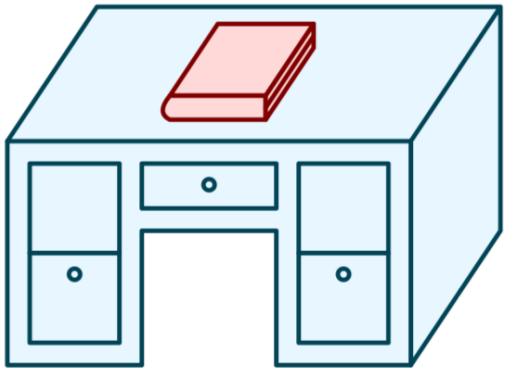
Lecture 14: Caches

CS 105

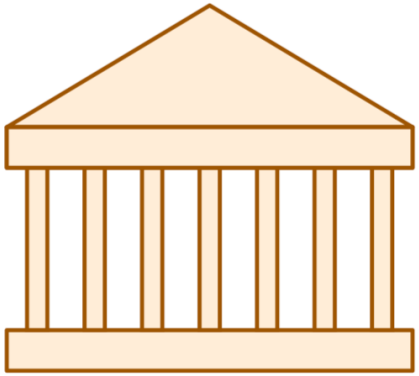
Spring 2021

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:

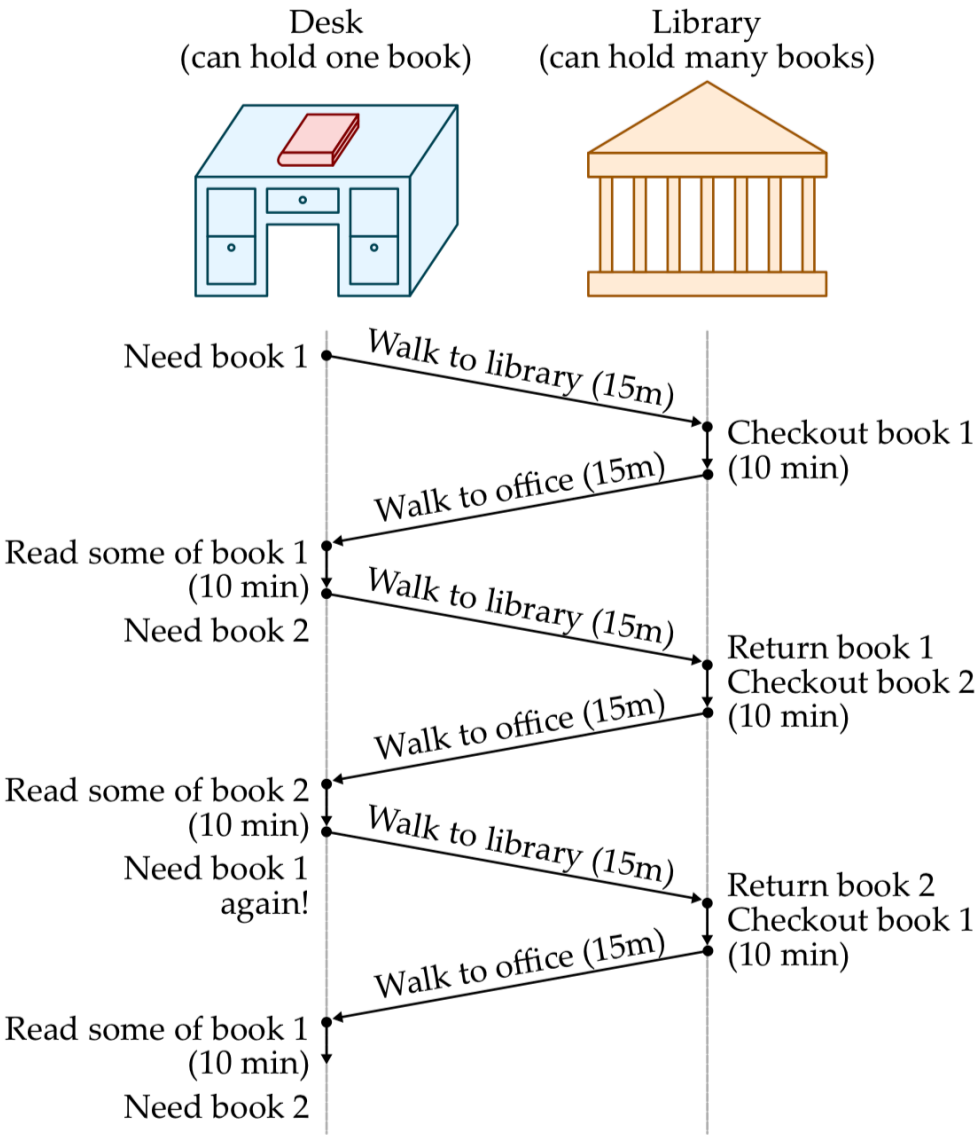


Desk
(can hold one book)



