# Computer Architecture and Organization

## Introduction

## Introduction

- In this Module, we have three lectures, viz.
- 1. Introduction to computer System and its sub modules.
- 2. Number System and Representation of information.
- 3. Brief History of Comp. Evolution

## Representation of Basic Information

- The basic functional units of computer are made of electronics circuit and it works with electrical signal.
- We provide input to the computer in form of electrical signal and get the output in form of electrical signal.
- There are two basic types of electrical signals, namely, **analog and digital.**
- The analog signals are continuous in nature and digital signals are discrete in nature.

# Representation of Basic Information

- Computer is a digital device, which works on two levels of signal.
- **LOW (L) - will represent 0 V and**
- **HIGH (H) - will represent 5V**
- **0 means LOW**
- **1 means HIGH**
- **Binary number system**
- Four bits together is known as **Nibble, and Eight bits together is known as Byte.**

## Introduction :: Computer Organization and Architecture

- Computer technology has made incredible improvement in the past half century
- The task that the computer designer handles is a complex one:
- Determine what attributes are important for a new machine, then design a machine to maximize performance while staying within cost constraints.
- This task has many aspects, including instruction set design, functional organization, logic design, and implementation.
- While looking for the task for computer design, both the terms computer organization and computer architecture come into picture.

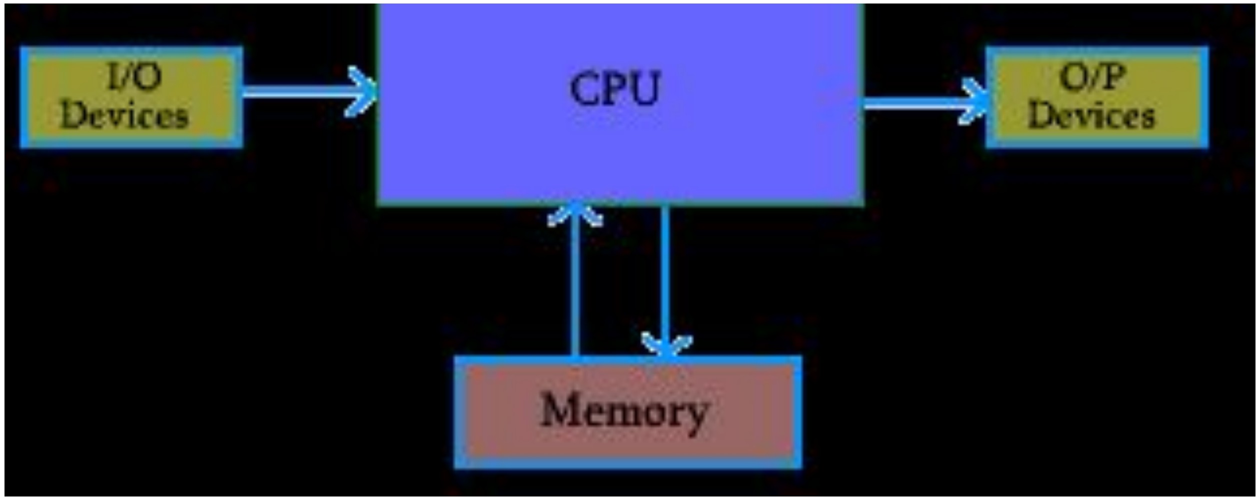
## Computer architecture VS Computer organization

- Computer architecture refers to those parameters of a computer system that are visible to a programmer or those parameters that have a direct impact on the logical execution of a program. Examples of architectural attributes include the instruction set, the number of bits used to represent different data types, I/O mechanisms, and techniques for addressing memory.
- Computer organization refers to the operational units and their interconnections that realize the architectural specifications. Examples of organizational attributes include those hardware details transparent to the programmer, such as control signals, interfaces between the computer and peripherals, and the memory technology used.

## Basic Computer Model and different units of Computer

- The model of a computer can be described by four basic units in high level abstraction. These basic units are:
- Central Processor Unit
- Input Unit
- Output Unit
- Memory Unit

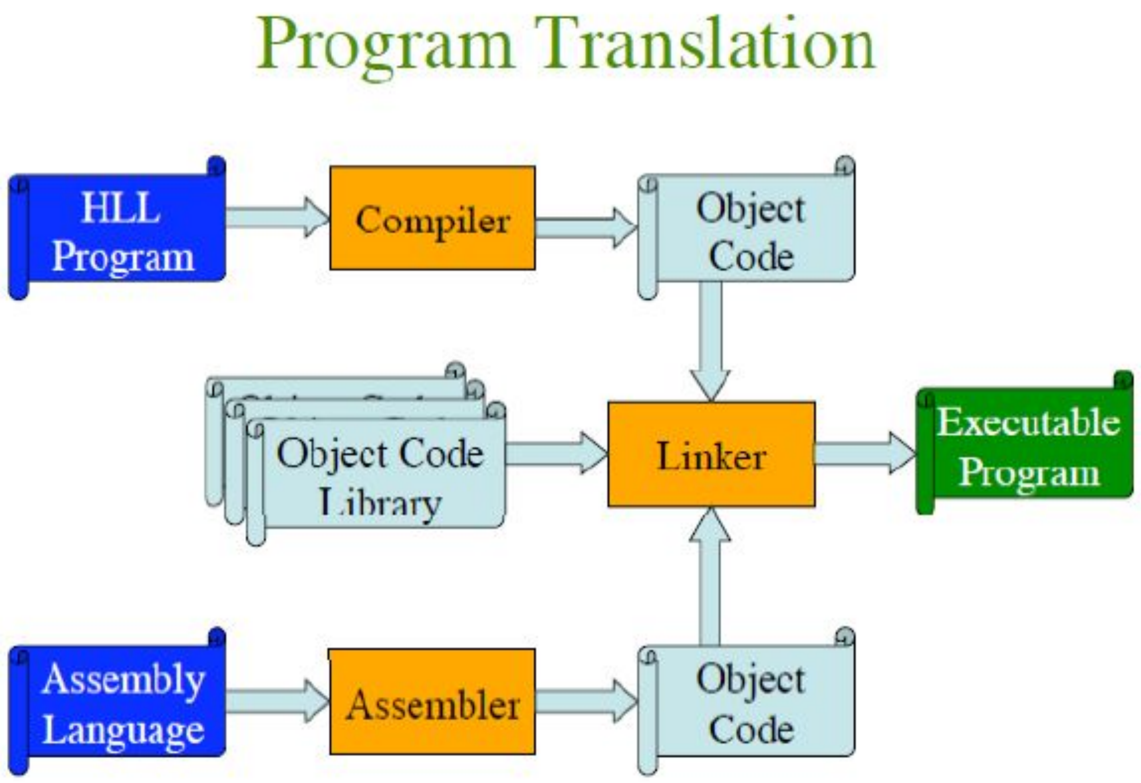
# Basic Computer Model and different units of Computer



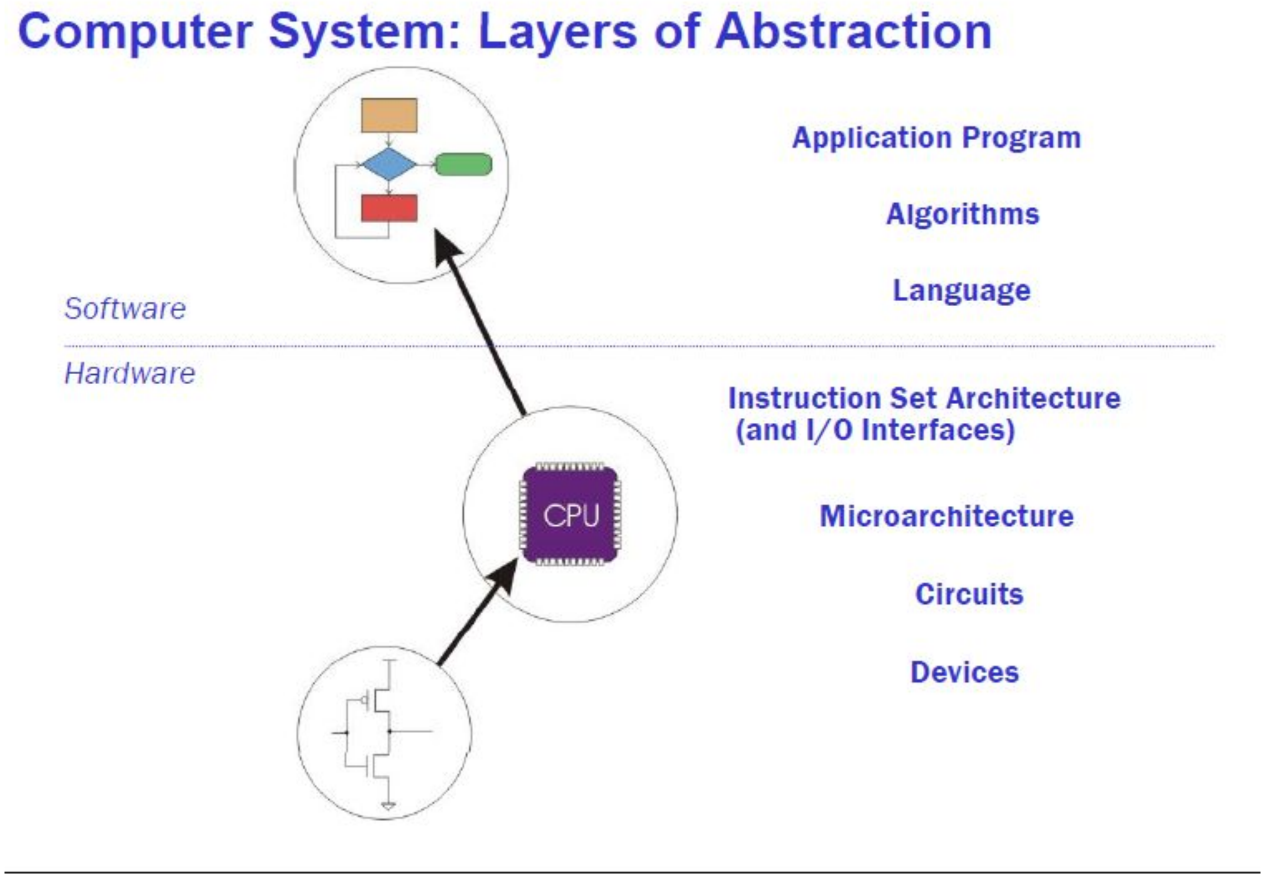
## Component of computers

- Hardware:
- Four major Blocks:
- CU,MU,ALU,I/O Unit
- CU and ALU both combined form CPU.
- Software :
- OS
- Machine Language
- Assembly Language
- High level Language
- Software can be classified into two catagories:
- System Software
- Application Software

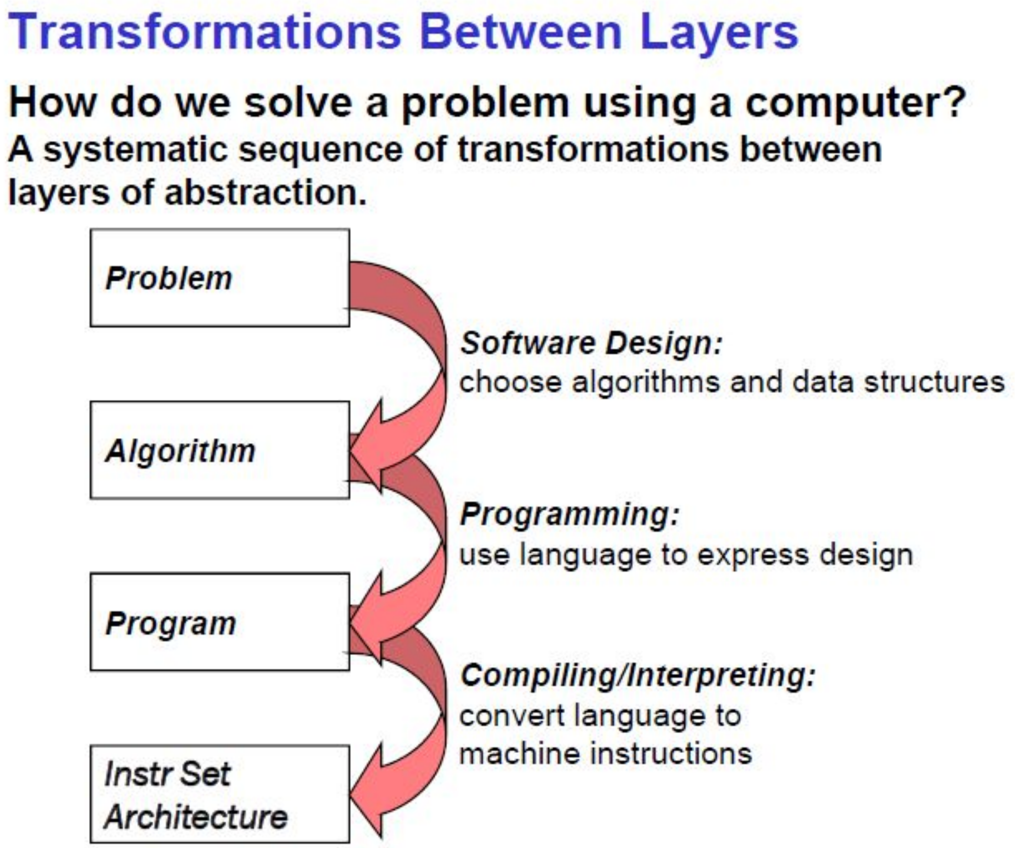
## Program Translation



# Layers of Abstraction



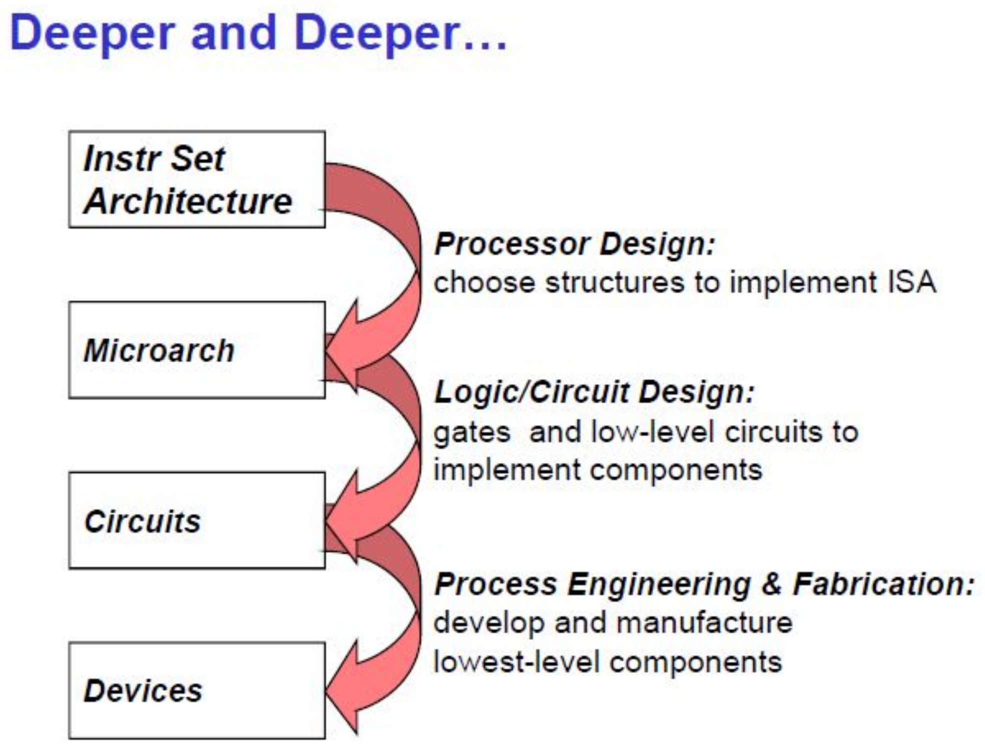
# Transformations Between Layers



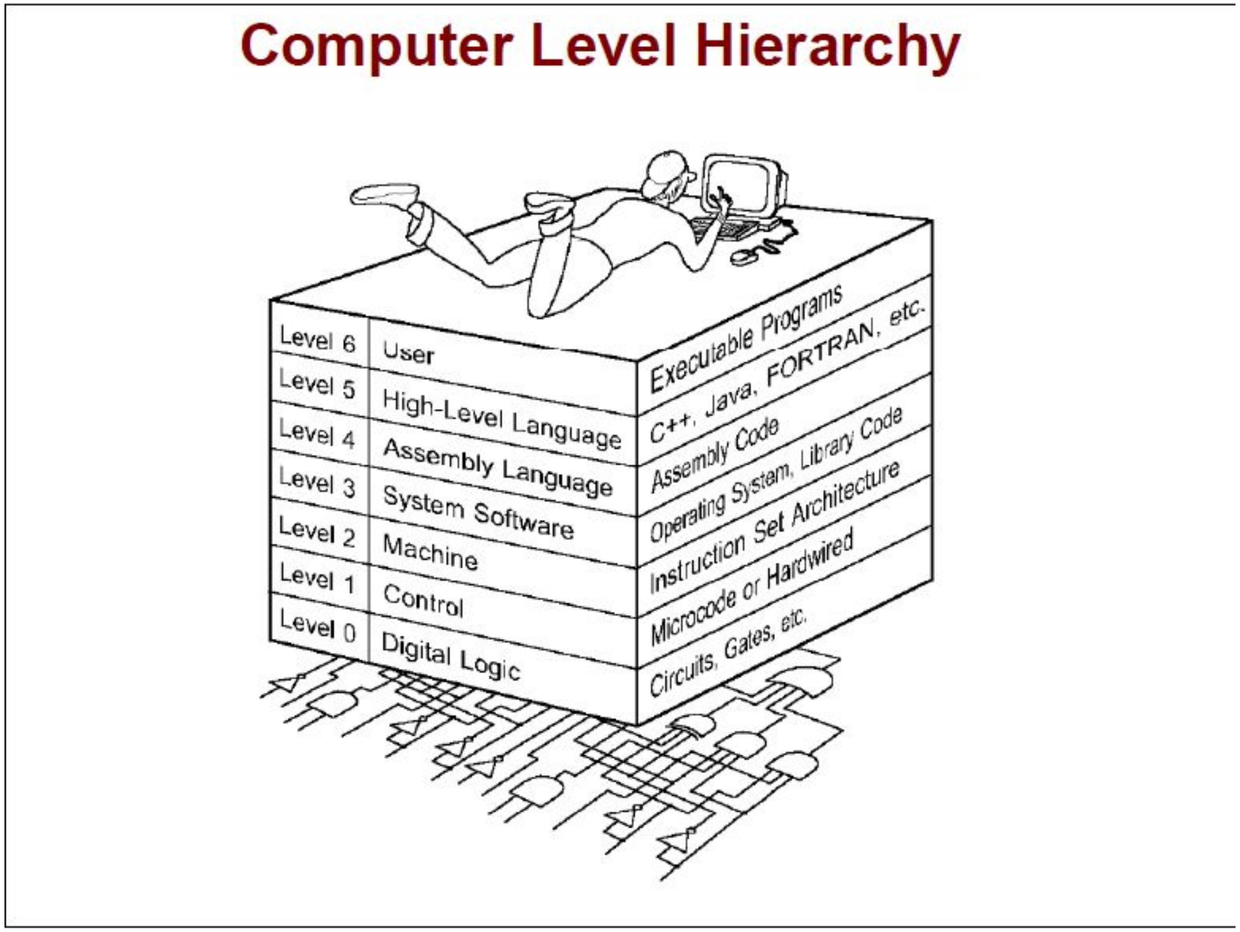
## Transformations Between Layers

**How do we solve a problem using a computer?**
**A systematic sequence of transformations between layers of abstraction.**

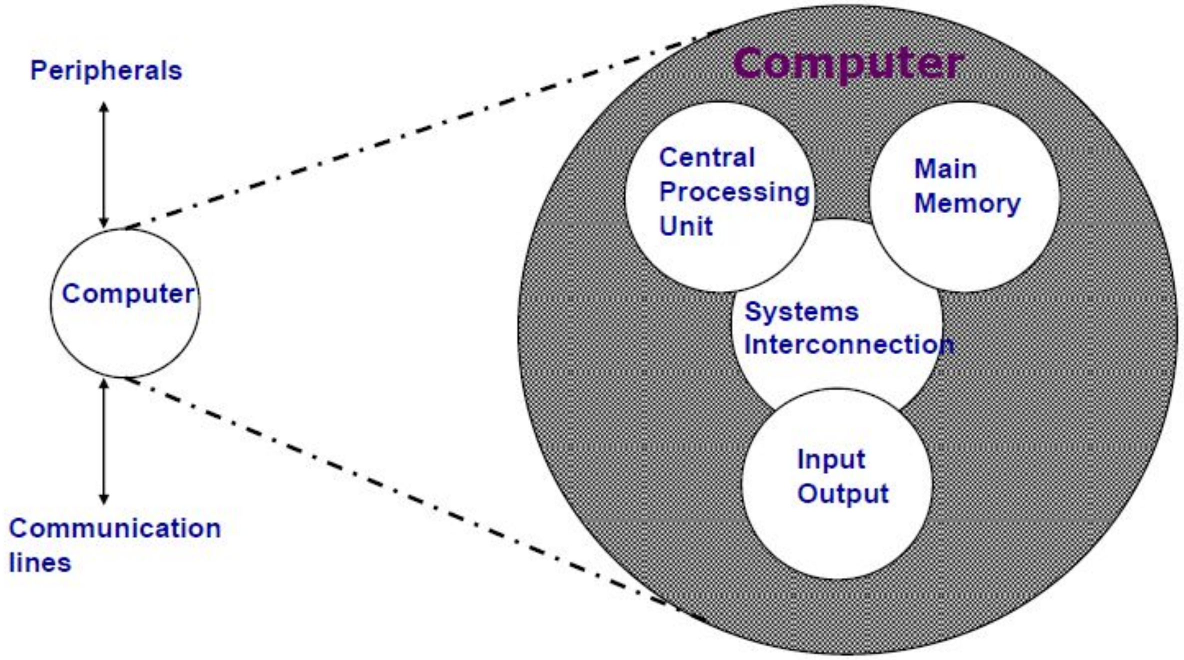
# Transformations Between Layers


Deeper and Deeper…
Instr Set Architecture
Microarch
Circuits
Devices
Processor Design:
choose structures to implement ISA
Logic/Circuit Design:
gates and low-level circuits to implement components
Process Engineering & Fabrication:
develop and manufacture lowest-level components

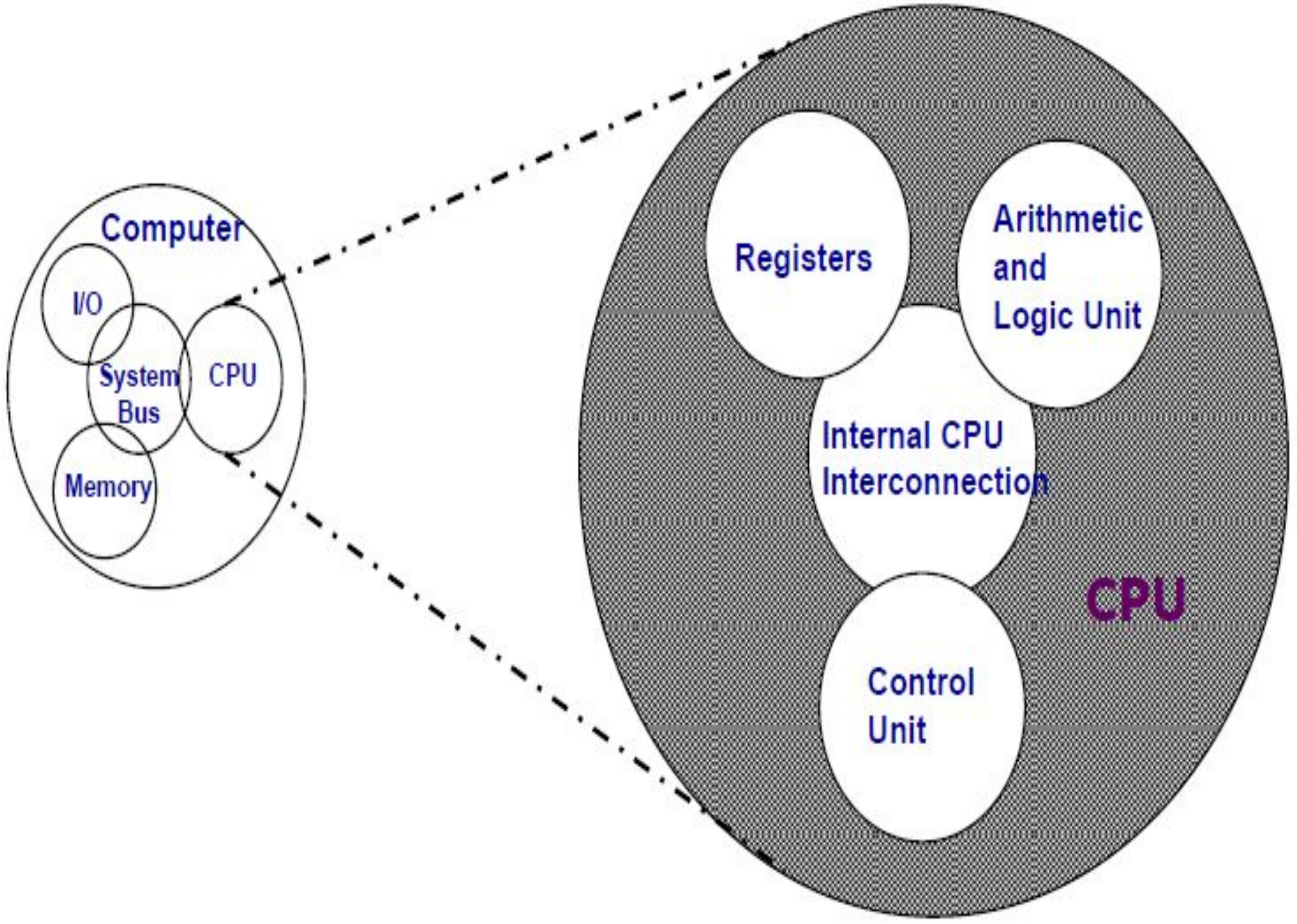
## Computer Level Hierarchy


Computer Level Hierarchy
Level 6 User Executable Programs
Level 5 High-Level Language C++, Java, FORTRAN, etc.
Level 4 Assembly Language Assembly Code
Level 3 System Software Operating System, Library Code
Level 2 Machine Instruction Set Architecture
Level 1 Control Microcode or Hardwired
Level 0 Digital Logic Circuits, Gates, etc.

