Unit - I
Basics structure of computer

Computer Architecture: Computer Architecture deals with giving operational attributes of the computer or Processor to be specific. It deals with details like physical memory, ISA (Instruction Set Architecture) of the processor, the number of bits used to represent the data types, Input Output mechanism and technique for addressing memories.

Computer Organization: Computer Organization is realization of what is specified by the computer architecture .It deals with how operational attributes are linked together to meet the requirements specified by computer architecture. Some organizational attributes are hardware details, control signals, peripherals.

1. COMPUTER TYPES Classification based on Operating Principles Based on the operating principles, computers can be classified into one of the following types: -

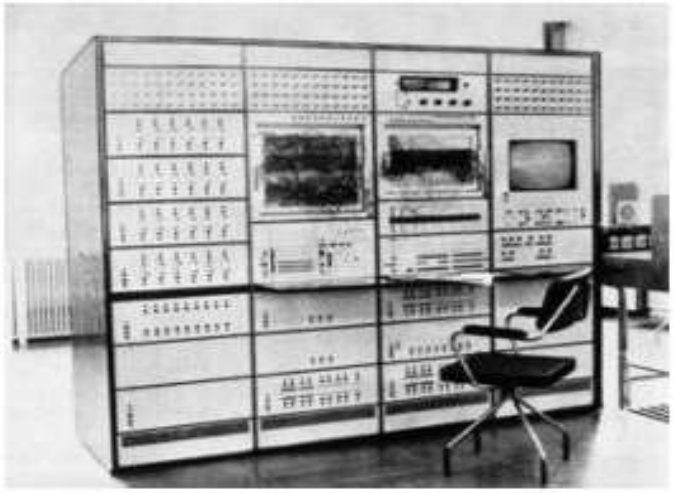
1) Digital Computers
2) Analog Computers
3) Hybrid Computers

Digital Computers: -

Operate essentially by counting. All quantities are expressed as discrete or numbers. Digital computers are useful for evaluating arithmetic expressions and manipulations of data (such as preparation of bills, ledgers, solution of simultaneous equations etc).

Analog Computers:- An analog computer is a form of computer that uses the continuously changeable aspects of physical phenomena such as electrical, mechanical, or hydraulic quantities to model the problem being solved. In contrast, digital computers represent varying quantities symbolically, as their numerical values change.

Hybrid Computers:- are computers that exhibit features of analog computers and digital computers. The digital component normally serves as the controller and provides logical operations, while the analog component normally serves as a solver of differential equations.



Classification digital Computer based on **size and Capability** Based on size and capability, computers are broadly classified into

Micro Computers(Personal Computer) A microcomputer is the smallest general purpose processing system. The older pc started 8 bit processor with speed of 3.7MB and current pc 64 bit processor with speed of 4.66 GB.
Examples: - IBM PCs, APPLE computers
Microcomputer can be classified into 2 types**: 1. Desktops 2. Portables**

The difference is portables can be used while travelling whereas desktops computers cannot be carried around.

The different portable computers are: -

1) Laptop
2) Notebooks
3) Palmtop (hand held)
4) Wearable computers

Laptop: - this computer is similar to a desktop computers but the size is smaller. They are expensive than desktop. The weight of laptop is around 3 to 5 kg.

Notebook: - These computers are as powerful as desktop but size of these computers are comparatively smaller than laptop and desktop. They weigh 2 to 3 kg. They are more costly than laptop.

Palmtop (Hand held): - They are also called as personal Digital Assistant (PDA). These computers are small in size. They can be held in hands. It is capable of doing word processing, spreadsheets and hand writing recognition, game playing, faxing and paging. These computers are not as powerful as desktop computers. Ex: - 3com palmV.

Wearable computer: - The size of this computer is very small so that it can be worn on the body. It has smaller processing power. It is used in the field of medicine. For example pace maker to correct the heart beats. Insulin meter to find the levels of insulin in the blood.

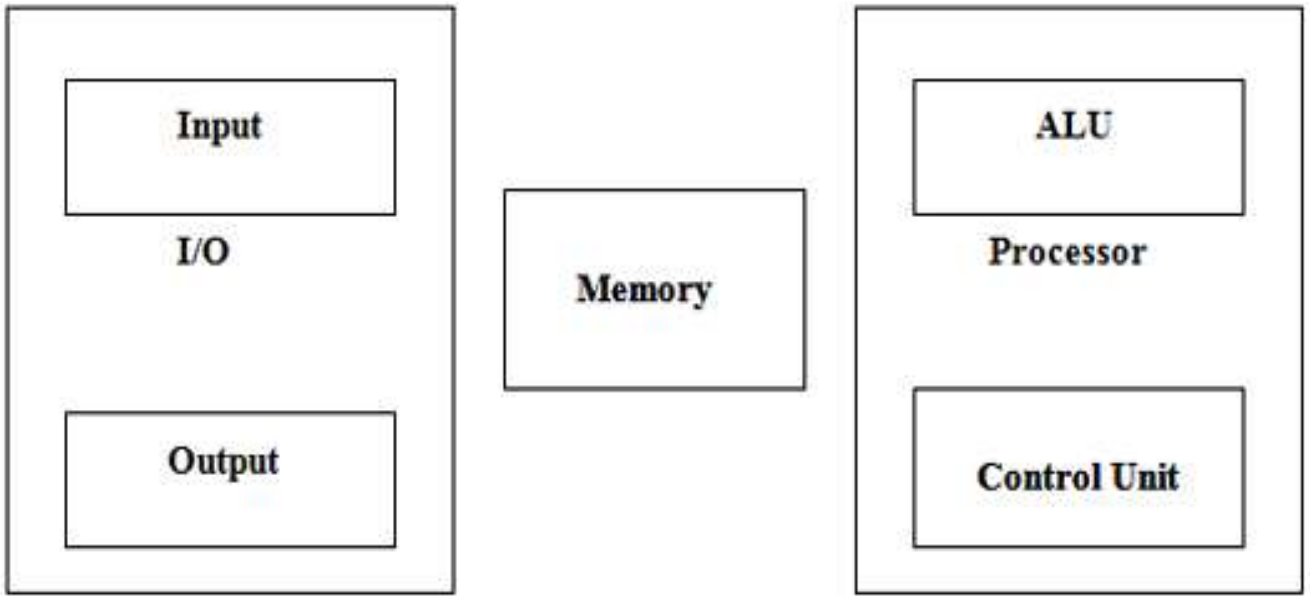
Work stations: - These have high resolution input/output (I/O) graphics capability, but with same dimensions as that of desktop computer. These are used in engineering applications of interactive design work.

Enterprise systems: - These are used for business data processing in medium to large corporations that require much more computing power and storage capacity than work stations. Internet associated with servers have become a dominant worldwide source of all types of information.

Super computers: - These are used for large scale numerical calculations required in the applications like weather forecasting etc.,

2. FUNCTIONAL UNIT-

A computer consists of five functionally independent main parts input, memory, arithmetic logic unit (ALU), output and control unit.



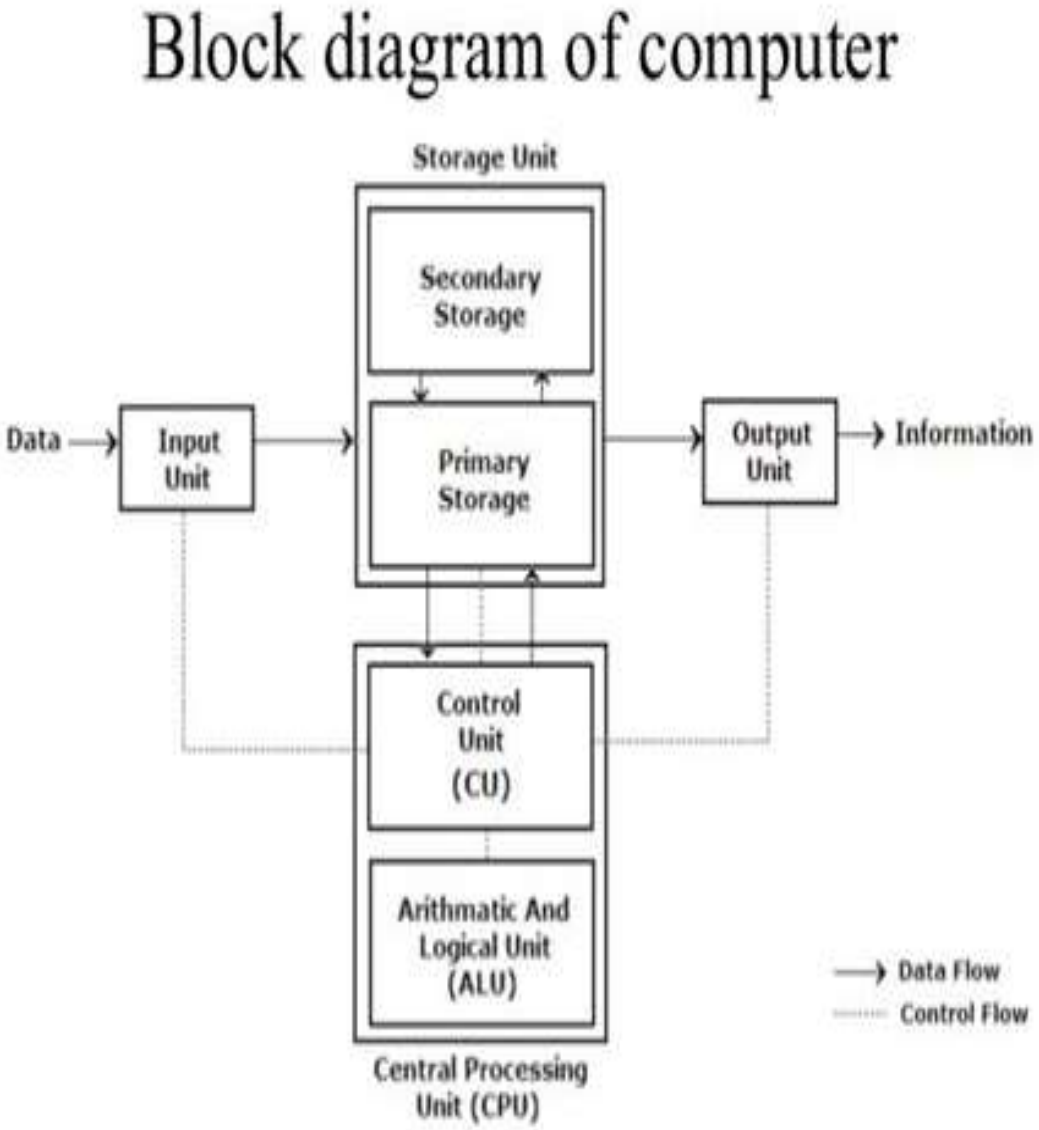
Functional units of computer

Input device accepts the coded information as source program i.e. high level language. This is either stored in the memory or immediately used by the processor to perform the desired operations.

This is either stored in the memory or immediately used by the processor to perform the desired operations.

The program stored in the memory determines the processing steps. Basically the computer converts one source program to an object program. i.e. into machine language.

Finally the results are sent to the outside world through output device. All of these actions are coordinated by the control unit.



Input unit: -

The source program/high level language program/coded information/simply data is fed to a computer through input devices keyboard is a most common type. Whenever a key is pressed, one corresponding word or number is translated into its equivalent binary code over a cable & fed either to memory or processor.

Joysticks, trackballs, mouse, scanners etc are other input devices.

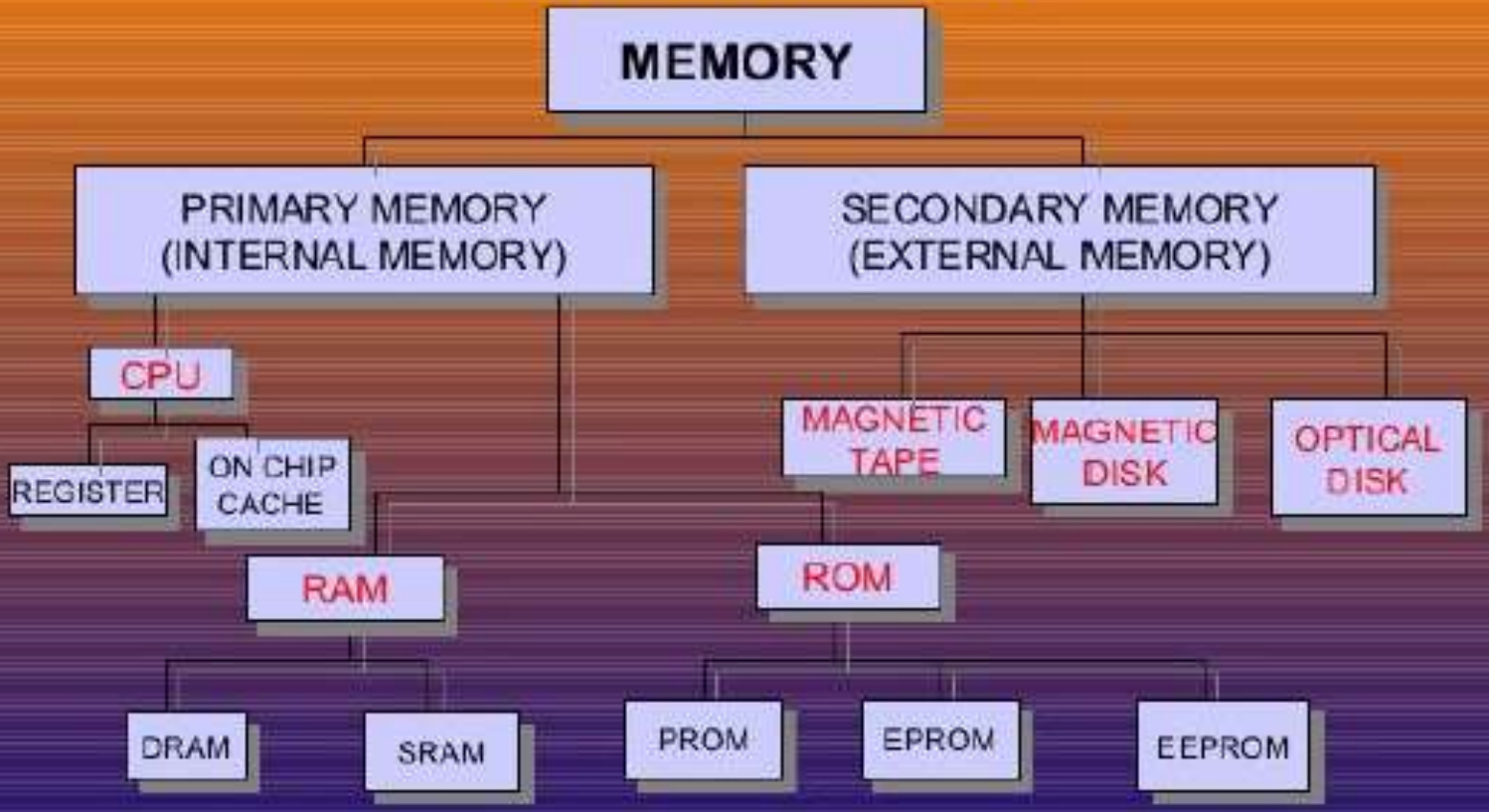
Memory unit: - Its function into store programs and data. It is basically to two types

1. Primary memory 2. Secondary memory

The number of bits in each word is known as word length. Word length refers to the number of bits processed by the CPU in one go. With modern general purpose computers, word size can be 16 bits to 64 bits.

The time required to access one word is called the memory access time. The small, fast, RAM units are called caches. They are tightly coupled with the processor and are often contained on the same IC chip to achieve high performance.

Types of memory



1. Primary memory: - Is the one exclusively associated with the processor and operates at the electronics speeds programs must be stored in this memory while they are being executed. The memory contains a large number of semiconductors storage cells.

- ✓ To provide easy access to a word in memory, a distinct address is associated with each word location. **Addresses are** numbers that identify memory location.
- ✓ Programs must reside in the memory during execution.
- ✓ Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random access memory (RAM) time.
- ✓ Memory which is only readable by the user and contents of which can't be altered is called read only memory (ROM) it contains operating system.

- ✓ Caches are the small fast RAM units, which are coupled with the processor and are
- ✓ often contained on the same IC chip to achieve high performance

2 Secondary memory: - Is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently.

Examples: - Magnetic disks & tapes, optical disks (ie CD-ROM's), floppies etc.,

Arithmetic logic unit (ALU):-

Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, etc. the operands are brought into the ALU from memory and stored in high speed storage elements called register. Then according to the instructions the operation is performed in the required sequence.

The control and the ALU are may times faster than other devices connected to a computer system. This enables a single processor to control a number of external devices such as key boards, displays, magnetic and optical disks, sensors and other mechanical controllers.

Output unit:-

These actually are the counterparts of input unit. Its basic function is to send the processed results to the outside world.

Examples:- Printer, speakers, monitor etc.

Control unit:-

It effectively is the nerve center that sends signals to other units and senses their states. The actual timing signals that govern the transfer of data between input unit, processor, memory and output unit are generated by the control unit.

3. BASIC OPERATIONAL CONCEPTS

To perform a given task an appropriate program consisting of a list of instructions is stored in the memory. Individual instructions are brought from the memory into the processor, which executes the specified operations. Data to be stored are also stored in the memory.

Examples: - Add R0, R1

This instruction adds the operand at memory location R0, to operand in register R1 & places the sum into register. This instruction requires the performance of several steps,

1. First the instruction is fetched from the memory into the processor.
2. The operand at R0 is fetched and added to the contents of R1
3. Finally the resulting sum is stored in the register R1

The preceding add instruction combines a memory access operation with an ALU Operations.

In some other type of computers, these two types of operations are performed by separate instructions for performance reasons.

Load R2, R1
Add R1, R0

Transfers between the memory and the processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals.
The data are then transferred to or from the memory.

Register:

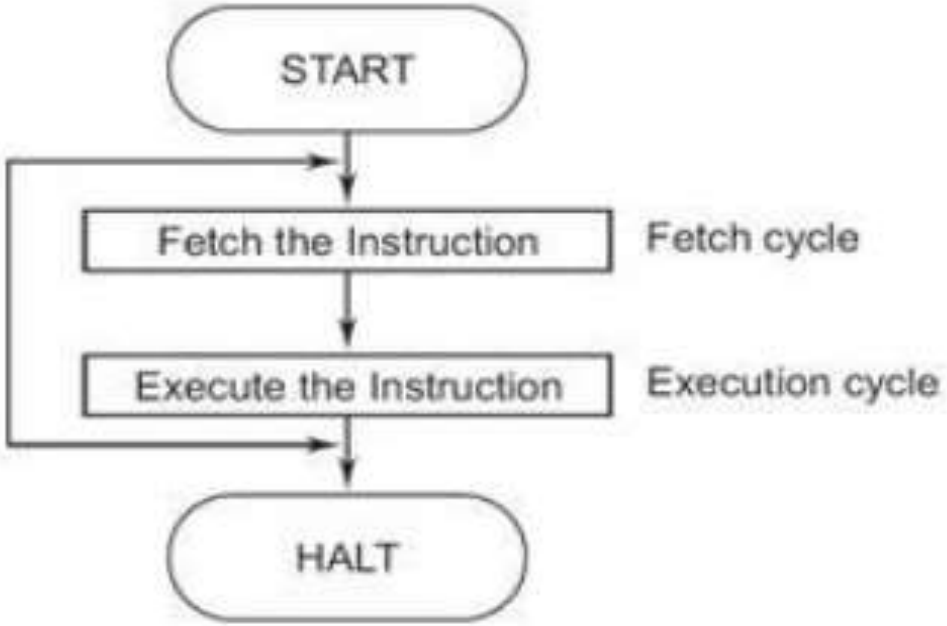
It is a special, high-speed storage area within the CPU. All data must be represented in a register before it can be processed.
For example, if two numbers are to be multiplied, both numbers must be in registers, and the result is also placed in a register.
(The register can contain the address of a memory location where data is stored rather than the actual data itself.)

Instruction Format:

Computer instructions are the basic components of a machine language program.
They are also known as macro operations, since each one is comprised of sequences of micro operations.

Instructions are encoded as binary instruction codes. Each instruction code contains of a operation code, or opcode, which designates the overall purpose of the instruction (e.g. add, subtract, move, input, etc.).

INSTRUCTION CYCLE:

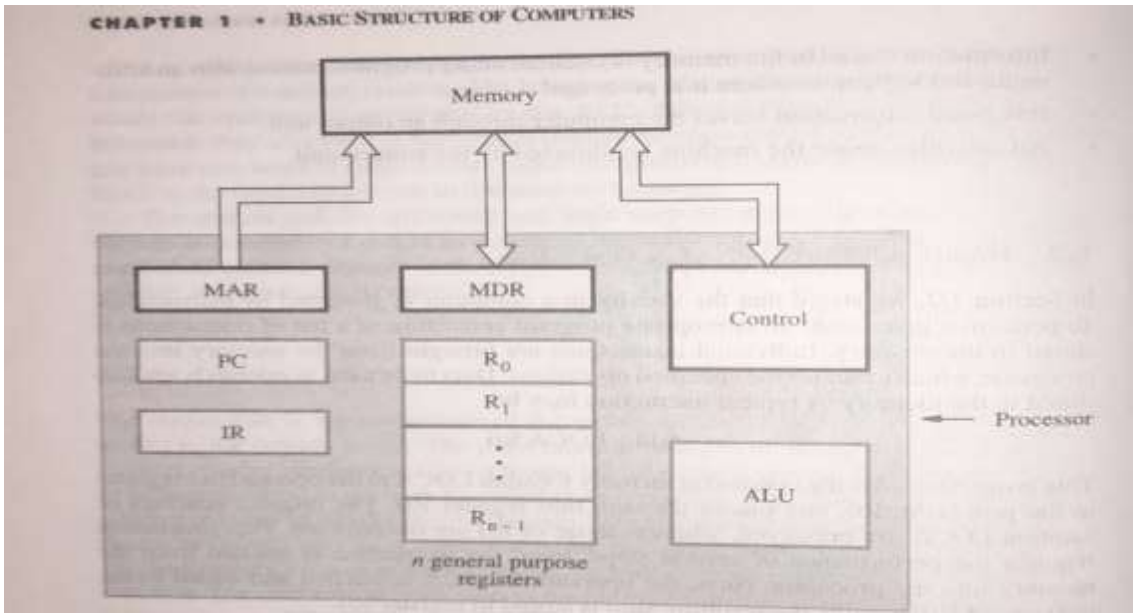


instruction register (IR):- Holds the instructions that are currently being executed. Its output is available for the control circuits which generates the timing signals that control the various processing elements in one execution of instruction.

The program counter PC:-
This is another specialized register that keeps track of execution of a program. It contains the memory address of the next instruction to be fetched and executed.

Besides IR and PC, there are n-general purpose registers R0 through Rn-1.

The other two registers which facilitate communication with memory are: -
1**. MAR – (Memory Address Register):-** It holds the address of the location to be accessed.
2**. MDR – (Memory Data Register):-** It contains the data to be written into or read out of the address location.

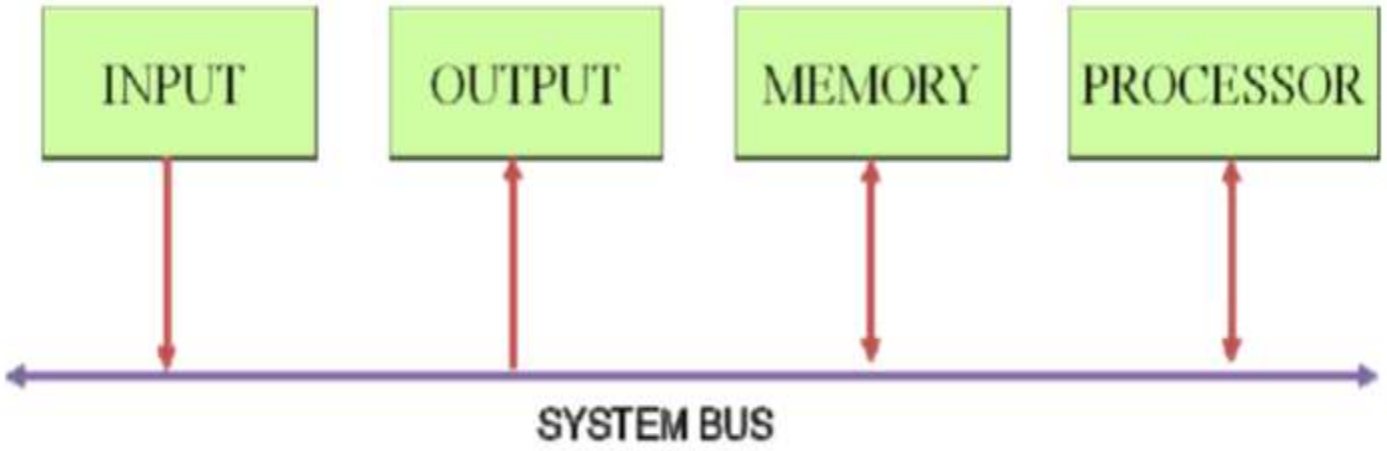


Operating steps are

- Programs reside in the memory & usually get these through the I/P unit.
- Execution of the program starts when the PC is set to point at the first instruction of the program.
- Contents of PC are transferred to MAR and a Read Control Signal is sent to the memory.
- After the time required to access the memory elapses, the address word is read out of the memory and loaded into the MDR.
- Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed.
- If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.
- An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.
- When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.
- After one or two such repeated cycles, the ALU can perform the desired operation.
- If the result of this operation is to be stored in the memory, the result is sent to MDR.
- Address of location where the result is stored is sent to MAR & a write cycle is initiated.
- The contents of PC are incremented so that PC points to the next instruction that is to be executed.

4. BUS STRUCTURES:

- Bus structure and multiple bus structures are types of bus or computing.
- A bus is basically a subsystem which transfers data between the components of Computer components either within a computer or between two computers.
- It connects peripheral devices at the same time.



Types of Buses

1. Data Bus:

- Data bus is the most common type of bus. It is used to transfer data between different components of computer.
- The number of lines in data bus affects the speed of data transfer between different components. The data bus consists of 8, 16, 32, or 64 lines. A 64-line data bus can transfer 64 bits of data at one time.
- The data bus lines are bi-directional. It means that:
- CPU can read data from memory using these lines CPU can write data to memory locations using these lines

2. Address Bus:

- Many components are connected to one another through buses. Each component is assigned a unique ID.
- This ID is called the address of that component. If a component wants to communicate with another component, it uses address bus to specify the address of that component.
- The address bus is a unidirectional bus. It can carry information only in one direction. It carries address of memory location from microprocessor to the main memory.

3. Control Bus:

- Control bus is used to transmit different commands or control signals from one component to another component.
- Suppose CPU wants to read data from main memory. It will use RD control is also used to transmit control signals like ASKS (Acknowledgement signals).

A control signal contains the following:

1 Timing information: It specifies the time for which a device can use data and address bus.

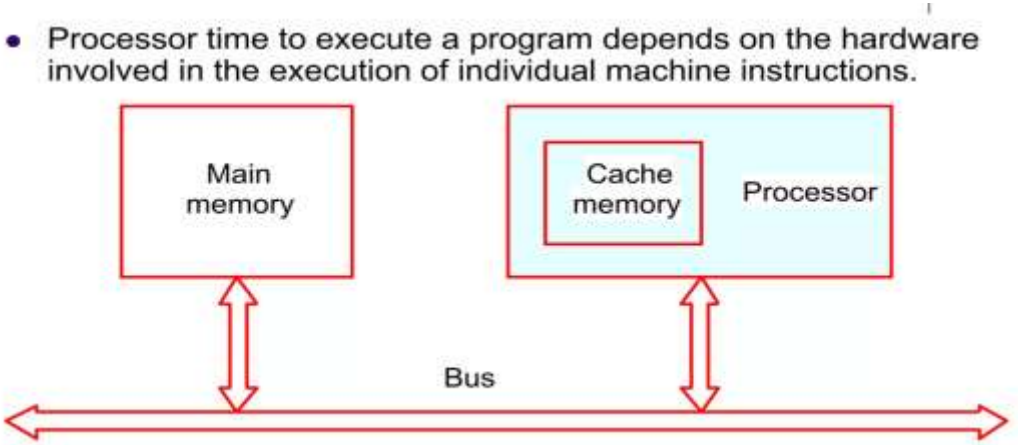
2 Command Signal: It specifies the type of operation to be performed.

Example: Suppose that CPU gives a command to the main memory to write data. The memory sends acknowledgement signal to CPU after writing the data successfully. CPU receives the signal and then moves to perform some other action.

5. PERFORMANCE & METRICS

- The most important measure of the performance of a computer is how quickly it can execute programs.
- The speed with which a computer executes program is affected by the design of its hardware.
- Generally 3 factors affects performance: **Hardware design, instruction set, Compiler**
- The total time required to execute the program is elapsed time is a measure of the performance of the entire computer system.
- It is affected by the speed of the processor, the disk and the printer. The time needed to execute a instruction is called the processor time.

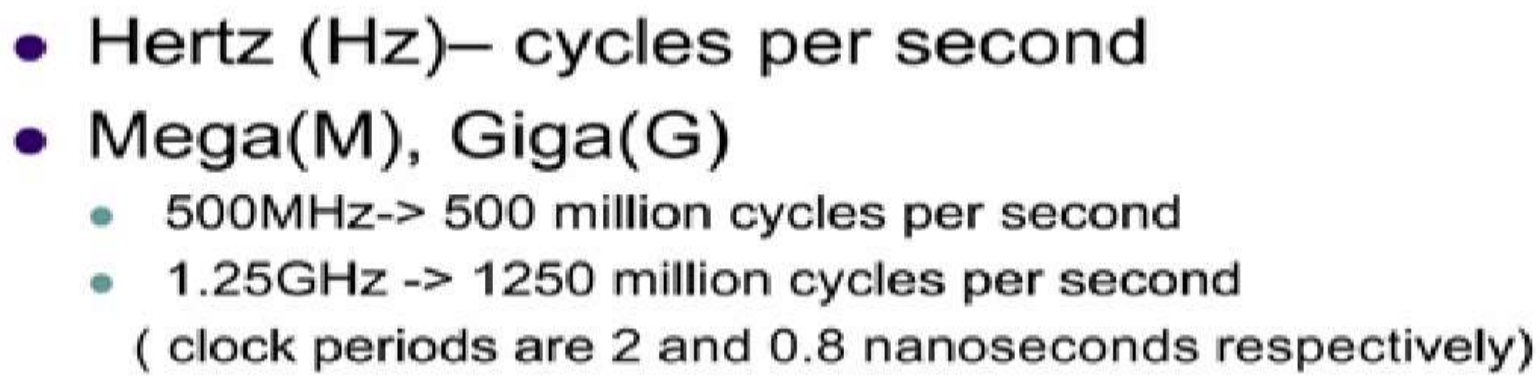
.



- Processor & relatively small cache memory can be fabricated on a single Integrated circuit chip – Speed, Cost, Memory Management

Processor clock: -

- Processor circuits are controlled by a timing signal called clock. The clock defines the regular time intervals called clock cycles (P).
- To execute a machine instruction the processor divides the action into a sequence of basic steps that each step can be completed in one clock cycle.
- Length of P affects Processor performance, Clock rate R= (1/P) cycles per second



Pipelining and Superscalar operation

a substantial performance is improved by overlapping the execution of successive instructions, using a technique called pipelining.

Add R1,R2,R3

Multiprocessors and Multicomputers

- Multiprocessor computer
  - Execute a number of different application tasks in parallel
  - Execute subtasks of a single large task in parallel
  - All processors have access to all of the memory – shared-memory multiprocessor
  - Cost – processors, memory units, complex interconnection networks
- Multicomputers
  - Each computer only have access to its own memory
  - Exchange message via a communication network – message-passing multicomputers

multicomputer	multiprocessors
1. A computer made up of several computers.	1. A computer that has more than one CPU on its motherboard.
2. Distributed computing deals with hardware and software systems containing more than one processing element, multiple programs	2. Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system.
3. It can run faster	3. Speed depends on the all processors speed
4. A multi-computer is multiple computers, each of which can have multiple processors.	4. Single Computer with multiple processors
5. Used for true parallel processing.	5. Used for true parallel processing.
6. Processor can not share the memory.	6. Processors can share the memory.
7. Called as message passing multi computers	7. Called as shared memory multi processors
8. Cost is more	8. Cost is low

7. Numbers

Numbering Systems		
System	Base	Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

There are many methods or techniques which can be used to convert numbers from one base to another. We'll demonstrate here the following −

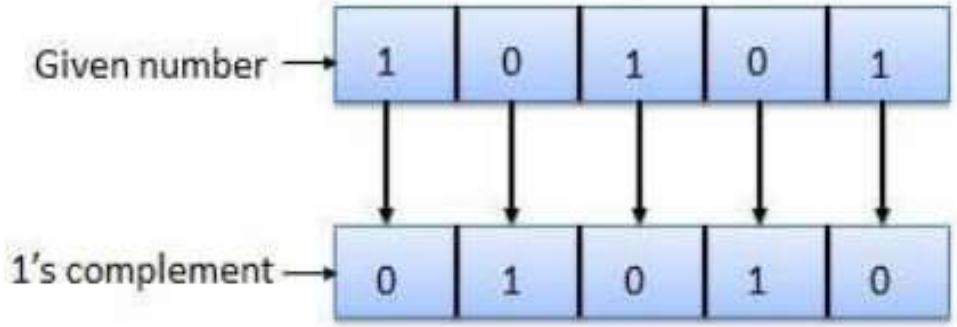
- Decimal to Other Base System
- Other Base System to Decimal
- Other Base System to Non-Decimal

Binary system complements

As the binary system has base r = 2. So the two types of complements for the binary system are 2's complement and 1's complement.

1's complement

The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement. Example of 1's Complement is as follows.

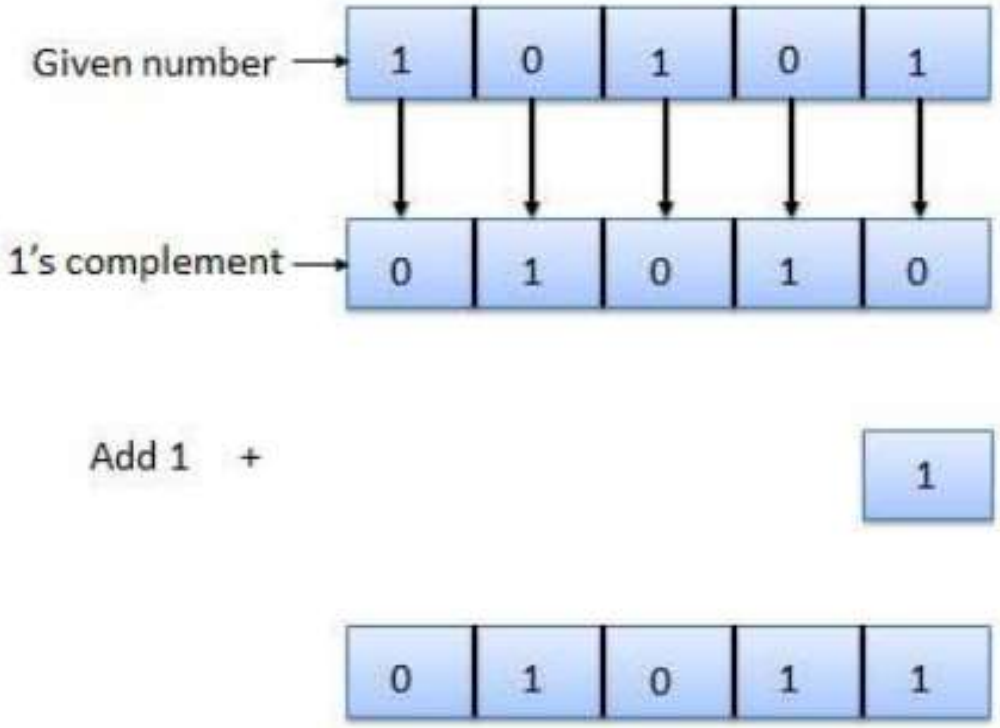


2's complement

The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number.

2's complement = 1's complement + 1

Example of 2's Complement is as follows.



Binary Arithmetic

Binary Addition

It is a key for binary subtraction, multiplication, division. There are four rules of binary addition.

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

In fourth case, a binary addition is creating a sum of (1 + 1 = 10) i.e. 0 is written in the given column and a carry of 1 over to the next column.

Example – Addition

0011010 + 001100 = 00100110

```
   1 1          carry
 0 0 1 1 0 1 0  = 26(10)
+0 0 0 1 1 0 0  = 12(10)
---------------
 0 1 0 0 1 1 0  = 38(10)
```

Binary Subtraction

Subtraction and Borrow, these two words will be used very frequently for the binary subtraction. There are four rules of binary subtraction.

Case	A - B	Subtract	Borrow
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	0	1

Example – Subtraction

0011010 - 001100 = 00001110

```
      1 1      borrow
  0 0 1 1 0 1 0  = 26(10)
- 0 0 0 1 1 0 0  = 12(10)
-----------------
  0 0 0 1 1 1 0  = 14(10)
```

Binary Multiplication

Binary multiplication is similar to decimal multiplication. It is simpler than decimal multiplication because only 0s and 1s are involved. There are four rules of binary multiplication.

Case	A x B	Multiplication
1	0 x 0	0
2	0 x 1	0
3	1 x 0	0
4	1 x 1	1

Example – Multiplication

Example:

0011010 x 001100 = 100111000

$$
\begin{array}{rl}
0\,0\,1\,1\,0\,1\,0 & = 26_{10} \\
\times\, 0\,0\,0\,1\,1\,0\,0 & = 12_{10} \\
\hline
0\,0\,0\,0\,0\,0\,0 & \\
0\,0\,0\,0\,0\,0\,0 & \\
0\,0\,1\,1\,0\,1\,0 & \\
0\,0\,1\,1\,0\,1\,0 & \\
\hline
0\,1\,0\,0\,1\,1\,1\,0\,0\,0 & = 312_{10}
\end{array}
$$

8. Memory Locations

Memory Location, Addresses, and Operation

- Operands and instructions are stored in memory
- Memory consists of many millions of storage cells, each of which can store 1 bit.
- Word of information.
- Data is usually accessed in *n*-bit groups. *n* is called word length.
- Word length 16 to 64 bits.

