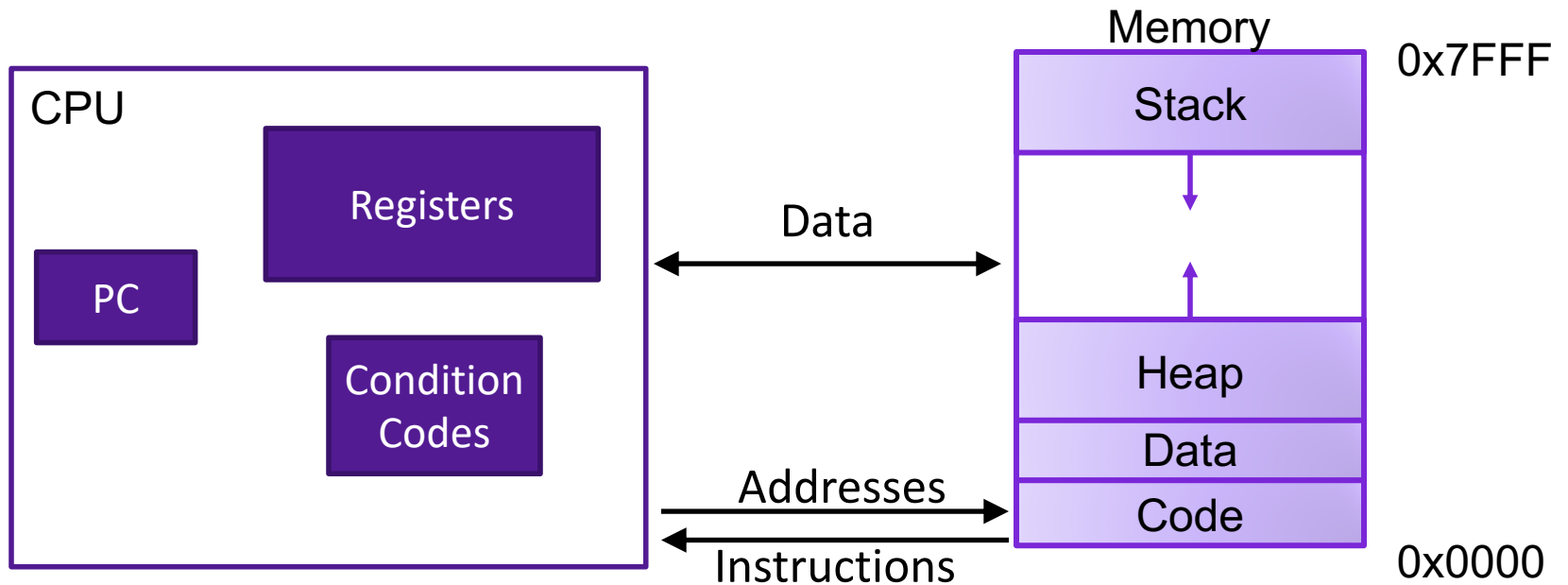


Lecture 11: Caches

CS 105

Review: Assembly/Machine Code View



Programmer-Visible State

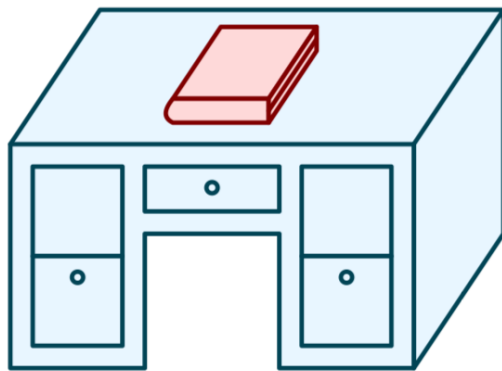
- ▶ PC: Program counter
- ▶ 16 Registers
- ▶ Condition codes

Memory

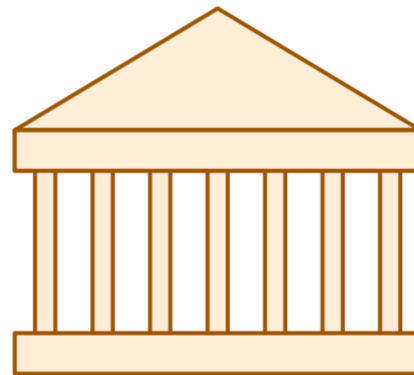
- ▶ Byte addressable array
- ▶ Code and user data
- ▶ Stack to support procedures

Life without caches

- You decide that you want to learn more about computer systems than is covered in this course
- The library contains all the books you could possibly want, but you don't like to study in libraries, you prefer to study at home.
- You have the following constraints:



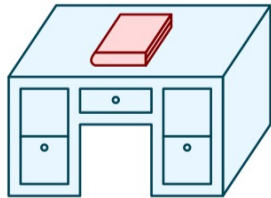
Desk
(can hold one book)



