Computer Science & IT

Computer Organization & Architecture

Comprehensive Theory

with Solved Examples and Practice Questions

MADE EASY Publications

MADE EASY Publications

Corporate Office: 44-A/4, Kalu Sarai (Near Hauz Khas Metro Station), New Delhi-110016
E-mail: infomep@madeeasy.in
Contact: 011-45124660, 8860378007
Visit us at: www.madeeasypublications.org

Computer Organization & Architecture

First Edition: 2015
Second Edition : 2016
Third Edition : 2017
Fourth Edition : 2018
Fifth Edition : 2019
Sixth Edition : **2020**

Contents

Computer Organization & Architecture

■■■■

Computer Organization & Architecture

Goal of the Subject

Basic understanding of computer organization includes:

- Understanding roles of processors, main memory, and input/output devices.
- Understanding the concept of programs as sequences of machine instructions.
- Understanding the relationship between assembly language and machine language; development of skill in assembly language programming;
- Understanding the relationship between high-level compiled languages and assembly language.
- Understanding arithmetic and logical operations with integer operands, floating-point number systems and operations.
- Understanding simple data path and control designs for processors, memory organization, including cache structures and virtual memory schemes.

Computer Organization & Architecture

INTRODUCTION

In this book we tried to keep the syllabus of Computer Organization around the GATE syllabus. Each topic required for GATE is crisply covered with illustrative examples and each chapter is provided with Student Assignment at the end of each chapter so that the students get the thorough revision of the topics that he/she had studied. This subject is carefully divided into eight chapters as described below.

1. Basics of Computer Design: In this chapter we discuss the Computer System, Data storage in the memory and Machine instructions
2. CPU Design: In this chapter we discuss the Datapath and Control unit design first is hardwired control unit second is microprogrammed.
3. Instruction Pipelining: In this chapter we discuss Performance, Instruction processing, Pipeline design and issue, Pipeline hazards, Pipeline performance analysis and Speedup.
4. Memory Hierarchy Design: In this chapter we discuss Primary memory, Associative memory, Address space and Cache memory design.
5. Input-Output and Secondary Storage: In this chapter we discuss Interface design, Input-output mode and Secondary memory.
6. Data Representation: In this chapter we discuss the Fixed and floating point formate, IEEE floating - point number representation, Computer arithmetic, Adding 2's complement numbers and Multiplying floating-point numbers.

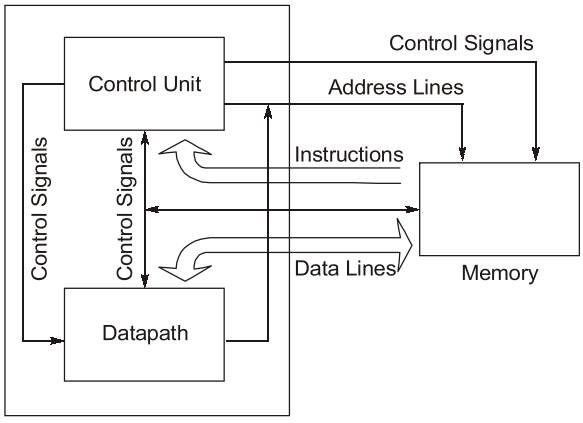
■■■■

CHAPTER 02

CPU Design

2.1 Introduction

Basic processing unit has two units Datapath and Control unit. Datapath includes the register section and the arithmetic/logic unit (ALU). An Arithmetic and Logic Unit (ALU) is a combinational circuit that performs LOGIC and ARITHMETIC Operations. The ALU does the actual computation or processing of data. The basic operations are implemented in hardware level.



Basic Processing Unit (Data Path/Control unit)

Datapath

It is combination of functional units and registers . The layout of bus wide logic that operates on data signals is called a datapath.

- Datapath is the component of the processor that performs arithmetic operations.
- Functional units and Registers are called as elements of datapath.
- Functional units are ALU, multipliers, dividers, etc.
- Registers are program counter, shifters, storage registers, etc.

Control Unit

The Control Unit controls the movement of data and instruction into and out of the CPU and controls the operation of the ALU.

Types of CPU Organizations

1. Single Accumulator CPU: *Example:* ADD X ; AC ← AC + M[X]
2. General Register CPU: *Example:* ADD R1, R2 ; R1 ← R1 + R2
3. Stack CPU: *Example:* PUSH X
   ADD: It is zero address instruction, which pops two numbers from the top of stack and adds those two numbers then pushes the result onto stack.
4. Combination of CPU organizations: Combination of any of three CPU organizations.

