Unit 2 – Memory Organization 8 Hrs

Memory

- Computer memory refers to the hardware device that are used to store and access data or programs on a temporary or permanent basis.
- **Memory** is used for storing programs and data that are required to perform a specific task.
- For CPU to operate at its maximum speed, it required an uninterrupted and high speed access to these **memories** that contain programs and data.

Characteristics of Memory Systems

Location Internal (e.g. processor registers, cache, main memory) External (e.g. optical disks, magnetic disks, tapes) **Capacity** Number of words Number of bytes **Unit of Transfer** Word Block **Access Method** Sequential Direct Random Associative	**Performance** Access time Cycle time Transfer rate **Physical Type** Semiconductor Magnetic Optical Magneto-optical **Physical Characteristics** Volatile/nonvolatile Erasable/nonerasable **Organization** Memory modules

Characteristics of Memory Systems

1) Location

- Refers to whether memory is internal and external to the computer
- Internal memory is often equated with main memory
- Processor requires its own local memory, in the form of registers
- Cache is another form of internal memory
- External memory consists of peripheral storage devices that are accessible to the processor via I/O controllers.

2) Capacity

- For internal memory, this is typically expressed in terms of bytes (1 byte = 8 bits) or words.
- Common word lengths are 8, 16, and 32 bits.
- External memory capacity is typically expressed in terms of bytes.

Characteristics of Memory Systems

3) Unit of Transfer

- For internal memory, the unit of transfer is equal to the number of electrical lines into and out of the memory module.
- This may be equal to the word length, but is often larger, such as 64, 128, or 256 bits.

4) Method of accessing

- **Sequential access**
- **Direct access**
- **Random access**
- **Associative**

Characteristics of Memory Systems

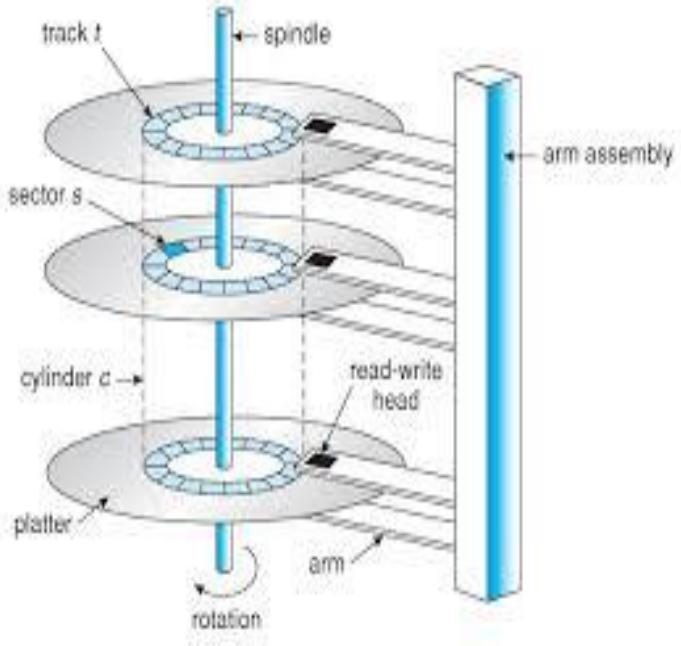
- **Sequential access**
  - Memory is organized into units of data, called records.
  - Access must be made in a specific linear sequence.
  - Time to access an arbitrary record is highly variable.
  - Tape units are sequential access.

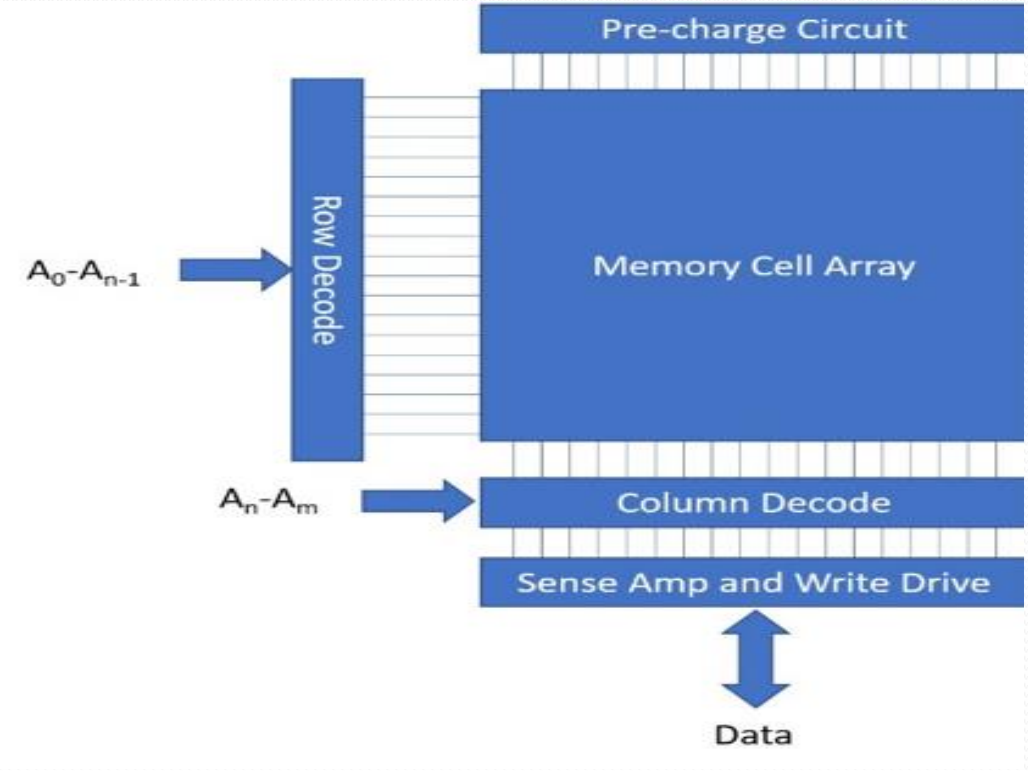
- **Direct Access**
  - As with sequential access, direct access involves a shared read–write mechanism.
  - However, individual blocks or records have a unique address based on physical location.
  - Access time is variable.
  - Disk units are direct access.

Characteristics of Memory Systems

- **Random access:**
  - Each addressable location in memory has a unique, physically wired-in addressing mechanism.
  - The time to access a given location is independent of the sequence of prior accesses and is constant.
  - Thus, any location can be selected at random and directly addressed and accessed.
  - Main memory and some cache systems are random access.
- **Associative:**
  - This is a random access type of memory.
  - A word is retrieved based on a portion of its contents rather than its address.
  - Cache memories may employ associative access.


D90
NOISE REDUCTION [ ] IN OUT
A
TYPE I
NORMAL BIAS 120 µS EQ


track t
spindle
arm assembly
sector s
cylinder c
read-write head
platter
arm
rotation


Pre-charge Circuit
Row Decode
Memory Cell Array
A_0-A_{n-1}
A_n-A_m
Column Decode
Sense Amp and Write Drive
Data

Characteristics of Memory Systems

5) Performance:

Three performance parameters are used:

a) **Access time (latency):**

- For random-access memory, this is the time it takes to perform a read or write operation.
- For non-random-access memory, access time is the time it takes to position the read–write mechanism at the desired location.

b) **Memory cycle time:**

- This concept is primarily applied to random-access memory and consists of the access time plus any additional time required before a second access can commence.
- It is the total time that is required to store next memory access operation from the previous memory access operation.

c) **Transfer rate:**

- This is the rate at which data can be transferred into or out of a memory unit.
- For random-access memory, it is equal to 1/(cycle time).

Characteristics of Memory Systems

6)Physical Types:

- A variety of **physical types** of memory have been employed.
- The most common today are
  a) semiconductor memory (RAM)
  b) magnetic surface memory, used for disk and tape, and
  c) optical (CD, DVD)
  d) magneto-optical (Re-Writable)

7) Physical characteristics:

- In a volatile memory, information is lost when electrical power is switched off.
- In a nonvolatile memory, information once recorded remains until intentionally changed; no electrical power is needed to retain information.
- Magnetic-surface memories are nonvolatile.
- Semiconductor memory (memory on integrated circuits) may be either volatile or nonvolatile.
- Nonerasable memory cannot be altered, except by destroying the storage unit.
- Semiconductor memory of this type is known as read- only memory (ROM)

Characteristics of Memory Systems

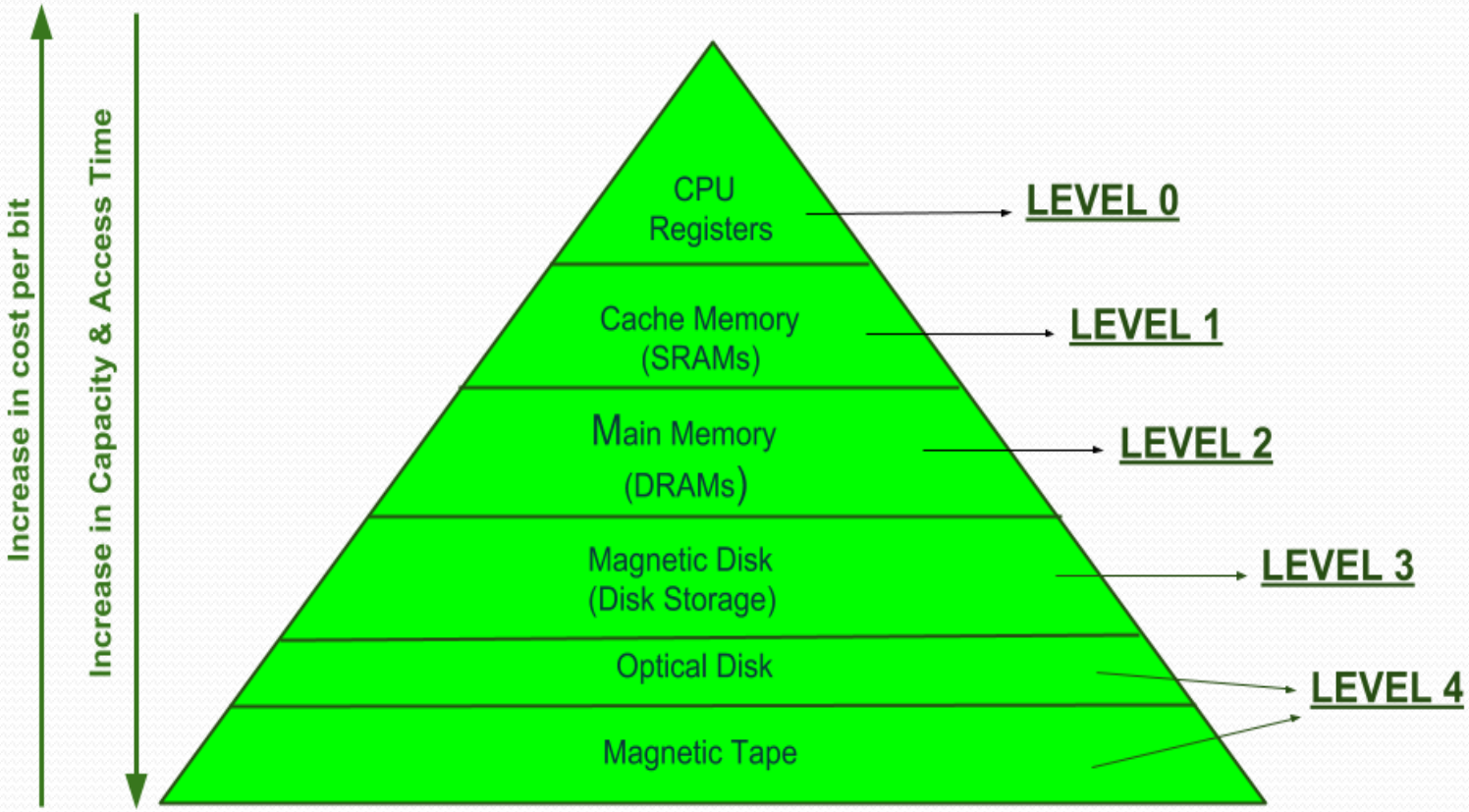
8)Organization:

- For random- access memory, the organization is a key design issue.
- Organization refers to the physical arrangement of bits to form words

Memory Hierarchy

- In our computer system, a processor and many memory devices have been used.
- However, the main problem is, these devices are expensive.
- So the memory organization of the system is done with the help of the memory hierarchy.
- It has various levels of memory with different access times and performance rates.
- But all these can give us an exact purpose, such that the access time can be reduced.
- Therefore the memory hierarchy was developed based on the program.

Memory Hierarchy



MEMORY HIERARCHY DESIGN

Memory Hierarchy

- The memory hierarchy is the arrangement of various types of storage on a computing system based on access speed.
- It organizes computer storage according to response time.
- Since response time, complexity, and capacity are all connected, the levels can also be distinguished by their performance and controlling technologies.
- As shown in fig,, the computer memory has a pyramid-like structure.
- It is used to describe the different levels of memory.

Memory Hierarchy

- This Memory Hierarchy Design is divided into 2 types:
  - **Primary or internal memory**
    It consists of CPU registers, Cache Memory, Main Memory, and these are directly accessible by the processor.
  - **Secondary or external memory**
    It consists of a Magnetic Disk, Optical Disk, Magnetic Tape, which are accessible by processor via I/O Module.

Memory Hierarchy

Internal memory

- It is also called primary memory.
- Internal memory is a part of computer that, when running, can store small amounts of data that need to be accessed quickly.
- It consists of RAM, ROM, and cache memory.
  - **RAM**
    - It is a volatile memory that is used to store whatever is in use by the computer.
    - it acts as a middle man between the CPU and the storage device, which helps speed up the computer.
    - It has many forms, but the most common type is DDR3.

Memory Hierarchy

- **ROM**
  - ROM, unlike other internal memory, is non-volatile.
  - ROM stands for read-only memory, meaning the user cannot write data to the ROM without special access.
  - It was designed so that the computer could access the bios without other parts of hardware.

- **Cache**
  - It is a really small, super-fast memory found in various computer components.
  - The CPU cache stores small bits of frequently accessed data from the RAM.
  - So that the processor doesn't have to wait for the RAM to respond every time it wants the same piece of information.
  - Like RAM, it is volatile and contents are lost whenever the computer gets turned off.

Memory Hierarchy

External memory

- It is also known as secondary memory.
- Since it has a huge capacity, it stores massive data.
- The critical property of external memory is that stored information will not be lost whenever the computer switches off.
- It consists of magnetic tape, a magnetic disk, and an optical disk.
  - **Magnetic tape:**
    - It is a medium for magnetic storage, a thin, magnetic coating on a long, narrow strip of plastic film.
    - It is used for many purposes like recording audio in a recording tape, to store the data on a hard disk.

Memory Hierarchy

- **Magnetic disk**
  - Magnetic disks are flat circular plates of metal or plastic coated on both sides with iron oxide.
  - Input signals, which may be audio, video, or data, are recorded on the surface of a disk as magnetic patterns or spots in spiral tracks by a recording head while a drive unit rotates the disk.
  - It is relatively cheap per storage unit, with Fast access and retrieval times compared to other storage devices.
- **Optical disk**
  - It is an electronic data storage medium that can be written to and read using a low-powered laser beam.
  - Optical disks are most commonly used for digital preservation, music, video, or data and programs for personal computers.
  - Optical media provides many advantages for storing data over conventional magnetic disks, such as mass storage capacity, mountable/un mountable storage units, and low cost per bit of storage.

Characteristics of Memory Hierarchy

- The memory characteristics mainly include:
  - **Access Time**
    - The access time in the memory hierarchy of a computer system is the interval of the time between the data availability and request to read or write.
  - **Capacity**
    - It is the amount of information that can be stored.
    - The capacity increases as we move from top to bottom in the hierarchy

Characteristics of Memory Hierarchy

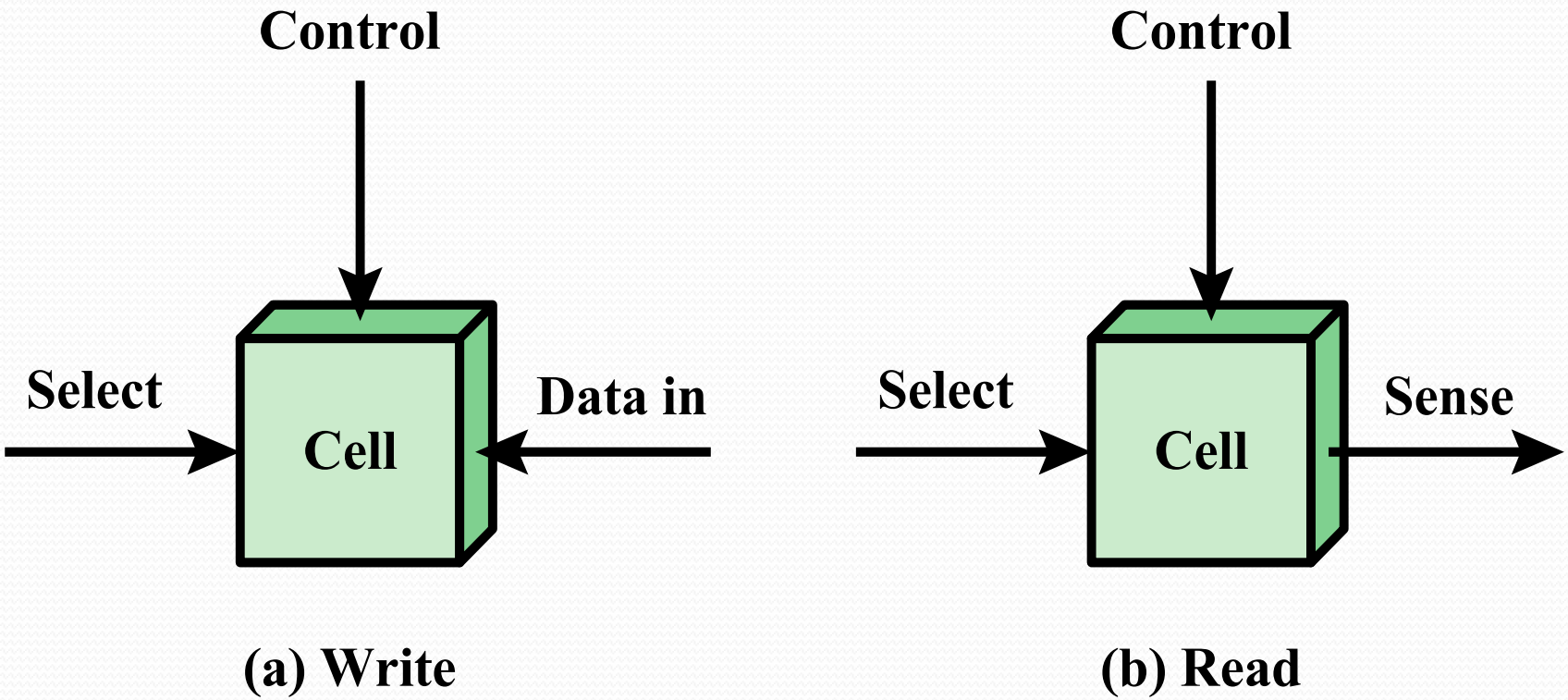
- **Performance**
  - In old times, designing a computer system was done without memory hierarchy.
  - The gap of speed between the main memory and the CPU registers is enhanced because of the huge inconsistency in access time, which will cause the lower performance of the system.
  - So, the enhancement in the memory was mandatory.
  - This enhancement was designed in the memory hierarchy model due to which the system's performance increase.

- **Cost per bit**
  - As we will move from bottom to top in the system's hierarchy, the cost per bit increases,
  - i.e., Internal Memory is costlier than External Memory.

Semiconductor Memory

- The basic element of a **semiconductor memory** is the memory cell.
- Although a variety of electronic technologies are used, all semiconductor memory cells share certain properties:
  - They exhibit two stable (or semi stable) states, which can be used to represent binary 1 and 0.
  - They are capable of being written into (at least once), to set the state.
  - They are capable of being read to sense the state.