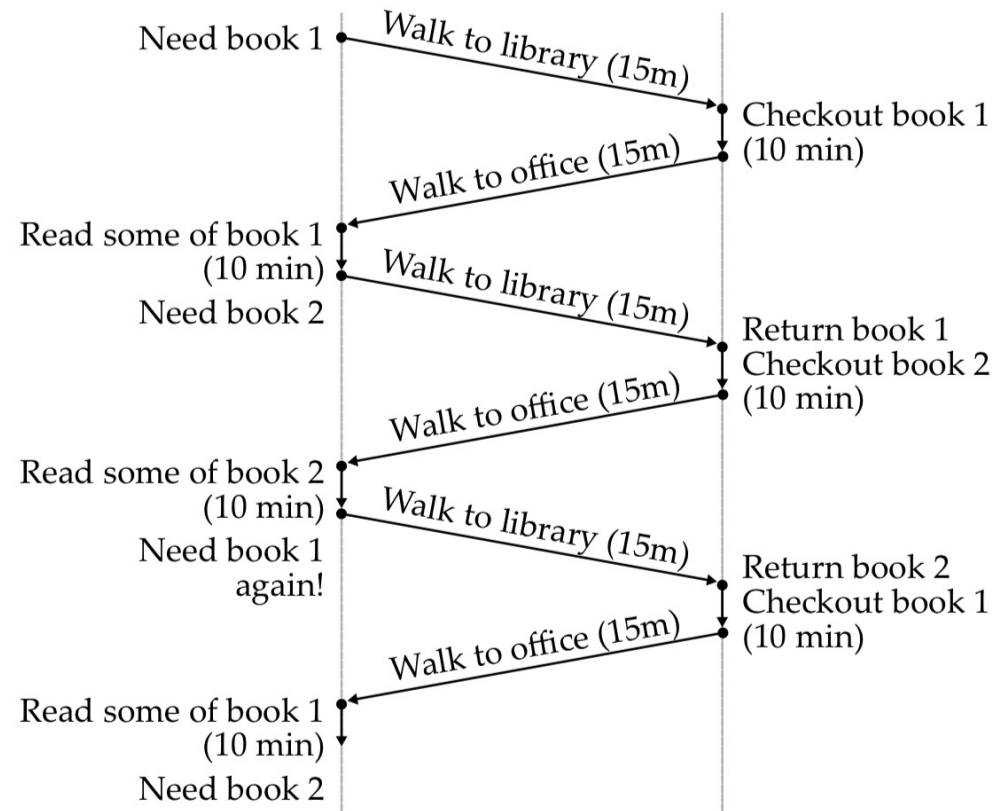
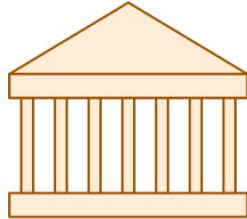
Library
(can hold many books)

Life without caches

Desk
(can hold one book)

