TEXAS TECH UNIVERSITY
Department *of* Computer Science

CS5375 Computer Systems Organization and Architecture

Lecture 5

Instructor: Yong Chen, Ph.D.
Department of Computer Science
Texas Tech University
Yong.Chen@ttu.edu, 806-834-0284

Review of Last Lecture

- Measuring, Reporting and Summarizing Performance (cont.)
  - Measuring the performance with benchmarks, e.g., SPEC (Standard Performance Evaluation Corporation) CPU2017 benchmark suite
- Amdahl's Law (Amdahl's speedup model) and Scaled Computing

$$Speedup_{Amdahl} = \frac{1}{(1-f)+\frac{f}{n}}$$

$$\lim_{n\to\infty} Speedup_{Amdahl} = \frac{1}{1-f}$$

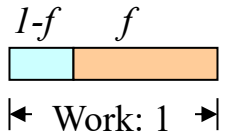
- Amdahl's Law in the Multicore Era
  - Symmetric v.s. asymmetric v.s. dynamic

Outline

- Amdahl's Law and Scaled Computing (cont.)
- Memory Hierarchy Design

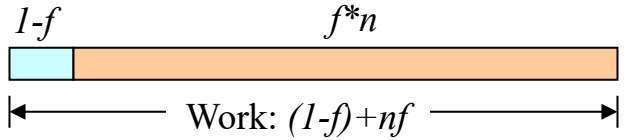
Scaled Computing Concept

- Tacit assumption in Amdahl's law
  - The problem size is fixed
  - The speedup emphasizes time reduction



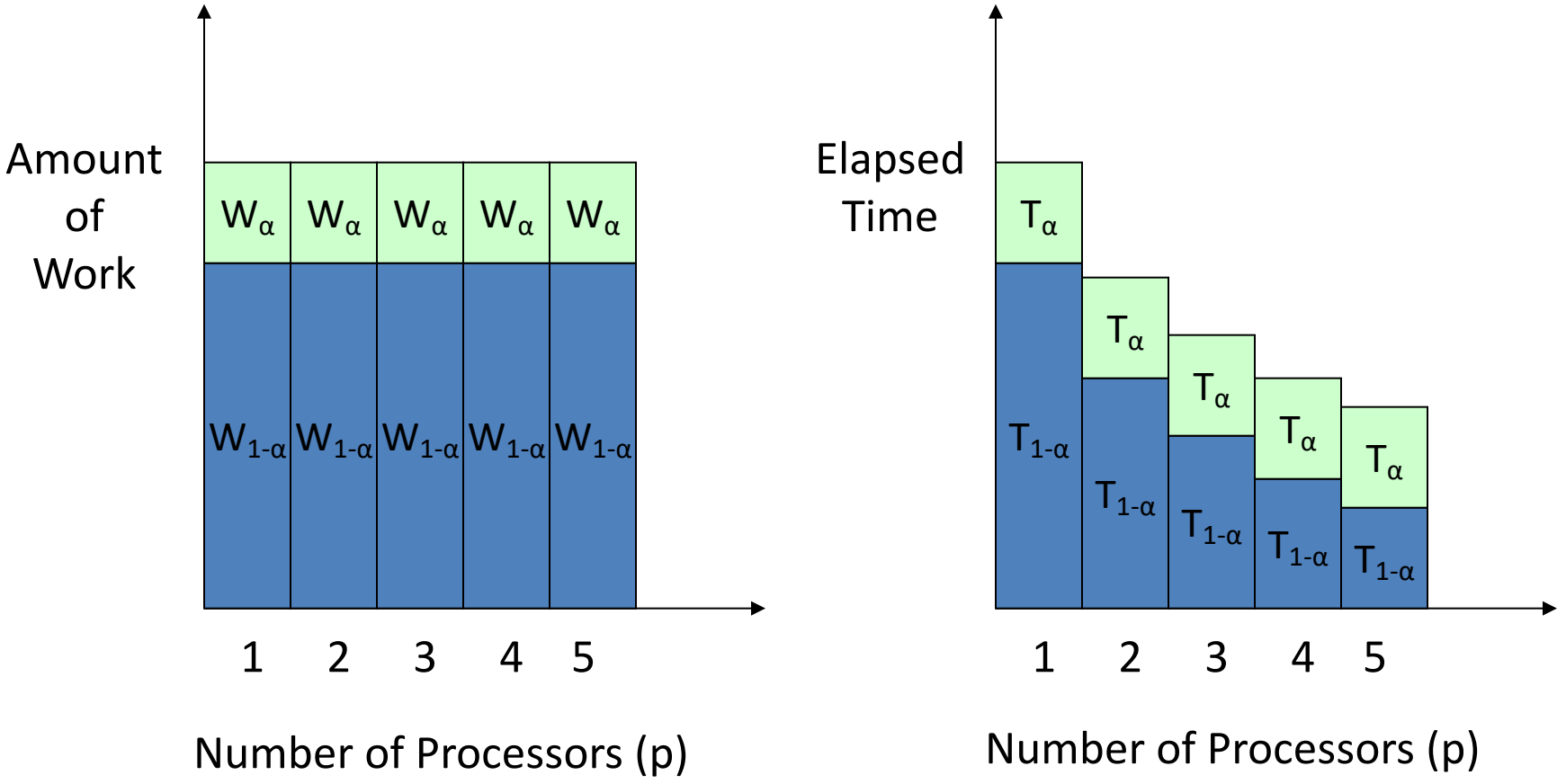
- Gustafson's Law, 1988
  - Fixed-time speedup model

$$Speedup_{fixed-time} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload} = (1-f)+nf$$