Memory Cell operation



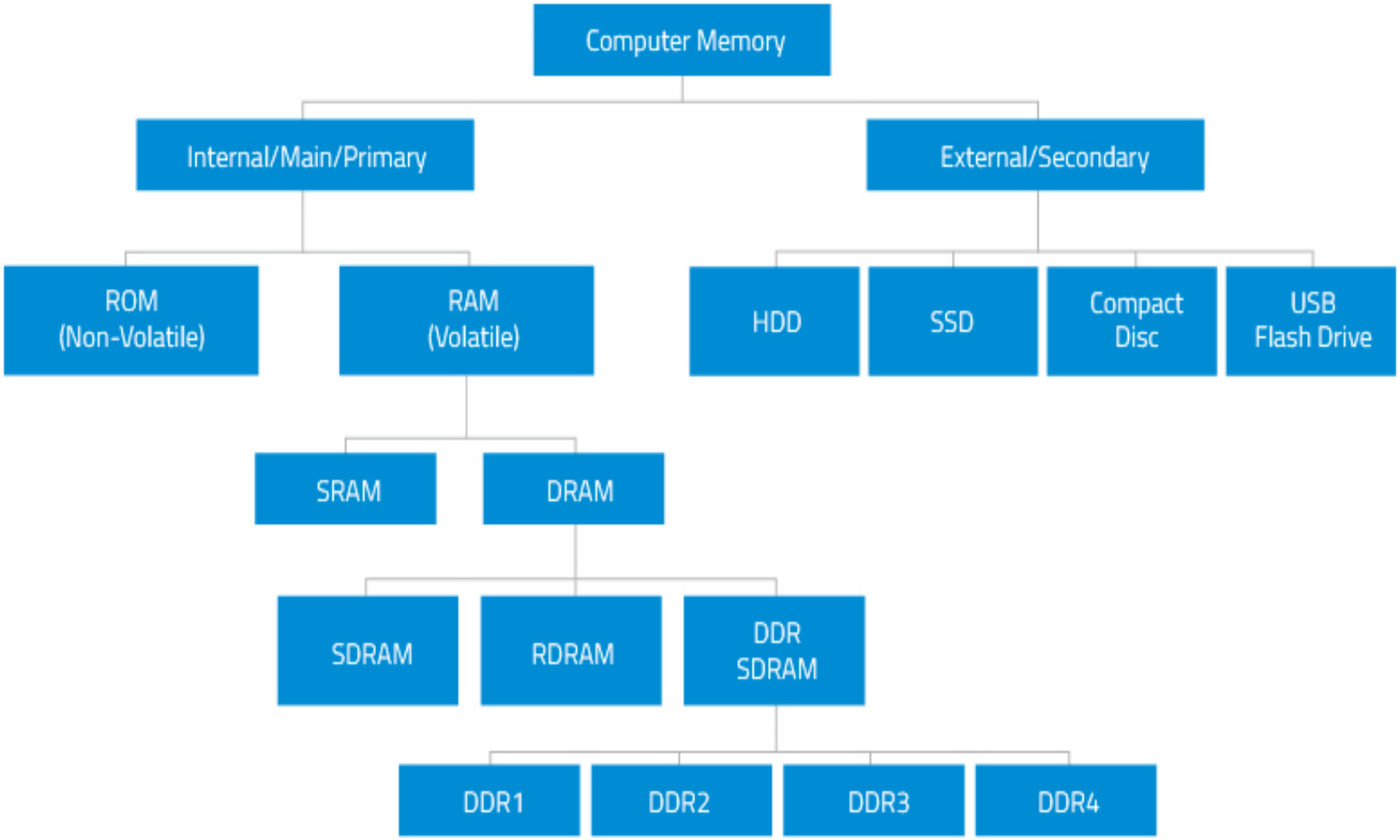
Semiconductor Memory

- Figure shows the operation of a memory cell.
- Most commonly, the cell has three functional terminals capable of carrying an electrical signal.
- The select terminal, as the name suggests, selects a memory cell for a read or write operation.
- The control terminal indicates read or write.
- For writing, the other terminal provides an electrical signal that sets the state of the cell to 1 or 0.
- For reading, that terminal is used for output of the cell's state.

Random Access Memory

- It is a type of Semiconductor memory.
- One distinguishing characteristic of memory is that it is possible both to read data from the memory and to write new data into the memory easily and rapidly.
- Both the reading and writing are accomplished through the use of electrical signals.
- The other distinguishing characteristic of traditional RAM is that it is volatile.
- A RAM must be provided with a constant power supply.
- If the power is interrupted, then the data are lost.
- Thus, RAM can be used only as temporary storage.
- The two traditional forms of RAM used in computers are DRAM and SRAM.

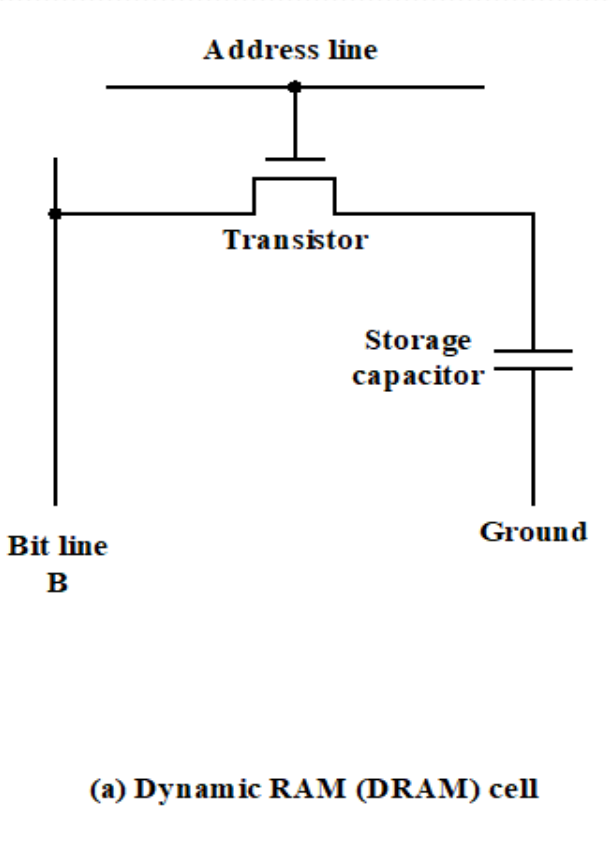
Random Access Memory



Dynamic ram(DRAM)

- A **dynamic RAM (DRAM)** is made with cells that store data as charge on capacitors.
- The presence or absence of charge in a capacitor is interpreted as a binary 1 or 0.
- Because capacitors have a natural tendency to discharge, dynamic RAMs require periodic charge refreshing to maintain data storage.
- The term *dynamic* refers to this tendency of the stored charge to leak away, even with power continuously applied.
- Figure shows a typical DRAM structure for an individual cell that stores one bit.

Dynamic ram(DRAM)



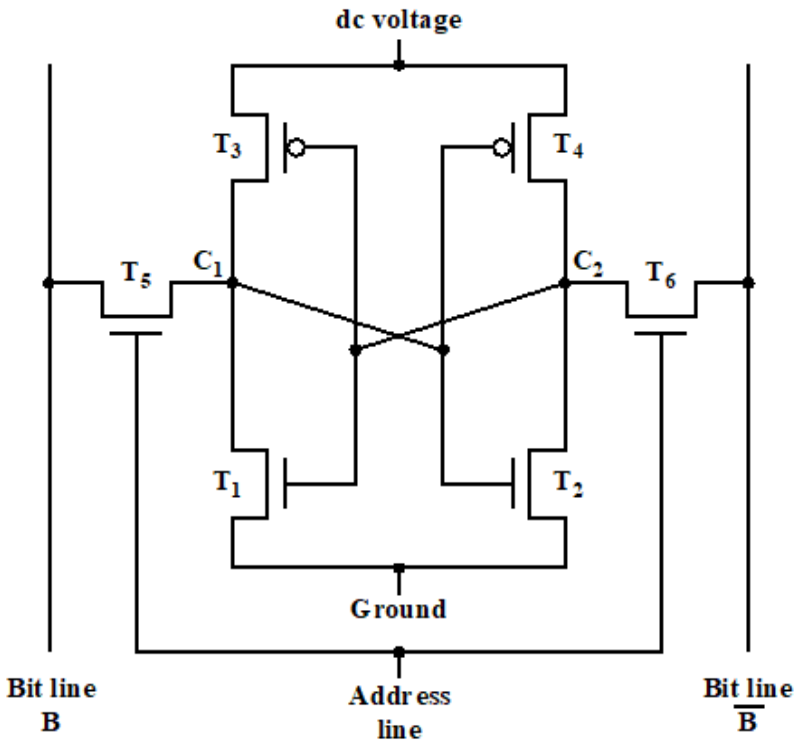
(a) Dynamic RAM (DRAM) cell

- The address line is activated when the bit value from this cell is to be read or written.
- The transistor acts as a switch, and it is closed when a voltage is applied to the address line (allowing current to flow) . And open (no current flows) if no voltage is present on the address line.
- For the write operation, a voltage signal is applied to the bit line; a high voltage represents 1, and a low voltage represents 0.
- A signal is then applied to the address line, allowing a charge to be transferred to the capacitor.
- For the read operation, when the address line is selected, the transistor turns on and the charge stored on the capacitor is fed out onto a bit line and to a sense amplifier.
- The sense amplifier compares the capacitor voltage to a reference value and determines if the cell contains a logic 1 or a logic 0.
- The readout from the cell discharges the capacitor, which must be restored to complete the operation.
- Although the DRAM cell is used to store a single bit (0 or 1), it is essentially an analog device.
- The capacitor can store any charge value within a range; a threshold value determines whether the charge is interpreted as 1 or 0.

Static ram (SRAM)

- A **static RAM (SRAM)** is a digital device that uses the same logic elements used in the processor.
- In a SRAM, binary values are stored using traditional flip-flop logic-gate configurations.
- A static RAM will hold its data as long as power is supplied to it.
- Figure shows a typical SRAM structure for an individual cell.

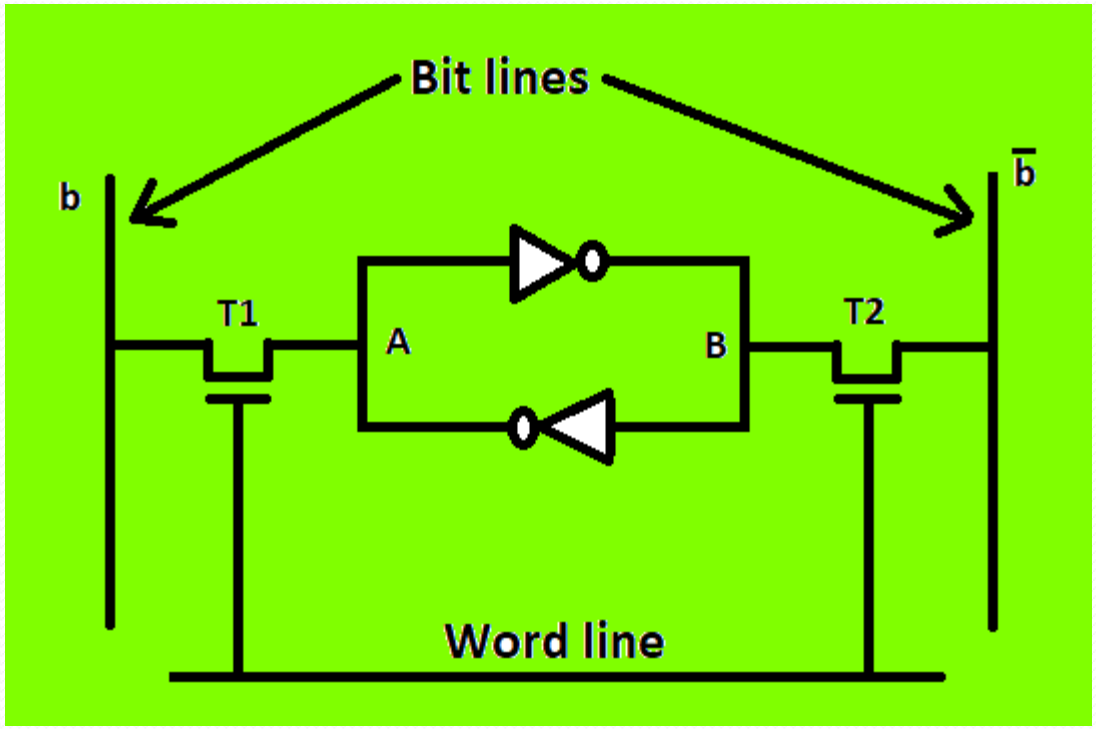
Static ram (SRAM)



(b) Static RAM (SRAM) cell

- Four transistors (T1, T2, T3, T4) are cross connected in an arrangement that produces a stable logic state.
- In logic state 1, point C1 is high and point C2 is low; in this state, T1 and T4 are off and T2 and T3 are on.
- In logic state 0, point C1 is low and point C2 is high; in this state, T1 and T4 are on and T2 and T3 are off.
- Both states are stable as long as the direct current (dc) voltage is applied.
- Unlike the DRAM, no refresh is needed to retain data.
- As in the DRAM, the SRAM address line is used to open or close a switch.
- The address line controls two transistors (T5 and T6).
- When a signal is applied to this line, the two transistors are switched on, allowing a read or write operation.
- For a write operation, the desired bit value is applied to line B, while its complement is applied to line B' .
- This forces the four transistors (T1, T2, T3, T4) into the proper state.
- For a read operation, the bit value is read from line B.

A latch is formed by two inverters



SRAM	DRAM
It is a static memory as it does not need to be refreshed repeatedly.	It is a dynamic memory as it needs to be refreshed continuously or it will lose the data.
Its memory cell is made of 6 transistors.	Its memory cell is made of one transistor and one capacitor.
Its cells occupy more space on a chip and offer less storage capacity (memory) than a DRAM of the same physical size.	Its cells occupy less space on a chip and provide more memory than a SRM of the same physical size.
It is more expensive than DRAM.	It is less expensive than SRAM
It has a lower access time, e.g. 10 nanoseconds. So, it is faster than DRAM.	It has a higher access time, e.g. more than 50 nanoseconds. So, it is slower than SRAM.
It stores information in a bistable latching circuitry. It requires regular power supply so it consumes more power.	The information or each bit of data is stored in a separate capacitor within an integrated circuit so it consumes less power.
It is mostly used in registers in the CPU and cache memory of various devices.	It is used in the motherboard because it is cheaper to manufacture and requires less space.
It is located on processors or between a processor and main memory.	It is mostly located on the motherboard.
Examples: L2 and L1 cache in a CPU.	Example: DDR3, DDR4 in mobile phones, computers, etc.
Size ranges from 1 MB to 16MB.	Size ranges from 1 GB to 3 GB in smartphones and 4GB to 16GB in laptops.

Synchronous DRAM

- One of the most widely used forms of DRAM is the **synchronous DRAM (SDRAM)**.
- Unlike the traditional DRAM, which is asynchronous, the SDRAM exchanges data with the processor synchronized to an external clock signal and running at the full speed of the processor/memory bus without imposing wait states.

Synchronous DRAM

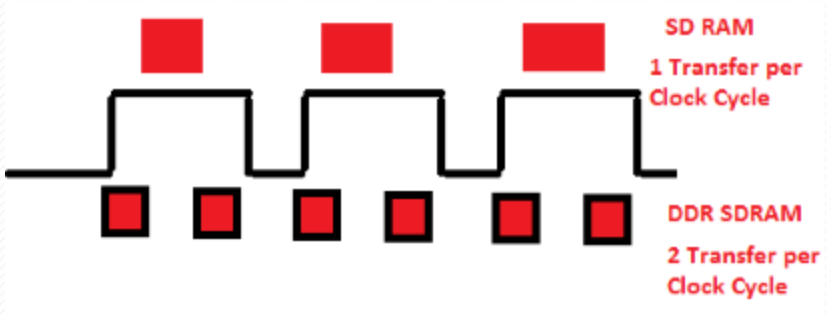
- In a typical DRAM, the processor presents addresses and control levels to the memory, to indicate that a set of data at a particular location in memory should be either read from or written into the DRAM.
- After a delay, the access time, the DRAM either writes or reads the data.
- During the access-time delay, the DRAM performs various internal functions, such as activating the high capacitance of the row and column lines, sensing the data, and routing the data out through the output buffers.
- The processor must simply wait through this delay, slowing system performance.

Synchronous DRAM

- With synchronous access, the DRAM moves data in and out under control of the system clock.
- The processor issues the instruction and address information, which is latched by the DRAM.
- The DRAM then responds after a set number of clock cycles.
- Meanwhile, the processor can safely do other tasks while the SDRAM is processing the request.

DDR SDRAM

- The next generation of the synchronous DRAM is known as the DDR RAM.
- It was developed to overcome the limitations of SDRAM and was used in PC memory at the beginning of the year 2000.
- In DDR SDRAM (DDR RAM), the data is transferred twice during each clock cycle; during the positive edge (rising edge) and the negative edge (falling edge) of the cycle.
- So, it is known as the double data rate SDRAM.

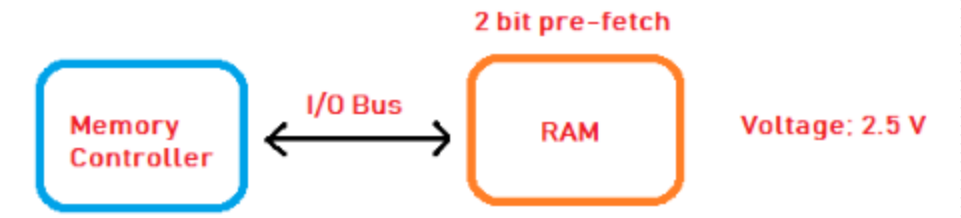


DDR SDRAM

- There are different generations of DDR SDRAM which include DDR1, DDR2, DDR3, and DDR4.
- Today, the memory that we use inside the desktop, laptop, mobile, etc., is mostly either DDR3 or DDR4 RAM.

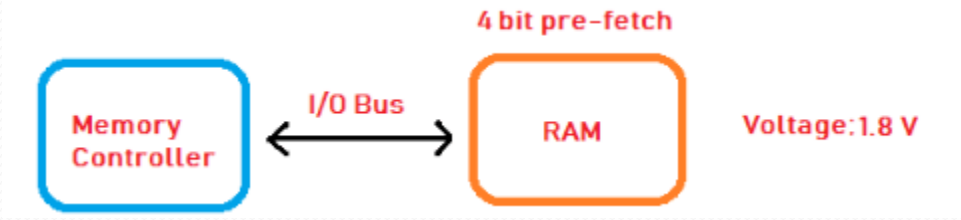
DDR1 SDRAM

- DDR1 SDRAM is the first advanced version of SDRAM.
- In this RAM, the voltage was reduced from 3.3 V to 2.5 V.
- The data is transferred during both the rising as well as the falling edge of the clock cycle.
- So, in each clock cycle, instead of 1 bit, 2 bits are being pre-fetched which is commonly known as the 2 bit pre-fetch.
- It is mostly operated in the range of 133 MHz to the 200 MHz.
- Furthermore, the data rate at the input-output bus is double the clock frequency because the data is transferred during both the rising as well as falling edge.
- So, if a DDR1 RAM is operating at 133 MHz, the data rate would be double, 266 Mega transfer per second.
- its maximum density was 128 Mb (so there were no modules with more than 1 GB)



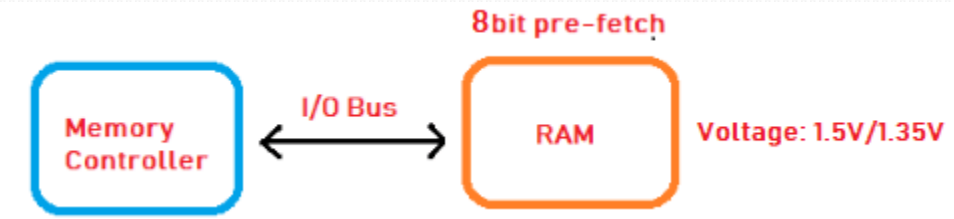
DDR2 SDRAM

- It is an advanced version of DDR1.
- It operates at 1.8 V instead of 2.5V.
- Its data rate is double the data rate of the previous generation due to the increase in the number of bits that are pre-fetched during each cycle; 4 bits are pre-fetched instead of 2 bits.
- The internal bus width of this RAM has been doubled.
- For example, if the input-output bus is 64 bits wide, the internal bus width of it will be equal to 128 bits.
- So, a single cycle can handle double the amount of data.
- Its maximum density was doubled to 256 Mb (2 GB per module)



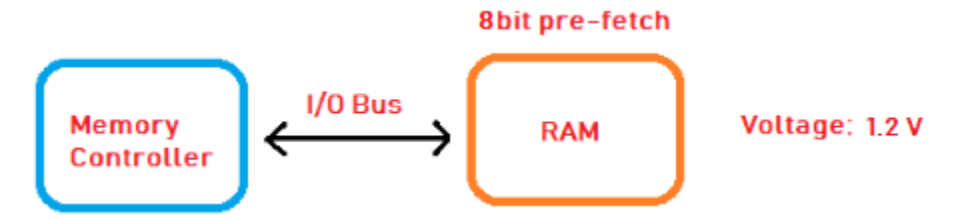
DDR3 SDRAM

- In this version, the voltage is further reduced from 1.8 V to the 1.5 V.
- The data rate has been doubled than the previous generation RAM as the number of bits that are pre-fetched has been increased from 4 bits to the 8 bits.
- We can say that the internal data bus width of RAM has been increased 2 times than that of the last generation.
- the density reached up to 8 GB per module.



DDR4 SDRAM

- In this version, the operating voltage is further reduced from 1.5 V to 1.2 V, but the number of bits that can be pre-fetched is same as the previous generation; 8 bits per cycle.
- The Internal clock frequency of the RAM is double of the previous version.
- If you are operating at 400 MHz the clock frequency of the input-output bus would be four times, 1600 MHz and the transfer rate would be equal to 3200 Mega transfer per second.
- Currently, there are already 32 GB modules, but this is also being expanded



DDR5 SDRAM

- DDR5 is the next evolution of PC main memory.
- This version is primarily focused on increasing the density and bandwidth of RAM while lowering its power consumption.
- The base DDR5 bandwidth (4,800 MT/s) performs over 50 percent better compared to DDR4 (3,200 MT/s).

