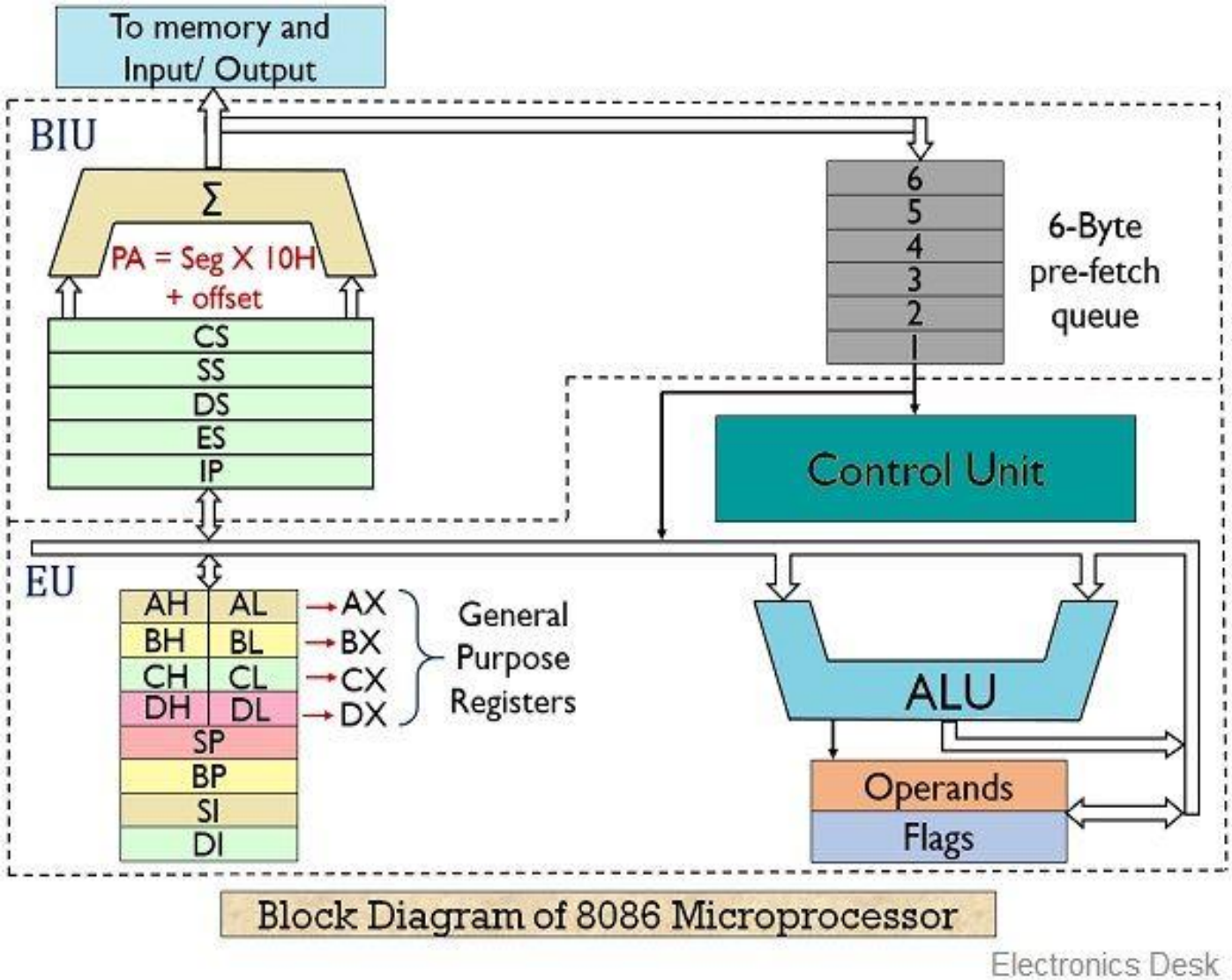
Block diagram of 8086:-

8086 is a 16-bit microprocessor and was designed in 1978 by Intel. Unlike, 8085, an 8086 microprocessor has 20-bit address bus. Thus, is able to access 220 i.e., 1 MB address in the memory. As we know that a microprocessor performs arithmetic and logic operations. And an 8086 microprocessor is able to perform these operations with 16-bit data in one cycle. Hence is a 16-bit microprocessor. Thus the size of the data bus is 16-bit as it can carry 16-bit data at a time.

