# Bio-Computer

**Dr. Mohannad K. Sabir Al Lami**

**Biomedical Engineering Department**

**2018-2019**

# Bio-Computer

- **Head lines**

1- Software Design and Validation,

2- Microprocessor,

3- Computer Architecture,

4- Real Time computing,

5- Embedded Software,

6- Graphical User Interface, and

7- Networking,

# Bio-Computer

## Suggested Readings


- John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach* (Third Edition ed.). Morgan Kaufmann Publishers.
- Gamma E. et al.: *Design patterns, WNT, Warszawa 2005*
- Miles R., Hamilton K.: *Learning UML 2.0, Helion, Gliwice 2007*
- Pressman R. S.: *Software Engineering: A Practitioner's Approach,*
- WNT, Warszawa 2004
- Sommerville I.: *Software Engineering, WNT, Warszawa 2003*

# Bio-Computer

**Importance of software engineering**

- National infrastructures are controlled by computer based Systems
- More and more systems require reliable software
- Software engineering is about theory, methods and tools used in software development
- Software development is an important part of economy in every developed country

# Bio-Computer

- **Software** is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.
- **Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods.

# Bio-Computer

Software Overview

- **Software engineering** is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.
- IEEE defines software engineering as: Software Engineering Tutorial

  (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

  (2) The study of approaches as in the above statement.

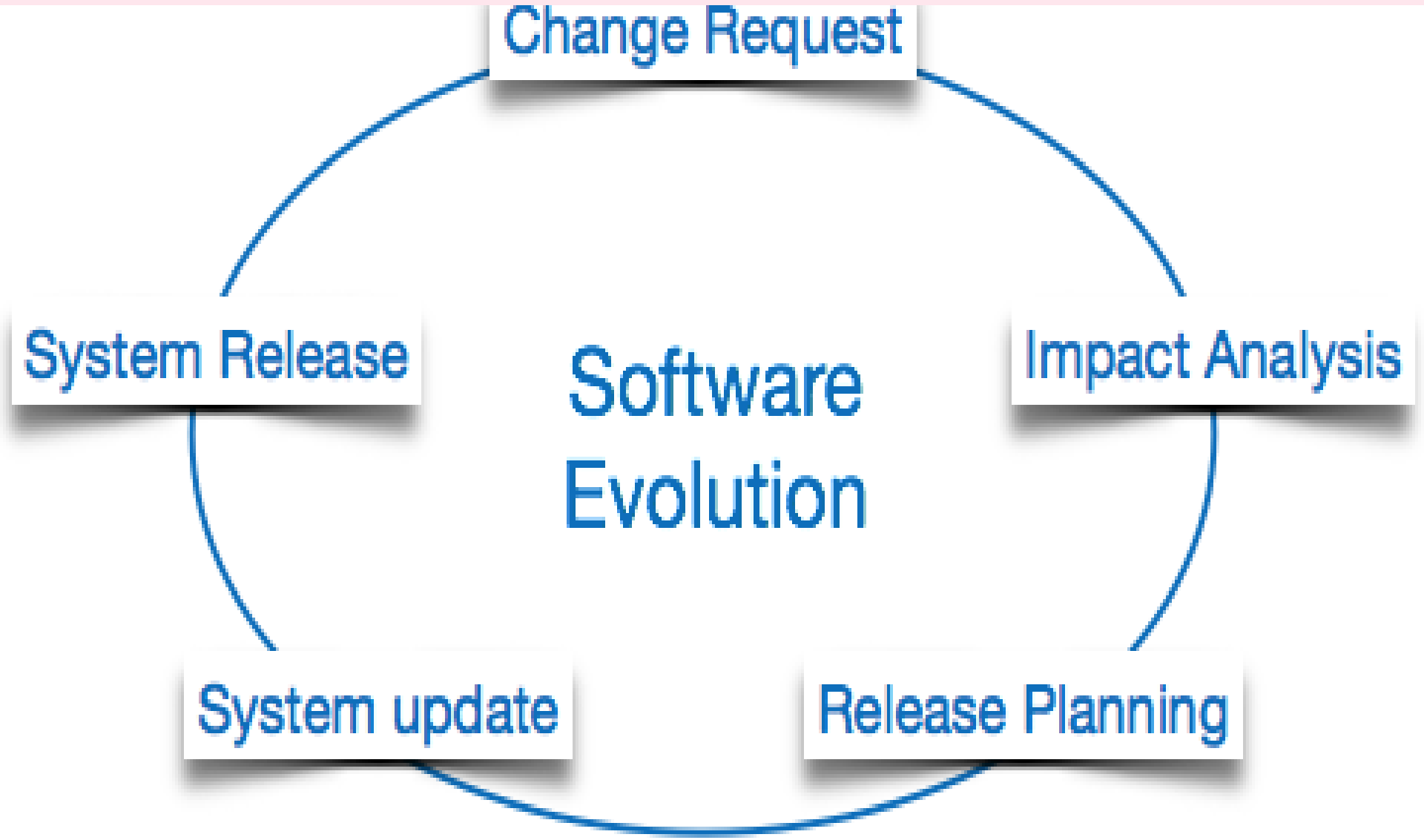
# Bio-Computer

# Bio-Computer

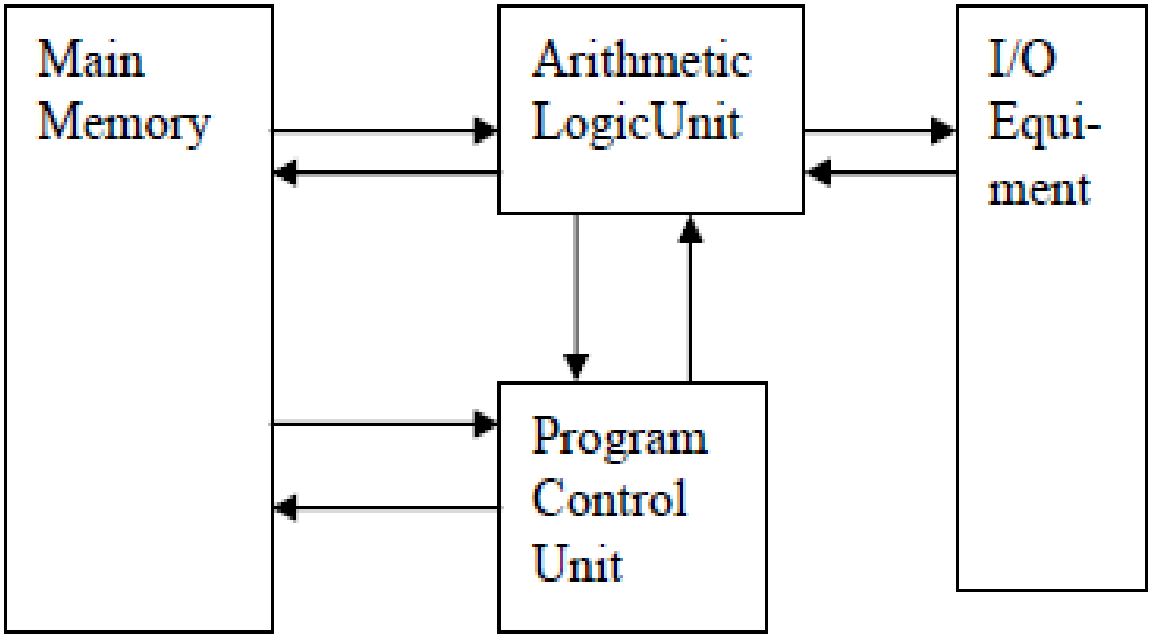
## Software Evolution

The process of developing a software product using software engineering principles and methods is referred to as **Software Evolution.** This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.

# Bio-Computer



# Bio-Computer




**Von Neumann Architecture**

# Bio-Computer

**General Structure of the IAS computer.**

- A main memory which stores data and instructions
- An arithmetic and logical unit(ALU) capable of operating on binary data.
- A control unit , which interprets the instruction in memory and causes them to be executed.
- Input and Output(I/O) equipment operated by control unit.

# Bio-Computer

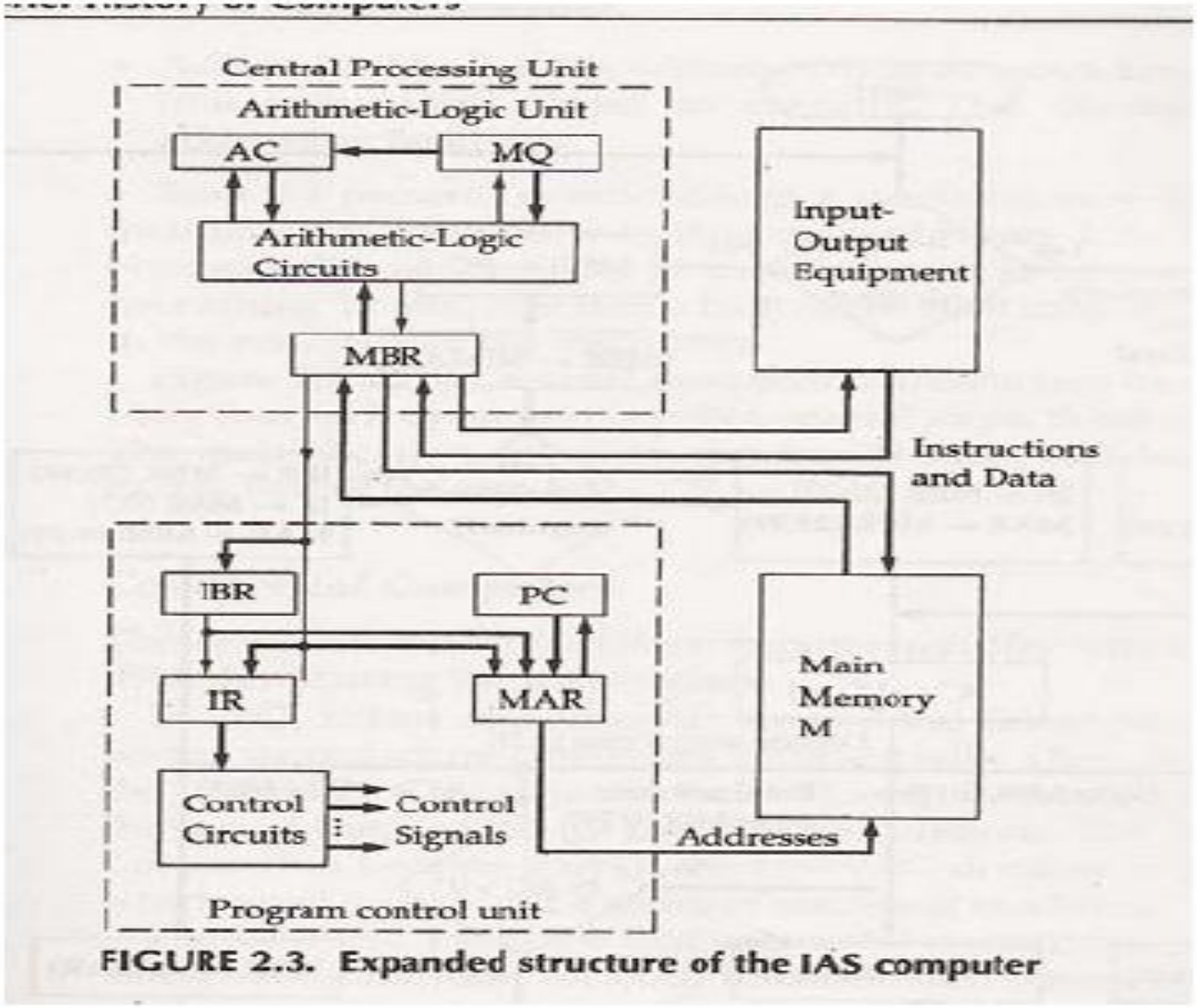



FIGURE 2.3. Expanded structure of the IAS computer

*Fig Expanded structure of the IAS computer*

# Bio-Computer

According to the above fig.

- (MBR) Memory Buffer Register : Contains the word to be stored in memory , or is used to receive a word from memory.
- (MAR) Memory Address Register: Specifies the address in the memory of the word to be written from or read into the MBR.
- (IR)Instruction register: Contains the 8 bit opcode instruction being executed
- (IBR)Instruction Buffer Register: Employed to temporartily hold the right – hand instruction from a word in memory.
- (PC)Program counter: Contains the address of the next instruction-pair to be fetched from memory.
- (AC)Accumulator and Multiplier-Quotient(MQ):Employed to temporarily hold operands and results of ALU operations. For eg. The result of multiplying two 40-bit numbers is an 80-bit number; the most significant 40 bits are stored in AC and the least significant in the MQ.

# Bio-Computer

- **STRUCTURE**
- The computer is an entity that interacts with its external environment. In general all its linkages to the external environment can be classified as peripheral devices or communication lines. There are four main structural components.
- *Central Processing Unit(CPU) : Controls the operation of the computer and performs its* data processing functions. Simply referred to as Processors.
- *Main memory : Store data.*
- *I/O : Moves data between computer and its external environment*
- *System Interconnection : Some mechanism that provides for communication among CPU,* main memory, and I/O.

# Bio-Computer

**FACTORS AFFECTING THE PERFORMANCE FACTORS:**

- Performance is specific to a particular program
- Total execution time is a consistent summary of performance
- Performance doesn't depend on any single factor: need to know Instruction Count, Clocks Per Instruction and Clock Rate to get valid estimations
- For a given architecture performance increases come from: increases in clock rate (without adverse CPI affects) – improvements in processor organization that lower CPI compiler enhancements that lower CPI and/or instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance

# Bio-Computer

- **Performance Calculation (1/2)**

CPU execution time for program= Clock Cycles for program x Clock Cycle Time

Substituting for clock cycles:

CPU execution time for program = (Instruction Count x CPI) x Clock Cycle Time = Instruction Count x CPI x Clock Cycle Time

# Bio-Computer

- **How to estimate a performance of computer?**

  One of the generic measures is MIPS (millions of instructions per second). This is only meaningful when comparing machines with the same architecture, since some architectures may require substantially more instructions than others for the same program. This method also can be very dependent on the mix of instructions and hence on the program used to measure MIPS.

  Some manufacturers report "peak MIPS" on carefully designed but useless programs.

  It is obvious, that all major computer components such as CPU, memory and IO devices together affect computer's performance. Slow RAM or hard disk is going to be a bottleneck for fast CPU. In reality, however, high performance of PC is always a trade off to low cost:

# Bio-Computer

| Option | High performance | Low cost |
|---|---|---|
| Bus architecture | Separate address/data | Multiplex address/data |
| Data bus width | Wider means faster | Low pin count is cheaper |
| Bus masters | Multiple (requires arbitration) | Single (no arbitration) |
| Transfer size | Multiple words | Single word |
| Clocking | Synchronous | Asynchronous |

# Bio-Computer

- **1. The CPU.**

  CPU architecture is important. The higher the generation, the better. For example, because of high performance new features, Pentium 75 (fifth generation with the clock rate 75 MHz) outperforms 80486DX100 (which is the fourth generation CPU with the clock rate 100MHz). One of the techniques, enhancing the performance , is *parallel processing. For example, while an* instruction is being executed in the ALU (E), the next instruction can be fetched from memory (F) and decoded (D).

  *Instruction Prefetching is another idea, first appeared in 286 (6 byte prefetching). It is based on* the fact, that CPU is normally performing sequential code fetching. Only jump instructions alter program flow and they are statistically rare. Rather than wait for the execution unit to request next instruction fetch, CPU during next cycle prefetches the next instruction from memory and put it into prefetch queue to have it ready. If jump instruction is executed the information in prefetch queue is marked as invalid.

# Bio-Computer

**2. Data bus width.**

80486 processors have data bus 32 bits wide, whereas Pentiums are 64 bit processors, thus Pentiums can transfer twice as much data at a time compared to fourth generation CPUs.

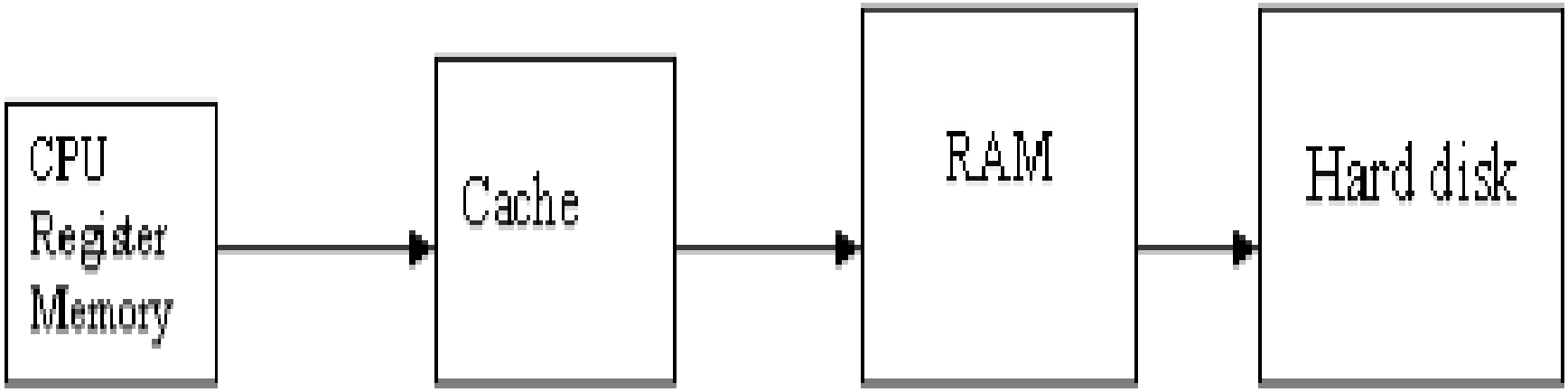
**3. Clock rate.**

Since any step of processing can happen only on the "tick" of the clock , the faster the rate the quicker the CPU works.

**4. Memory.**

- The diagram illustrates a general memory hierarchy of PC:

# Bio-Computer




| | | | | |
|---|---|---|---|---|
| Size | <1KB | <4MB | <512KB | <100GB |
| Access time | 2-5 nS | 3-10 nS | 50-100 nS | 5mS |

# Bio-Computer

- The amount of RAM really depends on your applications. Reasonable performance today calls for 128 MB. Adding more RAM will speed up the performance if you run several applications at the same time or work with large files and documents.
- L1 cache resides on-chip. The bigger the on-chip cache size - the better, since more instructions and data can be stored on the chip, reducing the number of times the processor has to access slower off-chip memory areas to get data.

# Bio-Computer

**5- IO devices**

Speaking of effective interfacing I/O devices to CPU, *synchronous protocol (includes a clock* in the control lines) is more effective than *asynchronous. A synchronous interface means* data and address are transmitted relative to the clock. Since little or no logic is needed to decide what to do next, a synchronous interface can be both fast and inexpensive. A disadvantage of this protocol is that it can not be long because of the clock-skew problem.