- MT – Mega Transfer or Million Transfer

DDR1

DDR2

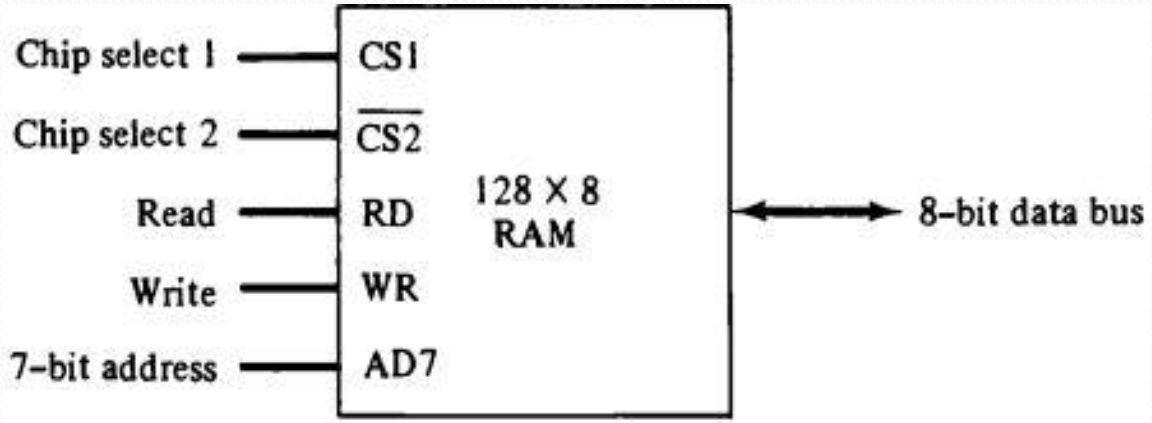
DDR3

DDR4


crucial


DOMINATOR
PLATINUM RGB

RAM chip



CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

- The block diagram of a RAM Chip is shown in Fig
- The capacity of memory is 128 words of eight bits (one byte) per word.
- This requires a 7-bit address and an 8-bit bidirectional data bus.
- The read and write inputs specify the memory operation.
- The two chips select (CS) control inputs are enabling the chip only when it is selected by the microprocessor.
- High impedance state indicates open circuit i.e. output does not carry a signal and has no logic significance.

ROM (Read Only Memory)

- **read-only memory (ROM)** contains a permanent pattern of data that cannot be changed.
- A ROM is nonvolatile; that is, no power source is required to maintain the bit values in memory.
- While it is possible to read a ROM, it is not possible to write new data into it.
- An important application of ROMs is microprogramming.
- The data or program is permanently in main memory and need never be loaded from a secondary storage device.

ROM (Read Only Memory) Chip

ROM (Read Only Memory)

- A ROM is created like any other integrated circuit chip, with the data actually wired into the chip as part of the fabrication process.
- This presents two problems:
  - The data insertion step includes a relatively large fixed cost, whether one or thousands of copies of a particular ROM are fabricated.
  - There is no room for error. If one bit is wrong, the whole batch of ROMs must be thrown out.
- When only a small number of ROMs with a particular memory content is needed, a less expensive alternative is the **programmable ROM (PROM)**.

programmable ROM (PROM).

- The PROM is **nonvolatile** and may be written into only once.
- For the PROM, the writing process is performed electrically and may be performed by a supplier or customer at a time later than the original chip fabrication.
- Special equipment is required for the writing or "programming" process.
- PROMs provide flexibility and convenience.
- The ROM remains attractive for high-volume production runs.

ROM

- Another variation on read-only memory is the **read-mostly memory.**
- **This** memory is useful for applications in which read operations are far more frequent than write operations
- But for these memories nonvolatile storage is required.
- There are three common forms of read-mostly memory: EPROM, EEPROM, and flash memory.

EPROM

- The optically **erasable programmable read-only memory (EPROM)** is read and written electrically, as with PROM.
- However, before a write operation, all the storage cells must be erased to the same initial state by exposure of the packaged chip to ultraviolet radiation.
- Erasure is performed by shining an intense ultraviolet light through a window that is designed into the memory chip.
- This erasure process can be performed repeatedly; each erasure can take as much as 20 minutes to perform.
- Thus, the EPROM can be altered multiple times and, like the ROM and PROM, holds its data virtually indefinitely.
- For comparable amounts of storage, the EPROM is more expensive than PROM, but it has the advantage of the multiple update capability.

EPROM

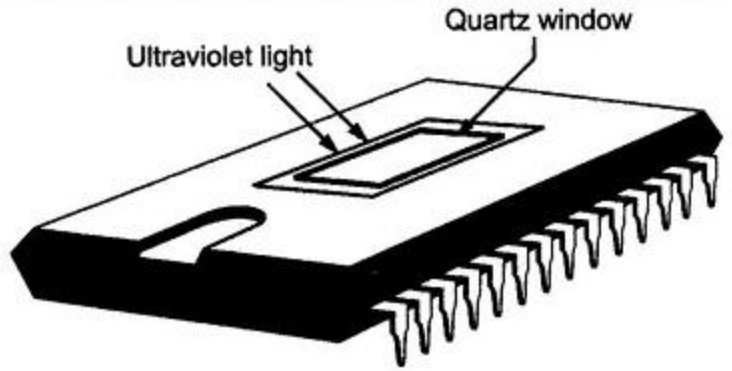


Fig. 3.72 EPROM

EEPROM

- A more attractive form of read-mostly memory is **electrically erasable programmable read-only memory (EEPROM)**.
- This is a read-mostly memory that can be written into at any time without erasing prior contents; only the byte or bytes addressed are updated.
- The write operation takes considerably longer than the read operation, on the order of several hundred microseconds per byte.
- The EEPROM combines the advantage of nonvolatility with the flexibility of being updatable in place, using ordinary bus control, address, and data lines.
- EEPROM is more expensive than EPROM and also is less dense, supporting fewer bits per chip.

flash memory

- Another form of semiconductor memory is **flash memory** (so named because of the speed with which it can be reprogrammed).
- Flash memory is intermediate between EPROM and EEPROM in both cost and functionality.
- Like EEPROM, flash memory uses an electrical erasing technology.
- An entire flash memory can be erased in one or a few seconds, which is much faster than EPROM.
- In addition, it is possible to erase just blocks of memory rather than an entire chip.
- Flash memory gets its name because the microchip is organized so that a section of memory cells are erased in a single action or "flash."
- However, flash memory does not provide byte-level erasure.
- Like EPROM, flash memory uses only one transistor per bit, and so achieves the high density (compared with EEPROM) of EPROM.

ROM Chip

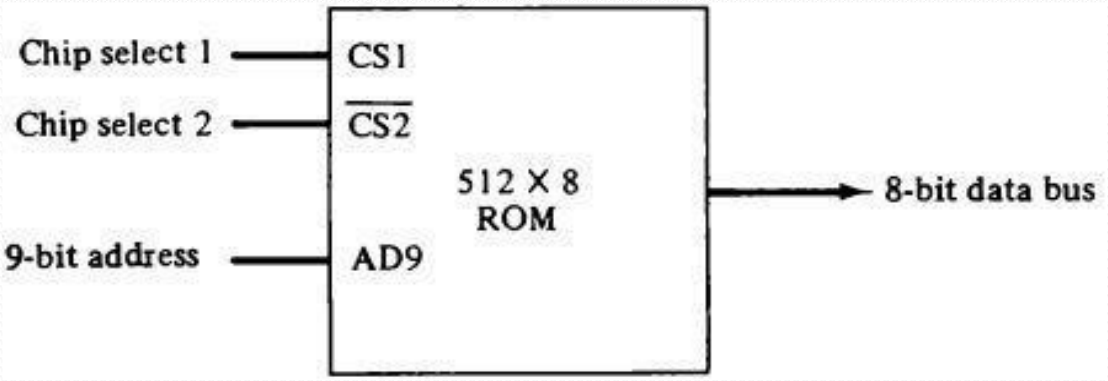


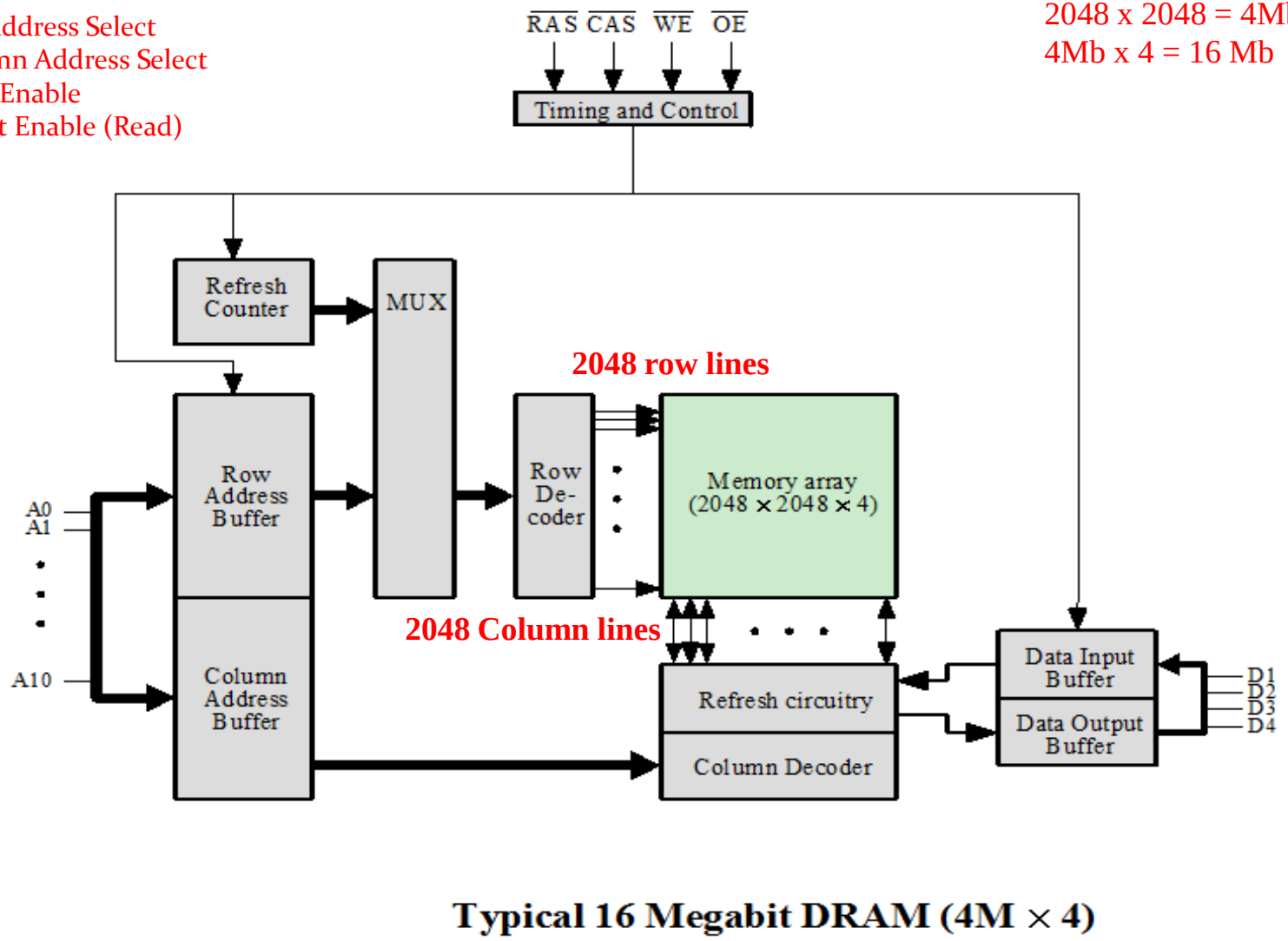
Fig: Typical ROM chip (512 byte ROM)

- As shown in fig. as a ROM chip can only read, data bus is unidirectional (output mode only).
- There are 9 address lines to address 512 bytes.
- There are two chip select (CS) inputs $CS_1=1$ and $(CS_2)'=0$ for the unit to operate, otherwise the data bus is in high-impedance state.
- There is no need for read or write control since the unit can only read.

Memory Organization

RAS- Row address Select
CAS- Column Address Select
WE- Write Enable
OE- Output Enable (Read)

2048 x 2048 = 4Mb
4Mb x 4 = 16 Mb



Typical 16 Megabit DRAM (4M × 4)

Memory Organization

- Figure shows a typical organization of a 16-Mbit DRAM.
- In this case, 4 bits are read or written at a time.
- Logically, the memory array is organized as four square arrays of 2048 by 2048 elements.
- Various physical arrangements are possible.
- In any case, the elements of the array are connected by both horizontal (row) and vertical (column) lines.
- Each horizontal line connects to the Select terminal of each cell in its row; each vertical line connects to the Data-In/Sense terminal of each cell in its column.

Memory Organization

- Address lines supply the address of the word to be selected.
- As 2^{11} = 2048 so in our example, 11 address lines are needed to select one of 2048 rows.
- These 11 lines are fed into a row decoder, which has 11 lines of input and 2048 lines for output.
- The logic of the decoder activates a single one of the 2048 outputs depending on the bit pattern on the 11 input lines (2^{11} = 2048).

Memory Organization

- An additional 11 address lines select one of 2048 columns of 4 bits per column.
- Four data lines are used for the input and output of 4 bits to and from a data buffer.
- On input (write), the bit driver of each bit line is activated for a 1 or 0 according to the value of the corresponding data line.
- On output (read), the value of each bit line is passed through a sense amplifier and presented to the data lines.
- The row line selects which row of cells is used for reading or writing.

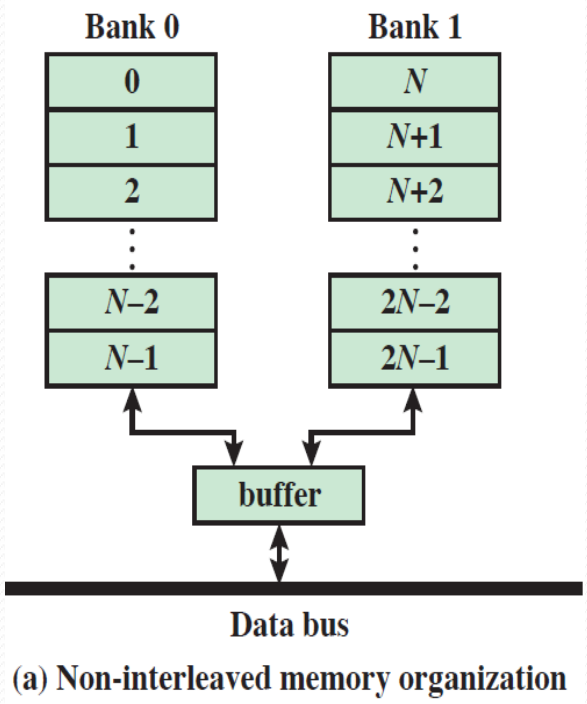
Memory Organization

- First, 11 address signals are passed to the chip to define the row address of the array, and then the other 11 address signals are presented for the column address.
- These signals are accompanied by row address select (RAS) and column address select (CAS) signals to provide timing to the chip.
- The write enable (WE) and output enable (OE) pins determine whether a write or read operation is performed.
- Two other pins, not shown in Figure, are ground (Vss) and a voltage source (Vcc).

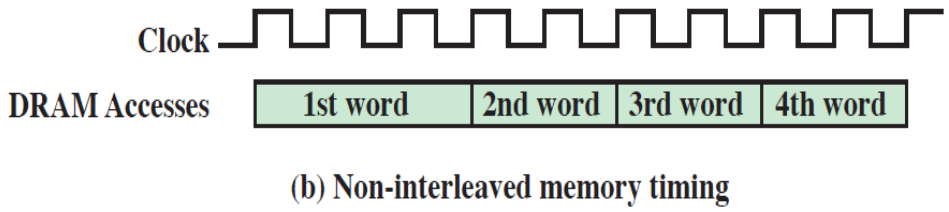
Interleaved Memory

- Main memory is typically of a series of DRAM chips.
- A number of these chips grouped together to form a memory bank.
- Multiple memory banks can be connected together to form an interleaved memory system.
- Because each bank can service a request, an interleaved memory system with K banks can service K requests simultaneously.
- Thus it increases the peak data transfer rate by a factor of K over the data transfer rate of a single bank.
- In most memory systems, the number of banks is a power of 2

Interleaved Memory



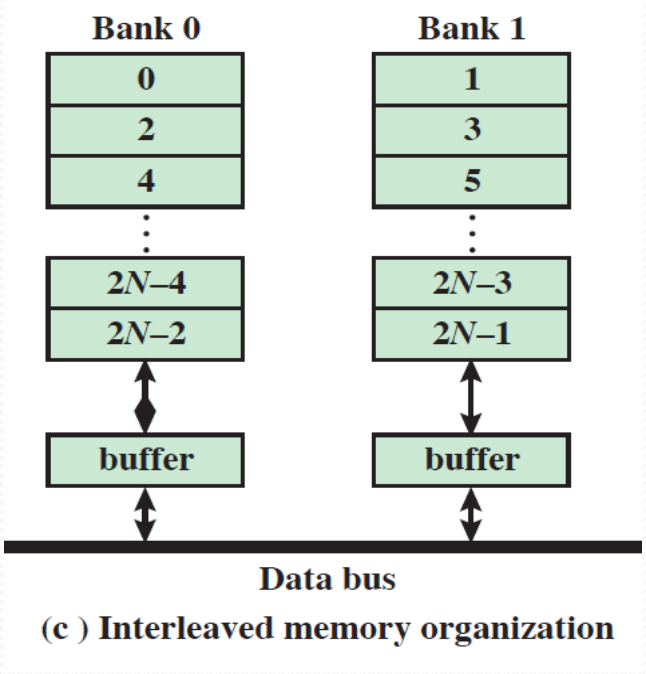
(a) Non-interleaved memory organization



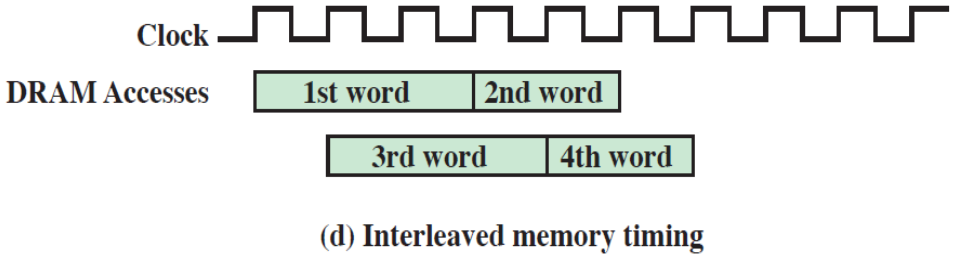
(b) Non-interleaved memory timing

- let us consider a simple system consisting of two DRAM memory banks.
- If the memory controller does not support interleaving, then the memory addresses are assigned sequentially in the first bank, followed by addresses in the second bank.
- Figure shows this organization for two banks of N words.
- Typically, when the memory controller will perform a read or write of multiple words.
- Then all the words in the block come from one bank of DRAM in a non-interleaved memory organization.
- so the time required to complete the transfer is a linear function of the number of words transferred.
- Figure b shows the timing of the transfer.

Interleaved Memory

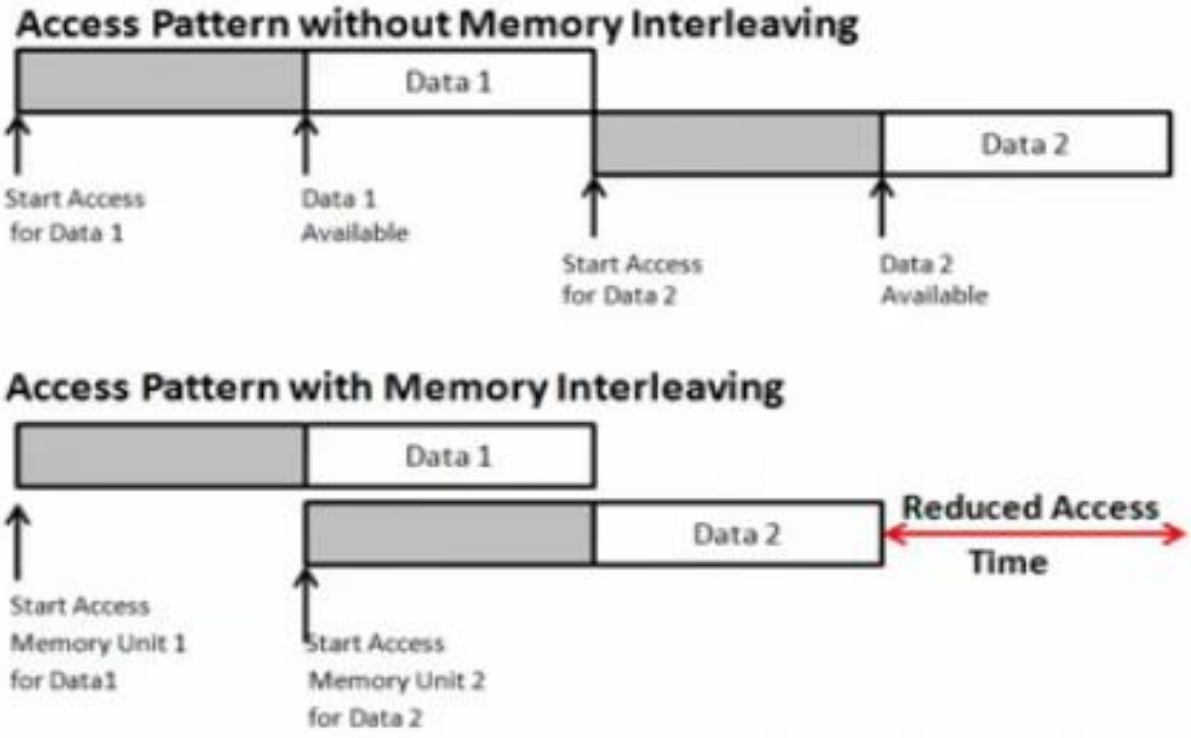


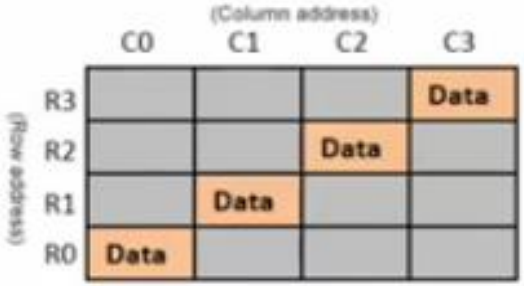
(c) Interleaved memory organization



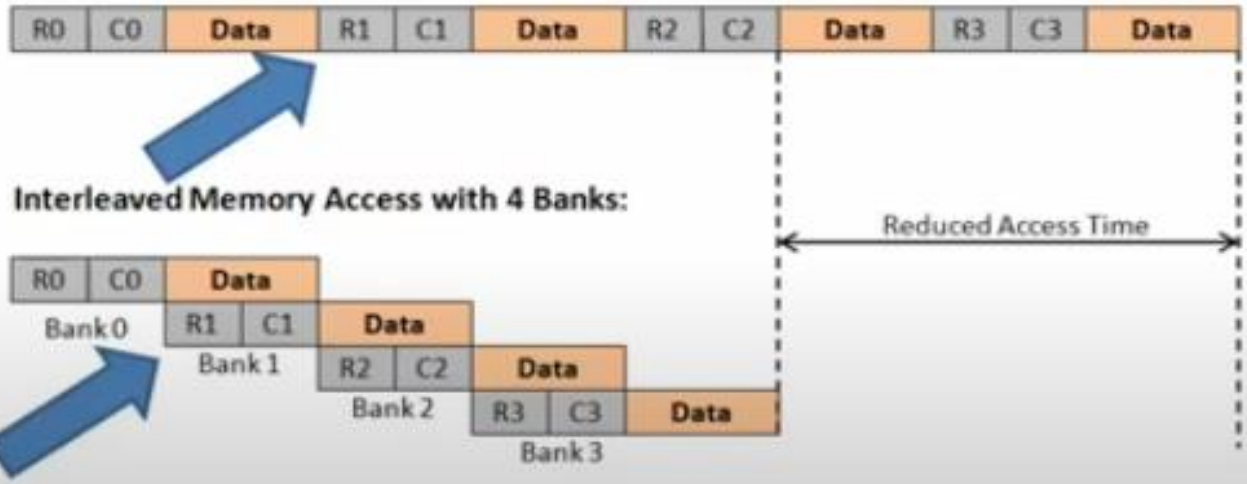
(d) Interleaved memory timing

- If the memory controller supports interleaved memory, then memory addresses are organized as shown in Figure c.
- Memory location addresses alternate between the two banks.
- This configuration speeds up the multiple transfer of four words, as shown in the timing diagram of Figure d.
- Because the four words of a multiple access are spread across two physical banks of DRAM, the individual accesses can be overlapped to hide part, or all, of the DRAM access time delay.


