Syllabus

- Basics of Computer Hardware and Software
    - Basics of Computer Architecture:
        - Processor, Memory, Input& Output devices
    - Application Software & System software
        - Compilers, Interpreters
    - High level and low level languages
    - Introduction to structured approach to programming
- Flow chart
- Algorithms
- Pseudo code (bubble sort, linear search - algorithms and pseudocode)

Syllabus

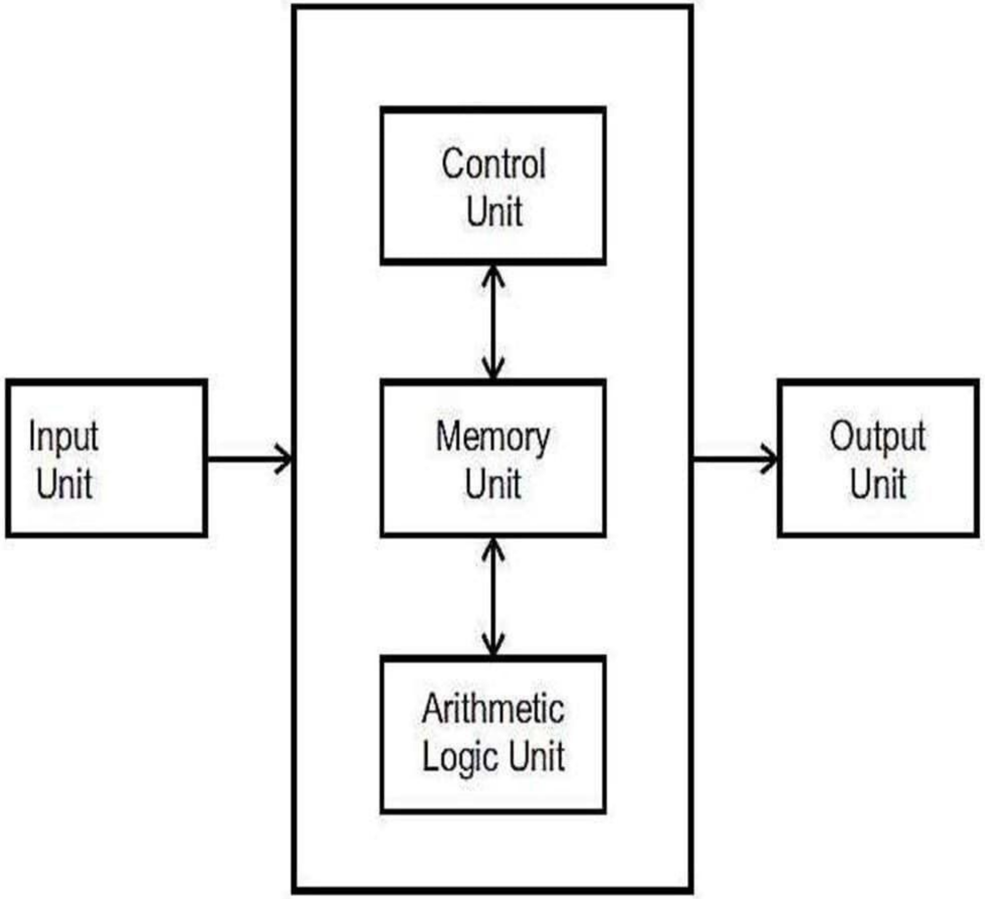
- Basics of Computer Hardware and Software
    - **Basics of Computer Architecture:**
        - **Processor, Memory, Input& Output devices**
    - Application Software & System software
        - Compilers, Interpreters
    - High level and low level languages
    - Introduction to structured approach to programming
- Flow chart
- Algorithms
- Pseudo code (bubble sort, linear search - algorithms and pseudocode)

INTRODUCTION TO COMPUTERS

- Computer is an electronic device that accepts data, process those data, store the data and produces the output.
  - Data are accepted by keyboard or any other input devices.
  - Data are processed by means of central processing (CPU).
  - Some kind of memories are used to store the data and results.
  - Output by monitors or any other output devices.

