Unit 4
MEMORY

Characteristics of Computer Memory Systems

Location

- CPU (registers and L1 cache)
- Internal Memory (main)
- External (secondary)

Capacity

- Word Size - typically equal to the number of bits used to represent a number and to the instruction length.
- Number of Words - has to do with the number of addressable units (which are typically words, but are sometimes bytes, regardless of word size). For addresses of length A (in bits), the number of addressable units is 2A.

Unit of Transfer

- Word
- Block

Access Method

- Sequential Access
- ✓ information used to separate or identify records is stored with the records
- ✓ access must be made in a specific linear sequence
- ✓ the time to access an arbitrary record is highly variable
- Direct Access
- ✓ individual blocks or records have an address based on physical location
- ✓ access is by direct access to general vicinity of desired information, then some search
- ✓ access time is still variable, but not as much as sequential access
- Random Access
- ✓ each addressable location has a unique, physical location
- ✓ access is by direct access to desired location
- ✓ access time is constant and independent of prior accesses
- Associative
- ✓ desired units of information are retrieved by comparing a sub-part of the unit with a desired mask -- location is not needed
- ✓ all matches to the mask are retrieved simultaneously
- ✓ access time is constant and independent of prior accesses
- ✓ most useful for searching - a search through N possible locations would take O(N) with Random Access Memory, but O(1) with Associative Memory

Performance

- Access Time
- Memory Cycle Time - primarily for random-access memory = access time + additional time required before a second access can begin (refresh time, for example)
- Transfer Rate
- ✓ Generally measured in bits/second

- ✓ Inversely proportional to memory cycle time for random access memory

Physical Type

- Most common - semiconductor and magnetic surface memories
- Others - optical, bubble, mechanical (e.g. paper tape), core, esoteric/theoretical (e.g.biological)

Physical Characteristics

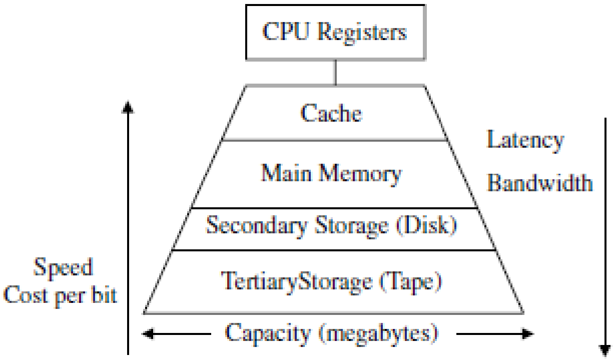
- volatile - information decays or is lost when power is lost
- non-volatile - information remains without deterioration until changed -- no electrical power needed
- non-erasable
- ✓ information cannot be altered with a normal memory access cycle
- ✓ As a practical matter, must be non-volatile

Organization - the physical arrangement of bits to form words.

- Obvious arrangement not always used
- Ex. Characters vs. Integers vs. Floating Point Numbers

Memory Hierarchy

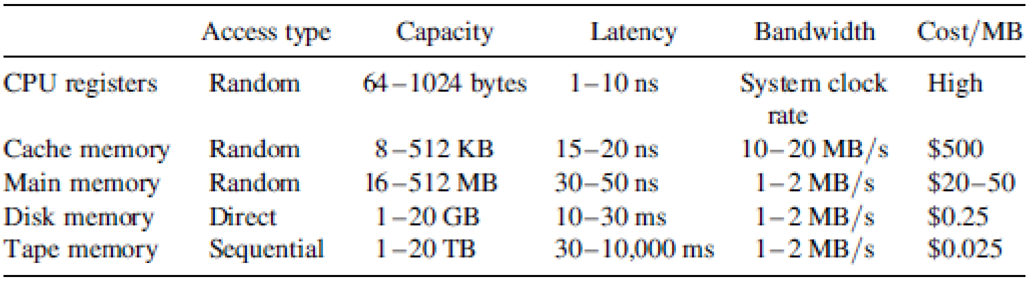
The memory hierarchy can be characterized by a number of parameters. Among these parameters are the access type, capacity, cycle time, latency, bandwidth, and cost.



Typical memory hierarchy

The effectiveness of a memory hierarchy depends on the principle of moving information into the fast memory infrequently and accessing it many times before replacing it with new information. This principle is possible due to a phenomenon called **locality of reference**; that is, within a given period of time, programs tend to reference a relatively confined area of memory repeatedly. ***There exist two forms of locality: spatial and temporal locality***. **Spatial locality** refers to the phenomenon that when a given address has been referenced, it is most likely that addresses near it will be referenced within a short period of time, for example, consecutive instructions in a straight line program. **Temporal locality**, on the other hand, refers to the phenomenon that once a particular memory item has been referenced, it is most likely that it will be referenced next, for example, an instruction in a program loop.

Unit 4
MEMORY

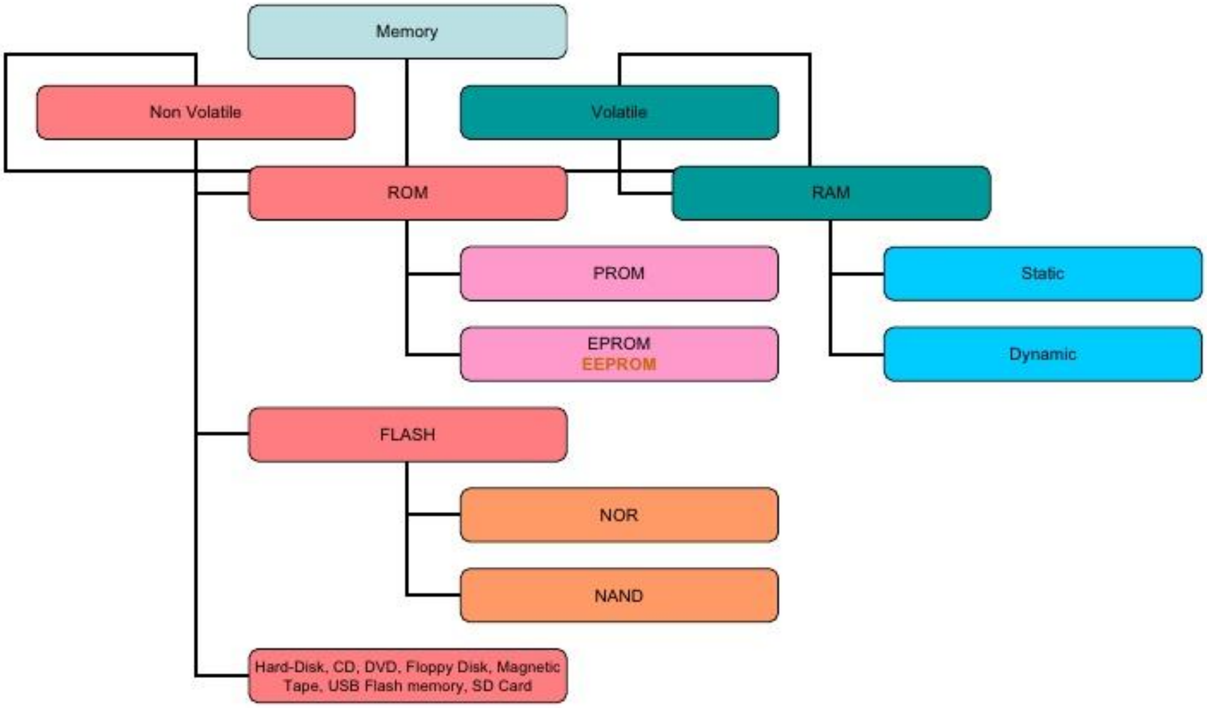


	Access type	Capacity	Latency	Bandwidth	Cost/MB
CPU registers	Random	64–1024 bytes	1–10 ns	System clock rate	High
Cache memory	Random	8–512 KB	15–20 ns	10–20 MB/s	$500
Main memory	Random	16–512 MB	30–50 ns	1–2 MB/s	$20–50
Disk memory	Direct	1–20 GB	10–30 ms	1–2 MB/s	$0.25
Tape memory	Sequential	1–20 TB	30–10,000 ms	1–2 MB/s	$0.025

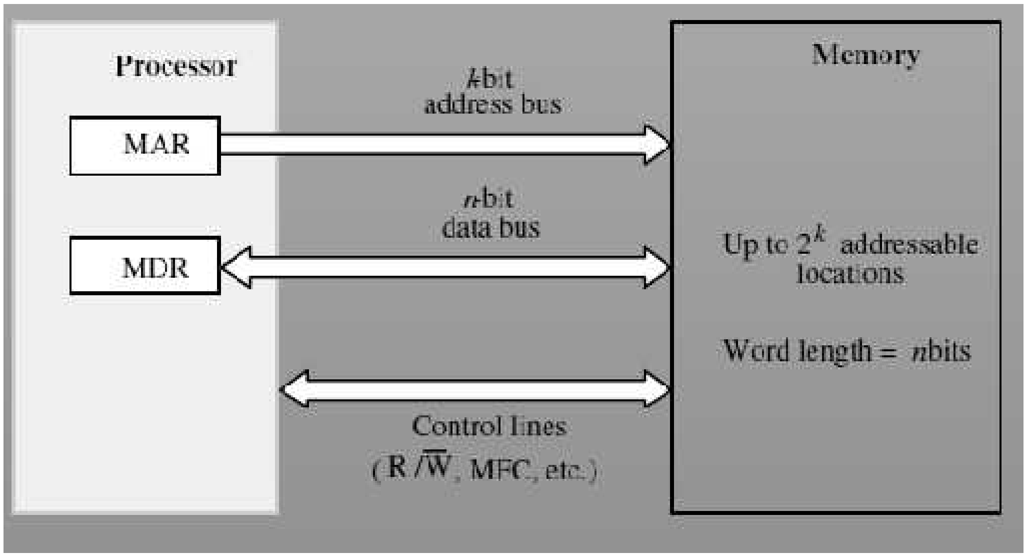
Memory Hierarchy Parameters

Semiconductor RAM memories

Memory Structure

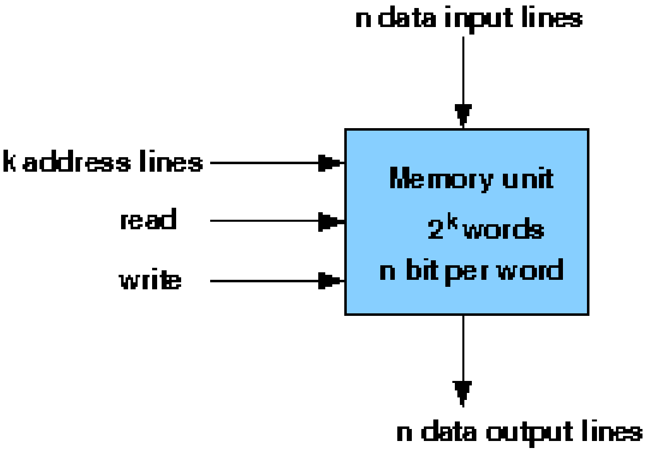

Memory
Non Volatile
Volatile
ROM
RAM
PROM
Static
EPROM
EEPROM
Dynamic
FLASH
NOR
NAND
Hard-Disk, CD, DVD, Floppy Disk, Magnetic Tape, USB Flash memory, SD Card

