UNIT-1 OVERVIEW AND INSTRUCTIONS

STORE PROGRAM CONTROL CONCEPT:

The term Stored Program Control Concept refers to the storage of instructions in computer memory to enable it to perform a variety of tasks in sequence or irregularly.

The idea was introduced in the late 1040s by John von Neumann who proposed that a program be electronically stored in the binary-number format in a memory device so that instructions could be modified by the computer as determined by intermediate computational results.

ENIAC (Electronic Numerical Integrator and Computer) was the first computing system designed in the early 1940s. It was based on Stored Program Concept in which machine use memory for processing data.

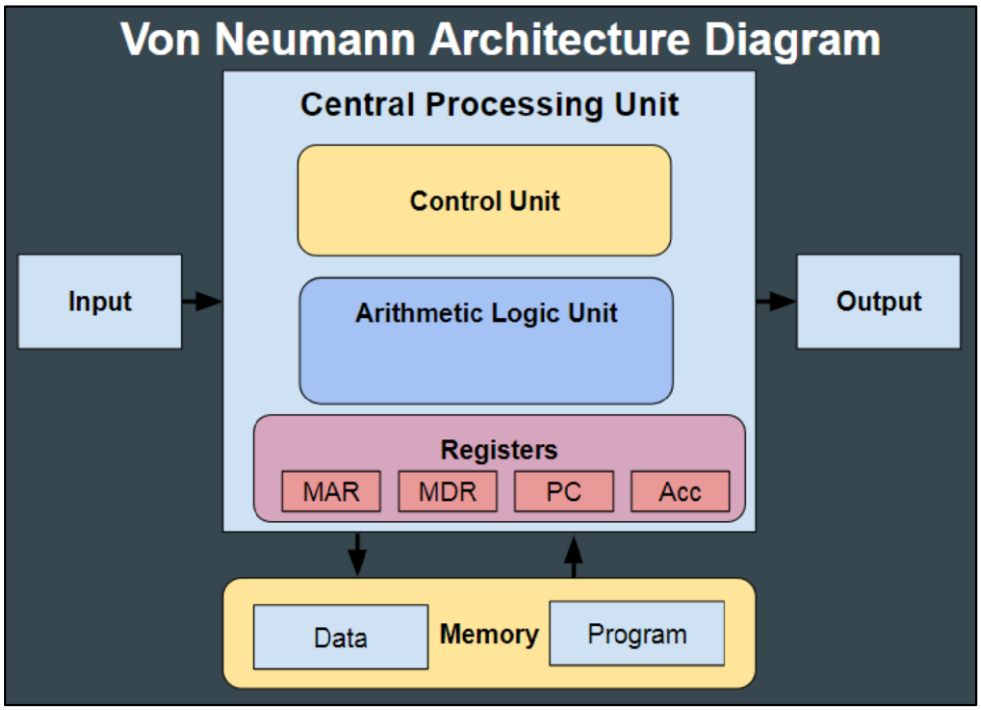
Stored Program Concept can be further classified in three basic ways:

1. Von-Neumann Model
2. General Purpose System
3. Parallel Processing

Von-Neumann Model:

- Von-Neumann proposed his computer architecture design in 1945 which was later known as Von-Neumann Architecture.
- It consisted of a Control Unit, Arithmetic, and Logical Memory Unit (ALU), Registers and Inputs/Outputs.
- Von Neumann architecture is based on the stored-program computer concept, where

instruction data and program data are stored in the same memory.



A Von Neumann-based computer:

- Uses a single processor
- Uses one memory for both instructions and data.
- Executes programs following the fetch-decode-execute cycle

Components of Von-Neumann Model: Central Processing Unit**,** Buses and Memory Unit. **1. Central Processing Unit:**

- The part of the Computer that performs the bulk of data processing operations is called the Central Processing Unit and is referred to as the CPU.
- The Central Processing Unit can also be defined as an electric circuit responsible for executing the instructions of a computer program.
- The CPU performs a variety of functions dictated by the type of instructions that are incorporated in the computer.

The major components of CPU are: ALU, Control Unit and registers.

(i) Arithmetic and Logic Unit (ALU): It performs the required micro-operations for executing the instructions. In simple words, ALU allows arithmetic (add, subtract, etc.) and logic (AND, OR, NOT, etc.) operations to be carried out.

(ii) Control Unit: It controls the operations of components like ALU, memory and input/output devices. It consists of a program counter that contains the address of the instructions to be fetched and an instruction register into which instructions are fetched from memory for execution.

(iii) Registers: Registers refer to high-speed storage areas in the CPU. The data processed by the CPU are fetched from the registers.

Following is the list of registers that plays a crucial role in data processing:

Registers	Description
MAR (Memory Address Register)	This register holds the memory location of the data that needs to be accessed.
MDR (Memory Data Register)	This register holds the data that is being transferred to or from memory.
AC (Accumulator)	This register holds the intermediate arithmetic and logic results.
PC (Program Counter)	This register contains the address of the next instruction to be executed.
CIR (Current Instruction Register)	This register contains the current instruction during processing.

2. Buses:

- Buses are the means by which information is shared between the registers in a multipleregister configuration system.
- A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.
- Control signals determine which register is selected by the bus during each particular register transfer.

