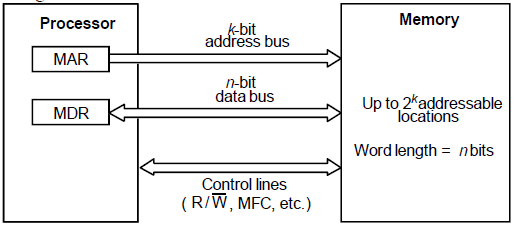
**Basic concepts of memory system**

**Introduction:** Computer should have a large memory to facilitate execution of programs that are large and deal with huge amounts of data. **The memory should be fast, large, and inexpensive. Unfortunately, it is impossible to meet all three of these requirements simultaneously. Increased speed and size are achieved at increased cost.** To solve this problem, much work has gone into developing clever structures that improve the apparent speed and size of the memory, yet keep the cost reasonable.

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 up to 216 = 64K (65536) memory locations. Similarly, machines whose instructions generate 32-bit addresses can utilize a memory that contains up to 232 = 4G (giga) memory locations, whereas machines with 4O-bit addresses can access up to 240 = 1 T (tera) locations. The number of locations represents the size of the address space of the computer.

From the system standpoint, we can view the memory unit as a black box. Data transfer between the memory and the processor takes place through the use of two processor 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 memory unit may contain up to 2 k addressable locations. During a memory cycle, n bits of data are transferred between the memory and the processor. This transfer takes place over the processor bus, which has k address lines and n data lines. The bus also includes the con1rollines Read / Write (R / W ) and Memory Function Completed (MFC) for coordinating data transfers. Other control lines may be added to indicate the number of bytes to be transferred. The connection between the processor and the memory is shown schematically in Figure 4.1.



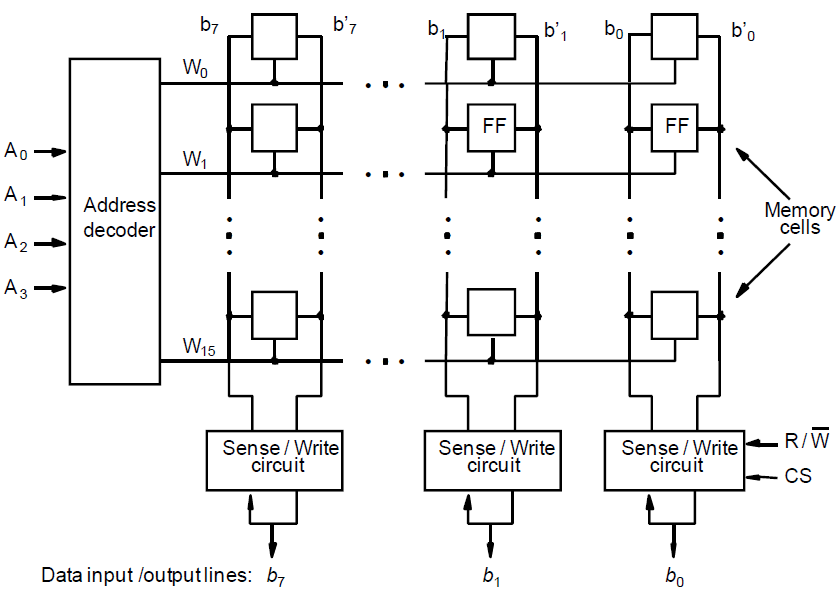
**Figure 4.1 Connection of the memory to the processor**

* The processor reads data from the memory by loading the address of the required memory location into the MAR register 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 the MFC signal. Upon receipt of the MFC signal, the processor loads the data on the data lines into the MDR register. The processor writes data into a memory location by loading the address of this location into MAR and loading the data into MDR. It indicates that a write operation is involved by setting the R/ W line to 0. If read or write operations involve consecutive address locations in the main memory, then a "block transfer" operation can be performed in which the only address sent to the memory is the one that identifies the first location.
* The time between the Read and the MFC signals is referred to as the memory access time. The memory cycle time is the minimum time delay required between the initiations of two successive memory operations, If any location can be accessed for a Read or Write operation some fixed amount of time that is independent of the location's address in a memory unit is called random-access memory (RAM). One way to reduce the memory access time is to use a cache memory. This 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 is used to increase the apparent size of the physical memory. Data are addressed in a virtual address space that can be as large as the addressing capability of the processor. But at any given time, only the active portion of this space is mapped onto locations in the physical memory. The remaining virtual addresses are mapped onto the bulk: storage devices used, which are usually magnetic disks. 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, often called the memory management unit.

**Semiconductor RAM Memories**

**Discuss Semiconductor RAM Memories:**Semiconductor memories are available in a wide range of speeds. Their cycle times range from l00ns to less than 10 ns.

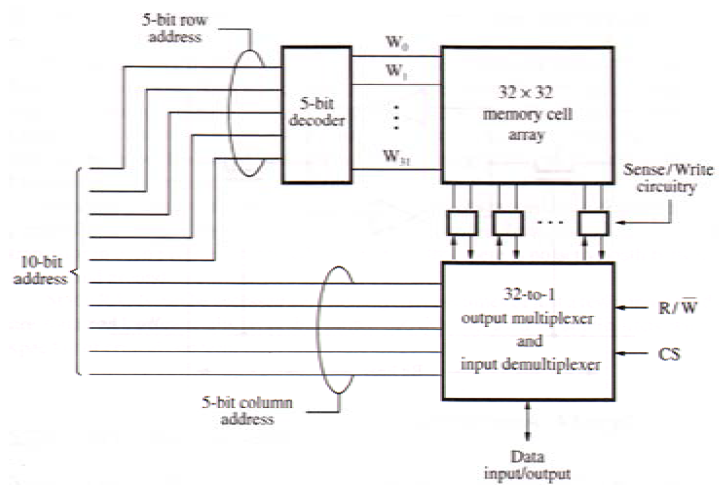
**INTERNAL ORGANIZATION OF MEMORY CHIPS**: Memory cells are usually organized in the form of an array, in which each cell is capable of storing one bit of information. A possible organization is illustrated in Figure 4.2.



**Figure 4.2 Organization of bit cell in a memory chip (16(words)X8(bits))**

* Each row of cells constitutes a memory word, and all cells of a row are connected to a common line referred to as the word line, which is driven by the address decoder on the chip.
* The cells in each column are connected to a Sense/Write circuit by two bit lines. The Sense/Write circuits are connected to the data input/output lines of the chip.
* During a Read operation, these circuits sense, or read, the information stored in the cells selected by a word line and transmit this information to the output data lines.
* During a Write operation, the Sense/Write circuits receive input information and store it in the cells of the selected word.

Figure 4.2 is an example of a very small memory chip consisting of 16 words of 8 bits each. This is referred to as a 16 x 8 organization. The data input and the data output of each Sense/Write circuit are connected to a single bidirectional data line that, can be connected to the data bus of a computer. Two control lines, R/W and CS, are provided in addition to address and data lines. The R/W (Read/ Write ) input specifies the required operation, and the CS (Chip Select) input selects a given chip in a multichip memory system.



**Fig 4.3 Organization of a 1kX1 memory chip**

* The memory circuit in Figure 4.3 stores 128 bits and requires 14 external connections for address, data, and control lines. Of course, it also needs two lines for power supply and ground connections. Consider now a slightly larger memory circuit, one that has l K (1024) memory cells.
* This circuit can be organized as a 128 x 8 memory, requiring a total of 19 external connections. Alternatively, the same number of cells can be organized into a l K x 1 format. In this case, a 100bit address is needed, but there is only one data line, resulting in 15 external connections. Figure 4.3 shows such an organization.
* The required 100bit address is divided into two groups of 5 bits each to form the row and column addresses for the cell array. A row address selects a row of 32 cells, all of which are accessed in parallel. However, according to the column address, only one of these cells is connected to the external data line by the output multiplexer and input de multiplexer. For an example, a 4M-bit chip may have a 512K x 8 organization, in which case 19 address and 8 data input/output pins are needed.

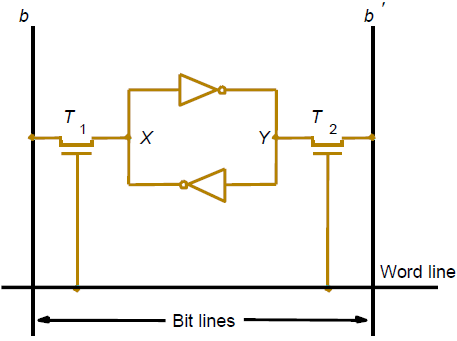
**Static Memories**

**STATIC MEMORIES**: Static memories are the Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memories.

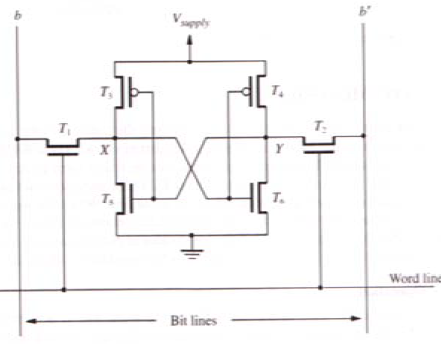
* Figure 4.4 illustrates how a static RAM (SRAM) cell may be implemented. Two inverters are cross-connected to form a latch. The latch is connected to two bit lines by transistors Tl and T2.
* These transistors act as switches that can be opened or closed under control of the word line. When the word line is at ground level, the transistors are turned off and the latch retains its state.
* For example, let us assume that the cell is in state 1 if the logic value at point X is 1 and at point Y is 0. This state is maintained as long as the signal on the word line is at ground level.

**Read operation**: In order to read the state of the SRAM cell, the word line is activated to close switches Tl and T2. If the cell is in state 1, the signal on bit line b is high and the signal on bit line b' is low. The opposite is true if the cell is in state 0. Thus, b and b' are complements of each other. Sense/Write circuits at the end of the bit lines monitor 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 signals on the bit lines are generated by the Sense/Write circuit. if Tl and T2 are turned on (closed), it lines b and b ' will have high and low signals, respectively.



