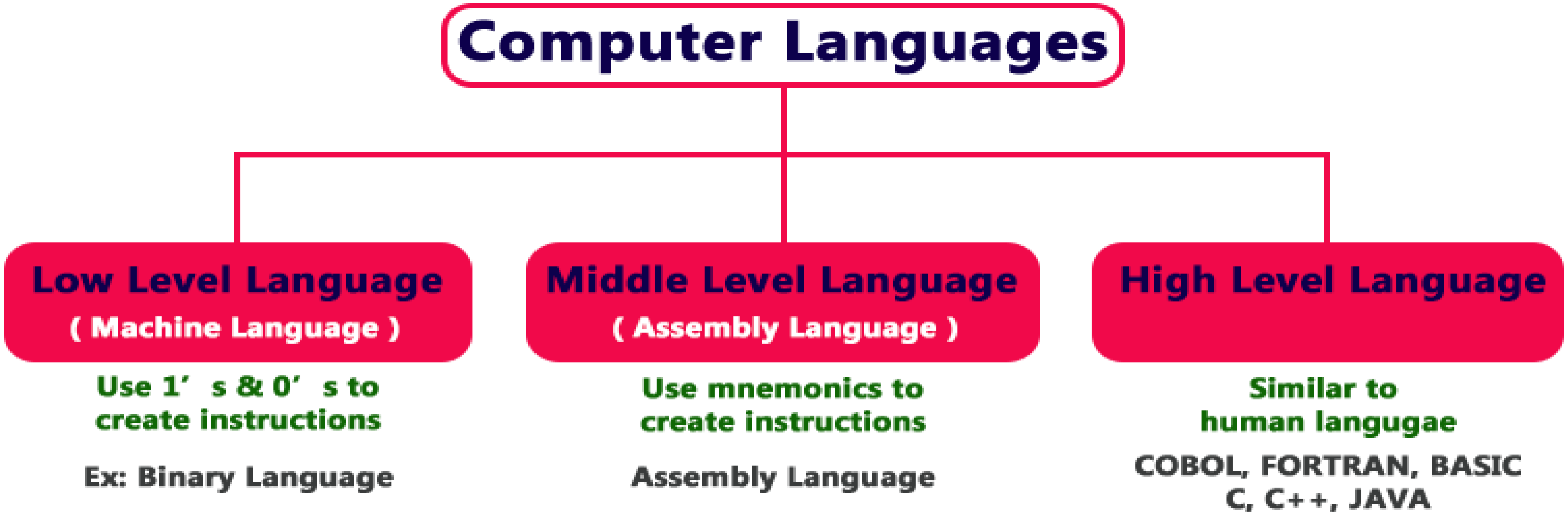
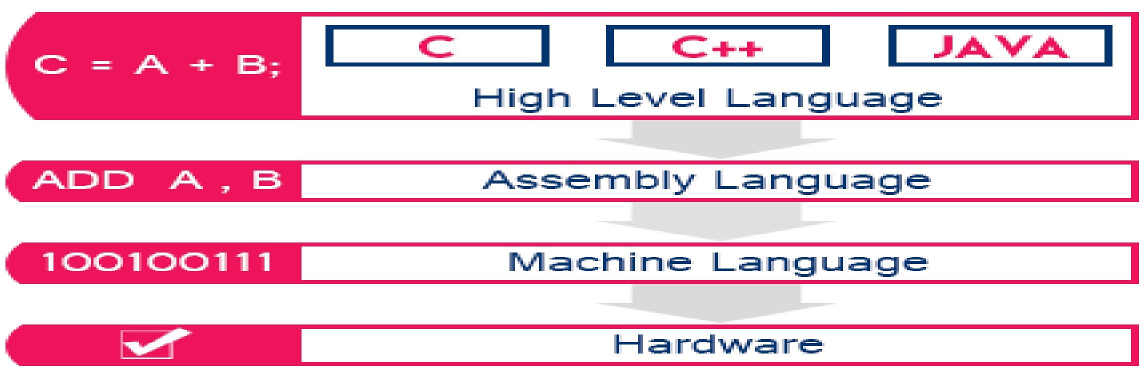
# Module-3