BASICS OF COMPUTER ARCHITECTURE

- A computer system has five basics units
  1. Input Unit
  2. Output Unit
  3. Storage Unit
  4. Arithmetic Logic Unit
  5. Control Unit



BASICS OF COMPUTER ARCHITECTURE

1. Input Unit

- It provides data and instructions to the computer system
- Examples: keyboard, mouse, microphone, webcam etc.

- Input unit performs following **tasks**:
  - Accept the data and instructions from the outside environment.
  - Convert it into machine language.
  - Supply the converted data to computer system.

BASICS OF COMPUTER ARCHITECTURE

2. Output Unit

- It provides the results of any computation, or instructions to the outside world
- Examples: monitor, printer, projector, speaker etc.

Input/Output Devices

- It can receive data from users, or another device (input), and send data to another device (output).
- Examples: CD-RW drive, DVD-RW drive, USB drive.

BASICS OF COMPUTER ARCHITECTURE

3. Storage Unit

- This unit holds the data and instructions

- Storage unit can be divided into two
  - Primary Storage
  - Secondary Storage

BASICS OF COMPUTER ARCHITECTURE

- **Primary Storage**
- This memory is used to store the data which is being currently executed.
- It is used for temporary storage of data.

- It is divided into two
  - Random Access Memory(RAM)
  - Read Only Memory(ROM)

BASICS OF COMPUTER ARCHITECTURE

- **Random Access Memory(RAM)**
  - It is a **volatile memory**: The data is lost, when the computer is switched off.
  - Both program and data are stored in RAM while the program is being executed
  - Divided into two:
  - DRAM
  - SRAM: Used as Cache memory

BASICS OF COMPUTER ARCHITECTURE

DRAM	SRAM
Constructed of tiny capacitors that leak electricity	Constructed of circuits similar to D Flip-Flop
Requires a recharge every few milliseconds to maintain its data	Hold its contents as long as power is available
Inexpensive	Expensive
Slower than SRAM	Faster than DRAM
Can store many its per chip	Cannot store many bits per chip
Use less power	Use more power
Generate less heat	Generate more heat
Used for main memory	Used for Cache

BASICS OF COMPUTER ARCHITECTURE

- **Read Only Memory(ROM)**
  - It is a **non-volatile memory**
  - Stores crucial information essential to operate the system, like Boot program
  - Used in embedded systems or where the programming needs no change

BASICS OF COMPUTER ARCHITECTURE

- **Read Only Memory(ROM)**
  - ROM is classified into 4 types

  1. **MROM(Masked ROM)**
     - The information is stored in the chip while manufacturing.
     - The information can’t be erased or rewritten/modified.
  2. **PROM(Programmable read-only memory)**
     - It can be programmed by user.
     - Once programmed, the data and instructions in it cannot be changed

BASICS OF COMPUTER ARCHITECTURE

3. **EPROM(Erasable Programmable read only memory)**
   - It can be reprogrammed.
   - To erase data from it, expose it to ultra violet light.
   - To reprogram it, erase all the previous data.
4. **EEPROM(Electrically erasable programmable read only memory)**
   - The data can be erased by applying electric field.
   - We can erase only portions of the chip.

BASICS OF COMPUTER ARCHITECTURE

- **Secondary Storage**
- It is used for permanent storage\long term storage of data.
- It is a **non-volatile memory.**
- The secondary memory is slower and cheaper than primary memory.
- Examples: Hard Disk Drive(HDD), Solid State Drive(SSD), CD etc.