**<--Figure 4.4 A static RAM cell**



**Figure 4.5 An example of a CMOS memory cell**-->

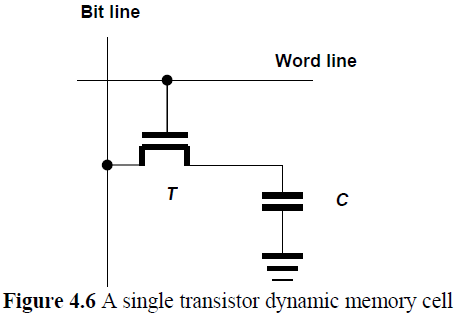
The power supply voltage, Vsupply, is 5 V in older CMOS SRAMs or 3.3 V in new low voltage versions. Note that "continuous power is needed for the cell to retain its state. If power is interrupted, the cell's contents will be lost. When power is restored, the latch will settle into a stable state, but it will not necessarily be the same state the cell was in before the interruption. Hence, SRAMs are said to be volatile memories because their contents are lost when power is interrupted.

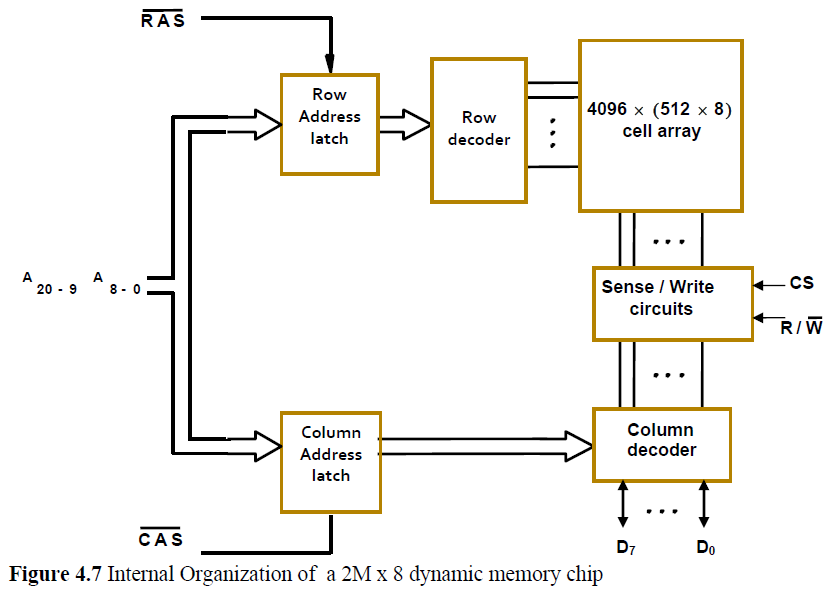
A major advantage of CMOS SRAMs is their very low power consumption because current flows in the cell only when the cell is being accessed. Otherwise, Tl, T2 and one transistor in each inverter are turned off, ensuring that there is no active path between V supply and ground. Static RAMs can be accessed very quickly. Access times of just a few nanoseconds are found in commercially available chips. SRAMs are used in applications where speed is of critical concern.

**Asynchronous DRAMs**

**ASYNCHRONOUS DRAMS**: Static RAMs are fast, but they come at a high cost because their cells require several transistors. Less expensive RAMs can be implemented if simpler cells are used.

* However, such cells do not retain their state indefinitely; hence, they are called dynamic RAMs (DRAMs).
* Information is stored in a dynamic memory cell in the form of a charge on a capacitor, and this charge can be maintained for only tens of milliseconds. Since the cell is required to store information for a much longer time, its contents must be periodically refreshed by restoring the capacitor charge to its full value.
* An example of a dynamic memory cell that consists of a capacitor, C, and a transistor, T, is shown in Figure 4.6. In order to store information in this cell, transistor T is turned on and an appropriate voltage is applied to the bit line. This causes a known amount of charge to be stored in the capacitor.





* A 16-megabit DRAM chip, configured as 2M x 8, is shown in Figure 4.7. The cells are organized in the form of a 4 K x 4 K array.
* The 4096 cells in each row are divided into 512 groups of 8, so that a row can store 512 bytes of data
* Therefore, 12 address bits are needed to select a row. Another 9 bits are needed to specify a group of 8 bits in the selected row. Thus, a 21-bit address is needed to access a byte in this memory.
* The high-order 12 bits and the low-order 9 bits of the address constitute the row and column addresses of a byte, respectively.
* To reduce the number of pins needed for external connections, the row and column addresses are multiplexed on 12 pins.
* During a Read or a Write operation, the row address is applied first. It is loaded into the row address latch in response to a signal pulse on the Row Address Strobe (RAS) input of the chip.
* Then a Read operation is initiated, in which 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 the column address latch under control of the Column Address Strobe (CAS) signal.
* The information in this latch is decoded and the appropriate group of 8 Sense/Write circuits are selected.
* If the R/W control signal indicates a Read operation, the output values of the selected circuits are transferred to the data lines, D7-0.
* For a Write operation, the information on the D7-0 lines is transferred to the selected circuits. This information is then used to overwrite the contents of the selected cells in the corresponding 8 columns.
* Applying a row address causes all cells on the corresponding row to be read and refreshed during both Read and Write operations.
* To ensure that the contents of a DRAM are maintained, each row of cells must be accessed periodically. A refresh circuit usually performs 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 DRAMs.
* Because of their high density and low cost, DRAMs are widely used in the memory units of computers. Available chips range in size from 1M to 256M bits, and even larger chips are being developed.
* Figure 4.7. To provide flexibility in designing memory systems, these chips are manufactured in different organizations. For example, a 64-Mbit chip may be organized as 16M x 4, 8M x 8, or 4M x 16.

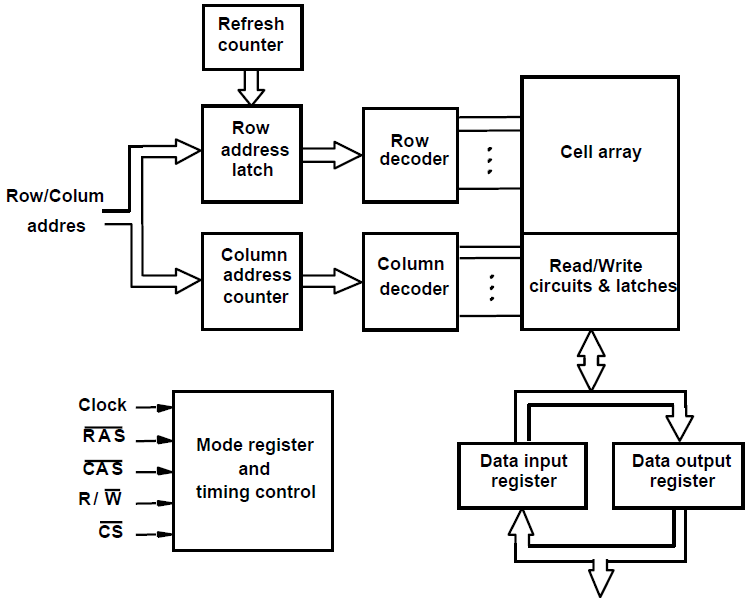
**Synchronous DRAMs**

**SYNCHRONOUS DRAMS**: DRAMs whose operation is directly synchronized with a clock signal are known as synchronous DRAMs (SDRAMs). Figure 4.8 indicates the structure of an SDRAM. The cell array is the same as in asynchronous DRAMs. The address and data connections are buffered by means of registers.

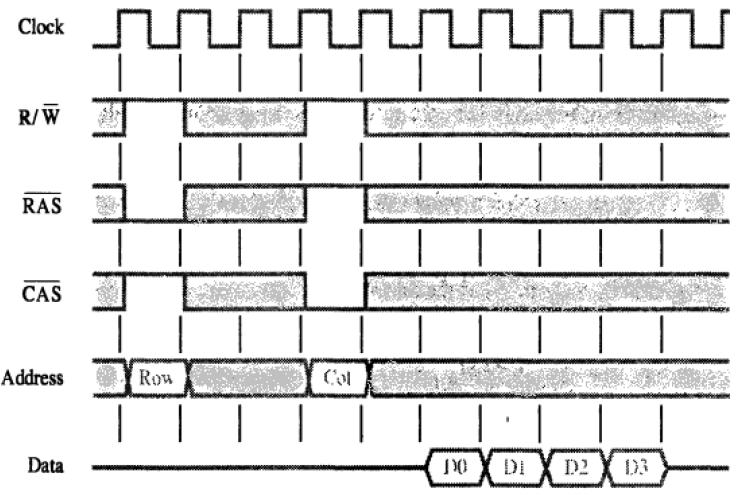
* We should particularly note that the output of each sense amplifier is connected to a latch.
* A Read operation causes the contents of all cells in the selected row to be loaded into these latches. But, if an access is made for refreshing purposes only, it will not change the contents of these latches
* SDRAMs have several different modes of operation, which can be selected by writing control information into a mode register.
* In SDRAMs, it is not necessary to provide externally generated pulses on the CAS line to select successive columns.

Figure 4.9 shows a timing diagram for a typical burst read of length 4. First, the row address is latched under control of the RAS signal.

* The memory typically takes 2 or 3 clock cycles (we use 2 in the figure) to activate the selected row. Then, the column address is latched under control of the 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 three sets of bits in the selected row, which are placed on the data lines in the next 3 clock cycles.
* SDRAMs have built-in refresh circuitry. A part of this circuitry is a refresh counter, which provides the addresses of the rows that are selected for refreshing.
* In a typical SDRAM, each row must be refreshed at least every 64 ms. Commercial SDRAMs can be used with clock speeds above 100 MHz.
* Transfers between the memory and the processor involve single words of data or small blocks of words. Large blocks, constituting a page of data, are transferred between the memory and the disks.
* The speed and efficiency of these transfers have a large impact on the performance of a computer system.