An asynchronous interface does not need clock. Instead, self-timed, handshaking protocols are used between sender and receiver.

# Bio-Computer

Most I/O devices today are *interrupt-driven , i.e. CPU does not do anything for the I/O* device until it notifies the CPU by sending *interrupt (IRQ). First computers used polling – a* simple interface, when the CPU periodically checked status bits to see if it is time for the next I/O operation. Since CPU is much faster than any I/O device, it is obvious that polling is a waste of the CPU's time. In general-purpose applications, using IRQ is the key to multitasking operating systems and good response time.

- Since I/O events often involve block transfers, *direct memory access (DMA) hardware is* added to many computer systems. DMA is when I/O device acts as a master and transfers large number of words to/from memory without intervention by the CPU.

# Bio-Computer

- **Byte – The amount of space in memory or on a disk needed to store one character.**

  **8 bits = 1 Byte**
- Kilo means 1000 kilobyte (KB) = 1000 Bytes
- Mega means 1,000,000 megabyte (MB) = 1,000,000 Bytes
- Giga Means 1,000,000,000 gigabyte (GB) = 1,000,000,000 Bytes

# Bio-Computer

## Central Processing Unit (CPU)

- The central processing unit is one of the two most important components of your microcomputer.
- It is the electronic brain of your computer. In addition to processing data, it controls the function of all the other components. The most popular microprocessors in IBM compatible computers are made by Intel.

# Bio-Computer

The generations of microprocessors are listed below.

- 1981 8088
- 1984 80286
- 1987 80386
- 1990 80486
- 1993 Pentium
- 1996 P-1
- 2002 P-4

# Bio-Computer

## System Software

- System software will come provided with each computer and is necessary for the computer's operation.
- This software acts as an interpreter between the computer and user. It interprets your instructions into binary code and likewise interprets binary code into language the user can understand.
- In the past you may have used MS-DOS or Microsoft Disk Operating System which was a command line interface.

# Bio-Computer

**Program Software**

- Program software is software used to write computer programs in specific computer languages.
- Application software is any software used for specified applications such as:

1. Word Processing
2. Spreadsheet
3. Database
4. Presentation Graphics
5. Communication
6. Tutorials
7. Entertainment, Games

# Bio-Computer

## Emerging Trends

- There are standard buses such as Industry Standard Architecture (ISA), Extended Industry Standard Architecture (EISA), Micro- Channel Architecture (MCA), and so on. The standard bus permits the user to purchase the components from different vendors and connect them easily.
- The various input and output devices have a standard way of connecting to the CPU and Memory. These are called interface standards. Some popular interface standards are the RS-232C and Small Computer System Interconnect (SCSI).

# Bio-Computer

## Main Memory

- A **flip-flop** made of electronic semiconductor devices is used to fabricated a memory cell. These memory cells organized as a **Random Access Memory (RAM).**
- A memory cell, which does not loose the bit stored in it when no power is supplied to the cell, is know as a **non-volatile cell.**
- A RAM may be fabricated with permanently stored

information, which cannot be erased. Such a memory is called a **Read Only Memory (ROM).**

# Bio-Computer

- a user can store his won special functions or programs in a ROM. Such ROM's are called **Programmable ROM (PROM).**
- The time taken to write a word is known as the Write time. The time to retrieve information is called the **Access time of the memory.**

# Bio-Computer

## Application Software

- It is the set of programs necessary to carry out operations for a specified application.

  Example Programs:
  - To solve a set of equations
  - To process examination results
  - To prepare a Pay-Bill for an organization
  - To prepare Electricity-Bill for each month.

# Bio-Computer

## System Software

- These are general program written for the system, which provide the environment to facilitate writing of Application software. Some of the system programs are given below:
- **Compiler**: It is a translator system program used to translate a High-level language program into a Machine language program.
- **Assembler**: It is another translator program used to translate an Assembly language program into a Machine language program.
- **Interpreter**: It is also a translator system program used to translate a High level language program into a Machine language program, but it translates and executes line by line.
- **Loader:** It is a system program used to store the machine language program into the memory of the computer.

# Bio-Computer

## Computer Languages

1. **Machine language:**The computers can execute a program written using binary digits only.
2. **Assembly Language:** In assembly language mnemonic codes are used to develop program for problem solving.

- **Program code Description**

| | |
|---|---|
| READ A | It reads the value of A. |
| ADD B | The value of B is added with A |
| STORE C | The result is store in C. |
| PRINT C | The result in 'C' is printed. |
| HALT | Stop execution. |

# Bio-Computer

3. **High Level Languages:** High level language are developed to allow application programs, which are machine independent.

- **FORTRAN (FORmula TRANslation),**
- **BASIC (Beginner's All-purpose Symbolic**
- **Instruction Code),**
- **COBOL (COmmon Business Oriented Language).**
- Visual Foxpro, Visual Basic (VB), Visual C++ (VC++) are more
- popular among the software developers.

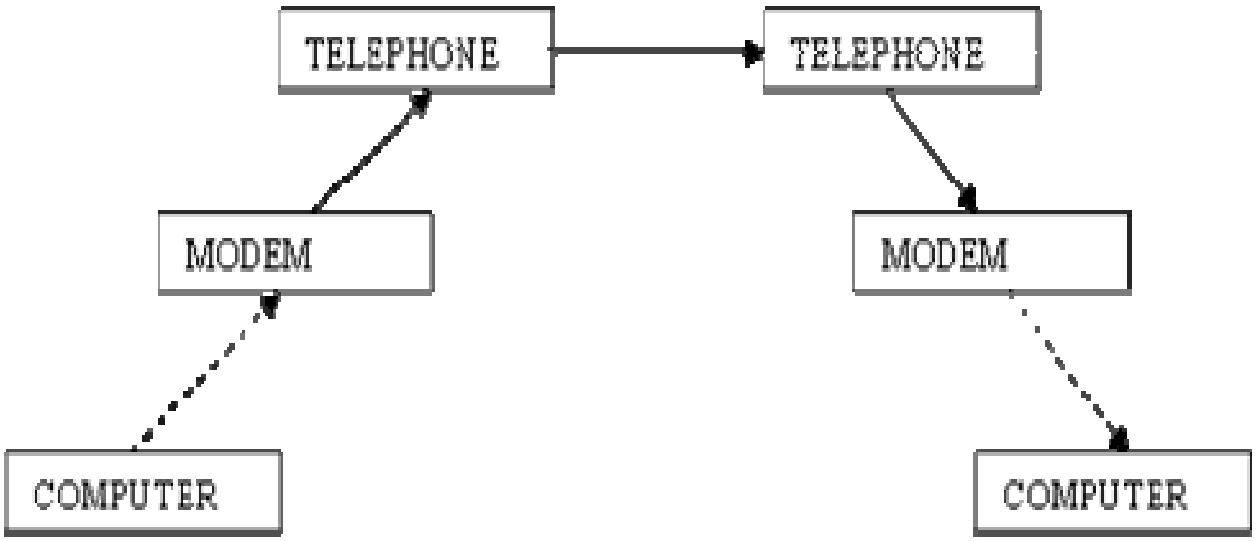
# Bio-Computer

## Computers and Communications

**Local Area Network (LAN) & Wide Area Network (WAN)**

- One way of connecting the computers is by using devices called **modems.**
- A modem is used to transfer data from one computer to another using the telephone lines.
- Interconnection of computers which are within the same building or nearby locations forms a
- network of computers and this network is called a **Local Area Network (LAN).**
- A LAN permits sharing of data files, computing resources and peripherals.
- Interconnection of computers located in far away locations using telecommunication system is known as **Wide Area Network (WAN).**

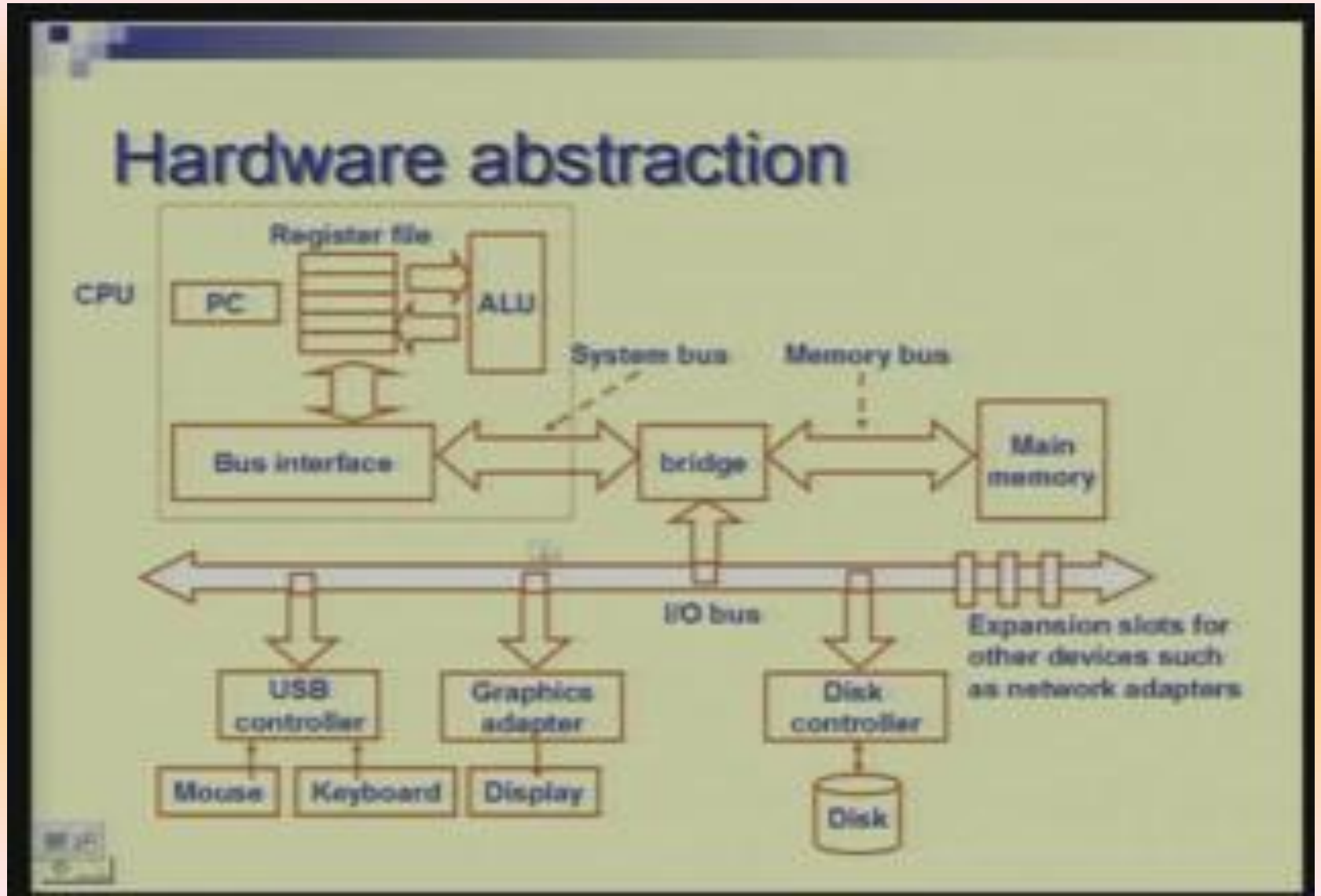
# Bio-Computer




Indicating Digital signal

Indicating Digital signal

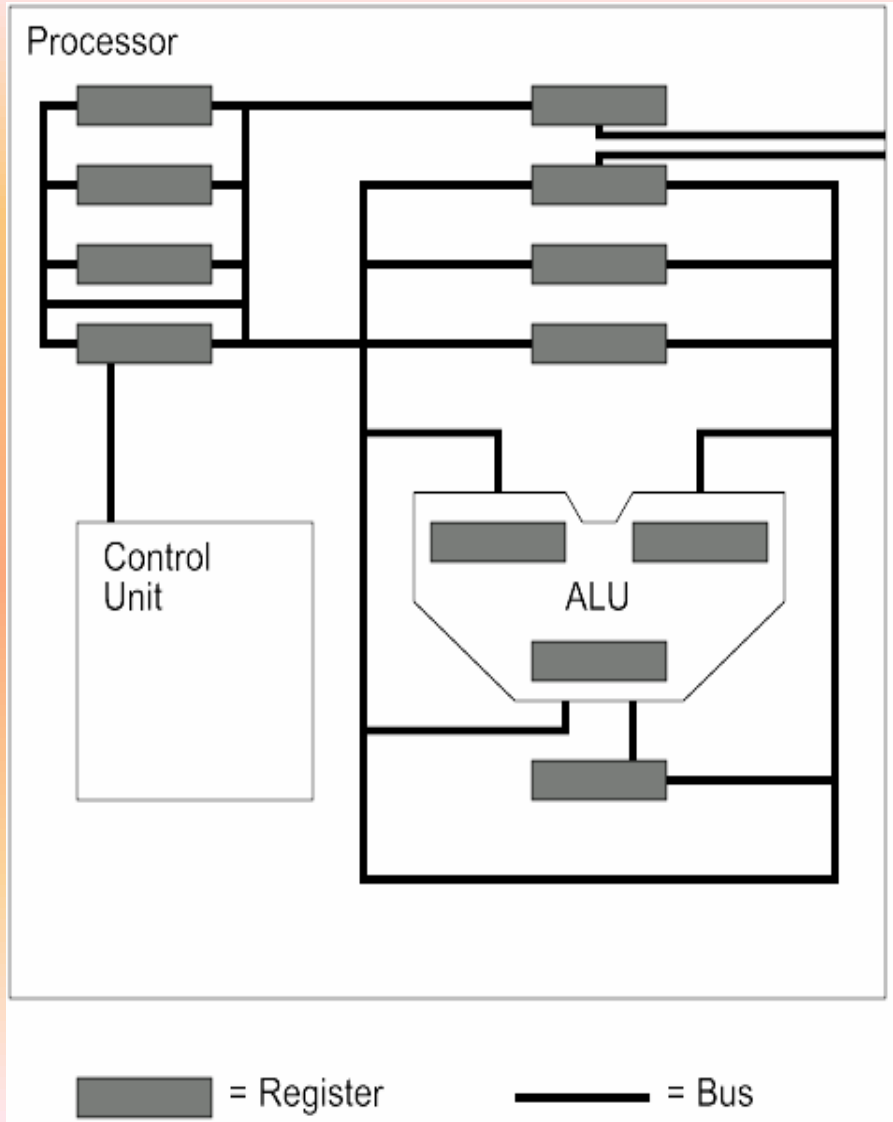
# Bio-Computer



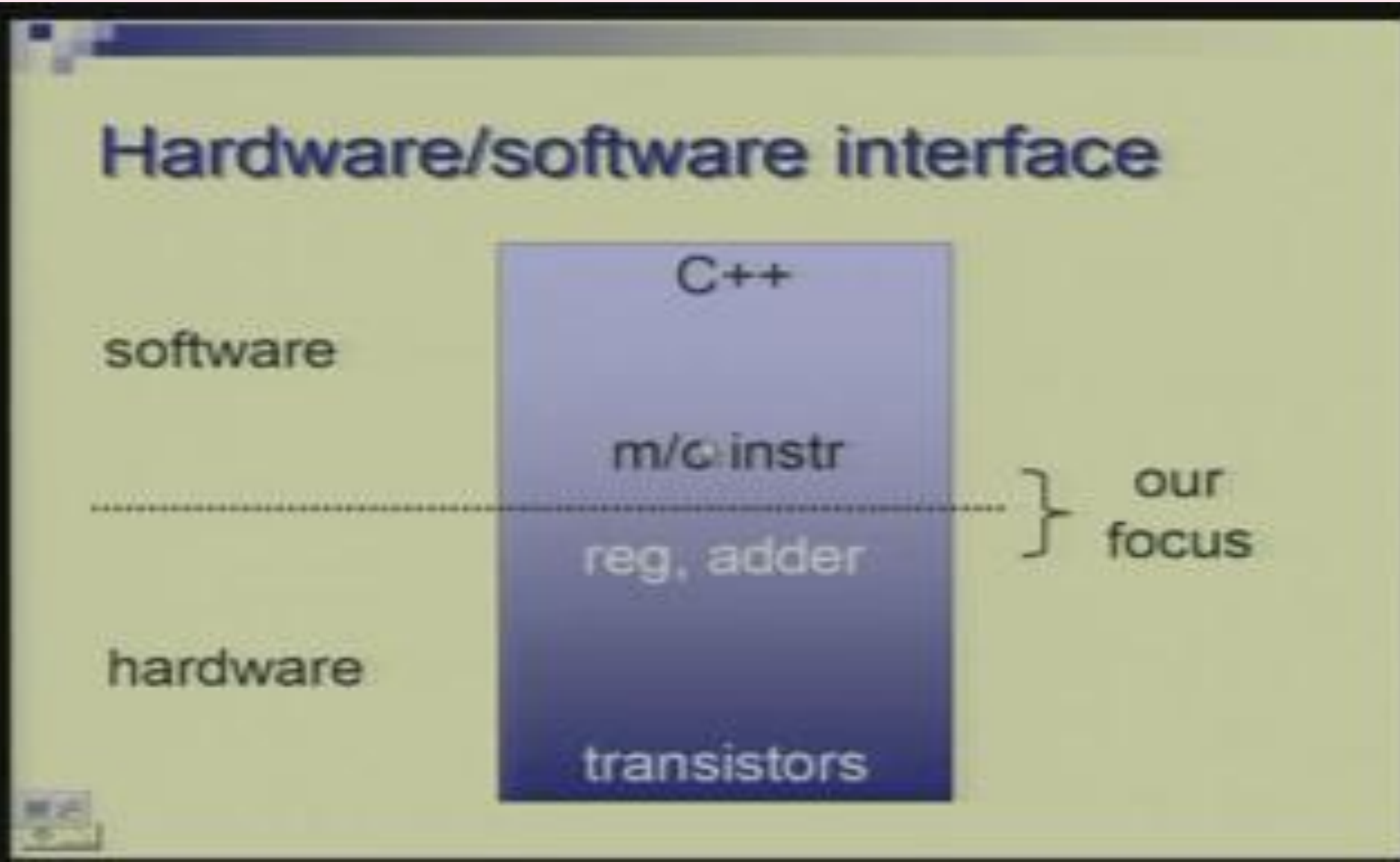
# Bio-Computer

# Bio-Computer

**To build a simple processor**
we need the following components:
– Some *Registers - a register is a* store where we can place one piece of data;
– An *Arithmetic Logic Unit, or ALU - a very basic calculator* for our processor. The ALU will have some registers inside it, as we will see later;
– A *Control Unit, or CU - to run* the processor;
– Some buses - to allow us to move data from one component to another.