Access Pattern without Memory Interleaving
Data 1
Data 2
Start Access for Data 1
Data 1 Available
Start Access for Data 2
Data 2 Available
Access Pattern with Memory Interleaving
Data 1
Data 2
Reduced Access Time
Start Access Memory Unit 1 for Data1
Start Access Memory Unit 2 for Data 2



Non-Interleaved Memory Access with No Bank Division:

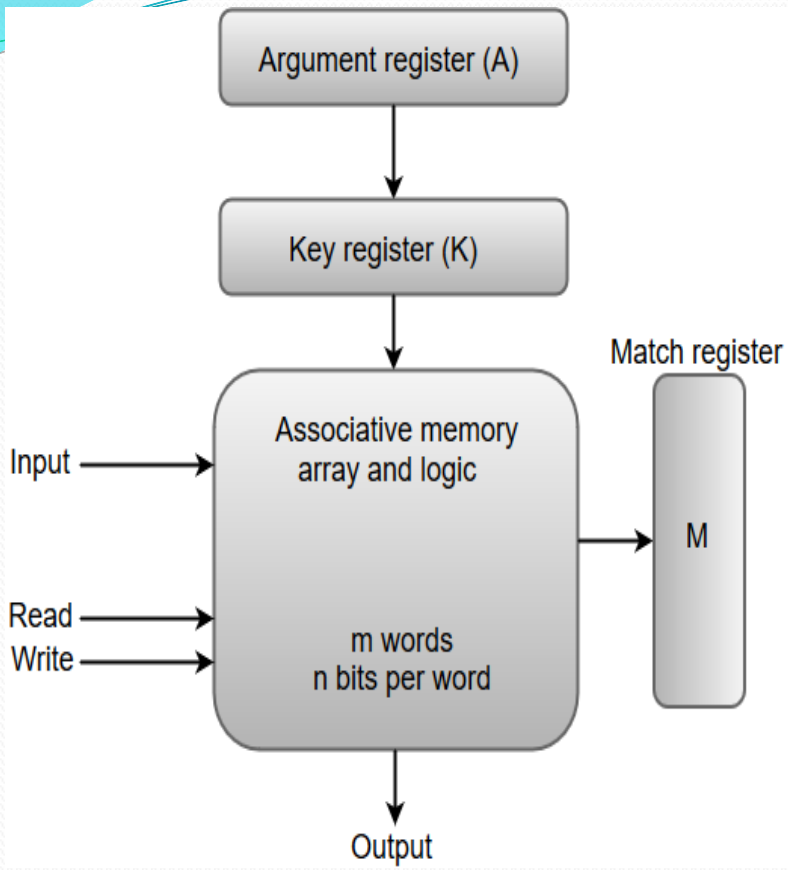


Interleaved Memory Access with 4 Banks:

Associative Memory

- An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.
- Associative memory is often referred to as **Content Addressable Memory (CAM)**.
- When a write operation is performed on associative memory, no address or memory location is given to the word.
- The memory itself is capable of finding an empty unused location to store the word.
- On the other hand, when the word is to be read from an associative memory, the content of the word, or part of the word, is specified.
- The words which match the specified content are located by the memory and are marked for reading.

Associative Memory



A	1 1 0 1 1	0 1 0	
K	0 0 0 0 0	1 1 1	
Word 1	0 1 0 1 0	0 1 0	match
Word 2	1 1 0 1 1	1 0 0	no match

- The Fig. 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.

Solid State Drives (SSD)

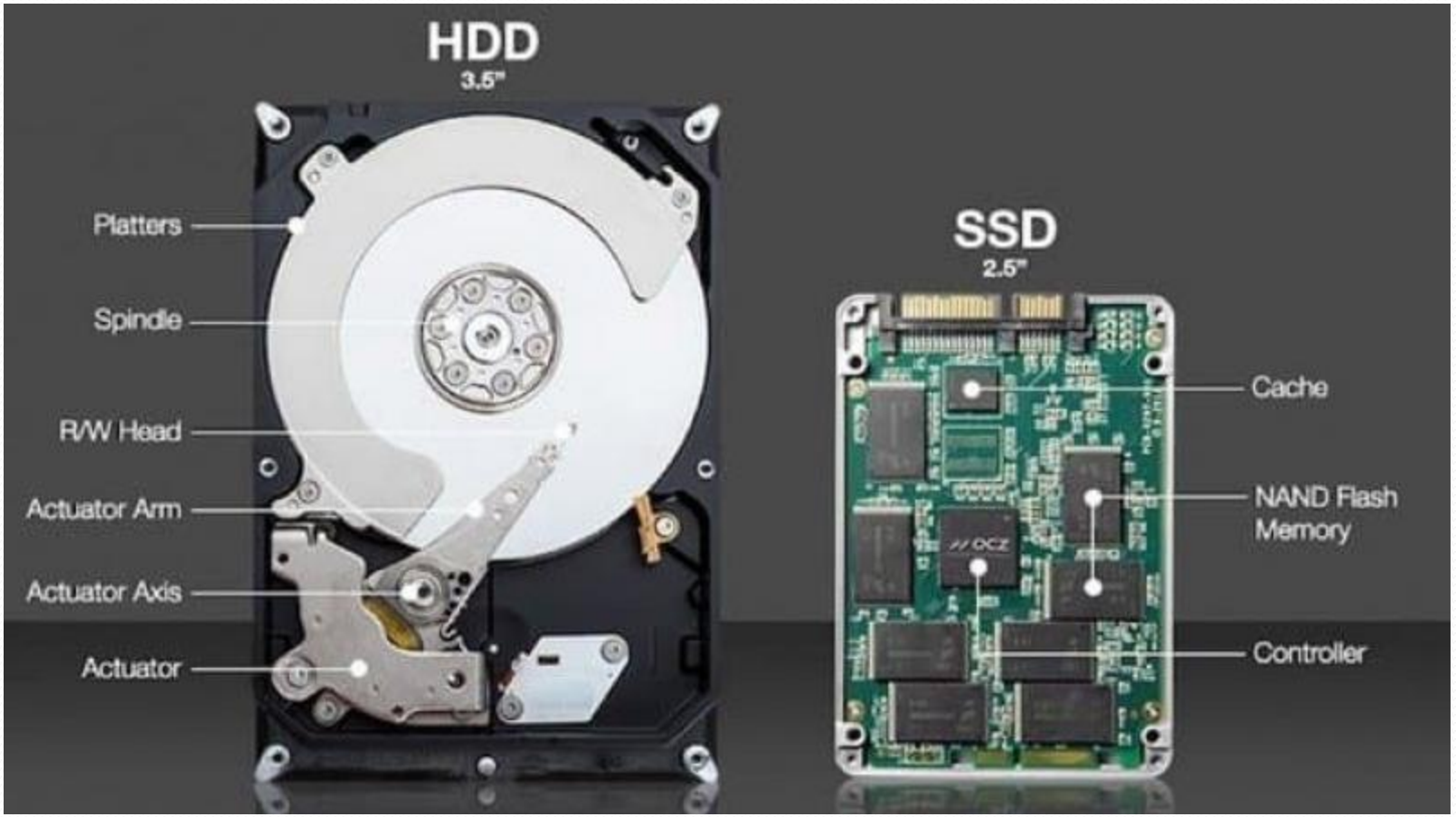
- One of the most significant developments in computer architecture in recent years is the increasing use of solid state drives (SSDs) to complement or even replace **hard disk drives (HDDs)**, both as internal and external secondary memory.
- The term *solid state* refers to electronic circuitry built with semiconductors.
- An SSD is a memory device made with solid state components that can be used as a replacement to a hard disk drive

SSD

- As the cost of flash-based SSDs has dropped and the performance and bit density increased, SSDs have become increasingly competitive with HDDs.
- SSDs have the following advantages over HDDs:
  - **High-performance input/output operations per second (IOPS):** Significantly increases performance I/O subsystems.
  - **Durability:** Less susceptible to physical shock and vibration.
  - **Longer lifespan:** SSDs are not susceptible to mechanical wear.
  - **Lower power consumption:** SSDs use considerably less power than comparable-size HDDs.
  - **Quieter and cooler running capabilities:** Less space required, lower energy costs.
  - **Lower access times and latency rates:** Over 10 times faster than the spinning disks in an HDD.

SSD

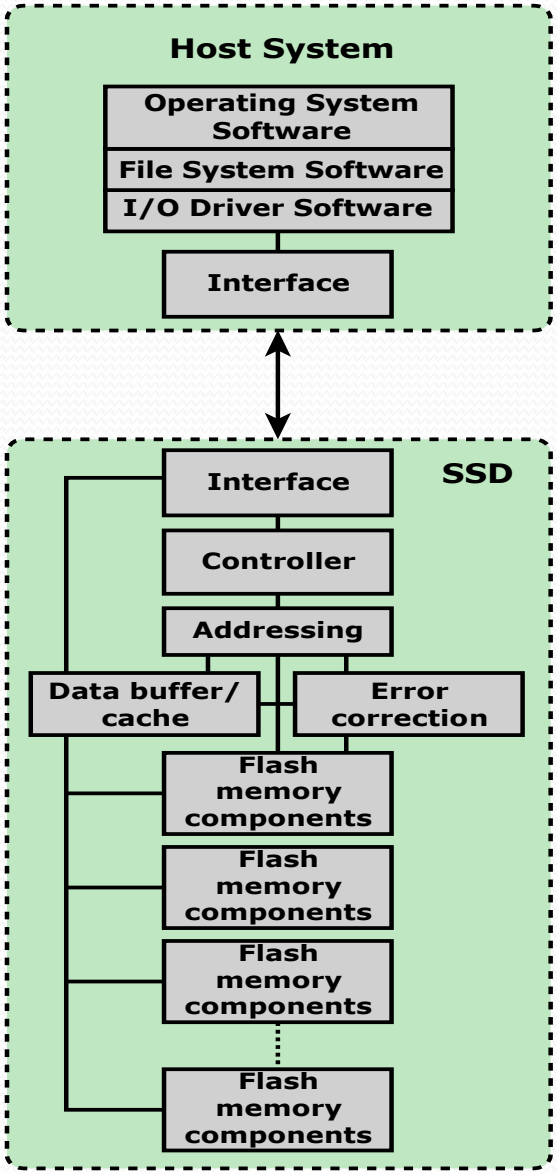
SSD, the device's operating system will boot up more rapidly, programs will load quicker and files can be saved faster.


HDD
3.5"
Platters
Spindle
R/W Head
Actuator Arm
Actuator Axis
Actuator
SSD
2.5"
Cache
NAND Flash
Memory
Controller

SSD organization

- Figure illustrates a general view of the common architectural system component associated with any SSD system.
- On the host system, the operating system invokes file system software to access data on the disk.
- The file system, in turn, invokes I/O driver software.
- The I/O driver software provides host access to the particular SSD product.
- The interface component in Figure refers to the physical and electrical interface between the host processor and the SSD peripheral device.
- If the device is an internal hard drive, a common interface is PCI.
- For external devices, one common interface is USB.

SSD organization



SSD organization

- In addition to the interface to the host system, the SSD contains the following components:
  - **Controller:** Provides SSD device level interfacing and firmware execution.
  - **Addressing:** Logic that performs the selection function across the flash memory components.
  - **Data buffer/cache:** High speed RAM memory components used for speed matching and to increased data throughput.
  - **Error correction:** Logic for error detection and correction.
  - **Flash memory components:** Individual NAND flash chips.

Practical Issues

- SDD performance has a tendency to slow down as the device is used
  - The entire block must be read from the flash memory and placed in a RAM buffer
  - Before the block can be written back to flash memory, the entire block of flash memory must be erased
  - The entire block from the buffer is now written back to the flash memory
- Flash memory becomes unusable after a certain number of writes.

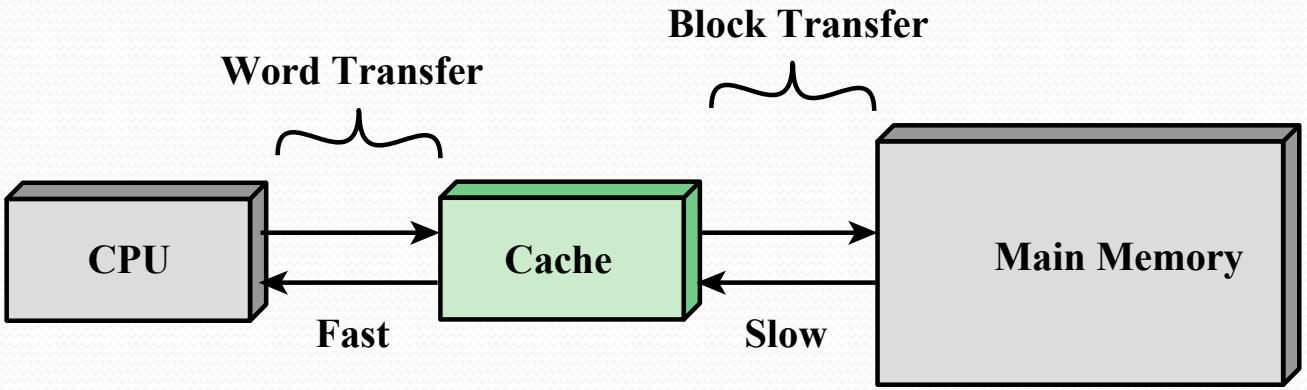
Cache memory

- Cache Memory is a special very high-speed memory.
- It is used to speed up and synchronizing with high-speed CPU.
- *Cache* is memory placed in between the processor and main memory.
- Cache is responsible for holding copies of main memory data for faster retrieval by the processor.
- When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache.

Cache memory

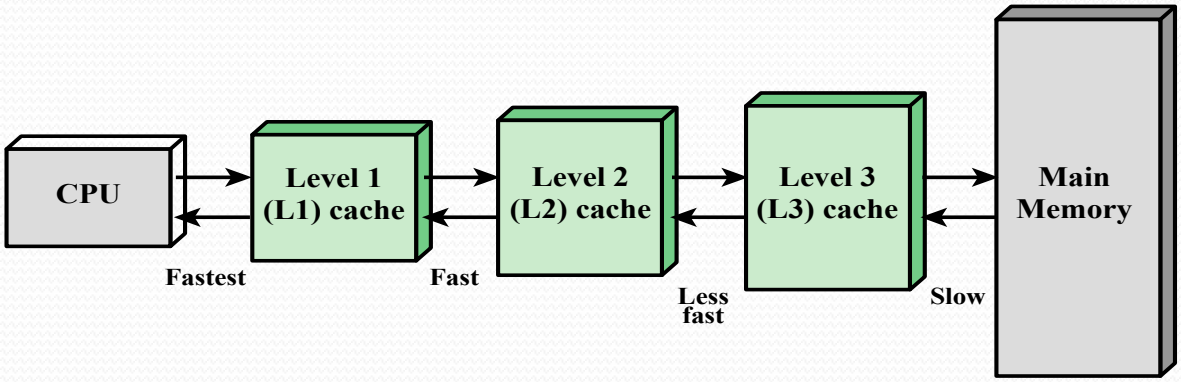
- If so, the word is delivered to the processor.
- If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor.
- Because of the phenomenon of locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to that same memory location or to other words in the block

Cache memory



(a) Single cache

Cache memory



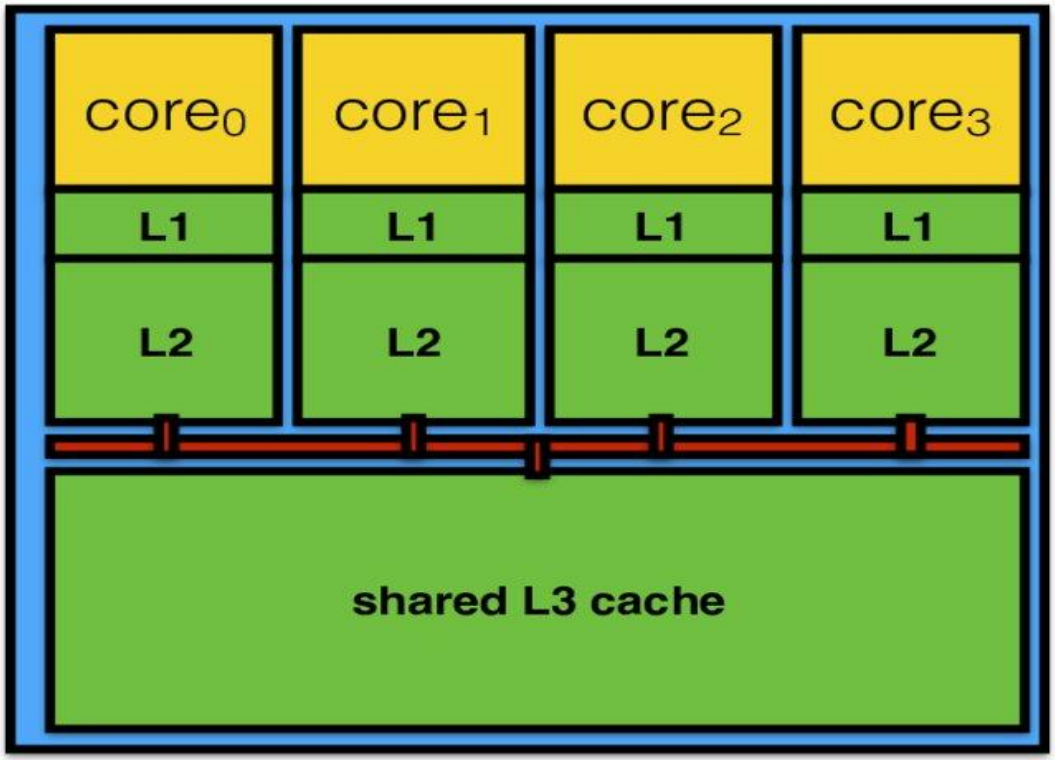
(b) Three-level cache organization

- Figure shows the use of multiple levels of cache.
- The L2 cache is slower and typically larger than the L1 cache, and the L3 cache is slower and typically larger than the L2 cache.