| MODULE-3 | 8 Hr |
|---|---|
| **Basic Structure of Computers:** Functional Units, Basic Operational Concepts, Bus structure, Performance – Processor Clock, Basic Performance Equation, Clock Rate, Performance Measurement.**Machine Instructions and Programs:** Memory Location and Addresses, Memory Operations, Instruction and Instruction sequencing, Addressing Modes.<br><br>**Text book 2: 1.2, 1.3, 1.4, 1.6, 2.2, 2.3, 2.4, 2.5** | |

# Why Computer Organisation



# Basic Structure of Computers

# Functional Units

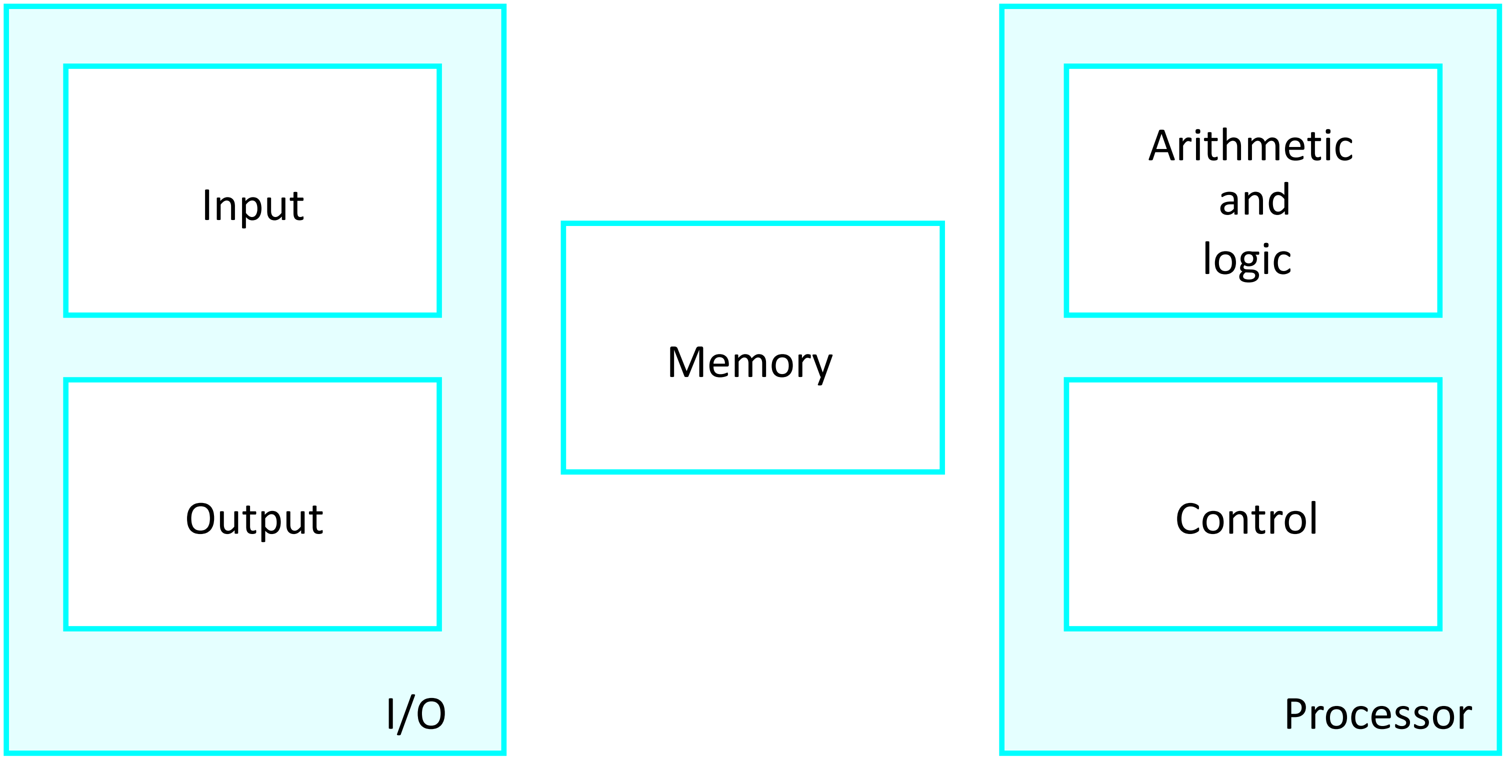



Figure 1.1. Basic functional units of a computer.