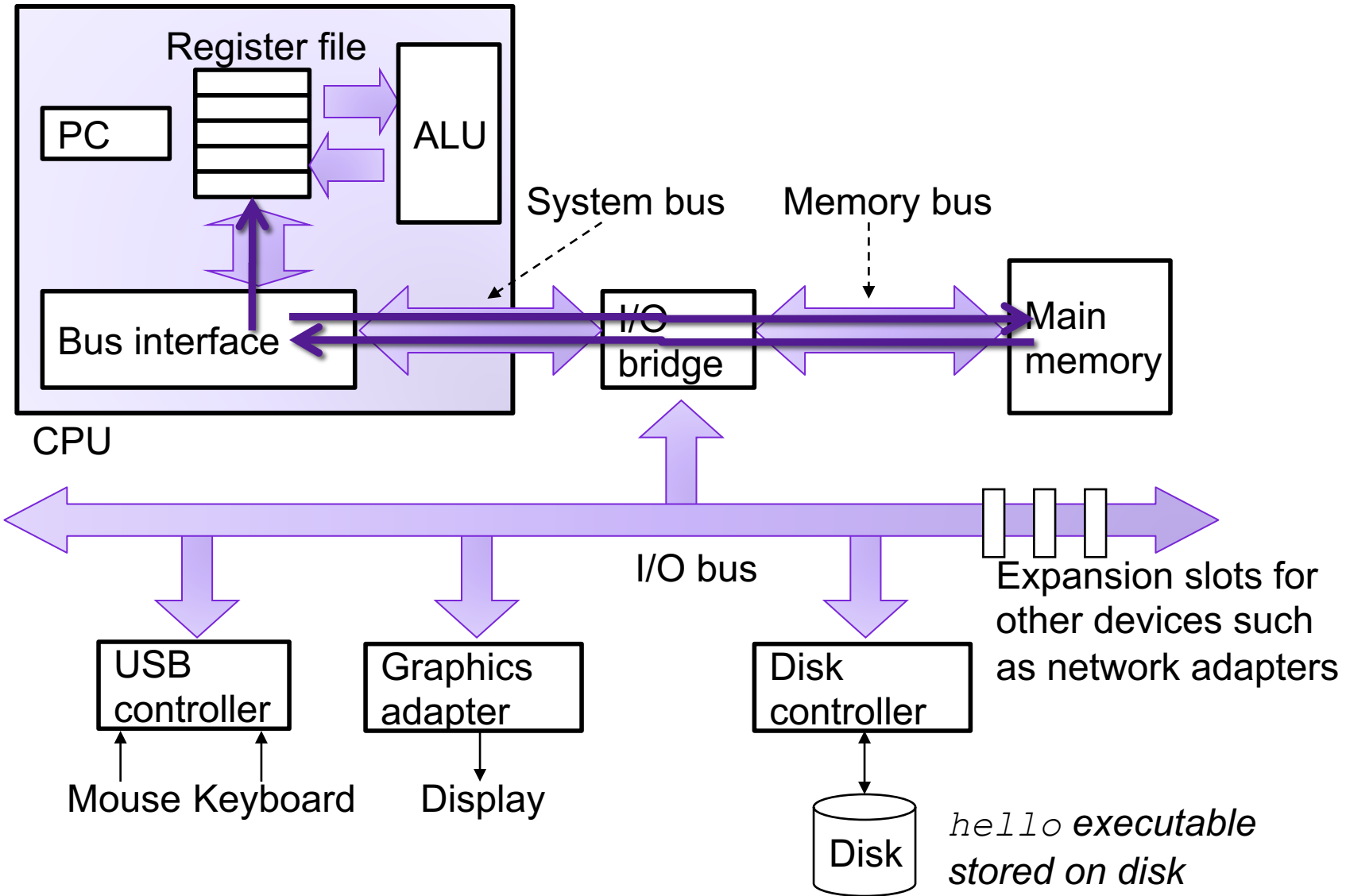
Library
(can hold many books)

