Memory Hierarchy & Interfacing Overview

Memory hierarchy ranks storage units based on parameters like access time, size, cost, and usage frequency.

The purpose is to bridge the **speed mismatch** between the fast **processor** and slower memory, while keeping costs reasonable.

Memory Hierarchy Ranking

By Access Time (Fastest to Slowest)

- **Registers**: Made of **flip-flops**, embedded in processor → **least access time**.
- **SRAM Caches**: Next fastest.
- **DRAM Main Memory**.
- **Secondary Memory**: Slowest (e.g., disks).

By Cost & Usage Frequency (Cheapest/Least Used to Most Expensive/Most Used)

Hierarchy reverses: Secondary memory (cheap, low usage) → Main memory → Caches → Registers (expensive, high usage).

Key Concepts: Hit, Miss, Hit Ratio

Information at level n is a subset of level n+1 (higher levels have more data).

- **Hit**: Data found at requested level n.
- **Miss**: Data not at level n → check level n+1.
- **Hit Ratio (h)**: Probability data is found at that level (e.g., 80% = 0.8).
    - Example: 100 instructions in secondary memory, 80 in main memory → **h = 80/100 = 0.8**.
- **Miss Ratio**: 1 - h.

Last level (secondary storage) always has h = 100% (program must exist to execute).

Memory Interfacing: Two Methods

Memory interfacing connects memory levels to **processor** and **I/O peripherals**.

Processor speed measured in MIPS (Million Instructions Per Second) → goal: feed instructions fast + cost-effective.

Way 1: Parallel/Simultaneous Connection

All memory levels (M1, M2, M3) connected directly to processor → checks all **side-by-side**.

Assume: t_1 < t_2 < t_3 (access times), hit ratios h_1, h_2.

Effective/Average Access Time formula:
$T_{avg} = h_1 \cdot t_1 + (1 - h_1) \cdot h_2 \cdot t_2 + (1 - h_1) \cdot (1 - h_2) \cdot t_3$

Logic:

- h_1 chance: Found in M1 (time t_1).
- (1-h_1) \cdot h_2: Miss M1, hit M2 (time t_2).
- (1-h_1) cdot (1-h_2): Miss M1 & M2, go to M3 (time t_3).

Parallel access = operations happen simultaneously.

Way 2: Sequential/Hierarchical Connection

Processor checks levels one-by-one: M1 → if miss, M2 → if miss, M3.

Effective Access Time formula:
$T_{avg} = h_1 \cdot t_1 + (1 - h_1) \cdot (t_1 + t_2) + (1 - h_1) \cdot (1 - h_2) \cdot (t_1 + t_2 + t_3)$

Key Difference: Misses include **prior levels' access times** (sequential).

Practical Context

- Programs loaded from **secondary storage** (non-volatile) to **main memory**.

- **Frequently accessed parts → cache**.
- No program in storage = no execution (like no audio file = no playback).

Key Takeaways for Revision

Concept	Key Point	Formula/Detail
Hierarchy	Fast/small/expensive at top	Registers → Cache → Main → Secondary
Hit Ratio	% chance of finding data	h = hits/total (e.g., 0.8)
Way 1 (Parallel)	Simultaneous check	No prior times on miss
Way 2 (Sequential)	Level-by-level	$t_1 + t_2$ on M2 miss
Goal	Match processor MIPS to memory speed	Cost-effective