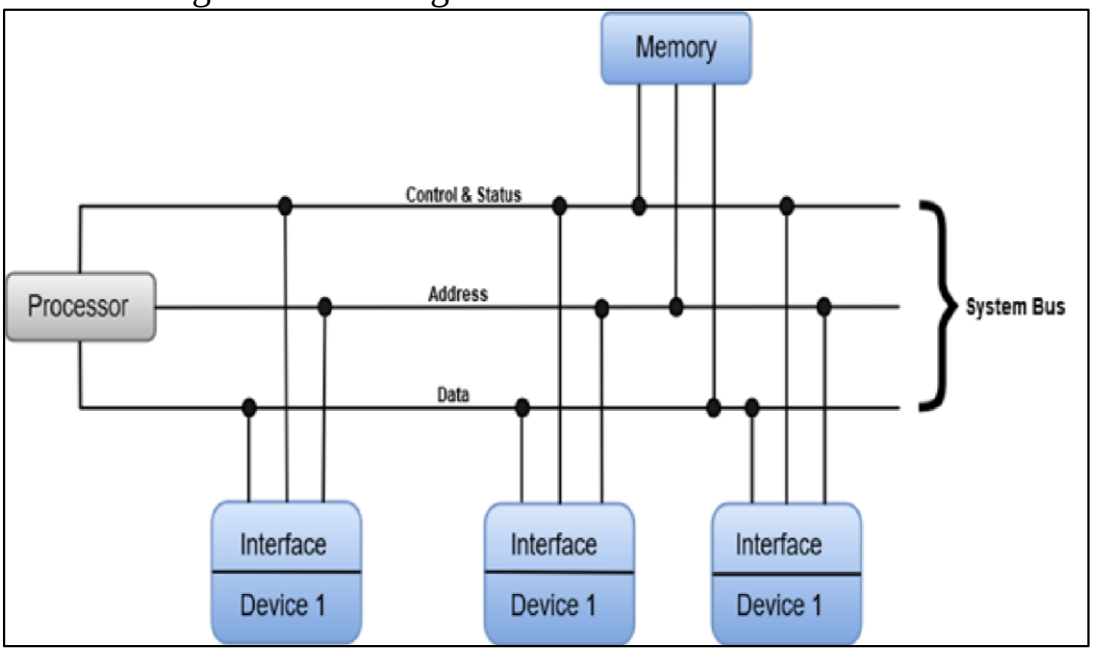
Von-Neumann Architecture comprised of three major bus systems for data transfer.

Bus	Description
Address Bus	Address Bus carries the address of data (but not the data) between the processor and the memory.
Data Bus	Data Bus carries data between the processor, the memory unit and the input/output devices.
Control Bus	Control Bus carries signals/commands from the CPU.

3. Memory Unit:

- A memory unit is a collection of storage cells together with associated circuits needed to transfer information in and out of the storage.
- The memory stores binary information in groups of bits called words.

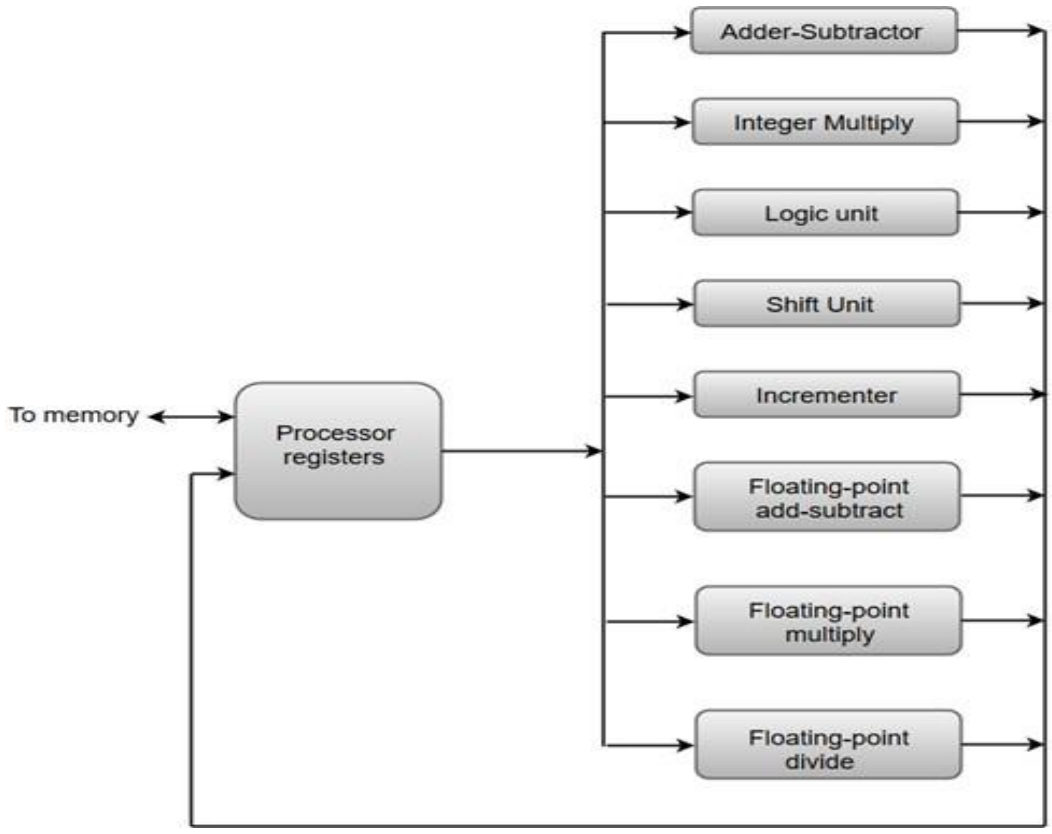
- The internal structure of a memory unit is specified by the number of words it contains and the number of bits in each word.
- Two major types of memories are used in computer systems: RAM (Random Access Memory) and ROM (Read-Only Memory) **General Purpose System:**
- The General Purpose Computer System is the modified version of the Von-Neumann Architecture. In simple words, we can say that a general purpose computer system is a modern day architectural representation of Computer System.
- The CPU (Central Processing Unit) consists of the ALU (Arithmetic and Logic Unit), Control Unit and various processor registers.
- The CPU, Memory Unit and I/O subsystems are interconnected by the system bus which includes data, address, and control-status lines.
- The following image shows how CPU, Memory Unit and I/O subsystems are connected through common single bus architecture.



