UNIT-IV: THE MEMORY SYSTEM

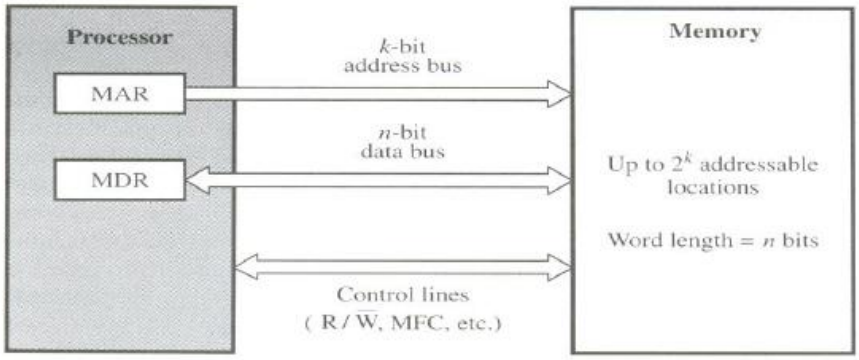
Basic Concepts, Semiconductor RAM, Types of Read-only Memory (ROM), Cache Memory, Performance Considerations, Virtual Memory, Secondary Storage.

4.1 Basic Concepts:

The maximum size of the memory that can be used in any computer is determined by the addressing scheme.

Address	Memory Locations
16 Bit	$2^{16} = 64$ K
32 Bit	$2^{32} = 4$G (Giga)
40 Bit	$2^{40} =$ IT (Tera)

Fig: Connection of Memory to Processor:



If MAR is k bits long and MDR is n bits long, then the memory may contain upto 2^K addressable locations and the n-bits of data are transferred between the memory and processor.

This transfer takes place over the processor bus.

The processor bus has,

- Address Line
- Data Line
- Control Line (R/W, MFC – Memory Function Completed)

The control line is used for co-ordinating data transfer.

The processor reads the data from the memory by loading the address of the required memory location into MAR and setting the R/W line to 1.

The memory responds by placing the data from the addressed location onto the data lines and confirms this action by asserting MFC signal.

Upon receipt of MFC signal, the processor loads the data onto the data lines into MDR register.

The processor writes the data into the memory location by loading the address of this location into MAR and loading the data into MDR sets the R/W line to 0.

- Measures for the speed of a memory:
  - memory access time.
    - It is the time that elapses between the initiation of an Operation and the completion of that operation.
  - memory cycle time.
    - It is the minimum time delay that required between the initiation of the two successive memory operations.

RAM (Random Access Memory):

In RAM, if any location that can be accessed for a Read/Write operation in fixed amount of time, it is independent of the location's address.

Cache Memory:

It is a small, fast memory that is inserted between the larger slower main memory and the processor.
It holds the currently active segments of a program and their data.

Virtual memory:

The address generated by the processor does not directly specify the physical locations in the memory.

The address generated by the processor is referred to as a virtual / logical address.

The virtual address space is mapped onto the physical memory where data are actually stored.

The mapping function is implemented by a special memory control circuit is often called the memory management unit.

Only the active portion of the address space is mapped into locations in the physical memory.

The remaining virtual addresses are mapped onto the bulk storage devices used, which are usually magnetic disk.

As the active portion of the virtual address space changes during program execution, the memory management unit changes the mapping function and transfers the data between disk and memory.

Thus, during every memory cycle, an address processing mechanism determines whether the addressed in function is in the physical memory unit.

If it is, then the proper word is accessed and execution proceeds. If it is not, a page of words containing the desired word is transferred from disk to memory.

This page displaces some page in the memory that is currently inactive.

Semiconductor RAM

Semi-Conductor memories are available is a wide range of speeds. Their cycle time ranges from 100ns to 10ns.

INTERNAL ORGANIZATION OF MEMORY CHIPS:

Memory cells are usually organized in the form of array, in which each cell is capable of storing one bit of information.

Each row of cells constitute a memory word and all cells of a row are connected to a common line called as word line.

The cells in each column are connected to Sense / Write circuit by two bit lines.

The Sense / Write circuits are connected to data input or output lines of the chip. During a write operation, the sense / write circuit receive input information and store it in the cells of the selected word.

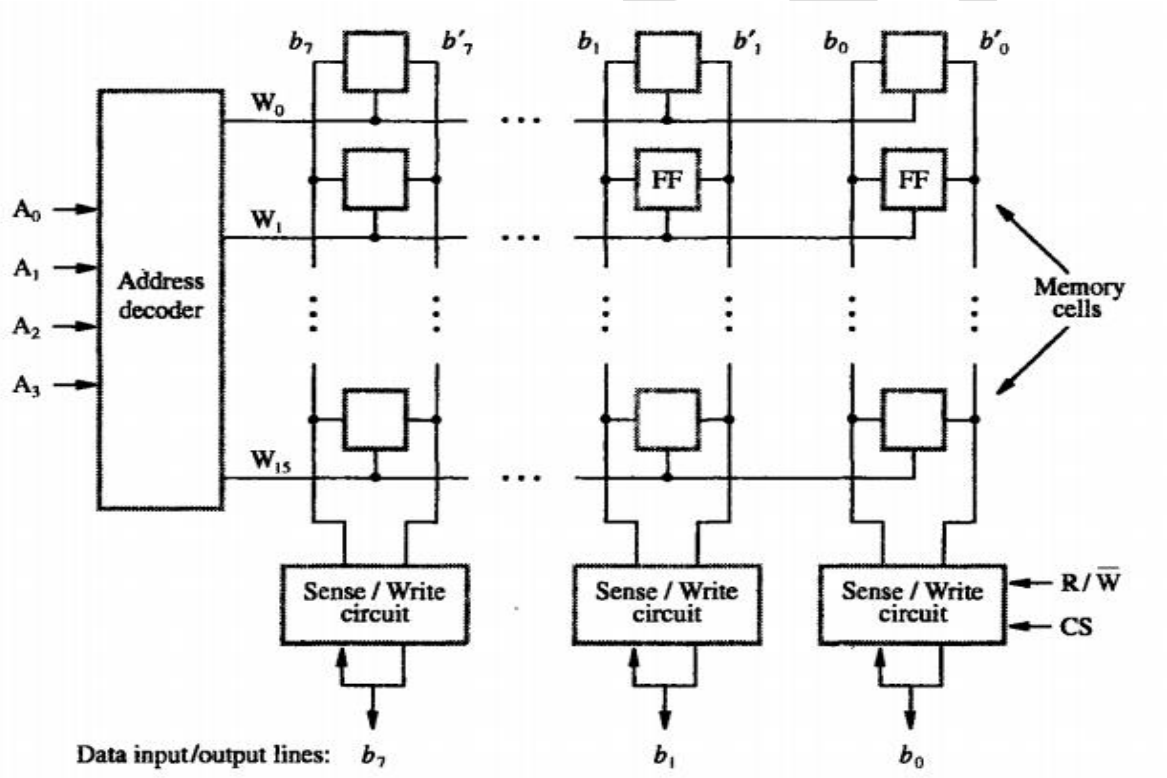


Figure 5.2 Organization of bit cells in a memory chip.

The data input and data output of each senses / write ckt are connected to a single bidirectional data line that can be connected to a data bus of the cptr.

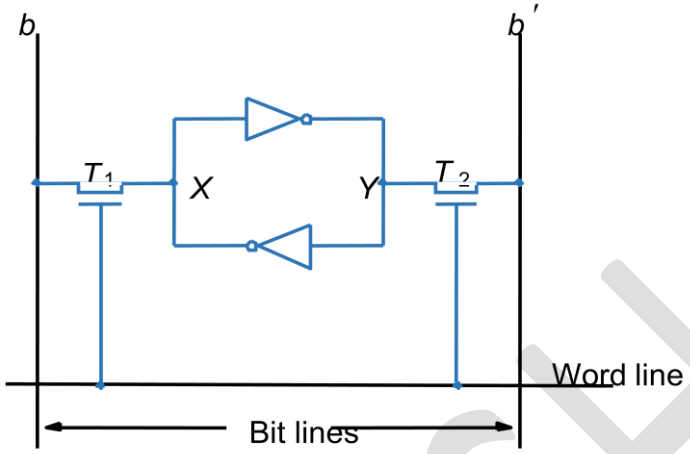
R / W → Specifies the required operation.

CS → Chip Select input selects a given chip in the multi-chip memory system

Static Memories:

Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memory.

Fig: Static RAM cell



- Two inverters are cross connected to form a batch.
- The batch is connected to two bit lines by transistors T1 and T2.
- These transistors act as switches that can be opened / closed under the control of the word line.
- When the word line is at ground level, the transistors are turned off and the latch retain its state.

Read Operation:

- In order to read the state of the SRAM cell, the word line is activated to close switches T1 and T2.
- If the cell is in state 1, the signal on bit line b is high and the signal on the bit line b is low. Thus b and b are complements of each other.
- Sense / write circuit at the end of the bit line monitors the state of b and b' and set the output accordingly.

Write Operation:

- The state of the cell is set by placing the appropriate value on bit line b and its complement on b and then activating the word line. This forces the cell into the corresponding state.
- The required signal on the bit lines are generated by Sense / Write circuit.

Fig:CMOS cell (Complementary Metal oxide Semi Conductor):

Figure 5.4 A static RAM cell.

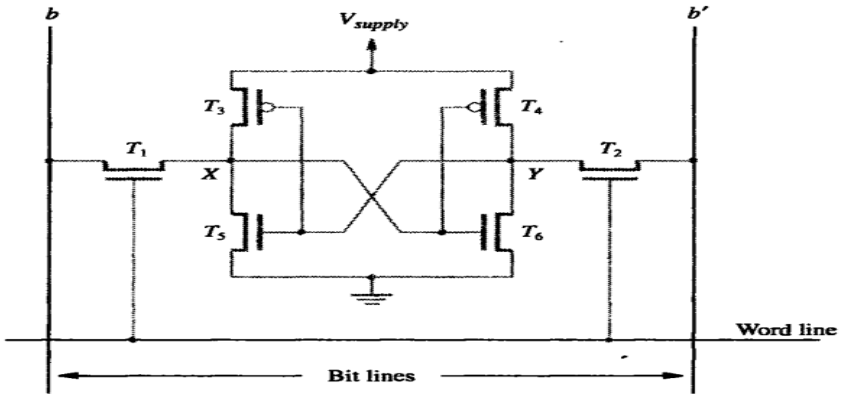


Figure 5.5 An example of a CMOS memory cell.

- Transistor pairs (T3, T5) and (T4, T6) form the inverters in the latch.
- In state 1, the voltage at point X is high by having T5, T6 on and T4, T5 are OFF.
- Thus T1 and T2 returned ON (Closed), bit line b and b will have high and low signals respectively.
- The CMOS requires 5V (in older version) or 3.3.V (in new version) of power supply voltage.
- The continuous power is needed for the cell to retain its state

Merit :

- It has low power consumption because the current flows in the cell only when the cell is being activated accessed.
- Static RAM's can be accessed quickly. It access time is few Nano seconds.

Demerit:

- SRAM's are said to be volatile memories because their contents are lost when the power is interrupted.

Asynchronous DRAMS:-

- Less ex pensive RAMs can be implemented if simplex call s are used such c ell s cannot retain their state indefinitely. Hence they are called Dynamic RAM's (DRAM).
- The information stored in a dynamic memory cell in the form of a charge on a capacitor and this charge can be maintained only for tens of Milliseconds.
- The contents must be periodically refreshed by restoring by restoring this capacitor charge to its full value.
- In order to store information in the cell, the transistor T is turned on & the appropriate voltage is applied to the bit line, which charges the capacitor.
- After the transistor is turned off, the capacitor begins to discharge which is caused by the capacitor's own leakage resistance.
- Hence the information stored in the cell can be retrieved correctly before the threshold value of the capacitor drops down.

