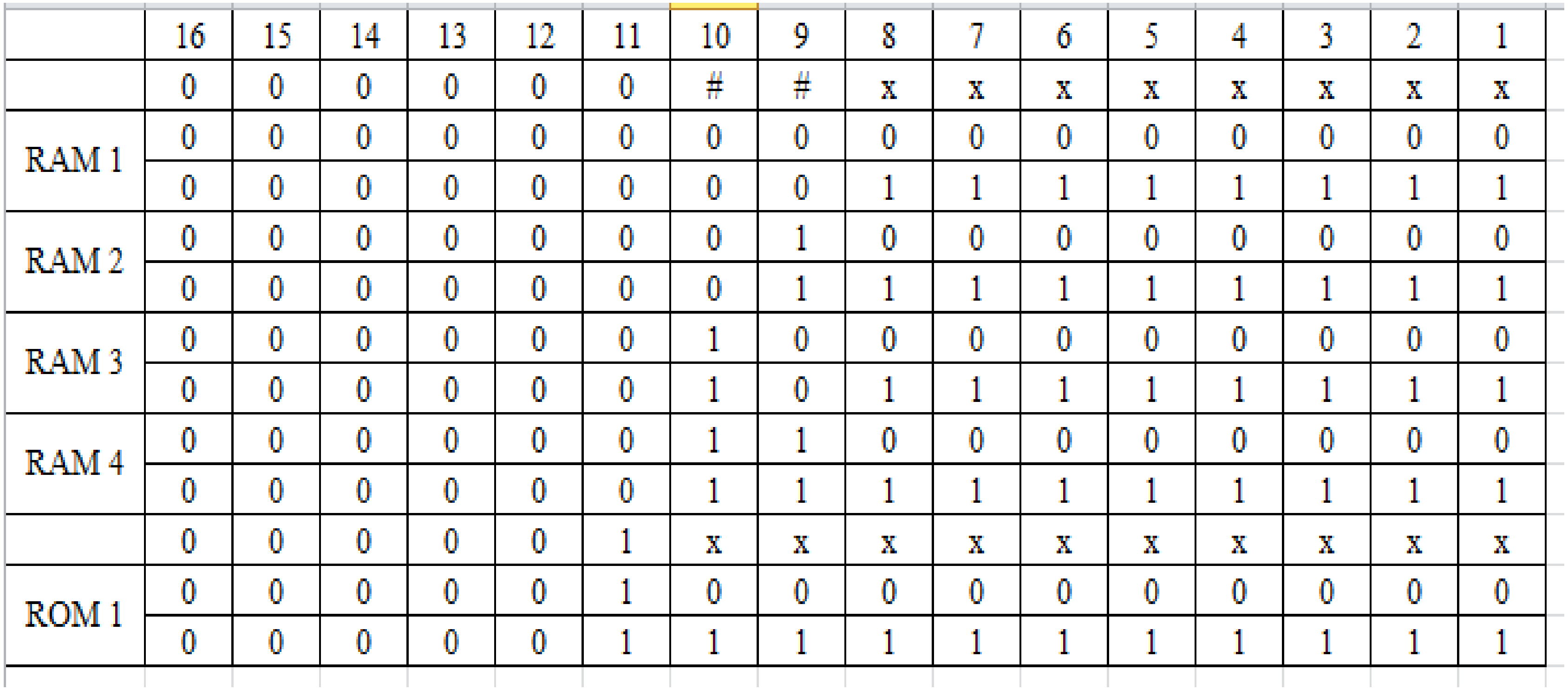
We have a chip size of RAM =256=2^8

We have a chip size of ROM = 1024=2^10

So, total RAM required = 1024/256 = 4

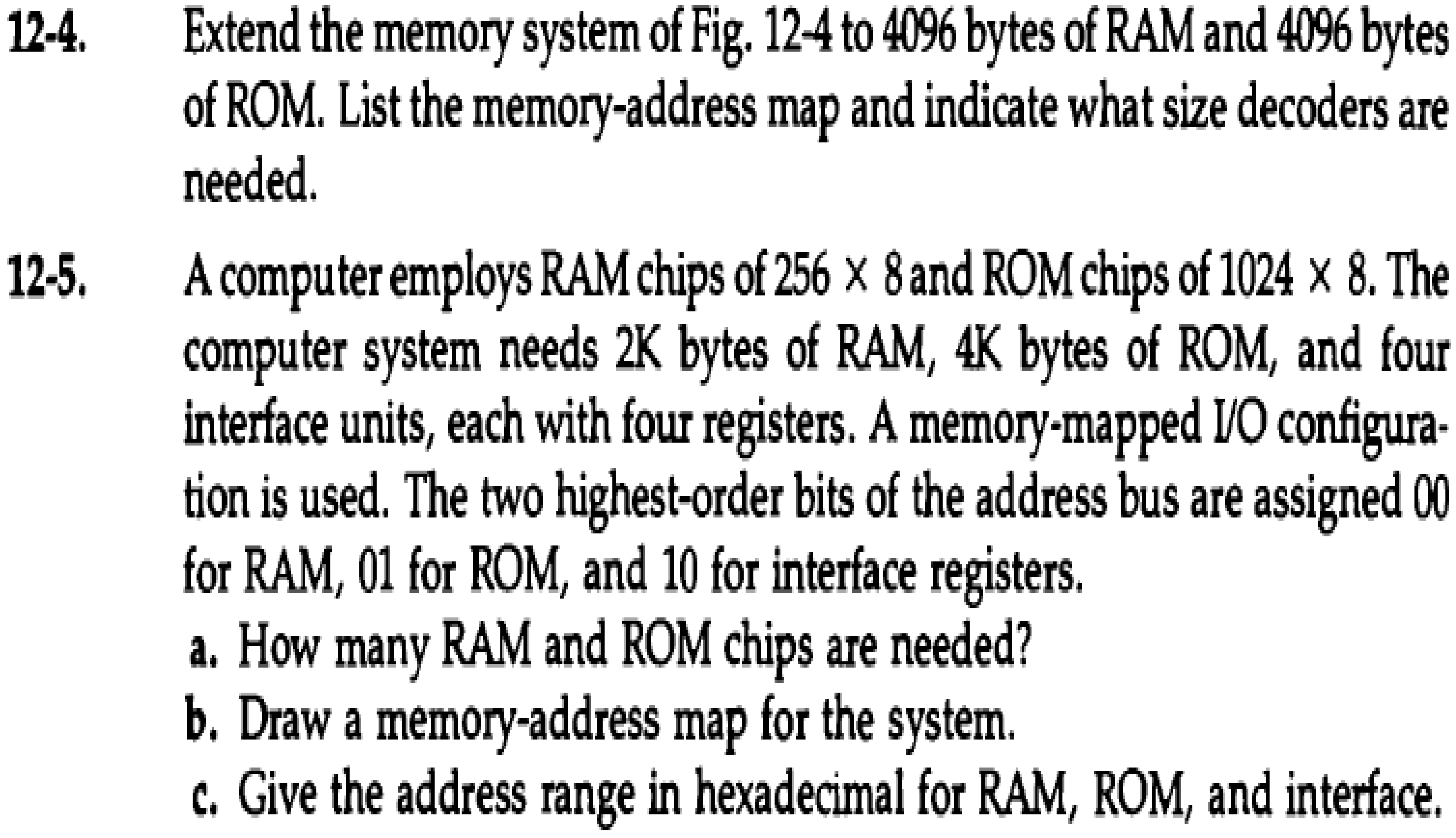
So, total ROM required = 1024/1024 =1

Therefore, 4 RAM and 1 ROM is required.



	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	0	0	0	0	0	0	#	#	x	x	x	x	x	x	x	x
RAM 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
RAM 2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
RAM 3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1
RAM 4	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	1	x	x	x	x	x	x	x	x	x	x
ROM 1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

ASSIGNMENT



12-4. Extend the memory system of Fig. 12-4 to 4096 bytes of RAM and 4096 bytes of ROM. List the memory-address map and indicate what size decoders are needed.

12-5. A computer employs RAM chips of 256 × 8 and ROM chips of 1024 × 8. The computer system needs 2K bytes of RAM, 4K bytes of ROM, and four interface units, each with four registers. A memory-mapped I/O configuration is used. The two highest-order bits of the address bus are assigned 00 for RAM, 01 for ROM, and 10 for interface registers.

a. How many RAM and ROM chips are needed?

b. Draw a memory-address map for the system.

c. Give the address range in hexadecimal for RAM, ROM, and interface.

Associative Memory

- The time require to find an item store in the memory can be reduced considerably if the stored data can be identified for excess by the content of data itself rather than by address.
- A memory unit accessed by the 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 then by specific address or location.
- When a word is written in a associative memory, no address is given.
- Memory is capable of finding empty unused location to store the word.
- When a word is to be read from an associative memory, the content of the word or the part of the word is specified.
- The memory locates all words which match the specified content and marks them for reading.
- An associative memory is more expensive than RAM, as each cell must have storage capability as well as logic circuits for matching its content with an external argument.
- Associative memories are used in applications where the search time is very critical and short.

Hardware Organization

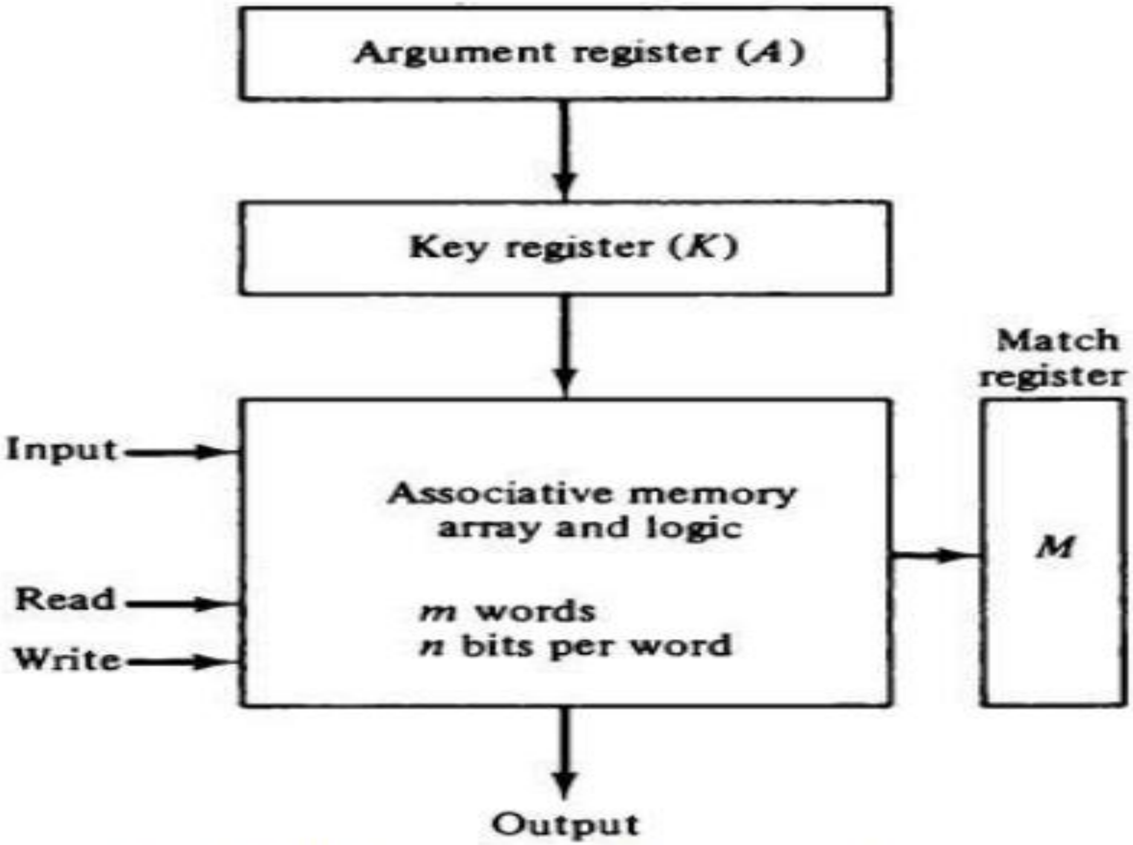
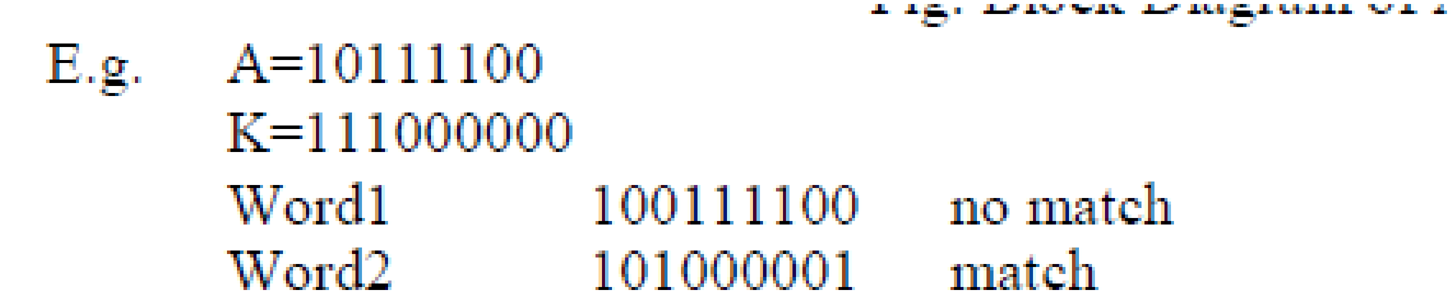


Fig: Block Diagram of Associative Memory

- Associative memory consists of a memory array and logic for m words and n bits per word.
- 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 words. 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 matching process, those bits in the match register that have been set indicate the fact 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 1’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.