Life without caches

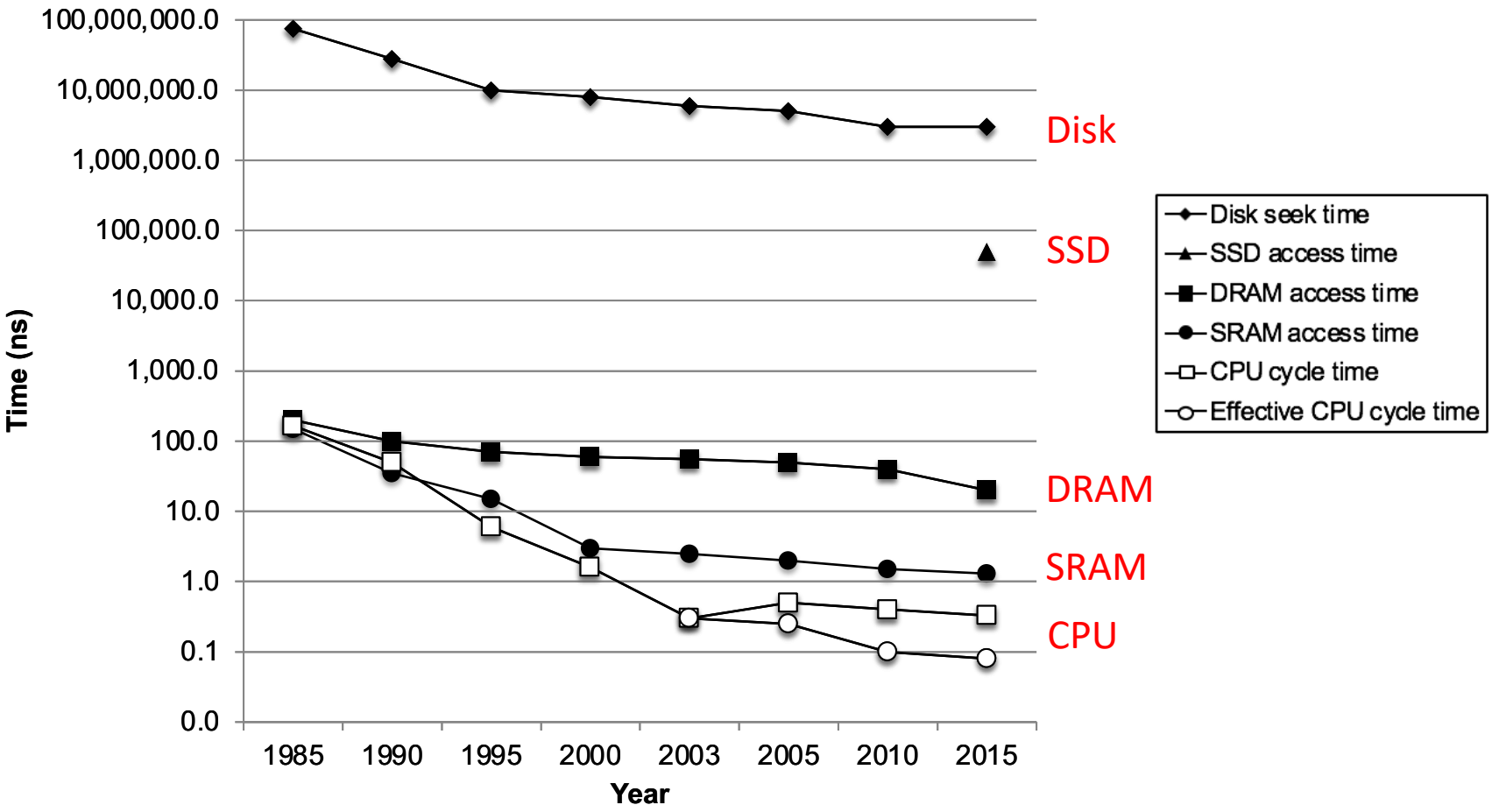


- Average latency to access a book: 40mins
- Average throughput (incl. reading time): 1.2 books/hr