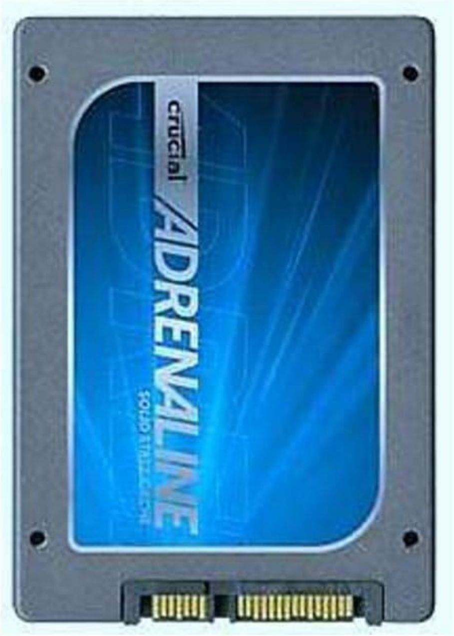
BASICS OF COMPUTER ARCHITECTURE

- **HDD(Hard Disk Drive)**
- The hard disk is a device that can store data permanently.
- It is a **non-volatile memory**
- Computer's operating system is stored on the hard disk
- Programs that installed on the computer are stored on the hard disk.
- Our own programs are stored.

- When the computer is told to execute a program, it loads the program from the hard disk into RAM and executes it from there.

BASICS OF COMPUTER ARCHITECTURE

SSD	HDD
SSD has no moving parts	HDD has moving parts
Noiseless operation	Noisy operation
Faster access time	Slower than SSD
Lower power consumption	Consume more power than SSD



BASICS OF COMPUTER ARCHITECTURE

4. Arithmetic and Logic Unit(ALU)

- All the calculations(addition, subtraction, division, multiplication etc) are performed by ALU.

- Whenever calculations are required, the control unit transfers the data from storage unit to ALU.
- When the operations are done, the result is transferred back to the storage unit

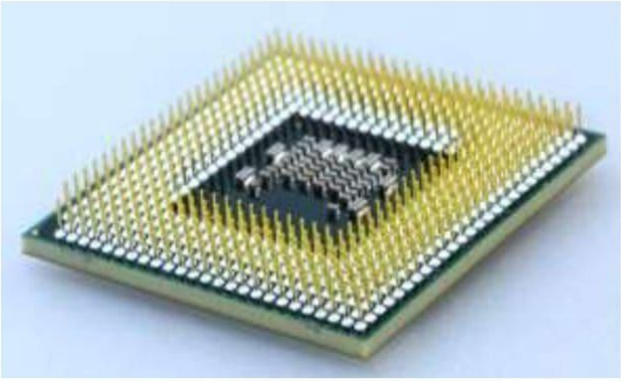
BASICS OF COMPUTER ARCHITECTURE

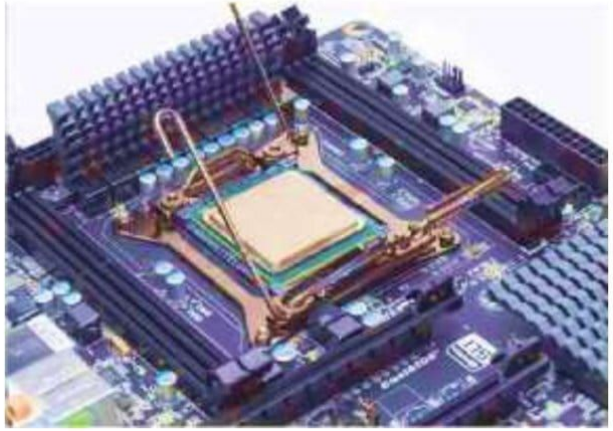
5. **Control Unit**

- It controls all other units of the computer.
- It controls the flow of data and instructions to and from the storage unit to ALU.
- It is also known as **central nervous system** of the computer.