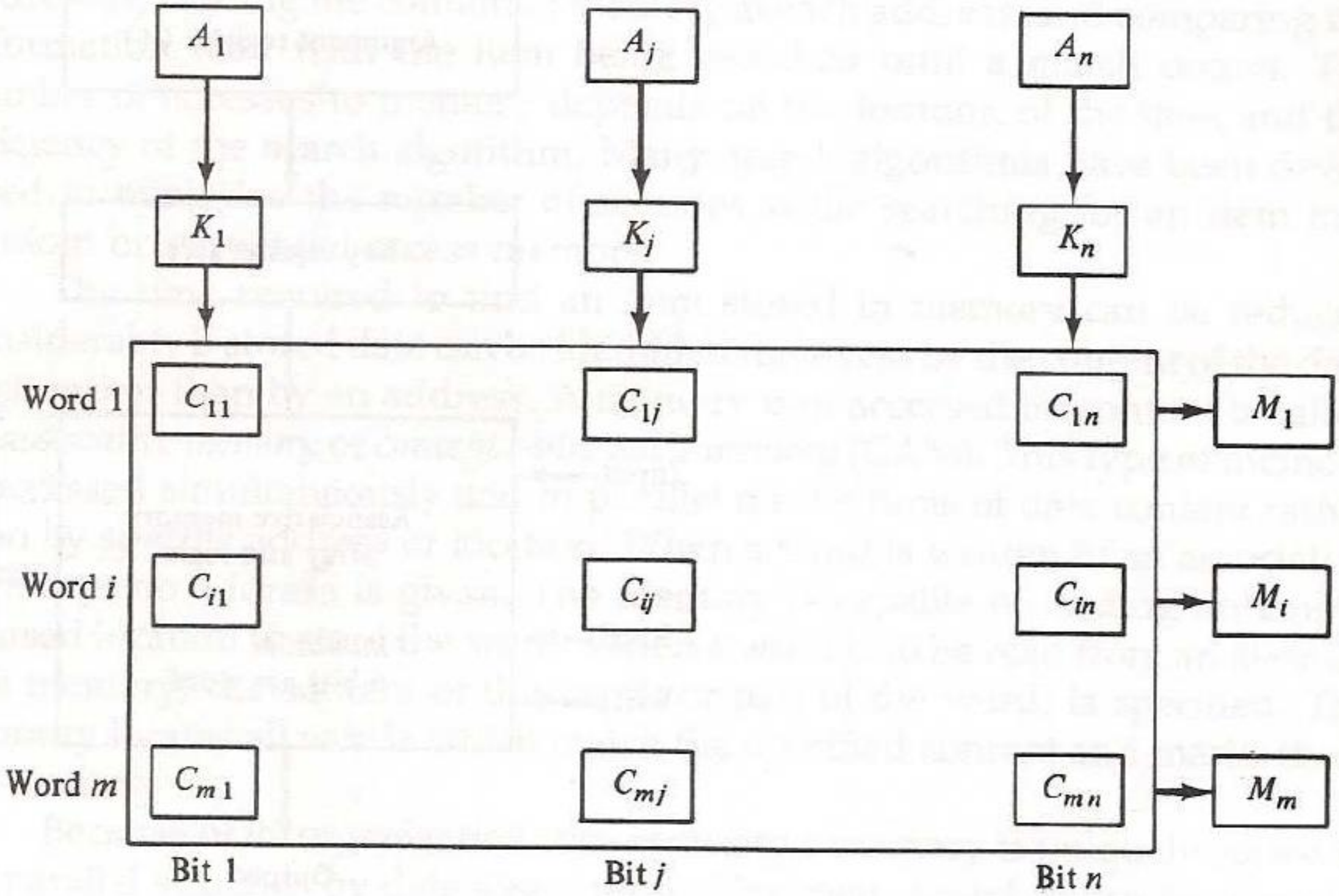


E.g. A=10111100

K=111000000

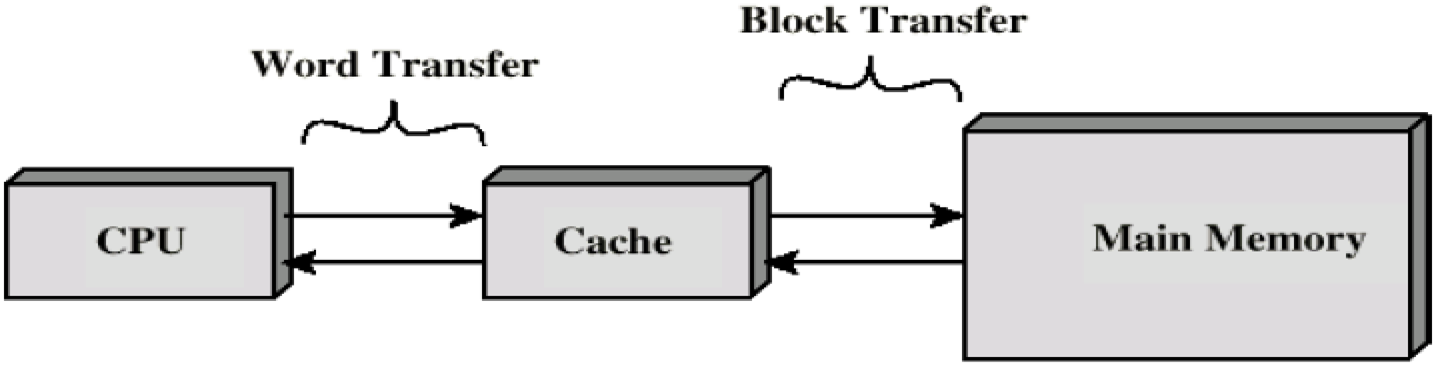
Word1	100111100	no match
Word2	101000001	match

Figure Associative memory of m word, n cells per word.



Cache Memory

- Cache is extremely fast memory that is built into a computer’s central processing unit (CPU), or located next to it on a separate chip.
- The CPU uses cache memory to store instructions that are repeatedly required to run programs, improving overall system speed.
- The advantage of cache memory is that the CPU does not have to use the motherboard’s system bus for data transfer.
- Whenever data must be passed through the system bus, the data transfer speed slows to the motherboard’s capability. The CPU can process data much faster by avoiding the bottleneck created by the system bus.



- A cache is a small amount of very fast associative memory.
- It sits between normal main memory and CPU.
- When CPU needs to access contents of memory location, then cache is examined for this data.
- If present, get from cache (fast).
- If not present, read required block from main memory to cache, then deliver from cache to CPU.
- Cache includes tags to identify which block of main memory is in each cache slot.
- The performance of cache memory is frequently measured in terms of a quantity called hit ratio. When CPU refers to memory and finds the word in cache, then it is said to produce hit.
- If word is not found in cache, it is in main memory and it counts as a miss.

- The ratio of the number of hits (success in finding the words in cache) to the total CPU references to memory (hits + misses) is known as hit ratio.
- The basic characteristic of cache memory is its fast access time.
- Hit Ratio = $\frac{Hit}{Hit+Miss}$
- Average memory access time

  = hit time + miss rate * miss time

- Example:

A computer with cache access time of 100 ns, a main memory access time of 1000 ns at the hit ratio of 0.9, what is the average access time?

Solution,

Hit time = 100 ns

Memory access time (miss time) = 1000 ns

Hit ratio = 0.9

Miss ratio = 1 – Hit ratio = 1-0.9 = 0.1

- Average memory access time

  = hit time + miss rate * miss time

  = 100 + 0.1*1000

  = 200 ns

- Given a cache memory with access time of 2ns and RAM with excess time 10 ns. If the hit ratio is 80%. Calculate the memory access time.

Hit time = 2 ns

Memory access time (miss time) = 10 ns

Hit ratio = 80% = 0.8

Miss ratio = 1 – Hit ratio = 1-0.8 = 0.2

- Average memory access time

  = hit time + miss rate * miss time

  = 2 + 0.2*10

  = 4 ns

The access time of a main memory is 1200 ns and that of cache memory 10 times faster. It is estimated that 80 percent of the memory requests are for read and remaining 20 percent for write. The hit ratio for read accesses only is 0.9. A write through procedure is used.

a. What is the average access time of the system considering only memory read cycles?
b. What is the average access time of the system considering only memory write cycles?
c. What is the average access time of the system considering both read and write memory cycles?
d. What is the hit ratio taking into consideration the write cycles?

Solution, Tavg = h*c +(1-h)*m

h = hit ratio,   1-h = miss ration

c = time to access from cache memory

m = time to access from main memory

Access time of main memory (Tm) = 1200ns

Cache time (Tc) = 120 ns (10 times faster)

Hit ratio for read access (Hr) = 0.9

80% of the memory request are for read = 0.8

20% of the memory request are for write = 0.2

a. Average access time of the system considering only memory read cycles = Hr *Tc + (1-Hr)* Tm

   = 0.9*120 + (1-0.9)*1200

   = 228 ns

b. Average access time of the system considering only memory write cycles = Hw *Tc + (1-Hw)* Tm

= 0*120+(1-0)*1200

= 1200 ns

Hw = 0 since the data must be immediate written onto memory.

c. Average access time of the system considering both read and write memory cycles = 0.8*228 + 0.2*1200

= 422.4 ns

d. hit ratio taking into consideration the write cycles = 0.8*0.9

= 0.72

Locality of Reference

- Analysis of large number of program shows that reference to memory at any given interval of time tend to be confined to few localized area in memory. This is known as locality of reference.
- Since size of cache memory is less as compared to main memory. So to check which part of main memory should be given priority and loaded in cache is decided based on locality of reference.
- Phenomenon in which the same values, or related storage locations, are frequently accessed, depending on the memory access pattern.
- There are two basic types of reference locality

  - Temporal locality

  - Spatial locality

Temporal Locality –

Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data. When CPU accesses the current main memory location for reading required data or instruction, it also get stored in cache memory which is based on the fact that same data or instruction may be needed in near future. This is known as temporal locality.

Spatial Locality –

Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in near future. This is slightly different from temporal locality. Here we are taking about nearly located memory locations while in temporal locality we were taking about the actual memory location that were being fetched.

Cache Mapping Techniques

The transformation of data from main memory to cache is known as mapping process. Three types of mapping procedures are:

- Associative Mapping
- Direct Mapping
- Set-Associative Mapping

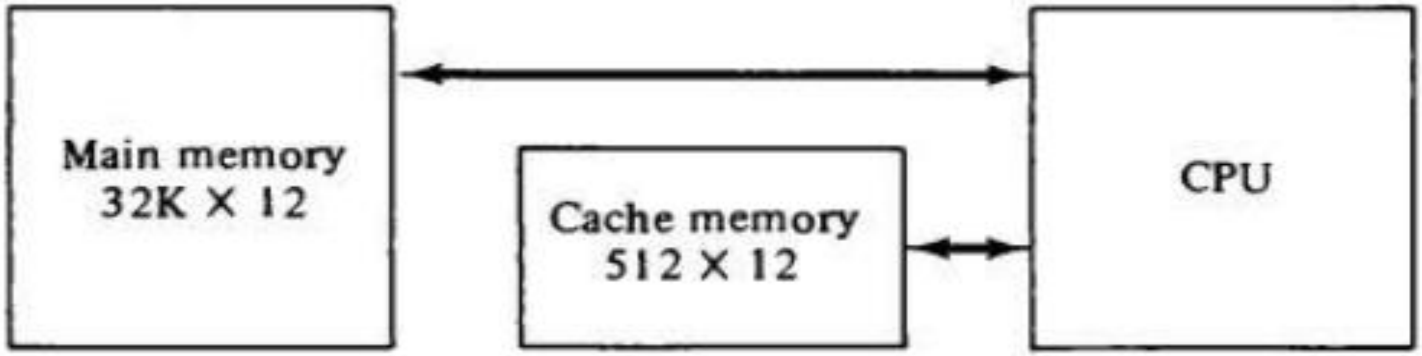


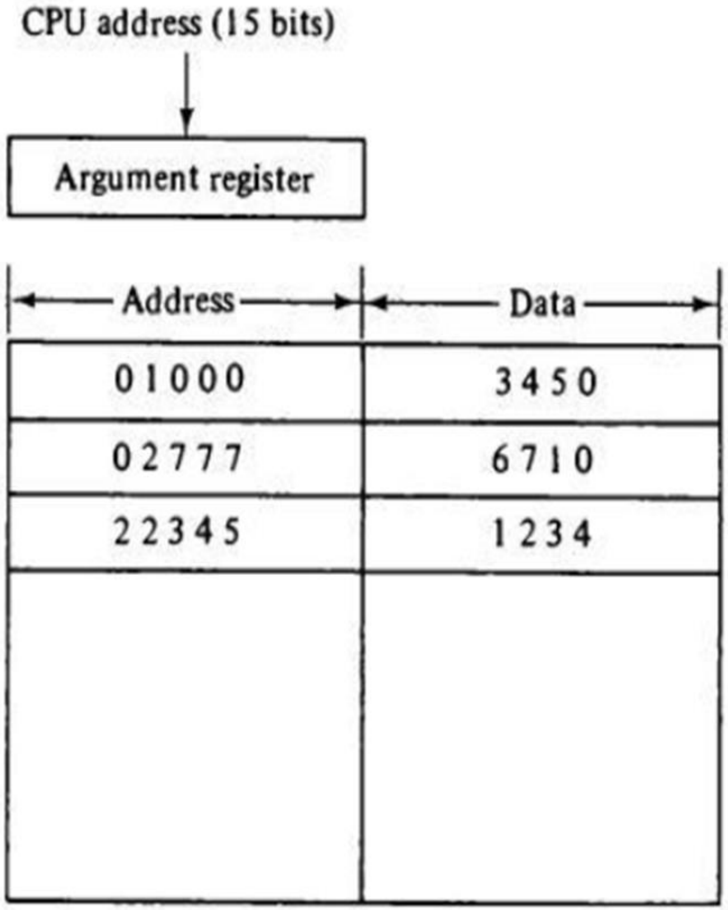
Fig: Example of Cache memory

- The main memory can store 32K words of 12 bits each. The cache is capable of storing 512 of these words at any given time.
- For every word stored in cache, there is a duplicate copy in main memory. The CPU communicates with both memories. It first sends a 15-bit address to cache.
- If there is a hit, the CPU accepts the 12-bit data from cache. If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

a. 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.
- 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.
- 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 no match occurs, the main memory is accessed for the word.
- The address-data pair is then transferred to the associative cache memory.
- 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.