Parallel Processing:

- Parallel processing can be described as a class of techniques which enables the system to achieve simultaneous data-processing tasks to increase the computational speed of a computer system.
- A parallel processing system can carry out simultaneous data-processing to achieve faster execution time. **For example,** while an instruction is being processed in the ALU component of the CPU, the next instruction can be read from memory.
- The primary purpose of parallel processing is to enhance the computer processing capability and increase its quantity, i.e. the amount of processing that can be accomplished during a given interval of time.

- A parallel processing system can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. The data can be distributed among various multiple functional units.
- The following diagram shows one possible way of separating the execution unit into eight functional units operating in parallel.
- The adder & integer multiplier performs the arithmetic operation with integer numbers.
- The logic, shift, and increment operations can be performed simultaneously on different data.
- The floating-point operations are separated into three circuits operating in parallel.
- All units are independent of each other, so one number can be shifted while another number is being incremented.
- The operation performed in each functional unit is indicated in each block of the diagram:

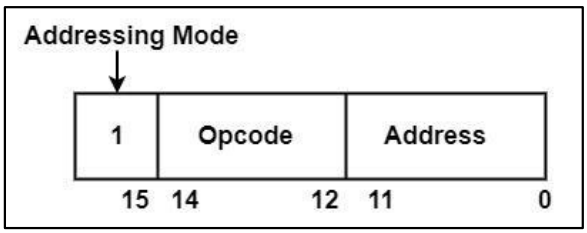


INSTRUCTION CODES:

- A computer instruction is a binary code that determines the micro-operations in a sequence for a computer. They are saved in the memory along with the information.
- Each computer has its specific group of instructions.

- They can be categorized as follows:
  - **Opcodes** specify the operation for specific instructions.
  - **Operands** are definite elements of computer instruction that show what information is to be operated on.
  - **Address** determines the registers or the areas that can be used for that operation.
- It consists of 12 bits of memory that are required to define the address as the memory includes 4096 words.
- The 15th bit of the instruction determines the addressing mode (where **direct addressing** corresponds to **0**, **indirect addressing** corresponds to **1**).
- Therefore, the instruction format includes
  - 12 bits of address.
  - 3 bits for Opcodes.
  - 1 bit for the addressing mode.

The following block diagram shows the instruction format for a basic computer.



1. What are the types of Instruction Code Format?

There are three parts of the Instruction Format which are as follows −

1. Addressing Modes:

- Instructions that define the address of a definite memory location are known as **memory reference instructions**.
- The method in which a target address or effective address is known within the instruction is known as **addressing mode**.
- The address field for instruction can be represented in two different ways are as follows:
  - **Direct Addressing** − It uses the address of the operand.
  - **Indirect Addressing** − It facilitates the address as a pointer to the operand.
- **Effective Address (EA)**: An effective address is the location of an operand which is stored in memory.

2. Opcodes:

- An opcode is a collection of bits that represents the basic operations including add, subtract, multiply, complement, and shift.
- The total number of operations provided through the computer determines the number of bits needed for the opcode.
- These operations are implemented on information that is saved in registers or memory.

3. Address/ Operand:

- The address is represented as the location where a specific instruction is constructed in the memory.
- The address bits of an instruction code is used as an operand and not as an address. In such methods, the instruction has an **immediate operand**.
- If the second part has an address, the instruction is referred to have a **direct address**.
- There is another possibility in the second part including the address as a pointer to operand. This is referred to as an **indirect address**.
- In the instruction code, one bit can signify if the direct or indirect addressing mode is executed.

Types of Instruction Code: The following are some common types of instruction codes:

1. **One-operand instructions:**
    - These instructions have one operand and perform an operation on that operand.
    - For example, the **"neg" instruction** in the x86 assembly language negates the value of a single operand.
2. **Two-operand instructions:**
    - These instructions have two operands and perform an operation involving both.
    - For example, the **"add" instruction** in x86 assembly language adds two operands together.
3. **Three-operand instructions:**
    - These instructions have three operands and perform an operation that involves all three operands.
    - For example, the **"fma" (fused multiply-add) instruction** in some processors multiplies two operands together, adds a third operand, and stores the result in a fourth operand.
4. **Data transfer instructions:**

- These instructions move data between memory and registers or between registers.
- For example, the **"mov" instruction** in the x86 assembly language moves data from one location to another.

5. **Control transfer instructions:**
   - These instructions change the flow of program execution by modifying the program counter.
   - For example, the **"jmp" instruction** in the x86 assembly language jumps to a different location in the program.
6. **Arithmetic instructions:**
   - These instructions perform mathematical operations on operands.
   - For example, the **"add" instruction** in x86 assembly language adds two operands together.
7. **Logical instructions:**
   - These instructions perform logical operations on operands.
   - For example, the **"and" instruction** in x86 assembly language performs a bitwise AND operation on two operands.
8. **Comparison instructions:**
   - These instructions compare two operands and set a flag based on the result.
   - For example, the **"cmp" instruction** in x86 assembly language compares two operands and sets a flag indicating whether they are equal, greater than, or less than.