A Computer System



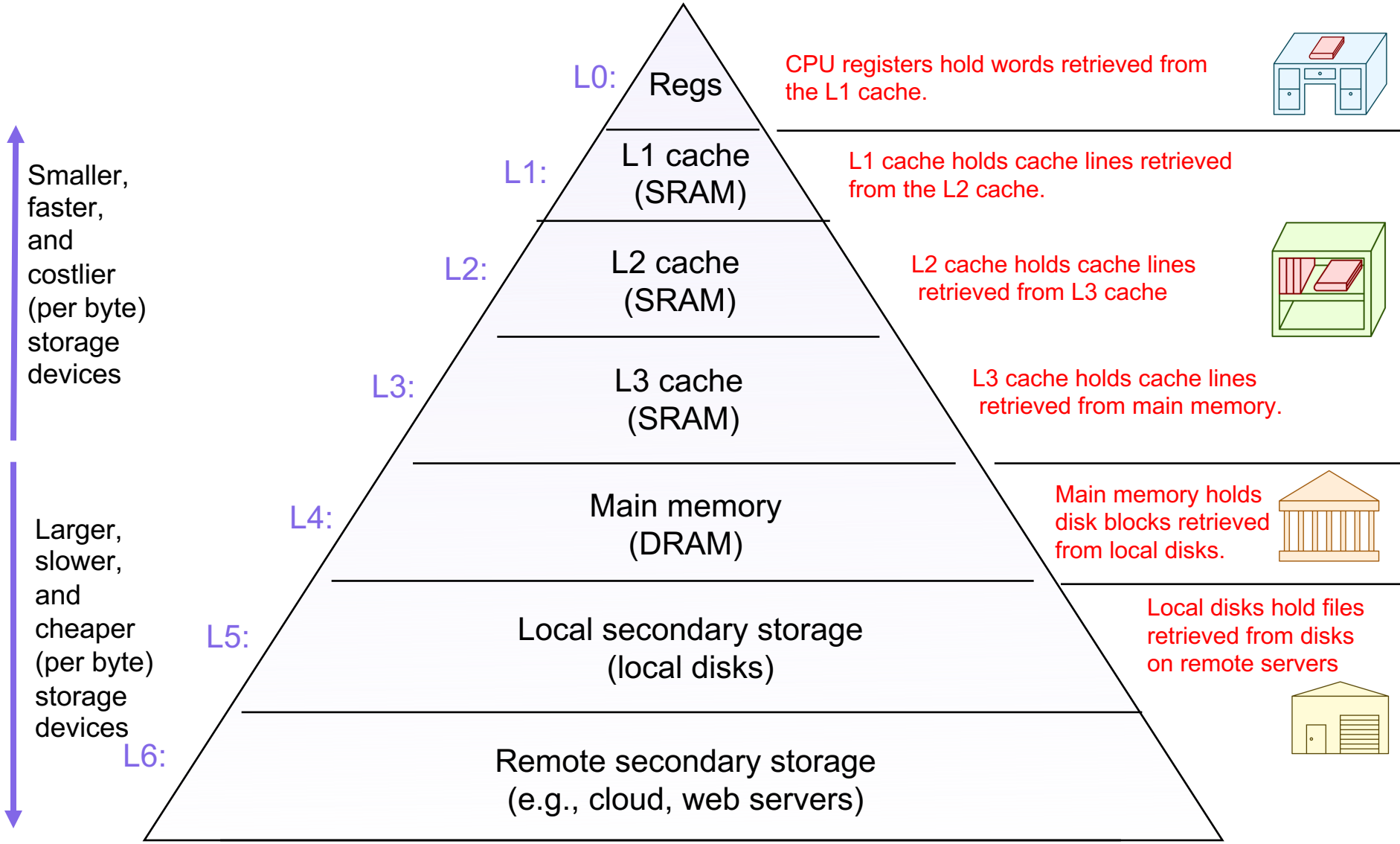
The CPU-Memory Gap



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)