CENTRAL PROCESSING UNIT - CPU

- It is an integrated electronic circuit
- The **control unit and ALU** are together known as CPU
- CPU is the brain of computer system
- It performs following tasks:
  - It performs all calculations
  - It takes all decisions
  - It controls all the units of computer
- Two main competitors in the processor market are **Intel** and **AMD**

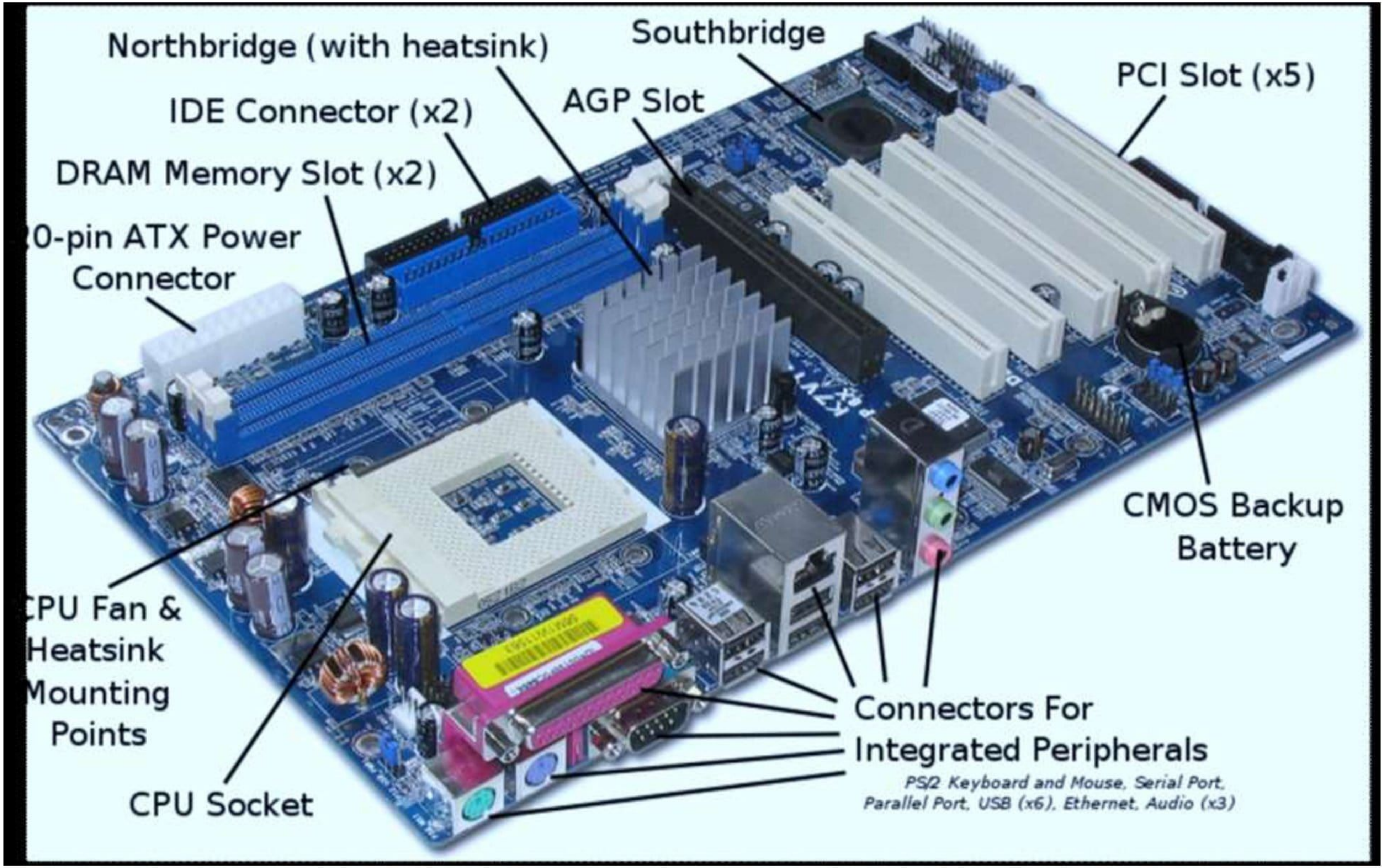




DIFFERENT TYPE OF PROCESSORS

- **Multi-core Processor**
  - IC contains two or more processors
  - This number can increase up to12 cores
  - It enhanced performance, reduced power consumption and more efficient simultaneous processing of multiple tasks
- **Single-core Processor**
  - CPU can only process a single set of instructions at one time
- **Dual-core Processor**
  - CPU can process two sets of instructions at a time.
- **Quad-core processor**
  - CPU can process four sets of instructions at a time.

MOTHERBOARD


Northbridge (with heatsink)
Southbridge
PCI Slot (x5)
AGP Slot
IDE Connector (x2)
DRAM Memory Slot (x2)
0-pin ATX Power Connector
CMOS Backup Battery
PU Fan & Heatsink Mounting Points
CPU Socket
Connectors For Integrated Peripherals
PS/2 Keyboard and Mouse, Serial Port, Parallel Port, USB (x6), Ethernet, Audio (x3)

MACHINE CYCLE

- A machine cycle consists of the steps that a computer's processor executes whenever it receives a machine language instruction
- One machine cycle involves
  - Fetching of instruction
  - Decoding the instruction
  - Operand fetching
  - Executing the instruction

Syllabus