2.1.1 ALU Design

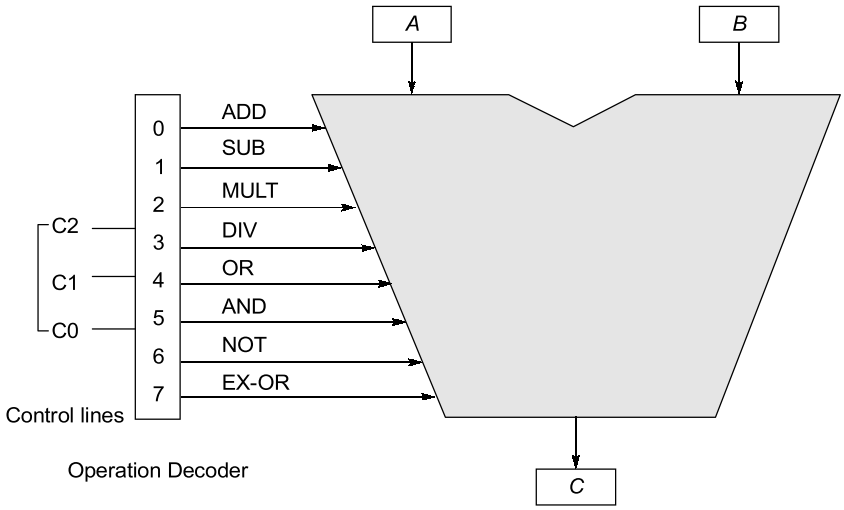
Consider an ALU is having four arithmetic operations (Addition, subtraction, multiplication and division) and four logical operations (OR, AND, NOT & EX-OR). We need three control lines to identify any one of these eight operations. Truth table for 8 operations:

C_1	C_0	Arithmetic $C_2 = 0$	Logical $C_2 = 1$
0	0	Addition	OR
0	1	Subtraction	AND
1	0	Multiplication	NOT
1	1	Division	EX-OR

Control line $C_2 = 0$ identifies as arithmetic operation, and $C_2 = 1$ identifies the logical operation. Control lines C_0 and C_1 are used to identify any one of the four operations in a group.

2.1.2 3 to 8 Decoder Circuit for 8 Operations

A 3 to 8 decoder can be used to decode the instruction to implement those 8 operations.



The input data are stored in registers A and B, and according to the operation specified in the control lines, the ALU perform the operation and put the result in register C.

C = A op B where op is any of 8 operations.

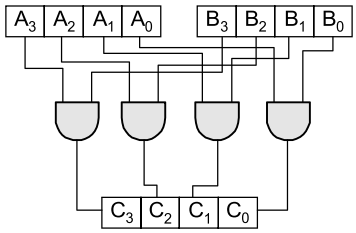
NOT: Complements the input.

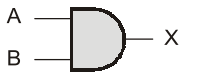
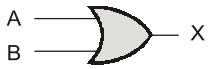
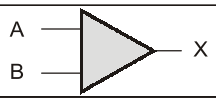
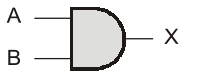
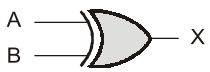
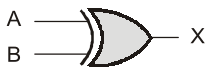
AND: The output is high if both the inputs are high.

OR: The output is high if any one of the input is high.
EX-OR: The output is high if either of the input is high.

2.1.3 4-bit AND Implementation

The following circuit perform the AND operation on two 4-bit number.



Name	Graphic Symbol	Algebraic Function	Truth Table
AND		X = A · B	A B X 0 0 0 0 1 0 1 0 0 1 1 1
OR		X = A + B	A B X 0 0 0 0 1 1 1 0 1 1 1 1
Inverter		X = A′	A X 0 1 1 1
Buffer		X = A	A X 0 0 1 1
NAND		X = (AB)′	A B X 0 0 1 0 1 1 1 0 1 1 1 0
NOR		X = (A+B)′	A B X 0 0 1 0 1 0 1 0 0 1 1 0
Exclusive-OR (XOR)		X = A ⊕ B	A B X 0 0 0 0 1 1 1 0 1 1 1 0
Exclusive-NOR or equivalence		X = (A ⊕ B)′	A B X 0 0 1 0 1 0 1 0 0 1 1 1

2.2 Datapath

System Bus

The CPU is connected to the rest of the system through system bus. Through system bus, data or information gets transferred between the CPU and the other component of the system. The system bus may have three components:

1. **Data Bus:** Data bus is used to transfer the data between main memory and CPU. A single memory unit is used for both instructions and data. In CPU instructions and data is stored in separate registers.
2. **Address Bus:** Address bus is used to access a particular memory location by putting the address of the memory location.
3. **Control Bus:** Control bus is used to provide the different control signals generated by CPU to different parts of the system. As for example, memory read is a signal generated by CPU to indicate that a memory read operation has to be performed. Through control bus this signal is transferred to memory module to indicate the required operation.