Block Diagram of 8086 Microprocessor

The architecture of 8086 microprocessor is composed of 2 major units, the BIU i.e., Bus Interface Unit and EU i.e., Execution Unit. The figure below shows the block diagram of the architectural representation of the 8086 microprocessor:



Block Diagram of 8086 Microprocessor

Electronics Desk

Bus Interface Unit (BIU)

The Bus Interface Unit (BIU) manages the data, address and control buses.

The BIU functions in such a way that it:

- Fetches the sequenced instruction from the memory,
- Finds the physical address of that location in the memory where the instruction is stored and
- Manages the 6-byte pre-fetch queue where the pipelined instructions are stored.

An 8086 microprocessor exhibits the property of pipelining the instructions in a queue while performing decoding and execution of the previous instruction. This saves the processor time of operation by a large amount. This pipelining is done in a 6-byte queue. Also, the BIU contains 4 segment registers. Each segment register is 16-bit. The segments are present in the memory and these registers hold the address of all the segments. These registers are as follows:

1**. Code segment register:** It is a 16-bit register and holds the address of the instruction or program stored in the code segment of the memory.

Also, the IP in the block diagram is the instruction pointer which is a default register that is used by the processor in order to get the desired instruction. The IP contains the offset address of the next byte that is to be taken from the code segment.

2. **Stack segment register:** The stack segment register provides the starting address of the stack segment in the memory. Like in stack pointer, PUSH and POP operations are used in this segment to give and take the data to/from it.

3. **Data segment register**: It holds the address of the data segment. The data segment stores the data in the memory whose address is present in this 16-bit register.

4. **Extra segment register**: Here the starting address of the extra segment is present. This register basically contains the address of the string data. It is to be noteworthy that the physical address of the instruction is achieved by combining the segment address with that of the offset address.

6-byte pre-fetch queue: This queue is used in 8086 in order to perform pipelining. As at the time of decoding and execution of the instruction in EU, the BIU fetches the sequential upcoming instructions and stores it in this queue.

The size of this queue is 6-byte. This means at maximum a 6-byte instruction can be stored in this queue. The queue exhibits FIFO behavior, first in first out.

Execution Unit (EU)

The Execution Unit (EU) performs the decoding and execution of the instructions that are being fetched from the desired memory location.

Control Unit: Like the timing and control unit in 8085 microprocessor, the control unit in 8086 microprocessor produces control signal after decoding the opcode to inform the general purpose register to release the value stored in it. And it also signals the ALU to perform the desired operation.

ALU: The arithmetic and logic unit carries out the logical tasks according to the signal generated by the CU. The result of the operation is stored in the desired register.

Flag: Like in 8085, here also the flag register holds the status of the result generated by the ALU. It has several flags that show the different conditions of the result.

Operand: It is a temporary register and is used by the processor to hold the temporary values at the time of operation.

The reason behind two separate sections for BIU and EU in the architecture of 8086 is to perform fetching and decoding-executing simultaneously.

Registers and applications of Microprocessor:-

In the 8086 Microprocessor, the registers are categorized into mainly four types:

- General Purpose Registers
- Segment Registers
- Pointers and Index Registers
- Flag or Status Register

1) General Purpose Registers

The use of general-purpose registers is to store temporary data. While the instructions are executed in the control unit, they may work on some numeric value or some operands. These need to be stored somewhere so that the processor can operate on them easily. So, these registers are used in these cases. There are 4 general-purpose registers of 16-bit length each. Each of them is further divided into two subparts of 8-bit length each: one high, which stores the higher-order bits and another low which stores the lower order bits.

I. AX = [AH:AL]
II. BX = [BH:BL]
III. CX = [CH:CL]
IV. DX = [DH:DL]

2) Segment Registers

There are 4 segment registers in 8086 Microprocessor and each of them is of 16 bit. The code and instructions are stored inside these different segments.

- **Code Segment (CS) Register:** The user cannot modify the content of these registers. Only the microprocessor's compiler can do this.
- **Data Segment (DS) Register:** The user can modify the content of the data segment.
- **Stack Segment (SS) Registers:** The SS is used to store the information about the memory segment. The operations of the SS are mainly Push and Pop/pull.
- **Extra Segment (ES) Register:** By default, the control of the compiler remains in the DS where the user can add and modify the instructions. If there is less space in that segment, then ES is used. ES is also used for copying purpose.