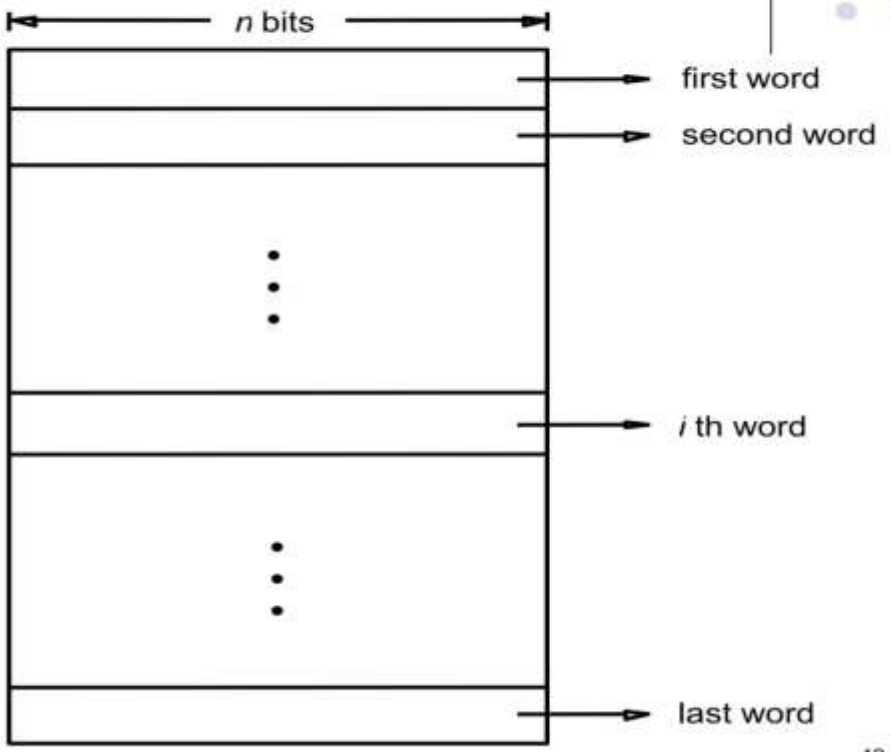


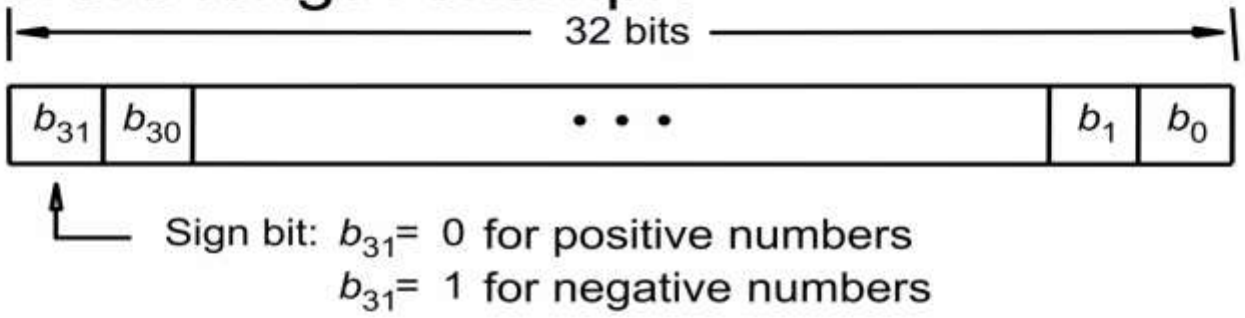
Fig: Memory words.

42

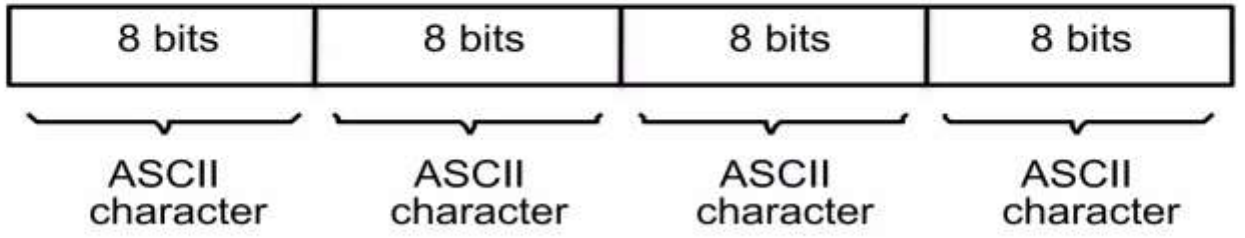
Byte Addressability

Memory Location, Addresses, and Operation

- 32-bit word length example



(a) A signed integer



(b) Four characters

Memory Location, Addresses, and Operation

- To retrieve information from memory, either for one word or one byte (8-bit), addresses for each location are needed.
- A *k*-bit address memory has 2^k memory locations, namely $0 - 2^k-1$, called memory space.
- 24-bit memory: 2^{24} = 16,777,216 = 16M ($1M=2^{20}$)
- 32-bit memory: 2^{32} = 4G ($1G=2^{30}$)
- 1K(kilo)=2^{10}
- 1T(tera)=2^{40}

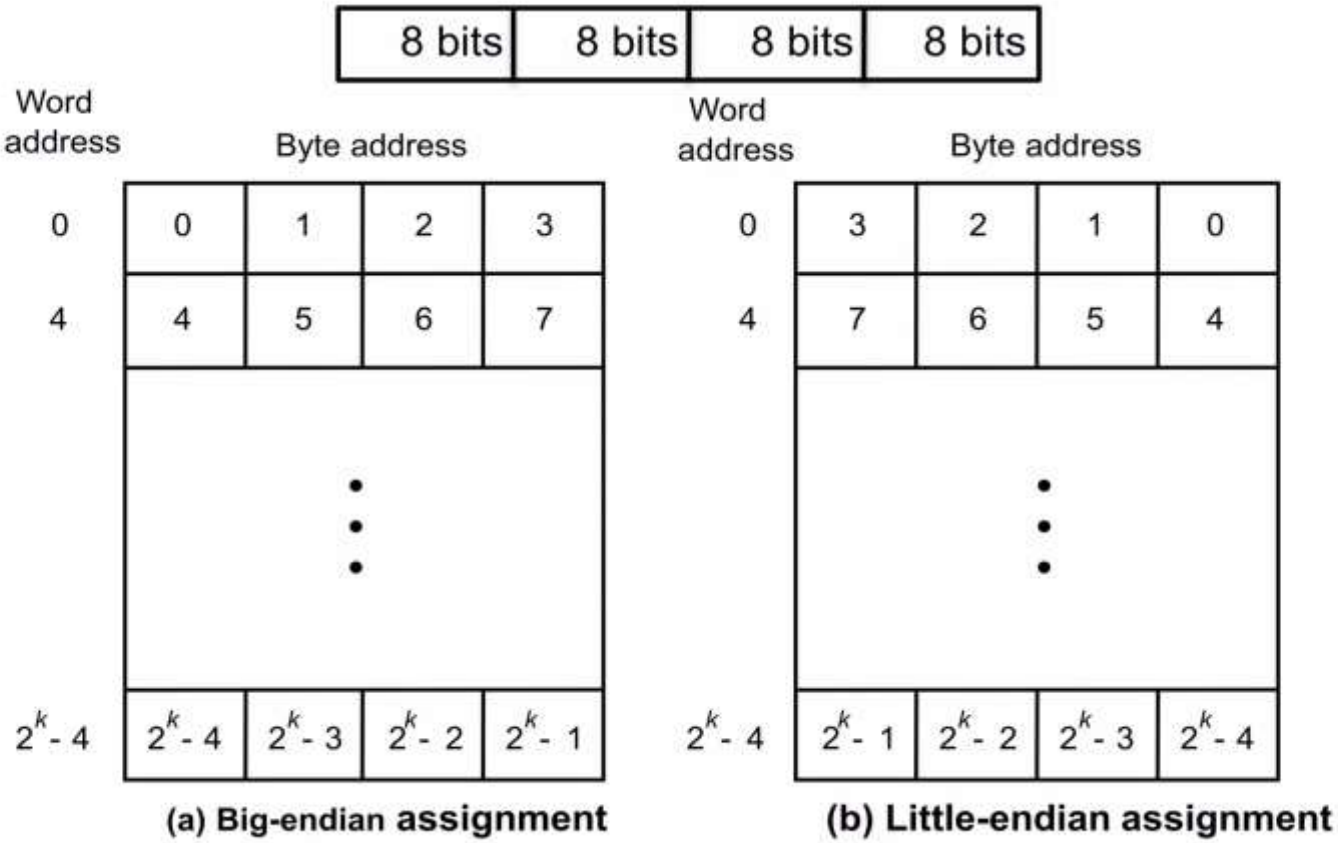
Memory Location, Addresses, and Operation

- It is impractical to assign distinct addresses to individual bit locations in the memory.
- The most practical assignment is to have successive addresses refer to successive byte locations in the memory – byte-addressable memory.
- Byte locations have addresses 0, 1, 2, … If word length is 32 bits, they successive words are located at addresses 0, 4, 8,… with each word consisting of 4 bytes.

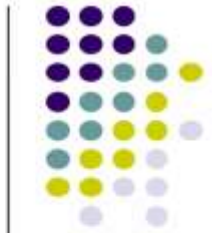
Big-Endian and Little-Endian Assignments

Big-Endian: lower byte addresses are used for the most significant bytes of the word

Little-Endian: opposite ordering. lower byte addresses are used for the less significant bytes of the word



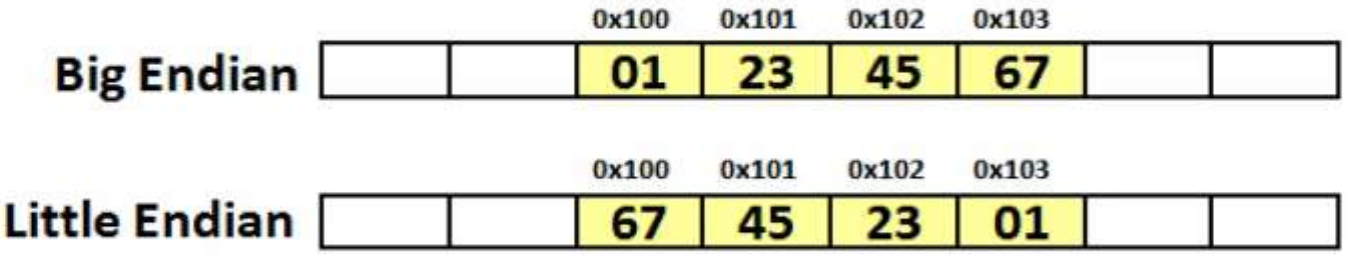
Big-Endian and Little-Endian Assignments



Big-Endian: MSB Byte will store first. MSB Byte will store at the lowest memory address

Little-Endian: LSB byte will store first. So the LSB Byte will store at the lowest memory address

On **big-endian (PowerPC, SPARC, and Internet)**, the 32-bit value x01234567 is stored as four bytes 0x01, 0x23, 0x45, 0x67, while on **little-endian (Intel x86)**, it will be stored in reverse order:



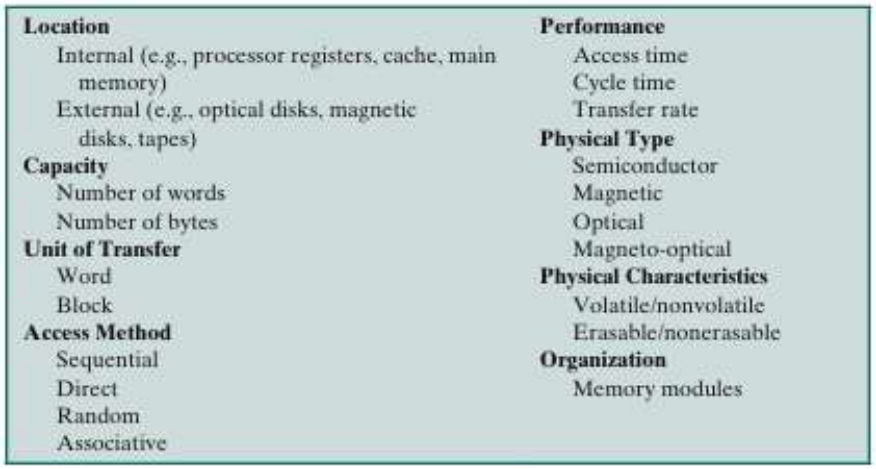
Memory Location, Addresses, and Operation

- Bits, Byte, Word
- Address ordering of bytes
- Word alignment
  - Words locations have aligned addresses in memory if they begin at a byte address. that is a multiple of the number of bytes in a word.
    - 16-bit word: word addresses: 0, 2, 4,….
    - 32-bit word: word addresses: 0, 4, 8,….
    - 64-bit word: word addresses: 0, 8,16,….
- Access number(1 word), character(1 Byte), and character string(multiple bytes)

Unit II: Basic Processing Unit

- Computer memory system overview: Characteristics, memory hierarchy, cache memory principles.
- Memory operations, memory locations and addresses,
- Instructions and instruction sequencing : Register transfer notations, assembly language notations, basic instruction types.
- Instruction execution
- Addressing modes and Assembler directives
- Basic IO operations
- Stacks & queues

1. Computer memory system overview



Characteristics of Memory Systems

Location:

The term location refers to whether memory is internal or external to the computer. Internal memory is often equated with main memory, but there are other forms of internal memory.

The processor requires its own local memory, in the form of registers.

External memory consists of peripheral storage devices, such as disk and tape, that are accessible to the processor via I/O controllers.

Capacity :

An obvious characteristic of memory is its 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

Unit of Transfer:

A related concept is the 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.

Access Method:

- **Sequential access:**

Memory is organized into units of data, called records. Access must be made in a specific linear sequence.

- **Direct access:**

As with sequential access, direct access involves a shared read– write mechanism. However, individual blocks or records have a unique Computer memory System address based on physical location.

- **Random access:**

Each addressable location in memory has a unique, physically wired- in addressing mechanism.. 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 that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously.

Performance:

- **Access time (latency):** For random- access memory, this is the time it takes to perform a read or write operation, that is, the time from the instant that an address is presented to the memory to the instant that data have been stored or made available for use.
- **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. Note that memory cycle time is concerned with the system bus, not the processor.
- **Transfer rate:** This is the rate at which data can be transferred into or out of a memory unit.

Hierarchy of Memory :

As one goes down the hierarchy, the following occur:

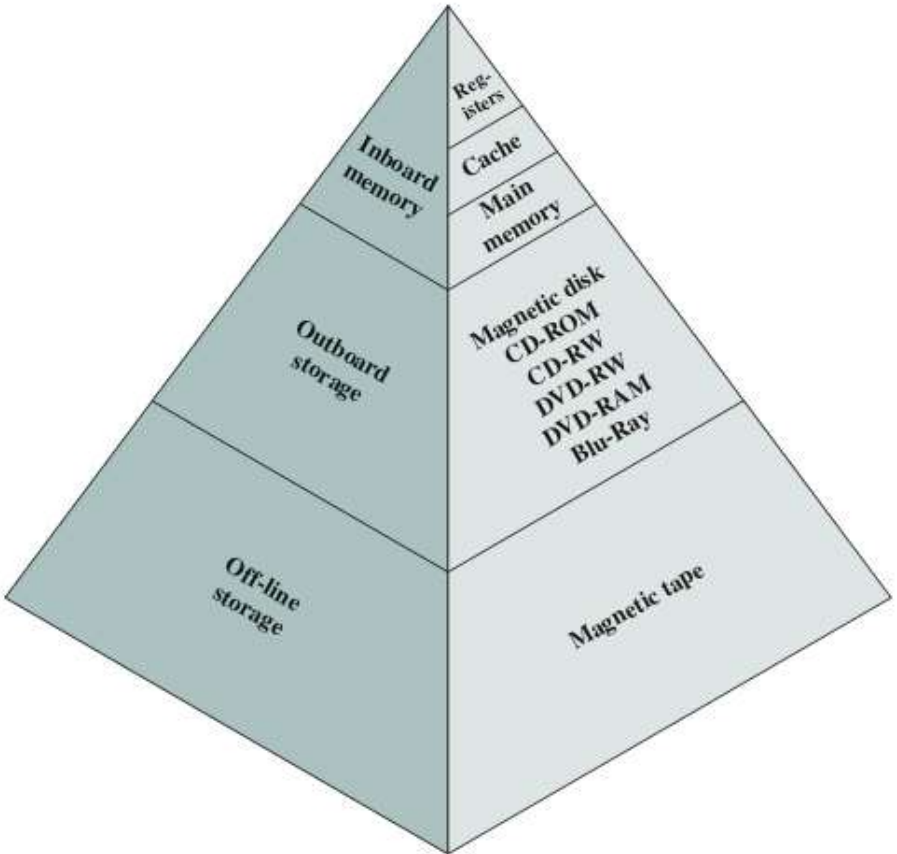
a. Decreasing cost per bit;

b. Increasing capacity;

c. Increasing access time;

d. Decreasing frequency of access of the memory by the processor.

Thus, smaller, more expensive, faster memories are supplemented by larger, cheaper, slower memories.



Cache memory is designed to combine the memory access time of expensive, high- speed memory combined with the large memory size of less expensive, lower- speed memory.

- There is a relatively large and slow main memory together with a smaller, faster cache memory. The cache contains a copy of portions of main memory.
- When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache.
- 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.

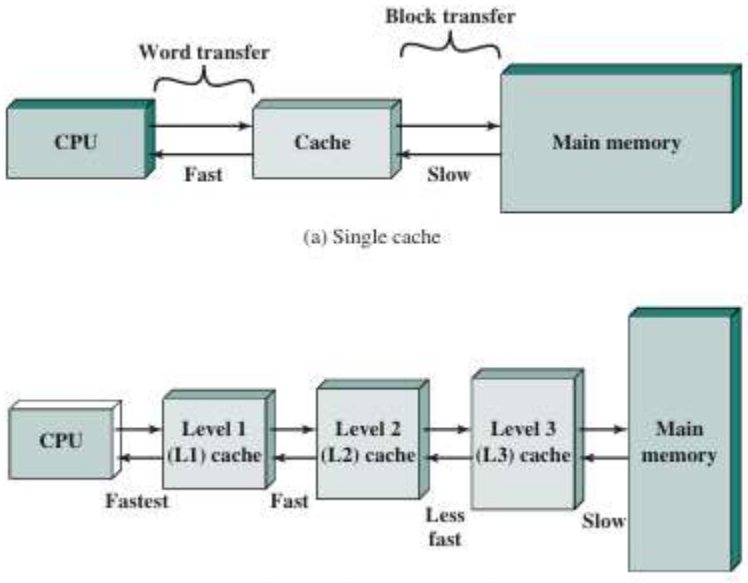


Figure depicts the use of multiple levels of cache. The L2 cache is slower and typically larger than the L1 cache, and the L3 cache is slower and typically larger than the L2 cache.

2. Memory Operations

- Both program instructions and data operands are stored in memory.
- To execute an instruction, the processor control ckts must cause the word to be transferred from memory to processor.
- Operands and results must also be moved between the Memory & Processor. Hence two basic operations involving the memory and the processor.

Load (or Read or fetch)
Store(or Write)

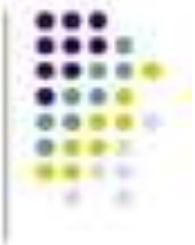
Load:

- Load operations transfers a copy of the contents of a specific memory location to the processor. The memory contents remain unchanged.
- To start the load operation, the proc. sends the address of the desired locations to the memory and requests that its contents be read.
- The mem. reads the data stored at that address and sends them to the processor.

Store:

- The Store operation transfers an item of information from the proc.to a specific mem location, destroying the former contents of that location. The proc. sends the addr. of the desired location to the mem together with the data to be written into that location.
- Either one word or byte can be transferred between the proc. and the mem in a single operation

"Must-Perform" Operations



- Data transfers between the memory and the processor registers
- Arithmetic and logic operations on data
- Program sequencing and control
- I/O transfers

3. Register Transfer Notations

- We need to describe the transfer of information from one location to another. Possible locations that may be involved are mem locations, registers, or registers in I/O subsystem. Most of the time we identify locations by a symbolic name.

- Identify a location by a symbolic name standing for its hardware binary address (LOC, PLACE,R0,R1, DATAIN,DATAOUT…)

- Contents of a location are denoted by placing square brackets around the name of the location (R1←[LOC], R3 ←[R1]+[R2])

- Register Transfer Notation (RTN)

  RHS : denotes value

  LHS : name of location where value is placed, overwriting old content of location.

Assembly Language Notations

We need another type of notation to represent machine instructions and programs. For this we use Assembly Language Format.

- Represent machine instructions and programs.
- Move LOC, R1 => R1←[LOC]
- Add R1, R2, R3 => R3 ←[R1]+[R2]

Basic Instruction Types:

One address:

Operation Source

Two Address:

Operation Source , Destination

Three Address:

Operation Source 1, Source2, Destination

Instruction Formats

- C=A+B
- Three-Address Instructions
  - ADD A,B,C — C ← [A] + [B]
  - ADD R2, R3, R1 — R1 ← [R2] + [R3]
- Two-Address Instructions
  - ADD A,B — B ← [A] + [B]
  - Move B,C — C← [B]
  - ADD R2, R1 — R1 ← [R1] + [R2]
- One-Address Instructions
  - ADD M — AC ← [AC] + [M]
  - Load A — AC ← [A]
  - Store A — A ← [AC]
  - Load A, ADD B, Store C
- Zero-Address Instructions
  - ADD — TOS ← [TOS] + [TOS – 1]

Instruction Formats

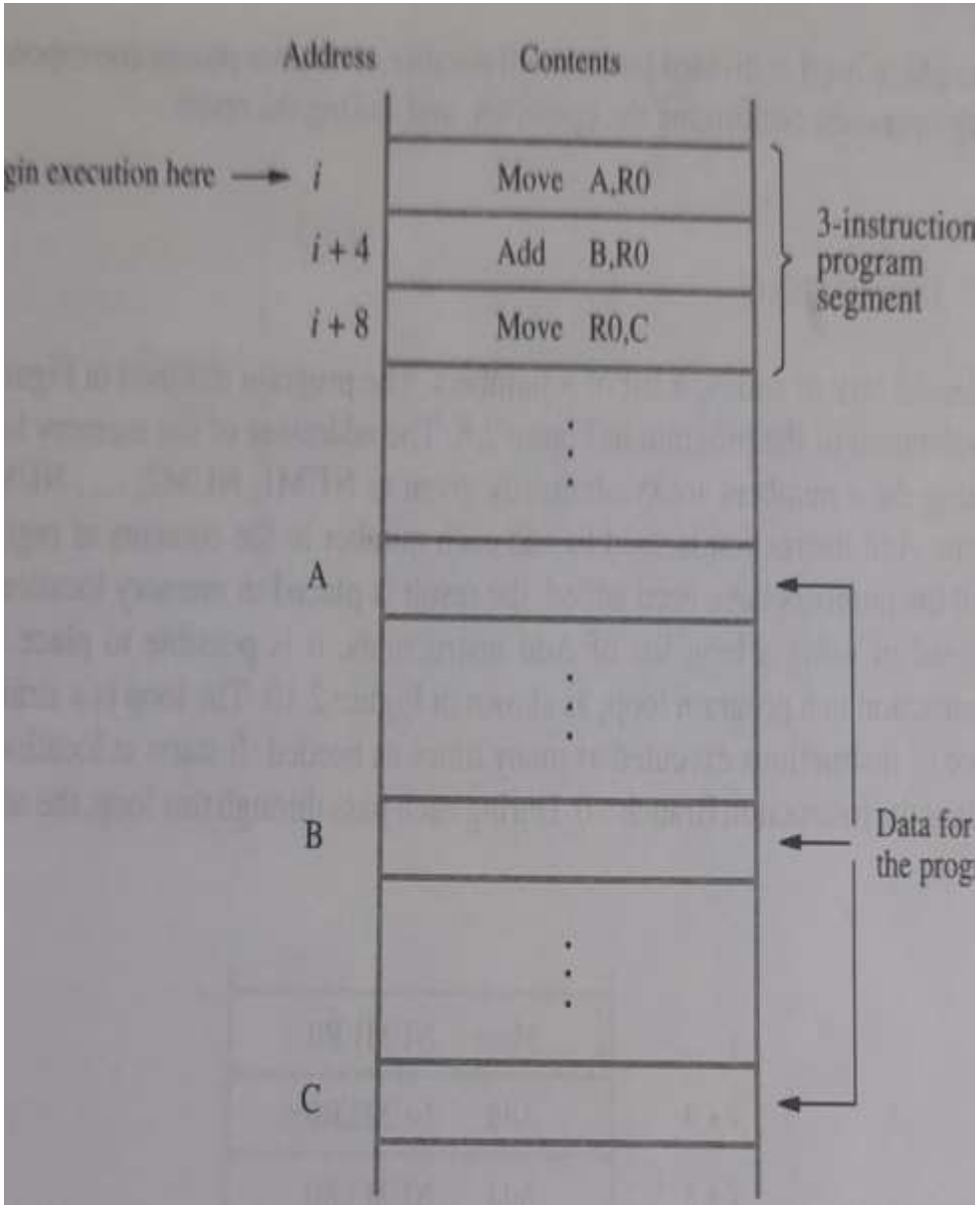
Example: Evaluate (A+B) * (C+D)

- Three-Address

```
1. ADD   A, B, R1      ; R1 ← [A] + [B]
2. ADD   C, D, R2      ; R2 ← [C] + [D]
3. MUL   R1, R2, X     ; X ← [R1] * [R2]
```

4. Instruction Execution and straight Line Sequencing

- For c← [A] +[B], it appears in memory of computer. We assumed that computer allows one memory operand per instruction and has no.of processor registers.
- Assumed that word length is 32 bits and memory is byte addressable.
- Three instructions of the program are in successive word locations, starting at location 'i'. Since each instruction is 4 bytes long, the second & third instructions start at address i+4 and i+8.
- Let us consider this instruction, PC hold the address of first instruction and also hold next instruction address.
- Processor control ckts use the information in the PC to fetch and execute instructions, and at a time in order of increasing addresses. This is called **straight Line Sequencing.**
- During execution, PC is incremented by 4 to point to next instruction then i+8 is executed.
- **Execution of instruction is in two phase procedure:**
- **Instruction Fetch & Instruction Execution**



Assumptions:

- One memory operand per instruction
- 32-bit word length
- Memory is byte addressable
- Full memory address can be directly specified in a single-word instruction

Two-phase procedure

-Instruction fetch

-Instruction execute

FIG. Program For c <- [A]+[B]

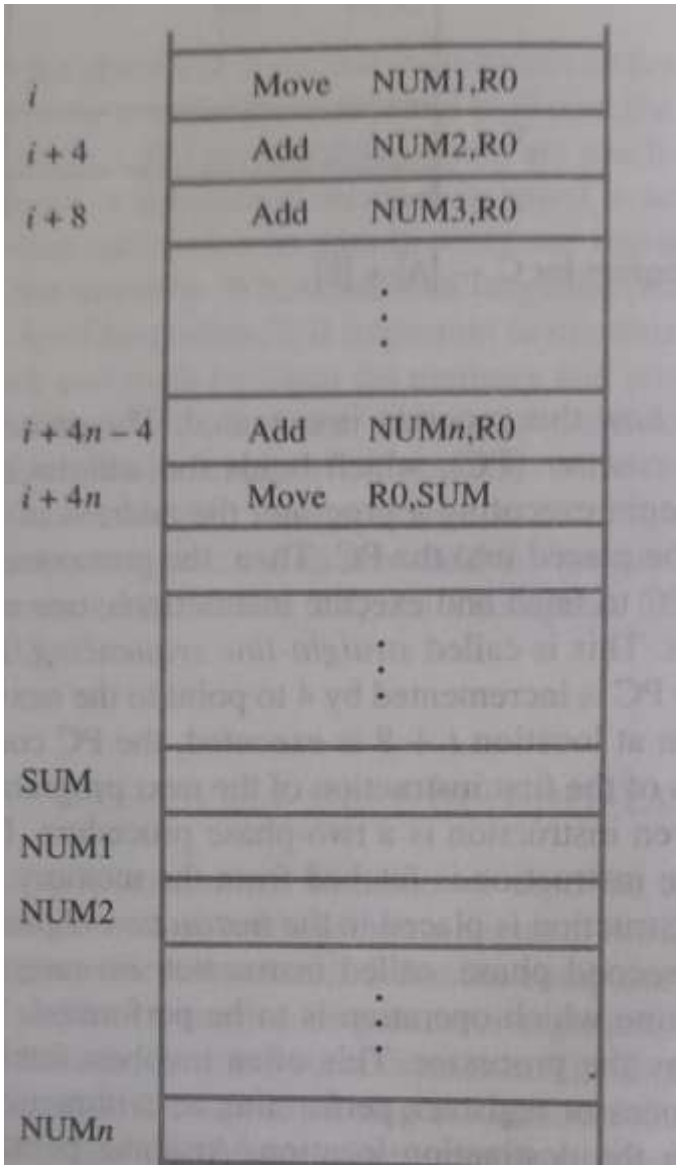


Fig. A St. line program for adding n numbers

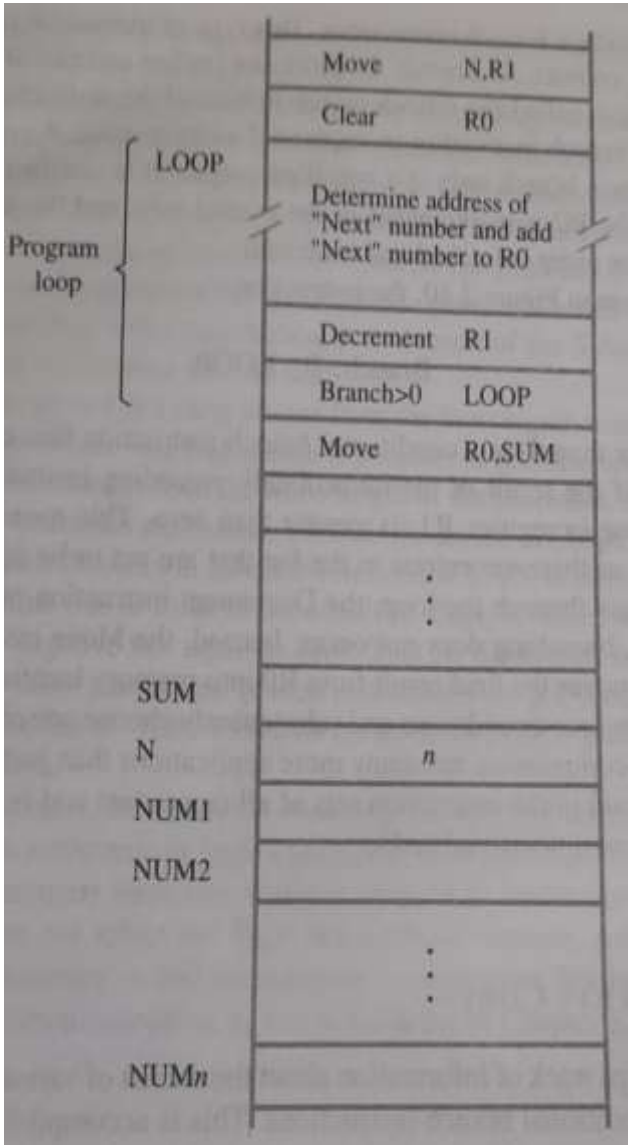


Fig. Using loop to add numbers

5. Addressing Modes

- The different ways in which the location of an operand is specified in an instruction are referred to as Addressing Modes.

- **Addressing modes** - how architectures specify the address of an object they will access. Addressing modes specify constants and registers in addition to locations in memory.

- When a memory location is used, the actual memory address specified by the addressing mode is called the Immediate or literals are usually considered memory addressing modes

Addressing Modes

- Every instruction of a program has to operate on a data.
- The different ways in which a source operand is denoted in an instruction are known as addressing modes.

1. Register Addressing
2. Immediate Addressing

Group I & II : Addressing modes for register and immediate data

3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing

Group III : Addressing modes for memory data

The different ways in which the location of an operand is specified in an instruction are referred to as Addressing Modes.

Name	Assembler syntax	Addressing function
Immediate	#Value	Operand = Value
Register	R*i*	EA = R*i*
Absolute (Direct)	LOC	EA = LOC
Indirect	(R*i*) (LOC)	EA = [R*i*] EA = [LOC]
Index	X(R*i*)	EA = [R*i*] + X
Base with index	(R*i*,R*j*)	EA = [R*i*] + [R*j*]
Base with index and offset	X(R*i*,R*j*)	EA = [R*i*] + [R*j*] + X
Relative	X(PC)	EA = [PC] + X
Autoincrement	(R*i*)+	EA = [R*i*]; Increment R*i*
Autodecrement	−(R*i*)	Decrement R*i*; EA = [R*i*]

EA = effective address
Value = a signed number

Register Mode:

The operand is the contents of a processor register, the name of the register is given in instruction

Ex: Move A, B EA= Ri

Immediate Mode:

The operand is given explicitly in the instruction

Ex: Move 200 ,Ro Operand= Value

Absolute (Direct):

The operand is in the memory location, the address of this location is given explicitly in the instruction.

Ex: Move LOC, R2 EA =LOC

Indirect Mode:

The Effective address of the operand is the contents of register or memory location whose address appears in the instruction

Ex: Add (A), R1 EA=[Ri], or EA=[LOC]

Indexing Mode: The Effective address of the operand is generated by adding constant value to the contents of a register. We represent it by X(Ri). X=constant value and Ri is the name of register invloved. EA= X + [Ri]

Relative Addressing:

The Effective address is determined by the Index mode using Program counter in place of general purpose register (Ri)

This mode can be used to access the data operands, but its most common useis to specify the target address in branch instruction

Autoincrement mode:

The Effective address of the operand is the contents of register specified in the insteuction . After accessing the operand, the contents of this register are automatically incremented to point to the next item in a list

(Ri)+

AutoDecrement mode:

The contents of this register are first automatically decremented and then used as the the effective address of the operand

(Ri)+

Assembler Directives:

- The assembly language allows the programmer to specify other information needed to translate the source program into the object program.
- we need to assign numerical values to any names used in a program.. suppose that the name SUM is used to represent the value 200. this fact may be conveyed to the assembler program through a statement such as

SUM EQU 200

- It will informs the assembler that the name SUM should be replaces]d by the value 200 wherever it appears in the program. Such statement is called Assemble directives or commands.

6. Basic Input/Output Operations:

- We now examine the means by which data are transferred between the memory and outside world.0
- 0I/O operations are essential and the way are performed can have a significant effect on the performance of the computer.
- Consider a task that reads in character input from a KB and produces character on a display screen. A simple way of performing such I/O tasks is to use a method known as **Program-Controlled I/O.**
- The rate of data transfer from the KB to a computer is limited by typing speed of the user, which is unlikely to exceed a few characters per second **but the rate of output transfers** to display is much higher.
- However this is much slower than the speed of a processor that can execute many millions of instructions per second.
- The difference is speed between the processor and I/O devices creates the need for mechanism to synchronize the transfer of data between them

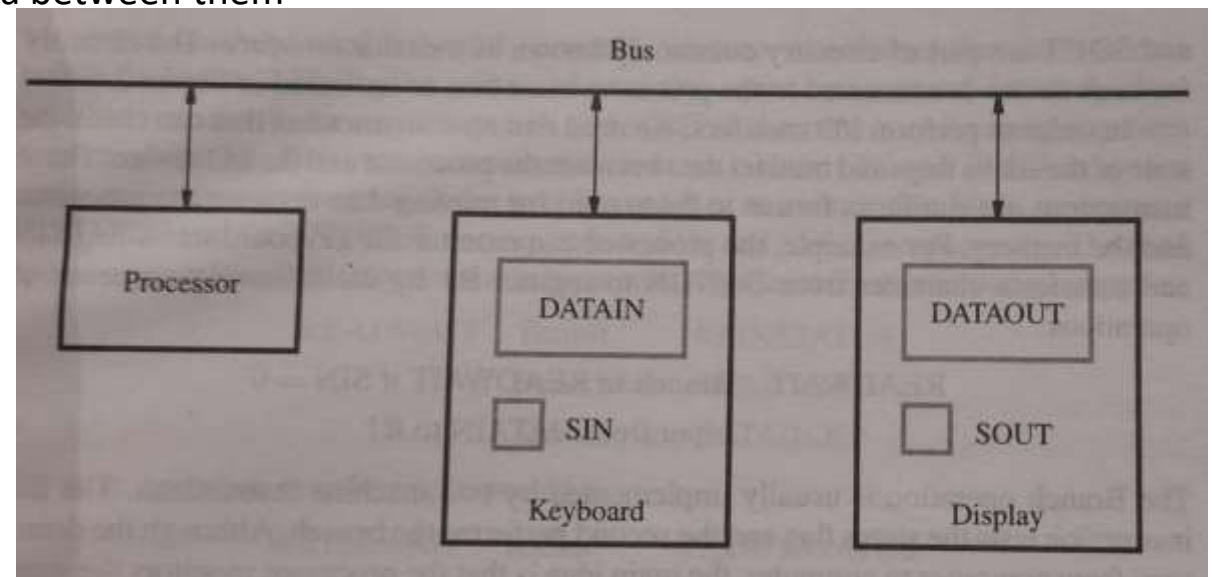


Fig. : Bus connection for Processor, KB and Display.