# Functional Units

- A computer consists of five functionally independent main parts: input, output, memory, arithmetic and logic, and control units

1. **Input Unit**

- The most common input device is the keyboard. Whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted to the processor

# Functional Units

**2. Memory Unit**

The function of the memory unit is to store programs and data. There are two classes of storage, called primary and secondary.

**a) Primary Memory**

- *Primary memory*, also called *main memory*, is a fast memory that operates at electronic speeds.
- Programs must be stored in this memory while they are being executed.

# Functional Units

**b) Secondary Storage**

- Although primary memory is essential, it tends to be expensive and does not retain information when power is turned off.
- Thus additional, less expensive, permanent *secondary storage* is used when large amounts of data and many programs have to be stored, particularly for information that is accessed infrequently.

# Functional Units

**c) Cache Memory**

- As an adjunct (addition) to the main memory, a smaller, faster RAM unit, called a cache, is used to hold sections of a program that are currently being executed, along with any associated data.

**3. Arithmetic and Logic Unit**

Most computer operations are executed in the *arithmetic and logic unit* (ALU) of the processor.

Any arithmetic or logic operation, such as addition, subtraction, multiplication, division, or comparison of numbers.

# Functional Units

is initiated by bringing the required operands into the processor, where the operation is performed by the ALU.

**4. Output Unit**

- The output unit is the counterpart of the input unit. Its function is to send processed results to the outside world.
- A familiar example of such a device is a *printer*. Most printers employ either photocopying techniques, as in laser printers, or ink jet streams

# Functional Units

## 5. Control Unit

- The memory, arithmetic and logic, and I/O units store and process information and perform input and output operations.
- The operation of these units must be coordinated in some way. This is the responsibility of the control unit.

# Basic Operational Concepts

- An Instruction consists of 2 parts,

1) Operation code (Opcode) and

2) Operands.

mnemonic

| OPCODE | OPERANDS |
|---|---|

- The data/operands are stored in memory.
- The individual instruction are brought from the memory to the processor.
- Then, the processor performs the specified operation.

# Basic Operational Concepts

Let us see a typical instruction

***ADD LOCA, R0***

- This instruction is an addition operation. The following are the steps to execute the instruction:
- Step 1: Fetch the instruction from main-memory into the processor.
- Step 2: Fetch the operand at location LOCA from main-memory into the processor.
- Step 3: Add the memory operand (i.e. fetched contents of LOCA) to the contents of register R0.
- Step 4: Store the result (sum) in R0.

# Basic Operational Concepts

- **Processor components**
- The **processor** contains ALU, control-circuitry and many registers.

# Basic Operational Concepts

- The processor contains “n” general-purpose registers **R0** through **Rn-1**.
- The **IR** holds the instruction that is currently being executed.
- The **control-unit** generates the timing-signals that determine when a given action is to take place.
- The **PC** contains the memory-address of the next-instruction to be fetched & executed.

# Basic Operational Concepts

- During the execution of an instruction, the contents of PC are updated to point to next instruction.
- The **MAR** holds the address of the memory-location to be accessed.
- The **MDR** contains the data to be written into or read out of the addressed location.
- MAR and MDR facilitates the communication with memory.
- (IR : Instruction-Register, PC : Program Counter)
- (MAR :Memory Address Register, MDR : Memory Data Register)

# Basic Operational Concepts

- **STEPS TO EXECUTE AN INSTRUCTION**

1) The address of first instruction (to be executed) gets loaded into PC.