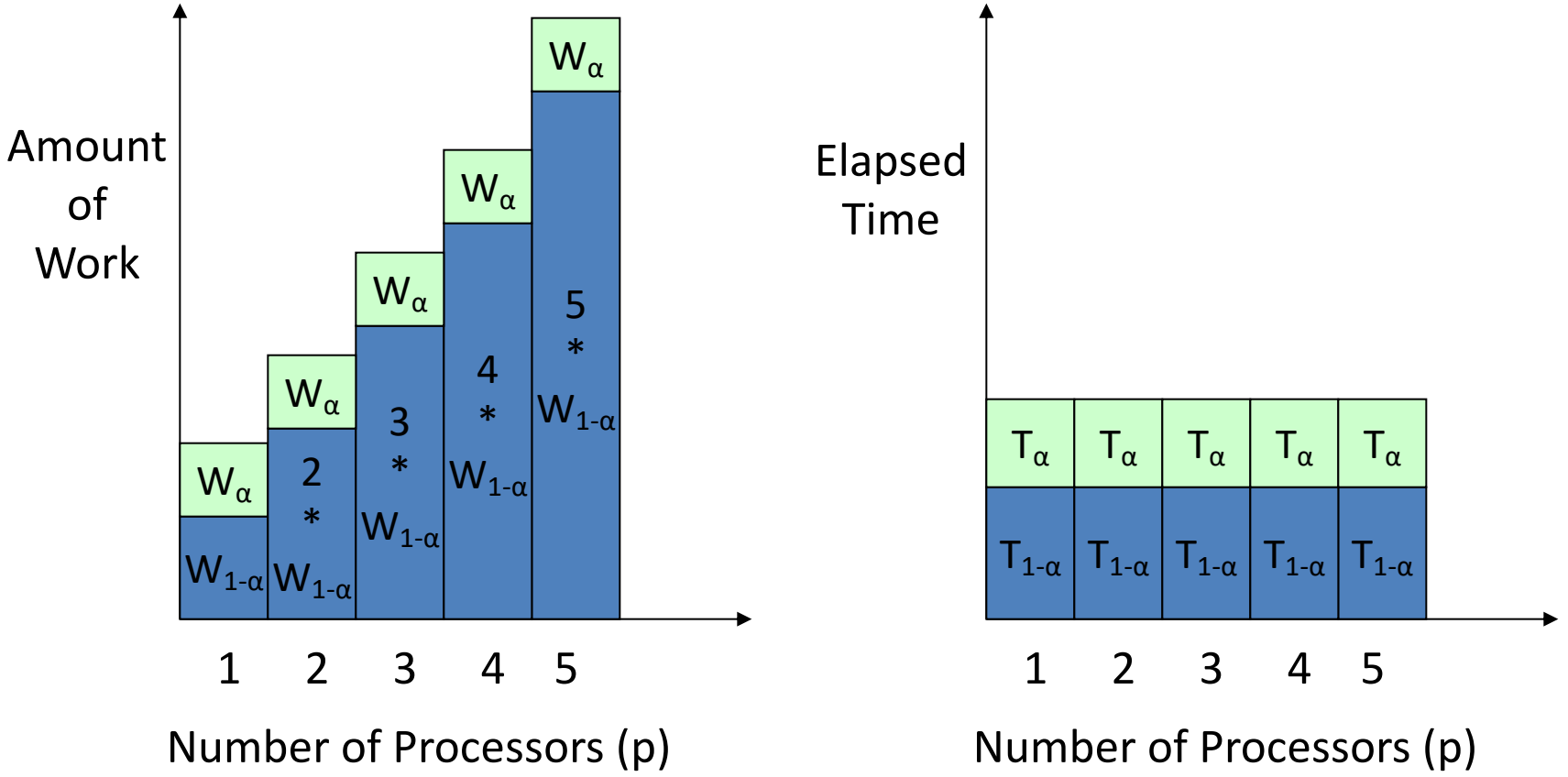


  - Shows the scaled speedup is a linear function of *n*
- Demonstrates great promise of parallelism