Fig:A single transistor dynamic Memory cell

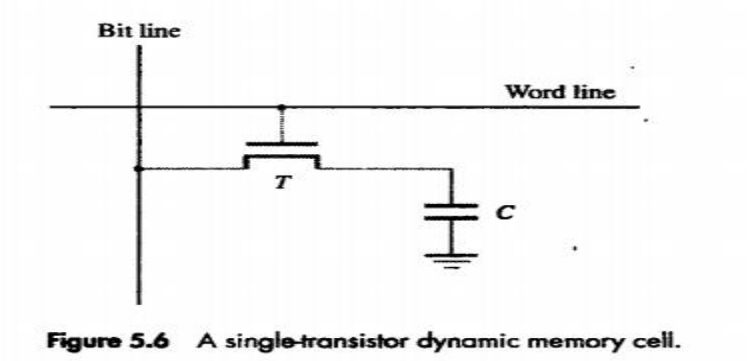


Figure 5.6 A single-transistor dynamic memory cell.

During are ad operation, the transistor is turned „o n‟ & a sense amplifier connected to the bit line detects whether the charge on the capacitor is above the threshold value.

If charge on capacitor > threshold value -> Bit line will have logic value 1.

If charge on capacitor < threshold value -> Bit line will set to logic value 0.

Fig:Internal organization of a 2M X 8 dynamic Memory chip.

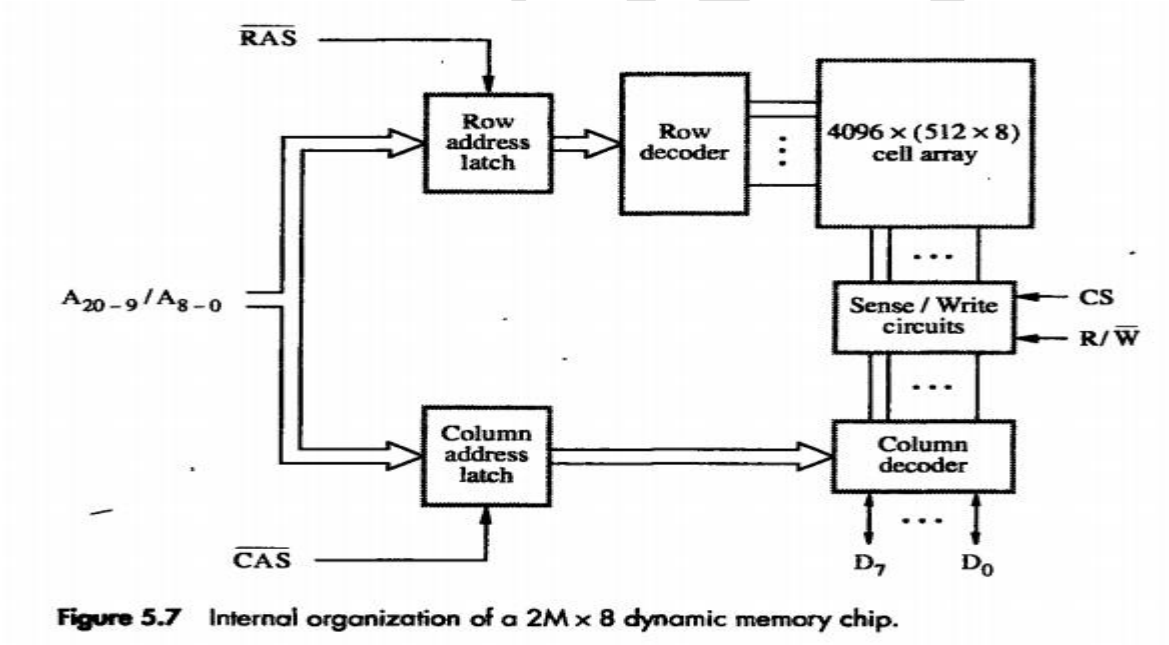


Figure 5.7 Internal organization of a 2M × 8 dynamic memory chip.

DESCRIPTION:

- The 4 bit cells in each row are divided into 512 groups of 8.
- 21 bit address is needed to access a byte in the memory(12 bit→To select a row,9 bit→Specify the group of 8 bits in the selected row).

A_{8-0} →Row address of a byte.
A_{20-9} →Column address of a byte.

- During Read/ Write operation ,the row address is applied first. It is loaded into the row address latch in response to a signal pulse on **Row Address Strobe(RAS)** input of the chip.
- When a Read operation is initiated, all cells on the selected row are read and refreshed.
- Shortly after the row address is loaded,the column address is applied to the address pins & loaded into **Column Address Strobe(CAS).**
- The information in this latch is decoded and the appropriate group of 8 Sense/Write circuits are selected.
- R/W =1(read operation)→The output values of the selected circuits are transferred to the data lines D0 - D7.
- R/W =0(write operation)→The information on D0 - D7 are transferred to the selected circuits.
- RAS and CAS are active low so that they cause the latching of address when they change from high to low. This is because they are indicated by RAS & CAS.
- To ensure that the contents of a DRAM 's are maintained, each row of cells must be accessed periodically.
- Refresh operation usually perform this function automatically.
- A specialized memory controller circuit provides the necessary control signals RAS & CAS, that govern the timing.
- The processor must take into account the delay in the response of the memory. Such memories are referred to as **Asynchronous DRAM's.**

Fast Page Mode:

Transferring the bytes in sequential order is achieved by applying the consecutive sequence of column address under the control of successive CAS signals.

This scheme allows transferring a block of data at a faster rate. The block of transfer capability is called as Fast Page Mode.

Synchronous DRAM:

- Here the operations are directly synchronized with clock signal.
- The address and data connections are buffered by means of registers.
- The output of each sense amplifier is connected to a latch.
- A Read operation causes the contents of all cells in the selected row to be loaded in these latches.

Fig: Synchronous DRAM

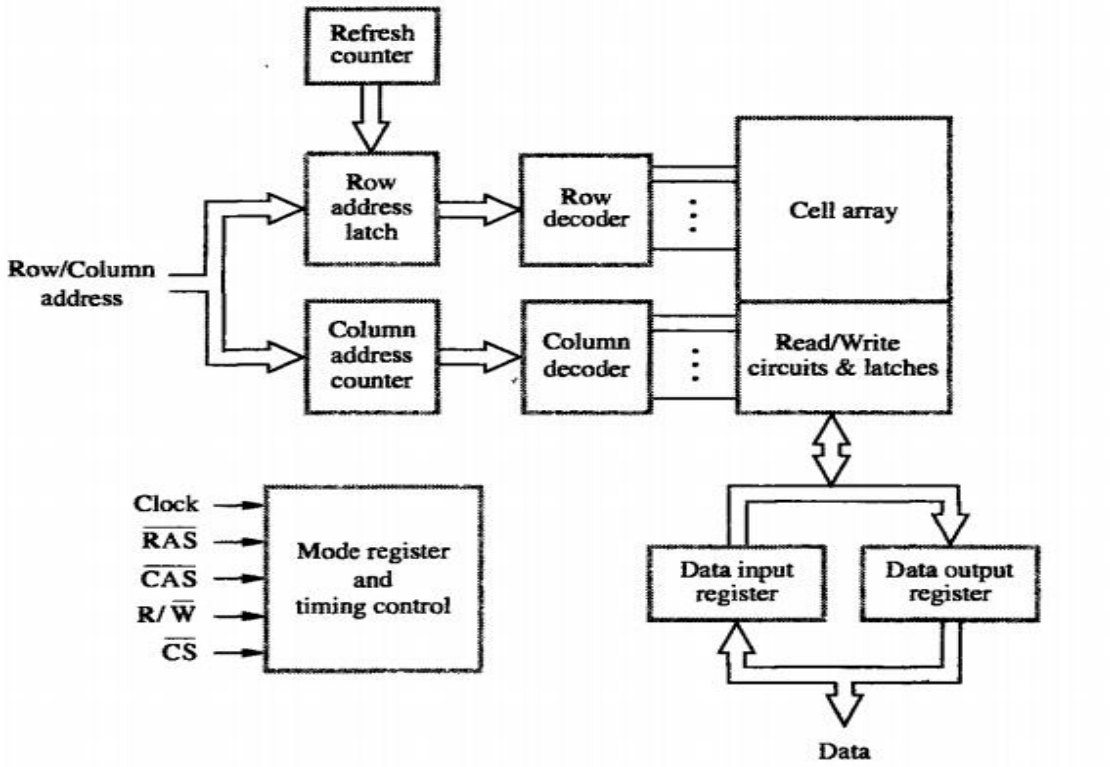
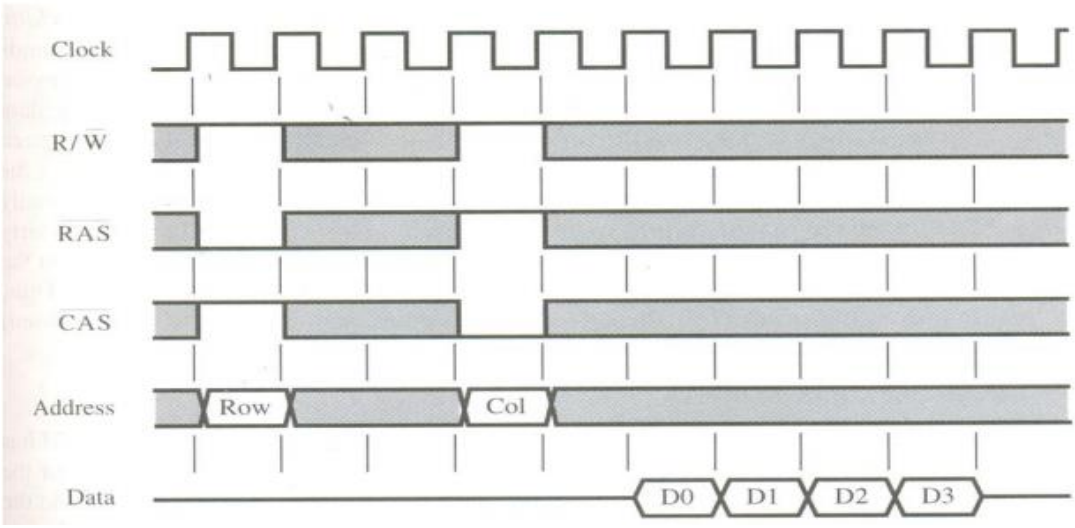


Figure 5.8 Synchronous DRAM.

- Data held in the latches that correspond to the selected columns are transferred into the data output register, thus becoming available on the data output pins.

Fig: Timing Diagram → Burst Read of Length 4 in an SDRAM

Fig: Timing Diagram → Burst Read of Length 4 in an SDRAM



- First, the row address is latched under control of RAS signal.
- The memory typically takes 2 or 3 clock cycles to activate the selected row.
- Then the column address is latched under the control of CAS signal.
- After a delay of one clock cycle, the first set of data bits is placed on the data lines.
- The SDRAM automatically increments the column address to access the next 3 sets of bits in the selected row, which are placed on the data lines in the next 3 clock cycles.

Latency & Bandwidth:

A good indication of performance is given by two parameters. They are,

- Latency
- Bandwidth

Latency:

- It refers to the amount of time it takes to transfer a word of data to or from the memory.
- For a transfer of single word, the latency provides the complete indication of memory performance.
- For a block transfer, the latency denotes the time it takes to transfer the first word of data.

Bandwidth:

- It is defined as the number of bits or bytes that can be transferred in one second.
- Bandwidth mainly depends upon the speed of access to the stored data & on the number of bits that can be accessed in parallel.

Double Data Rate SDRAM (DDR-SDRAM):

- The standard SDRAM performs all actions on the rising edge of the clock signal.
- The double data rate SDRAM transfer data on both the edges (loading edge, trailing edge).
- The Bandwidth of DDR-SDRAM is doubled for long burst transfer.
- To make it possible to access the data at high rate, the cell array is organized into two banks.
- Each bank can be accessed separately.
- Consecutive words of a given block are stored in different banks.
- Such interleaving of words allows simultaneous access to two words that are transferred on successive edge of the clock.