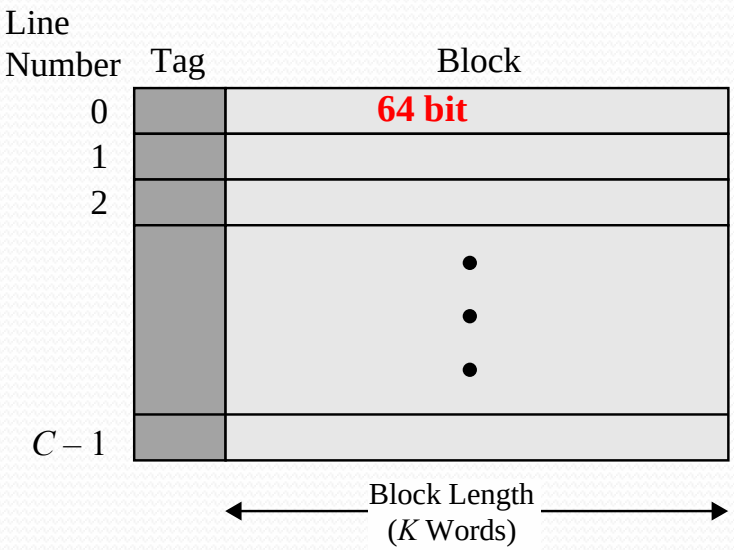
Cache Levels

- L1 cache
  - it is the level 1 cache memory also called as **primary cache**.
  - It operates at the same speed as the CPU and is the fastest cache among all other caches.
  - it is smaller than other caches, which are L2 and L3.
  - Each core in the CPU has their own L1 cache memory.
- L2 Cache
  - L2 cache is the level 2 cache.
  - It stores the data that is not stored in the L1 cache.
  - In other words, if the CPU cannot find the data it is looking for in the L1 cache, it checks the L2 cache.
  - L2 cache is larger than L1 cache but smaller than L3 cache.
  - Each core in the CPU has their own L2 cache memory.
- L3 cache
  - It is the level 3 cache.
  - It stores data that is not stored in L1 and L2 cache.
  - In other words, if the CPU cannot find the data it is looking for in L1 and L2 cache, it checks L3 cache.
  - L3 cache is the largest among all caches and all the cores in the CPU share the same L3 cache memory.

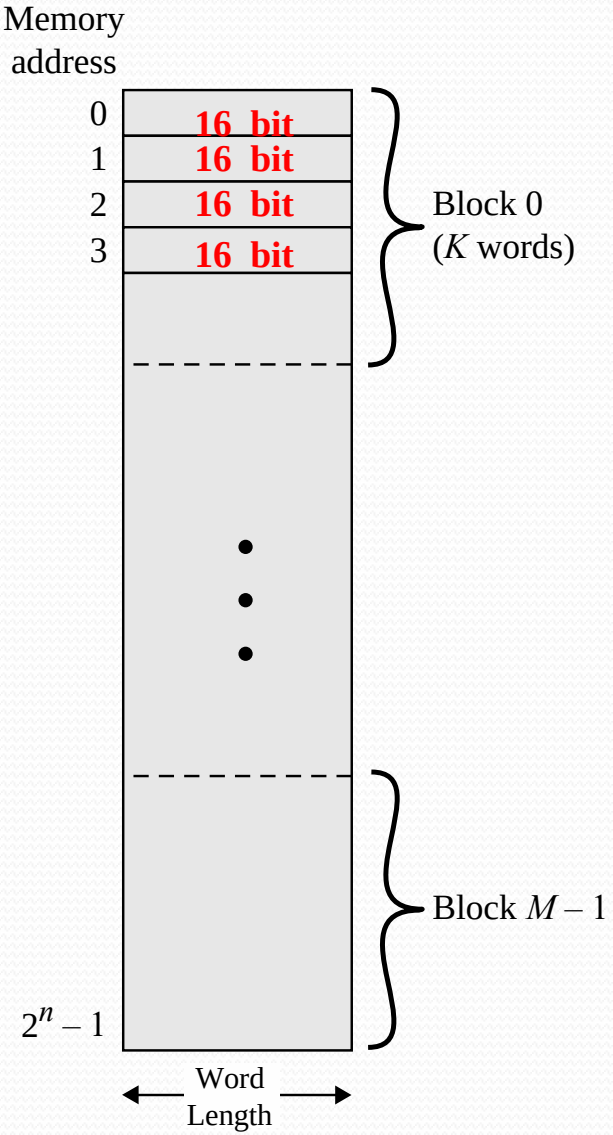
Cache Levels



Cache/Main Memory Structure



(a) Cache



(b) Main memory

n=5

2^n -1 = 32-1=31

MM size = 32x 16= 512 bits

K=4 words

M=32/4 = 8 blocks

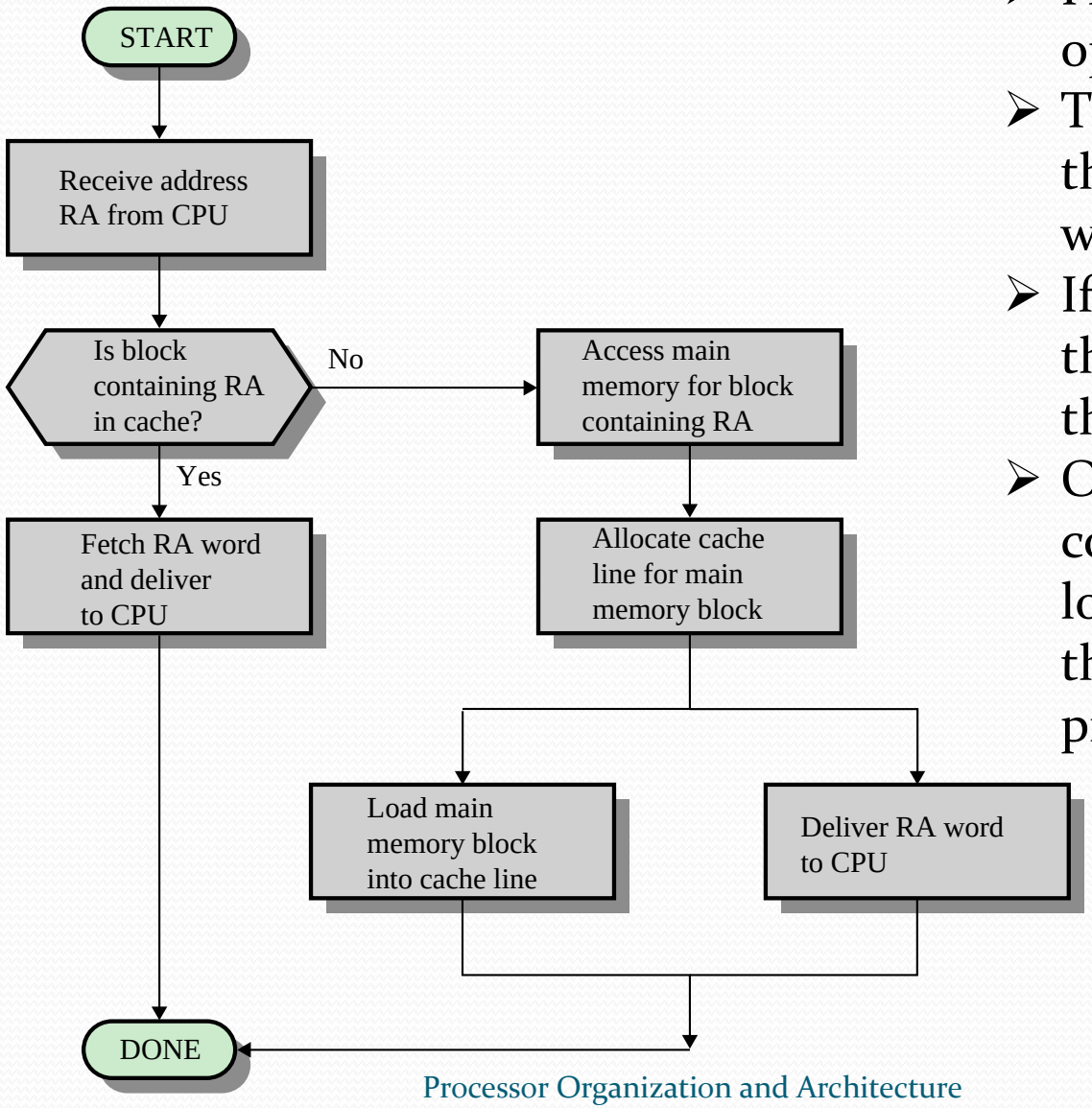
Cache/Main Memory Structure

- Figure shows the structure of a cache/main-memory system.
- Main memory consists of up to 2^n addressable words, with each word having a unique n-bit address.
- For mapping purposes, this memory is considered to consist of a number of fixed-length blocks of K words each.
- That is, there are $M = 2^n/K$ blocks in main memory.
- The cache consists of m blocks, called **lines**.
- Each line contains K words, plus a tag of a few bits.
- Each line also includes control bits (not shown), such as a bit to indicate whether the line has been modified since being loaded into the cache.
- The length of a line, not including tag and control bits, is the **line size**.

Cache/Main Memory Structure

- The line size may be as small as 32 bits, with each "word" being a single byte; in this case the line size is 4 bytes.
- The number of lines is considerably less than the number of main memory blocks ($m << M$).
- At any time, some subset of the blocks of memory resides in lines in the cache.
- If a word in a block of memory is read, that block is transferred to one of the lines of the cache.
- Because there are more blocks than lines, an individual line cannot be uniquely and permanently dedicated to a particular block.
- Thus, each line includes a **tag** that identifies which particular block is currently being stored.
- The tag is usually a portion of the main memory address.

Cache Read Operation

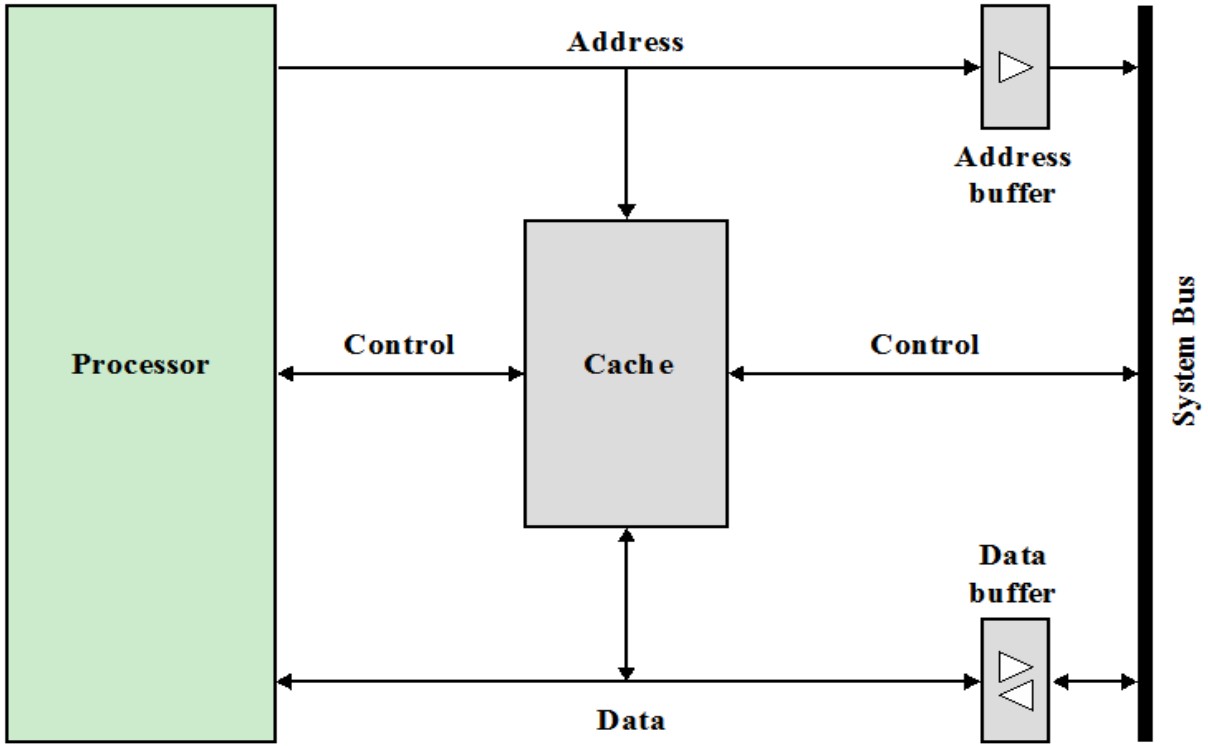


- Figure shows the read operation.
- The processor generates the read address (RA) of a word to be read.
- If the word is contained in the cache, it is delivered to the processor.
- Otherwise, the block containing that word is loaded into the cache, and the word is delivered to the processor

Cache Performance:

- When the processor needs to read or write a location in main memory, it first checks for a corresponding entry in the cache.
- If the processor finds that the memory location is in the cache, a **cache hit** has occurred and data is read from cache
- If the processor **does not** find the memory location in the cache, a **cache miss** has occurred.
- For a cache miss, the cache allocates a new entry and copies in data from main memory, then the request is fulfilled from the contents of the cache.
- The performance of cache memory is frequently measured in terms of a quantity called **Hit ratio.**

Typical Cache Organization



- In this organization, the cache connects to the processor via data, control, and address lines.
- The data and address lines also attach to data and address buffers, which attach to a system bus from which main memory is reached.

Typical Cache Organization

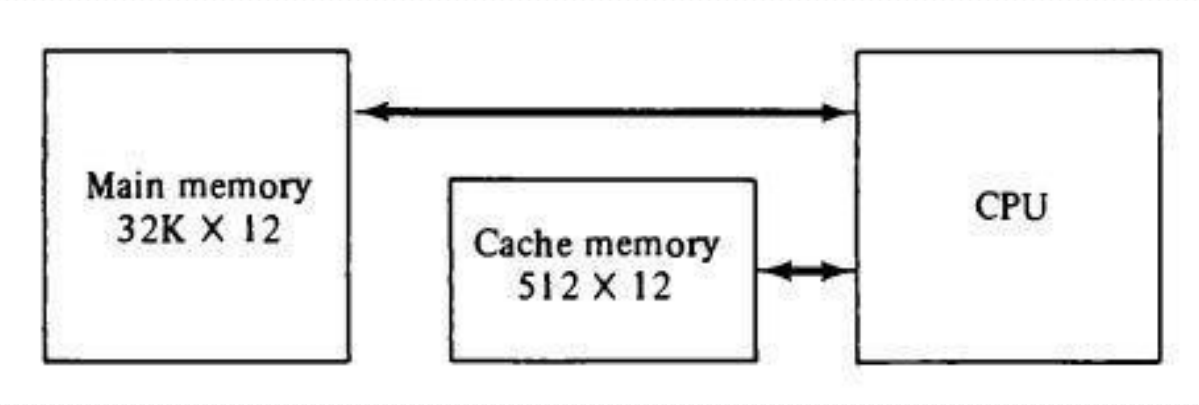
- When a cache hit occurs, the data and address buffers are disabled and communication is only between processor and cache, with no system bus traffic.
- When a cache miss occurs, the desired address is loaded onto the system bus and the data are returned through the data buffer to both the cache and the processor.
- In other organizations, the cache is physically interposed between the processor and the main memory for all data, address, and control lines.
- In this latter case, for a cache miss, the desired word is first read into the cache and then transferred from cache to processor.

Cache mapping

- The process of transferring the data from main memory to cache is known as mapping process. There are three types of cache mapping techniques:
  - Associative mapping
  - Direct mapping
  - Set-associative mapping
- For these three mapping procedures consider the example of a memory organization as shown in Figure

Cache mapping

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



Associative mapping

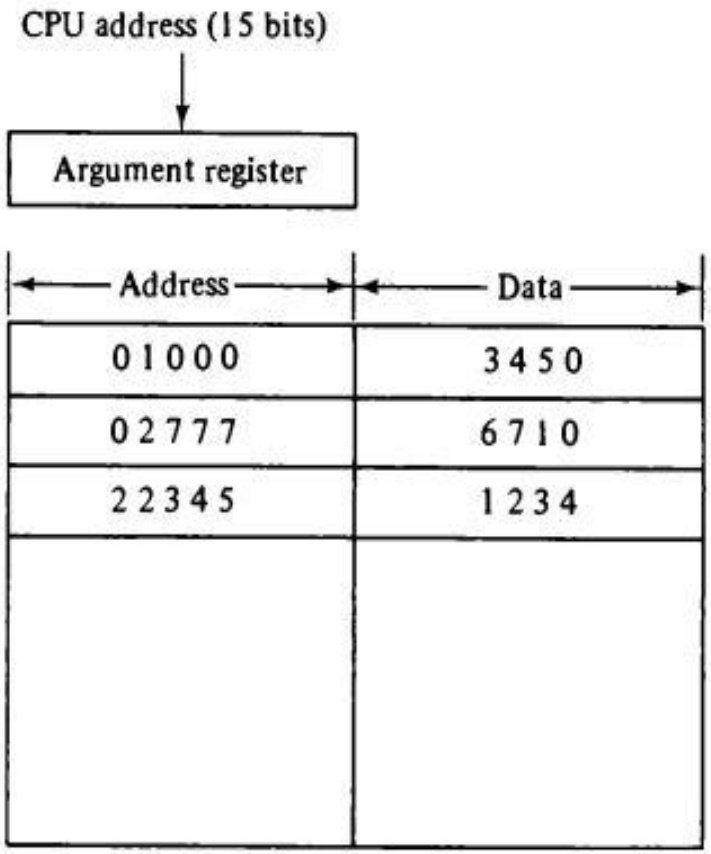
- The fastest and most flexible cache organization uses an associative memory.
- In associative mapping cashes are made of associative memory.
- Associative memory is used to store both the address and content of memory word.
- It permits any location in cache to store any word from main memory.
- That is it enables any word from main memory at any place in cache memory.

Associative mapping

- For example Main Memory size = 32K×12 = 2^{15}× 12 i.e. 15 Address lines and 12 data lines.
- Cache Memory size = 512 × 12 = 2^{9}× 12 i.e. 9 Address lines and 12 data lines.

- The address value of 15 bits is shown as a five-digit octal number and its corresponding 12 –bit word is shown as a four-digit octal number.

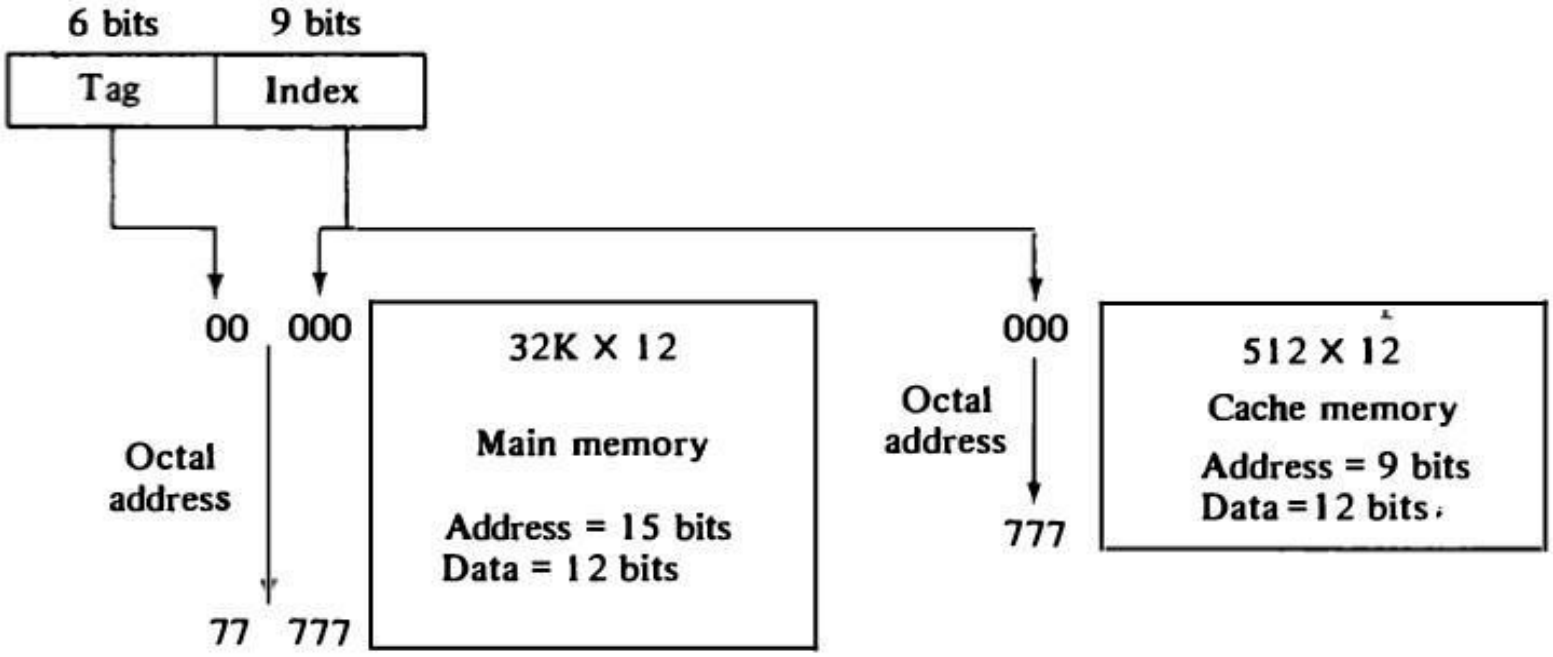
Associative mapping



- The diagram shows three words presently stored in the cache.
- The address value of 15 bits is shown as a five digit octal number and its corresponding 12 bit word is shown as a four digit octal number.
- A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.
- If the address is found, the corresponding 12 bit data is read and sent to the CPU.
- If the address is not matched, then CPU accesses the main memory for the data.
- The address and data pair is then transferred to the associative cache memory.
- If the cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in the cache.

