control store. In Figure19, only 20 bits are needed to store the patterns for the 42 signals.

So far we have considered grouping and encoding only mutually exclusive control signals. We can extend this idea by enumerating the patterns of required signals in all possible microinstructions. Each meaningful combination of active control signals can then be assigned a distinct code that represents the microinstruction

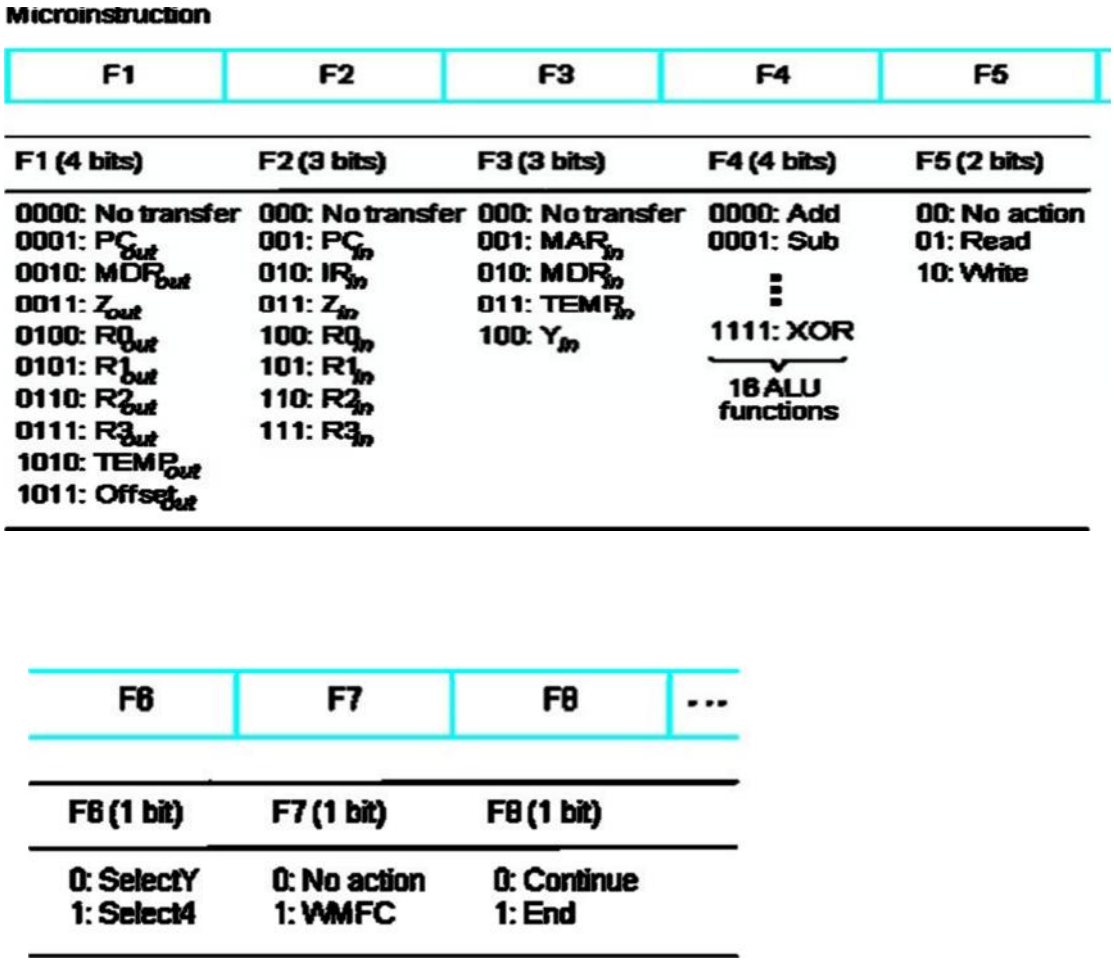


Fig 19

Such full encoding is likely to further reduce the length of micro words but also to increase the complexity of the required decoder circuits.

Highly encoded schemes that use compact codes to specify only a small number of control functions in each microinstruction are referred to as a *vertical organization*. On the other hand, the minimally encoded scheme of Figure 15, in which many resources can be controlled with a single microinstruction, is called a *horizontal organization*. The horizontal approach is useful when a higher operating speed is desired and when the machine structure allows parallel use of resources. The vertical approach results in considerably slower operating speeds because more microinstructions are needed to perform the desired control functions. Although fewer bits are required for each microinstruction, this does not imply that the total number of bits in the control store is smaller. The significant factor is that less hardware is needed to handle the execution of microinstructions. Horizontal and vertical organizations represent the two organizational extremes in micro programmed control. Many intermediate schemes are also possible, in which the degree of encoding is a design parameter. The layout in Figure 19 is a horizontal organization because it groups only mutually exclusive micro operations in thesame fields. As a result, it does not limit in any way the processor's ability to perform various micro operations in parallel.

Although we have considered only a subset of all the possible control signals, this subset is representative of actual requirements. We have omitted some details that are not essential for understanding the principles of operation.

UNIT 3
MEMORY SYSTEM

3.1 BASIC CONCEPTS

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

For example, a 16-bit computer that generates 16-bit addresses is capable of addressing upto 2^{16} =64K memory locations. If a machine generates 32-bit addresses, it can access upto 2^{32} = 4G memory locations. This number represents the size of address space of the computer.

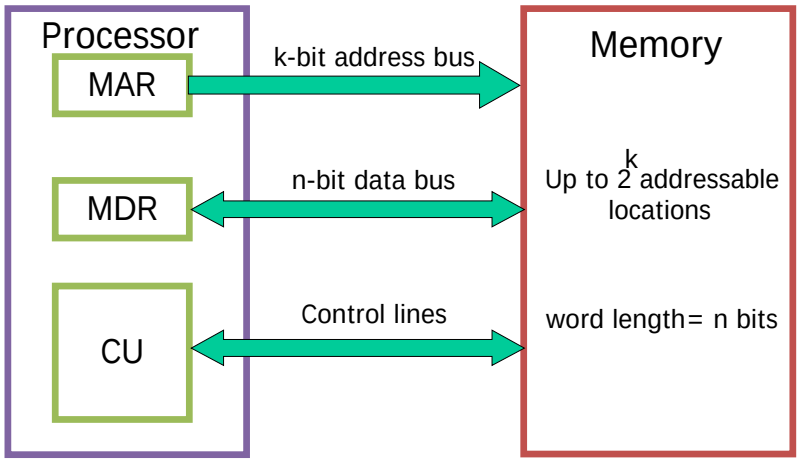
Address	Memory Locations
16 Bit	2^{16} = 64 K
32 Bit	2^{32} = 4G (Giga)
40 Bit	2^{64} = IT (Tera)

If the smallest addressable unit of information is a memory word, the machine is called word-addressable. If individual memory bytes are assigned distinct addresses, the computer is called byte-addressable. Most of the commercial machines are byte-addressable. For example in a byte-addressable 32-bit computer, each memory word contains 4 bytes. A possible word-address assignment would be:

Word Address	Byte Address			
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11
.		…..		
.		…..		
.		…..		

With the above structure a READ or WRITE may involve an entire memory word or it may involve only a byte. In the case of byte read, other bytes can also be read but ignored by the CPU. However, during a write cycle, the control circuitry of the MM must ensure that only the specified byte is altered. In this case, the higher-order 30 bits can specify the word and the lower-order 2 bits can specify the byte within the word.

CPU-Main Memory Connection – A block schematic: -



From the system standpoint, the Main Memory (MM) unit can be viewed as a "block box". Data transfer between CPU and MM takes place through the use of two CPU registers, usually called MAR (Memory Address Register) and MDR (Memory Data Register). If MAR is K bits long and MDR is 'n' bits long, then the MM unit may contain upto 2^k addressable locations and each location will be 'n' bits wide, while the word length is equal to 'n' bits. During a "memory cycle", n bits of data may be transferred between the MM and CPU. This transfer takes place over the processor bus, which has k address lines (address bus), n data lines (data bus) and control lines like Read, Write, Memory Function completed (MFC), Bytes specifiers etc (control bus). For a read operation, the CPU loads the address into MAR, set READ to 1 and sets other control signals if required. The data from the MM is loaded into MDR and MFC is set to 1. For a write operation, MAR, MDR are suitably loaded by the CPU, write is set to 1 and other control signals are set suitably. The MM control circuitry loads the data into appropriate locations and sets MFC to 1. This organization is shown in the following block schematic

Some Basic Concepts

Memory Access Times: -

It is a useful measure of the speed of the memory unit. It is the time that elapses between the initiation of an operation and the completion of that operation (for example, the time between READ and MFC).

Memory Cycle Time :-

It is an important measure of the memory system. It is the minimum time delay required between the initiations of two successive memory operations (for example, the time between two successive READ operations). The cycle time is usually slightly longer than the access time.

RANDOM ACCESS MEMORY (RAM):

A memory unit is called a Random Access Memory if any location can be accessed for a READ or WRITE operation in some fixed amount of time that is independent of the location's address. Main memory units are of this type. This distinguishes them from serial or partly serial access storage devices such as magnetic tapes and disks which are used as the secondary storage device.

Cache Memory:-

The CPU of a computer can usually process instructions and data faster than they can be fetched from compatibly priced main memory unit. Thus the memory cycle time become the bottleneck in the system. One way to reduce the memory access time is to use cache memory. This is a small and fast memory that is inserted between the larger, slower main memory and the CPU. This holds the currently active segments of a program and its data. Because of the locality of address references, the CPU can, most of the time, find the relevant information in the cache memory itself (cache hit) and infrequently needs access to the main memory (cache miss) with suitable size of the cache memory, cache hit rates of over 90% are possible leading to a cost-effective increase in the performance of the system.