3) Pointers and Index Registers

The pointers will always store some address or memory location. In 8086 Microprocessor, they usually store the offset through which the actual address is calculated.

- **Instruction Pointer (IP):** The instruction pointer usually stores the address of the next instruction that is to be executed. Apart from this, it also acts as an offset for CS register.
- **Base Pointer (BP):** The Base pointer stores the base address of the memory. Also, it acts as an offset for Stack Segment (SS).
- **Stack Pointer (SP):** The Stack Pointer Points at the current top value of the Stack. Like the BP, it also acts as an offset to the Stack Segment (SS). The indexes are used with the extra segment and they usually are used for copying the contents of a particular block of memory to a new location.
- **Source Index (SI):** It stores the offset address of the source.
- **Destination Index (DI):** It stores the offset address of the Destination.

4) **Flag or Status Register**

The Flag or Status register is a 16-bit register which contains 9 flags, and the remaining 7 bits are idle in this register. These flags tell about the status of the processor after any arithmetic or logical operation. IF the flag value is 1, the flag is set, and if it is 0, it is said to be reset.

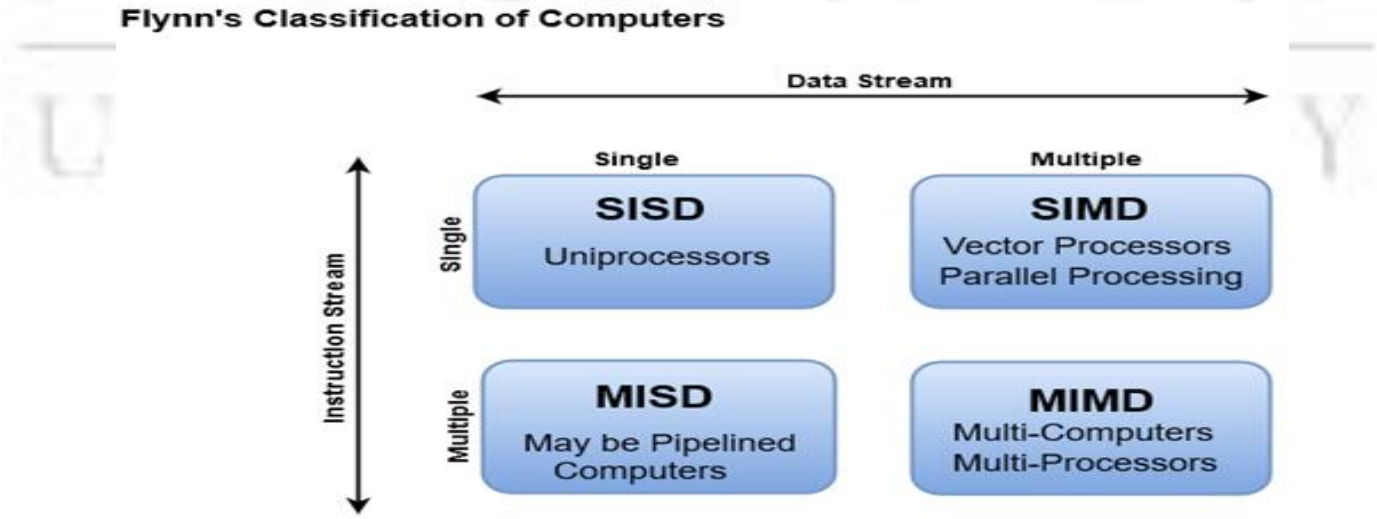
➢ Applications of the Microprocessors

- It is present in single-board microcomputers as they use low configuration with software and hardware.
- It is embedded in the PC making it suitable to access and use applications.
- It's present in superminis and CAD enhancing their performance.
- It acts as an instrument because of its accepting programmability.

- It acts as a controller in many home appliances like toasters, televisions, stereo systems, etc. In the science industry, it is useful for measuring speed, temperature, moisture, etc.
- The telecom sector uses it for a digital telephone system, telephone exchange, and modem while the hospitality sector uses it for railway and airline reservation systems.
- Office automation uses it for word processing, spreadsheet operations, storage, etc.
- The publication uses it for automatic photocopies, high-quality printing, and good speed.
- Consumers are using it for toys, amusement devices, and house held devices frequently nowadays.
- It is also present in wireless communication equipment allowing them to interact and connect with devices.