- Basics of Computer Hardware and Software
  - Basics of Computer Architecture:
    - Processor, Memory, Input& Output devices
  - **Application Software & System software**
    - **Compilers, Interpreters**
  - High level and low level languages
  - Introduction to structured approach to programming
- Flow chart
- Algorithms
- Pseudo code (bubble sort, linear search - algorithms and pseudocode)

SOFTWARE

- **Software** is a set of instructions, data or programs used to operate computers and execute specific tasks

- Two types of softwares
  1. System Software
  2. Application Software

SYSTEM SOFTWARE

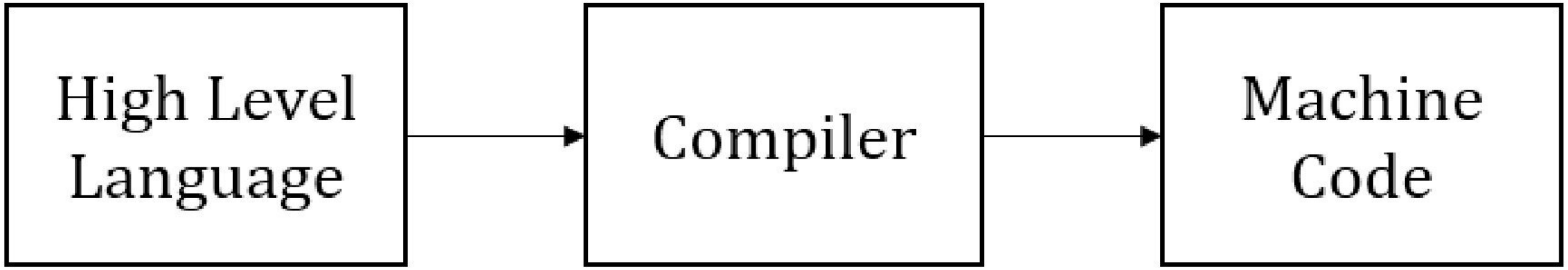
- **System software** is software designed to provide a platform for other software.
- It serves as the interface between the hardware and the end users.
- These software's consists of programs written in low-level languages, used to interact with the hardware at a very basic level.
- These are commonly prepared by the computer manufacturers.
- Examples: Operating systems, Game engines, Search engines, Computer language translators, Device Driver, Linker, Loader etc

OPERATING SYSTEM

- An Operating System (OS) is an interface between a computer user and computer hardware.
- Important **functions** of an operating System.
  - Memory Management
  - Processor Management
  - Device Management
  - File Management
  - Security
  - Control over system performance
  - Job accounting
  - Error detecting aids
  - Coordination between other software and users
- Examples: Windows, Linux, MAC iOS etc..

