ASSIGNMENT 2

Computer Architectures

Name: Sudha Kumari
Roll No. 2210013135115
B.Tech CSE 3rd Year 6 th Sem

Q1 explain the basic and intermediate concept of pipelining in detail. also explain the types, limitation and benefits of pipeline in Computer architecture.

Pipelining is a technique used in computer architecture to improve the performance of the CPU.
It works by overlapping the execution of multiple instructions like an assembly line in a factory.

Typical Stages:

Instruction Fetch: Get instruction from memory

Instruction Decode: Decode instruction and fetch registers.

Execute Perform arithmetic/logic operation

Memory Access Read/Write memory if needed

Write Back Write result back to register.

Types of Pipelining:

Arithmetic Pipelining: Used for operations like floating-point arithmetic, where a big operation like multiply is broken into stages.

Instruction Pipelining: Splits instruction execution into stages the usual IF, ID, EX, MEM, WB.

Processor Pipelining: Multiple functional units operate in a pipeline fashion like a superscalar processor.

Vector Pipelining: Works with vector processors where operations are done on large data arrays.

Benefits of Pipelining:

Increased Instruction

Throughput:
Multiple instructions processed at once.

Better CPU Resource Utilization:
All parts of CPU are kept busy.

Higher Clock Speed Potential:
Since each stage is simple and fast, clock cycle time can be reduced.

Scalability:
Modern processors can use deep pipelines

Limitations of Pipelining:

Pipeline Hazards:
Can cause stalls, reducing performance.

Complex Design:
Managing dependencies, branches, and hazards makes pipeline control logic complicated.

Unequal Stage Times:
Some stages may take longer than others, causing

imbalance pipeline bubbles.

Diminishing Returns:
Beyond a point, adding more pipeline stages gives minimal performance improvement.

Penalty on Branch Misprediction:
If the branch prediction is wrong, the entire pipeline needs to be flushed.

Q2. Write difference between linear and nonlinear pipeline processors. Also explain advance optimization of cache performance.

1. Linear Pipeline Linear pipeline is a pipeline in which a series of processors are connected together in a serial manner. In linear pipeline the data flows from the first block to the final block of processor. The processing of data is done in a linear and sequential manner. The input is supplied to the first block and we get the output

from the last block till which the processing of data is being done. The linear pipelines can be further be divided into synchronous and asynchronous models. Linear pipelines are typically used when the data transformation process is straightforward and can be performed in a single path.

2. Non-Linear Pipeline Non-Linear pipeline is a pipeline which is made of different pipelines that are present at different stages. The different pipelines are connected to perform multiple functions. It also has feedback and feed-forward connections. It is made such that it performs various function at different time intervals. In Non-Linear pipeline the functions are dynamically assigned. In Non-Linear pipeline the functions are dynamically assigned.

Advanced Optimization of Cache Performance:

1. Cache Prefetching
2. Victim Cache
3. Multi-level Caches
4. Associativity Enhancement
5. Cache Replacement Policies
6. Write Optimization Techniques

7. Non-Blocking Caches
8. Cache Compression

Q3. What are issues which can affect the performance of pipelining?

1. Pipeline Hazards.

a. Structural Hazards
Occur when hardware resources are insufficient for simultaneous execution.

Example: Two instructions need to access memory at the same time.

Solution: Duplicate resources like separate memory for instructions and data.

b. Data Hazards

c. Control Hazards

Occur with branch instructions like if, for, goto.

Pipeline does not know which instruction to fetch next until branch is resolved.

Solution: Branch prediction, speculative execution, branch delay slots.

3. Pipeline Stalls

Stalls are intentional delays inserted into the pipeline to resolve hazards.

Every stall wastes a clock cycle it reduces performance.

Solution: Minimize stalls through smarter hazard handling

Pipeline Depth

Deep pipelines many stages can increase instruction throughput.

However, deeper pipelines suffer more from hazards, branch misprediction penalties, and higher complexity.

Q4. What do you mean by term cache cohency? Also explain systolic architecture in details.