**<------Figure 4.8 Synchronous DRAM**



**Figure 4.9 Burst read of length 4 in an SDRAM**

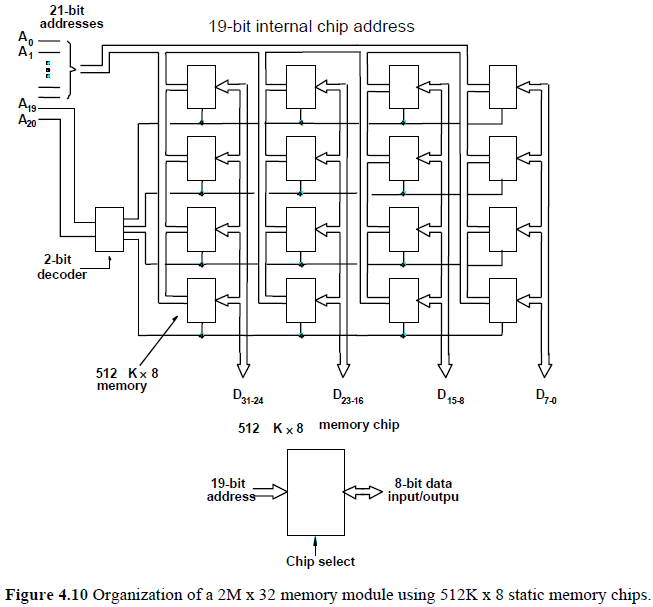
* A good indication of the performance is given by two parameters: latency and bandwidth.
* The term memory latency is used to refer to the amount of time it takes to transfer a word of data to or from the memory.
* In the case of reading or writing a single word of data, the latency provides a complete indication of memory performance.
* But, in the case of burst operations that transfer a block of data, the time needed to complete the operation depends also on the rate at which successive words can be transferred and on the size of the block. In block transfers, the term latency is used to denote the time it takes to transfer the first word of data.
* When transferring blocks of data, it is of interest to know how much time is needed to transfer an entire block. Since blocks can be variable in size, it is useful to define a performance measure in terms of the number of bits or bytes that can be transferred in one second. This measure is often referred to as the memory bandwidth.

**Structure of Larger Memories**

**STRUCTURE OF LARGER MEMORIES**: Many memory chips are connected to form a much larger memory.

**Static Memory System**: Consider a memory consisting of 2M (2,097,152) words of 32 bits each. Figure 4.10 shows how we can implement this memory using 512K x 8 static memory chips. Each column in the figure consists of four chips, which implement one byte position. Four of these sets provide the required 2M x 32 memory.

* Each chip has a control input called Chip Select. When this input is set to 1, it enables the chip to accept data from or to place data on its data lines. The data output for each chip is of the three-state type.
* Only the selected chip places data on the data output line, while all other outputs are in the high-impedance state. Twenty one address bits are needed to select a 32-bit word in this memory.
* The high-order 2 bits of the address are decoded to determine which of the four Chip Select control signals should be activated and the remaining 19 address bits are used to access specific byte locations inside each chip of the selected row. The R/W inputs of all chips are tied together to provide a common Rea d/Write control (not shown in the figure).



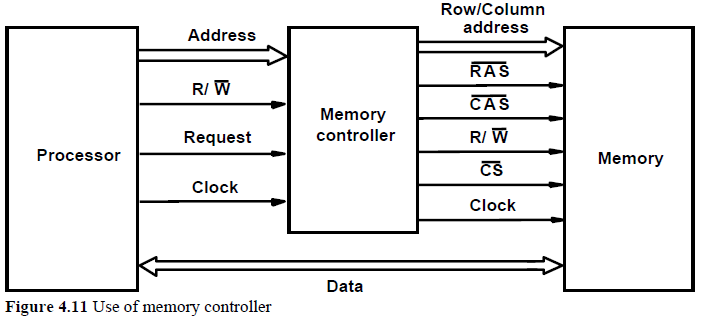
**Dynamic Memory System**: The organization of large dynamic memory systems is essentially the same as the memory shown in Figure 4.10. However, physical implementation is often done more conveniently in the form of memory modules. Modem computers use very large memories; even a small personal computer is likely to have at least 32M bytes of memory.

* Typical workstations have at least 128M bytes of memory. A large memory leads to better performance because more of the programs and data used in processing can be held in the memory, thus reducing the frequency of accessing the information in secondary storage. However, 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 a motherboard, it will occupy an unacceptably large amount of space on the board.
* Also, it is awkward to provide for future expansion of the memory, because space must be allocated and wiring provided for the maximum expected size. These packaging considerations have led to the development of larger memory units known as SIMMs (Single In-line Memory Modules) and DIMMs (Dual In-line Memory Modules).

**MEMORY SYSTEM CONSIDERATION**: There are several factors that determine the choice of a RAM chip for a given application. These factors are the cost, speed, power dissipation, and size of the chip. Static RAMs are generally used only when very fast operation is the primary requirement. Their cost and size are adversely affected by the complexity of the circuit that realizes the basic cell. They are used mostly in cache memories. Dynamic RAMs are the predominant choice for implementing computer main memories. The high densities achievable in these chips make large memories economically feasible.

**Memory' Controller**: To reduce the number of pins, the dynamic memory chips use multiplexed address inputs. The address is divided into two parts. The high-order address bits, which select a row in the cell array, are provided first and latched into the memory chip under control of the RAS signal. Then, the low-order address bits, which select a column, are provided on the same address pins and latched using the CAS signal.

A typical processor issues all bits of an address at the same time. The required multiplexing of address bits is usually performed by a memory controller circuit, which is interposed between the processor and the dynamic memory as shown in Figure 4.11. The controller accepts a complete address and the R/W signal from the processor, under 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 the RAS and CAS signals. Thus, the controller provides the RAS-CAS timing, in addition to its address multiplexing function. It also sends the R/W and CS signals to the memory. The CS signal is usually active low, hence it is shown as CS in Figure 4.11. Data lines are connected directly between the processor and the memory. Note that the clock signal is needed in SDRAM chips.



When used with DRAM chips, which do not have self-refreshing capability, the memory controller has to provide all the information needed to control the refreshing process. It contains a refresh counter that provides successive row addresses. Its function is to cause the refreshing of all rows to be done within the period specified for a particular device.

**Refresh overhead**: All dynamic memories have to be refreshed. In older DRAMs, a typical period for refreshing all rows was 16ms. In typical SDRAMs, a typical period is 64 ms.

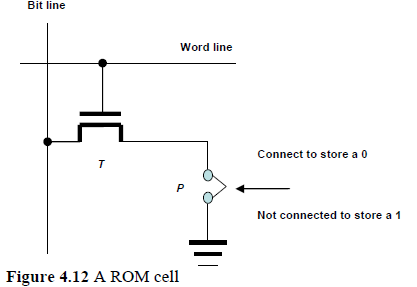
**Read Only Memories (ROM)**

**Read Only Memories (ROM)**: RAM memories are volatile memories, which mean that they lose the stored information if power is turned off.

* There are many applications that need memory devices which retain the stored retained even if power is turned off. A practical solution is to provide a small amount of non volatile memory that holds the instructions whose execution results in loading the boot program from the disk.
* The contents of such memory can be read as if they were SRAM or DRAM memories. But, a special writing process is needed to place the information into this memory. Since its normal operation involves only reading of stored data, a memory of this type is called read-only memory (ROM).

**ROM**: Figure 4.12 shows a possible configuration for a ROM cell. A logic value 0 is stored in the cell if the transistor is connected to ground at point P; otherwise, a 1 is stored.

* The bit line is connected through a resistor to the power supply. To read the state of the cell, the word line is activated. Thus, the transistor switch is closed and the voltage on the bit line drops to near zero if there is a connection between the transistor and ground.
* If there is no connection to ground, the bit line remains at the high voltage, indicating a 1. A sense circuit at the end of the bit line generates the proper output value. Data are written into a ROM when it is manufactured.



**PROM**: **Some ROM designs allow the data to be loaded by the user, thus providing a programmable ROM (PROM).**Programmability is achieved by inserting a fuse at point P in Figure 4.12.

* Before it is programmed, the memory contains all 0s. The user can insert 1s at the required locations by burning out the fuses at these locations using high-current pulses. Of course, this process is irreversible. PROMs provide flexibility and convenience not available with ROMs.
* The latter are economically attractive for storing fixed programs and data when high volumes of ROMs are produced. However, the cost of preparing the masks needed for storing a particular information pattern in ROMs makes them very expensive when only a small number are required. In this case, PROMs provide a faster and considerably less expensive approach because they can be programmed directly by the user.

**EPROM**: **Another type of ROM chip allows the stored data to be erased and new data to be loaded. Such an erasable, reprogrammable ROM is usually called an EPROM.**

* It provides considerable flexibility during the development phase of digital systems. Since EPROMs are capable of retaining stored information for a long time, they can be used in place of ROMs while software is being developed.
* In this way, memory changes and updates can be easily made. An EPROM cell has a structure similar to the ROM cell in Figure 4.12. In an EPROM cell, however, the connection to ground is always made at point 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. Thus, an EPROM cell can be used to construct a memory in the same way as the previously discussed ROM cell.
* The important advantage of EPROM chips is that their contents can be erased and reprogrammed. Erasure requires dissipating the charges trapped in the transistors of memory cells; this can be done by exposing the chip to ultraviolet light. For this reason, EPROM chips are mounted in packages that have transparent windows.

**EEPROM**: **A significant disadvantage of EPROMs is that a chip must be physically removed from the circuit for reprogramming and that its entire contents are erased by the ultraviolet light.**

* **It is possible to implement another version of erasable PROMs that can be both programmed and erased electrically. Such chips, called EEPROMs,** do not have to be removed for erasure.
* Moreover, it is possible to erase the cell contents selectively. The only disadvantage of EEPROMs is that different voltages are needed for erasing, writing, and reading the stored data.

**Flash Memories**

**FLASH MEMORIES**: An approach similar to EEPROM technology has more recently given rise to flash memory devices. A flash cell is based on a single transistor controlled by trapped charge, just like an EEPROM cell. While similar in some respects, there are also substantial differences between flash and EEPROM devices. In EEPROM it is possible to read and write the contents of a single cell. In a flash device it is possible to read the contents of a single cell, but it is only possible to write an entire block of cells. Prior to writing, the previous contents of the block are erased. Flash devices have greater density, which leads to higher capacity and a lower cost per bit. They require a single power supply voltage, and consume less power in their operation. The low power consumption of flash memory makes it attractive for use in portable equipment that is battery driven. 1Ypical applications include hand-held computers, cell phones, digital cameras, and MP3 music players.

Single flash chips do not provide sufficient storage capacity for the applications mentioned above. Larger memory modules consisting of a number of chips are needed. There are two popular choices for the implementation of such modules: flash cards and flash drives.