9. **Floating-point instructions:**
   - These instructions perform arithmetic and other operations on floating-point numbers.
   - For example, the **"fadd" instruction** in the x86 assembly language adds two floating-point numbers together.

2. Define Computer Registers. What are its types?

COMPUTER REGISTERS:

- Computer registers are memory storing units that operate at high speed.
- It's a component of a computer's processor.
- Registers are a type of computer memory used to accept, store, and transfer data and instructions used by the CPU right away.

- It can hold any type of data, including a bit sequence or a single piece of data.
- During the execution of a program, registers are used to store data temporarily.

The following are the various computer registers and their functions:

1. **Accumulator Register (AC):**
   - It is a general-purpose Register.
   - The initial data to be processed, the intermediate result, and the final result of the processing operation are all stored in this register.
   - If no specific address for the result operation is specified, the result of arithmetic operations is transferred to AC.
   - The number of bits in the accumulator register equals the number of bits per word.
   - A **word** is the natural unit of data used by a particular processor design. A word is a fixed-sized piece of data handled as a unit by the instruction set.
2. **Program Counter (PC):**
   - A program counter (PC) is a CPU register in the computer processor which has the address of the next instruction to be executed from memory.
   - It is a digital counter needed for faster execution of tasks as well as for tracking the current execution point.
3. **Address Register (AR):**
   - It is the address of the memory location or Register where data is stored or retrieved.
   - The size of the Address Register is equal to the width of the memory address is directly related to the size of the memory because it contains an address.
4. **Data Register (DR):**
   - The operand is stored in the Data Register from memory.
   - When a direct or indirect addressing operand is found, it is placed in the Data Register.
   - This value was then used as data by the processor during its operation. It's about the same size as a word in memory.
5. **Instruction Register (IR):**
   - The instruction is stored in the Instruction Register.
   - The instruction register contains the currently executed instruction. Because it includes instructions, the number of bits in the Instruction Register equals the number of bits in the instruction, which is n bits for an n-bit CPU.

6. **Temporary Register (TR):**
   - The Temporary Register is used to hold data while it is being processed. As Temporary Register stores data, the number of bits it contains is the same as the number of bits in the data word.
7. **Input Register (INPR):** Input Register is a register that stores the data from an input device. The computer's alphanumeric code determines the size of the input register.
8. **Output Register (OUTR):** The data that needs to be sent to an output device is stored in the Output Register. Its size is determined by the alphanumeric code used by the computer.

Following is the list of some of the most common registers in computer:

Register	**Symbol**	**No. of Bits**	**Function**
Accumulator	AC	16	It's a processor register.
Program counter	PC	12	It stores the address of the instruction.
Address Register	AR	12	It is used for storing memory addresses.
Data Register	DR	16	A general-purpose register used for storing data during calculations.
Instruction Register	IR	16	It stores current instruction being executed.
Temporary Register	TR	16	It holds the temporary data.
Input Register	INPR	8	It carries the input character.
Output Register	OUTR	8	It carries the output character

3. What do you mean by Instruction Cycle?

INSTRUCTION CYCLE:

- It is also known as **Fetch-Execute-Cycle.**
- The instruction cycle is a basic computer system that deals with the central processor unit's core operations.
- The instruction cycle is defined as the basic cycle in which a computer system fetches an instruction from memory, decodes it, and then executes it.
- All instructions in a computer system are executed in the RAM of the computer system. The CPU is in charge of carrying out the instruction.

Each instruction cycle in a basic computer includes the following procedures:

- It has the ability to retrieve instructions from memory.
- It's used to decode the command.
- If the instruction has an indirect address, it can read the effective address from memory.
- It is capable of carrying out the command.

The instruction cycle is divided into five stages, which are described below:

1. Initiating Cycle
2. Fetching of Instruction
3. Decoding of Instruction
4. Read of an Effective Address
5. Execution of Instruction **Initiating Cycle**:

During this phase, the computer system boots up and the Operating System loads into the central processing unit's main memory. It begins when the computer system starts.

Fetching of Instruction:

- The **first phase** is **instruction retrieval**.
- Each instruction executed in a central processing unit uses the fetch instruction.
- During this phase, the central processing unit sends the PC to MAR and then the READ instruction to a control bus.
- After sending a read instruction on the data bus, the memory returns the instruction that was stored at that exact address in the memory.
- The CPU then copies data from the data bus into MBR, which it then copies to registers.
- The pointer is incremented to the next memory location, allowing the next instruction to be fetched from memory.

Decoding of Instruction:

- The **second phase** is **instruction decoding**. During this step, the CPU determines which instruction should be fetched from the instruction and what action should be taken on the instruction.
- The instruction's opcode is also retrieved from memory, and it decodes the related operation that must be performed for the instruction.

Read of an Effective Address:

- The **third phase** is the **reading of an effective address**. The operation's decision is made during this phase.
- Any memory type operation or non-memory type operation can be used.
- Direct memory instruction and indirect memory instruction are the two types of memory instruction available.

Execution of Instruction:

- The **last step** is to **carry out the instructions**. The instruction is finally carried out at this stage and the result is saved in the register.
- The CPU gets prepared for the execution of the next instruction after the completion of each instruction.
- The execution time of each instruction is calculated, and this information is used to determine the processor's processing speed.