Direct Mapping

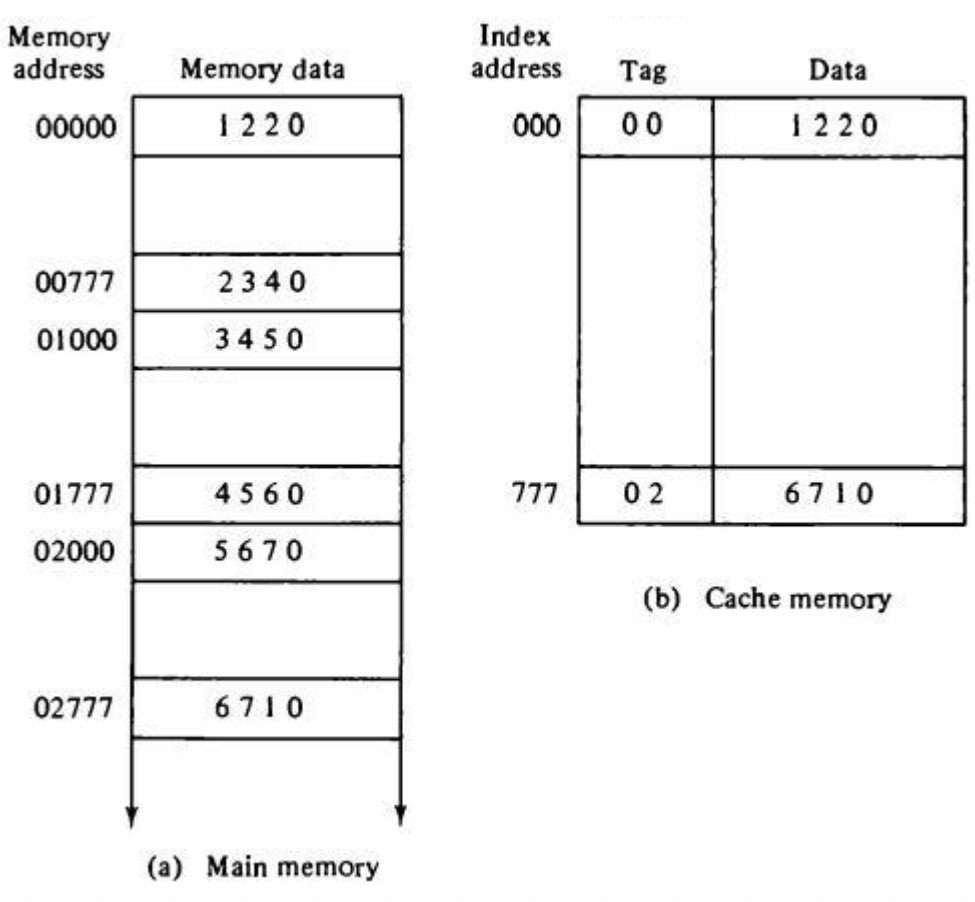
- Associative memories are expensive compared to random-access memories because of the added matching logic associated with each cell. So direct mapping is used.
- Direct mapping maps each block of main memory into only one possible cache line.
- That is it assigns each memory block to a specific line in the cache.
- The CPU address of 15 bits is divided into two fields.
- The nine least significant bits constitute the *index field and the remaining six bits form the tag field.*



Direct Mapping

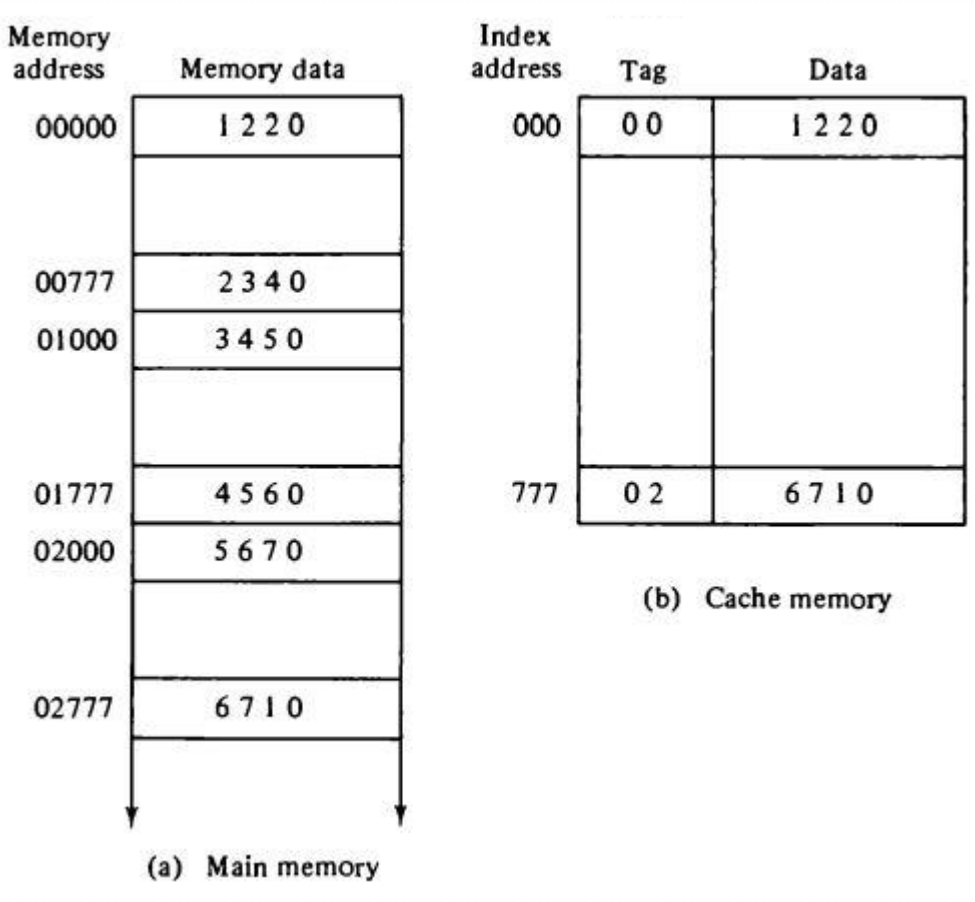
- The number of bits in the index field is equal to the number of address bits required to access the cache memory.
- The number of bits for tag field is the number of bits required to represent the address of Main Memory minus number of bits required to represent the address of Cache Memory.
- The n-bit memory address is divided into two fields as *k* bits for the *index* field and *n* - *k* bits for the *tag* field.
- The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.
- The internal organization of the words in the cache memory is as shown in figure.

Direct Mapping



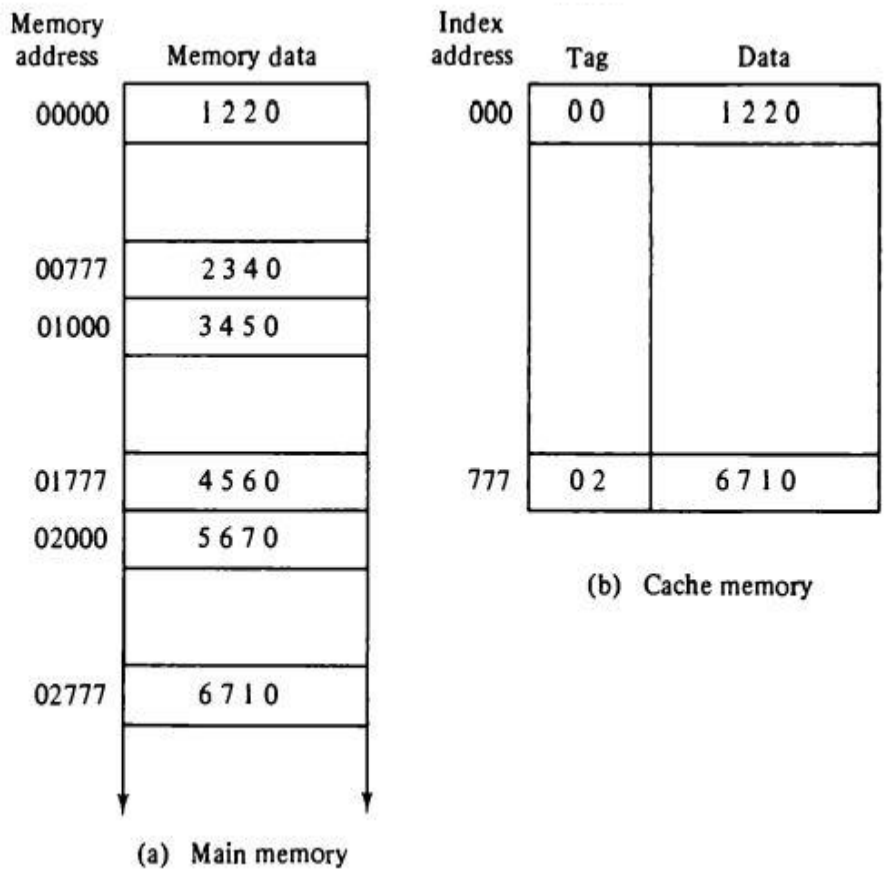
- Each word in cache consists of the data word and its associated tag.
- When a new word is first brought into the cache, the tag bits are stored along the data bits.
- When the CPU generates a memory request, the index field is used for the address to access the cache.
- The tag field of the CPU address is compared with the tag in the word read from the cache.
- if the two tags match, there is a hit and the desired data word is in cache.

Direct Mapping



- Eg: Suppose that the CPU now wants to access the word at address 02777.
- So the index value is 777 and tag value is 02.
- The index value 777 is used as address to access the cache memory.
- In the figure, index address 777 is available, and then tag value 02 is compared with tag value in the cache memory associated with index address 777.
- Here tag value of CPU address is match with tag value of cache with index address also.
- So desired data is available in cache memory.
- So it is hit operation.

Direct Mapping



- Eg: Suppose that the CPU now wants to access the word at address 01777.
- So the index value is 777 and tag value is 01.
- The index value 777 is used as address to access the cache memory.
- In the figure, index address 777 is available, and then tag value 01 is compared with tag value in the cache memory associated with index address 777.
- Here tag value of CPU address 01 is not match with tag value of cache memory is 02.
- So desired data was not available in cache memory.
- So it is *miss* operation.
- Whenever a miss operation was occur, the required word is read from main memory.
- It is then stored in the cache together with the new tag, replacing the previous value.

Direct Mapping

- The disadvantage of direct mapping is that the hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly.
- However this possibility is minimized by the fact that such words are far apart in the address range.(for e.g. 512 locations)
- That is two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

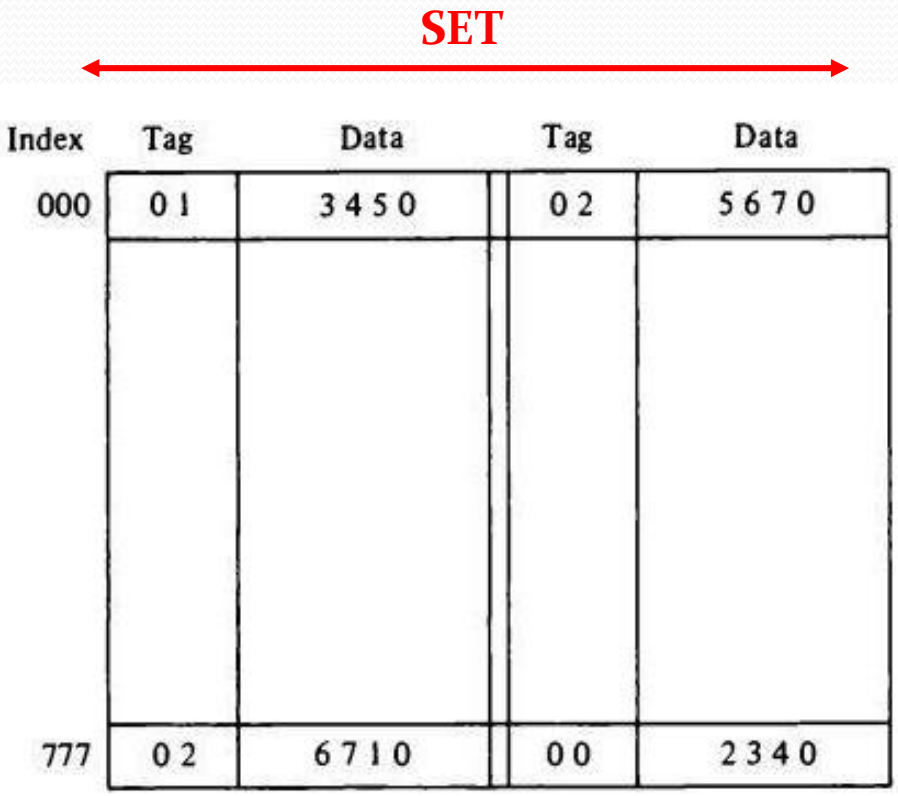
Set-Associative Mapping

- To overcome the disadvantage of direct mapping, i.e. two words with the same index in their address but with different tag values cannot reside in cache memory at the same time, Set- associative mapping is proposed.
- Here each word of cache can store two or more words stored under the same index address, creating a **Set.**
- Each data word is stored together with its tag.
- The number of tag-data items in one word is said to form a set.
- An example of a set-associative cache organization for a set size of two is shown in Figure

Set-Associative Mapping

- Each index address refers to two data words and their associated tags.
- Each tag requires six bits and each data word has 12 bits, so the word length is 2(6 + 12) = 36 bits.
- An index address of nine bits can accommodate 512 words.
- Thus, the size of cache memory is 512*36.
- It can accommodate 1024 words of main memory since each word of cache contains two data words.

Set-Associative Mapping



- The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000 with different tag values 01 and 02 respectively.
- Similarly, the words at addresses 02777 and 00777 of main memory are stored in cache memory at index address 777 with different tag values 01 and 02 respectively.
- When the CPU generates a memory request, the index value of the address is used to access the cache.
- The tag field of the CPU address is then compared with both tags in the cache. If tags are match then it is a *hit* operation.
- If the tag field of the CPU address and tags in the cache are not matched then it is a *miss* Operation.
- When a miss occurs in a set-associative cache and the set is full, it is necessary to replace one of the tag-data items with a new value.

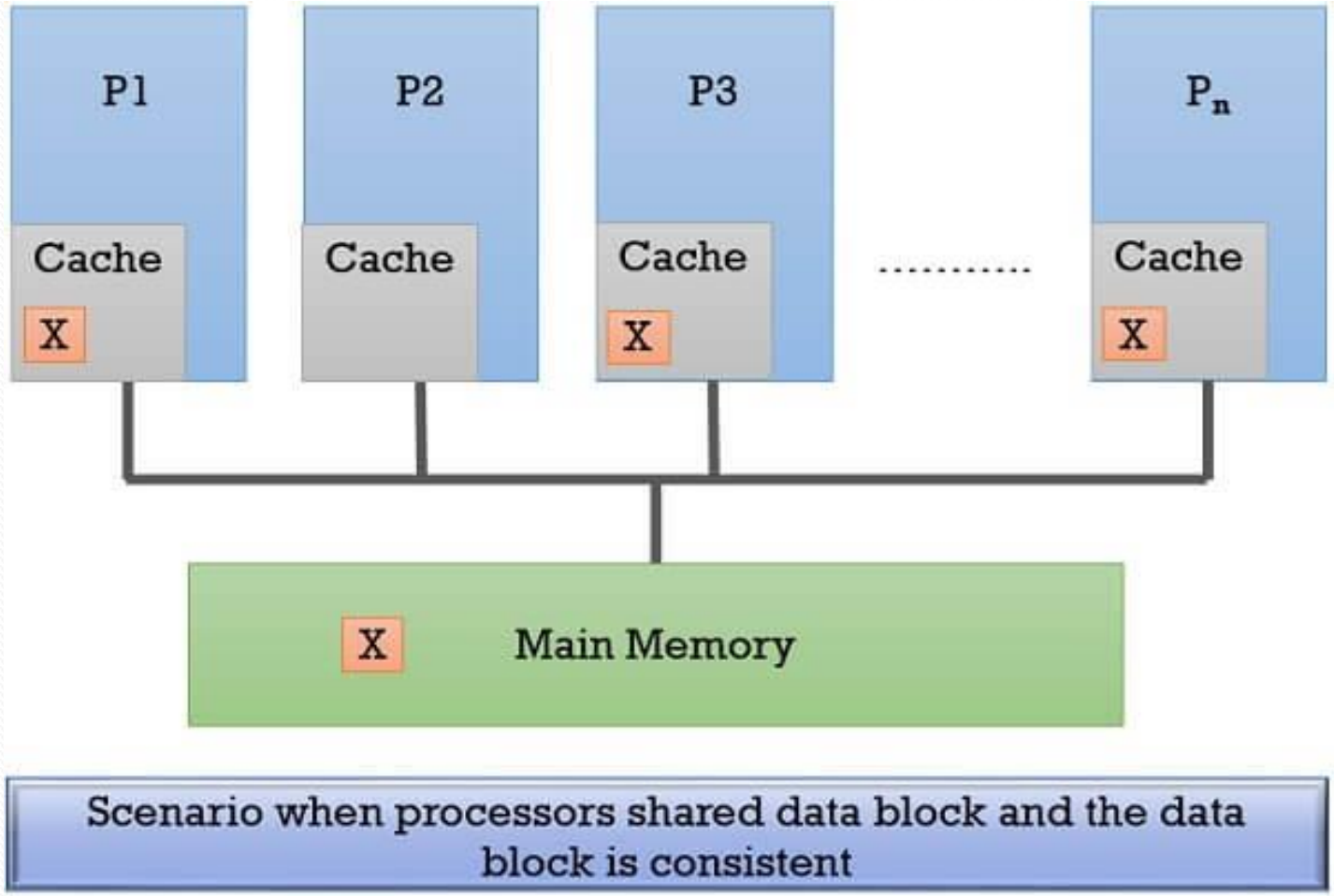
Writing into Cache

- An important aspect of cache organization is concerned with memory write requests.
- There are two ways for writing into Cache.
- The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache memory being updated in parallel.
- This is called the *write-through* method.
- This method has the advantage that main memory always contains the same data as the cache.
- The second procedure is called the write-back method.
- In this method only the cache location is updated during a write operation.
- The location is then marked by a flag so that later when the word is removed from the cache it is updated into main memory.

Cache Coherence

- In a multiprocessor environment, all the processors in the system share the **main memory** via a **bus**.
- Now, keeping a common cache for all the processors will enhance the size of the cache thereby slowing down the performance of the system.
- For better performance, each processor implements its own **cache**.
- Processors may share the same data block by keeping a copy of this data block in their cache.
- The figure below shows how processors P1, P3 & Pn have the copy of shared data block X of main memory in their caches.

Cache Coherence

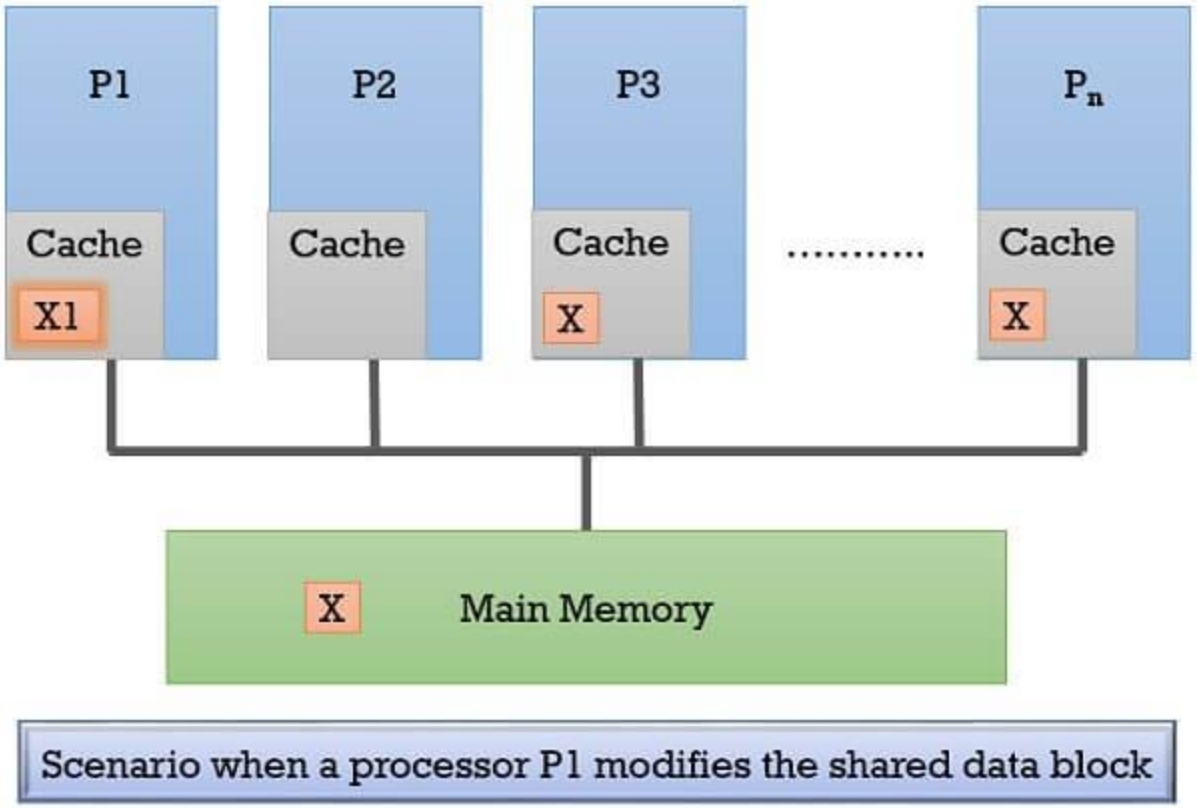


Scenario when processors shared data block and the data block is consistent

Cache Coherence

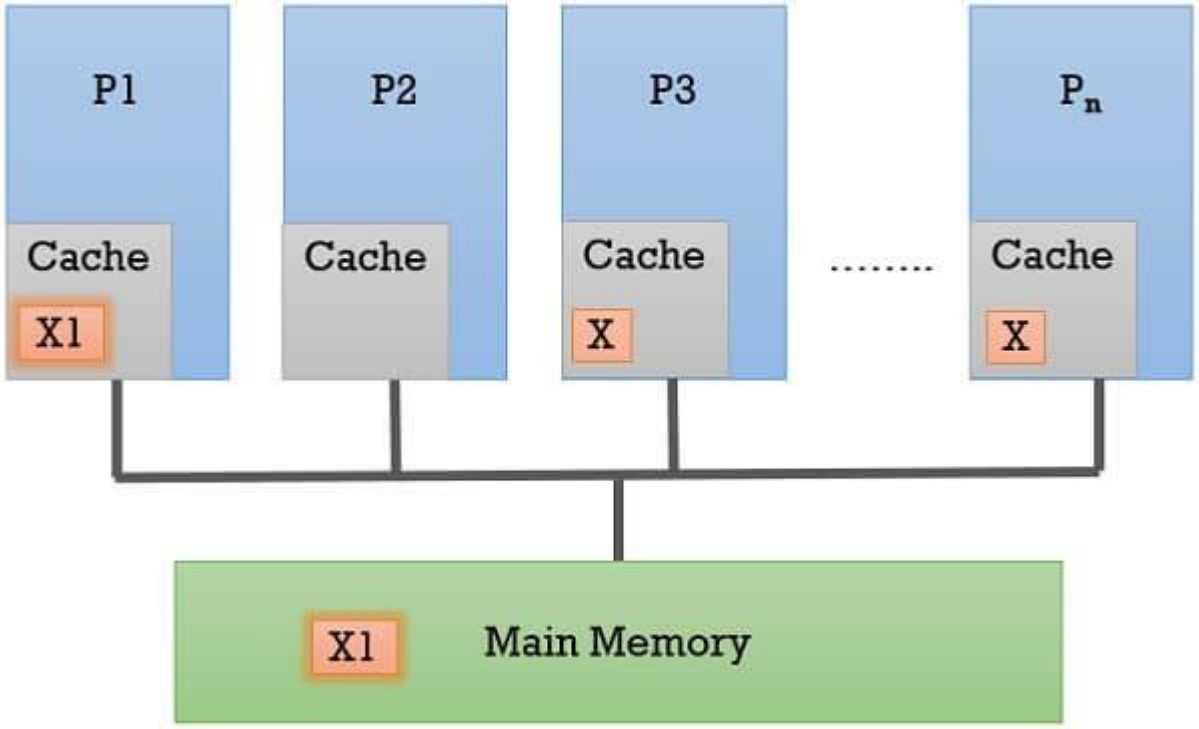
- In case, the processor P1 modifies the copy of shared memory block X present in its cache.
- It would result in **data inconsistency**.
- As the processor **P1** will have the **modified copy** of the shared memory block i.e. X1.
- But, the **main memory** and other processors' **cache** will have the **old copy** of the shared memory block X.
- And this problem is the **cache coherence problem**.
- The figure below shows the cache coherence problem in a multiprocessing environment.

