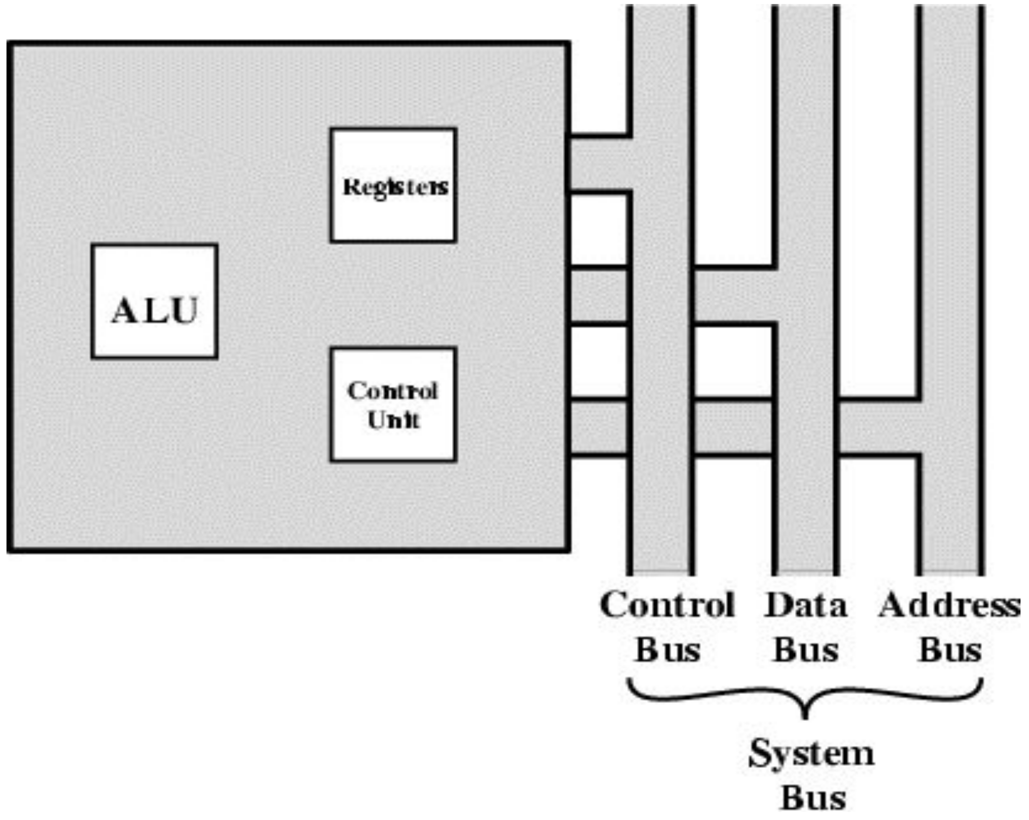
Module 3: Processor Organization and Architecture

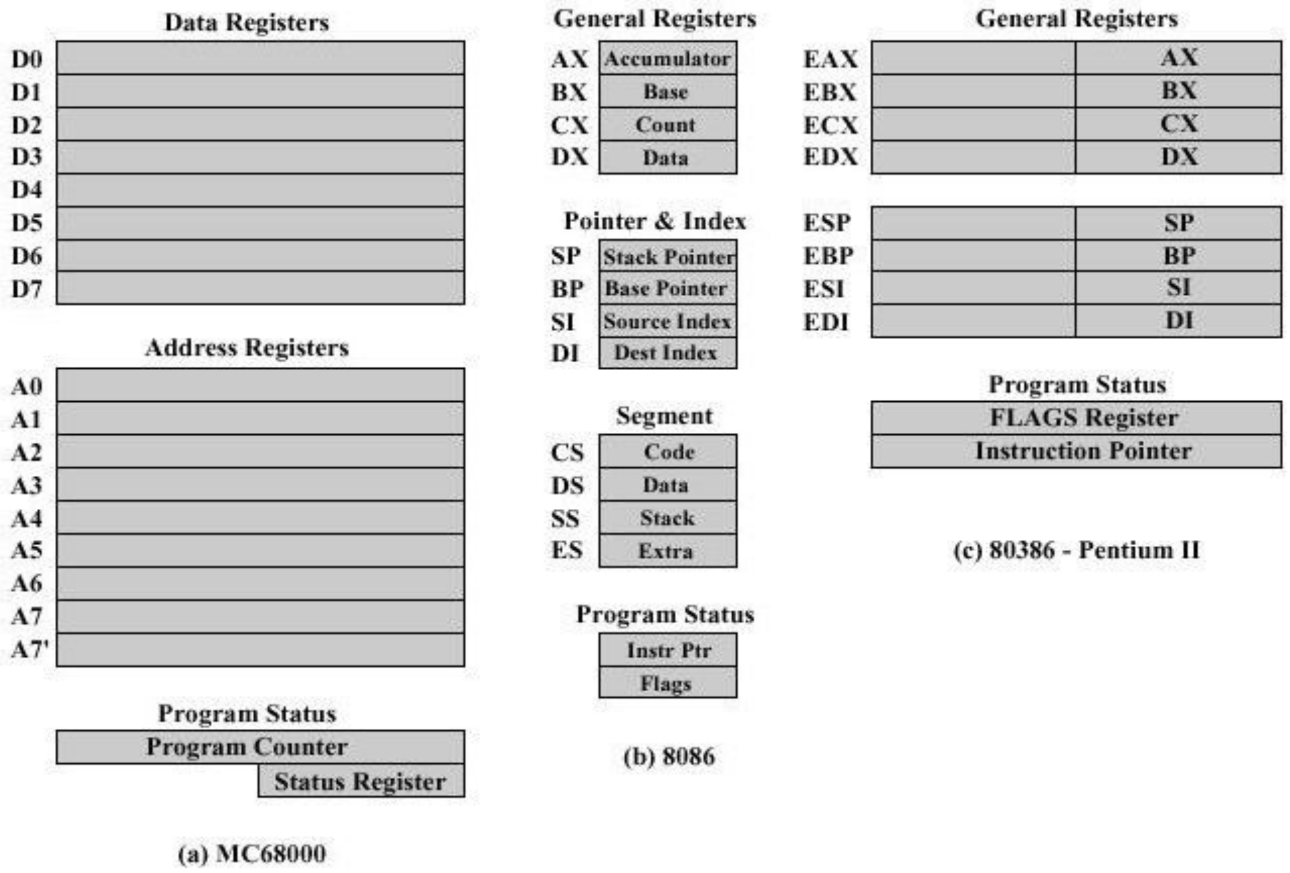
Processor organization

Things a CPU must do:

- Fetch Instructions
- Interpret Instructions
- Fetch Data
- Process Data
- Write Data