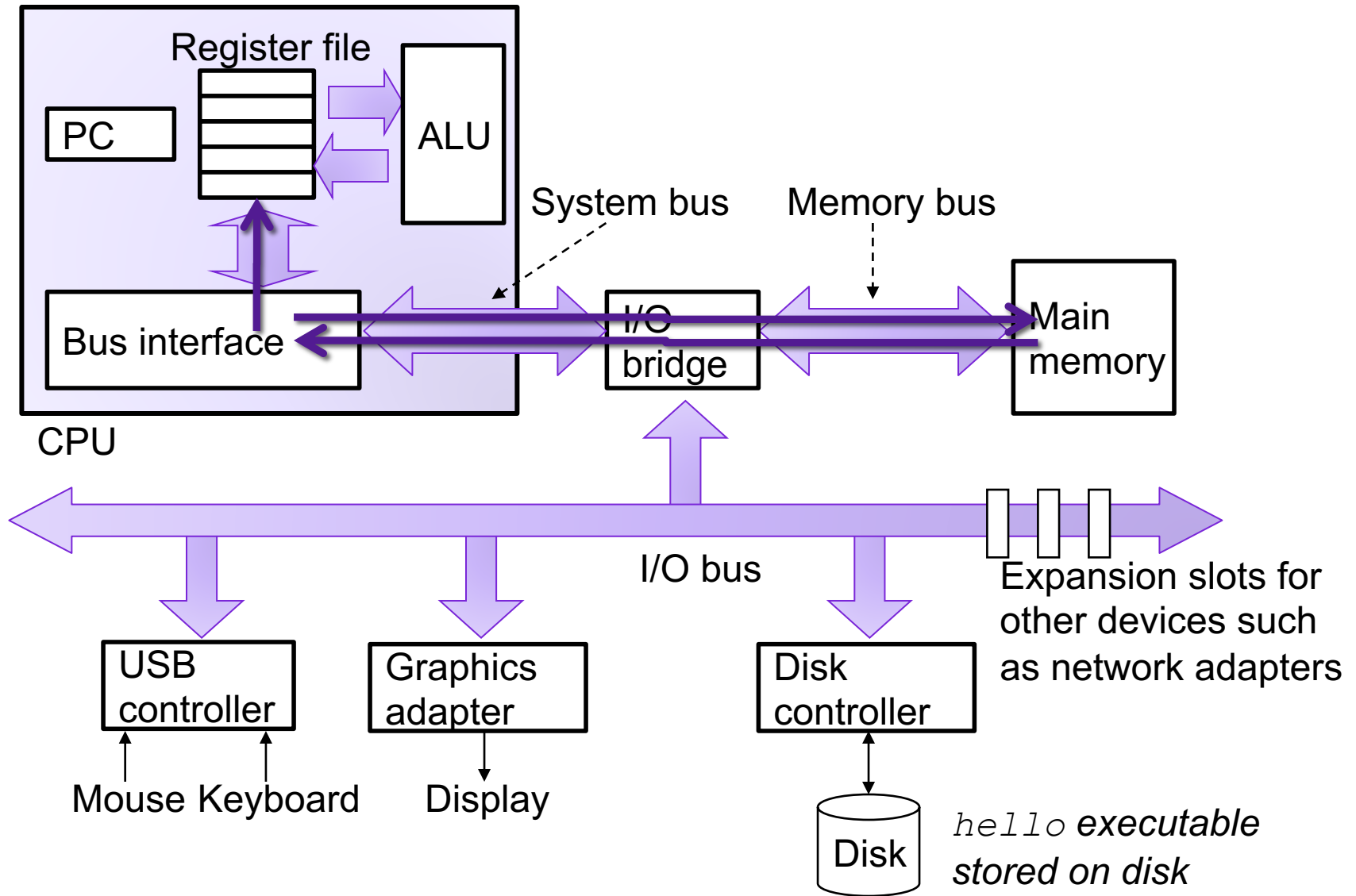


Library
(can hold many books)

