CHAPTER-9

Advance Architecture

9.1. RISC and CISC fundamentals

RISC:

Reduced Instruction set Architecture (RISC) computer is a type of computer that emphasizes simplicity and efficiently in instruction execution.

↳ The fundamental idea behind RISC architecture is to streamline the instruction set by using a reduced number of instructions with fixed formats, making them easier to decode and execute.

↳ there are some key characteristics of RISC architecture:

1. **Simplicity**: RISC architectures have a simplified instruction set with a small number of instructions.

2. **Load-store Architecture**: Registers are used to load and memory address are for store instructions. This simplifies the instruction set and are reduces memory access latency.

3. **fixed Length Instructions**:
RISC architectures often use fixed-length instructions, which simplifies instructions decoding and allows for more regular instruction fetching. This improves pipelining and instruction fetching efficiency.

4. **Register usage**
RISC architectures heavily rely on registers for storing and manipulating data. Registers are fast storage locations built into the processor itself.

CISC:

↳ CISC (complex Instruction set computer) is a type of computer architecture that supports a large number of complex and specialized instructions.

↳ In contrast to RISC architecture, CISC processors aim to provide instructions that can perform more complex operations in a single instruction.

RISC	CISC
1. Instructions Takes one or two cycles.	1. Instruction takes multiple cycles
2. Only load/store instructions are used to access memory.	2. In addition to load and store, memory can be accessed using other instructions too.
3. Instructions executed by hardware.	3. Instructions executed by software or microprogram.
4. fixed format instructions.	4. variable format instructions
5. few instruction sets.	5. complex instruction sets.
6. Most of the instructions are multiple register based.	6. single register Based.
7. Highly pipelined.	7. Less pipelined.
8. Complexity in compiler.	8. Complexity in microprogram
9. fast	9. slow.

8.2. Pipelining:

Pipelining is a technique of decomposing a sequential process into sub-operation with each sub process begin executed in a special dedicated segment that operates concurrently with all other segments.

Each segment of a pipeline has a capability of performing a special task. The register provide isolation between each segment, so that each segment can operate on distinct data simultaneously. Hence pipeline structure can be viewed as segments with separate input register followed by combinational circuits. All the segments are synchronized by clock.

The working of pipeline can be demonstrated by following example.

$A_i * B_i + C_i$ for $i = 1, 2, 3, 7$

Here above operation can be decomposed into following sub operations:

i) $R_1 \leftarrow A_i$ ii) $R_2 \leftarrow B_i$; input $A_i + B_i$

iii) $R_3 \leftarrow A_i * B_i$ iv) $R_4 \leftarrow C_i$; multiply $A_i * B_i$, input C_i

Add C_i to the product.

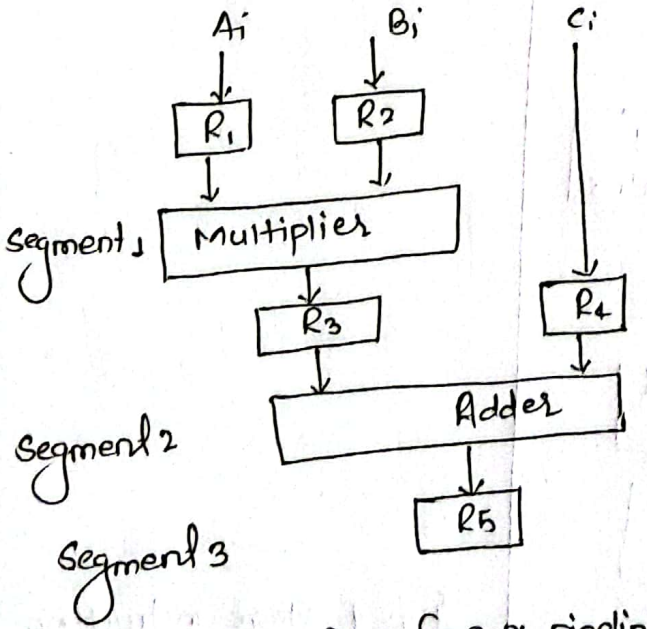

fig: flow of pipeline process.

Instruction Pipeline:

⇒ Pipeline processing can occur not only in data stream but in instruction as well. An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments. Instruction fetch and execut- phases are overlapped and perform simultaneous operations.