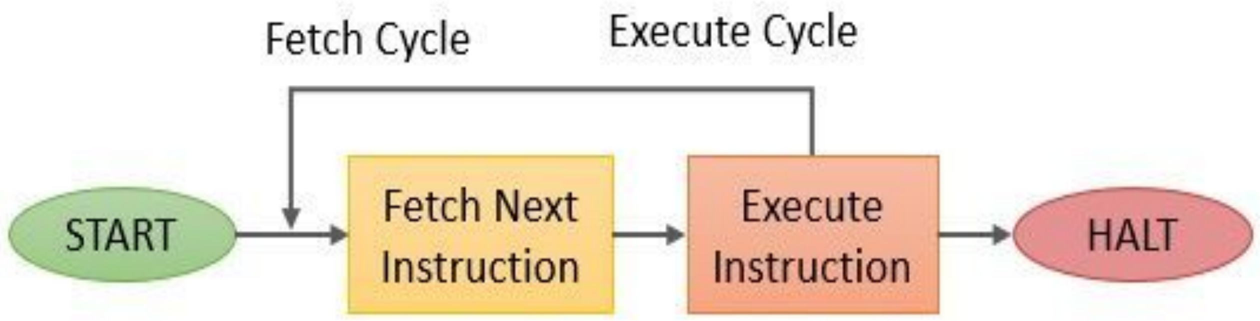


Microprocessor register organization



Basic instruction cycle

- The instruction cycle defines the processing or execution of a single instruction.
- It consist of two main steps:
  1. Fetch cycle to fetch the operation
  2. Execute cycle to execute operation.



Basic instruction cycle

- To start the execution of a program, the processor runs the fetch cycle and fetches the first instruction from the memory. Program counter (PC) holds address of the instruction to be fetched next.
- The execution cycle interprets the operation and performs the operations specified in the instruction accordingly.
- This cycle repeats until all the instructions are executed from the program and after the execution of the last instruction the instruction cycle get halt. So, this was the scenario where there were no interrupts.

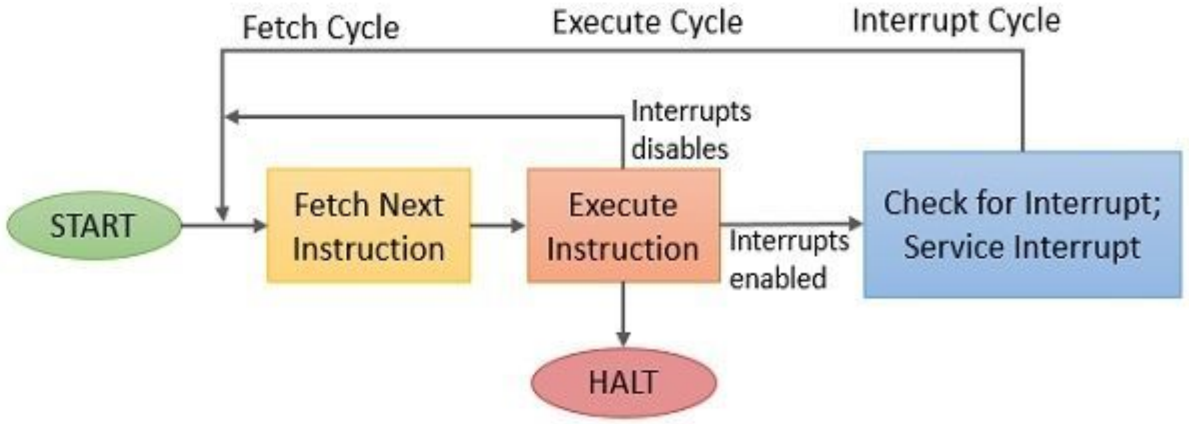
Interrupt

- Interrupt is **the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor**.
- The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

Instruction cycle with interrupt

Instruction cycle contains the following sub-cycles.

- Fetch - read next instruction from memory into CPU
- Execute - Interpret the opcode and perform the indicated operation
- Interrupt - if interrupts are enabled and one has occurred, save the current process state and service the interrupt



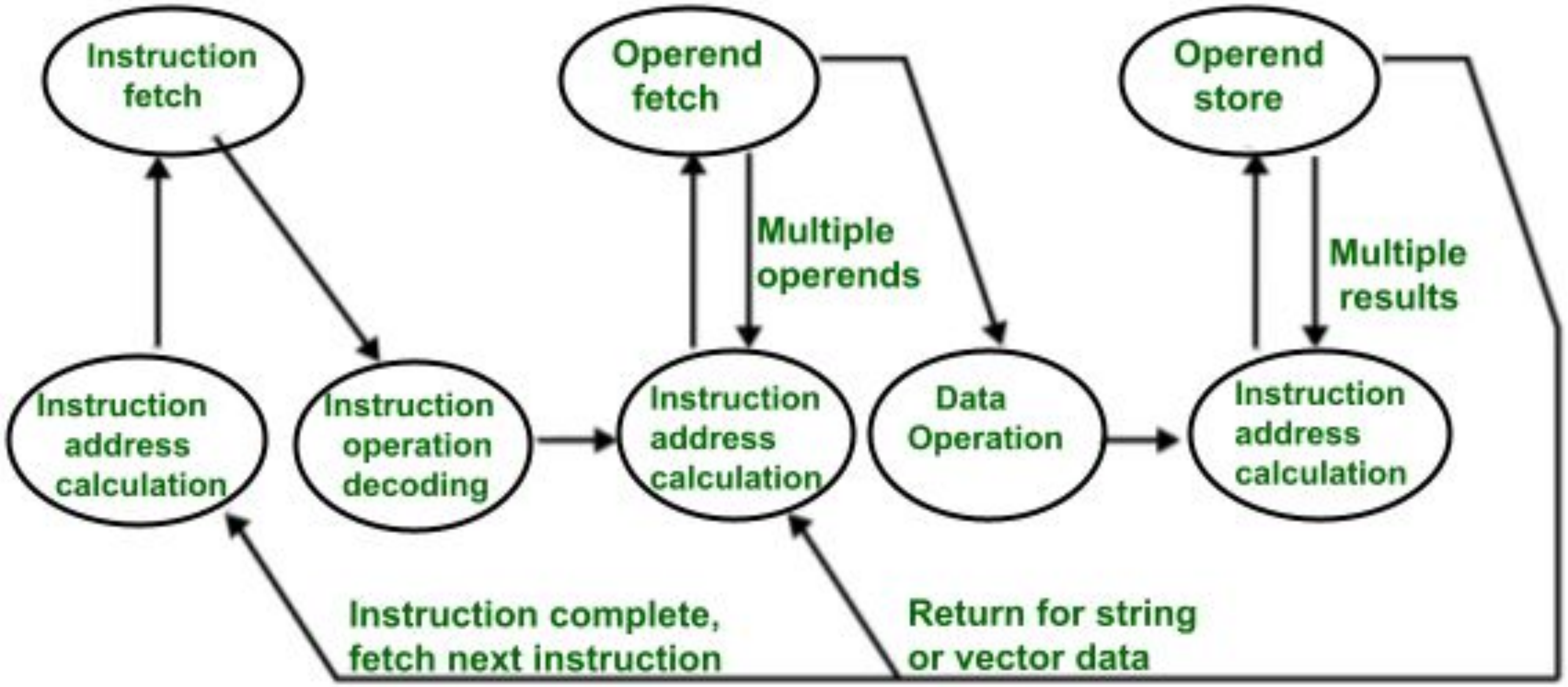
Instruction and Instruction sequencing

A computer program consist of a sequence of small steps.