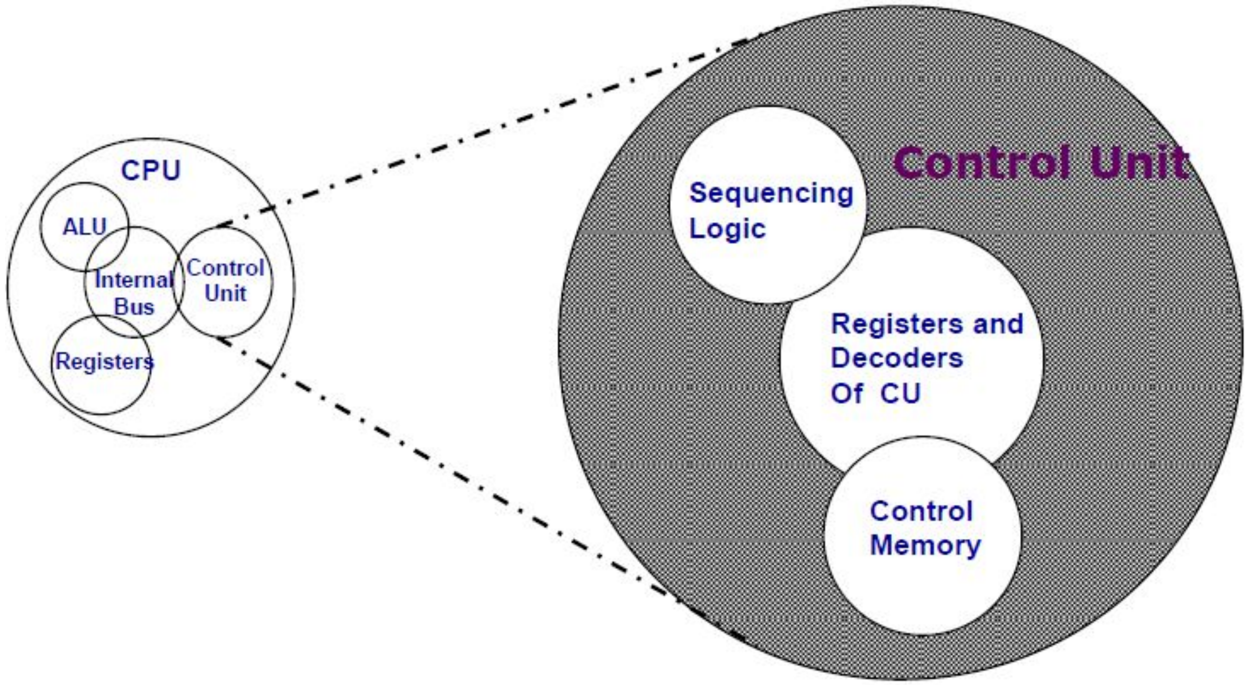
# Structure - Top Level



# Structure - The CPU



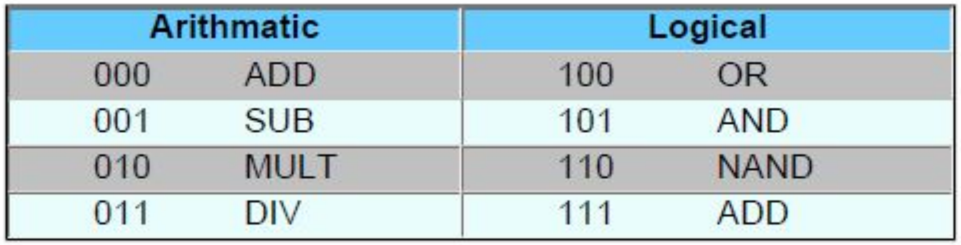
## Structure - The Control Unit



## Basic Working Principle of a Computer

- **Consider the Arithmatic and Logic Unit (ALU) of Central Processing Unit**
- 0 - in that signal, represents an arithmetic operation and 1 - in that signal, represents a logical operation.

## Basic Working Principle of a Computer



| Arithmatic | | Logical | |
|---|---|---|---|
| 000 | ADD | 100 | OR |
| 001 | SUB | 101 | AND |
| 010 | MULT | 110 | NAND |
| 011 | DIV | 111 | ADD |

# Introduction to Computer System

- Consider the part of control unit, its task is to generate the appropriate signal at right moment.
- There is an instruction decoder in CPU which decodes this information in such a way that computer can perform the desired task
- The simple model for the decoder may be considered that there is three input lines to the decoder and correspondingly it generates eight output lines.

## Introduction to Computer System

- In our simple model, we use three storage units in CPU,
- Two -- for storing the operand and
- one -- for storing the results.

These storage units are known as register.

## Introduction to Computer System

- **The CPU can work with the information available in main memory only.**
- To access the data from memory, we need two special registers one is known as **Memory Data Register (MDR)** and the second one is **Memory Address Register (MAR).**

## Main Memory Organization

- The capacity of a memory module is specified by the number of memory location and the information stored in each location.
- A memory module of capacity 16 X 4 indicates that, there are 16 location in the memory module and in each location, we can store 4 bit of information.
- We need two operation to work with memory.
- **READ Operation: This operation is to retrieve the data from memory and bring it to CPU register**
- **WRITE Operation: This operation is to store the data to a memory location from CPU register**

## Introduction to Computer System

- To transfer the data from CPU to memory module and vice-versa, we need some connection. This is termed as **DATA BUS.**
- These signal lines use to identify a memory location is termed as **ADDRESS BUS.**
- **Size of address bus depends on the memory size. For a memory module of capacity of 2*n*** location, we need n address lines, that is, an address bus of size n.

## Introduction to Computer System

- We use a address decoder to decode the address that are present in address bus
- As for example, consider a memory module of 16 location and each location can store 4 bit of information
- The size of address bus is 4 bit and the size of the data bus is 4 bit
- The size of address decoder is 4 X 16.

## Introduction to Computer System

- If the contents of address bus is 0101 and contents of data bus is 1100 and R/W = 1, then 1100 will be written in location 5.
- If the contents of address bus is 1011 and R/W=0, then the contents of location 1011 will be placed in data bus.

# Memory Instruction

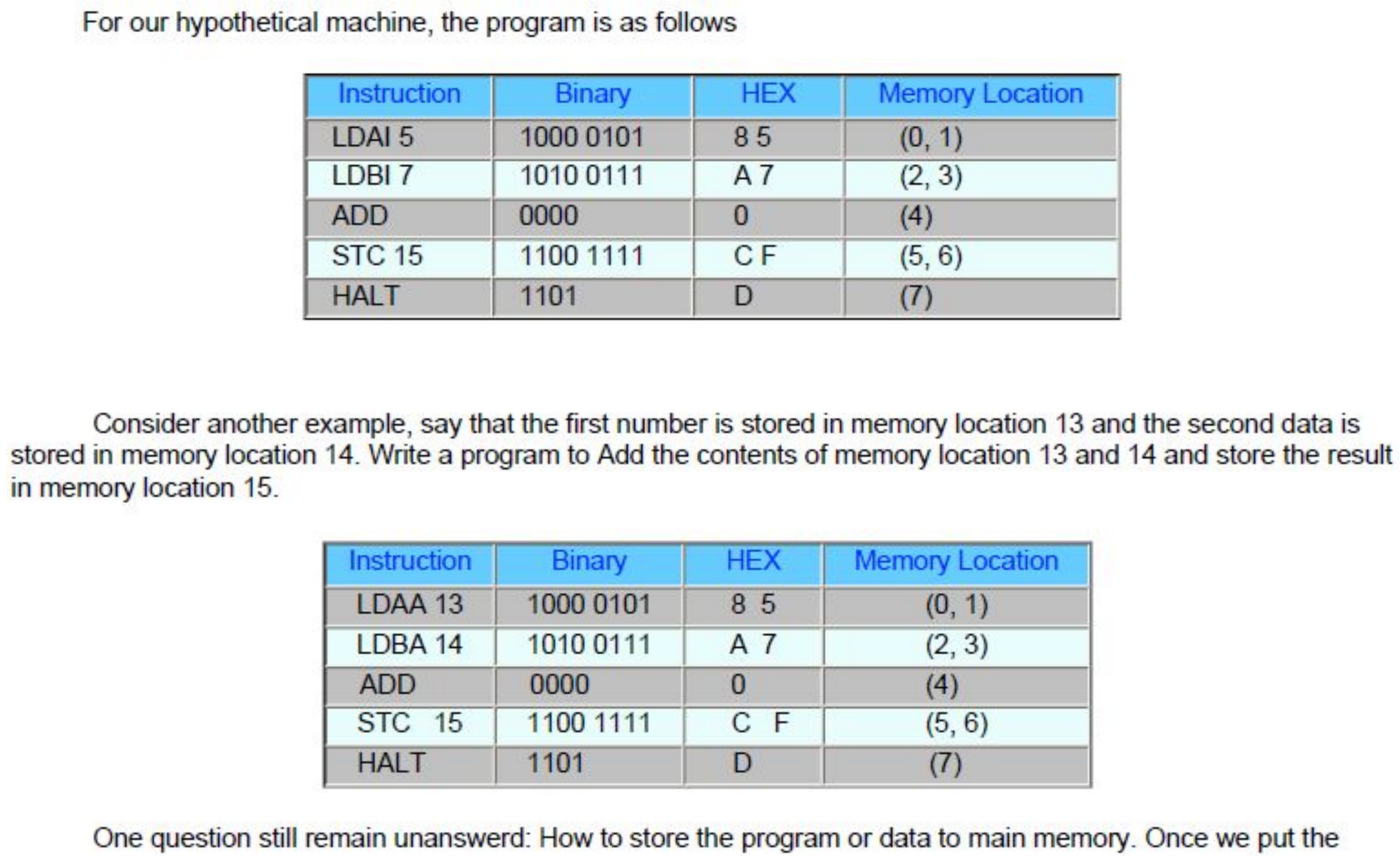
We need some more instruction to work with the computer. Apart from the instruction needed to perform task inside CPU, we need some more instructions for data transfer from main memory to CPU and vice versa.

In our hypothetical machine, we use three signal lines to identify a particular instruction. If we want to include more instruction, we need additional signal lines.

| Instruction | Code | Meaning |
|---|---|---|
| 1000 | LDAI imm | Load register A with data that is given in the program |
| 1001 | LDAA addr | Load register A with data from memory location addr |
| 1010 | LDBI imm | Load register B with data |
| 1011 | LDBA addr | Load register B with data from memory location addr |
| 1100 | STC addr | Store the value of register C in memory location addr |
| 1101 | HALT | Stop the execution |
| 1110 | NOP | No operation |
| 1111 | NOP | No operation |

With this additional signal line, we can go upto 16 instructions. When the signal of this new line is 0, it will indicate the ALU operation. For signal value equal to 1, it will indicate 8 new instructions. So, we can design 8 new memory access instructions.

# Memory Instruction



For our hypothetical machine, the program is as follows

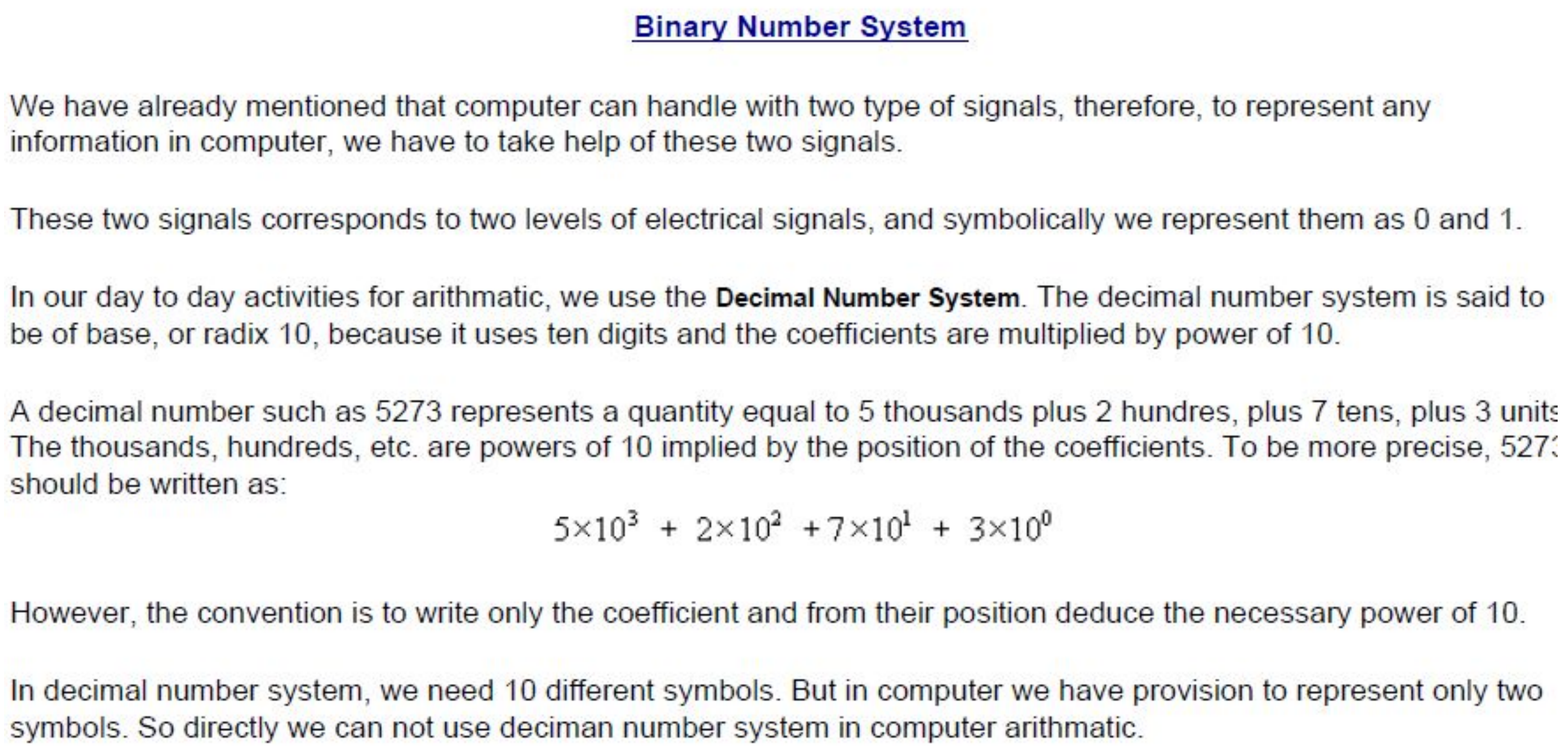
| Instruction | Binary | HEX | Memory Location |
|---|---|---|---|
| LDAI 5 | 1000 0101 | 8 5 | (0, 1) |
| LDBI 7 | 1010 0111 | A 7 | (2, 3) |
| ADD | 0000 | 0 | (4) |
| STC 15 | 1100 1111 | C F | (5, 6) |
| HALT | 1101 | D | (7) |

Consider another example, say that the first number is stored in memory location 13 and the second data is stored in memory location 14. Write a program to Add the contents of memory location 13 and 14 and store the result in memory location 15.

| Instruction | Binary | HEX | Memory Location |
|---|---|---|---|
| LDAA 13 | 1000 0101 | 8 5 | (0, 1) |
| LDBA 14 | 1010 0111 | A 7 | (2, 3) |
| ADD | 0000 | 0 | (4) |
| STC 15 | 1100 1111 | C F | (5, 6) |
| HALT | 1101 | D | (7) |

One question still remain unanswerd: How to store the program or data to main memory. Once we put the

# Number System and Representation



## Binary Number System

We have already mentioned that computer can handle with two type of signals, therefore, to represent any information in computer, we have to take help of these two signals.

These two signals corresponds to two levels of electrical signals, and symbolically we represent them as 0 and 1.

In our day to day activities for arithmatic, we use the **Decimal Number System**. The decimal number system is said to be of base, or radix 10, because it uses ten digits and the coefficients are multiplied by power of 10.

A decimal number such as 5273 represents a quantity equal to 5 thousands plus 2 hundres, plus 7 tens, plus 3 units The thousands, hundreds, etc. are powers of 10 implied by the position of the coefficients. To be more precise, 527 should be written as:

$$5\times10^3 + 2\times10^2 + 7\times10^1 + 3\times10^0$$

However, the convention is to write only the coefficient and from their position deduce the necessary power of 10.

In decimal number system, we need 10 different symbols. But in computer we have provision to represent only two symbols. So directly we can not use deciman number system in computer arithmatic.

## Number System and Representation

For computer arithmatic we use **binary number system**. The binary number system uses two symbols to represent the number and these two symbols are 0 and 1.

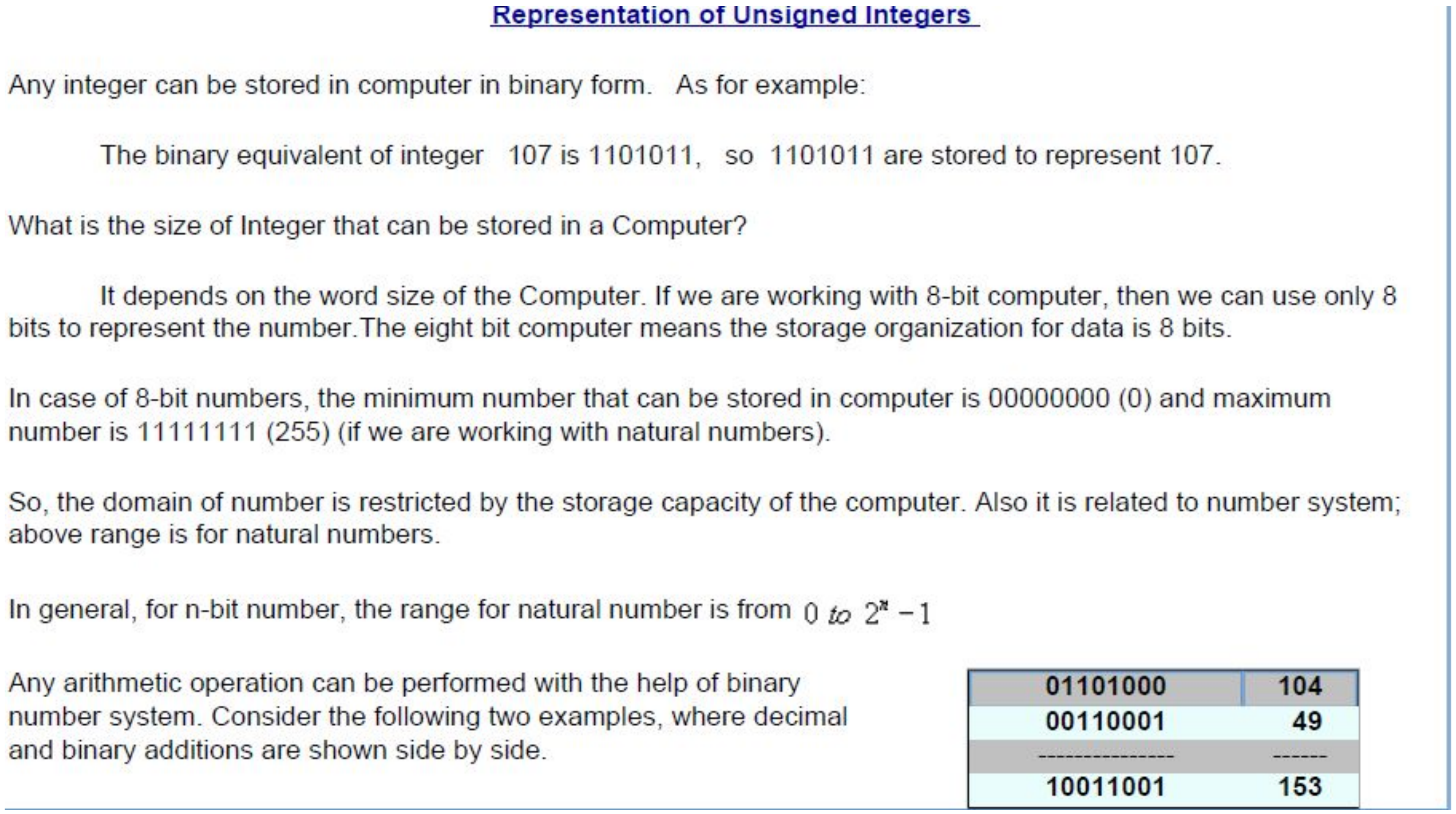
The binary number system is said to be of base 2 or radix 2, because it uses two digits and the coefficients are multiplied by power of 2.