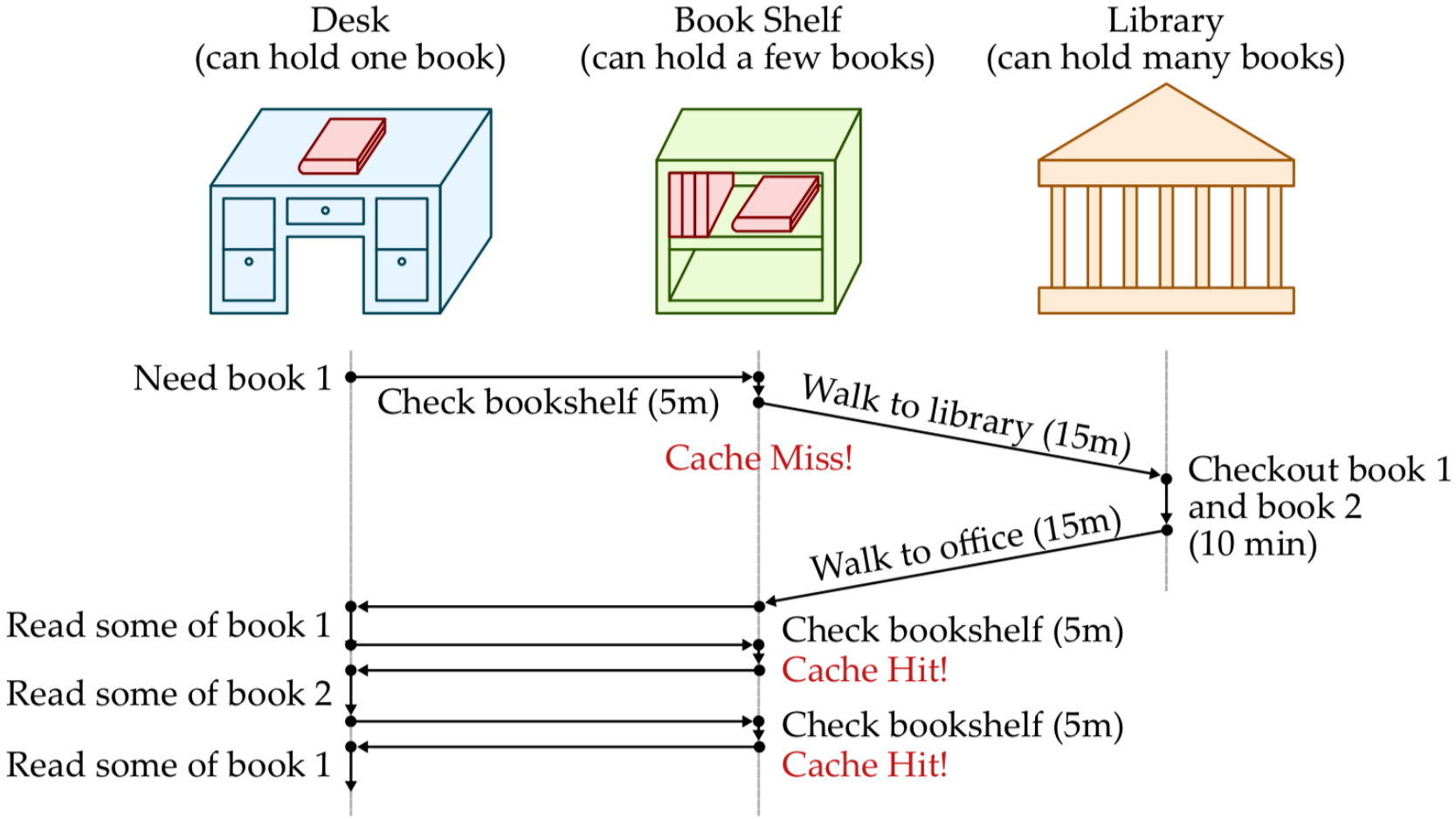
Memory Hierarchy



Latency numbers every programmer should know (2020)

L1 cache reference	1 ns	
Branch mispredict	3 ns	
L2 cache reference	4 ns	
Main memory reference	100 ns	
memory 1MB sequential read	3,000 ns	3 μs
SSD random read	16,000 ns	16 μs
SSD 1MB sequential read	49,000 ns	49 μs
Magnetic Disk seek	2,000,000 ns	2 ms
Magnetic Disk 1MB sequential read	825,000 ns	825 μs
Round trip in Datacenter	500,000 ns	500 μs
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

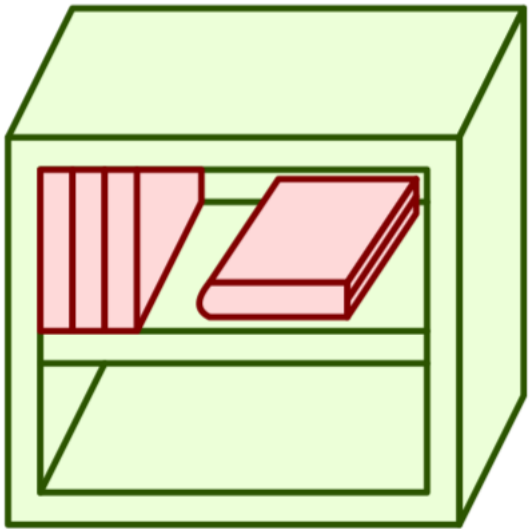
Caching—The Vocabulary

- **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?