2) The contents of PC (i.e. address) are transferred to the MAR & control-unit issues Read signal to memory.

3) After certain amount of elapsed time, the first instruction is read out of memory and placed into MDR.

4) Next, the contents of MDR are transferred to IR. At this point, the instruction can be decoded &executed.

# Basic Operational Concepts

5) To fetch an operand, it's address is placed into MAR & control-unit issues Read signal. As a result, the operand is transferred from memory into MDR, and then it is transferred from MDR to ALU.

6) Likewise required number of operands is fetched into processor.

7) Finally, ALU performs the desired operation.

8) If the result of this operation is to be stored in the memory, then the result is sent to the MDR.

# Basic Operational Concepts

9) The address of the location where the result is to be stored is sent to the MAR and a Write cycle is initiated.

10) At some point during execution, contents of PC are incremented to point to next instruction in the program.

# BUS STRUCTURE

- A bus is a group of lines that serves as a connecting path for several devices.
- A bus may be lines or wires.
- The lines carry data or address or control signal.
- There are 2 types of Bus structures:

1) Single Bus Structure and 2) Multiple Bus Structure.

**1) Single Bus Structure**

- Because the bus can be used for only one transfer at a time, only 2 units can actively use the bus at any given time. Bus control lines are used to arbitrate multiple requests for use of the bus.

# Bus Structure

- Single-bus

# BUS STRUCTURE

## 2) Multiple Bus Structure

- Systems that contain multiple buses achieve more concurrency in operations . Two or more transfers can be carried out at the same time.
- **Advantage:** Better performance.
- **Disadvantage:** Increased cost
- The devices connected to a bus vary widely in their speed of operation.

• To synchronize their operational-speed, buffer-registers can be used.

# BUS STRUCTURE

- **Buffer Registers**
- are included with the devices to hold the information during transfers.
- prevent a high-speed processor from being locked to a slow I/O device during data transfers.

# PERFORMANCE

- The most important measure of performance of a computer is how quickly it can execute programs.
- The **speed of a computer is affected** by the design of

1) Instruction-set.

2) Hardware & the technology in which the hardware is implemented.

3) Software including the operating system.

- Because programs are usually written in a HLL, performance is also affected by the compiler that translates programs into machine language. (HLL :High Level Language).

# PERFORMANCE

- For best performance, it is necessary to design the compiler, machine instruction set and hardware in a co-ordinated way.