Cache Coherence



Cache Coherence Protocols

1. Write-Through Protocol
   - In write-through protocol when a processor modifies a data block in its cache, it immediately **updates the main memory** with the new copy of the same data block.
   - So, the main memory here always has consistent data.



1. Write-Through Protocol

- The write-through protocols have two versions and those are:
  - Updating Values in Other Caches
  - Invalidating Values in Other Caches.

- **Updating Values**
  - Whenever a processor modifies a shared data block in its cache, it immediately updates the same data block in the main memory.
  - It also broadcast the modified data to all the other processors in the system.
  - When the other processors in the system receive the broadcasted modified data they modify the content of that data block as specified in broadcasted data.
- **Invalidating Values**
  - Whenever a processor modifies a data block present in its cache memory, it immediately updates the same data block in the main memory.
  - The processor modifying the data block broadcast request to other processors present in the system to **invalidate the copies of the same data block** in their caches.

2. Write-Back Protocol

- This protocol permits the processor to **modify a data block** only if it acquires **ownership**.
- Initially, the **memory** is the **owner** of all the data blocks and it retains that ownership.
- when a processor reads a data block and sites its copy in its cache and wants to **modify** a data block in its cache it has to confirm that it is an **exclusive owner** of that data block.
- For this, it has to first **invalidate** the copies of this data block in the **other caches** by broadcasting an invalidating request to all processors.
- Once it has become the exclusive owner, it can modify the data block.

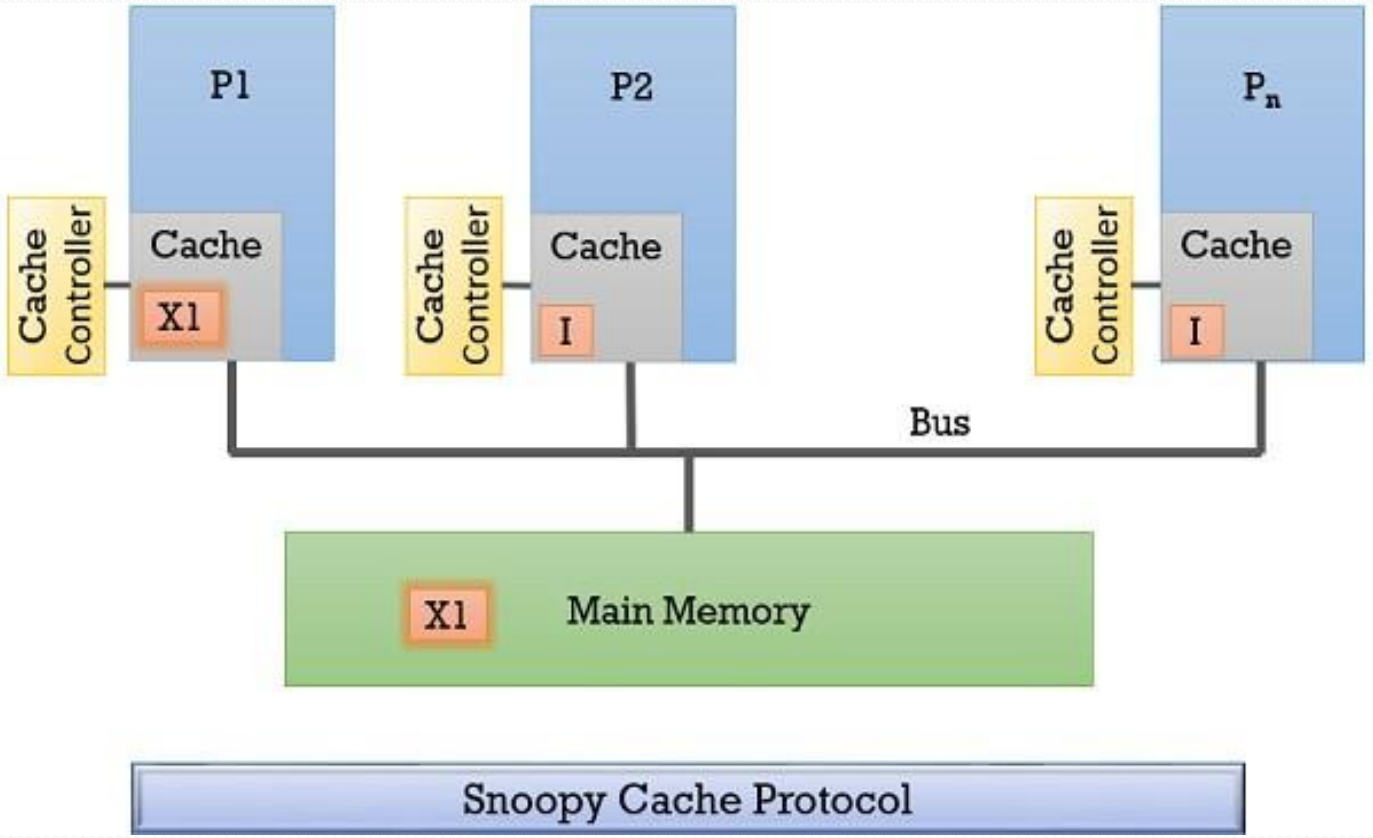
2. Write-Back Protocol

- If any processor wants to **read** this modified data block it has to send the **request** to the **current owner processor** of that data block.
- The owner forwards the data to the **requesting processor** and to the **main memory**.
- The **main memory** updates the content of the data block that has been modified and **reacquires its ownership** again over the data block.
- If any processor requires this data block it will be serviced by the main memory.

3. Snoopy Protocol

- In the multiprocessor environment, all the processors are connected to memory modules via a **single bus**.
- The transaction between the processors and the memory module i.e. read, write, invalidate request for the data block occurs via bus.
- If we implement the **cache controller** to every processor's cache in the system, it will **snoop** all the **transactions over the bus** and perform the appropriate action.
- So, we can say that the Snoopy protocol is the **hardware solution** to the cache coherence problem.
- It is used for small multiprocessor environments as the large shared-memory multiprocessors are connected via the interconnection network.

3. Snoopy Protocol



States of Memory Block in Cache memory

- As cache memory is subdivided into a number of blocks.
- And whenever a processor requires a data block it first checks it in its own cache memory.
- If it is not there the data is retrieved from the main memory and a copy of it is placed in the cache block.
- Now to maintain the cache coherency the cache controller maintain some information to keep the caches of other processors in the system synchronized, while a processor is modifying its copy of data that is also shared by other processors in the system.
- So the cache controller maintains the state for every cache block of the cache memory which helps in maintaining the coherency.

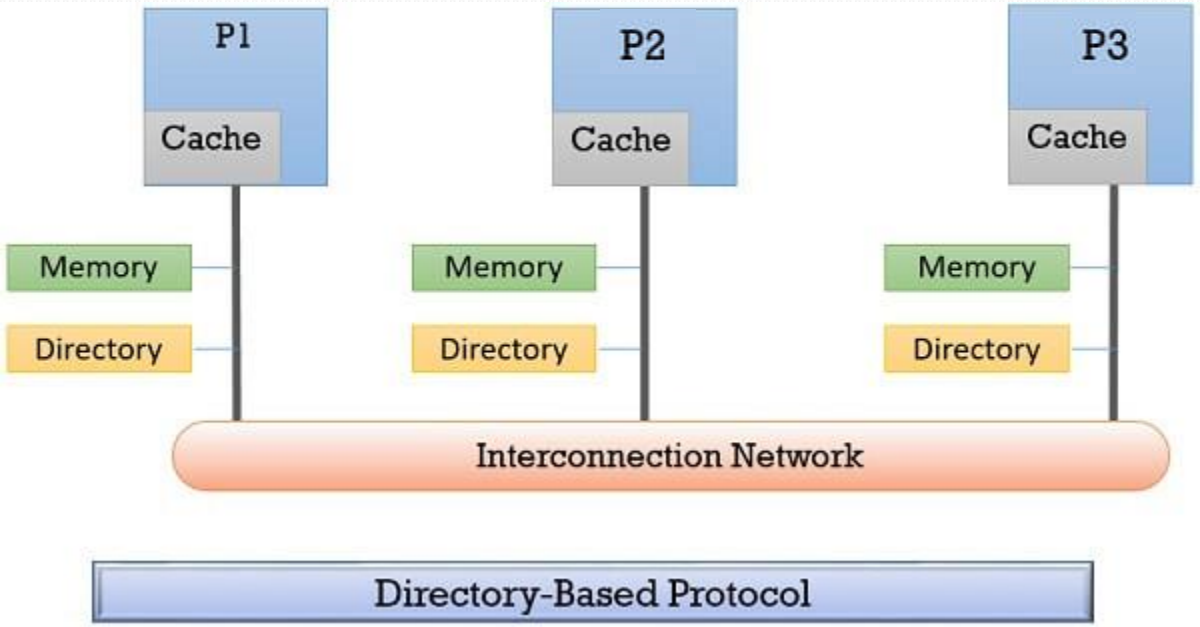
States of Memory Block in Cache memory

- **Shared (S):**
  - A data block in the main memory is shared by many processors in the system and all the processors have a **valid copy of the data block** in their caches.
- **Exclusive (E):**
  - When the processor wants to modify a data block in its cache, it broadcast the request to invalidate the copy of the same data block in other caches.
  - So, the data block to be modified is now only with the **processor that wishes to modify** it and with the **main memory**. Here, the processor is the exclusive owner of the data block.
- **Modify (M):**
  - The data block in a cache is **modified** (different from main memory) and is available only in this cache.
  - The processor modifying the data block is the owner of that data block.
- **Invalid (I):** The cache has a data block that does **not have valid data**.

4. Directory-Based Cache Coherence Protocol

- Directory-Based cache coherence protocol is a **hardware solution** to the cache coherence problem.
- It is implemented in a **large multiprocessor system** where the shared memory and processors are connected using the **interconnection network**.
- The **directories** are implemented in **each memory module** of the multiprocessors system.
- These directories keep the record of all the actions taken to each data block i.e. whether a data block in the cache of a processor is invalid, or is being modified, or is in the shared state.
- Due to its **cost** and **complexity** directory-based cache coherence protocols are implemented only to large multiprocessors systems.

4. Directory-Based Cache Coherence Protocol



Directory-Based Cache Coherence Protocol

- Cache directories keep track of status of all cache blocks.
- Global state information about the contents of various local cache is stored.
- if any entry is changed the directory either updates it or invalidates the other caches with that entry.
- It is used for distributed systems where processors are having their own cache, directory and memory.
- E.g.: if P1 wants A then it sees directory to get A, if A is with P3 then P3 sends A and makes A as shared in its memory.

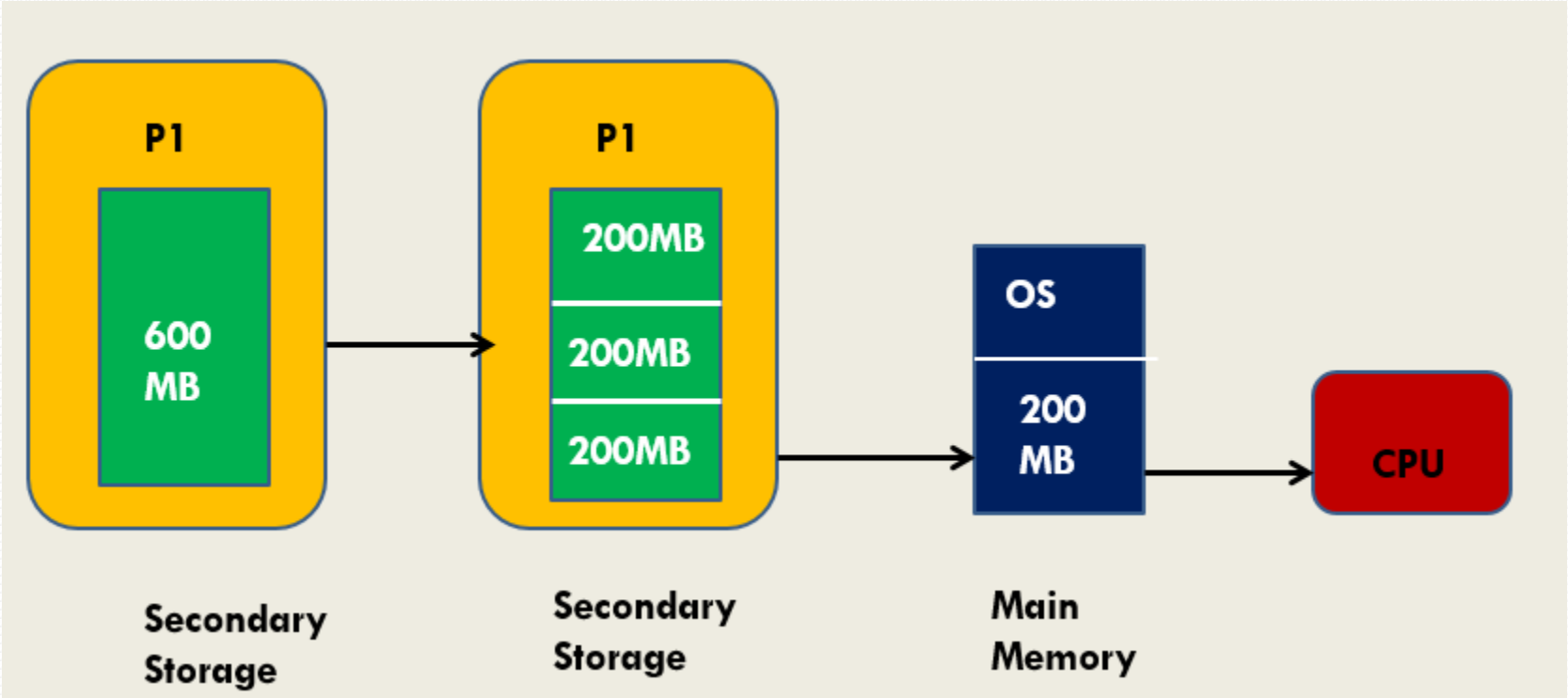
Virtual Memory

- Virtual memory is a technique that allows the execution of process having size larger than physical memory.
- The virtual memory is the memory that does not exist physically but still it appears to the programmer that it is available in the system.
- The concept of virtual memory is implemented using physical memory and the secondary memory.

Virtual Memory

- Initially the data and programs are stored in secondary memory and during execution the part of the program or data is loaded from secondary memory to the main memory.
- After the execution of the first part , the next part required for further execution is again loaded from secondary memory.
- While the result of the executed part are stored back in to the secondary memory.
- This continues till the complete program is executed.

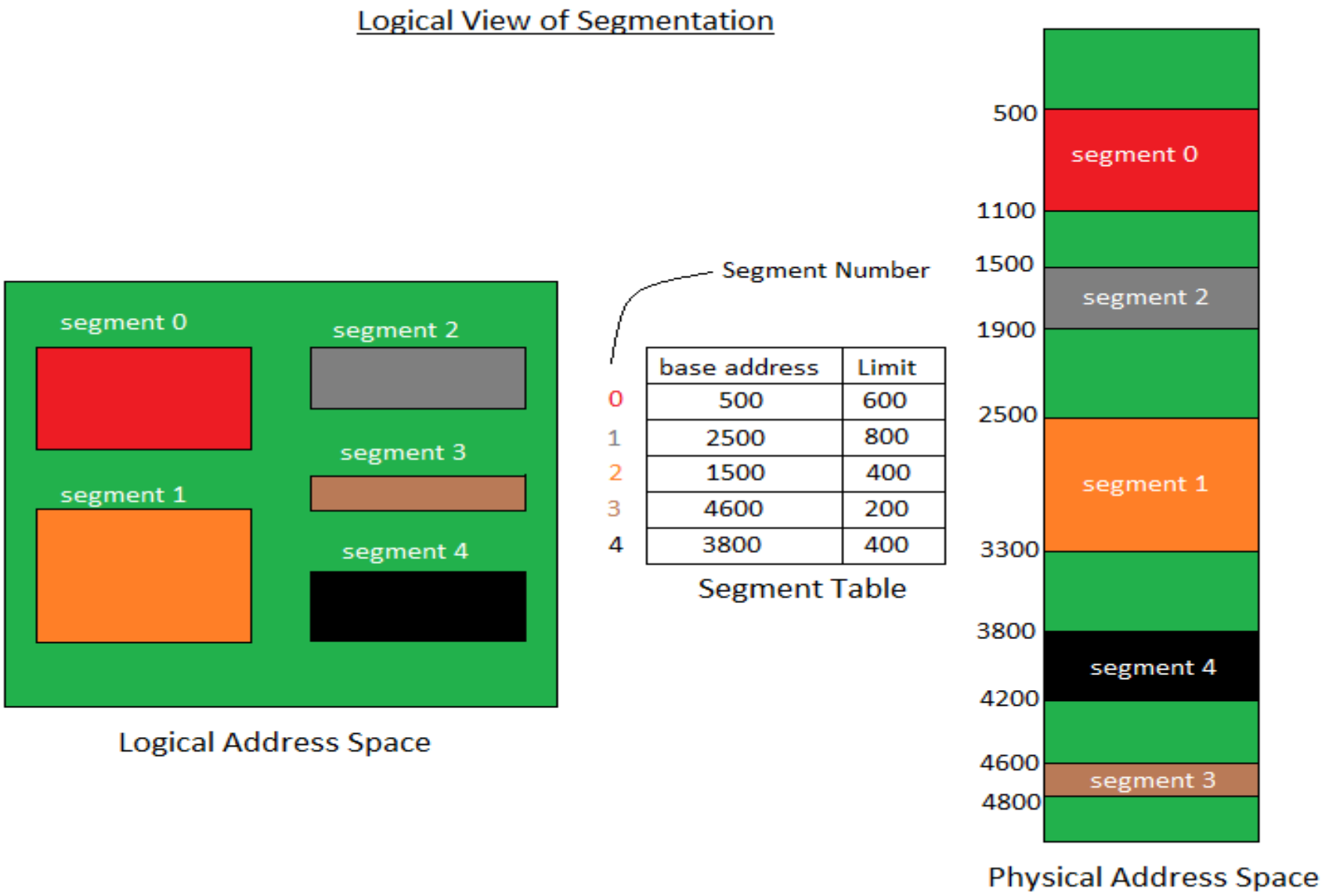
Virtual Memory



Segmentation

- Segmentation is a memory-management scheme in which the memory is divided in to variable size parts.
- Each part is known as segment which can be allocated to a process.
- A logical address space is a collection of segments
- Each segment has a name and a length.
- The details about each segment are stored in a table called as segment table.
- Segment table is stored in one (or many) of the segments.