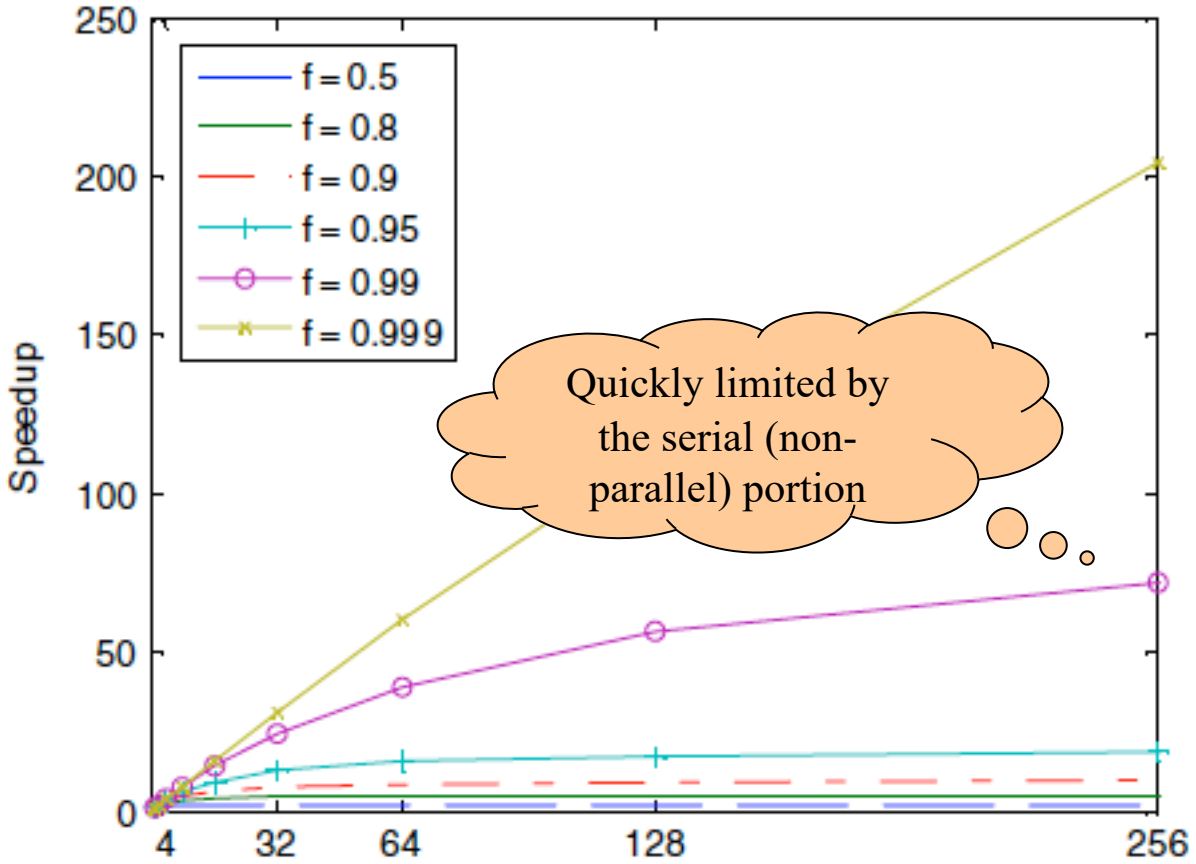
Fixed-Size Speedup



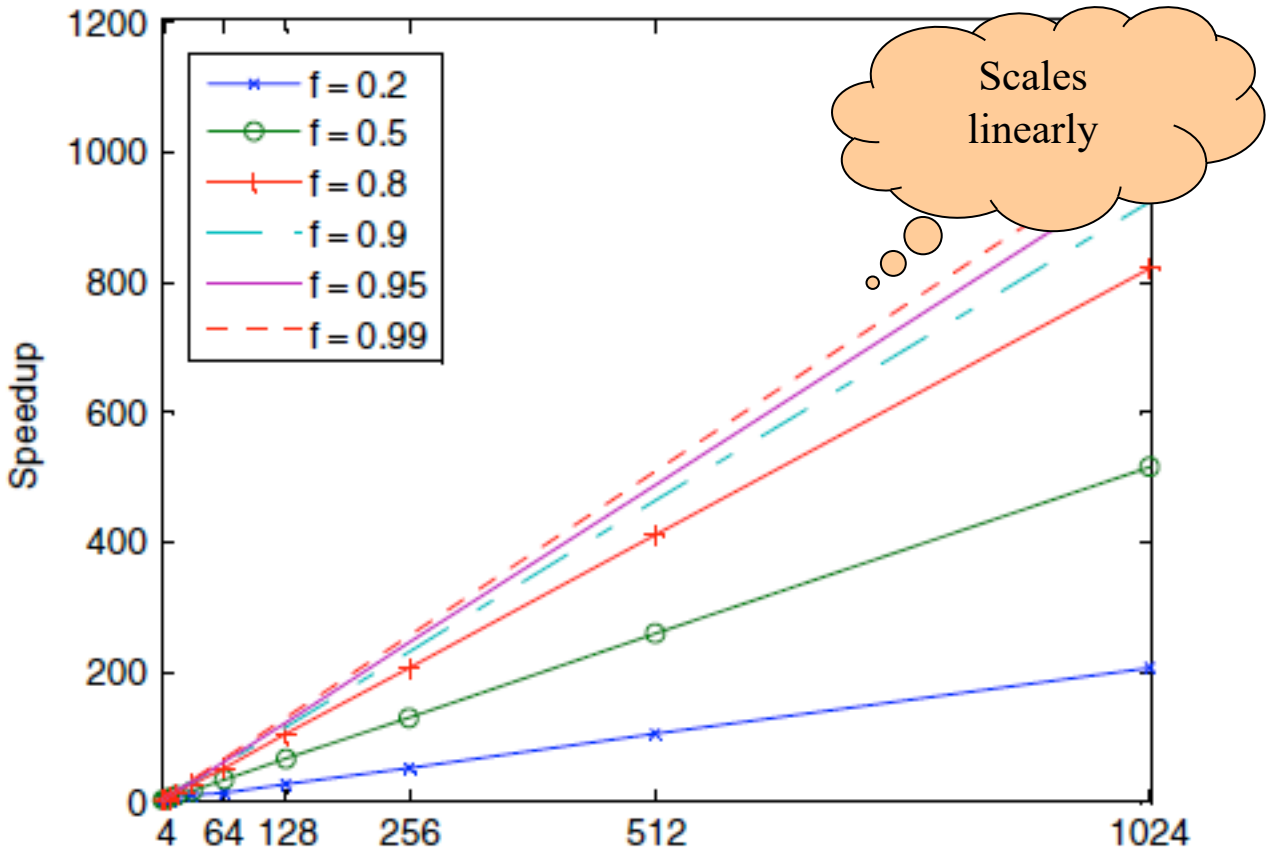
Fixed-Time Speedup



Fixed-size Speedup



Fixed-time Speedup

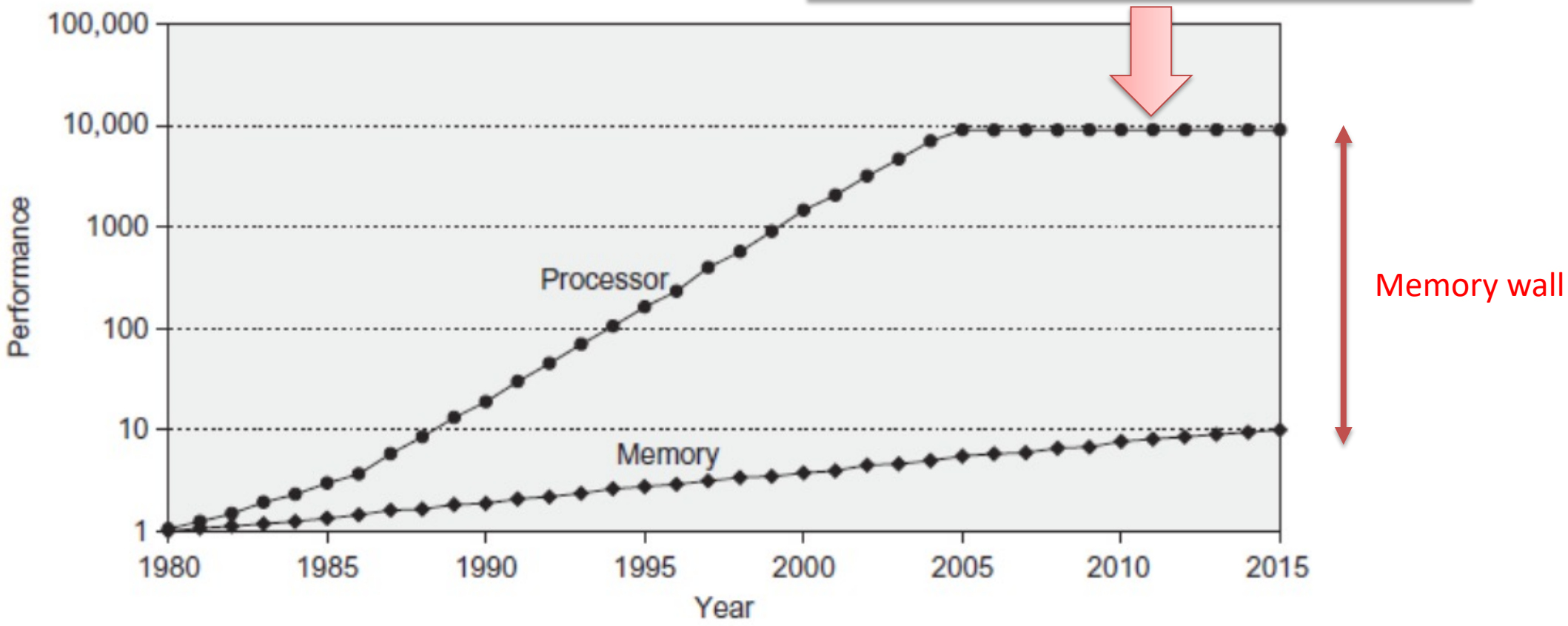


Outline

- Amdahl's Law and Scaled Computing (cont.)
- Memory Hierarchy Design

Performance Gap b.t. Processor and Memory

Even larger gap with the aggregate performance of multiple cores for multicore processors



Memory Hierarchy Design

- Memory hierarchy design becomes more crucial to bridge the performance gap with recent multicore processors

- Aggregate peak bandwidth grows with # cores:
  - Intel Core i7 can generate two references (memory accesses) per core per clock cycle
  - Four cores and 3.2 GHz clock
    - 25.6 billion 64-bit data references/second + 12.8 billion 128-bit instruction references/second
    - = 409.6 GB/s!
  - DRAM bandwidth is only 8% of this (34.1 GB/s)
  - Requires:
    - Two levels of cache per core
    - Shared third-level cache on chip
    - Multi-port, pipelined caches

Memory Hierarchy Concept

- Analogy: writing a paper in the library
  - Library shelves
  - Library Desk
  - The book you read now
  - Your memory
- Another example
  - Farm
  - Grocery store
  - Fridge

Principle of Locality

- Programs access a small proportion of instruction/data (address space) at any time

