MEMORY HIERARCHY DESIGN

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

THE UNIVERSITY OF UTAH

CS/ECE 6810: Computer Architecture

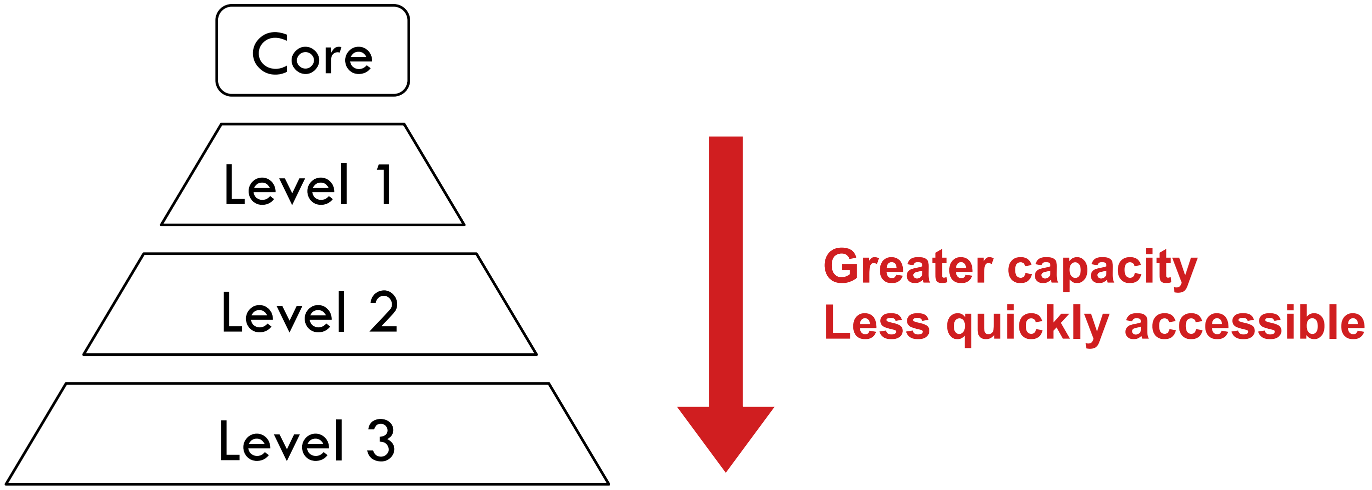
Overview

- Basics
  - Why use a memory hierarchy
  - Memory technologies
  - Principle of locality
- Caches
  - Concepts: block, index, tag, address, cache policies
  - Performance metric (AMAT) and optimization techniques
- Virtual Memory
  - Paging, address translation, TLB

Memory Hierarchy

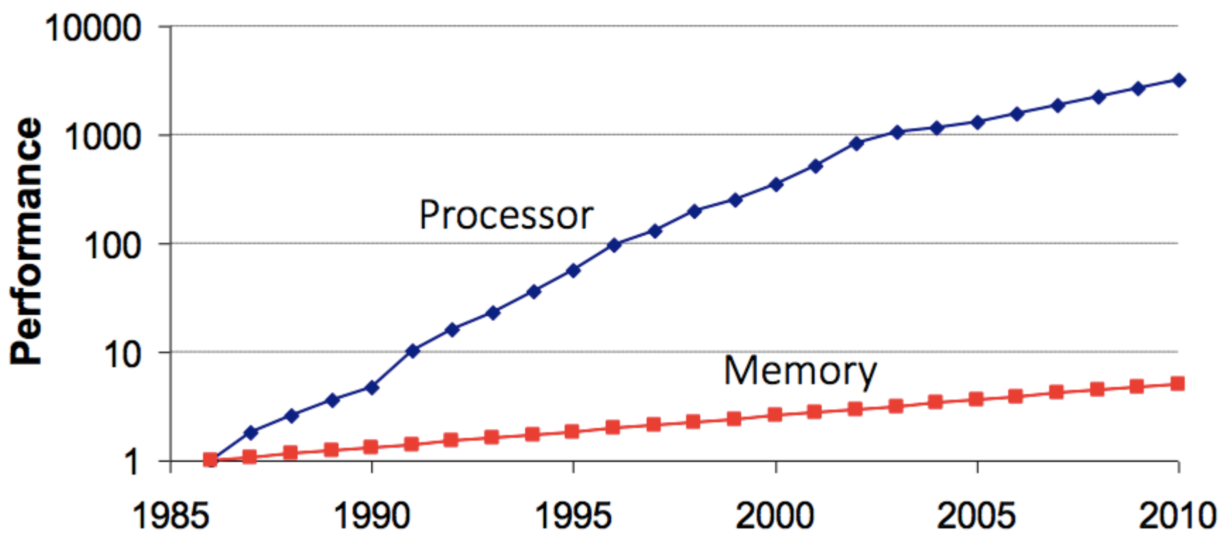
"Ideally one would desire an indefinitely large memory capacity such that any particular [...] word would be immediately available [...] We are [...] forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."