The binary number 110011 represents the quantity equal to:

$$1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 51 \text{ (in decimal)}$$

We can use binary number system for computer arithmatic.

# Number System and Representation



## Representation of Unsigned Integers

Any integer can be stored in computer in binary form. As for example:

The binary equivalent of integer 107 is 1101011, so 1101011 are stored to represent 107.

What is the size of Integer that can be stored in a Computer?

It depends on the word size of the Computer. If we are working with 8-bit computer, then we can use only 8 bits to represent the number.The eight bit computer means the storage organization for data is 8 bits.

In case of 8-bit numbers, the minimum number that can be stored in computer is 00000000 (0) and maximum number is 11111111 (255) (if we are working with natural numbers).

So, the domain of number is restricted by the storage capacity of the computer. Also it is related to number system; above range is for natural numbers.

In general, for n-bit number, the range for natural number is from $0 \text{ to } 2^n - 1$

Any arithmetic operation can be performed with the help of binary number system. Consider the following two examples, where decimal and binary additions are shown side by side.

| | |
|---|---|
| 01101000 | 104 |
| 00110001 | 49 |
| --------------- | ------ |
| 10011001 | 153 |

# Number System and Representation

In the above example, the result is an 8-bit number, as it can be stored in the 8-bit computer, so we get the correct results.

| | |
|---|---|
| 10000001 | 129 |
| 10101010 | 178 |
| ----------------- | ------ |
| 100101011 | 307 |

In the above example, the result is a 9-bit number, but we can store only 8 bits, and the most significant bit (msb) cannot be stored.

The result of this addition will be stored as (00101011) which is 43 and it is not the desired result. Since we cannot store the complete result of an operation, and it is known as the overflow case.

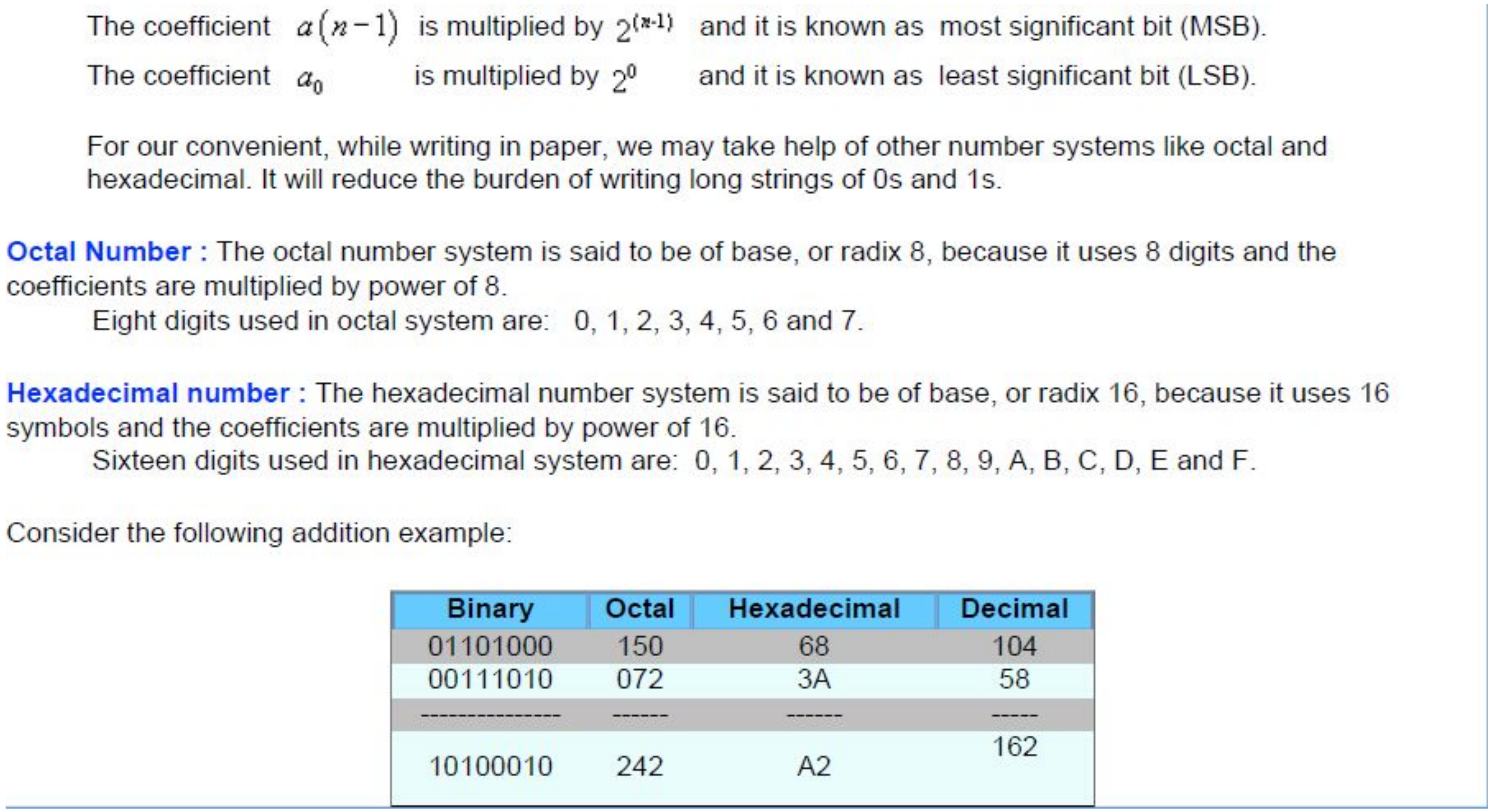
The smallest unit of information is known as **BIT** (BInary digiT).

The binary number 110011 consists of 6 bits and it represents:

$$1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

For an n-bit number the coefficient is - $a_j$ multiplied by $2^j$ where, $( 0 \leq j < n )$

# Number System and Representation



The coefficient $a(n-1)$ is multiplied by $2^{(n-1)}$ and it is known as most significant bit (MSB).

The coefficient $a_0$ is multiplied by $2^0$ and it is known as least significant bit (LSB).

For our convenient, while writing in paper, we may take help of other number systems like octal and hexadecimal. It will reduce the burden of writing long strings of 0s and 1s.

**Octal Number :** The octal number system is said to be of base, or radix 8, because it uses 8 digits and the coefficients are multiplied by power of 8.

Eight digits used in octal system are: 0, 1, 2, 3, 4, 5, 6 and 7.

**Hexadecimal number :** The hexadecimal number system is said to be of base, or radix 16, because it uses 16 symbols and the coefficients are multiplied by power of 16.

Sixteen digits used in hexadecimal system are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.

Consider the following addition example:

| Binary | Octal | Hexadecimal | Decimal |
|---|---|---|---|
| 01101000 | 150 | 68 | 104 |
| 00111010 | 072 | 3A | 58 |
| --------------- | ------ | ------ | ----- |
| 10100010 | 242 | A2 | 162 |

# Number System and Representation

## Signed Integer

We know that for n-bit number, the range for natural number is from $0 \; to \; 2^n - 1$.

For n-bit, we have all together $2^n$ different combination, and we use these different combination to represent $2^n$ numbers, which ranges from $0 \; to \; 2^n - 1$.

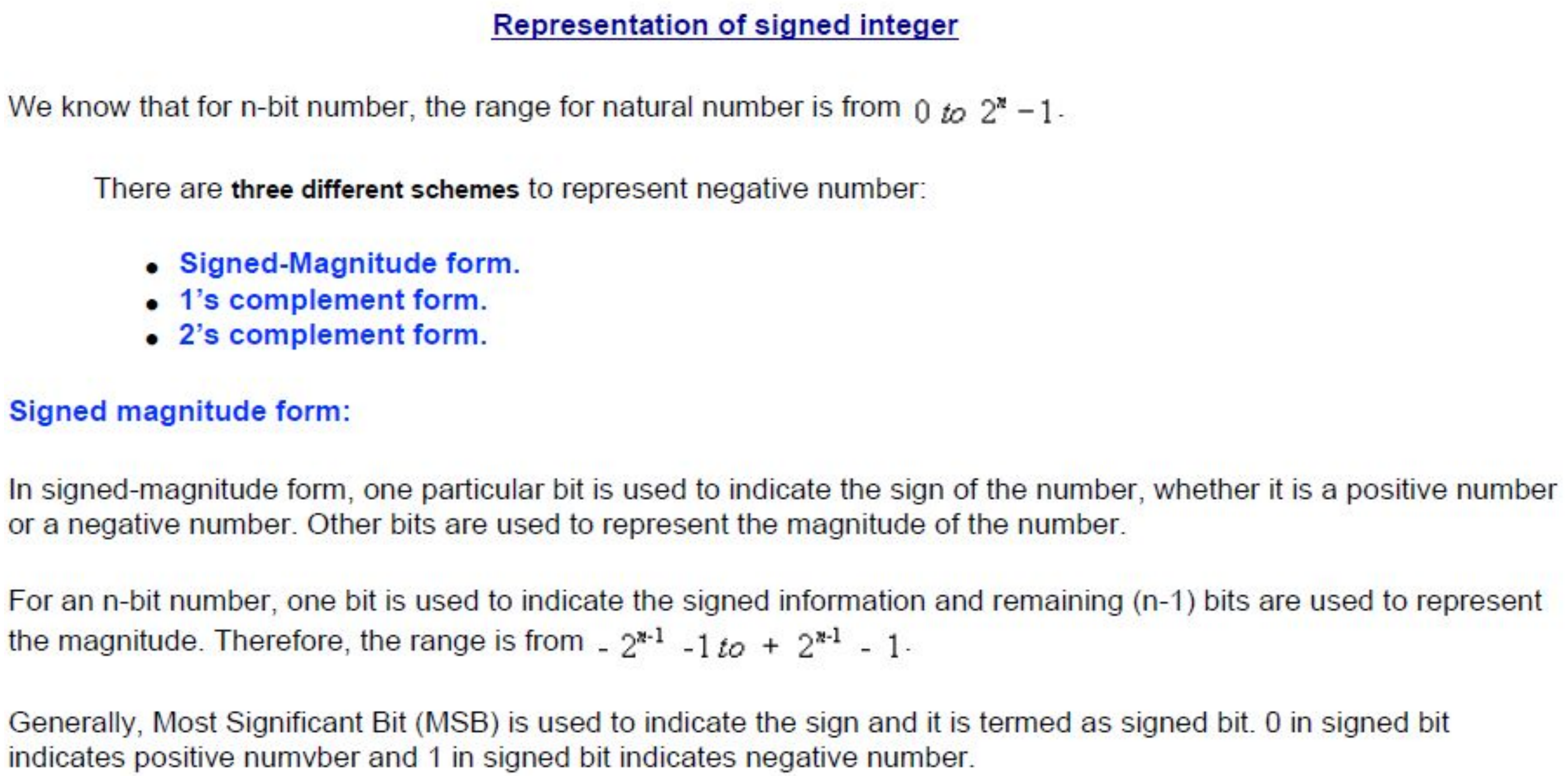
If we want to include the negative number, naturally, the range will decrease. Half of the combinations are used for positive number and other half is used for negative number.

For n-bit represenatation, the range is from $- 2^{n-1} - 1 \; to \; + \; 2^{n-1} - 1$.

For example, if we consider 8-bit number, then range

for natural number is from 0 to 255; but
for signed integer the range is from -127 to +127.

# Number System and Representation



**Representation of signed integer**

We know that for n-bit number, the range for natural number is from $0 \text{ to } 2^n - 1$.

There are **three different schemes** to represent negative number:

- **Signed-Magnitude form.**
- **1's complement form.**
- **2's complement form.**

**Signed magnitude form:**

In signed-magnitude form, one particular bit is used to indicate the sign of the number, whether it is a positive number or a negative number. Other bits are used to represent the magnitude of the number.

For an n-bit number, one bit is used to indicate the signed information and remaining (n-1) bits are used to represent the magnitude. Therefore, the range is from $- 2^{n-1} - 1 \text{ to } + 2^{n-1} - 1$.

Generally, Most Significant Bit (MSB) is used to indicate the sign and it is termed as signed bit. 0 in signed bit indicates positive numvber and 1 in signed bit indicates negative number.

# Number System and Representation

For example, 01011001 represents + 169 and
11011001 represents - 169

What is 00000000 and 10000000 in signed magnitude form?

**The concept of complement**

The concept of complements is used to represent signed number.

Consider a number system of base-r or radix-r. There are two types of complements,

- The radix complement or the r's complement.
- The diminished radix complement or the (r - 1)'s complement.

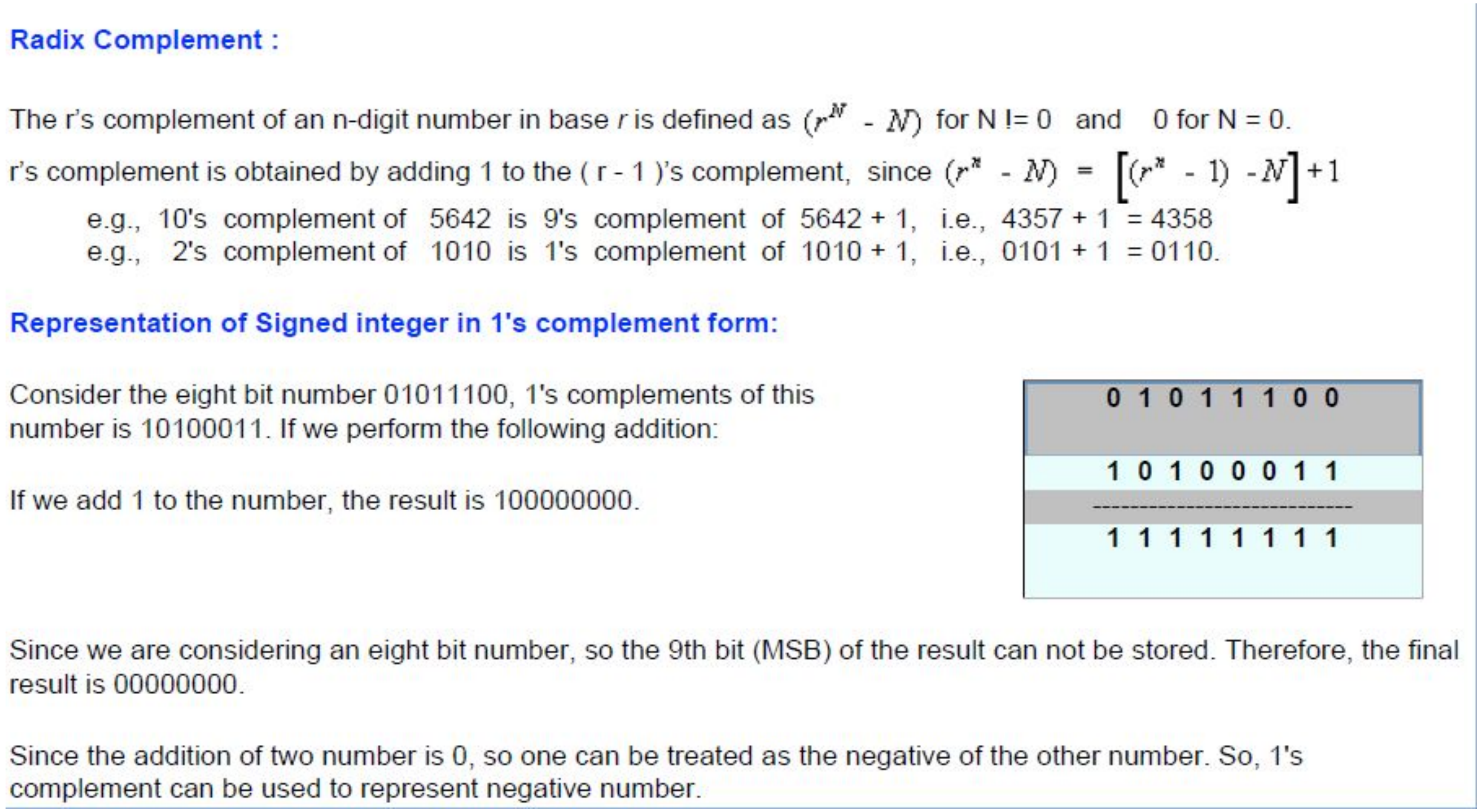
**Diminished Radix Complement :**

Given a number N in base r having n digits, the (r - 1)'s complement of N is defined as $(r^n - 1) - N$.

For decimal numbers, r = 10 and r - 1 = 9, so the 9's complement of N is $(10^n - 1) - N$.

e.g., 9's complement of 5642 is 9999 - 5642 = 4357.

# Number System and Representation



**Radix Complement :**

The r's complement of an n-digit number in base *r* is defined as $(r^N - N)$ for N != 0 and 0 for N = 0.

r's complement is obtained by adding 1 to the ( r - 1 )'s complement, since $(r^n - N) = \left[(r^n - 1) - N\right] + 1$

e.g., 10's complement of 5642 is 9's complement of 5642 + 1, i.e., 4357 + 1 = 4358
e.g., 2's complement of 1010 is 1's complement of 1010 + 1, i.e., 0101 + 1 = 0110.

**Representation of Signed integer in 1's complement form:**

Consider the eight bit number 01011100, 1's complements of this number is 10100011. If we perform the following addition:

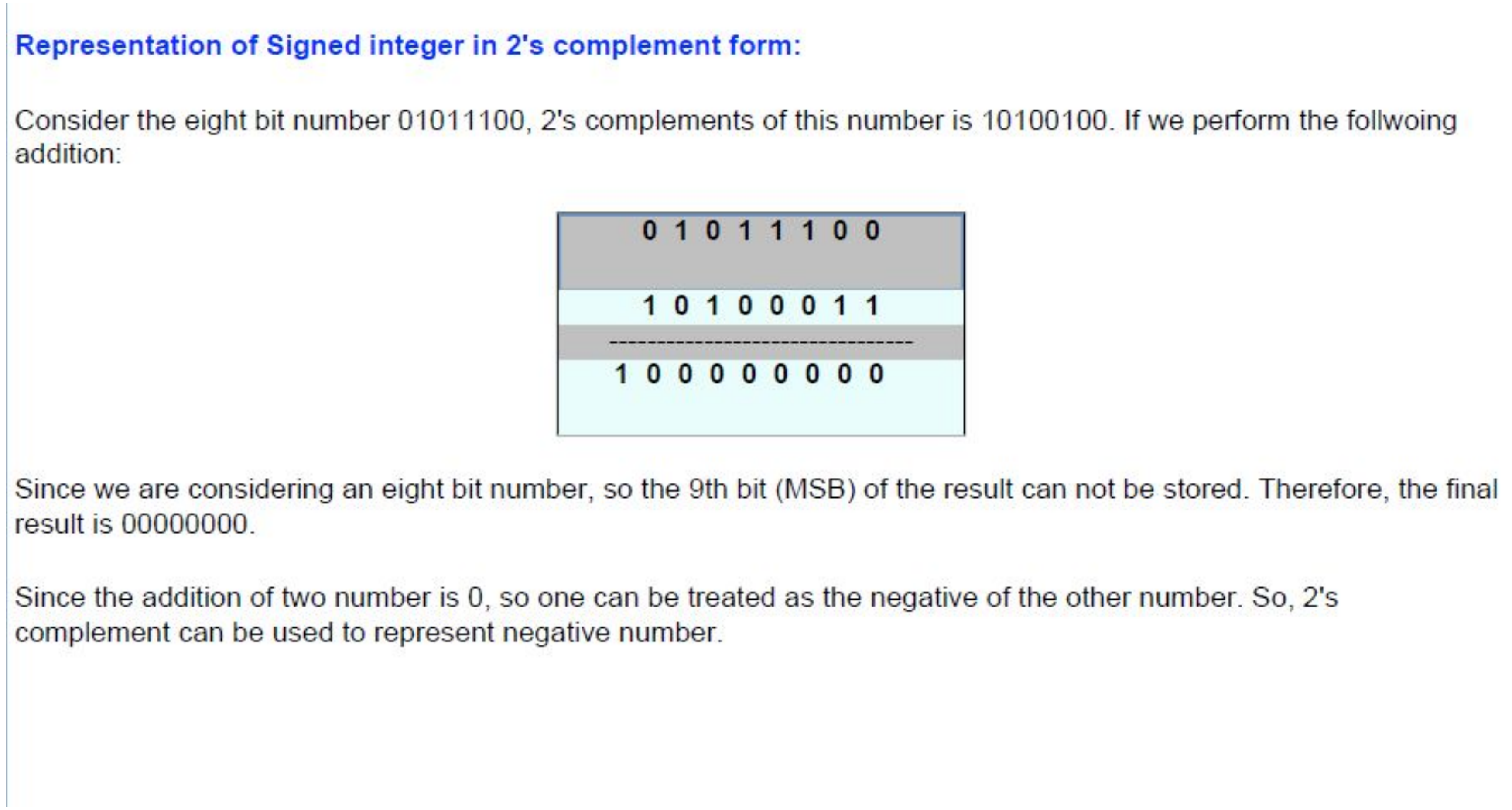
```
0 1 0 1 1 1 0 0
1 0 1 0 0 0 1 1
----------------------------
1 1 1 1 1 1 1 1
```

If we add 1 to the number, the result is 100000000.

Since we are considering an eight bit number, so the 9th bit (MSB) of the result can not be stored. Therefore, the final result is 00000000.

Since the addition of two number is 0, so one can be treated as the negative of the other number. So, 1's complement can be used to represent negative number.