Parallel Processing – Flynn's classification:-

Flynn's classic taxonomy (Flynn, 1966) depends on the number of control units and the multiple processors available in a computer. Flynn's introduced the following notions –

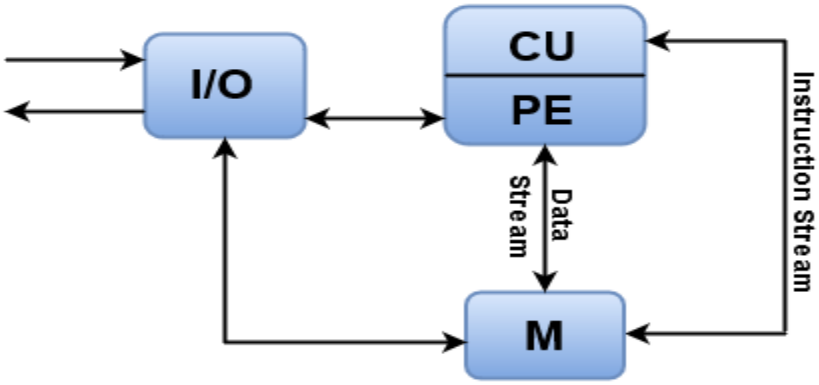


SISD:-

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.

SISD:



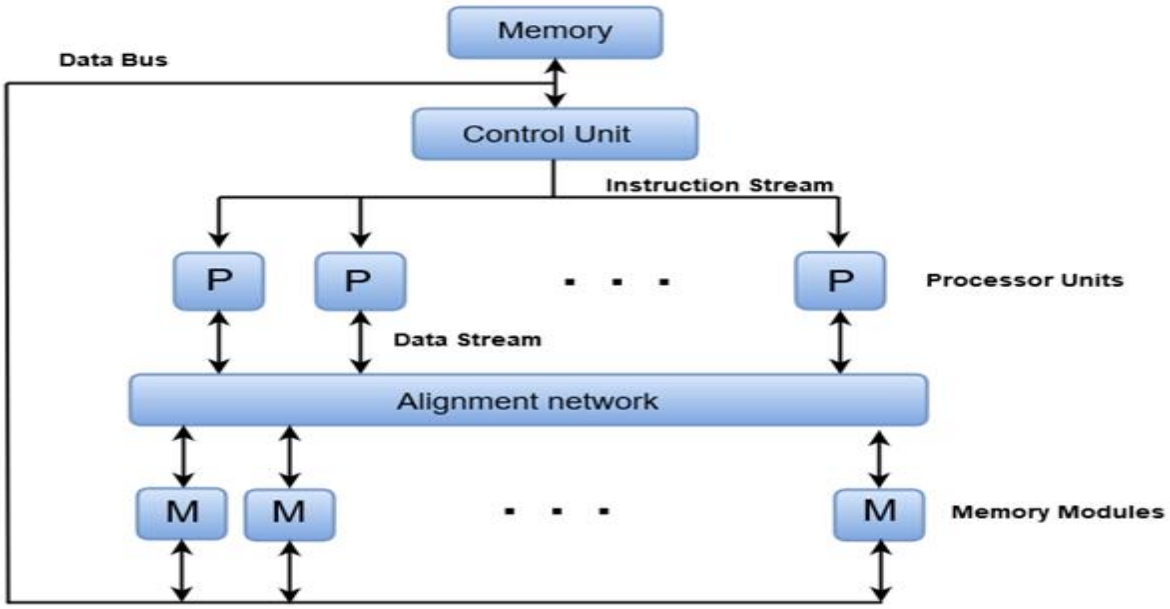
Where, CU = Control Unit, PE = Processing Element, M = Memory.
Instructions are decoded by the Control Unit and then the Control Unit sends the instructions to the processing units for execution. Data Stream flows between the processors and memory bi-directionally.