Unit 4
MEMORY



A typical CPU and main memory interface

Ram design

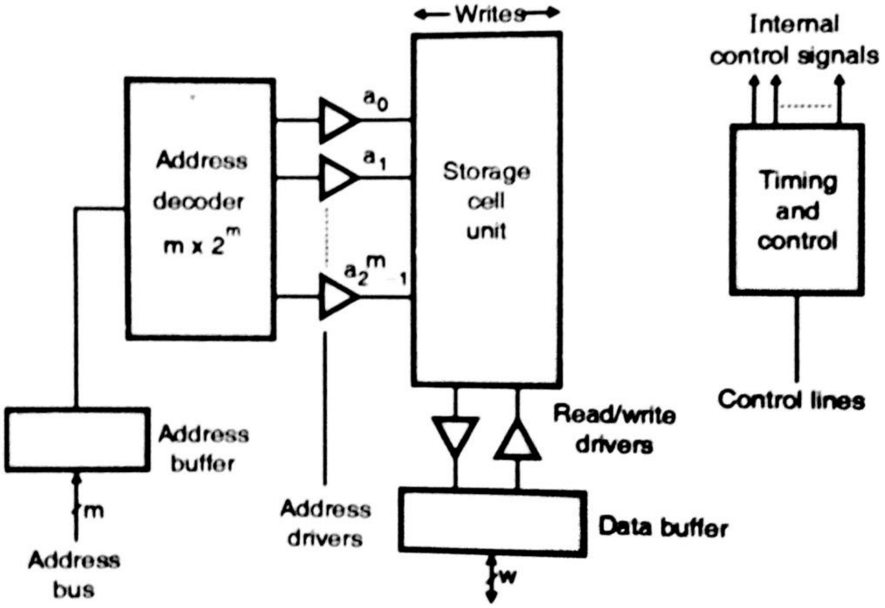


- m-bit address lines can be used to select a word
- There are 2^k words in RAM
- Each word is of n-bits

2-D (One-dimensional) Organization

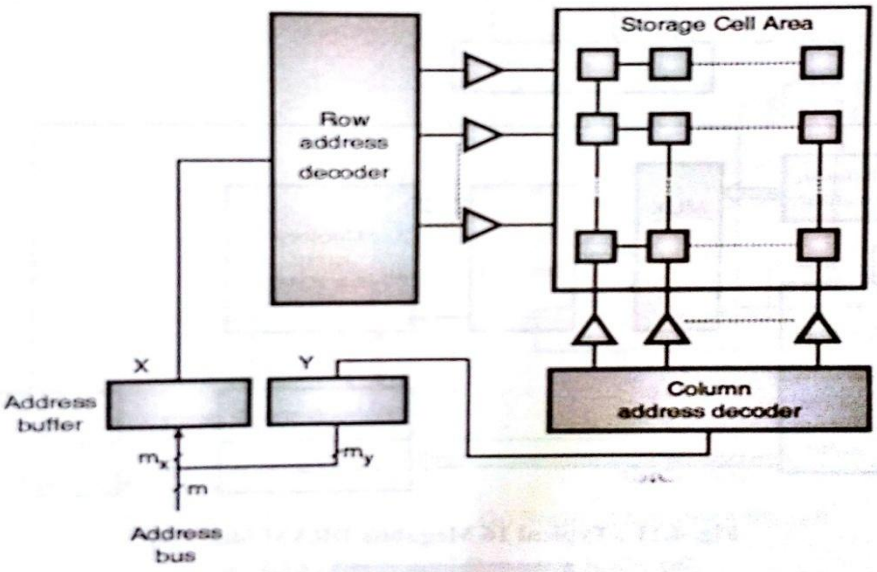
RAM with 2^m words of n bit each. The matrix of cells is formed by 2^m rows and n columns.

Unit 4
MEMORY



- Storage unit is composed of a large number (2^m) of addressable locations. Each location stores a w-bit word.
- An m-bit address is generated by the processor. It is stored into an address buffer. Output of address buffer is fed to the address decoder.
- The address decoder selects one of the address locations in the memory. The output of address decoder passes through tri-state buffers.
- The contents of a selected location are outputted through tri-state buffer to data buffer during read operation. The data bus is w-bit wide.

2 1/2D Ram Organization



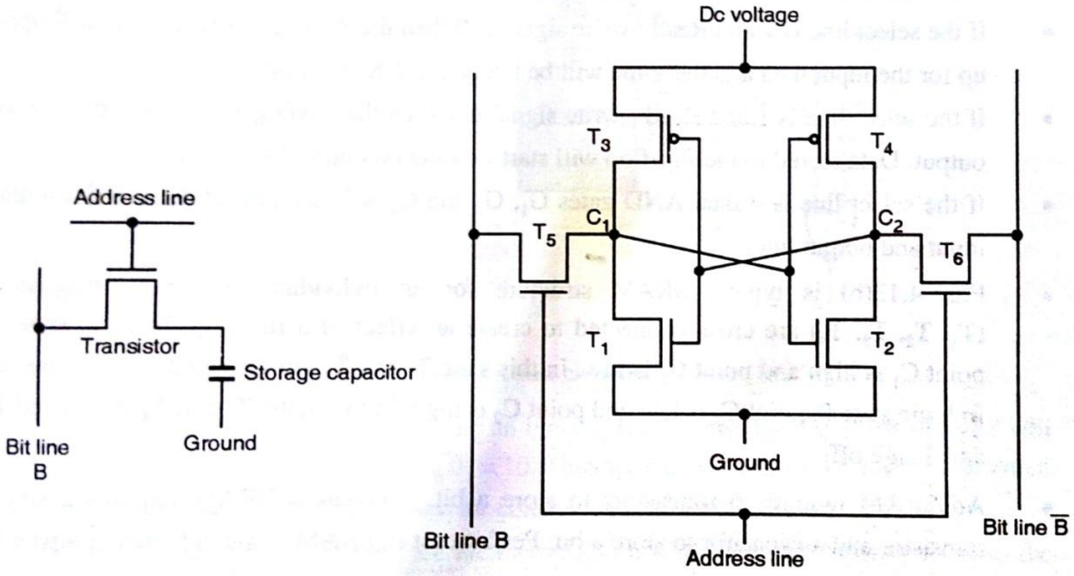
Unit 4
MEMORY

- The memory is organized as the matrix of cells, each of which stores a bit.
- A particular cell is selected using row-decoder and column decoder.
- It is cheaper to implement hamming error correction code. Correction is implemented row-wise.

Static Ram

SRAM is constructed using flip-flops. SRAM is capable of retaining its data (1/0) as long as power is applied.

- SRAM does not require refreshing as the data in the cell is retained as long as power is applied.
- Four transistors (T1, T2, T3, and T4) are cross connected to create an effect of a flip-flop.
- In logic state 1, C1=1 and C2=0. T1 and T4 are off & T2 and T3 are on.
- In logic state 0, C1=0 and C2=1. T1 and T4 are on & T2 and T3 are off.
- An SRAM require 6 transistors to store a bit. Where DRAM requires a single transistor and capacitor to store a bit.
- Cache is constructed using SRAM.



(a) Dynamic RAM (DRAM) cell **(b) Static RAM (SRAM) cell**

Dynamic Ram

- In DRAM cell the presence of data bit 1 or 0 corresponds to the pressure or absence of a stored charge in the capacitor.
- A dynamic cell can be constructed with a single transistor and hence a higher storage density can be achieved with DRAM
- DRAM is less expensive than a SRAM.
- Since the data in DRAM is stored in the form of charge, it leaks out a period of time DRAM requires periodic refreshment to preserve the charge in capacitor.
- R/w is suspended when refresh cycle is going on. This increase the effective access time of DRAM

Unit 4
MEMORY

Read only memory

ROM (Read Only Memory). Usually they are used in microprogramming of systems, library routines of frequent use, etc. The manufacturers use it when they produce components in a massive way.

PROM (Programmable Read Only Memory). The writing process is carried out electrically and can be made either by the provider or by the client after the manufacturing of the original chip, unlike the ROM that is recorded when it is assembled. PROM allows just a single recording and it is more expensive than ROM.

EPROM (Erasable Programmable Read Only Memory). EPROM can be written several times using electrical power. However, using ultra-violet raises all the data can be erased. This type of memory is more expensive than PROM.

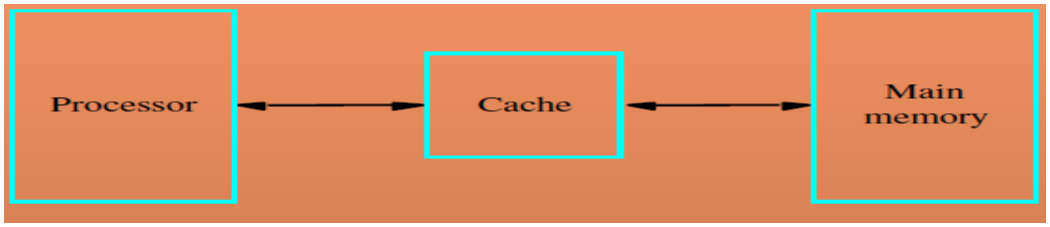
EEPROM (Electrically Erasable Programmable Read Only Memory). Data can be erased at byte level in a selective way using electrical power. It's more expensive than EPROM.

Flash Memory Denominated thus by the speed with which it can be reprogrammed. It uses selective electrical erasure at block of bytes level. It's cheaper than EEPROM.

High Speed Memories

Cache Memory & *Interleaved memory*

Cache Memory



Cache memory is a very high speed semiconductor memory which can speed up CPU. It acts as a buffer between the CPU and main memory. It is used to hold those parts of data and program which are most frequently used by CPU. The parts of data and programs are transferred from disk to cache memory by operating system, from where CPU can access them.

Advantages

The advantages of cache memory are as follows:

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.

Unit 4

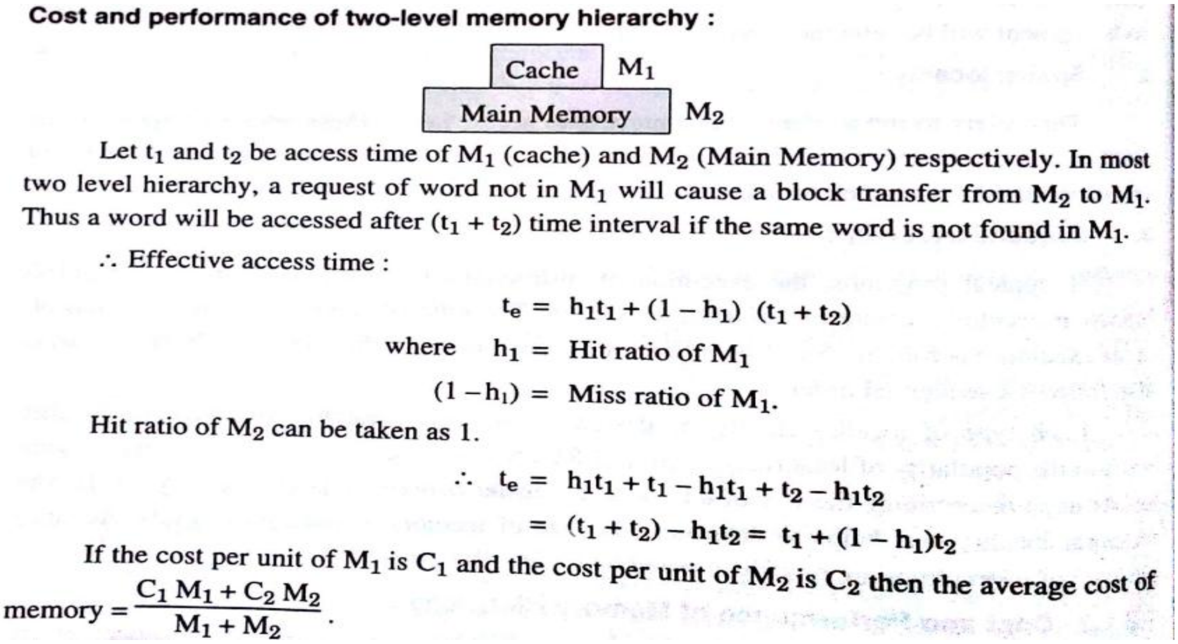
MEMORY

- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

Disadvantages

The disadvantages of cache memory are as follows:

- Cache memory has limited capacity.
- It is very expensive.