- **A solution to this problem is:**
- On output, the processor sends the first character and then wait for signal from the display that the character has been received it then sends the second character and so on.
- Input is sent from the keyboard in a similar way, the processor waits for a signal indicating that the character key has been struck and is available in some buffer register associated with the keyboard then the processor processed to read that code.
- The keyboard and the display are separate devices shown fig., the action of striking on the keyboard does not automatic click cause the corresponding character to be displayed on the screen one block of instructions in the I/O program transfer the character into the processor and another associated block of instructions causes the character to be displayed.
- **Consider the problem of moving a character from keyboard to the processor,**
- **Striking a key stores the corresponding character code in an 8-bit buffer register associated with the keyboard. Call this register DATAIN shown in fig.**
- **To inform the processor that a valid character is in DATAIN, Status control flag SIN is set to 1.**
- **A program monitors a SIN, and when SIN = 1, the processor reads the contents of DATAIN. When the character is transferred to the processor SIN is automatically cleared to 0. If a second character is entered at the keyboard SIN is again set 1 and process repeats.**

- Consider the problem of characters are transferred from processor to display.
- A buffer register DATAOUT, and status control flag, SOUT, are used for this transfer. when SOUT equals 1, the display is ready to receive a character.
- Under program control, the processor monitors, SOUT, and when SOUT is set to 1, the processor transfer a character code to DATAOUT.
- The transfer of a character to DATAOUT clears SOUT to 0; when the display device is ready to receive a second character, SOUT is again set to 1.
- The buffer register DATAIN and DATAUT, and the status flags are part of circuitry commonly known as *Device Interface. T*his circuitry for each device is connected to the processor via a bus indicated in figure.

7. STACKS and QUEUES:

- To organize the control and information formation linkage between the main program and the Subroutine, a data structure, called **stack** is used as well as closely related data structure called a **queue.**
- Stack is a list of data elements usually words or bytes with the accessing restriction that elements can be added or removed at one end of the list only. This end is called the top of this stack (TOS), called SP Register and other end is called bottom.
- This structure is sometimes referred to as a post Pushdown stack.

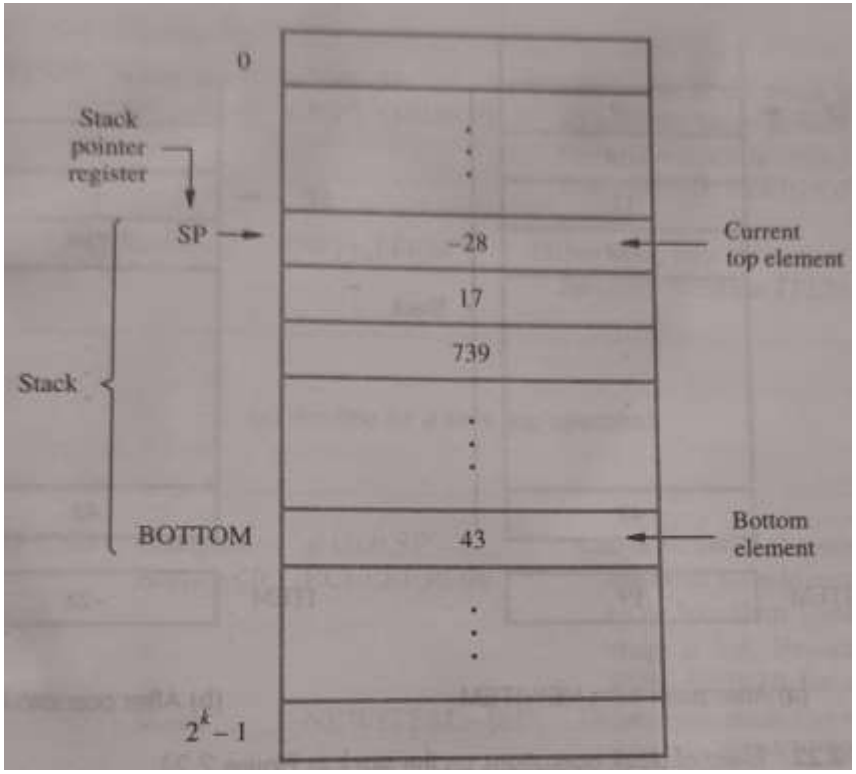


Fig: A stack of Words in the memory

- Another descriptive phrase last in first out before stack is also used to describe this type of storage mechanism; the last data item placed on this stack is the first one removed when it retrieval begins.
- The terms PUSH and POP are used to describe placing a new item on this stack and removing the top item from the stack respectively.
- Data stored in the memory can be organized as a stack with successive elements occupying successive memory locations as in that the first element is placed in location BOTTOM and when new elements are pushed on to stack they are place in successively lower address locations.
- we use the stack that grows in the direction of decreasing memory addresses.

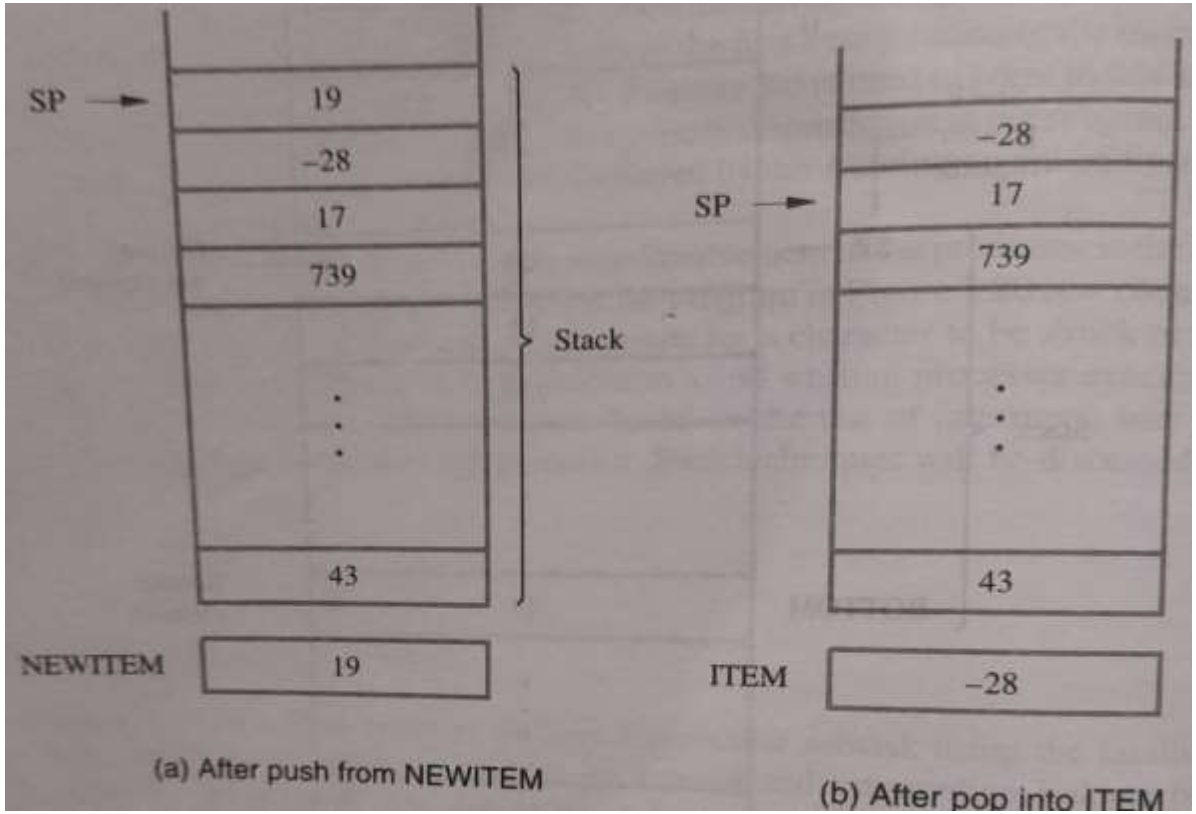


Fig: Effect of stack operation on the stack

- In figure it contains numerical values, with 43 at the bottom, and -28 the top.
- processor registers is used to keep track of the address of the element of the stack that is at the top of at any given time. This register is called stack pointer. If we assume a byte addressable memory with 32 bit mode operation can be implemented as

```
Subtract    #4, SP
Move NEWITEM, (SP)
```

- Similarly POP can be implemented,

```
Move (SP), ITEM
Add#4, SP
```

If the processor has autoincrement and auto decrement mode, then PUSH and POP operation can be performed by single instruction

```
Move  NEWITEM, -(SP)      --- -Push
Move (SP), + ITEM         ----- Pop
```

QUEUE:

- Another useful data structure that is called a **queue.** **D**ata are stored in and retrieved from a queue on a FIRST IN FIRST OUT (FIFO) basis.
- Hence if we assume that the **queue** grows in the direction of increasing addresses in the memory, new data are added at the back (High address end)and retrieved from the front (Low address end) of the queue
- A single pointer is needed to point to the top of the stack the other hand two pointers are needed to keep track of the two ends of the queue.

Unit III: I/O Organization

- Accessing I/O devices: I/O interface for an input device
- Interrupts: Interrupt hardware, enabling and disabling, Interrupt Priority scheme
- Direct Memory Access (DMA): Bus arbitration
- Interface circuits: Parallel Port & Serial Port.
- Std I/O Interfaces: Processor bus, PCI bus & SCSI bus standard.

1. Accessing I/O devices

- Simple arrangement to connect IO devices to a computer is to use a single bus arrangement shown fig.
- the bus enables all the devices connected to it to exchange information. Typically it consist of three sets of lines used to carry address, data and control signals.
- Each I/O devices is assigned a unique sets of addresses. When the processor places a particular address on the address lines the device that recognizes this address respond to the commands issued on the control lines.
- The processor request either a read or write operation, and the requested data are transferred over the data line. When I/O devices and the memory share the same address space this arrangement is called **Memory Mapped I/O**

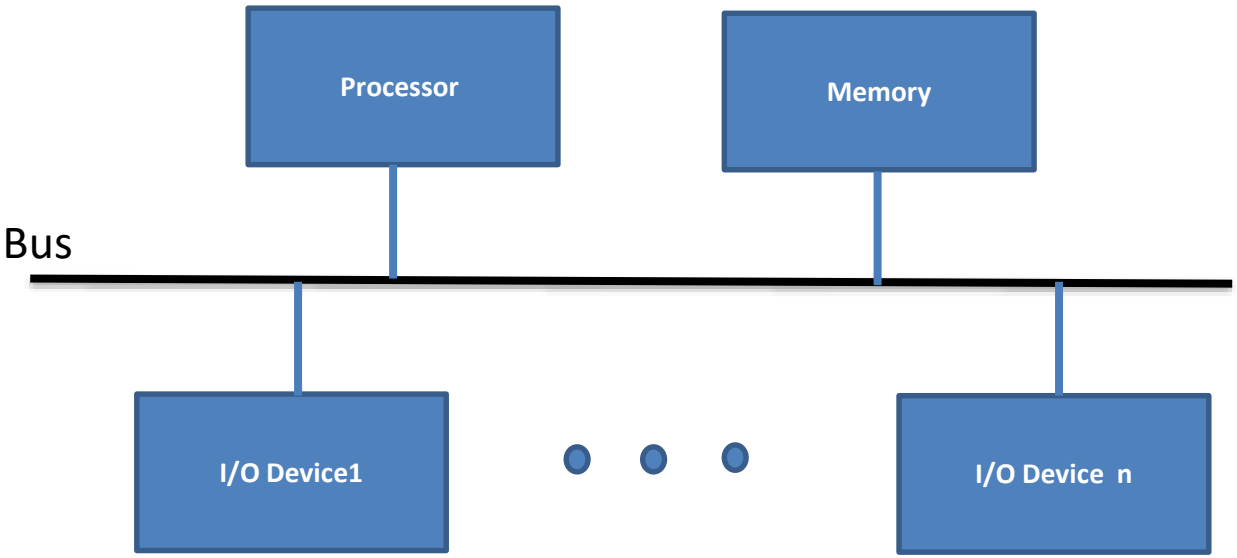


Fig. : A single bus Structure

With mem-mapped I/O any machine instruction that access memory can be used to transfer data to or from an I/O device.

For Ex: Move DATAIN, R0 - address of the input buffer associated with KB

Move R0, DATAOUT - output data buffer of a display unit or printer.

- I/O devices operate at speeds that are vastly different from that of the processor. When a human operator entering a characters at a KB, the processor is capable of executing millions of successive character entries.
- An instruction that reads a character from the KB should be executed only when a character is available in the input buffer of the KB interface..

- Following fig. shows the hardware required to connect an I/O devices to the bus.
- The address decoder enables the device to recognize its address when this address appears on the address lines.

- The data register holds the data being transferred to or from the processor. The status register contains info relevant to the operation of the I/O device.

- Both the data & Status registers are connected to the data bus and assigned unique addresses. The address decoder, the data and status registers, and the control circuitry required to coordinate I/O transfers constitutes the devices interface circuit.

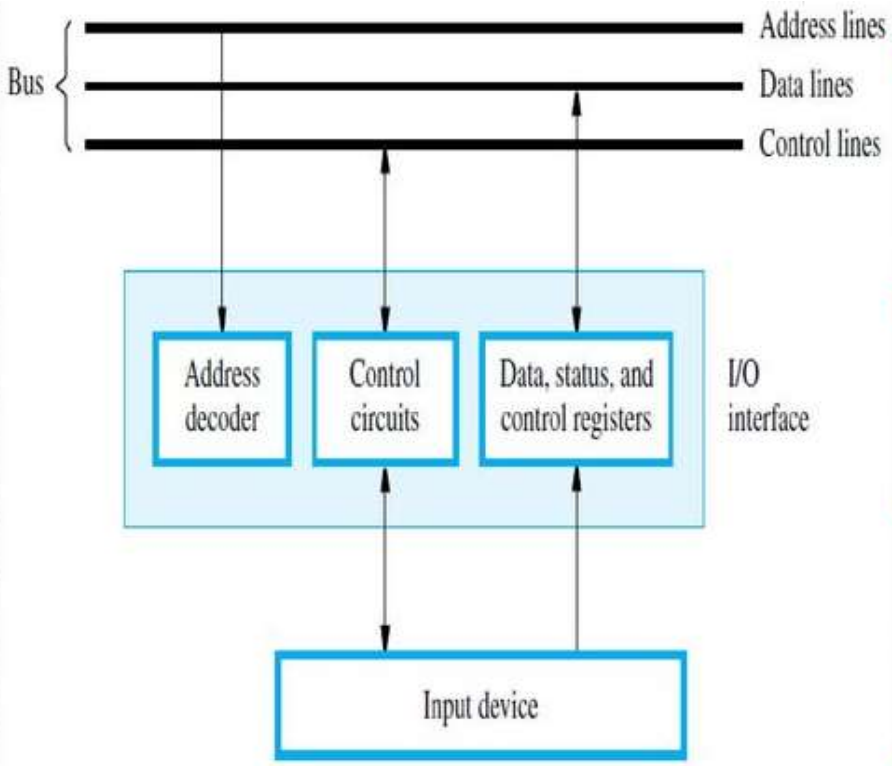


Fig: I/O interface for an Input device

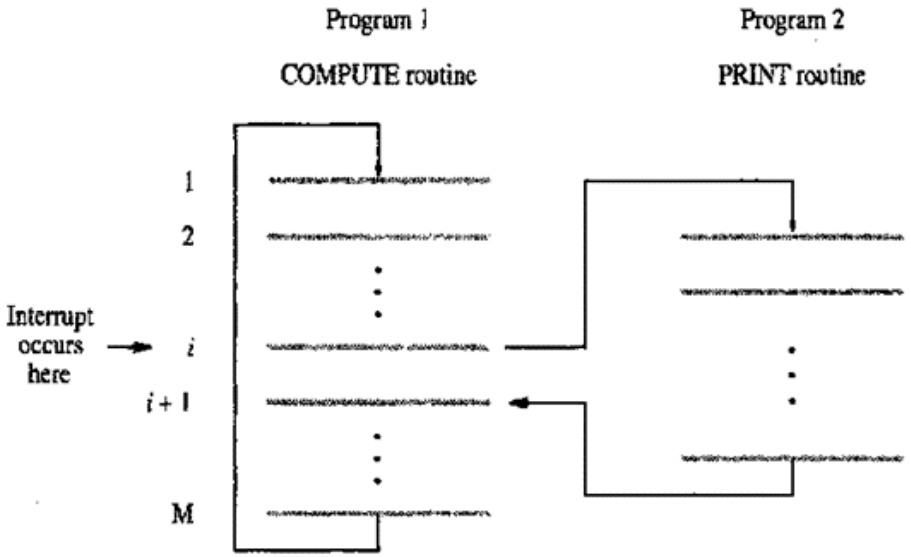
- Processor places a particular address on the address lines, it is examined by the address decoders of all devices on the bus.
- The device that recognizes this address responds to the commands issued on the control lines.
- The processor uses the control lines to request either a Read or a Write operation, and the requested data are transferred over the data lines.
- Intel processor uses I/O mapped I/O.

Program Controlled I/O

- In this processor repeatedly checks a status flags to achieve the required synchronization between the processor and an input or output device
- Also called **Processor Polls the device**
- Other methods are Interrupt and DMA

2. Interrupts

- The program enters a waiting loop in which it repeatedly test the device status, during the time the processor is not performing any useful computation.
- There are many situations where other task can be performed while waiting for an IO device to become ready.
- To allow this to happen, we can arrange for the IO device to alert the processor when it becomes ready.
- It can do so by sending the hardware signal called and **Interrupt** to the processor. At least one of the bus control lines called and **Interrupt request** line is usually dedicated for this purpose.
- since the processor is no longer required to continuously check the status of external devices, it can use the waiting period to perform other useful functions. By using interrupt such waiting periods can be ideally eliminated.



Ex:

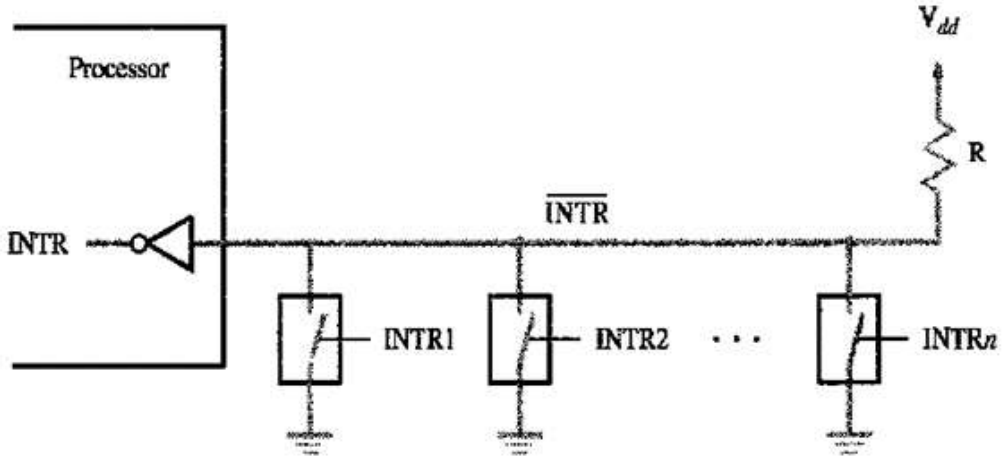
Consider a task that requires some computations to be performed and the results to be printed on a line printer. This is followed by more computations and output, and so on. Let the program consist of two routines, COMPUTE and PRINT. Assume that COMPUTE produces a set of n lines of output, to be printed by the PRINT routine.

The required task may be performed by repeatedly executing first the COMPUTE routine and then the PRINT routine. The printer accepts only one line of text at a time. Hence, the PRINT routine must send one line of text, wait for it to be printed, then send the next line, and so on, until all the results have been printed. The disadvantage of this simple approach is that the processor spends a considerable amount of time waiting for the printer to become ready. If it is possible to overlap printing and computation, that is, to execute the COMPUTE routine while printing is in progress, a faster overall speed of execution will result. This may be achieved as follows. First, the COMPUTE routine is executed to produce the first n lines of output. Then, the PRINT routine is executed to send the first line of text to the printer. At this point, instead of waiting for the line to be printed, the PRINT routine may be temporarily suspended and execution of the COMPUTE routine continued. Whenever the printer becomes ready, it alerts the processor by sending an interrupt-request signal. In response, the processor interrupts execution of the COMPUTE routine and transfers control to the PRINT routine. The PRINT routine sends the second line to the printer and is again suspended. Then the interrupted COMPUTE routine resumes execution at the point of interruption. This process continues until all n lines have been printed and the PRINT routine ends.

1. Interrupt Hardware

- It is found that I/O devices request on interrupt by activating a bus line called ***Interrupt Request***.
- Most computers are likely to have several I/O devices that can request an interrupt.
- A single interrupt request line may be used to serve ***n*** devices as given in figure.
- All devices are connected to the line via switches to ground. To request an interrupt, a device closes its associated switch. Hence if all interrupt signals $INTR_1$ to $INTR_n$ are inactive i.e, if all switches are open the voltage on the interrupt request line will be equal to V_{dd} .
- When a device request an interrupt by closing it switch, the voltage on the line drops to 0, causing the interrupt request signals, **INTR** received by the processor to go to 1. Since the closing of one or more switches will cause the line voltage to 0, i.e the signal is active when in the low voltage state.

INTR= ***INTR1 + INTR2 +….. +INTRn***



- In fig. special Gates are known as Open-collector or Open-drain is equivalent to switch to GND i.e open when Gates input is in the 0 state and closed when it is in the 1 state.
- The resistor R is called a pull-up register as it pulls the line voltage up to the high voltage state when the switches are open.

2. Enabling and Disabling Interrupts

- The facilities provided in a computer must give the programmer complete control over the events that takes place during the program execution.
- The arrival of an interrupt request from an external device causes to processor to suspend execution of one program and start the execution of another.
- Because interrupts can arrived anytime they may alter the sequence of events by the programmer.
- Hence the interaction of a program execution must be carefully controlled. A fundamental facility found in all computers is the ability to **Enable and Disable** such interruptions are desired.

Ex : **Compute- Print Routine**

Assuming that Interrupts are enable, following is the typical scenario :

1. The device raises an Interrupt-request
2. The processor interrupts the program currently being executed.
3. Interrupts are disabled by changing the control bits is PS.
4. The device is informed that its request has been recognized, and in response, it deactivates the interrupt-request signal.
5. The action requested by the interrupt is performed by the ISR (Interrupt service Routine)
6. Interrupts are enabled the execution of the interrupted program is resumed.

3. Handling Multiple Devices:

Vectored Interrupts, Interrupt Nesting, Simultaneous Interrupts

Let us now consider the situation where a number of devices capable of initiating interrupts are connected to the processor. Because these devices are operationally independent, there is no definite order in which they will generate interrupts. For example, device X may request an interrupt while an interrupt caused by device Y is being serviced, or several devices may request interrupts at exactly the same time. This gives rise to a number of questions:

1. How can the processor recognize the device requesting an interrupt?
2. Given that different devices are likely to require different interrupt-service routines, how can the processor obtain the starting address of the appropriate routine in each case?
3. Should a device be allowed to interrupt the processor while another interrupt is being serviced?
4. How should two or more simultaneous interrupt requests be handled?

The polling scheme is easy to implement. Its main disadvantage is the time spent interrogating the IRQ bits of all the devices that may not be requesting any service. An alternative approach is to use vectored interrupts, which we describe next.

1. Vectored Interrupt

- To reduce the time involved in the polling process a device requesting an interrupt may identified self directly to the process, Then the processor can immediately start executing a corresponding interrupt service routine (ISR). The term Vectored interrupt refers to all interrupt handling schemes based on this approach.
- A Device requesting an interrupt can identified itself by sending a special code to the processor over the bus. This enables the processor to identify individual devices even if the share a single interrupt-request line. The code supplied by the device may represent the starting address of the interrupt service routine (ISR) for that device. The code link is typically in the range of 4 to 8 bits.
- In most computers, I/O devices send the interrupt vector code over the data bus. using the bus control signals to ensure that devices do not interfere with each other. When a device sends and interrupt request, the processor may not be ready to receive the interrupt-vector code immediately.
- For Ex: it must first complete the execution of the current instruction which may require the use of the bus. There may be at the delays , the interrupting device must wait to put data on the bus when only when the processor is ready to receive it. When the processor is ready to receive the interrupt-vector code, activates INTA. I/O device responds by sending its Interrupt vector code and turning OFF the INTR signal.

2. Interrupt Nesting

A multiple-level priority organization means that during execution of an interrupt-service routine, interrupt requests will be accepted from some devices but not from others, depending upon the device's priority. To implement this scheme, we can assign a priority level to the processor that can be changed under program control. The priority level of the processor is the priority of the program that is currently being executed. The processor accepts interrupts only from devices that have priorities higher than its own.

A multiple-priority scheme can be implemented easily by using separate interrupt-request and interrupt-acknowledge lines for each device, as shown in Figure 4.7. Each of the interrupt-request lines is assigned a different priority level. Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor. A request is accepted only if it has a higher priority level than that currently assigned to the processor.

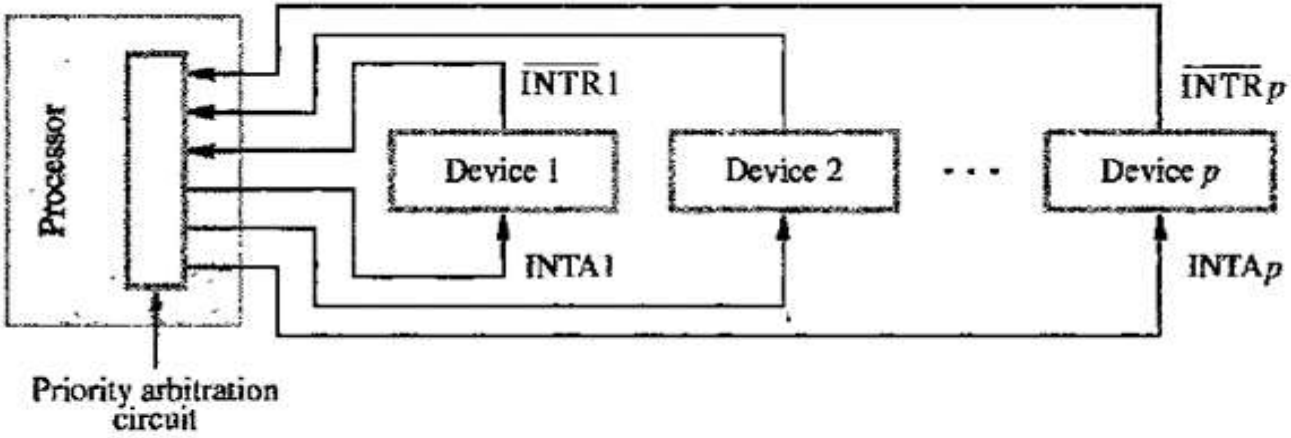
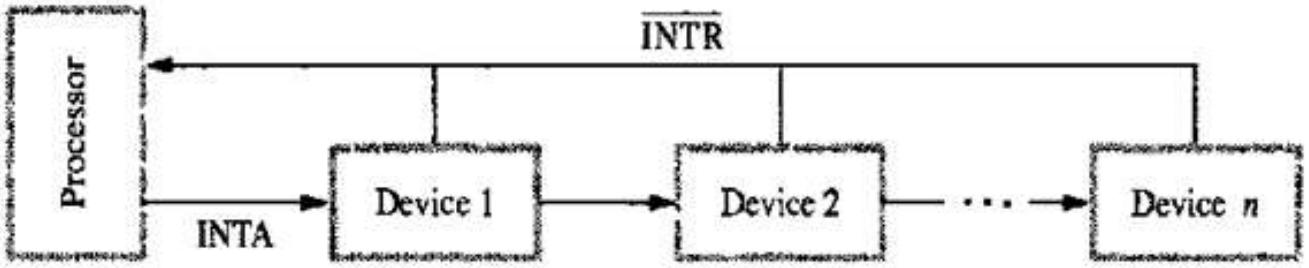


Figure 4.7 Implementation of interrupt priority using individual interrupt-request and acknowledge lines.

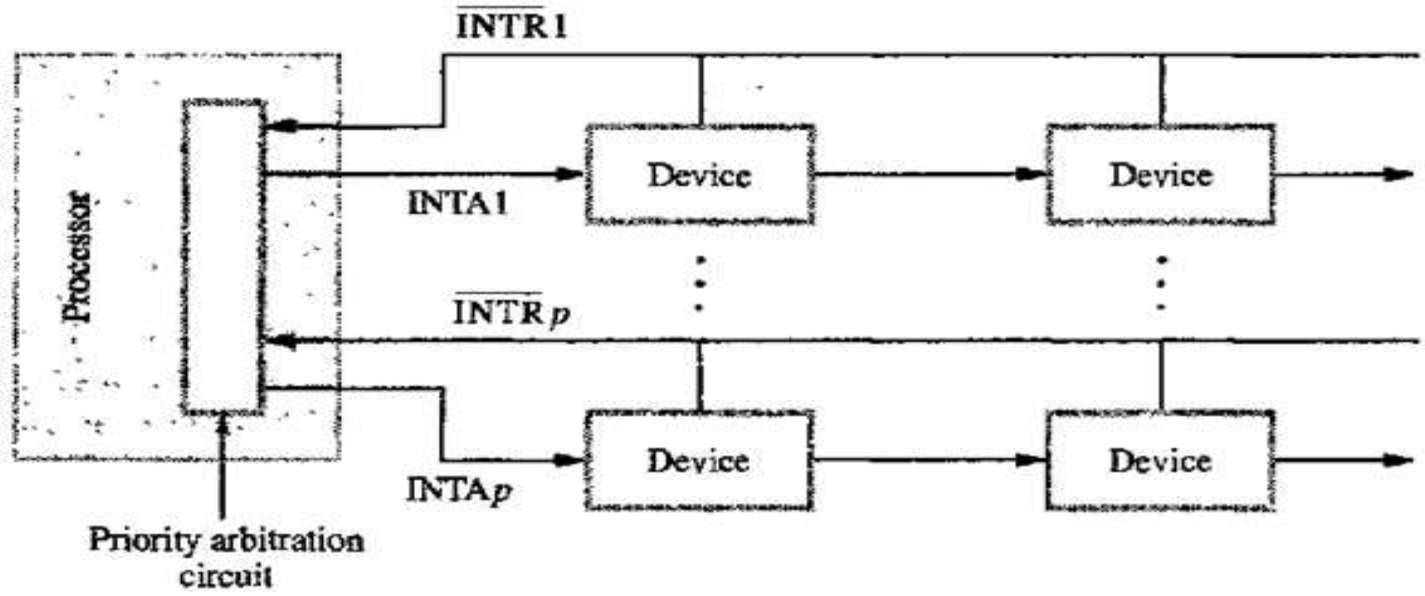
3. Simultaneous Request

Let us now consider the problem of simultaneous arrivals of interrupt requests from two or more devices. The processor must have some means of deciding which request to service first. Using a priority scheme such as that of Figure 4.7, the solution is straightforward. The processor simply accepts the request having the highest priority. If several devices share one interrupt-request line, as in Figure 4.6, some other mechanism is needed.

Polling the status registers of the I/O devices is the simplest such mechanism. In this case, priority is determined by the order in which the devices are polled. When vectored interrupts are used, we must ensure that only one device is selected to send its interrupt vector code. A widely used scheme is to connect the devices to form a *daisy chain*, as shown in Figure 4.8*a*. The interrupt-request line $\overline{\text{INTR}}$ is common to all devices. The interrupt-acknowledge line, INTA, is connected in a daisy-chain fashion, such that the INTA signal propagates serially through the devices. When several devices raise an interrupt request and the $\overline{\text{INTR}}$ line is activated, the processor responds by setting the INTA line to 1. This signal is received by device 1. Device 1 passes the signal on to device 2 only if it does not require any service. If device 1 has a pending request for



(a) Daisy chain



(b) Arrangement of priority groups

interrupt, it blocks the INTA signal and proceeds to put its identifying code on the data lines. Therefore, in the daisy-chain arrangement, the device that is electrically closest to the processor has the highest priority. The second device along the chain has second highest priority, and so on.

The scheme in Figure 4.8*a* requires considerably fewer wires than the individual connections in Figure 4.7. The main advantage of the scheme in Figure 4.7 is that it allows the processor to accept interrupt requests from some devices but not from others, depending upon their priorities. The two schemes may be combined to produce the more general structure in Figure 4.8*b*. Devices are organized in groups, and each group is connected at a different priority level. Within a group, devices are connected in a daisy chain. This organization is used in many computer systems.

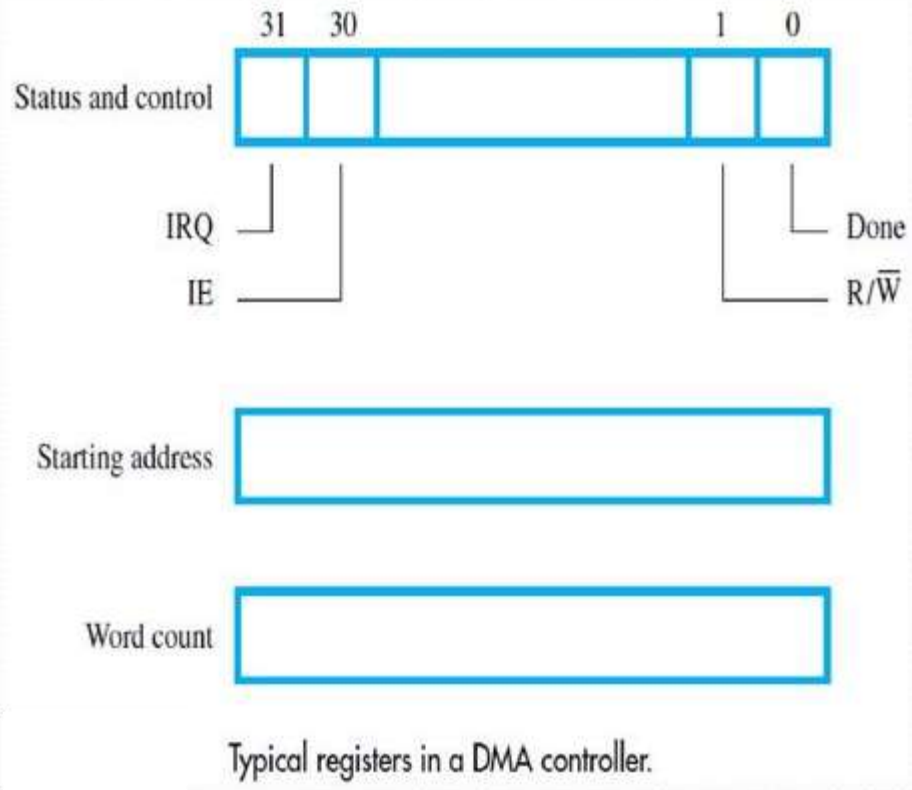
Direct Memory Access (DMA)

- Data transferred by executing instruction such as
- **MOV DATAIN,R0**
- An instruction to transfer input or output data is executed only after the processor determines that the IO devices is ready.
- To do this the processor either polls status flag in the device interface or waits for the device to send an interrupt request.
- In either case considerable overhead is incurred because several programs instructions must be executed for each data word transfer.
- When interrupts are used, that is the additional overhead associated with program counter and other state of information

- To transfer large blocks of data at high speed, an alternative approach is used.
- A special control unit provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor. This approach is called DMA.

- DMA transfers are performed by a control circuit that is part of the I/O device interface called **DMA controller**.
- **For each word transferred, processor provides the memory address and all the bus signals that control data transfer.**
- Since processor has to transfer blocks of data, the DMA controller must increment the memory address for successive words and keep track of the number of transfers.

- DMA controller can transfer data without intervention by the processor, but its operation under the control of a program executed by the processor.
- **To initiate the transfer of a block of words, the processor sends the starting address, the number of words in the block, and the direction of the transfer.**
- On receiving this information, the DMA controller proceeds to perform the requested operation.
- **When the entire block has been transferred, the controller informs the processor by raising an interrupt signal.**
- While a DMA transfer is taking place, the program that requested the transfer cannot continue, and the processor can be used to execute another program.
- After the DMA transfer is completed, the processor can return to the program that requested the transfer.



Typical registers in a DMA controller.

- DMA controller registers that are accessed by the processor to initiate transfer operations.
- Two registers are used for storing the Starting address and the word count.
- The third register contains status and control flags.
- The R/W bit determines the direction of the transfer.
  - When this bit is set to 1 by a program instruction, the controller performs a read operation, that is, it transfers data from the memory to the I/O device.
- When the controller has completed transferring a block of data and is ready to receive another command, it sets the Done flag to 1.
- Bit 30 is the Interrupt-enable flag, IE. When this flag is set to 1, it causes the controller to raise an interrupt after it has completed transferring a block of data.
- The controller sets the IRQ bit to 1 when it has requested an interrupt.