Why do we need an Instruction Cycle?

- The instruction cycle of a computer system is necessary for understanding the flow of instructions and the execution of an instruction in a computer processor.
- It is responsible for the complete flow of instructions from the start of the computer system till its shutdown.
- The instruction cycle helps to understand the internal flow of the central processing unit, allowing any faults to be immediately resolved.
- It deals with a computer processor's basic operations and demands a detailed understanding of the many steps involved.
- It is common for all instruction sets to require a thorough understanding to perform all operations efficiently.
- The processing time of a programme can be easily calculated using the instruction cycle, which helps in determining the processor's speed. The processor's speed determines how many instructions can be executed simultaneously in the central processing unit.
- All instructions for the computer processor system follow the fetch-decode-execute cycle.

4.Briefly describe Memory Reference Instructions.

MEMORY REFERENCE INSTRUCTIONS:

Memory reference instructions are those commands or instructions which are in the custom to generate a reference to the memory and approval to a program to have an approach to

the instructed information and that states as to from where the data is cache continually. These instructions are known as Memory Reference Instructions.

There are seven memory reference instructions which are as follows:

1. **AND:**
   The AND instruction implements the AND logic operation on the bit collection from the register and the memory word that is determined by the effective address. The result of this operation is moved back to the register.

2. **ADD:**
   The ADD instruction adds the content of the memory word that is denoted by the effective address to the value of the register.

3. **LDA:**
   The LDA instruction shares the memory word denoted by the effective address to the register.

4. **STA:**
   STA saves the content of the register into the memory word that is defined by the effective address. The output is next used to the common bus and the data input is linked to the bus.
   It needed only one micro-operation.

5. **BUN:**
   The **Branch Unconditionally (BUN)** instruction can send the instruction that is determined by the effective address. They understand that the address of the next instruction to be performed is held by the PC and it should be incremented by one to receive the address of the next instruction in the sequence. If the control needs to implement multiple instructions that are not next in the sequence, it can execute the BUN instruction.

6. **BSA:**
   BSA stands for **Branch and Save-return Address**. These instructions can branch a part of the program (known as subroutine or procedure). When this instruction is performed, BSA will store the address of the next instruction from the PC into a memory location that is determined by the effective address.

7. **ISZ:**
   The **Increment if Zero (ISZ)** instruction increments the word determined by effective address. If the incremented cost is zero, thus PC is incremented by 1. A negative value is saved in the memory word through the programmer. It can influence the zero value after getting incremented repeatedly. Thus, the PC is incremented and the next instruction is skipped.

5. Briefly describe Flynn's classification of computers.

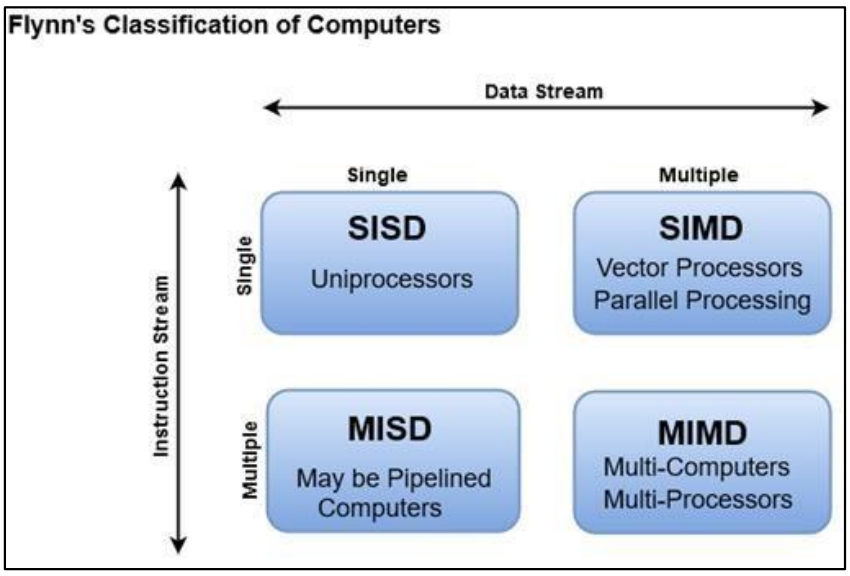
FLYNN'S CLASSIFICATION OF COMPUTERS:

- M.J. Flynn proposed a classification for the organization of a computer system by the number of instructions and data items that are manipulated simultaneously.
- The sequence of instructions read from memory constitutes an **instruction stream**.
- The operations performed on the data in the processor constitute a **data stream**.

Note: The term 'Stream' refers to the flow of instructions or data.

Flynn's classification divides computers into four major groups that are:

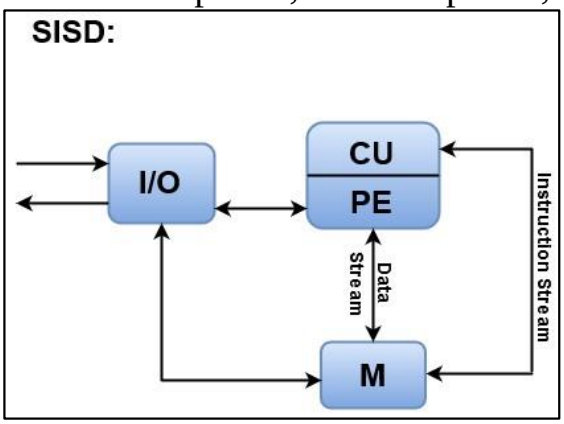
1. Single instruction stream, single data stream (SISD)
2. Single instruction stream, multiple data stream (SIMD)
3. Multiple instruction stream, single data stream (MISD)
4. Multiple instruction stream, multiple data stream (MIMD)



1. SISD: (Single Instruction and Single Data Stream)

- SISD stands for ***'Single Instruction and Single Data Stream'***.
- 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 capabilities.
- Most conventional computers have SISD architecture like the traditional Von-Neumann computers.
- Parallel processing, in this case, may be achieved by means of multiple functional units or by pipeline processing.