# Number System and Representation



**Representation of Signed integer in 2's complement form:**

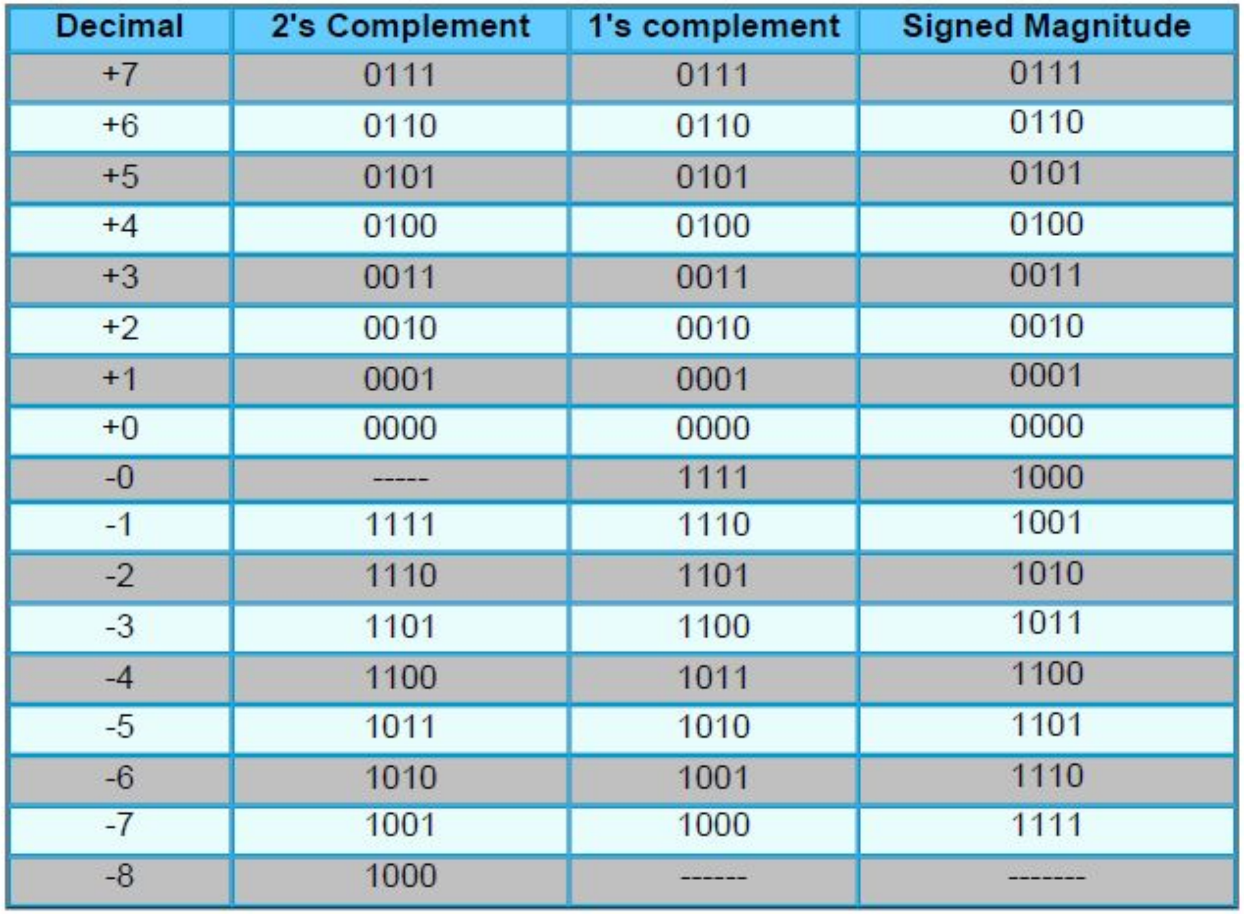
Consider the eight bit number 01011100, 2's complements of this number is 10100100. If we perform the follwoing addition:

| |
|---|
| 0 1 0 1 1 1 0 0 |
| 1 0 1 0 0 0 1 1 |
| -------------------------------- |
| 1 0 0 0 0 0 0 0 0 |

Since we are considering an eight bit number, so the 9th bit (MSB) of the result can not be stored. Therefore, the final result is 00000000.

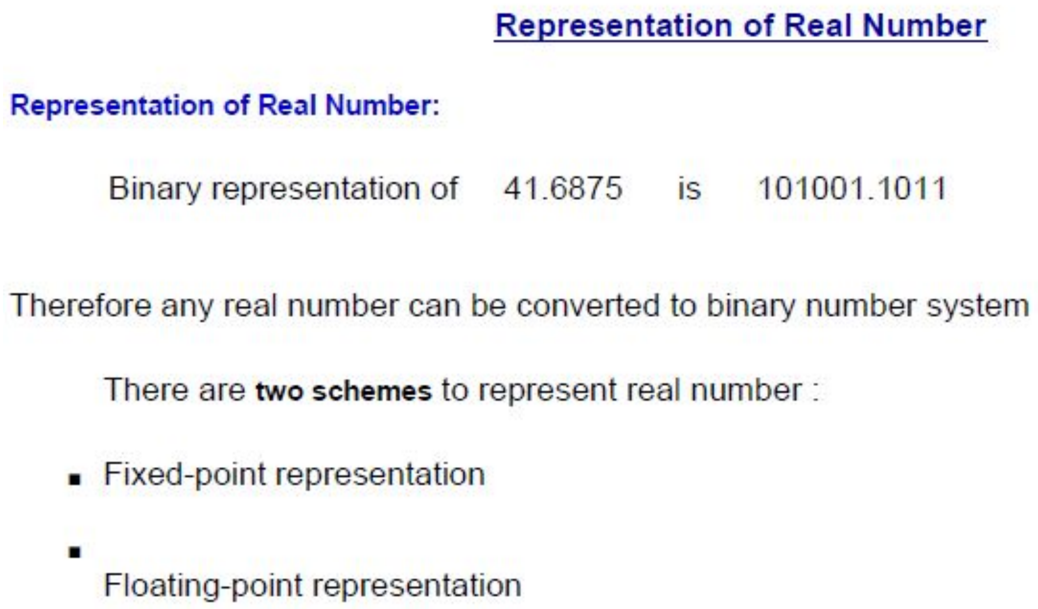
Since the addition of two number is 0, so one can be treated as the negative of the other number. So, 2's complement can be used to represent negative number.

# Number System and Representation



| Decimal | 2's Complement | 1's complement | Signed Magnitude |
|---|---|---|---|
| +7 | 0111 | 0111 | 0111 |
| +6 | 0110 | 0110 | 0110 |
| +5 | 0101 | 0101 | 0101 |
| +4 | 0100 | 0100 | 0100 |
| +3 | 0011 | 0011 | 0011 |
| +2 | 0010 | 0010 | 0010 |
| +1 | 0001 | 0001 | 0001 |
| +0 | 0000 | 0000 | 0000 |
| -0 | ----- | 1111 | 1000 |
| -1 | 1111 | 1110 | 1001 |
| -2 | 1110 | 1101 | 1010 |
| -3 | 1101 | 1100 | 1011 |
| -4 | 1100 | 1011 | 1100 |
| -5 | 1011 | 1010 | 1101 |
| -6 | 1010 | 1001 | 1110 |
| -7 | 1001 | 1000 | 1111 |
| -8 | 1000 | ------ | ------- |

# Number System and Representation



## Representation of Real Number

**Representation of Real Number:**

Binary representation of 41.6875 is 101001.1011

Therefore any real number can be converted to binary number system

There are **two schemes** to represent real number :

- Fixed-point representation
- Floating-point representation

# Number System and Representation

**Fixed-point representation:**

Binary representation of 41.6875 is 101001.1011

To store this number, we have to store **two information**,

-- the part before decimal point and
-- the part after decimal point.

This is known as fixed-point representation where the position of decimal point is fixed and number of bits before and after decimal point are also predefined.

If we use 16 bits before decimal point and 8 bits after decimal point, in signed magnitude form, the range is

$$-2^{16} - 1 \text{ } \textit{to} \text{ } +2^{16} - 1 \quad \text{and the precision is } 2^{(-8)}$$

One bit is required for sign information, so the total size of the number is 25 bits

( 1(sign) + 16(before decimal point) + 8(after decimal point) ).

# Number System and Representation

**Floating-point representation:**

In this representation, numbers are represented by a mantissa comprising the significant digits and an exponent part of Radix R. The format is:

$$mantissa * R^{exponent}$$

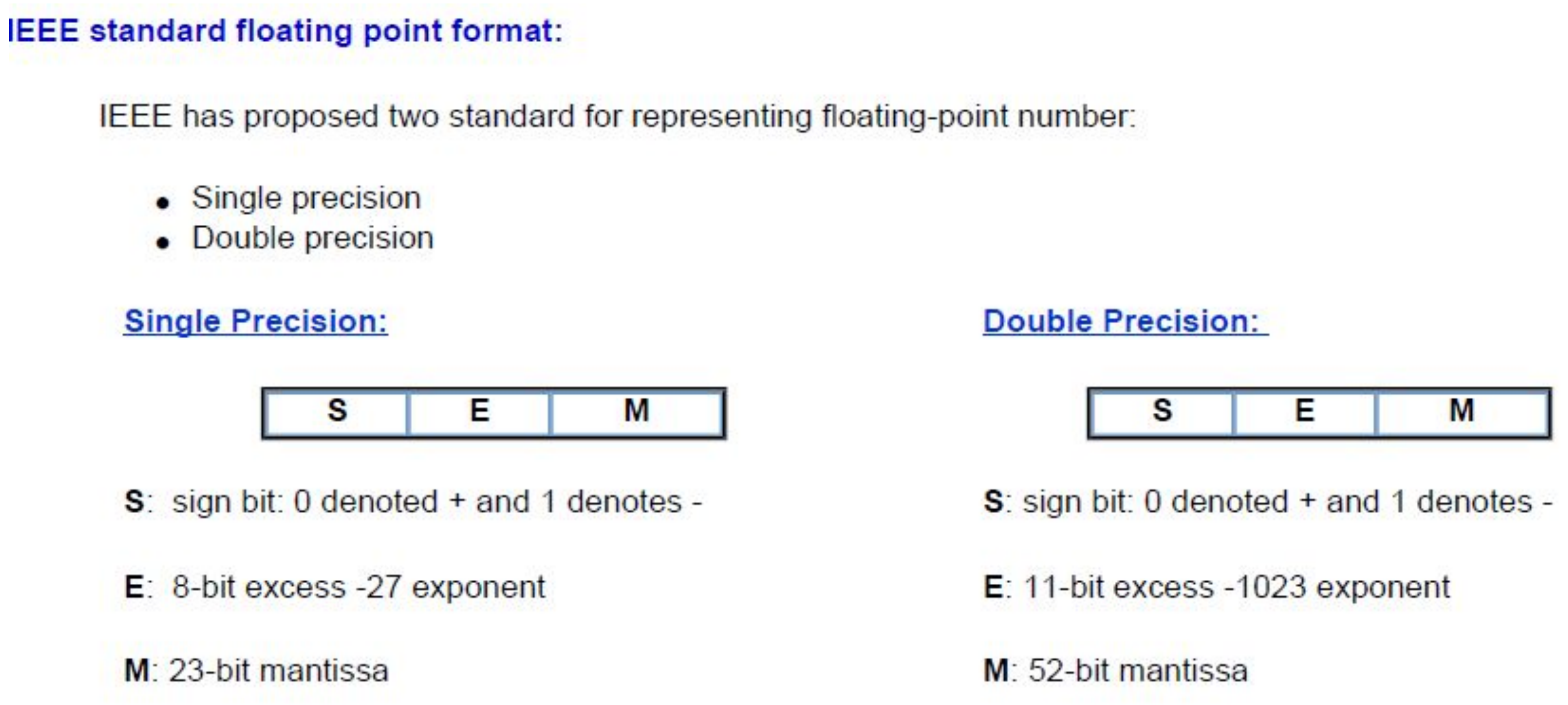
Numbers are often normalized, such that the decimal point is placed to the right of the **first non zero digit**.

For example, the decimal number,

5236 is equivqlent to $5.236 * 10^3$

To store this number in floating point representation, we store 5236 in mantissa part and 3 in exponent part.

# Number System and Representation



**IEEE standard floating point format:**

IEEE has proposed two standard for representing floating-point number:

- Single precision
- Double precision

**Single Precision:**

| S | E | M |
|---|---|---|

**S**: sign bit: 0 denoted + and 1 denotes -

**E**: 8-bit excess -27 exponent

**M**: 23-bit mantissa

**Double Precision:**

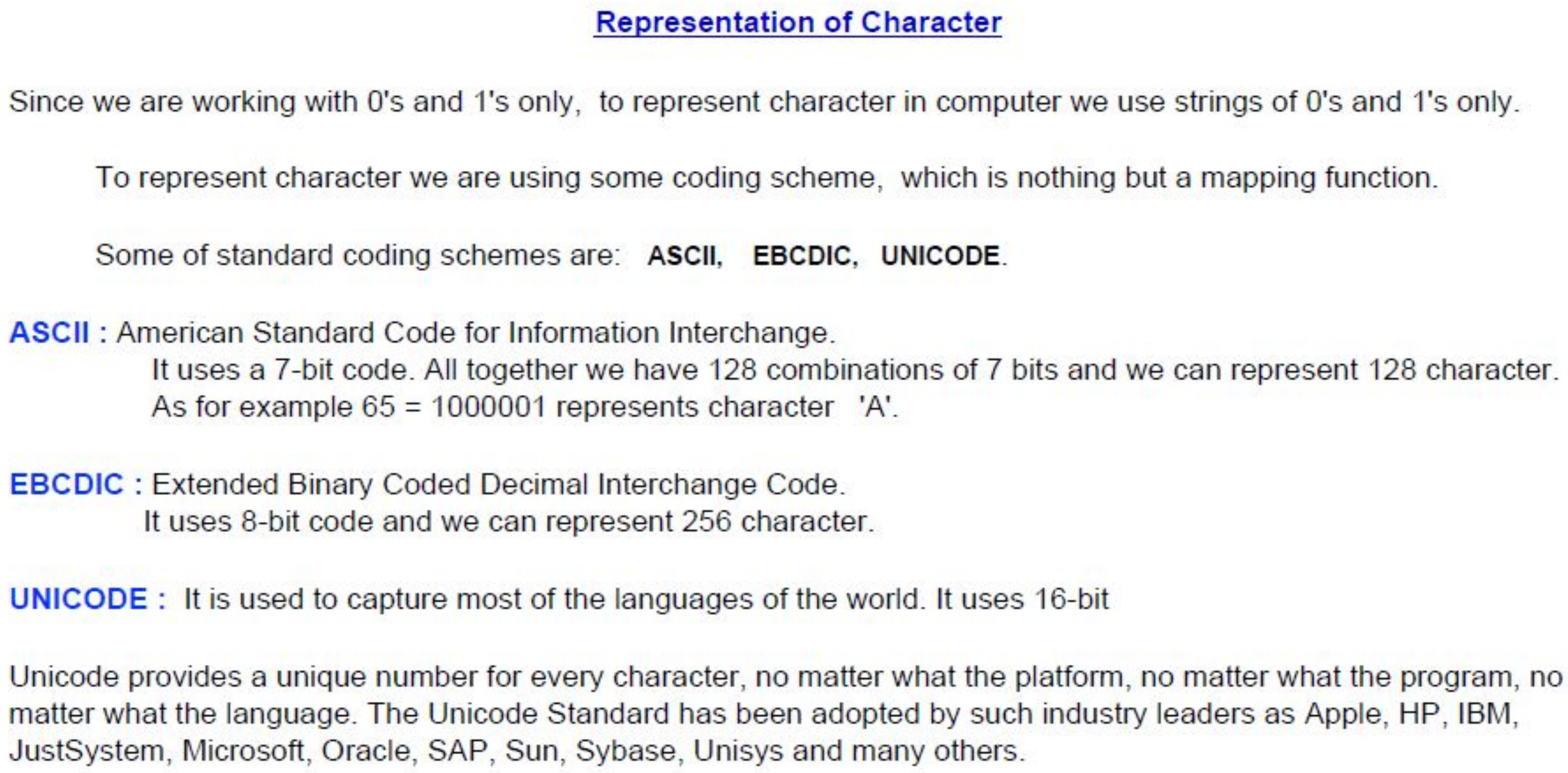
| S | E | M |
|---|---|---|

**S**: sign bit: 0 denoted + and 1 denotes -

**E**: 11-bit excess -1023 exponent

**M**: 52-bit mantissa

# Number System and Representation



## Representation of Character

Since we are working with 0's and 1's only, to represent character in computer we use strings of 0's and 1's only.

To represent character we are using some coding scheme, which is nothing but a mapping function.

Some of standard coding schemes are: **ASCII, EBCDIC, UNICODE**.

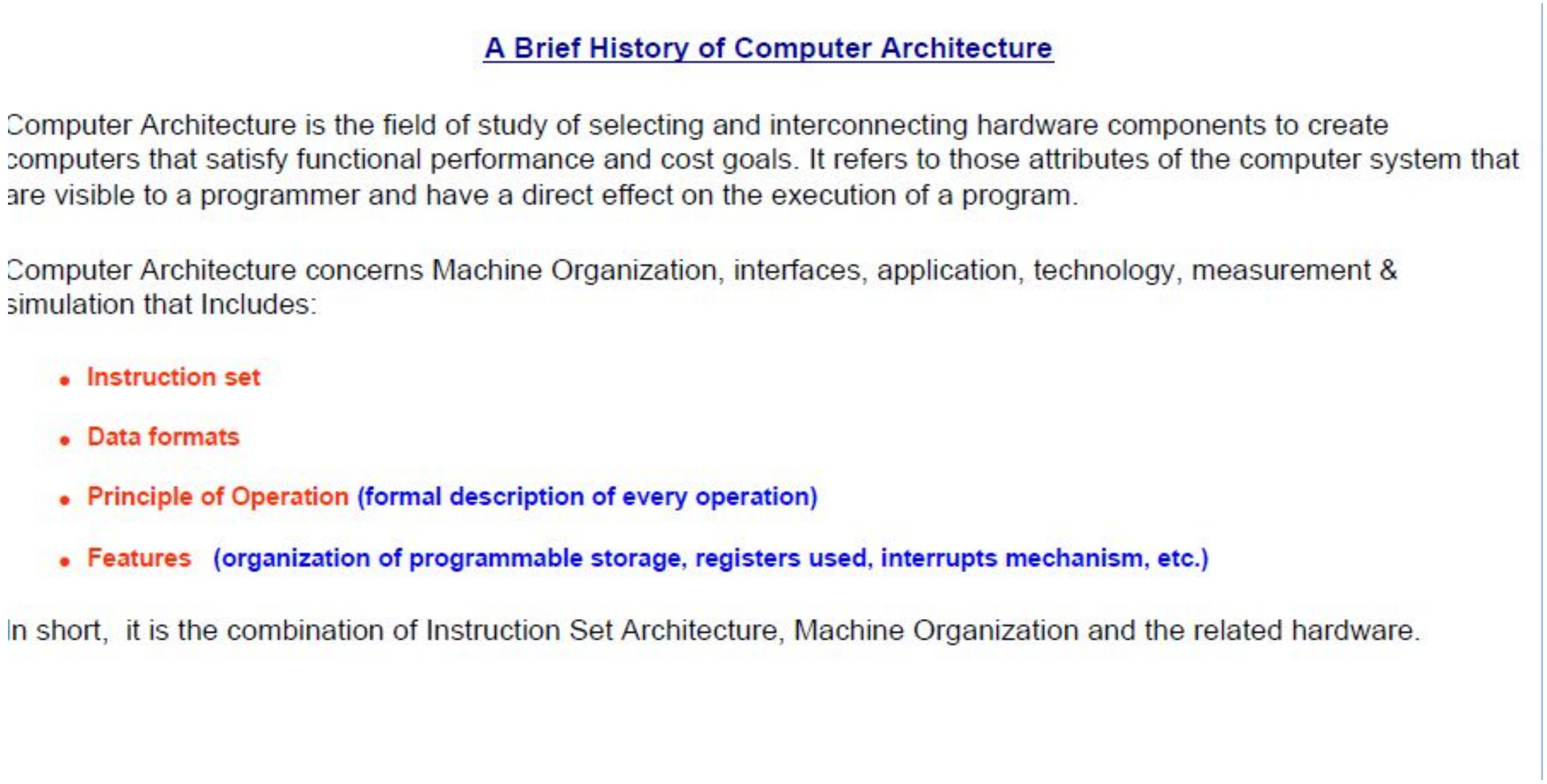
**ASCII :** American Standard Code for Information Interchange.
It uses a 7-bit code. All together we have 128 combinations of 7 bits and we can represent 128 character.
As for example 65 = 1000001 represents character 'A'.

**EBCDIC :** Extended Binary Coded Decimal Interchange Code.
It uses 8-bit code and we can represent 256 character.

**UNICODE :** It is used to capture most of the languages of the world. It uses 16-bit

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and many others.

# Brief History of Computer Evolution



## A Brief History of Computer Architecture

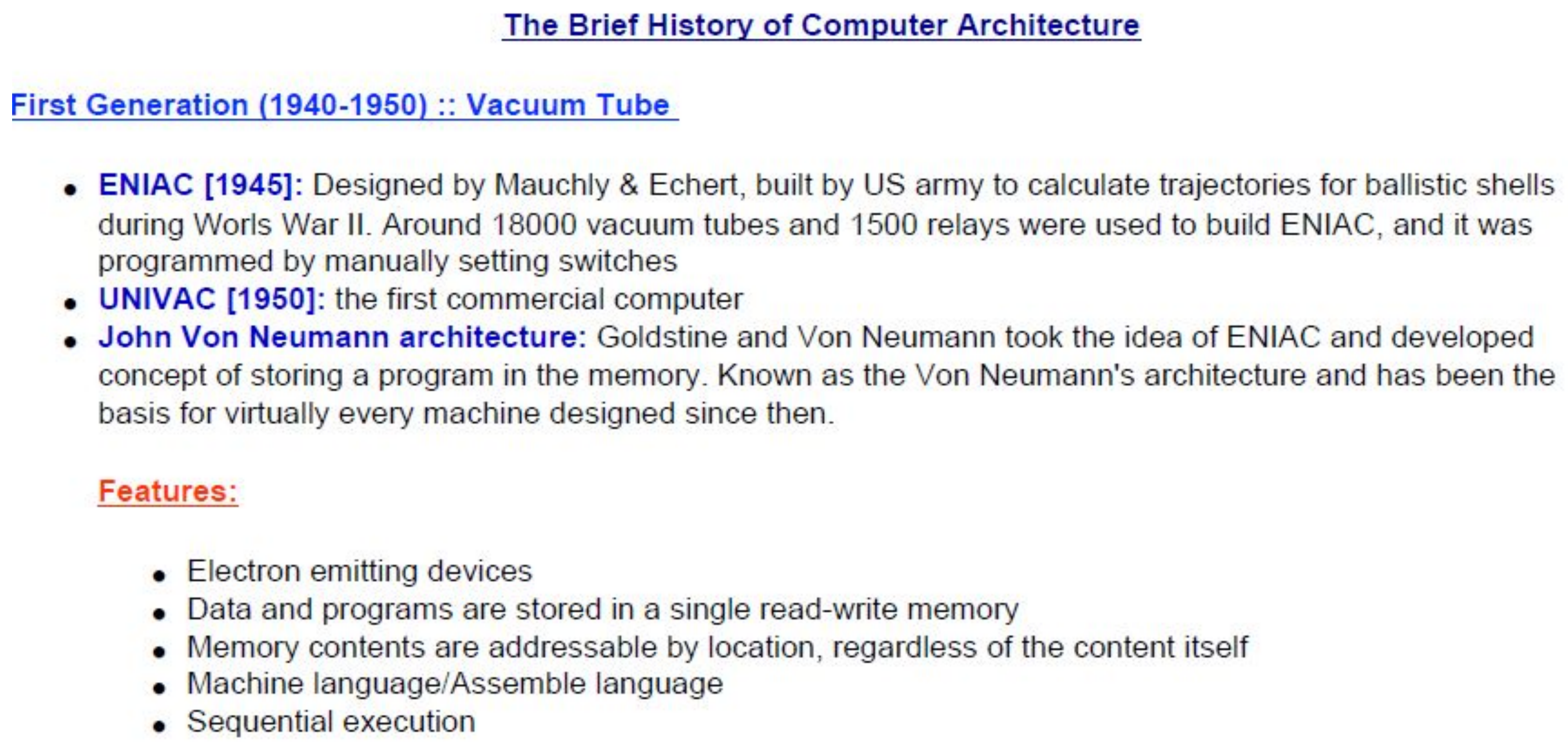
Computer Architecture is the field of study of selecting and interconnecting hardware components to create computers that satisfy functional performance and cost goals. It refers to those attributes of the computer system that are visible to a programmer and have a direct effect on the execution of a program.

Computer Architecture concerns Machine Organization, interfaces, application, technology, measurement & simulation that Includes:

- **Instruction set**
- **Data formats**
- **Principle of Operation** **(formal description of every operation)**
- **Features** **(organization of programmable storage, registers used, interrupts mechanism, etc.)**

In short, it is the combination of Instruction Set Architecture, Machine Organization and the related hardware.