Memory Interleaving: -

This technique divides the memory system into a number of memory modules and arranges addressing so that successive words in the address space are placed in different modules. 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.

Virtual Memory: -

In a virtual memory System, the address generated by the CPU is referred to as a virtual or logical address. The corresponding physical address can be different and the required mapping is implemented by a special memory control unit, often called the memory management unit. The mapping function itself may be changed during program execution according to system requirements.

Because of the distinction made between the logical (virtual) address space and the physical address space; while the former can be as large as the addressing capability of the CPU, the actual physical memory can be much smaller. Only the active portion of the virtual address space is mapped onto the physical memory and the rest of the virtual address space is mapped onto the bulk storage device used. If the addressed information is in the Main Memory (MM), it is accessed and execution proceeds. Otherwise, an exception is generated, in response to which the memory management unit transfers a contigious block of words containing the desired word from the bulk storage unit to the MM, displacing some block that is currently inactive. If the memory is managed in such a way that, such transfers are required relatively infrequency (ie the CPU will generally find the required information in the MM), the virtual memory system can provide a reasonably good performance and succeed in creating an illusion of a large memory with a small, in expensive MM.

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 constitutes 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. Figure 3..1 shows the possible arrangements of memory cells.

The Sense / Write circuits are connected to data input or output lines of the chip. During a write operation, the sense / write circuit receives input information and stores it in the cells of the selected word. The data input and data output of each sense / write circuits are connected to a single bidirectional data line that can be connected to a data bus of the cpu.

R / W → specifies the required operation.

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

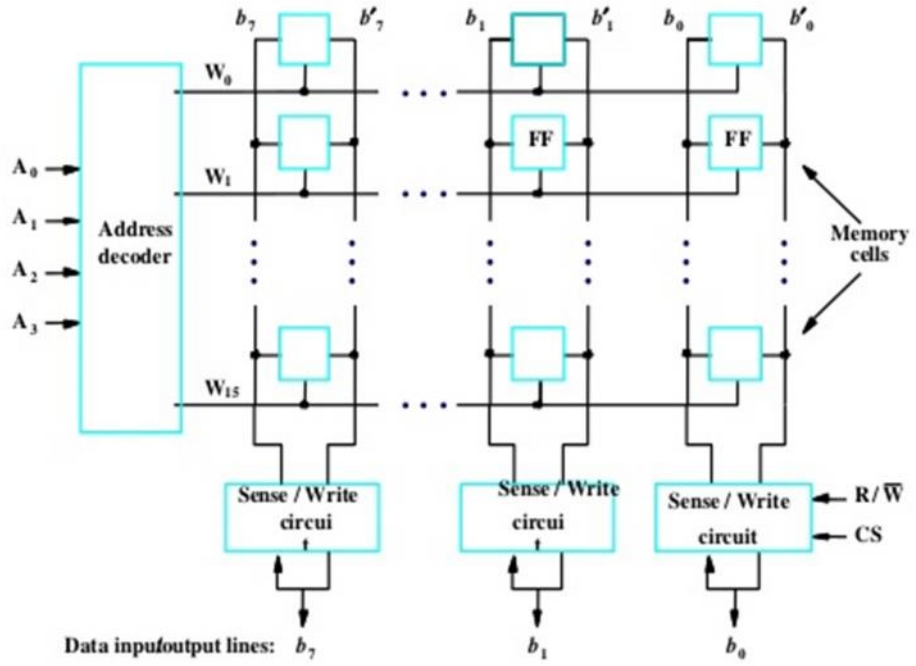


Figure Organization of bit cells in a memory chip.

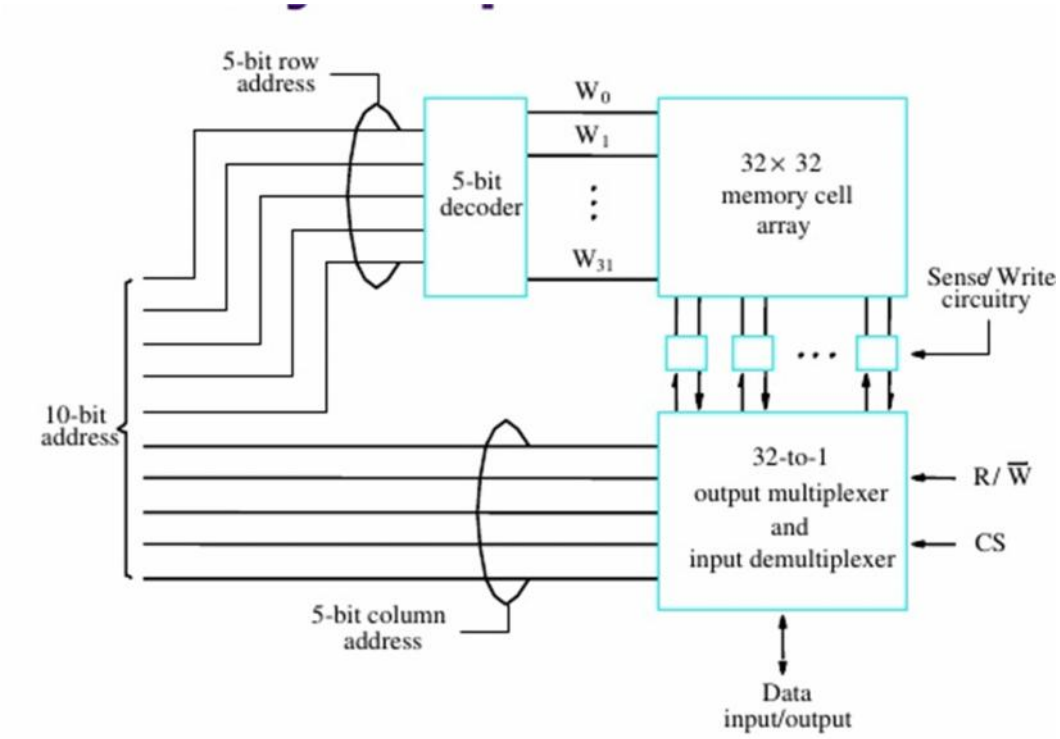


Figure 5.3. Organization of a 1K × 1 memory chip.

3.3 SEMI CONDUCTOR RAM MEMORIES:

Semi-Conductor memories are available is a wide range of speeds. Their cycle time ranges from 100ns to 10ns. When first introduced in the late 1960s, they were much more expensive. But now they are very cheap, and used almost exclusively in implementing main memories.

3.3.1STATIC MEMORIES:

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

Static random-access memory (**SRAM**) is a type of semiconductor memory that uses bistable latching circuitry to store each bit. The term *static* differentiates it from *dynamic* RAM (DRAM) which must be periodically refreshed. SRAM exhibits data remanence, but is still *volatile* in the conventional sense that data is eventually lost when the memory is not powered. Figure 3.2 shows the implementation of static RAM.

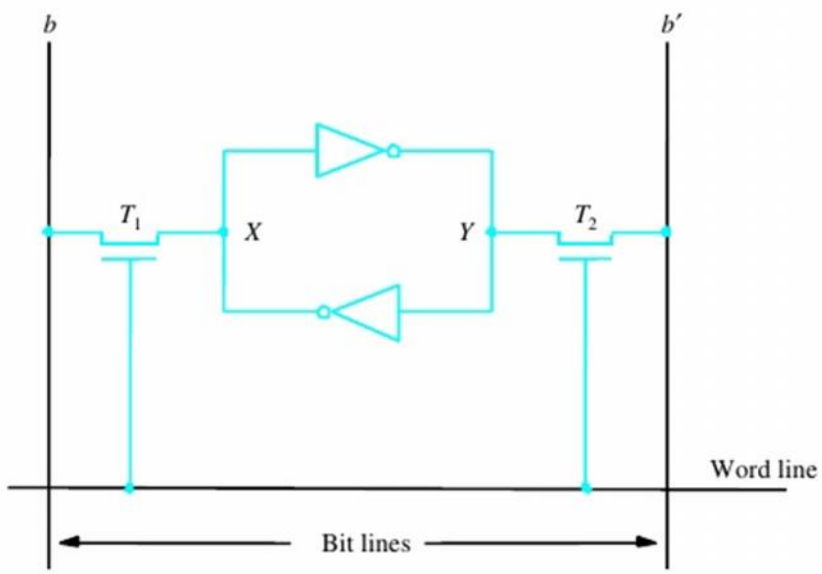


fig: 3.2Static RAM cell

Two inverters are cross connected to form a latch. The latch is connected to two bit lines by transistors T_1 and T_2. 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 retains its state.

Read Operation:

In order to read the state of the SRAM cell, the word line is activated to close switches T_1 and T_2. 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 complement of each other. Sense / write circuit at the end of the bit line monitors the state of b and b' and set the output according.

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.

CMOS RAM CELL

Transistor pairs (T_3, T_5) and (T_4, T_6) form the inverters in the latch. In state 1, the voltage at point X is high by having T_5, T_6 on and T_4, T_5 are OFF. Thus T_1 and T_2 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.