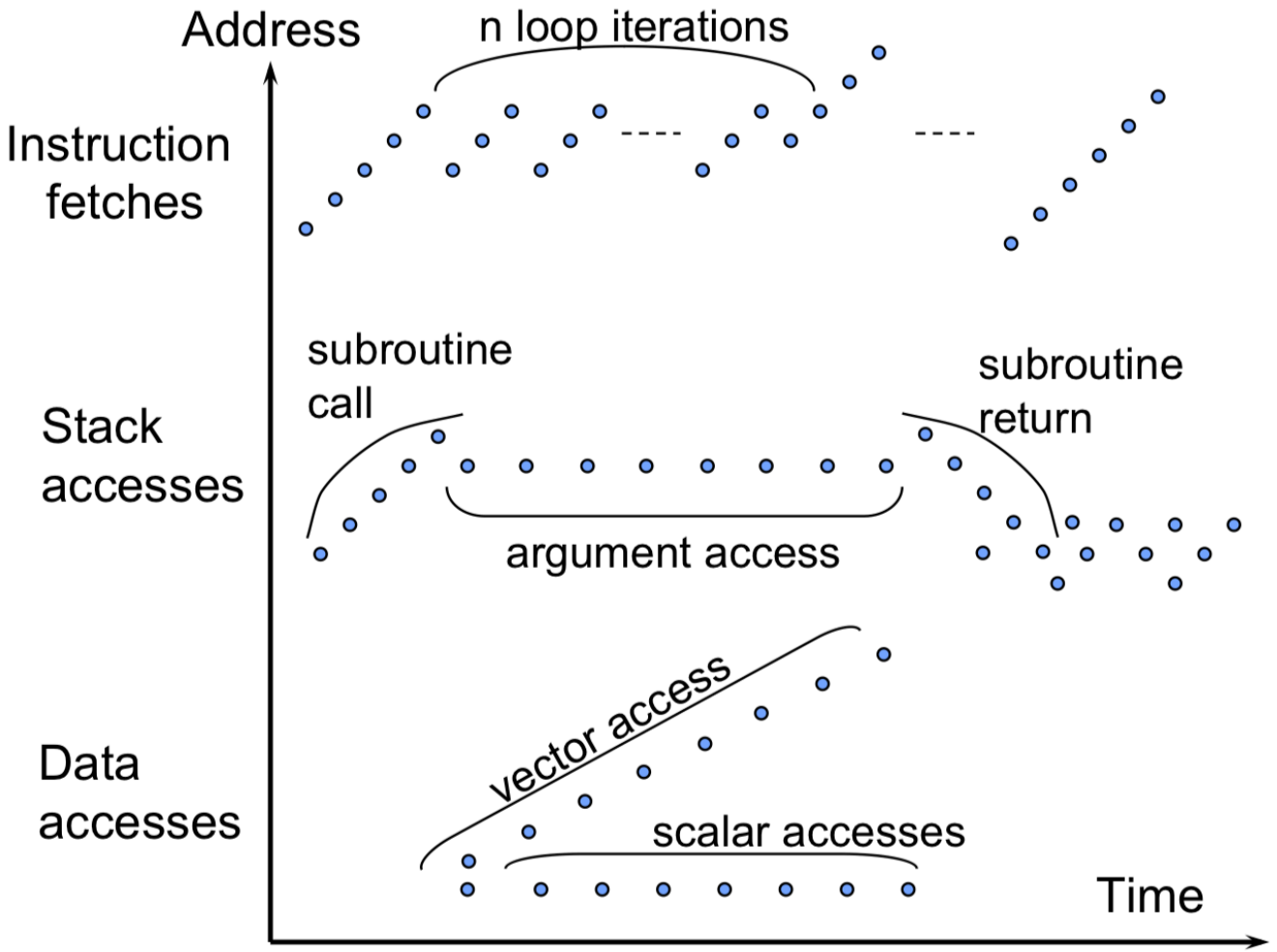


Example Access Patterns

```
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.

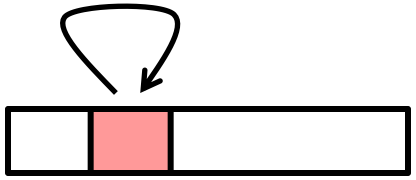
Example Access Patterns



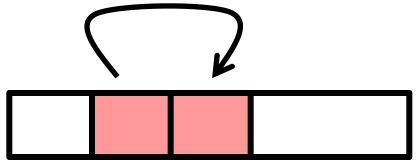
Principle of Locality

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

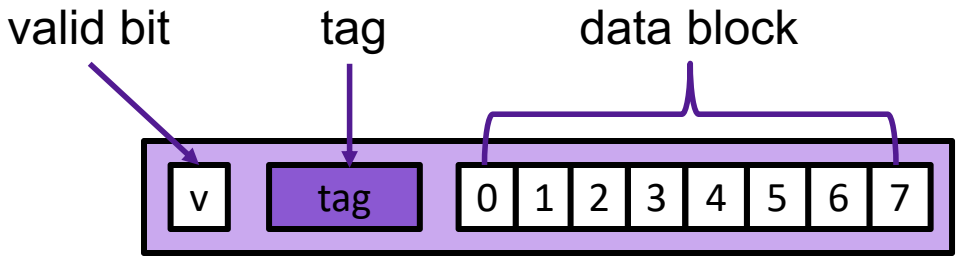


- Spatial locality:
  - Items with nearby addresses tend to be referenced close together in time



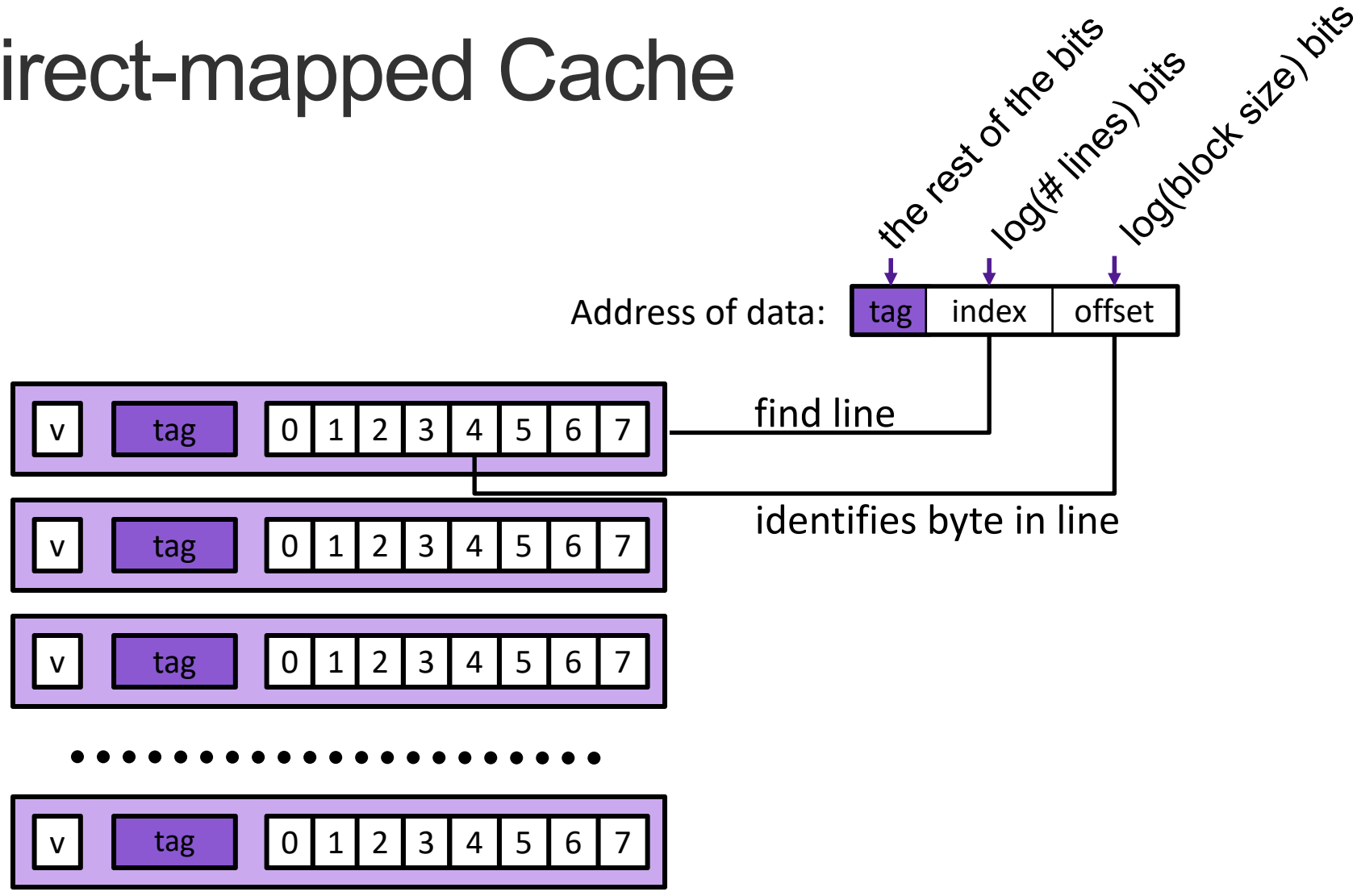
CACHE ORGANIZATION

Cache Lines



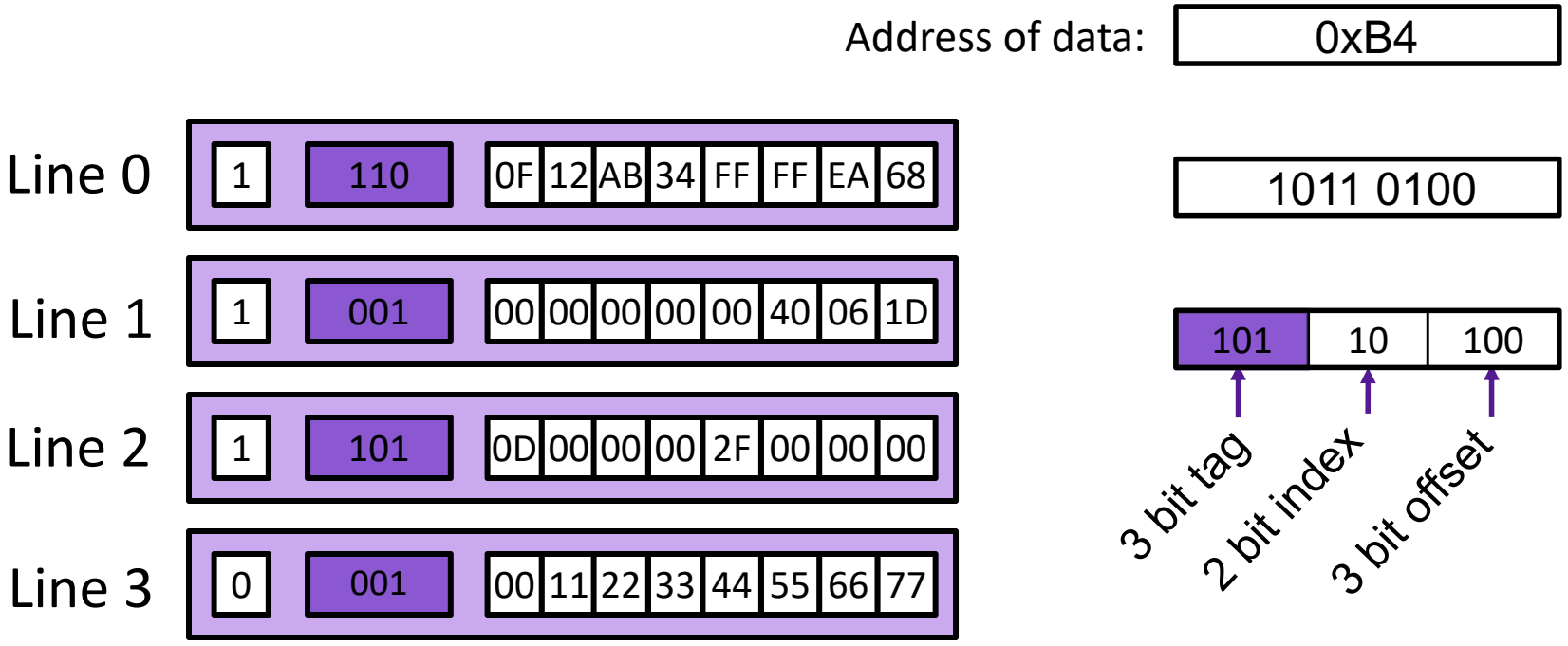
- **data block:** cached data (i.e., copy of bytes from memory)
- **tag:** uniquely identifies which data is stored in the cache line
- **valid bit:** indicates whether or not the line contains meaningful information

Direct-mapped Cache



Example: Direct-mapped Cache

Assume: cache block size 8 bytes
Assume: assume 8-bit machine



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