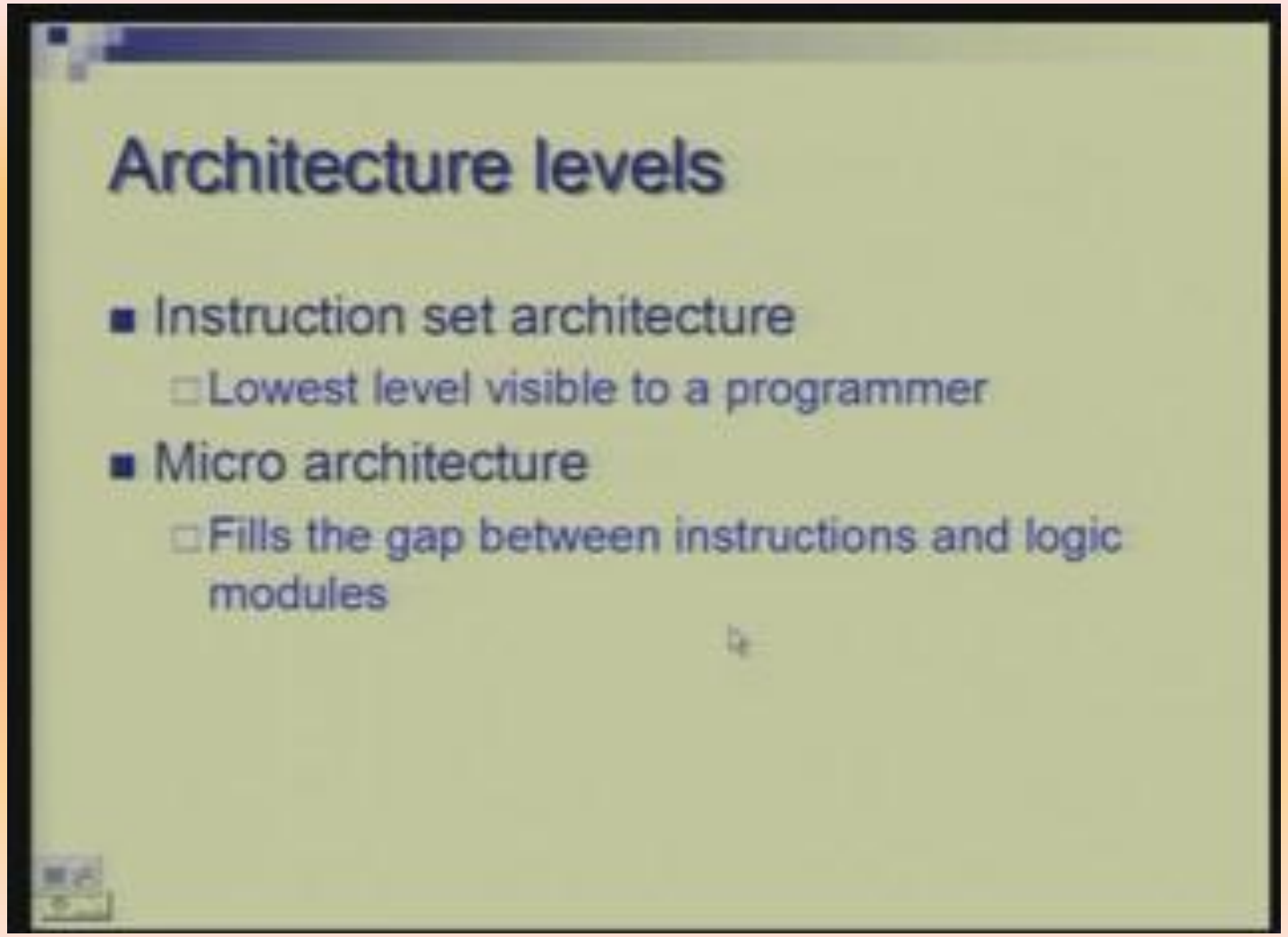
# Bio-Computer



# Bio-Computer

On the hardware side we saw major building blocks, registers, adders and so on and at the bottom you have individual components and transistors. So our focus would be somewhere here in the middle where you see hardware software boundary. So, what exactly is hardware software boundary? It is where you have a set of instructions which define the basic capability of a processor and major hardware components which are able to understand those instructions. So, if you are a programmer you will see the machine as defined by a set of instructions whereas if you are a hardware designer you will see software in terms of those machine instructions which you need to interpret. So there are levels of hierarchy here within hardware and level of hierarchy within software.

# Bio-Computer

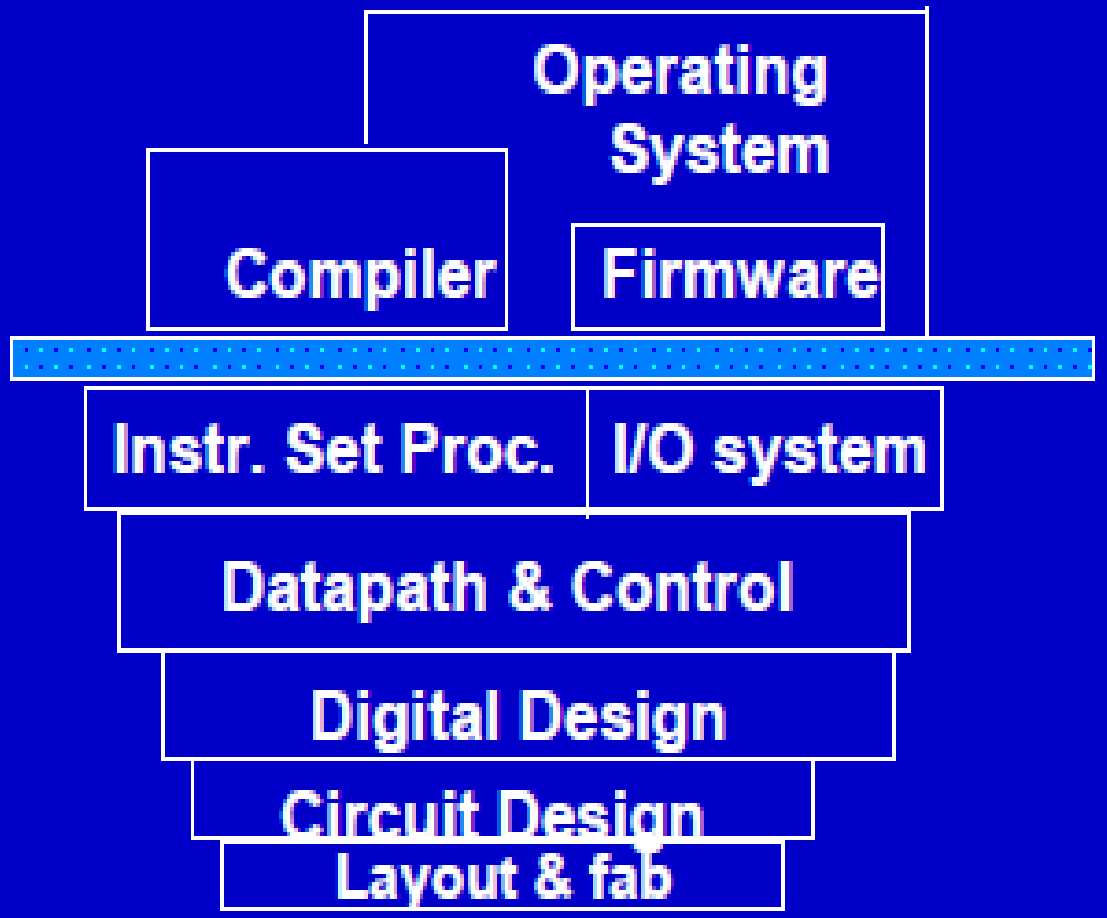


# Bio-Computer

- Architecture itself, which is around interface between hardware and software
- The instruction set architecture refers to the lowest level visible to a programmer. The programmer is not concerned about your transistors or your gates or your flip-flops or adders and so on
- The basic unit of computation is an instruction whereas micro architecture is what concerns a hardware designer more and it fills up the gap between the instruction and the logic modules

# Bio-Computer

## What is "Computer Architecture"?




*Semiconductor Materials*

# Bio-Computer

- This is the typical view of where we place instruction set architecture in between software and hardware
- At the top you have application programs which are able to run on a processor with the help of some system software.
- Below ISA level you have the broad CPU design then at a lower level you have circuit design and for fabricating the circuit, for physically realizing this you need to have a layout where you need to worry about where you place the transistor where you place a wire how you interconnect them and so on

# Bio-Computer

**Instruction Set Architecture**

**-- Organization of Programmable Storage**

**-- Data Types & Data Structures: Encodings & Representations**

**-- Instruction Formats**

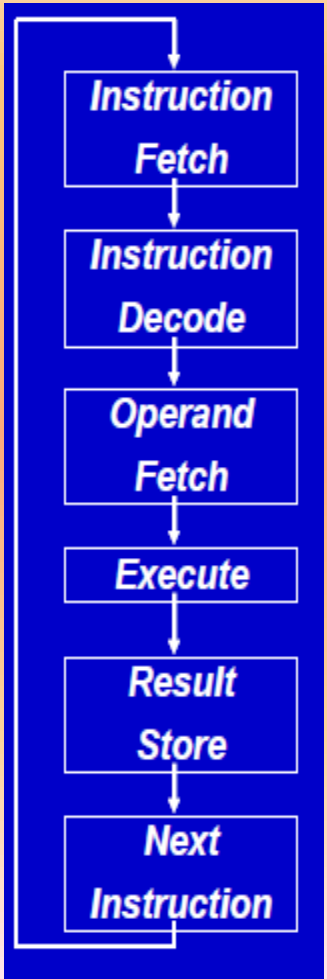
**-- Instruction (or Operation Code) Set**

**-- Modes of Addressing and Accessing Data Items and Instructions**

**-- Exceptional Conditions**

# Bio-Computer

## Fundamental Execution Cycle




- **Obtain instruction from program storage**
- **Determine required actions and instruction size**
- **Locate and obtain operand data**
- **Compute result value or status**
- **Deposit results in storage for later use**
- **Determine successor instruction**

# Bio-Computer

Elements of an ISA

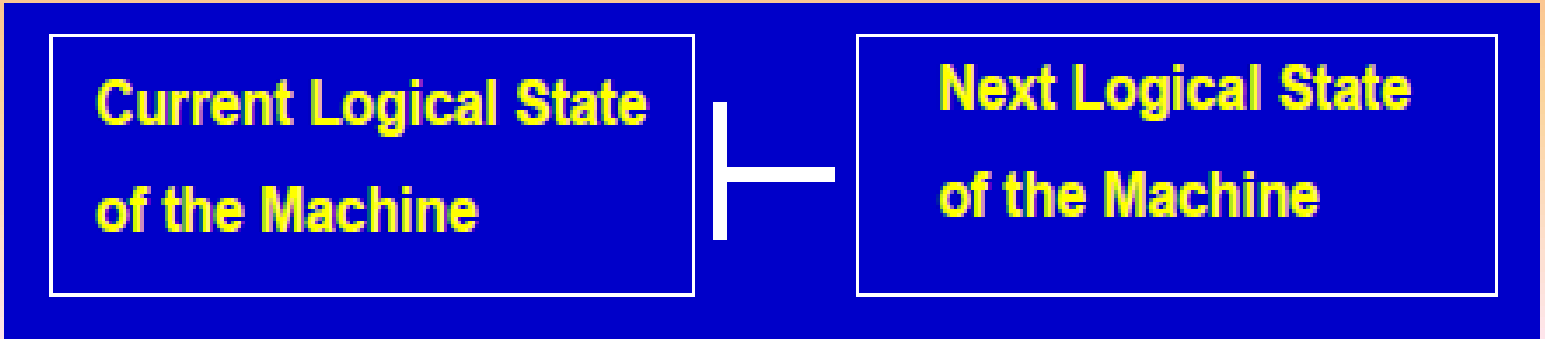
- Set of machine-recognized data types
  - **bytes, words, integers, floating point, strings, . . .**
- Operations performed on those data types
  - **Add, sub, mul, div, xor, move, ….**
- Programmable storage
  - **regs, PC, memory**
- Methods of identifying and obtaining data referenced by instructions

  (addressing modes)
  - **Literal, reg., absolute, relative, reg + offset, …**
- Format (encoding) of the instructions
  - **Op code, operand fields, …**

# Bio-Computer

## Computer as a State Machine

- State: defined by storage
  - **Registers, Memory, Disk, …**
- Next state is influenced by the operation
  - **Instructions, I/O events, interrupts, …**
- When is the next state decided?
  - **Result Store: Register write, Memory write**
  - **Output: Device (disk, network) write**



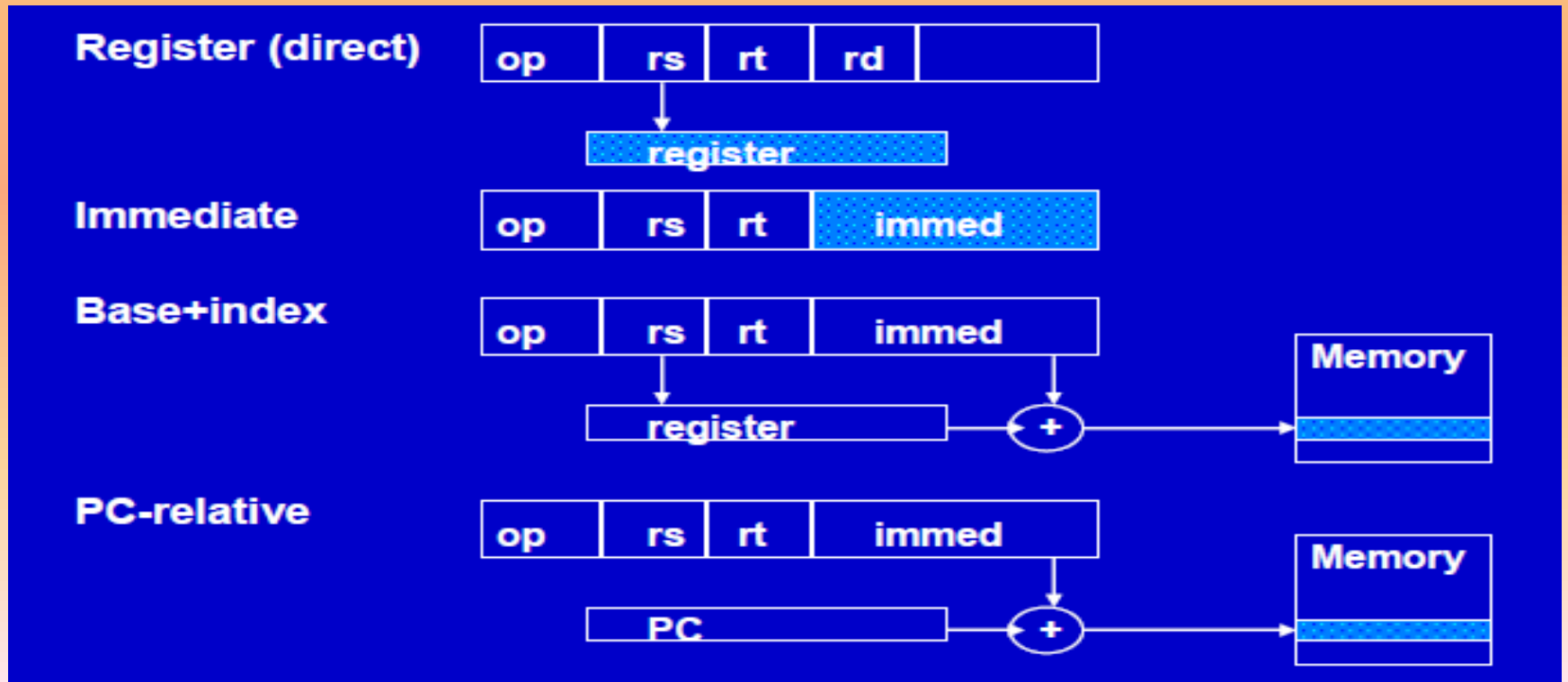
# Bio-Computer

- **Accumulator**:
  1 address add A — acc ← acc + mem[A]
  1+x address addx A — acc ← acc + mem[A + x]
- **Stack**:
  0 address add — tos ← tos + next
- **General Purpose Register**:
  2 address add A B — EA(A) ← EA(A) + EA(B)
  3 address add A B C — EA(A) ← EA(B) + EA(C)
- **Load/Store**:
  3 address add Ra Rb Rc — Ra ← Rb + Rc
  load Ra Rb — Ra ← mem[Rb]
  store Ra Rb — mem[Rb] ← Ra