- Examples: Older generation computers, minicomputers, and workstations

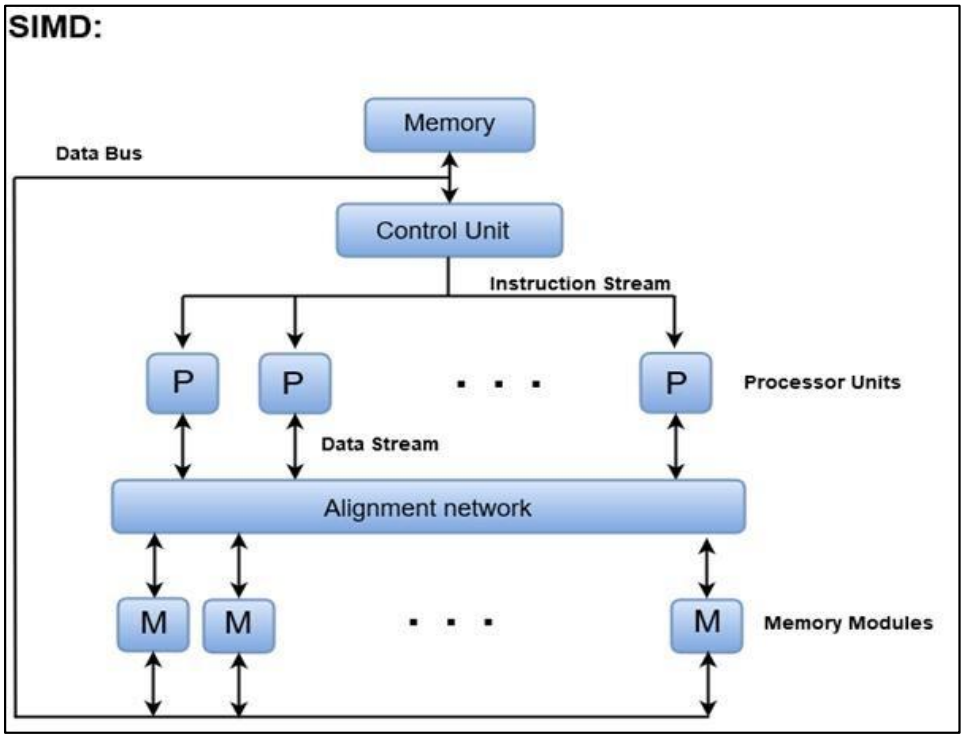


- In above diagram, CU = Control Unit, PE = Processing Element, M = Memory Unit
- Instructions are decoded by the Control Unit and then the Control Unit sends the instructions to the processing units for execution.
- Data Stream flows between the processors and memory bi-directionally.

2. SIMD: (*Single Instruction and Multiple Data Stream*)

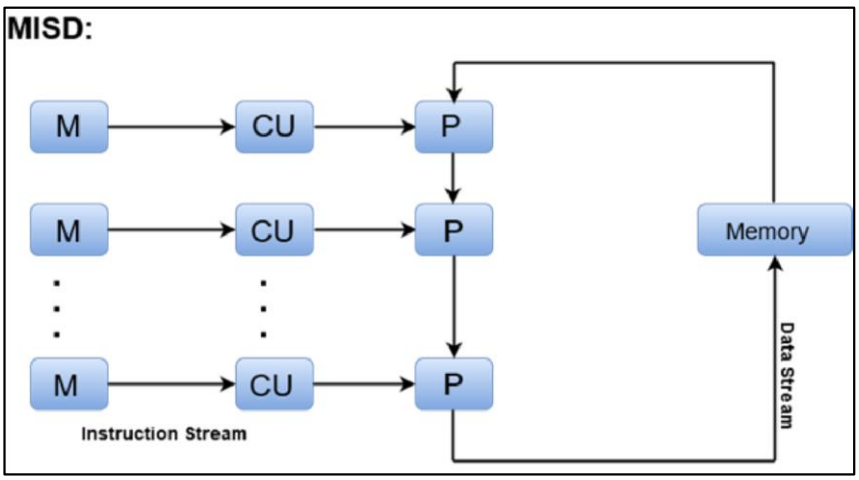
- **SIMD** stands for ***'Single Instruction and Multiple Data Stream'***.
- It represents an organization that includes many processing units under the supervision of a common control unit.
- All processors receive the same instruction from the control unit but operate on different items of data.
- The shared memory unit must contain multiple modules so that it can communicate with all the processors simultaneously.

- SIMD is mainly dedicated to array processing machines.