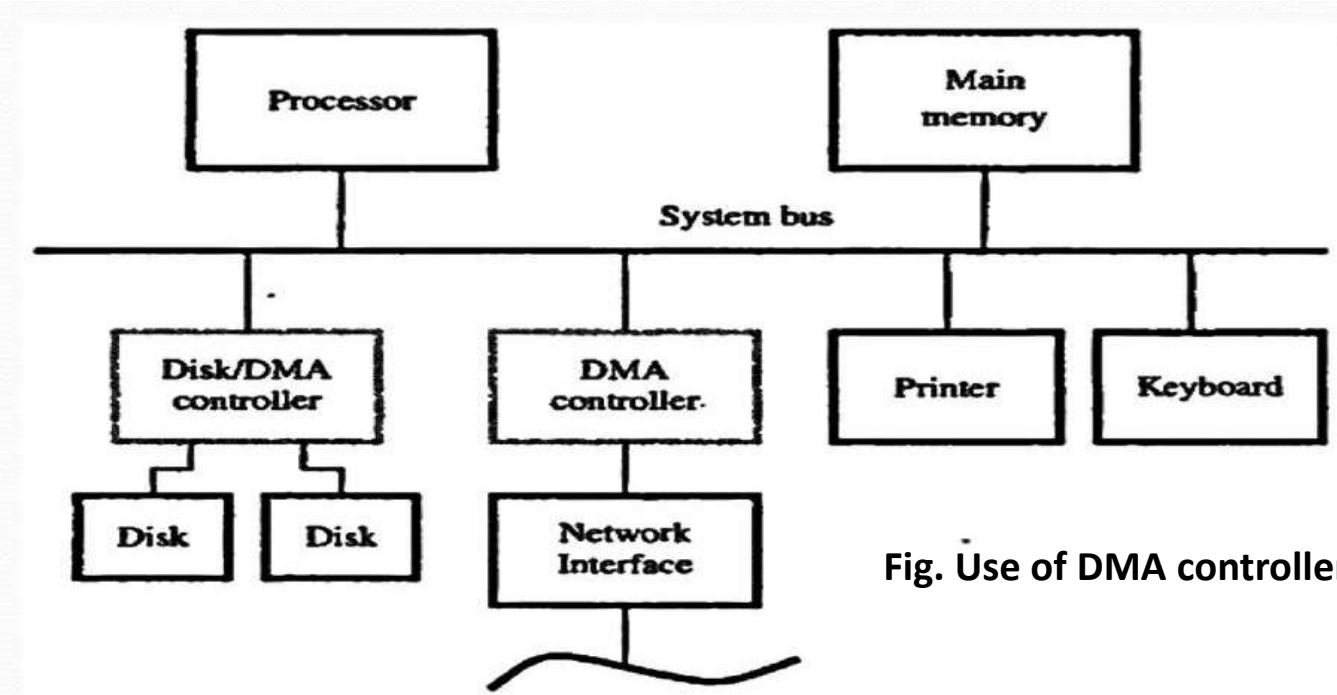


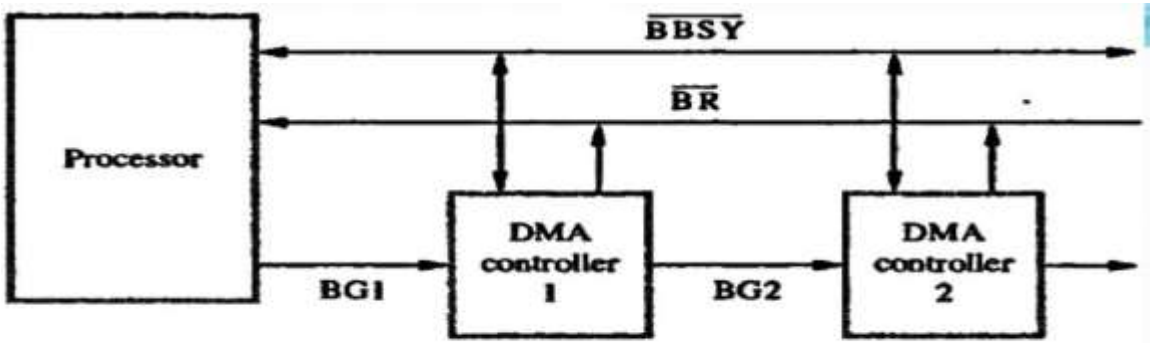
Fig. Use of DMA controllers in a PC

- Requests by DMA devices for using the bus are always given higher priority than processor requests.
- Among different DMA devices, top priority is given to high-speed peripherals such as a disk, a high-speed network interface, or a graphics display device.
- Since the processor originates most memory access cycles, the DMA controller can be said to "**steal**" **memory cycles from the processor**.
- **Interweaving technique is usually called cycle stealing**.
- Alternatively, the DMA controller may be given exclusive access to the main memory to transfer a block of data without interruption. This is known as **block or burst mode**.

Bus Arbitration

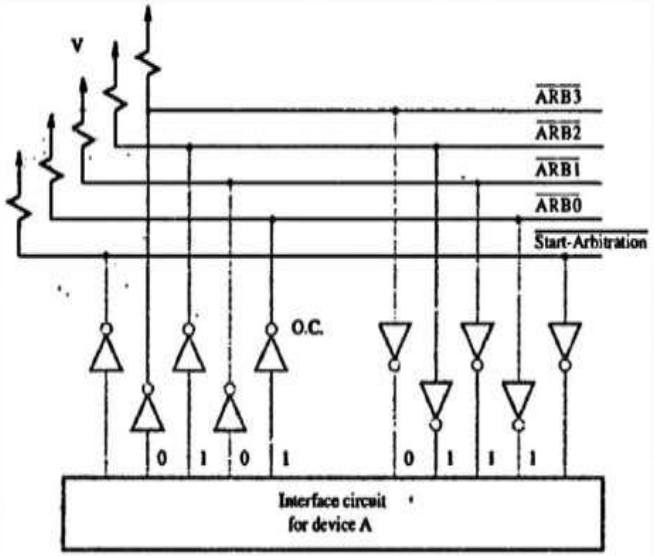
- A conflict may arise if both the processor and a DMA controller or two DMA controllers try to use the bus at the same time to access the main memory. To resolve these conflicts, an arbitration procedure is implemented on the bus to coordinate the activities of all devices requesting memory transfers.
  - The device that is allowed to initiate data transfers on the bus at any given time is called **the bus master**.
  - When the current master relinquishes control of the bus, another device can acquire this status.
  - **Bus arbitration** is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.
  - There are two approaches to bus arbitration: **centralized and distributed.**
  - In centralized arbitration, a single bus arbiter performs the required arbitration. In distributed arbitration, all devices participate in the selection of the next bus master.

- **Centralized Arbitration**
- The bus arbiter may be the processor or a separate unit connected to the bus. The processor is normally the bus master unless it grants bus mastership to one of the DMA controllers.
- A DMA controller indicates that it needs to become the bus master by activating the **Bus-Request line BR (Active Low)**.
- When **Bus-Request is activated,** the processor activates the **Bus-Grant signal, BG1**, indicating to the DMA controllers that they may use the bus when it becomes free.
- This signal is connected to all **DMA controllers using a daisy-chain arrangement**.
- The current bus master indicates to all device that it is using the bus by activating line called **Bus-Busy (Active Low)**
- Bus-Busy to prevent other devices from using the bus at the same time



A simple arrangement for bus arbitration using a daisy chain.

- **Distributed Arbitration**
- Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.
- Each device on the bus assigned a 4-bit identification number. When one or more devices request the bus, they assert the **Start Arbitration (Active Low)** signal and place their 4-bit ID numbers on four line, ARBo through ARB3 .
- A winner is selected as a result of the interaction among the signals transmitted over those liens by all contenders.
- The net outcome is that the code on the four lines represents the request that has the highest ID number.

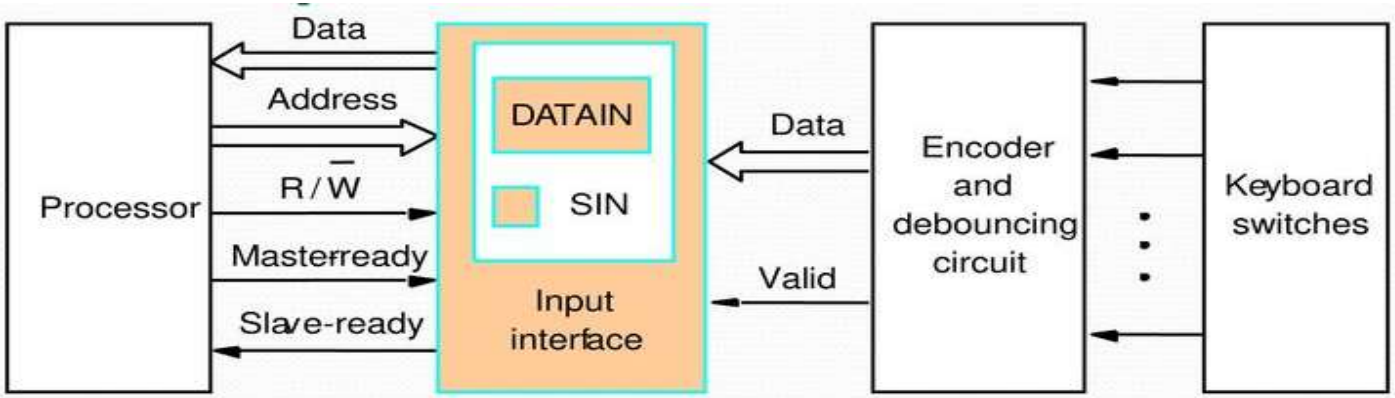


- Device A and B having ID 5 (0101) and 6 (0110).
- Both sent the ID
- Code seen by both device is 0111.
- Each device compares the pattern on the arbitration lines to it own ID, starting from MSB. If it detects a difference at any bit position, it disables its drivers at that bit position and for all lower order bits.

Interface circuits- Parallel Port, Serial Port

- I/O interface consists of the circuitry required to connect an I/O device to a computer bus.
- Side of the interface which connects to the computer has bus signals for:
  - Address,
  - Data
  - Control
- Side of the interface which connects to the I/O device has:
  - Datapath and associated controls to transfer data between the interface and the I/O device.
  - This side is called as a "port".
- Ports can be classified into two:
  - Parallel port,
  - Serial port.
- Parallel port transfers data in the form of a number of bits, normally 8 or 16 to or from the device.
- Serial port transfers and receives data one bit at a time.
- Processor communicates with the bus in the same way, whether it is a parallel port or a serial port.
  - Conversion from the parallel to serial and vice versa takes place inside the interface circuit.

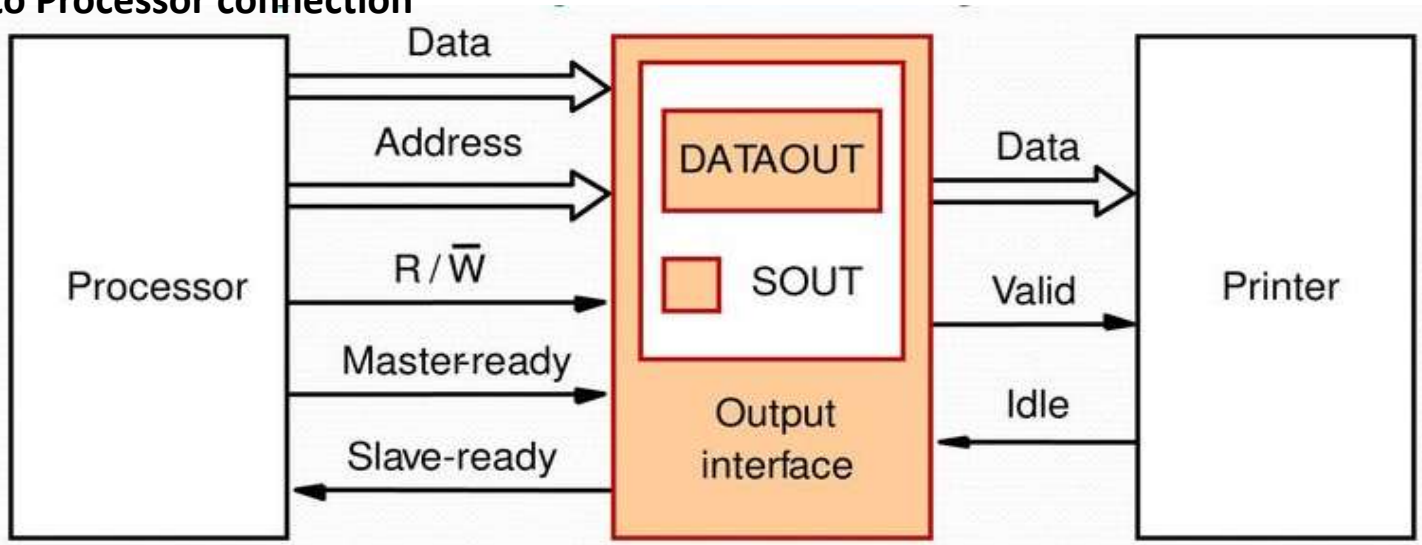
Parallel Port **KB to Processor connection**



- *Keyboard is connected to a processor using a parallel port.*
- *Processor is 32-bits and uses memory-mapped I/O and the asynchronous bus protocol.*
- *On the processor side of the interface we have:*
  - *Data lines.*
  - *Address lines*
  - *Control or R/W line.*
  - *Master-ready signal and*
  - *Slave-ready signal.*

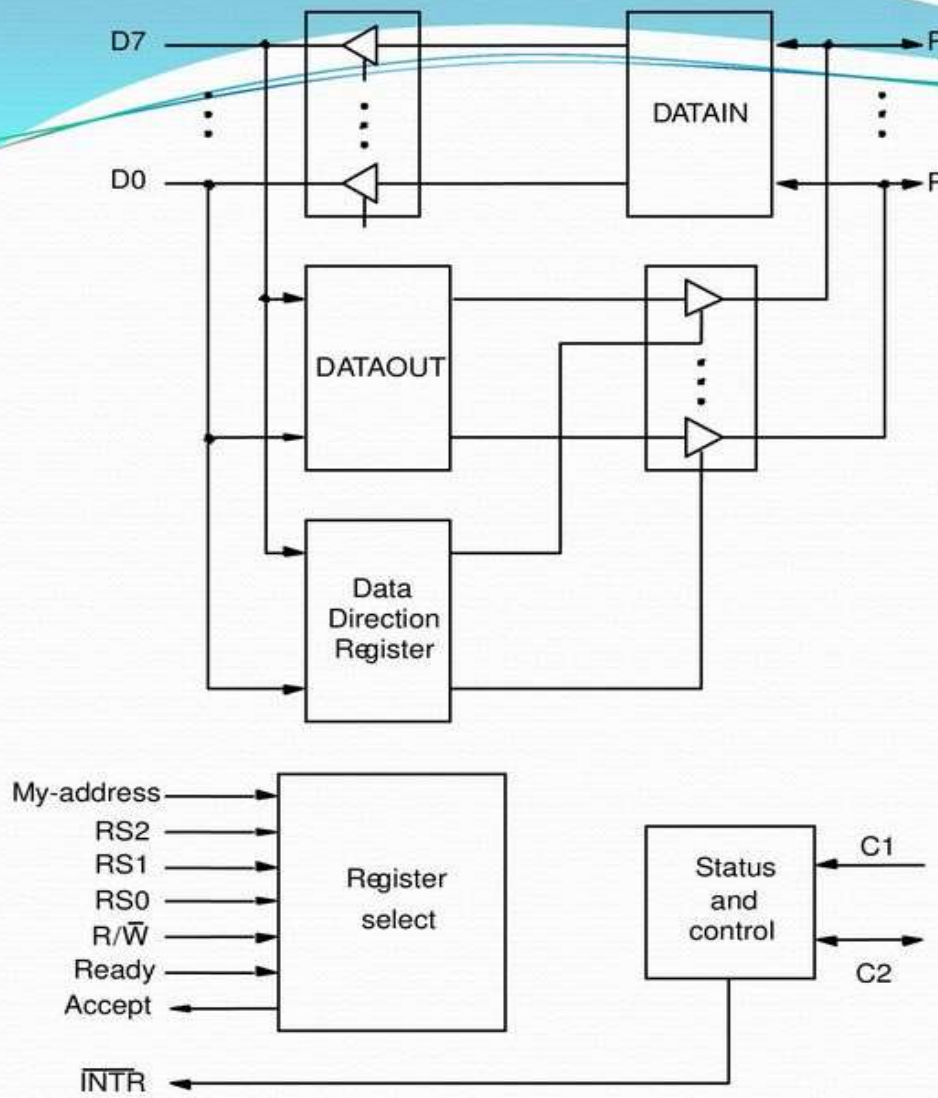
- *On the keyboard side of the interface:*
  - *Encoder circuit which generates a code for the key pressed.*
  - *Debouncing circuit which eliminates the effect of a key bounce (a single key stroke may appear as multiple events to a processor).*
  - *Data lines contain the code for the key.*
  - *Valid line changes from 0 to 1 when the key is pressed. This causes the code to be loaded into DATAIN and SIN to be set to 1.*

Printer to Processor connection



- *Printer is connected to a processor using a parallel port.*
- *Processor is 32 bits, uses memory-mapped I/O and asynchronous bus protocol.*
- *On the processor side:*
  - *Data lines.*
  - *Address lines*
  - *Control or R/W line.*
  - *Master-ready signal and*
  - *Slave-ready signal.*
- *On the printer side:*
  - *Idle signal line which the printer asserts when it is ready to accept a character. This causes the SOUT flag to be set to 1.*
  - *Processor places a new character into a DATAOUT register.*
  - *Valid signal, asserted by the interface circuit when it places a new character on the data lines.*

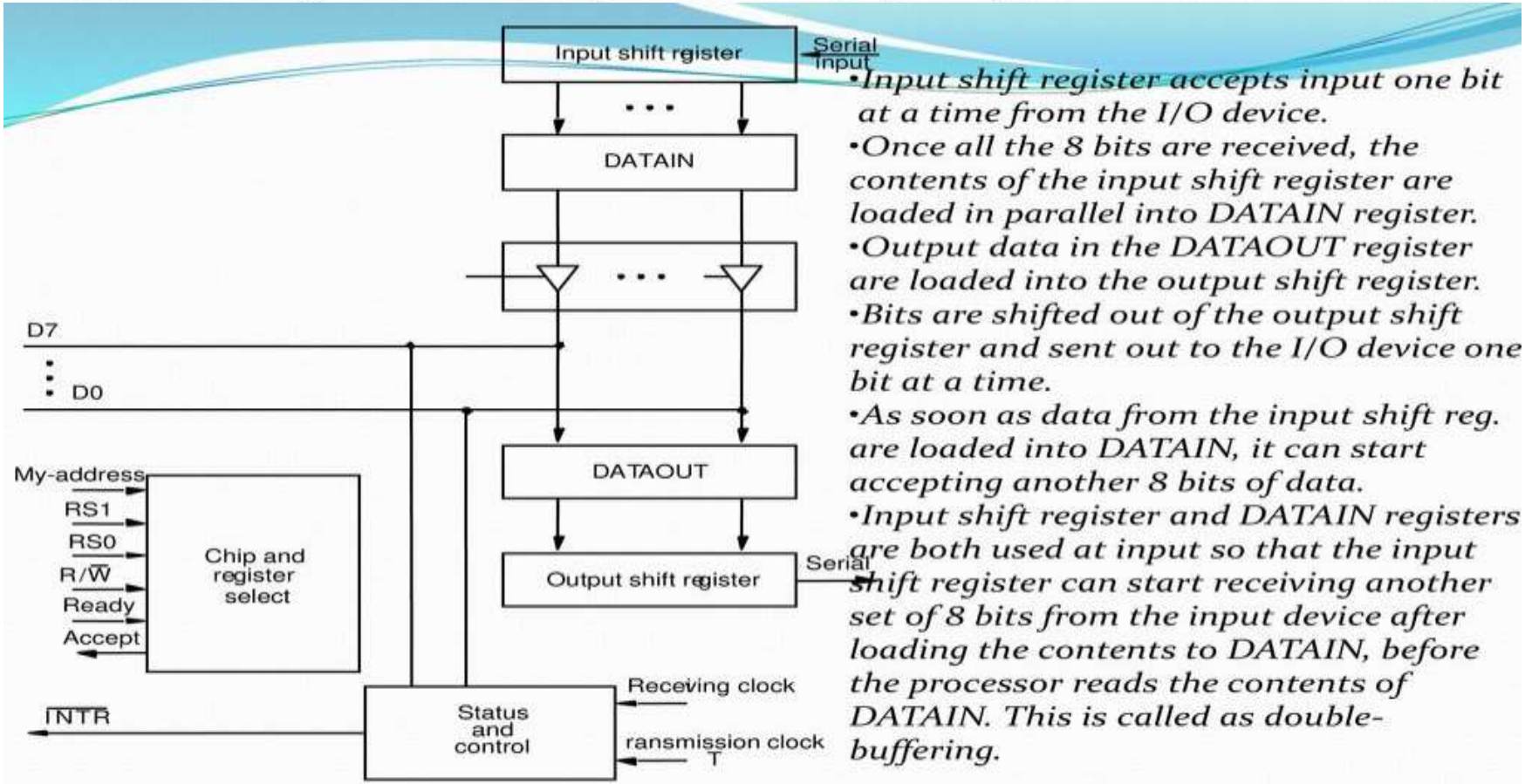
A general 8-bit parallel Interface



- *Data lines to I/O device are bidirectional.*
- *Data lines P7 through Po can be used for both input, and output.*
- *In fact, some lines can be used for input & some for output depending on the pattern in the Data Direction Register (DDR).*
- *Processor places an 8-bit pattern into a DDR*
- *If a given bit position in the DDR is 1, the corresponding data line acts as an output line, otherwise it acts as an input line.*
- *C1 and C2 control the interaction between the interface circuit and the I/O devices.*
- *Ready and Accept lines are the handshake control lines on the processor bus side, and are connected to Master-ready & Slave-ready*
- *Input signal My-address is connected to the output of an address decoder.*
- *Three register select lines that allow up to 8 registers to be selected.*

Serial Port - Serial Interface

- Serial port is used to connect the processor to I/O devices that require transmission of data one bit at a time.
- Serial port communicates in a bit-serial fashion on the device side and bit parallel fashion on the bus side.
  - Transformation between the parallel and serial formats is achieved with shift registers that have parallel access capability.



•Input shift register accepts input one bit at a time from the I/O device.
•Once all the 8 bits are received, the contents of the input shift register are loaded in parallel into DATAIN register.
•Output data in the DATAOUT register are loaded into the output shift register.
•Bits are shifted out of the output shift register and sent out to the I/O device one bit at a time.
•As soon as data from the input shift reg. are loaded into DATAIN, it can start accepting another 8 bits of data.
•Input shift register and DATAIN registers are both used at input so that the input shift register can start receiving another set of 8 bits from the input device after loading the contents to DATAIN, before the processor reads the contents of DATAIN. This is called as double-buffering.

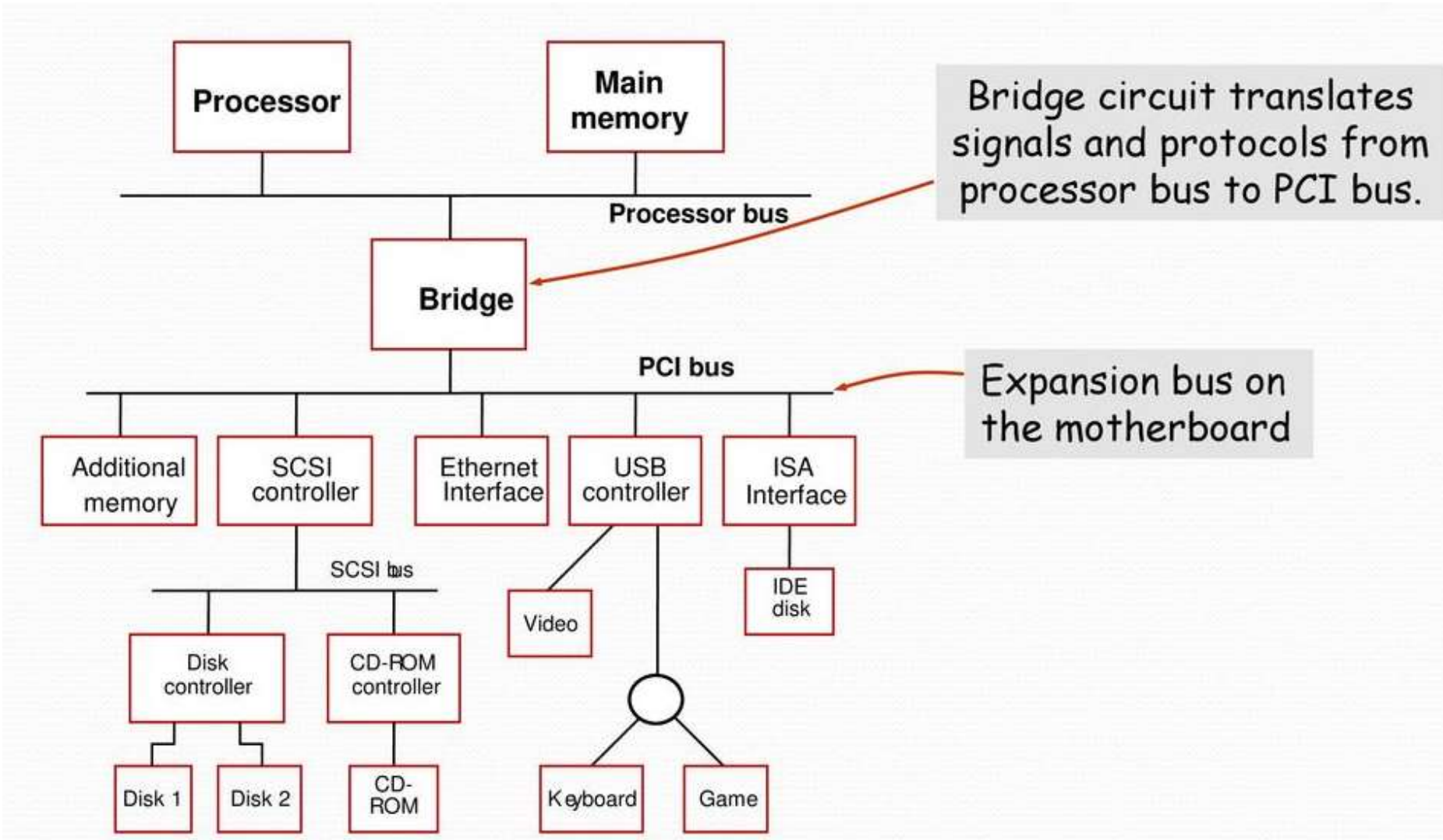
- Serial interfaces require fewer wires, and hence serial transmission is convenient for connecting devices that are physically distant from the computer.
- Speed of transmission of the data over a serial interface is known as the "bit rate".
  - Bit rate depends on the nature of the devices connected.
- In order to accommodate devices with a range of speeds, a serial interface must be able to use a range of clock speeds.
- Several standard serial interfaces have been developed:
  - Universal Asynchronous Receiver Transmitter (UART) for low-speed serial devices.
  - RS-232-C for connection to communication links.

Standard I/O Interfaces

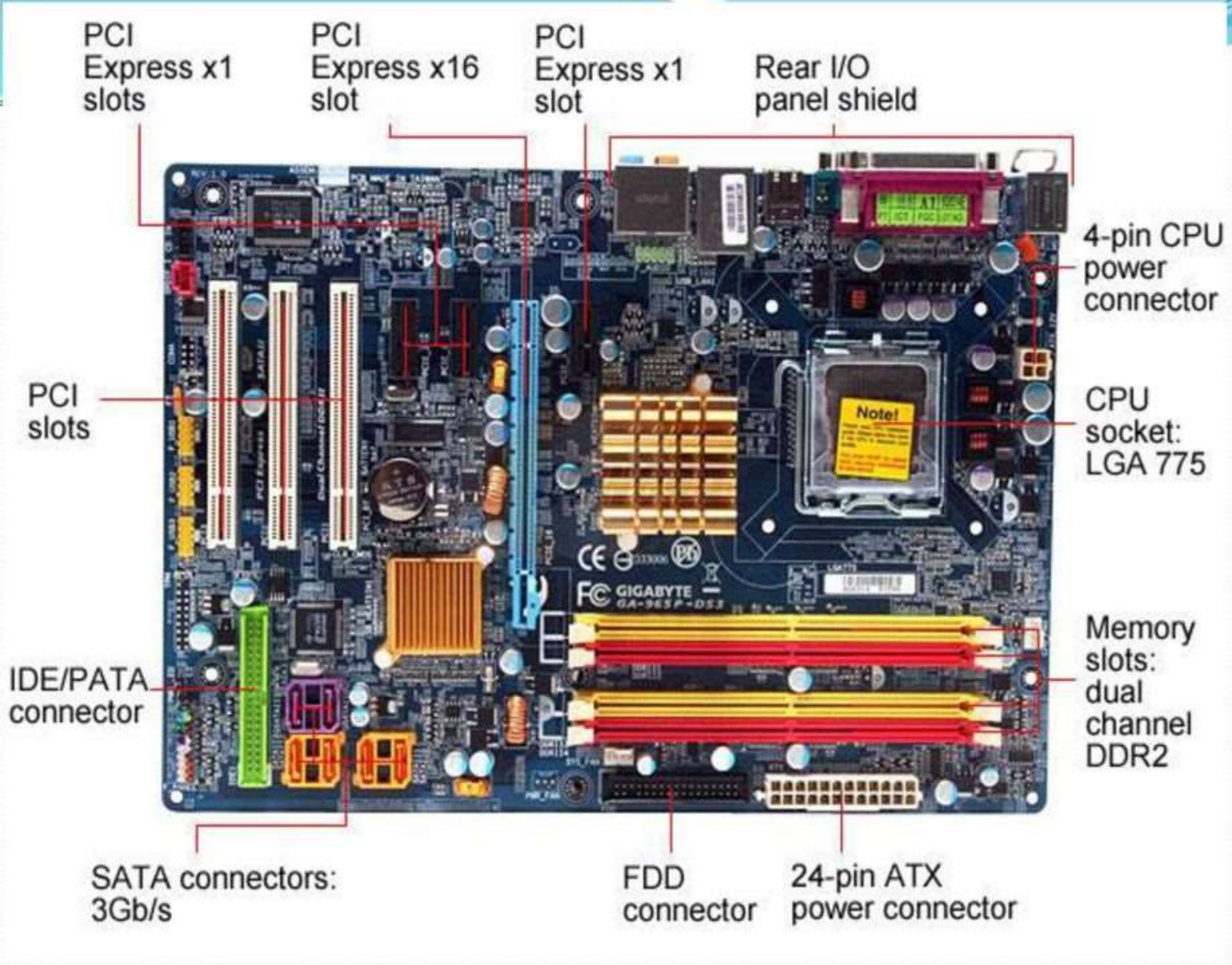
- I/O device is connected to a computer using an interface circuit.
- Do we have to design a different interface for every combination of an I/O device and a computer?
- A practical approach is to develop standard interfaces and protocols.
- A personal computer has:
  - A motherboard which houses the processor chip, main memory and some I/O interfaces.
  - A few connectors into which additional interfaces can be plugged.
- Processor bus is defined by the signals on the processor chip.
  - Devices which require high-speed connection to the processor are connected directly to this bus.

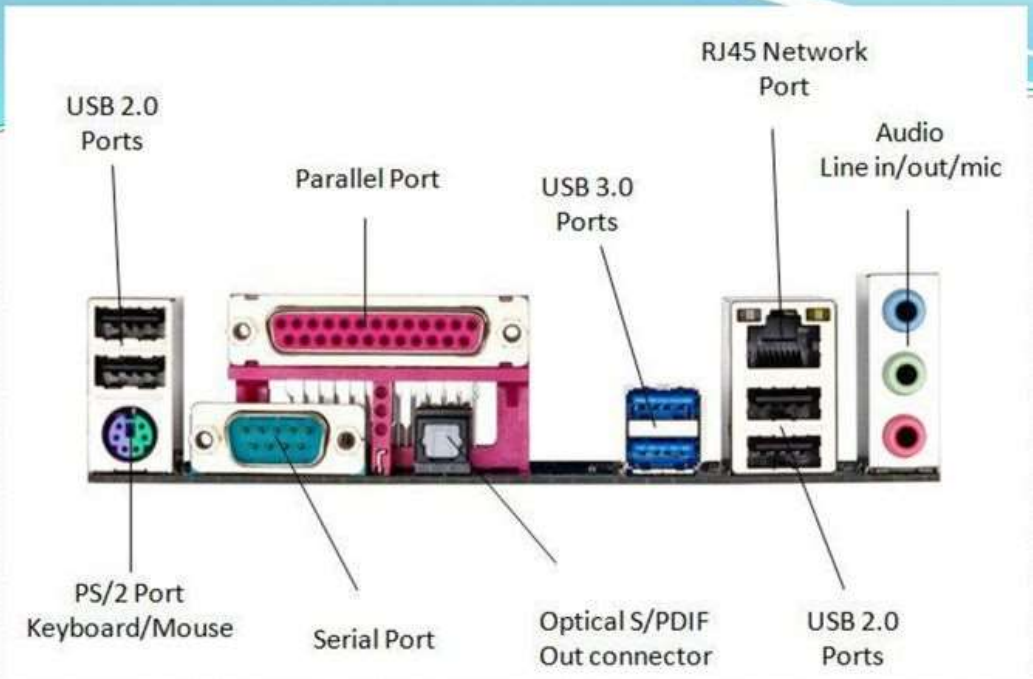
- Because of electrical reasons only a few devices can be connected directly to the processor bus.
- Motherboard usually provides another bus that can support more devices.
  - Processor bus and the other bus (called as expansion bus) are interconnected by a circuit called "bridge".
  - Devices connected to the expansion bus experience a small delay in data transfers.
- Design of a processor bus is closely tied to the architecture of the processor.
  - No uniform standard can be defined.
- Expansion bus however can have uniform standard defined.

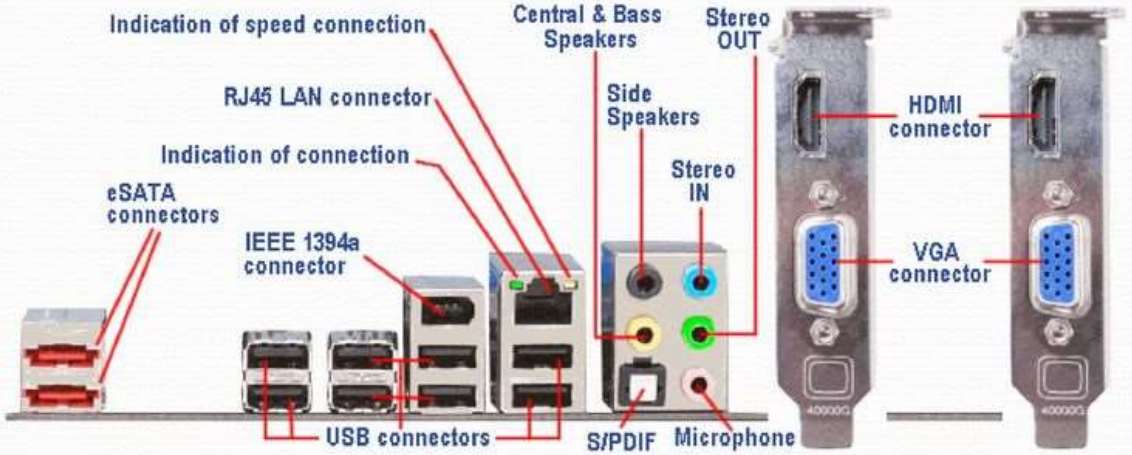
- A number of standards have been developed for the expansion bus.
  - Some have evolved by default.
  - For example, IBM's Industry Standard Architecture.
- Three widely used bus standards:
  - PCI (Peripheral Component Interconnect)
  - SCSI (Small Computer System Interface)
  - USB (Universal Serial Bus)



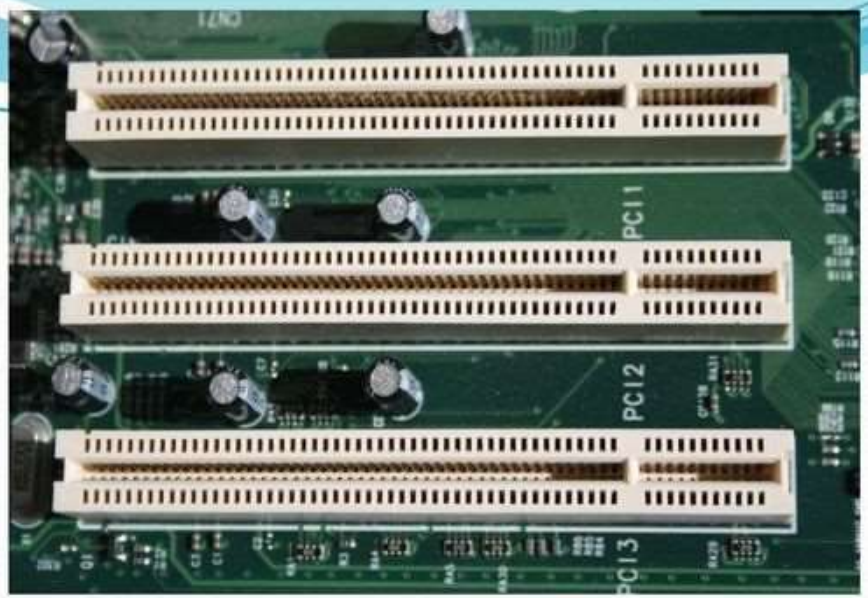
https://slideplayer.com/slide/13496322/


PCI Express x1 slots
PCI Express x16 slot
PCI Express x1 slot
Rear I/O panel shield
4-pin CPU power connector
PCI slots
CPU socket: LGA 775
Memory slots: dual channel DDR2
IDE/PATA connector
SATA connectors: 3Gb/s
FDD connector
24-pin ATX power connector
Note!
GIGABYTE
GA-965P-DS3


RJ45 Network Port
USB 2.0 Ports
Audio Line in/out/mic
Parallel Port
USB 3.0 Ports
PS/2 Port Keyboard/Mouse
Serial Port
Optical S/PDIF Out connector
USB 2.0 Ports


Indication of speed connection
Central & Bass Speakers
Stereo OUT
RJ45 LAN connector
Side Speakers
HDMI connector
Indication of connection
Stereo IN
eSATA connectors
IEEE 1394a connector
VGA connector
USB connectors
S/PDIF
Microphone

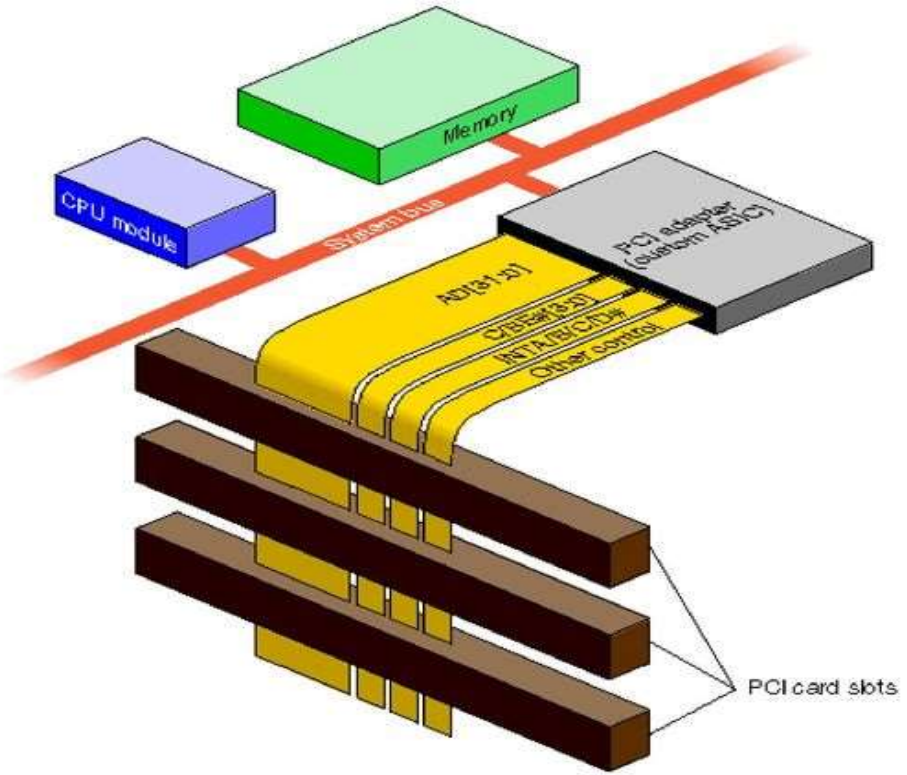
PCI Bus



- *Peripheral Component Interconnect*
- Introduced in 1992
- Low-cost bus
- Processor independent
- Plug-and-play capability
- In today's computers, most memory transfers involve a burst of data rather than just one word. The PCI is designed primarily to support this mode of operation.
- The bus supports three independent address spaces: memory, I/O, and configuration.
- We assumed that the master maintains the address information on the bus until data transfer is completed. But, the address is needed only long enough for the slave to be selected. Thus, the address is needed on the bus for one clock cycle only, freeing the address lines to be used for sending data in subsequent clock cycles. The result is a significant cost reduction.
- A master is called an initiator in PCI terminology. The addressed device that responds to read and write commands is called a target.