Registers in Datapath Design

There are mainly two types of registers: (1) User visible Registers and (2) Control and Status Registers.

User-Visible Registers

These enables the machine - or assembly-language programmer to minimize main memory reference by optimizing use of registers. The user-visible registers can be categorized as follows:

- **General-purpose registers** can be assigned to a variety of functions by the programmer. In some cases, general- purpose registers can be used for addressing functions (e.g., register indirect, displacement). In other cases, there is a partial or clean separation between data registers and address registers.
- **Data registers** may be used to hold only data and cannot be employed in the calculation of an operand address.
- **Address registers** are used for general purpose, or they may be devoted to a particular addressing mode. The following address registers may be used in the system.
- **Segment pointer:** In a machine with segment addressing, a segment register holds the address of the base of the segment. There may be multiple registers, one for the code segment and one for the data segment.
- **Index registers:** These are used for indexed addressing and may be auto-indexed.
- **Stack pointer:** If there is user visible stack addressing, then typically the stack is in memory and there is a dedicated register that points to the top of the stack.
- **Condition Codes** (also referred to as flags) are bits set by the CPU hardware as the result of the operations.

  For example, an arithmetic operation may produce a positive, negative, zero or overflow result. In addition to the result itself begin stored in a register or memory, a condition code is also set. The code may be subsequently be tested as part of a condition branch operation. Condition code bits are collected into one or more registers.

Control and Status Registers

These are used by the control unit to control the operation of the CPU. Operating system programs may also use these in privileged mode to control the execution of program. Processor status word is used as synonym for control and status registers.

Program Status Word (PSW): All CPU designs include a register or set of registers, often known as the processor status word (PSW), that contains status information. The PSW typically contains condition codes plus other status information. Common fields or flags include the following:

(a) **Sign:** Contains the sign bit of the result of the last arithmetic operation.

(b) **Zero:** Set when the result is zero.

(c) **Carry:** Set if an operation resulted in a carry (addition) into or borrow (subtraction) out of a high order bit.

(d) **Equal:** Set if a logical compare result is equal.

(e) **Overflow:** Used to indicate arithmetic overflow.

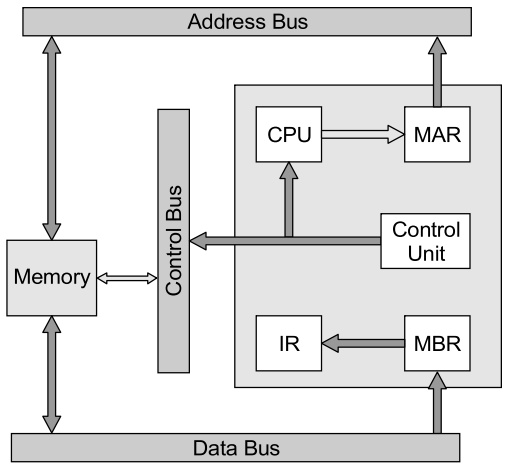
(f) **Interrupt enable/disable:** Used to enable or disable interrupts.

(g) **Supervisor:** Indicate whether the CPU is executing in supervisor or user mode. Certain privileged instructions can be executed only in supervisor mode, and certain areas of memory can be accessed only in supervisor mode.

Apart from these, a number of other registers related to status and control might be found in a particular CPU design. In addition to the PSW, there may be a pointer to a block of memory containing additional status information (e.g. process control blocks).

Essential Registers for Instruction Execution

The following figure shows simple design of CPU datapath and control.