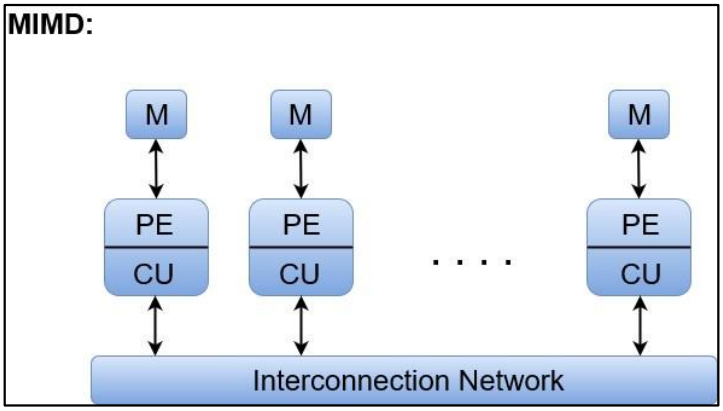
3. MISD: (*Multiple Instruction and Single Data stream)*

- MISD stands for ***'Multiple Instruction and Single Data stream'***.
- MISD structure is only of theoretical interest since no practical system has been constructed using this organization.
- In MISD, multiple processing units operate on one single-data stream. Each processing unit operates on the data independently via separate instruction stream.
- Here, M = Memory Unit, CU = Control Unit, P = Processor Units
- Example: The experimental Carnegie-Mellon computer (1971)



4. MIMD: (*Multiple Instruction and Multiple Data stream)*

- MIMD stands for ***'Multiple Instruction and Multiple Data Stream'***.
- In this organization, all processors in a parallel computer can execute different instructions and operate on various data at the same time.
- In MIMD, each processor has a separate program and an instruction stream is generated from each program.
- Here, M = Memory Module, PE = Processing Element, and CU = Control Unit
- Examples: Cray T90, Cray T3E, IBM-SP2



INSTRUCTION SET ARCHITECTURE:

Instruction Set Based Classification of Processors (RISC, CISC and Their Comparisons)

RISC PROCESSOR:

- RISC stands for **Reduced Instruction Set Computer Processor**, a microprocessor architecture with a simple collection and highly customized set of instructions.
- It is built to minimize the instruction execution time by optimizing and limiting the number of instructions. It means each instruction cycle requires only one clock cycle, and each cycle contains three parameters: fetch, decode and execute.
- It is used to perform various complex instructions by combining them into simpler ones.
- RISC chips require several transistors, making it cheaper to design and reduce the execution time for instruction.
- Examples: SUN's SPARC, PowerPC, Microchip PIC processors, RISC-V.

Features of RISC Processor:

1. Simpler instruction, hence simple instruction decoding.
2. Instruction comes undersize of one word.
3. Instruction takes a single clock cycle to get executed.
4. More general-purpose registers.

5. Simple Addressing Modes and Fewer Data types.
6. A pipeline can be achieved.

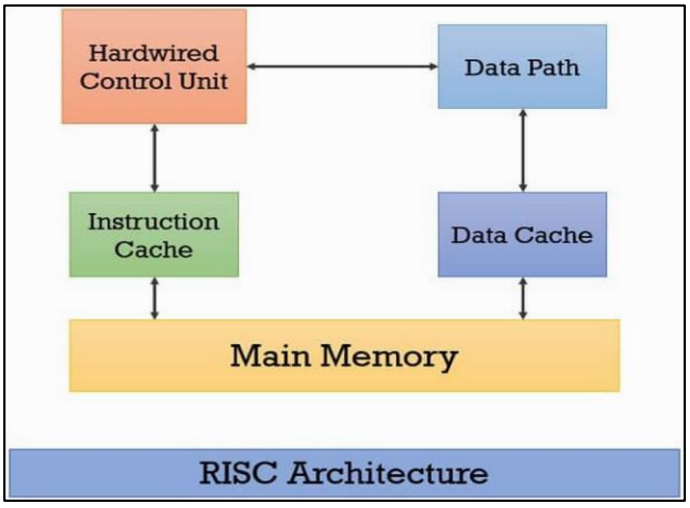
Advantages of RISC:

1. **Simpler instructions:** RISC processors use a smaller set of simple instructions, which makes them easier to decode and execute quickly. This results in faster processing times.
2. **Faster execution:** Because RISC processors have a simpler instruction set, they can execute instructions faster than CISC processors.
3. **Lower power consumption:** RISC processors consume less power than CISC processors, making them ideal for portable devices.

Disadvantages of RISC:

1. **More instructions required:** RISC processors require more instructions to perform complex tasks than CISC processors.
2. **Increased memory usage:** RISC processors require more memory to store the additional instructions needed to perform complex tasks.
3. **Higher cost:** Developing and manufacturing RISC processors can be more expensive than CISC processors.

RISC Processor Architecture:



- RISC processor is implemented using the hardwired control unit**.** The **hardwired control unit** produces **control signals** which regulate the working of processors hardware. RISC architecture emphasizes on using the **registers** rather than memory.
- This is because the registers are the 'fastest' available memory source. The registers are physically small and are placed on the same chip where the ALU and the control unit are placed on the processor. The RISC instructions **operate** on the operands present in **processor's registers**.

- The **hardware** of RISC architecture is designed to execute the instruction quickly, which is possible because of the more precise and smaller number of instructions and a large number of registers.
- The **data path** is used to store and manipulate data in a computer. It is responsible for managing data within the processor and its movement between processor and memory. The processor uses a **cache** to reduce the access time to the main memory.
- The **instruction cache** is beneficial for retrieving and storing the data of frequently used instructions. It speeds up the process of instruction execution. The **data cache** provides storage for frequently used data from the main memory.

Example – Suppose we have to add two 8-bit numbers:

CISC approach: There will be a single command or instruction for this like ADD which will perform the task.
RISC approach: Here programmer will write the first load command to load data in registers then it will use a suitable operator and then it will store the result in the desired location.