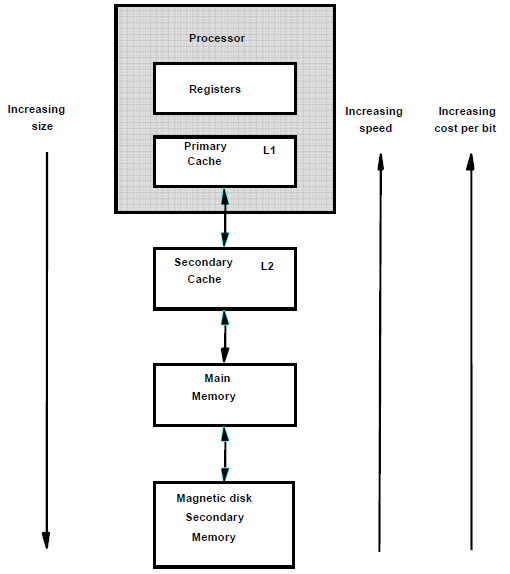
**FlashCards:-**  
One way of constructing a larger module is to mount flash chips on a small card. Such flash cards have a standard interface that makes them usable in a variety of products. A card is simply plugged into a conveniently accessible slot. Flash cards come in a variety of memory sizes. Typical sizes are 8, 32, 64 MB and so on.

**Flashdrive:-**  
Larger flash memory modules have been developed to replace, hard disk drives. These flash drives are designed to fully emulate the bard disks, to the point that they can be fitted into standard disk drive bays. The fact that flash drives are solid state electronic devices that have no movable parts provides some important advantages. They have shorter seek and access times, which results in faster response. They have lower power consumption, which makes them attractive for battery driven applications, and they are also insensitive to vibration. Another type of ROM chip allows the stored data to be erased and new data to be loaded. The disadvantages of flash drives are less storage capacity, higher cost per bit and it will become weak after it has been written several times.

**Speed, Size and Cost of Memories**

**Introduction:**An ideal memory would be fast, large, and inexpensive. a very fast memory can be implemented if SRAM chips are used. But these chips are expensive because their basic cells have six transistors, which preclude packing a very large number of cells onto a single chip. Thus, for cost reasons, it is impractical to build a large memory using SRAM chips. The alternative is to use Dynamic RAM chips, which have much simpler basic cells and thus are much less expensive. But such memories are significantly slower.

Although dynamic memory units in the range of hundreds of megabytes can be implemented at a reasonable cost, the affordable size is still small compared to the demands of large programs with voluminous data. A solution is provided by using secondary storage, mainly magnetic disks, to implement large memory spaces. Very large disks are available at a reasonable price, and they are used extensively in computer systems. However, they are much slower than the semiconductor memory units. So A huge amount of cost-effective storage can be provided by magnetic disks. A large, yet affordable, main memory can be built with dynamic RAM technology. This leaves SRAMs to be used in smaller units where speed is of the essence, such as in cache memories.



**Figure 4.13 Memory hierarchy:-**  
All of these different types of memory units are employed effectively in a computer. The entire computer memory can be viewed as the hierarchy depicted in Figure 4.13. The fastest access is to data held in processor registers. Therefore, if we consider the registers to be part of the memory hierarchy, then the processor registers are at the top in terms of the speed of access. Of course, the registers provide only a minuscule portion of the required memory.

At the next level of the hierarchy is a relatively small amount of memory that can be  
implemented directly on the processor chip. This memory, called a processor cache, holds copies of instructions and data stored in a much larger memory that is provided externally. There are often two levels of caches. A primary cache is always located on the processor chip. This cache is small because it competes for space on the processor chip, which must implement many other functions.

The primary cache is referred to as level (L1) cache. A larger, secondary cache is placed between the primary cache and the rest of the memory. It is referred to as level 2 (L2) cache. It is usually implemented using SRAM chips. It is possible to have both Ll and L2 caches on the processor chip.  
The next level in the hierarchy is called the main memory. This rather large memory is  
implemented using dynamic memory components, typically in the form of SIMMs, DIMMs, or RIMMs. The main memory is much larger but significantly slower than the cache memory. In a typical computer, the access time for the main memory is about ten times longer than the access time for the L 1 cache.

Disk devices provide a huge amount of inexpensive storage. They are very slow compared to the semiconductor devices used to implement the main memory. A hard disk drive (HDD; also hard drive, hard disk, magnetic disk or disk drive) is a device for storing and retrieving digital information, primarily computer data. It consists of one or more rigid (hence "hard") rapidly rotating discs (often referred to as platters), coated with magnetic material and with magnetic heads arranged to write data to the surfaces and read it from them. During program execution, the speed of memory access is of utmost importance. The key to managing the operation of the hierarchical memory system in Figure 4.13 is to bring the instructions and data that will be used in the near future as close to the processor as possible. This can be done by using the hardware mechanisms.

**Cache memories**

**Cache memories**: **In most of the computer system, the speed of the main memory is very low than the speed of modern processors**. **For good performance, it is important to devise a scheme that reduces the time needed to access the necessary information.**

* Since the speed of the main memory unit is limited by electronic and packaging constraints, the solution must be sought in a different architectural arrangement an efficient solution is to use a fast cache memory which essentially makes the main memory appear to the processor to be faster.
* The effectiveness of the cache mechanism is based on a property of computer programs called **locality of reference.**
* Analysis of programs shows that most of their execution time is spent on routines in which many instructions are executed repeatedly. These instructions may constitute a simple loop, nested loops, or a few procedures that repeatedly call each other.
* Many instructions in localized areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently. This is referred to as **locality of reference**.
* It manifests itself in two ways: temporal and spatial. The temporal means that a recently executed instruction is likely to be executed again very soon.
* The spatial aspect means that instructions in close proximity to a recently executed instruction (with respect to the instructions' addresses) are also likely to be executed soon.
* Block refers to a set of contiguous address locations of some size. Another term that is often used to refer to a cache block is cache line.

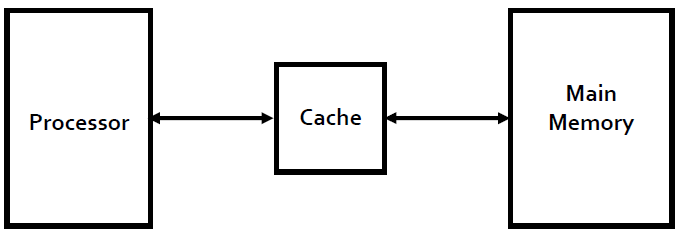
Consider the simple arrangement in Figure 4.14. When a Read request is received from the processor, the contents of a block of memory words containing the location specified are transferred into the cache one word at a time.

Subsequently, when the program references any of the locations in this block, the **desired contents are read directly from the cache**.

Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.

The correspondence between the main memory blocks and those in the cache is specified by a mapping function.

When the cache is full and a memory word (instruction or data) 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 contains the referenced word. The collection of rules for making this decision constitutes the replacement algorithm.



**Figure 4.14 Use of cache memory**  
The processor does not need to know explicitly about the existence of the cache. It simply issues Read and Write requests using addresses that refer to locations in the memory.

The cache control circuitry determines whether the requested word currently exists in the cache. If it does, the Read or Write operation is performed on the appropriate cache location. In this case, a **read or write hit** is said to have occurred.

In a Read operation, the main memory is not involved.

For a Write operation, the system can proceed in two ways. In the **first technique**, called the **write through protocol**, the **cache location and the main memory location are updated simultaneously.**

The **second technique** is to **update only the cache location** and to mark it as updated with an associated flag bit, often called the **dirty or modified bit**.

The main memory location of the word is updated later, when the block containing this marked word is to be removed from the cache to make room for a new block. This technique is known as the **write- back, or copy-back, protocol**.

The write-through protocol is simpler, but it results in unnecessary Write operations in the main memory when a given cache word is updated several times during its cache residency.

Note that the write-back protocol may also result in unnecessary Write operations because when a cache block is written back to the memory all words of the block are written back, even if only a single word has been changed while the block was in the cache.

When the addressed word in a Read operation is not in the cache, a **read** **miss** occurs.

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

After the entire block is loaded into the cache, the particular word requested is forwarded to the processor. Alternatively, this word may be sent to the processor as soon as it is read from the main memory.

The latter approach, which is called **load-through, or early restart**, reduces the processor's waiting period somewhat, but at the expense of more complex circuitry.

During a Write operation, if the addressed word is not in the cache, a **write miss** occurs. Then, if the write-through protocol is used, the information is written directly into the main memory.

In the case of the write-back protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

**Mapping Functions**

**MAPPING FUNCTIONS**: **The process of keeping information of moved data blocks from main memory to cache memory is known as mapping.** For example:

Consider a cache consisting of 128 blocks of 16 words each, for a total of 2048 (2K) words, and assumes that the main memory is addressable by a 16-bit address.

The main memory has 64K words, which we will view as 4K (4096) blocks of 16 words each. These mappings can be done in three ways. **They are Direct mapping, Associative mapping and Set associative mapping.**

1. **Direct mapping**: The simplest way to determine cache locations in which to store memory blocks is the direct-mapping technique. In this technique, block j of the main memory maps onto block j modulo 128 of the cache, as depicted in Figure 4.15.

Thus, whenever one of the main memory blocks 0, 128, 256,… is loaded in the cache, it is stored in cache block 0.

Blocks 1, 129, 257, are stored in cache block 1, and so on.

Since more than one memory block is mapped onto a given cache block position, contention may arise for that position even when the cache is not full.

For example, instructions of a program may start in block 1 and continue in block 129, possibly after a branch. As this program is executed, both of these blocks must be transferred to the block-1 position in the cache. Contention is resolved by allowing the new block to overwrite the currently resident block. In this case, the replacement algorithm is trivial.

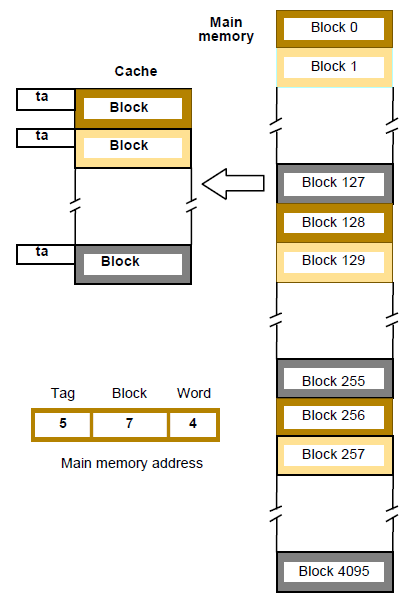
Placement of a block in the cache is determined from the memory address. The memory address can be divided into three fields, as shown in Figure 4.15. The low-order 4 bits select one of 16 words in a block.

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 high-order 5 bits of the memory address of the block are stored in 5 tag bits associated with its location in the cache. They identify which of the 32 blocks that are mapped into this cache position are currently resident in the cache.