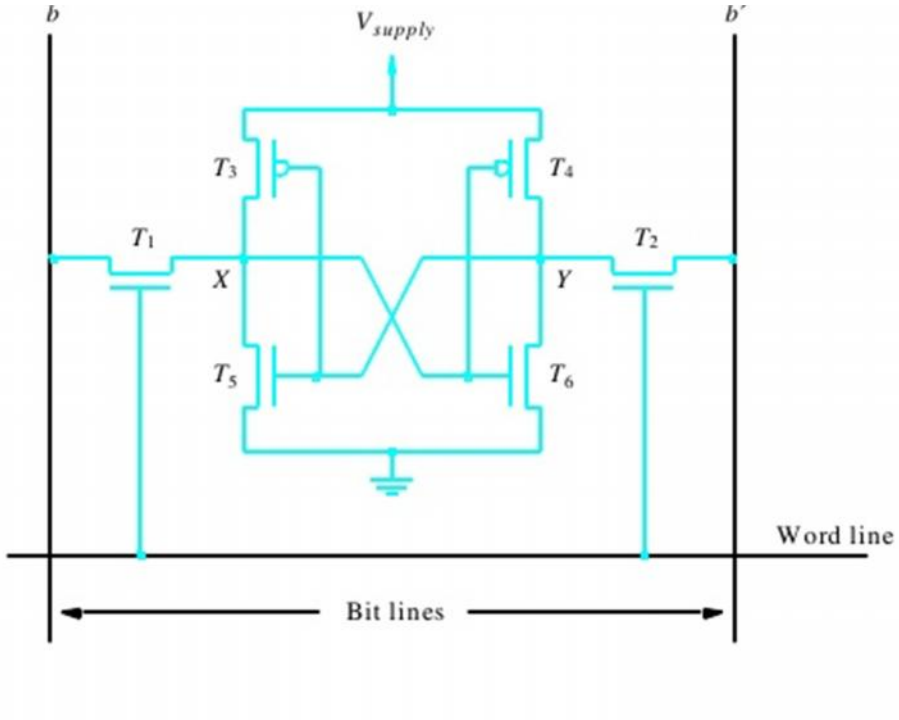


Figure3.3 : CMOS cell (Complementary Metal oxide Semi-Conductor):

Merit:

- It has low power consumption because the current flows in the cell only when the cell is being accessed.
- Static RAMs can be accessed quickly. It access time is few nanoseconds.

Demerit:

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

Asynchronous DRAMS:

Less expensive RAM‟s can be implemented if simpler cells are used. Such cells 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 a few milliseconds. The contents must be periodically refreshed by restoring this capacitor charge to its full value.

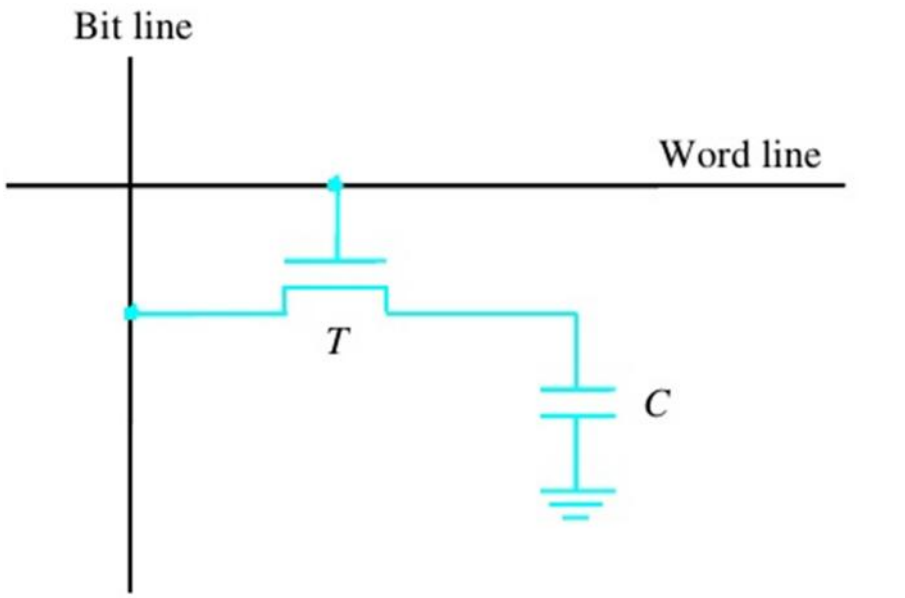


Figure 3.4: A single transistor dynamic Memory cell

In order to store information in the cell, the transistor T is turned on and 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, as shown in Figure 3.4.

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**.
During a read operation, the transistor is turned on and a sense amplifier connected to the bit line detects whether the charge on the capacitor is above the threshold value.
A 16-megabit DRAM chip configured as 2M x 8, is show n in Figure 3.5.

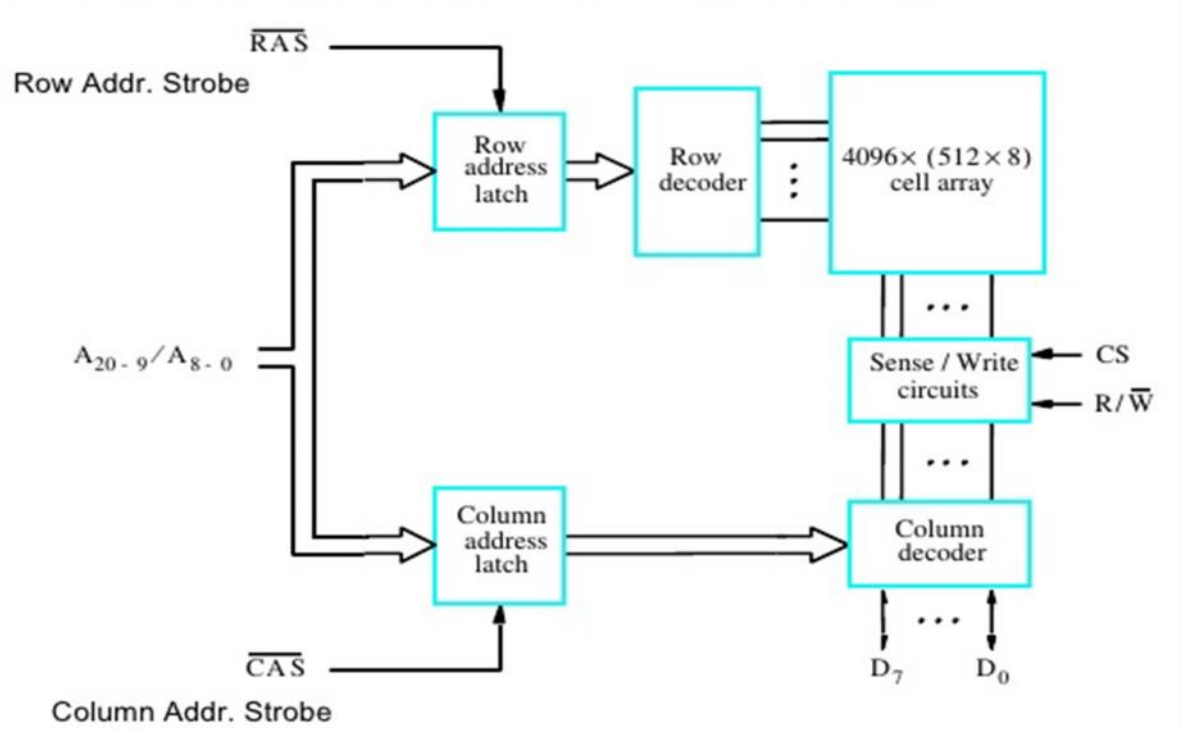


Figure 3.5: Internal organization of a 2M X 8 dynamic Memory chip.

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, and 9 bits specify the group of 8 bits in the selected row).

A (0-8) → Row address of a byte.

A (9-20) → 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 and loaded into Column Address Strobe (CAS). The information in this latch is decoded and the appropriate group of 8 Sense/Write circuits is selected. R/W =1(read

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 is 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 and 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 and 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. The Figure 3.6 shows the structure of SDRAM.

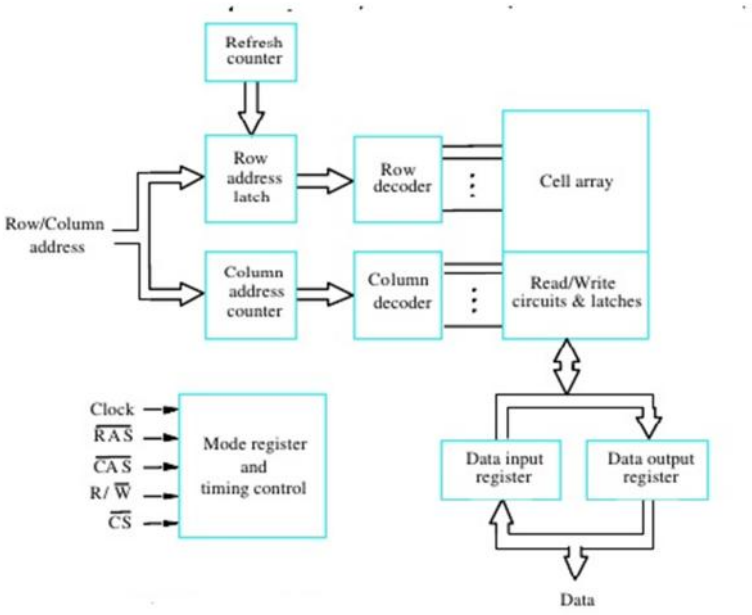


Figure 3.6: Synchronous DRAM

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.

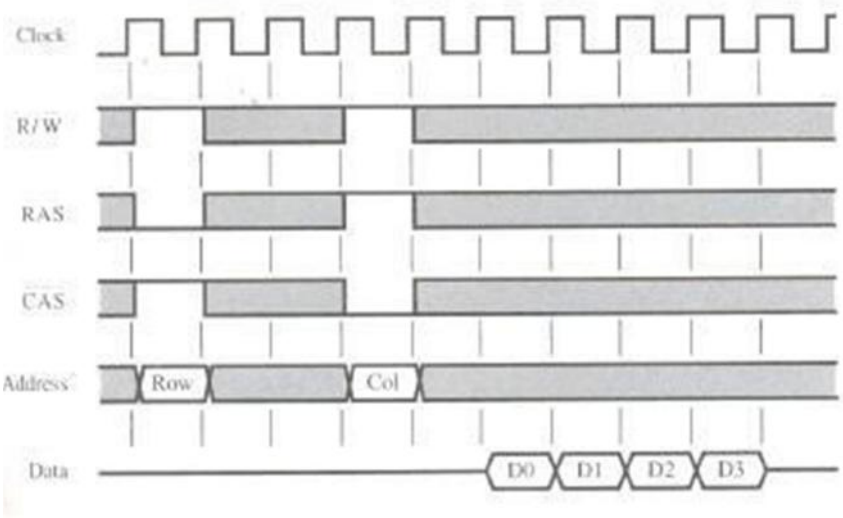


Figure 3.7: Timing Diagram Burst Read of Length 4 in an SDRAM

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. A timing diagram for a typical burst of read of length 4 is shown in Figure 3.7.