Examples: Older generation computers, minicomputers, and workstations

SIMD:-

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:

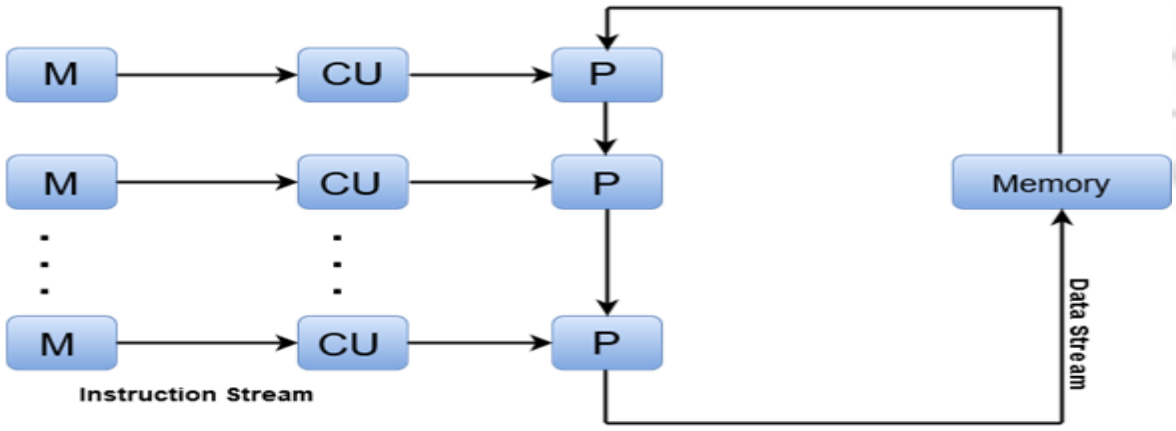


SIMD is mainly dedicated to array processing machines. However, vector processors can also be seen as a part of this group.

MISD:-

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.

MISD:



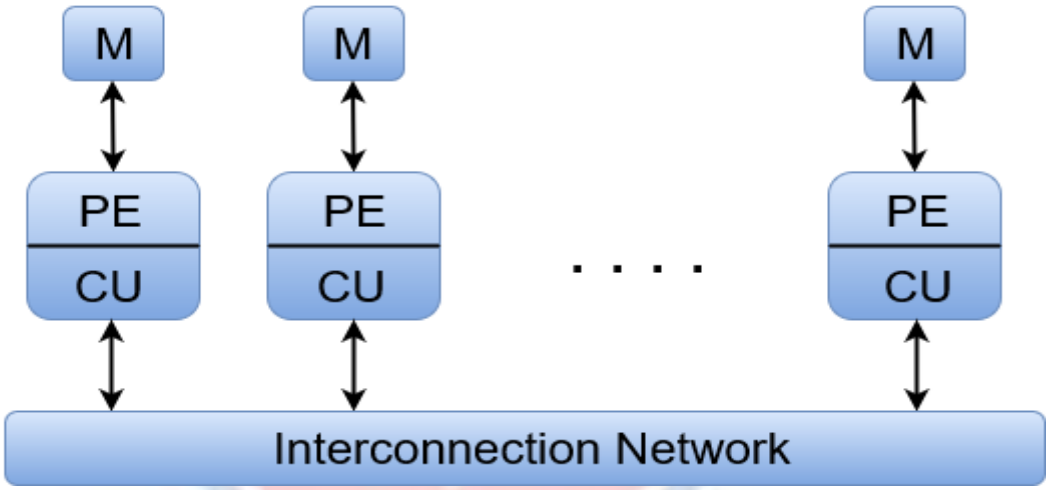
Where, M = Memory Modules, CU = Control Unit, P = Processor Units.

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

MIMD:-

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.

MIMD:



Where, M = Memory Module, PE = Processing Element, and CU = Control Unit.

Examples: Cray T90, Cray T3E, IBM-SP2.

Pipelining:-

The term Pipelining refers to a technique of decomposing a sequential process into sub-operations, with each sub-operation being executed in a dedicated segment that operates concurrently with all other segments.

The most important characteristic of a pipeline technique is that several computations can be in progress in distinct segments at the same time. The overlapping of computation is made possible by associating a register with each segment in the pipeline. The registers provide isolation between each segment so that each can operate on distinct data simultaneously.

The structure of a pipeline organization can be represented simply by including an input register for each segment followed by a combinational circuit.