- Temporal locality (locality in time)
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, induction variables (variable that gets increased or decreased by a fixed amount on every iteration)

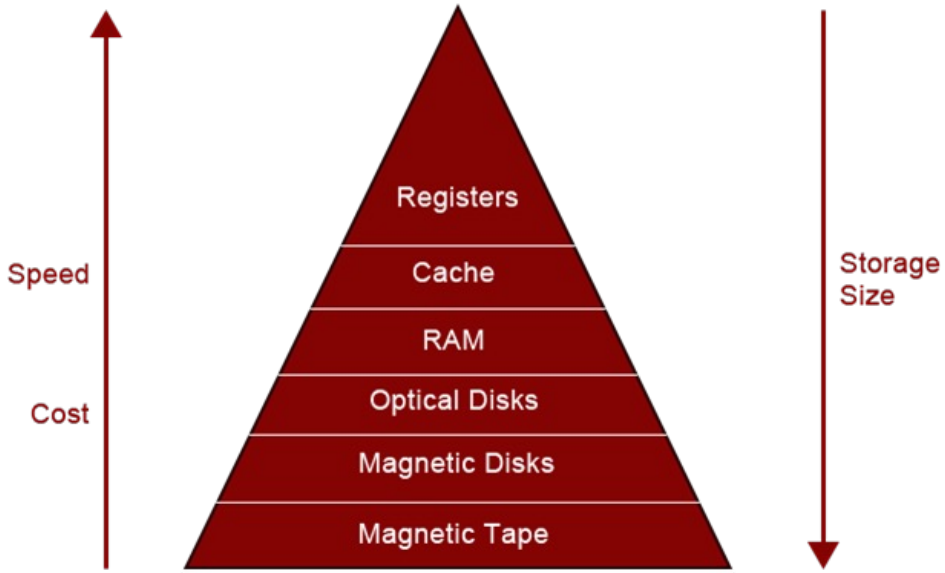
- Spatial locality (locality in space)
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

Memory Hierarchy: Taking Advantage of Locality

- We take advantage of the principle of locality and implement the memory of a computer as a memory hierarchy

- Memory hierarchy: consists of multiple levels of memory with different speeds and sizes

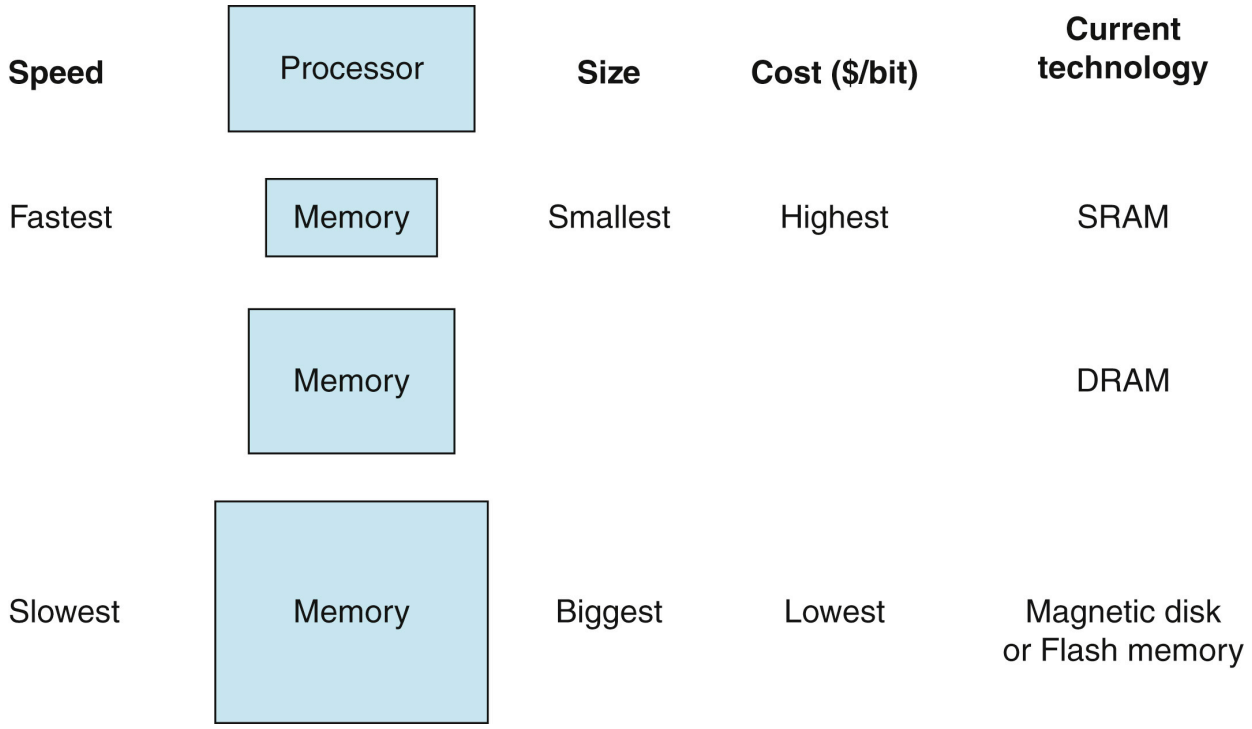
  – The faster memories are more expensive per bit than the slower memories and thus are smaller