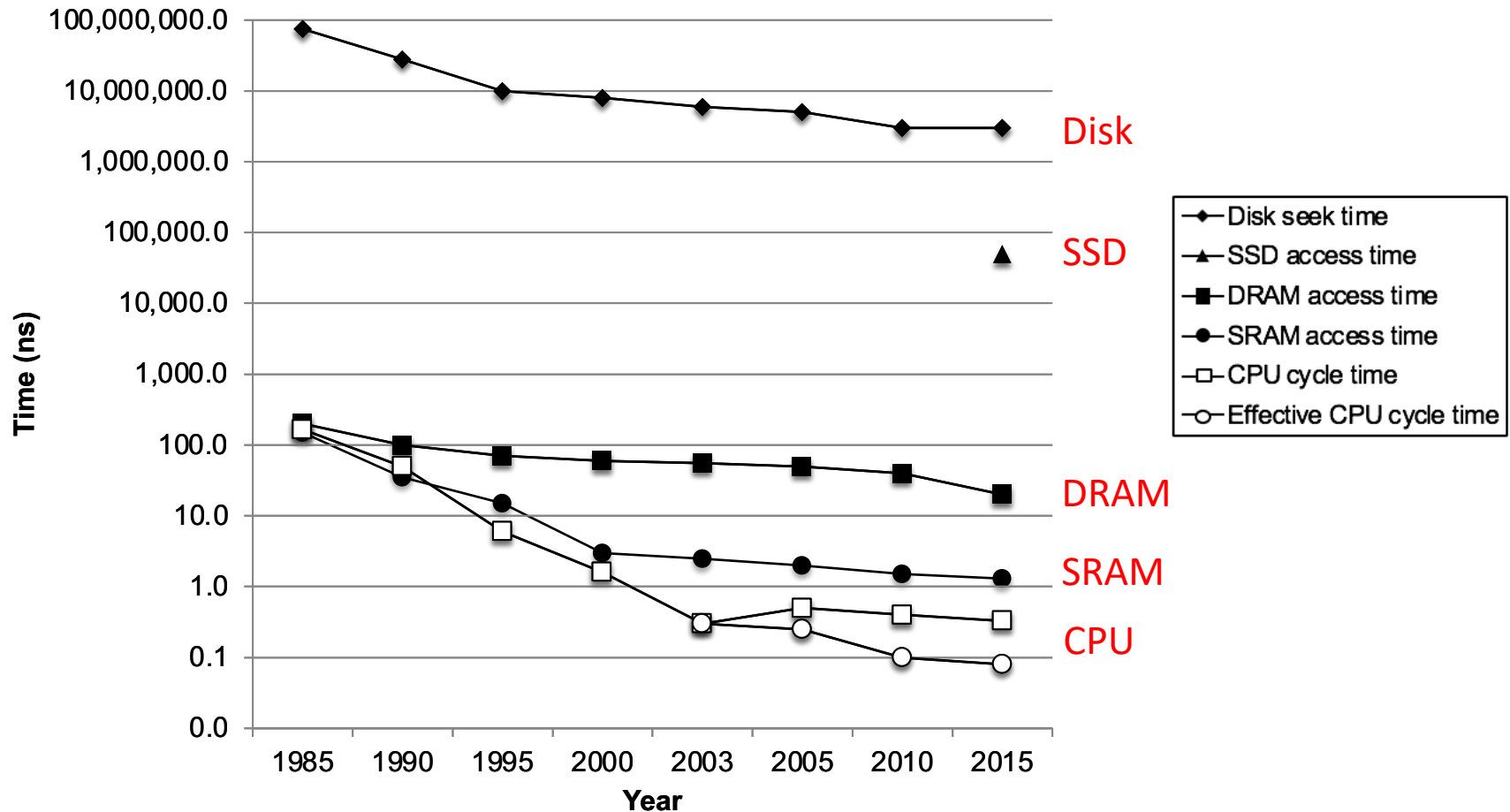


- Average latency to access a book: 40mins
- Average throughput (incl. reading time): 1.2 books/hr

A Computer System



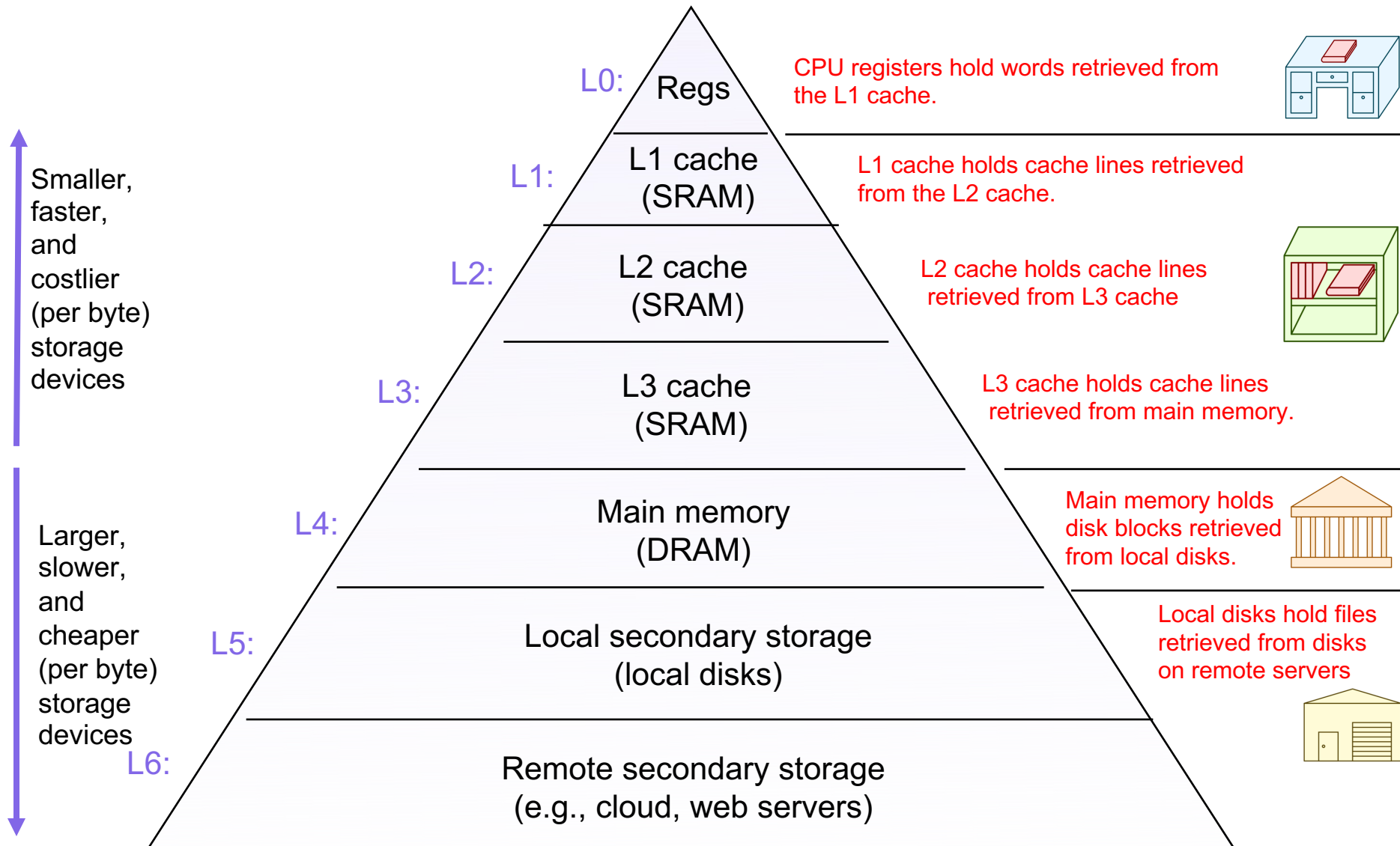
The CPU-Memory Gap



Caching—The Very Idea

- Keep some memory values nearby in fast memory
- Modern systems have 3 or even 4 levels of caches
- Cache idea is widely used:
 - Disk controllers
 - Web
 - (Virtual memory: main memory is a “cache” for the disk)