Data transfer signals on the PCI bus.

Name	Function
CLK	A 33-MHz or 6
FRAME#	Sent by the initi transaction.
AD	32 address/data increased to 64
C/BE#	4 command/byte-
IRDY#, TRDY#	Initiator-ready a
DEVSEL#	A response from recognized its a transfer transacti
IDSEL#	Initialization D

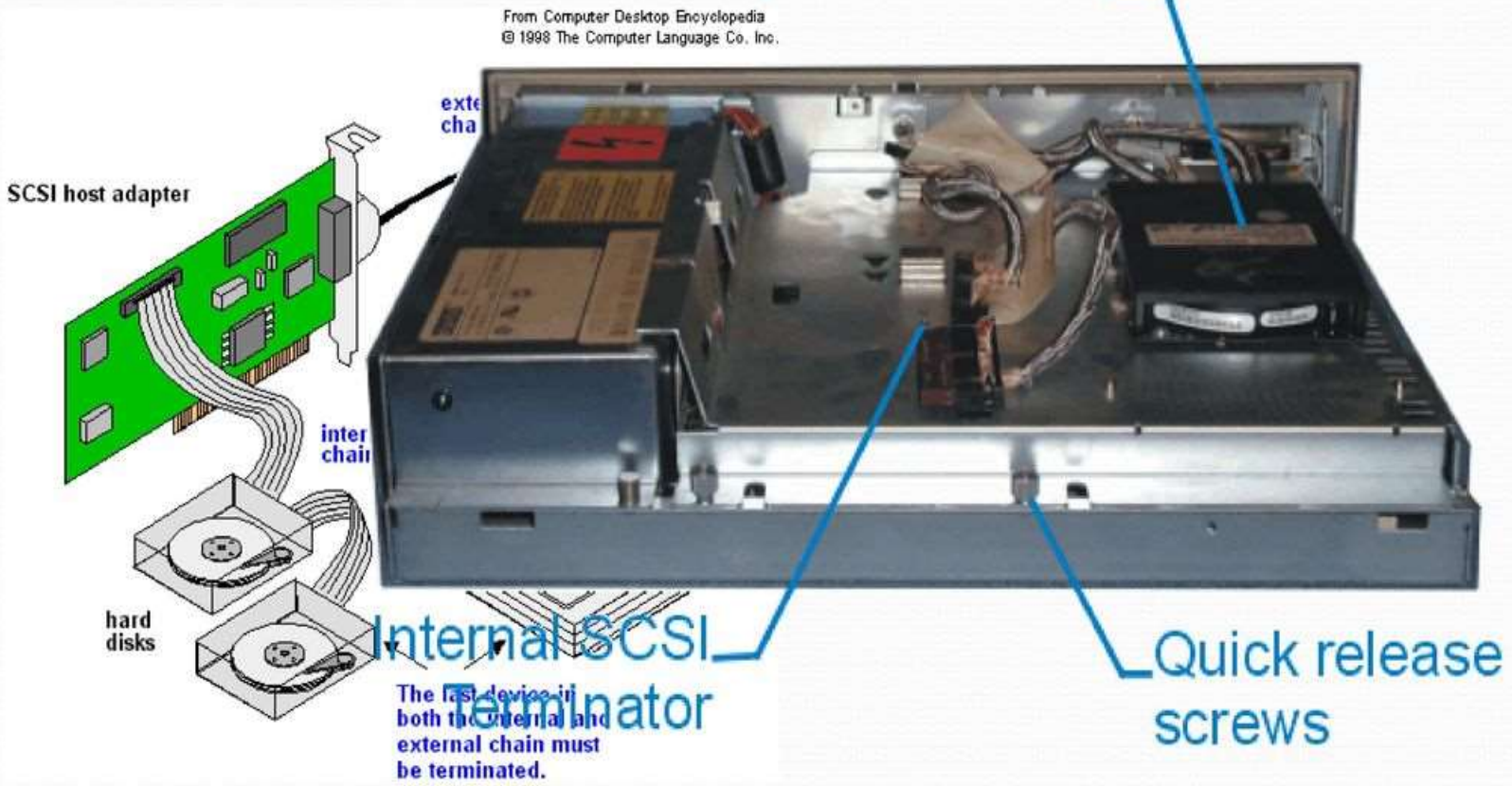


A signal whose name ends with the symbol # is asserted when in the low-voltage state.

SCSI Bus

- The acronym SCSI stands for Small Computer System Interface.
- It refers to a standard bus defined by the American National Standards Institute (ANSI) under the designation X3.131 .
- In the original specifications of the standard, devices such as disks are connected to a computer via a 50-wire cable, which can be up to 25 meters in length and can transfer data at rates up to 5 megabytes/s.
- The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years.
- SCSI-2 and SCSI-3 have been defined, and each has several options.
- Because of various options SCSI connector may have 50, 68 or 80 pins.

SCSI Hard Drive (SCSI BUS A)



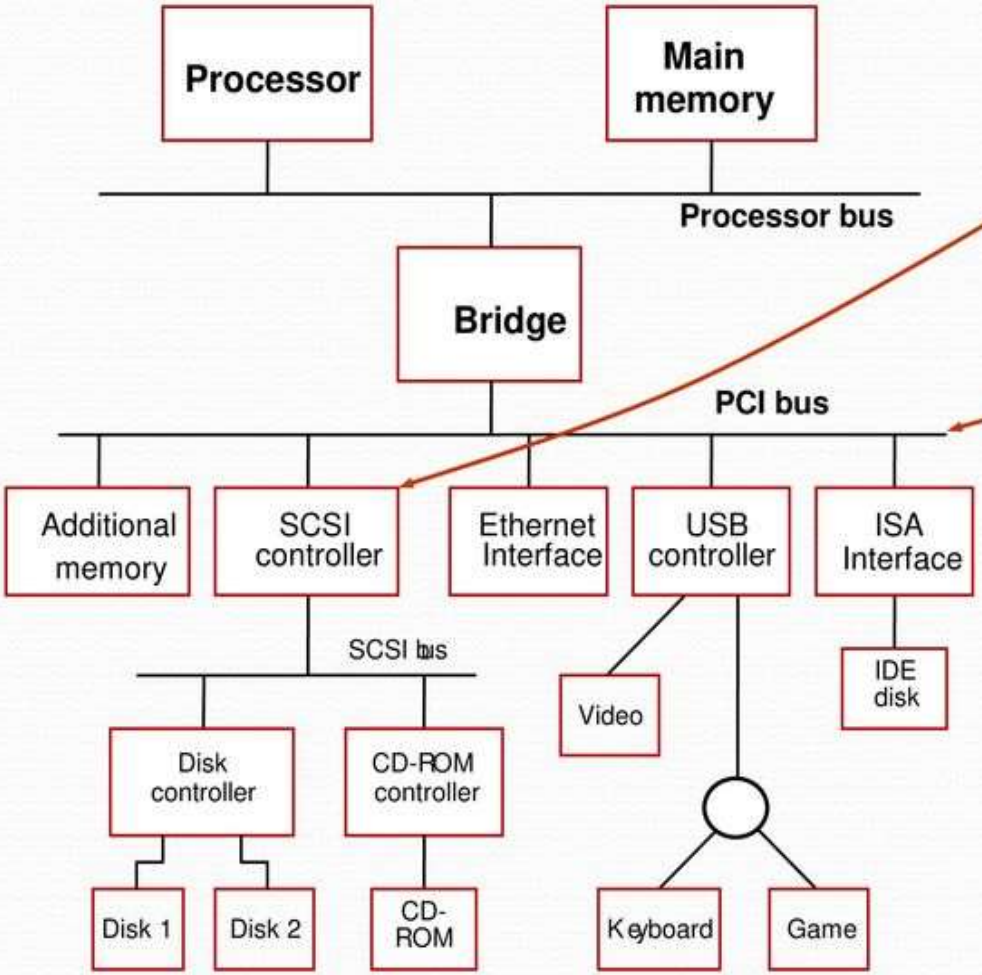
SCSI Bus (Contd.,)

- Devices connected to the SCSI bus are not part of the address space of the processor
- The SCSI bus is connected to the processor bus through a SCSI controller. This controller uses DMA to transfer data packets from the main memory to the device, or vice versa.
- A packet may contain a block of data, commands from the processor to the device, or status information about the device.
- A controller connected to a SCSI bus is one of two types – an initiator or a target.
- An initiator has the ability to select a particular target and to send commands specifying the operations to be performed. The disk controller operates as a target. It carries out the commands it receives from the initiator.
- The initiator establishes a logical connection with the intended target.
- Once this connection has been established, it can be suspended and restored as needed to transfer commands and bursts of data.
- While a particular connection is suspended, other device can use the bus to transfer information.
- This ability to overlap data transfer requests is one of the key features of the SCSI bus that leads to its high performance.

SCSI Bus (Contd.,)

- Data transfers on the SCSI bus are always controlled by the target controller.
- To send a command to a target, an initiator requests control of the bus and, after winning arbitration, selects the controller it wants to communicate with and hands control of the bus over to it.
- Then the controller starts a data transfer operation to receive a command from the initiator.

Standard I/O interfaces (contd..)



Bridge circuit translates signals and protocols from processor bus to SCSI bus.

Expansion bus on the motherboard

Bus Signals

Operation of SCSI bus from H/W point of view

Category	Name	Function
Data	– DB(0) to – DB(7)	Data lines: Carry one byte of information during the information transfer phase and identify device during arbitration, selection and reselection phases
	– DB(P)	Parity bit for the data bus
Phase	– BSY	Busy: Asserted when the bus is not free
	–SEL	Selection: Asserted during selection and reselection
Information type	– C/D	Control/Data: Asserted during transfer of control information (command, status or message)
	– MSG	Message: indicates that the information being transferred is a message

Table 4. The SCSI bus signals.

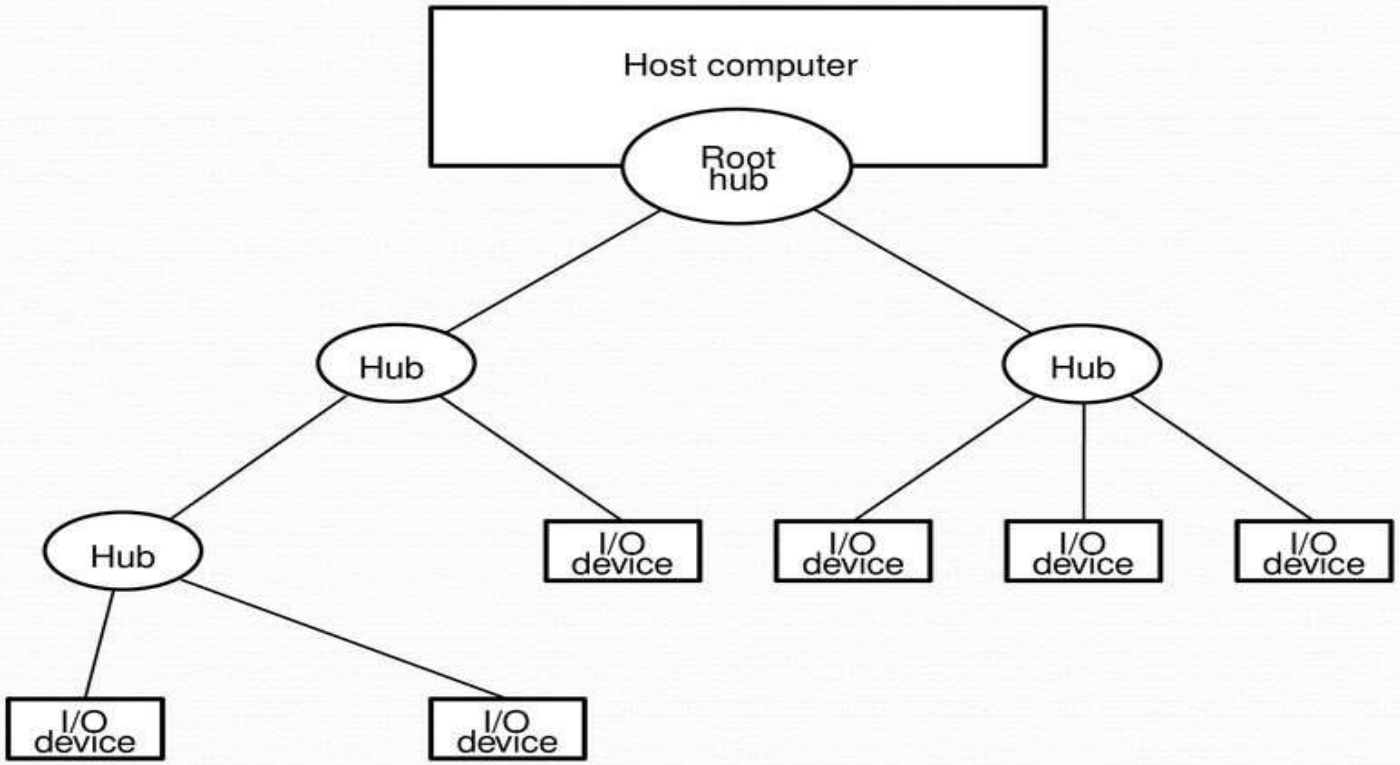
Table 4. The SCSI bus signals.(*cont.*)

Category	Name	Function
Handshake	− REQ	Request: Asserted by a target to request a data transfer cycle
	− ACK	Acknowledge: Asserted by the initiator when it has completed a data transfer operation
Direction of transfer	− I/O	Input/Output: Asserted to indicate an input operation (relative to the initiator)
Other	− ATN	Attention: Asserted by an initiator when it wishes to send a message to a target
	− RST	Reset: Causes all device cortrols to disconnect from the bus and assume their start-up state

USB

- Universal Serial Bus (USB) is an industry standard developed through a collaborative effort of several computer and communication companies, including Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, Nortel Networks, and Philips.
- Speed
  - Low-speed(1.5 Mb/s)
  - Full-speed(12 Mb/s)
  - High-speed(480 Mb/s)
- Port Limitation
- Device Characteristics
- Plug-and-play

Universal Serial Bus tree structure



USB Protocols

- All information transferred over the USB is organized in packets, where a packet consists of one or more bytes of information. There are many types of packets that perform a variety of control functions.
- The information transferred on the USB can be divided into two broad categories: control and data.
  - Control packets perform such tasks as addressing a device to initiate data transfer, acknowledging that data have been received correctly, or indicating an error.
  - Data packets carry information that is delivered to a device.
- A packet consists of one or more fields containing different kinds of information. The first field of any packet is called the packet identifier, PID, which identifies the type of that packet.
- They are transmitted twice. The first time they are sent with their true values, and the second time with each bit complemented
- The four PID bits identify one of 16 different packet types. Some control packets, such as ACK (Acknowledge), consist only of the PID byte.

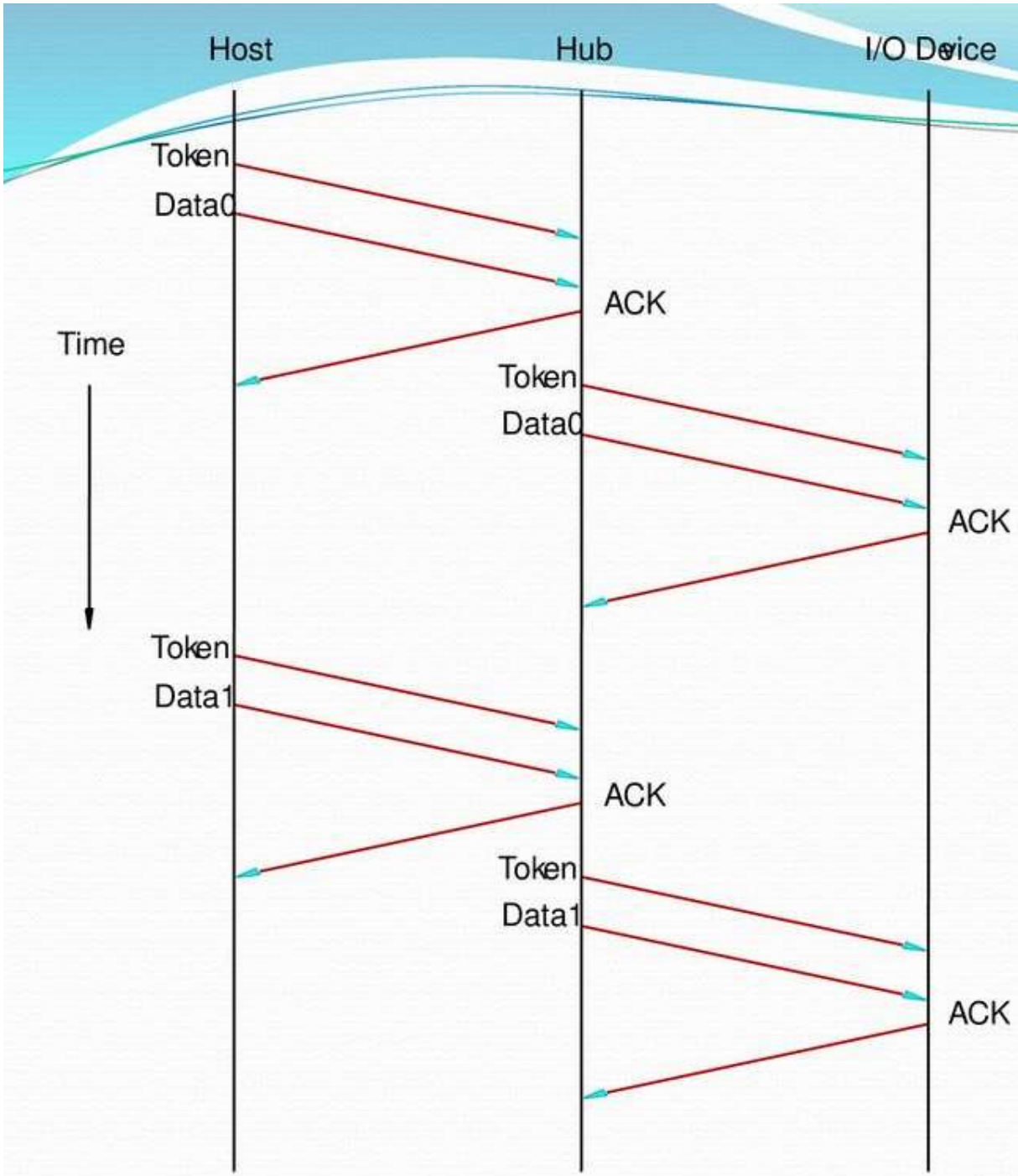


Figure: An output transfer

Unit IV: Arithmetic and Logical Unit computation **14M**

- ALU Inputs and outputs, number systems: Decimal, Binary, Hexadecimal notations.
- Integer representations: Sign magnitude, Two’s complementary addition and subtraction of numbers.
- Block diagram of hardware for addition and subtraction
- Hardware implementation of unsigned binary multiplication,
- Booths algorithm for two's complement multiplication.
- Hardwired control unit, Microprogrammed control unit.

1. ALU Inputs and outputs

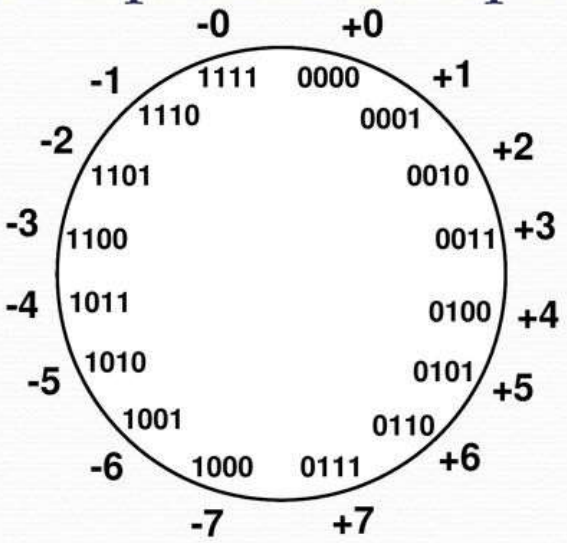
2. Integer Representations

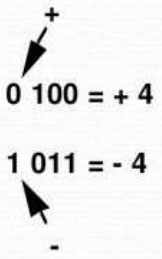
- 3 major representations:
  - Sign and magnitude
  - One's complement
  - Two's complement

Sign and Magnitude Representation

High order bit is sign: 0 = positive (or zero), 1 = negative
Three low order bits is the magnitude: 0 (000) thru 7 (111)
Number range for n bits = $+/-2^{n-1}$ -1
Two representations for 0

One's Complement Representation

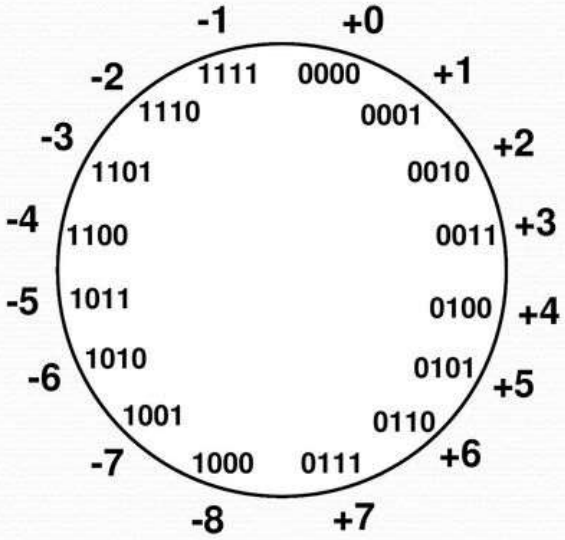


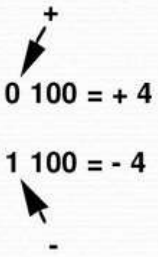


- Subtraction implemented by addition & 1's complement
- Still two representations of 0! This causes some problems
- Some complexities in addition

Two's Complement Representation

like 1's comp except shifted one position clockwise





- Only one representation for 0
- One more negative number than positive number

Binary, Signed-Integer Representations

B	Values represented	
$b_3 b_2 b_1 b_0$	1's complement	2's complement
0 1 1 1	+7	+7
0 1 1 0	+6	+6
0 1 0 1	+5	+5
0 1 0 0	+4	+4
0 0 1 1	+3	+3
0 0 1 0	+2	+2
0 0 0 1	+1	+1
0 0 0 0	+0	+0
1 0 0 0	-7	-8
1 0 0 1	-6	-7
1 0 1 0	-5	-6
1 0 1 1	-4	-5
1 1 0 0	-3	-4
1 1 0 1	-2	-3
1 1 1 0	-1	-2
1 1 1 1	-0	-1

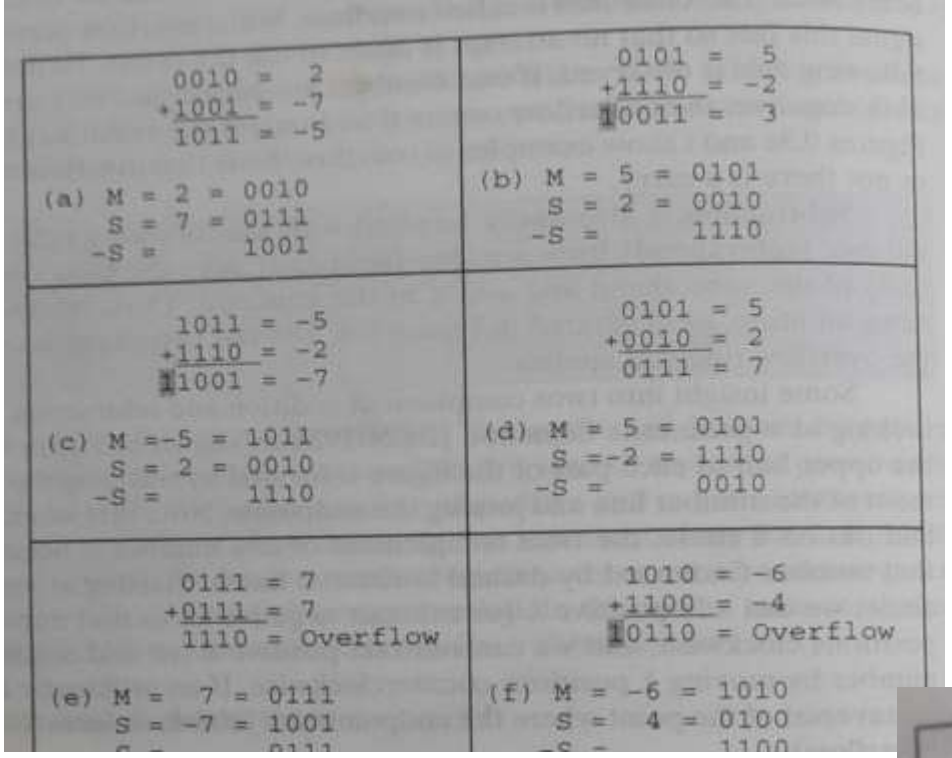
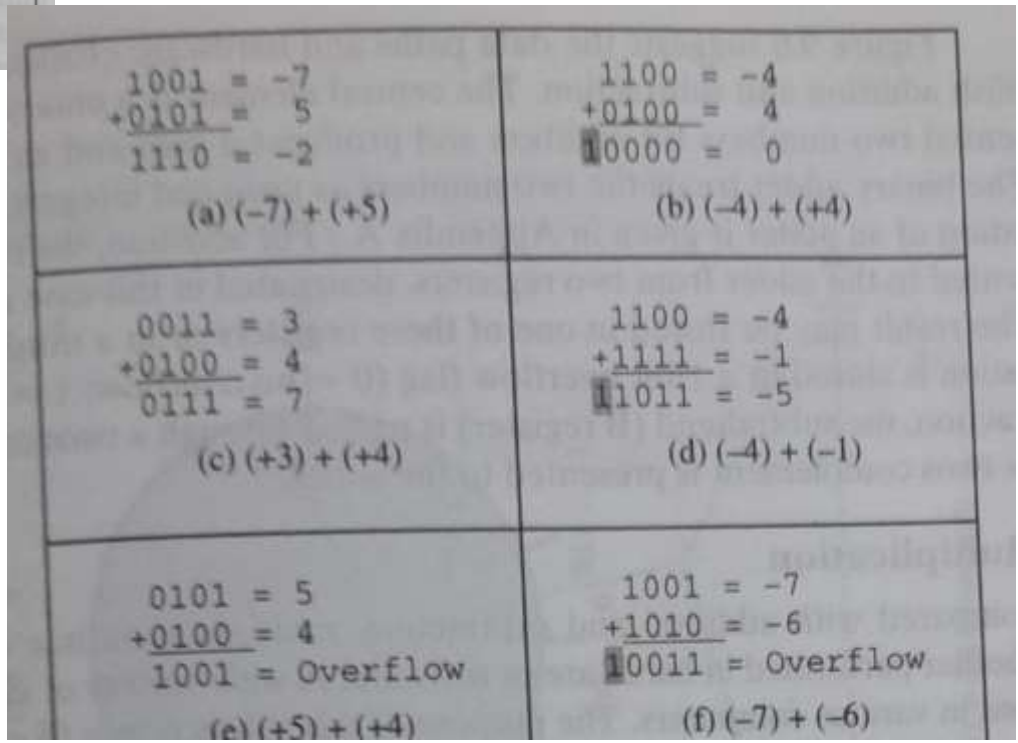


Fig. Subtraction of no's in twos complement representation

Fig. Addition of no's in twos complement representation

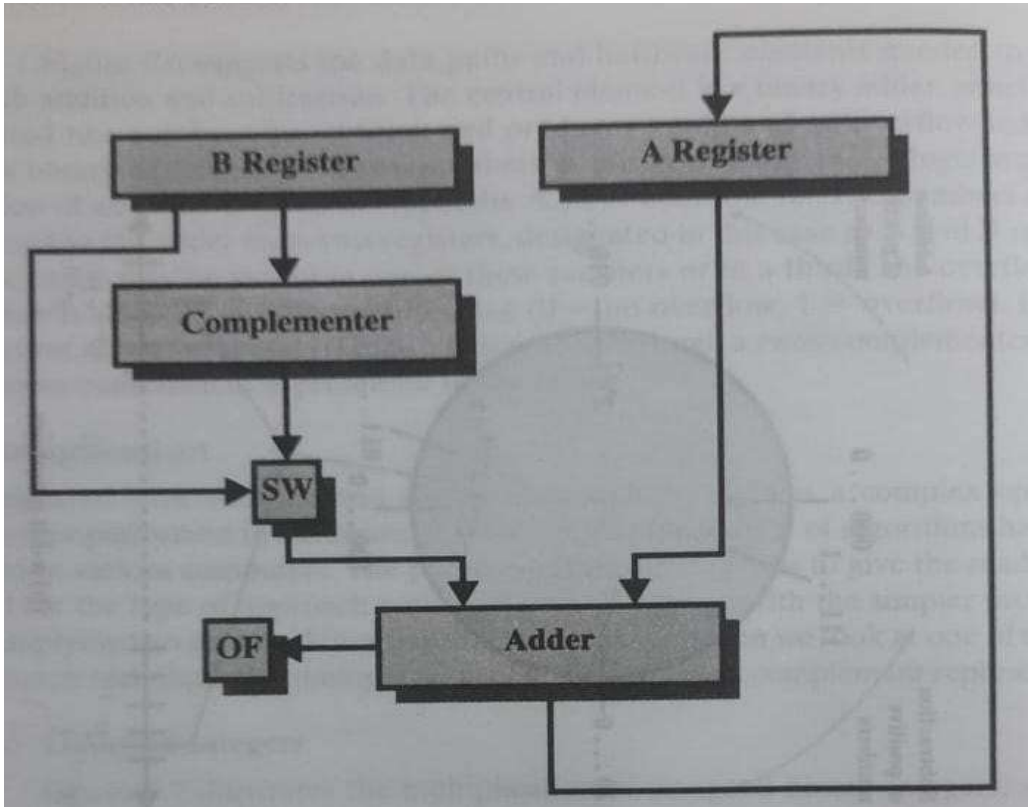


2's-Complement Add and Subtract Operations

(a)	0 0 1 0	(+2)		(b)	0 1 0 0	(+4)	
	+ 0 0 1 1	(+3)			+ 1 0 1 0	(- 6)	
	0 1 0 1	(+5)			1 1 1 0	(- 2)	
(c)	1 0 1 1	(- 5)		(d)	0 1 1 1	(+7)	
	+ 1 1 1 0	(- 2)			+ 1 1 0 1	(- 3)	
	1 0 0 1	(- 7)			0 1 0 0	(+4)	
(e)	1 1 0 1	(- 3)	⇒		1 1 0 1		
	- 1 0 0 1	(-7)			+ 0 1 1 1		
					0 1 0 0	(+4)	
(f)	0 0 1 0	(+2)	⇒		0 0 1 0		
	- 0 1 0 0	(+4)			+ 1 1 0 0		
					1 1 1 0	(- 2)	
(g)	0 1 1 0	(+6)	⇒		0 1 1 0		
	- 0 0 1 1	(+3)			+ 1 1 0 1		
					0 0 1 1	(+3)	
(h)	1 0 0 1	(- 7)	⇒		1 0 0 1		
	- 1 0 1 1	(- 5)			+ 0 1 0 1		
					1 1 1 0	(- 2)	
(i)	1 0 0 1	(- 7)	⇒		1 0 0 1		
	- 0 0 0 1	(+1)			+ 1 1 1 1		
					1 0 0 0	(- 8)	
(j)	0 0 1 0	(+2)	⇒		0 0 1 0		
	- 1 1 0 1	(- 3)			+ 0 0 1 1		
					0 1 0 1	(+5)	

Figure 2.4. 2's-complement Add and Subtract operations.

3. Block diagram of Hardware for addition and subtraction



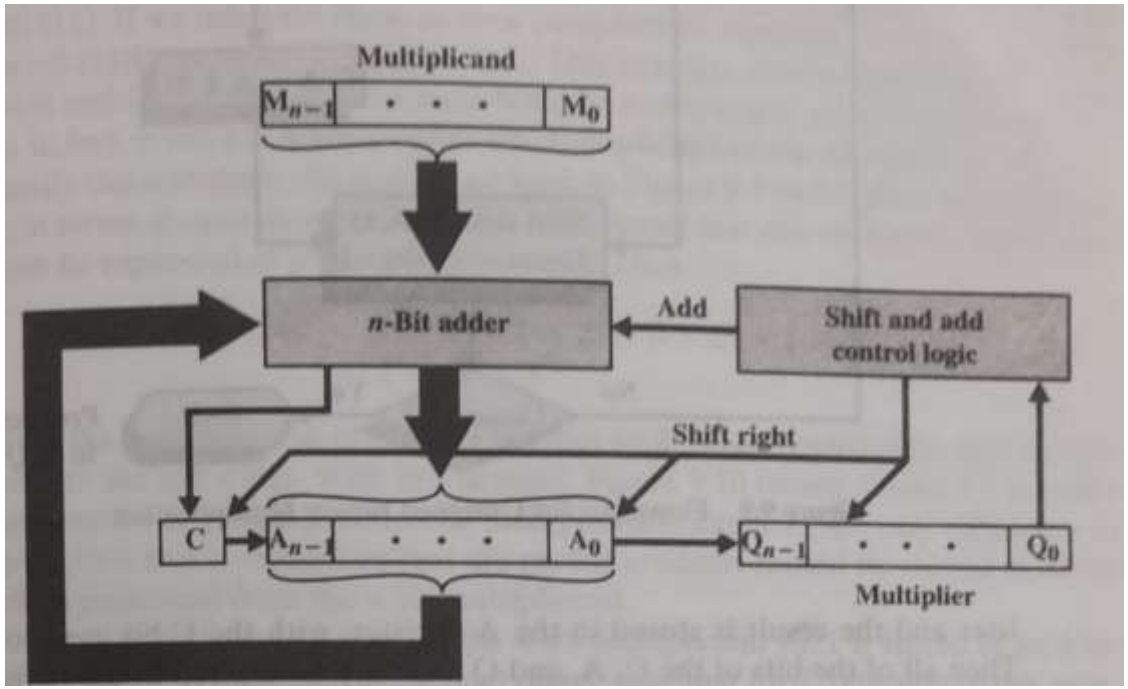
Block diagram of Hardware for addition & Subtraction

- Data paths and hardware elements needed to accomplish addition & subtraction. The central element is a binary adder, presented two numbers for addition and produces a sum and an overflow indication.
- For addition register A & B presented to the adder for addition. The result may be stored either one of register or in third.
- The overflow indication is stored in one bit flag, OF.
- For subtraction, the subtrahend (B register) is passed through a 2's complementer & is presented to adder.