# Brief History of Computer Evolution



**The Brief History of Computer Architecture**

**First Generation (1940-1950) :: Vacuum Tube**

- **ENIAC [1945]:** Designed by Mauchly & Echert, built by US army to calculate trajectories for ballistic shells during Worls War II. Around 18000 vacuum tubes and 1500 relays were used to build ENIAC, and it was programmed by manually setting switches
- **UNIVAC [1950]:** the first commercial computer
- **John Von Neumann architecture:** Goldstine and Von Neumann took the idea of ENIAC and developed concept of storing a program in the memory. Known as the Von Neumann's architecture and has been the basis for virtually every machine designed since then.

**Features:**

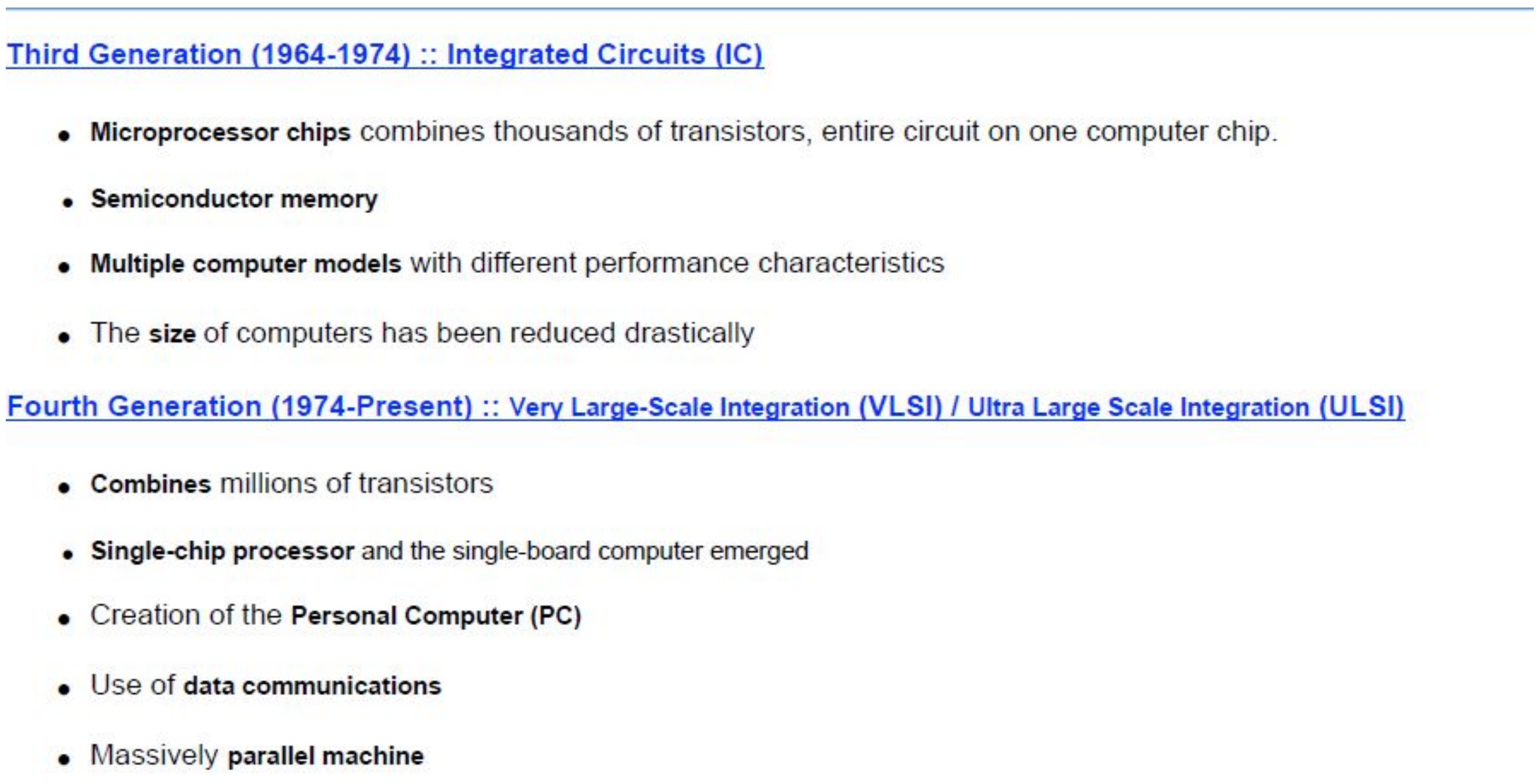
- Electron emitting devices
- Data and programs are stored in a single read-write memory
- Memory contents are addressable by location, regardless of the content itself
- Machine language/Assemble language
- Sequential execution

## Brief History of Computer Evolution

**Second Generation (1950-1964) :: Transistors**

- William Shockley, John Bardeen, and Walter Brattain invent the **transistor** that reduce size of computers and improve reliability. Vacuum tubes have been replaced by transistors.
- **First operating Systems:** handled one program at a time
- **On-off switches** controlled by electronically.
- **High level languages**
- **Floating point arithmetic**

# Brief History of Computer Evolution



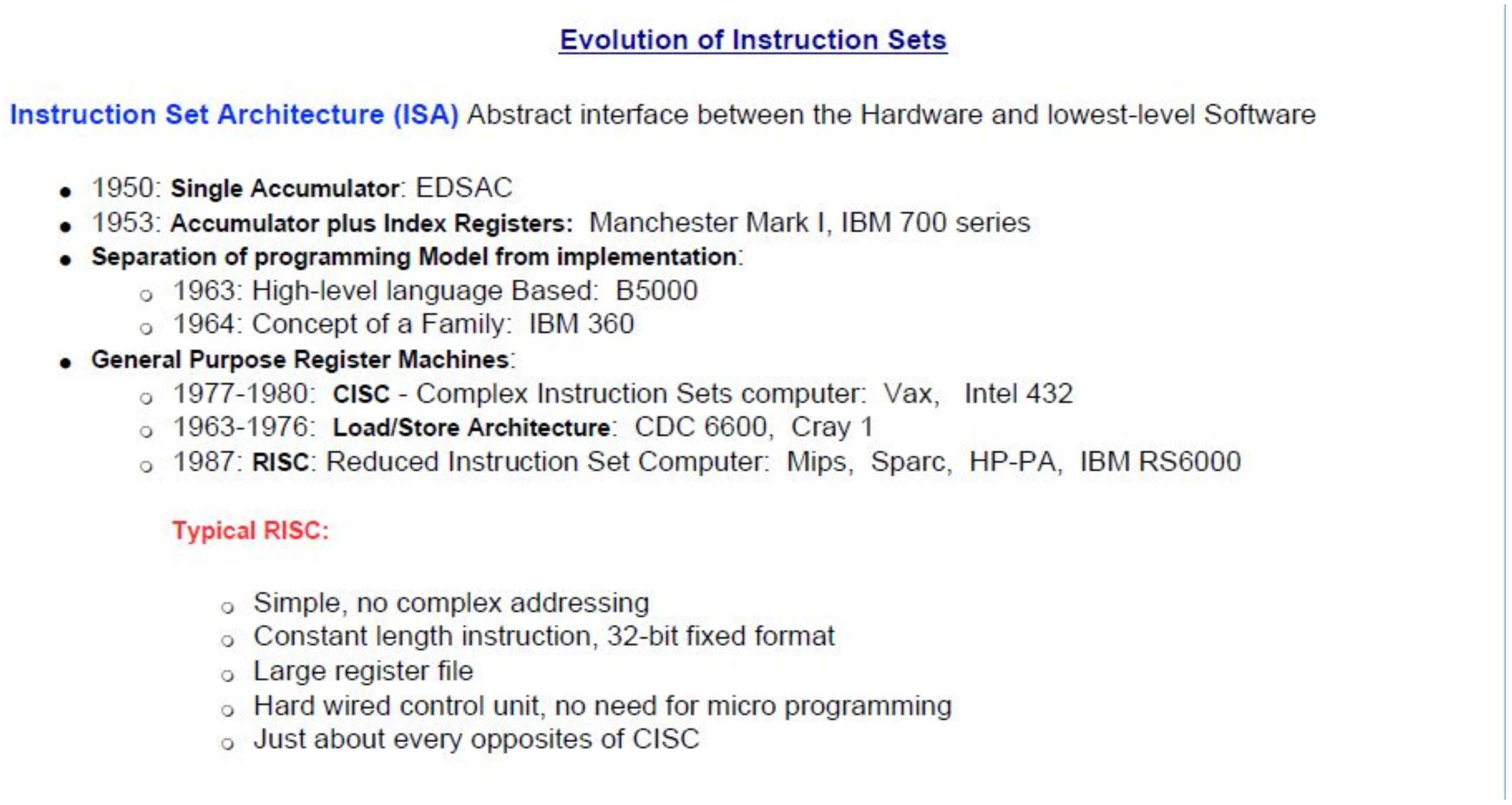
**Third Generation (1964-1974) :: Integrated Circuits (IC)**

- **Microprocessor chips** combines thousands of transistors, entire circuit on one computer chip.
- **Semiconductor memory**
- **Multiple computer models** with different performance characteristics
- The **size** of computers has been reduced drastically

**Fourth Generation (1974-Present) :: Very Large-Scale Integration (VLSI) / Ultra Large Scale Integration (ULSI)**

- **Combines** millions of transistors
- **Single-chip processor** and the single-board computer emerged
- Creation of the **Personal Computer (PC)**
- Use of **data communications**
- Massively **parallel machine**

# Brief History of Computer Evolution



**Evolution of Instruction Sets**

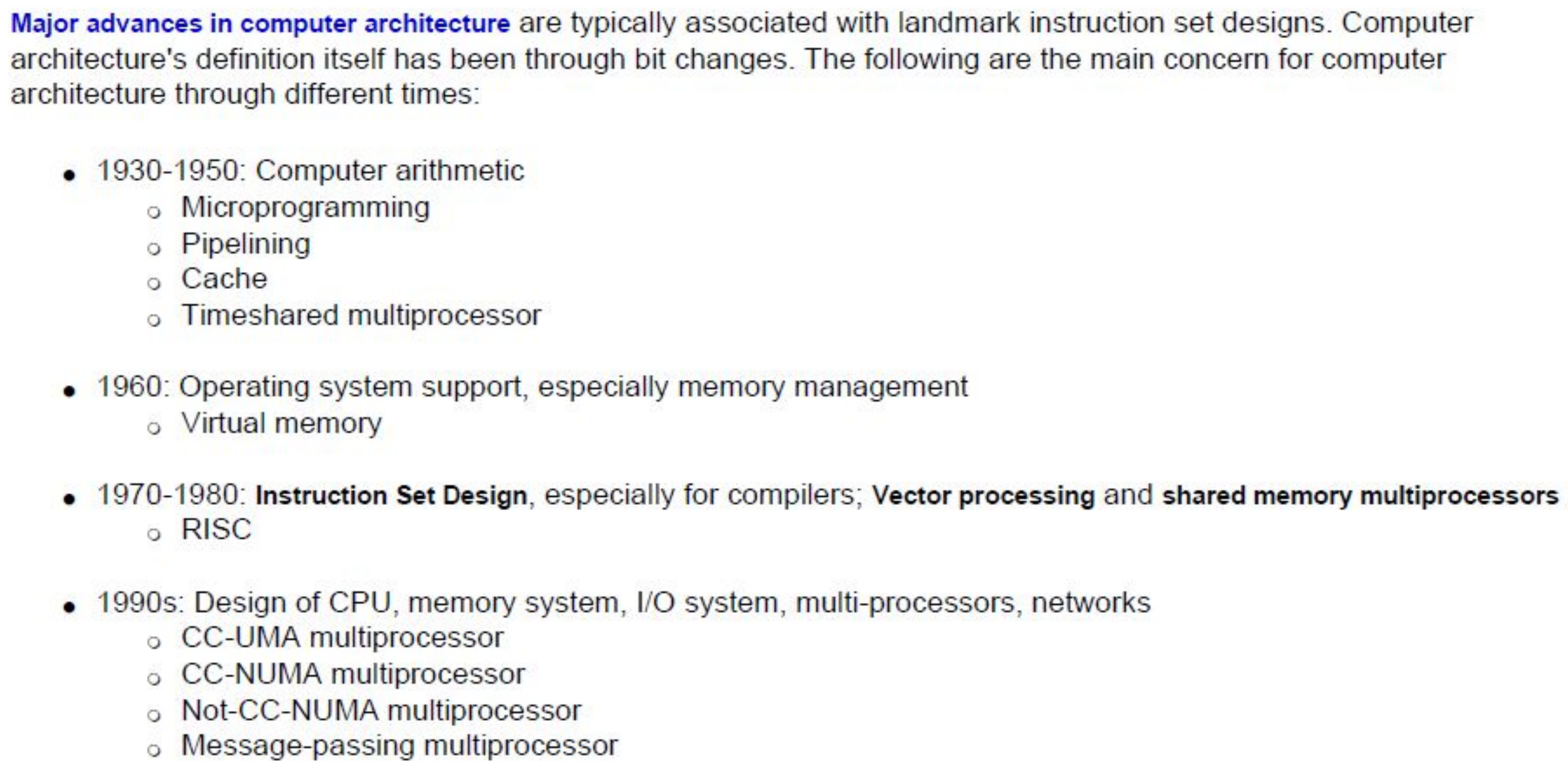
**Instruction Set Architecture (ISA)** Abstract interface between the Hardware and lowest-level Software

- 1950: **Single Accumulator**: EDSAC
- 1953: **Accumulator plus Index Registers:** Manchester Mark I, IBM 700 series
- **Separation of programming Model from implementation**:
  - 1963: High-level language Based: B5000
  - 1964: Concept of a Family: IBM 360
- **General Purpose Register Machines**:
  - 1977-1980: **CISC** - Complex Instruction Sets computer: Vax, Intel 432
  - 1963-1976: **Load/Store Architecture**: CDC 6600, Cray 1
  - 1987: **RISC**: Reduced Instruction Set Computer: Mips, Sparc, HP-PA, IBM RS6000

  **Typical RISC:**

    - Simple, no complex addressing
    - Constant length instruction, 32-bit fixed format
    - Large register file
    - Hard wired control unit, no need for micro programming
    - Just about every opposites of CISC

# Brief History of Computer Evolution



**Major advances in computer architecture** are typically associated with landmark instruction set designs. Computer architecture's definition itself has been through bit changes. The following are the main concern for computer architecture through different times:

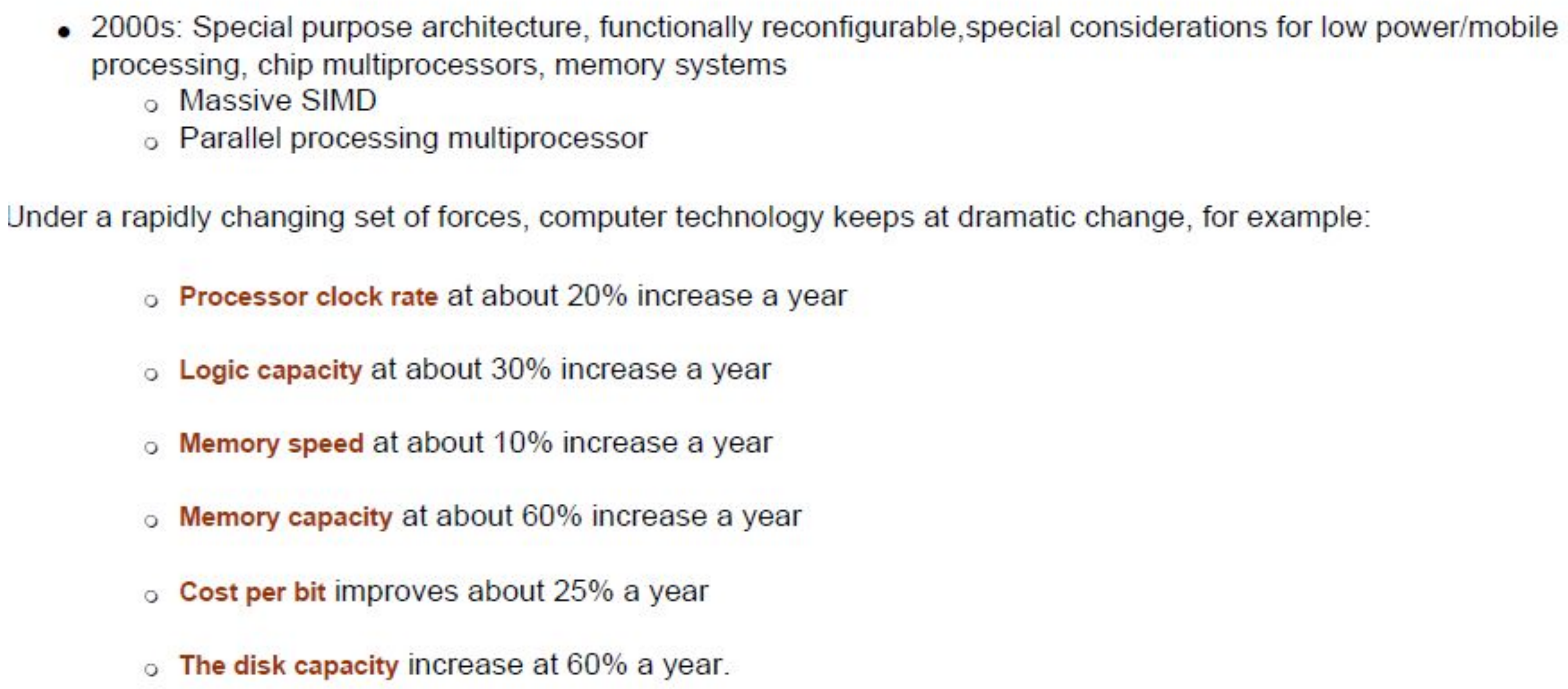
- 1930-1950: Computer arithmetic
  - Microprogramming
  - Pipelining
  - Cache
  - Timeshared multiprocessor

- 1960: Operating system support, especially memory management
  - Virtual memory

- 1970-1980: **Instruction Set Design**, especially for compilers; **Vector processing** and **shared memory multiprocessors**
  - RISC

- 1990s: Design of CPU, memory system, I/O system, multi-processors, networks
  - CC-UMA multiprocessor
  - CC-NUMA multiprocessor
  - Not-CC-NUMA multiprocessor
  - Message-passing multiprocessor

# Brief History of Computer Evolution

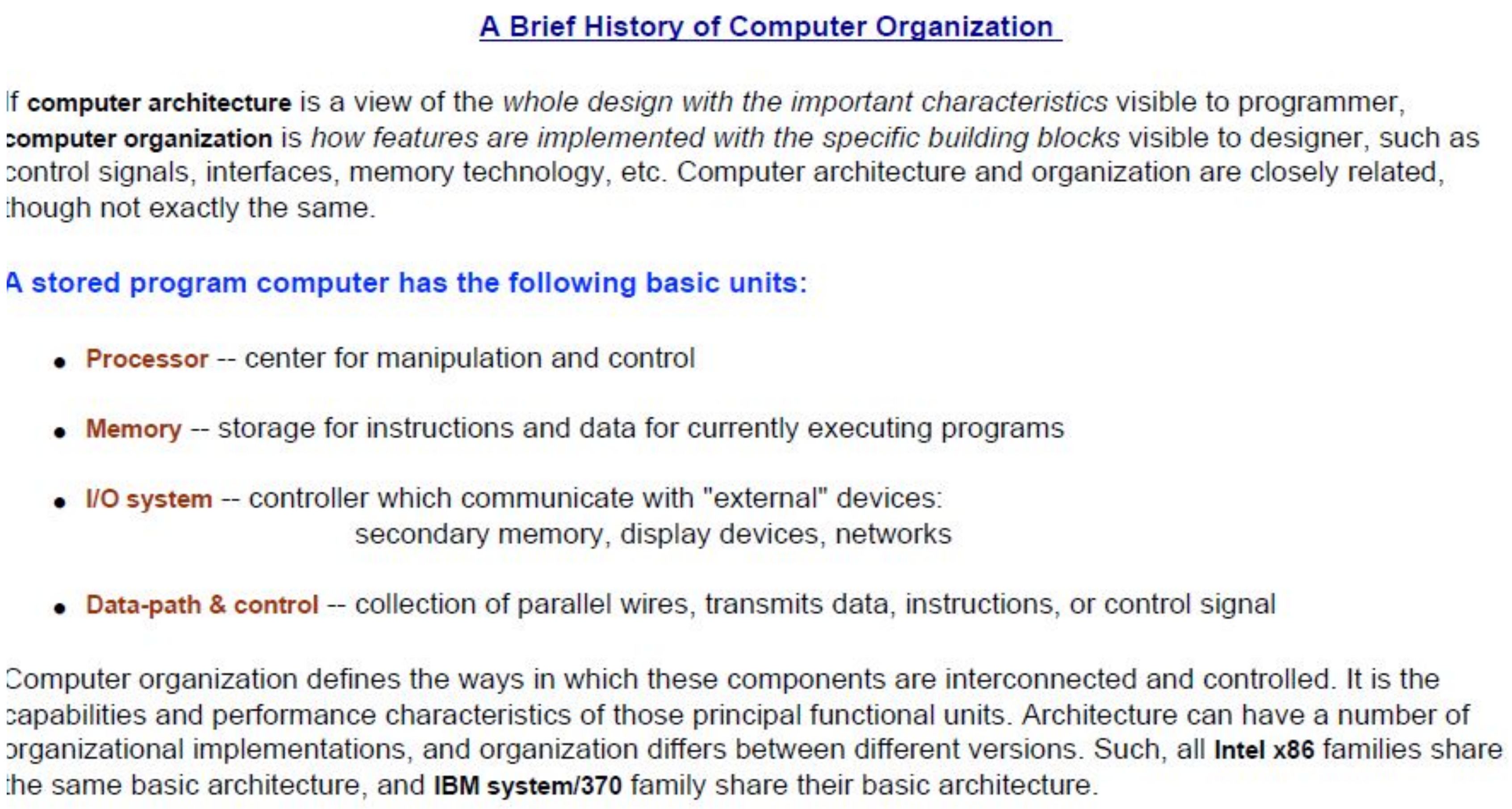


- 2000s: Special purpose architecture, functionally reconfigurable,special considerations for low power/mobile processing, chip multiprocessors, memory systems
  - Massive SIMD
  - Parallel processing multiprocessor

Under a rapidly changing set of forces, computer technology keeps at dramatic change, for example:

- **Processor clock rate** at about 20% increase a year
- **Logic capacity** at about 30% increase a year
- **Memory speed** at about 10% increase a year
- **Memory capacity** at about 60% increase a year
- **Cost per bit** improves about 25% a year
- **The disk capacity** increase at 60% a year.

