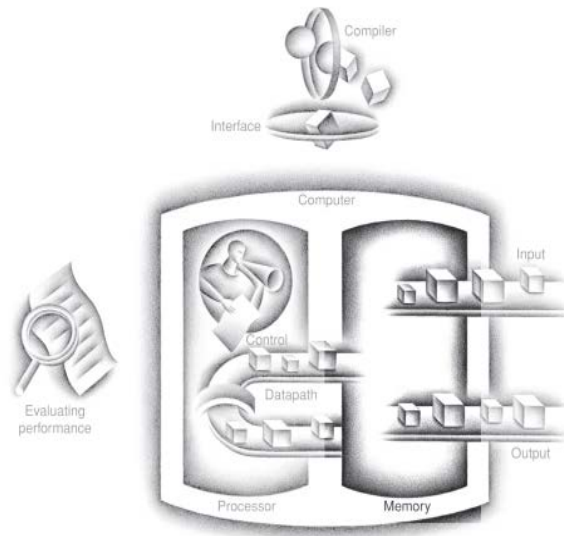

Chapter 5

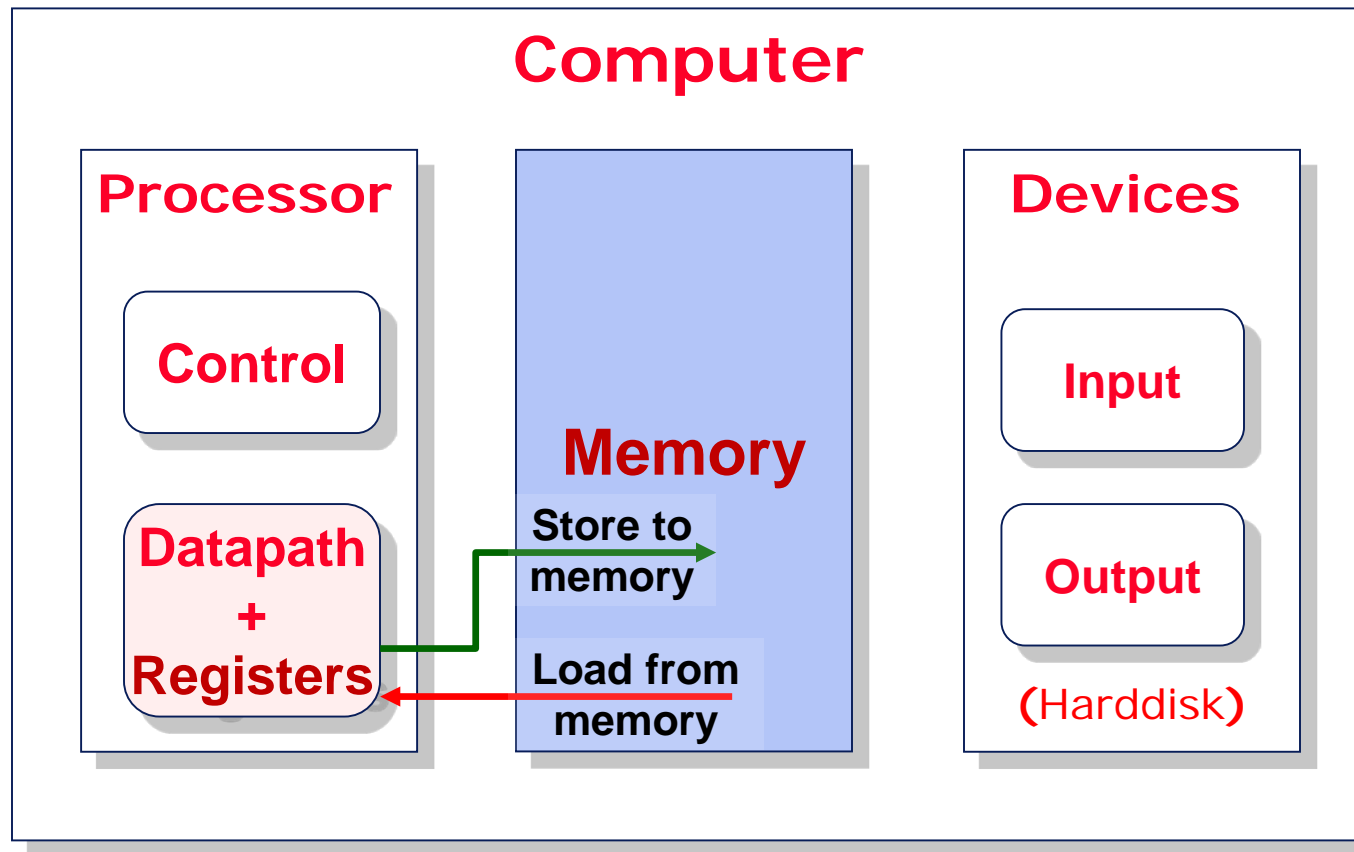
Large and Fast : Exploiting the Memory Hierarchy