Let us consider an example of combined multiplication and addition operation to get a better understanding of the pipeline organization.

The combined multiplication and addition operation is done with a stream of numbers such as:

Ai* Bi + Ci for i = 1, 2, 3, ..., 7

The operation to be performed on the numbers is decomposed into sub-operations with each sub-operation to be implemented in a segment within a pipeline. The sub-operations performed in each segment of the pipeline are defined as:

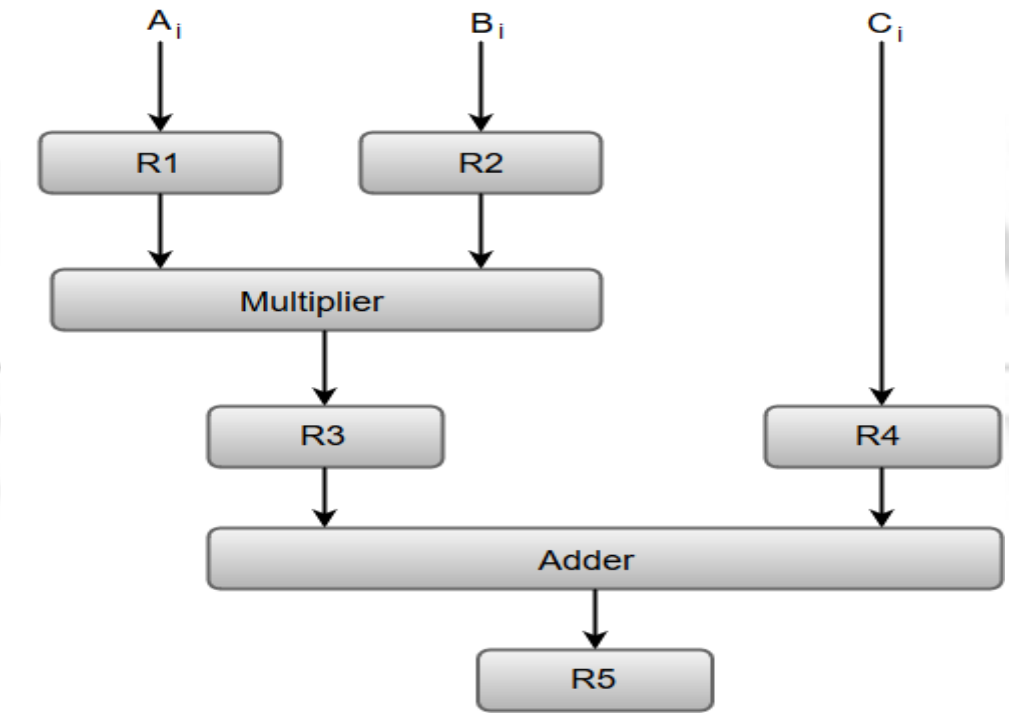
R1 ← Ai, R2 ← Bi Input Ai, and Bi

R3 ← R1 * R2, R4 ← Ci multiply, and input Ci

R5 ← R3 + R4 Add Ci to product

The following block diagram represents the combined as well as the sub-operations performed in each segment of the pipeline.

Pipeline Processing:



Registers R1, R2, R3, and R4 hold the data and the combinational circuits operate in a particular segment.

The output generated by the combinational circuit in a given segment is applied as an input register of the next segment. For instance, from the block diagram, we can see that the register R3 is used as one of the input registers for the combinational adder circuit.

In general, the pipeline organization is applicable for two areas of computer design which includes:

- **Arithmetic Pipeline**:-

Arithmetic Pipelines are mostly used in high-speed computers. They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in scientific problems.

- **Instruction Pipeline:-**

Pipeline processing can occur not only in the data stream but in the instruction stream as well. Most of the digital computers with complex instructions require instruction pipeline to carry out operations like fetch, decode and execute instructions. In general, the computer needs to process each instruction with the following sequence of steps.

- Fetch instruction from memory.
- Decode the instruction.
- Calculate the effective address.
- Fetch the operands from memory.
- Execute the instruction.
- Store the result in the proper place.

Each step is executed in a particular segment, and there are times when different segments may take different times to operate on the incoming information. Moreover, there are times when two or more segments may require memory access at the same time, causing one segment to wait until another is finished with the memory.

Thank You