# Brief History of Computer Evolution



A Brief History of Computer Organization

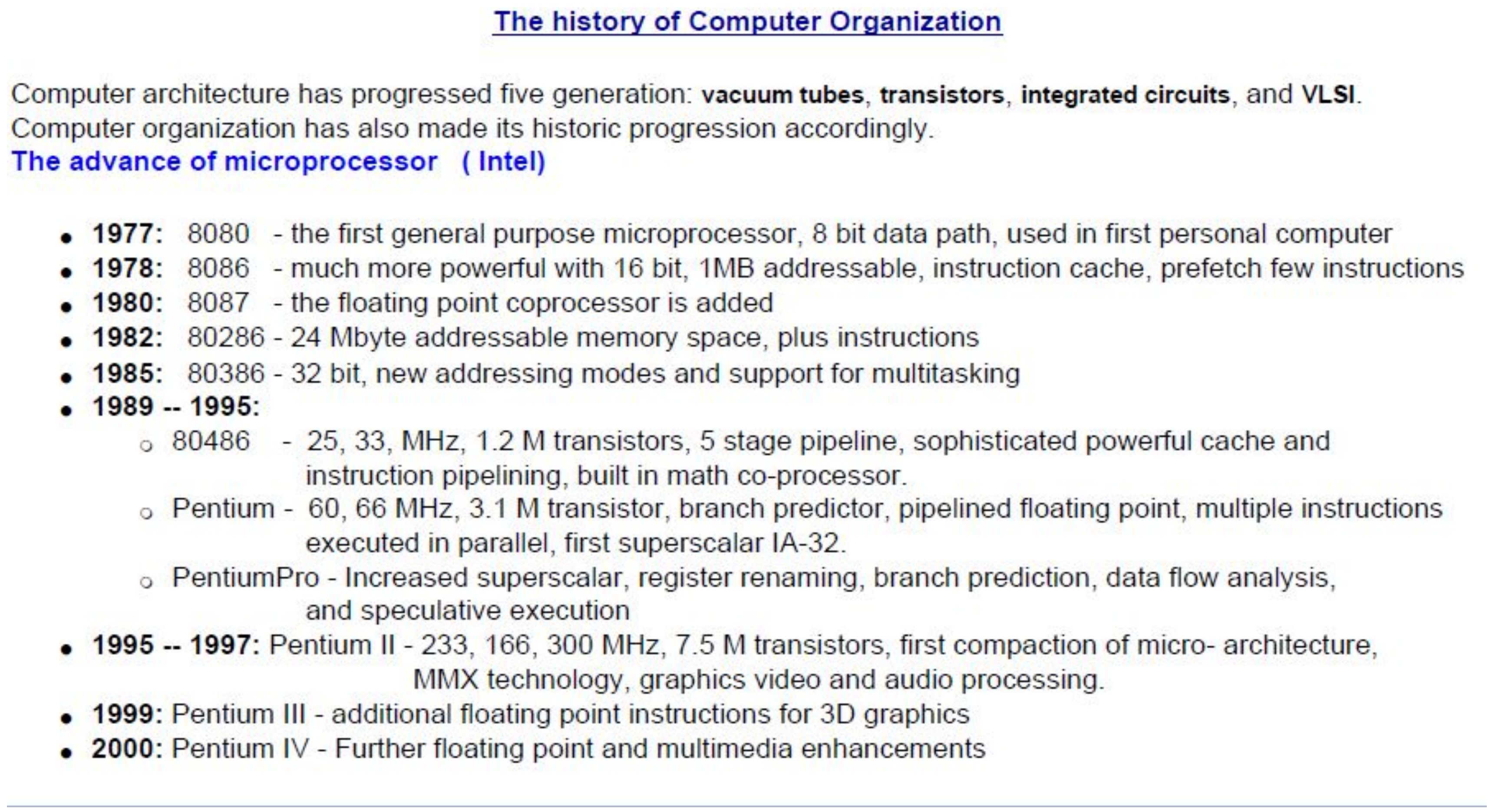
If **computer architecture** is a view of the *whole design with the important characteristics* visible to programmer, **omputer organization** is *how features are implemented with the specific building blocks* visible to designer, such as ontrol signals, interfaces, memory technology, etc. Computer architecture and organization are closely related, hough not exactly the same.

**A stored program computer has the following basic units:**

- **Processor** -- center for manipulation and control
- **Memory** -- storage for instructions and data for currently executing programs
- **I/O system** -- controller which communicate with "external" devices:
  secondary memory, display devices, networks
- **Data-path & control** -- collection of parallel wires, transmits data, instructions, or control signal

Computer organization defines the ways in which these components are interconnected and controlled. It is the apabilities and performance characteristics of those principal functional units. Architecture can have a number of rganizational implementations, and organization differs between different versions. Such, all **Intel x86** families share he same basic architecture, and **IBM system/370** family share their basic architecture.

# Brief History of Computer Evolution



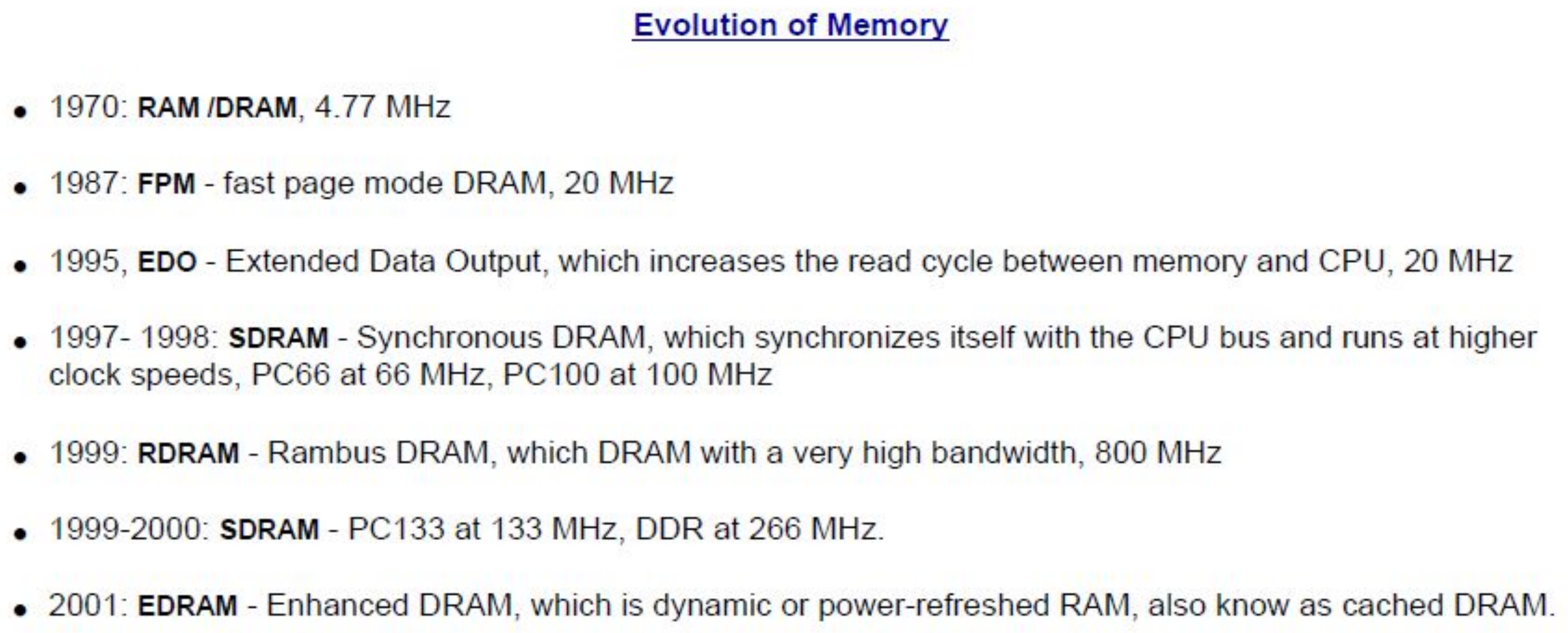
**The history of Computer Organization**

Computer architecture has progressed five generation: **vacuum tubes**, **transistors**, **integrated circuits**, and **VLSI**. Computer organization has also made its historic progression accordingly.

**The advance of microprocessor ( Intel)**

- **1977:** 8080 - the first general purpose microprocessor, 8 bit data path, used in first personal computer
- **1978:** 8086 - much more powerful with 16 bit, 1MB addressable, instruction cache, prefetch few instructions
- **1980:** 8087 - the floating point coprocessor is added
- **1982:** 80286 - 24 Mbyte addressable memory space, plus instructions
- **1985:** 80386 - 32 bit, new addressing modes and support for multitasking
- **1989 -- 1995:**
  - 80486 - 25, 33, MHz, 1.2 M transistors, 5 stage pipeline, sophisticated powerful cache and instruction pipelining, built in math co-processor.
  - Pentium - 60, 66 MHz, 3.1 M transistor, branch predictor, pipelined floating point, multiple instructions executed in parallel, first superscalar IA-32.
  - PentiumPro - Increased superscalar, register renaming, branch prediction, data flow analysis, and speculative execution
- **1995 -- 1997:** Pentium II - 233, 166, 300 MHz, 7.5 M transistors, first compaction of micro- architecture, MMX technology, graphics video and audio processing.
- **1999:** Pentium III - additional floating point instructions for 3D graphics
- **2000:** Pentium IV - Further floating point and multimedia enhancements

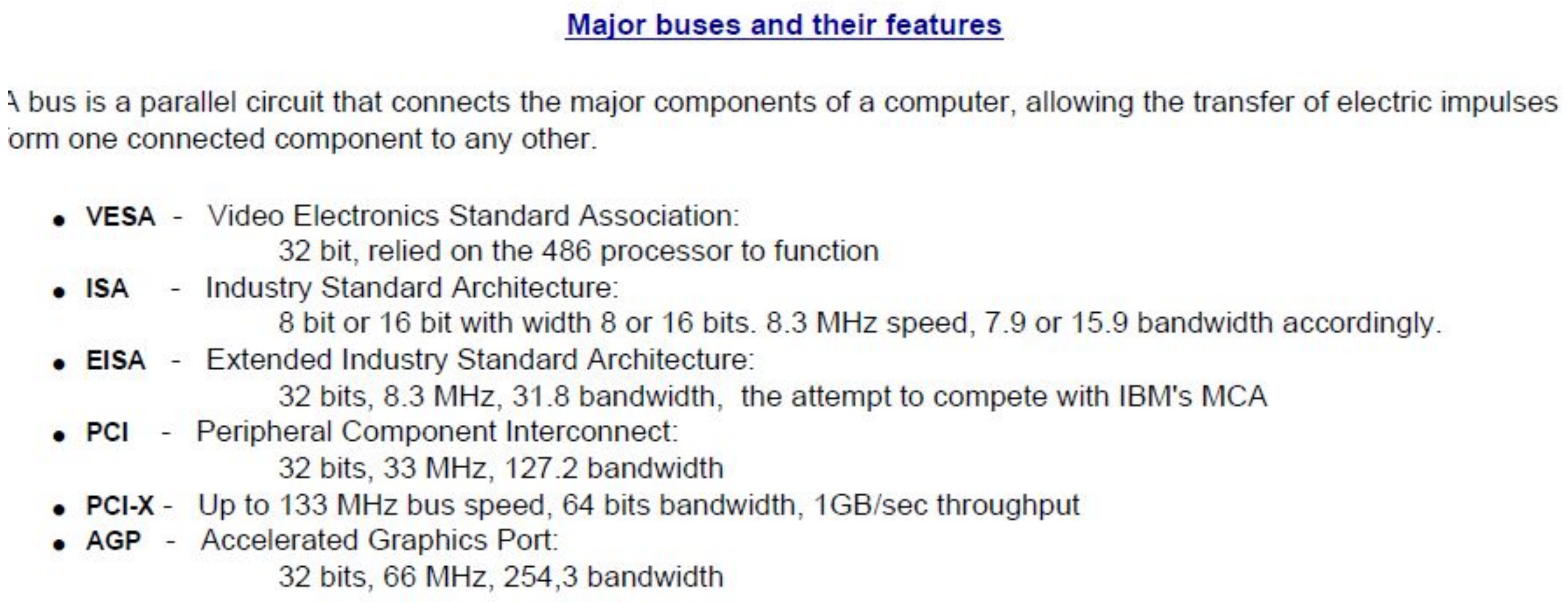
## Brief History of Computer Evolution



### Evolution of Memory

- 1970: **RAM /DRAM**, 4.77 MHz
- 1987: **FPM** - fast page mode DRAM, 20 MHz
- 1995, **EDO** - Extended Data Output, which increases the read cycle between memory and CPU, 20 MHz
- 1997- 1998: **SDRAM** - Synchronous DRAM, which synchronizes itself with the CPU bus and runs at higher clock speeds, PC66 at 66 MHz, PC100 at 100 MHz
- 1999: **RDRAM** - Rambus DRAM, which DRAM with a very high bandwidth, 800 MHz
- 1999-2000: **SDRAM** - PC133 at 133 MHz, DDR at 266 MHz.
- 2001: **EDRAM** - Enhanced DRAM, which is dynamic or power-refreshed RAM, also know as cached DRAM.

# Brief History of Computer Evolution

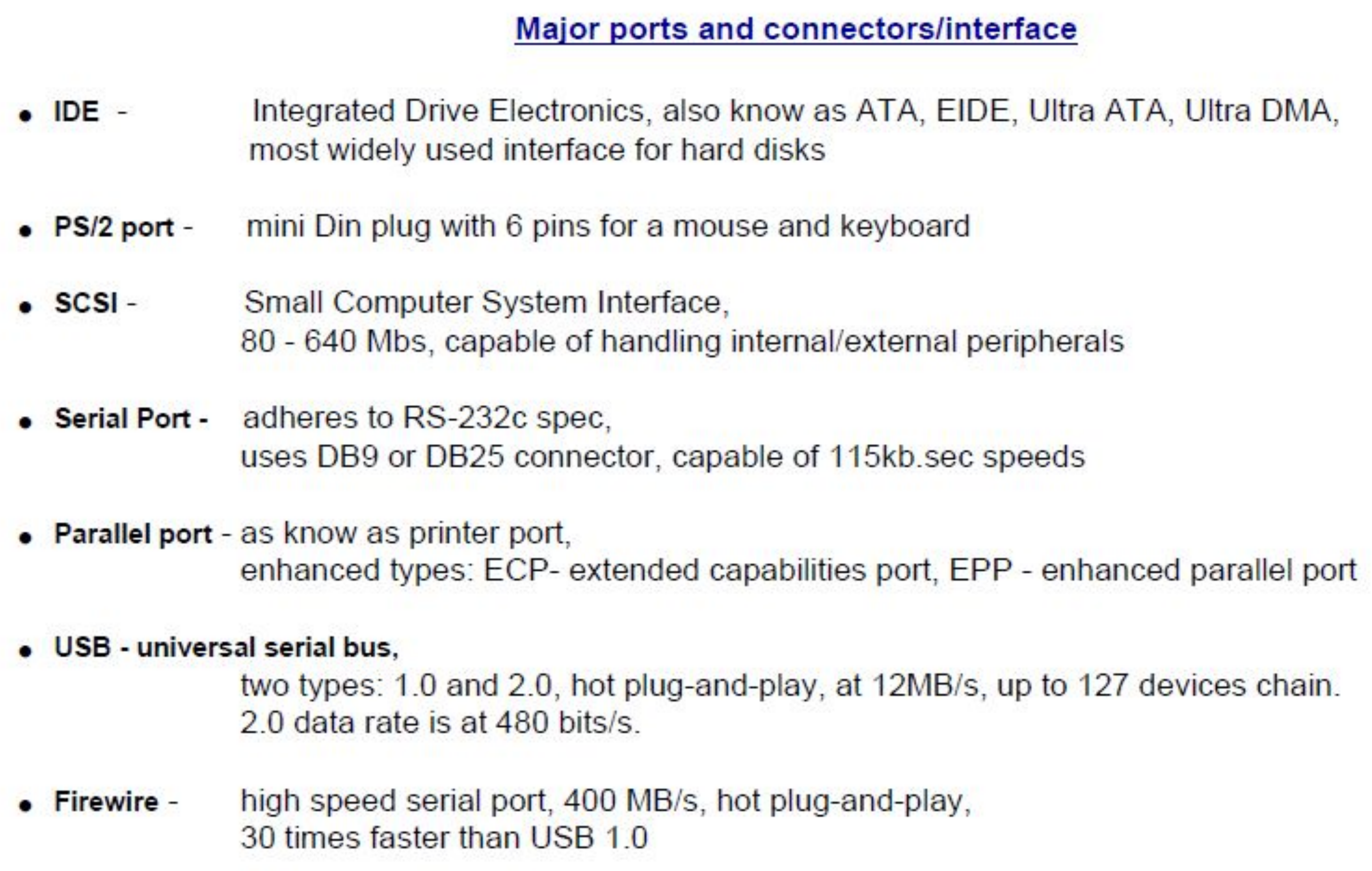


**Major buses and their features**

A bus is a parallel circuit that connects the major components of a computer, allowing the transfer of electric impulses orm one connected component to any other.

- **VESA** - Video Electronics Standard Association:
  32 bit, relied on the 486 processor to function
- **ISA** - Industry Standard Architecture:
  8 bit or 16 bit with width 8 or 16 bits. 8.3 MHz speed, 7.9 or 15.9 bandwidth accordingly.
- **EISA** - Extended Industry Standard Architecture:
  32 bits, 8.3 MHz, 31.8 bandwidth, the attempt to compete with IBM's MCA
- **PCI** - Peripheral Component Interconnect:
  32 bits, 33 MHz, 127.2 bandwidth
- **PCI-X** - Up to 133 MHz bus speed, 64 bits bandwidth, 1GB/sec throughput
- **AGP** - Accelerated Graphics Port:
  32 bits, 66 MHz, 254,3 bandwidth

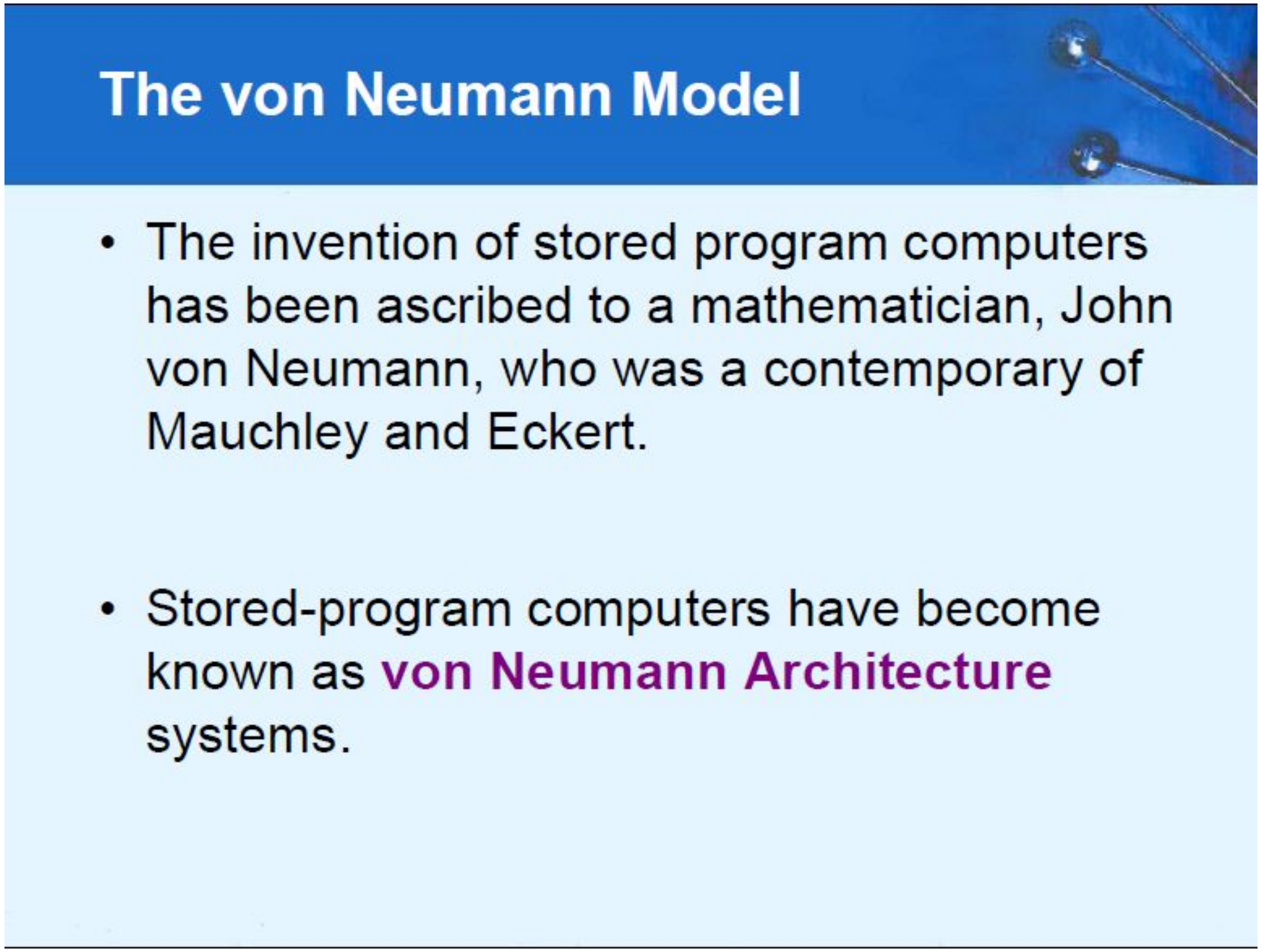
# Brief History of Computer Evolution



**Major ports and connectors/interface**

- **IDE** - Integrated Drive Electronics, also know as ATA, EIDE, Ultra ATA, Ultra DMA, most widely used interface for hard disks
- **PS/2 port** - mini Din plug with 6 pins for a mouse and keyboard
- **SCSI** - Small Computer System Interface, 80 - 640 Mbs, capable of handling internal/external peripherals
- **Serial Port -** adheres to RS-232c spec, uses DB9 or DB25 connector, capable of 115kb.sec speeds
- **Parallel port** - as know as printer port, enhanced types: ECP- extended capabilities port, EPP - enhanced parallel port
- **USB - universal serial bus,** two types: 1.0 and 2.0, hot plug-and-play, at 12MB/s, up to 127 devices chain. 2.0 data rate is at 480 bits/s.
- **Firewire** - high speed serial port, 400 MB/s, hot plug-and-play, 30 times faster than USB 1.0

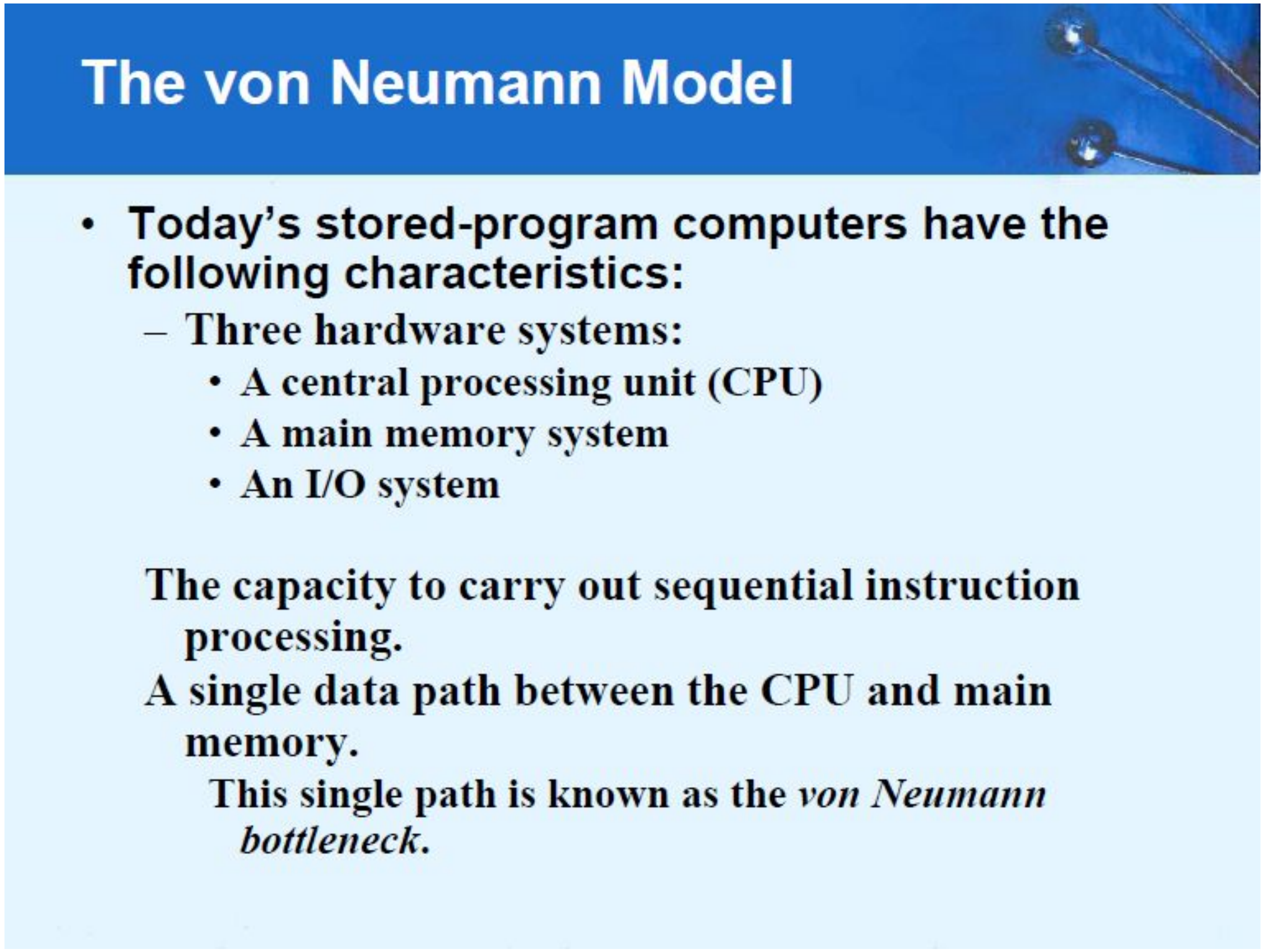
# The von Neumann Modal



## The von Neumann Model

- The invention of stored program computers has been ascribed to a mathematician, John von Neumann, who was a contemporary of Mauchley and Eckert.
- Stored-program computers have become known as **von Neumann Architecture** systems.