Address	Data
01000	3450
02777	6710
22345	1234

Fig: Associative mapping cache (all numbers in octal)

b. Direct Mapping

- The CPU address of 15 bits is divided into two fields. The nine least significant bits constitute the index field and the remaining six bits form the tag field.

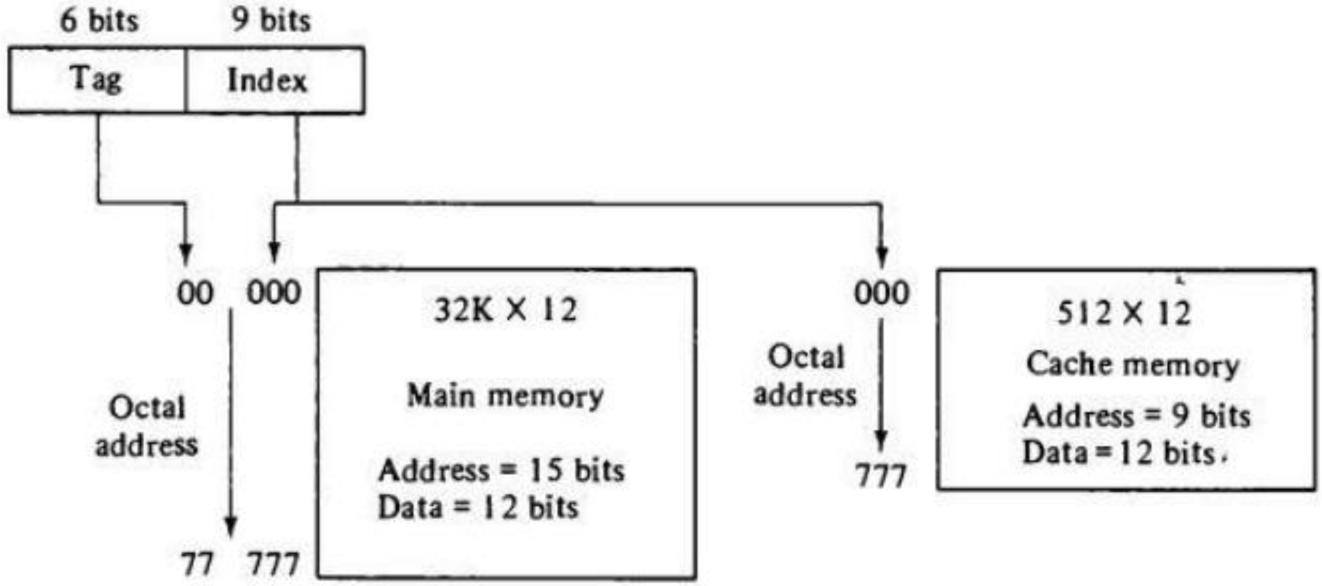
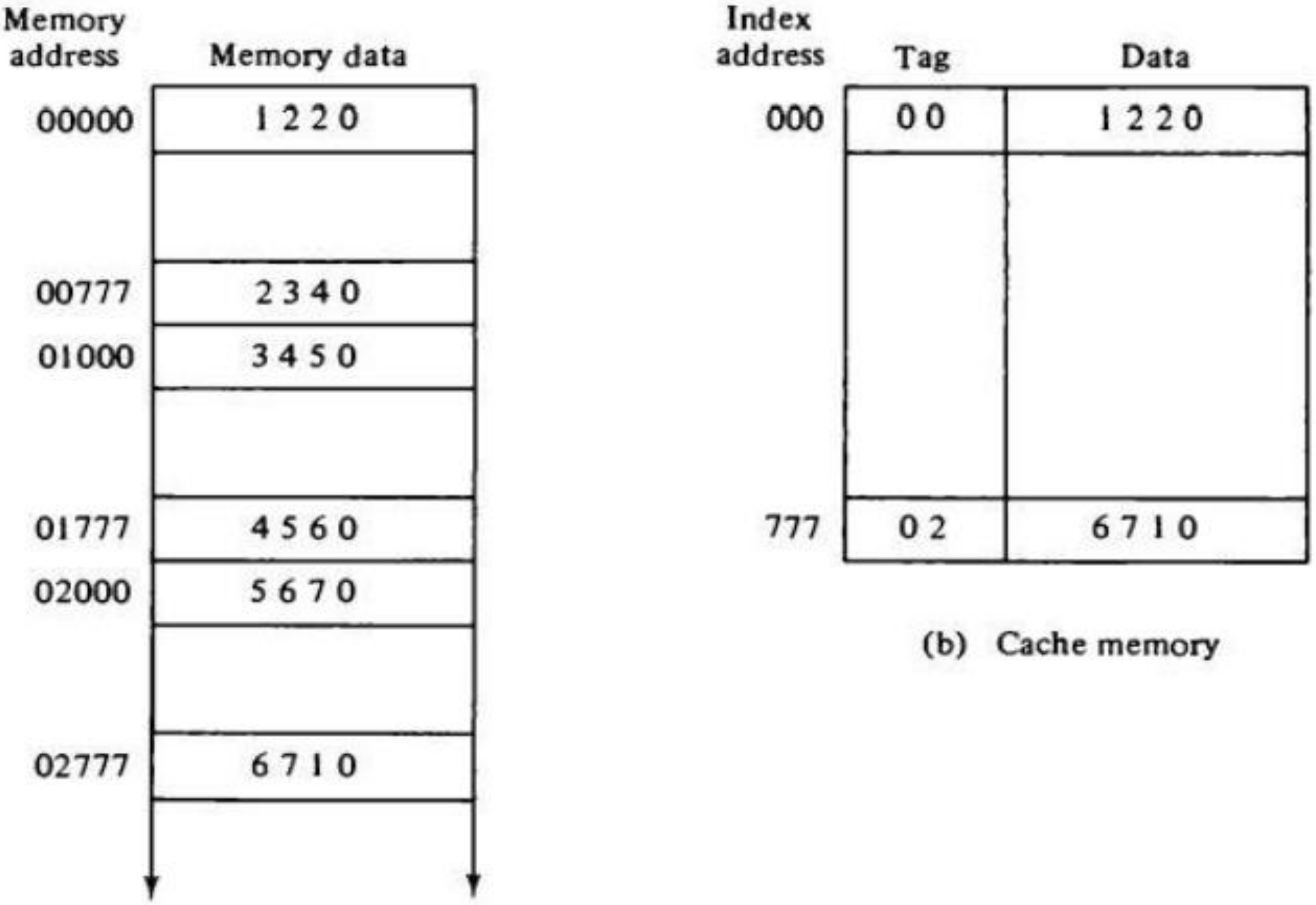


Fig: Addressing relationships between main and cache memories.

- The figure shows that main memory needs an address that includes both the tag and the index bits. The number of bits in the index field is equal to the number of address bits required to access the cache memory.
- There are 2K words in cache memory and 2^n 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.
- 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 alongside the data bits. When the CPU generates a memory request, the index field is used for the address to access the cache.



Memory address	Memory data
00000	1 2 2 0
00777	2 3 4 0
01000	3 4 5 0
01777	4 5 6 0
02000	5 6 7 0
02777	6 7 1 0