Introduction	374
Memory Technologies	378
The Basics of Caches	383
Measuring and Improving Cache Performance	398
Dependable Memory Hierarchy	418
Virtual Machines	424
Virtual Memory	427

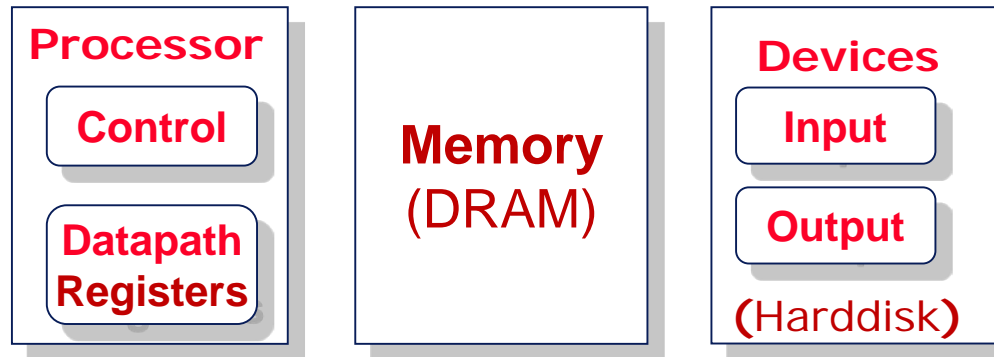
A Common Framework for Memory Hierarchy	454
Using a Finite-State Machine to Control a Simple Cache	461
Parallelism and Memory Hierarchies: Cache Coherence	466
Parallelism and Memory Hierarchy: Redundant Arrays of Inexpensive Disks	470
Advanced Material: Implementing Cache Controllers	470
Real Stuff: The ARM Cortex-A8 and Intel Core i7 Memory Hierarchies	471
Going Faster: Cache Blocking and Matrix Multiply	475
Fallacies and Pitfalls	478
Concluding Remarks	482
Historical Perspective and Further Reading	483
Exercises	483

Data Transfer: The Big Picture



Registers are in the datapath of the processor. If operands are in memory we have to **load** them to processor (registers), operate on them, and **store** them back to memory.

Quality vs. Quantity



	Capacity	Latency	Cost/GB
Register	100s Bytes	20 ps	\$\$\$\$
SRAM	100s KB	0.5-5 ns	\$\$\$
DRAM	100s MB	50-70 ns	\$
Hard Disk	100s GB	5-20 ms	Cents
Ideal	1 GB	1 ns	Cheap

The “Memory Wall”

- Processor vs. DRAM speed disparity continues to grow
- What we want:
 - A **BIG** and **FAST** memory
 - Memory system should perform like 1GB of SRAM (1ns access time) but cost like 1GB of slow memory