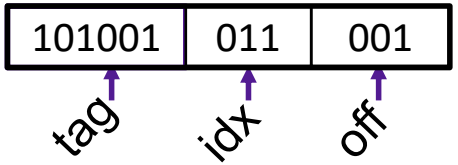
3. A direct-mapped cache with 16 cache lines and 8-byte data blocks

Exercise 2: Interpreting Addresses

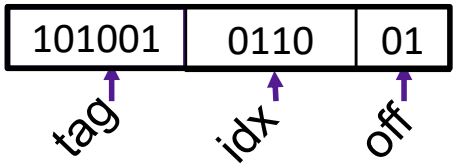
Consider the hex address 0xA59. What would be the tag, index, and offset for this address with each of the following cache configurations?

1010 0101 1001

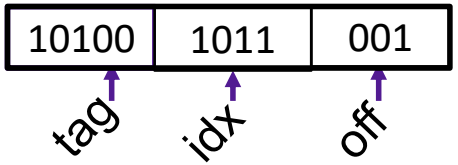
1. A direct-mapped cache with 8 cache lines and 8-byte data blocks



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

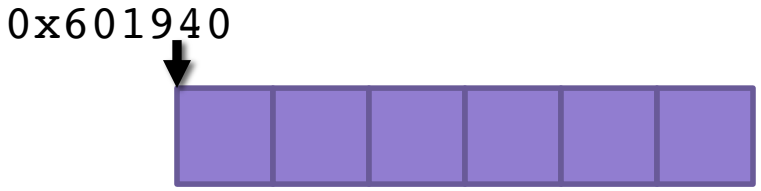


3. A direct-mapped cache with 16 cache lines and 8-byte data blocks



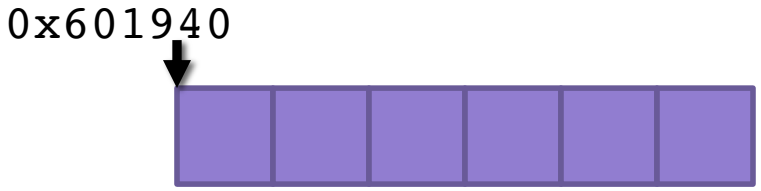
Exercise 3: Cache Indices

- 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?



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?



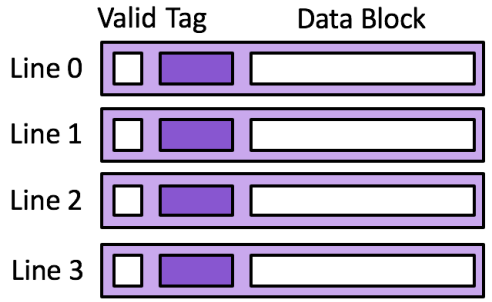
Element	Address	Binary Address	Index	Offset
a[0]	0x601940	… 0100 0000	000	000