A computer must have instructions capable of performing four types of operations:

- Data transfers between the memory and the processor registers
- Arithmetic and logic operations on data
- Program sequencing and control
- I/O transfers

Instruction cycle state diagram


Instruction fetch
Operend fetch
Operend store
Multiple operends
Multiple results
Instruction address calculation
Instruction operation decoding
Instruction address calculation
Data Operation
Instruction address calculation
Instruction complete, fetch next instruction
Return for string or vector data

Instruction cycle state diagram

- The processor has a register named **program counter** which has the address of the instruction that has to be executed next. It gives the address of the instruction which needs to be fetched from the memory.
- If the instruction is fetched then, the instruction opcode is decoded. On decoding, the processor identifies the number of operands.
- If there is any operand to be fetched from the memory, then that operand address is calculated.

Instruction cycle state diagram

- Operands are fetched from the memory. If there is more than one operand, then the operand fetching process may be repeated (i.e. address calculation and fetching operands).
- After this, the data operation is performed on the operands, and a result is generated.
- If the result has to be stored in a register, the instructions end here.
- Side by side, the PC is incremented to calculate the address of the next instruction.
- Above instruction cycle is repeats for further instructions.

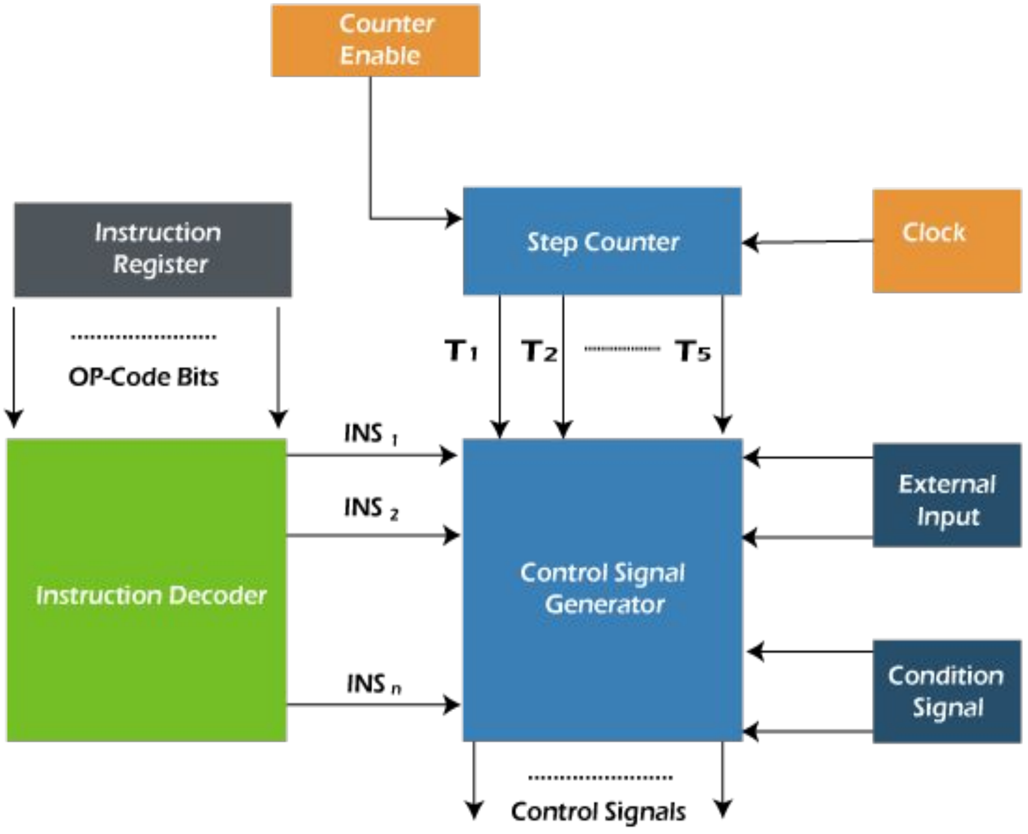
Design of control unit

The Control Unit is classified into two major categories:

1. Hardwired Control
2. Microprogrammed Control

1. Hardwired Control

A hardwired control is a method of generating control signals with the help of sequential logic circuit or Finite State Machines. It is designed with the help of gates, flip-flops, decoders, and other digital circuits.



1. Hardwired Control

- Instruction Register: The instruction fetched from the main memory is placed in the instruction register and the instruction remains there till its execution is completed.
- Instruction Decoder: The instruction decoder interprets the opcode and the addressing mode from the instruction register and determines what actions have to be taken.
- Step Counter: The step counter is used to track the progress in the execution of the instruction. The step counter indicates which step among the five i.e. instruction fetch, decode, operand fetch, execute, operand store steps is being carried out.

1. Hardwired Control

- Control Signal Generator: It is a combinational circuit that generates the control signals depending upon their input.
- Clock: The clock implement in the control circuitry is such that it completes one clock cycle for each step of instruction execution.
- External Inputs: The external input component acknowledges the control circuitry about the external signal such as interrupts.
- Conditional Signals: These components help the control unit in generating the control signals for branching instructions.

1. Hardwired Control

Initially, the instruction to be executed is fetched from the main memory and is a place in the instruction register which in turn generates the opcode which is interpreted by the instruction decoder. After interpreting the opcode bits the instruction decoder activates the corresponding INS*i* signal to the control circuitry.

With each clock cycle, one of the timing signals from T1 to T5 is activated indicating which step is from instruction fetch to operand store is being carried out. Based on the timing signals from the step counter and signals from the instruction decoder the control unit generates the control signals. The control signals are even influenced by the external signal and the conditional signals.

1. Hardwired Control

Factors Considered for the design of the hardwired control unit.

- Amount of hardware - Minimise the number of hardware used.
- Speed of operation - If a single IC can replace a group of IC, replace it. The amount of hardware and speed of operation are inversely proportional to each other.
- Cost

1. Hardwired Control

Advantages:

- Extremely fast
- Instruction set size is small as it relies on hardware more.
- The rapid mode of operation can be produced by optimising it.

Disadvantages:

- Modification becomes tougher as each time we have to play with complex circuits.
- Difficult to handle complex instructions.
- Design is complicated, and decoding is complex.