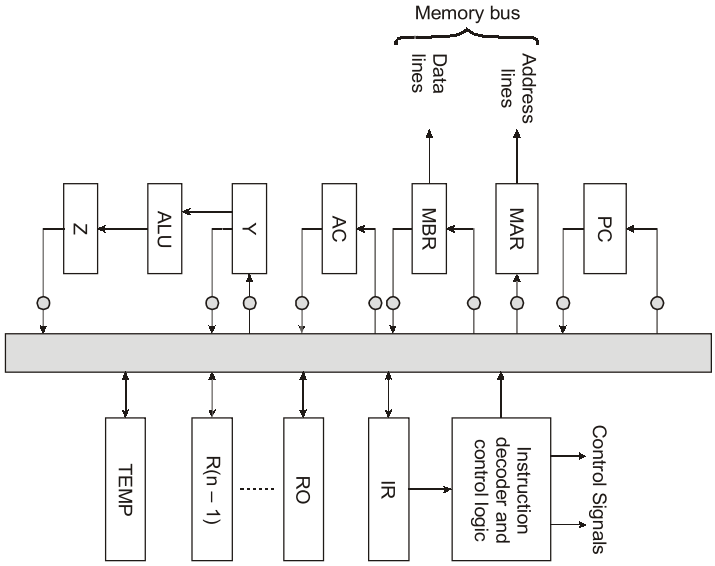
- **Program Counter (PC):** Contains the address of an instruction to be fetched. Typically, the PC is updated by the CPU after each instruction fetched so that it always points to the next instruction to be executed. A branch or skip instruction will also modify the contents of the PC.
- **Instruction Register (IR):** Contains the instruction most recently fetched. The fetched instruction is loaded into an IR, where the opcode and operand specifiers are analyzed.
- **Memory Address Register (MAR):** Contain address of a location of main memory from where information has to be fetched or information has to be stored. Contents of MAR is directly connected to address bus.
- **Memory Buffer Register (MBR or MDR):** Contains a word of data to be written to memory or the word most recently read. Contents of MBR is directly connected to the data bus. It is also known as Memory Data Register (MDR).

Apart from these specific registers, we may have some temporary registers which are not visible to the user. As such, there may be temporary buffering registers at the boundary to the ALU; these registers serve as input and output registers for the ALU and exchange data with the MBR and user visible registers.

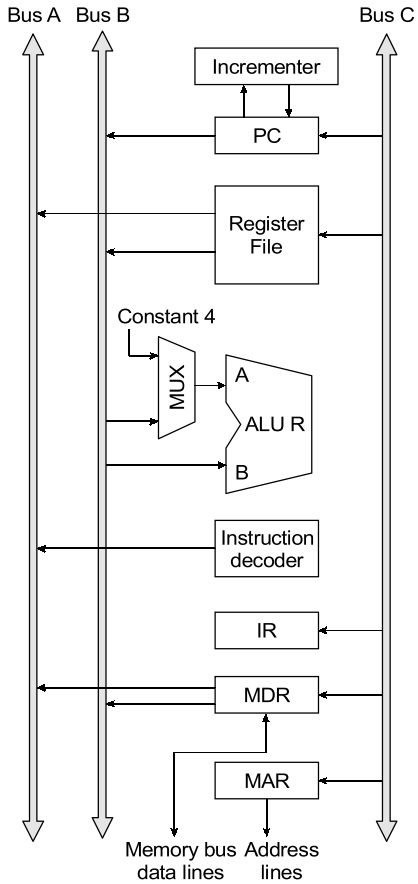
2.2.1 Bus Organizations of Datapath

Single Bus Organization (CPU Architecture)

CPU always stores the results of most calculations in one special register called Accumulator.



Three-bus Organization of Datapath (CPU Architecture)



2.2.2 Implementations of Datapath

1. **A single cycle datapath instruction implementation:** All micro-operations of an instruction are to be carried out in a single system clock cycle.
2. **Multi cycle data path instruction implementation:** In this multi-cycle design, we assume that the clock cycle can accommodate at most one of the following operations:
   (a) A memory access,
   (b) A register file access (two reads or one write), or
   (c) An ALU operation.

Hence, any data produced by one of these three functional units (the memory, the register file, or the ALU) must be saved, into a temporary register for use on a later cycle. If it were not saved then the possibility of a timing race could occur, leading to the use of an incorrect value.

Example - 2.1 Suppose every instruction is one word long, as well as every address. How may memory accesses require the following instructions?

`ADD` r_1, r_2, r_3; Register addressing

`ADD` r_1, r_2, (r_3); Direct addressing

`ADD` r_1, r_2, @r_3; Memory indirect addressing

Solution:

ADD r_1, r_2, r_3 → require only one memory access, reading the instruction.