Larger Memories:

Dynamic Memory System:

- The physical implementation is done in the form of Memory Modules.
- If a large memory is built by placing DRAM chips directly on the main system printed circuit board that contains the processor, often referred to as Motherboard; it will occupy large amount of space on the board.
- These packaging considerations have led to the development of larger memory unit s known as SIMMs & DIMMs.
    - SIMM-Single Inline memory Module

- DIMM-Dual Inline memory Module

- SIMM & DIMM consists of several memory chips on a separate small board that plugs vertically into single socket on the motherboard.

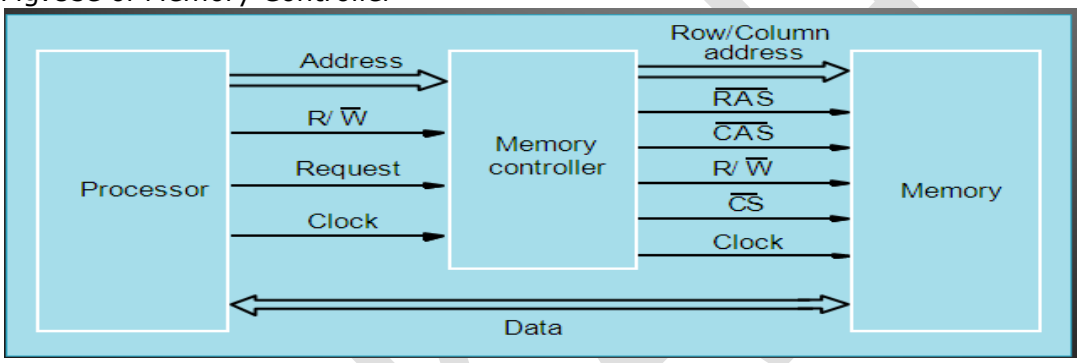
MEMORY SYSTEM CONSIDERATION:

To reduce the number of pins, the dynamic memory chips use multiplexed Address inputs. The address is divided into two parts. They are,

- High Order Address Bit (Select a row in cell array & it is provided first and latched into memory chips under the control of RAS signal).
- Low Order Address Bit(Selects a column and they are provided on same Address pins and latched using CAS signals).

The Multiplexing of address bit is usually done by Memory Controller Circuit.

Fig:Use of Memory Controller



- The Controller accepts a complete address & R/W signal from the processor, under the control of a Request signal which indicates that a memory access operation is needed.
- The Controller then forwards the row & column portions of the address to the memory and generates RAS &CAS signals.
- It also sends R/W &CS signals to the memory. The CS signal is usually active low, hence it is shown as CS.
-

Refresh Overhead:

All dynamic memories have to be refreshed. In DRAM ,the period for refreshing all rows is 16ms whereas 64ms in SDRAM.

Eg:Given a cell array of 8K(8192).

Clock cycle=4
Clock Rate=133MHZ
No of cycles to refresh all rows =8192*4 =32,768
Time needed to refresh all rows=32768/133*10=246*10-6 sec=0.246sec
Refresh Overhead=0.246/64
Refresh Overhead =0.0038

Rambus Memory:

- The usage of wide bus is expensive.
- Rambus developed the implementation of narrow bus.

- Rambus technology is a fast signaling method used to transfer information between chips.
- Instead of using signals that have voltage levels of either 0 or V_{supply} to represent the logical values, the signals consists of much smaller voltage swings around a reference voltage Vref.
- The reference Voltage is about 2V and the two logical values are represented by 0.3V swings above and below Vref
- This type of signaling is generally is known as Differential Signalling.
- Rambus provides a complete specification for the design of communication links (Special Interface circuits) called as Rambus Channel.
- Rambus memory has a clock frequency of 400MHZ. The data are transmitted on both the edges of the clock so that the effective data transfer rate is 800MHZ.
- The circuitry needed to interface to the Rambus channel is included on the chip. Such chips are known as Rambus DRAMs (RD RAM).
  Rambus channel has,
  - 9 Data lines(1-8→ Transfer the data,9^{th} line→parity checking).
  - Control line
  - Power line

A two channel rambus has 18 data lines which has no separate address lines. It is also called as Direct RD RAM's. Communication between processor or some other device that can serves as a master and RDRAM modules are serves as slaves, is carried out by means of packets transmitted on the data lines.

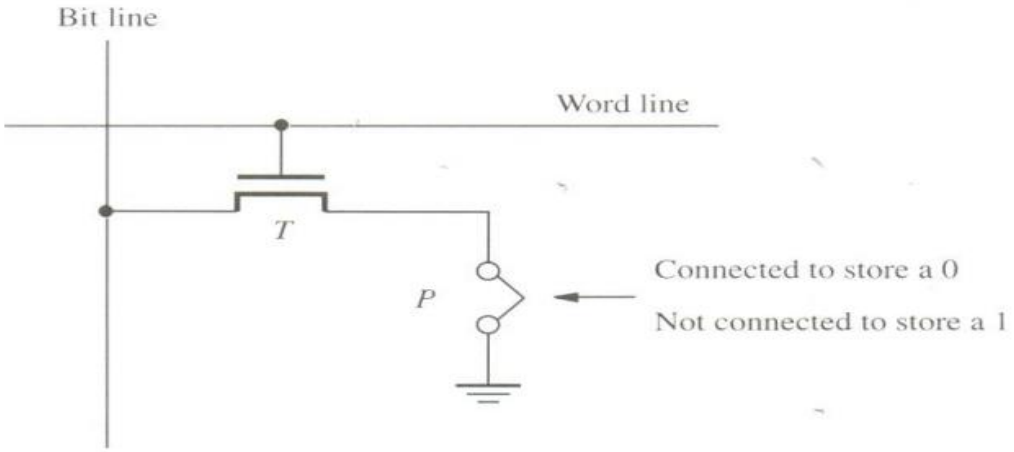
There are 3 types of packets. They are,

- Request
- Acknowledge
- Data

Types of Read-only Memory (ROM)

- Both SRAM and DRAM chips are volatile, which means that they lose the stored information if power is turned off.
- Many application requires Non-volatile memory (which retain the stored information if power is turned off).
- Eg: Operating System software has to be loaded from disk to memory which
- requires the program that boots the Operating System ie. It requires non-volatile memory.
- Non-volatile memory is used in embedded system.
- Since the normal operation involves only reading of stored data ,a memory of this type is called ROM.

Fig: ROM cell



At Logic value '0' → Transistor (T) is connected to the ground point(P).
Transistor switch is closed & voltage on bitline nearly drops to zero.
At Logic value '1' → Transistor switch is open.
The bitline remains at high voltage.
To read the state of the cell, the word line is activated.
A Sense circuit at the end of the bitline generates the proper output value.
Types of ROM:
Different types of non-volatile memory are,

- PROM
- EPROM
- EEPROM
- Flash Memory

PROM:-Programmable ROM:

- PROM allows the data to be loaded by the user.
- Programm ability is achieved b y inserting a fuse at point P in a ROM cell .
- Before it is programmed, the memory contains all 0"s
- The user can insert 1s at the required location by burning out t he fuse at these locations using high-current pulse.
- This process is irreversible.

Merit:

- It provides flexibility.
- It is faster.
- It is less expensive because they can be programmed directly by the user.

EPROM:-Erasable reprogrammable ROM:

- EPROM allows the stored data to be erased and new data to be loaded.
- In an EPROM cell , a connection to ground is always made at P and a special transistor is used, which has the ability to function either as a normal transistor or as a disabled transistor that is always turned off.
- This transistor can be programmed to behave as a permanently open switch, by injecting charge into it that becomes trapped inside.
- Erasure requires dissipating the charges trapped in the transistor of memory cells. This can be done by exposing the chip to ultra-violet light, so that EPROM chips are mounted in packages that have transparent windows.

Merits:

- It provides flexibility during the development phase of digital system.
- It is capable of retaining the stored information for a long time.

Demerits:

- The chip must be physically removed from the circuit for reprogramming and its entire contents are erased by UV light.

Electrically erasable programmable read-only memory (EEPROM)

Electrically erasable programmable read-only memory (EEPROM) chips that can be electrically programmed and erased. EEPROMs are typically changed 1 byte at time. Erasing EEPROM takes typically quite long. The drawback of EEPROM is their speed. EEPROM chips are too slow to use in many products that make quick changes to the data stored on the chip. Typically EEPROMs are found in electronics devices for storing the small amounts of nonvolatile data in applications where speed is not the most important. Small EEPROMs with serial interfaces are commonly found in many electronics devices.

Flash Memory:

- In EEPROM, it is possible to read & write the contents of a single cell.
- In Flash device, it is possible to read the contents of a single cell but it is only possible to write the entire contents of a block.
- Prior to writing, the previous contents of the block are erased. Eg. In MP3 player, the flash memory stores the data that represents sound.
- Single flash chips cannot provide sufficient storage capacity for embedded system application.
- There are 2 methods for implementing larger memory modules consisting of number of chips. They are,
    - Flash Cards
    - Flash Drives.

Merits:

- Flash drives have greater density which leads to higher capacity & low cost per bit.
- It requires single power supply voltage & consumes less power in their operation.

Flash Cards:

- One way of constructing larger module is to mount flash chips on a small card.
- Such flash card have standard interface.
- The card is simply plugged into a conveniently accessible slot.
- Its memory size are of 8,32,64MB.
- Eg:A minute of music can be stored in 1MB of memory. Hence 64MB flash cards can store an hour of music.

Flash Drives:

- Larger flash memory module can be developed by replacing the hard disk drive.
- The flash drives are designed to fully emulate the hard disk.
- The flash drives are solid state electronic devices that have no movable parts.

Merits:

- They have shorter seek and access time which results in faster response.
- They have low power consumption which makes them attractive for battery driven application.
- They are insensitive to vibration.

Demerits:

- The capacity of flash drive (<1GB) is less than hard disk(>1GB).
- It leads to higher cost per bit.
- Flash memory will deteriorate after it has been written a number of times(typically at least 1 million times.)

SPEED,SIZE COST:

Characteristics	SRAM	DRAM	Magnetis Disk
Speed	Very Fast	Slower	Much slower than DRAM
Size	Large	Small	Small
Cost	Expensive	Less Expensive	Low price

Magnetic Disk:

- A huge amount of cost effective storage can be provided by magnetic disk;The main memory can be built with DRAM which leaves SRAM's to be used in smaller units where speed is of essence.