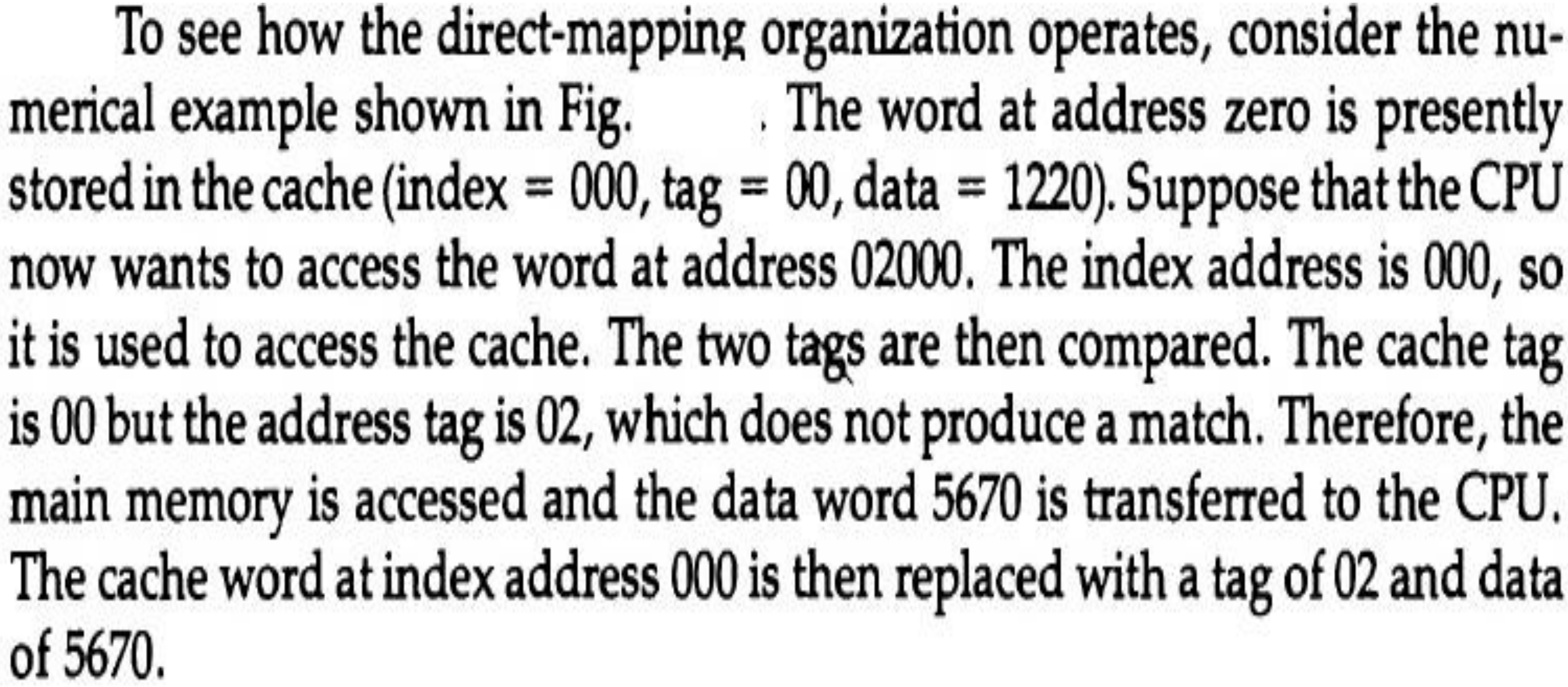
(a) Main memory

Index address	Tag	Data
000	0 0	1 2 2 0
777	0 2	6 7 1 0

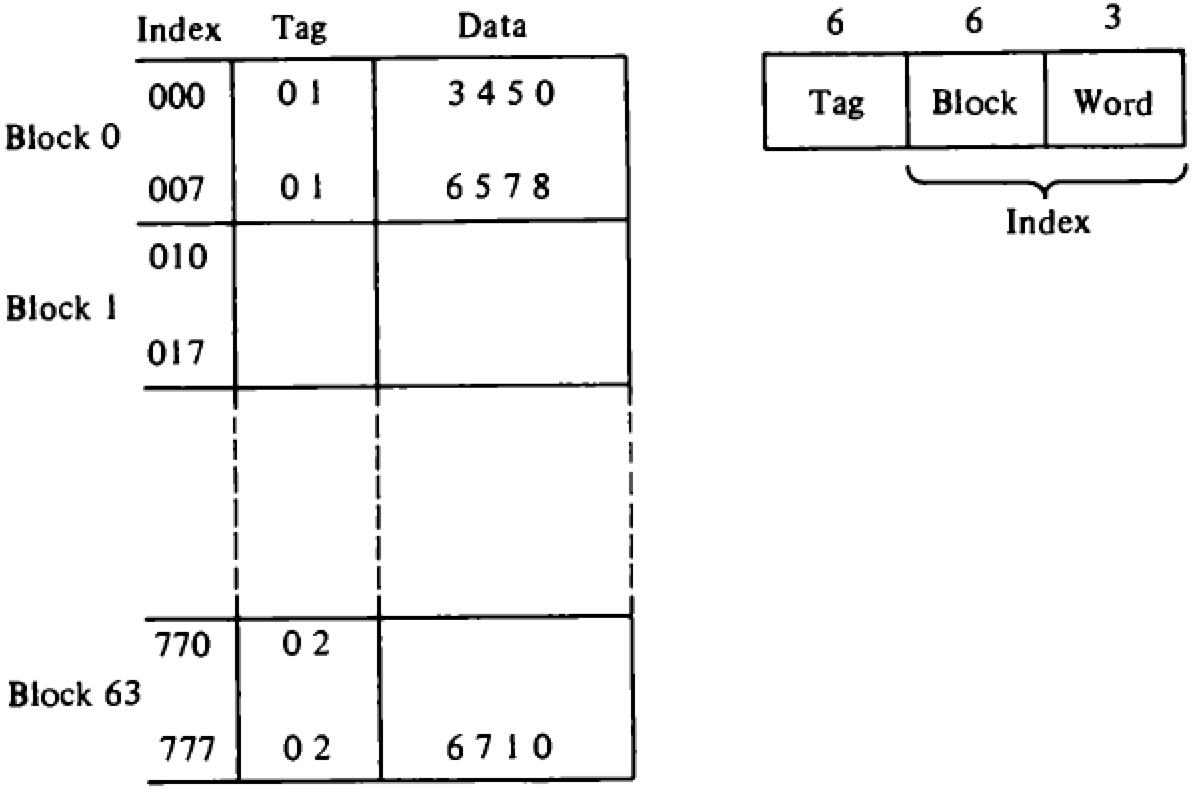
(b) Cache memory

Fig: 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 two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

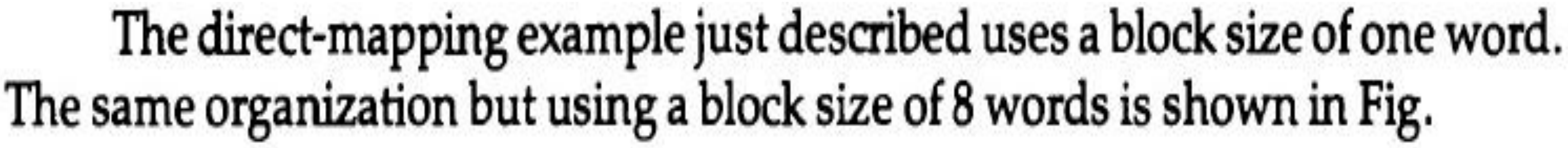


To see how the direct-mapping organization operates, consider the numerical example shown in Fig. . The word at address zero is presently stored in the cache (index = 000, tag = 00, data = 1220). Suppose that the CPU now wants to access the word at address 02000. The index address is 000, so it is used to access the cache. The two tags are then compared. The cache tag is 00 but the address tag is 02, which does not produce a match. Therefore, the main memory is accessed and the data word 5670 is transferred to the CPU. The cache word at index address 000 is then replaced with a tag of 02 and data of 5670.

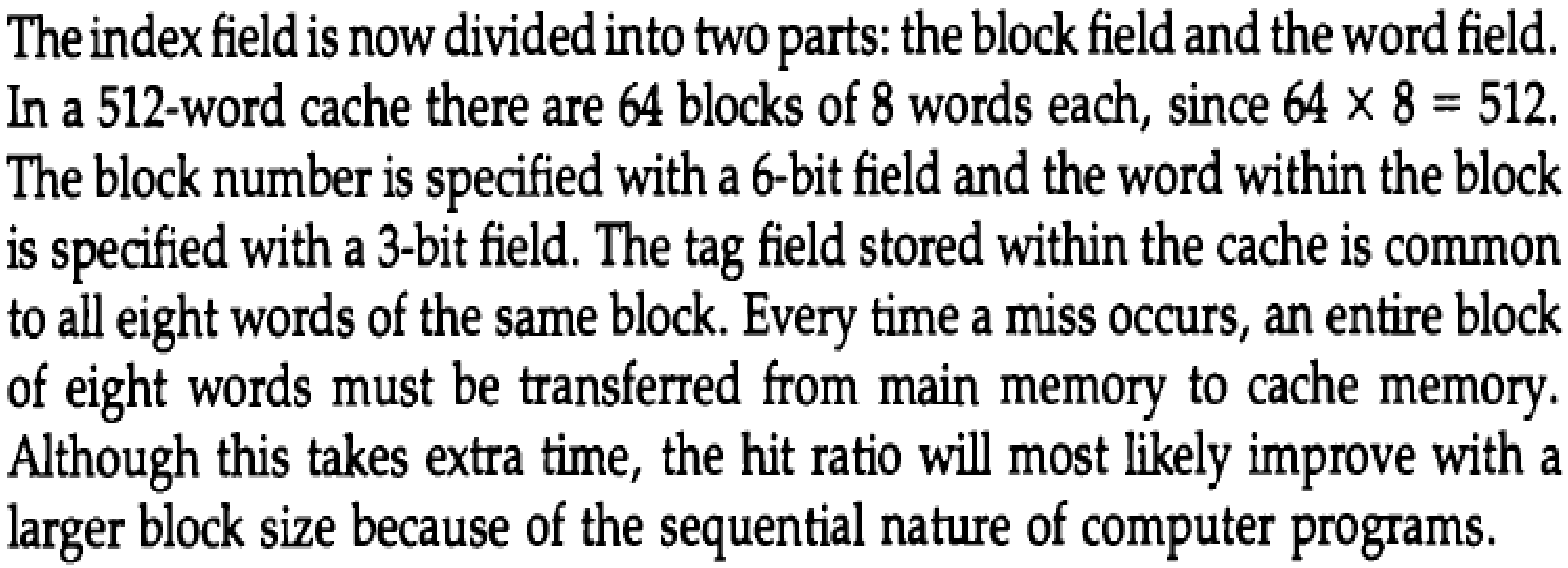


	Index	Tag	Data
Block 0	000	0 1	3 4 5 0
	007	0 1	6 5 7 8
Block 1	010		
	017		
Block 63	770	0 2	
	777	0 2	6 7 1 0

Direct mapping cache with block size of 8 words.



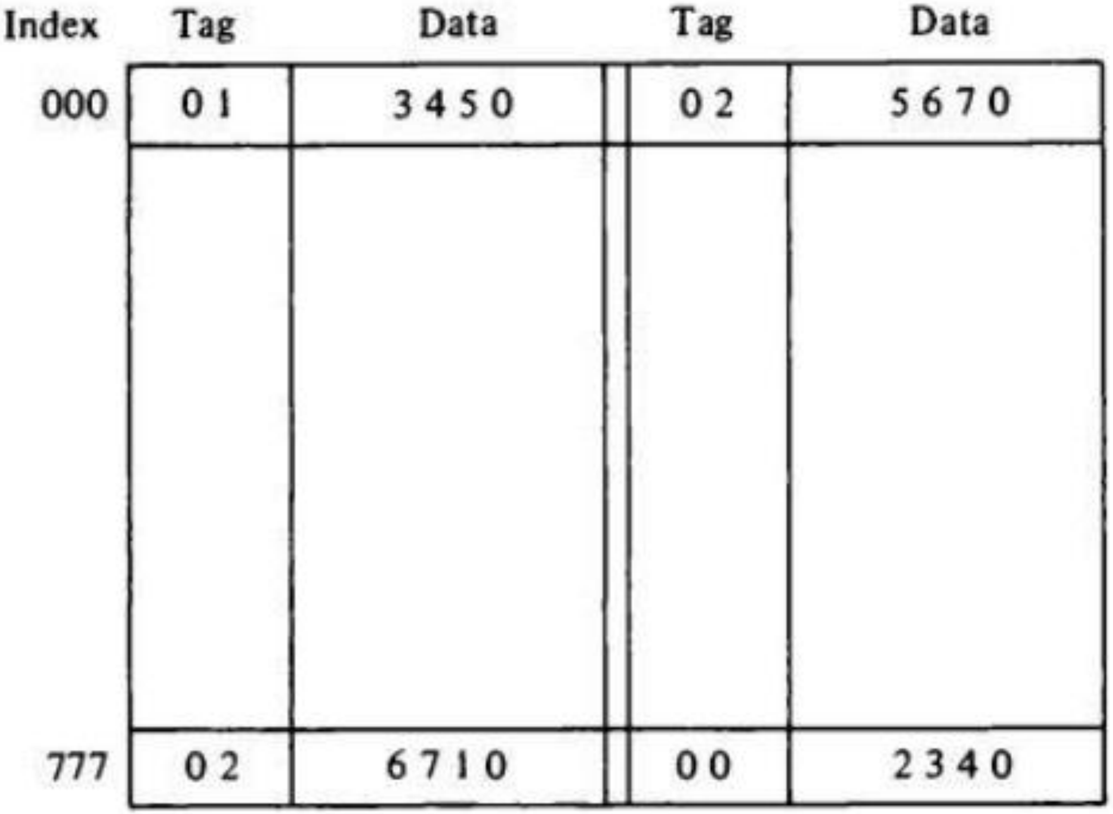
The direct-mapping example just described uses a block size of one word. The same organization but using a block size of 8 words is shown in Fig.



The index field is now divided into two parts: the block field and the word field. In a 512-word cache there are 64 blocks of 8 words each, since 64 × 8 = 512. The block number is specified with a 6-bit field and the word within the block is specified with a 3-bit field. 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.

c. Set Associative Mapping

- It 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.
- 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*36. It can accommodate 1024 words of main memory since each word of cache contains two data words.



Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

Fig: Two-way set associative mapping cache.

- The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Similarly, the words at addresses 02777 and 00777 are stored in cache at index address 777.
- When CPU generates a memory request, index value of address is used to access the cache. The tag field of CPU address is compared with both tags in cache to determine if a match occurs. The comparison logic is done by an associative search of a tag in the set similar to an associative memory searched.
- 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).

Writing into Cache

Writing into cache can be done in two ways:

- Write through

- Write Back

- In **write through**, whenever write operation is performed in cache memory, main memory is also updated in parallel with the cache.
- In **write back**, only cache is updated and marked by the flag. When the word is removed from cache, flag is checked if it is set the corresponding address in main memory is updated.

Virtual Memory

- A virtual memory system attempts to optimize the use of the main memory (the higher speed portion) with the hard disk (the lower speed portion). In effect, **virtual memory** is a technique for using the secondary storage to extend the apparent limited size of the physical memory beyond its actual physical size. It is usually the case that the available physical memory space will not be enough to host all the parts of a given active program.

- Virtual memory gives programmers the illusion that they have a very large memory and provides mechanism for dynamically translating program-generated addresses into correct main memory locations. The translation or mapping is handled automatically by the hardware by means of a mapping table.

Address Space and Memory Space

- An address used by the programmer is a virtual address (virtual memory addresses) and the set of such addresses is the Address Space. An address in main memory is called a location or physical address. The set of such locations is called the memory space. Thus the address space is the set of addresses generated by the programs as they reference instructions and data; the memory space consists of actual main memory locations directly addressable for processing. Generally, the address space is larger than the memory space.

- Example: consider main memory: 32K words (K = 1024) = 2^15 and auxiliary memory 1024K words = 2^20. Thus we need 15 bits to address physical memory and 20 bits for virtual memory (virtual memory can be as large as we have auxiliary storage).