# Bio-Computer

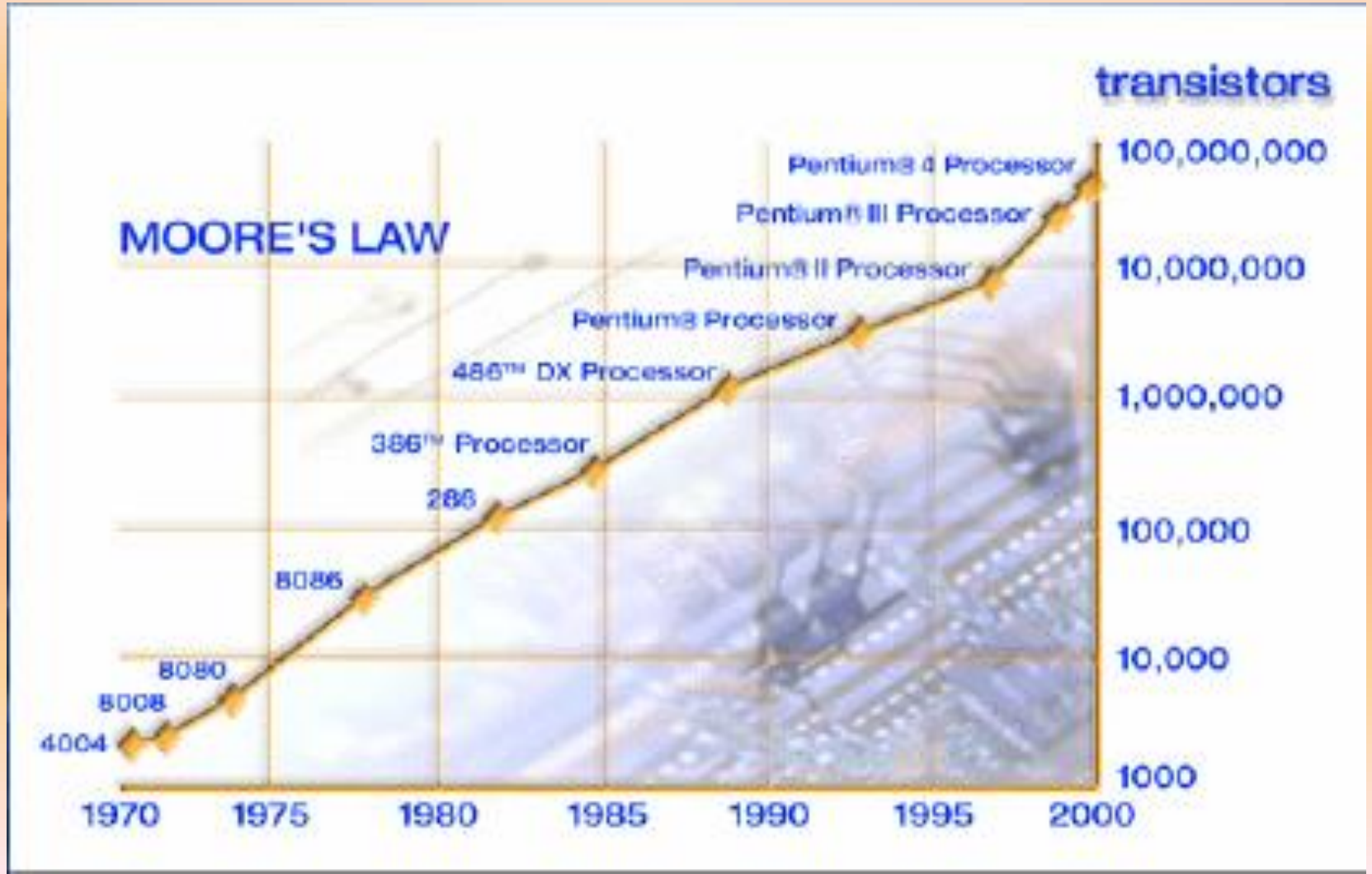
## Addressing Modes & Formats

- Simple addressing modes
- All instructions 32 bits wide



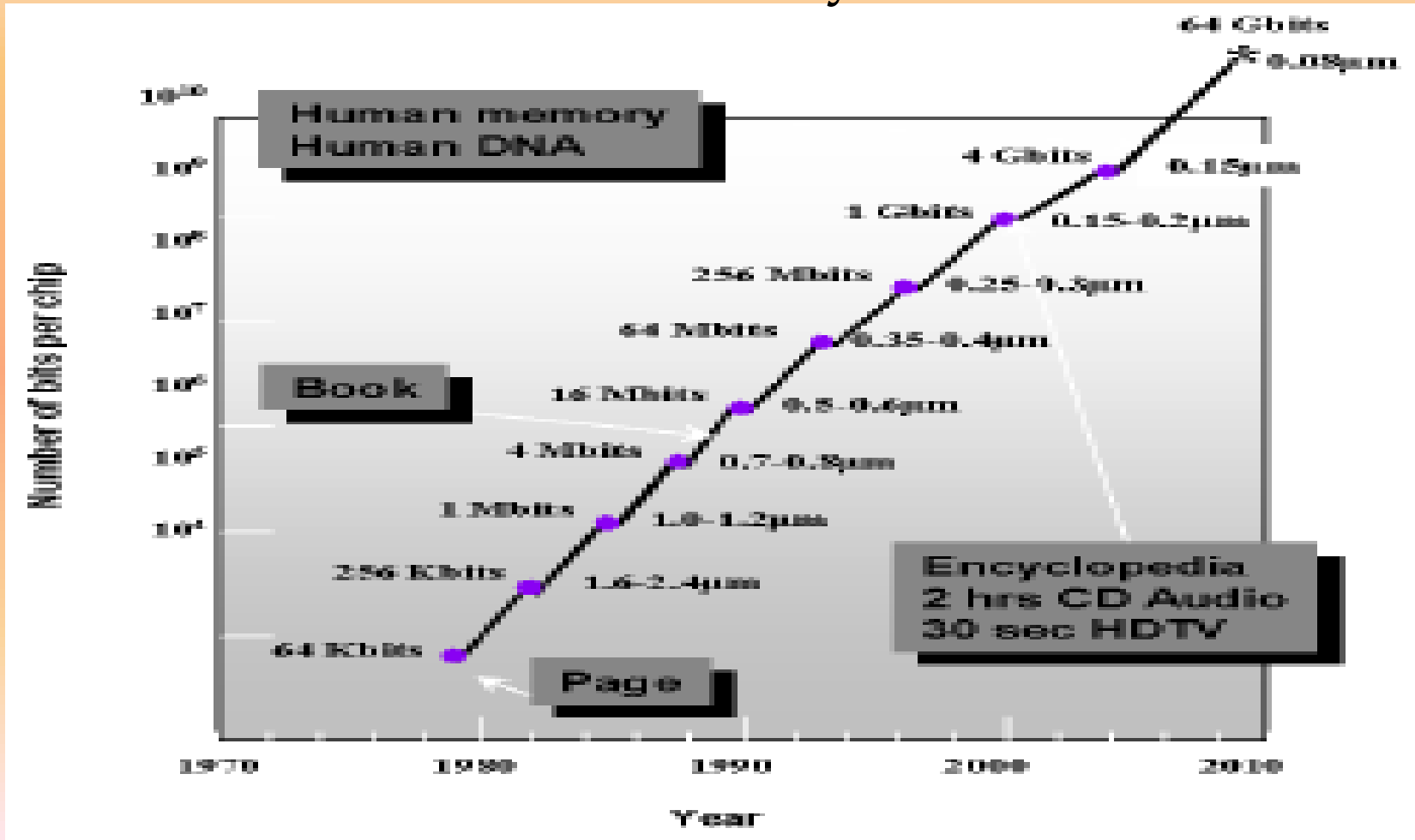
# Bio-Computer

Illustration of Moor's Law



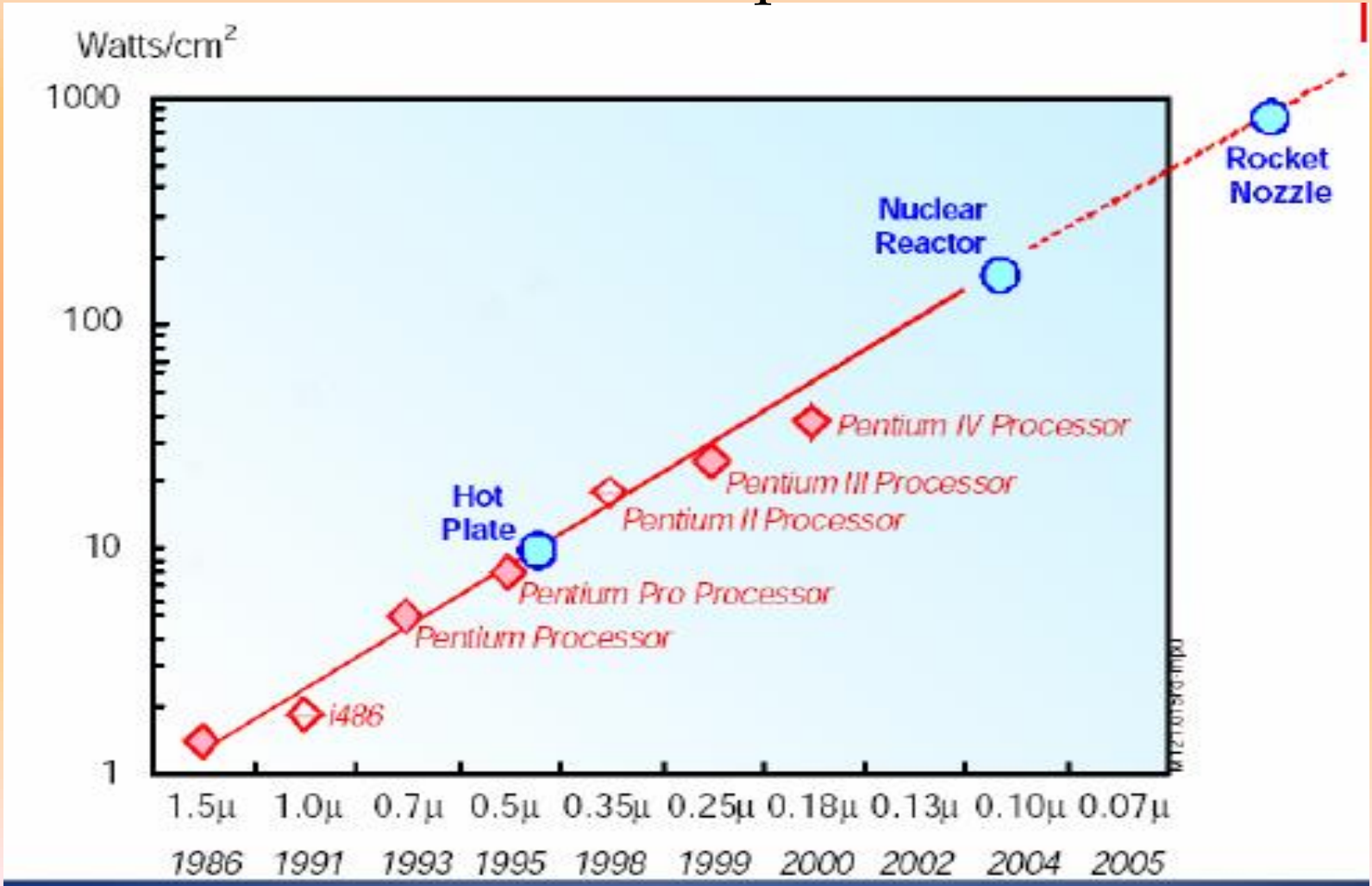
# Bio-Computer

Memory



# Bio-Computer

Power dissipation



# Bio-Computer

## Application Software Issues

- Regardless of the language you use, most of the important software issues are:
- How numbers are represented by bit patterns,
- Round-off error in computer arithmetic
- The computational speed of different types of processors, etc.

# Bio-Computer

## Computer Numbers

- A fixed number of bits are allocated to store each number, usually 8, 16, 32 or 64.
- This is usually provided by an algorithm or formula for converting between the represented value and the corresponding bit pattern, and back again.
- only two general formats have become common, ***fixed point*** *(also called integer numbers) and* ***floating point*** (also called real numbers).

# Bio-Computer

- When you evaluate the formats presented pages, try to understand them in terms of their **range** (the largest and smallest numbers they can represent) and their **precision (**the size of the gaps between numbers).

**Fixed Point (Integers)**

- Fixed point representation is used to store *integers, the positive and negative* whole numbers: -3,-2,-1, 0, 1, 2, 3,.. .

# Bio-Computer

- **unsigned integer format**, the possible bit patterns are assigned to the numbers 0 through $2^N$ .
- The disadvantage of unsigned integer is that negative numbers cannot be represented.
- **Offset binary** is similar to unsigned integer, except the decimal values are *shifted to allow for negative numbers*.
- The most important use of offset binary is in ADC and DAC.

# Bio-Computer

- **Sign and magnitude** is another simple way of representing negative integers.
- The far left bit is called the **sign bit,** and is made a *zero for positive numbers,* and a *one for negative numbers.*

| UNSIGNED INTEGER | | OFFSET BINARY | | SIGN AND MAGNITUDE | | TWO'S COMPLEMENT | |
|---|---|---|---|---|---|---|---|
| Decimal | Bit Pattern | Decimal | Bit Pattern | Decimal | Bit Pattern | Decimal | Bit Pattern |
| 15 | 1111 | 8 | 1111 | 7 | 0111 | 7 | 0111 |
| 14 | 1110 | 7 | 1110 | 6 | 0110 | 6 | 0110 |
| 13 | 1101 | 6 | 1101 | 5 | 0101 | 5 | 0101 |
| 12 | 1100 | 5 | 1100 | 4 | 0100 | 4 | 0100 |
| 11 | 1011 | 4 | 1011 | 3 | 0011 | 3 | 0011 |
| 10 | 1010 | 3 | 1010 | 2 | 0010 | 2 | 0010 |
| 9 | 1001 | 2 | 1001 | 1 | 0001 | 1 | 0001 |
| 8 | 1000 | 1 | 1000 | 0 | 0000 | 0 | 0000 |
| 7 | 0111 | 0 | 0111 | 0 | 1000 | -1 | 1111 |
| 6 | 0110 | -1 | 0110 | -1 | 1001 | -2 | 1110 |
| 5 | 0101 | -2 | 0101 | -2 | 1010 | -3 | 1101 |
| 4 | 0100 | -3 | 0100 | -3 | 1011 | -4 | 1100 |
| 3 | 0011 | -4 | 0011 | -4 | 1100 | -5 | 1011 |
| 2 | 0010 | -5 | 0010 | -5 | 1101 | -6 | 1010 |
| 1 | 0001 | -6 | 0001 | -6 | 1110 | -7 | 1001 |
| 0 | 0000 | -7 | 0000 | -7 | 1111 | -8 | 1000 |
| 16 bit range: 0 to 65,535 | | 16 bit range -32,767 to 32,768 | | 16 bit range -32,767 to 32,767 | | 16 bit range -32,768 to 32,767 | |

# Bio-Computer

- **Two's complement** is the format loved by hardware engineers, and is how integers are usually represented in computers.
- Converting between decimal and two's complement is straightforward for positive numbers, a simple decimal to binary conversion.
- For negative numbers, the following algorithm is often used:

(1) take the absolute value of the decimal number,

(2) convert it to binary,

(3) complement all of the bits (ones become zeros and zeros become ones),

(4) add 1 to the binary number.

# Bio-Computer

## Floating Point (Real Numbers)

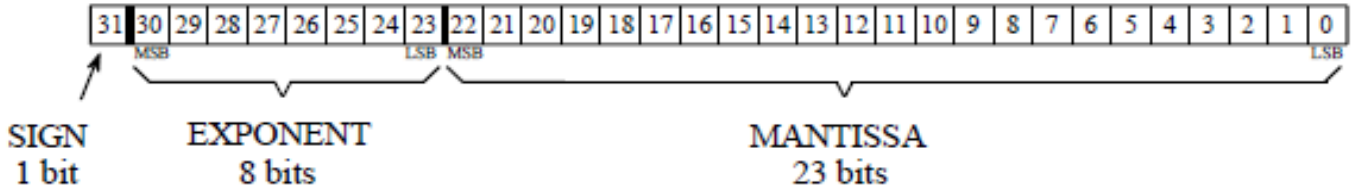
- The basic idea is the same as used in scientific notation, where a **mantissa** is multiplied by ten raised to some **exponent**.
- ANSI/IEEE Std. 754-1985 is the most common standard defines the format for 32 bit numbers called **single precision**, as well as 64 bit numbers called **double precision**.
- These bits form the floating point number, *v*, *by* the following relation:

$$v = (-1)^S \times M \times 2^{E-127}$$

# Bio-Computer

- Floating point numbers are *normalized in the same way as scientific notation, that is, there* is only one nonzero digit left of the decimal point (called a *binary point in* base 2).
- Since the only nonzero number that exists in base two is 1, the leading digit in the mantissa will always be a 1, and therefore does not need to be stored.

# Bio-Computer




Example 1

0 00000111 11000000000000000000000

\+ 7 0.75

$$+1.75 \times 2^{(7-127)} = +1.316554 \times 10^{-36}$$

Example 2

1 10000001 01100000000000000000000

\- 129 0.375

$$-1.375 \times 2^{(129-127)} = -5.500000$$

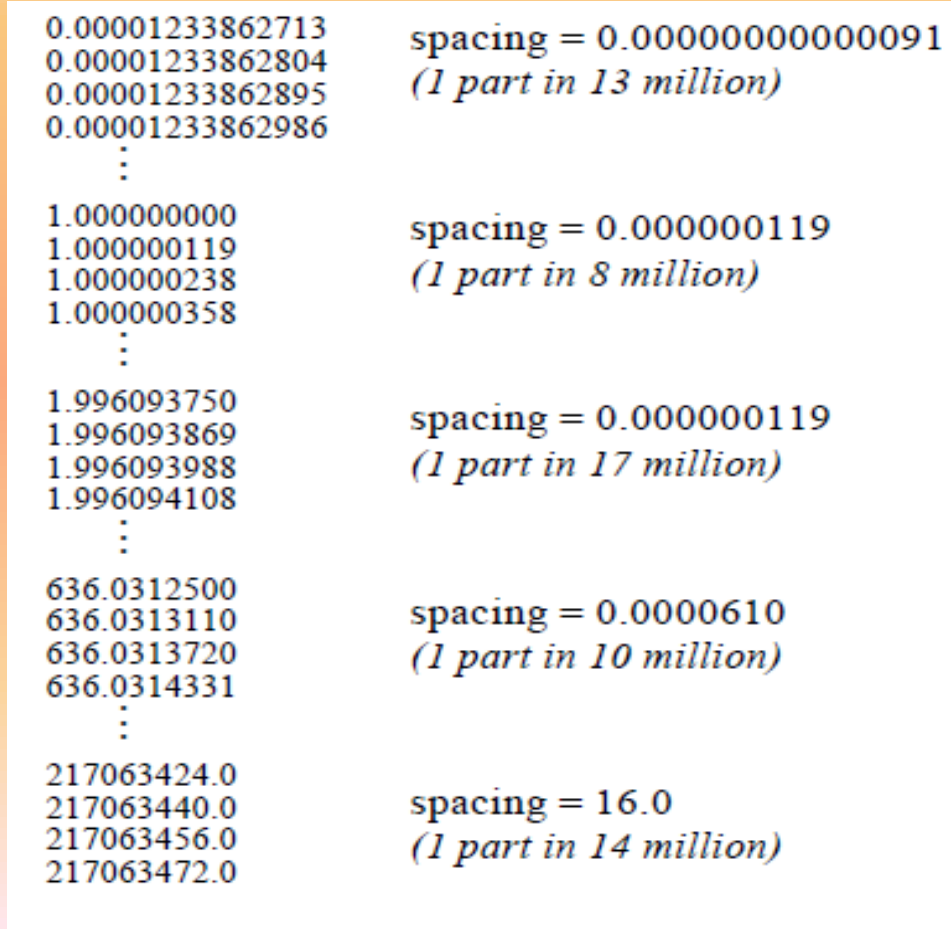
# Bio-Computer

## Number Precision

- The errors associated with number representation are very similar to quantization errors during ADC.
- With fixed point variables, the gaps between adjacent numbers are always *exactly one*.
- In floating point notation, the gaps between adjacent numbers vary over the represented number range.

# Bio-Computer

This is a key concept of floating point notation: large numbers have large gaps between them, while small numbers have small gaps.