Latency and Bandwidth:

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

- Latency
- Bandwidth

Latency 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 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 and on the number of bits that can be accessed in parallel.

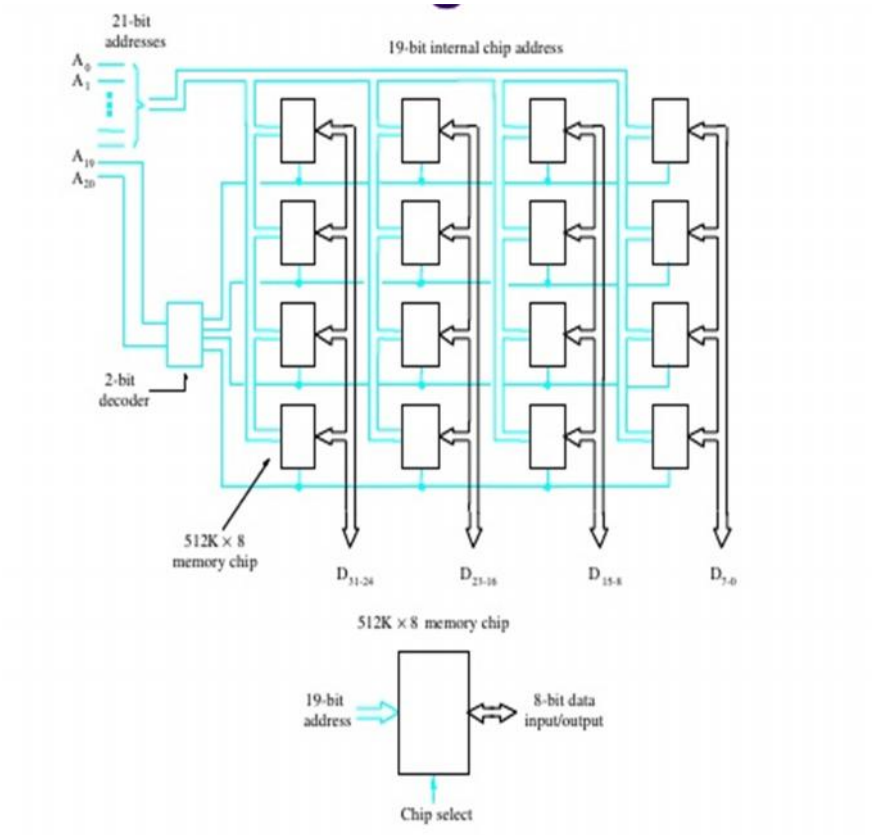
Double Data Rate SDRAM (DDR-SDRAM):

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.

3.3.2 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 consideration have led to the development of larger memory units known as SIMM's and DIMM's

SIMM-Single Inline memory Module
DIMM-Dual Inline memory Module



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 and 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, as shown in Figure 3.8.

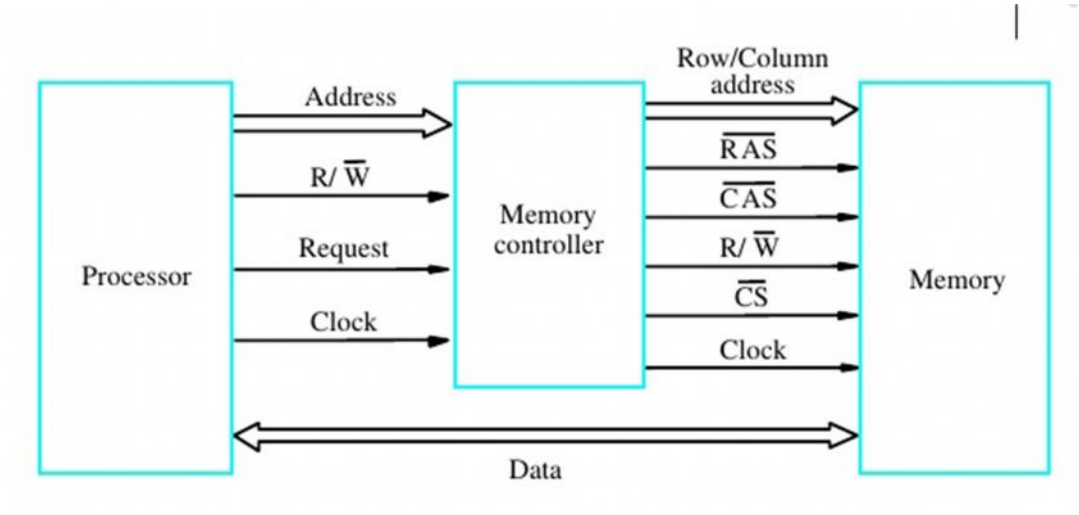


Figure 3.8: Use of Memory Controller

The Controller accepts a complete address and 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 and column portions of the address to the memory and generates RAS and CAS signals. It also sends R/W and 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.

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 Vsupply to represent the logical values, the signals consist 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 Signaling. 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 (RDRAM). Rambus channel has,

- 9 Data lines (1-8 Transfer the data, 9th line → Parity checking).
- Control line
- Power line

A two channel Rambus has 18 data lines which have no separate address lines. It is also called as Direct RDRAM's. Communication between processor or some other device that can serves as a master and RDRAM modules are served 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

3.4 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 applications require Non-volatile memory (which retains the stored information if power is turned off).

E.g.: Operating System software has to be loaded from disk to memory which requires the program that boots the Operating System. i.e., 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.

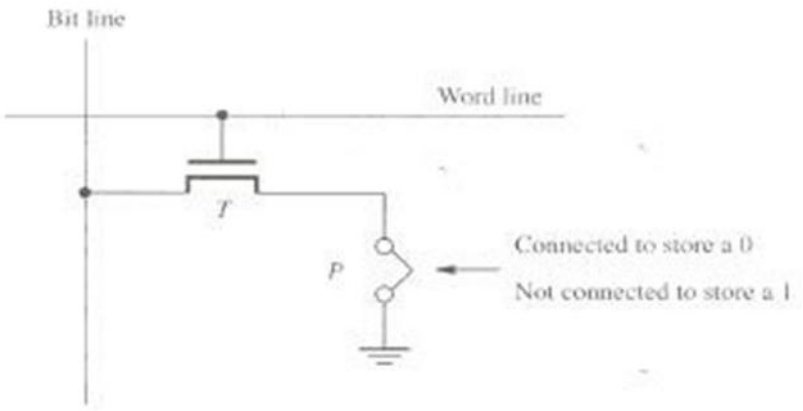


Figure 3.9: ROM cell

At Logic value ‗0' → Transistor (T) is connected to the ground point (P). Transistor switch is closed and voltage on bit line nearly drops to zero.

At Logic value ‗1' → Transistor switch is open. The bit line remains at high voltage. To read the state of the cell, the word line is activated. A Sense circuit at the end of the bit line generates the proper output value.

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. Programmability is achieved by inserting a fuse at point P in a ROM cell. Before it is programmed, the memory contains all 0's. The user can insert 1's at the required location by burning out the 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 ultraviolet 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.

EEPROM:- Electrically Erasable ROM:

EEPROM (also written **E^2PROM** and pronounced "e-e-prom," "double-e prom," "e-squared," or simply "e-prom") stands for **E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory and is a type of non-volatile memory used in computers and other electronic devices to store small amounts of data that must be saved when power is removed, e.g., calibration tables or device configuration.

When larger amounts of static data are to be stored (such as in USB flash drives) a specific type of EEPROM such as flash memory is more economical than traditional EEPROM devices. EEPROMs are realized as arrays of floating-gate transistors. EEPROM is user modifiable read only memory (ROM) that can be erased and reprogrammed (written to) repeatedly through the application of higher than normal electrical voltage generated externally or internally in the case of modern EEPROMs.

EPROM usually must be removed from the device for erasing and programming, whereas EEPROMs can be programmed and erased in circuit. Originally, EEPROMs were limited to single byte operations which made them slower, but modern EEPROMs allow multi-byte page operations. It also has a limited life - that is, the number of times it could be reprogrammed was limited to tens or hundreds of thousands of times.

That limitation has been extended to a million write operations in modern EEPROMs. In an EEPROM that is frequently reprogrammed while the computer is in use, the life of the EEPROM can be an important design consideration. It is for this reason that EEPROMs were used for configuration information, rather than random access memory.

Merits:

It can be both programmed and erased electrically.

It allows the erasing of all cell contents selectively.

Demerits:

It requires different voltage for erasing, writing and reading the stored data.

FLASH MEMORY:

In EEPROM, it is possible to read and 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.

E.g.: 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 and low cost per bit.
- It requires single power supply voltage and 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 is of 8, 32,64MB. E.g.: 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.

Demerit:

- 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 SRA‘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

3.5 MEMORY HIERARCHY

To this point in our study of systems, we have relied on a simple model of a computer system as a CPU that executes instructions and a memory system that holds instructions and data for the CPU. In our simple \model, the memory system is a linear array of bytes, and the CPU can access each memory location in a constant amount of time. While this is an effective model as far as it goes, it does not reflect the way that modern systems really work. In practice, a memory system is a hierarchy of storage devices with different capacities, costs, and access times. CPU registers hold the most frequently used data. Small, fast cache memories nearby the CPU act as staging areas for a subset of the data and instructions stored in the relatively slow main memory.