COMPILER

- **Compiler** is a computer software that translates source code written in a high-level language into a set of machine-language instructions that can be understood by a digital computer's CPU.



- Some compilers translate high-level language into an intermediate assembly language, which is then translated into machine code by an assembler.
- Other compilers generate machine language directly.

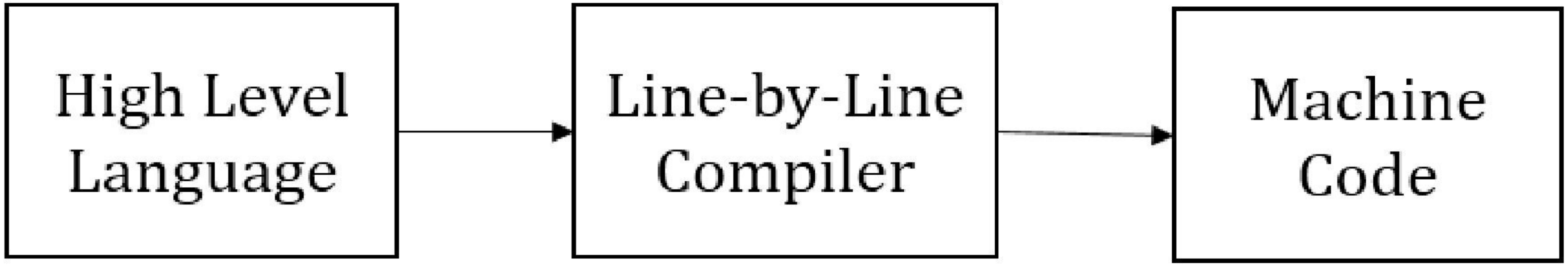
COMPILER

- Phases of compilation:
  - Lexical analysis
  - Parsing
  - Semantic analysis
  - Intermediate Code Generation
  - Code Optimization
  - Code Generation

- Examples:
  - gcc (C compiler)
  - g++ (C++ Compiler)
  - javac (Java Compiler).

INTERPRETER

- An interpreter converts the high level language in to the machine language
- It translates only one statement of the program at a time.
- Programming languages like JavaScript, Python, Ruby use interpreters.



Interpreter

COMPILER	INTERPRETER
Scans the entire program and translates it as a whole into machine code.	Translates program one statement at a time.
Generates Object Code which further requires linking, hence requires more memory.	No Object Code is generated, hence are memory efficient.
It does not require source code for later execution	It requires source code for later execution.
Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters.	Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers.
Errors are displayed at the end together. So debugging is difficult.	Errors are shown line by line. So debugging is easy.

ASSEMBLER

- An assembler is a program that converts assembly language into machine code

DEVICE DRIVER

- A device driver controls a particular type of device that is attached to the computer, such as a keyboard or a mouse.
- The driver program converts the more general input/output instructions of the operating system to messages that the device type can understand.

APPLICATION SOFTWARE

- Application software is a software that performs specific tasks for an end-user
- Examples: Railways Reservation Software, Microsoft Office Suite Software, Microsoft Word, Microsoft PowerPoint, Photoshop, Web browser etc.