ADD r_1, r_2, (r_3) → require two memory access, the first to read the instruction and the other one to read the value from memory location.

Add r_1, r_2, @ r_3 → require three memory address, first to read the instruction second to get M[r_3], and third one to get M[m[r_3]].

Example - 2.2 A 16 bit register contains the binary configuration:

0000 0000 0001 0010

that is the register contain 18_{10} in an unsigned representation. Show the register's content after left shifting by one, two and three positions respectively, and the decimal representation of the number?

Solution:

One bit left shift: 0000 0000 0010 0100

It is the binary representation of 36.

$$36 = 18 * 2$$

Two bit left shift: 0000 0000 0100 1000

It is the binary representation of 72.

$$72 = 18 * 4$$

Three bit left shift: 0000 0000 1001 0000

It is the binary representation of 144.

$$144 = 18 * 8$$

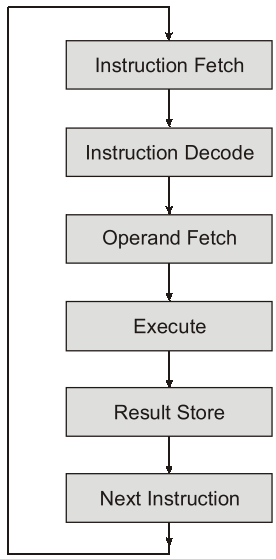
Example - 2.3 An array of two integer (each integer = 32 bits) is placed in memory starting with address 100. Show how to increment each element of the array using register addressing mode.

Solution:

```
LOAD   r1, 100        # the base
LOAD   r2, 1          # 1 will be used for increment
ADD    r3, r2, (r1)
STORE  (r1), r3
ADD    r1, r1, 4      # the next array element is at 104
STORE  (r1), r3
```

2.3 Control Unit

A CPU contains three main sections: the register section, the arithmetic/logic unit (ALU), and the control unit. These sections work together to perform the sequences of micro-operations needed to perform the fetch, decode, and execution of every instruction in the CPU's instruction set (shown in the following Figure). The operation or task that must perform by CPU to execute an instruction are:



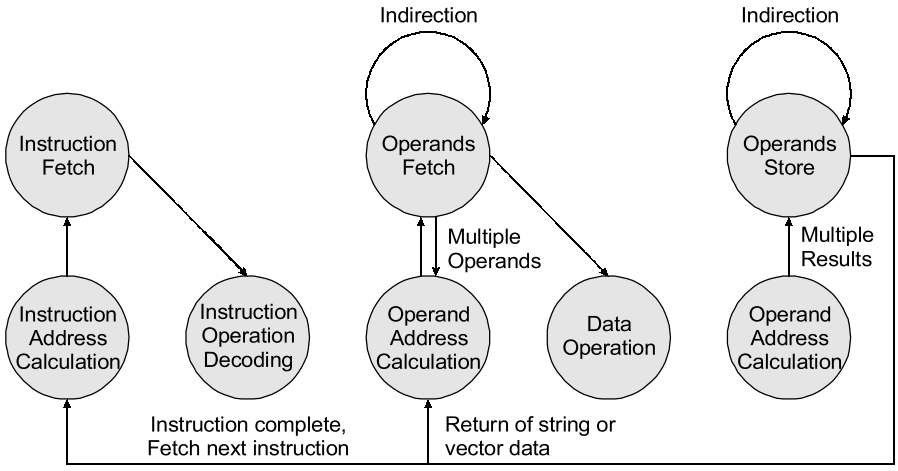
Instruction Execution Cycle

1. **Fetch Instruction:** An instruction, stored in the memory, is fetched into the control unit by supplying the memory with the address of the instruction.
2. **Decode (Interpret Instruction):** The control unit decodes the instruction in order to find the sequence of operation necessary to execute it.
3. **Memory (Operand Fetch/Data Fetch):** Any data necessary for an instruction is fetched from the memory by the control unit and stored in the datapath.
4. **Execute (Process data):** The operation (Arithmetic or Logical) is performed within the datapath.
5. **Write (result store):** The result of the operation is possibly written to the memory.

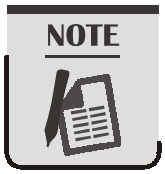
Instruction Cycle with States

For any given instruction cycle, some states may be null and other may be visited more than once.