Cost and performance of two-level memory hierarchy :

Cache
M_1
Main Memory
M_2

Let t_1 and t_2 be access time of M_1 (cache) and M_2 (Main Memory) respectively. In most two level hierarchy, a request of word not in M_1 will cause a block transfer from M_2 to M_1. Thus a word will be accessed after $(t_1 + t_2)$ time interval if the same word is not found in M_1.

∴ Effective access time :

$$t_e = h_1 t_1 + (1 - h_1)\ (t_1 + t_2)$$

$$\text{where } h_1 = \text{Hit ratio of } M_1$$

$$(1 - h_1) = \text{Miss ratio of } M_1.$$

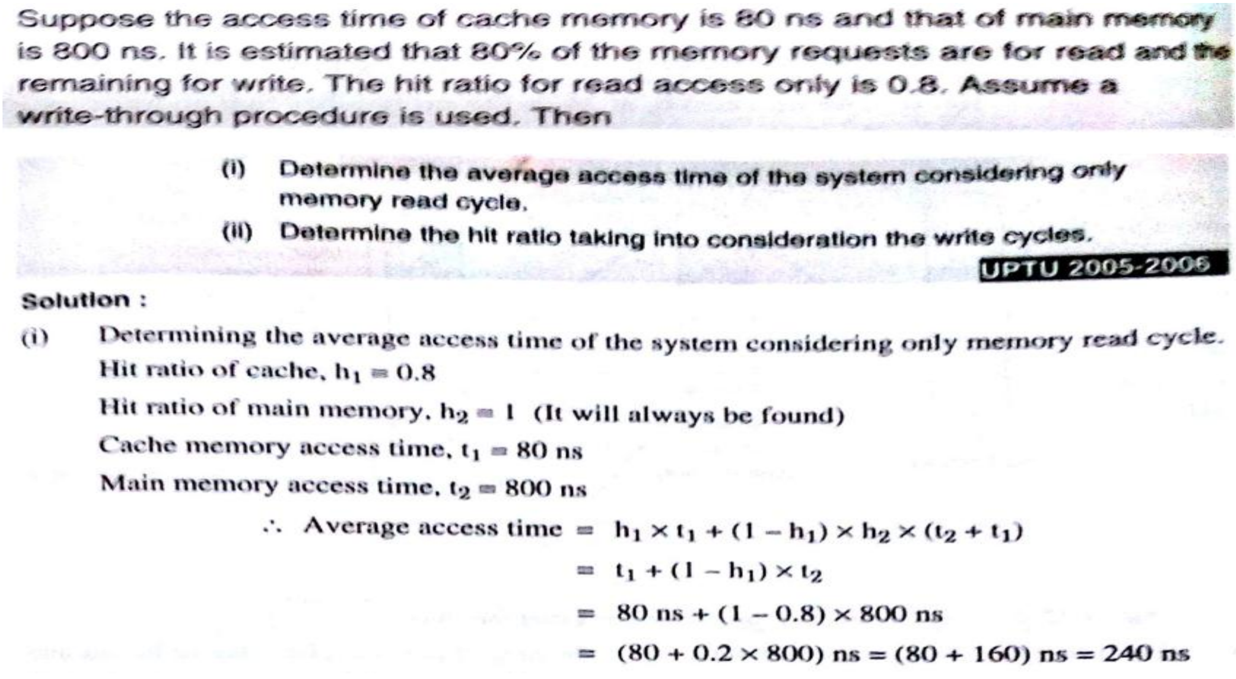
Hit ratio of M_2 can be taken as 1.

$$\therefore\ t_e = h_1 t_1 + t_1 - h_1 t_1 + t_2 - h_1 t_2$$

$$= (t_1 + t_2) - h_1 t_2 = t_1 + (1 - h_1) t_2$$

If the cost per unit of M_1 is C_1 and the cost per unit of M_2 is C_2 then the average cost of memory $= \dfrac{C_1 M_1 + C_2 M_2}{M_1 + M_2}$.

Example:



Suppose the access time of cache memory is 80 ns and that of main memory is 800 ns. It is estimated that 80% of the memory requests are for read and the remaining for write. The hit ratio for read access only is 0.8. Assume a write-through procedure is used. Then

(i) Determine the average access time of the system considering only memory read cycle.

(ii) Determine the hit ratio taking into consideration the write cycles.

UPTU 2005-2006

Solution :

(i) Determining the average access time of the system considering only memory read cycle.

Hit ratio of cache, $h_1 = 0.8$

Hit ratio of main memory, $h_2 = 1$ (It will always be found)

Cache memory access time, $t_1 = 80$ ns

Main memory access time, $t_2 = 800$ ns

$$\therefore \text{ Average access time } = h_1 \times t_1 + (1 - h_1) \times h_2 \times (t_2 + t_1)$$

$$= t_1 + (1 - h_1) \times t_2$$

$$= 80 \text{ ns} + (1 - 0.8) \times 800 \text{ ns}$$

$$= (80 + 0.2 \times 800) \text{ ns} = (80 + 160) \text{ ns} = 240 \text{ ns}$$

(ii) Determine the hit ratio taking into consideration the write cycle.

Since a write operation in a write through cache is treated as cache miss. The effective hit ratio $= 0.8 \times \frac{80}{100} = \mathbf{0.64}$.

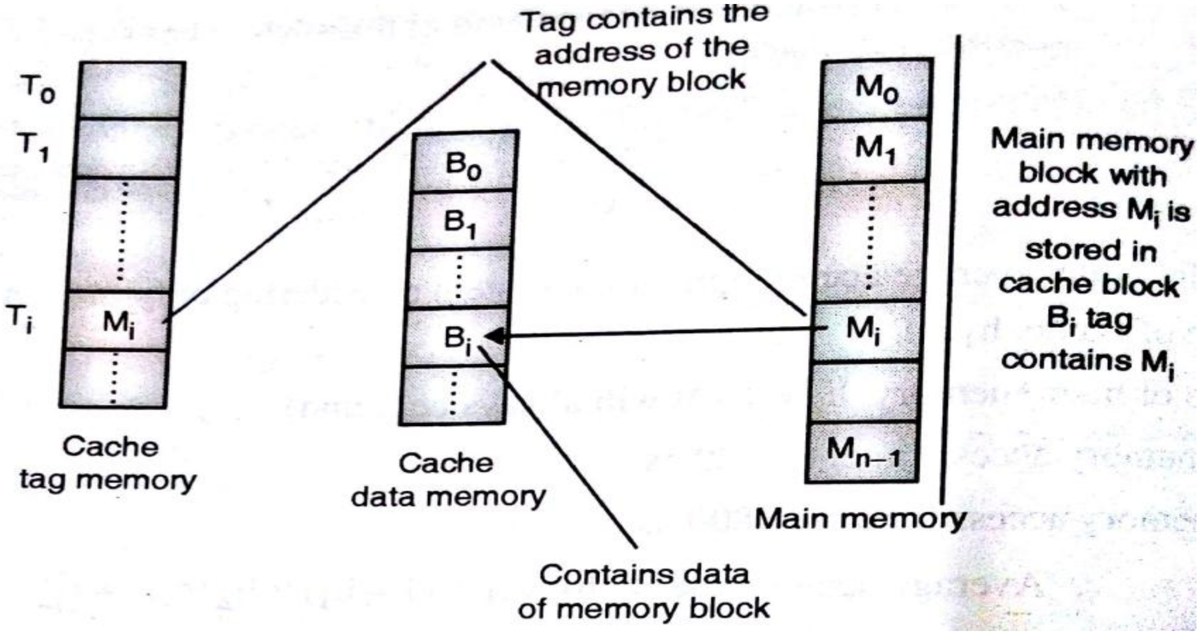
Cache Organization

Cache consists of

(a) cache data memory

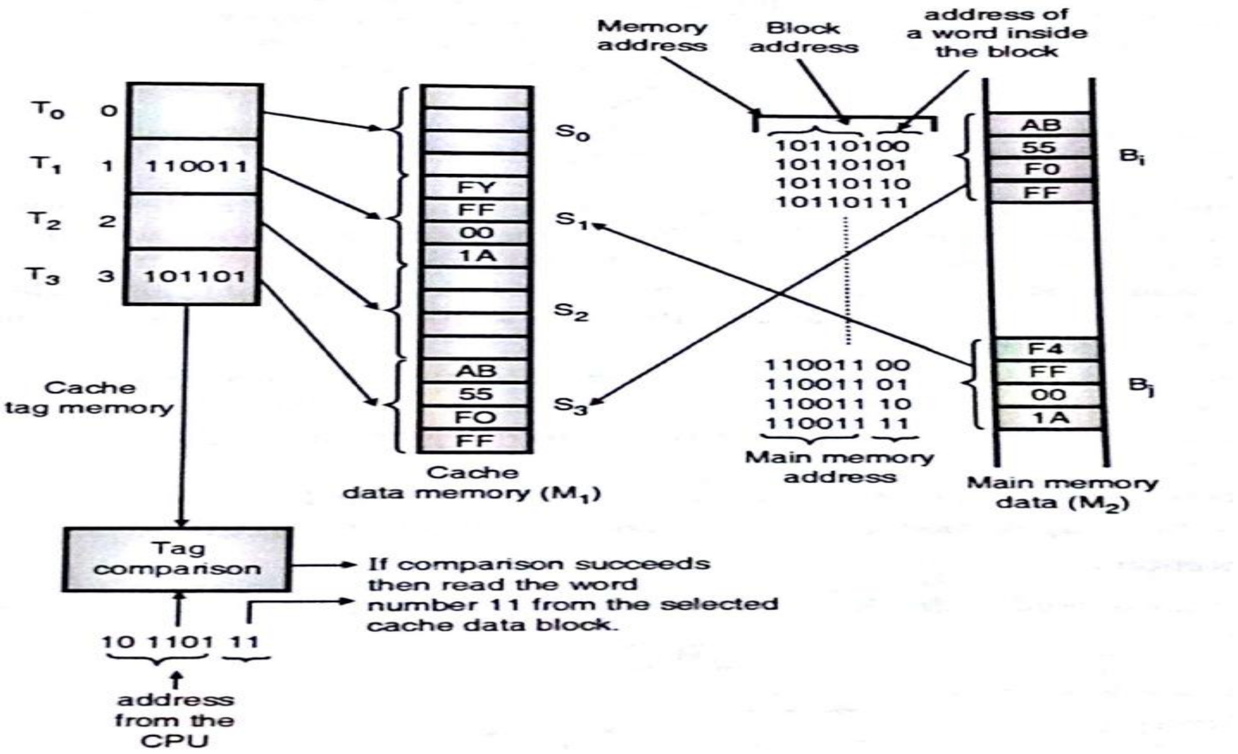
(b) tag

- Memory words are stored in a cache data memory. These words are grouped into a block. Basic data transfer unit between main memory and cache is a block of words.
- Each cache block is marked with its block address (address of memory block residing in cache), referred to as a tag.
- Tag indicates to what part of the memory space the block belongs.
- The collection of tag addresses is stored in special memory known as the cache tag memory or directory.
- Whenever a memory block with block address M_i is assigned to a cache block B_i then M_i is in the cache tag memory.