The main memory stages data stored on large, slow disks, which in turn often serve as staging areas for data stored on the disks or tapes of other machines connected by networks.

Memory hierarchies work because well written programs tend to access the storage at any particular level more frequently than they access the storage at the next lower level. So the storage at the next level can be slower, and thus larger and cheaper per bit. The overall effect is a large pool of memory that costs as much as the cheap storage near the bottom of the hierarchy, but that serves data to programs at the rate of the fast storage near the top of the hierarchy.

As a programmer, you need to understand the memory hierarchy because it has a big impact on the performance of your applications. If the data your program needs are stored in a CPU register, then they can be accessed in zero cycles during the execution of the instruction. If stored in a cache, 1 to 30 cycles. If stored in main memory, 50 to 200 cycles.

And if stored in disk tens of millions of cycles!. The entire computer memory can be realized as the hierarchy shown in the Figure 14.10.

Here, then, is a fundamental and enduring idea in computer systems: If you understand how the system moves data up and down the memory hierarchy, then you can write your application programs so that their data items are stored higher in the hierarchy, where the CPU can access them more quickly. This idea centers on a fundamental property of computer programs known as locality.

Programs with good locality tend to access the same set of data items over and over again, or they tend to access sets of nearby data items. Programs with good locality tend to access more data items from the upper levels of the memory hierarchy than programs with poor locality, and thus run faster.

For example, the running times of different matrix multiplication kernels that perform thesame number of arithmetic operations, but have different degrees of locality, can vary by a factor of 20!

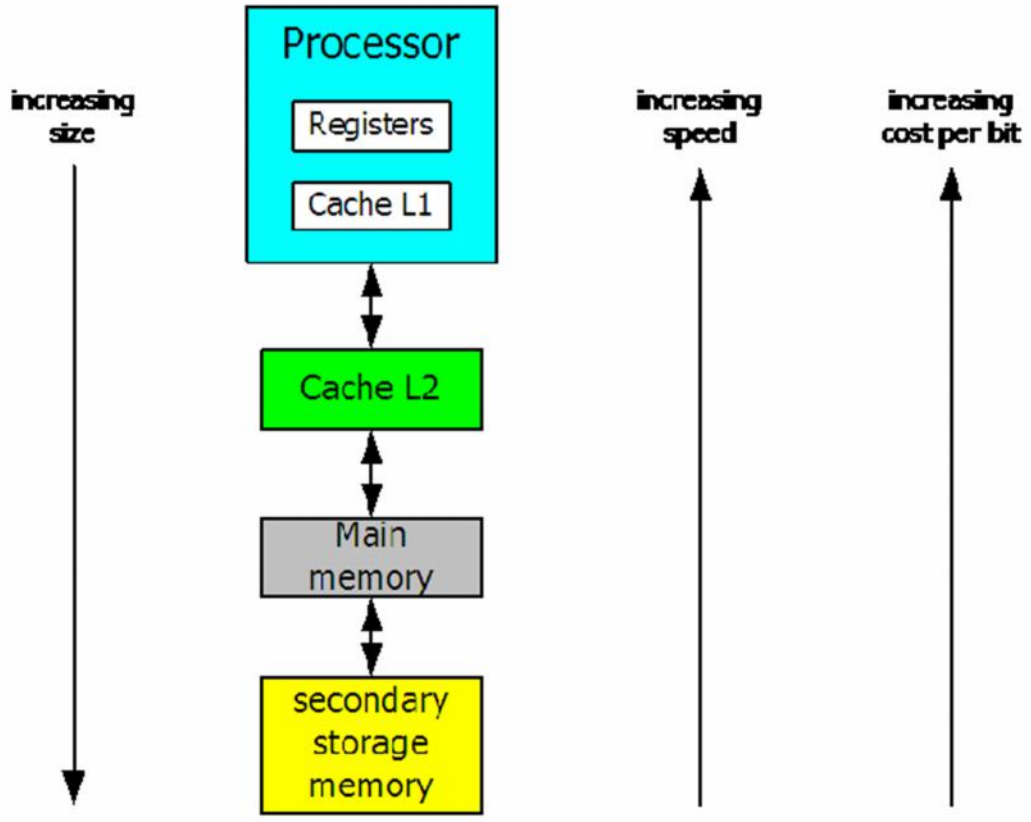


Figure : Memory Hierarchy

Types of Cache Memory:

The Cache memory is of 2 types. They are:

- Primary /Processor Cache (Level1 or L1 cache)
- Secondary Cache (Level2 or L2 cache)

Primary Cache → It is always located on the processor chip.

Secondary Cache → It is placed between the primary cache and the rest of the memory.

The main memory is implemented using the dynamic components (SIMM, RIMM, and DIMM). The access time for main memory is about 10 times longer than the access time for L1 cache.

3.6 CACHE MEMORY

Cache memory is an integral part of every system now. Cache memory is random access memory (RAM) that a computer microprocessor can access more quickly than it can access regular RAM. As the microprocessor processes data, it looks first in the cache memory and if it finds the data there (from a previous reading of data), it does not have to do the more time consuming reading of data from larger memory.

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

Locality of Reference:

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 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.

If the active segment of a program can be placed in a fast cache memory, then the total execution time can be reduced significantly. The operation of a cache memory is very simple. The memory control circuitry is designed to take advantage of the property of locality of reference. 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.

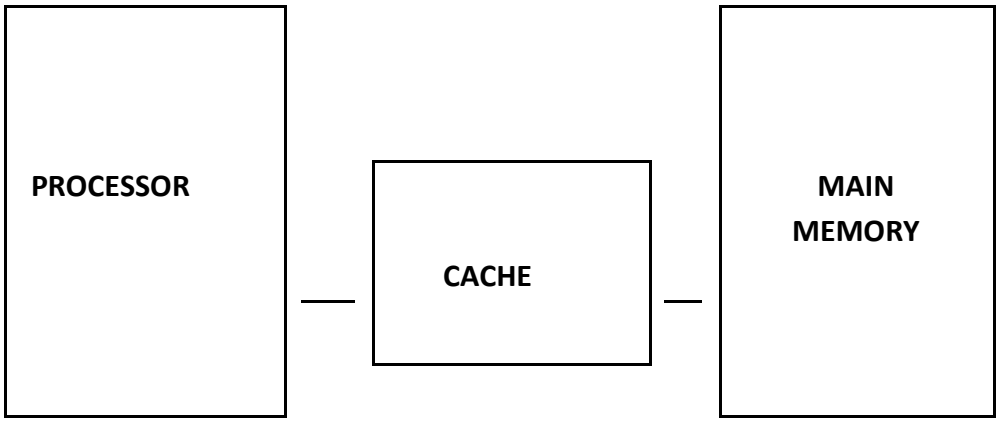


Figure : Use of Cache Memory

The Figure shows arrangement of Cache between processor and main 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 decides 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 be involved.

The write operation proceeds 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 does not exist 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 does exist 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 blocks containing the addressed word is first brought into the cache and then the desired word in the cache is overwritten with the new information.

3.6.1 CACHE MEMORY DESIGN PARAMETERS

There are two cache design parameters that dramatically influence the cache performance: the block size and the cache associability. There are also many other implementation techniques both hardware and software that improve the cache performance but they are not discussed here.

The simplest way to reduce the miss rate is to increase the block size. However increasing the block size also increases the miss penalty (which is the time to load a block from main memory into cache) so there is a trade–off between the block size and miss penalty. We can increase the block size up to a level at which the miss rate is decreasing but we also have to be sure that this benefit will not be exceeded by the increased miss penalty.

The second cache design parameter that reduces cache misses is the associability. There is an empirical result called the 2:1 rule of thumb which states that a direct mapped cache of size N has about the same miss rate as a 2 way set associative cache of size N/2. Unfortunately an increased associability will have a bigger hit time. More time will be taken to retrieve a block inside of an associative cache than in a direct mapped cache. To retrieve a block in an associative cache, the block must be searched inside of an entire set since there is more than one place where the block can be stored.

Based on the cause that determines a cache miss we can classify the cache misses as compulsory, capacity and conflict misses. This classification is called the 3C model. Compulsory misses are issued when a first access is done to a block that is not in the memory, so the block must be brought into cache. Increasing block size can reduce compulsory misses due to perfecting the other elements in the block. If the cache cannot contain all the blocks needed during the execution of a program, capacity misses will occur due to blocks being discarded and later retrieved. If the block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory and capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. Increasing the associability in general reduce the number of conflict misses and implicitly the runtime of the programs. However this is not true all the time. Minimizing the cache misses does not necessarily minimize the runtime. For example, there can be fewer cache misses with more memory accesses.

3.6.2 Mapping Function:

Direct Mapping:

It is the simplest technique in which block j of the main memory maps onto block ‗J' modulo 128 of the cache. Thus whenever one of the main memory blocks 0, 128, 256 is loaded in the cache, it is stored in block 0. Block 1, 129, 257 are stored in cache block 1 and so on. The contention may arise when the cache is full, when more than one memory block is mapped onto a given cache block position. The contention is resolved by allowing the new blocks to overwrite the currently resident block. Placement of block in the cache is determined from memory address.

The memory address is divided into 3 fields, namely,

- **Low Order 4 bit field (word) →** Select one of 16 words in a block.
- **7 bit cache block field →** When new block enters cache, 7 bit determines the cache position in which this block must be stored.
- **5 bit Tag field →** The high order 5 bits of the memory address of the block is stored in 5 tag bits associated with its location in the cache.