As execution proceeds, the 7 -bit cache block field of each address generated by the processor points to a particular block location in the cache. The high-order 5 bits of the address are compared with the 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 first be read from the main memory and loaded into the cache. The direct-mapping technique is easy to implement, but it is not very flexible.



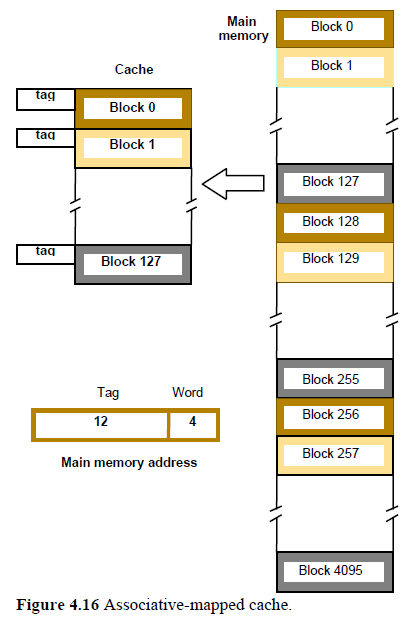
**Figure 4.15 Direct-mapped cache.**

1. **Associative mapping**: Figure 4.16 shows a much more flexible mapping method, in which a 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 the **associative-mapping technique**. It gives complete freedom in choosing the cache location in which to place the memory block.

Thus, the space in the cache can be used more efficiently. A new block that has to be brought into the cache has to replace (eject) an existing block only if the cache is full. In this case, we need an algorithm to select the block to be replaced.

Many replacement algorithms are possible; the cost of an associative cache is higher than the cost of a direct-mapped cache because of the need to search all 128 tag patterns to determine whether a given block is in the cache. A search of this kind is called an associative search. For performance reasons, the tags must be searched in parallel.



1. **Set-Associative mapping**

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 the main memory to reside in any block of a specific set. Hence, the contention problem of the direct method is eased by having a few choices for block placement. At the same time, the hardware cost is reduced by decreasing the size of the associative search. An example of this set-associative-mapping technique is shown in Figure 4.17 for a cache with two blocks per set. In this case, memory blocks 0, 64, 128, …, 4032 map into cache set 0, and they can occupy either of the two block positions within this set. Having 64 sets means that the 6-bit set field of the address determines which set of 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-way associative search is simple to implement.

The number of blocks per set is a parameter that can be selected to suit the requirements of a particular computer. For the main memory and cache sizes in Figure 4.17, four blocks per set can be accommodated by a 5-bit set field, eight blocks per set by a 4-bit set field, and so on. The extreme condition of 128 blocks per set requires no set bits and corresponds to the fully associative technique, with 12 tag bits. The other extreme of one block per set is the direct-mapping method. A cache that has k blocks per set is referred to as a k-way set-associative cache.

**Cache performance Improvement**

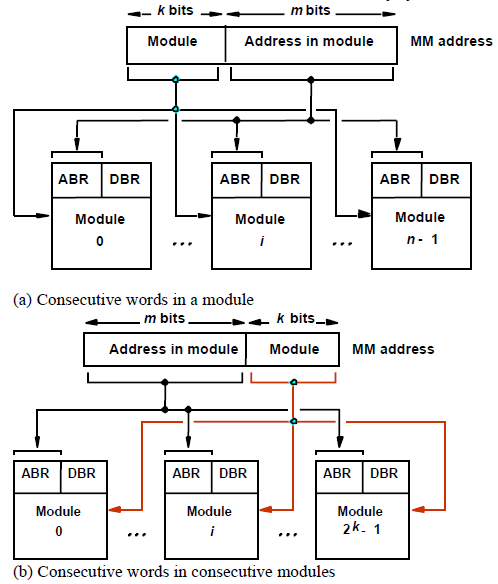
**improving cache performance**: Two key factors in the commercial success of a computer are **performance and cost;** the best possible performance at the lowest cost is the objective.

The challenge in considering design alternatives is to improve the performance without increasing the cost. A common measure of success is the price/performance ratio. Performance depends on how fast machine instructions can be brought into the processor for execution and how fast they can be executed.

The memory hierarchy is used for the best price/performance ratio. The main purpose of this hierarchy is to create a memory I that the processor sees as having a short access time and a large capacity. Each level of the hierarchy plays an important role.

The speed and efficiency of data transfer between various levels of the hierarchy are also of great significance. It is beneficial if transfers to and from the faster units can done at a rate equal to that of the faster unit. This is not possible if both the slow and the fast units are accessed in the same manner, but it can be achieved when parallelism is used in the organization of the slower unit. An effective way to introduce parallelism is to use an interleaved organization.

**INTERLEAVING**: If the main memory of a computer is structured as a collection of physically separate modules, each with 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.



**Figure 4.18 Addressing multiple-module memory systems.**

How individual addresses are distributed over the modules is critical in determining the average number of modules that can be kept busy as computations proceed.

Two methods of address layout are indicated in Figure 4.18. In the first case, the memory address generated by the processor is decoded as shown in Figure 4.18a. 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, as happens when a block of data is transferred to a cache, only one module is involved. At the same time, however, devices with direct memory access (DMA) ability may be accessing information in other memory modules.

The second and more effective way to address the modules is shown in Figure 4.18b. It 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 that module. In this way, consecutive addresses are located in successive modules. Thus, any component of the system that generates requests for access to consecutive memory locations can keep several modules busy at anyone time. This results in both faster accesses to a block of data and higher average utilization of the memory system as a whole. To implement the interleaved structure, there must be 2k modules; otherwise, there will be gaps of nonexistent locations in the memory address space.

**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 processor as a single memory unit that has the access time of a cache on the processor 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, well 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 processor 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.

**CACHES ON THE PROCESSOR CHIP**: From the speed point of view, the optimal place for a cache is on the processor chip. Unfortunately, space on the processor 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. 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.

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.

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 L 1 cache. A workstation computer may include an Ll 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  
tave = h1 C1 (1 - h1) h2 C2 ( 1 - h1 ) ( 1 - h2 ) M  
where h1 is the hit rate in the Ll cache.  
h2 is the hit rate in the L2 cache.  
C1 is the time to access information in the Ll cache.  
C2 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 - h1 ) ( 1 - h2 ), should be low. If both h1 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.

**Write Buffer**: When the write-through protocol is used, each write operation results in writing a new value into the main memory. If the processor must wait for the memory function to be completed, as we have assumed until now, then the processor is slowed down by all write requests. Yet the processor typically does not immediately depend on the result of a write operation, so it is not necessary for the processor 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 processor 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 processor usually cannot proceed without the data that are to be read from the memory. Hence, these requests are given priority over write requests.

**Write-back protocol**: 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 processor 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 cache mechanism, 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 processor has to pause until the new data arrive, which is the effect of the miss penalty.

To avoid stalling the processor, 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 processor 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 processor.

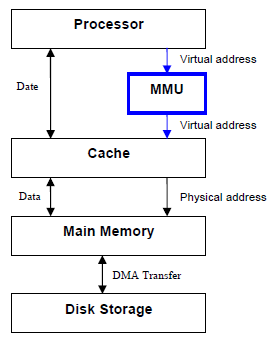
Prefetch instructions can be inserted into a program either by the programmer or by the compiler. The overall effect of software prefetching on performance is positive, and many processors have machine instructions to support this feature. Prefetching can also be done through hardware. This involves adding circuitry that attempts to discover a pattern in memory references and then prefetches data according to this pattern.

**Lockup-Free cache**: The software prefetching 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 prefetching 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. This problem can be solved by modifying the basic cache structure to allow the processor to access the cache while a miss is being serviced. 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. Due to the overlapping of 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.

**Virtual memory**

**Virtual memory**: Te physical main memory is not as large as the address space spanned by an address issued by the processor. When a program does not completely fit into the main memory, the parts of it not currently being executed are stored on secondary storage devices, such as magnetic disks. Of  
course, all parts of a program that are eventually executed are first brought into the main memory. When a new segment of a program is to be moved into a full memory, it must replace another segment already in the memory. The operating system moves programs and data automatically between the main memory and secondary storage. This process is known as swapping. Thus, the application programmer does not need to be aware of limitations imposed by the available main memory.

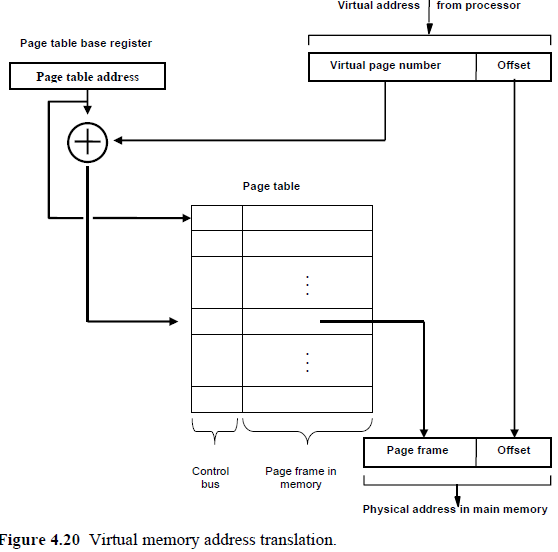
Techniques that automatically move program and data blocks into the physical main memory when they are required for execution are called virtual-memory techniques. Programs, and hence the processor, reference an instruction and data space that is independent of the available physical main memory space. The binary addresses that the processor issues for either instructions or data are called virtual or logical addresses. These addresses are translated into physical addresses by a combination of hardware and software components. If a virtual address refers to a part of the program or data space that is currently in the physical memory, then the contents of the appropriate location in the main memory are accessed immediately. On the other hand, if the referenced address is not in the main memory, its contents must be brought into a suitable location in the memory before they can be used.



**Figure 4.19 Virtual memory organization.**  
Figure 4.19 shows a typical organization that implements virtual memory. A special hardware unit, called the Memory Management Unit (MMU), translates virtual addresses into physical addresses. When the desired data (or instructions) are in the main memory, these data are fetched as described in our presentation of the ache mechanism. If the data are not in the main memory, the MMU causes the operating system to bring the data into the memory from the disk. The DMA scheme is used to perform the data Transfer between the disk and the main memory.