Tag contains the address of the memory block
T_0
T_1
T_i
M_i
Cache tag memory
B_0
B_1
B_i
Cache data memory
Contains data of memory block
M_0
M_1
M_i
M_{n-1}
Main memory
Main memory block with address M_i is stored in cache block B_i tag contains M_i

Unit 4
MEMORY

Cache Operation:



Address Mapping:

There are three mapping techniques—

(i) Direct Mapping (ii) Full associative mapping (iii) Set associative Mapping

Direct Mapping

$$\text{Cache slot (set) number} = \begin{pmatrix}\text{Block number of}\\ \text{main memory}\end{pmatrix} \begin{matrix}\text{modulo}\\ \text{operator}\end{matrix} \begin{pmatrix}\text{Total number of}\\ \text{slots in cache}\end{pmatrix}$$

In this technique it can easily be determined whether a memory block is in cache or not

Example:

Suppose number of ***main memory*** blocks --- 16 (B_0-B_{15})

Number of slots (sets) in cache memory – 4

Suppose we want to find a cache memory set in which the block number B_{11} should be stored

This can be done using above formula i.e. **(11) modulo operator (4)**

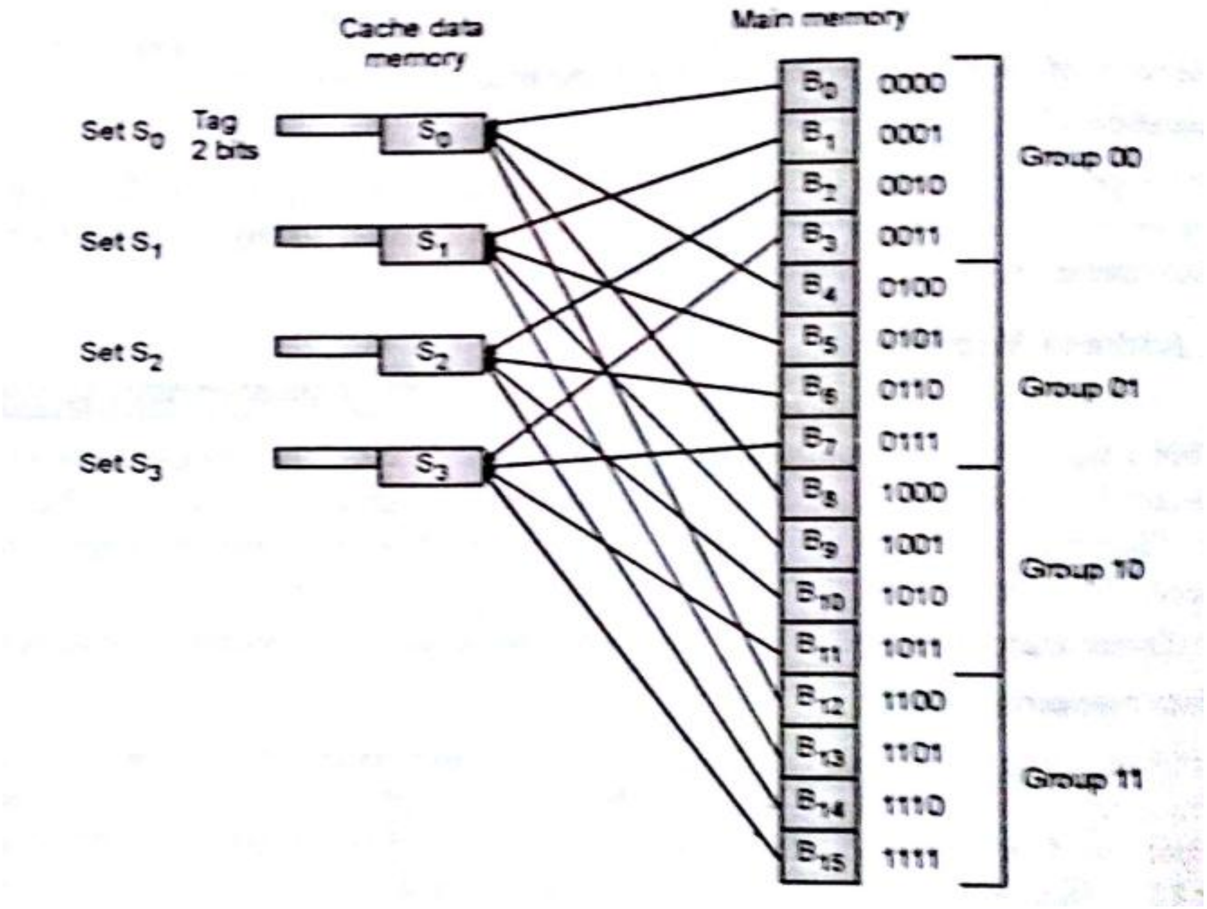
$$\frac{11}{4} = 2\frac{3}{4}$$

Quotient = 2, gives the group number/Tag Value

Remainder = 3, gives the set number of cache data memory

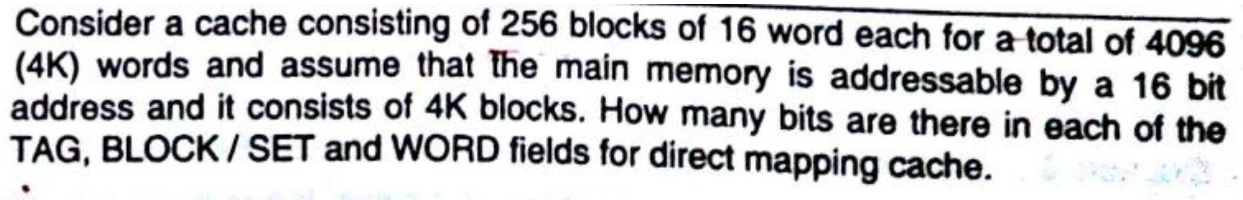
If the block no. is represented in binary then we can easily find the quotient and remainder

$(11)_{10} = (1011)_2$ (Here **10 is Tag value** and **11 set no. of cache**)



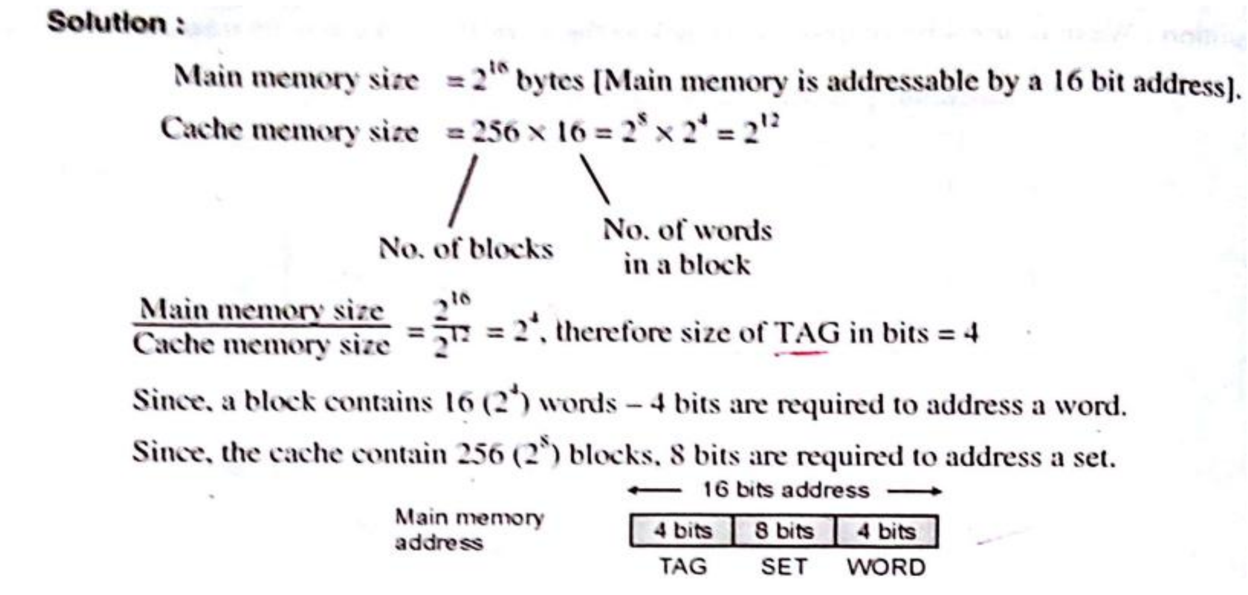
Direct Mapping

Examples:



Consider a cache consisting of 256 blocks of 16 word each for a total of 4096 (4K) words and assume that the main memory is addressable by a 16 bit address and it consists of 4K blocks. How many bits are there in each of the TAG, BLOCK / SET and WORD fields for direct mapping cache.

Unit 4
MEMORY



Solution :

Main memory size $= 2^{16}$ bytes [Main memory is addressable by a 16 bit address].

Cache memory size $= \underbrace{256}_{\text{No. of blocks}} \times \underbrace{16}_{\text{No. of words in a block}} = 2^8 \times 2^4 = 2^{12}$

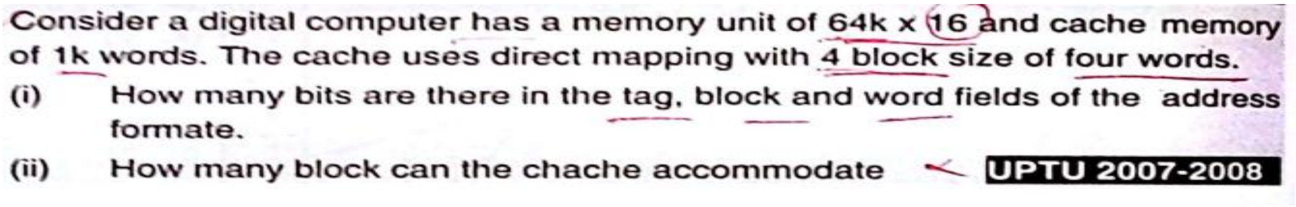
$\frac{\text{Main memory size}}{\text{Cache memory size}} = \frac{2^{16}}{2^{12}} = 2^4$, therefore size of TAG in bits = 4

Since, a block contains 16 (2^4) words – 4 bits are required to address a word.

Since, the cache contain 256 (2^8) blocks, 8 bits are required to address a set.

Main memory address ⟵ 16 bits address ⟶

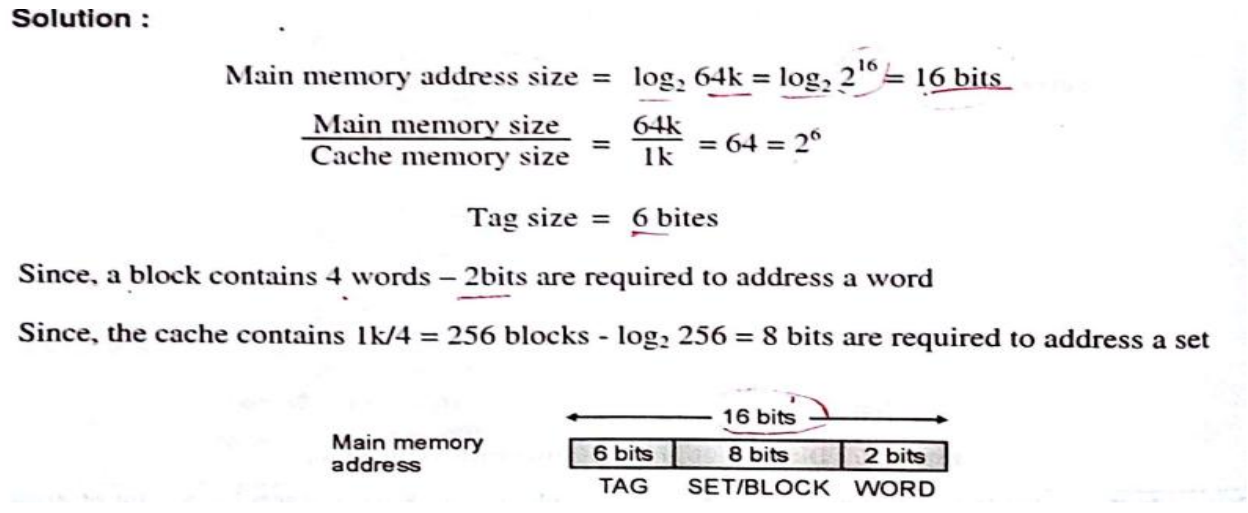
4 bits	8 bits	4 bits
TAG	SET	WORD



Consider a digital computer has a memory unit of 64k x 16 and cache memory of 1k words. The cache uses direct mapping with 4 block size of four words.

(i) How many bits are there in the tag, block and word fields of the address formate.