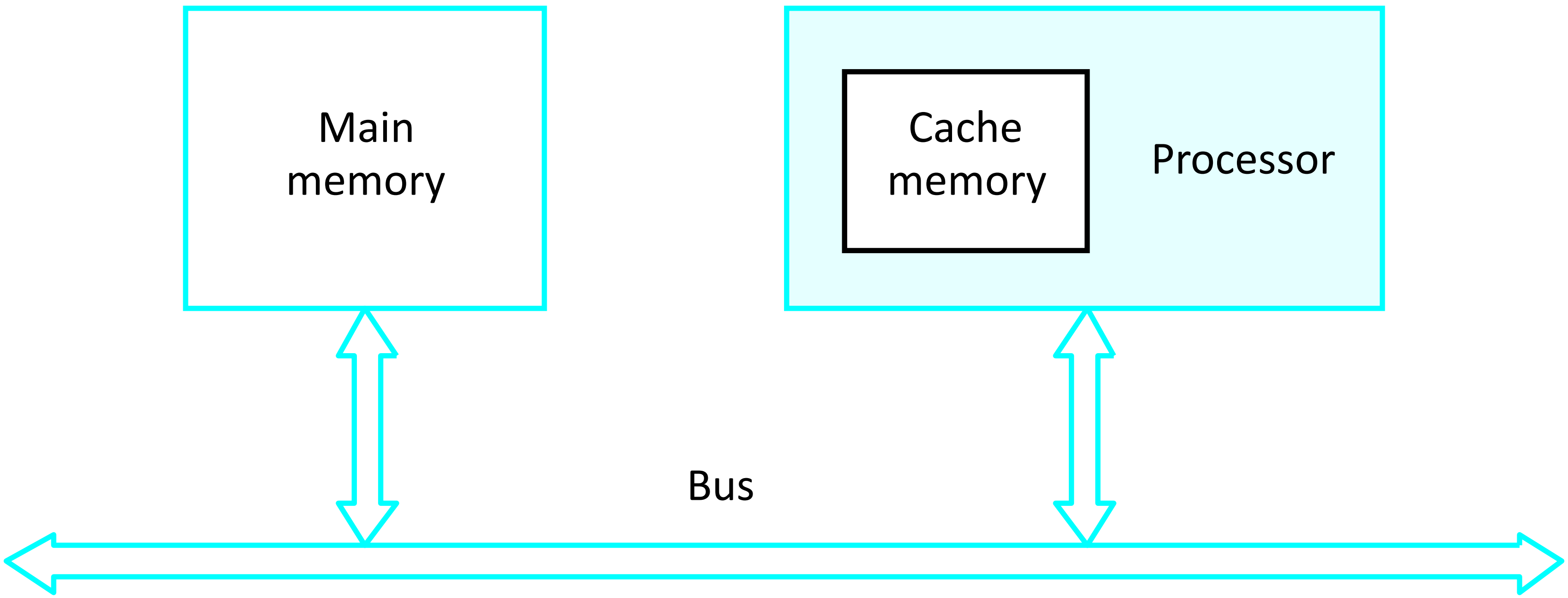


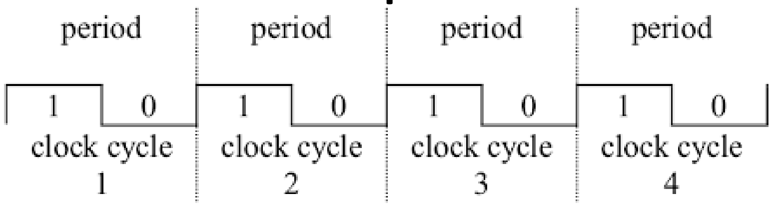

Figure 1.5. The processor cache.

# PERFORMANCE

- Let us examine the flow of program instructions and data between the memory & the processor.
- At the start of execution, all program instructions are stored in the main-memory.
- As execution proceeds, instructions are fetched into the processor, and a copy is placed in the cache.
- Later, if the same instruction is needed a second time, it is read directly from the cache.
- A program will be executed faster if movement of instruction/data between the main-memory and the processor is minimized which is achieved by using the cache

# PERFORMANCE

- **PROCESSOR CLOCK**
- Processor circuits are controlled by a **timing signal called a Clock.**
- The clock defines regular time intervals called **Clock Cycles**.
- To execute a machine instruction, the processor divides the action to be performed into a sequence of basic steps such that each step can be completed in one clock cycle.




- Let P = Length of one clock cycle
- R = Clock rate

# PERFORMANCE

- Relation between P and R is given by R= $1/P$
- R is measured in cycles per second.
- Cycles per second is also called Hertz (Hz)
- **BASIC PERFORMANCE EQUATION**
- Let T = Processor time required to executed a program.
- N = Actual number of instruction executions.

# PERFORMANCE

- S = Average number of basic steps needed to execute one machine instruction.
- R = Clock rate in cycles per second.
- The program execution time is given by
  - **T=** $(N \ X \ S)/ \ R$
- To achieve high performance, the computer designer must reduce the value of T, which means reducing N and S, and **increasing R**.
- **CLOCK RATE**
- There are 2 possibilities for increasing the clock rate R:

# PERFORMANCE

- 1) Improving the **IC technology** makes logic-circuits faster.
- This reduces the **time needed to compute** a basic step. (IC :integrated circuits).
- This allows the **clock period P** to be **reduced and the clock rate R** to be increased.
- 2) Reducing the amount of processing done in one basic step also **reduces the clock period P**.
- In presence of a **cache**, the percentage of accesses to the main-memory is small.
- Hence, much of performance-gain expected from the use of faster technology can be realized.

