Programming in C

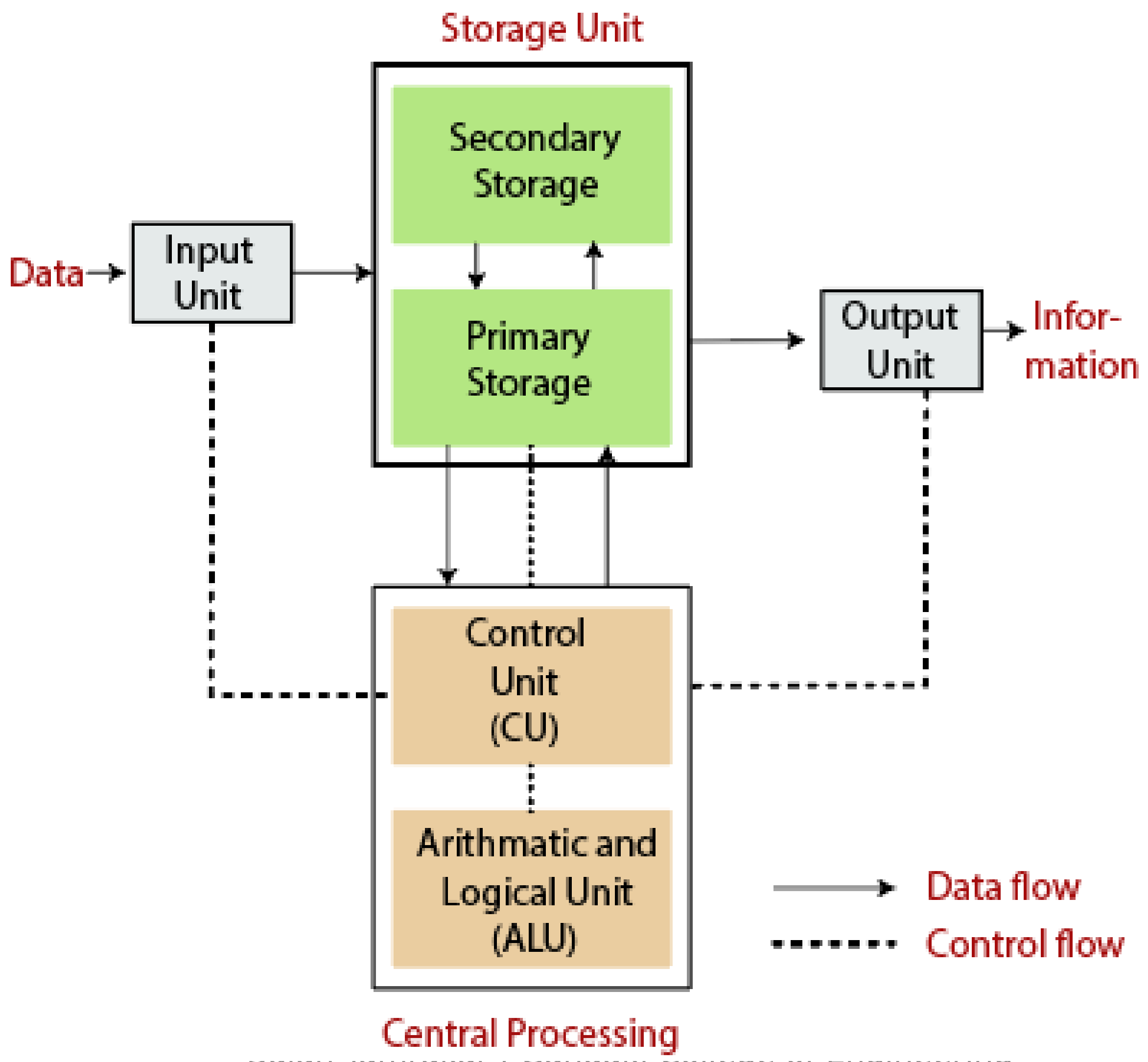
Sujan Tamrakar

Introduction

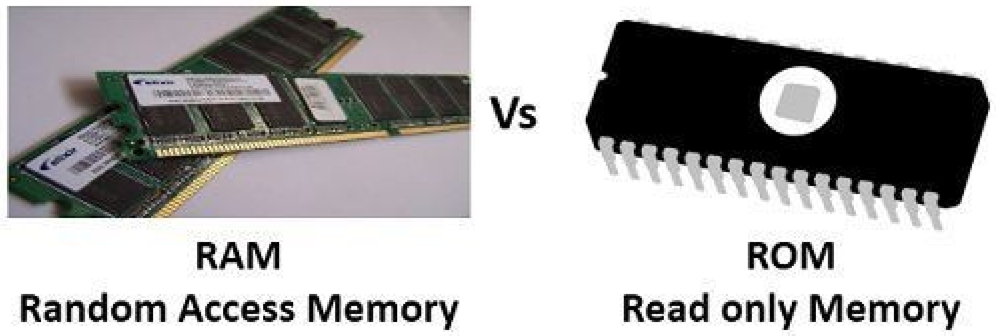
Introduction to Computer

- A computer is a programmable electronic device that accepts data and instruction from input devices, processes the data and provides result as information in the output.
- It works on the principle of IPOS cycle (Input – Process – Output – Storage).
- It is made up of both hardware and software components.
  - Hardware consists of the physical components of a computer system like input devices, output devices, processing device and storage device.
  - Software represents the set of programs and instructions that govern the operation of a computer system.

Block diagram of Computer



RAM v/s ROM

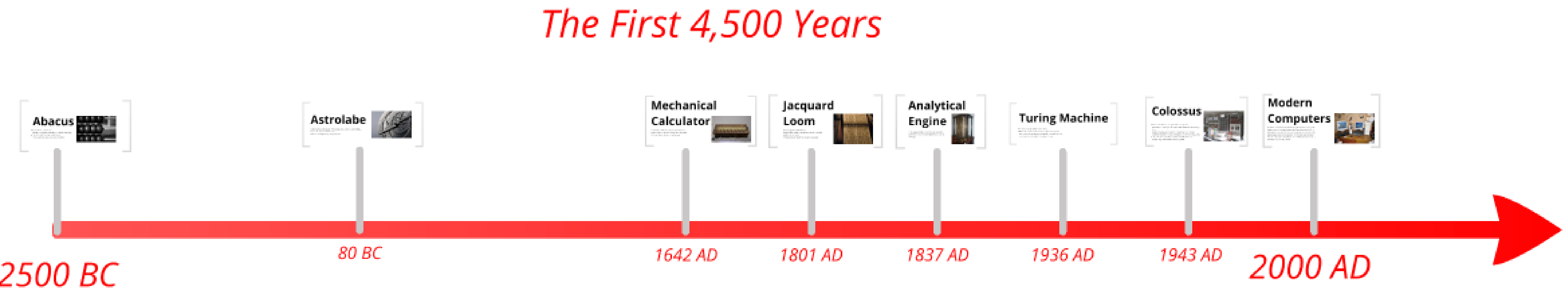


RAM	ROM
Random Access Memory	Read Only Memory
Temporary storage	Permanent storage
Volatile memory	Non-volatile memory
Read data quickly to run applications.	Stores the program required to boot the computer.
Allows reading & writing.	Allows reading (generally).
Comparatively expensive.	Comparatively cheap.
Types: Static RAM, Dynamic RAM	Types: PROM, EPROM, EEPROM