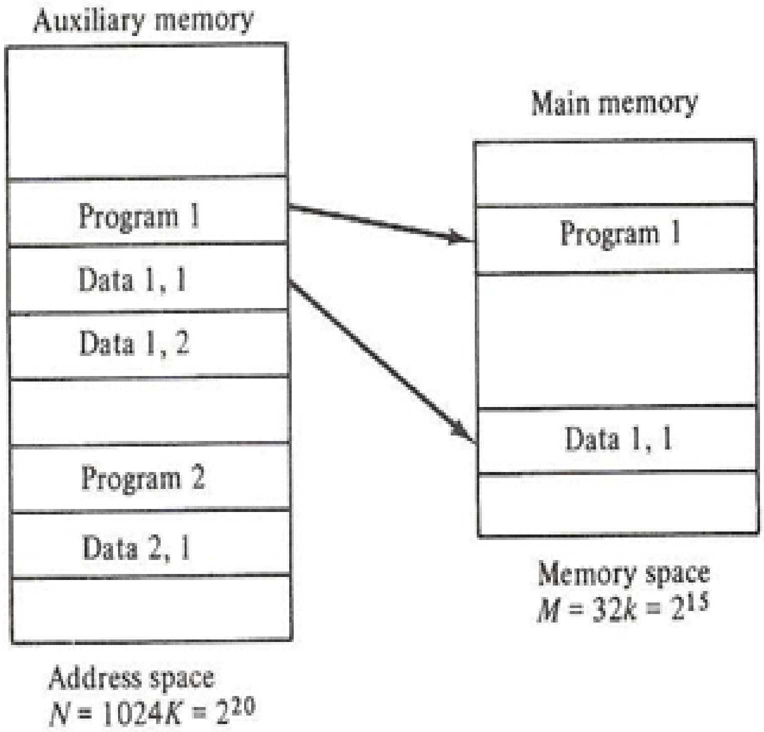


Fig: Relation between address and memory space in a virtual memory system

- Here auxiliary memory has the capacity of storing information equivalent to 32 main memories.
- Address space N = 1024K
- Memory space M = 32K
- In multiprogram computer system, programs and data are transferred to and from auxiliary memory and main memory based on the demands imposed by the CPU.

- In virtual memory system, address field of an instruction code has a sufficient number of bits to specify all virtual addresses. In our example above we have 20-bit address of an instruction (to 20-bit virtual address) but physical memory addresses are specified with 15-bits. So a table is needed to map a virtual address of 20-bits to a physical address of 15-bits. Mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU

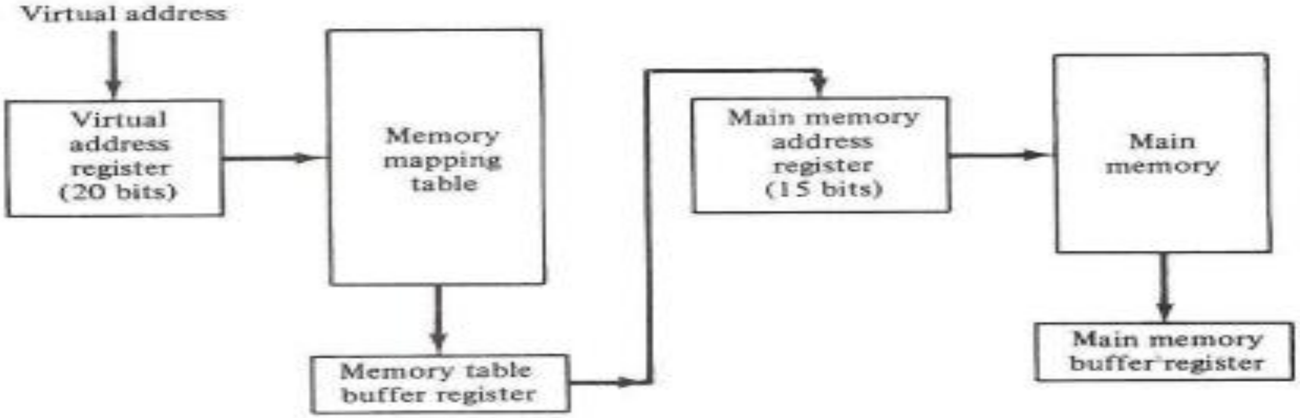

Virtual address
Virtual address register (20 bits)
Memory mapping table
Memory table buffer register
Main memory address register (15 bits)
Main memory
Main memory buffer register

Fig: Memory table for mapping a virtual address

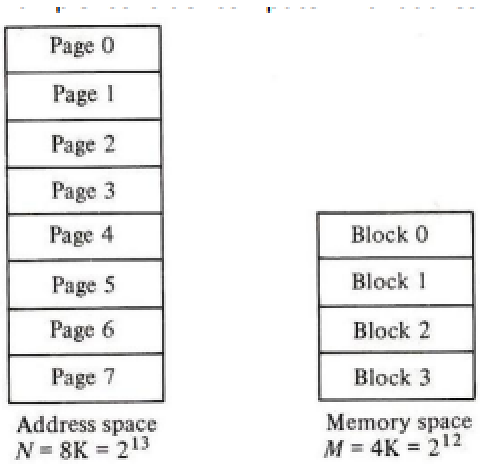
Address Mapping using Pages

Above memory table implementation of address mapping is simplified if the information in address space and memory space are each divided into groups of fixed size.

Blocks (or page frame): The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each.

Pages: refers to a portion of subdivided virtual memory having same size as blocks i.e. groups of address space.

Example: consider computer with address space = 8K and memory space = 4K



- ➔ If we spit both spaces into groups of 1K words, we obtain 8 pages and 4 blocks.
- ➔ Virtual address has 13 bits. Since each page consists of 210 = 1024 words, high-order 3 bits will specify one of 8 pages and low-order 10 bits give the line address within the pages.

- The mapping from address space to memory space becomes easy if virtual address is represented by two numbers: a page number address and a line with in the page. In a computer with 2^p words per page, p bits are used to specify a line address and remaining high-order bits of the virtual address specify the page number.
- NOTE: line address in address space and memory space is same; only mapping required is from page number to a block number.

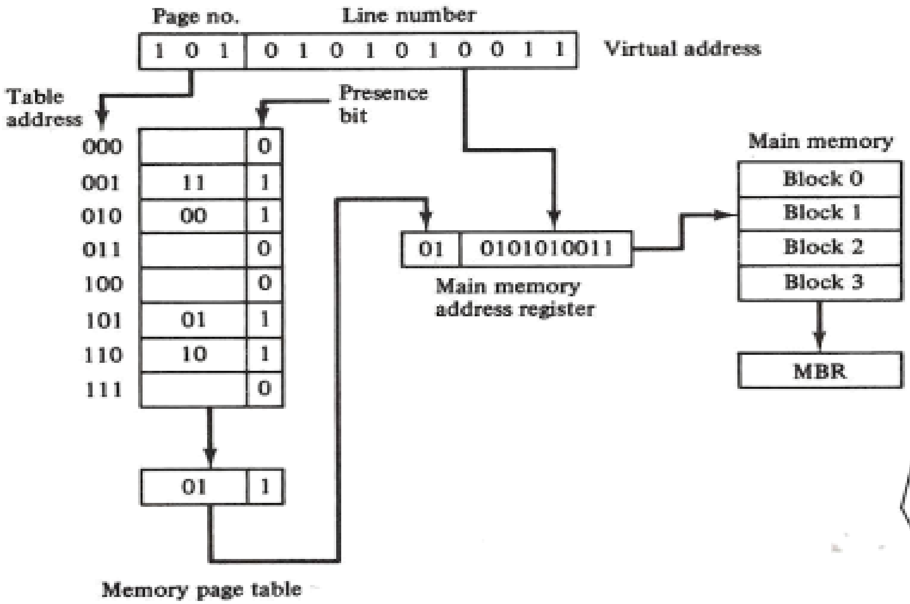
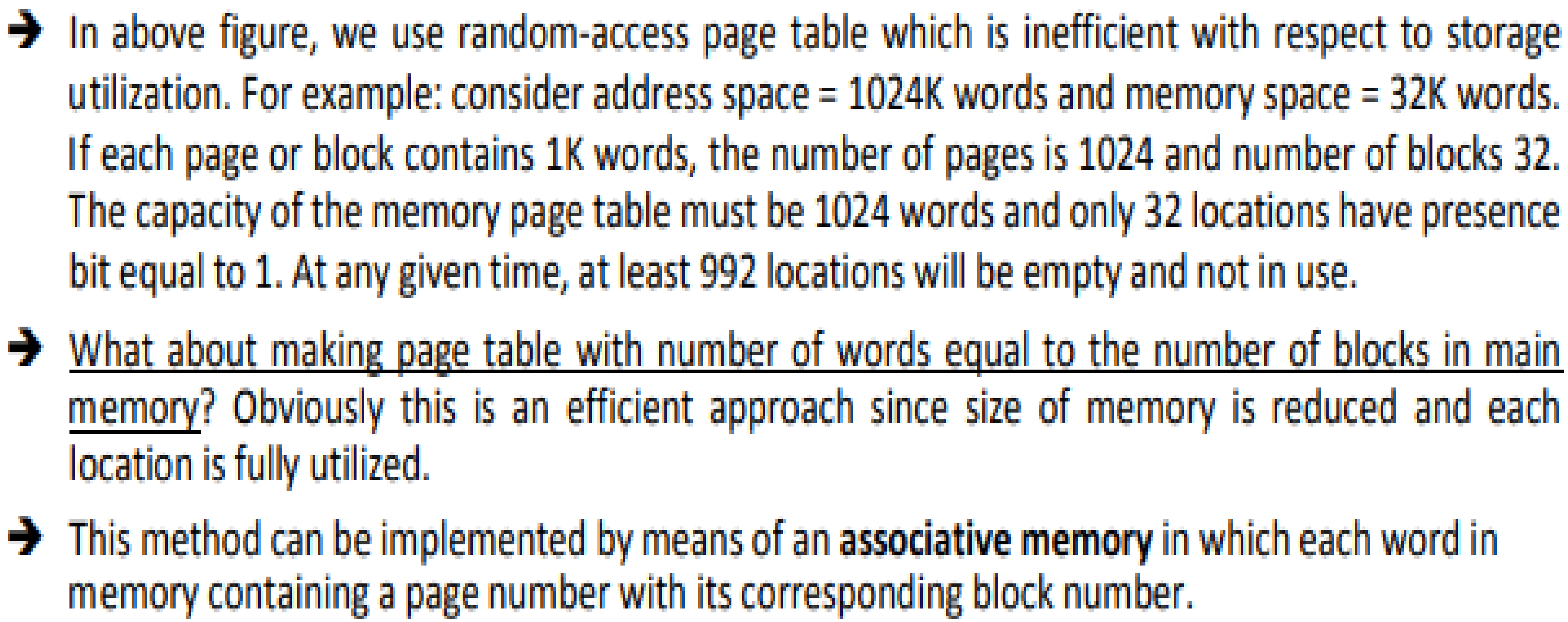


Table address		Presence bit
000		0
001	11	1
010	00	1
011		0
100		0
101	01	1
110	10	1
111		0

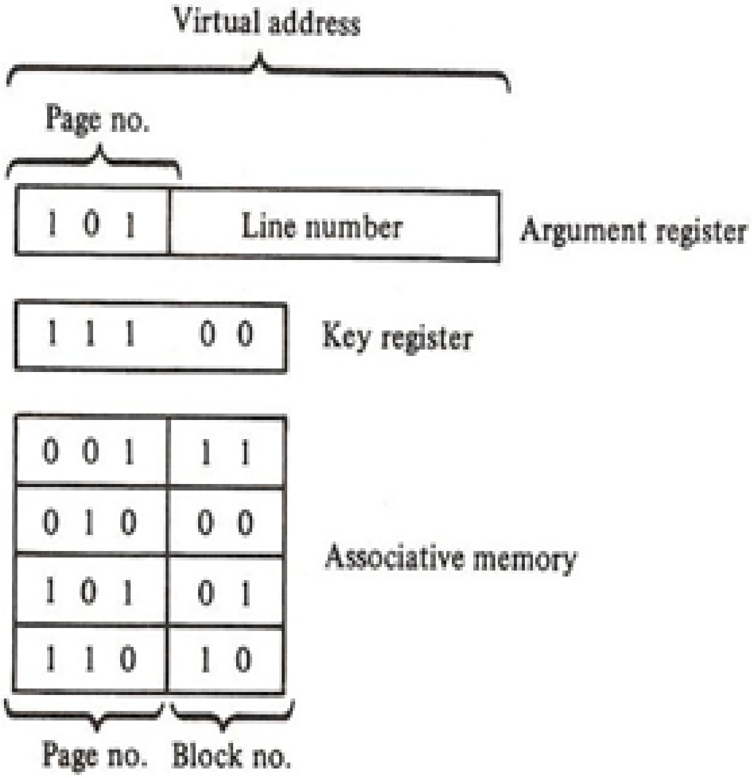
Fig: Memory table in paged system

- ➔ The memory-page table consists of 8 words, one for each page.
- ➔ The address in the page table denotes page number and the content of the word gives the block number where the page is stored in main memory.
- ➔ Presence bit when 0 indicates page is not available in main memory and when 1 says that page has been transferred to main memory.
- ➔ Table shows that pages 1,2,5 and 6 are now available in main memory in blocks 3,0,1 and 2 respectively.

Associative Memory Page Table



- In above figure, we use random-access page table which is inefficient with respect to storage utilization. For example: consider address space = 1024K words and memory space = 32K words. If each page or block contains 1K words, the number of pages is 1024 and number of blocks 32. The capacity of the memory page table must be 1024 words and only 32 locations have presence bit equal to 1. At any given time, at least 992 locations will be empty and not in use.
- What about making page table with number of words equal to the number of blocks in main memory? Obviously this is an efficient approach since size of memory is reduced and each location is fully utilized.
- This method can be implemented by means of an **associative memory** in which each word in memory containing a page number with its corresponding block number.