(ii) How many block can the chache accommodate **UPTU 2007-2008**



Solution :

Main memory address size = $\log_2 64k = \log_2 2^{16} = 16$ bits

$\frac{\text{Main memory size}}{\text{Cache memory size}} = \frac{64k}{1k} = 64 = 2^6$

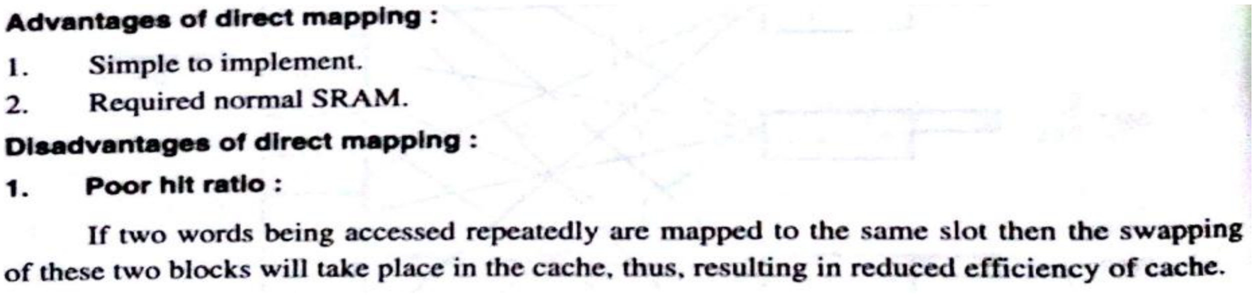
Tag size = 6 bites

Since, a block contains 4 words – 2bits are required to address a word

Since, the cache contains 1k/4 = 256 blocks - $\log_2 256 = 8$ bits are required to address a set

Main memory address ⟵ 16 bits ⟶

6 bits	8 bits	2 bits
TAG	SET/BLOCK	WORD



Advantages of direct mapping :

1. Simple to implement.
2. Required normal SRAM.

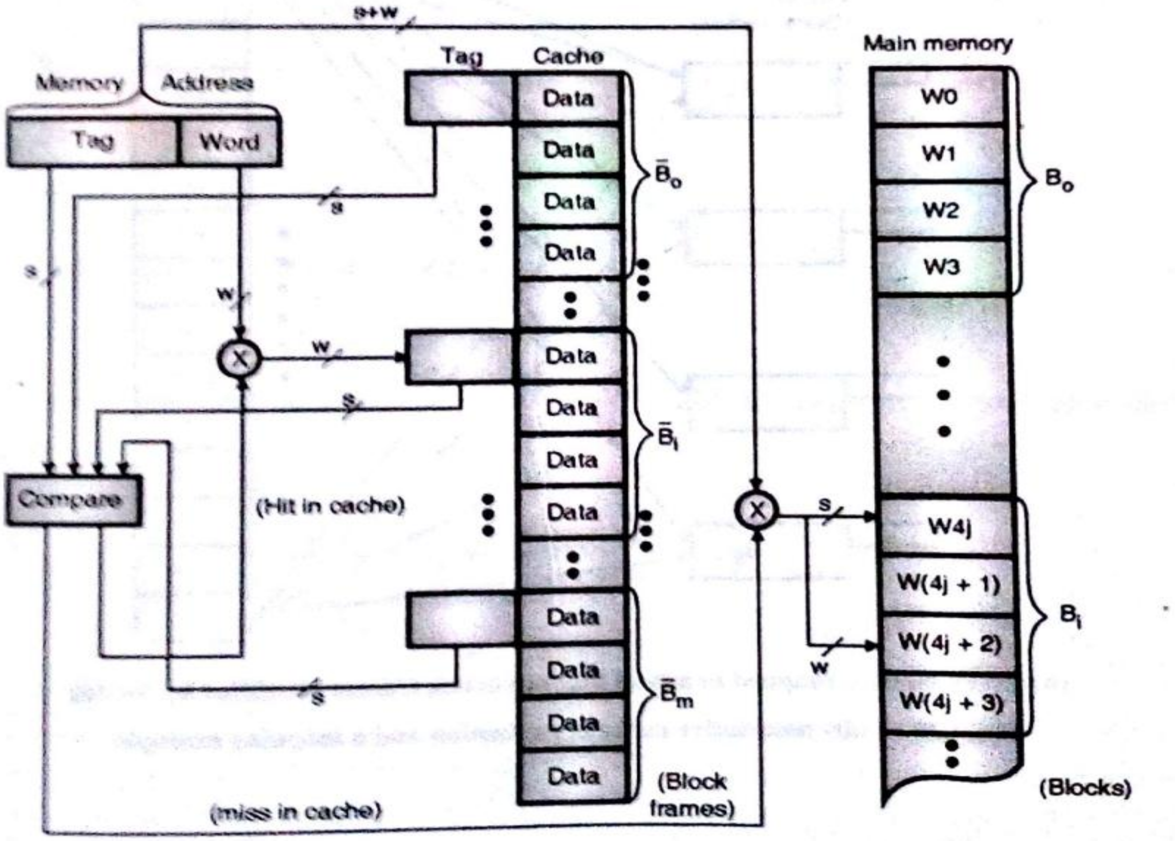
Disadvantages of direct mapping :

1. **Poor hit ratio :**

If two words being accessed repeatedly are mapped to the same slot then the swapping of these two blocks will take place in the cache, thus, resulting in reduced efficiency of cache.

Full Associative Mapping

The fastest technique for implementing tag comparison is associative or content addressing. Associative mapping permits the input tag to be compared simultaneously to all tags in the cache-tag memory. In an associative memory any stored item can be accessed by using the contents of the item in question, generally some specified sub-field as an address. Associative memories are also commonly known as content addressable memories (CAM).



Advantages :

1. Very high hit ratio.
2. Allows, better block replacement strategy with reduced contention.

Disadvantages :

1. Associative memory is very costly.
2. Significant increase in tag length.
3. Scheme is comparatively complex.

Unit 4
MEMORY

Consider a cache consisting of 16 words. Each block consists of 4 words. Size of the main memory is 256 bytes. Find number of bits in each of TAG, BLOCK / SET, and WORD for associative mapped cache ?

Solution :

$$\text{Number of blocks in main memory} = \frac{256}{4} = 64$$

$$\therefore \text{ Tag size} = \log_2 64 = \log_2 2^6 = 6 \text{ bits}$$

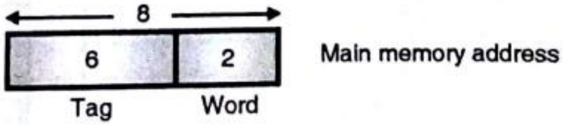
$$\text{Word length} = \log_2 \text{(Number of words in a block)}$$

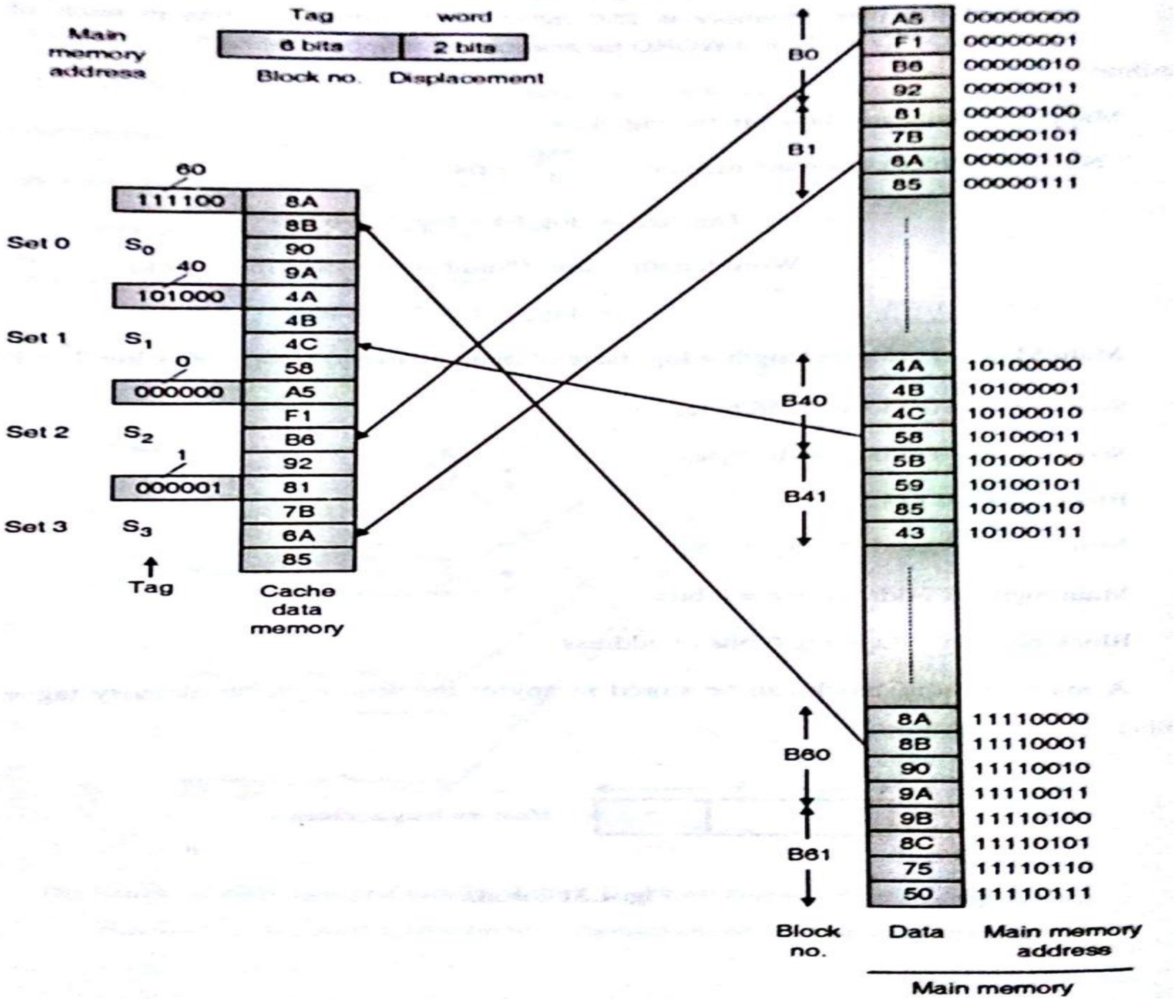
$$= \log_2 4 = \log_2 2^2 = 2 \text{ bits}$$

$$\text{Main Memory address length} = \log_2 \text{(size of main memory)} = \log_2 256 = \log_2 2^8 = 8 \text{ bits}$$

- Size of main memory = 256 bytes.
- Size of cache memory = 16 bytes.
- Block size = 4 bytes.
- Number of sets = 4 (S_0, S_1, S_2, S_3).
- Main memory address size = 8 bits.
- Block number = Leading 6 bits of address.

A main memory block can be stored in any of the sets of cache memory tag = block number of main memory.





Set Associative Mapping:

Set-associative cache tries to take advantage of both direct mapping and associative mapping.

1. Direct mapped cache requires normal SRAM.
2. Associate mapping offers high hit-ratio.

Set associative cache is implemented using normal SRAM and provides a high hit ratio.