History of Computing

- Abacus was the very early counting tools approx. 3000 BC
- Calculators 1600s
- Punched card devices 1800s
- First electronic computers 1940s
- Mainframes 1950s
- Mini computers 1960s
- Micro computers 1970s
- Micro computer systems 1980s
- Internet 1990s
- Smartphone 1990s

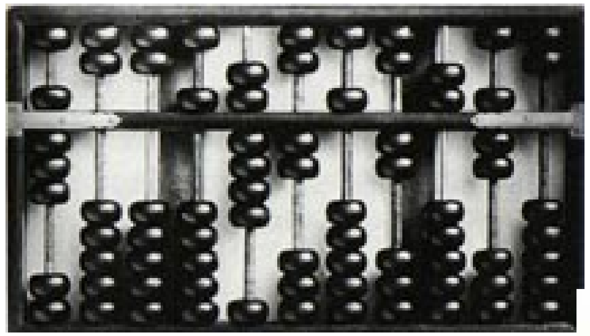
History of Computing

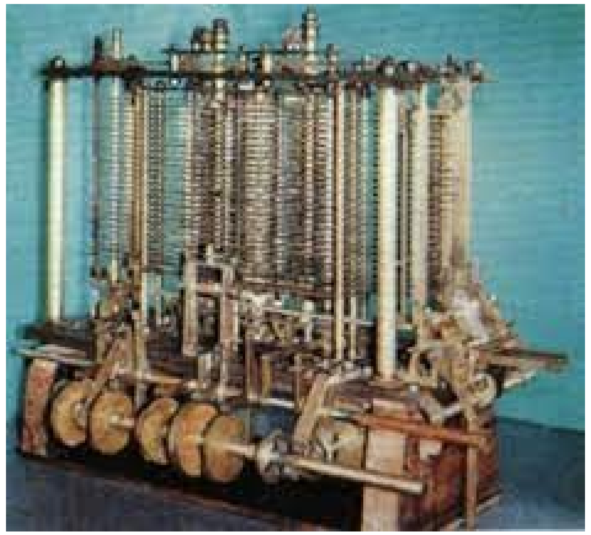


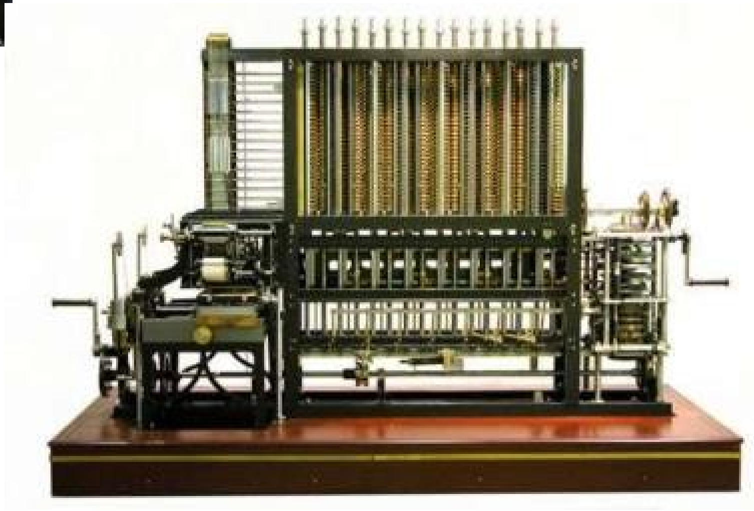
History of Computing

- Development history can be divided as:

1. Mechanical Era
    - Computers were developed using mechanical parts like wheel, liver, gears and shafts.
    - Eg. Abacus, Napier’s bone, Slide rule, Pascaline, Stepped reckoner, Jacquards loom, Difference engine, Analytical engine, Tabulating machine, etc.