As execution proceeds, the high order 5 bits of the address is compared with tag bits associated with that cache location. If they match, then the desired word is in that block of the cache. If there is no match, then the block containing the required word must be first read from the main memory and loaded into the cache. The direct mapping is shown in Figure 15.2.

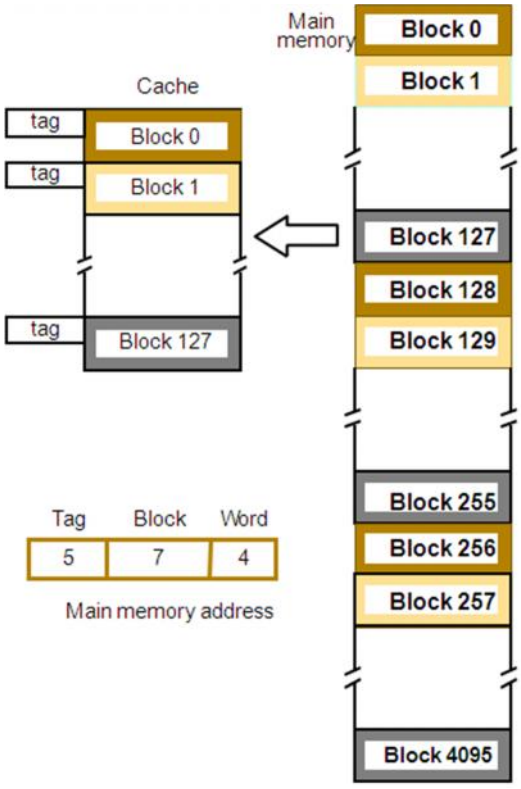


Figure 15.2: Direct Mapped Cache

Merit:

- It is easy to implement.

Demerit:

- It is not very flexible.

Associative Mapping:

Here, the main memory block can be placed into any cache block position.12 tag bits will identify a memory block when it is resolved 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.** It gives complete freedom in choosing the cache location. A new block that has to be brought into the cache has to replace (eject) an existing block if the cache is full.

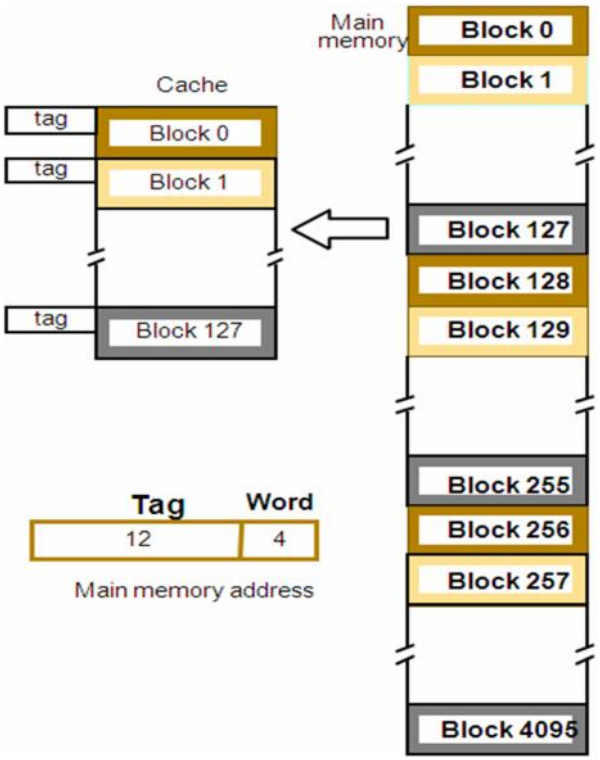


Figure 15.3: Associative Mapped Cache.

In this method, the memory has to determine whether a given block is in the cache. A search of this kind is called an associative Search. The associative-mapped cache is shown in Figure 15.3.

Merit:

- It is more flexible than direct mapping technique.

Demerit:

- Its cost is high.

Set-Associative Mapping:

It is the combination of direct and associative mapping. The blocks of the cache are grouped into sets and the mapping allows a block of the main memory to reside in any block of the specified set. In this case, the cache has two blocks per set, so the memory blocks 0, 64, 128……..4032 map into cache set to 0 and they can occupy either of the two block position within the set.

6 bit set field → Determines which set of cache contains the desired block.

6 bit tag field → the tag field of the address is compared to the tags of the two blocks of the set to clock if the desired block is present.

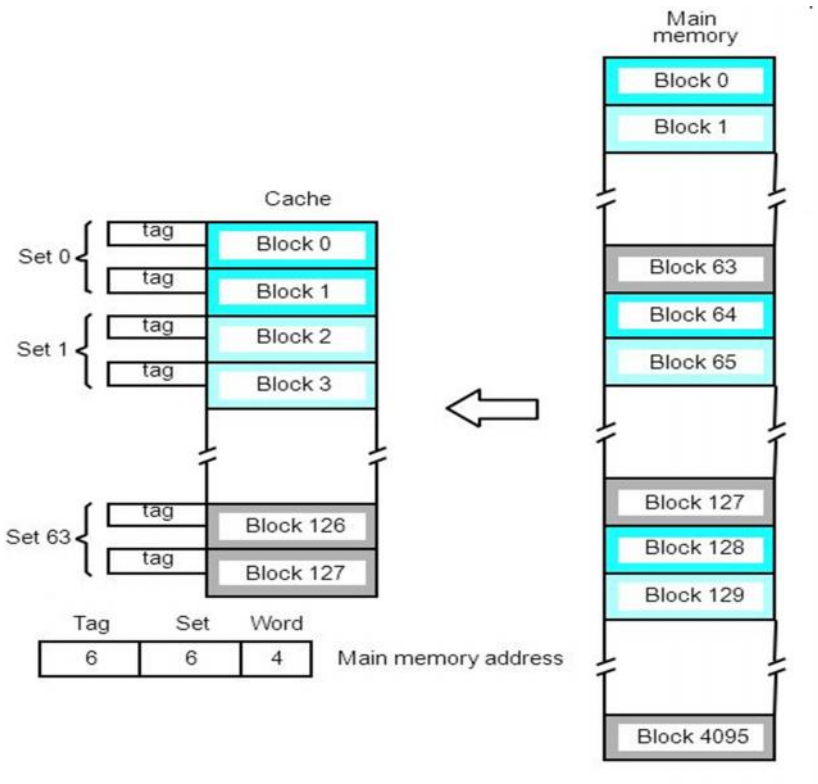


Figure 15.4: Set-Associative Mapping:

No of blocks per set	No of set field
2	6
3	5
8	4
128	no set field

The cache which contains 1 block per set is called direct Mapping. A cache that has ‗k' blocks per set is called as k-way set associative cache. Each block contains a control bit called a valid bit. The Valid bit indicates that whether the block contains valid data. The dirty bit indicates that whether the block has been modified during its cache residency. The set-associative mapping is shown in Figure 15.4.

Valid bit=0→When power is initially applied to system
Valid bit =1→When the block is loaded from main memory at first time.

If the main memory block is updated by a source and if the block in the source is already exists in the cache, then the valid bit will be cleared to ‗0'. If Processor and DMA use the same copies of data then it is called as the Cache Coherence Problem.

Merit:

- The Contention problem of direct mapping is solved by having few choices for block placement.
- The hardware cost is decreased by reducing the size of associative search.

3.7 REPLACEMENT ALGORITHM:

In a direct-mapped cache, the position of each block is predetermined: hence, no replacement strategy exists. In associative and set-associative caches there exists some flexibility. 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 an important issue, because the decision can be a strong determining factor in system performance. In general, the objective is to keep blocks in the cache that are likely to be referenced in the near future. However, it is not easy to determine which blocks are about to be referenced.

The property of locality of reference in programs gives a clue to a reasonable strategy. Because, programs usually stay in localized areas for reasonable periods of time, there is a high probability that the blocks that have been referenced recently will be referenced again soon. Therefore, when a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced. This block is called the least recently used (LW) block, and the technique is called the LRU replacement algorithm. To use the LRU algorithm, the cache controller must track references to all blocks as computation proceeds.

Suppose it is required to track the LRU block of a four block set in a set-associative cache. A 2-bit counter can be used for each block. When a hit occurs, the counter of the block that is referenced is set to 0. Counters with values originally lower than the referenced one are incremented by one, and all others remain unchanged. When a miss occurs and the set is not full, the counter associated with the new block loaded from the main memory is set to 0, and the values of all other counters are increased by one. When a miss occurs and the set is full, the block with the counter value 3 is removed, the new block is put in its place, and its counter is set to 0.

The other three block counters are incremented by one (It can be easily verified that the counter values of occupied blocks are always distinct). The LRU algorithm has been used extensively. Although it performs well for many access patterns, 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 the LRU algorithm can be improved by introducing a small amount of randomness in deciding which block to replace.

Several other replacement algorithms are also used in practice. An intuitively reasonable rule would be to remove the "oldest" block from a full set when a new block must be brought in. However, because this algorithm does not take into account the recent pattern of access to blocks in the cache, it is generally not as effective as the LRU algorithm in choosing the best blocks to remove. The simplest algorithm is to randomly choose the block to be overwritten. Interestingly enough, this simple algorithm has been found to be quite effective in practice.

Example: Let us **c**onsider 4 blocks/set, in set associative cache, where 2 bit counter can be used for each block. When a ‗hit' occurs, then block counter = 0, the counter with values originally lower than the referenced one are incremented by 1 and all others remain unchanged. When a ‗miss' occurs and if the set is full, the blocks with the counter value 3 is removed, the new block is put in its place and its counter is set to ‗0' and other block counters are incremented by 1.

Merit:

The performance of LRU algorithm is improved by randomness in deciding which block is to be overwritten.

3.8 PERFORMANCE CONSIDERATION:

Two Key factors in the commercial success are the performance and cost where the best possible performance is at low cost. A common measure of success is called the Price Performance ratio.