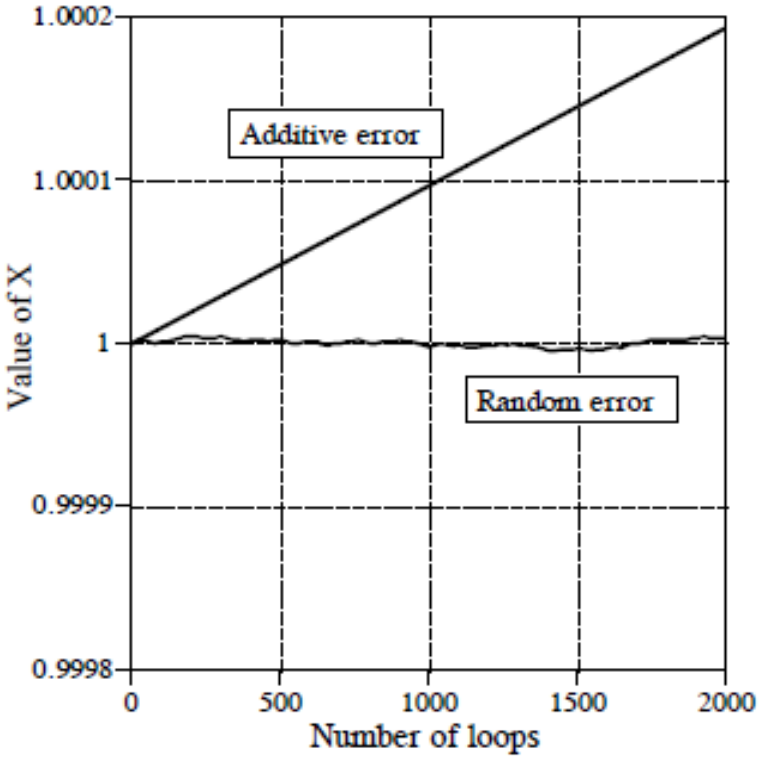


# Bio-Computer

- The ***additive error*** is roughly equal to the round-off error from a single operation, multiplied by the total number of operations.
- The ***random error*** only increases in proportion to the *square root of the number of operations.*
- Additive error can be hundreds of times worse than random error for common DSP algorithms.

# Bio-Computer

```
100  X = 1                  'initialize X
110  '
120  FOR I% = 0 TO 2000
130    A = RND              'load random numbers
140    B = RND              'into A and B
150    '
160    X = X + A            'add A and B to X
170    X = X + B
180    X = X - A            'undo the additions
190    X = X - B
200    '
210    PRINT X              'ideally, X should be 1
220  NEXT I%
230  END
```



# Bio-Computer

**Execution Speed: Program Language**

- DSP programming can be loosely divided into three levels of sophistication:

1. *Assembly,*
2. *Compiled, and*
3. *Application Specific.*

- Assembly programming requires an extensive understanding of the internal construction of the particular microprocessor you intend to use.

# Bio-Computer

- **Compiled or high-level languages**, is the next level of sophistication can manipulate *abstract variables without any reference to the* particular hardware.
- A program called a **compiler** is used to transform the high-level source code directly into machine code.
- An **interpreter** converts a *single line of source code into machine code, executes* that machine code, and then goes on to the next line of source code.

# Bio-Computer

## Execution Speed: Programming Tips

- *How you program* can be changed at any time, and will drastically affect how long the program will require to execute.
- Here are three suggestions:

1. **First, use integers instead of floating point variables whenever possible.**
2. **Second, avoid using functions such as: sin(*x*), *log*(*x*), $y^x$ , etc.**

- Another option is to pre-calculate these slow functions, and store the values in a **look-up table (LUT).**

3. **Third, learn what is *fast and what is slow on your particular system.***

# Bio-Computer

## ELEMENTS OF BUS DESGIN

- ***Type***
  **Dedicated**
  Multiplexed
- ***Method of Arbitration***
  Centralized
  Distributed
- **Timing**
  Synchronous
  Asynchronous

***Bus Width***
**Address**
Data

***Data Transfer Type***
Read
Write
Read-modify-write
Read-after-write
Block

# Bio-Computer

## Bus Types

- **Dedicated**

  Separate data & address lines

- **Multiplexed**

➢ Shared lines

➢ Address valid or data valid control line

| ➢ Advantage | Disadvantages |
| --- | --- |
| fewer lines | More complex control |
| | Ultimate performance |

# Bio-Computer

## *Method of Arbitration*

- In all but the simplest systems, more than one module may need control of the bus.
- Since only one unit at a time can successfully transmit over the bus, some method of arbitration is needed.
- The various methods can be roughly classified as being either centralized or distributed.

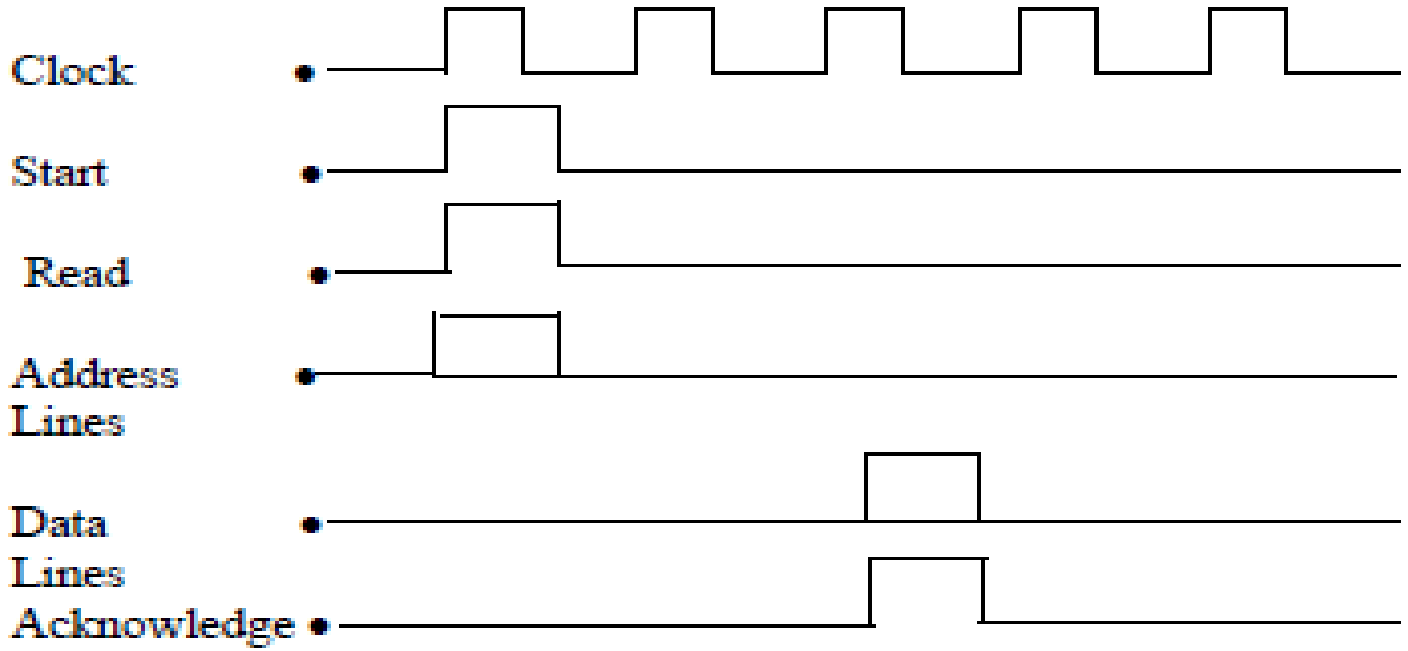
# Bio-Computer

- **centralized scheme**, a single hardware device (separate module or part of the CPU), referred to as bus controller or arbiter, is responsible for allocating time on the bus.
- **distributed scheme**, there is no central controller, each module contains access control logic and the modules act together to share the bus.
- With both methods of arbitration, the purpose is to designate one device as **master**.
- The **master** may then initiate a data transfer (e.g. read or write) with some **other device**, which acts as **slave** for this particular exchange.

# Bio-Computer

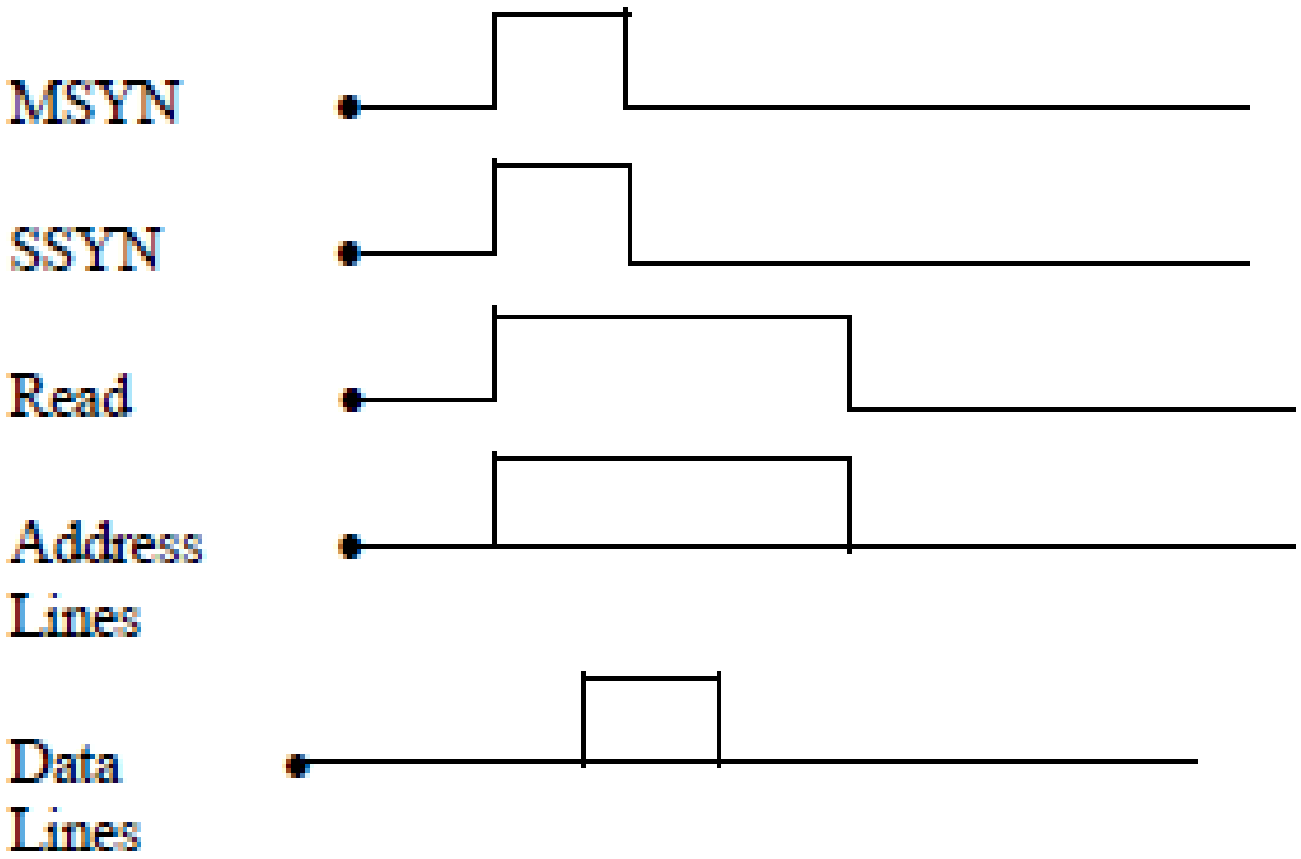
## *Timing*

- Timing refers to the way in which events are coordinated on the bus.




(a) Synchronous Timing

# Bio-Computer




(b) Asynchronous Timing

# Bio-Computer

## Bus Width

- The width of the **data bus** has an impact on system performance: the **wider** the **data bus**, the greater the number of bits transferred at one time.
- The width of the **address bus** has an impact on system capacity: the **wider** the **address bus**, the greater the range of locations that can be referenced.

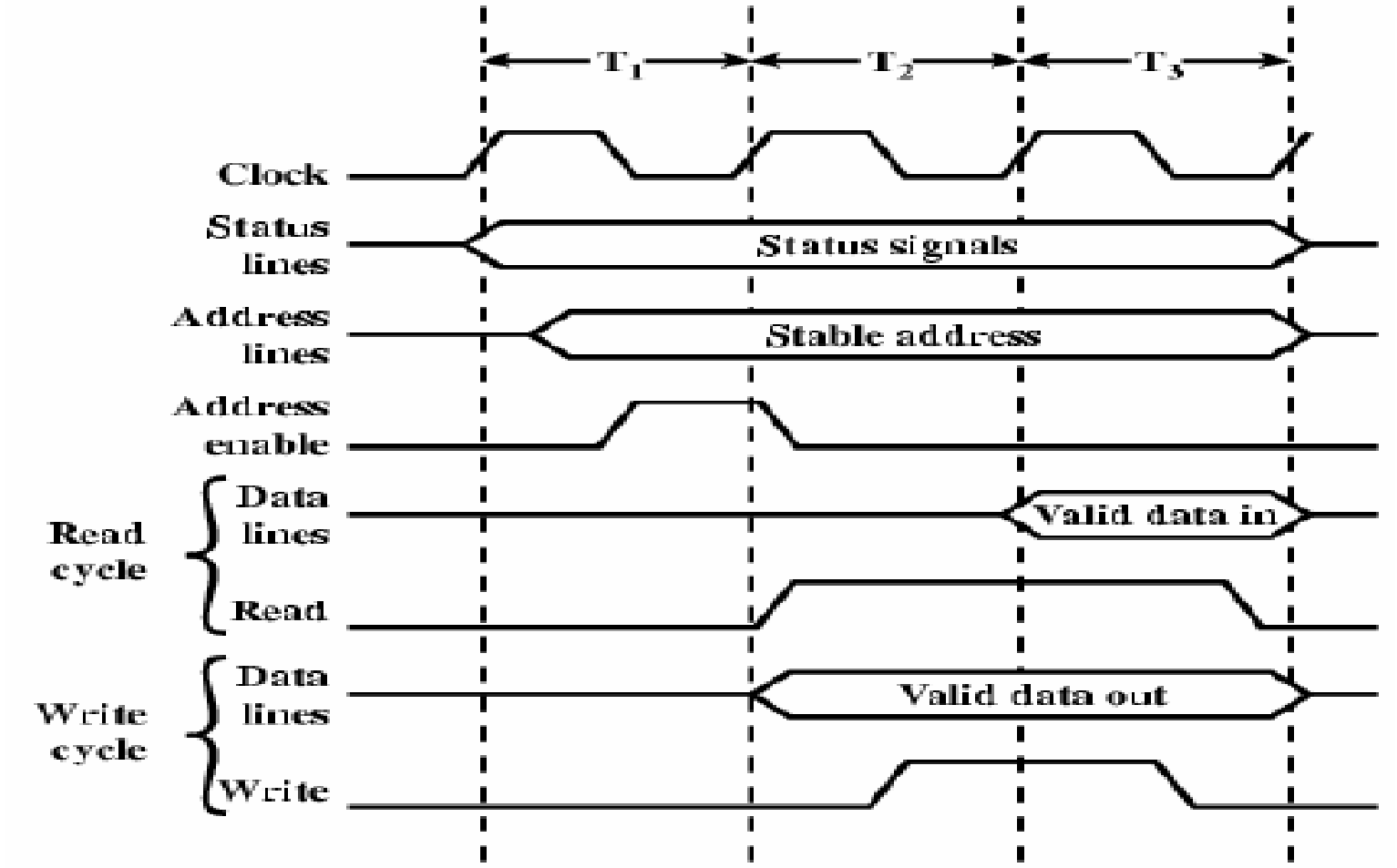
# Bio-Computer

## Data Transfer Type

- All buses support both write (master to slave) and read (slave to master) transfers.
- In the case of a **multiplexed** address/data bus, the bus is **first** used for specifying the address and **then** for transferring the data.
- For a read operation, there is typically a **wait** while the data is being fetched from the slave to be put on the bus.