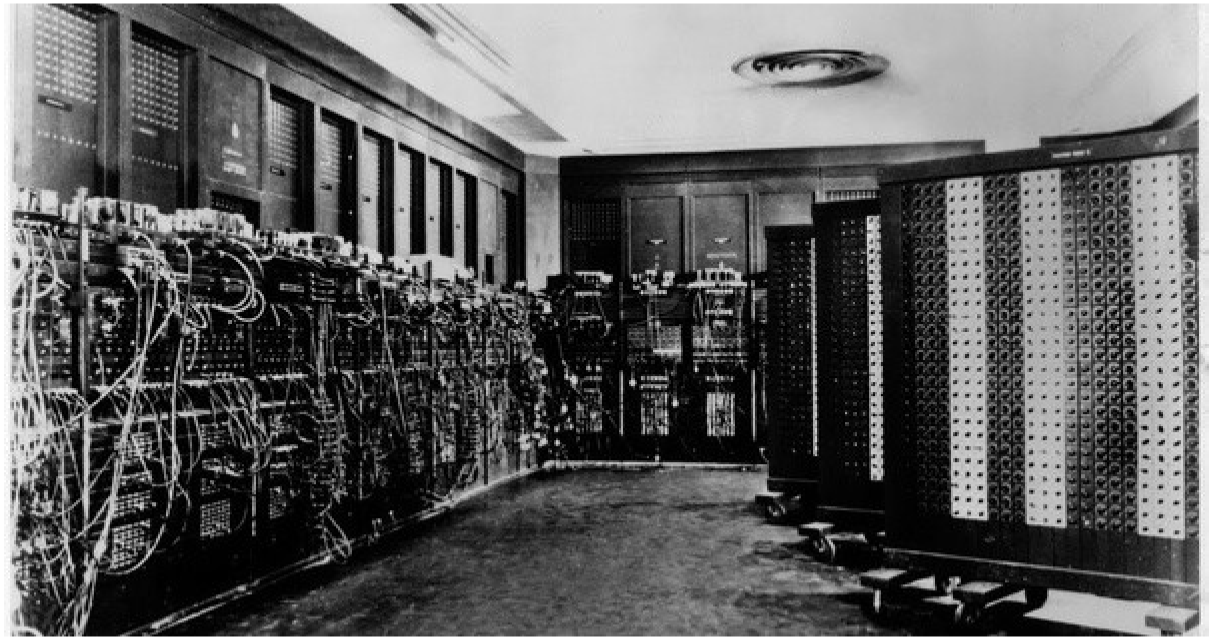




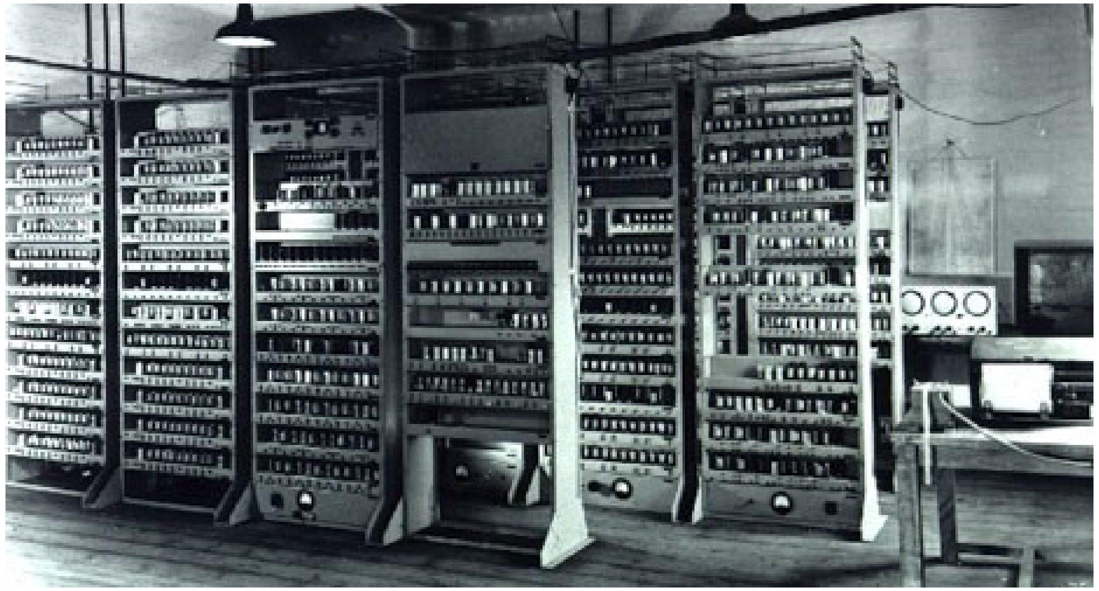
History of Computi

2. Electro-Mechanical Era
    - Such computers used electricity and were partly programmable.
    - Some of the parts consist of mechanical parts as well.
    - Ex. Mark-I by Howard Aiken, ABC, Zuse.

History of Computi



3. Electronic Era
   - Mechanical devices were replaced with electronic circuits.
   - Calculations were digital performed with electronic circuits.
   - Such computers were faster due to use of vacuum tubes, transistors, ICs.
   - Ex. ENIAC (Electronic Numerical Integrator and Computer), EDSAC (Electronic delay storage automatic calculator), EDVAC (Electronic Discrete Variable Automatic Computer), etc.



Generation of Computer

- It refers to the time period in which computers used the same technology.
- 5 generations:
  1. First Generation (1945-1956) – Vacuum tubes
  2. Second Generation (1957-1964) – Transistors
  3. Third Generation (1965-1974) – Integrated Circuits
  4. Fourth Generation (1975-1990) – Very Large Scale Integration
  5. Fifth Generation (1990 onward) – Ultra Large Scale Integration

Generation of Computer

1. First Generation (1945-1956) – Vacuum tubes

- Used Vacuum tubes for circuitry and Magnetic drum for storage
- Large in size, very slow speed, less accurate and consumed more power
- Used punched card as input and output were in print outs
- Used machine level language
- Solved single problem at a time
- Produced immense heat and were bulky
- Ex. UNIVAC, EDVAC, ENIAC

Generation of Computer

2. Second Generation (1957-1964) – Transistors

- Used transistors
- Comparatively smaller, faster and chea
- Used magnetic disk and tape for storag
- Assembly languages were used
- Ex. IBM 1620, PDP-I, ATLAS, etc.

Generation of Computer

3. Third Generation (1965-1974) – Integrated Circuits

- Used IC technology
- More reliable, smaller, easier to operate and cheaper
- Used semiconductor device as memory
- Required less power
- High level programming languages were
- Keyboard was introduced
- Ex. IBM 360 series, IBM 370 series.

Generation of Computer

4. Fourth Generation (1975-1990) – Very Large Scale Integration

- Used LSI, VLSI and microprocessor
- Portable and quite reliable
- Consumed less power
- Used GUI based OS
- Pointing device (mouse) was introduced
- Ex. IBM PC, Apple Macintosh, etc.

Generation of Computer

5. Fifth Generation (1990 onward) – Ultra Large Scale Integration

- Based on AI.
- Still in development phase.
- Offers ULSI technology.
- PCs are portable, smaller and handy.
- Embedded AI like Voice recognition syster Parallel processing, Supercomputing, Rob Game Playing, Expert system.
- Uses NLP with quantum computation and molecular technology.

Types of Computer

- On the basis of:
  - Operation
  - Size
  - Model
  - Brand

Note: You are suggested to make self note on these.

Types of Computer

- On the basis of Operation:

Types of Computer

- On the basis of Size:

Types of Computer

- On the basis of Model:

PS/2 computer

XT computer

AT computer

Types of Computer

- On the basis of Brand:

IBM PC

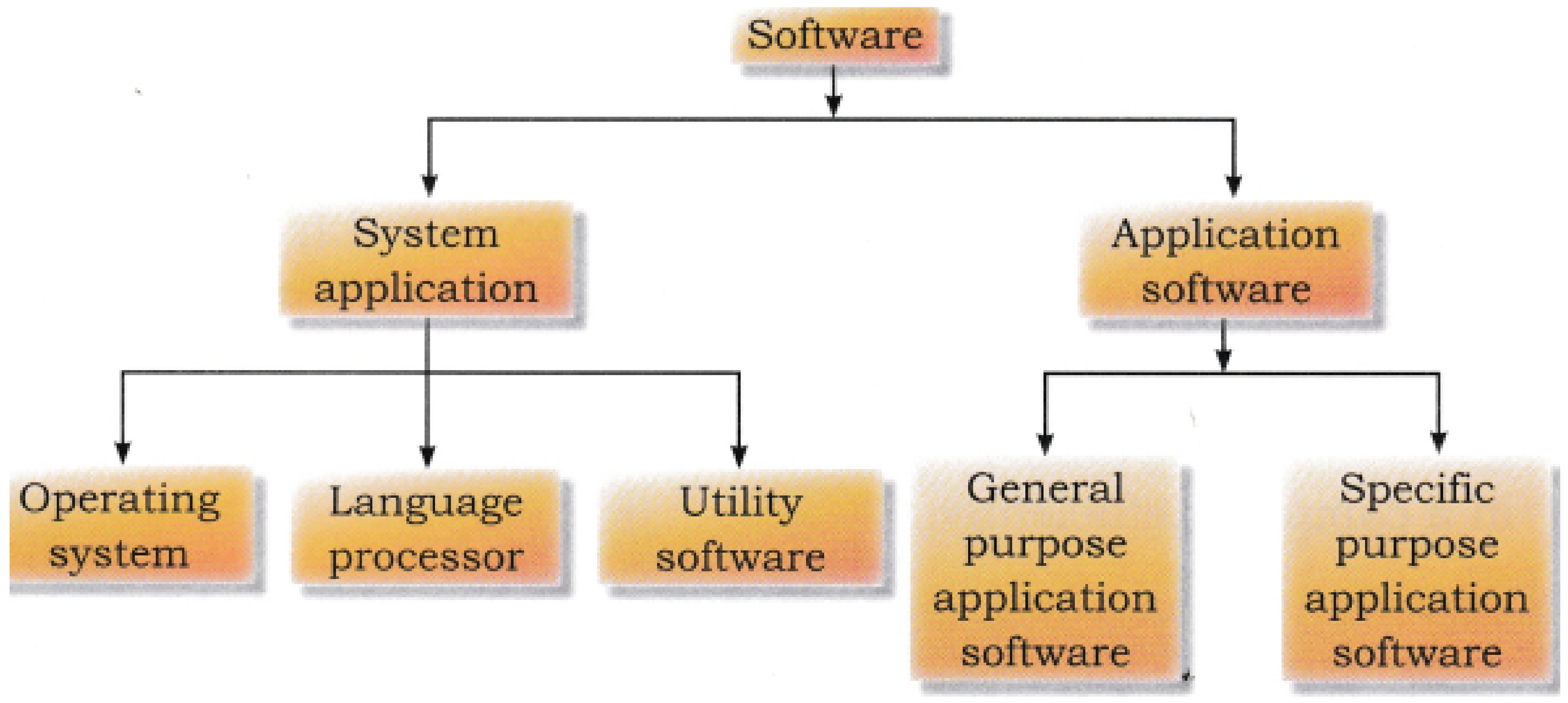
IBM Compatible

Apple / Macintosh

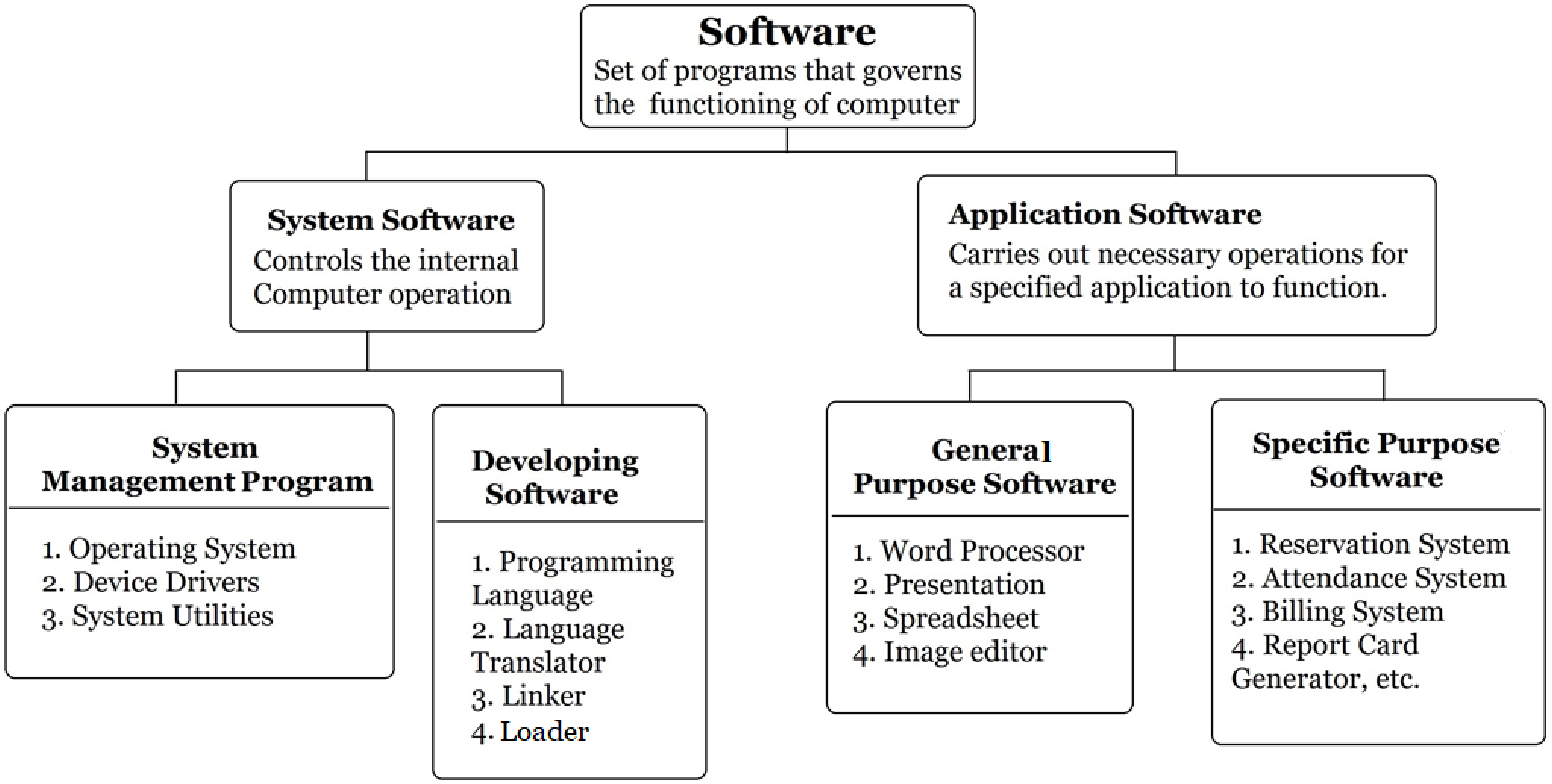
Types of Software

- A single instruction given to the computer to perform a single job is called ‘Command’.
- Collection of such commands in logical and sequential order is called ‘Program’.
- A set of computer programs to perform a specific task is called ‘Software’.

Types of Software



Types of Software



Generation of Programming Languages

- First Generation Programming Language (1GL)
  - Machine language (used 0s and 1s)
  - Machine dependent

- Second Generation Programming Language (2GL)
  - Assembly language
  - Used mnemonics
  - Machine dependent
  - Need of assembler as language translator

Generation of Programming Languages

- Third Generation Programming Language (3GL)
  - High level languages (FORTRAN, COBOL, BASIC, C, C++)
  - Procedural language