4. Hardware implementation of unsigned binary multiplication

1. Multiplication involves the generation of partial products, one for each digit in the multiplier. These partial products are then summed to produce the final product.
2. The partial products are easily defined. When the multiplier bit is 0, the partial product is 0. When the multiplier is 1, the partial product is the multiplicand.
3. The total product is produced by summing the partial products. For this operation, each successive partial product is shifted one position to the left relative to the preceding partial product.
4. The multiplication of two n-bit binary integers results in a product of up to $2n$ bits in length (e.g., $11 \times 11 = 1001$).

```
  1011      Multiplicand (11)
×1101       Multiplier   (13)
------
  1011   }
  0000   }
 1011    }  Partial products
1011     }
--------
10001111    Product (143)
```



Block Diagram: **Hardware implementation of unsigned binary multiplication**

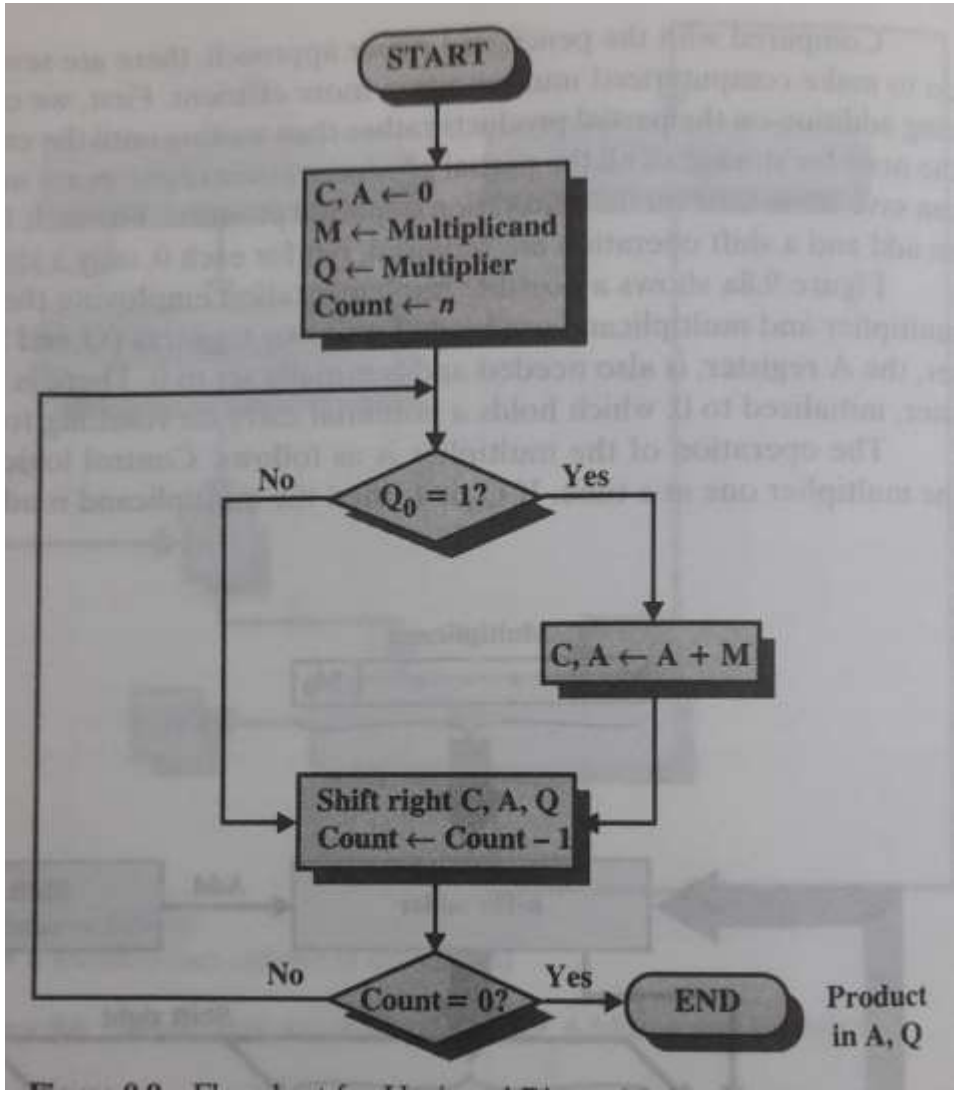


Fig. Flowchart for unsigned Binary Multiplication

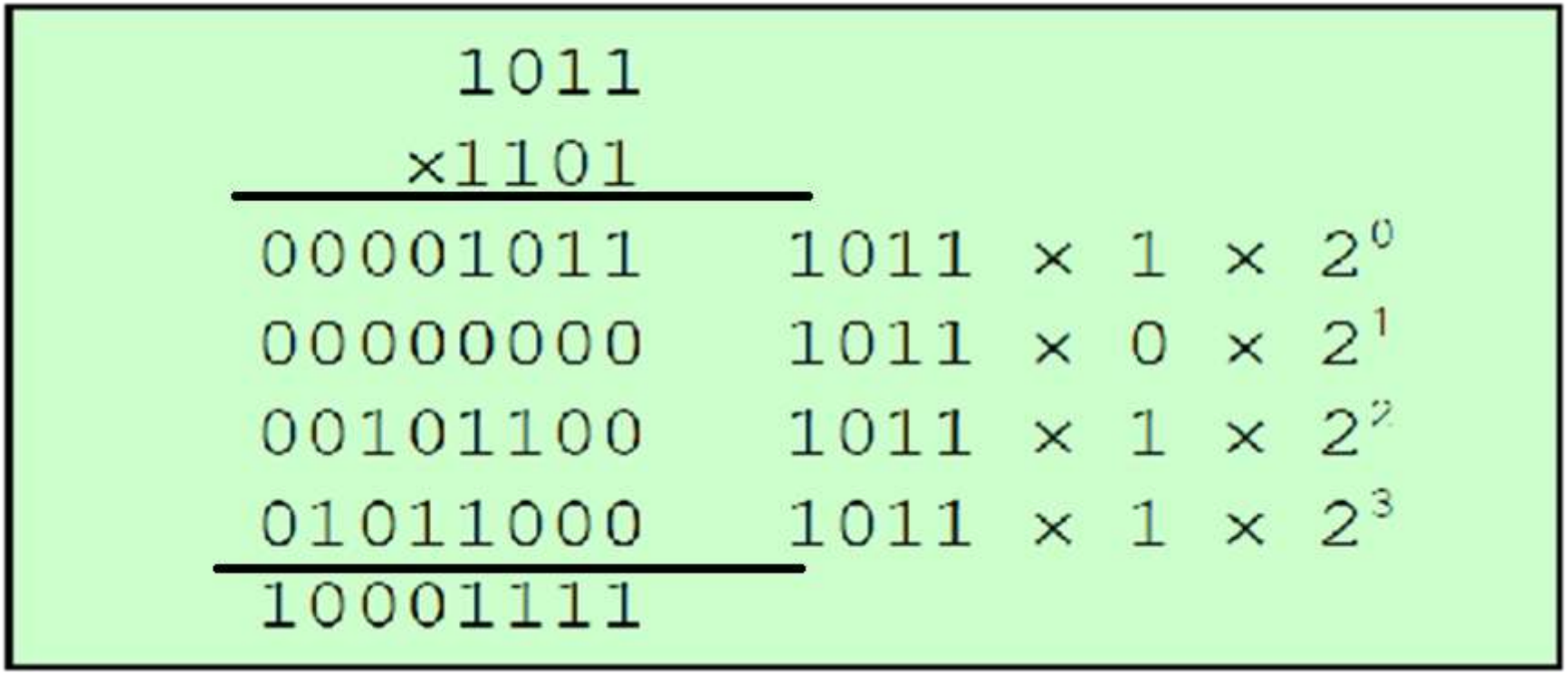
Algorithm

- The multiplier (Q) and Multiplicand (M) is loaded in two registers. Third register (A) is needed and initially set to 0.
- There is also a 1 bit register (C) ,initialized to 0 holds a potential carry bit resulting from addition.
- Control logic reads the bits of the multiplier one at a time. If Q_0=1, then the multiplicand is added to the A register and the result is stored in the A register, with the C bit is used for overflow.
- Then all of the bits of C, A, Q registers are shifted to the right one bit, so that C bit goes to A_{n-1}, A_0 goes into Q_{n-1} and Q_0 is lost.
- If Q_0=0, then no addition is performed, just the shift. This process is repeated for each bit of the original multiplier.
- The resulting 2n-bit product is contained in A & Q register.

Hardware implementation of unsigned binary multiplication

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

5.Booths algorithm for two's complement Multiplication



Multiplication of Two Unsigned 4-Bit Integers Yielding an 8-Bit Result

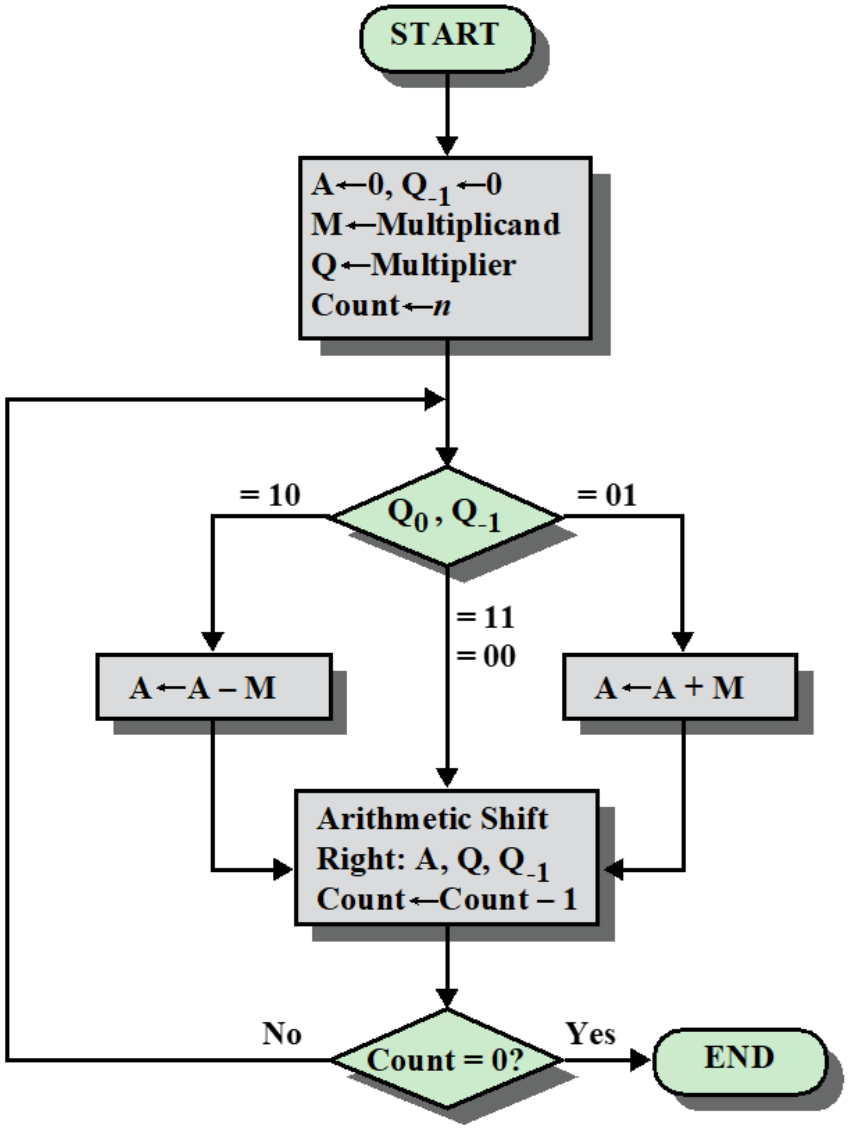
https://github.com/mguludag/Stallings-Computer-Org-and-Arch-Slides/blob/master/CH10-COA10e.pptx

1001 (9) ×0011 (3) 00001001 1001 × 2^0 00010010 1001 × 2^1 00011011 (27)	1001 (−7) ×0011 (3) 11111001 (−7) × 2^0 = (−7) 11110010 (−7) × 2^1 = (−14) 11101011 (−21)

(a) Unsigned integers (b) Twos complement integers

Comparison of Multiplication of Unsigned and Twos Complement Integers

- There are a number of ways out of this dilemma.
- **One would be to convert both multiplier and multiplicand to positive numbers, perform the multiplication, and then take the twos complement of the result** if and only if the sign of the two original numbers differed.
- Implementers have preferred to use techniques that do not require this final transformation step. One of the most common of these is Booth's algorithm.
- This algorithm also has the benefit of speeding up the multiplication process, relative to a more straightforward approach.



Flowchart: Booths algorithm for Two's complementary Multiplication

Booth's Algorithm

- The multiplier and multiplicand are placed in the Q and M registers, respectively.
- There is also a 1-bit register placed logically to the right of the least significant bit (Q_0) of the Q register and designated Q_{-1};
- The results of the multiplication will appear in the A and Q registers. A and Q_{-1} are initialized to 0.
- As before, control logic scans the bits of the multiplier one at a time. Now, as each bit is examined, the bit to its right is also examined.
- If the two bits are the same (1–1 or 0–0), then all of the bits of the A, Q, and Q_{-1} registers are shifted to the right 1 bit.
- If the two bits differ, then the multiplicand is added to or subtracted from the A register, depending on whether the two bits are 0–1 or 1–0.
- Following the addition or subtraction, the right shift occurs.
- In either case, the right shift is such that the leftmost bit of A, namely A_{n-1}, not only is shifted into A_{n-2}, but also remains in A_{n-1}.
- This is required to preserve the sign of the number in A and Q. It is known as an arithmetic shift, because it preserves the sign bit.

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A ← A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A ← A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

```
   0111
  ×0011     (0)
11111001    1-0
0000000     1-1
000111      0-1
--------
00010101    (21)
```

(a) (7) × (3) = (21)

```
   0111
  ×1101     (0)
11111001    1-0
0000111     0-1
111001      1-0
--------
11101011    (-21)
```

(b) (7) × (–3) = (–21)

```
   1001
  ×0011     (0)
--------
00000111    1-0
0000000     1-1
111001      0-1
11101011    (-21)
```

(c) (–7) × (3) = (–21)

```
   1001
  ×1101     (0)
--------
00000111    1-0
1111001     0-1
000111      1-0
00010101    (21)
```

(d) (–7) × (–3) = (21)

Examples Using Booth's Algorithm

Fundamental Concepts

- Processor fetches one instruction at a time and perform the operation specified.
- Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.
- Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).
- Instruction Register (IR)

Executing an Instruction

- Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).

$$IR \leftarrow [[PC]]$$

- Assuming that the memory is byte addressable, increment the contents of the PC by 4 (fetch phase).

$$PC \leftarrow [PC] + 4$$

- Carry out the actions specified by the instruction in the IR (execution phase).

Internal organization of the processor

- ALU
- Registers for temporary storage
- Various digital circuits for executing different micro operations.(gates, MUX,decoders,counters).
- Internal path for movement of data between ALU and registers.
- Driver circuits for transmitting signals to external units.
- Receiver circuits for incoming signals from external units.

- PC:
  - Keeps track of execution of a program
  - Contains the memory address of the next instruction to be fetched and executed.

MAR:

- Holds the address of the location to be accessed.
- I/P of MAR is connected to Internal bus and an O/p to external bus.

MDR:

- Contains data to be written into or read out of the addressed location.
- IT has 2 inputs and 2 Outputs.
- Data can be loaded into MDR either from memory bus or from internal processor bus.

The data and address lines are connected to the internal bus via MDR and MAR

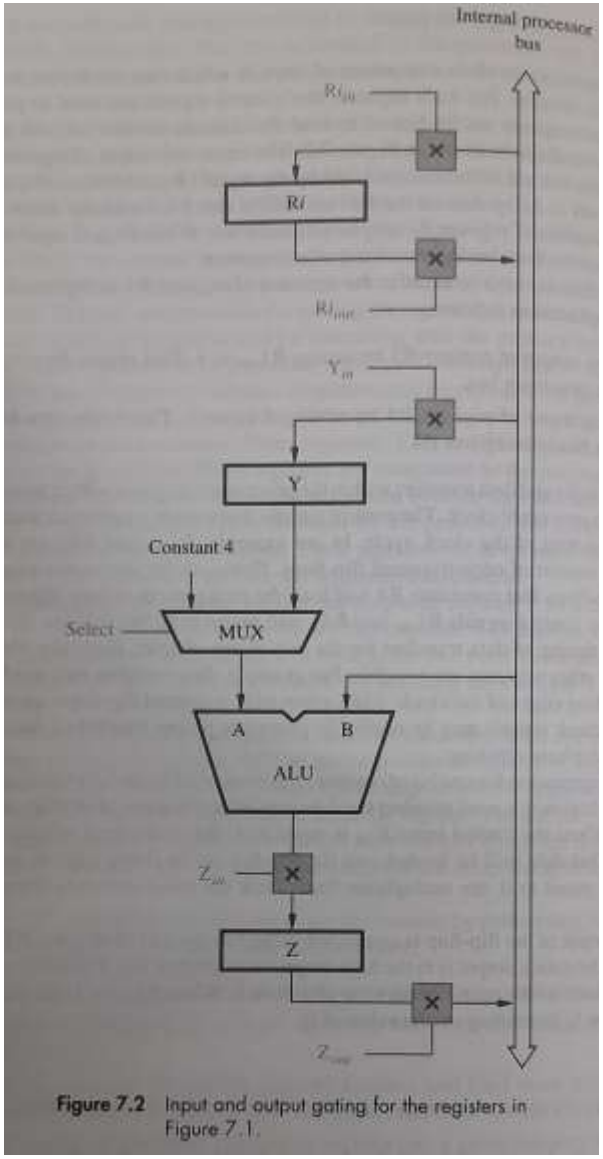


Figure 7.2 Input and output gating for the registers in Figure 7.1.

- The input and output gates for register Ri are controlled by signals isR_{in} and R_{iout} .
- R_{in} Is set to1 – data available on common bus are loaded into Ri.
- R_{iout} Is set to1 – the contents of register are placed on the bus.
- R_{iout} Is set to 0 – the bus can be used for transferring data from other registers .

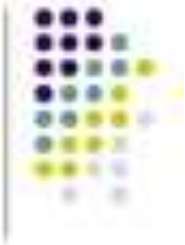
Data transfer between two registers:

EX:

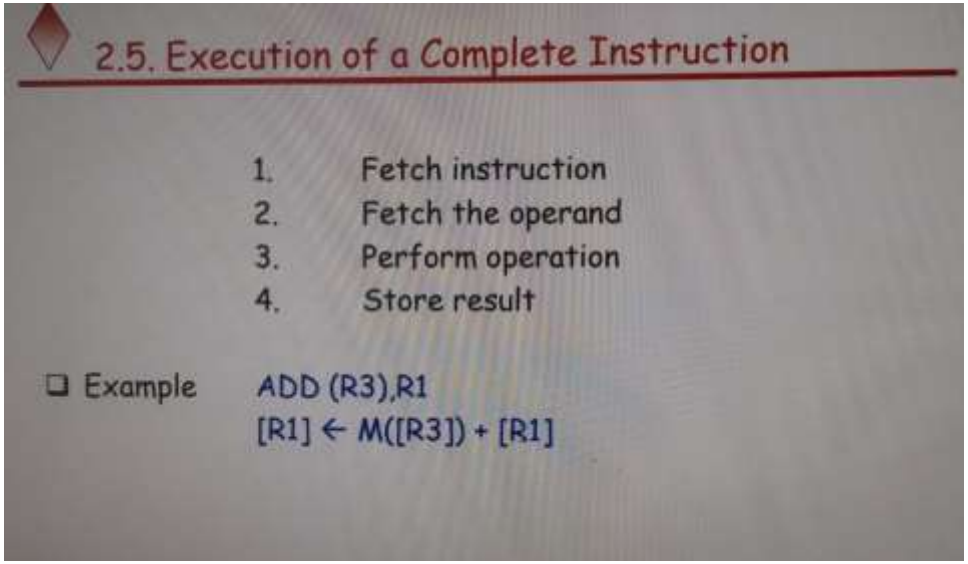
Transfer the contents of R1 to R4.

1. Enable output of register R1 by setting R1out=1. This places the contents of R1 on the processor bus.
2. Enable input of register R4 by setting R4in=1. This loads the data from the processor bus into register R4.

Execution of a Complete Instruction



- Add (R3), R1
- Fetch the instruction
- Fetch the first operand (the contents of the memory location pointed to by R3)
- Perform the addition
- Load the result into R1



Execution of a Complete Instruction

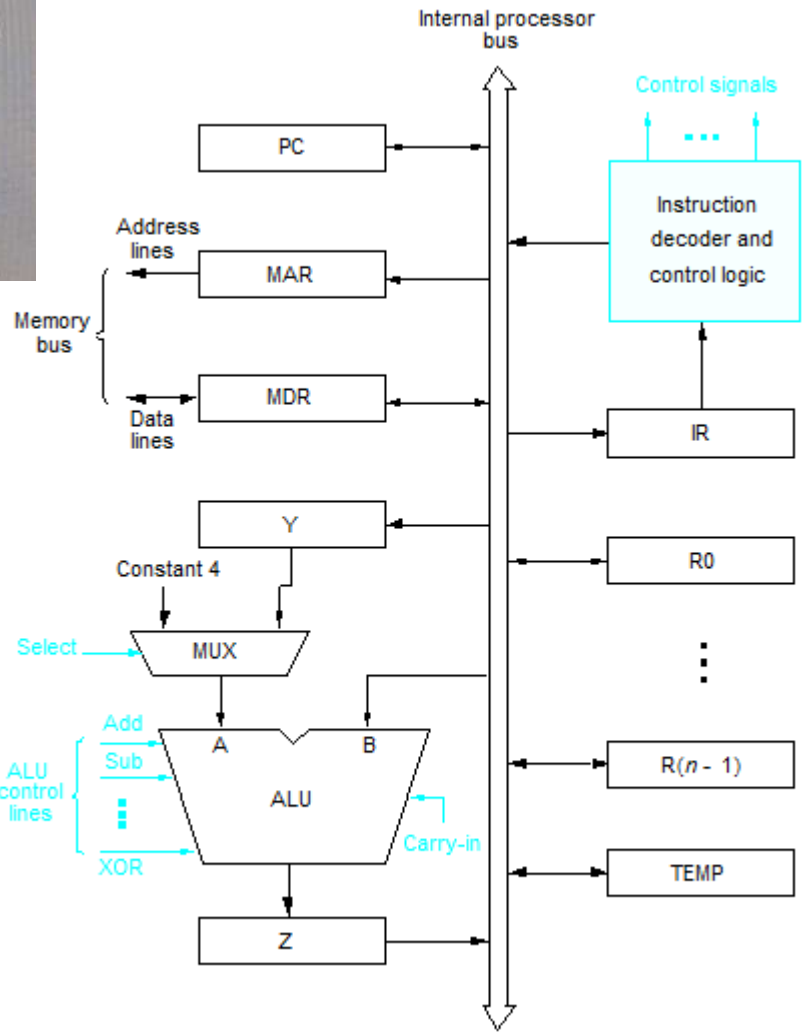
1. Fetch instruction
2. Fetch the operand
3. Perform operation
4. Store result

ADD (R3),R1
[R1] ← M([R3]) + [R1]

Add (R3), R1

1	PC_{out} , MAR_{in} , Read, Select4,Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMFC
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMFC
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

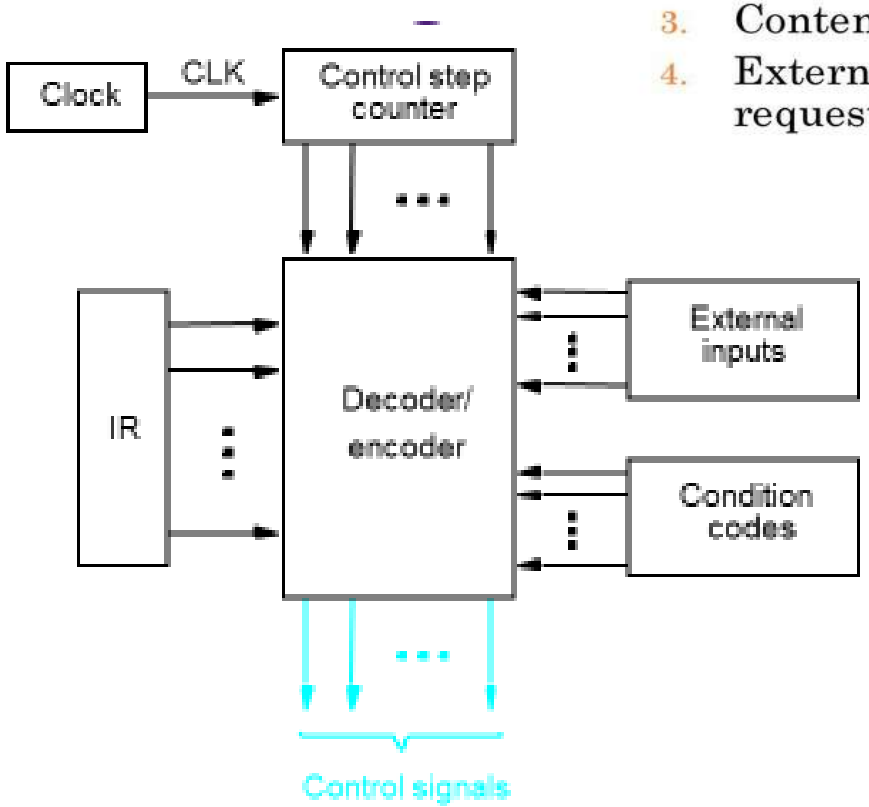
Control sequence for execution of the instruction Add (R3),R1.



Hardwired Control Unit

- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
- Two categories: Hardwired control and Microprogrammed control
- Hardwired system can operate at high speed; but with little flexibility.

- The sequence of control signals step is complete in one clock period.
- A counter may be used to keep track of the control steps, as shown in previous slide. Each state, or count, of this counter corresponds to one control step.
- The required control signals are determined by the following information:
  1. Contents of the control step counter
  2. Contents of the instruction register
  3. Contents of the condition code flags
  4. External input signals, such as MFC and interrupt requests



Control Unit Organization

- This diagram can be viewed as a state machine that changes from one machine to another in every clock cycle , depending on the contents of IR, condition codes, and the external inputs
- The sequence of operation carried out by this machine is determined by the wiring of the logic elements, hence the name “Hardwired”.

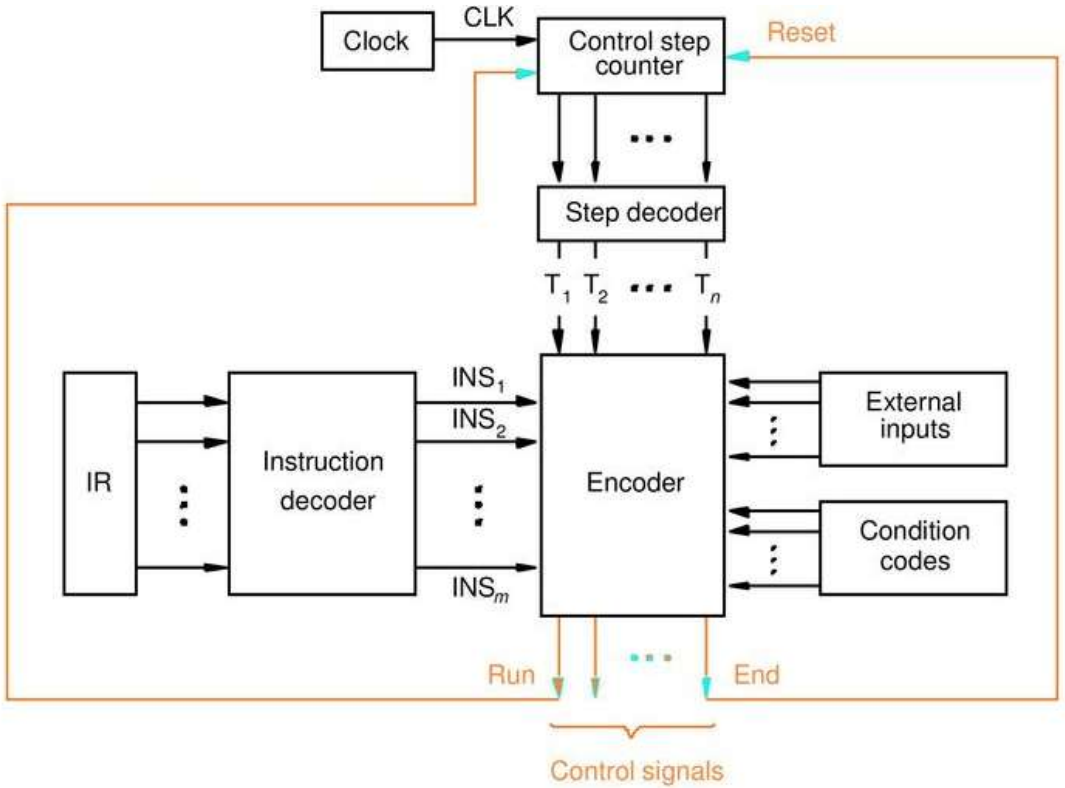


Figure 7.11. Separation of the decoding and encoding functions.

- The step decoder provides a separate signal line for each step, or time slot, in the control sequence.
- Similarly, the output of the instruction decoder consists of a separate line for each machine instruction. For any instruction loaded in the IR, one of the output lines INS1 through INSm is set to 1, and all other lines are set to 0.
- The input signals to the encoder block in diagram combined to generate the individual control signals Yin, PCout, Add, End, and so on.

Microprogrammed Control Unit

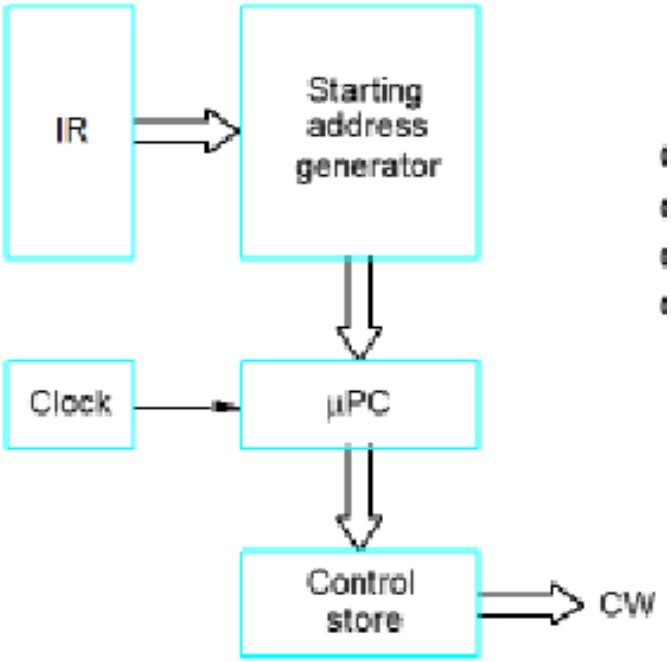
- Control signals are generated by a program similar to machine language programs.
- Control Word (CW); microroutine; microinstruction

Micro - instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

- A **control word (CW)** is a word whose individual bits represent the various control signals. Each of the control steps in the control sequence of an instruction defines a unique combination of 1s and 0s in the CW.
- A sequence of CWs corresponding to the control sequence of a machine instruction constitutes the **microroutine** for that instruction, and the individual control words in this micro routine are referred to as **microinstructions**.

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4,Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Control sequence for execution of the instruction Add (R3),R1.



- The micro routine for all instructions in the instruction set of a computer are stored in a special memory called the program Store.
- The control unit can generate the control signals for any instruction by sequentially reading the control words of the corresponding micro routines from the control store.
- This suggest organizing the control unit as shown in figure. To read the control words sequentially from the control store a micro program(uPC) counter is used.
- Everytime a new instruction is loaded into the IR, the output of the block labeled “Starting address generator” is loaded into the uPC , the uPC is then automatically incremented by the clock causing successive micro instructions to be read from the control store
- And hence the control signals are delivered to the various parts of the microprocessor in the correct sequence

Unit V :Memory system

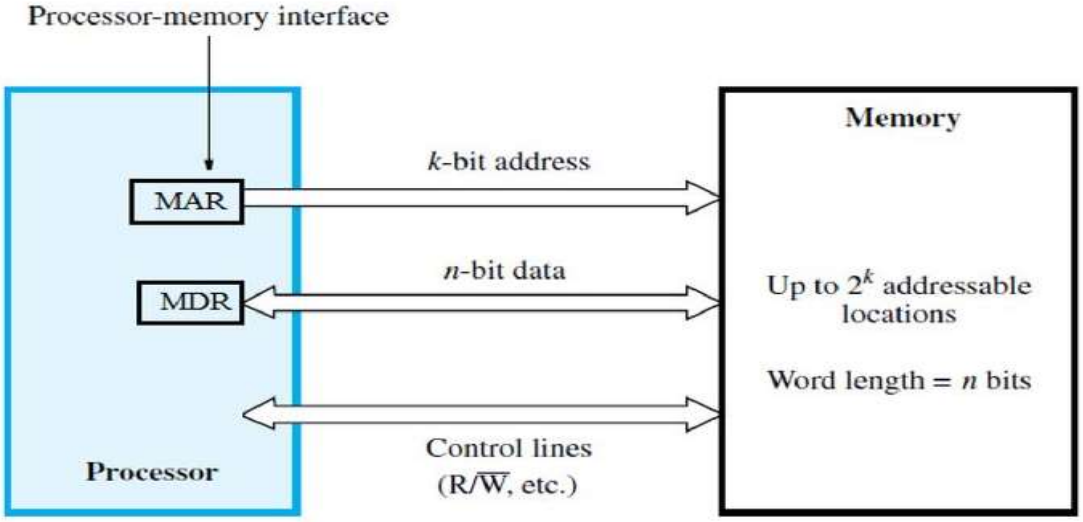
Basic Concepts, Computer memory system overview: Characteristics, memory hierarchy, Basic concepts of pipelining.
Semiconductor Memory, Internal organization of memory chips, static memories, speed, size and cost.
Cache Memories: Basic concepts of cache, elements of cache design, cache performance.
Secondary Storage :Magnetic disk and physical characteristics

1. Basic Concepts of Memory

- Execution speed of programs is highly dependent on the speed with which instructions and data can be transferred between the processor and the memory.
- Ideally, the memory would 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.
- Much work has gone into developing structures that improve the effective 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 computer that generates 16-bit addresses is capable of addressing up to

  $2^{16} = 64K$ (kilo) memory locations

The connection between the processor and its memory consists of address, data, and control lines,



Connection of the memory to the processor.

- The processor uses the address lines to specify the memory location involved in a data transfer operation, and uses the data lines to transfer the data.
- At the same time, the control lines carry the command indicating a Read or a Write operation and whether a byte or a word is to be transferred. The control lines also provide the necessary timing information and are used by the memory to indicate when it has completed the requested operation.
- When the processor-memory interface receives the memory's response, it asserts the MFC (Memory Function Complete) signal

- A useful measure of the speed of memory units is the time that elapses between the initiation of an operation to transfer a word of data and the completion of that operation.
- This is referred to as the ***memory access time***.
- Another important measure is the ***memory cycle time***, which is the minimum time delay required between the initiation of two successive memory operations, for example, the time between two successive Read operations.
- The cycle time is usually slightly longer than the access time.

- **Cache and Virtual Memory**
- The processor of a computer can usually process instructions and data faster than they can be fetched from the main memory. Hence, the memory access time is the bottleneck in the system.
- One way to reduce the memory access time is to use a ***cache memory.***
- This is a small, fast memory inserted between the larger, slower main memory and the processor. It holds the currently active portions of a program and their data.
- ***Virtual memory*** With this technique, only the active portions of a program are stored in the main memory, and the remainder is stored on the much larger secondary storage device. Sections of the program are transferred back and forth between the main memory and the secondary storage device in a manner that is transparent to the application program. As a result, the application program sees a memory that is much larger than the computer's physical main memory.

2.Characteristics of Memory Systems

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

Location:

The term location refers to whether memory is internal or external to the computer. Internal memory is often equated with main memory, but there are other forms of internal memory.

The processor requires its own local memory, in the form of registers.

External memory consists of peripheral storage devices, such as disk and tape, that are accessible to the processor via I/O controllers.

Capacity :

An obvious characteristic of memory is its 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

Unit of Transfer:

A related concept is the 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.

Access Method:

- **Sequential access:**

Memory is organized into units of data, called records. Access must be made in a specific linear sequence.

- **Direct access:**

As with sequential access, direct access involves a shared read– write mechanism. However, individual blocks or records have a unique Computer memory System address based on physical location.

- **Random access:**

Each addressable location in memory has a unique, physically wired- in addressing mechanism.. 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 that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously.

Performance:

- **Access time (latency):** For random- access memory, this is the time it takes to perform a read or write operation, that is, the time from the instant that an address is presented to the memory to the instant that data have been stored or made available for use.
- **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. Note that memory cycle time is concerned with the system bus, not the processor.
- **Transfer rate:** This is the rate at which data can be transferred into or out of a memory unit.