-- Burks, Goldstine, and von Neumann, 1946



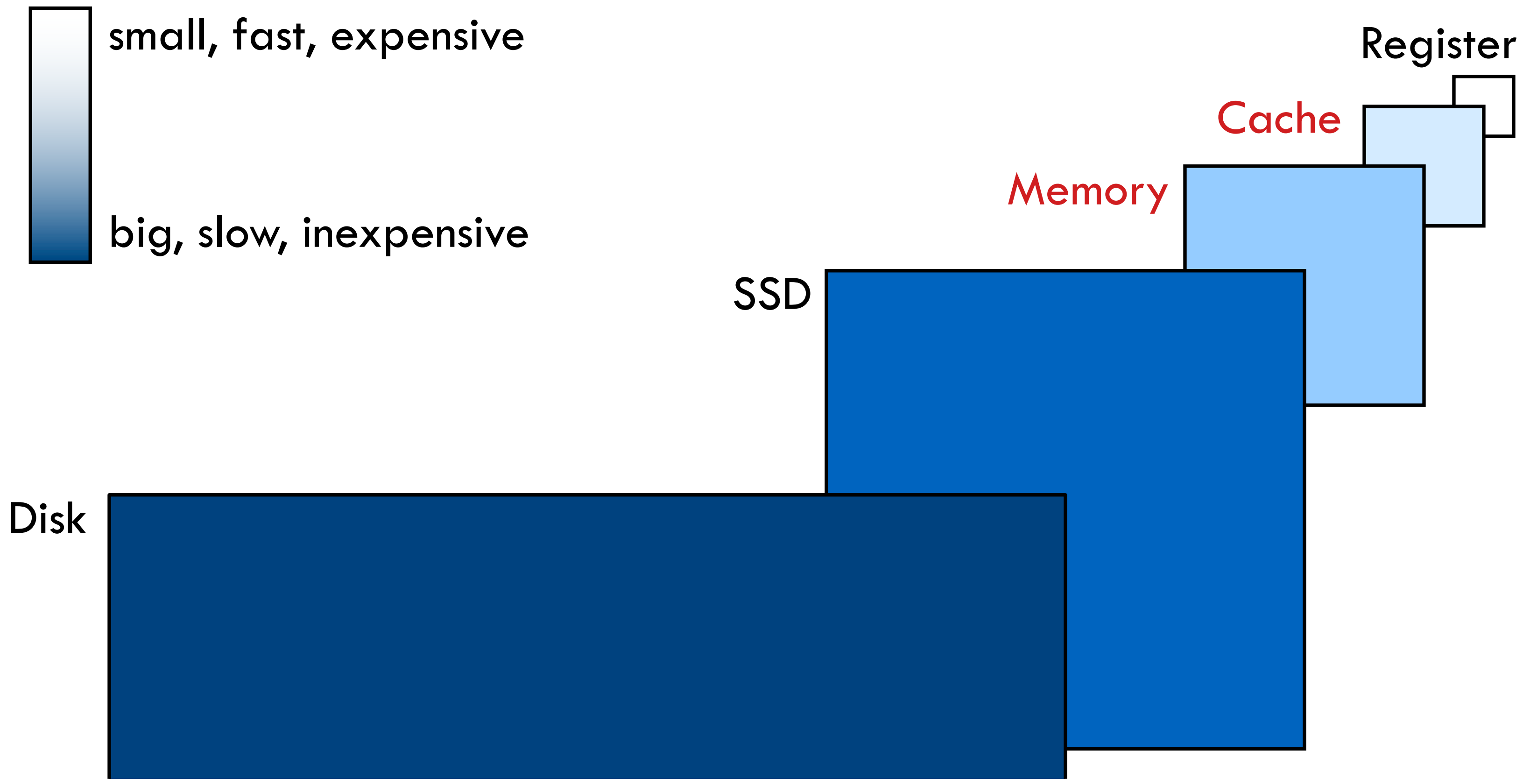
The Memory Wall

- Processor-memory performance gap increased over 50% per year
  - Processor performance historically improved ~60% per year
  - Main memory access time improves ~5% per year