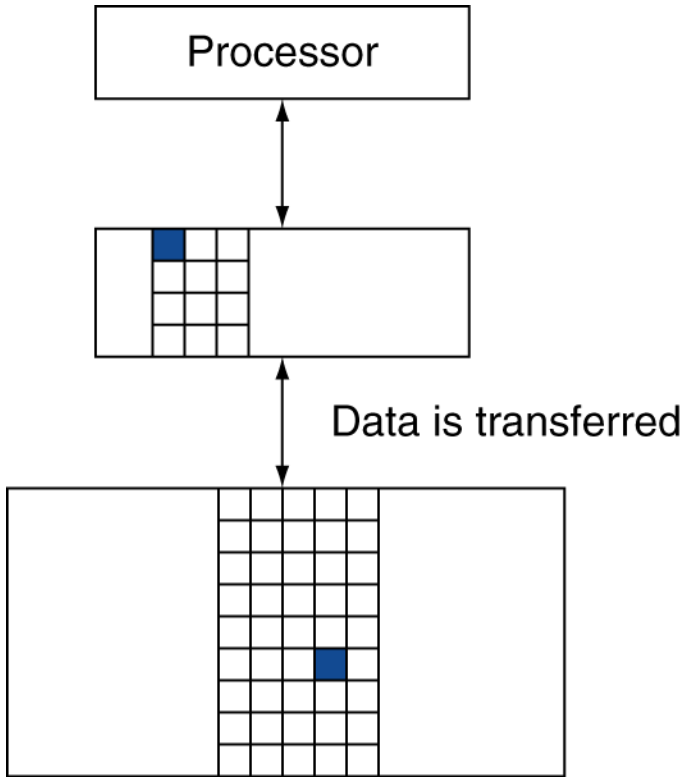
The structure of a memory hierarchy: as the distance from the processor increases, the speed and cost (per unit) decreases and the size increases

Memory Hierarchy Structure

- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU



Memory Hierarchy Levels

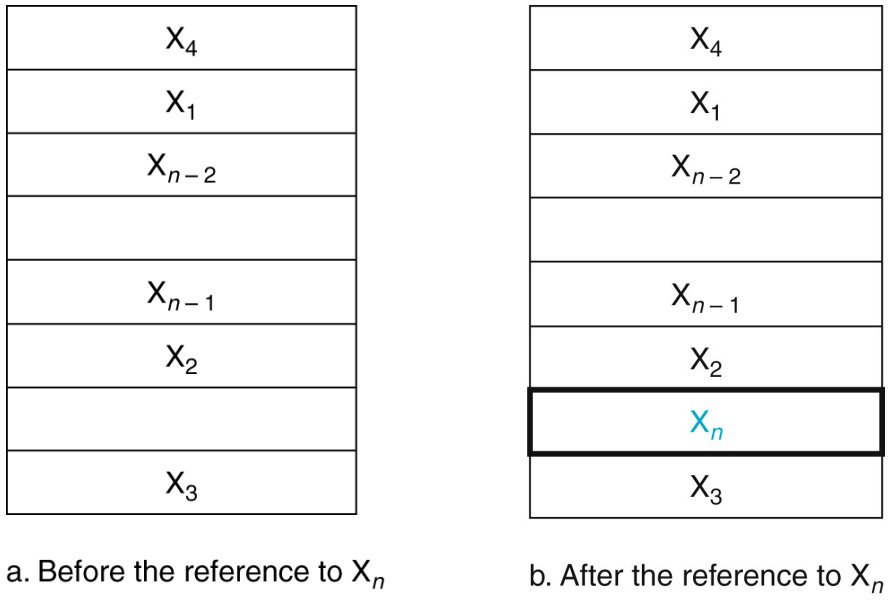


Every pair of levels in the memory hierarchy can be considered having an upper level and a lower level

- Block (aka line): unit of copying
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
    - Time taken: hit time
    - Hit rate/ratio: hits/accesses
- If accessed data is absent
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss rate/ratio: misses/accesses
      = 1 – hit ratio
  - Then accessed data supplied from upper level

Cache Memory

- Cache memory
  - The level of the memory hierarchy closest to the CPU
- Given accesses (or memory references, i.e., instruction fetch/load/store): x_1, …, x_{n-1}, x_n