2. Microprogrammed Control

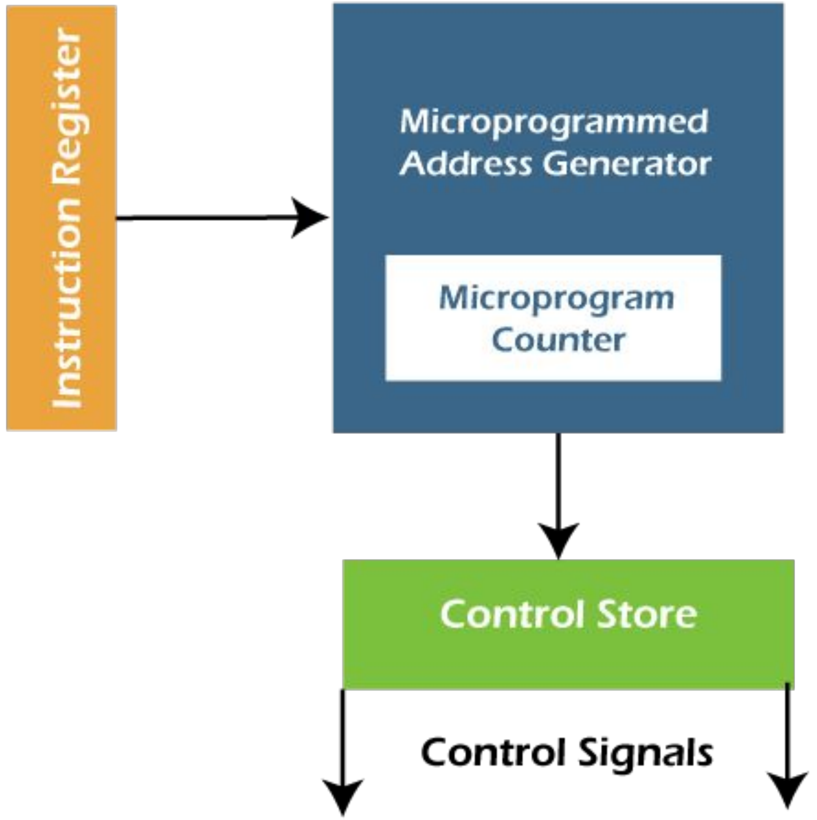
As the name suggests, these control units are designed with the help of a micro-program. This micro-program is a collections of micro-instructions stored in the control memory.

A micro-instruction consist of one or more micro-operations to be executed and the address of the next micro-instruction.

Example of micro-instruction:
$MAR \leftarrow R3$
In the above instruction, we are fetching the operand.
The control signal for the above example:
MAR_{i}□ , $R3_{ou}$□

2. Microprogrammed Control unit

- In this, the micro-operations are performed by executing a program consisting of micro-instructions.
- The implementation of this CU is very easy and flexible, but it is slower as compared to the Hardwired control unit.



2. Microprogrammed Control

1. **Instruction** fetch is the **first step**. In this step, the instruction is fetched from the IR with the help of a Microinstruction address register.
2. **Decode** is the **second step**. In this step, the instructions obtained from the instruction register will be decoded with the help of a microinstruction address generator.
3. **Increment** is the third step. In this step, the control word, which corresponds to the starting address of a micro-program, will be read. When the execution proceeds, the micro-program counter will be increased so that it can read the successive control words of a micro-routine.

2. Microprogrammed Control

4. **End bit** is the fourth step. In this step, the microinstruction of a micro-routine contains a bit, which is known as the end bit. The execution of the microinstruction will be successfully completed when the end bit is set to 1.

5. In last step, the micro-program address generator will again go back to **Step 1** so that we can fetch a new instruction, and this process or cycle goes on.

So in the micro-programmed control unit, the micro-programs are stored with the help of Control memory or Control store.

Concept of Nano programming

- In most microprogrammed processors, an instruction fetched from memory is interpreted by a micro program stored in a single control memory (CM).
- In some microprogrammed processors, the micro instructions are not directly used by the decoder to generate control signals. They use second control memory called a nano control memory (nCM).

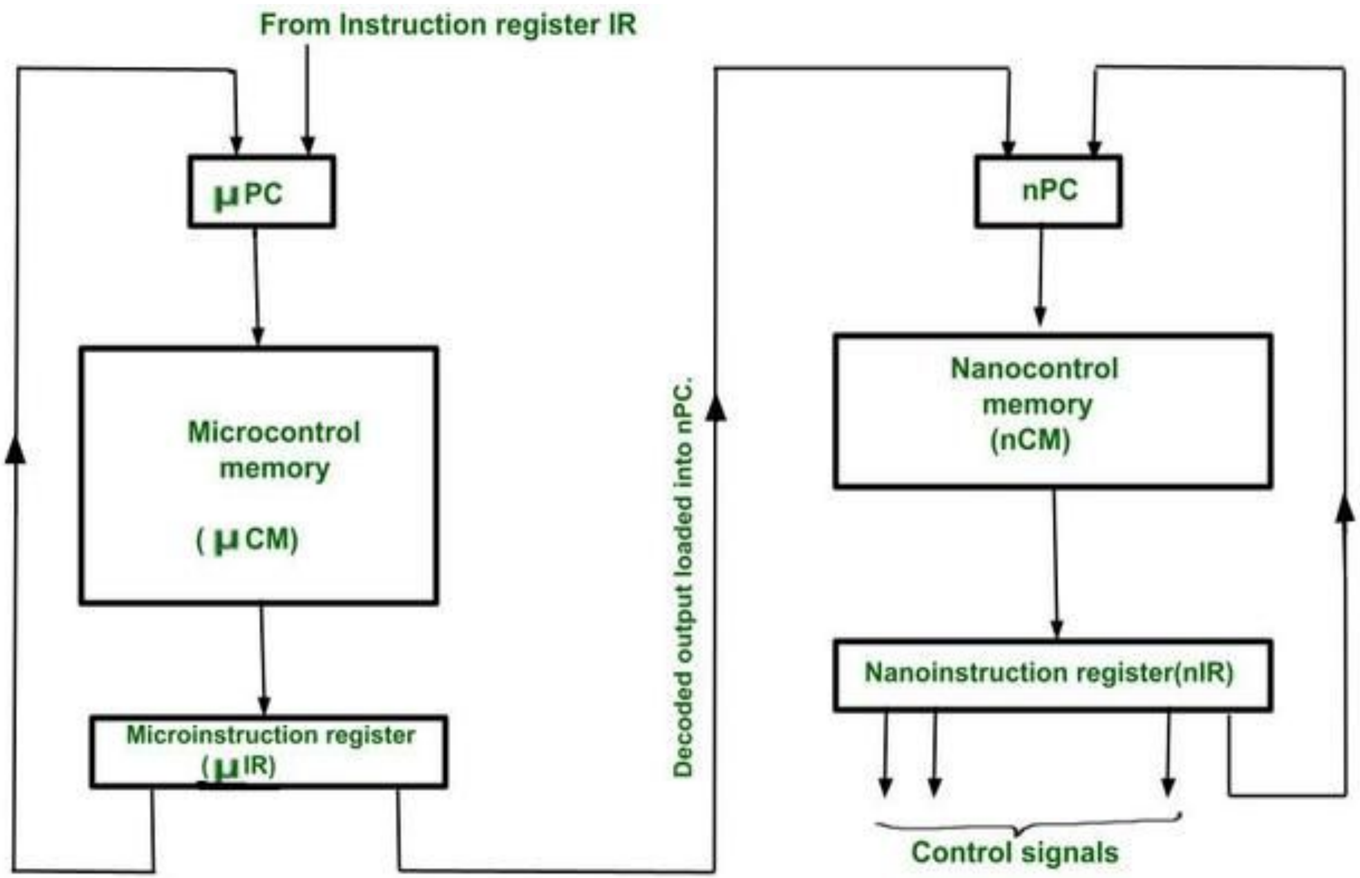
Concept of Nano programming

So they are two levels of control memories,

- A higher level control memories is known as micro control memory (µCM)
- A lower level control memories is known as nano control memory (nCM).