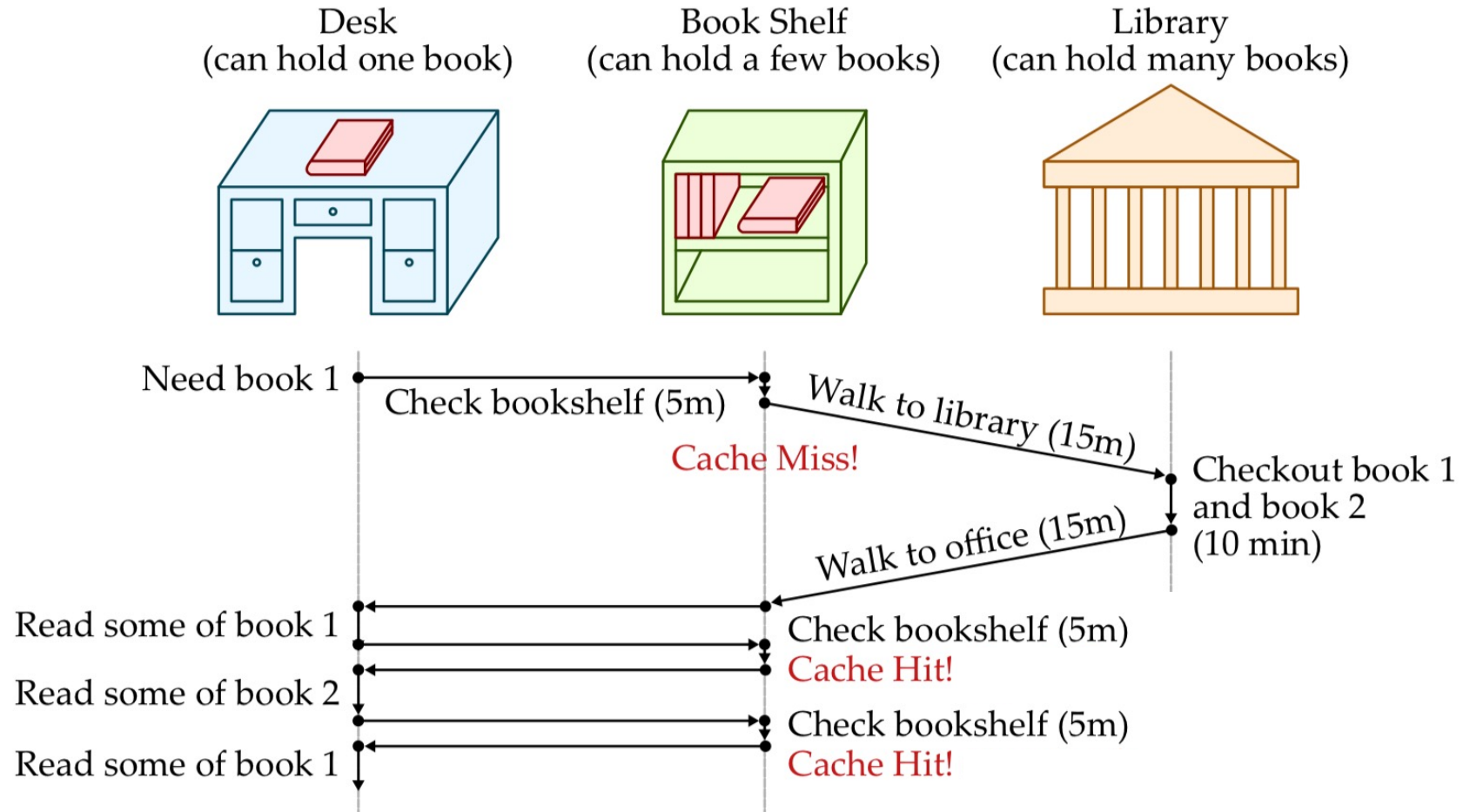
Memory Hierarchy



Latency numbers (2020)

Register access	0.3 ns	
L1 cache reference	1 ns	
L2 cache reference	4 ns	
Main memory reference	100 ns	
memory 1MB sequential read	3,000 ns	3 μ s
SSD random read	16,000 ns	16 μ s
SSD 1MB sequential read	49,000 ns	49 μ s
Magnetic Disk seek	2,000,000 ns	2 ms
Magnetic Disk 1MB sequential read	825,000 ns	825 μ s
Round trip in Datacenter	500,000 ns	500 μ s
Round trip CA<->Europe	150,000,000 ns	150 ms