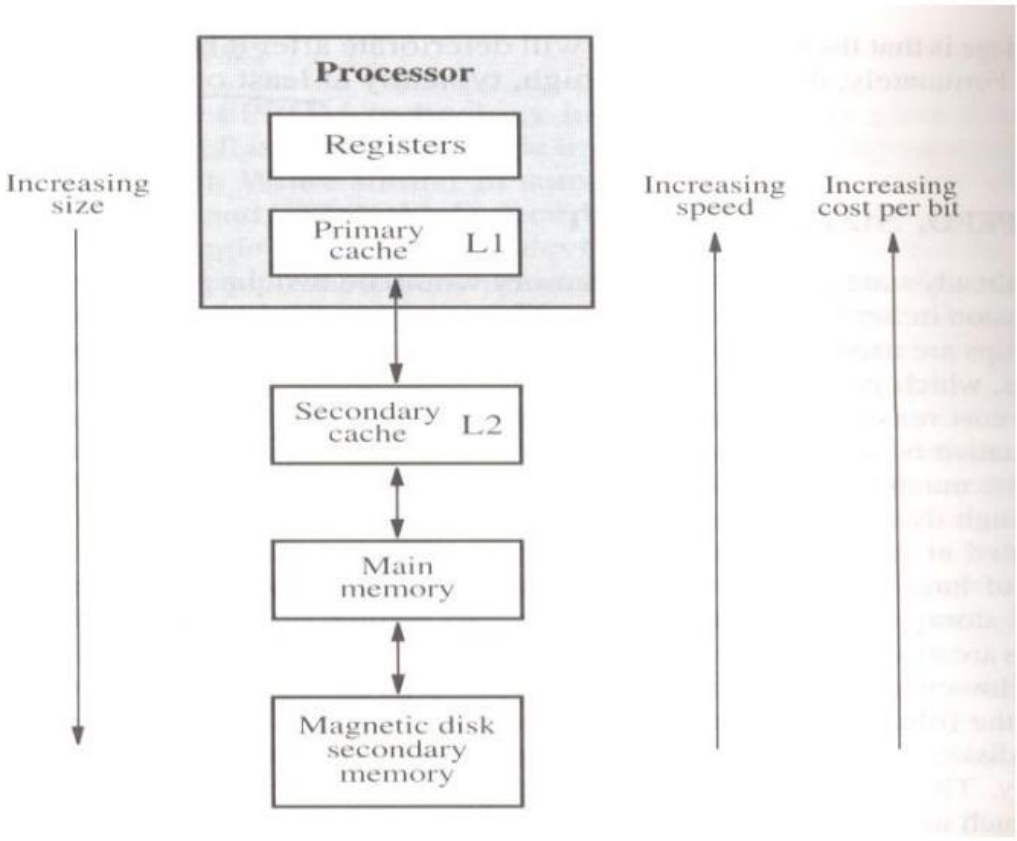
Memory	Speed	Size	Cost
Registers	Very high	Lower	Very Lower
Primary cache	High	Lower	Low
Secondary cache	Low	Low	Low
Main memory	Lower than Seconadry cache	High	High
Secondary Memory	Very low	Very High	Very High

4.4 Cache Memory

Ideally, computer memory should be fast, large and inexpensive. Unfortunately, it is impossible to meet all the three requirements simultaneously. Increased speed and size are achieved at increased cost. Very fast memory systems can be achieved if SRAM chips are used. These chips are expensive and for the cost reason it is impracticable to build a large main memory using SRAM chips. The alternative used to use DRAM chips for large main memories. The processor fetches the code and data from the main memory to execute the program. The DRAMs which form the main memory are slower devices. So it is necessary to insert wait states in memory read/write cycles. This reduces the speed of execution. The solution for this problem is in the memory system small section of SRAM is added along with the main memory, referred to as cache memory. The program which is to be executed is loaded in the main memory, but the part of the program and data accessed from the cache memory. The cache controller looks after this swapping between main memory and cache memory with the help of DMA controller, Such cache memory is called **secondary cache**. Recent processors have the built in cache memory called **primary cache.** The size of the memory is still small compared to the demands of the large programs with the voluminous

data. A solution is provided by using secondary storage, mainly magnetic disks and magnetic tapes to implement large memory spaces, which is available at reasonable prices. To make efficient computer system it is not possible to rely on a single memory component, but to employ a memory hierarchy which uses all different types of memory units that gives efficient computer system. A typical memory hierarchy is illustrated below in the figure:

Fig:Memory Hierarchy



- Fastest access is to the data held in processor registers. Registers are at the top of the memory hierarchy.
- Relatively small amount of memory that can be implemented on the processor chip. This is processor cache.
- Two levels of cache. Level 1 (L1) cache is on the processor chip. Level 2 (L2) cache is in between main memory and processor.
- Next level is main memory, implemented as SIMMs. Much larger, but much slower than cache memory.
- Next level is magnetic disks. Huge amount of inexpensive storage.
- Speed of memory access is critical, the idea is to bring instructions and data that will be used in the near future as close to the processor as possible.

The effectiveness of cache mechanism is based on the property of " Locality of reference'.

Locality of Reference:

Many instructions in the localized areas of the program are executed repeatedly during some time period and remainder of the program is accessed relatively infrequently.

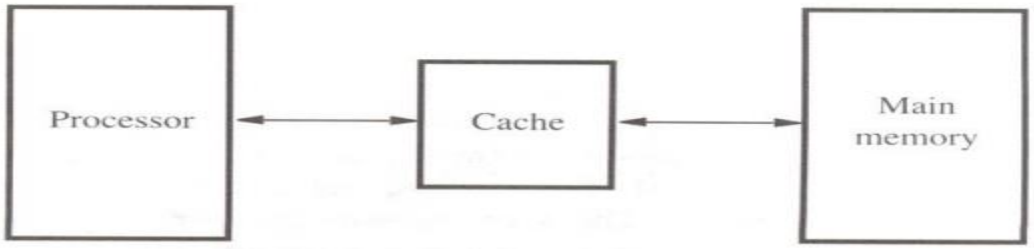
It manifests itself in 2 ways. They are,

- **Temporal**(The recently executed instruction are likely to be executed again very soon.)
- **Spatial**(The instructions in close proximity to recently executed instruction are also likely to be executed soon.) If the active segment of the program is placed in cache memory, then the total execution time can be reduced significantly.

The term Block refers to the set of contiguous address locations of some size.

The cache line is used to refer to the cache block.

Fig: Use of Cache Memory



- The Cache memory stores a reasonable number of blocks at a given time but this number is small compared to the total number of blocks available in Main Memory.
- The correspondence between main memory block and the block in cache memory is specified by a mapping function.
- The Cache control hardware decide that which block should be removed to create space for the new block that contains the referenced word.
- The collection of rule for making this decision is called the replacement algorithm.
- The cache control circuit determines whether the requested word currently exists in the cache.
- If it exists, then Read/Write operation will take place on appropriate cache location. In this case Read/Write hit will occur.
- In a Read operation, the memory will not involve.
- The write operation is proceeding in 2 ways. They are,
    - Write-through protocol
    - Write-back protocol

Write-through protocol:

Here the cache location and the main memory locations are updated simultaneously.

Write-back protocol:

- This technique is to update only the cache location and to mark it as with associated flag bit called dirty/modified bit.

- The word in the main memory will be updated later, when the block containing this marked word is to be removed from the cache to make room for a new block.
- If the requested word currently not exists in the cache during read operation, then read miss will occur.
- To overcome the read miss Load – through / Early restart protocol is used.

Read Miss:
The block of words that contains the requested word is copied from the main memory into cache.

Load – through:

- After the entire block is loaded into cache, the particular word requested is forwarded to the processor.
- If the requested word not exists in the cache during write operation, then Write Miss will occur.
- If Write through protocol is used, the information is written directly into main memory.
- If Write back protocol is used then block containing the addressed word is first brought into the cache and then the desired word in the cache is over-written with the new information.

Cache Memories – Mapping Functions

First generation processors, those designed with vacuum tubes in 1950 or those designed with integrated circuits in 1965 or those designed as microprocessors in 1980 were generally about the same speed as main memory. On such processors, this naive model was perfectly reasonable. By 1970, however, transistorized supercomputers were being built where the central processor was significantly faster than the main memory, and by 1980, the difference had increased, although it took several decades for the performance difference to reach today's extreme.

Solution to this problem is to use what is called a ***cache memory*** between the central processor and the main memory. Cache memory takes advantage of the fact that, with any of the memory technologies available for the past half century, we have had a choice between building large but slow memories or small but fast memories. This was known as far back as 1946, when Berks, Goldstone and Von Neumann proposed the use of a memory hierarchy, with a few fast registers in the central processor at the top of the hierarchy, a large main memory in the middle, and a library of archival data, stored off-line, at the very bottom.

A cache memory sits between the central processor and the main memory. During any particular memory cycle, the cache checks the memory address being issued by the processor. If this address matches the address of one of the few memory locations held in the cache, the cache handles the memory cycle very quickly; this is called a **cache hit.** If the address does not, then the memory cycle must be satisfied far more slowly by the main memory; this is called a **cache miss.**

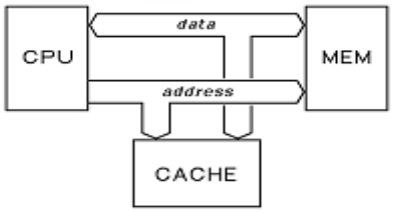


Figure 7:Adding a cache to the naive view

The correspondence between the main memory and cache is specified by a Mapping function. When the cache is full and a memory word that is not in the cache is referenced, the cache control hardware must decide which block should be removed to create space for the new block that constitutes the Replacement algorithm.

Mapping Functions

There are three main mapping techniques which decides the cache organization:

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

To discuss possible methods for specifying where memory blocks are placed in the cache, we use a specific small example, a cache consisting of 128 blocks of 16 word each, for a total of 2048(2k) word, and assuming that the main memory is addressable by a 16-bit address. The main memory has 64k word, which will be viewed as 4K blocks of 16 word each, the consecutive addresses refer to consecutive word.

Direct Mapping Technique

The cache systems are divided into three categories, to implement cache system. As shown in figure, the lower order 4-bits from 16 words in a block constitute a **word field**. The second field is known as **block field** used to distinguish a block from other blocks. Its length is 7-bits, when a new block enters the cache; the 7-bit cache block field determines the cache position in which this block must be stored. The third field is a **Tag field**, used to store higher order 5-bits of the memory address of the block, and to identify which of the 32blocks are mapped into the cache.

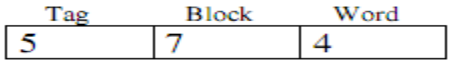


Figure 8:Main Memory Address

It is the simplest mapping technique, in which each block from the main memory has only one possible location in the cache organization. For example, the block I of the main memory maps on to block i module128 of the cache. Therefore, whenever one of the main memory blocks 0, 128, 256, ……. Is loaded in the cache, it is stored in the block 0. Block 1, 129, 257,….. are stored in block 1 of the cache and so on.

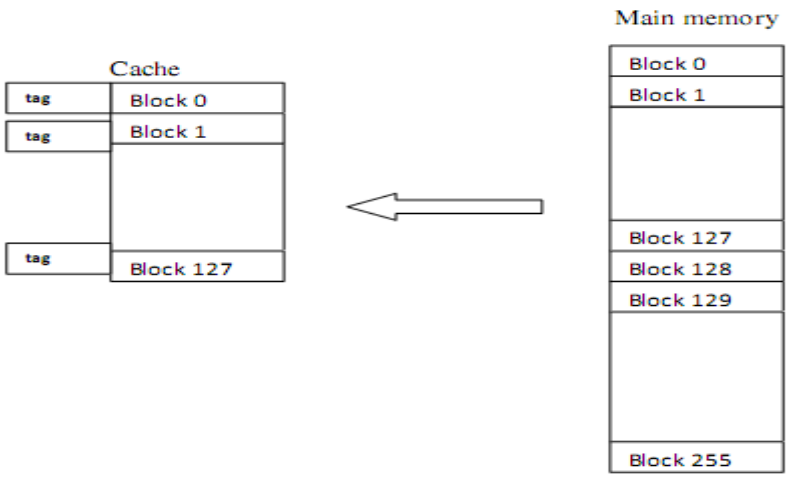


Figure 9 :Direct Mapped Cache

Associative Mapping Technique

The figure shows the associative mapping, where in which main memory block can be placed into any cache block position, in this case, 12 tag bits are required to identify a memory block when it is resident in the cache. The tag bits of an address received from the processor are compared to the tag bits of each block of the cache, to see if the desired

block is present. This is called associative-mapping technique. It gives the complete freedom in choosing the cache location in which to place the memory block.