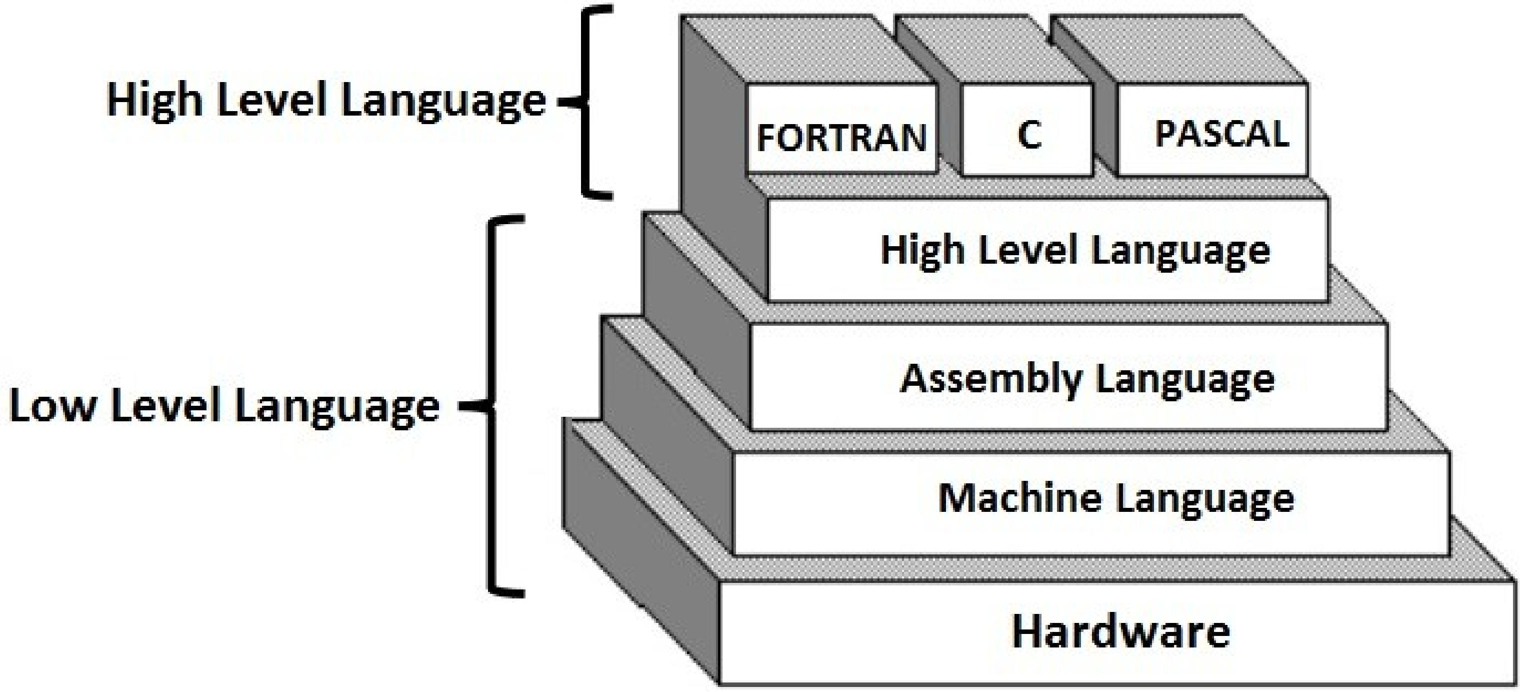
- Fourth Generation Programming Language (4GL)
  - Non-procedural language
  - User friendly and English like language
  - Ex: Query language like SQL.

- Fifth Generation Programming Language (5GL)
  - Based on Artificial Intelligence
  - Eg: Prolog, Mercury, P4.

Programming Language

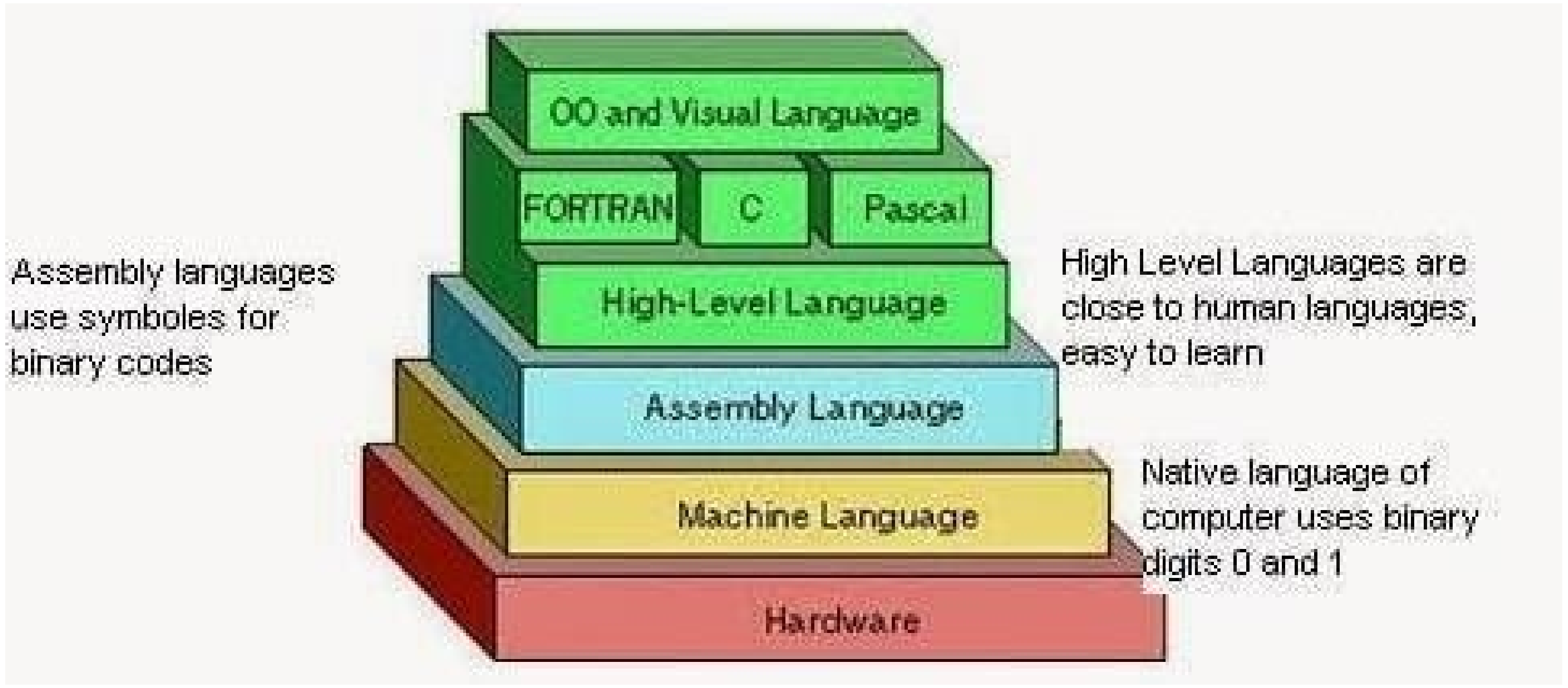
- The languages which are used to instruct the computer to perform certain tasks are called Programming languages (PL).
- Ex. BASIC, C, C++, Pascal, Fortran, Visual Basic, Java, Python, C#, etc.
- It is a standardized communication technique for describing instructions for a computer.
- Every PL has a set of rules used to define the program.
- It enables us to develop different kinds of software.
- It can be classified into 2 main categories.
  - Low level language
  - High Level language

Programming Language



Computer Language and its Types

Programming Language


OO and Visual Language
FORTRAN
C
Pascal
High-Level Language
Assembly Language
Machine Language
Hardware
Assembly languages use symboles for binary codes
High Level Languages are close to human languages, easy to learn
Native language of computer uses binary digits 0 and 1

Programming Language

<u>Low level language</u>

- Each instruction is directly translated into a single machine code.
- Much closer to the hardware.
- A sound knowledge of a hardware is necessary before writing a program for hardware.
- They are hardware specific (machine dependent).
- It’s types are:
  - Machine language
  - Assembly language

Programming Language

Low level language

- Machine language
    - The lowest level programming language
    - Computer understands the programs written only in this language
    - They consist of entire numbers of 0s and 1s (also called as bits)
    - Machine code may be different from machine to machine
    - Programs are faster and efficient
    - Writing programs is very tedious, time consuming and difficult to debug.