- **Instruction Address Calculation:** Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number of the address of the previous instruction.
- **Instruction Fetch:** Read instruction from the memory location into the processor.
- **Instruction Operation Decoding:** Analyze instruction to determine type of operation to be performed and operand(s) to be used.
- **Operand Address Calculation:** If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.
- **Operand Fetch:** Fetch the operand from memory or read it from I/O.
- **Data Operation:** Perform the operation indicated in the instruction.
- **Operand Store:** Write the result into memory or out to I/O.



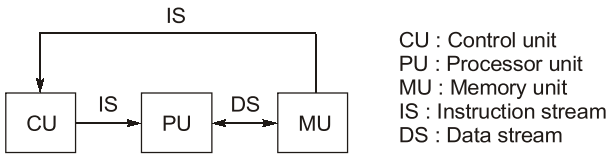
State Diagram for Instruction Cycle



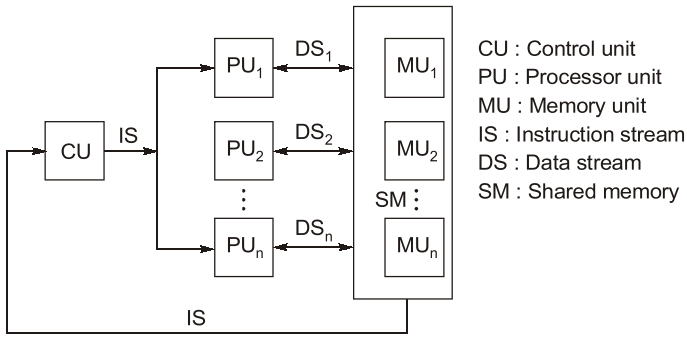
In some machines, a single instruction can specify an operation to be performed on a vector (one-dimensional array) of numbers or a string of characters. This would involve repetitive operand fetch and/or store operation for a particular opcode (opcode is fetched only once).

Flynn's Classification

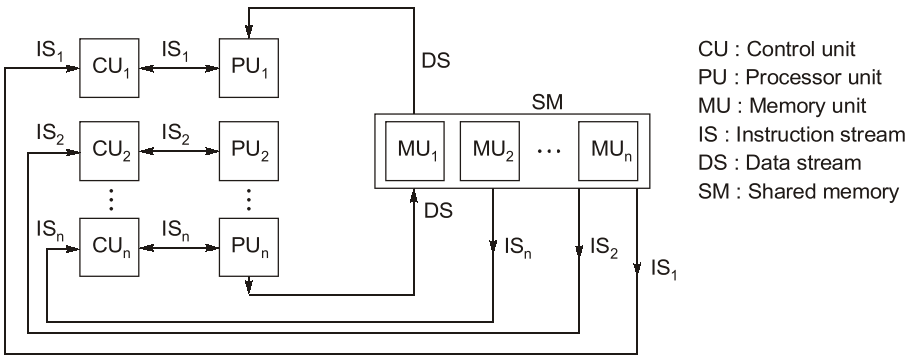
1. **Single Instruction Stream, Single Data Stream (SISD):** A computer with a single processor is called a Single Instruction Stream, Single Data Stream (SISD) Computer. It represents the organization of a single computer containing a control unit, a processor unit, and a memory unit. Instructions are executed sequentially and the system may or may not have internal parallel processing. Parallel processing may be achieved by means of a pipeline processing.
   In such a computer a single stream of instructions and a single stream of data are accessed by the processing elements from the main memory, processed and the results are stored back in the main memory. SISD computer organization is shown in figure below.



2. **Single Instruction Stream, Multiple Data Stream (SIMD):** It represents an organization of computer which has multiple processors under the supervision of a common control unit. All processors receive the same instruction from the control unit but operate on different items of the data. SIMD computers are used to solve many problems in science which require identical operations to be applied to different data set synchronously. Examples are added a set of matrices simultaneously, such as $\sum_i \sum_k (a_{ik} + a_{ik})$. Such computers are known as array processors. SIMD computer organization is shown in figure below.



3. **Multiple Instruction Stream, Single Data Stream (MISD):** It refers to the computer in which several instructions manipulate the same data stream concurrently. In the structure different processing element run different programs on the same data. This type of processor may be generalized using a 2-dimensional arrangement of processing element. Such a structure is known as systolic processor. MISD computer organization is shown in figure below.



4. **Multiple Instruction Stream, Multiple Data Stream (MIMD):** MIMD computers are the general purpose parallel computers. Its organization refers to a computer system capable of processing several programs at a same time. MIMD systems include all multiprocessing systems. MIMD computer organization is shown in figure below.]