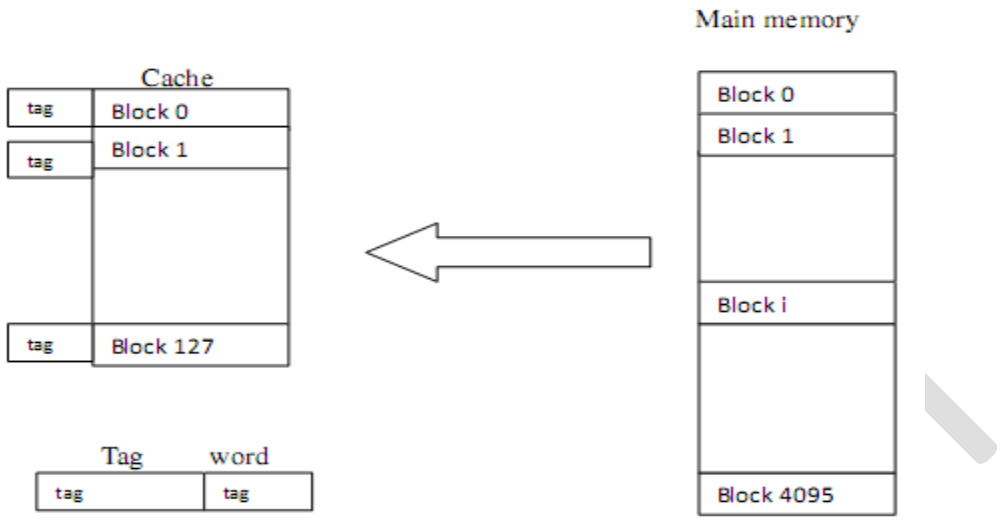


Figure 10: Associative mapped cache

Set-Associative Mapping

It is a combination of the direct and associative-mapping techniques can be used. Blocks of the cache are grouped into sets and the mapping allows a block of main memory to reside in any block of the specific set. In this case memory blocks 0, 64,128……4032 mapped into cache set 0, and they can occupy either of the two block positions within this set. The cache might contain the desired block. The tag field of the address must then be associatively compared to the tags of the two blocks of the set to check if the desired block is present this two associative search is simple to implement.

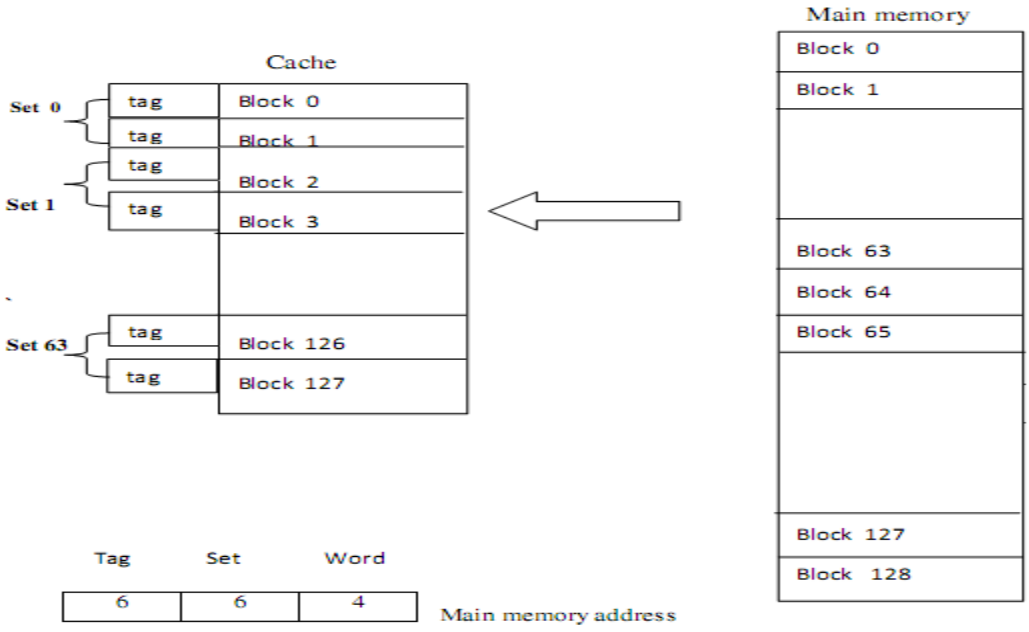


Figure 11: Set-Associative Mapped Cache

Replacement Algorithms

In a direct-mapped cache, the position of each block is fixed, hence no replacement strategy exists. In associative and set-associative caches, when a new block is to be brought into the cache and all the Positions that it may occupy are full, the cache controller must decide which of the old blocks to overwrite. This is important issue because the decision can be factor in system performance.

The objective is to keep blocks in the cache that are likely to be referenced in the near future. Its not easy to determine which blocks are about to be referenced. The property of locality of reference gives a clue to a reasonable strategy. When a block is to be over written, it is sensible to overwrite the one that has gone the longest time without being referenced. This block is called the least recently used(LRU) block, and technique is called the **LRU Replacement algorithm.**

The LRU algorithm has been used extensively for many access patterns, but it can lead to poor performance in some cases. For example, it produces disappointing results when accesses are made to sequential elements of an array that is slightly too large to fit into the cache. Performance of LRU algorithm can be improved by introducing a small amount of randomness in deciding which block to replace.

Example of mapping techniques

We now consider a detailed example to illustrate the effects of different cache mapping techniques. Assume that a processor has separate instruction and data caches. To keep the example simple, assume the data cache has space for only eight blocks of data. Also assume that each block consists of only one 16-bit word of data and the memory is word-addressable with 16-bit addresses. (These parameters are not realistic for actual computers, but they allow us to illustrate mapping techniques clearly.) Finally, assume the LRU replacement algorithm is used for block replacement in the cache.

Let us examine changes in the data cache entries caused by running the following application: A 4 × 10 array of numbers, each occupying one word, is stored in main memory locations 7A00 through 7A27 (hex). The elements of this array, A, are stored in column order, as shown in Figure 5.18. The figure also indicates how tags for different cache mapping techniques are derived from the memory address. Note that no bits are needed to identify a word within a block, as was done in Figures 5.15 through 5.17, because we have assumed that each block contains only one word. The application normalizes the elements of the first row of A with respect to the average value of the elements in the row. Hence, we need to compute the average of the elements in the row

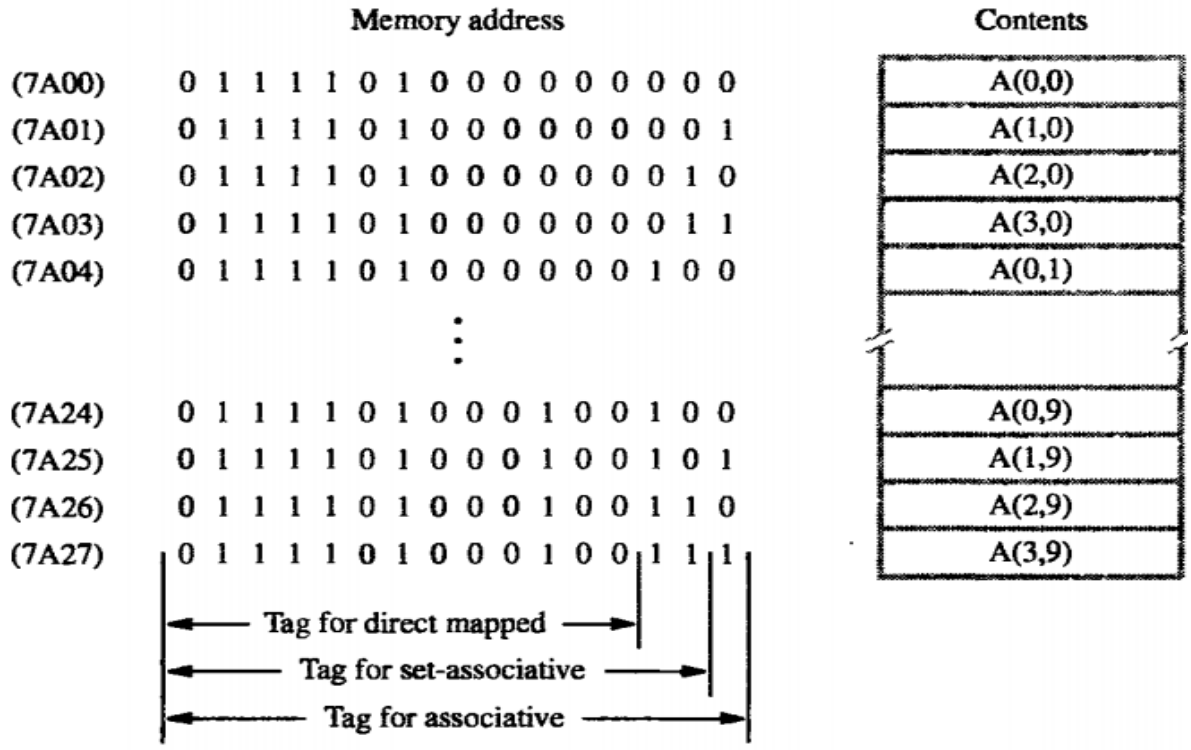


Figure 5.18 An array stored in the main memory.

```
SUM := 0
for j:= 0 to 9 do
        SUM := SUM + A(0,j)
end
AVE := SUM / 10
for i:= 9 downto 0 do
        A(0,i) := A(0,i) / AVE
end
```

Figure 5.19 Task for example in Section 5.5.3.

Block position	Contents of data cache after pass:								
	$j = 1$	$j = 3$	$j = 5$	$j = 7$	$j = 9$	$i = 6$	$i = 4$	$i = 2$	$i = 0$
0	A(0,0)	A(0,2)	A(0,4)	A(0,6)	A(0,8)	A(0,6)	A(0,4)	A(0,2)	A(0,0)
1									
2									
3									
4	A(0,1)	A(0,3)	A(0,5)	A(0,7)	A(0,9)	A(0,7)	A(0,5)	A(0,3)	A(0,1)
5									
6									
7									

Figure 5.20 Contents of a direct-mapped data cache.

and divide each element by that average. The required task can be expressed as

$$A(0, i) \leftarrow \frac{A(0, i)}{\left(\sum_{j=0}^{9} A(0, j)\right) / 10} \qquad \text{for } i = 0, 1, \ldots, 9$$

Figure 5.19 gives the structure of a program that corresponds to this task. In a machine language implementation of this program, the array elements will be addressed as memory locations. We use the variables SUM and AVE to hold the sum and average values, respectively. These variables, as well as index variables i and j, will be held in processor registers during the computation.

Direct mapped Cache:

In a direct-mapped data cache, the contents of the cache change as shown in Figure 5.20. The columns in the table indicate the cache contents after various passes through the two program loops in Figure 5.19 are completed. For example, after the second pass through the first loop ($j = 1$), the cache holds the elements $A(0, 0)$ and $A(0, 1)$. These elements are in block positions 0 and 4, as determined by the three least-significant bits of the address. During the next pass, the $A(0, 0)$ element is replaced by $A(0, 2)$, which maps into the same block position. Note that the desired elements map into only two positions in the cache, thus leaving the contents of the other six positions unchanged from whatever they were before the normalization task was executed.

After the tenth pass through the first loop ($j = 9$), the elements $A(0, 8)$ and $A(0, 9)$ are found in the cache. Since the second loop reverses the order in which the elements are handled, the first two passes through this loop ($i = 9, 8$) will find the required data in the cache. When $i = 7$, the element $A(0, 9)$ is replaced with $A(0, 7)$. When $i = 6$, element $A(0, 8)$ is replaced with $A(0, 6)$, and so on. Thus, eight elements are replaced while the second loop is executed.

Associate mapped cache:

Figure 5.21 presents the changes if the cache is associative-mapped. During the first eight passes through the first loop, the elements are brought into consecutive block positions, assuming that the cache was initially empty. During the ninth pass ($j = 8$), the LRU algorithm chooses $A(0, 0)$ to be overwritten by $A(0, 8)$. The next and last pass through the j loop sees $A(0, 1)$ replaced by $A(0, 9)$. Now, for the first eight passes through the second loop ($i = 9, 8, \ldots, 2$) all required elements are found in the cache. When $i = 1$, the element needed is $A(0, 1)$, so it replaces the least recently used element, $A(0, 9)$. During the last pass, $A(0, 0)$ replaces $A(0, 8)$.