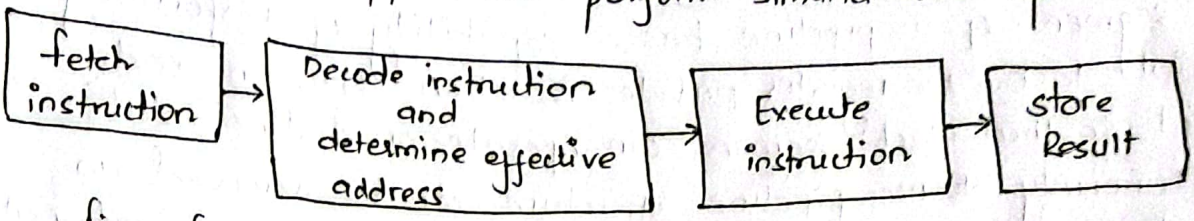

fig: four stage instruction pipeline.

Generally instructions are processed in following sequence:

1. fetch the instruction from memory.
2. Decode the instruction
3. Calculate the effective address
4. fetch the operands from memory
5. Execute the instruction
6. store the result in proper place.

The working of 4 stage instruction pipeline is explained by following flowchart.

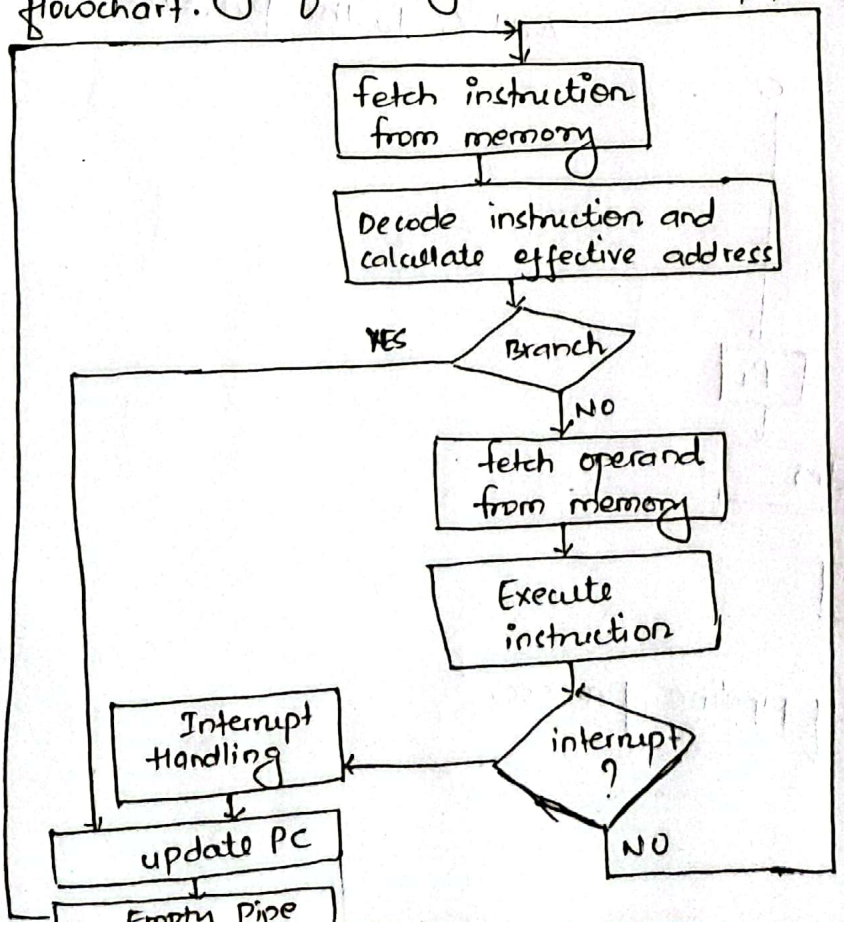

fig: 4 stage instruction pipeline.

	1	2	3	4	5	6	7	8	9
1	FI	DA	Fo	Ex					
2	-	FI	DA	Fo	Ex				
3	-	-	FI	DA	Fo	Ex			
4	-	-	-	FI	DA	fo	Ex		
5	-	-	-	-	FI	DA	fo	Ex	
6	-	-	-	-	-	FI	DA	fo	Ex

where, FI = fetch Instruction
DA = Decode instruction and calculate effective address
Fo = fetch operand
EX = Execute instruction.

Register window

In computer architecture, a register window is a technique used in some processor designs to optimize function calls and improve performance. It is commonly found in architectures with a large number of registers.

The idea behind the register window is to create a small set of registers (called a window) that can be accessed quickly by current function. This window typically contains both general-purpose registers and special-purpose registers used for function calls, return addresses and saved registers.