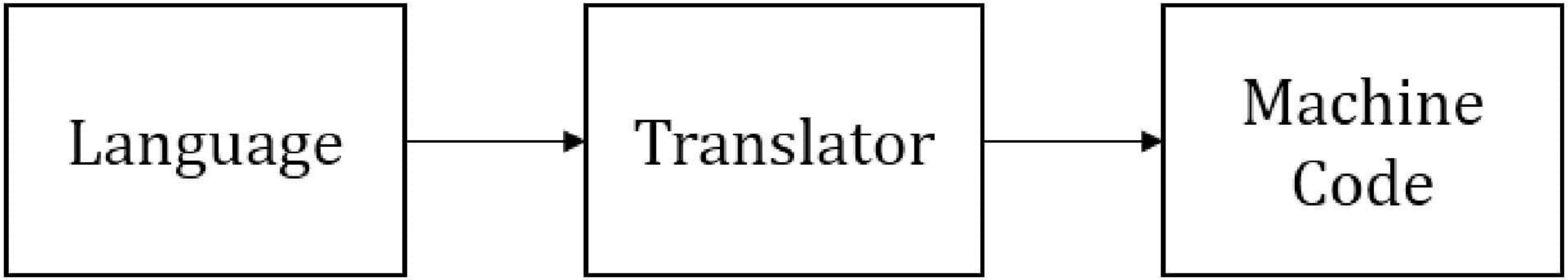
Syllabus

- Basics of Computer Hardware and Software
  - Basics of Computer Architecture:
    - Processor, Memory, Input& Output devices
  - Application Software & System software
    - Compilers, Interpreters
  - **High level and low level languages**
  - Introduction to structured approach to programming
- Flow chart
- Algorithms
- Pseudo code (bubble sort, linear search - algorithms and pseudocode)

CLASSIFICATION OF COMPUTER LANGUAGES

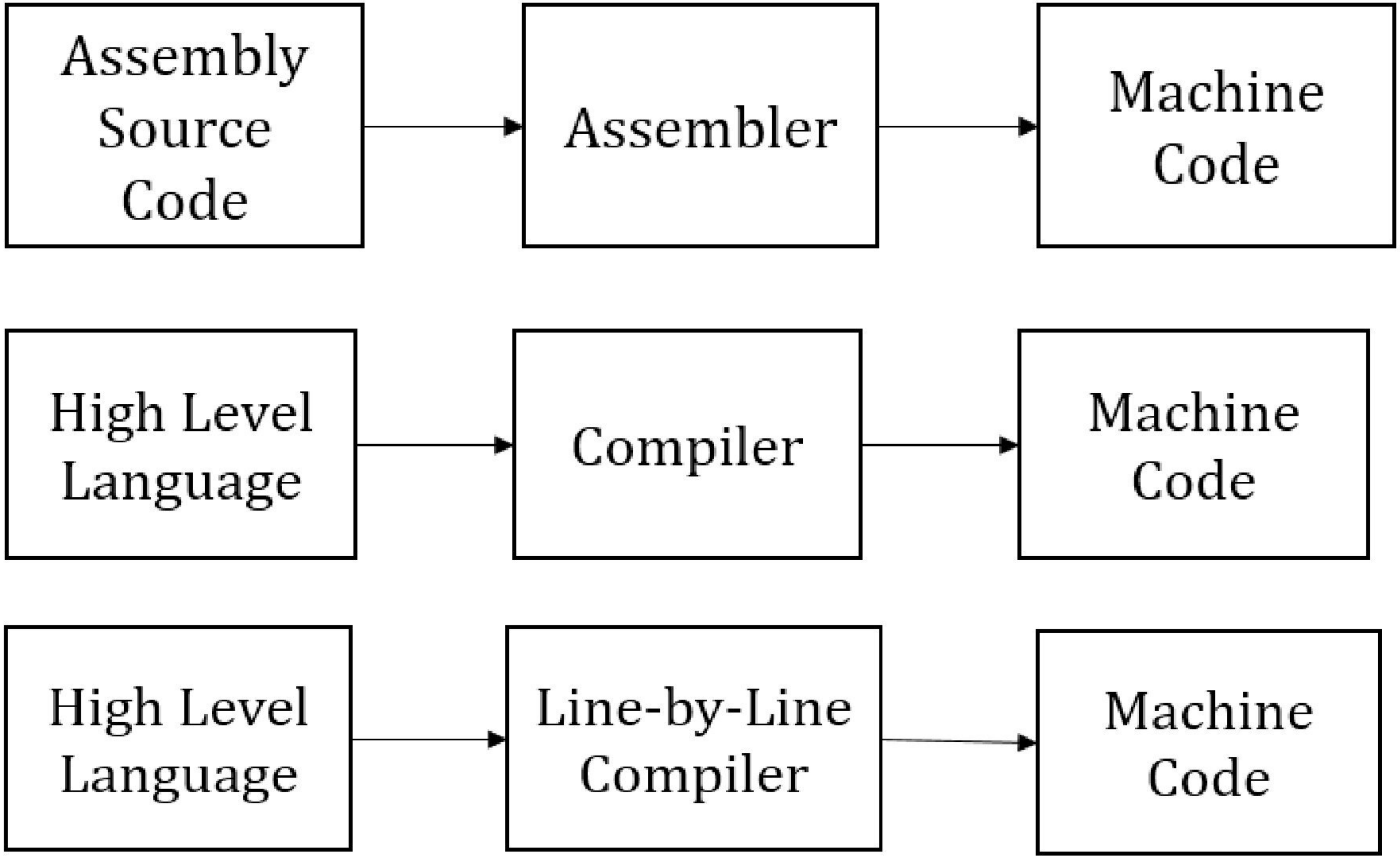
1. **Low Level Language**
   - Programs written in the form of binary (1s and 0s) are known as Machine Language.
   - This is the only language that can be understood or interpreted by a computer.
   - Programs written in machine language or assembly language is known as low level language.
2. **High Level Language**
   - Programs that are written in structured languages such as English are known as High Level Languages.
   - High level languages are not directly interpreted or understood by a computer. It requires a **translator** which translate the high level language in to machine code.