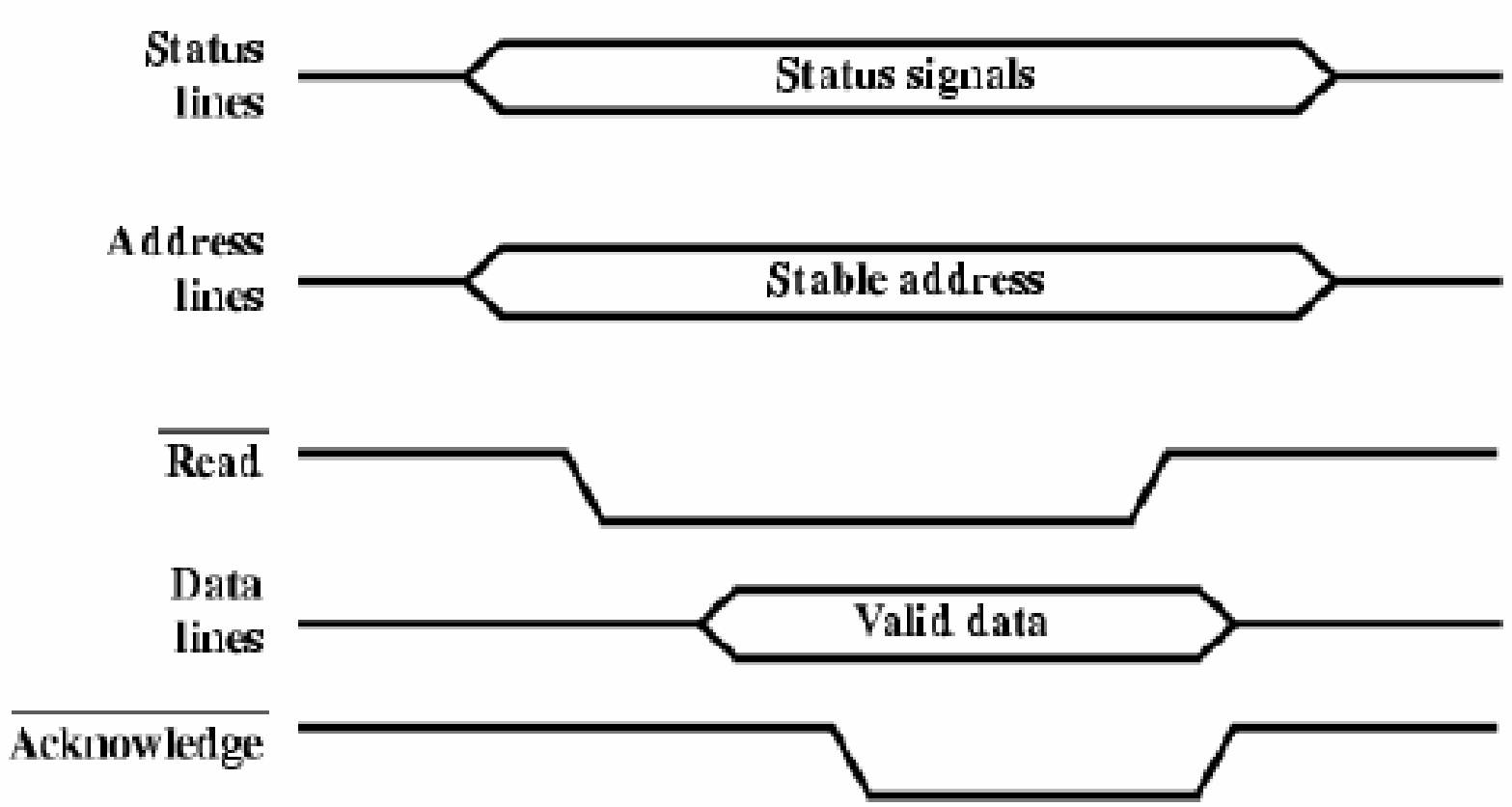
# Bio-Computer

## Synchronous Timing Diagram



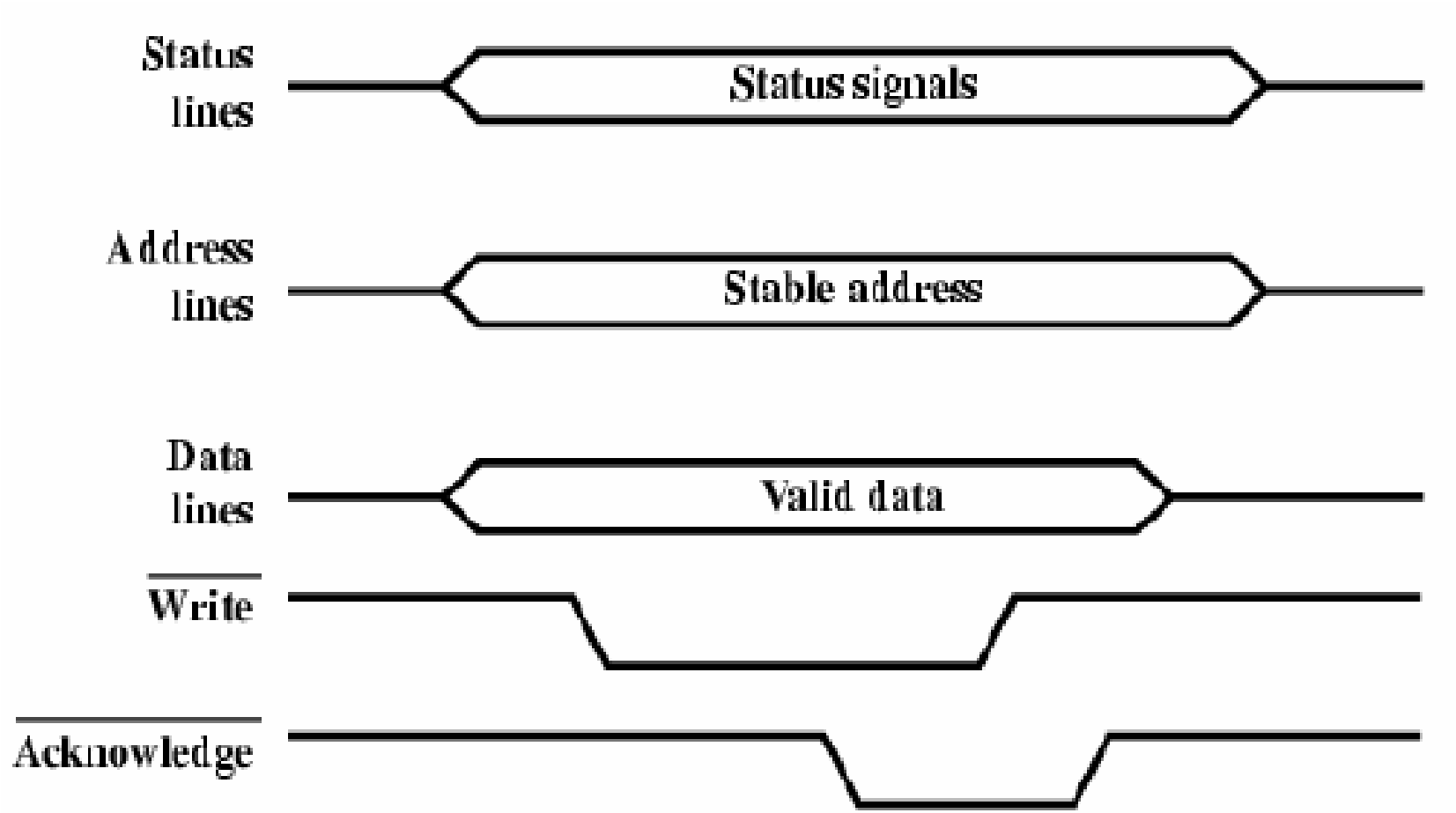
# Bio-Computer

## Asynchronous Timing Diagram – Read Diagram



# Bio-Computer

## Asynchronous Timing Diagram – Write Diagram



# Bio-Computer

**PCI Bus**

- Peripheral Component Interconnection
- Intel released to public domain
- 32 or 64 bit
- 50 lines

**PCI Bus Lines (required)**

- Systems lines
  - Including clock and reset
- Address & Data
  - 32 time mux lines for address/data
  - Interrupt & validate lines
- Interface Control
- Arbitration
  - Not shared
  - Direct connection to PCI bus arbiter
- Error lines

# Bio-Computer

**PCI Bus Lines (Optional)**

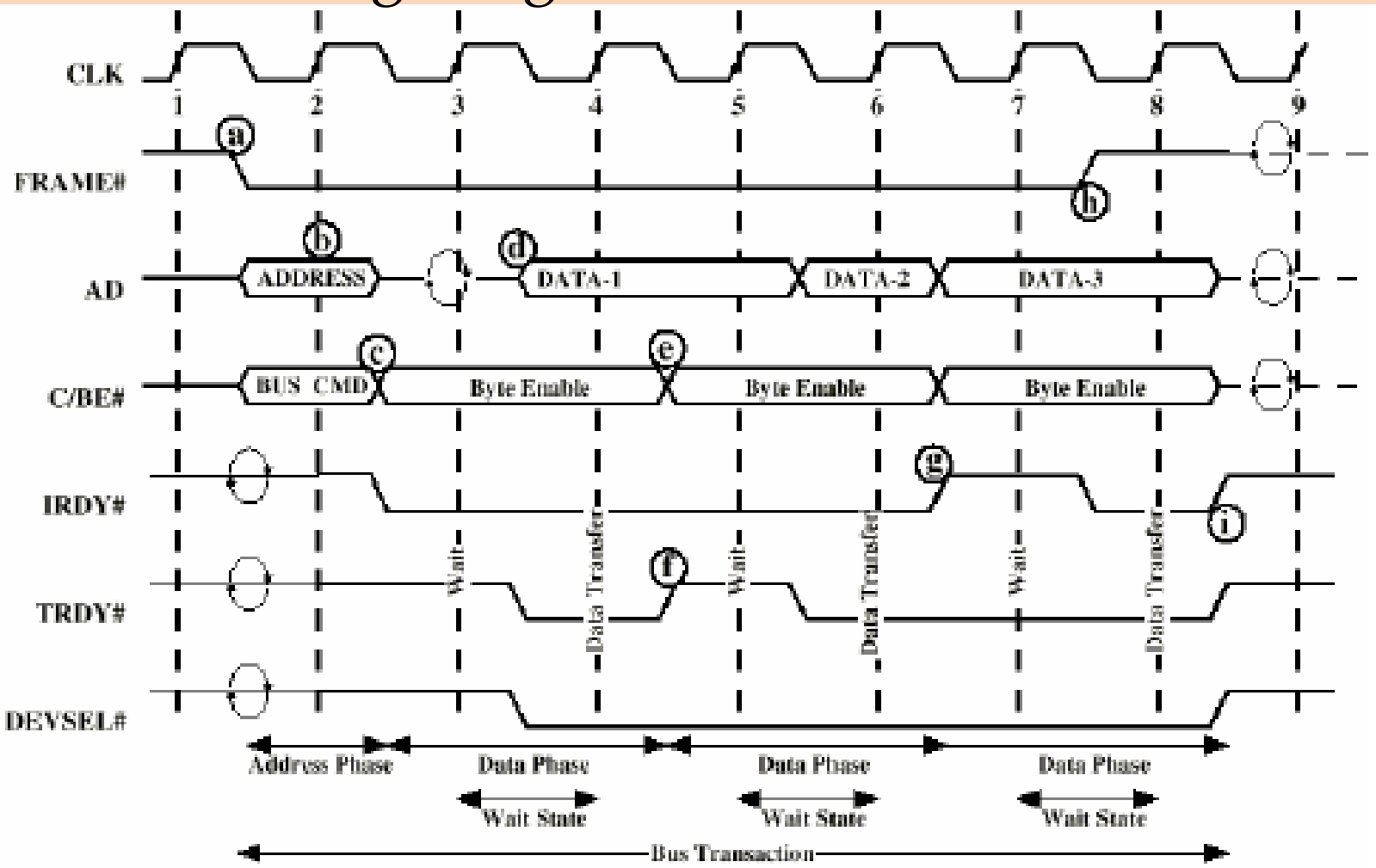
- Interrupt lines
  - Not shared
- Cache support
- 64-bit Bus Extension
  - Additional 32 lines
  - Time multiplexed
  - 2 lines to enable devices to agree to use 64-bit transfer
  - JTAG/Boundary Scan
  - For testing procedures

**PCI Commands**

- Transaction between initiator (master) and target
- Master claims bus
- Determine type of transaction
  - e.g. I/O read/write
- Address phase
- One or more data phases

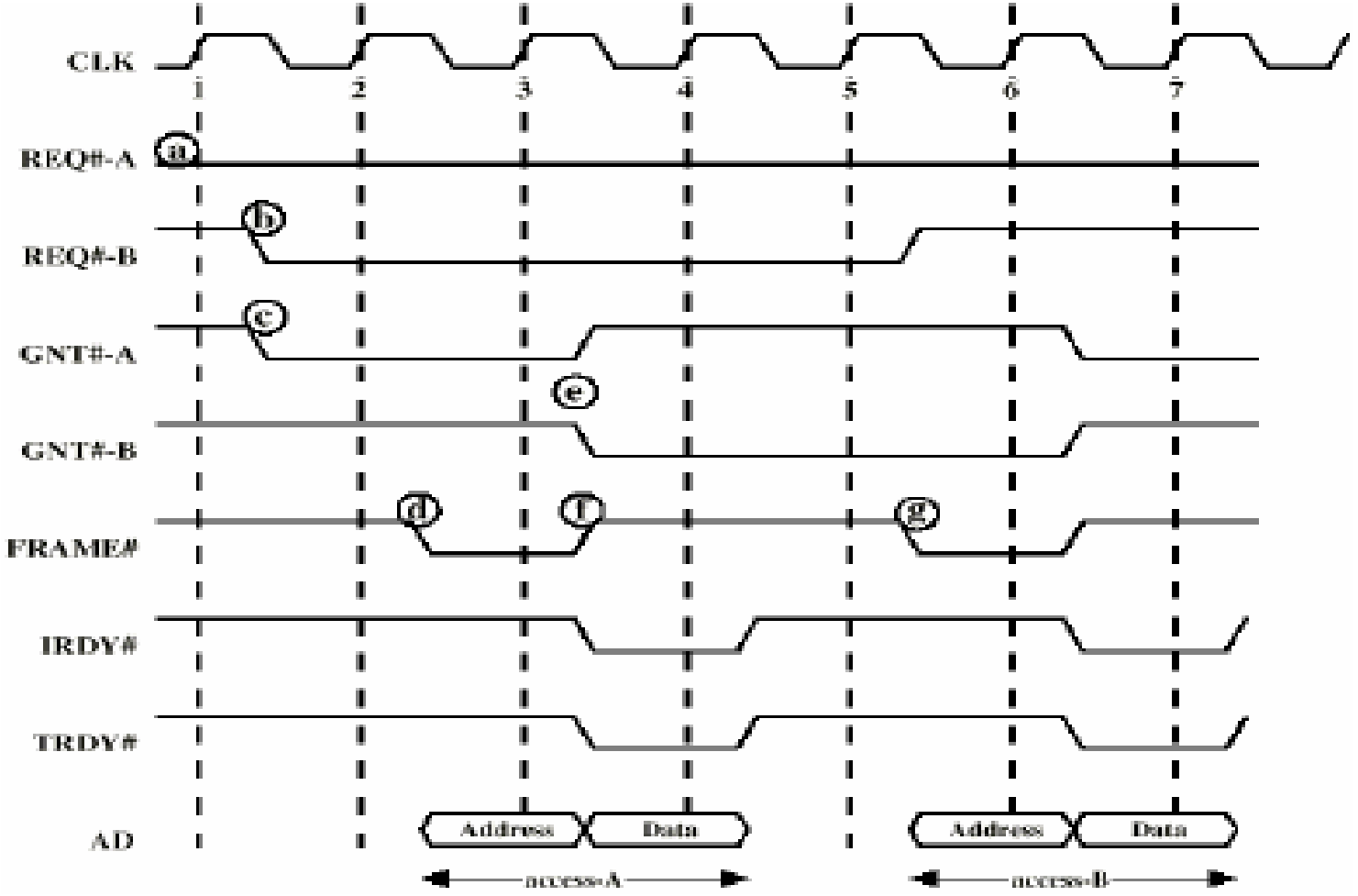
# Bio-Computer

## PCI Read Timing Diagram



# Bio-Computer

## PCI Bus Arbitration



# Bio-Computer

Software Paradigm

# Bio-Computer

- **Software Development Paradigm**; This paradigm is known as software engineering paradigms; where all the engineering concepts pertaining to the development of software are applied.
  - Requirement gathering
  - Software design
  - Programming
- **Software Design Paradigm;** is a part of Software Development and includes –
  - Design
  - Maintenance
  - Programming

# Bio-Computer

- **Programming Paradigm;** This paradigm is related closely to programming aspect of software development. This includes –
  - Coding
  - Testing
  - Integration

# Bio-Computer

## Need of Software Engineering

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working. Following are some of the needs stated

- Large software
- Scalability
- Cost
- Dynamic Nature
- Quality Management

# Bio-Computer

## Characteristics of good software

1. **Operational**; this tells us how well the software works in operations. It can be measured on:
   - Budget
   - Usability
   - Efficiency
   - Correctness
   - Functionality
   - Dependability
   - Security
   - Safety

# Bio-Computer

2. **Transitional**; this aspect is important when the software is moved from one platform to another:
   - Portability
   - Interoperability
   - Reusability
   - Adaptability
3. **Maintenance**; this aspect briefs about how well the software has the capabilities to maintain itself in the ever-changing environment:
   - Modularity
   - Maintainability
   - Flexibility
   - Scalability

# Bio-Computer

- **Software Development Life Cycle, SDLC;** is a well-defined, structured sequence of stages in software engineering to develop the intended software product.
- **SDLC Activities**

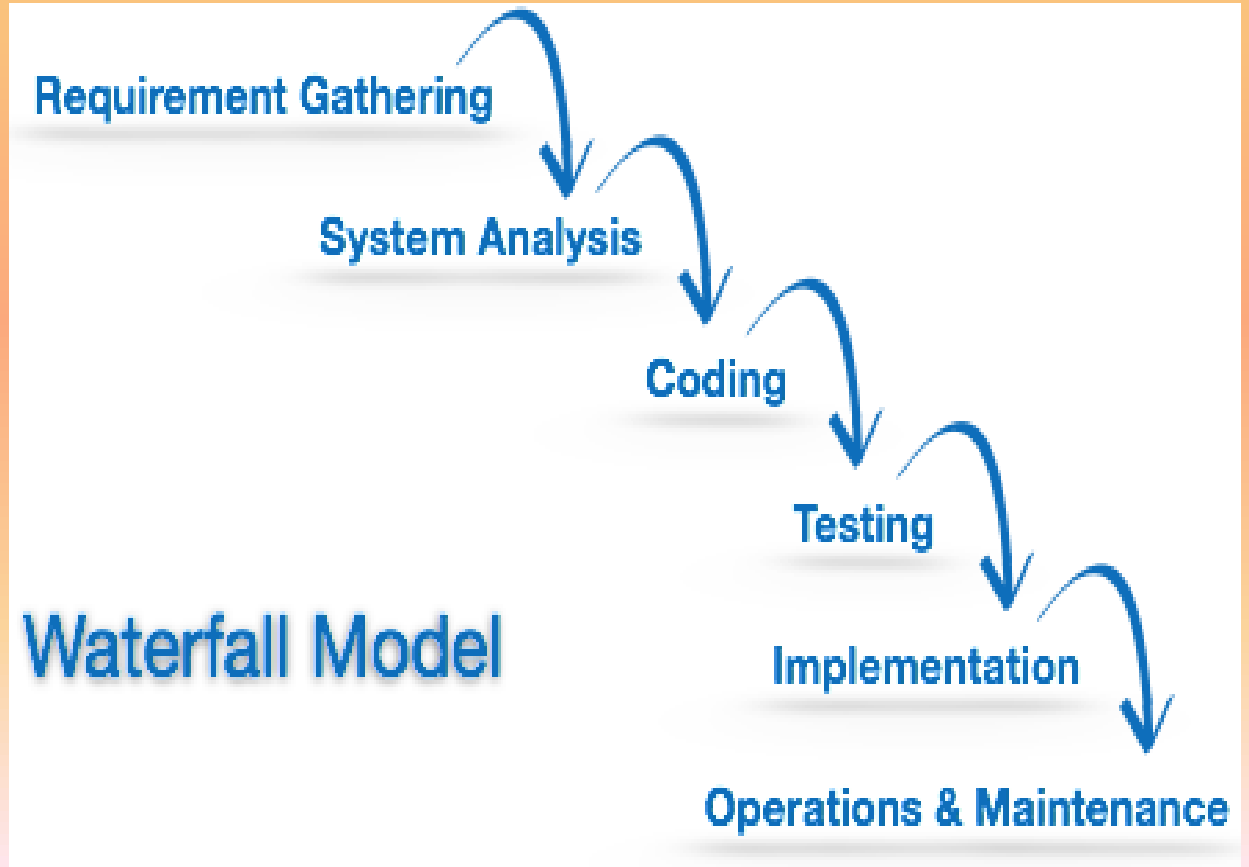
# Bio-Computer

# Bio-Computer

Software Development Paradigm; helps a developer to select a strategy to develop the software. A software development paradigm has its own set of tools, methods, and procedures, which are expressed clearly and defines software development life cycle. A few of software development paradigms or process models are defined as follows:

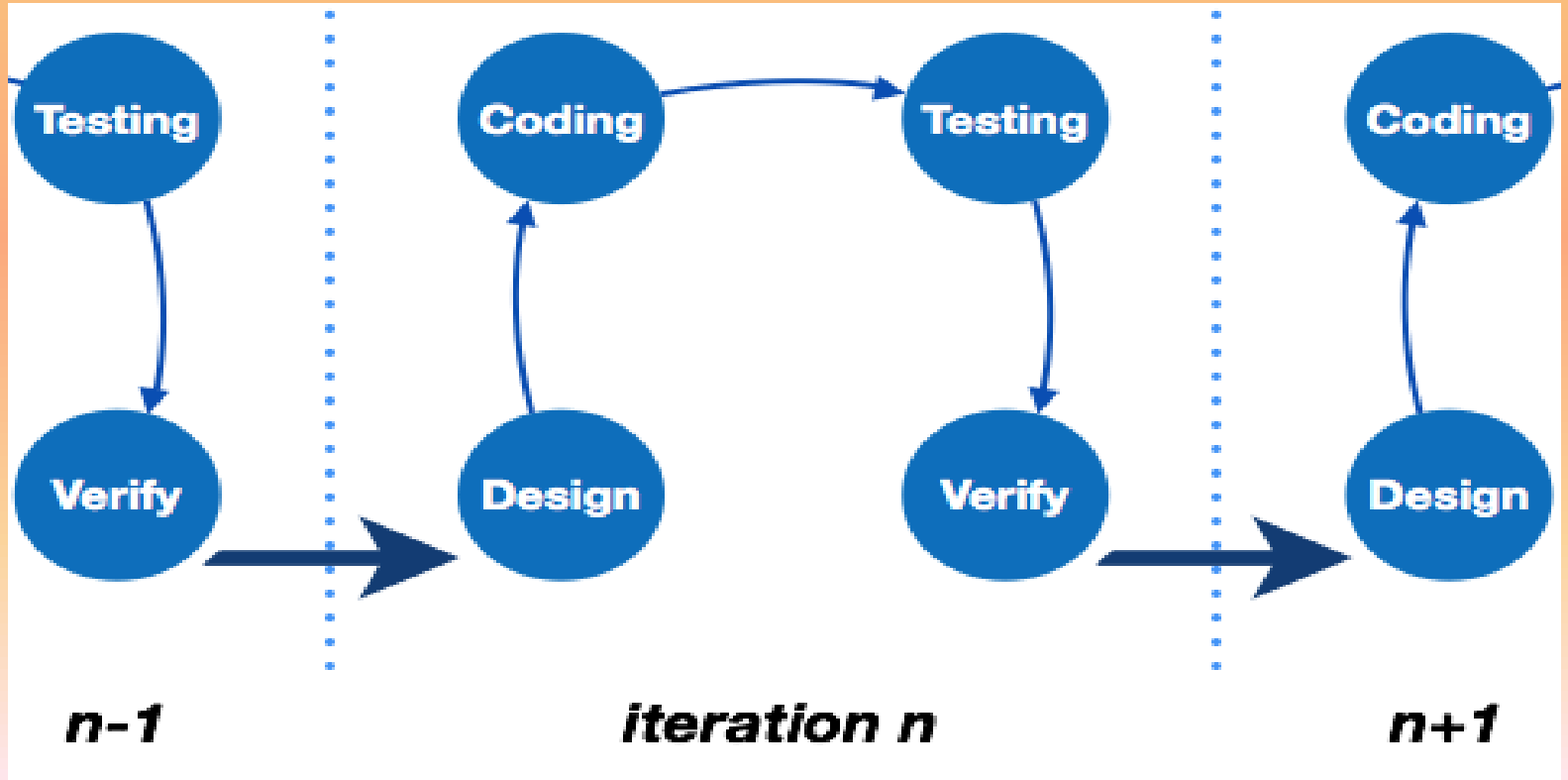
# Bio-Computer

**Waterfall Model**; is the simplest model of software development paradigm. All the phases of SDLC will function one after another in linear manner.



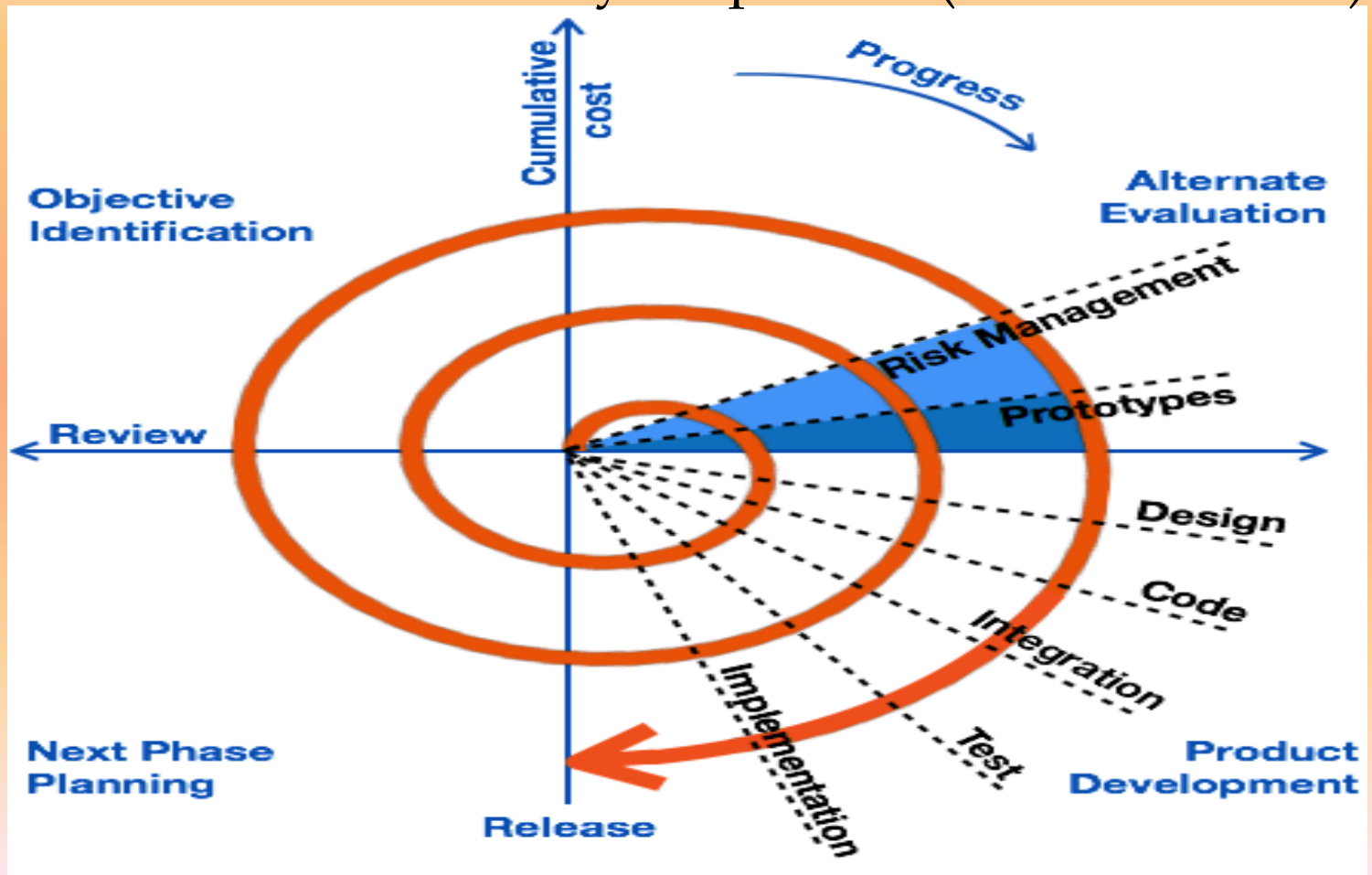
# Bio-Computer

**Iterative Model**; leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process



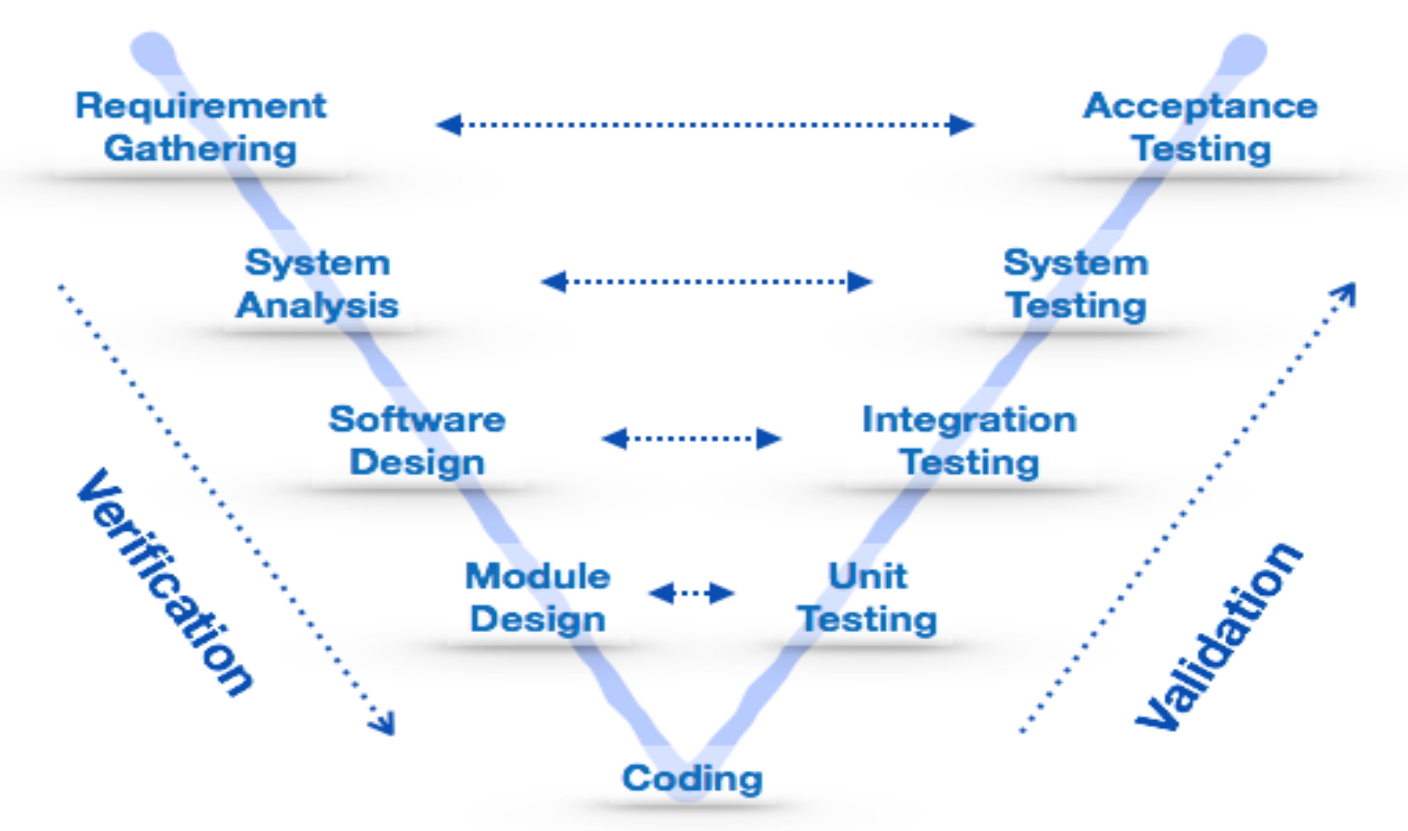
# Bio-Computer

**Spiral Model**; is a combination of both, iterative model and one of the SDLC model. It can be seen as if you choose one SDLC model and combined it with cyclic process (iterative model).



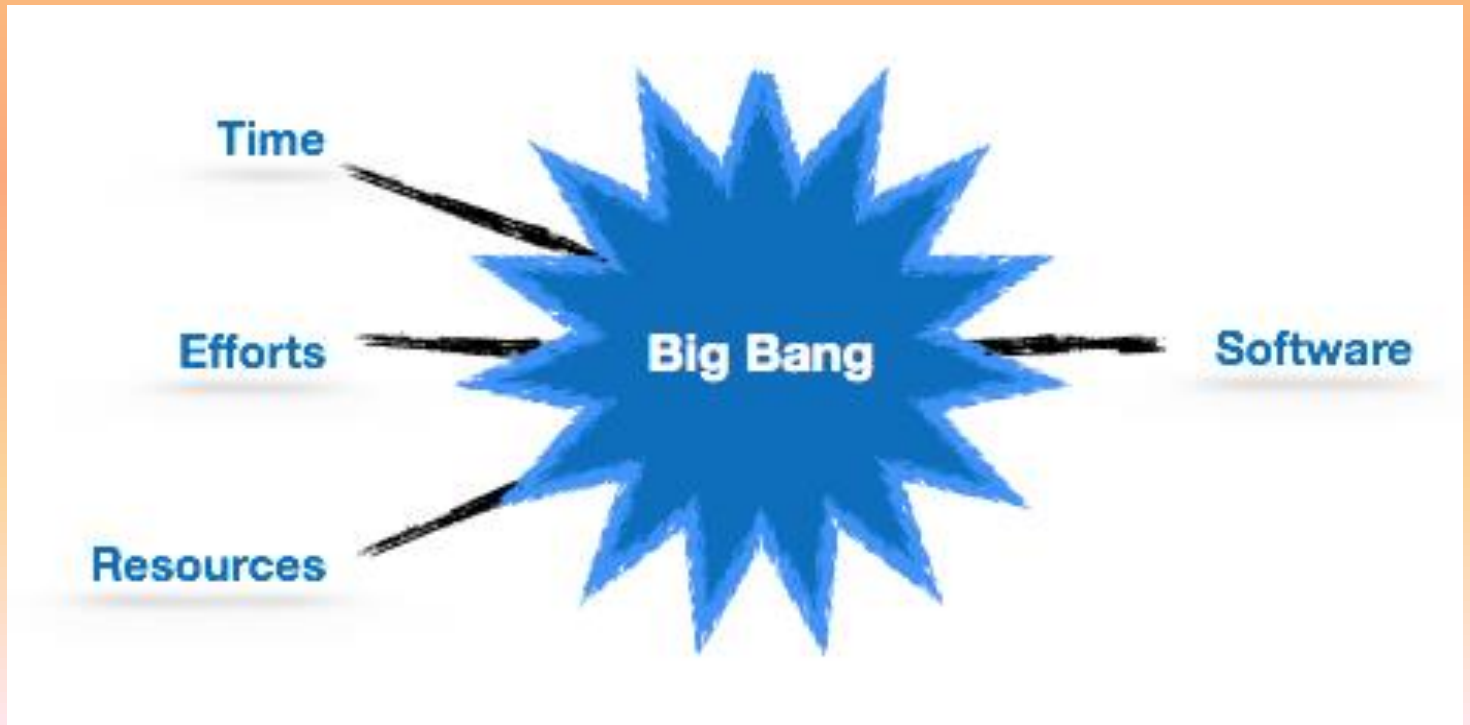
# Bio-Computer

**V – model**; the major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner



# Bio-Computer

**Big Bang Model**; is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets, and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product. This model is not suitable for large software projects but good one for learning and experimenting



# Bio-Computer

**Software Project Management**

Software development job can be splitted into two parts:

1- Software creation.

2- Software Project Management

A Project is well-defined task, which is a collection of several operations done in order to achieve a goal.

# Bio-Computer

Project can be characterized as:

1- Every project may have a unique and distinct goal.

2- Project is not a routine activity or day-to-day operation.

3- Project comes with a start and end time.

4- Project ends when its goal is achieved. Hence, it is a temporary phase in the life time of an organization.

5- Project needs adequate resources in terms of time, manpower, finance, material, and knowledge-bank.

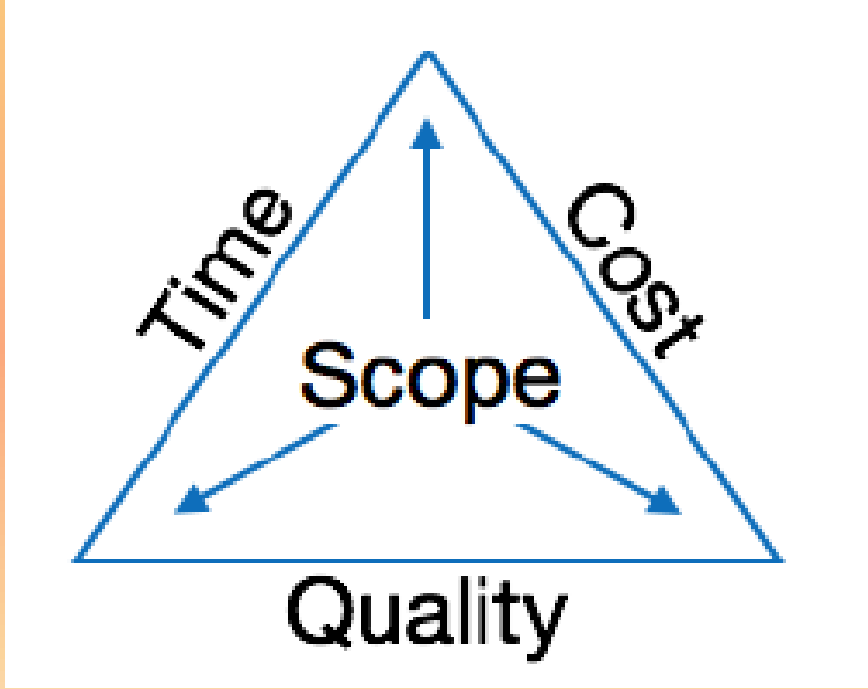
# Bio-Computer

**Software Project**: is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.

- Most software products are tailor made to fit client's requirements.
- The most important is that the underlying technology changes and advances so frequently and rapidly that the experience of one product may not be applied to the other one.

# Bio-Computer

- Triple constraints for software projects are shown below:



# Bio-Computer

- It is essential part of software organization

1- To deliver quality product,

2- Keeping the cost within client's budget constrain, and

3- Deliver the project as per scheduled.

- Any of the three factors can severely impact the other two.
- There are several factors, both internal and external, which may impact this triple constrain triangle.
- Software project management is essential to incorporate user requirements along with budget and time constraints.

# Bio-Computer

- **Software Project Manager** is a person who undertakes the responsibility of executing the software project. He may never directly involve in producing the end product but he controls and manages the activities involved in production.

# Bio-Computer

- **Below few responsibilities that he should do:**

**A) Managing People**

1- Act as project leader.
2- Lesion with stakeholders,
3- Managing human resources.
4- setting up reporting hierarchy etc.

**B) Managing Project**

1- Defining and setting up project scope.
2- Managing project management activities.
3- Monitoring progress and performance.
4- Risk analysis at very phase.
5- Take necessary step to avoid or come out of problems.
6- Act as project spokesperson.

# Bio-Computer

Software Management Activities may include:

1- Project Planning.

2- Scope Management.

3- Project Estimation.

- **Project Planning**: it is a task, which is performed before the production of software actually starts. It is a set of multiple processes, which facilitates software production.