When function is called, a new window is created by pushing the previous window onto a stack. This effectively saves the previous state of the registers. The new window provides a fresh set of registers for the called function to use. As functions call other functions, additional windows are created and stacked on top of each other.

When a function returns, the current window is discarded, and the previous window is restored from the stack. This process allows for fast and efficient function calls without the need to save and restore registers to memory.

By using register windows, the processor can reduce the number of memory accesses required for saving and restoring registers during function calls, resulting in improved performance. Primarily, register window are used in RISC architecture such as SPARC (Scalable Processor Architecture).

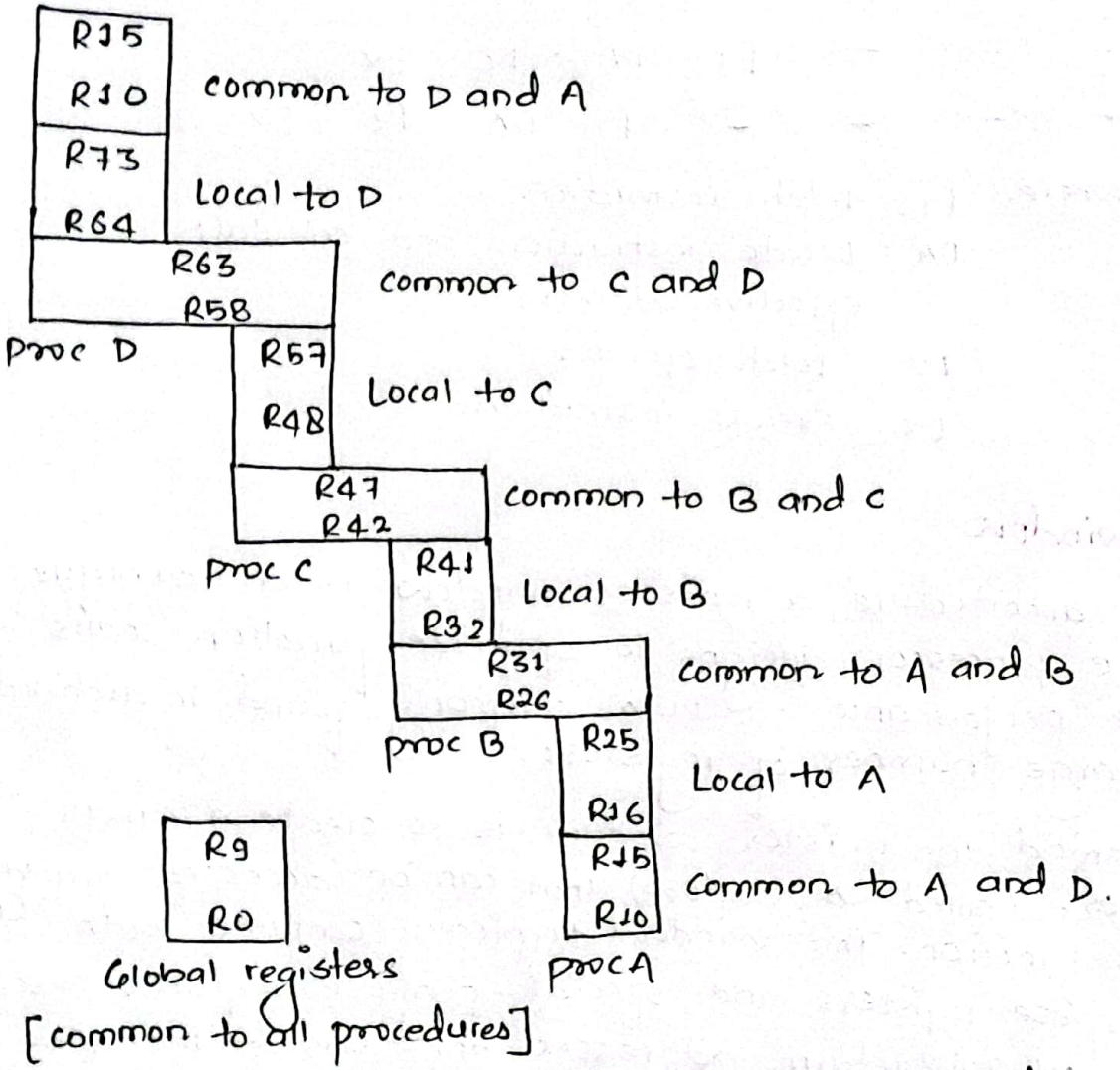

fig: Register window in RISC Architecture.