Performance depends on how fast the machine instructions are brought to the processor and how fast they are executed. To achieve parallelism (i.e., both the slow and fast units are accessed in the same manner) interleaving is used.

3.8.1 Interleaving:

If the main memory is structured as a collection of physically separated modules, each with its own ABR (Address buffer register) and DBR(Data buffer register), memory access operations may proceed in more than one module at the same time. Thereby the aggregate rate of transmission of words to and from the main memory system can be increased.

Two methods of address layout are indicated in Figure 3.5. In the first case, memory address generated by the processor is decoded as shown in part (a) of the figure. The high-order k bits name one of n modules and the low-order m bits name a particular word in that module. When consecutive locations are accessed, only one module is

involved. At the same time, devices with DMA ability may be accessing information in other modules.

In the second case, as shown in part (b) of the figure, which is called memory interleaving. The low-order k bits of the memory address select a module, and the high-order m bits name a location within the module. Thus, any component of the system that generates requests for access to consecutive memory locations can keep several modules busy at any one time which results in both faster access to a block of data and higher average utilization of the memory system as a whole.

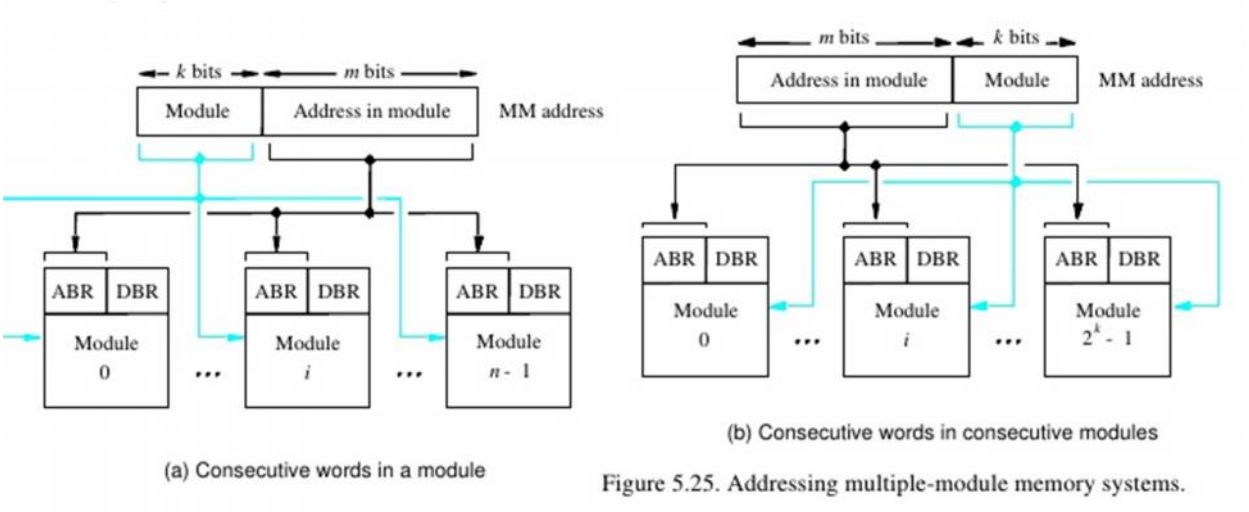


Figure : Addressing multiple-module memory system

3.8.2 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 levels of the hierarchy. Recall that a successful access to data in a cache is called a hit. The number of hits stated as a fraction of all attempted accesses is called the hit rate, and the miss rate is the number of misses stated as a fraction of attempted accesses. Ideally, the entire memory hierarchy would appear to the CPU as a single memory unit that has the access time of a cache on the CPU chip and the size of a magnetic disk. How close we get to this ideal depends largely on the hit rate at different levels of the hierarchy. High hit rates,over 0.9, are essential for high performance computers. Performance is adversely affected by the actions that must be taken after a miss. The extra time needed to bring the desired information into the cache is called the Miss penalty. This penalty is

ultimately reflected in the time that the CPU is stalled because the required instructions or data are not available for execution. In general, the miss penalty is the time needed to bring a block of data from a slower unit in the memory hierarchy to a faster unit. The miss penalty is reduced if efficient mechanisms for transferring data between the various units of the hierarchy are implemented. The previous section shows how an interleaved memory can reduce the miss penalty substantially. 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 CPU is hC + (1 – h) M.

3.8.3 CACHES ON PROCESSING CHIPS

When information is transferred between different chips, considerable delays are introduced in driver and receiver gates on the chips. Thus, from the speed point of view, the optimal place for a cache is on the CPU chip. Unfortunately, space on the CPU chip is needed for many other functions: this limits the size of the cache that can be accommodated. All high performance processor chips include some form of a cache. Some manufacturers have chosen to implement two separate caches, one for instructions and another for data, as in the 68040 and PowerPC 604 processors. Others have implemented a single cache for both instructions and data, as in the PowerPC 601 processor. A combined cache for instructions and data is likely to have a somewhat better hit rate, because it offers greater flexibility in mapping new information into the cache. However, if separate caches arc used, it is possible to access both caches at the same time, which leads to increased parallelism and, hence, better performance. The disadvantage of separate caches is that the increased parallelism comes at the expense of more complex circuitry. Since the Size of a cache on the CPU chip is limited by space constraints, a good strategy for designing a high performance system is to use such a cache as a primary cache. An external secondary cache, constructed with SRAM chips, is

When added to provide the desired capacity. If both primary and secondary caches are used, the primary cache should be designed to allow very fast access by the CPU, because its access time will have a large effect on the clock rate of the CPU. 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 in the cache is to access more than one word simultaneously and then let the CPU use them one at a time. The secondary cache can be considerably 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 primary cache. A workstation computer may include a primary cache with the capacity of tens of kilobytes and a secondary cache of several megabytes.

3.8.4 OTHER ENHANCEMENTS

In addition to the main design issues just discussed, several other possibilities exist for enhancing performance. We discuss three of them in this section.

Write Buffer

When the write-through protocol is used, each write operation results in writing a new value into the main memory. If the CPU must wait for the memory function to be completed, as we have assumed until now, then the CPU is slowed down by all write requests. Yet the CPU typically does not immediately depend on the result of a write operation, so it is not necessary for the CPU to wait for the write request to be completed. To improve performance, a write buffer can be included for temporary storage of write requests. The CPU places each write request into this buffer and continues execution of the next instruction. The write requests stored in the write buffer are sent to the main memory whenever the memory is not responding to read requests. Note that it is important that the read requests be serviced immediately, because the CPU usually cannot proceed without the data that is to be read from the memory. Hence, these requests are given priority over write requests. The write buffer may hold a number of write requests. Thus, it is possible that a subsequent read request may refer to data that are still in the write buffer. To ensure correct operation, the addresses of data to be read

from the memory are compared with the addresses of the data in the write buffer. In case of a match, the data in the write buffer are used. This need for address comparison entails considerable cost. But the cost is justified by improved performance. A different situation occurs with the write-back protocol. In this case, the write operations are simply performed on the corresponding word in the cache. But consider what happens when a new block of data is to be brought into the cache as a result of a read miss, which replaces an existing block that has some dirty data. The dirty block has to be written into the main memory. If the required write-back is performed first, then the CPU will have to wait longer for the new block to be read into the cache. It is more prudent to read the new block first. This can be arranged by providing a fast write buffer for temporary storage of the dirty block that is ejected from the cache while the new block is being read. Afterward, the contents of the buffer are written into the main memory. Thus, the write buffer also works well for the write-back protocol.

Prefetching

In the previous discussion of the cache mechanism, we assumed that new data are brought into the cache when they are first needed. A read miss occurs, and the desired data are loaded from the main memory. The CPU has to pause until the new data

arrive, which is the effect of the miss penalty. To avoid stalling the CPU, it is possible to prefetch the data into the cache before they are needed. The simplest way to do this is through software. A special prefetch instruction may be provided in the instruction set of the processor. Executing this instruction causes the addressed data to be loaded into the cache, as in the case of a read miss. However, the processor does not wait for the referenced data. A prefetch instruction is inserted in a program to cause the data to be loaded in the cache by the time they are needed in the program. The hope is that prefetching will take place while the CPU is busy executing instructions that do not result in a read miss, thus allowing accesses to the main memory to be overlapped with computation in the CPU. Prefetch instructions can be inserted into a program either by the programmer or by the compiler. It is obviously preferable to have the compiler insert these instructions, which can be done with good success for many applications. Note that software prefetching entails a certain overhead; because inclusion of prefetch instructions increases the length of programs. Moreover, some prefetches may load into the cache data that will not be used by the instruction. This can happen if the prefetched data are ejected from the cache by a read miss involving other data. However, the overall effect of software prefetching on performance is positive, and many processors (including the PowerPC) have machine instructions to support this feature. Prefetching can also be done through hardware. This involves adding circuitry that attempts to discover a pattern in memory references, and then prefetches data according to this pattern.

Lockup-Free

Cache the software prefetehing scheme just discussed does not work well if it interferes significantly with the normal execution of instructions. This is the case if the action of prefetehing 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. We can solve this problem by modifying the basic cache structure to allow the CPU to access the cache while a miss is being serviced. In fact, it is desirable that more than one outstanding miss can be supported. A cache that can support multiple outstanding misses is called lockup-free. Since it can service only one miss at a time, it must include circuitry that keeps track of all outstanding misses. This may be done with special registers that hold the pertinent inFormation about these misses. Lockup-free caches were first used in the early l980s in the Cyber series of computers manufactured by Control Data Company.

We have used software prefetching as an obvious motivation for a cache that is not locked by a read miss. A much more important reason is that, in a processor that uses a pipelined organization, which overlaps the execution of several instructions, a read miss caused by one instruction could stall the execution of other instructions. A lockup-free cache reduces the likelihood of such stalling.