Programming Language

Low level language

- Assembly language
  - This overcomes the difficulties of Machine language programs
  - It uses symbols (ADD, MOV, COMP) in program instead of numbers only.
  - Such symbols are called ‘mnemonics’.
  - Writing a program was easier using Assembly language instead of writing in numbers only.
  - However, programs written using assembly language must be converted (done by assembler) to machine language to execute.
  - They are also machine dependent.

Programming Language

High level language

- Named as high level due to their syntax resembling close to human language.
- It makes writing program easier (read/write/understand) and fast.
- Most of the programs uses English language (words, symbols, mathematical notations) syntax.
- They have their own set of syntax(grammar/rules) to represent a set of instructions.
- Programs written in HLL must be translated to machine language (done by compiler or interpreter) to execute.
- Ex. C, C++, Java.
- It’s types are:
  - Procedural language
  - Object oriented language

Programming Language

High level language

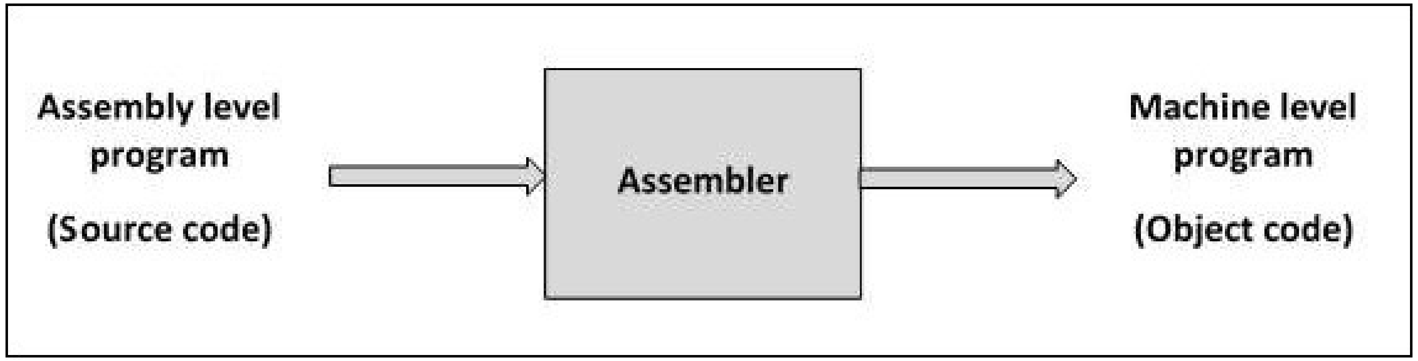
- Procedural language
  - Languages which are used to write procedures to perform the defined task.
  - Focus on control flow rather than the data.
  - Ex. C, Fortran, BASIC, etc.

- Object Oriented language
  - Languages that wraps the set of data and functions into a class and such properties are held by objects of those classes.
  - Focus on data (privacy & security).
  - Ex. C++, Java, etc.