8.3. Flynn's Taxonomy

Flynn's Taxonomy is a specific classification of parallel computer architectures that are based upon numbers of concurrent instructions and data stores available in the architecture. It was proposed by M.J Flynn in 1966. According to Flynn's Taxonomy, computers are classified into following types:

i) Single Instruction streams, Single Data stream (SISD)
ii) Single Instruction streams, Multiple Data stream (SIMD)
iii) Multiple Instruction stream, Single Data stream (MISD)
iv) Multiple Instruction stream, Multiple Data stream (MIMD)

i) SISD:

- Represents organization of a single computer containing a control unit, a processor unit and a memory unit.
- Instructions are executed sequentially and the systems may or maynot have internal parallel processing capabilities
- Parallel processing may be achieved by use of multiple functional units or pipelining processing.

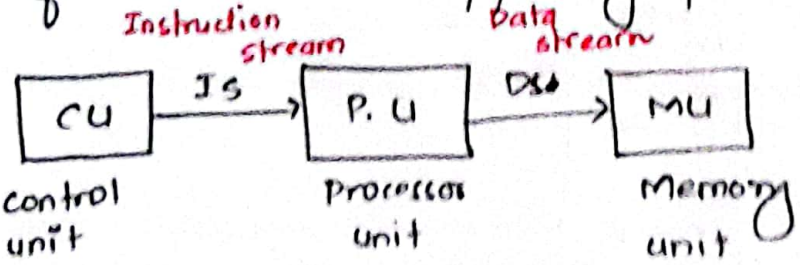

fig: SISD computer.

ii) SIMD

- Represents organizations of a system containing multiple processing unit under control of a single control unit.
- All processors receive same instructions from control unit but operate on different data.
- Shared memory unit must have multiple modules to communicate with all processors simultaneously.

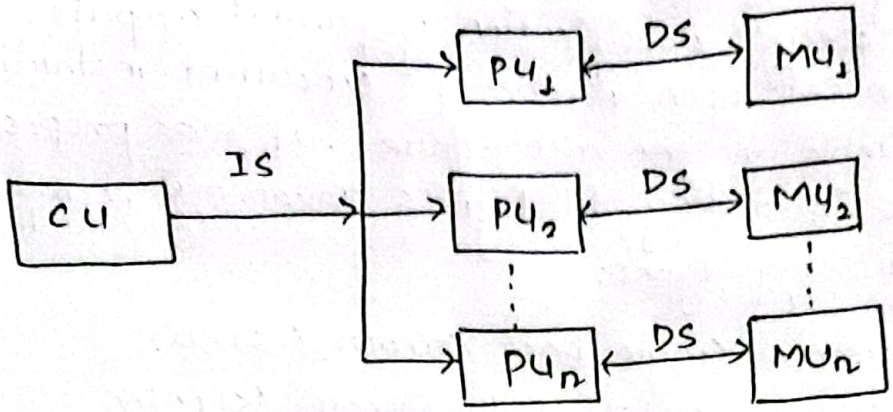

fig: SIMD computer.

(iii) MISD

↳ Represents organization of a system containing multiple processing unit under control of multiple control unit.

↳ Each processor receive different instruction.

↳ Only theoritical organization, implementation is impractical.

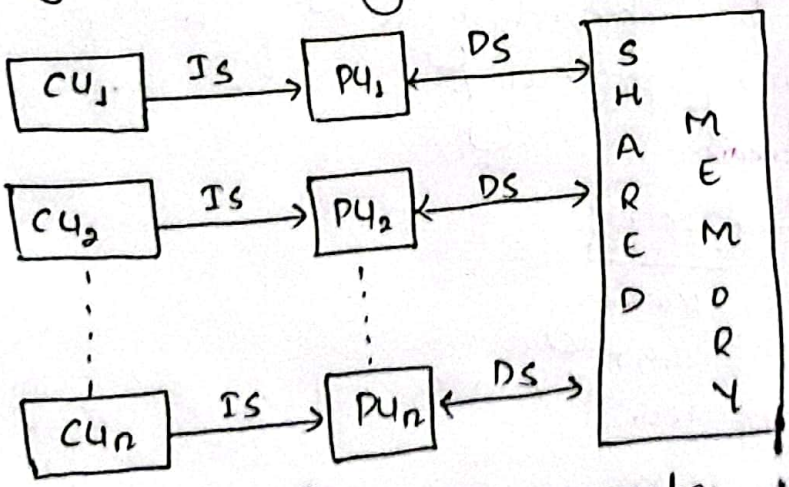

fig: MISD computer architecture.