Page no.	Block no.
0 0 1	1 1
0 1 0	0 0
1 0 1	0 1
1 1 0	1 0

- ➔ The page field in each **associative memory table** word is compared with page number bits in an argument register (which contains page number in the virtual address), if match occurs, the word is read form memory and its corresponding block number is extracted.
- ➔ This is the associative page table for same example in previous section where we were using random-access page table.

Page Replacement

A virtual memory system is a combination of hardware and software techniques. A memory management software system handles:

1. Which page in main memory ought to be removed to make room for a new page?

2. When a new page is to be transferred from auxiliary memory to main memory?

3.Where the page is to be placed in main memory?

- Mechanism: when a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position. The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called **page fault**. When page fault occurs, the execution of the present program is suspended until required page is brought into memory. Since loading a page from auxiliary memory to main memory is basically an I/O operation, OS assigns this task to I/O processor. In the mean time, control is transferred to the next program in memory that is waiting to be processed in the CPU. Later, when memory block has been assigned, the original program can resume its operation.

- When **a page fault** occurs in a virtual memory system, it signifies that the page referenced by the program is not in main memory. A new page is then transferred from auxiliary memory to main memory. If main memory is full, it would be necessary to remove a page from a memory block to make a room for a new page. **The policy for choosing pages to remove is determined** from the replacement algorithm that is used.
- **GOAL**: try to remove the page least likely to be referenced by in the immediate future. There are numerous page replacement algorithms, two of which are:

- **First-in First-out (FIFO):** replaces a page that has been in memory longest time.
- **Least Recently Used (LRU):** assumes that least recently used page is the better candidate for removal than the least recently loaded page.

Memory Management Hardware

- A memory management system is a collection of hardware and software procedures for managing various programs (effect of multiprogramming support) residing in memory. Basic components of memory management unit (MMU) are:

- A facility for dynamic storage relocation that maps logical memory references into physical memory addresses.
- A provision for sharing common programs by multiple users
- Protection of information against unauthorized access.

- The dynamic storage relocation hardware is a mapping process similar to paging system.
- **Segment:** It is more convenient to divide programs and data into logical parts called segments despite of fixed-size pages. A segment is a set of logically related instructions or data elements. Segments may be generated by the programmer or by OS. Examples are: a subroutine, an array of data, a table of symbols or user's program.
- **Logical address:** The address generated by the segmented program is called a logical address. This is similar to virtual address except that logical address space is associated with variable-length segments rather than fixed-length pages

Segmented-Page Mapping

The length of each segment is allowed to grow and contract according to the needs of the program being executed. One way of specifying the length of a segment is by associating with it a number of equal-sized pages.

Consider diagram below:

Logical address = Segment + page + Word

Where **segment** specifies segment number, **page** field specifies page with in the segment and **word** field specifies specific word within the page.

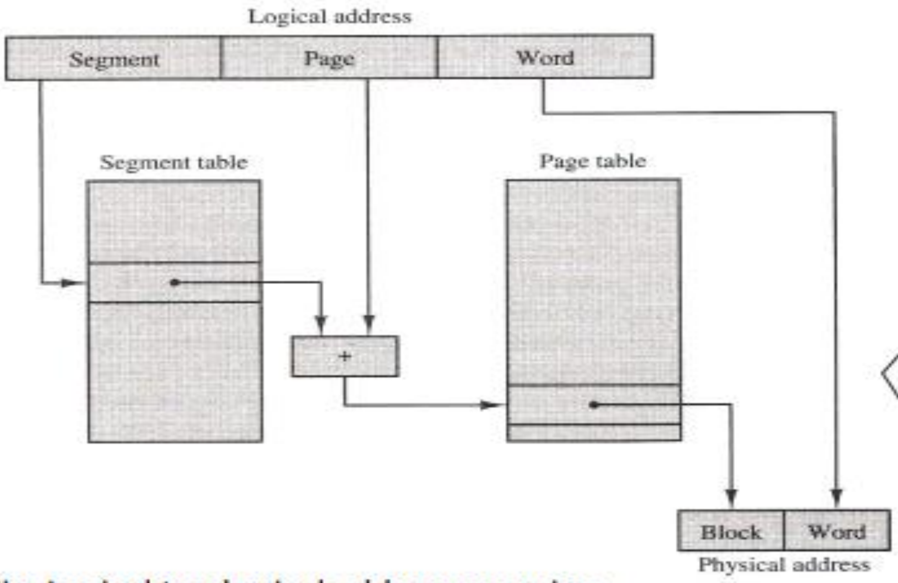


Fig: Logical to physical address mapping

- Mapping of logical address to physical address is done by using two tables: segment and page table.
- The entry in the segment table is a pointer address for the page table base, which is then added to page number (given in logical address). The sum point to some entry in page table and content of that page is the address of physical block. The concatenation of block field with the word field produces final **physical mapped address**.

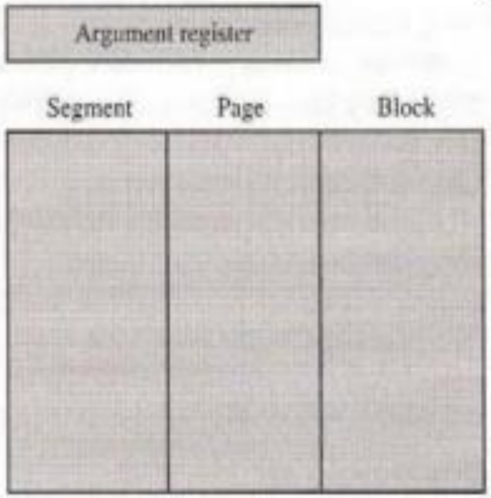
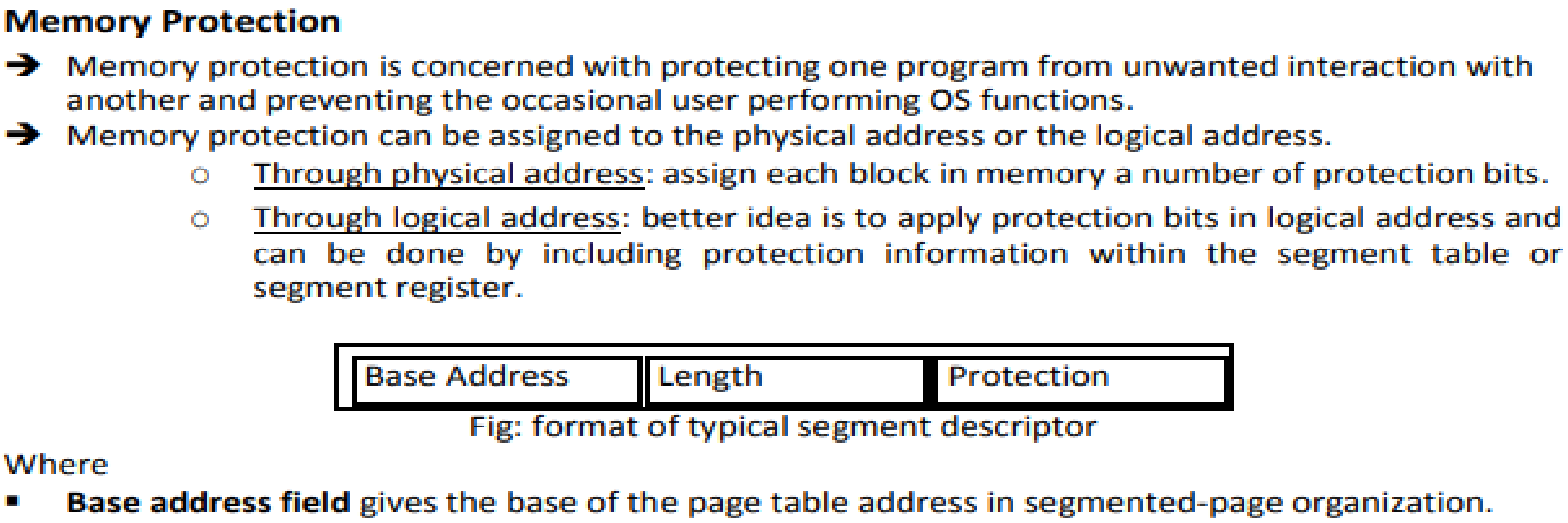


Fig: Associative memory Translation Look-aside Buffer (TLB)

➔ This is a fast associative memory (TLB) and holds most recently referenced entries.

(Alternatively we could store above two tables: segment table and page table, in two separate small memories which really increases the CPU access time)



Memory Protection

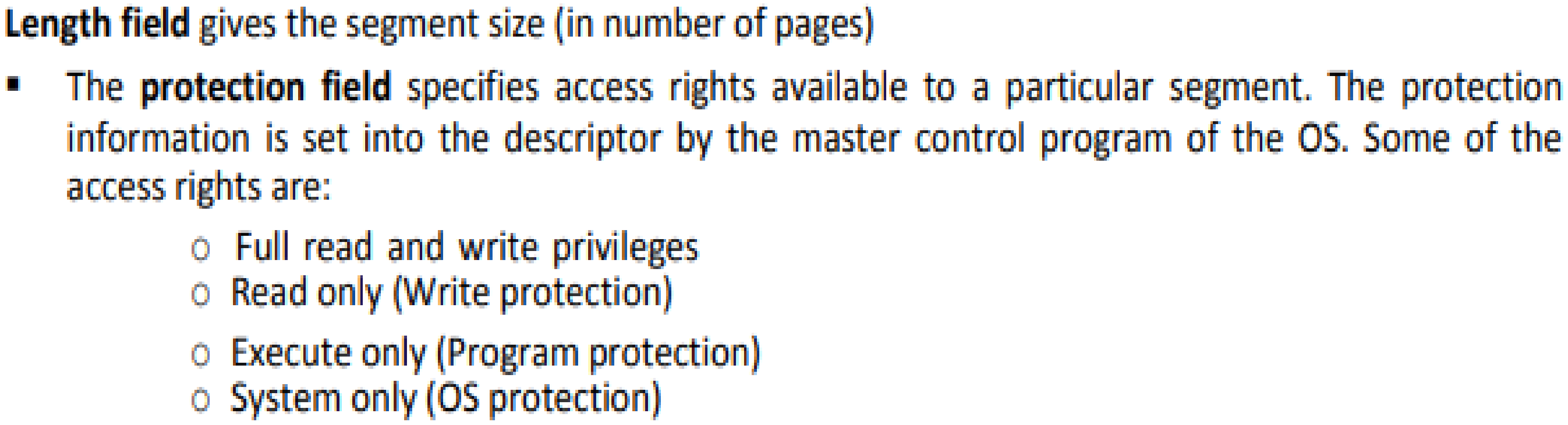
- Memory protection is concerned with protecting one program from unwanted interaction with another and preventing the occasional user performing OS functions.
- Memory protection can be assigned to the physical address or the logical address.
  - Through physical address: assign each block in memory a number of protection bits.
  - Through logical address: better idea is to apply protection bits in logical address and can be done by including protection information within the segment table or segment register.

Base Address	Length	Protection

Fig: format of typical segment descriptor

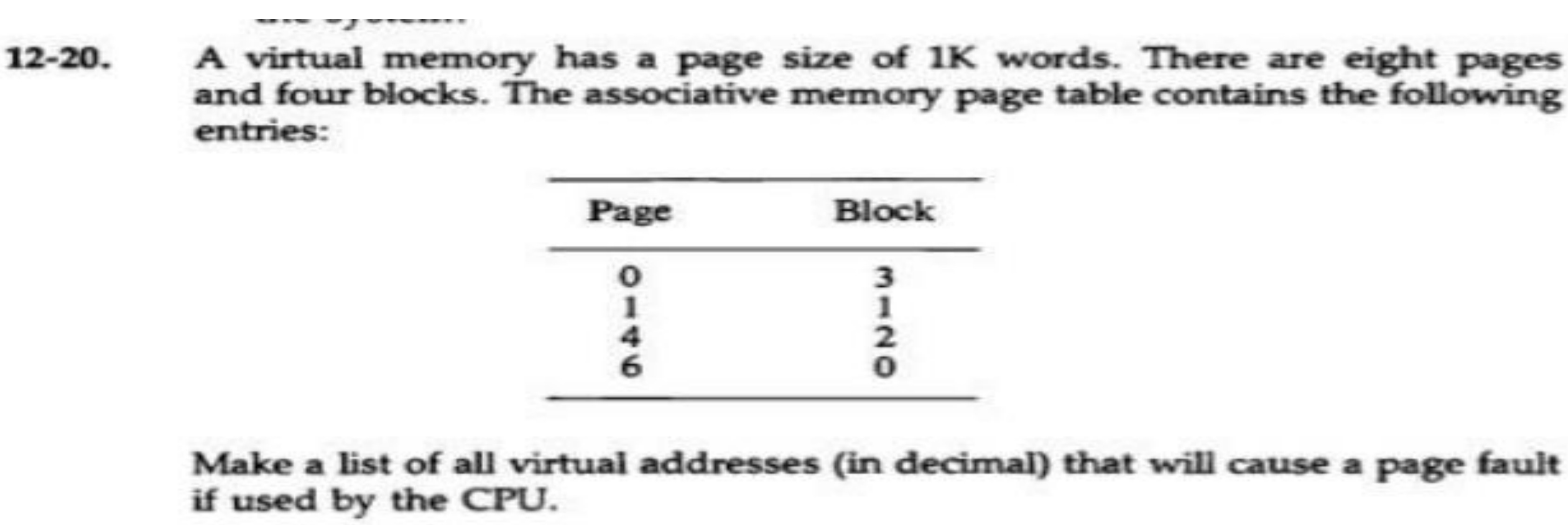
Where

- **Base address field** gives the base of the page table address in segmented-page organization.