Life with caching



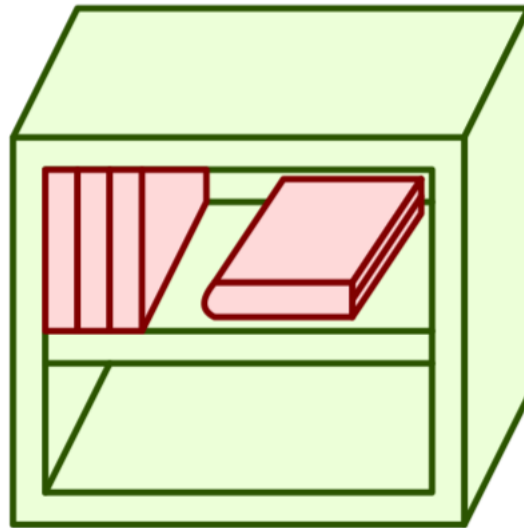
- Average latency to access a book: <20mins
- Average throughput (incl. reading time): ~2 books/hr

Caching—The Vocabulary

- **Size:** the total number of bytes that can be stored in the cache
- **Cache Hit:** the desired value is in the cache and returned quickly
- **Cache Miss:** the desired value is not in the cache and must be fetched from a more distant cache (or ultimately from main memory)

Exercise 1: Caching Strategies

How should we decide which books to keep in the bookshelf?

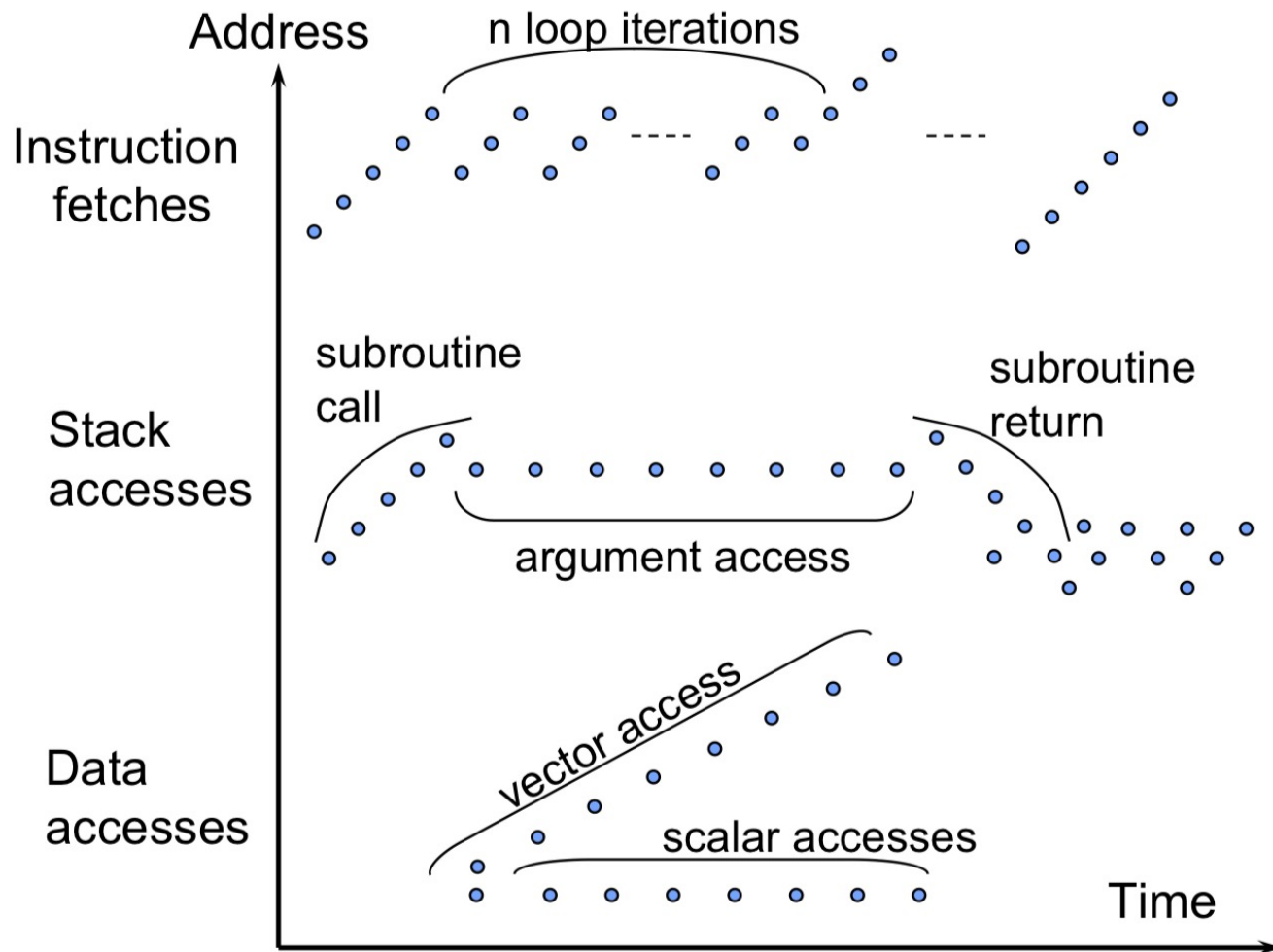


Example Access Patterns

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

- Data references
 - Reference array elements in succession.
 - Reference variable **sum** each iteration.
- Instruction references
 - Reference instructions in sequence.
 - Cycle through loop repeatedly.