Assembler, Compiler, Interpreter

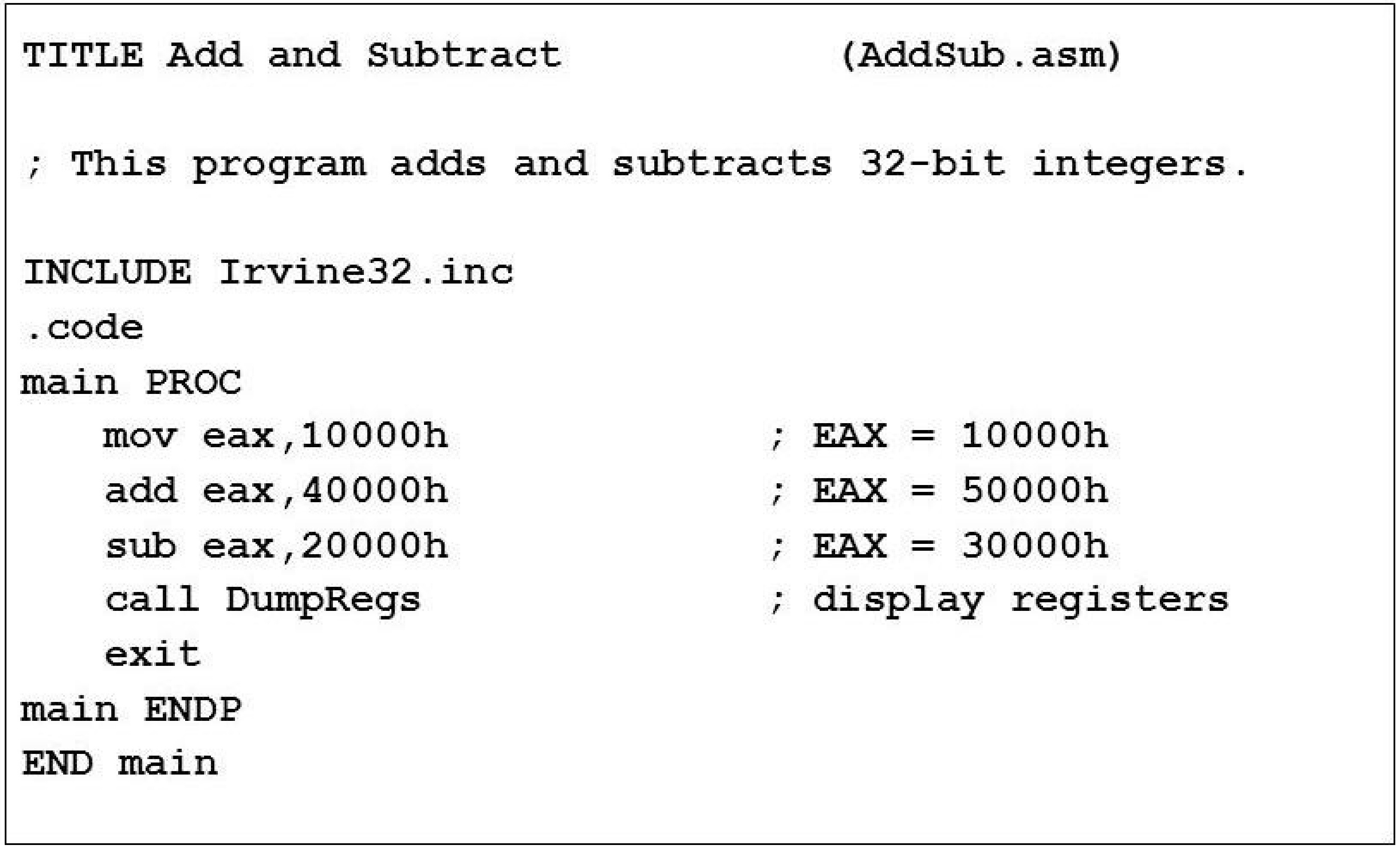
Assembler

- Program which translates the program written in assembly language to machine language.
- It produces one machine instruction for each source instruction and symbolic address to the machine address.
- It includes necessary linkage for closed sub-routine, allocates area of storage, detects and indicates invalid source language instructions.
- Ex. Pseudo assembly.



Assembler, Compiler, Interpreter

Assemble



```
TITLE Add and Subtract              (AddSub.asm)

; This program adds and subtracts 32-bit integers.

INCLUDE Irvine32.inc
.code
main PROC
   mov eax,10000h               ; EAX = 10000h
   add eax,40000h               ; EAX = 50000h
   sub eax,20000h               ; EAX = 30000h
   call DumpRegs                ; display registers
   exit
main ENDP
END main
```