**Address Translation**

**ADDRESS TRANSLATION**: The process of translating a virtual address into physical address is known as address translation. It can be done with the help of MMU. A simple method for translating virtual addresses into physical addresses is to assume that all programs and data are composed of fixed-length units called pages, each of which consists of a block of words that occupy contiguous locations in the main memory. Pages commonly range from 2K to 16K bytes in length. They constitute the basic unit of information that is moved between the main memory and the disk whenever the translation mechanism determines that a move is required. Pages should not be too small, because the access time of a magnetic disk is much longer (several milliseconds) than the access time of the main memory. The reason for this is that it takes a considerable amount of time to locate the data on the disk, but once located, the data can be transferred at a rate of several megabytes per second. On the other hand, if pages are too large it is possible that a substantial portion of a page may not be used, yet this unnecessary data will occupy valuable space in the main memory.



The cache bridges the speed gap between the processor and the main memory and is implemented in hardware. The virtual-memory mechanism bridges the size and speed gaps between the main memory and secondary storage and is usually implemented in part by software techniques. Conceptually, cache techniques and virtual- memory techniques are very similar. They differ mainly in the details of their implementation.

A virtual-memory address translation method based on the concept of fixed-length pages is shown schematically in Figure 4.20. Each virtual address generated by the processor, whether it is for an instruction fetch or an operand fetch/store operation, is interpreted as a virtual page number (high-order bits) followed by an offset (low-order bits) that specifies the location of a particular byte (or word) within a page. Information about the main memory location of each page is kept in a page table. This information includes the main memory address where the page is stored and the current status of the page. An area in the main memory that can hold one page is called a page frame. The starting address of the page table is kept in a page table base register. By adding the virtual page number to the contents of this register, the address of the corresponding entry in the page table is obtained.  The contents of this location give the starting address of the page if that page currently resides in the main memory.

Each entry in the page table also includes some control bits that describe the status of the page while it is in the main memory. One bit indicates the validity of the page, that is, whether the page is actually loaded in the main memory. This bit allows the operating system to invalidate the page without actually removing it. Another bit indicates whether the page has been modified during its residency in the memory. As in cache memories, 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. Other control bits indicate various restrictions that may be imposed on accessing the page. For example, a program may be given full read and write permission, or it may be restricted to read accesses only.

**Translation Lookaside Buffer**

**TRANSLATION LOOKASIDE BUFFER**: The MMU must use the page table information for every read and write access; so ideally, the page table should be situated within the MMU. Unfortunately, the page table may be rather large, and since the MMU is normally implemented as part of the processor chip (along with the primary cache), it is impossible to include a complete page table on this chip. Therefore, the page table is kept in the main memory. However, a copy of a small portion of the page table can be accommodated within the MMU.

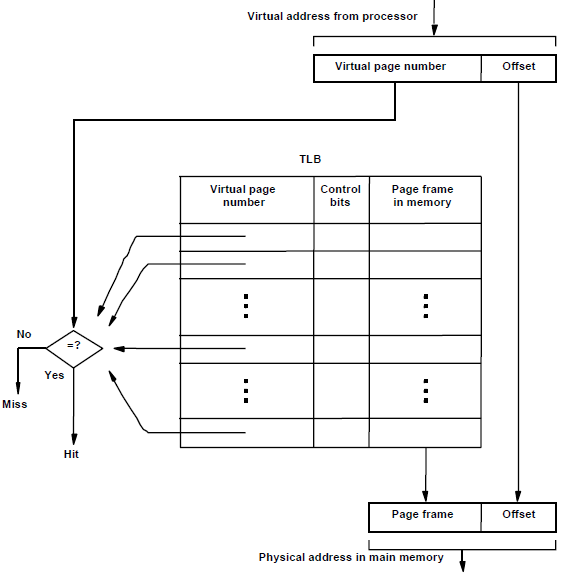
This portion consists of the page table entries that correspond to the most recently accessed pages. A small cache, usually called the Translation Lookaside Buffer (TLB) is incorporated into the MMU for this purpose. The operation of the TLB with respect to the page table in the main memory is essentially the  
same as the operation of cache memory; the TLB must also include the virtual address of the entry. Figure 4.21 shows a possible organization of a TLB where the associative-mapping technique is used. Setassociative mapped TLBs are also found in commercial products.

An essential requirement is that the contents of the TLB be coherent with the contents of page tables in the memory. When the operating system changes the contents of page tables, it must simultaneously invalidate the corresponding entries in the TLB. One of the control bits in the TLB is provided for this purpose. When an entry is invalidated, the TLB will acquire the new information as part of the MMU's normal response to access misses. Address translation proceeds as follows. Given a virtual address, the MMU looks in the TLB for the referenced page. IT the page table entry for this page is found in the TLB, the physical address is obtained immediately. If there is a miss in the TLB, then the required entry is obtained from the page table in the main memory and the TLB is updated.

When a program generates an access request to a page that is not in the main memory, a page fault is said to have occurred. The whole page must be brought from the disk into the memory before access can proceed.

When it detects a page fault, the MMU asks the operating system to intervene by raising an exception (interrupt). Processing of the active task is interrupted, and control is transferred to the operating system. The operating system then copies the requested page from the disk into the main memory and returns control to the interrupted task. Because a long delay occurs while the page transfer takes place, the operating system may suspend execution of the task that caused the page fault and begin execution of another task whose pages are in the main memory.

It is essential to ensure that the interrupted task can continue correctly when it resumes execution. A page fault occurs when some instruction accesses a memory operand that is not in the main memory, resulting in an interruption before the execution of this instruction is completed. Hence, when the task resumes, either the execution of the interrupted instruction must continue from the point of interruption, or the instruction must be restarted. The design of a particular processor dictates which of these options should be used.



**Figure 4.21 Use of an associative-mapped TLB.**

If a new page is brought from the disk when the main memory is full, it must replace one of the resident pages. The problem of choosing which page to remove is just as critical here as it is in a cache, and the idea that programs spend most of their time in a few localized areas also applies. Because main memories are considerably larger than cache memories, it should be possible to keep relatively larger portions of a program in the main memory. This will reduce the frequency of transfers to and from the disk. Concepts similar to the FIFO, Optimal and LRU replacement algorithms can be applied to page replacement and the control bits in the page table entries can indicate usage. One simple scheme is based on a control bit that is set to 1 whenever the corresponding page is referenced (accessed). The operating system occasionally clears this bit in all page table entries, thus providing a simple way of determining which pages have not been used recently.

A modified page has to be written back to the disk before it is removed from the main memory. It is important to note that the write-through protocol, which is useful in the framework of cache memories, is not suitable for virtual memory. The access time of the disk is so long that it does not make sense to access it frequently to write small amounts of data. The address translation process in the MMU requires some time to perform, mostly dependent on the time needed to look up entries in the TLB. Because of locality of reference, it is likely that many successive translations involve addresses on the same page. This is particularly evident in fetching instructions. Thus, we can reduce the average translation time by including one or more special registers that retain the virtual page number and the physical page frame of the most recently performed translations. The information in these registers can be accessed more quickly than the TLB.

**Memory management requirements**

**Memory management requirements:** Virtual memory fills the gap between the physical memory and secondary memory (disc), when one large program is being executed and it does not fit into the available physical memory, parts of it (pages) are moved from the disk into the main memory when they are to be executed. Some software routines are needed to manage this movement of program segments. Management routines are part of the operating system of the computer. It is convenient to assemble the operating system routines into a virtual address space, called the system space, that is separate from the virtual space in which user application programs reside. The latter space is called the user space.

In fact, there may be a number of user spaces, one for each user. This is arranged by providing a separate page table for each user program. The MMU uses a page table base register to determine the address of the table to be used in the translation process. Hence, by changing the contents of this register, the operating system can switch from one space to another. The physical main memory is thus shared by the active pages of the system space and several user spaces. However, only the pages that belong to one of these spaces are accessible at any given time.

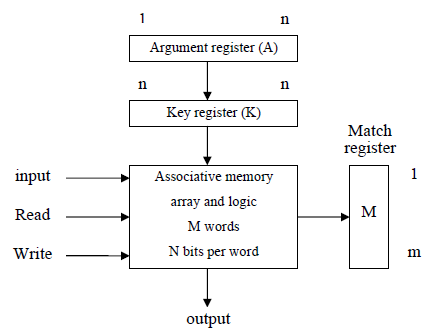
In any computer system in which independent user programs coexist in the main memory, the notion of protection must be addressed. No program should be allowed to destroy either the data or instructions of other programs in the memory. Such protection can be provided in several ways. Let us first consider the most basic form of protection. Recall that in the simplest case, the processor has two states, the supervisor state and the user state. As the names suggest, the  
processor is usually placed in the supervisor state when operating system routines are being executed and in the user state to execute user programs.

In the user state, some machine instructions (Access I/O devices, Poll for IO, perform DMA, catch hardware interrupt, Manipulate memory management, Set up page tables, load/flush the TLB and CPU aches, etc.) cannot be executed. These privileged instructions, which include such operations as modifying the page table base register, can only be executed while the processor is in the supervisor state. Hence, a user program is prevented from accessing the page tables of other user spaces or of the system space.

It is sometimes desirable for one application program to have access to certain pages belonging to another program. The operating system can arrange this by causing these pages to appear in both spaces. The shared pages will therefore have entries in two different page tables. The control bits in each table entry can be set to control the access privileges granted to each program. For example, one program may be allowed to read and write a given page, while the other program may be given only read access.

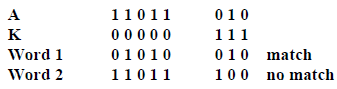
**Associative memory**

**Associative memory.**: Many data-processing applications require the search of items in a table stored in memory. They use object names or number to identify the location of the named or numbered object within a memory space. For example, an account number may be searched in a file to determine the holder's name and account status. To search an object, the number of accesses to memory depends on the location of the object and the efficiency of the search algorithm.  
The time required to find an object stored in memory can be reduced considerably if objects are selected based on their contents, not on their locations. A memory unit addressed by the content is called an associative memory or content addressable Memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location. In general, the instructions which are available in a special kind of memories like cache, ROM and Virtual memory are addressed by content and not by address location.