In this case, when the second loop is executed, only two elements are not found in the cache. In the direct-mapped case, eight of the elements had to be reloaded during the second loop. Obviously, the associative-mapped cache benefits from the complete freedom in mapping a memory block into any position in the cache. Good utilization

	Contents of data cache after pass:				
Block position	$j = 7$	$j = 8$	$j = 9$	$i = 1$	$i = 0$
0	A(0,0)	A(0,8)	A(0,8)	A(0,8)	A(0,0)
1	A(0,1)	A(0,1)	A(0,9)	A(0,1)	A(0,1)
2	A(0,2)	A(0,2)	A(0,2)	A(0,2)	A(0,2)
3	A(0,3)	A(0,3)	A(0,3)	A(0,3)	A(0,3)
4	A(0,4)	A(0,4)	A(0,4)	A(0,4)	A(0,4)
5	A(0,5)	A(0,5)	A(0,5)	A(0,5)	A(0,5)
6	A(0,6)	A(0,6)	A(0,6)	A(0,6)	A(0,6)
7	A(0,7)	A(0,7)	A(0,7)	A(0,7)	A(0,7)

Figure 5.21 Contents of an associative-mapped data cache.

	Contents of data cache after pass:					
	$j = 3$	$j = 7$	$j = 9$	$i = 4$	$i = 2$	$i = 0$
Set 0	A(0,0)	A(0,4)	A(0,8)	A(0,4)	A(0,4)	A(0,0)
	A(0,1)	A(0,5)	A(0,9)	A(0,5)	A(0,5)	A(0,1)
	A(0,2)	A(0,6)	A(0,6)	A(0,6)	A(0,2)	A(0,2)
	A(0,3)	A(0,7)	A(0,7)	A(0,7)	A(0,3)	A(0,3)
Set 1						

Figure 5.22 Contents of a set-associative-mapped data cache.

of this cache also occurred because we chose to reverse the order in which the elements are handled in the second loop of the program. It is interesting to consider what would happen if the second loop dealt with the elements in the same order as in the first loop (see Problem 5.12). Using the LRU algorithm, all elements would be overwritten before they are used in the second loop. This degradation in performance would not occur if a random replacement algorithm were used.

Set Associative mapped cache:

For this example, we assume that a set-associative data cache is organized into two sets, each capable of holding four blocks. Thus, the least-significant bit of an address determines which set the corresponding memory block maps into. The high-order 15 bits constitute the tag.

Changes in the cache contents are depicted in Figure 5.22. Since all the desired blocks have even addresses, they map into set 0. Note that, in this case, six elements must be reloaded during execution of the second loop.

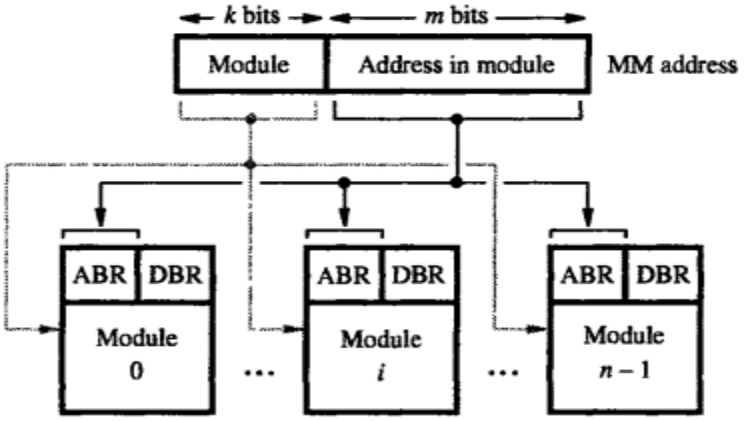
Even though this is a simplified example, it illustrates that in general, associative mapping performs best, set-associative mapping is next best, and direct mapping is the worst. However, fully associative mapping is expensive to implement, so set-associative mapping is a good practical compromise.

4.5 Performance Considerations:

- Two Key factors in the commercial success are the performance & cost ie the best possible performance at low cost.
- A common measure of success is called the Price/ Performance ratio. Performance depends on how fast the machine instruction are brought to the processor and how fast they are executed.
- To achieve parallelism(ie. Both the slow and fast units are accessed in the same manner),interleaving is used.

Interleaving:

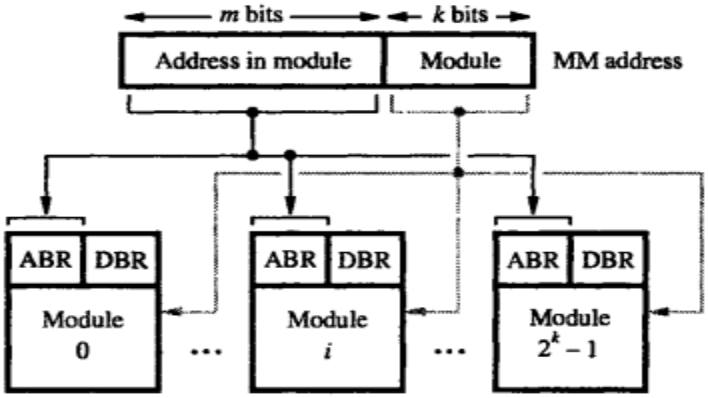
- If the main memory system is divided into a number of memory modules. Each module has its own address buffer register (ABR) and data buffer register (DBR).
- Memory access operations may proceed in more than one module at the same time. Thus the aggregate rate of transmission of words to and from the main memory system can be increased.
- Two methods of address layout are indicated they are
  - Consecutive words in a module
  - Consecutive words in a consecutive module
- **Consecutive words in a module**



(a) Consecutive words in a module

- Consecutive words are placed in a module.
- High-order k bits of a memory address determine the module.
- Low-order m bits of a memory address determine the word within a module.
- When a block of words is transferred from main memory to cache, only one module is busy at a time.

- **Consecutive words in a consecutive module**



- (b) Consecutive words in consecutive modules

- Consecutive words are located in consecutive modules.
- Consecutive addresses can be located in consecutive modules.
- While transferring a block of data, several memory modules can be kept busy at the same time.
- This is called **interleaving**
- When requests for memory access involve consecutive addresses, the access will be to different modules.
- Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.

Example:

The effect of interleaving is substantial. Consider the time needed to transfer a block of data from the main memory to the cache when a read miss occurs. Suppose that a cache with 8-word blocks is used, similar to our examples in Section 5.5. On a read miss, the block that contains the desired word must be copied from the memory into the cache. Assume that the hardware has the following properties. It takes one clock cycle to send an address to the main memory. The memory is built with relatively slow DRAM chips that allow the first word to be accessed in 8 cycles, but subsequent words of the block are accessed in 4 clock cycles per word. (Recall from Section 5.2.3 that, when consecutive locations in a DRAM are read from a given row of cells, the row address is decoded only once. Addresses of consecutive columns of the array are then applied to access the desired words, which takes only half the time per access.) Also, one clock cycle is needed to send one word to the cache.

If a single memory module is used, then the time needed to load the desired block into the cache is

$$1 + 8 + (7 \times 4) + 1 = 38 \text{ cycles}$$

Suppose now that the memory is constructed as four interleaved modules, using the scheme in Figure 5.25*b*. When the starting address of the block arrives at the memory, all four modules begin accessing the required data, using the high-order bits of the address. After 8 clock cycles, each module has one word of data in its DBR. These words are transferred to the cache, one word at a time, during the next 4 clock cycles. During this time, the next word in each module is accessed. Then it takes another 4 cycles to transfer these words to the cache. Therefore, the total time needed to load the

block from the interleaved memory is

$$1 + 8 + 4 + 4 = 17 \text{ cycles}$$

Thus, interleaving reduces the block transfer time by more than a factor of 2.

Hit Rate and Miss Penalty

An excellent indicator of the effectiveness of a particular implementation of the memory hierarchy is the success rate in accessing information at various level of the hierarchy. A successful access to data in a cache is called a hit.

The number of hits stated as fraction of all attempted access is called the hit rate, and the miss rate is the number of misses stated as a fraction of attempted accesses.

- Hit rate can be improved by increasing block size, while keeping cache size constant.
- Block sizes that are neither very small nor very large give best results.
- Miss penalty can be reduced if load-through approach is used when loading new blocks into cache.

Example:

Consider now the impact of the cache on the overall performance of the computer. Let h be the hit rate, M the miss penalty, that is, the time to access information in the main memory, and C the time to access information in the cache. The average access time experienced by the processor is

$$t_{ave} = hC + (1 - h)M$$

We use the same parameters as in Example 5.1. If the computer has no cache, then, using a fast processor and a typical DRAM main memory, it takes 10 clock cycles for each memory read access. Suppose the computer has a cache that holds 8-word blocks and an interleaved main memory. Then, as we showed in Section 5.6.1, 17 cycles are needed to load a block into the cache. Assume that 30 percent of the instructions in a typical program perform a read or a write operation, which means that there are 130 memory accesses for every 100 instructions executed. Assume that the hit rates in the cache are 0.95 for instructions and 0.9 for data. Let us further assume that the miss penalty is the same for both read and write accesses. Then, a rough estimate of the improvement in performance that results from using the cache can be obtained as follows:

$$\frac{\textit{Time without cache}}{\textit{Time with cache}} = \frac{130 \times 10}{100(0.95 \times 1 + 0.05 \times 17) + 30(0.9 \times 1 + 0.1 \times 17)} = 5.04$$

This result suggests that the computer with the cache performs five times better.

It is also interesting to consider how effective this cache is compared to an ideal cache that has a hit rate of 100 percent (in which case, all memory references take one cycle). Our rough estimate of relative performance for these caches is

$$\frac{100(0.95 \times 1 + 0.05 \times 17) + 30(0.9 \times 1 + 0.1 \times 17)}{130} = 1.98$$

This means that the actual cache provides an environment in which the processor effectively works with a large DRAM-based main memory that appears to be only two times slower than the circuits in the cache.

In this example, we made a simplifying assumption that the same clock is used to access the on-chip cache and the main memory via the system bus. A high-performance processor is likely to operate under the control of a clock that is much faster than the system bus clock, perhaps up to ten times faster. Let us consider the impact of a cache in a system of this type.

Example 2:

Suppose that there is a single cache that is implemented on the processor chip and that the main memory is realized using SDRAM chips. Assume that the system bus clock is four times slower than the processor clock. As in Example 5.2, assume that a cache block contains 8 words, and that the hit rates in the cache are 0.95 for instructions and 0.9 for data. The SDRAM timing diagram is similar to Figure 5.9. The only difference is that there is a burst of 8 data words rather than four. Thus, according to Figure 5.9, it will take 14 clock cycles from when the RAS signal is asserted to transfer a block of data between the main memory and the cache. Since the RAS and CAS signals are generated by the memory controller, as indicated in Figure 5.11, one more cycle is needed during which the processor sends the address of the first word in a block to the memory controller. Therefore, a total of 15 cycles is needed to transfer a block. The cycles shown in Figure 5.9 are the system bus clock cycles. If the processor clock is four times faster, then it takes 60 processor cycles to transfer an 8-word block to or from the main memory. Note also that Figure 5.9 indicates that the processor can read or write a single word in the main memory in 9 bus clock cycles, consisting of the 8 cycles indicated in Figure 5.9 plus one cycle needed to send an address to the memory controller. Hence, 36 processor cycles are needed to access a single word in the main memory. Yet, the processor accesses a word in the cache in one processor cycle!

Repeating the calculation in Example 5.2 gives:

$$\frac{\textit{Time without cache}}{\textit{Time with cache}} = \frac{130 \times 36}{100(0.95 \times 1 + 0.05 \times 60) + 30(0.9 \times 1 + 0.1 \times 60)} = 7.77$$

Thus, accounting for the differences between processor and system bus clock speeds shows that the cache has an even greater positive effect on the performance.