# The von Neumann Modal



## The von Neumann Model

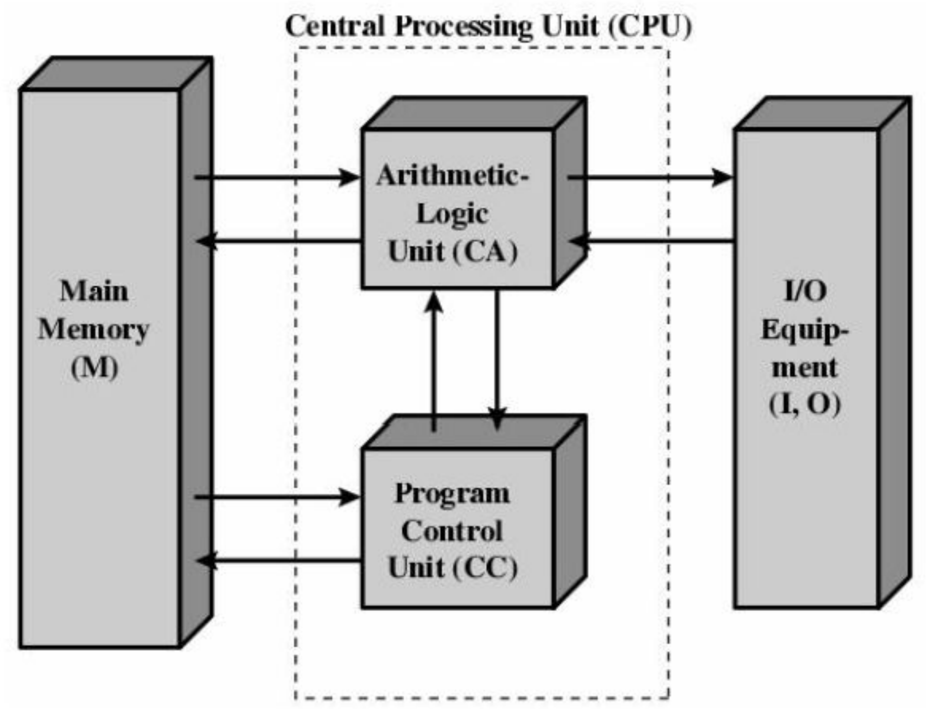
- Today's stored-program computers have the following characteristics:
  - Three hardware systems:
    - A central processing unit (CPU)
    - A main memory system
    - An I/O system

The capacity to carry out sequential instruction processing.

A single data path between the CPU and main memory.

This single path is known as the *von Neumann bottleneck*.

# The von Neumann Modal

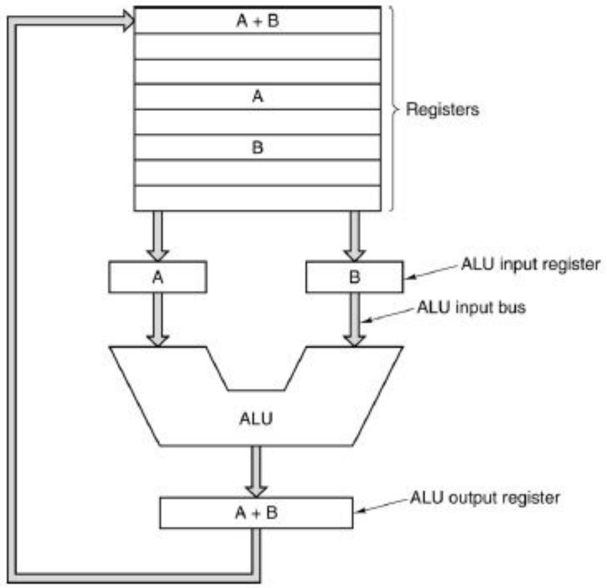


Central Processing Unit (CPU)
Main Memory (M)
Arithmetic-Logic Unit (CA)
Program Control Unit (CC)
I/O Equip-ment (I, O)

IAS (Princeton) computer model by Von Neumann's group.

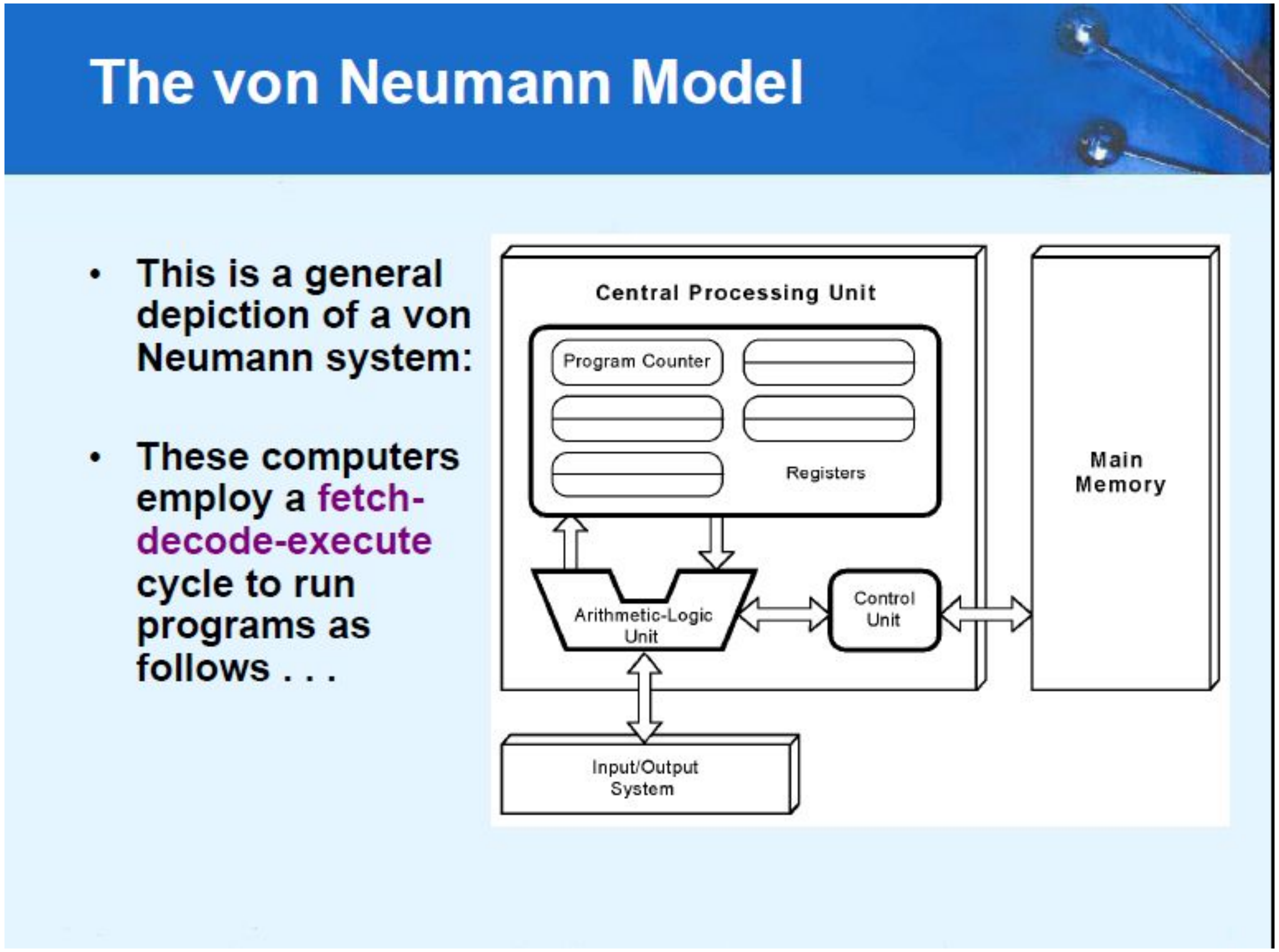
# The von Neumann Modal

## CPU Organization



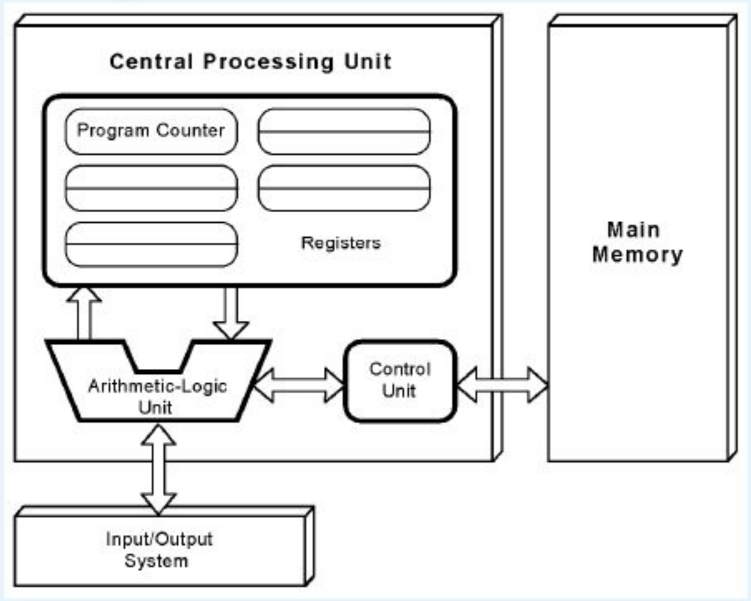
The data path of a typical Von Neumann machine.

# The von Neumann Modal



## The von Neumann Model

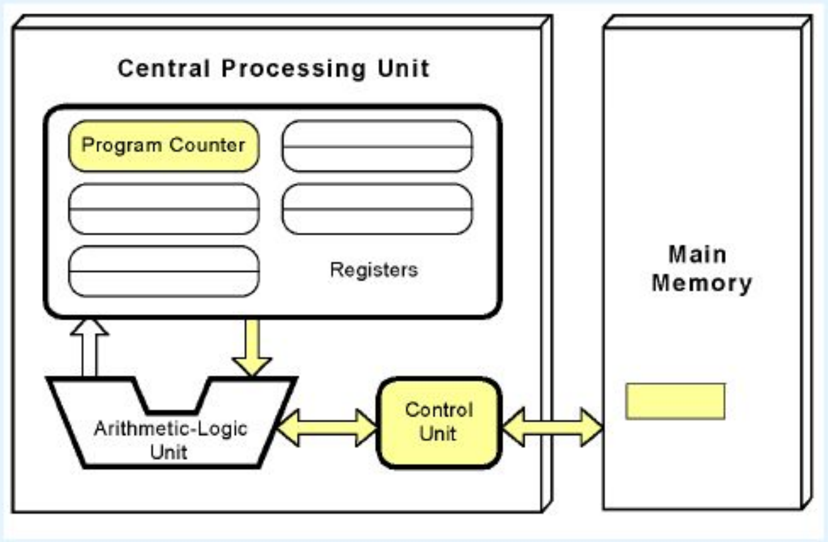
- This is a general depiction of a von Neumann system:
- These computers employ a fetch-decode-execute cycle to run programs as follows . . .



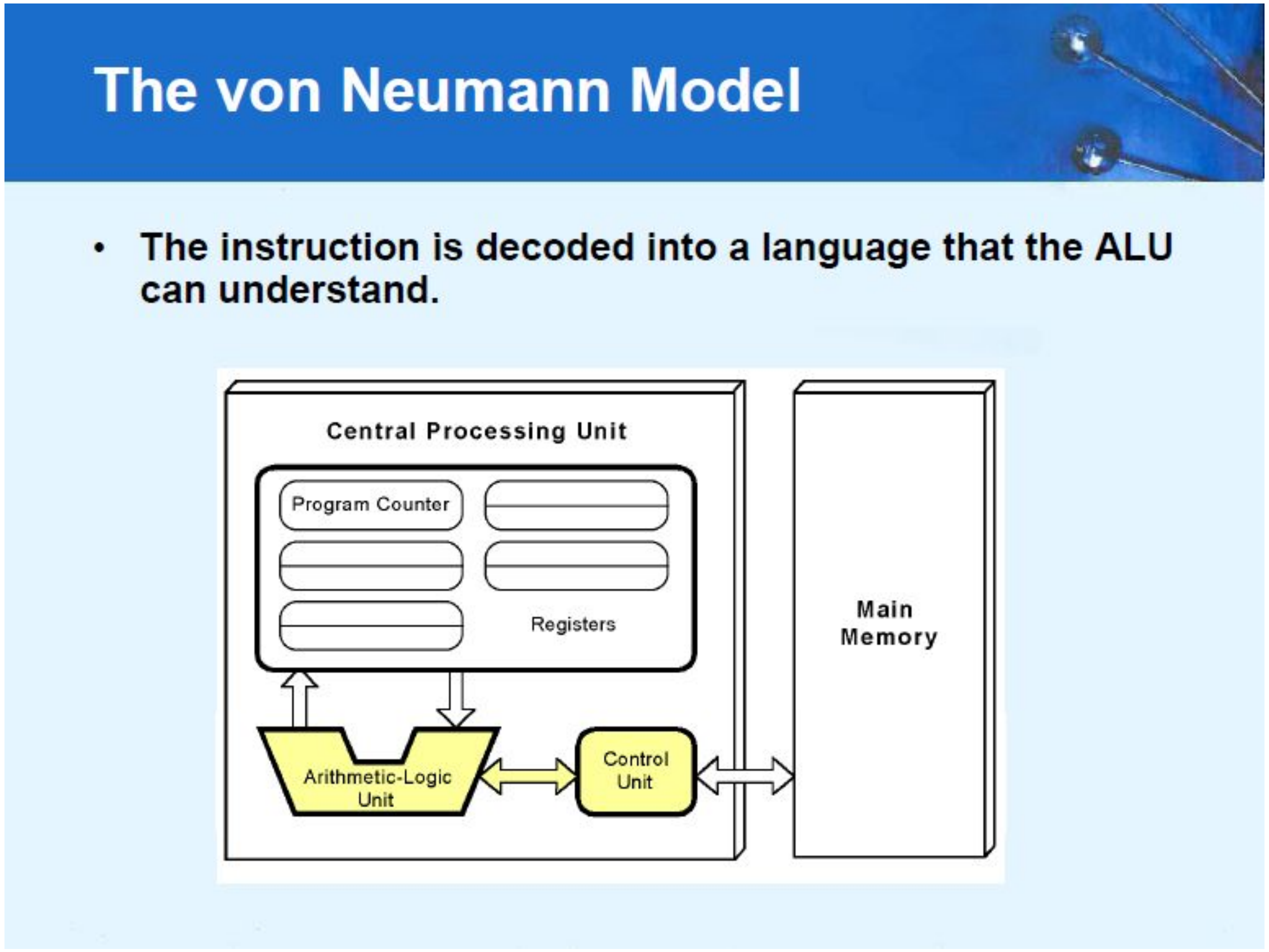
# The von Neumann Modal

## The von Neumann Model

- The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.

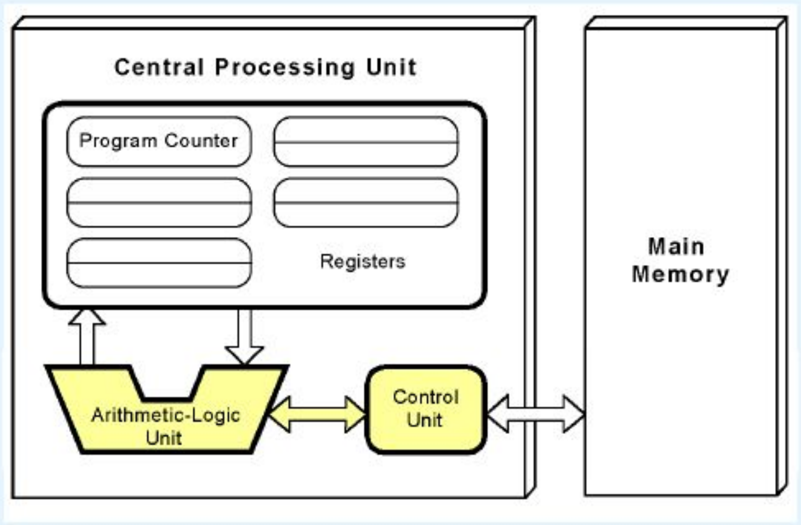


# The von Neumann Modal

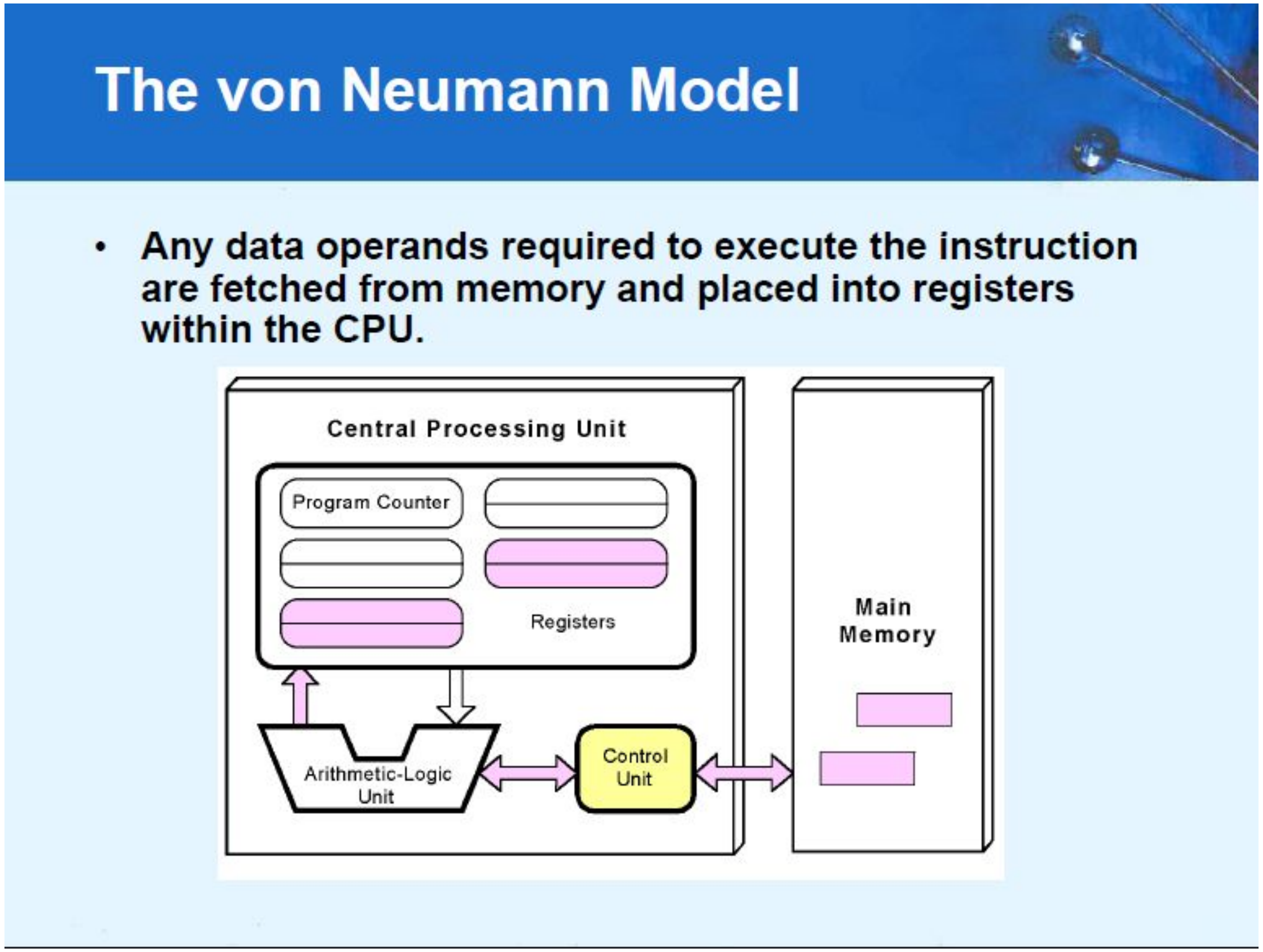


## The von Neumann Model

- The instruction is decoded into a language that the ALU can understand.