Modern Memory Hierarchy

- Trade-off among memory speed, capacity, and cost



Memory Technology

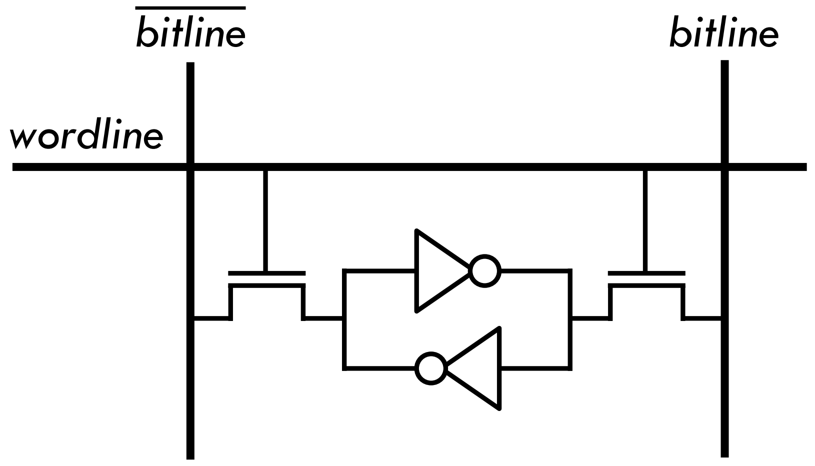
- Random access memory (RAM) technology
  - access time same for all locations (not so true anymore)

  - Static RAM (SRAM)
    - typically used for caches
    - 6T/bit; fast but – low density, high power, expensive

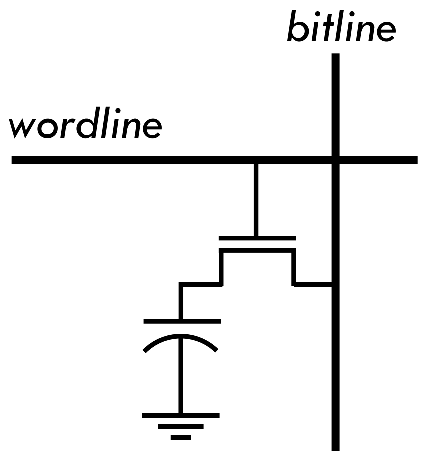
  - Dynamic RAM (DRAM)
    - typically used for main memory
    - 1T/bit; inexpensive, high density, low power – but slow