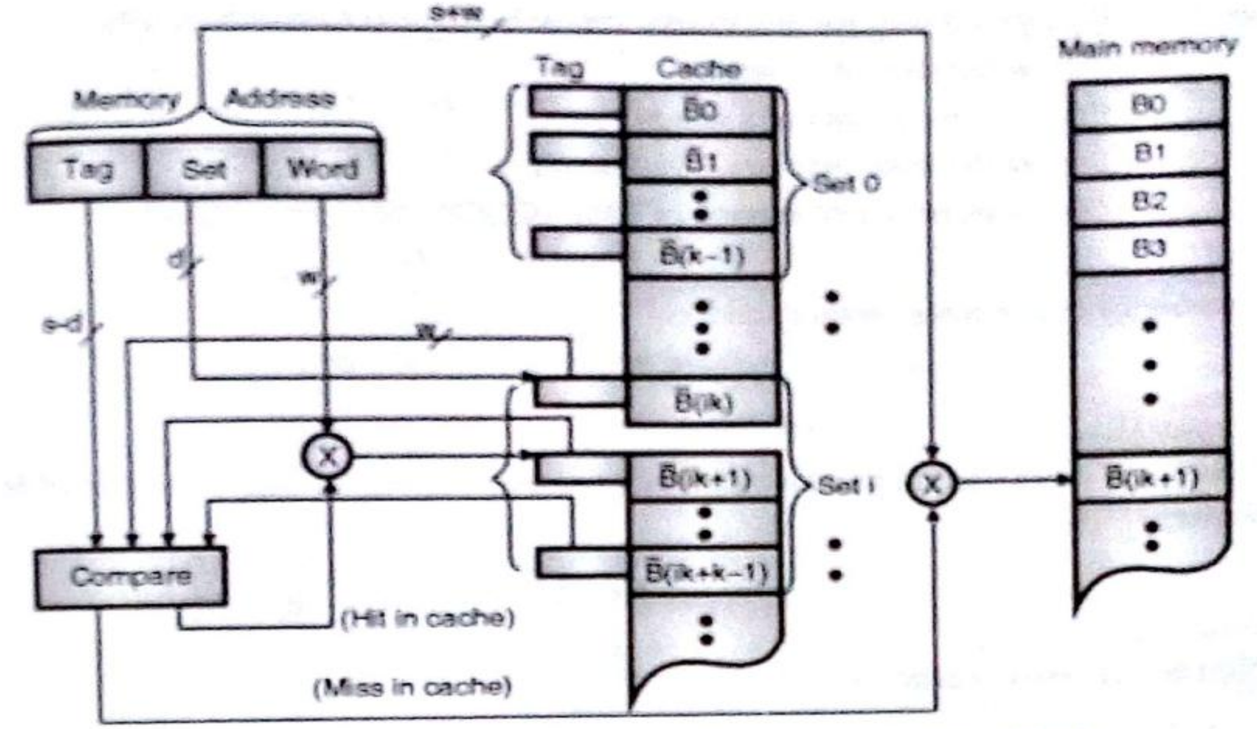
If properly designed, this cache may offer the best performance-cost ratio. Most high-performance computer system are based on this.

- In a K-way associative cache, the m cache blocks are divided into **V = m / k** sets.
- Each set contains K blocks.
- A memory block is mapped to a set using direct mapping.
- If each set is identified by d bits then

$$2^d = V$$

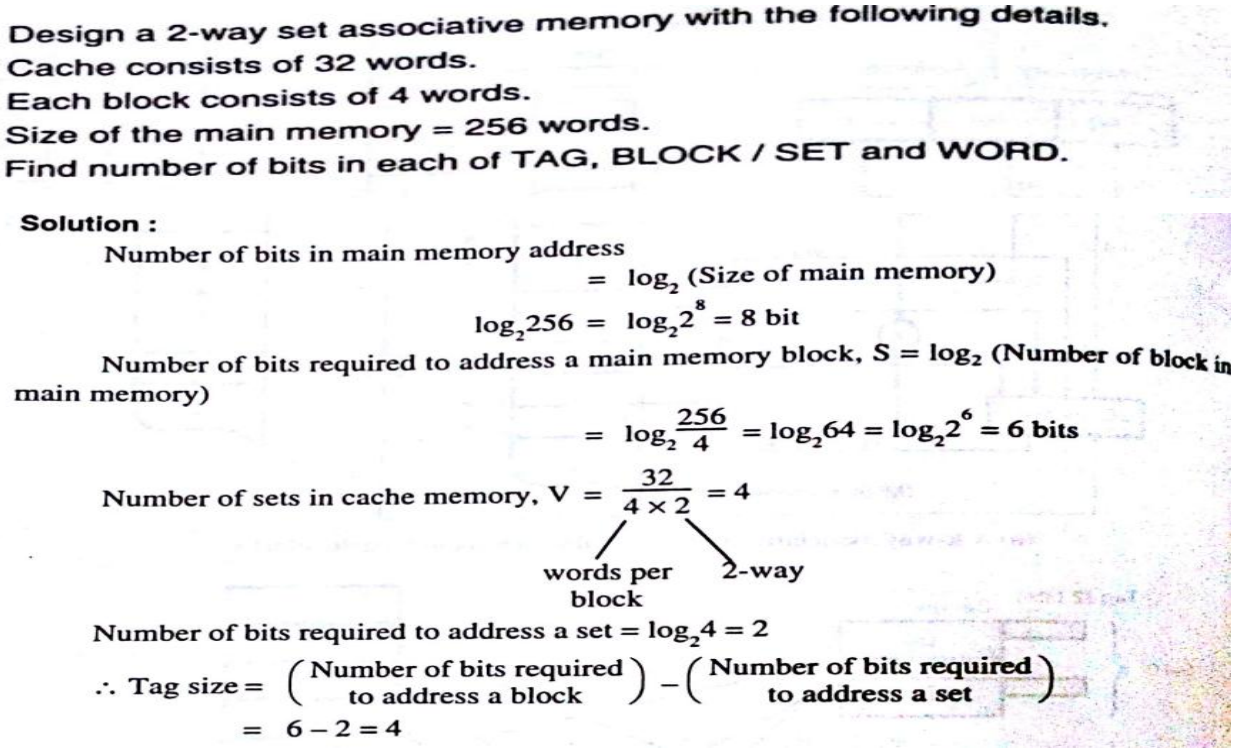
- If S bits are required to access a main memory block then tag length = S – d.
- A block of main memory can be stored in any of the frames inside the addressed set (associative mapping).

Unit 4
MEMORY



(a) A k-way associative search within each set of k cache blocks

Examples



Design a 2-way set associative memory with the following details.
Cache consists of 32 words.
Each block consists of 4 words.
Size of the main memory = 256 words.
Find number of bits in each of TAG, BLOCK / SET and WORD.

Solution :

Number of bits in main memory address

$$= \log_2 \text{(Size of main memory)}$$

$$\log_2 256 = \log_2 2^8 = 8 \text{ bit}$$

Number of bits required to address a main memory block, S = log₂ **(Number of block in** main memory)

$$= \log_2 \frac{256}{4} = \log_2 64 = \log_2 2^6 = 6 \text{ bits}$$

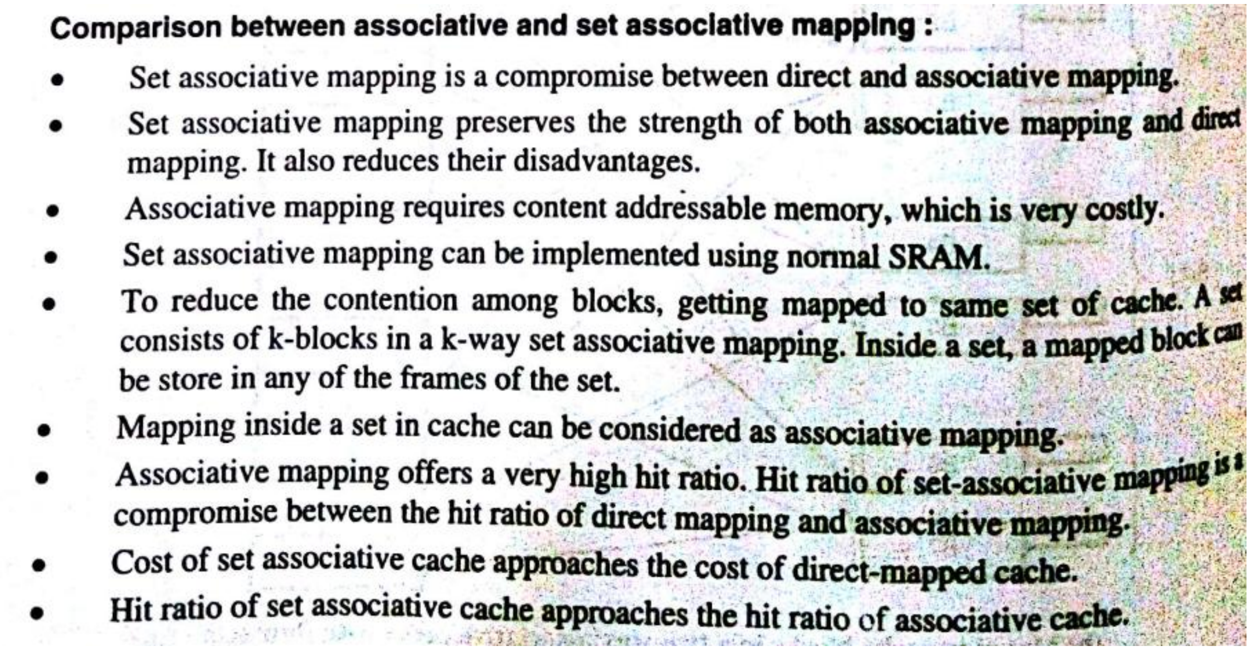
$$\text{Number of sets in cache memory, } V = \frac{32}{4 \times 2} = 4$$

words per block 2-way

Number of bits required to address a set $= \log_2 4 = 2$

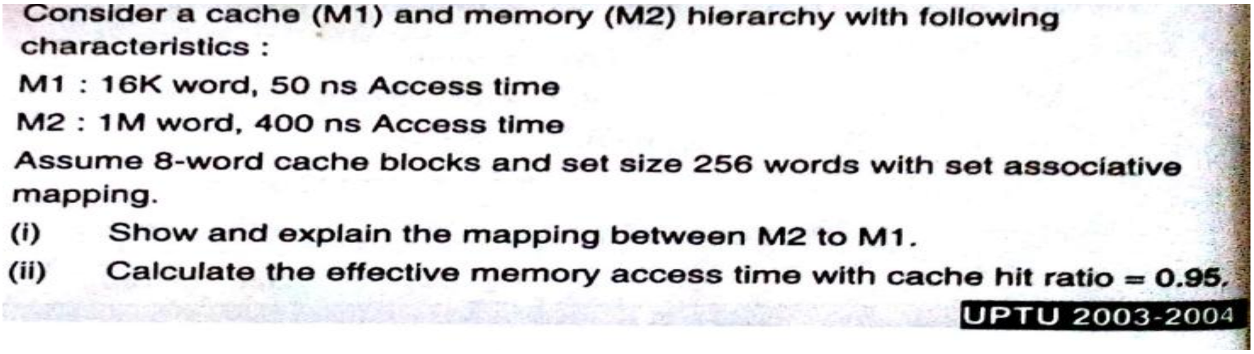
$$\therefore \text{ Tag size} = \left(\begin{matrix}\text{Number of bits required}\\ \text{to address a block}\end{matrix}\right) - \left(\begin{matrix}\text{Number of bits required}\\ \text{to address a set}\end{matrix}\right)$$

$$= 6 - 2 = 4$$



Comparison between associative and set associative mapping :

- Set associative mapping is a compromise between direct and associative mapping.
- Set associative mapping preserves the strength of both associative mapping and direct mapping. It also reduces their disadvantages.
- Associative mapping requires content addressable memory, which is very costly.
- Set associative mapping can be implemented using normal SRAM.
- To reduce the contention among blocks, getting mapped to same set of cache. A set consists of k-blocks in a k-way set associative mapping. Inside a set, a mapped block can be store in any of the frames of the set.
- Mapping inside a set in cache can be considered as associative mapping.
- Associative mapping offers a very high hit ratio. Hit ratio of set-associative mapping is a compromise between the hit ratio of direct mapping and associative mapping.
- Cost of set associative cache approaches the cost of direct-mapped cache.
- Hit ratio of set associative cache approaches the hit ratio of associative cache.