The µCM stores micro instructions whereas nCM stores nano instructions.

Concept of Nano programming


From Instruction register IR
μPC
Microcontrol memory
(μ CM)
Microinstruction register
(μ IR)
Decoded output loaded into nPC.
nPC
Nanocontrol memory
(nCM)
Nanoinstruction register(nIR)
Control signals

Concept of Nano programming

- The instruction is fetched from the main memory into instruction register IR.
- Using its opcode we load address of its first micro-instruction into μPC,
- Using this address we fetch the micro-instruction from micro control memory (μCM) into micro instruction register μIR.
- This is in vertical form and decoded by a decoder.
- The decoded output loads a new address in a nano program counter (nPC).

Concept of Nano programming

- By using this address, the nano-instruction is fetched from nano-control memory (nCM) into nano instruction register (nIR).
- This is in horizontal form and can directly generate control signals which can be multiple at a time.
- Such a combination gives advantage of both techniques.
- The size of the control Memory is small as micro-instructions are vertical.

Nano programming

Advantages:

Reduces total size of required control memory

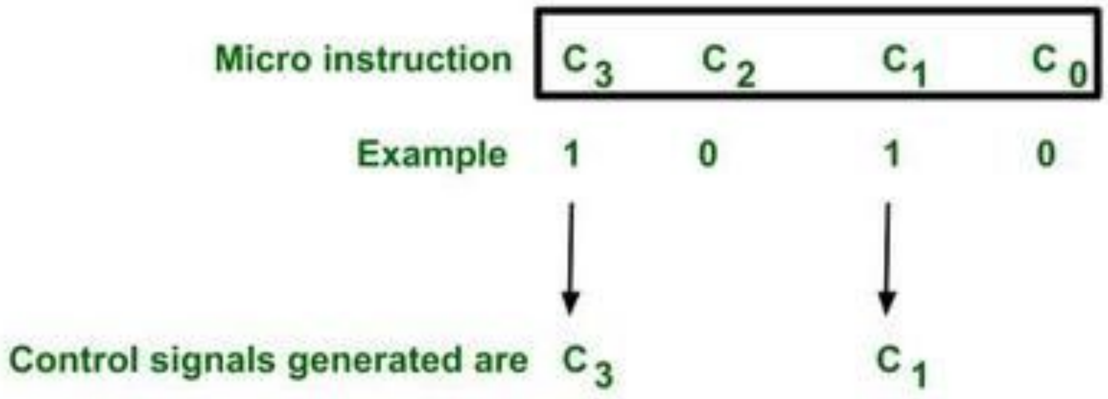
Disadvantage:

The main disadvantage of the two level memory approaches is the loss of speed due to the extra memory access required for nano control memory.

Micro instruction format

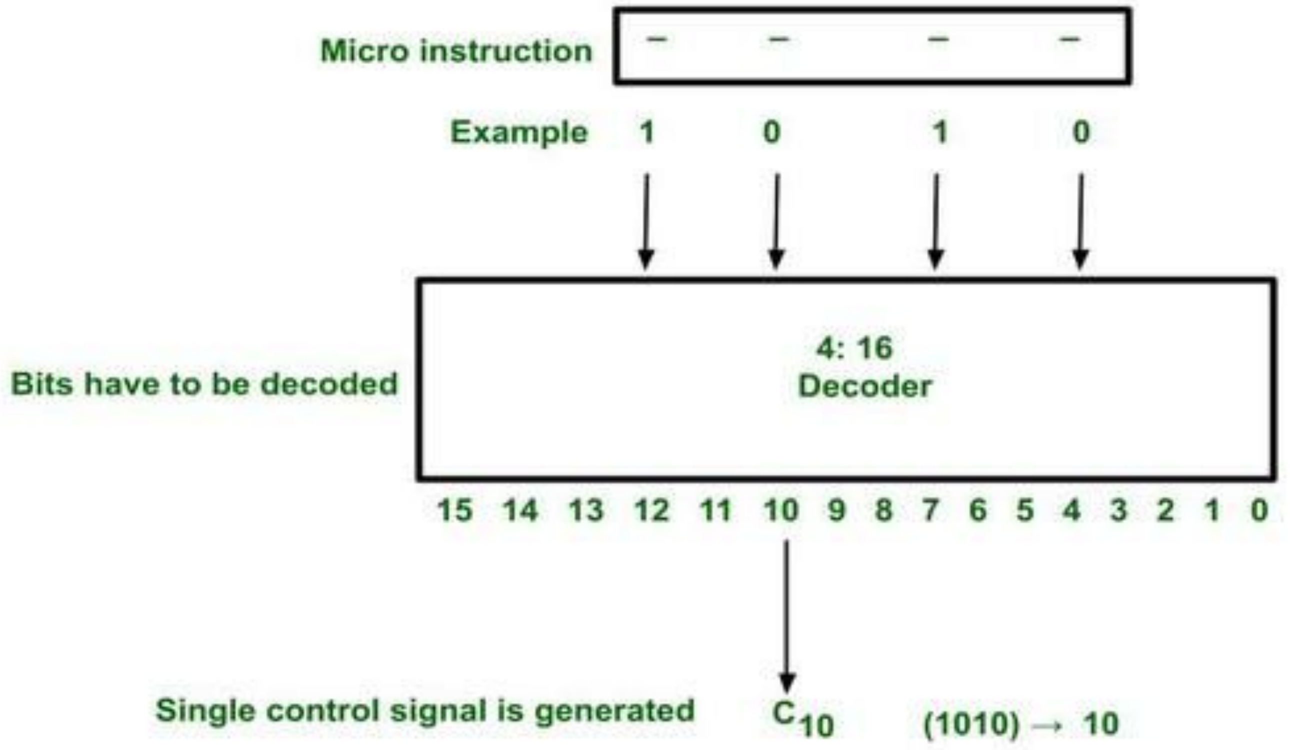
Type-I :

Horizontal micro-instruction :