LANGUAGE TRANSLATORS



- **Some language translators are**
  1. Assembler
  2. Compiler
  3. Interpreter

LANGUAGE TRANSLATORS



Assembly Source Code → Assembler → Machine Code

High Level Language → Compiler → Machine Code

High Level Language → Line-by-Line Compiler → Machine Code

Interpreter

HIGHLEVEL LANGUAGE	LOW LEVEL LANGUAGE
It is programmer friendly language.	It is a machine friendly language.
High level language is less memory efficient.	Low level language is high memory efficient.
It is easy to understand.	It is tough to understand.
It is simple to debug.	It is complex to debug comparatively.
It is simple to maintain/modify.	It is complex to maintain comparatively.
It is portable.	It is non-portable.
It can run on any platform.	It is machine-dependent.
It needs compiler or interpreter for translation.	It needs assembler for translation.
It is used widely for programming.	It is not commonly used now-a-days in programming.
Executed slower than low level languages	Execute with high speed
For writing programs, hardware knowledge is not required.	For writing programs, hardware knowledge is required.
Examples: C, C++, Java, Python	Examples: Machine language, Assembly languages(8085, 8086)

Syllabus

- Basics of Computer Hardware and Software
  - Basics of Computer Architecture:
    - Processor, Memory, Input& Output devices
  - Application Software & System software
    - Compilers, Interpreters
  - High level and low level languages
  - **Introduction to structured approach to programming**
- Algorithms
- Flow chart
- Pseudo code (bubble sort, linear search - algorithms and pseudocode)

STRUCTURED PROGRAMMING

- The main aim of **Structured programming** is to improve and increase quality, clarity, and development time of computer programs.
- Programs are divided into small programs or functions. Each of these functions has specific job. The entire problem is solved by collecting such functions.
- In this, functions are written globally and code lines are processed one by one i.e., Run sequentially.
- It generally follows "Top-Down Approach".
- It is all about facilitating creation of programs with readable code and reusable components.
- It is a method of organizing, managing and coding programs that can provide much easier modification and understanding.
- **Examples :** C, C++, Pascal, ALGOL etc