(iv) MIMD

↳ Multiprocessor and multicomputer systems are classified in this category.

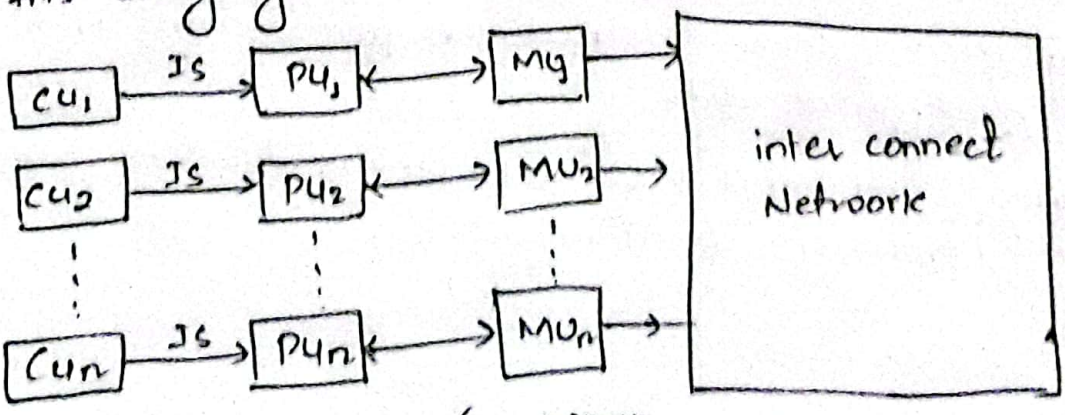

fig: MIMD.

9.4. Multicore Architecture.

↳ Multicore cpu chip is shown below.

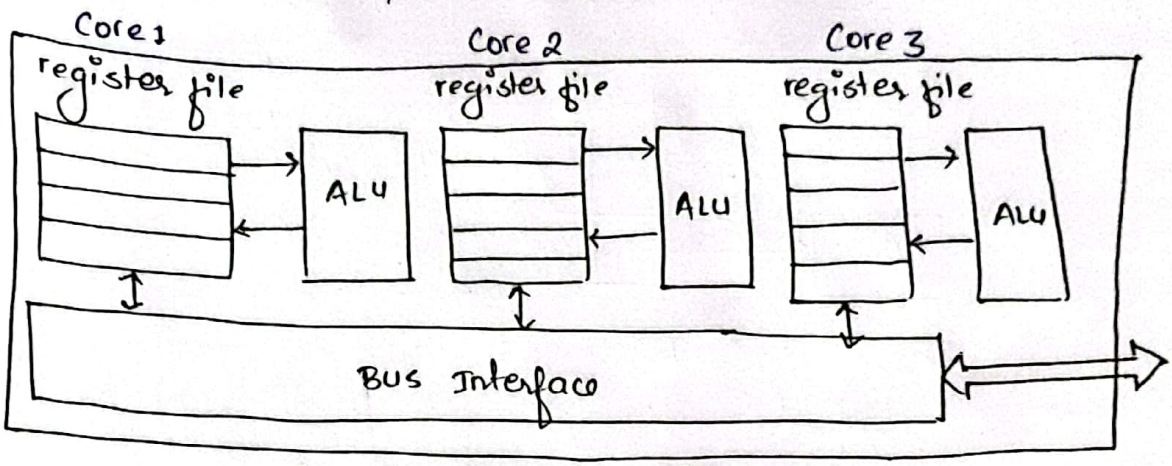

fig: Multicore cpu chip.

↳ Multicore refers to an architecture in which a single physical processor incorporates the core logic of more than one processor. A single integrated circuit is used to package or hold these processors. These single integrated circuits are known as die.

↳ Multicore architecture places multiple processor cores and bundles them as single physical processor. The objective is to create a system that can complete more tasks at the same time, thereby gaining better overall system performance.

↳ The concept of multicore technology is mainly centered on the possibility of parallel computing, which can significantly boost computer speed and efficiency by including two or more CPUs in a single chip. This reduces the system's heat and power consumption. This means much better performance with less or the same amount of energy.

↳ Multicore technology is very efficien· effective in challenging tasks and applications, such as encoding, 3-D gaming and video editing.