# PERFORMANCE

- **Problems**

1. A program contains l000 instruction Out of that 25% instructions requires 4 clock cycles. 40% instructions require 5 clocks and remaining 3 clock cycles for execution. Find the total time required to execute the program running in a 1GHz machine. (5 Marks)

**Answer**

- N=l000
- 25% of N= 250 instructions require 4 clock cycles
- 40% of N =400 instructions require 5 clock cycles
- 35% of N =350 instructions require 3 clock cycles

# PERFORMANCE

- T= (N*S)/R =250*4+400*5+350*3/1*$10^9$
- =(1000+2000+1050)/1* l$0^9$

=**4.05* l$0^{-6}$= 4.05 μs**

2. The Effective value of S is 1.25 and the average value of N is 200. If the clock rate is 500MHz, calculate the total program execution time required.

**Answer**

- T=(N*S)/R
- = (200*1.25)500* I$0^6$
- = (250)/500* I$0^6$

**=5*$10^{-7}$s**

# PERFORMANCE

- **PERFORMANCE MEASUREMENT**
- Benchmark refers to standard task used to measure how well a processor operates.
- The Performance Measure is the time taken by a computer to execute a given benchmark.
- SPEC selects & publishes the standard programs along with their test results for different application domains. (SPEC :**System Performance Evaluation Corporation**).
- SPEC Rating is given by

$$\text{SPEC rating} = \frac{\textit{Running time on the reference computer}}{\textit{Running time on the computer under test}}$$

# PERFORMANCE

- **PERFORMANCE MEASUREMENT**
- SPEC rating = 50 :The computer under test is 50 times as fast as reference-computer.
- The test is repeated for all the programs in the SPEC suite.
- Then, the geometric mean of the results is computed.
- Let SPECi = Rating for program „i' in the suite.
- Overall SPEC rating for the computer is given by

$$SPEC\ rating = \left(\prod_{i=1}^{n} SPEC_i\right)^{1/n}$$

# PERFORMANCE

- **PERFORMANCE MEASUREMENT**
- **Where n is the number of programs in the suite**
- **Geometrical mean= is defined as the nth root of the product of n numbers**
- **Problem**
- **List the steps needed to execute the machine instruction:** (Appeared in- Dec.2017/jan.2018)
- ***Load LOC A, R0***
- **Solution:**
- 1. Transfer the contents of register PC to register MAR.

2. Issue a Read command to memory. And, then wait until it has transferred the requested word into register MDR.

3. Transfer the instruction from MDR into IR and decode it.

4. Transfer the address LOCA from IR to MAR.

5. Issue a Read command and wait until MDR is loaded.

6. Transfer contents of MDR to the ALU.

7. Transfer contents of R0 to the ALU.

8. Perform addition of the two operands in the ALU and transfer result into R0.

9. Transfer contents of PC to ALU.

10. Add 1 to operand in ALU and transfer incremented address to PC.

**Problem :**

- List the steps needed to execute the machine instruction:
- Add R1, R2, R3
- **Solution:**

1. Transfer the contents of register PC to register MAR.

2. Issue a Read command to memory. And, then wait until it has transferred the requested word into register MDR.

3. Transfer the instruction from MDR into IR and decode it.

4. Transfer contents of R1 and R2 to the ALU.

5. Perform addition of two operands in the ALU and transfer answer into R3.

6. Transfer contents of PC to ALU.

7. Add 1 to operand in ALU and transfer incremented address to PC.

- **Problem :**

(a) Give a short sequence of machine instructions for the task “Add the contents of memory-location A to those of location B, and place the answer in location C”.

Instructions:

- Load R*i*, LOC

  and

- Store R*i*, LOC
- are the only instructions available to transfer data between memory and the general purpose registers.

- b) Is it possible to use fewer instructions of these types to accomplish the task in part (a)? If yes, give the sequence.
- **Solution:**
- (a)
- Load A, R0
- Load B, R1
- Add R0, R1
- Store R1, C

- (b) Yes;
- Move B, C
- Add A, C