So, add operation is divided into parts i.e. load, operate, store due to which RISC programs are longer and require more memory to get stored but require fewer transistors due to less complex command.

CISC PROCESSOR:

- The CISC Stands for **Complex Instruction Set Computer**, developed by the Intel.
- It has a large collection of complex instructions that range from simple to very complex and specialized in the assembly language level, which takes a long time to execute the instructions.
- So, CISC approaches reducing the number of instruction on each program and ignoring the number of cycles per instruction.
- However, CISC chips are relatively slower as compared to RISC chips but it uses little instruction than RISC.
- Examples of CISC processors are VAX, AMD, Intel x86 and the System/360.

Features of CISC Processor:

1. Complex instruction, hence complex instruction decoding.
2. Instructions are larger than one-word size.
3. Instruction may take more than a single clock cycle to get executed.

4. Less number of general-purpose registers as operations get performed in memory itself.
5. Complex Addressing Modes.
6. More Data types.

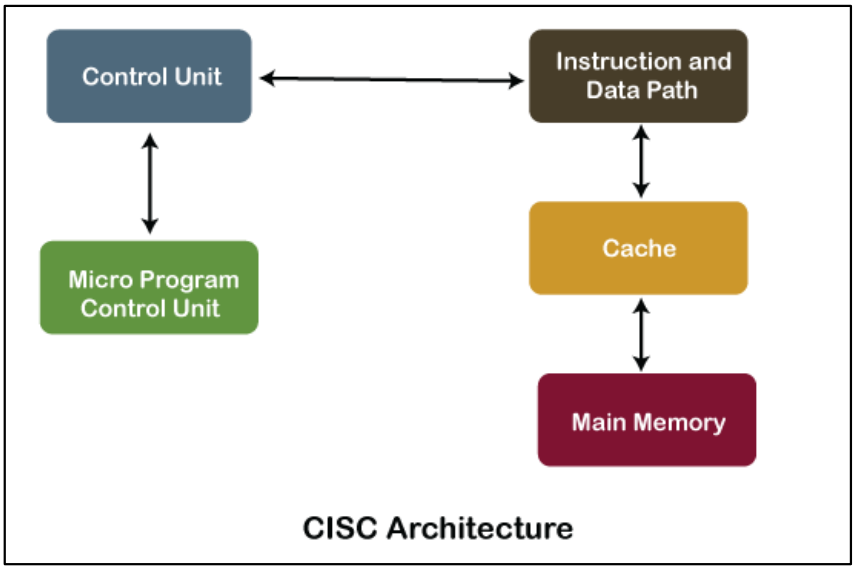
Advantages of CISC:

1. **Reduced code size:** CISC processors use complex instructions that can perform multiple operations, reducing the amount of code needed to perform a task.
2. **More memory efficient:** Because CISC instructions are more complex, they require few instructions to perform complex tasks, which can result in more memory-efficient code.
3. **Widely used:** CISC processors have been in use for a longer time than RISC processors, so they have a larger user base and more available software.

Disadvantages of CISC:

1. **Slower execution:** CISC processors take longer to execute instructions because they have more complex instructions and need more time to decode them.
2. **More complex design:** CISC processors have more complex instruction sets, which makes them more difficult to design and manufacture.
3. **Higher power consumption:** CISC processors consume more power than RISC processors because of their more complex instruction sets.

CISC Processors Architecture:



CISC Architecture

- In CISC architecture, we have a special **microprogram control unit** that uses a series of micro-instructions of the microprogram stored in the "control memory" of the microprogram control unit and generate the **control signals**.

- The **control units** access the **control signals** produced by the microprogram control unit and operate the functioning of processors hardware.
- The **Instruction and data path** fetches the opcode and operands of the instructions from the memory.
- The **Cache** and **main memory** is the location where the program instructions and operands are stored.

6. Difference between RISC and CISC architecture.

Difference between the RISC and CISC Processors:

S.No.	RISC	CISC
1.	**R**educed **I**nstruction **S**et **C**omputer.	**C**omplex **I**nstruction **S**et **C**omputer.
2	It emphasizes on software to optimize the instruction set.	It emphasizes on hardware to optimize the instruction set.
3.	It is a hard wired unit of programming in the RISC Processor.	Microprogramming unit in CISC Processor.
4	It requires multiple register sets to store the instruction.	It requires a single register set to store the instruction.
5.	It has simple decoding of instruction.	It has complex decoding of instruction.
6.	Uses of the pipeline are simple in RISC.	Uses of the pipeline are difficult in CISC.
7.	It uses a limited number of instruction that requires less time to execute the instructions.	It uses a large number of instruction that requires more time to execute the instructions.
8.	It uses LOAD and STORE that are independent instructions in the registerto-register a program's interaction.	It uses LOAD and STORE instruction in the memory-to-memory interaction of a program.
9.	RISC has more transistors on memory registers.	CISC has transistors to store complex instructions.
10.	The execution time of RISC is very short.	The execution time of CISC is longer.

11.	RISC architecture can be used with highend applications like telecommunication, image processing, video processing, etc.	CISC architecture can be used with lowend applications like home automation, security system, etc.
12.	It has fixed format instruction.	It has variable format instruction.
13.	The program written for RISC architecture needs to take more space in memory.	Program written for CISC architecture tends to take less space in memory.
14.	Example of RISC: ARM, PA-RISC, Power Architecture, Alpha, AVR, ARC and the SPARC.	Examples of CISC: VAX, Motorola 68000 family, System/360, AMD and the Intel x86 CPUs.