Micro instruction format

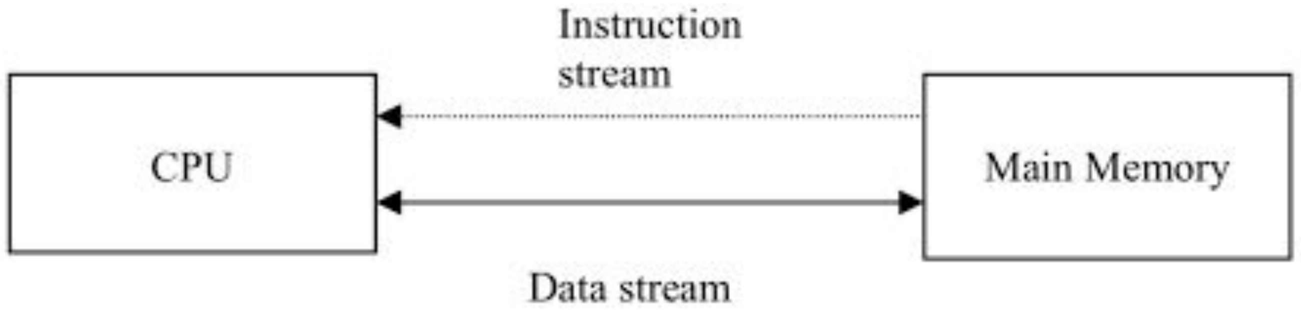
Type-2 : Vertical micro-instruction



Flynn's Classification

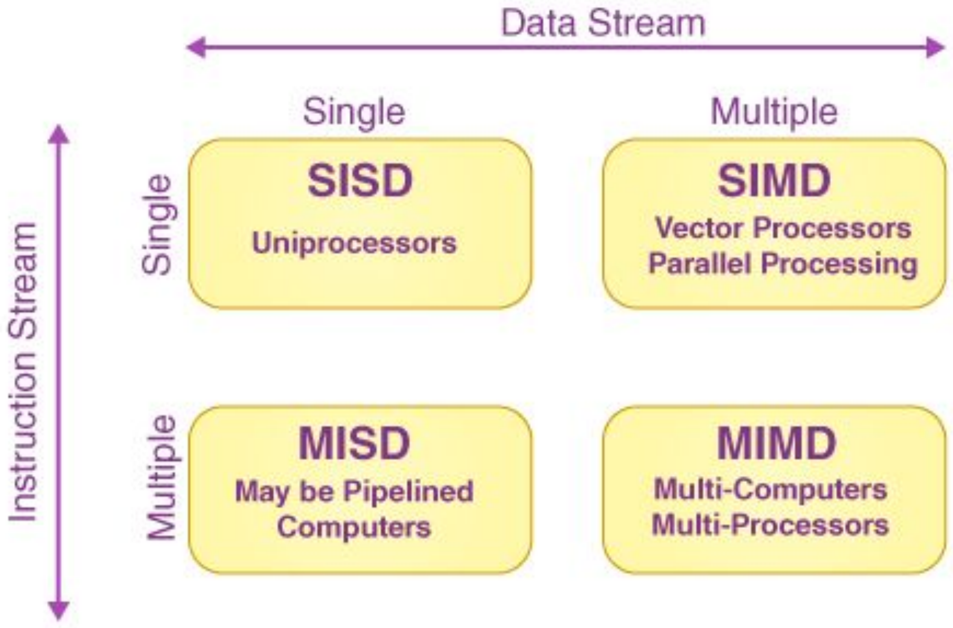
Flynn's Classification of Computer Architecture introduced by Michael Flynn in 1972. He introduced the concept of instruction and data stream for categorizing of computers.

- **Instruction stream** means flow of information from main memory to CPU.
- **Data stream** means flow of operands between main memory and CPU bi-directionally.



Flynn's Classification

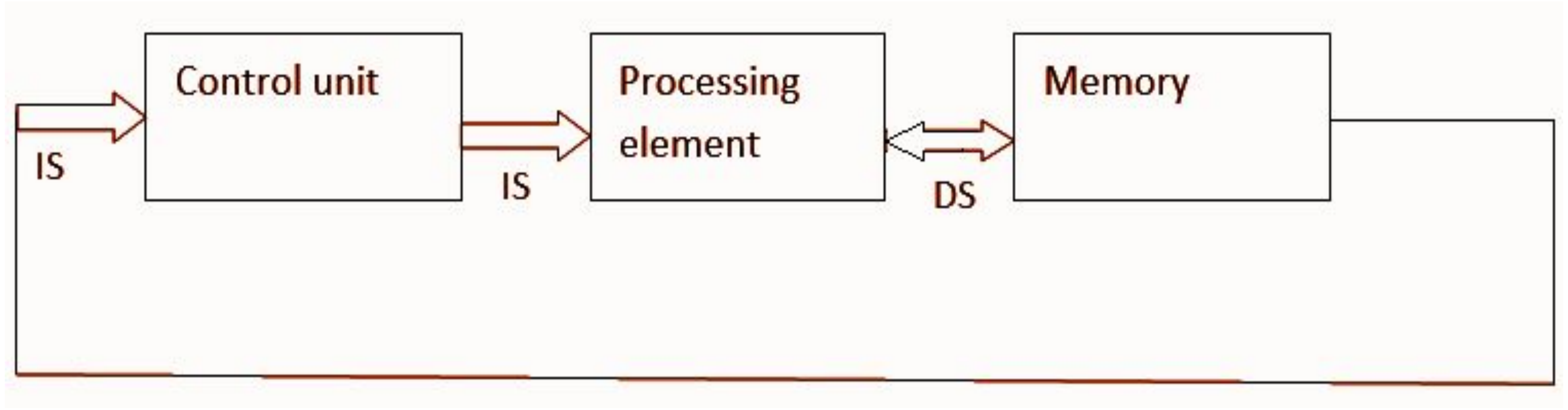
In Flynn's classification, either of the instruction or data streams can be single or multiple. Thus computer architecture can be classified into the following four distinct categories:



SISD (Single Instruction Single Data Stream)

Single instruction: Only one instruction stream is being acted or executed by CPU during one clock cycle.

Single data stream: Only one data stream is used as input during one clock cycle.