Example Access Patterns

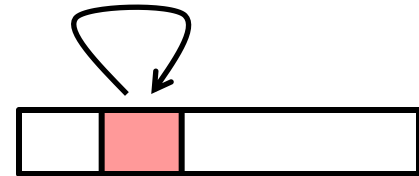


Principle of Locality

Programs tend to use data and instructions with addresses near or equal to those they have used recently

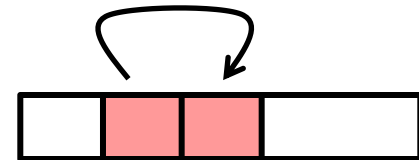
- ▶ **Temporal locality:**

- ▶ Recently referenced items are likely to be referenced again in the near future



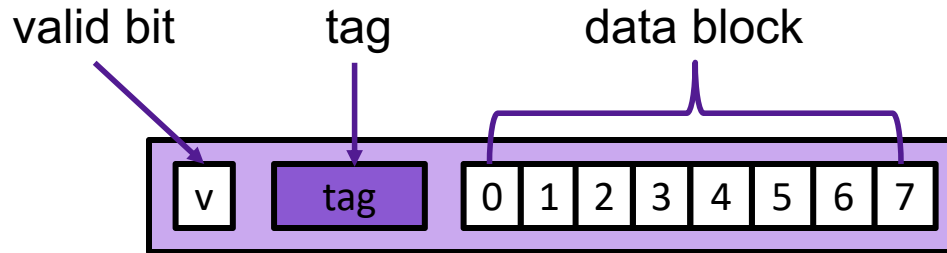
- ▶ **Spatial locality:**

- ▶ Items with nearby addresses tend to be referenced close together in time



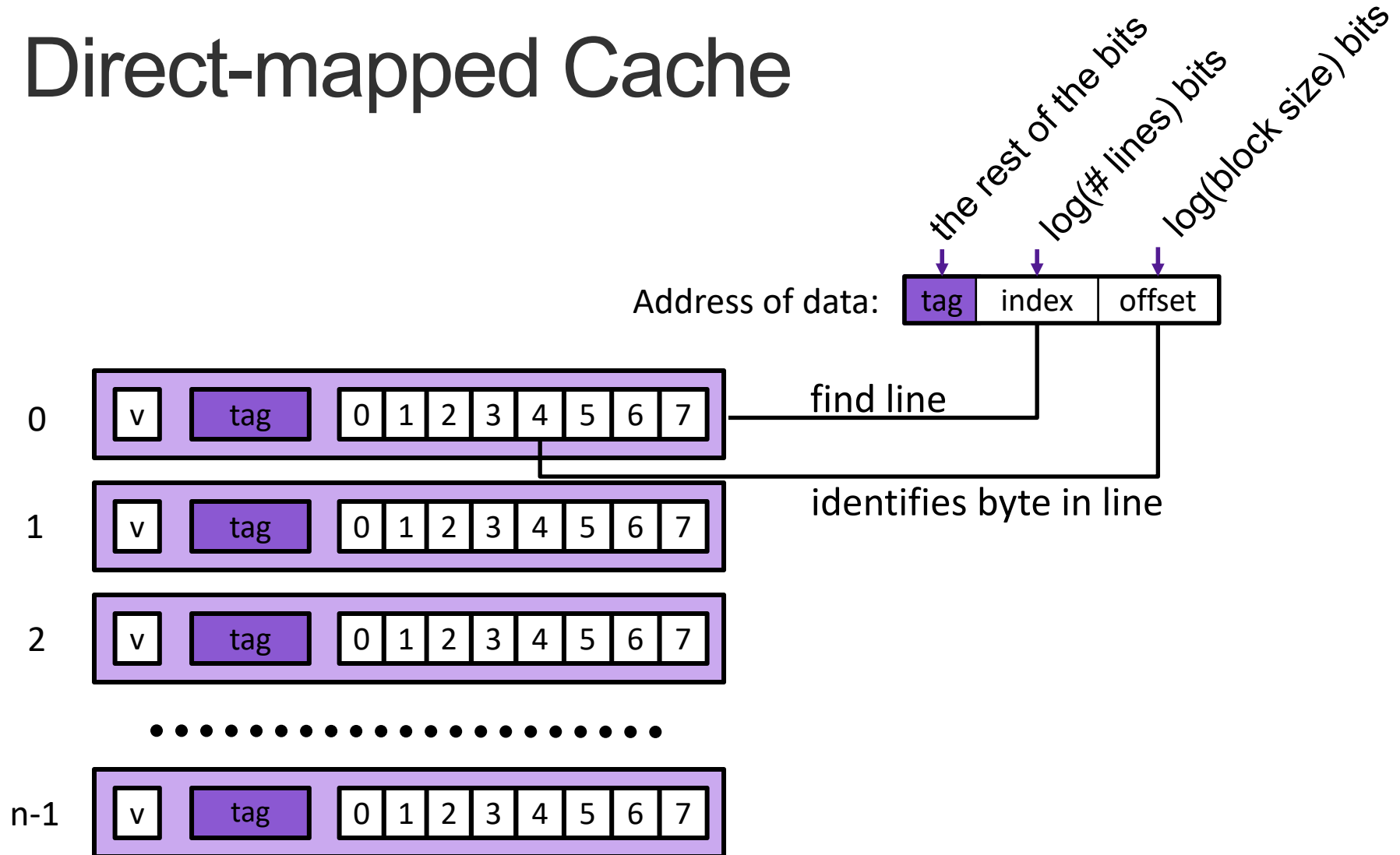
CACHE ORGANIZATION

Cache Lines



- **data block:** cached data (i.e., copy of bytes from memory)
- **tag:** uniquely identifies which data is stored in the cache line
- **valid bit:** indicates whether or not the line contains meaningful information

Direct-mapped Cache



Example: Direct-mapped Cache

Assume: cache block size 8 bytes, total cache size 32 bytes

Assume: assume 8-bit machine

Address of data:

0xB4

Line 0	1	110	0F	12	AB	34	FF	FF	EA	68
Line 1	1	001	00	00	00	00	00	40	06	1D
Line 2	1	101	0D	00	00	00	2F	00	00	00
Line 3	0	001	00	11	22	33	44	55	66	77

1011 0100

101 10 100

3 bit tag

2 bit index

3 bit offset

Exercise 2: Interpreting Addresses

Consider the hex address 0xA59. What would be the tag, index, and offset for this address with each of the following cache configurations?

1. A direct-mapped cache with 8 cache lines and 8-byte data blocks
2. A direct-mapped cache with 16 cache lines and 4-byte data blocks
3. A direct-mapped cache with 16 cache lines and 8-byte data blocks

Exercise 3: Cache Indices