Hierarchy of Memory :

As one goes down the hierarchy, the following occur:

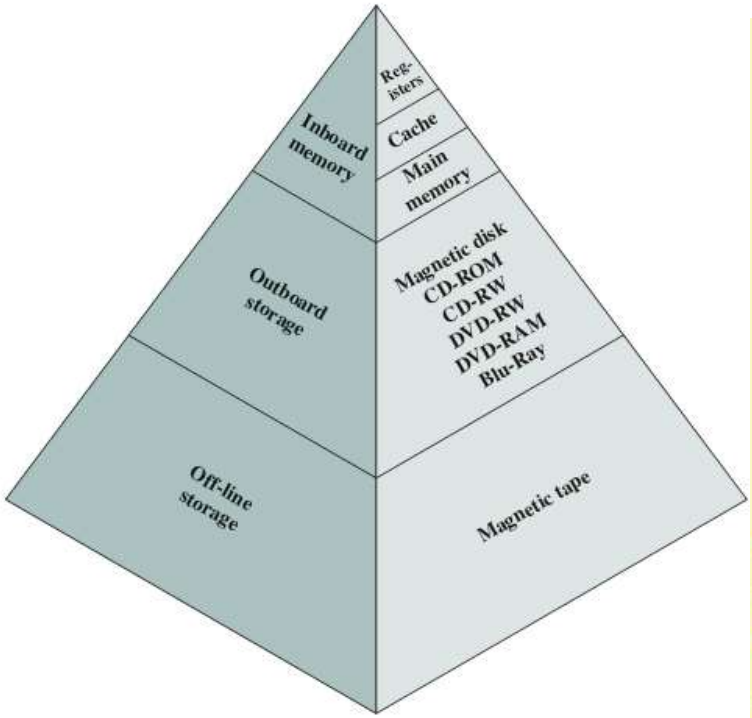
a. Decreasing cost per bit;

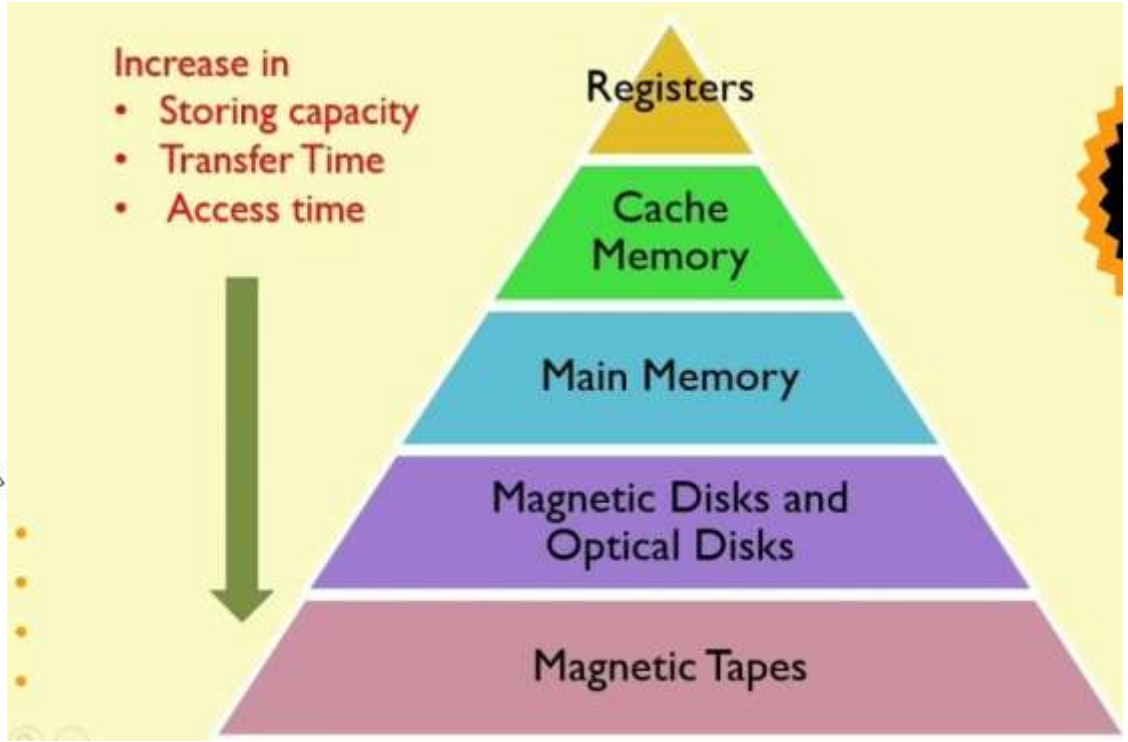
b. Increasing capacity;

c. Increasing access time;

d. Decreasing frequency of access of the memory by the processor.

Thus, smaller, more expensive, faster memories are supplemented by larger, cheaper, slower memories.





Basic concepts of Pipelining

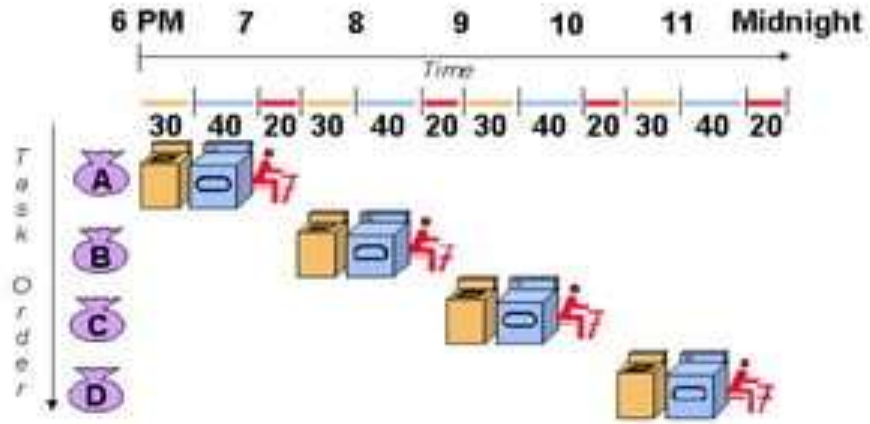
- Pipelining is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process often in parallel, to improve the speed and efficiency of a system.
- It is also known as pipeline processing. Pipelining is a technique where multiple instructions are overlapped during execution.

- A **Pipelining** is a series of stages, where some work is done at each stage in parallel.
- The stages are connected one to the next to form a pipe - instructions enter at one end, progress through the stages, and exit at the other end.

Pipelining Case: Laundry

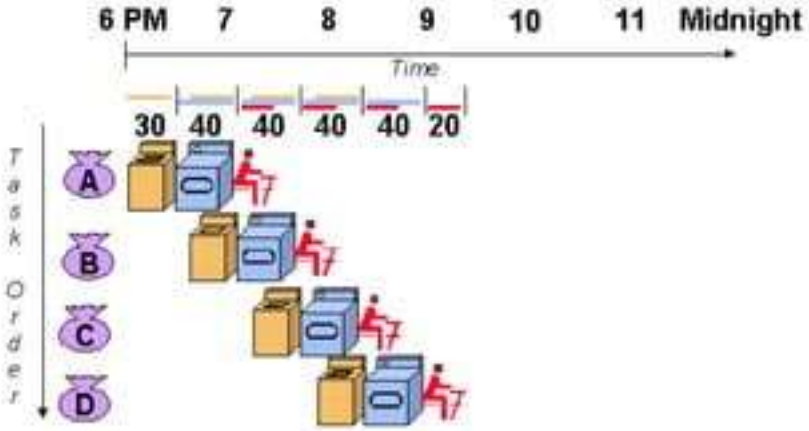
- 4 loads of laundry that need to washed, dried, and folded.
  - 30 minutes to wash, 40 min. to dry, and 20 min. to fold.
  - We have 1 washer, 1 dryer, and 1 folding station.
- What's the most efficient way to get the 4 loads of laundry done?

Non Pipelined Laundry



- Takes a total of 6 hours; nothing is done in parallel

Pipelined Laundry



- Using this method, the laundry would be done at 9:30.

Definition:

Pipelining is an speed up technique where multiple instructions are overlapped in execution on a processor.

Instruction Pipeline

- **Instruction execution process lends itself naturally to pipelining**
  - **overlap the subtasks of instruction fetch, decode and execute**

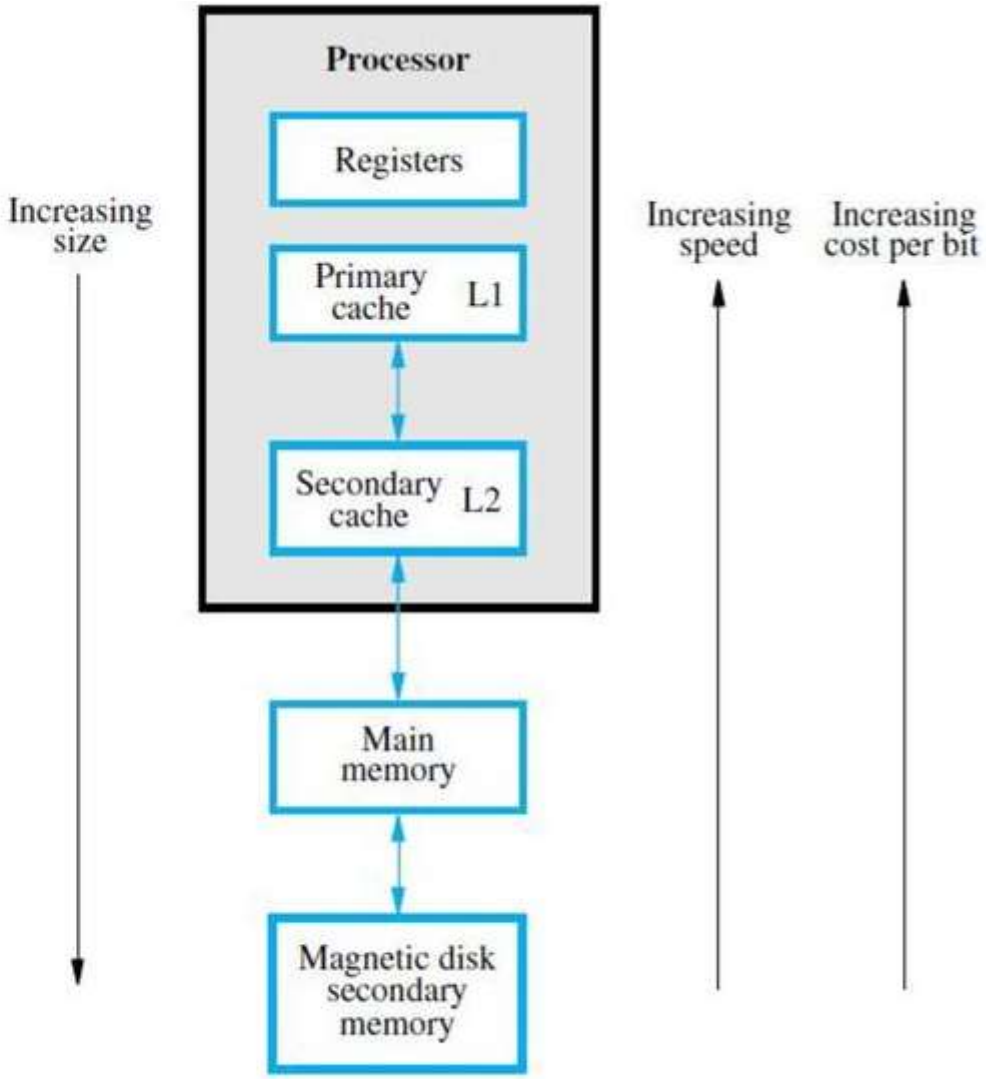
- **Instruction pipeline has six operations,**
  - Fetch instruction (FI)
  - Decode instruction (DI)
  - Calculate operands (CO)
  - Fetch operands (FO)
  - Execute instructions (EI)
  - Write result (WR)

  Overlap these operations

Pipelining: Processors

- Computers, like laundry, typically perform the exact same steps for every instruction:
  - Fetch an instruction from memory
  - Decode the instruction
  - Execute the instruction
  - Read memory to get input
  - Write the result back to memory

- **Memory hierarchy**
- The fastest access is to data held in processor registers.
- *Processor cache*, holds copies of the instructions and data stored in a much larger memory that is provided externally.
- A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers. The primary cache is referred to as the *level 1* (L1) cache.
- A larger, and hence somewhat slower, secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the *level 2* (L2) cache.



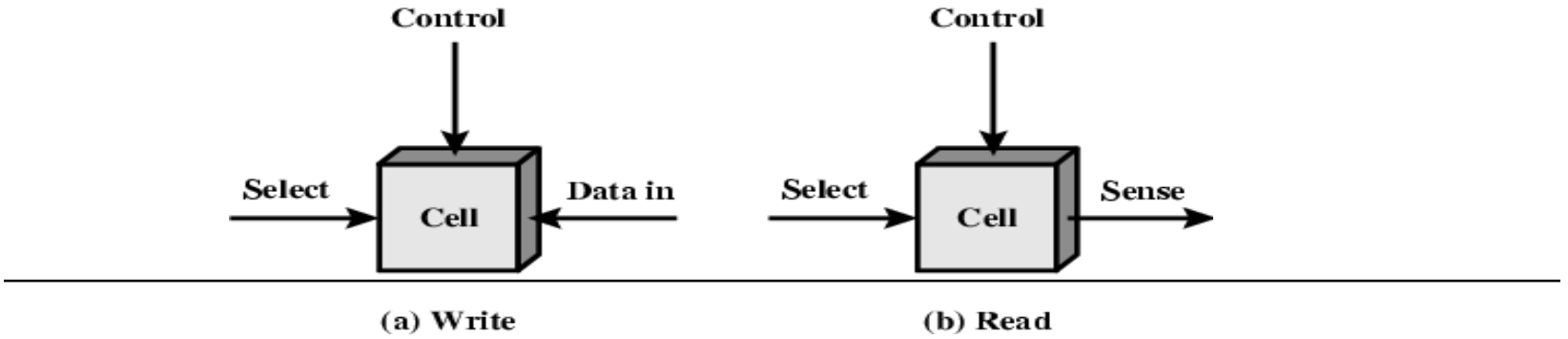
Memory hierarchy.

•Each memory cell can hold one bit of information.
•Memory cells are organized in the form of an array.
•One row is one memory word.
•All cells of a row are connected to a common line, known as the “word line”.
•Word line is connected to the address decoder.
•Sense/write circuits are connected to the data input/output lines of the memory chip.
•The storage part is modelled here with SR-latch, but in reality it is an electronics circuit made up of transistors.
•The memory constructed with the help of transistors is known as semiconductor memory.
•The semiconductor memories are termed as Random Access Memory(RAM), because it is possible to access any memory location in random.

Depending on the technology used to construct a RAM, there are two types of RAM :

SRAM: Static Random Access Memory.

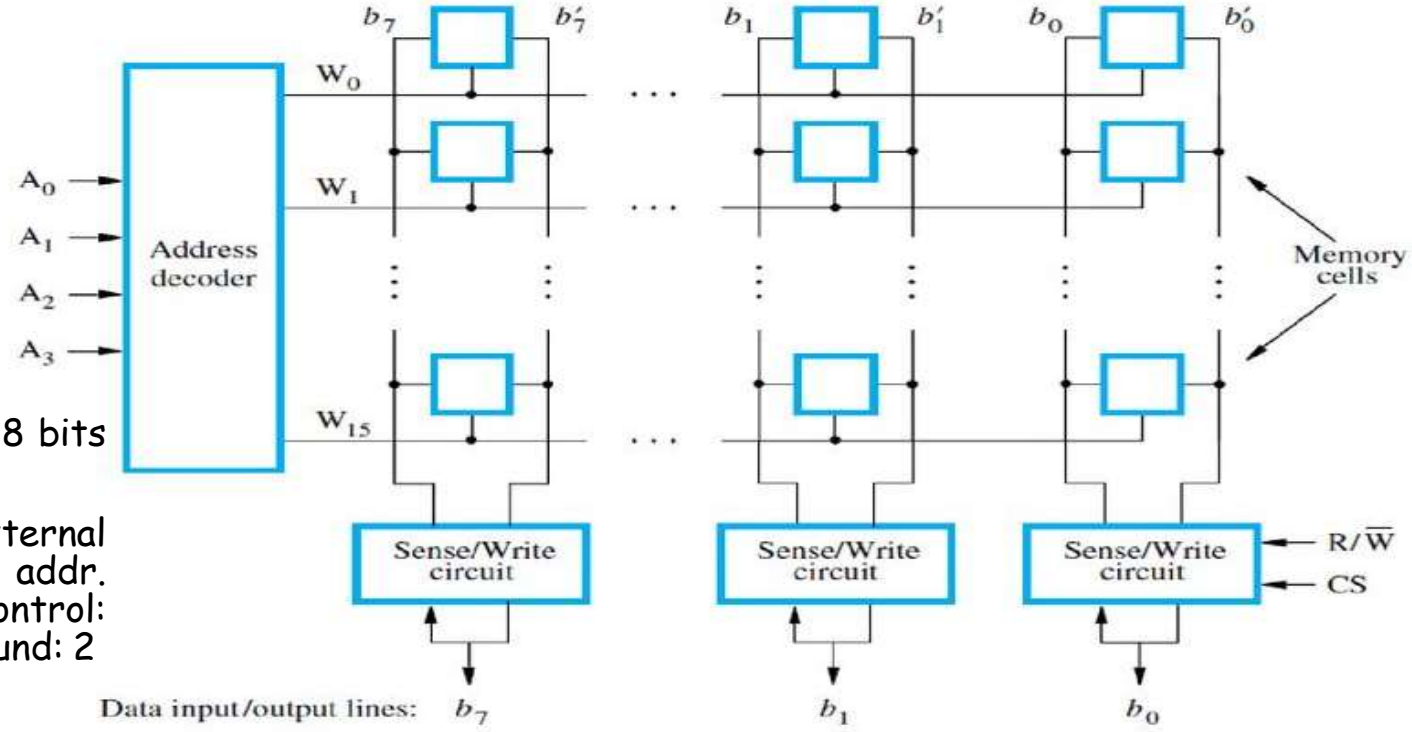
DRAM: Dynamic Random Access Memory.



Semiconductor RAM Memory

- Are available in wide range of speeds. Their cycle times range from 100ns to 10ns.
- Were much more expensive than magnetic core memories they replaced.
- Because of rapid advancements in VLSI technology, the cost of semiconductor memories has dropped dramatically.
- They are now used almost exclusively for implementing memories.

Internal organization of memory chip



16 words of 8 bits each: (16*8)

It has 16 external connections: addr. 4, data 8, control: 2,power/ground: 2

- Memory cells are usually organized in the form of an array, in which each cell is capable of **storing one bit** of information.
- Each row of cell constitutes a memory word, and all cells of a row are connected to a common line is word line driven by **address decoder** on the chip.
- The cells in each column are connected to **Sense/Write ckt** by two-bit lines. Sense/Write ckt are connected to the data I/O lines of the chip. During a Read operation, these ckt read the information stored in the cells selected by word line and transmit this information to the output data lines.
- During a Write operation, Sense/Write ckt receive input information and store it in the cells of the selected word.

- In above example of a very small memory circuit consisting of 16 words of 8 bits each.
- This is referred to as a 16 × 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 lines of a computer.
- 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.

Static Memory

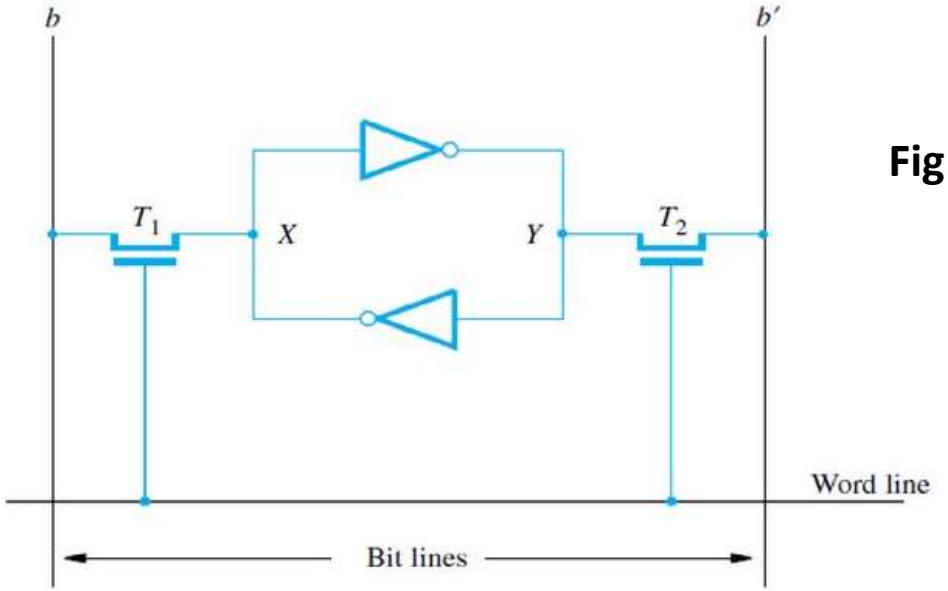


Fig: A static RAM Cell

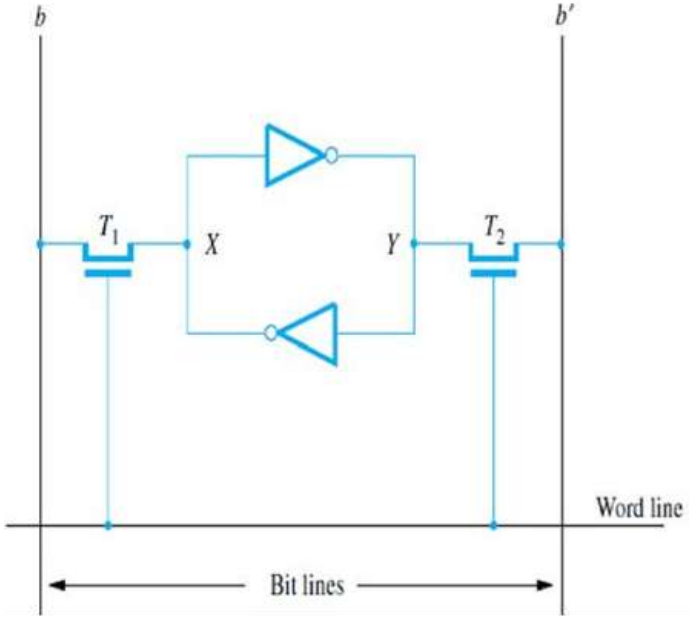
Memories that consists of circuits capable of retaining their state as long as power is applied are known as Static Memories.

- Two inverters are cross-connected to form a latch. The latch is connected to two bit lines by transistors *T*1 and *T*2. These transistors act as switches that can be opened 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, 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. Assume that this state represents the value 1.

- **Read Operation**
  - **Assume cell is in state 1.**
  - **To read keep Word line to 1.**
  - So signal on bit line *b* is high and the signal on bit line *b`* is low

- **Write Operation**
- During a Write operation, the Sense/Write circuit drives bit lines *b* and *b`*.
- It places the appropriate value on bit line *b* and its complement on *b`* and activates the word line.
- This forces the cell into the corresponding state, which the cell retains when the word line is deactivated.

- Sense/Write circuits at the end of the bit lines monitor the state of b and b’ and set the output accordingly.



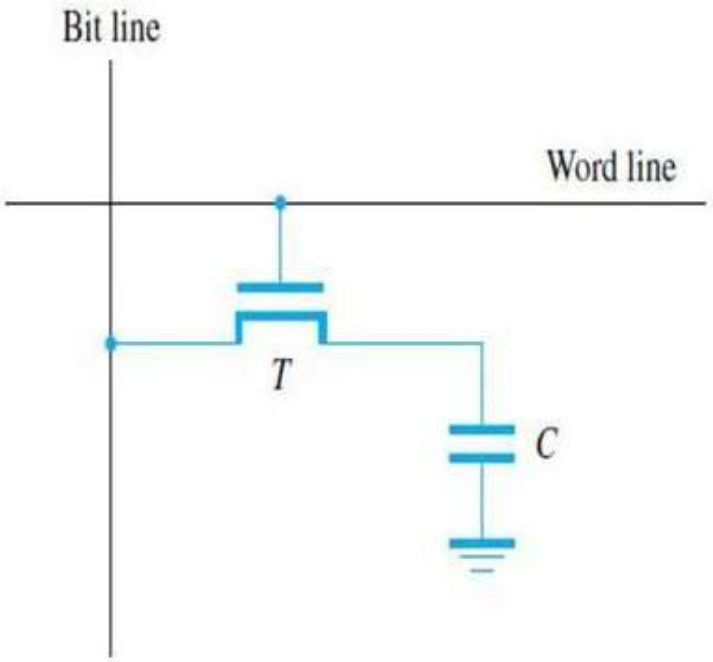
- Continuous power is needed for the cell to retain its state. If power is interrupted, the cell's contents are lost. When power is restored, the latch settles into a stable state, but not necessarily the same state the cell was in before the interruption. Hence, SRAMs are said to be *volatile* memories.
- 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.
- Static RAMs can be accessed very quickly. Access times on the order of a few nanoseconds are found in commercially available chips. SRAMs are used in applications where speed is of critical concern.

- Static RAMs are fast, but their cells require several transistors.
- Less expensive and higher density RAMs can be implemented with simpler cells.
- But, these simpler cells do not retain their state for a long period, unless they are accessed frequently (Refreshed) for Read or Write operations.
- Memories that use such cells are called *dynamic RAMs* (DRAMs).

- Information is stored in a dynamic memory cell in the form of a charge on a capacitor, but 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 (read or write)* by restoring the capacitor charge to its full value.

Asynchronous DRAM

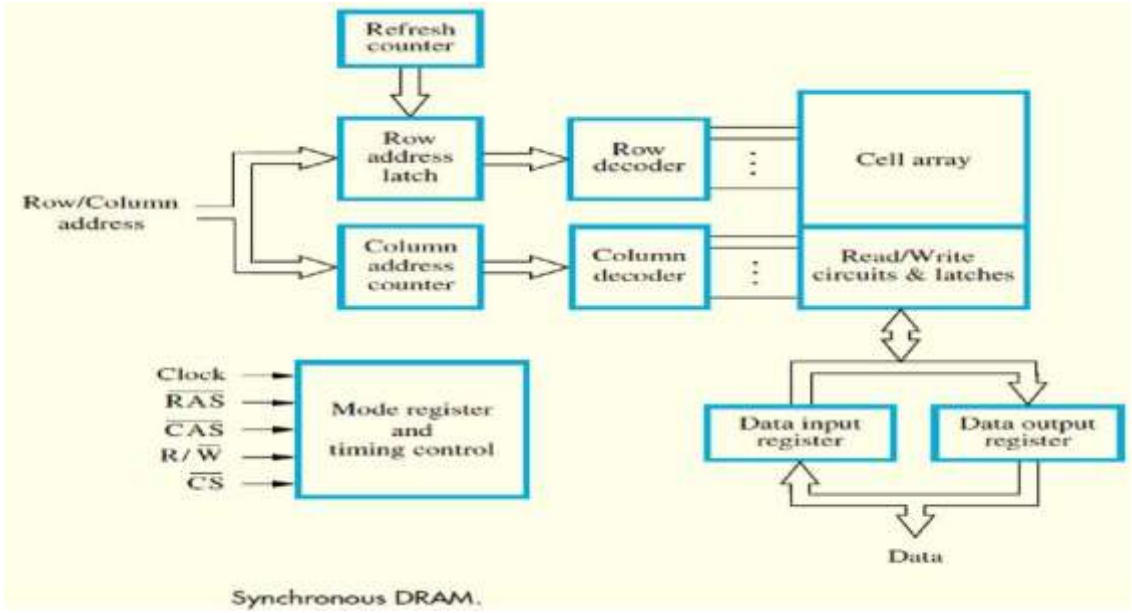
- A single-transistor dynamic memory cell.
- To store information in this cell, transistor *T* is turned on and an appropriate voltage is applied to the bit line.
- This causes a known amount of charge to be stored in the capacitor.
- After the transistor is turned off, the charge remains stored in the capacitor, but not for long. The capacitor begins to discharge.



A single-transistor dynamic memory cell.

- This is because the transistor continues to conduct a tiny amount of current, after it is turned off.
- Hence, the information stored in the cell can be retrieved correctly only if it is read before the charge in the capacitor drops below some threshold value.
- During a Read operation, the transistor in a selected cell is turned on.
- A sense amplifier connected to the bit line detects whether the charge stored in the capacitor is above or below the threshold value.
- If the charge is above the threshold, the sense amplifier drives the bit line to the full voltage representing the logic value 1. As a result, the capacitor is recharged.
- If the sense amplifier detects that the charge in the capacitor is below the threshold value, it pulls the bit line to ground level to discharge the capacitor fully.
- Thus, reading the contents of a cell automatically refreshes its contents.
- Since the word line is common to all cells in a row, all cells in a selected row are read and refreshed at the same time.

Synchronous DRAMs



Synchronous DRAM.

- a block of data can be transferred at a much faster rate than can be achieved for transfers involving random addresses.
- **DRAMs whose operation is synchronized with a clock signal.** Such memories are known as *synchronous DRAMs* (SDRAMs).
- The cell array is the same as in asynchronous DRAMs.
- **SDRAMs have built-in refresh circuitry, with a refresh counter to provide the addresses of the rows to be selected for refreshing.**
- As a result, the dynamic nature of these memory chips is almost invisible to the user.
- SDRAMs have several different modes of operation, which can be selected by writing **control information into a *mode* register.**
- For example, in **burst operations** of different lengths can be specified.
- It is not necessary to provide externally-generated pulses on the CAS line to select successive columns.
- The necessary control signals are generated internally using a **column counter and the clock signal.**
- New data are placed on the data lines **at the rising edge of each clock pulse.**
- 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 for simplicity) 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.

- Synchronous DRAM scan deliver data at a very high rate, because all the control signals needed are generated inside the chip.
- The initial commercial SDRAMs designed for clock speeds of up to 133 MHz.
- As technology evolved, much faster SDRAM chips were developed. Today's SDRAMs operate with clock speeds that can exceed 1 GHz.

What is cache memory?

- The cache memory is a small-sized high speed volatile memory that provides high speed data access to a processor.
- The cache memory stores the frequently used computer programs, applications and the program data.

Locality of Reference

- Analysis of programs indicates that many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently.
  - These instructions may be the ones in a loop, nested loop or few procedures calling each other repeatedly.
  - This is called "locality of reference".
- Temporal locality of reference:
  - Recently executed instruction is likely to be executed again very soon.

Cache memory is designed to combine the memory access time of expensive, high- speed memory combined with the large memory size of less expensive, lower- speed memory.

- There is a relatively large and slow main memory together with a smaller, faster cache memory. The cache contains a copy of portions of main memory.
- When the processor attempts to read a word of memory, **a check is made to determine if the word is in the cache.**
- 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.

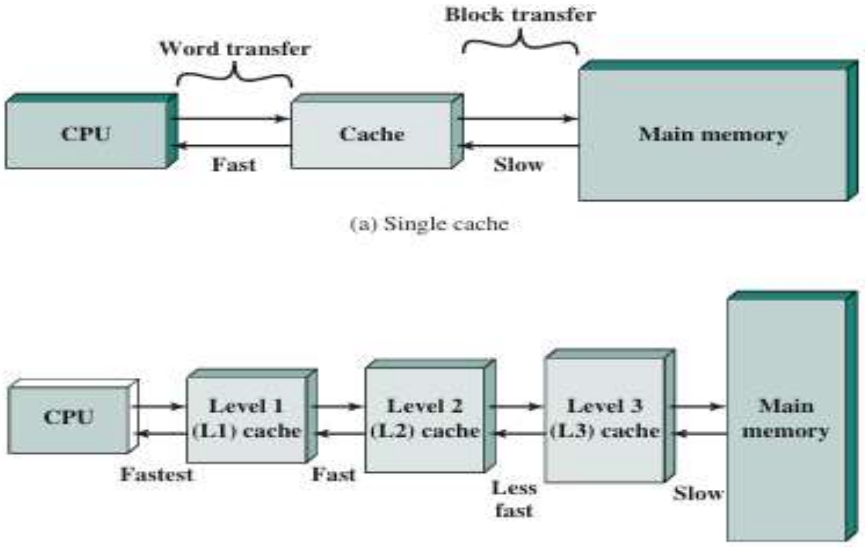


Figure depicts the use of multiple levels of cache. The L2 cache is slower and typically larger than the L1 cache, and the L3 cache is slower and typically larger than the L2 cache.

- **Cache Hits**
- The processor does not need to know explicitly about the existence of the cache. It simply issues R/W 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 R/W operation is performed on the appropriate cache location. In this case, a *read* or *write hit* is said to have occurred. The main memory is not involved when there is a **cache hit** in a Read/Write operation.

- **Cache Misses**
- A Read operation for a word that is not in the cache constitutes a *Read miss*.
- It causes the block of words containing the requested word to be 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.**

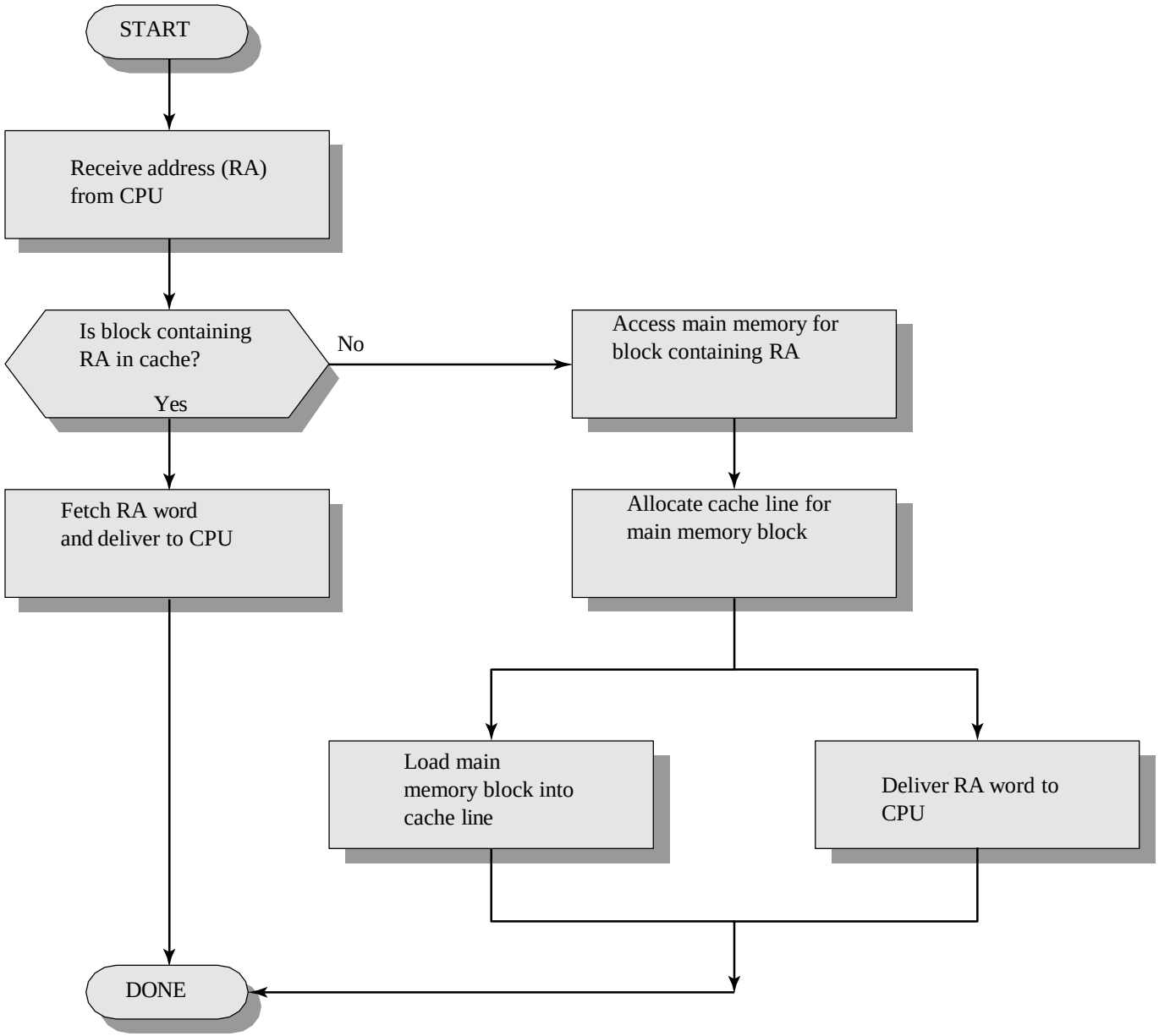


Fig.: Cache Read Operation

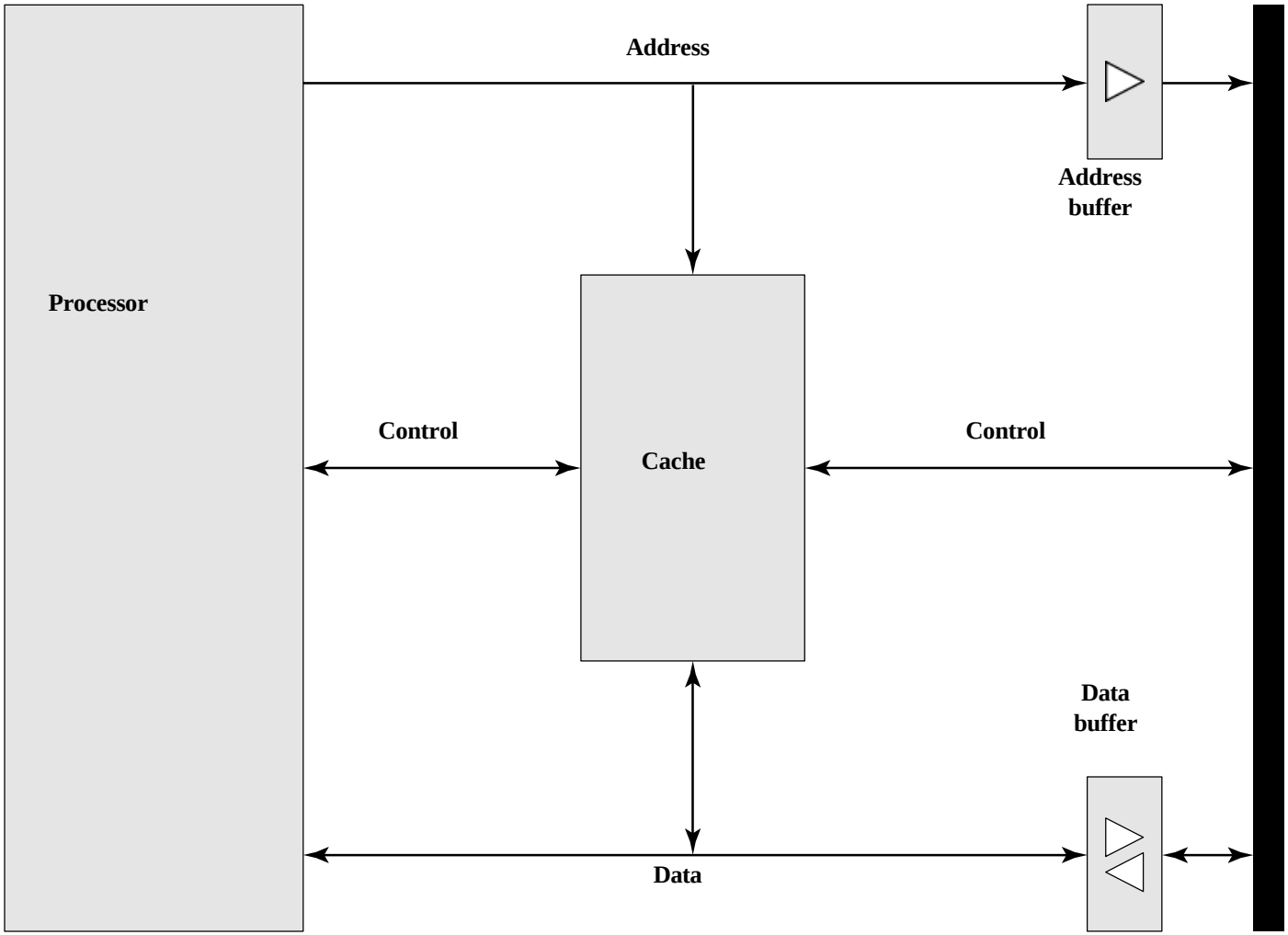


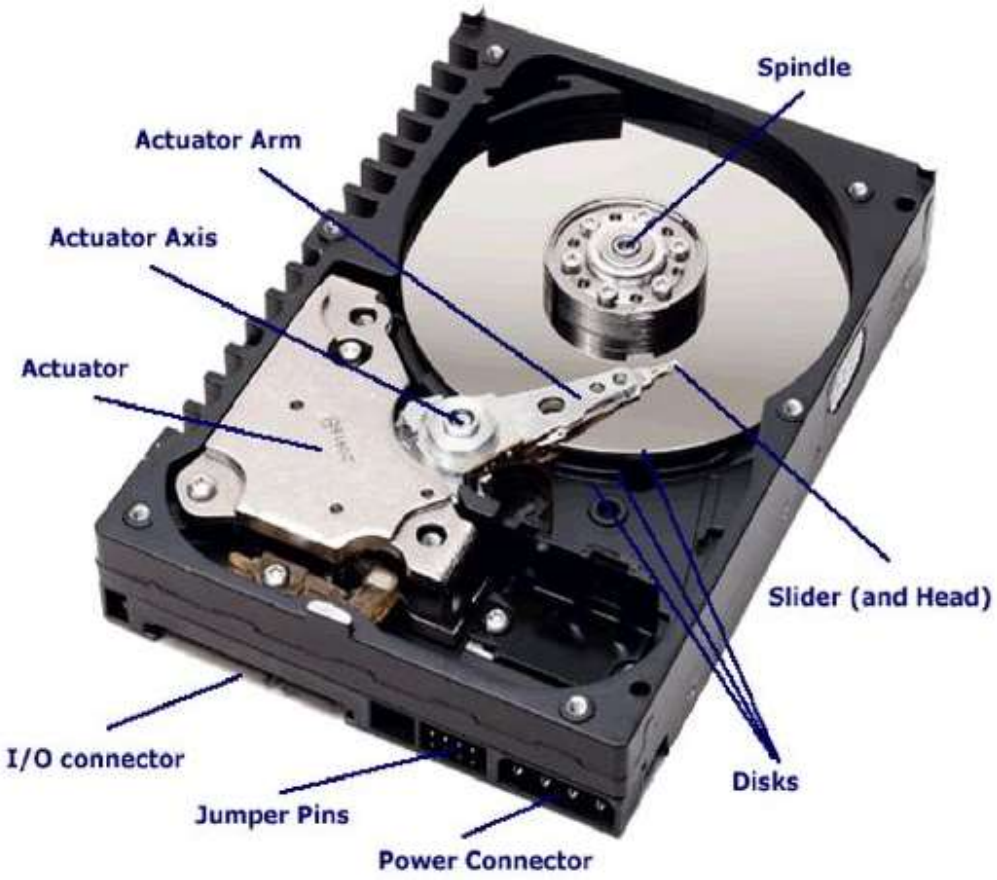
Fig: Typical Cache Organization

Elements of Cache Design

- **Cache Addresses**
  - Logical
  - Physical
- **Cache Size**
- **Mapping Function**
  - Direct
  - Associative
  - Set Associative
- **Replacement Algorithm**
  - Least recently used (LRU)
  - First in first out (FIFO)
  - Least frequently used (LFU)
  - Random
- **Write Policy**
  - Write through
  - Write back Write once
- **Line Size**
- **Number of caches**
  - Single or two level
  - Unified or split

Secondary Storage :Magnetic Hard Disc

- 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 *disk controller*, which is the electronic circuitry that controls the operation of the system.