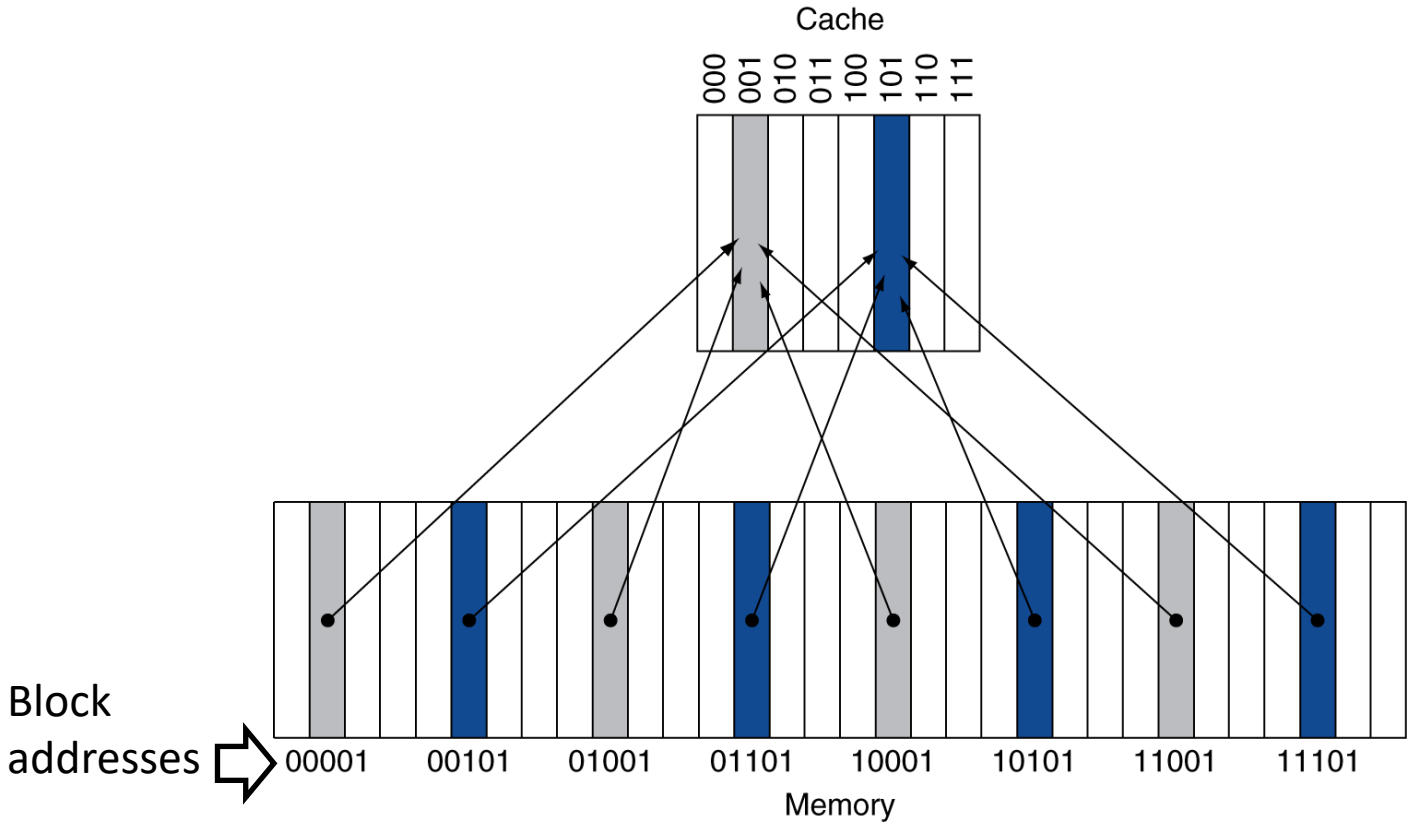
a. Before the reference to X_n

b. After the reference to X_n

- Where do we look?
- How do we know if the data is present?
- Etc.

Direct Mapped Cache

- Location determined by the memory address, more specifically by Block address = memory address (byte address) / block size (i.e., how many bytes in one block)
- Direct mapped: only one choice, use the below mapping to find a block
  (Block address) modulo (#Blocks in cache)
- Granularity/size of a cache block (or cache line)



- #Blocks is a power of 2
- Modulo can be simply calculated by using low-order $\log_2$(cache size in blocks) address bits
- E.g. in this example, cache size in blocks, i.e. #Blocks in cache, is 8. We use low-order 3 bits of block address

Tags and Valid Bits

- How do we know which particular block is stored in a cache location?
  - Store block address as well as the data
  - Actually, only need the high-order bits
  - Called the tag

- What if there is no data in a location?
  - Valid bit: 1 = present, 0 = not present
  - Initially 0

Cache Example

- 8-blocks, cache block size (or cache line size): 1 word, direct mapped
  - Since the block size is same as the word size, therefore, block address == word address
- Assume accessing a sequence of 9 words from memory: **22, 26, 22, 26, 16, 3, 16, 18, 16**
- Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