Consider a cache (M1) and memory (M2) hierarchy with following characteristics :

M1 : 16K word, 50 ns Access time

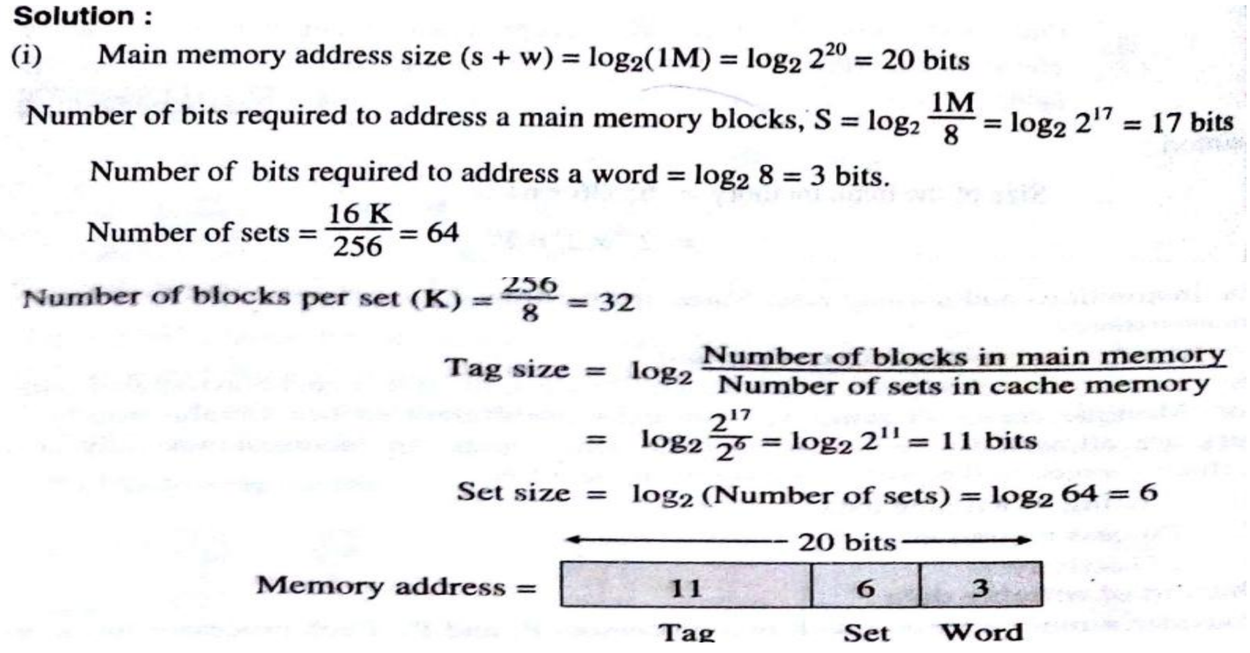
M2 : 1M word, 400 ns Access time

Assume 8-word cache blocks and set size 256 words with set associative mapping.

(i) Show and explain the mapping between M2 to M1.

(ii) Calculate the effective memory access time with cache hit ratio = 0.95.

UPTU 2003-2004



Solution :

(i) Main memory address size (s + w) = $\log_2(1M) = \log_2 2^{20} = 20$ bits

Number of bits required to address a main memory blocks, $S = \log_2 \frac{1M}{8} = \log_2 2^{17} = 17$ bits

Number of bits required to address a word = $\log_2 8 = 3$ bits.

Number of sets = $\frac{16\ K}{256} = 64$

Number of blocks per set (K) = $\frac{256}{8} = 32$

$$\text{Tag size} = \log_2 \frac{\text{Number of blocks in main memory}}{\text{Number of sets in cache memory}}$$

$$= \log_2 \frac{2^{17}}{2^6} = \log_2 2^{11} = 11 \text{ bits}$$

$$\text{Set size} = \log_2 (\text{Number of sets}) = \log_2 64 = 6$$

← 20 bits →

Memory address =

11	6	3
Tag	Set	Word

ii) Access time of cache, $t_1 = 50$ ns

Hit ratio of cache, $h_1 = 0.95$

Access time of main memory, $t_2 = 400$ ns

Hit ratio of main memory, $h_2 = 1$

$\therefore$ Effective access time $= h_1t_1 + (1 - h_1) \times h_2 \times (t_2 + t_1) = t_1 + (1 - h_1)\, t_2$

$= 50 \text{ ns} + 20 \text{ ns} = \mathbf{70\ ns}$

Cache Coherence:

The contents of cache and main memory can be altered by more than one devices e.g. CPU can write to cache and input / output modulus or DMA (Direct Memory Access) can directly write to main memory. This can result in inconsistencies in the values of cache and main memories.

Techniques for cache coherence in single CPU system :

UPTU 2001-2002

1. **Write through :** Write the data in cache as well as main memory. This will always keep the two copies of data (one in cache and other in main memory) consistent. The disadvantage of this technique is that a bottleneck is created due to large number of access to main memory.
2. **Write block :** In this method updates are made only in the cache, setting a bit called update bit. Only those blocks whose update bit is set is replaced in the main memory. But here all the accesses to main memory whether from the CPU or input / output modules need to be from the cache resulting in complex circuitry.
3. **Instruction cache :** An instruction cache is one, which is employed for accessing only the instructions and nothing else. Since, instructions do not change, there will not be any inconsistency.

Cache coherence in multiprocessor system :

general, three sources of the problems have been identified :

1. Sharing of writable data.
2. Process migration.
3. I/O activity.

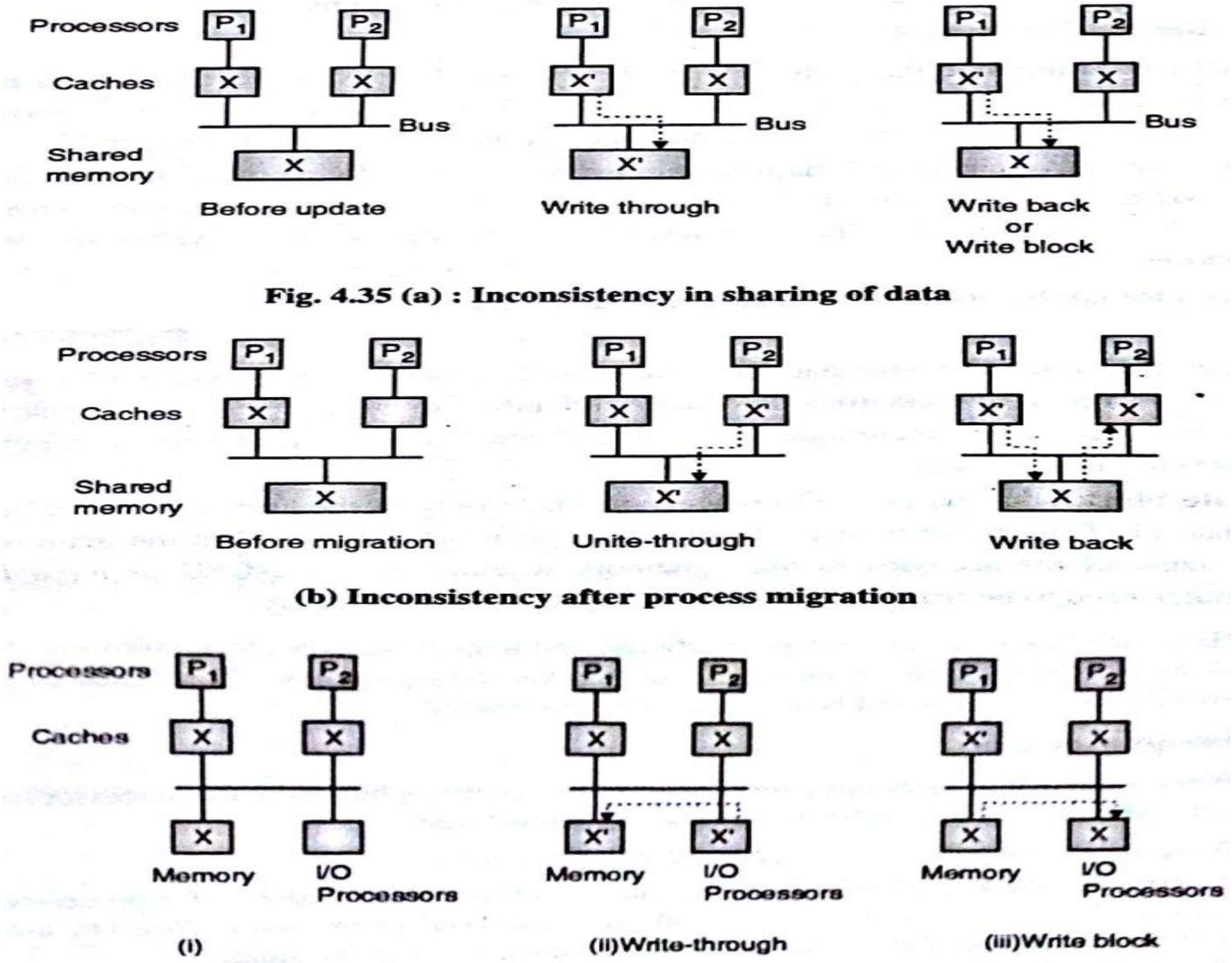


Fig. 4.35 (a) : Inconsistency in sharing of data

(b) Inconsistency after process migration

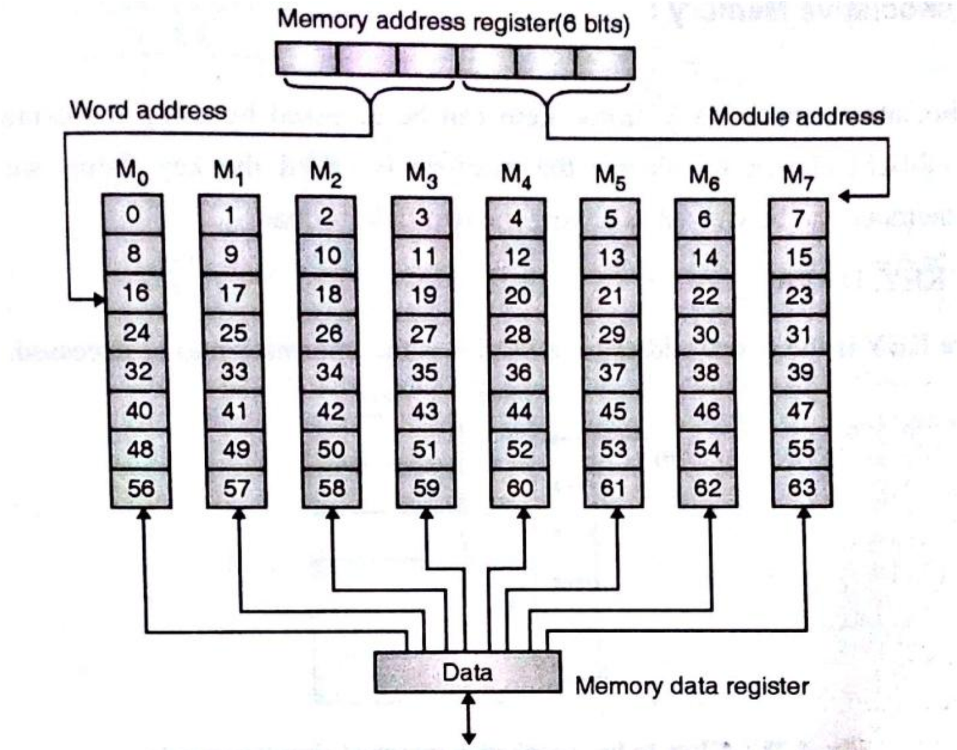
(c) I/O operation bypassing the cache

Cache coherence strategies have been divided into two categories :

1. **Software solutions**
2. **Hardware solutions**
   - **(i) Snoopy protocols**
     - **(a) Write-invalidate protocols (MESI)**
     - **(b) Write-update protocols**
   - **(ii) Directory protocols**

Interleaved memory:

Interleaved memory implements the concept of accessing more words in single memory access cycle. Memory can be partitioned into N separate memory modules. Thus N accesses can be carried out to the memory simultaneously. Once presented with a memory address, each memory module returns one word per cycle. It is possible to present different addresses to different memory modules so that parallel access to multiple words can be done simultaneously or in a pipelined fashion. The maximum processor bandwidth in interleaved memory can be equal to the number of modules i.e. N words per cycle.