Length field gives the segment size (in number of pages)

- The **protection field** specifies access rights available to a particular segment. The protection information is set into the descriptor by the master control program of the OS. Some of the access rights are:
  - Full read and write privileges
  - Read only (Write protection)
  - Execute only (Program protection)
  - System only (OS protection)



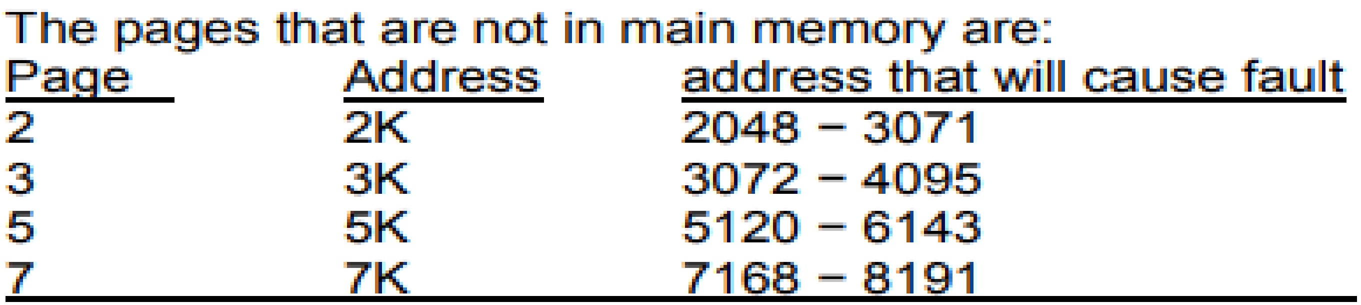
the system.

12-20. A virtual memory has a page size of 1K words. There are eight pages and four blocks. The associative memory page table contains the following entries:

Page	Block
0	3
1	1
4	2
6	0

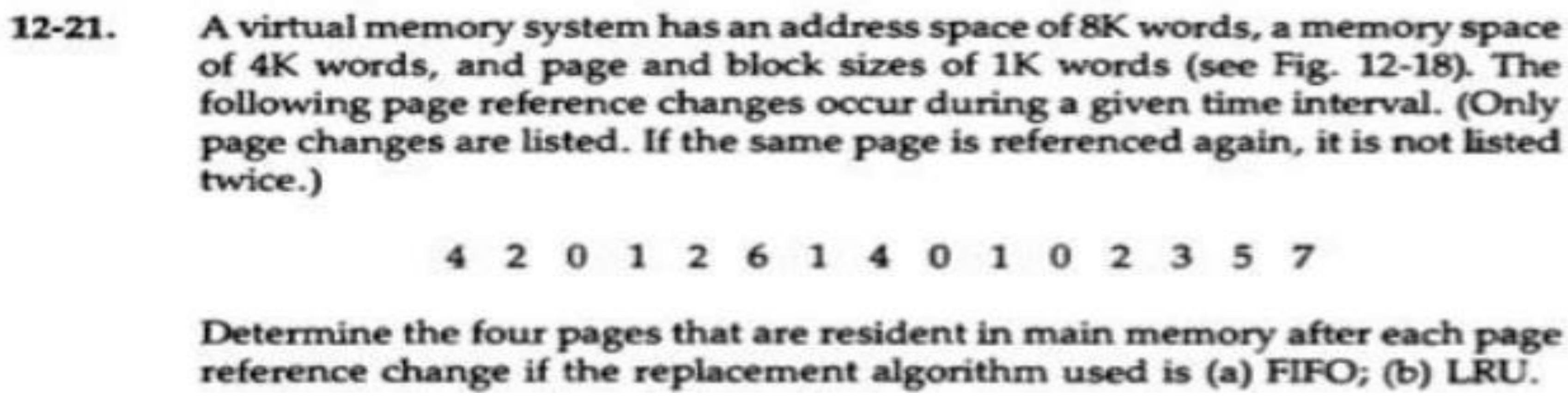
Make a list of all virtual addresses (in decimal) that will cause a page fault if used by the CPU.

Solution:



The pages that are not in main memory are:

Page	Address	address that will cause fault
2	2K	2048 – 3071
3	3K	3072 – 4095
5	5K	5120 – 6143
7	7K	7168 – 8191



12-21. A virtual memory system has an address space of 8K words, a memory space of 4K words, and page and block sizes of 1K words (see Fig. 12-18). The following page reference changes occur during a given time interval. (Only page changes are listed. If the same page is referenced again, it is not listed twice.)

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

Determine the four pages that are resident in main memory after each page reference change if the replacement algorithm used is (a) FIFO; (b) LRU.

- Solution:

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

Page Reference	FIFO	LRU
Initially	4201	4201
2	4201	4012
6	2016	0126
1	2016	0261
4	0164	2614
0	0164	6140
1	0164	6401
0	0164	6410
2	1642	4102
3	6423	1023
5	4235	0235
7	2357	2357

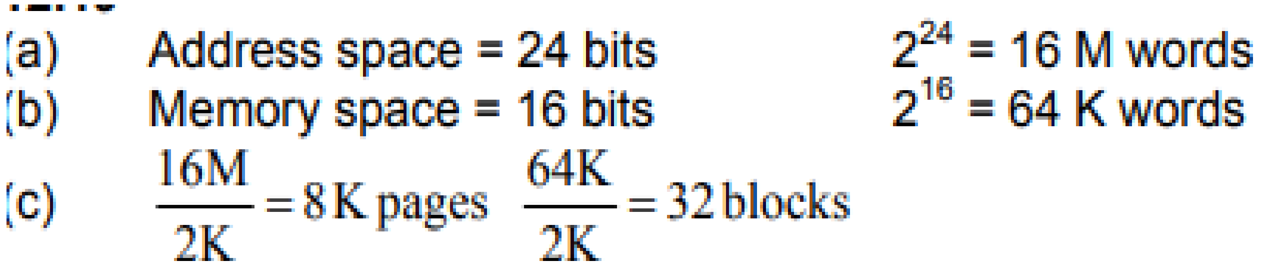
- An address space is specified by 24 bits and the corresponding memory space by 16 bits.

a. How many words are there in the address address?

b. How many words are there in the memory address?

c. If the page consists of 2K words, how many pages and blocks are there in the system?

Solution:



(a) Address space = 24 bits 2^{24} = 16 M words

(b) Memory space = 16 bits 2^{16} = 64 K words

(c) $\dfrac{16\text{M}}{2\text{K}} = 8\,\text{K pages}$ $\dfrac{64\text{K}}{2\text{K}} = 32\,\text{blocks}$

MULTIPROCESSOR

Characteristics of Multiprocessor:

- A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment.
- The term “processor” in multiprocessor can mean either a central processing unit (CPU) or an input-output processor (IOP).
- Multiprocessors are classified as multiple instruction stream, multiple data stream (MIMD) systems.
- Multiprocessing improves the reliability of the system.
- The benefit derived from a multiprocessor organization is an improved system performance.
  - Multiple independent jobs can be made to operate in parallel.
  - A single job can be partitioned into multiple parallel tasks.
- Multiprocessing can improve performance by decomposing a program into parallel executable tasks.
  - The user can explicitly declare that certain tasks of the program be executed in parallel.
  - The other is to provide a compiler with multiprocessor software that can automatically detect parallelism in a user’s program.

Advantages:

- Take little physical space
- Inexpensive
- Improves reliability, if one fails other works
- Improves system performance.

Types of multiprocessor

- Multiprocessors are classified by the way their memory is organized. They are:
-
- **Shared memory or tightly-coupled multiprocessor:** the multiprocessor system in which all processing elements share a common memory. In this type, there is no local memory with processor but they have their own cache memory.

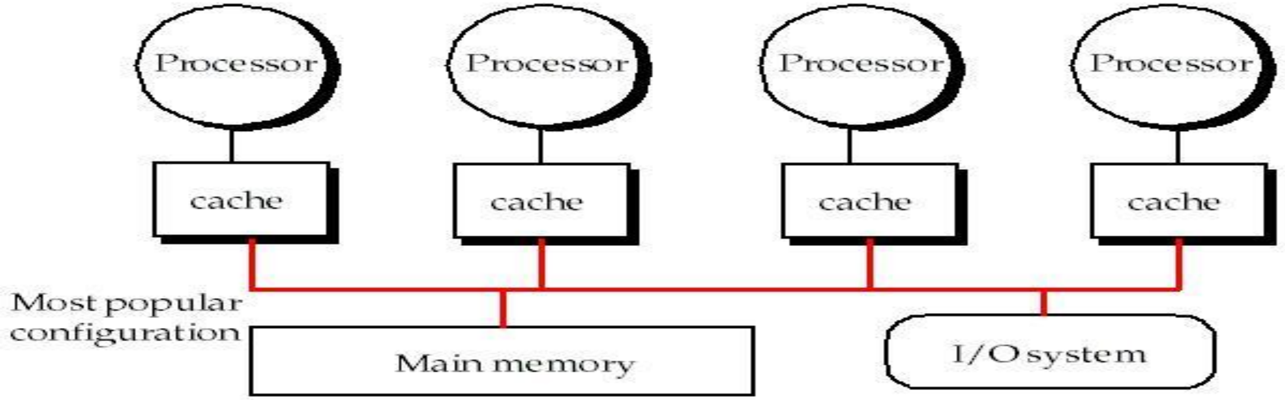


Fig: Shared Memory Architecture

-

- **Distributed memory or loosely-coupled multiprocessor:** the multiprocessor system in which each processing element has its own private local memory and all the processors are tied together by a switching mechanism to route information from one processor to another through a message passing scheme.

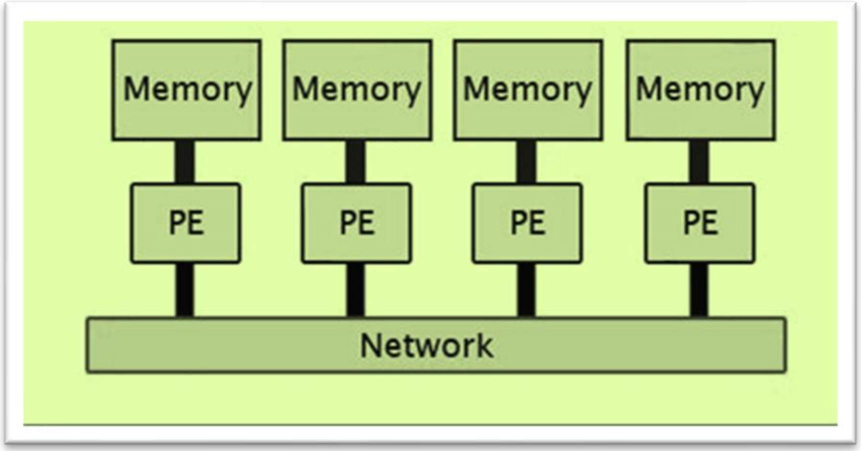


Fig: Distributed Memory Architecture

Interconnection Structures

- The components that form a multiprocessor system are CPUs, IOPs connected to I/O devices, and a memory unit that may be partitioned into a number of separate modules.
- The interconnection between the components can have different physical configurations, depending on the number of transfer paths that are available between the processors and memory in a shared memory system or among the processing elements in a loosely coupled system.
- There are several interconnection networks. Some of these schemes are given as:

1. Time shared common bus
2. Multiport memory
3. Crossbar switch
4. Multistage switching network
5. Hypercube system

1. Time-shared common bus

A common bus is used for all CPU to communicate with shared memory. At any given time, only one processor can communicate with the memory or another processor at any given time. When one processor is communicating with the memory, all other processors are either busy with their internal operation or idle waiting for the bus. Conflict is resolved by bus controller that establishes priority among the requesting units.

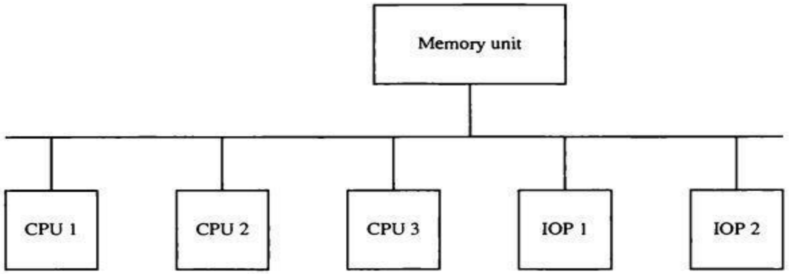


Fig: Time-shared Common Bus Organization.

Advantages:

1. Simplicity
2. Flexibility
3. Reliability

Disadvantages

1. Performance limited by bus cycle time
2. Each processor should have local cache
3. Reduce number of bus accesses

2. Multiport Memory

It uses separate buses between each memory module and each CPU.