When multiple processors or cores have their own local caches like in a multi-core CPU, they might all cache copies of the same memory location.
If one core updates its cached value, the others could have outdated stale copies.
This inconsistency creates a cache cohesrence problem.

Suppose:

Core 1 has cached variable x 5

Core 2 also has cached variable x 5

Core 1 updates x 10

Now, Core 2 still thinks x 5, which is wrong unless cache cohesrence is enforced!

Systolic Architecture:

A Systolic Architecture is a network of tightly coupled processing elements PEs that rhythmically compute and pass data through the system in a synchronized manner.

Key Characteristics of Systolic Architecture

Array of Processing Elements PEs:

A grid 1D, 2D of small processing units, each performing simple operations.

Data Movement:

Instead of moving data back and forth to memory, data moves between neighboring processors.

Synchronization:

All PEs work in lock-step synchronized clock pulses.

Local Communication:

PEs only communicate with neighbors, avoiding costly global communication.

Highly Parallel:

Many operations happen simultaneously.

Q5. Explain super scaler super pipeline design in detail.

Superscalar Design: Superscalar architecture allows a CPU to execute multiple instructions per clock cycle by using multiple execution units.

Key Concepts of Superscalar
Multiple Pipelines: The CPU has several functional units ALUs, FPUs, etc. Operating in parallel.

Instruction-Level Parallelism ILP: CPU looks for independent instructions that can be executed together.

Dynamic Scheduling: Hardware dynamically decides which instructions to execute together.

Issue Width: Number of instructions the CPU can issue per clock cycle.

1. ADD R1, R2, R3
2. MUL R4, R5, R6
3. SUB R7, R8, R9

In a superscalar CPU:

ADD goes to ALU-1

MUL goes to Multiplier Unit

SUB goes to ALU-2

→ All execute in one cycle!

2. Superpipelined Architecture:

Superpipelined architecture increases CPU speed by breaking stages into even smaller sub-stages, and allowing faster clock cycles.

Instead of executing multiple instructions per cycle like superscalar, superpipeline executes one instruction at a time, but stages complete faster, so more instructions are

in flight at once.

Suppose normal pipeline:

Stage IF ID EX MEM WB

Superpipeline splits into smaller stages:

Stage IF1 IF2 ID1 ID2 EX1 EX2 MEM1 MEM2 WB1 WB2

Thus, more instructions are working through different mini-stages.

Q6. What is clustering. Explain multi threaded architecture and diffences between distributed and shared MIMD.

Clustering means grouping multiple computers or nodes together so that they work together like a single system.

Features of Clustering

High Availability: If one node fails, others keep working.

Load Balancing: Workload is distributed among nodes.

Scalability: Easily add more nodes.

Parallelism: Tasks can be split across nodes.

Multithreaded Architecture:

Multithreaded Architecture is a CPU design where multiple threads are managed and executed within a single core or processor.

Types of Multithreaded Architectures:

Fine-Grained Multithreading Switches threads every clock cycle even if no stall happens.

Coarse-Grained Multithreading Switches threads only when a stall occurs like waiting for memory.

Simultaneous Multithreading Executes multiple threads at the same time in a superscalar CPU (example: Intel Hyper-Threading

1. Shared Memory MIMD:

Memory Access: All processors share a single, globally accessible memory space.

Communication: Data sharing and communication are efficient and transparent as processors can directly access each others memory.

Scalability: Scalability can be limited by memory contention as more processors compete for the same memory resources.

Programming: Easier to program compared to distributed memory due to the shared memory space.

Example: Silicon Graphics machines and Sun SMP Symmetric multi-Processing.

2. Distributed Memory MIMD:

Memory Access: Each processor has its own local memory.

Communication: Communication between processors is achieved through message passing, requiring explicit communication mechanisms.

Scalability: Scales better than shared memory systems as each processor has its own memory space, reducing contention.

Programming: More complex to program due to the need for explicit message passing and memory management.

Example: Multicomputers using interconnection networks like mesh or hypercube.