- Assume you have an array of 6 integers `a` that begins at address `0x601940`. Assume you are running on a machine that has a direct-mapped cache with 8 cache lines and 8-byte data blocks. Which cache line would each of the 6 integers be stored in when it is in cache?

`0x601940`



Exercise 3: Cache Indices

- Assume you have an array of 6 integers `a` that begins at address `0x601940`. Assume you are running on a machine that has a direct-mapped cache with 8 cache lines and 8-byte data blocks. Which cache line would each of the 6 integers be stored in when it is in cache?

`0x601940`



Element	Address	Binary Address	Index	Offset
<code>a[0]</code>				
<code>a[1]</code>				
<code>a[2]</code>				
<code>a[2]</code>				
<code>a[4]</code>				
<code>a[5]</code>				

Exercise 4: Direct-mapped Cache

Memory

0x14	18
0x10	17
0x0c	16
0x08	15
0x04	14
0x00	13

Cache

	Valid	Tag	Data Block
Line 0	<input type="checkbox"/>		
Line 1	<input type="checkbox"/>		
Line 2	<input type="checkbox"/>		
Line 3	<input type="checkbox"/>		

Assume 4 byte data blocks

Access	tag	idx	off	h/m
rd 0x00	0000	00	00	m
rd 0x04				
rd 0x14				
rd 0x00				
rd 0x04				
rd 0x14				

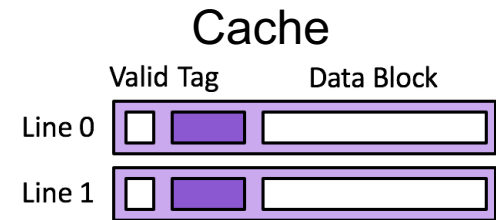
Line 0			Line 1			Line 2			Line 3		
0	0000	47	0	0000	47	0	0000	47	0	0000	47
1	0000	13									

How well does this take advantage of spacial locality?
 How well does this take advantage of temporal locality?

Exercise 5: Direct-mapped Cache

Memory

0x14	18
0x10	17
0x0c	16
0x08	15
0x04	14
0x00	13



Access	tag	idx	off	h/m
rd 0x00				
rd 0x04				
rd 0x14				
rd 0x00				
rd 0x04				
rd 0x14				

Assume 8 byte data blocks

Line 0				Line 1			
0	0000	47	48	0	0000	47	48

How well does this take advantage of spacial locality?
 How well does this take advantage of temporal locality?