Assembler, Compiler, Interpreter

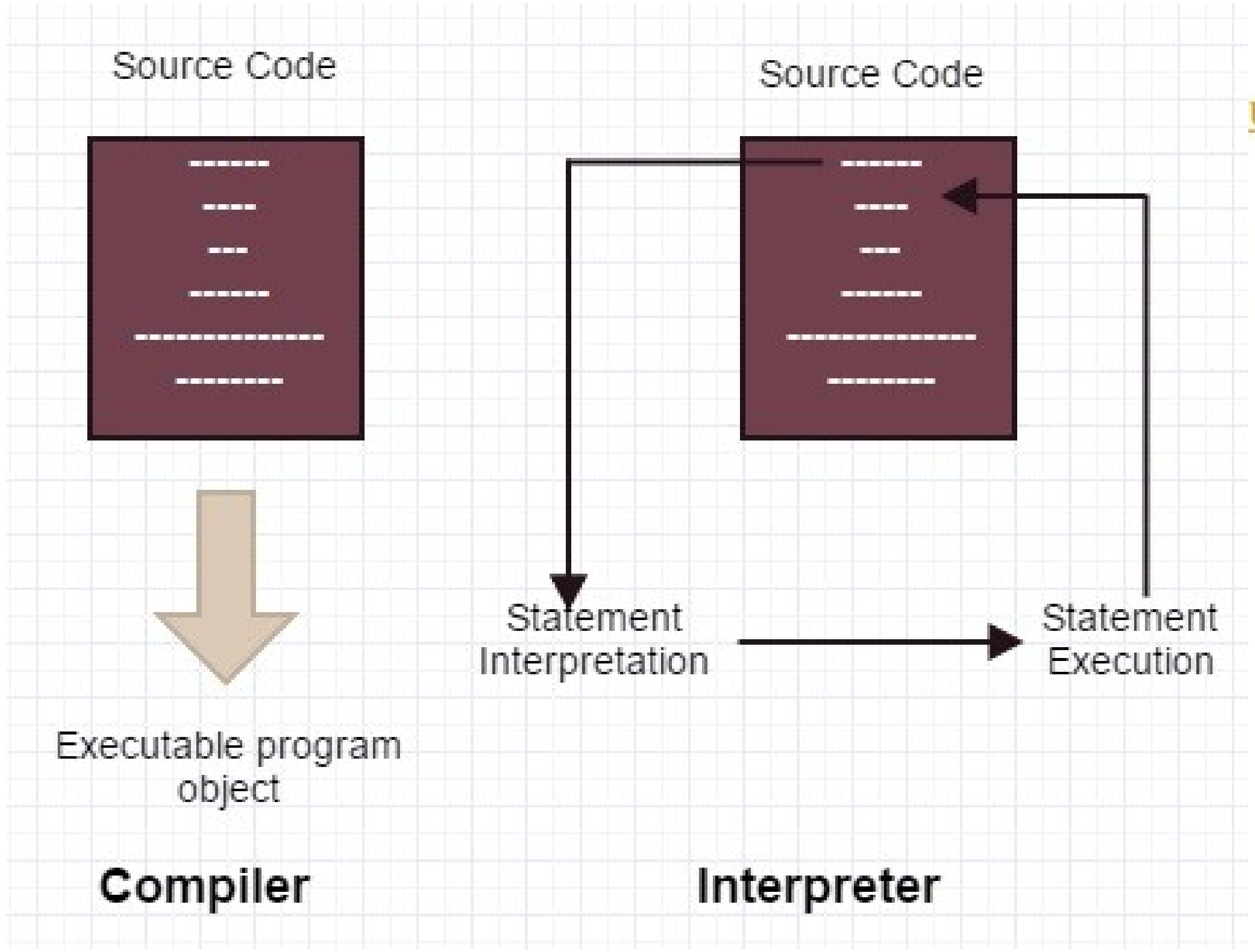
Compiler

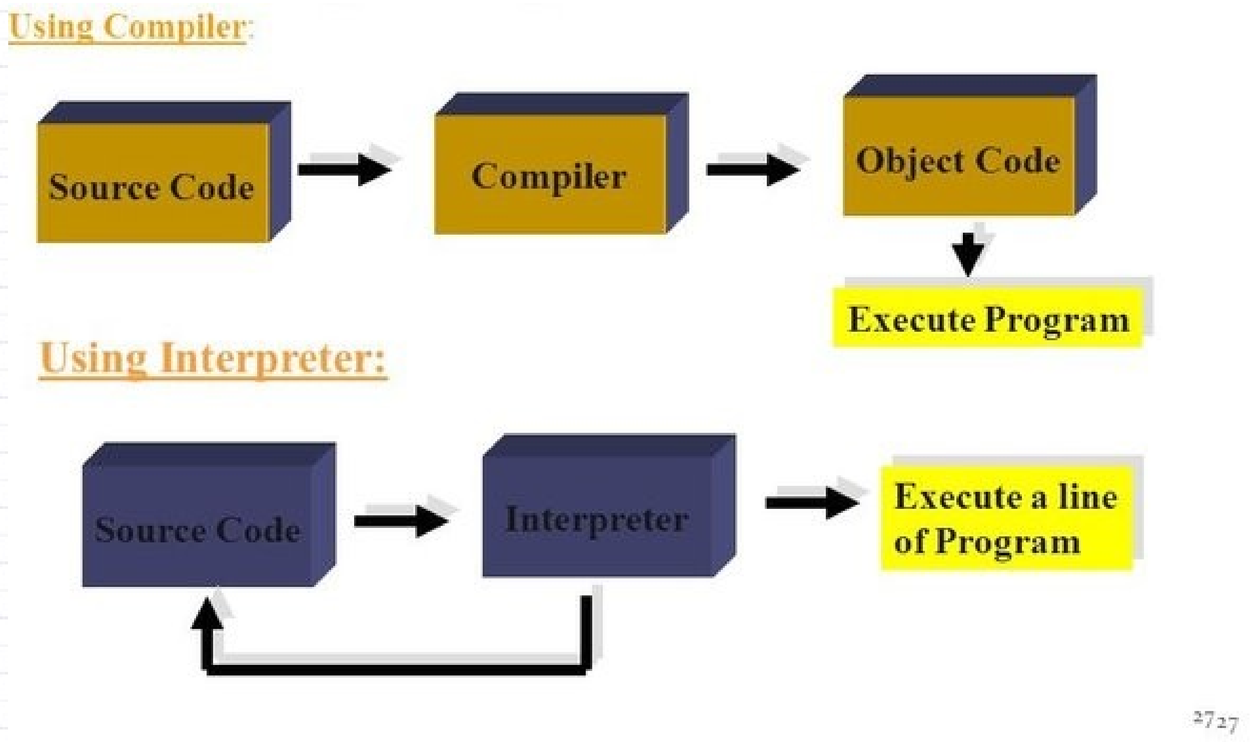
- Program which translates the source code into object code.
- It looks at the entire portion of the source code and recognizes the instructions.
- Every HLL comes with a compiler.
- It defines the acceptable instructions.

Interpreter

- Program which converts each HLL program statement into machine code just before the program statement is to be executed.
- Translation and execution occur immediately one statement at a time.

Assembler, Compiler, Interpreter





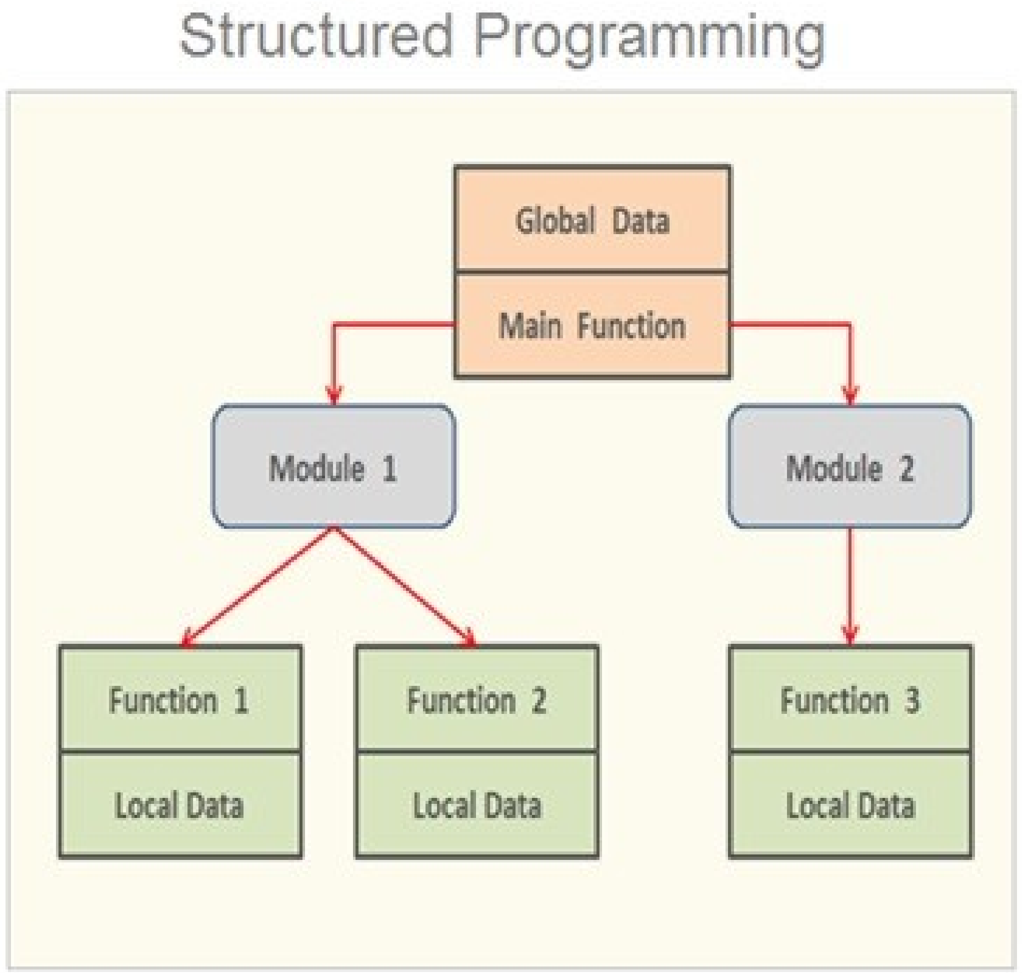
Traditional and Structural Programming concept

- Traditional, structured, programming has worked well for millions of programs and billions of lines of code.
- In particular, it's worked extremely well for number crunching and data processing programs that run once and produce an answer.
- Two characteristics tend to define problems that can be addressed well in a purely structured fashion:
  1. The data to be manipulated closely matches the built-in data types of the language, typically numbers and strings.
  2. The program follows a well-defined flow of control to produce a single result based on some input.

Traditional and Structural Programming concept

- Structured programming can be defined as a software application programming technique that follows a top down design approach with block oriented structures.
- This style of programming is characterized by the programmers tendency to divide his program source code into logically structured blocks which would normally consist of conditional statements, loops and logic blocks.
- This style of programming has the implementation of the source code being processed in the order in which bits of the code have been typed in.

Traditional and Structural Programming concept



Traditional and Structural Programming concept

- Such top-down approach uses sub-routines and functions that perform specific tasks.
- Focus is much more on the control flow rather than the data.
- Program is divided into several basic *structures* and they are the building blocks of the program.
  - Sequence structure – contains program statement one after another.
  - Selection or Conditional structure – conditions are tested and control flow is diverted.
  - Repetition or Loop structure – repeats a block of statements for several times as long as the condition is matched.

Traditional and Structural Programming concept

- Advantages:
  - Programs can be easily described.
  - Testing and debugging is easy.
  - Easy to modify and maintain.
  - Saves time of the programmer.

Assignment 1 – Due 9th May

1. Draw a block diagram of a computer and explain each component in brief.
2. Write about various Generations of:
   a) Computer
   b) Programming language
3. Create a hierarchy diagram of following and briefly write about them:
   a) Types of software
   b) Programming language
4. Differentiate between:
   a) Compiler and Interpreter.
   b) High Level Language and Low Level Language
   c) Primary memory and Secondary memory

Thank you