3.9 VIRTUAL MEMORY:

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

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.

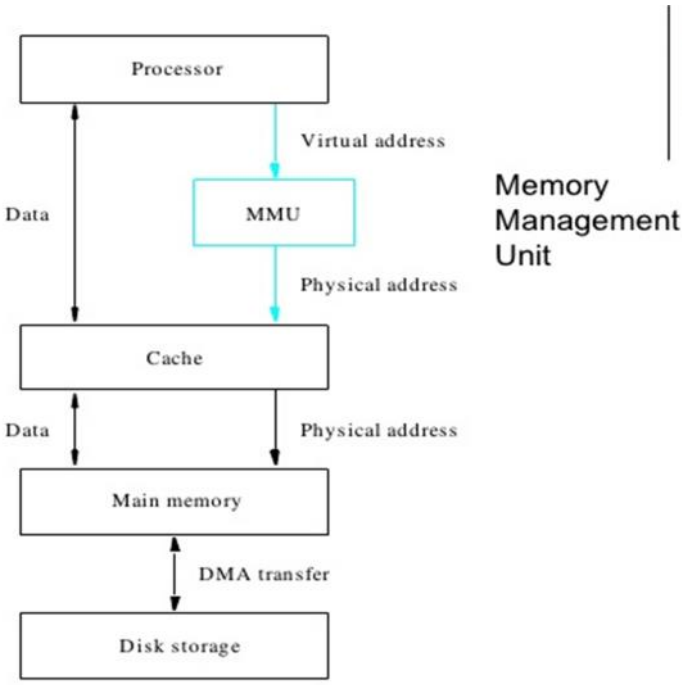


Fig:Virtual Memory Organisation

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 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 bits 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.

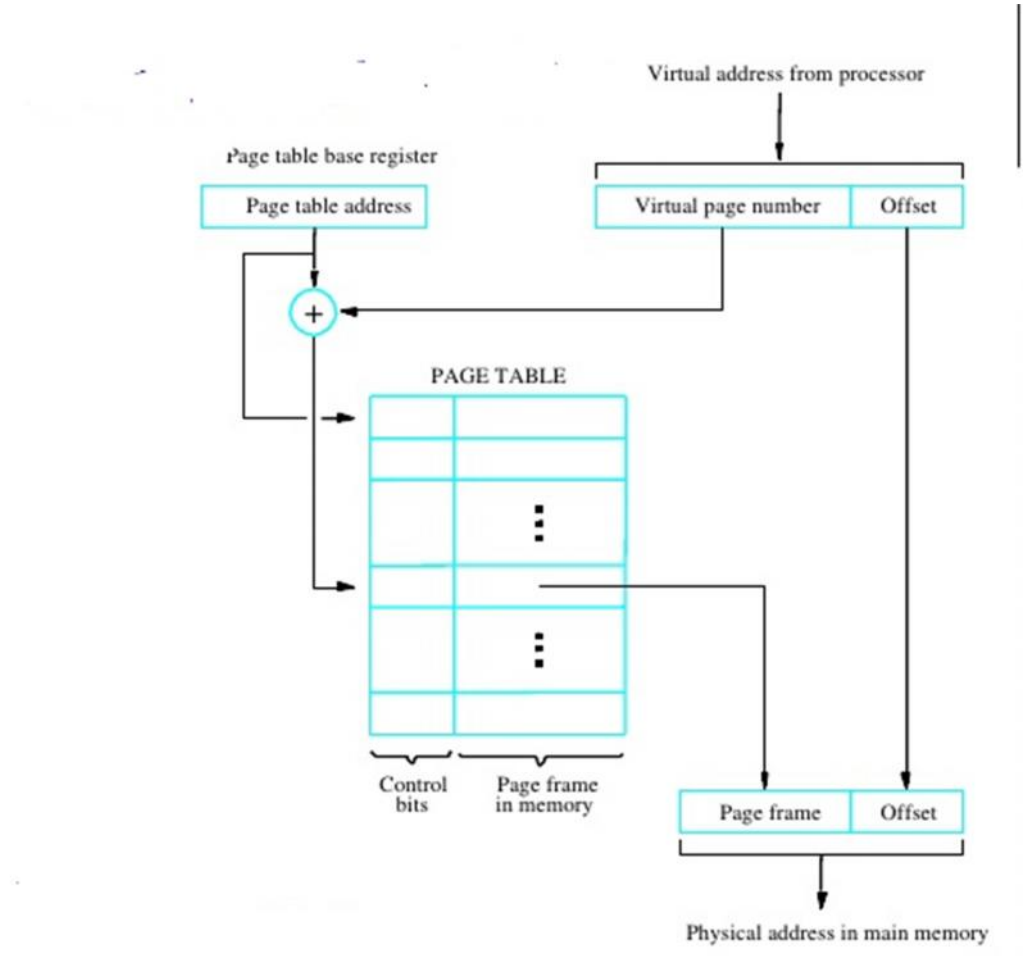


Fig:Virtual Memory Address Translation

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 **LookAside Buffer(TLB).** This portion consists of the page table enteries that corresponds to the most recently accessed pages and also contains the virtual address of the entry.

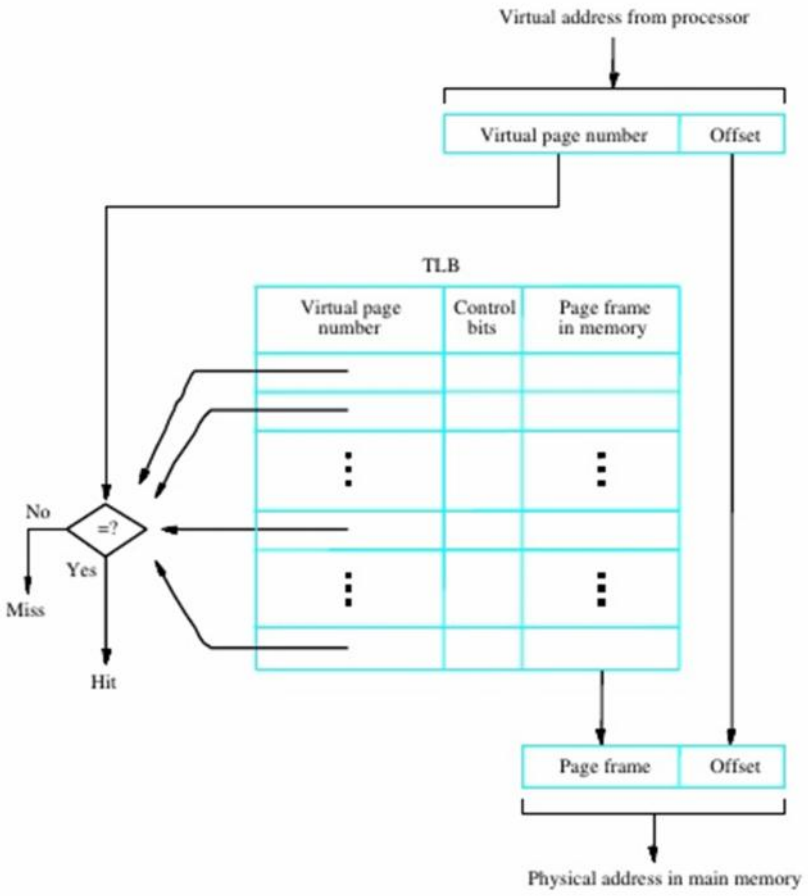


Fig:Use of Associative Mapped TLB

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 broght from disk into memry 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.

3.10 MEMORY MANAGEMENT REQUIREMENTS:

Management routines are part of the Operating system.Assembling the OS routine into virtual address space is called „**System Space**". The virtual space in which the user application program 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.One program may be allowed to read/write a given page,while the other programs may be given only red access.

3.11SECONDARY STORAGE:

The Semi-conductor memories donot 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 disk 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 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 disk 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.

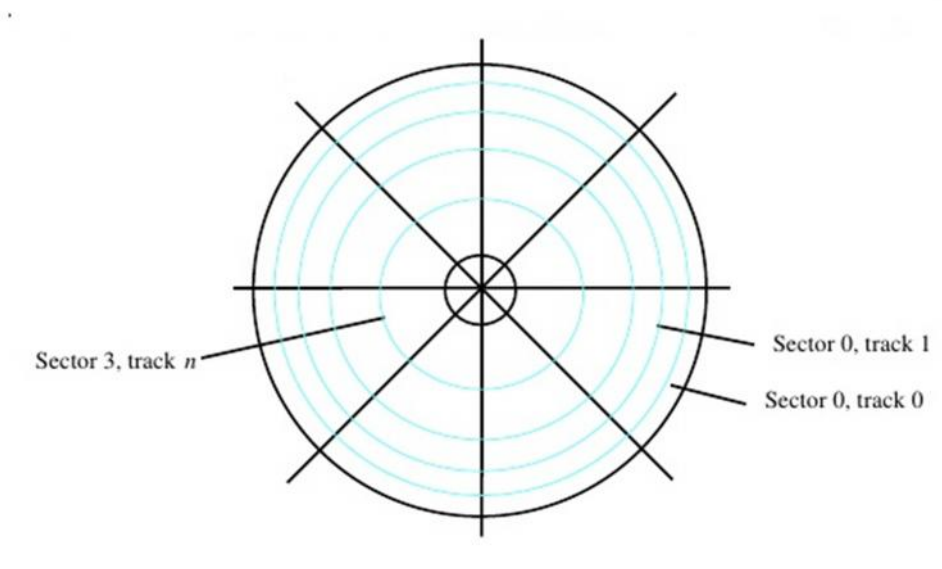
The disk system has 3 parts.They are,

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 eachtrack.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 matchbookCapacity -> 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