- Magnetic-disk system consists of one or more disk platters mounted on a common spindle.
- A thin magnetic film is deposited on each platter, usually on both sides.
- The assembly is placed in a drive that causes it to rotate at a constant speed.
- The magnetized surfaces move in close proximity to read/write heads.
- Data are stored on concentric tracks, and the read/write heads move radially to access different
- Platter, divided into billions of tiny areas. Each one of those areas can be independently magnetized (to store a 1) or demagnetized (to store a 0).

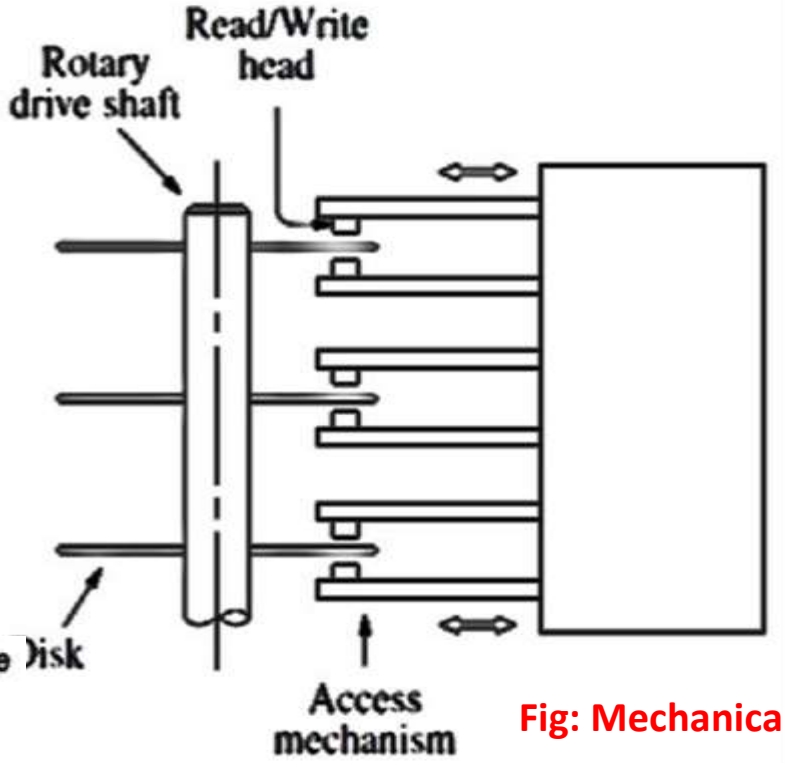
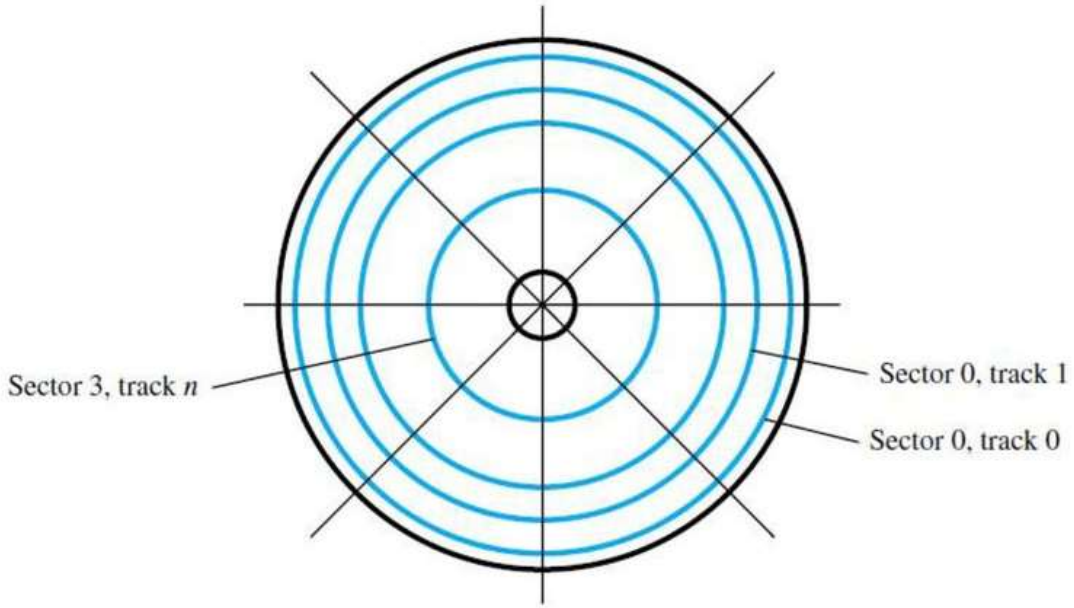
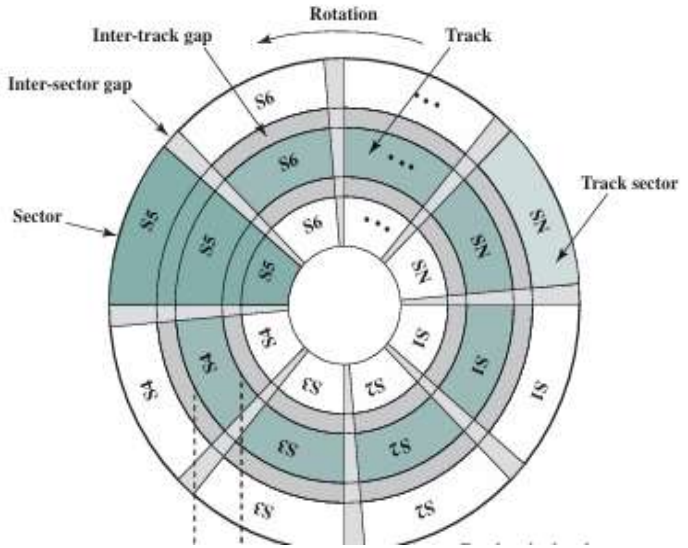


Fig: Mechanical Structure

Organization and Accessing of Data on a Disk





Organization of one surface of a disk.

- 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*.
- All tracks of a cylinder can be accessed without moving the read/write heads.
- Data are accessed by specifying the surface number, the track number, and the sector number.
- Read and Write operations always start at sector boundaries.
- Data bits are stored serially on each track. Each sector may contain 512 or more bytes.
- 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 the data bytes.
- There is a small *inter-sector gap* that enables the disk control circuitry to distinguish easily between two consecutive sectors.

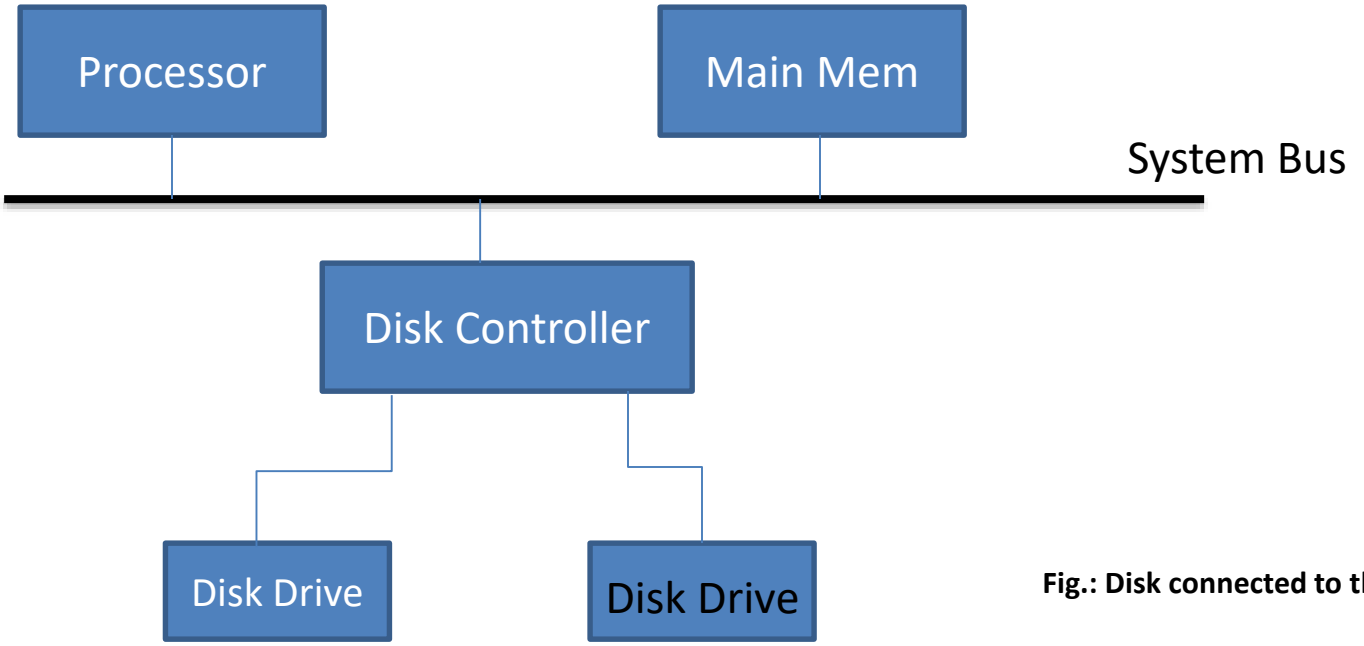


Fig.: Disk connected to the system bus

- **Disk Controller**
- One disk controller may be used to control more than one drive. A disk controller that communicates directly with the processor contains a number of registers that can be read and written by the operating system.
- Thus, communication between the OS and the disk controller is achieved in the same manner as with any I/O interface.
- The disk controller uses the DMA scheme to transfer data between the disk and the main memory.
- The OS initiates the transfers by issuing Read and Write requests, which entail loading the controller's registers with the necessary addressing and control information.

Physical Characteristics of Disc

Head Motion	**Platters**
Fixed head (one per track)	Single platter
Movable head (one per surface)	Multiple platter
Disk Portability	**Head Mechanism**
Nonremovable disk	Contact (floppy)
Removable disk	Fixed gap
Sides	Aerodynamic gap (Winchester)
Single sided	
Double sided	

Unit VI : Computer Peripherals **10M**

- Working of input and output devices like keyboard, mouse, video display, flat panel display and printers.
- Serial communication links: Asynchronous & Synchronous transmission with ADSL connection.
- Standard communication interfaces.

Input Devices : Keyboard

- The most commonly encountered input device is keyboard usually completed by a mouse or trackball. Together with video display as an output device, they are used for direct human interaction with the computer.
- Keyboards are available in two types: One type consists of an array of mechanical switches mounted on a printed circuit board. The switches are ground in rows and columns and connected to a microcontroller on the board.
- When a switch is pressed, the controller identify the row and column and determines which key is being pressed.
- After correcting for switch bounce, the controller generates a code representing that switch and sends it over serial link to the computer.
- The second time uses of flat structure consisting of three layer. Top layer is a plasticized material, with key positions marked on the top surface and conducting traces deposited on the underside.
- The middle layer is made of rubber, with holes at the key positions. The bottom layer is metallic, with raised bumps at key positions.
- When pressure is applied to the top layer at a key position, the trace underneath comes in contact with the corresponding bump on the bottom layer, thus completing and electrical circuit in the same way as a mechanical switch. The current that flows in this circuit is sensed by the microcontroller. This arrangement provides a low-cost. such keyboards are commonly encountered in applications such as point of sale terminals

Mouse

- The invention of the mouse in 1968 represented an important step in the development of the means of for people to communicate with computers.
- The mouse is made it possible to enter graphic information directly, by drawing the desired objects, and opened the door to many new and powerful ideas including Windows and pull down menus.
- The mouse is a device shift to fit comfortably in the operator's hand, such that it can be moved over flat surface.
- An electronic circuit senses this moment and sends some measures of the distance travelled in the X and Y directions to the computer. Movement is monitored either mechanically or optically.
- A mechanical mouse is fitted with a ball mounted, such that it can rotate freely as the mouse is moved. The rotation of the ball is sensed and used to advance two counters, one for each of the two axes of motion. The mouse is also fitted with two or three pushbuttons.
- The information from the counters and the buttons is collected by microcontroller, encoded as a 3-byte packet and send to the computer over a serial link.
- An optical mouse uses a light emitting diode(LED) to eliminate the surface on switch the mouse is placed, and light-sensitive device senses the light reflected from the surface.
- In some models, the mouse must be placed on a special pad that has a pattern of vertical & horizontal lines. The reflected light changes as the mouse moves from light to dark areas on the surface underneath, and the mouse measures the distance travelled by counting these changes,

Output Devices: Video Display

- Video displays are used whenever visual representation of computer output is needed. The most common display device uses a Cathode Ray Tube (CRT).
- Let us describing how a picture is formed on CRT. Focus beam of electrons strikes a fluorescent screen, causing emission of light that is seen as a bright spot against a dark background.
- The dot thus form disappears when the beam is turned off or moved to another spot.
- Thus in general, 3 independent variable variables need to be specified at all times, representing the position and intensity of the beam.
- The position of the beam corresponds to the X and Y coordinates of the spot on the screen.
- It's intensity, which is usually refer to as the Z-axis control, provides the gray-scale or brightness information at that spot. The smallest addressable spot on the screen is called a pixel. It consist of a number of smaller dots of different sizes, arranges in some geometrical pattern.
- The most common standard for computer video display is VGA (video graphics Array) and its higher quality variants.
- The basic VGA displays has 640*480 pixels. Variations of this standard specifies with higher resolution, such as 1024*768 (XVGA) and 1600*1200 (UXGA).
- Alphanumeric text and graphical pictures can be constructed using a technique called Raster Scan. The electron beam is sweft successfully across each row pixels from left to right, until all rows have been scanned from top to the bottom of the screen. Many video display use interlacing to increase the pursued rate at which the screen is refreshed.

Flat-Panel Display

- Flat panel displays are becoming increasing the popular, they are thinner and lighter in weight.
- They provide better linearity and in some cases even higher resolution. Several types of flat panel displays have been developed including liquid Crystal panels, Plasma panels and Electroluminescent Panels.
- The availability of low cost panel displays has been instrumental in the development of the Notebook computers.
- Liquid-crystal panels are constructed by sandwiching a thin layer of liquid crystal- a liquid that exhibits crystalline properties between two transparent plates. The top plate has transparent electrodes deposited on it, and the back plate is Mirror.
- By applying appropriate electrical signals across the plates, various segments of the liquid crystal can be activated, causing changes in their light diffusing or polarizing properties. Thus these segments either transmit or block the light.
- LCD come in 2 varieties, static displays have a simple structure in which electrodes are deposited along one axis of the lower plate, thus defining column and rows.
- Displays that uses this arrangement are simple to build and in-expensive, but the quality of the images they produce low. The illuminated area is not well defined, so edges in the image are not sharp.
- A higher quality display is produces by introducing a transistor at each intersection point. This provides faster response and better control over the area to be illuminated. The transistors are prepared on a thin film deposited on one of the plates. Hence this type display is called Thin-Film-Transistor (TFT). It is also known as active Matrix display this type of display most commonly found in high quality notebook computers.
- Plaza panels consists of two glass plates separated by a thin gap filled with a gas such as neon. Each plate has several parallel electrodes running across it. The electrodes on the two plates run at right angles to each other. Plasma displays can provide high resolution but are rather expensive.

Printers

- Printers are used to produce hard copy of output data text. They are usually classified as being either an impact on non impact type, depending on the nature of the printing mechanism used. Impact printers use mechanical printing mechanisms, and non-impact printer rely on optical, ink-jet or electrostatic techniques.
- Non-impact printers have few moving parts and can be operated at high speed. Laser printers use the same Technology as photocopiers. A drum coated with positively charged photo conductive material is scanned by laser beam.
- The positive charges that are illuminated by the beam are dissipated. Then a negatively charged toner powder is spread over the drum.
- And thus creating a page image that is then transferred to the paper. The drum is cleaned of any excess toner material to prepare it for printing the next page.
- other type of non-impact printers used Ink-jet, in which droplets of different color inks are fired at the paper from tiny nozzles to generate color output. They are also the most expensive
- A variety of techniques are used to fire the ink droplets. for example in a bubble ink-jet printer, the nozzle is attached to small chamber to which a heat pulse is applied. This causes the ink in a chamber to evaporate, forming a gas bubble that pushes a small amount of ink out of the nozzle.
- Most printers form characters and graphic images in the same way that images for formed on video screen that is , by printing dots in matrices. However because of the sensitivity of the human eyes to regular patterns, a regular Dot Matrix easily detected and interferes with the perceived quality of the image.
- Pixel consisting of several dots, ach having one of three colors, it is the geometrical arrangements of the dots in a pixel & the assignment of colors to each dots are varied.
- The highest quality printing is needed in applications such as graphic arts and photographic printing.

2. Serial Communication Links

- Devices such as the keyboard and mouse are connected directly to the computer with which they are used typically through a serial communication link.
- Other devices such as printers and scanners may be connected to a computer either directly or via communication network so that they may be share among several users as the important role in many computers applications.
- A devices called MODEM(Modulator-Demodulator)is installed at each end of a communication link to perform the desired signal transformations as shown in fig. as computer connected to a network server. This could be a permanent connection or a dialed connection over a telephone line

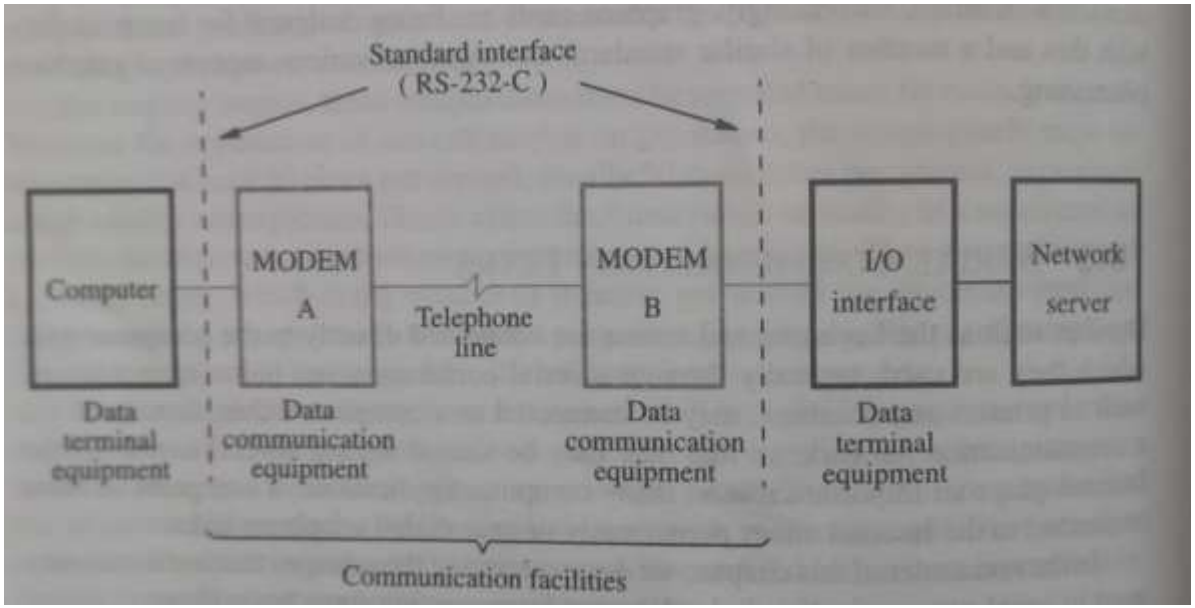


Fig: Remote connection to a network

- Serial communication means the data are sent one bit at a time. This requires that both the transmitting and the receiving devices used the same timing information for interpretation of individual bits.
- When the communicating devices are physically close to each other and multiple signal paths are available, a clock signal can be transmitted along with the data.
- However this is not feasible over longer links where only one signal path is available. Even if a second path is provided, the delays encountered by the data and clock signal could be different. Hence timing information and data are encoded on one transmission channel.

- Two types of Serial Transmission:

Asynchronous Transmission

- For data transmission at speeds not exceeding a few tens of KB per sec.
- Sender and receiver use independent clock signals having the same nominal frequency. No attempt is made to guarantee that two clocks have exactly the same frequency. Hence this scheme is called Asynchronous transmission

Synchronous transmission is needed to transmit data at higher speed. In this case the receiver recover the clock timing used by the transmitter by continuously observing the positions of the transitions in the received signal. As a result the receivers clock is synchronized with the transmitters clock and can be used to recover the transmitted data correctly.

Communication link may be operated with 3 schemes:

- **Simplex: Allows transmission in one direction**
- **Half duplex: Allows transmission in either direction, but not at the same time.**
- **Full duplex: Allows simultaneous transmission in both direction**

Asynchronous Transmission

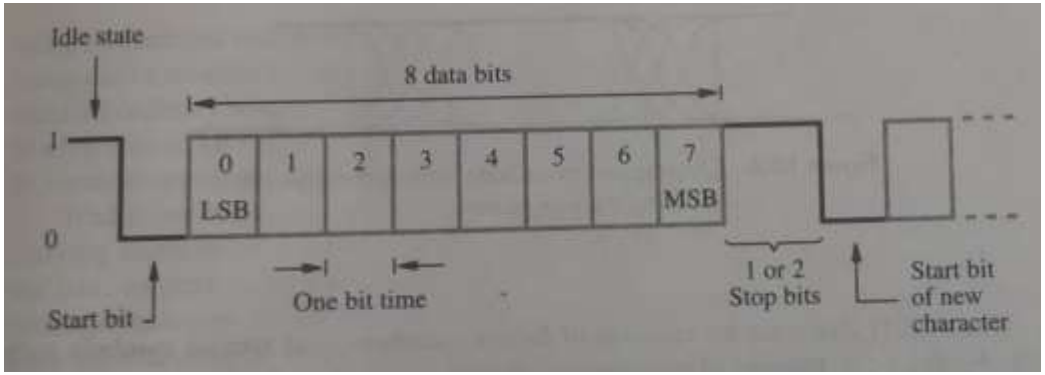


Fig: Asynchronous serial character Transmission

- The simplest scheme for serial communication is synchronous transmission using a technique called start stop.
- To facilitate timing recovery data organized in small groups of 6 to 8 bits, with a well defined beginning and end. In a typical arrangement alphanumeric characters encoded in 8 bits are transmitted as shown in fig.
- The line connecting the transmitter and the receiver is in the 1 state when Idle.
- Transmission of a characteristic by a zero bit referred to as the start Bit followed by data bits and one or two star bits stop bits have logic value of one the start bits alerts the receiver the data transmission is preceded by 0 bit, referred to as Start bit, followed by 8 data bits and 1 or 2 stop bits.
- The stop bits have logic value1. the Start bits alerts the receiver that the data transmission is about to begin.
- Its leading edge is used to synchronize the receiver clock with that of transmitter. The Stop bits at the end delineate consecutive characters in the 1 state after the end of the Stop bits.
- It is there responsibility of the sender and receiver circuitry to insert and remove the Start and Stop bits.

Synchronous Transmission

- In synchronous transmission data are transmitted in blocks consisting of several hundreds or thousands of Bits each.
- The beginning and end of each block are marked by appropriate codes and data within a block or organized according to an agreed upon set of rules. Modems requires a significant startup time to complete such operations as transmitting and detecting career frequencies and establishing synchronization. In some modems, the startup time is also used to adapt the modem circuits to the transmission properties of the link.
- In the start-stop scheme, the position of the 1 to 0 transition at the beginning of the start Bit given in fig. is the key to obtaining correct timing.
- Hence this scheme is useful only where the speed of transmission is sufficiently low and the conditions on the transmission link are such that the square waveform shown in fig. maintain their shape.

a. Network connection- ADSL

- Telephone Companies refer to the connection between central office and a subscriber as the subscriber loop hence when digital transmission is used the scheme is called **Digital subscriber Loop**. Hence, when a digital transmission is used, the scheme is called **Digital subscriber Line (DSL).**
- Computer communication is critically dependent on compatibility among equipments and services provided by several parties, including computer companies, modem manufacturers, and network service providers . Hence agreement on a few standards that are accepted by all parties is essential.
- In the DSL domain, a few such standard are developed, These include SDSL (Symmetric DSL), HDSL (high speed DSL) and **ADSL (Asymmetric DSL).** ADSL is most widely used for connecting home PC's to the internet.
- The symmetry in ADSL refers to the difference between transmission speed in the upstream and downstream directions.
- Most of the time the information sent from a computer to server on the Internet (Upstream direction) consists of input from the user. A low speed connection is sufficient in this case.
- On the other hand, information flowing to the user, such as image to be displayed on the computer screen, requires transmission at high speed to provide good response. For this reason the speed of Transmission in the downstream direction in ADSL is considerable higher than that used for Upstream transmission.
- The ADSL scheme uses different frequency bands and a technique called **Time Division Multiplexing** to create several channels of communication, one these allocated to regular telephone service. The others are allocated to the transmission of the data in the upstream and downstream directions. Typical arrangement is showing in figure

- A Single twisted pair wires carries information between the central office and a subscriber. At each end, a splitter separates data traffic from voice signals.
- At the subscriber end, data are directed to the computer over an appropriate data link, such as Ethernet or USB connection. Voice signals are sent to the telephone.
- At the central office, data are sent to the router device connected to the internet and voice signals are directed to the telephone switch.
- This arrangement makes it possible to have the computer connected to the Internet all the time, without the need for dialing. At the same time, regular dialed telephone service continues to be available.

(Router is a switching device to direct traffics in a data network)

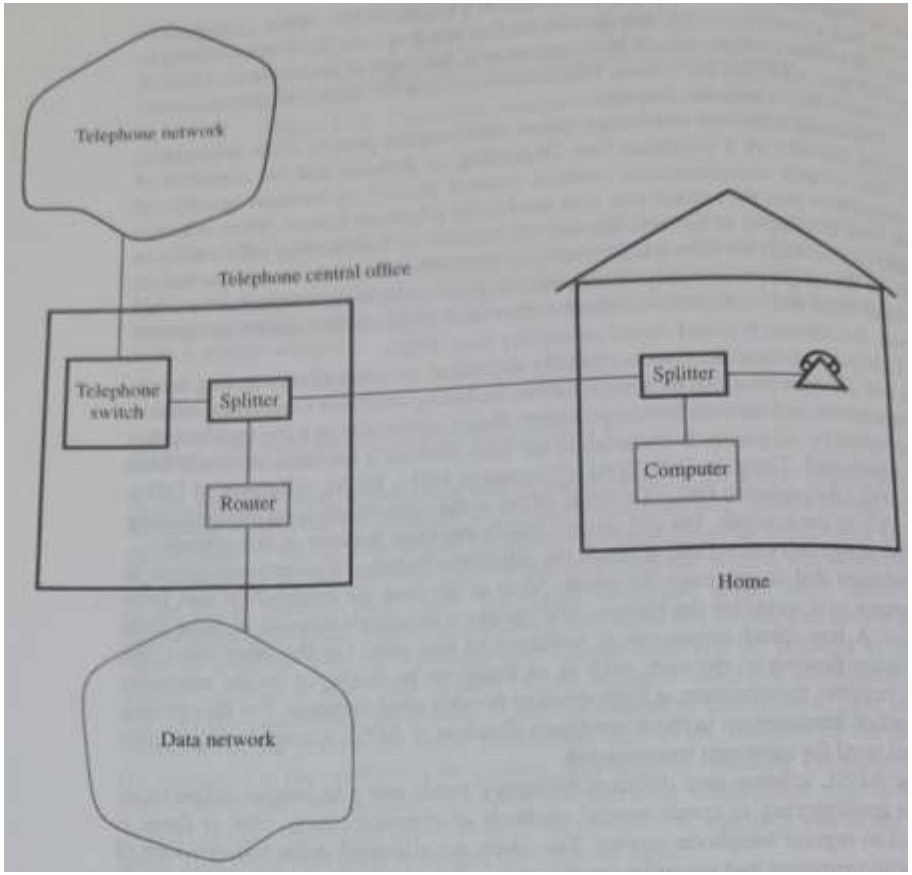


Fig. An ADSL Connection

b. Cable MODEM

- Cable Modem provides an alternative means for connecting a home computers to the internet. They use the cable TV connection instead of telephone connection.
- The Coaxial cable used in Cable TV as much higher Bandwidth than twisted pair wire. Hence the maximum speed possible with a cable modem is higher than what can be achieved with DSL schemes.
- However cable TV uses, a bus-like connection to all subscribers in a given neighborhood. Hence the information carrying capacity of the cable is shared among all those connected to it.
- The full capacity is available to one user only when no other users on the same cable are active, Shown in fig.
- The top speed available to any single user of a cable modem system is determined by the network service provider. It may vary from 600 KB/Sec to 10 MB/Sec.

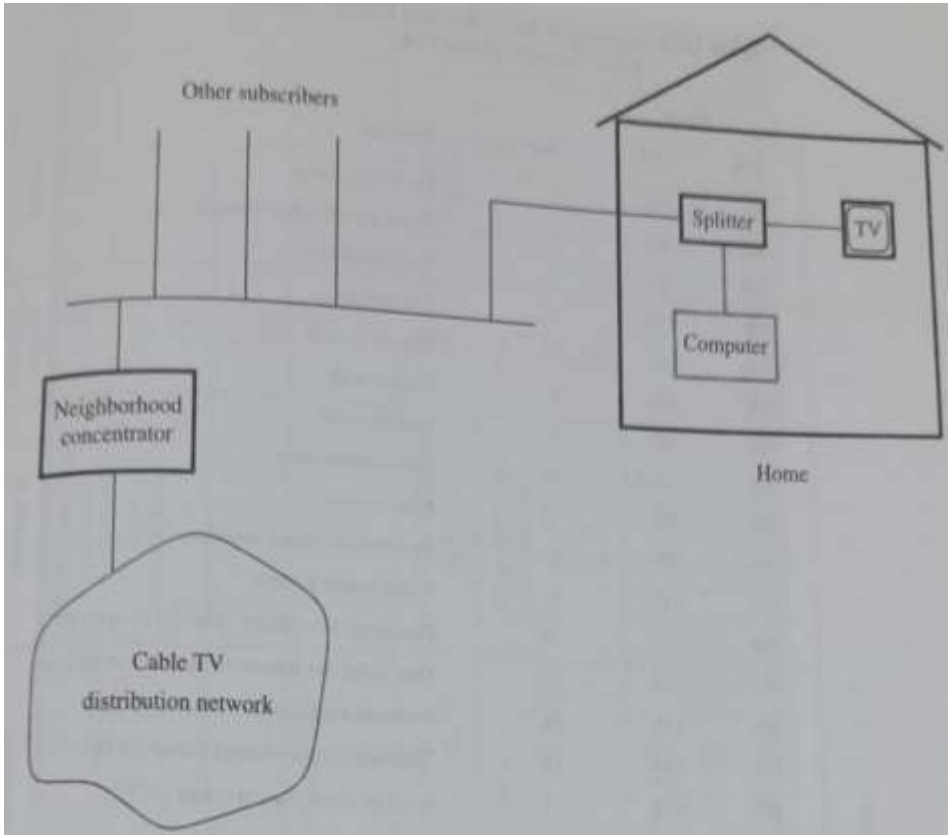


Fig. : A Cable-MODEM Connection

3. Std. Communication Interface

- A standard interface refers to the collection of points at which two devices are connected to each other.
- One such standard that has gained wide acceptance is the EIA (Electronics Industry Association) standard RS-232-C
- Outside north America, this is known as CCITT (Comite Consultatif International Telegrahique Telephonique) Recommendation V24.
- This standard completely specifies the interface between the data communication devices like modems and data equipment.
- The RS-232C interface consist of 25 connection points given in table.
- When the network server is ready to accept call, it sets the Data-terminal-ready signal (CD) to 1.
- Modem Monitors the telephone line, when it detects the ringing current that indicates incoming call (CE) to 1
- If CD=1 at the time the ringing current is detected, the modem automatically answer the call by setting modem ready signal CC to 1.

Table 10.2 Summary of the EIA Standard RS-232-C Signals (CCITT Recommendation V24).

Name			
EIA	CCITT	Pin* no.	Function
AA	101	1	Protective ground
AB	102	7	Signal ground-common return
BA	103	2	Transmitted data
BB	104	3	Received data
CA	105	4	Request to send
CB	106	5	Clear to send
CC	107	6	Data set ready
CD	108.2	20	Data terminal ready
CE	125	22	Ring indicator
CF	109	8	Received line signal detector
CG	110	21	Signal quality detector
CH	111	23§	Data signal rate selector (from DTE† to DCE‡)
CI	112	23§	Data signal rate selector (from DCE‡ to DTE†)
DA	113	24	Transmitter signal element timing (DTE†)
DB	114	15	Transmitter signal element timing (DCE‡)
DD	115	17	Receiver signal element timing (DCE‡)
SBA	118	14	Secondary transmitted data
SBB	119	16	Secondary received data
SCA	120	19	Secondary request to send
SCB	121	13	Secondary clear to send
SCF	122	12	Secondary received line signal detector

*Pins 9 and 10 are used for testing purposes, and pins 11, 18, and 25 are spare.
†Data terminal equipment.
‡Data communication equipment.
§The name of the signal on this pin depends on the signal's direction.