Caches on processor chip:

In high-performance processors two levels of caches are normally used. The L1 cache(s) is on the processor chip. The L2 cache, which is much larger, may be implemented externally using SRAM chips. But, a somewhat smaller L2 cache may also be implemented on the processor chip,

If both L1 and L2 caches are used, the L1 cache should be designed to allow very fast access by the processor because its access time will have a large effect on the clock rate of the processor. A cache cannot be accessed at the same speed as a register file because the cache is much bigger and, hence, more complex. A practical way to speed up access to the cache is to access more than one word simultaneously and then let the processor use them one at a time. This technique is used in many commercial processors.

The L2 cache can be slower, but it should be much larger to ensure a high hit rate. Its speed is less critical because it only affects the miss penalty of the L1 cache. A workstation computer may include an L1 cache with the capacity of tens of kilobytes and an L2 cache of several megabytes.

Including an L2 cache further reduces the impact of the main memory speed on the performance of a computer. The average access time experienced by the processor in a system with two levels of caches is

$$t_{ave} = h_1C_1 + (1 - h_1)h_2C_2 + (1 - h_1)(1 - h_2)M$$

where

h_1 is the hit rate in the L1 cache.

h_2 is the hit rate in the L2 cache.

C_1 is the time to access information in the L1 cache.

C_2 is the time to access information in the L2 cache.

M is the time to access information in the main memory.

The number of misses in the L2 cache, given by the term $(1 - h_1)(1 - h_2)$, should be low. If both h_1 and h_2 are in the 90 percent range, then the number of misses will be less than 1 percent of the processor's memory accesses. Thus, the miss penalty M will be less critical from a performance point of view.

Other enhancements:

Write buffer

- Write-through:
- Each write operation involves writing to the main memory.
- If the processor has to wait for the write operation to be complete, it slows down the processor.
- Processor does not depend on the results of the write operation.
- Write buffer can be included for temporary storage of write requests.
- Processor places each write request into the buffer and continues execution.
- If a subsequent Read request references data which is still in the write buffer, then this data is referenced in the write buffer.
- Write-back:
- Block is written back to the main memory when it is replaced.
- If the processor waits for this write to complete, before reading the new block, it is slowed down.
- Fast write buffer can hold the block to be written, and the new block can be read first.

Prefetching

- New data are brought into the processor when they are first needed.
- Processor has to wait before the data transfer is complete.
- Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.
- Prefetching can be accomplished through software by including a special instruction in the machine language of the processor.
    - Inclusion of prefetch instructions increases the length of the programs.
- Prefetching can also be accomplished using hardware:
    - Circuitry that attempts to discover patterns in memory references and then prefetches according to this pattern.

Lockup-Free Cache

- Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.
- A cache of this type is said to be "locked" while it services a miss.
- Cache structure which supports multiple outstanding misses is called a lockup free cache.
- Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.
- Special registers may hold the necessary information about these misses.

4.6 VIRTUAL MEMORY:

- Techniques that automatically move program and data blocks into the physical main memory when they are required for execution is called the **Virtual Memory**.
- The binary address that the processor issues either for instruction or data are called the virtual / Logical address.
- The virtual address is translated into physical address by a combination of hardware and software components. This kind of address translation is done by MMU (Memory Management Unit).

- When the desired data are in the main memory, these data are fetched /accessed immediately.
- If the data are not in the main memory, the MMU causes the Operating system to bring the data into memory from the disk.
- Transfer of data between disk and main memory is performed using DMA scheme.

Fig: Virtual Memory Organization

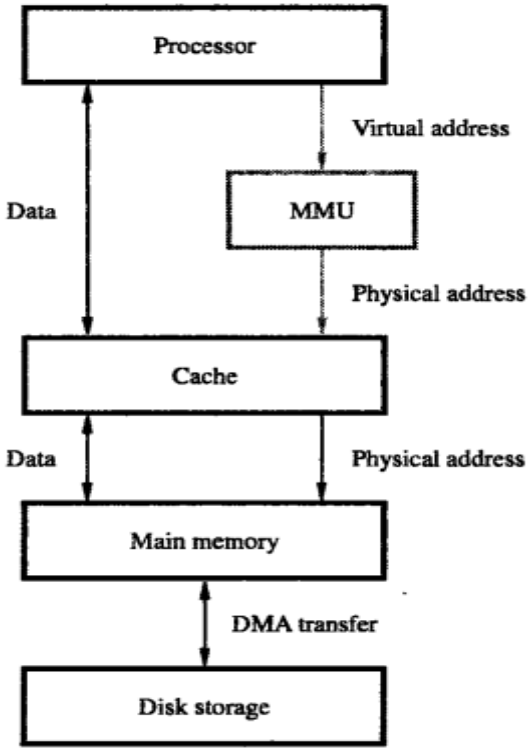


Figure 5.26 Virtual memory organization.

- Memory management unit (MMU) translates virtual addresses into physical addresses.
- If the desired data or instructions are in the main memory they are fetched as described previously.
- If the desired data or instructions are not in the main memory, they must be transferred from secondary storage to the main memory.
- MMU causes the operating system to bring the data from the secondary storage into the main memory.

Address Translation:

In address translation, all programs and data are composed of fixed length units called Pages.

The Page consists of a block of words that occupy contiguous locations in the main memory.

The pages are commonly range from 2K to 16K bytes in length. The cache bridge speed up the gap between main memory and secondary storage and it is implemented in software techniques.

Each virtual address generated by the processor contains virtual Page number (Low order bit) and offset(High order bit) Virtual Page number+ Offset→ Specifies the location of a particular byte (or word) within a page.

Page Table:

It contains the information about the main memory address where the page is stored & the current status of the page.

Page Frame:

An area in the main memory that holds one page is called the page frame.

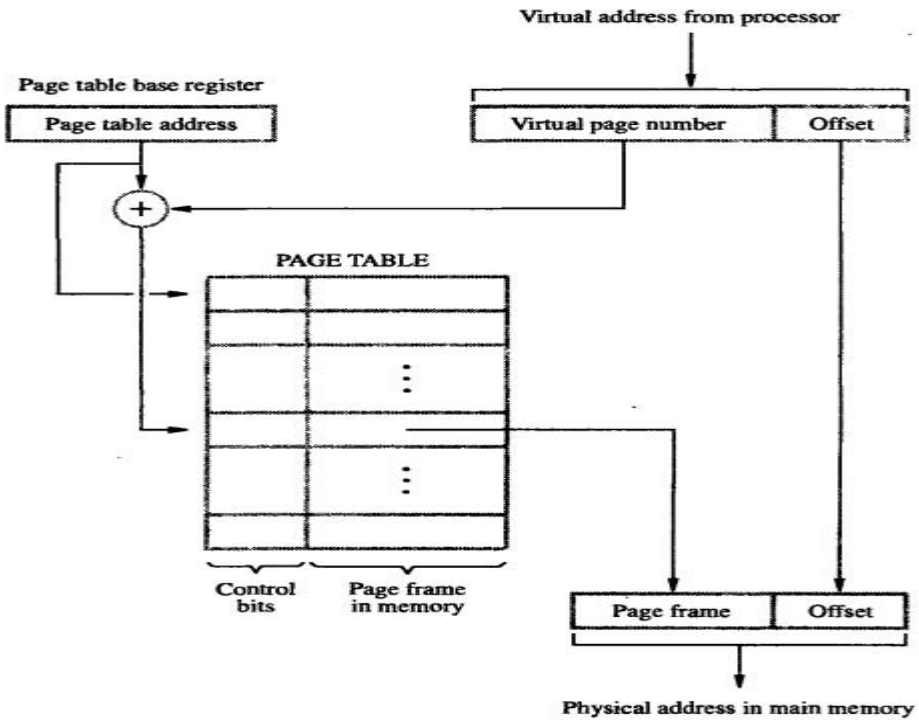
Page Table Base Register:

- It contains the starting address of the page table.
- Virtual Page Number+Page Table Base register→ Gives the address of the corresponding entry in the page table.ie)it gives the starting address of the page if that page currently resides in memory.

Control Bits in Page Table:

- The Control bit specifies the status of the page while it is in main memory. Function:
- The control bit indicates the validity of the page ie) it checks whether the page is actually loaded in the main memory.
- It also indicates that whether the page has been modified during its residency in the memory; this information is needed to determine whether the page should be written back to the disk before it is removed from the main memory to make room for another page.

Fig: Virtual Memory Address Translation



- The Page table information is used by MMU for every read & write access.
- The Page table is placed in the main memory but a copy of the small portion of the page table is located within MMU.
- This small portion or small cache is called Translation Look Aside Buffer (TLB).
- This portion consists of the page table entries that corresponds to the most recently accessed pages and also contains the virtual address of the entry.

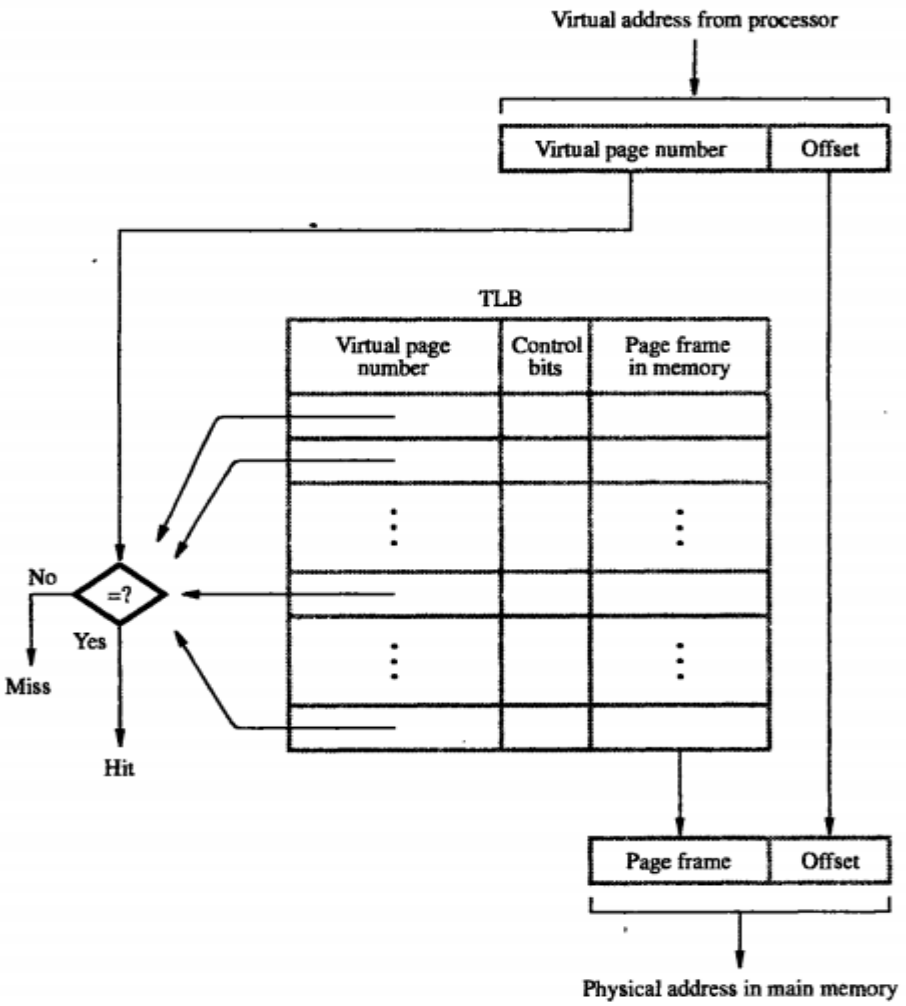


Figure 5.28 Use of an associative-mapped TLB.

- When the operating system changes the contents of page table , the control bit in TLB will invalidate the corresponding entry in the TLB. Given a virtual address, the MMU looks in TLB for the referenced page.
- If the page table entry for this page is found in TLB, the physical address is obtained immediately. If there is a miss in TLB, then the required entry is obtained from the page table in the main memory & TLB is updated.
- When a program generates an access request to a page that is not in the main memory, then Page Fault will occur.