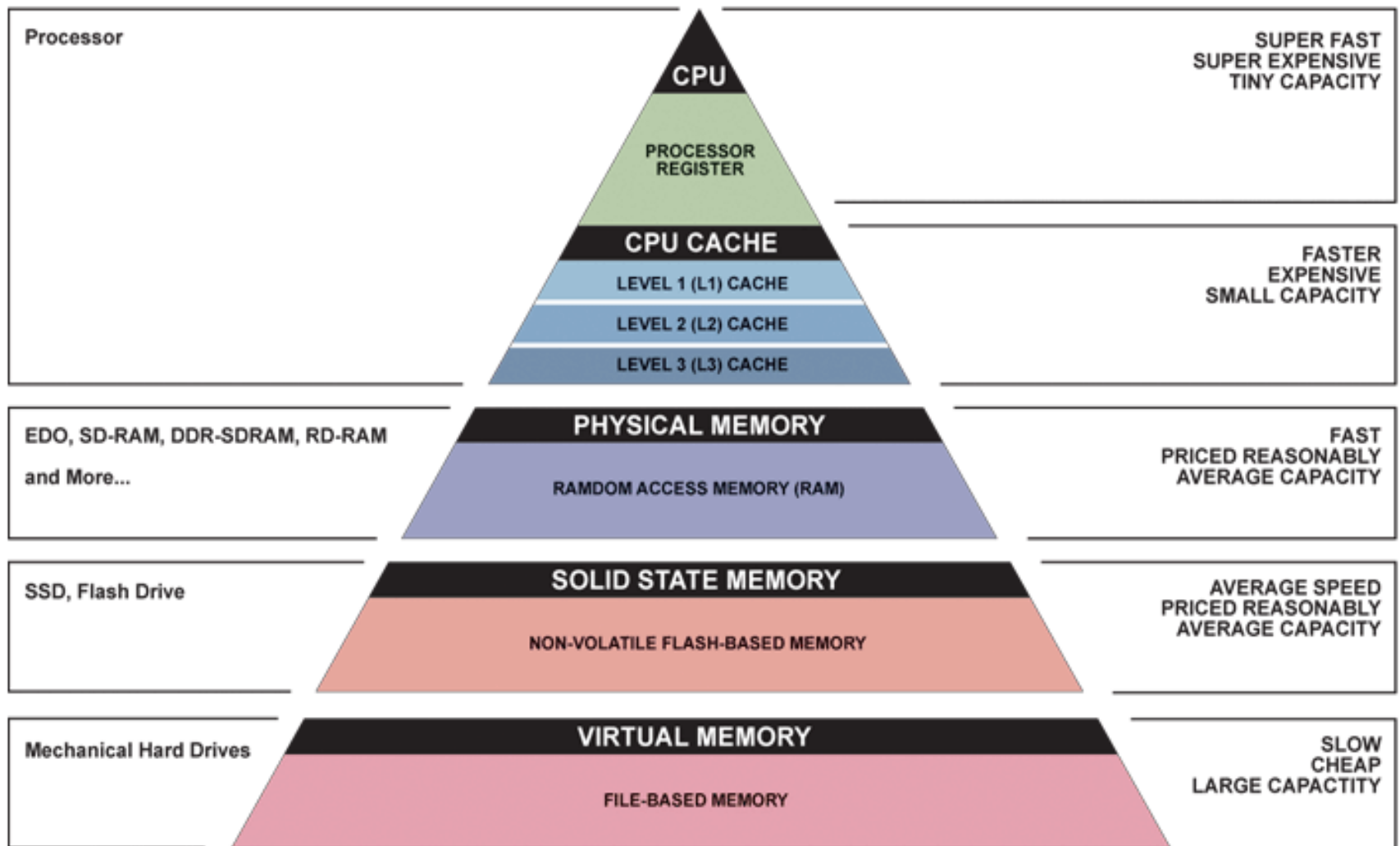
Key concept:

Use a **hierarchy** of memory technologies:

- ❖ Small but fast memory near CPU
- ❖ Large but slow memory farther away from CPU

- Good memory hierarchy (cache) design is increasingly important to overall performance
 - 1980: **no cache** in μ proc, 1989: first Intel CPU with **a cache on chip**
 - 1995: **2-level cache** on chip 2020 ?

Great Idea : Principle of Locality / Memory Hierarchy



Principle of Locality (pp. 374)

```
int sum = 0;
int x[1000];

for(int c = 0; c < 1000; c++) {
    sum += c;

    x[c] = 0;
}
```



❑ “Programs will access **only a small proportion** of the address space **at any time**”

┆ Not any place accessed randomly with equal probability!