SISD (Single Instruction Single Data Stream)

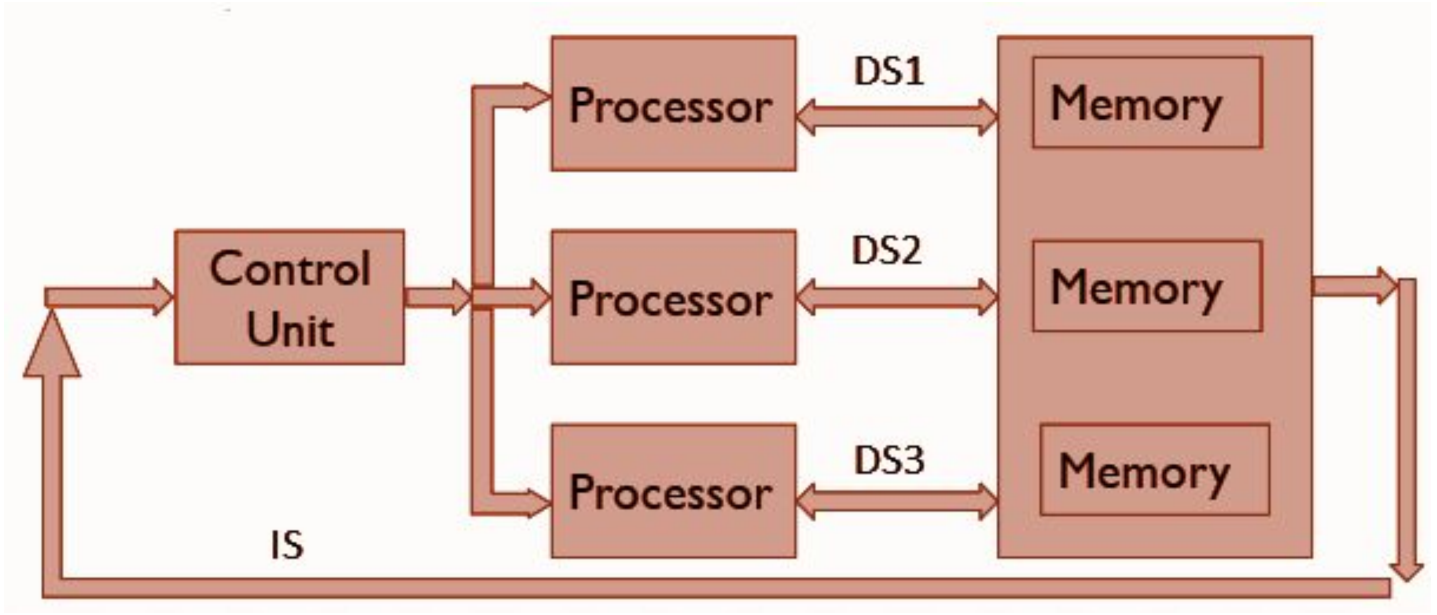
A SISD computing system is a uniprocessor machine that is capable of executing a single instruction operating on a single data stream.

Most conventional computers have SISD architecture where all the instruction and data to be processed have to be stored in primary memory.

Examples: Minicomputers, workstations, and computers from previous generations

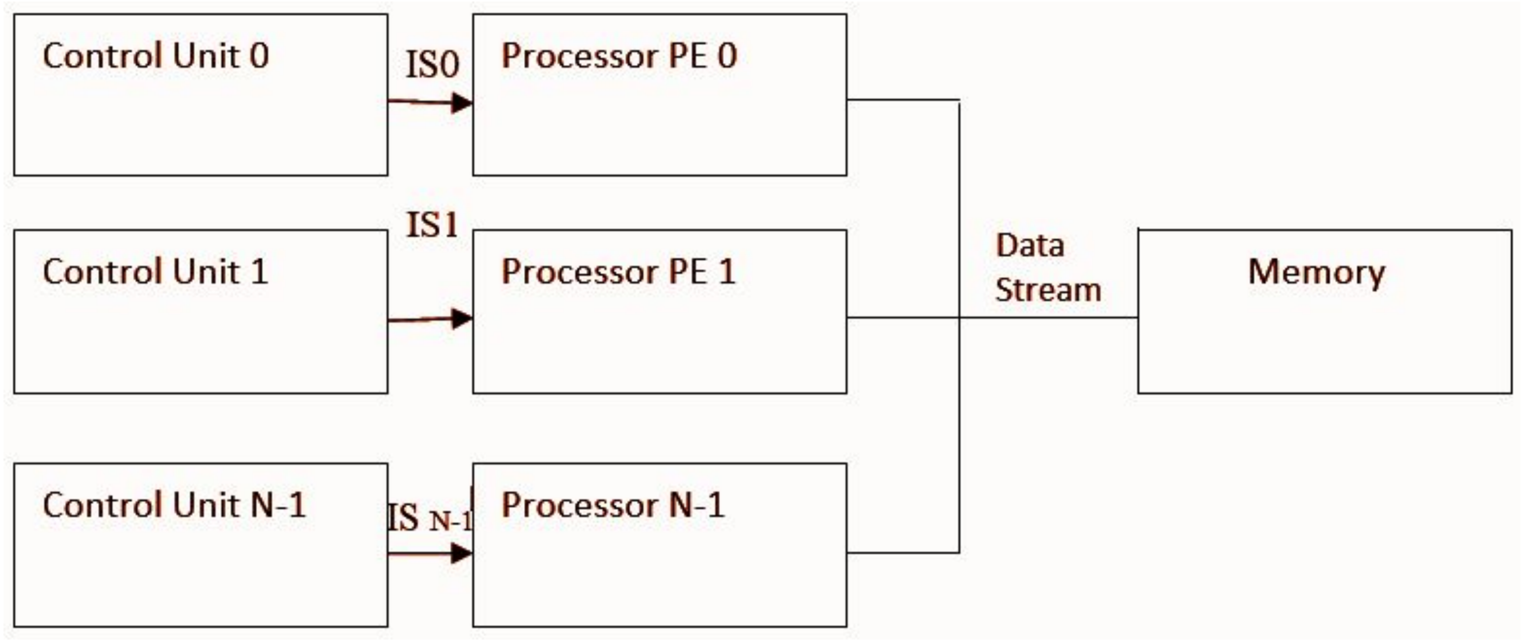
SIMD (Single Instruction Multiple Data Stream)

A SIMD system is a multiprocessor machine, capable of executing the same instruction on all the CPUs but operating on the different data stream.



MISD (Multiple Instruction Single Data Stream)

An MISD computing is a multiprocessor machine capable of executing different instructions on processing elements but all of them operating on the same data set.



MISD (Multiple Instruction Single Data Stream)