RAM Cells

- 6T SRAM cell
  - internal feedback maintains data while power on

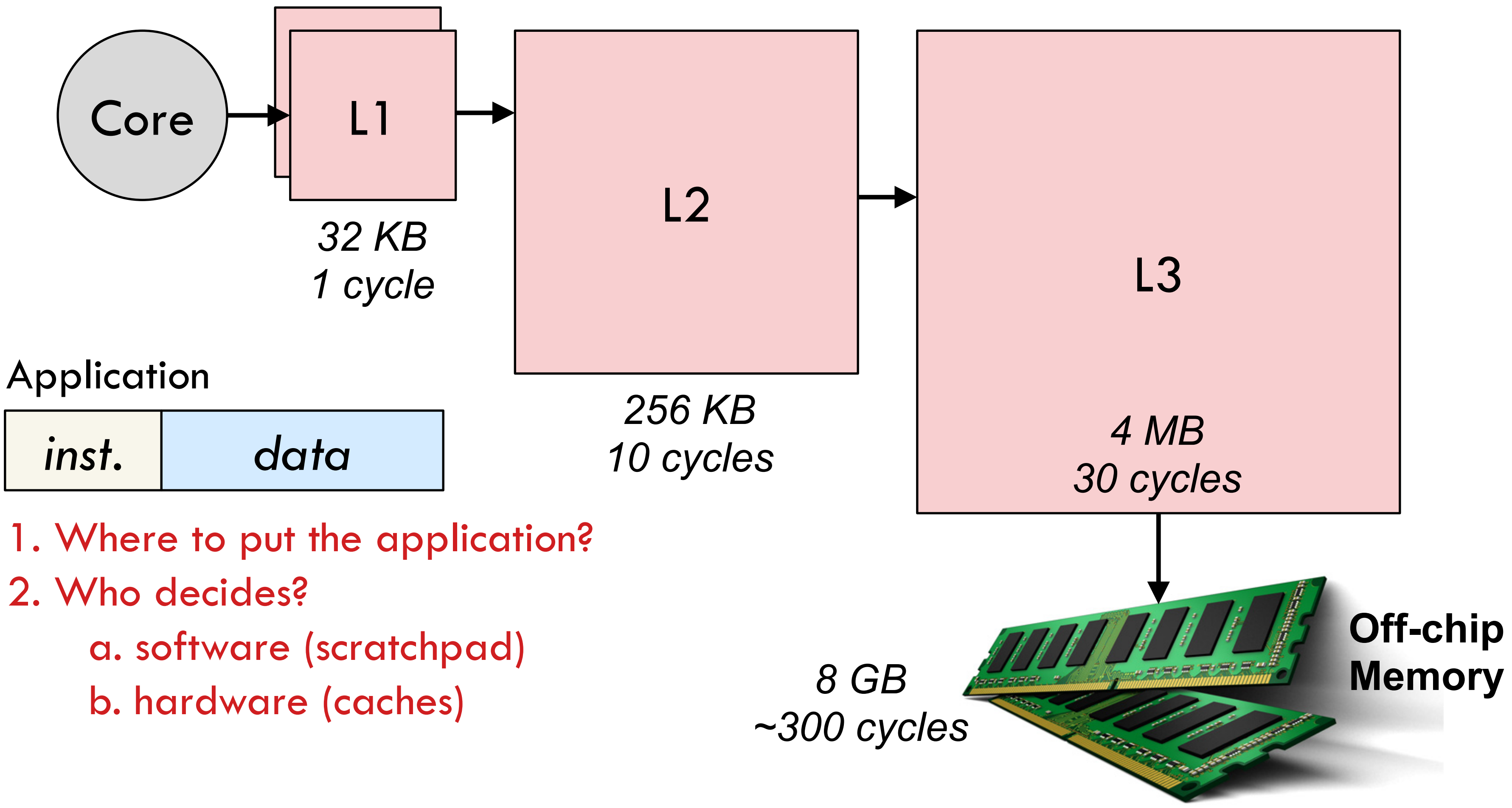


- 1T-1C DRAM cell
  - needs refresh regularly to preserve data



Cache Hierarchy

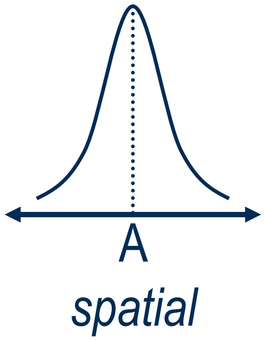
- Example three-level cache organization

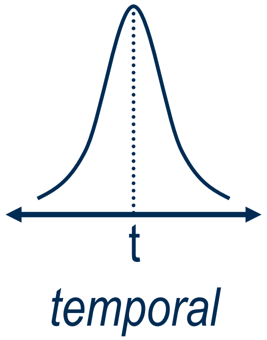


1. Where to put the application?
2. Who decides?
    a. software (scratchpad)
    b. hardware (caches)

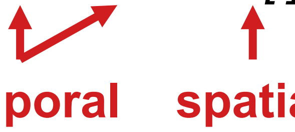
Principle of Locality

- Memory references exhibit localized accesses
- Types of locality
  - *spatial*: probability of access to $A+\delta$ at time $t+\varepsilon$ highest when $\delta \rightarrow 0$
  - *temporal*: probability of accessing $A+\varepsilon$ at time $t+\delta$ highest when $\delta \rightarrow 0$





```
for (i=0; i<1000; ++i) {
        sum = sum + a[i];
}
```



Key idea: store local data in fast cache levels

Cache Terminology

- Block (cache line): unit of data access
- Hit: accessed data found at current level
  - *hit rate: fraction of accesses that finds the data*
  - *hit time: time to access data on a hit*
- Miss: accessed data NOT found at current level
  - miss rate: 1 – hit rate
  - miss penalty: time to get block from lower level

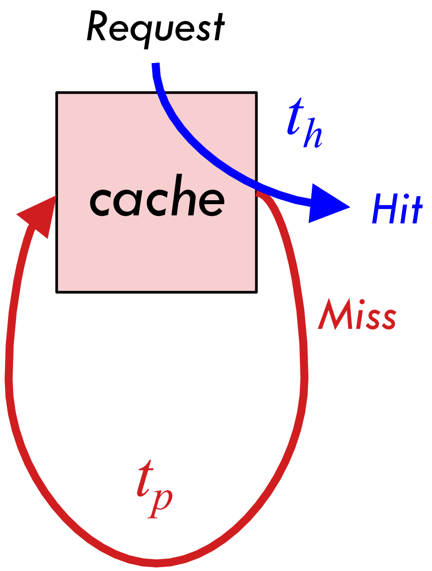
hit time << miss penalty

Cache Performance

- Average Memory Access Time (AMAT)

Outcome	Rate	Access Time
Hit	r_h	t_h
Miss	r_m	$t_h + t_p$

$$AMAT = r_h t_h + r_m (t_h + t_p)$$
$$r_h = 1 - r_m$$



$$AMAT = t_h + r_m t_p$$

problem: hit rate is 90%; hit time is 2 cycles; and accessing the lower level takes 200 cycles; find the average memory access time?

AMAT = 2 + 0.1x200 = 22 cycles

Example Problem

- Assume that the miss rate for instructions is 5%; the miss rate for data is 8%; the data references per instruction is 40%; and the miss penalty is 20 cycles; find performance relative to perfect cache with no misses
  - misses/instruction = 0.05 + 0.08 x 0.4 = 0.08
  - Assuming hit time =1
    - AMAT = 1 + 0.08x20 = 2.6
    - Relative performance = 1/2.6