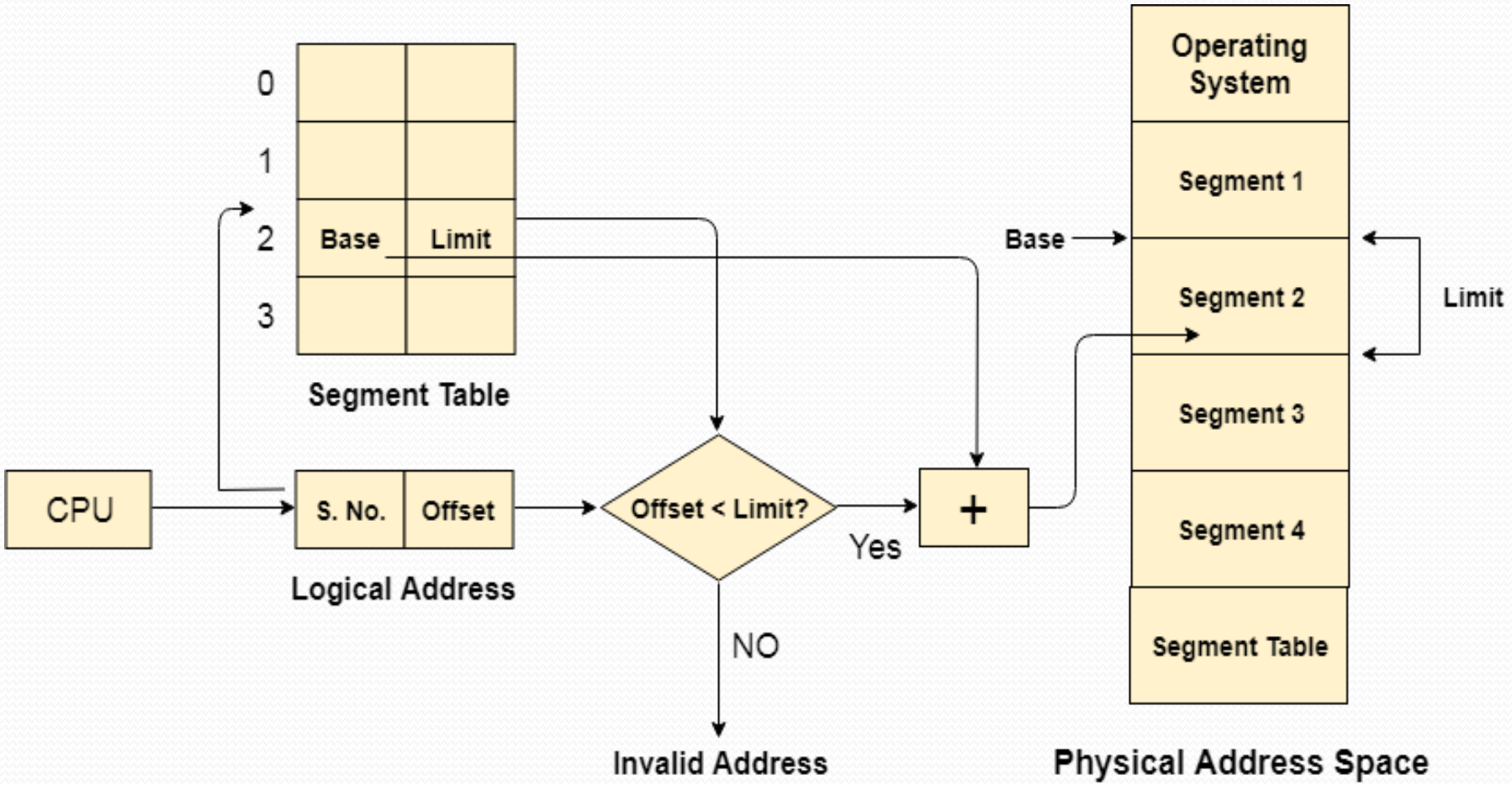
Segmentation



Segmentation

- Segment table contains mainly two information about segment:
- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Limit:** It specifies the length of the segment.

Translation of Two dimensional Logical Address to one dimensional Physical Address



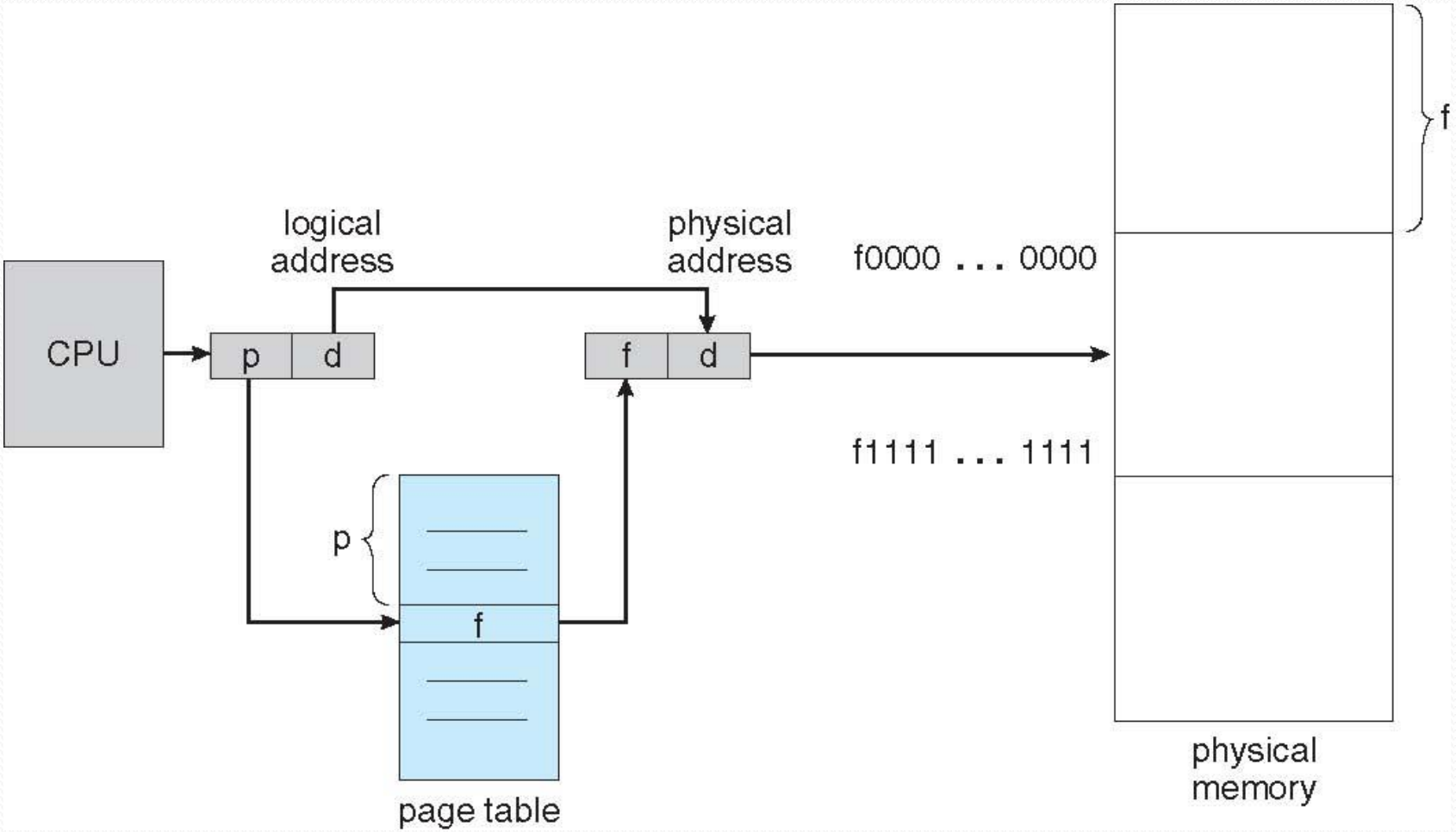
Segmentation

- CPU generates a logical address which contains two parts:
  1. Segment Number (p)
  2. Offset (d)
- The Segment number is mapped to the segment table.
- The limit of the respective segment is compared with the offset.
- If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.
- In the case of valid address, the base address of the segment is added to the offset to get the physical address of actual word in the main memory.

Paging

- Paging is memory management scheme in which the physical memory is divided in to fixed sized blocks called ***frames.***
- ***logical memory is also divided*** into blocks of the same size called ***pages.***
- When a process is to be executed, its pages are loaded into available frames in physical memory.
- The hardware support for paging is shown in fig.

Paging


CPU
logical address
p
d
physical address
f
d
f0000 . . . 0000
f1111 . . . 1111
f
p
f
page table
physical memory

Paging

- As shown in fig. every address generated by the CPU is divided into two parts:
  1) ***page number (p) and 2) page offset (d).***
- The page number is used as an index into a ***page table.***
- ***The page table*** contains the base address of each page in physical memory.
- This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.

Page Fault:

- If the CPU request for the page and that page is not available in physical memory then the page fault occurs.
- In this situation the required page is loaded from secondary memory into physical memory.
- This is called swapping.
- But to load the new page in to physical memory, the space must be created in physical memory by removing the pages that are not required by CPU this process is called unswapping.

Page Replacement

- When the page fault occurs, the OS has to select a page to remove from memory to make space for the page that is to be brought in.
- This is known as page replacement.
- There are various page replacement techniques:

1) FIFO page replacement

2) Optimal page replacement

3) Least recently used page replacement

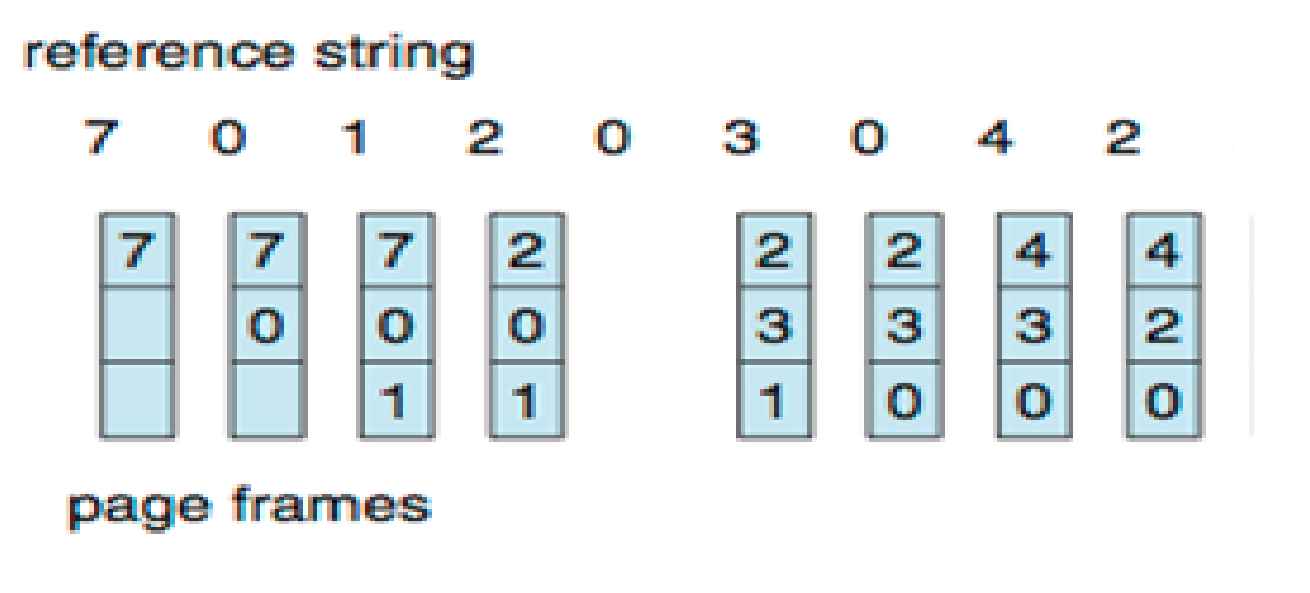
4) Not recently used page replacement.

First In First Out (FIFO) Page Replacement Algorithm

- The simplest page replacement algorithm is a first-in, first-out (FIFO) algorithm.
- In this algorithm the time when that page was loaded into memory is considered.
- When a page must be replaced, the oldest page is chosen.
- The FIFO queue is used to hold all pages in memory.
- A page is replaced at the head of the queue.
- When a page is brought into memory, it is inserted at the tail of the queue.

First In First Out (FIFO) Page Replacement Algorithm

Reference string: **7,0,1,2,0,3,0,4,2**



First In First Out (FIFO) Page Replacement Algorithm

- For our example reference string, our three frames are initially empty.
- The first three references (7, 0, 1) cause page faults and are brought into these empty frames.
- The next reference (2) and it replaces page 7, because page 7 was brought in first.
- The next reference is 0 and 0 is already in memory, so there is no page fault .
- The next reference to 3 which is to be replaced with 0, as it was brought before 2 and 1.
- The next reference is 0, which replaces 1 after that is 4 which replaces 2 and then 2 replaces 3.

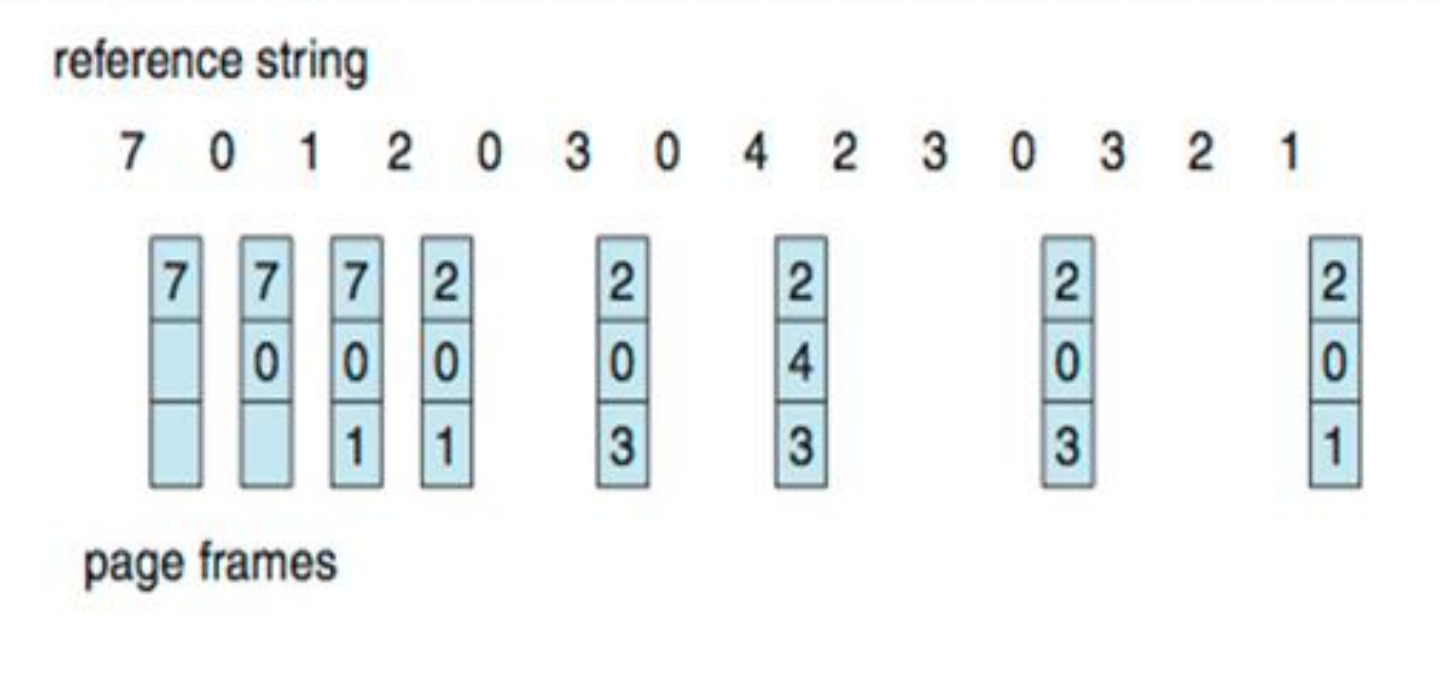
First In First Out (FIFO) Page Replacement Algorithm

- Thus there are 8 page faults.
- The FIFO page replacement algorithm is easy to understand and program.
- However, its performance is not always good.
- It increases the rate of page fault.

Optimal Page Replacement Algorithm

- This algorithm that has the lowest page-fault rate of all algorithms.
- This algorithm *Replace the page that will not be used for the longest period of time.*

Optimal Page Replacement Algorithm



Optimal Page Replacement Algorithm

- For example, consider the reference string: **7,0,1,2,0,3,0,4,2,3,0,3,2**
- Assume the 3 frames are empty and the first 3 references causes page fault and are stored in first three frames.
- The next reference is 2 which replaces 7 because it is not used for long period.
- The next reference is 0 which is already available after that is 3 which replaces 1

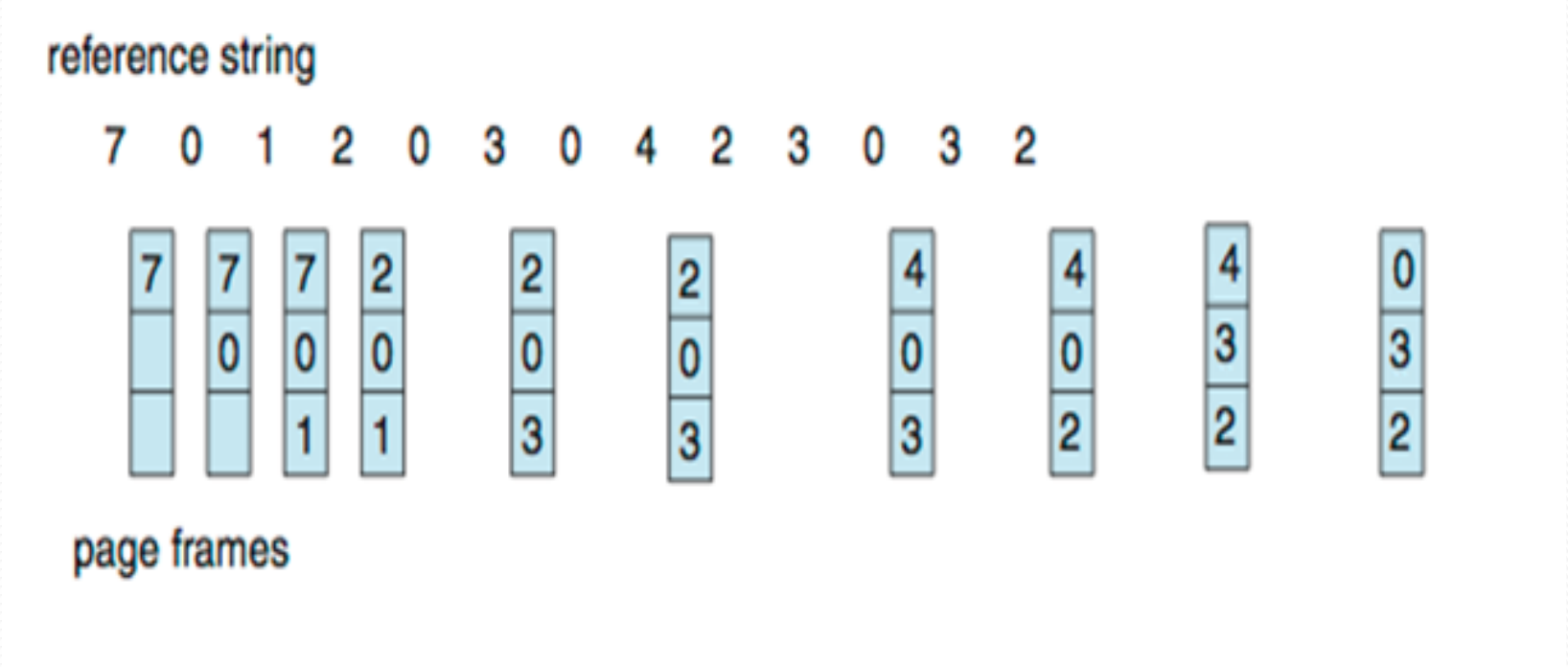
Optimal Page Replacement Algorithm

- The next reference is 0 which is already available, then next is 4 which replaces 0 and so on.
- The optimal page replacement algorithm is difficult to implement, because it requires future knowledge of the reference string.
- As a result, the optimal algorithm is used mainly for comparison studies.

Least Recently Used (LRU) Page Replacement Algorithm:

- In this algorithm *the page that* ***has not been used for the longest period of time is replaced.***
- LRU replacement associates with each page the time of that page's last use.
- When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.
- For example, consider the reference string: **7,0,1,2,0,3,0,4,2,3,0,3,2**

Least Recently Used (LRU) Page Replacement Algorithm:



Least Recently Used (LRU) Page Replacement Algorithm:

- For example, consider the reference string: **7,0,1,2,0,3,0,4,2,3,0,3,2**
- Assume the 3 frames are empty and the first 3 references causes page fault and are stored in first three frames.
- The next reference is 2 which replaces 7 because it is not used for long period.
- The next reference is 0 which is already available after that is 3 which replaces 1 and so on.

Least Recently Used (LRU) Page Replacement Algorithm:

- The major problem is how to implement LRU replacement.
- An LRU page replacement algorithm may require substantial hardware assistance.

Reference books

1. William Stallings- “Computer Organization and Architecture: Designing for Performance”, 11 Pearson Publication, 10th Edition, 2013.
2. John P. Hayes- “Computer Architecture and Organization”, McGraw-Hill, 1988.