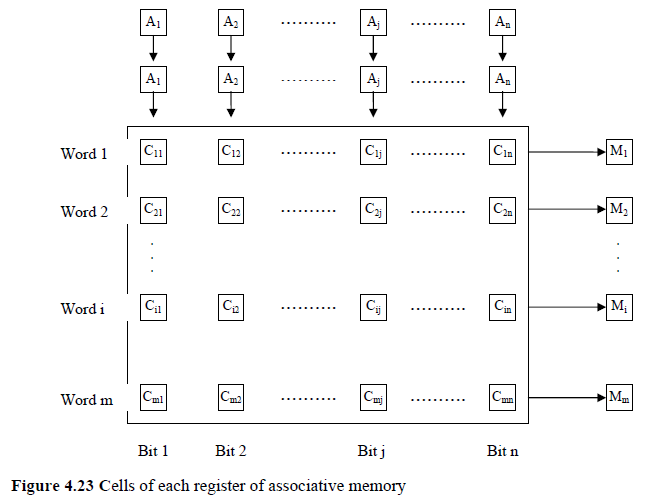


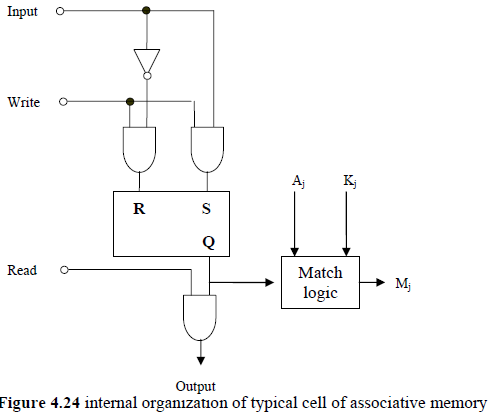
**Figure 4.22 Block diagram of associative memory**  
The Fig. 4.22 shows the block diagram of an associative memory it consists of memory array with match logic for m n-bit words and associated registers. The argument register (A) and key register (K) each have n-bits per word. Each word in memory is compared in parallel with the contents of the argument register. The words match with the word stored in the argument register set corresponding bits in the match register. Therefore, reading can be accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.

The key register provides a mask for choosing a particular field or bits in the argument word. Only those bits in the argument register having 1's in their corresponding position of the key register are compared. For example, if an argument register A and the key register K have the bit configuration shown below. Only the three rightmost bits of A are compared with memory words because K has 1's in these positions.

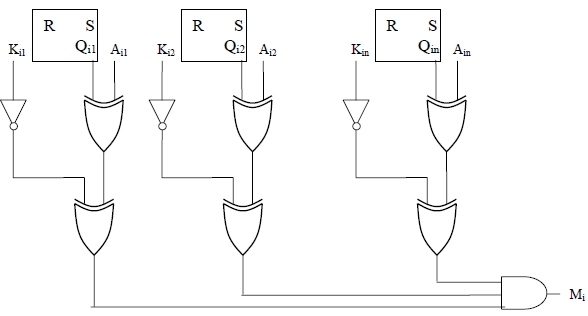


The Figure 4.23 shows the associated memory with cells of each register. The cells in the memory array are marked by the letter C with two subscripts. The first subscript gives the word number and the second subscript gives the bit position in the word





The Figure 4.24 shows the internal organization of a typical cell. It consists of RS flip-flop as a storage element and the circuit for reading, writing and matching the cell. By making the write signal logic 1, it is possible to transfer the input bit into the storage cell. To carry-out read operation read signal is made logic 1. The match logic compares the bit in the storage cell with the corresponding unmasked bit of the argument and provides an output for the decision logic that sets the corresponding bit the match register.

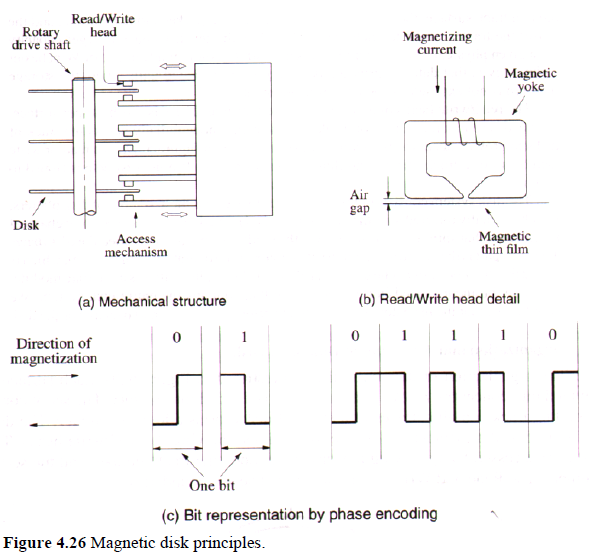


The Figure 4.25 shows the match logic for each word. The cell bit (Qij) is compared with the corresponding argument bit with Ex-NOR gate. Ex-NOR gate gives output logic1 only when its inputs are same. The output of Ex-NOR gate is valid only when the corresponding bit in key register is logic 1. This condition is implemented using 2-input OR gate. When corresponding bit in key register is logic 1, the inverter gives output 0 which forces the output of OR gate to follow the second input, i.e. the comparison output; otherwise output of OR-gate is logic 1. The outputs of all OR-gates are then ANDed with n-input AND gate to check whether all bits in the word are matched with the bits in the argument register.

**Secondary memory**

**Secondary memory**: Most computer systems are economically realized in the form of magnetic disks, optical disks, and magnetic tapes, which are usually referred to as secondary storage devices. Secondary storage memories are Non volatile, Low cost per bit and mass storage of information.

**MAGNETIC HARD DISKS**: The storage medium in a magnetic-disk system consists of one or more disks 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, as shown in Figure 4.26a. The disks rotate at a uniform speed. Each head consists of a magnetic yoke and a magnetizing coil, as indicated in Figure 4.26b.



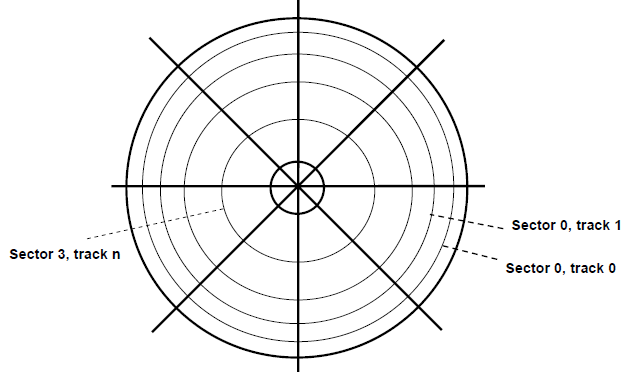
Digital information can be stored on the magnetic film by applying current pulses of suitable polarity to the magnetizing coil. This causes the magnetization of the film in the area immediately underneath the head to switch to a direction parallel to the applied field. The same head can be used for reading the stored information. In this case, changes in the magnetic field in the vicinity of the head caused by the movement of the film relative to the yoke induce a voltage in the coil, which now serves as a sense coil. The polarity of this voltage is monitored by the control circuitry to determine the state of magnetization of the film. Only changes in the magnetic field under the head can be sensed during the Read operation. Therefore, if the binary states 0 and 1 are represented by two opposite states of magnetization, a voltage is induced in the head only at 0-to-l and at I-to-O transitions in the bit stream. A long string of 0s or 1 s causes an induced voltage only at the beginning and end of the string. To determine the number of consecutive 0s or 1s stored, a  clock must provide information for synchronization. In some early designs, a clock was stored on a separate track, where a change in magnetization is forced for each bit period. Using the clock signal as a reference, the data stored on other tracks can be read correctly.

The modem approach is to combine the clocking information with the data. Several different techniques have been developed for such encoding. One simple scheme, depicted in Figure 4.26c, is known as phase encoding or Manchester encoding. In this scheme, changes in magnetization occur for each data bit, as shown in the figure. Note that a change in magnetization is guaranteed at the midpoint of each bit period, thus providing the clocking information. The drawback of Manchester encoding is its poor bit-storage density. The space required to represent each bit must be large enough to accommodate two changes in magnetization. We use the Manchester encoding example to illustrate how a self-clocking scheme may be implemented, because it is easy to understand. Other, more compact codes have been developed. They are much more efficient and provide better storage density. They also require more complex control circuitry.

Read/write heads must be maintained at a very small distance from the moving disk surfaces in order to achieve high bit densities and reliable read/write operations. When the disks are moving at their steady rate, air pressure develops between the disk surface and the head and forces the head away from the surface. This force can be counteracted by a spring -loaded mounting arrangement for the head that allows it to be pressed toward the surface. The flexible spring connection between the head and its arm mounting permits the head to fly at the desired distance away from the surface in spite of any small variations in the flatness of the surface. In most modem disk units, the disks and the read/write heads are placed in a sealed, airfiltered enclosure. This approach is known as Winchester technology. In such units, the read/write heads can operate closer to the magnetized track surfaces because dust particles, which are a problem in unsealed assemblies, are absent. The closer the heads are to a track surface, the more densely the data can be packed along the track, and the loser the tracks can be to each other. Thus, Winchester disks have a larger capacity for a given physical size compared to unsealed units. Another advantage of Winchester technology is that data integrity tends to be greater in sealed units where the storage medium is not exposed to contaminating elements.

The read/write heads of a disk system are movable. There is one head per surface. All heads are mounted on a comb-like arm that can move radially across the stack of disks to provide access to individual tracks, as shown in Figure 4.26a. To read or write data on a given track, the arm holding the read/write heads must first be positioned to that track. The disk system consists of three key parts. One part is the assembly of disk platters, which is usually referred to as the disk. The second part comprises the electromechanical mechanism that spins the disk and moves the read/write heads; it is called the disk drive. The third part is the electronic circuitry that controls the operation of the system, which is called the disk controller. The disk controller may be implemented as a separate module, or it may be incorporated into the enclosure that contains the entire disk system. The term disk is often used to refer to the combined package of the disk drive and the disk it contains.

The organization of data on a disk is illustrated in Figure 5.30. Each surface is divided into concentric tracks, and each track is divided into sectors. The set of corresponding tracks on all surfaces of a stack of disks forms a logical cylinder. The data on all tracks of a cylinder can be accessed without moving the read/write heads. The data are accessed by specifying the surface number, the track number, and the sector number. The Read and Write operations start at sector boundaries.



Data bits are stored serially on each track. Each sector usually contains 512 bytes of data, but other sizes may be used. The data are preceded by a sector header that contains identification (addressing) information used to find the desired sector on the selected track. Following the data, there are additional bits that constitute an error correcting code (ECC). The ECC bits are used to detect and correct errors that may have occurred in writing or reading of the 512 data bytes. To easily distinguish between two consecutive sectors, there is a small intersector gap.