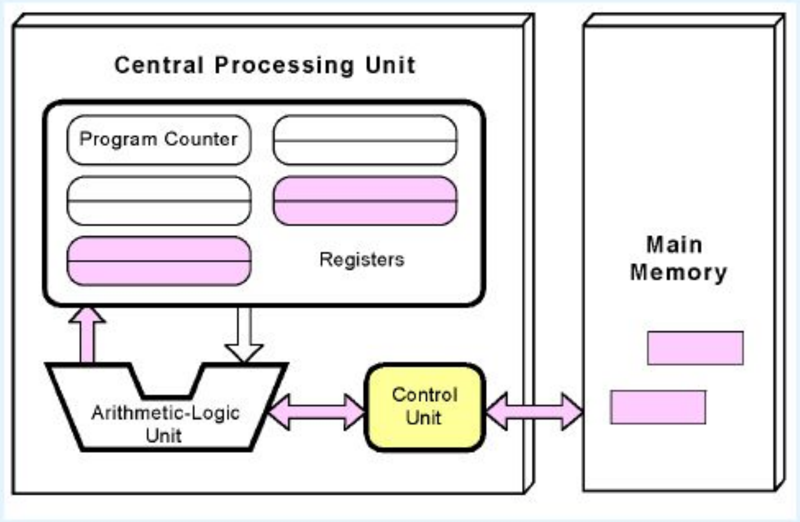


# The von Neumann Modal

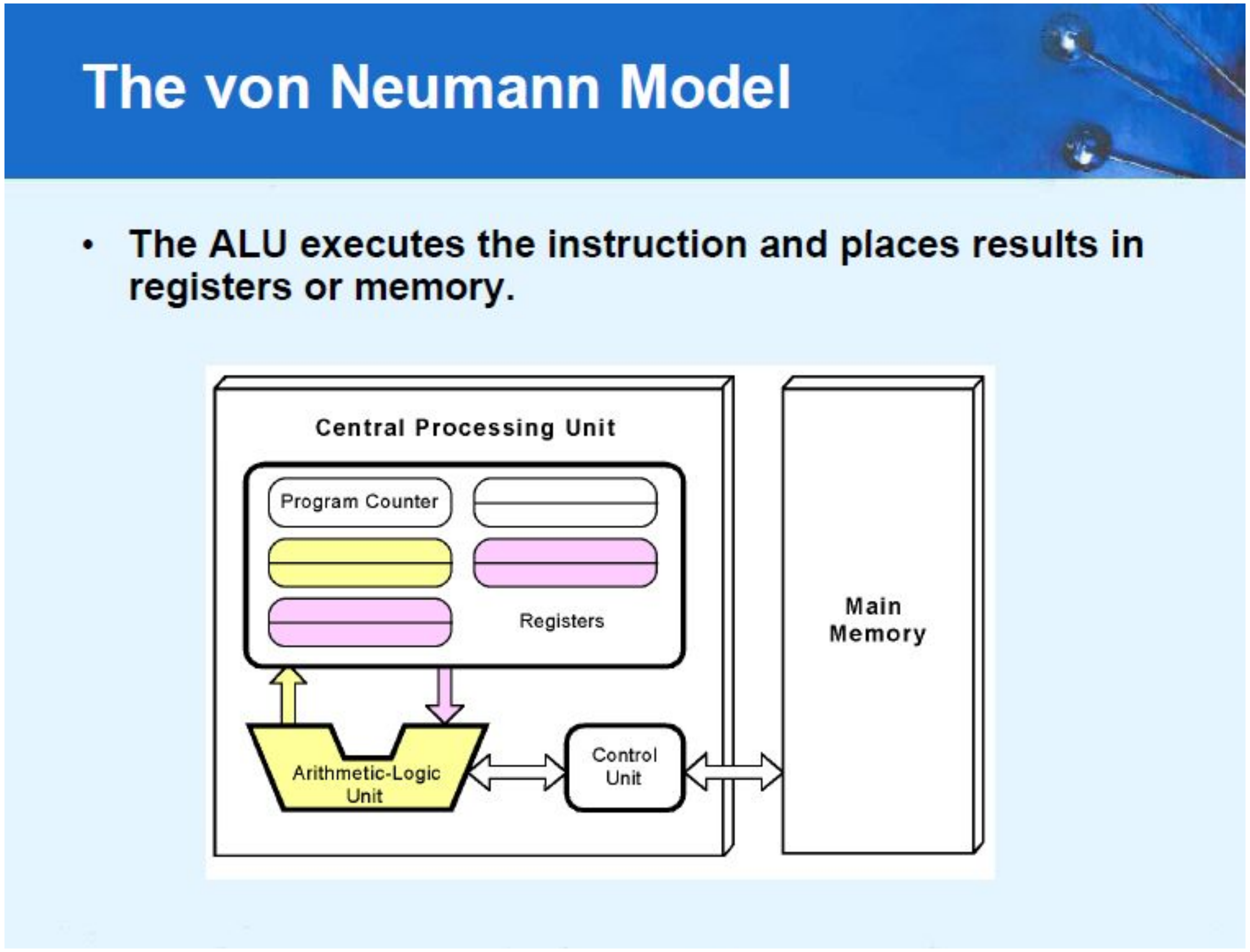


## The von Neumann Model

- Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.

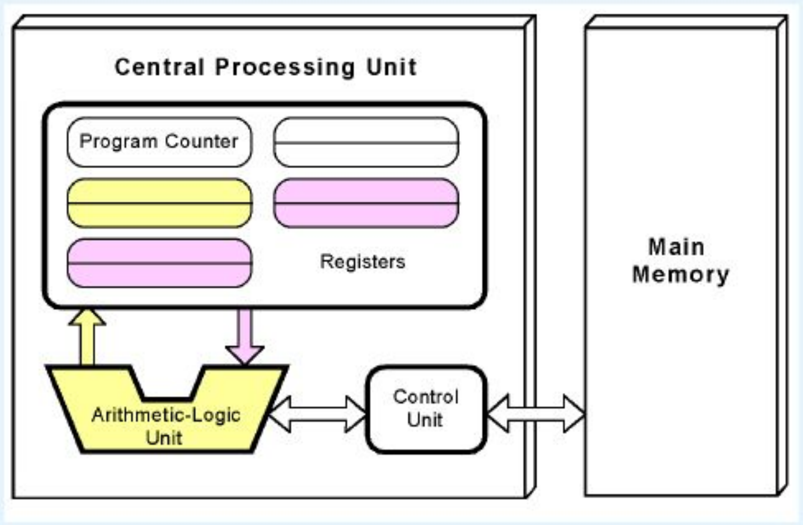


# The von Neumann Modal

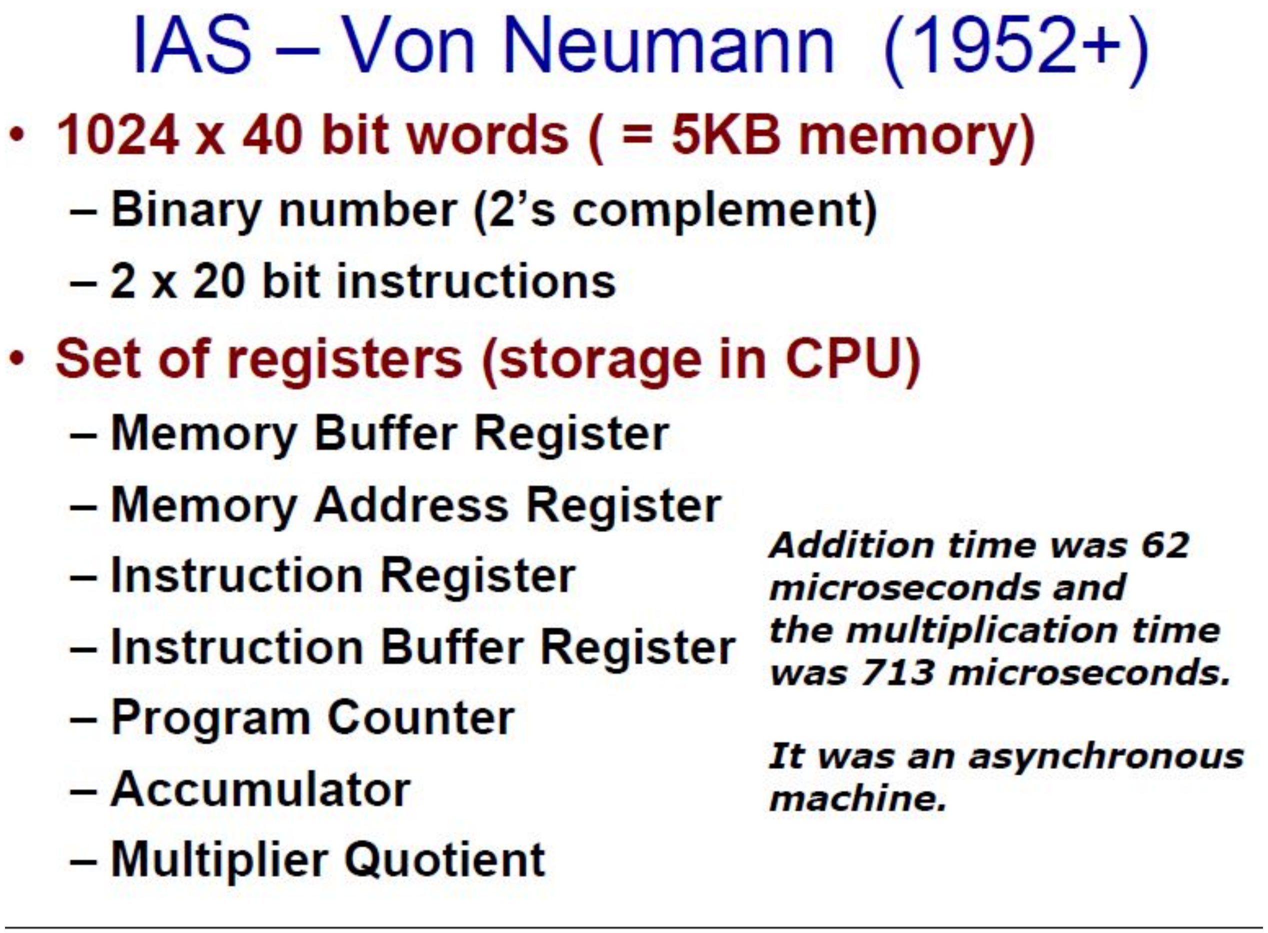


## The von Neumann Model

- The ALU executes the instruction and places results in registers or memory.



# IAS-Von Neumann Model



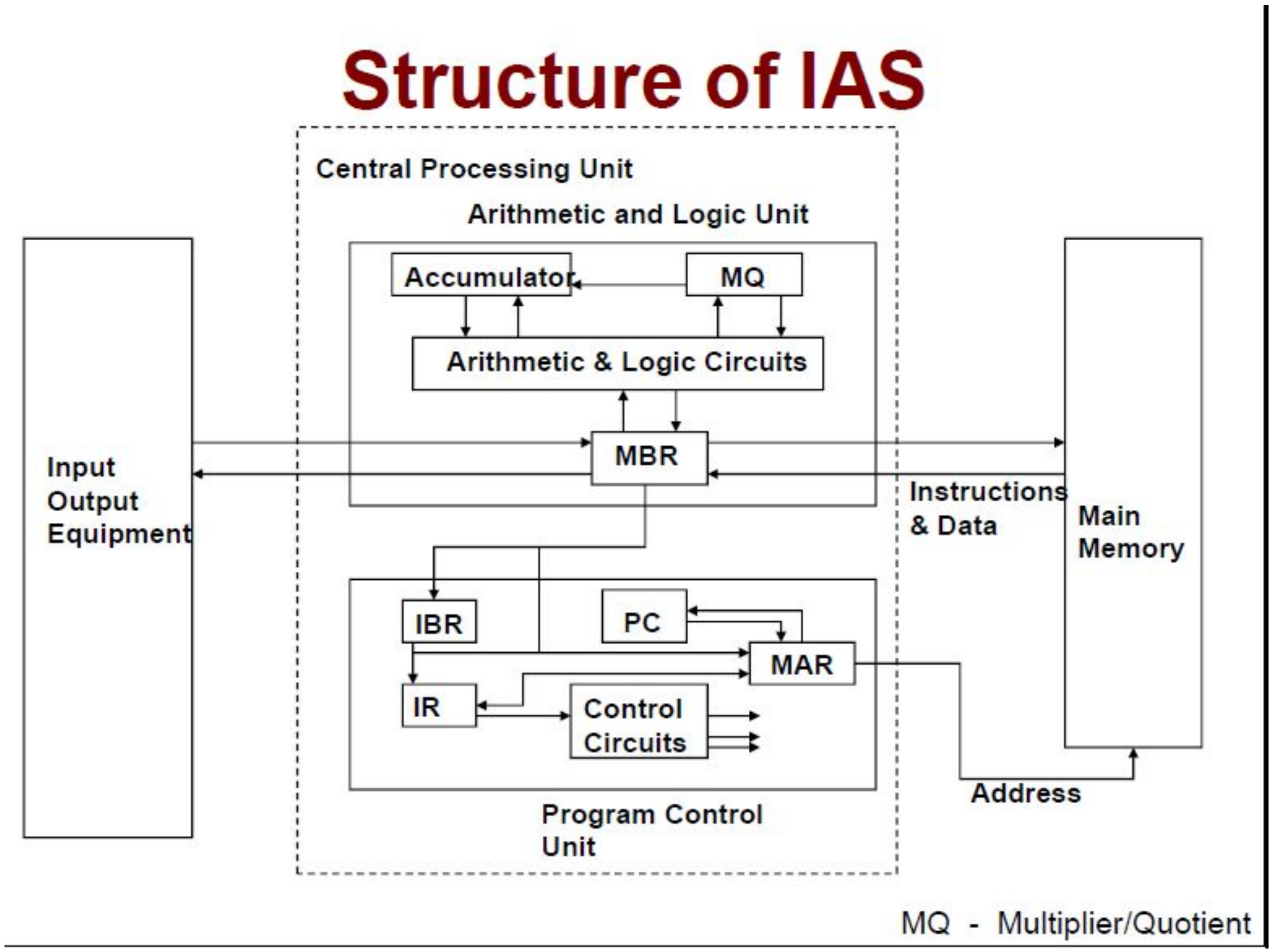
## IAS – Von Neumann (1952+)

- **1024 x 40 bit words ( = 5KB memory)**
  - Binary number (2's complement)
  - 2 x 20 bit instructions
- **Set of registers (storage in CPU)**
  - Memory Buffer Register
  - Memory Address Register
  - Instruction Register
  - Instruction Buffer Register
  - Program Counter
  - Accumulator
  - Multiplier Quotient

***Addition time was 62 microseconds and the multiplication time was 713 microseconds.***

***It was an asynchronous machine.***

# IAS-Von Neumann Model



# IAS-Von Neumann Model

**IAS computer consists of:**

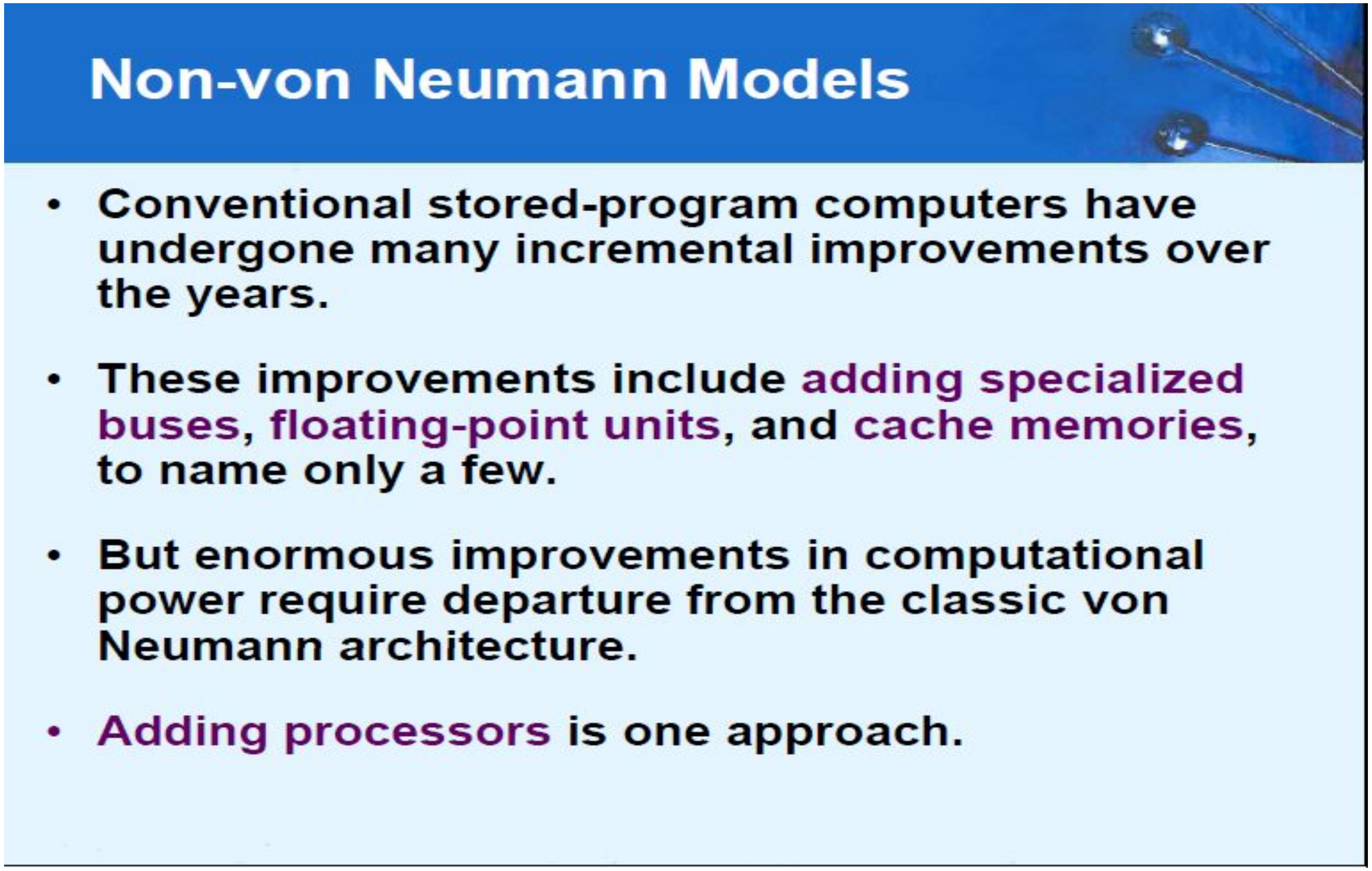
**- A main memory, which stores both data and instructions.**

**- An arithmetic-logical unit (ALU) capable of operating on binary data.**

**- A control unit, which interprets the instructions in memory and causes them to be executed.**

**- Input and output (I/O) equipment operated by the control unit.**

# Alternative Architecture

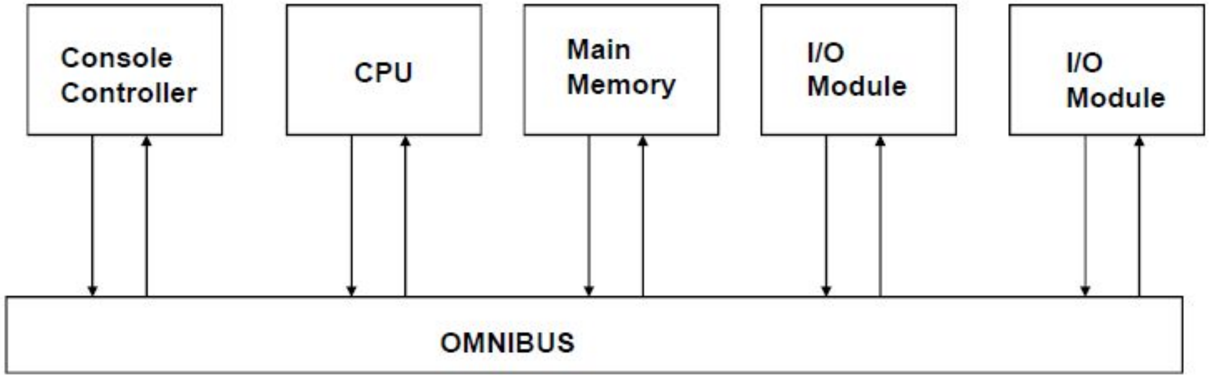


## Non-von Neumann Models

- Conventional stored-program computers have undergone many incremental improvements over the years.
- These improvements include adding specialized buses, floating-point units, and cache memories, to name only a few.
- But enormous improvements in computational power require departure from the classic von Neumann architecture.
- Adding processors is one approach.

## Alternative Architecture

### DEC - PDP-8 Bus Structure



The Omnibus - a backplane of undedicated slots;

## Architecture vs. Organization

## Computer Architecture

**Logical aspects of system implementation as seen by the programmer; such as, instruction sets (ISA) and formats, opcode, data types, addressing modes and I/O.**

**Instruction set architecture (ISA) is different from "microarchitecture", which consist of various processor design techniques used to implement the instruction set.**

**Computers with different microarchitectures can share a common instruction set.**

**For example, the Intel Pentium and the AMD Athlon implement nearly identical versions of the x86 instruction set, but have radically different internal designs.**

Computer architecture is the conceptual design and fundamental operational structure of a computer system. It is a functional description of requirements and design implementations for the various parts of a computer.

It is the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals.

It deals with the architectural attributes like physical address memory, CPU and how they should be designed and made to coordinate with each other keeping the goals in mind.

Analogy: "building the design and architecture of house" – architecture may take more time due to planning and then organization is building house by bricks or by latest technology keeping the basic layout and architecture of house in mind.

**Computer architecture comes before computer organization.**

**Computer organization (CO) is how operational attributes are linked together and contribute to realise the architectural specifications.**

**CO encompasses all physical aspects of computer systems**

**e.g. Circuit design, control signals, memory types.**

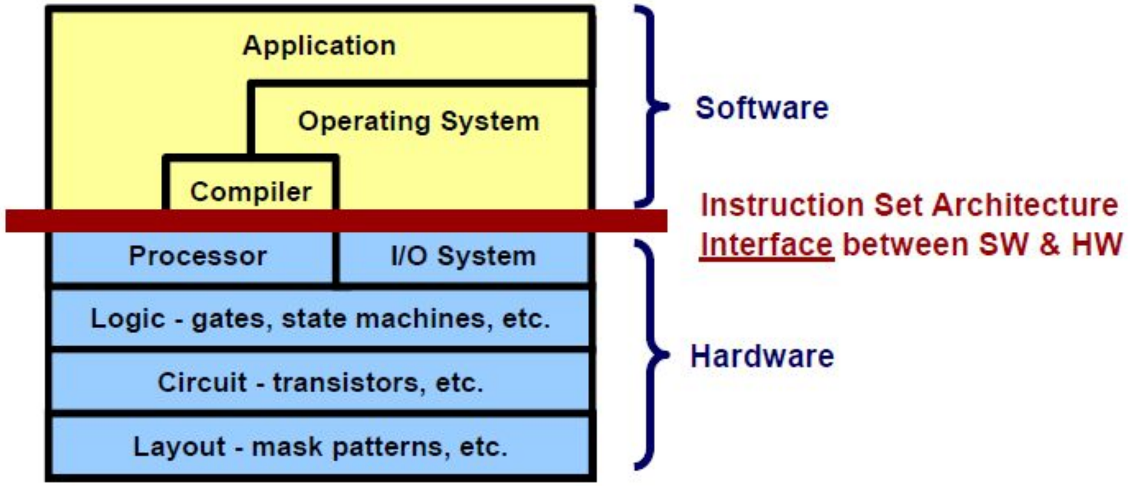
**Microarchitecture, also known as Computer organization is a lower level, more concrete and detailed, description of the system that involves how the constituent parts of the system are interconnected and how they interoperate in order to implement the ISA.**

**The size of a computer's cache, for example, is an organizational issue that generally has nothing to do with the ISA.**

**Another example: it is an architectural design issue whether a computer will have a multiply instruction. It is an organizational issue whether that instruction will be implemented by a special multiply unit or by a mechanism that makes repeated use of the add unit of the system.**

## Instruction Set Architecture (ISA) - The Hardware-Software Interface

- The **most important** abstraction of computer design


Application
Operating System
Compiler
Software
Instruction Set Architecture
Interface between SW & HW
Processor
I/O System
Logic - gates, state machines, etc.
Circuit - transistors, etc.
Layout - mask patterns, etc.
Hardware

# Thank you

????...