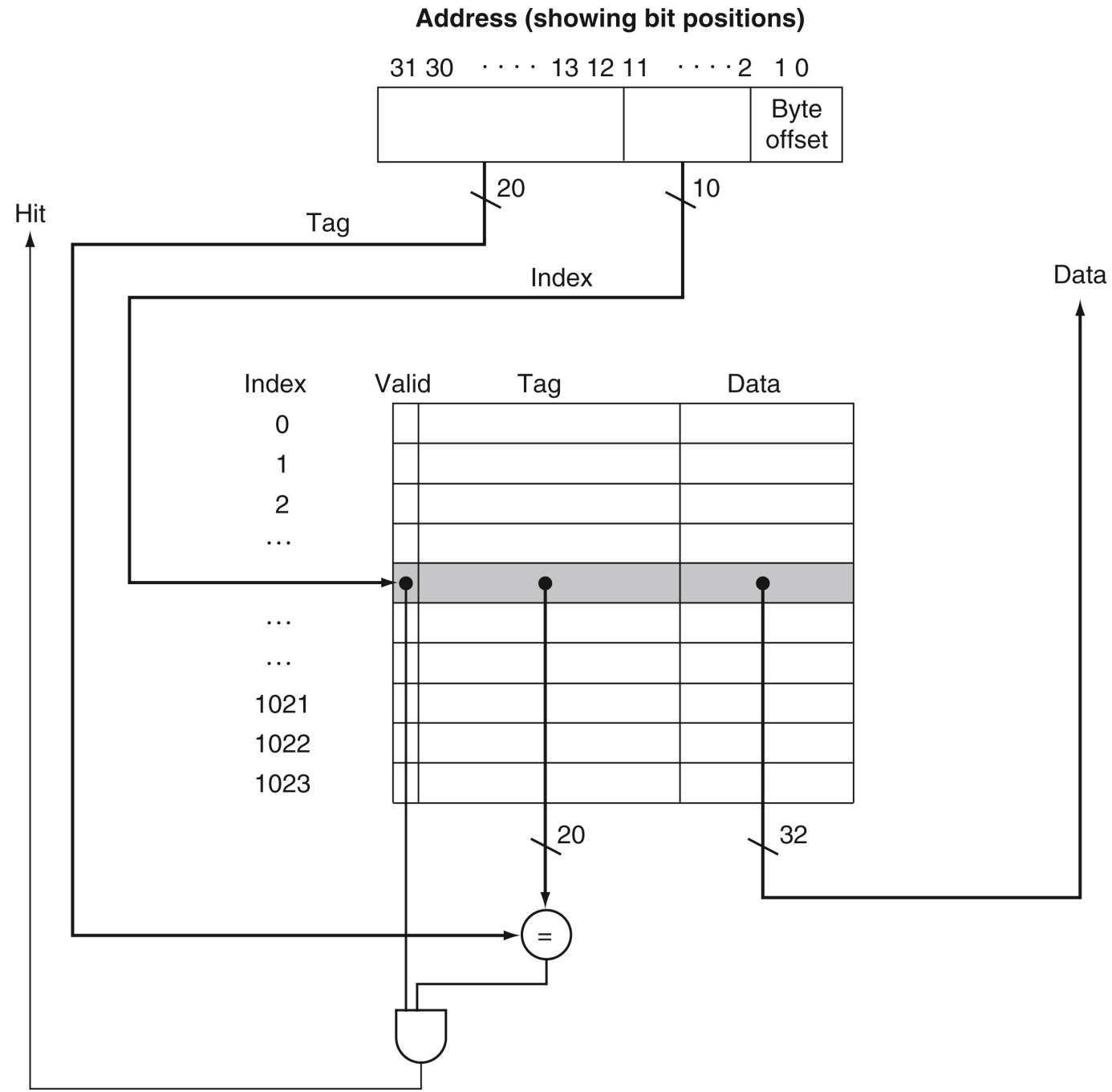
Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

A cache block is being replaced

Example: Accessing a Cache

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_{two}	miss	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss	$(10010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$



Accessing a Cache and Address Subdivision

Accessing a Cache and Address Subdivision

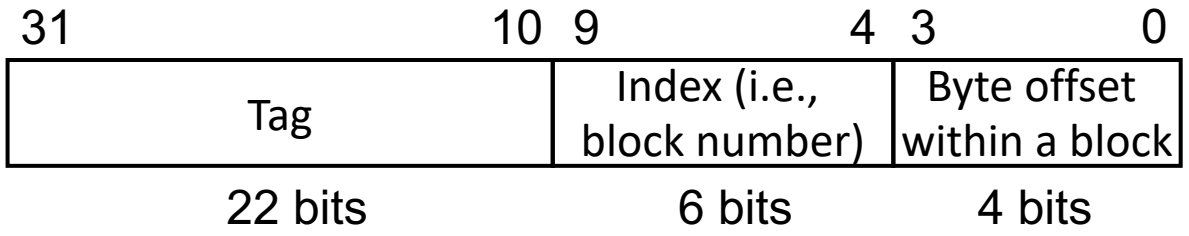
- What's the size of data in this cache?
  - 4 KiB (1K entries of 4B data each), called a 4 KiB cache as a convention
- The total number of bits needed for a cache includes the size of the valid field, the size of the tag field, and the size of the data
- What's the total number of bits needed for this particular direct-mapped cache?
  - Leave it as a homework question

Another Example: Larger Block Size

- Assume a 1KiB cache, 64 blocks, 16 bytes/block (or 4 words/block)
  - To what cache block number does memory address 1200_{ten} map?

- Block address = memory address (byte address) / block size = $\lfloor 1200_{ten}/16_{ten} \rfloor = 75_{ten}$
  - Block address is calculated as floor(byte address/bytes per block)

- Block number = (Block address) modulo (#Blocks in cache) = 75_{ten} modulo $64_{ten} = 11_{ten}$



Readings

- Chapter 1, 1.4-1.9
- Chapter 2, 2.1

- Gene M. Amdahl, “Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities”, 1967

- John L. Gustafson, “Reevaluating Amdahl’s Law”, 1988

- Hill & Marty, “Amdahl’s Law in the Multicore Era”, IEEE Computer 2008

- X.-H. Sun and Y. Chen, "Reevaluating Amdahl's Law in the Multicore Era,” Journal of Parallel and Distributed Computing, vol. 70, no. 2, pp. 183-188, Feb 2010.