An unformatted disk has no information on its tracks. The formatting process divides the disk physically into tracks and sectors. This process may discover some defective sectors or even whole tracks. The disk controller keeps a record of such defects and excludes them from use. The capacity of a formatted disk is a proper indicator of the storage capability of the given disk. The formatting information accounts for about 15 percent of the total information that can be stored on a disk. It comprises the sector headers, the ECC bits, and intersector gaps. In a typical computer, the disk is subsequently divided into logical partitions. There must be at least one such partition, called the primary partition. There may be a number of additional partitions.

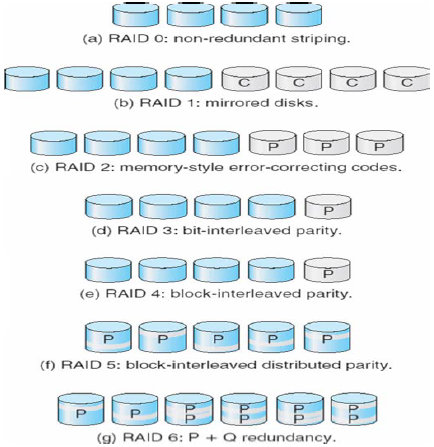
**Floppy Disks**

**FLOPPY DISKS**: The devices previously discussed are known as hard or rigid disk units. Floppy disks are smaller, simpler, and cheaper disk units that consist of a flexible, removable, plastic diskette coated with magnetic material. The diskette is enclosed in a plastic jacket, which has an opening where the read/write head makes contact with the diskette. A hole in the center of the diskette allows a spindle mechanism in the disk drive to position and rotate the diskette.

One of the simplest schemes used in the first floppy disks for recording data is phase or Manchester encoding mentioned earlier. Disks encoded in this way are said to have single density. A more complicated variant of this scheme, called double density, is most often used in current standard floppy disks. It increases the storage density by a factor of 2 but also requires more complex circuits in the disk controller.

The main feature of floppy disks is their low cost and shipping convenience. However, they have much smaller storage capacities, longer access times, and higher failure rates than hard disks. Current standard floppy disks are 3.25 inches in diameter and store 1.44 or 2 Mbytes of data. Larger super-floppy disks are also available. One type of such disks, known as the zip disk, can store more than 100 Mbytes. In recent years, the attraction of floppy-disk technology has been diminished by the emergence of rewritable compact disks.

**RAID DISK ARRAY:**In 1988, researchers at the University of California-Berkeley proposed a storage system based on multiple disks. They called it RAID, for Redundant Array of Inexpensive Disks. Using multiple disks also makes it possible to improve the reliability of the overall system. Six different configurations were proposed. They are known as RAID levels even though there is no hierarchy involved.



**Figure 4.29 RAID Levels**  
RAID 0 is the basic configuration intended to enhance performance. A single large file is stored in several separate disk units by breaking the file up into a number of smaller pieces and storing these pieces on different disks. This is called data striping. When the file is accessed for a read, all disks can deliver their data in parallel. RAID 1 is intended to provide better reliability by storing identical copies of data on two disks rather than just one. The two disks are said to be mirrors of each other. Then, if one disk drive fails, all read and write operations are directed to its mirror drive. This is a costly way to improve the reliability because all disks are duplicated. RAID 2, RAID 3, and RAID 4 levels achieve increased reliability through various parity checking schemes without requiring a full duplication of disks. All of the parity information is kept on one disk. RAID 5 also makes use of a parity-based error-recovery scheme. However, the parity information is distributed among all disks, rather than being stored on one disk.

A disk drive is connected to the rest of a computer system using some standard interconnection scheme. Normally, a standard bus such as the SCSI bus is used. A disk drive that incorporates the required SCSI interface circuitry is usually referred to as a SCSI drive. The SCSI bus is capable of transferring data at much higher rates than the rate at which data can be read from disk tracks. An efficient way to deal with the possible differences in transfer rates between the disk and the SCSI bus is to include a data buffer in the disk unit. This buffer is a semiconductor memory, capable of storing a few megabytes of data.

**Optical Disks**

**OPTICAL DISKS**: Large storage devices can also be implemented using optical means. The familiar compact disk (CD), used in audio systems, was the first practical application of this technology. Soon after, the optical technology was adapted to the computer environment to provide high-capacity readonly storage referred to as CD-ROM. The first generation of CDs was developed in the mid-1980s by the Sony and Philips companies, which also published a complete specification for these devices.

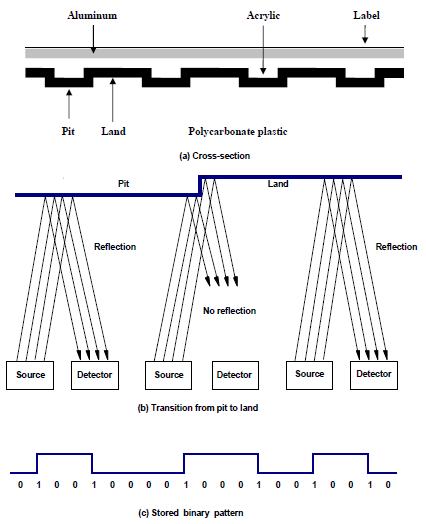
**CD Technology**: The optical technology that is used for CD systems is based on a laser light source. A laser beam is directed onto the surface of the spinning disk. Physical indentations in the surface are arranged along the tracks of the disk. They reflect the focused beam toward a photo detector, which detects the stored binary patterns.

The laser emits a coherent light beam that is sharply focused on the surface of the disk. Coherent light consists of synchronized waves that have the same wavelength. If a coherent light beam is combined with another beam of the same kind, and the two beams are in phase, then the result will be a brighter beam. But, if the waves of the two beams are 180 degrees out of phase, they will cancel each other. Thus, if a photo detector is used to detect the beams, it will detect a bright spot in the first case and a dark spot in the second case.

A cross-section of a small portion of a CD is shown in Figure 4.30a. The bottom layer is polycarbonate plastic, which functions as a clear glass base. The surface of this plastic is programmed to store data by indenting it with pits. The unintended parts are called lands. A thin layer of reflecting aluminum material is placed on top of a programmed disk. The aluminum is then covered by a protective acrylic. Finally, the topmost layer is deposited and stamped with a label. The total thickness of the disk is 1.2mm. Almost all of it is contributed by the polycarbonate plastic. The other layers are very thin. The laser source and the photodetector are positioned below the polycarbonate plastic. The emitted beam travels through this plastic, reflects off the aluminum layer, and travels back toward the photo detector. Note that from the laser side, the pits actually appear as bumps with respect to the lands.

Figure 4.30b shows what happens as the laser beam scans across the disk and encounters a transition from a pit to a land. Three different positions of the laser source and the detector are shown, as would occur when the disk is rotating. When the light reflects solely from the pit, or solely from the land, the detector will see the reflected beam as a bright spot. But, a different situation arises when the beam moves through the edge where the pit changes to the land, and vice versa. The pit is recessed one quarter of the wavelength of the light. Thus, the reflected wave from the pit will be 180 degrees out of phase with the wave reflected from the land, canceling each other. Hence, at the pit-land and land-pit transitions the detector will not see a reflected beam and will detect a dark spot.

Figure 4.30c depicts several transitions between lands and pits. If each transition, detected as a dark spot, is taken to denote the binary value 1, and the flat portions represent Os, then the detected binary pattern will be as shown in the figure. This pattern is not a direct representation of the stored data. CDs use a complex encoding scheme to represent data. Each byte of data is represented by a 14-bit code, which provides considerable error detection capability. We will not delve into details of this code.



**Figure 4.30 Optical Disk**  
**CD ROM**: Since information is stored in binary form in CDs, they are suitable for use as a storage medium in computer systems. The biggest problem is to ensure the integrity of stored data. Because the pits are very small, it is difficult to implement all of the pits perfectly. In audio and video applications, some errors in data can be tolerated because they are unlikely to affect the reproduced sound or image in a perceptible way. However, such errors are not acceptable in computer applications. Since physical imperfections cannot be avoided, it is necessary to use additional bits to provide error checking and correcting capability. CDs used in computer applications have such capability. They are called CD-ROMs, because after manufacture their contents can only be read, as with semiconductor ROM chips.

Stored data are organized on CD-ROM tracks in the form of blocks that are called sectors. There are several different formats for a sector. One format, known as Mode 1, uses 2352-byte sectors. There is a 16-byte header that contains a synchronization field used to detect the beginning of the sector and addressing information used to identify the sector. This is followed by 2048 bytes of stored data. At the end of the sector, there are 288 bytes used to implement the error-correcting scheme. The number of sectors per track is variable; there are more sectors on the longer outer tracks. Error detection and correction is done at more than one level. As mentioned in the introduction to CDs, each byte of stored information is encoded using a 14-bit code that has some error-correcting capability. This code can correct single-bit errors. Errors that occur in short bursts, affecting several bits, are detected and corrected using the error-checking bits at the end of the sector. CD- ROM drives operate at a number of different rotational speeds. The basic speed, known as IX, is 75 sectors per second. This provides a data rate of 153,600 bytes/s (150 Kbytes/s), using the Mode 1 format. With this speed and format, a CD-ROM based on the standard CD designed for 75 minutes of music has a data storage capacity of about 650 Mbytes.

**CD-ReWritables**: The most flexible CDs are those that can be written multiple times by the user. They are known as CD-RWs (CD-ReWritables). The basic structure of CD- RW s is similar to the structure of CD-R s. Instead of using an organic dye in the recording layer, an alloy of silver, indium, antimony and tellurium is used. This alloy has interesting and useful behavior when it is heated and cooled. If it is heated above its melting point (500 degrees C) and then cooled down, it goes into an amorphous state in which. It absorbs light. But, if it is heated only to about 200 degrees C and this temperature is maintained for an extended period, a process known as annealing takes place, which leaves the alloy in a crystalline state that allows light to pass through. If the crystalline state represents land area, pits can be created by heating selected spots past the melting point. The stored data can be erased using the annealing process, which returns the alloy to a uniform crystalline state. A reflective material is placed above the recording layer to reflect the light when the disk is read. The CD-RW drive uses three different laser powers. The highest power is used to record the pits. The middle power is used to put the alloy into its crystalline state; it is referred to as the "erase power." The lowest power is used to read the stored information. There is a limit on how many times a CD-RW disk can be rewritten. Presently, this can be done up to 1000 times.