M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

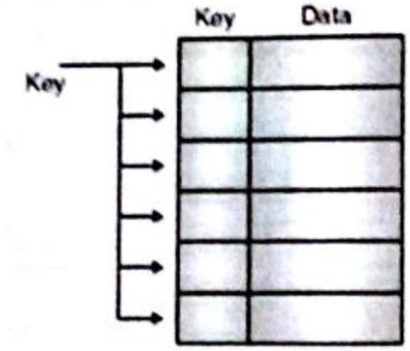
Eight-way low-order interleaving (absolute address shown in each memory word)

Associative Memory:

In associative memory any stored item can be accessed by using the contents of th item. The subfield chosen to address the memory is called the key. Items stored in a associative memory can be viewed as having the two field format :

KEY, DATA

Where KEY is the stored address and DATA is the information to be accessed.



Unit 4
MEMORY

Virtual Memory Technique:

Techniques that automatically move program and data blocks into the physical main memory when they are required for execution are called virtual memory techniques.

- Program or processor references an instruction and data space that is independent of available physical memory space.
- The address issued by the processor (either instruction or data) is called virtual or logical address.
- The virtual address is translated into physical address by a combination of hardware and software components.
- IF a virtual address refers to a part of the program or data space that is currently in the physical memory, then it is accessed immediately.
- If the referenced address is not in the main memory, its contents must be brought into main memory before it can be used.

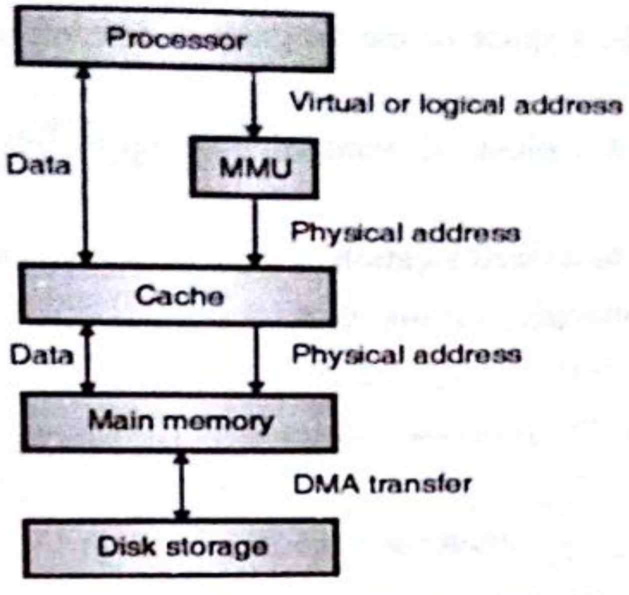
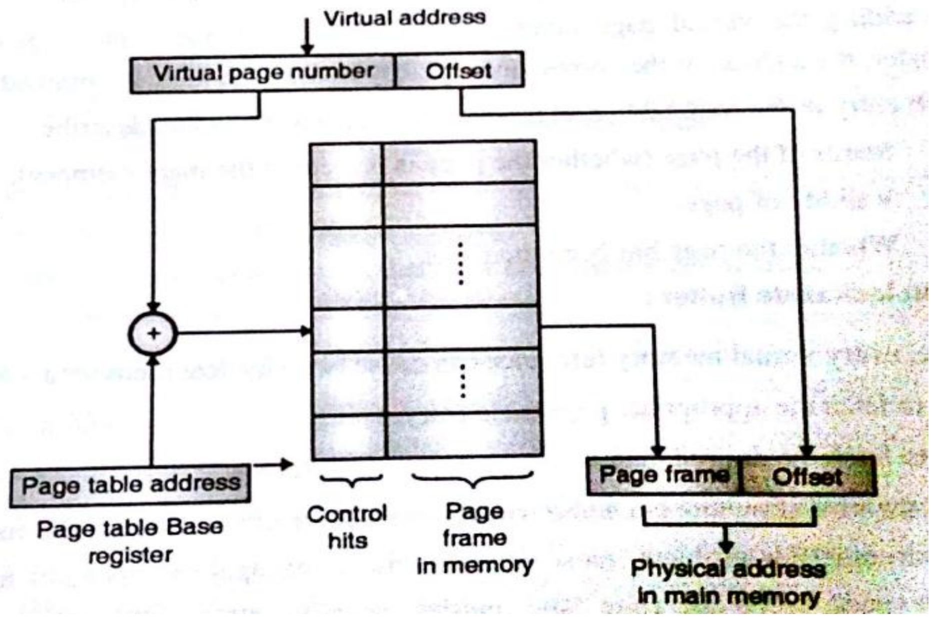


Fig. 4.40 : Virtual memory organization

Unit 4
MEMORY

Paging:



- Virtual (logical) address space of the program is divided into fixed length units called, pages.
- Each page consists of a block of words that occupy contiguous locations in the main memory.
- Each page is mapped to a fixed location in main memory called page frame.
- Page table stores the mapping information

  Virtual page number → Page frame
- Address generated by the processor to fetch a word memory can be divided into two parts :

Virtual page number	Offset

Translation lookaside Buffer :

In principle, every virtual memory reference can cause two physical memory accesses :

1. One to fetch the appropriate page table entry.
2. One to fetch the desired data.

Thus the virtual memory scheme would have the effect of doubling the memory access time. To overcome this problem, most virtual memory management schemes make use of a special high-speed cache for page table entries, usually called a **Translation Lookaside Buffer (TLB).**

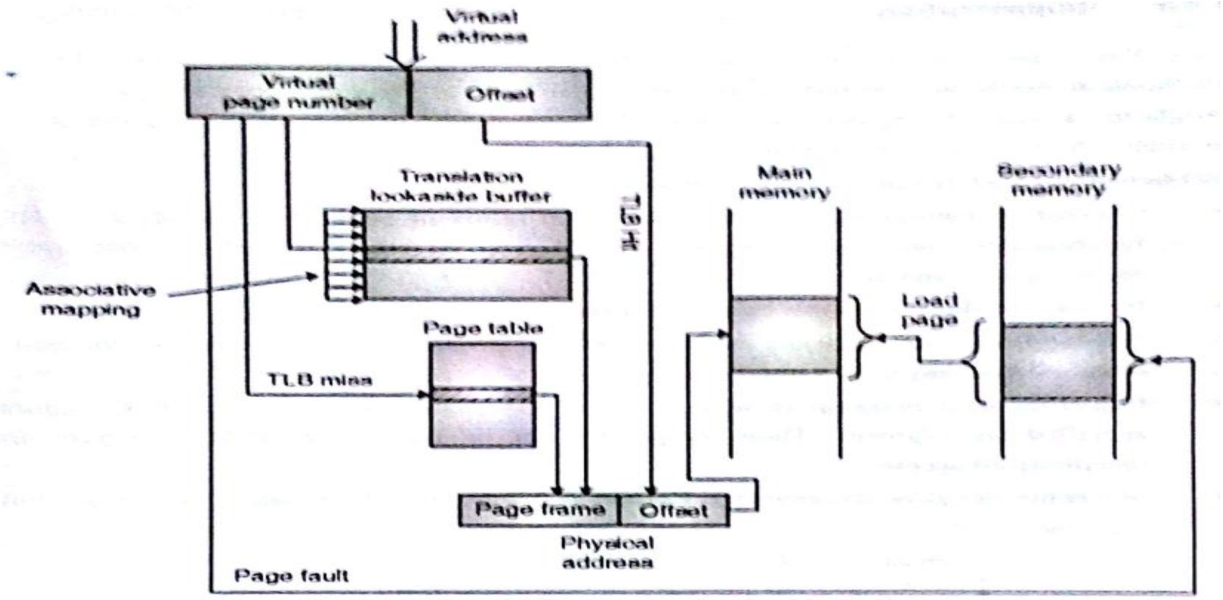
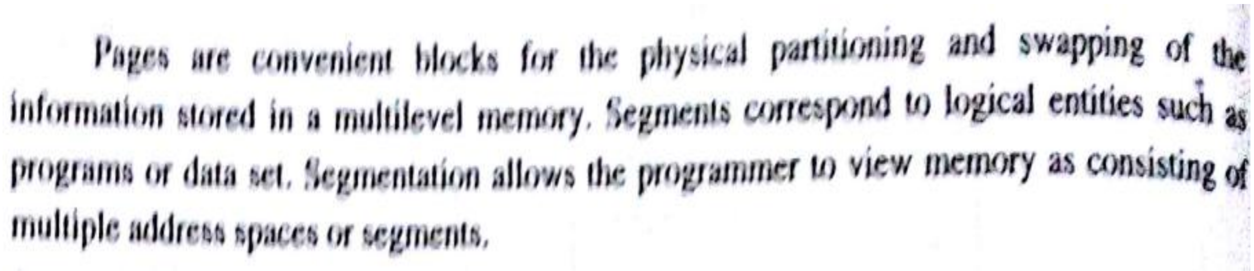


Fig. 4.42 : Use of a Translation Lookaside Buffer

- TLB contains those page table entries that have been most recently used.
- An entry of TLB contains :
  1. Virtual page number
  2. Control bits
  3. Page frame number in memory
- Address translation proceeds as follows. Given a virtual address, the processor looks in the TLB for the referenced page. If the page table entry for this page is found in the TLB, the physical address is obtained immediately. If there is a miss in the TLB, then the required entry is obtained from the page table in the main memory, and the TLB is updated.
- When a program generates an access request to a page that is not in the main memory, a page fault is said to have occurred. The whole page must be brought from the disk into the memory before access can proceed.

Segmentation:

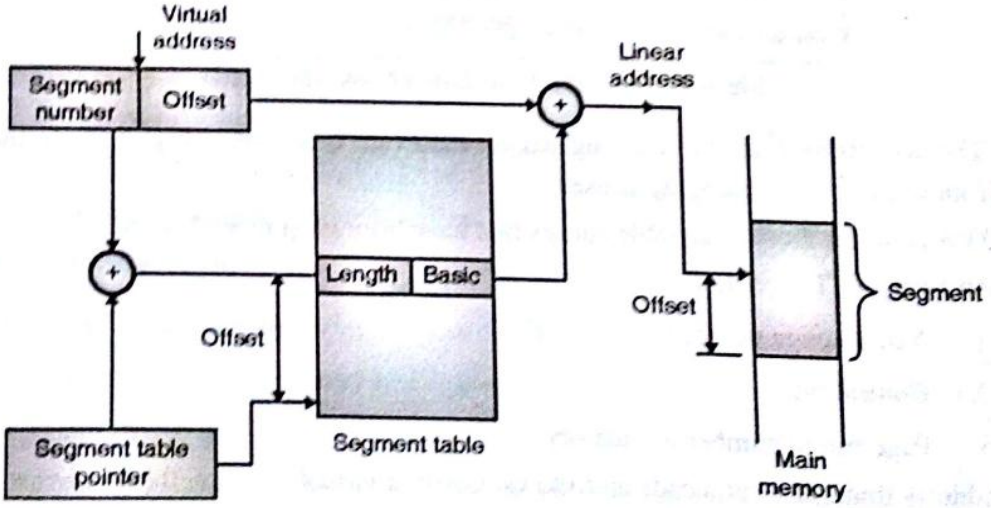


Pages are convenient blocks for the physical partitioning and swapping of the information stored in a multilevel memory. Segments correspond to logical entities such as programs or data set. Segmentation allows the programmer to view memory as consisting of multiple address spaces or segments.

Unit 4

MEMORY

- **Segments can grow dynamically. Certain segment types-stacks and queues grow during run time.**



(End Of Unit)