❑ **Temporal Locality** (locality in time)

┆ Items accessed are likely to be accessed again soon

┆ e.g., instructions and data in a loop

❑ **Spatial Locality** (locality in space)

┆ Items near the accessed item are likely to be accessed soon

┆ e.g., sequential instruction access, an array of data

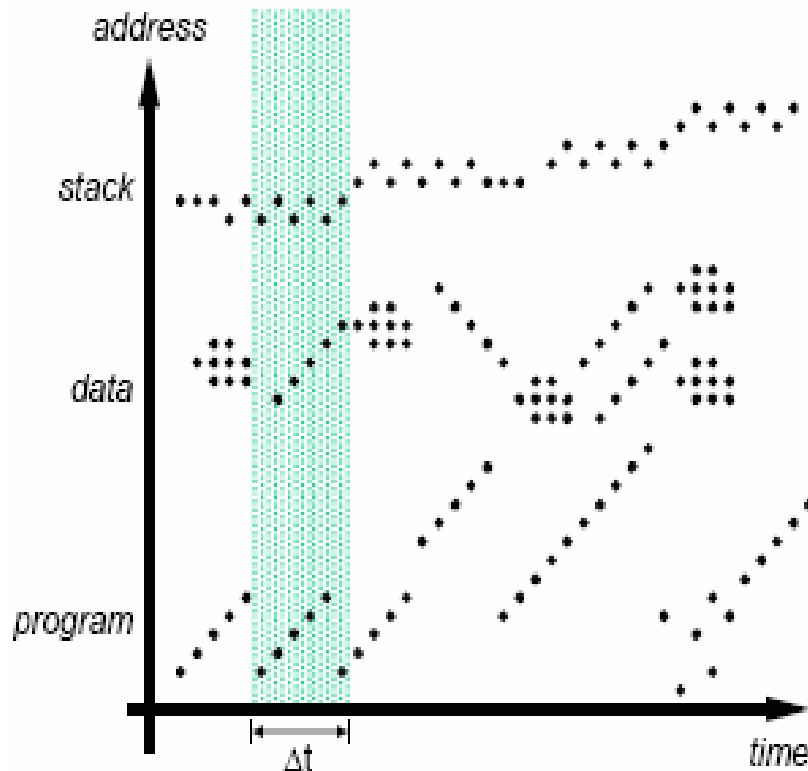
How to Take Advantage of Locality

- Keep *most recently accessed* data items closer to the processor

| Caches

Main memory

Hard Disk



Capture **the working set**
and keep it in the
memory closest to CPU

The Memory Hierarchy Levels : Terminology

- ❑ **Block** (or cache line): unit of copying

- | It has multiple words

- ❑ If accessed data is present in upper level

- | **Hit** : access satisfied by upper level

- | **Hit ratio** = $\#hits / \#accesses$

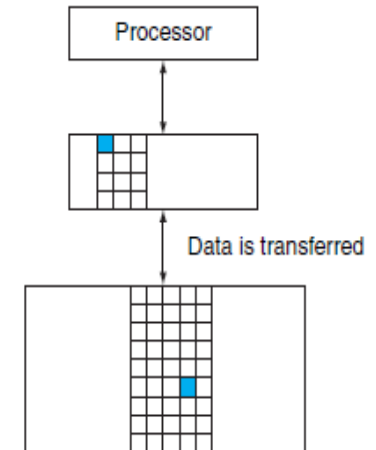
- ❑ If accessed data is absent

- | **Miss** : block copied from lower level

- | **Miss ratio** = $\#misses / \#accesses = 1 - \text{Hit ratio}$

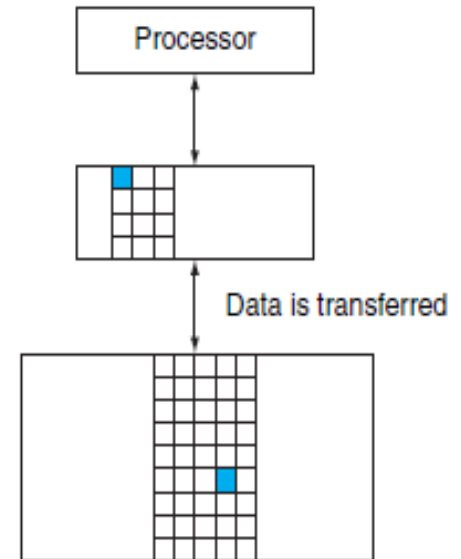
- | Time taken : **Miss Penalty** (Time to replace a block in that level with the corresponding block from a lower level)

- ❑ Hit Time \ll Miss Penalty



Common Framework for Memory Hierarchy

- ❑ Q1 block placement
- ❑ Q2 block identification
- ❑ Q3 block replacement
- ❑ Q4 write policy



❑ Modern systems?

Figure 5.44

Summary