# Bio-Computer

- **Scope Management** it includes all the activities, process need to be done in order to make a deliverable software product.
- It is essential because it creates boundaries of the project by clearly defining what would be done in the project and what would not be done.
- During Project Scope management, it is necessary to:

# Bio-Computer

1- Define the scope,

2- Decide its verification and control,

3- Divide the project into various smaller parts for ease of management,

4- Verify the scope,

5- Control the scope by incorporating changes to the scope.

# Bio-Computer

- **Project Estimation** determines an accurate estimation of various measures.
- It may involve the following:

1- **Software size estimation**; may be estimated either in terms of Kilo Line of code (KLOC) or by calculating number of function points in the software.

- Line of code depend upon coding practices.
- Function points vary according to the user or software requirement.

# Bio-Computer

2- **Effort estimation**; is in terms of personnel requirement and man-hour required to produce the software.

- For effort estimation software size should be known.
- It is derived by manager's experience, historical data of organization, or software size can be converted into efforts by using some standard formulae.

# Bio-Computer

3- **Time estimation;** once the size and efforts are estimated, the time required to produce the software can be estimated.

- Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure (WBS).
- The tasks are scheduled on day-to-day basis or in calendar months.
- The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.

# Bio-Computer

4- **Cost estimation;** is considered as the most difficult of all because it depends on more elements than any of the previous ones. It is required to consider:

- Size of the software,
- Software quality,
- Additional software or tools, licenses etc,
- Skilled personnel with task-specific skills,
- Travel involved,
- Communication,
- Training and support.

# Bio-Computer

**Project Estimation Techniques**

- Project estimation involved various parameters such as size, efforts, and cost.
- Project manager can estimate the listed factors using two broadly recognized techniques:

1- **Composition Technique**; assumes the software as a product of various compositions. There are two models:

- **Line of code**: the estimation is done on behalf of the number of line codes in the software product.
- **Function Points**: the estimation is done on behalf of number of function points in the software product.

# Bio-Computer

2- **Empirical Estimation Technique**; uses empirically derived formulae to make estimation.

- **Putnam Model**: is made by Lawrence H. Putnam. It maps time and efforts required with software size.
- **COCOMO**: stands for Constructive Cost Model, developed by Barry W. Boehm. It divides the software product into three categories of software: organic, semi-detached, and embedded.

# Bio-Computer

**Project Scheduling**: refers to roadmap of all activities to be done with specified order and within time slot allocated to each activity.

- Project managers trend to define various tasks, and project milestones and then arrange them keeping various factors in mind.
- For scheduling a project, it is necessary to:

# Bio-Computer

1- Break down the project tasks into smaller, manageable form,

2- Find out various tasks and correlate them,

3- Estimate time frame required for each task,

4- Divide time into work-units,

5-Assign adequate number of work- units for each task,

6- Calculate total time required for the project from start to finish.

# Bio-Computer

**Resource Management:** all elements used to develop a software product may be assumed as resource for that project.

- This may include human resource, productive tools, and software libraries.
- The shortage of resources hampers development of the project and it can lag behind the schedule.
- Allocating extra resources increases development cost in the end.

# Bio-Computer

- Resource management includes:

1- Defining proper organization project by creating a project team and allocating responsibilities to each team member.

2- Determining resources required at a particular stage and their availability.

3- Manage resources by generating resource request when they are required and de-allocating them when they are no more needed.

# Bio-Computer

**Project Risk Management**; involves all activities pertaining to identification, analyzing and making provision for predictable and non-predictable risks in the project. It may include:-

1- Experienced staff leaving the project and new staff coming in.

2- Change in organizational management.

3- Requirement change or misinterpreting requirement.

4- under-estimation of required time and resources.

5- Technological changes, environmental changes, business competition.

# Bio-Computer

- **Risk Management Process**; involves the following activities:

1- **Identification**- make note of all possible risks, which may occur in the project.

2- **Categorize**- known risks into high, medium and low risk intensity as per their possible impact on the project.

3- **Manage**- Analyze the probability of occurrence of risks at varies phases. Make plan to avoid or face risks. Attempt to minimize their side effects.

4- **Monitor**- Closely monitor the potential risks and their early symptoms. Also monitor the effective steps taken to mitigate or avoid them.

# Bio-Computer

**Project Execution and Monitoring**; are executing the tasks described in project according to their schedules.

- Execution needs monitoring in order to check whether everything is going according to the plan.
- These measures include:

1- **Activity Monitoring-** All activities scheduled within some task can be monitored on day-to-day basis. When all activities in task are completed, it is considered as complete.

2- **Status Reports-** The reports contain status of activities and tasks completed within a given time frame, generally a week. Status can be marked as Finished, Pending or Work-in-Progress etc.

3- **Milestones Checklist-** Every project is divided into multiple phases where major tasks are performed (milestones) based on the phases of SDLC. This milestone checklist is prepared once every few weeks and reports the status of milestones.

# Bio-Computer

**Project Communication Management**;

- plays a vital role in the success of a project.
- It bridges gaps between client and organization, among the team members as well as other stake holders in the project such as hardware suppliers.
- Communication can be oral or written.

# Bio-Computer

- Communication management process may have the following steps:

1- **Planning-** This step includes the identifications of all the stakeholders in the project and the mode of communication among them. It also considers if any additional communication facilities are required.

2- **Sharing-** manager focuses on sharing correct information with the correct person at the correct time. This keeps every one involved in the project up-to-date with project progress and its status.

3- **Closure-** At the end of each major event, end of a phase of SDLC or end of the project itself, administrative closure is formally announced to update every stakeholder by sending email, by distributing a hardcopy of document or by other mean of effective communication.

- After closure, the team moves to next phase or project.

# Bio-Computer

Configuration Management; is a process of tracking and controlling the changes in software in terms of the requirements, design, functions and development of the product.

- IEEE defines it as “the process of identifying and defining the items in the system, controlling the change of these items throughout their life cycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items”.
- The changes are addressed only with prior approval of higher management, as there is a possibility of cost and time overrun.

# Bio-Computer

Configuration Management

**1-Baseline**; A phase is baselined when all activities pertaining to it are finished and well documented. If it was not the final phase, its output would be used in next immediate phase.

- Configuration management is a discipline of organization administration, which takes care of occurrence of any changes (process, requirement, technological, strategical etc.) after a phase is baselined. CM keeps check on any changes done in software.

# Bio-Computer

Configuration Management

2- **Change Control**; is function of configuration management, which ensures that all changes made to software system are consistent and made as per organizational rules and regulations.

A change in the configuration of product goes through following steps –

i- **Identification -** A change request arrives from either internal or external source. When change request is identified formally, it is properly documented.

# Bio-Computer

Configuration Management

ii- **Validation -** Validity of the change request is checked and its handling procedure is confirmed.

iii- **Analysis -** The impact of change request is analyzed in terms of schedule, cost and required efforts. Overall impact of the prospective change on system is analyzed.

# Bio-Computer

Configuration Management

iv- **Control -** If the prospective change either impacts too many entities in the system or it is unavoidable, it is mandatory to take approval of high authorities before change is incorporated into the system. It is decided if the change is worth incorporation or not. If it is not, change request is refused formally.

# Bio-Computer

Configuration Management

v- **Execution -** If the previous phase determines to execute the change request, this phase takes appropriate actions to execute the change, through a thorough revision if necessary.

vi- **Close request** - The change is verified for correct implementation and merging with the rest of the system. This newly incorporated change in the software is documented properly and the request is formally closed.

# Bio-Computer

Project Management Tools

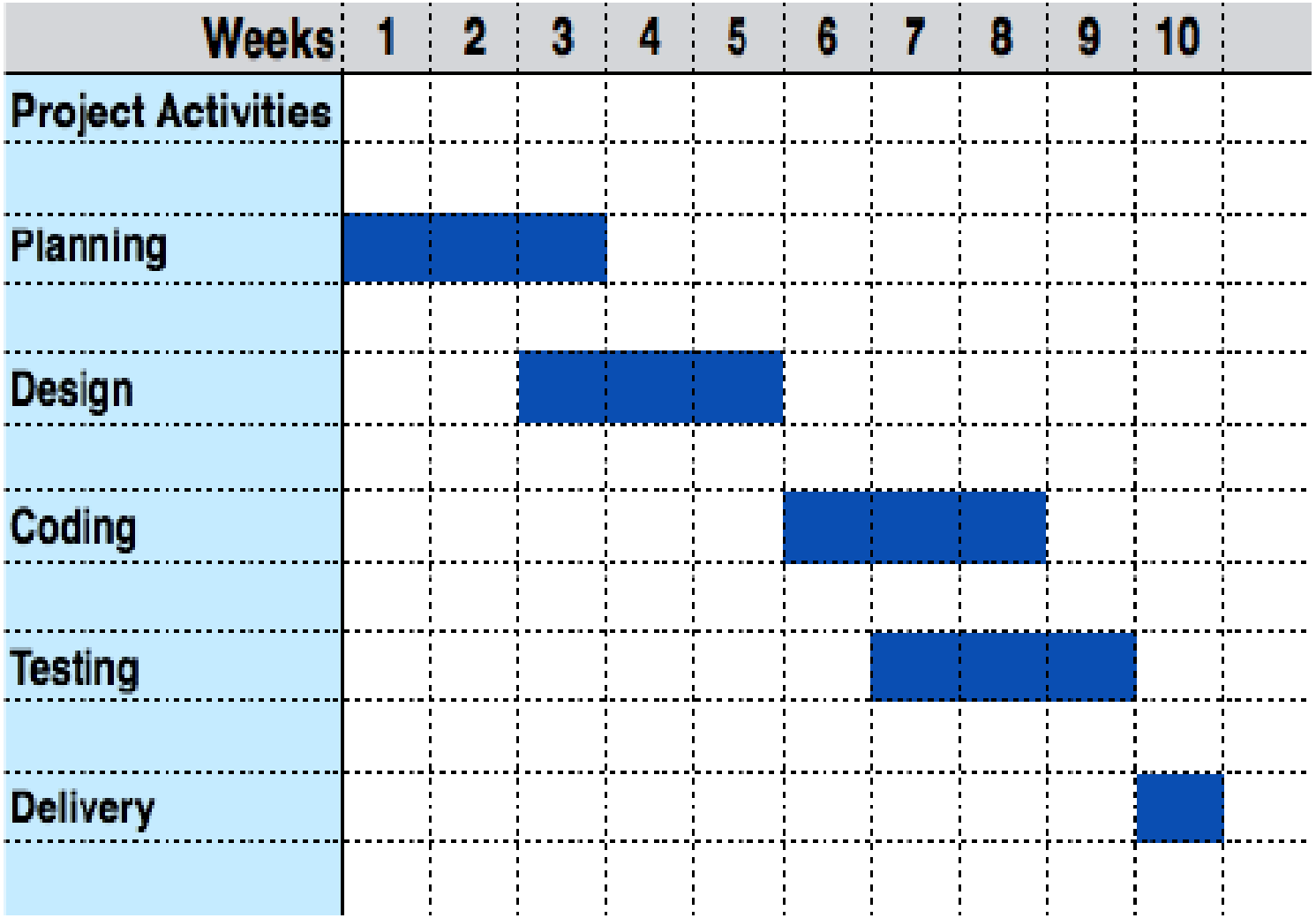
- The risk and uncertainty rises multifold with respect to the size of the project, even when the project is developed according to set methodologies.
- There are tools available, which aid for effective project management. A few described are:-

# Bio-Computer

Project Management Tools

1- **Gantt Chart**; Gantt chart was devised by Henry Gantt (1917). It represents project schedule with respect to time periods. It is a horizontal bar chart with bars representing activities and time scheduled for the project activities.

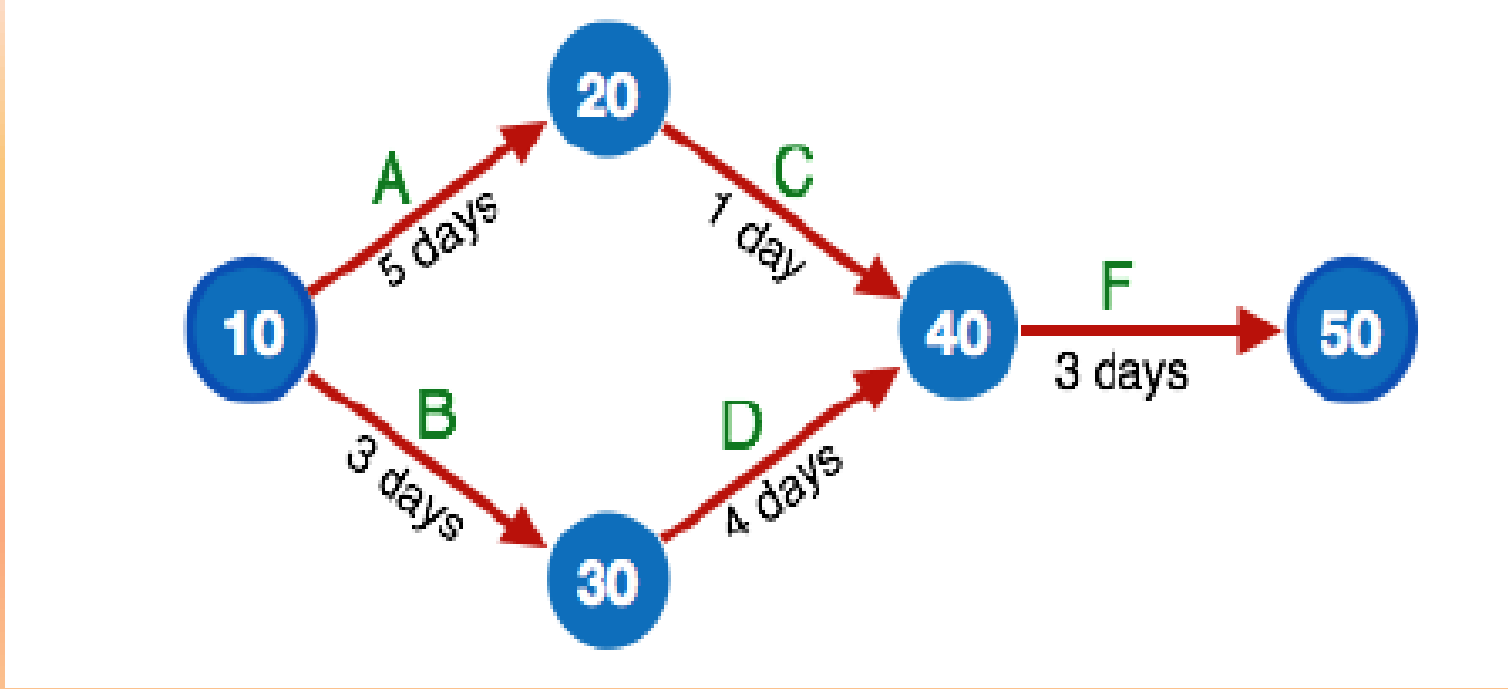
# Bio-Computer



# Bio-Computer

2- **PERT Chart**; Program Evaluation & Review Technique) (PERT) chart is a tool that depicts project as network diagram. It is capable of graphically representing main events of project in both parallel and consecutive ways. Events, which occur one after another, show dependency of the later event over the previous one.

# Bio-Computer



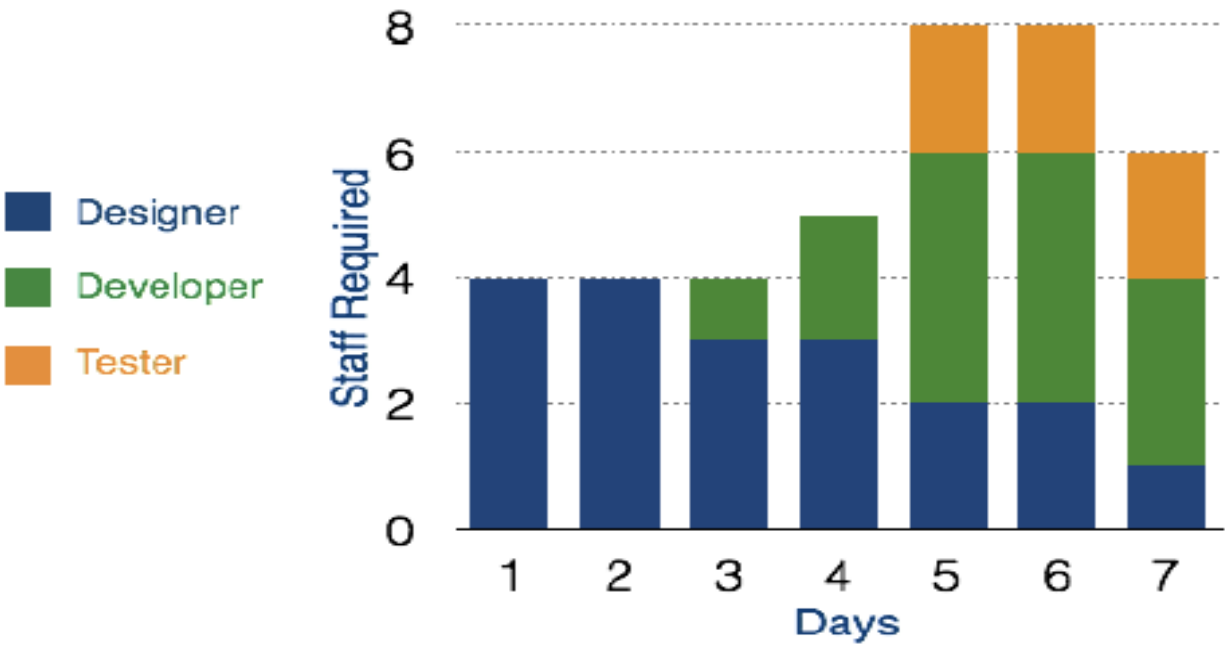

Events are shown as numbered nodes. They are connected by labeled arrows depicting the sequence of tasks in the project.

# Bio-Computer

3- **Resource Histogram**; this is a graphical tool that contains bar or chart representing number of resources (usually skilled staff) required over time for a project event (or phase). Resource Histogram is an effective tool for staff planning and coordination.

# Bio-Computer

| Staff | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 |
|---|---|---|---|---|---|---|---|
| Designer | 4 | 4 | 3 | 3 | 2 | 2 | 1 |
| Developer | 0 | 0 | 1 | 2 | 4 | 4 | 3 |
| Tester | 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| Total | 4 | 4 | 4 | 5 | 8 | 8 | 6 |



# Bio-Computer

4- **Critical Path Analysis**; this tools is useful in recognizing interdependent tasks in the project. It also helps to find out the shortest path or critical path to complete the project successfully. Like PERT diagram, each event is allotted a specific time frame. This tool shows dependency of event assuming an event can proceed to next only if the previous one is completed.

- The events are arranged according to their earliest possible start time. Path between start and end node is critical path which cannot be further reduced and all events require to be executed in same order.