Each processor has direct independent access of memory modules by their own bus connected to each module.

The modules must have internal control logic to determine which port will have access to memory at any given time.

Memory access conflicts are resolved by assigning fixed priorities to each memory port. CPU1 will have priority over CPU2, CPU2 will have priority over CPU3 and CPU4 will have the lowest priority. It is high speed due to multiple port between CPU and memory.

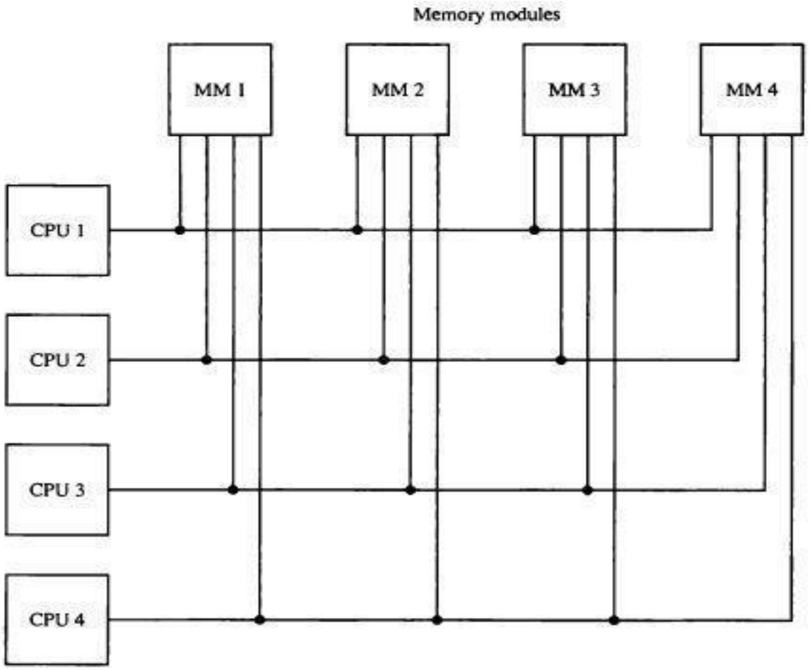


Fig: Multiport memory organization

Advantages:
Better performance because each processor has dedicated path to each module.

Can configure portions of memory as private to one or more processors so increased security.

Disadvantages:
Requires extra control logic so more complex and increase the cost.

Requires large connection wires.

3. Crossbar Switch

A crossbar switch (also known as cross-point switch or matrix switch) is a switch connecting multiple inputs to multiple outputs in a matrix manner.

The crossbar switch organization consists of a number of cross points that are placed at interconnection between processor bus and memory module path

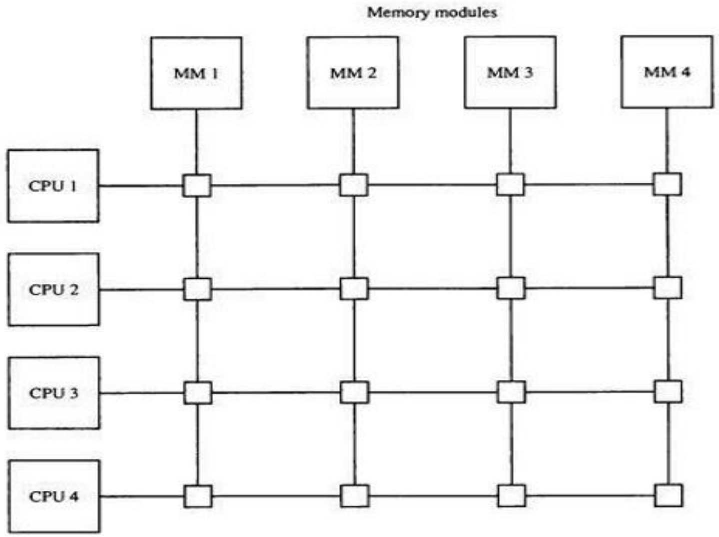


Fig: Crossbar switch.

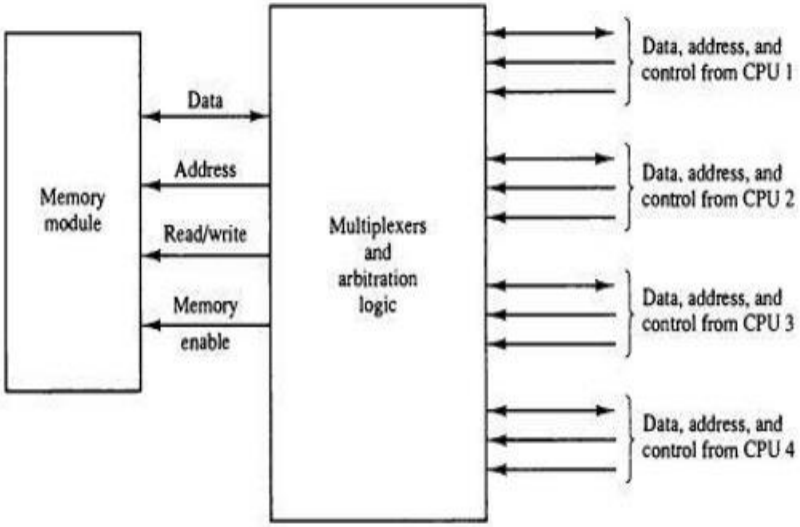

Memory module
Data
Address
Read/write
Memory enable
Multiplexers and arbitration logic
Data, address, and control from CPU 1
Data, address, and control from CPU 2
Data, address, and control from CPU 3
Data, address, and control from CPU 4

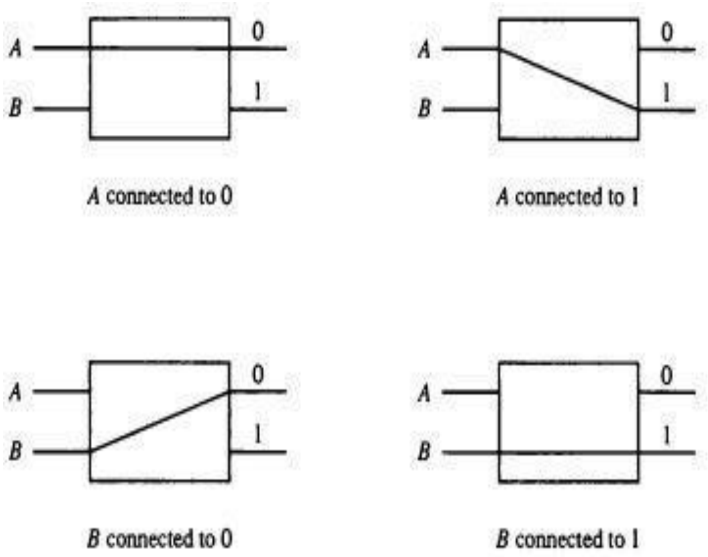
Fig: Block diagram of crossbar switch.

- Above figure shows a crossbar switch interconnection of four CPUs and four memory modules. The small square in each cross point is a switch that determines the path from a processor to a memory module.

- Each switch point has control logic to set up the transfer path between a processor and memory. It examines the address that is placed in the bus to determine whether its particular module is being addressed. It also resolves multiple requests for access to the same memory module on a predetermined priority basis.

4. Multistage Switching Network

- ➢ Controls the communication between a number of resources and destinations.
- ➢ Basic components of a multistage switching network are two-input, two-output interchange switch.



A connected to 0

A connected to 1

B connected to 0

B connected to 1

Fig: Operation of a 2*2 interchange switch.

- As shown in above figure, the 2*2 switch has two inputs labeled A and B, and two outputs 0 and 1. There are control signals associated with the switch that establish the interconnection between the input and output terminals.

- The switch has capacity of connecting input A to either of the outputs. Terminal B of the switch behaves in a similar fashion. The switch also has the capability to arbitrate between conflicting requests. If inputs A and B both request the same output terminal, only one of them will be connected, the other will be blocked.

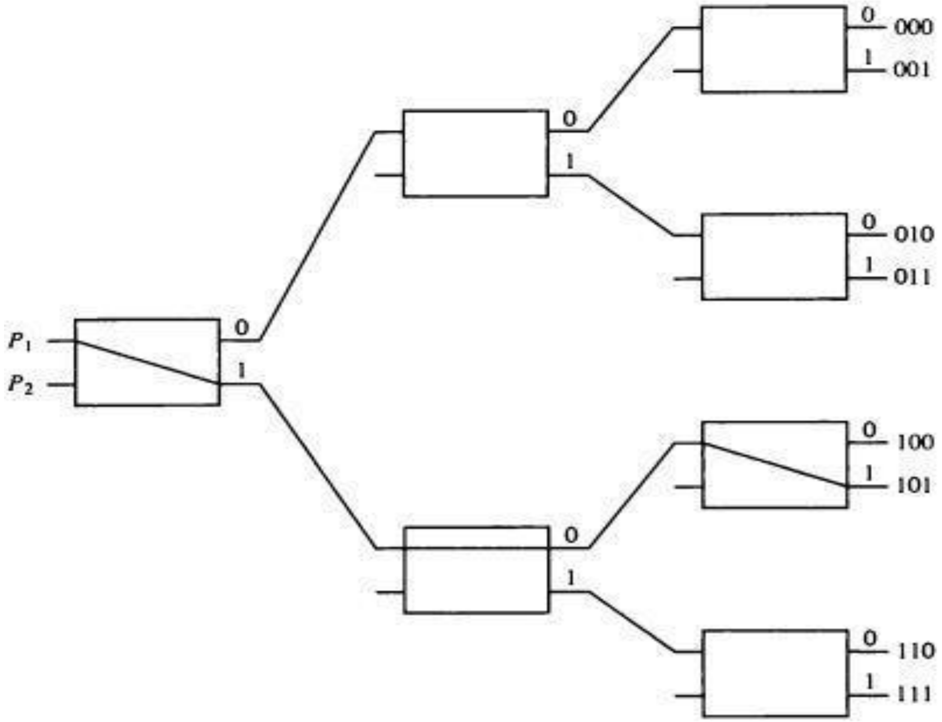


Fig: Binary tree with 2*2 switches.

- Using the 2*2 switch as a building block, it is possible to build a multistage network as in figure.

5. Hypercube Interconnection

- The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N=2^n$ processors interconnected in an n-dimensional binary cube.

- Each processor forms a node of the cube. Each processor has direct communication paths to n other neighbor processors. These paths correspond to the edges of the cube.

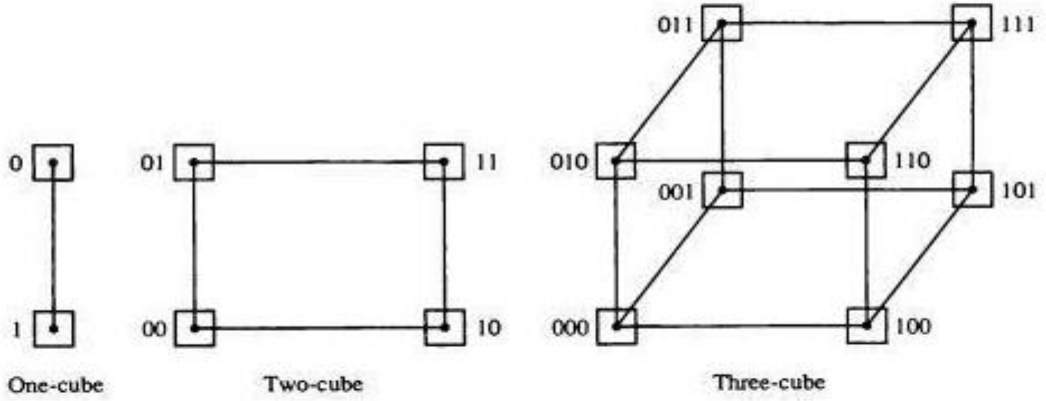


Fig: Hypercube structures for n = 1, 2, 3.

- A one-cube structure has n = 1 and $2^1 = 2$. It contains two processors interconnected by a single path.
- A two-cube structure has n = 2 and $2^2 = 4$. It contains four nodes interconnected as a square.

- A three-cube structure has eight nodes interconnected as a cube. An n-cube structure has 2^n nodes with a processor residing in each node. Each node is assigned a binary address in such a way that the addresses of two neighbors differ in exactly one bit position. For example, the three neighbors of the node with address 100 in a three cube structure are 000, 110, and 101. Each of these binary numbers differs from address 100 by one bit value.

- Routing messages through an n cube structure may take from one to n links from a source node to a destination node. For example, in a three cube structure, node 000 can communicate directly with node 001. It must cross at least two links to communicate with 011 (from 000 to O01 to 011 or from 000 to 010 to 011). It is necessary to go through at least three 1ines to communicate from node 000 to node 111.

- A routing procedure can be developed by computing the exclusive OR of the source node address with the destination node address. The resulting binary value will have 1 bits corresponding to the axes on which the two nodes differ. The message is then sent along any one of the axes. For example, in a three cube structure, a message at 010 going to 001 produces an exclusive OR of the two addresses equal to O11. The message can be sent along the second axis to 000 and then through the third axis to 001.