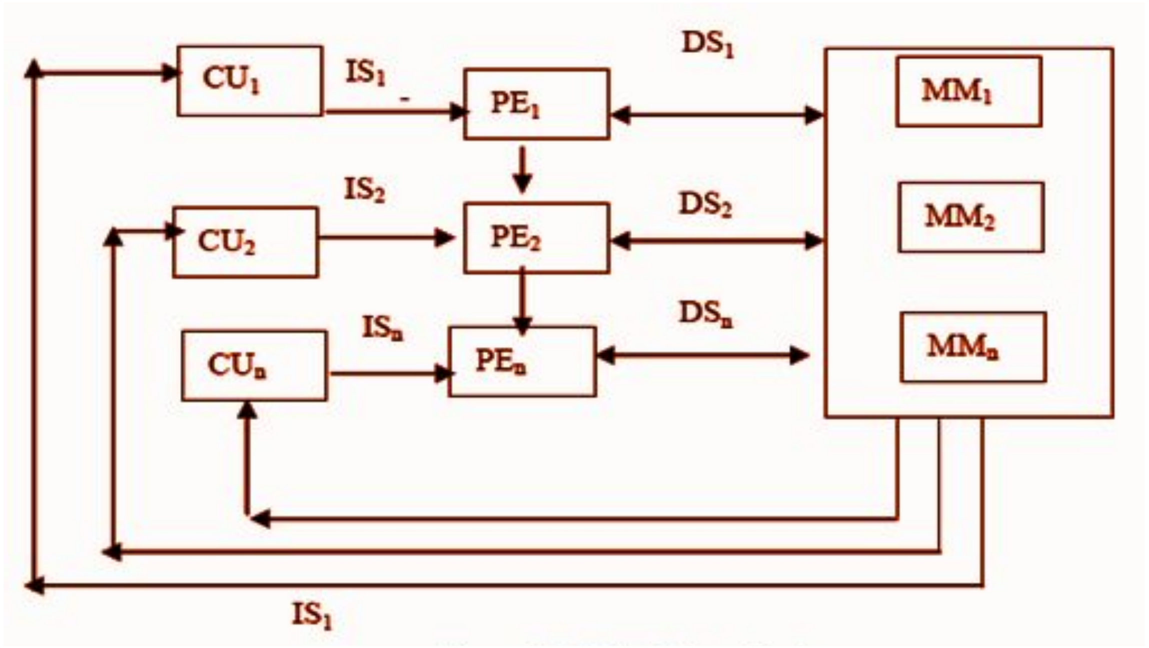
Because no real system has been built using the MISD structure, it is primarily of theoretical importance.

Multiple processing units work on a single data stream in MISD. Each processing unit works on the data in its own way, using its own instruction stream.

Example: The experimental Carnegie-Mellon computer C.mmp (in 1971)

MIMD (Multiple Instruction Multiple Data Stream)

A MIMD system is a multiprocessor machine that is capable of executing multiple instructions over multiple data streams. Each processing element has a separate instruction stream and data stream.



MIMD (Multiple Instruction Multiple Data Stream)

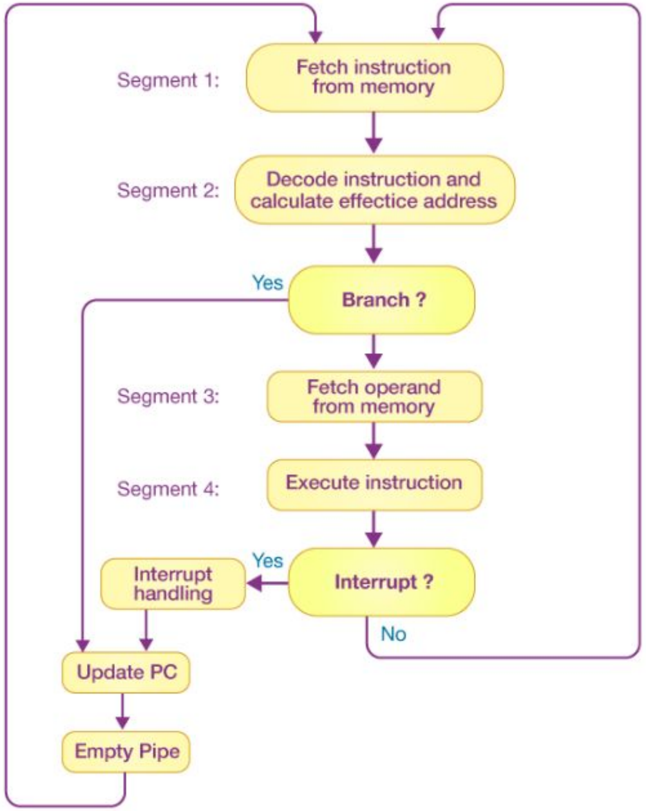
All multiprocessor systems drop under this classification.

Examples:

Burroughs D825, Cray X-MP, C.mmp Cray-2, S1, HEP etc.

6 stage instruction Pipeline

Pipeline processing can happen not only in the data stream but also in the instruction stream. To perform tasks such as fetching, decoding and execution of instructions, most digital computers with complicated instructions would require an instruction pipeline.



6 stage instruction Pipeline

In general, each and every instruction must be processed by the computer in the following order:

1. Fetching the instruction from memory
2. Decoding the obtained instruction
3. Calculating the effective address
4. Fetching the operands from the given memory
5. Execution of the instruction
6. Storing the result in a proper place

6 stage instruction Pipeline

A four-segment instruction pipeline is illustrated in the block diagram given above. The instructional cycle is divided into four parts:

Segment 1: The implementation of the instruction fetch segment can be done using the FIFO or first-in, first-out buffer.

Segment 2: In the second segment, the memory instruction is decoded, and the effective address is then determined in a separate arithmetic circuit.

6 stage instruction Pipeline

Segment 3

In the third segment, some operands would be fetched from memory.

Segment 4

The instructions would finally be executed in the very last segment of a pipeline organisation.

Pipeline Hazards

1. Pipeline hazards are situations that prevent the next instruction in the instruction stream from executing during its designated clock cycles.
2. Any condition that causes a stall in the pipeline operations can be called a hazard.
3. There are primarily three types of hazards:

   i. Data Hazards

   ii. Control Hazards or instruction Hazards

   iii. Structural Hazards.

1. Data Hazards

A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline.

As a result of which some operation has to be delayed and the pipeline stalls. Whenever there are two instructions one of which depends on the data obtained from the other.

A=3+A

B=A*4

Data Hazards

For the above sequence, the second instruction needs the value of ‘A’ computed in the first instruction.

Thus the second instruction is said to depend on the first.

If the execution is done in a pipelined processor, it is highly likely that the interleaving of these two instructions can lead to incorrect results due to data dependency between the instructions. Thus the pipeline needs to be stalled as and when necessary to avoid errors.

2. Structural Hazards

This situation arises mainly when two instructions require a given hardware resource at the same time and hence for one of the instructions the pipeline needs to be stalled.

The most common case is when memory is accessed at the same time by two instructions. One instruction may need to access the memory as part of the Execute or Write back phase while other instruction is being fetched.

Structural Hazards

In this case if both the instructions and data reside in the same memory. Both the instructions can’t proceed together and one of them needs to be stalled till the other is done with the memory access part.

Thus in general sufficient hardware resources are needed for avoiding structural hazards.

3. Branch Hazards

The instruction fetch unit of the CPU is responsible for providing a stream of instructions to the execution unit. The instructions fetched by the fetch unit are in consecutive memory locations and they are executed.

However the problem arises when one of the instructions is a branching instruction to some other memory location.

Branch Hazards

Thus all the instruction fetched in the pipeline from consecutive memory locations are invalid now and need to removed(also called flushing of the pipeline).This induces a stall till new instructions are again fetched from the memory address specified in the branch instruction.

Thus the time lost as a result of this is called a branch penalty. Often dedicated hardware is incorporated in the fetch unit to identify branch instructions and compute branch addresses as soon as possible and reducing the resulting delay as a result.