- The whole page must be brought from disk into memory before an access can proceed. When it detects a page fault, the MMU asks the operating system to generate an interrupt.
- The operating System suspend the execution of the task that caused the page fault and begin execution of another task whose pages are in main memory because the long delay occurs while page transfer takes place.
- When the task resumes,either the interrupted instruction must continue from the point of interruption or the instruction must be restarted.
- If a new page is brought from the disk when the main memory is full,it must replace one of the resident pages.In that case,it uses LRU algorithm which removes the least referenced Page.
- A modified page has to be written back to the disk before it is removed from the main memory. In that case,write – through protocol is used.

MEMORY MANAGEMENT REQUIREMENTS:

Management routines are part of the Operating system. Assembling the OS routine into virtual address sp ace is called "System Space". The virtual space in which the user application programs reside is called the "User Space". Each user space has a separate page table. The MMU uses the page table to determine the address of the table to be used in the translation process. Hence by changing the contents of this register, the OS can switch from one space to another. The process has two stages. They are,

- User State
- Supervisor state.

User State: In this state, the processor executes the user program.

Supervisor State: When the processor executes the operating system routines, the processor will be in supervisor state. Privileged Instruction:

In user state, the machine instructions cannot be executed. Hence a user program is prevented from accessing the page table of other user spaces or system spaces.

The control bits in each entry can be set to control the access privileges granted to each program. ie) One program may be allowed to read/write a given page, while the other programs may be given only red access.

4.7 SECONDARY STORAGE:

The Semi-conductor memories do not provide all the storage capability.

The Secondary storage devices provide larger storage requirements. Some of the Secondary Storage devices are,

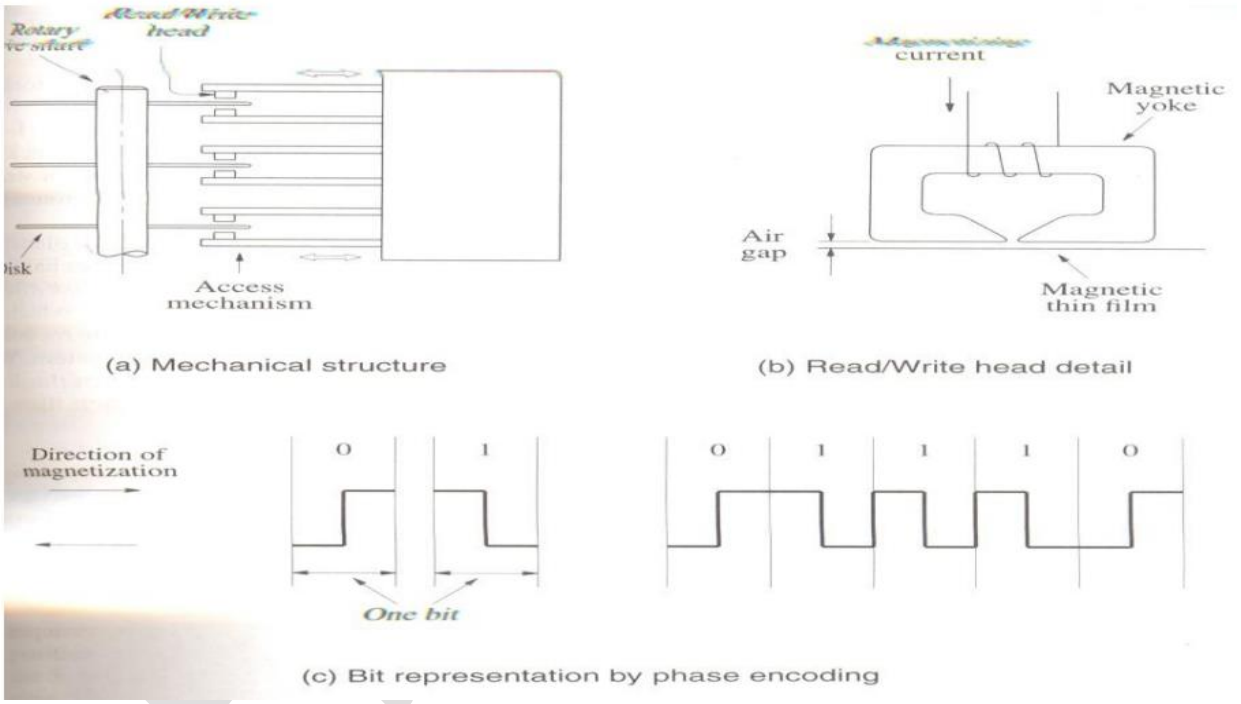
- Magnetic Disk
- Optical Disk
- Magnetic Tapes.

Magnetic Disk:

- Magnetic Disk system consists o one or more disk mounted on a common spindle.
- A thin magnetic film is deposited on each disk, usually on both sides.
- The disks are placed in a rotary drive so that the magnetized surfaces move in close proximity to read /write heads.
- Each head consists of magnetic yoke & magnetizing coil.

- Digital information can be stored on the magnetic film by applying the current pulse of suitable polarity to the magnetizing coil.
- Only changes in the magnetic field under the head can be sensed during the Read operation.
- Therefore if the binary states 0 & 1 are represented by two opposite states of magnetization, a voltage is induced in the head only at 0-1 and at 1-0 transition in the bit stream.
- A consecutive (long string) of 0‟s & 1‟s are determined by using the clock which is mainly used for synchronization.
- Phase Encoding or Manchester Encoding is the technique to combine the clocking information with data.
- The Manchester Encoding describes that how the self-clocking scheme is implemented.

Fig:Mechanical Structure



- The Read/Write heads must be maintained at a very small distance from the moving disk surfaces in order to achieve high bit densities.
- When the disks are moving at their steady state, the air pressure develops between the disk surfaces & the head & it forces the head away from the surface.
- The flexible spring connection between head and its arm mounting permits the head to fly at the desired distance away from the surface.

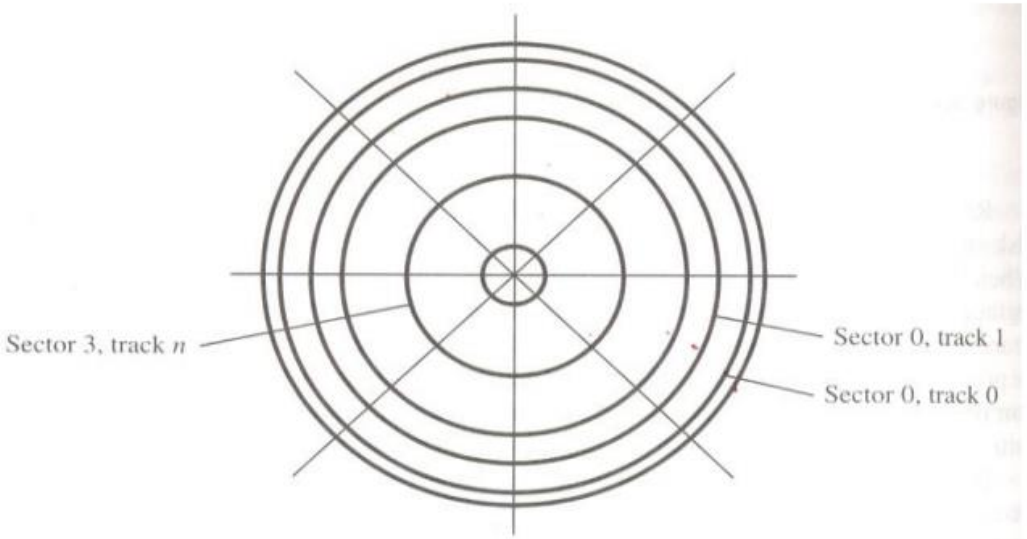
Wanchester Technology:

- Read/Write heads are placed in a sealed, air – filtered enclosure called the Wanchester Technology.
- In such units, the read/write heads can operate closure to magnetic track surfaces because the dust particles which are a problem in unsealed assemblies are absent.

Merits:

- It have a larger capacity for a given physical size. The data intensity is high because the storage medium is not exposed to contaminating elements.
- The read/write heads of a disk system are movable. The disk system has 3 parts.They are,
    - Disk Platter(Usually called Disk)
    - Disk Drive(spins the disk & moves Read/write heads)
    - Disk Controller(controls the operation of the system.)

Fig:Organizing & Accessing the data on disk



Each surface is divided into concentric **tracks.**

Each track is divided into sectors. The set of corresponding tracks on all surfaces of a stack of disk form a logical **cylinder**.

The data are accessed by specifying **the surface number, track number and the sector number.**

The Read/Write operation start at sector boundaries. Data bits are stored serially on each track.

Each sector usually contains 512 bytes.

Sector header -> contains identification information.

It helps to find the desired sector on the selected track.

ECC (Error checking code)- used to detect and correct errors.

An unformatted disk has no information on its tracks.

The formatting process divides the disk physically into tracks and sectors and this process may discover some defective sectors on all tracks.

The disk controller keeps a record of such defects.

The disk is divided into logical partitions. They are,

- Primary partition
- Secondary partition

In the diag, Each track has same number of sectors.

So all tracks have same storage capacity.

Thus the stored information is packed more densely on inner track than on outer track.

Access time

There are 2 components involved in the time delay between receiving an address and the beginning of the actual data transfer. They are,

- Seek time
- Rotational delay / Latency

Seek time – Time required to move the read/write head to the proper track.
Latency – The amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector passes under the read/write head.

Seek time + Latency = Disk access time

Typical disk

One inch disk- weight=1 ounce,
size -> comparable to match book
Capacity -> 1GB
Inch disk has the following parameter
Recording surface=20
Tracks=15000 tracks/surface
Sectors=400.
Each sector stores 512 bytes of data
Capacity of formatted disk=$20x15000x400x512=60x10^9$ =60GB
Seek time=3ms
Platter rotation=10000 rev/min
Latency=3ms
Internet transfer rate=34MB/s

Data Buffer / cache

- A disk drive that incorporates the required SCSI circuit is referred as SCSI drive. The SCSI can transfer data at higher rate than the disk tracks.
- An efficient method to deal with the possible difference in transfer rate between disk and SCSI bus is accomplished by including a data buffer.
- This buffer is a semiconductor memory.
- The data buffer can also provide cache mechanism for the disk (ie) when a read request arrives at the disk, then controller first check if the data is available in the cache (buffer).
- If the data is available in the cache, it can be accessed and placed on SCSI bus.
- If it is not available then the data will be retrieved from the disk.

Disk Controller:

- The disk controller acts as interface between disk drive and system bus.
- The disk controller uses DMA scheme to transfer data between disk and main memory.
- When the OS initiates the transfer by issuing Read/Write request, the controllers register will load the following information. They are,

- Main memory address(address of first main memory location of the block of words involved in the transfer)
- Disk address(The location of the sector containing the beginning of the desired block of words) (number of words in the block to be transferred).

Diskette or Floppy Disk

- spinning platter of special material
- Information stored by magnetically
- read/write head positioned by mechanical arm
- Storage capacity is at a few MBs
- Random access
- seek time from 10 to 40 milliseconds
- Easily portable

Optical Disks

- CD-ROM - read only (books, software releases)
- WORM - write once, read many (archival storage)
- Laser encoding, not magnetic
- 30-50 ms seek times
- 640MB - 17GB storage capacity
- Cheaper than hard disks per MB of storage capacity, but slower
- portable
- Jukeboxes of optical disks are becoming popular for storing really, really large collections of data. The Mercury-20 jukebox (no I'm not selling these, just using it as a typical example) provides access to up to 150 CD-ROMs, or in other words 94GBs of storage capacity. The Mercury jukebox takes a maximum of four seconds to exchange and load a disc into a drive, 2.5 seconds to spin up and access the data and 10 seconds to transfer a 6.0 MB file to the computer or server.