a[1]	0x601944	… 0100 0100	000	100
a[2]	0x601948	… 0100 1000	001	000
a[2]	0x60194c	… 0100 1100	001	100
a[4]	0x601950	… 0101 0000	010	000
a[5]	0x601954	… 0101 0100	010	100

Exercise 3: Direct-mapped Cache

Memory

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

Cache



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				

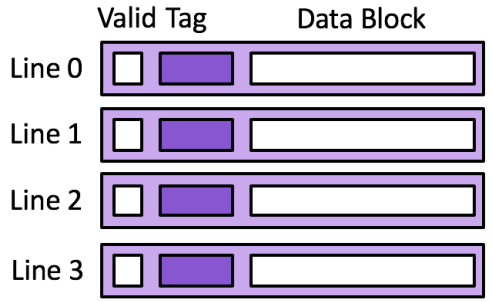
Line 0			Line 1			Line 2			Line 3		
0	0000	47	0	0000	47	0	0000	47	0	0000	47
1	0000	13									

Exercise 3: Direct-mapped Cache

Memory

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

Cache



Assume 4 byte data blocks

Access	tag	idx	off	h/m
rd 0x00	0000	00	00	m
rd 0x04	0000	01	00	m
rd 0x14	0001	01	00	m
rd 0x00	0000	00	00	h
rd 0x04	0000	01	00	m
rd 0x14	0001	01	00	m

Line 0			Line 1			Line 2			Line 3		
0	0000	47	0	0000	47	0	0000	47	0	0000	47
1	0000	13									
			1	0000	14						
			1	0001	18						
			1	0000	14						
			1	0001	18						

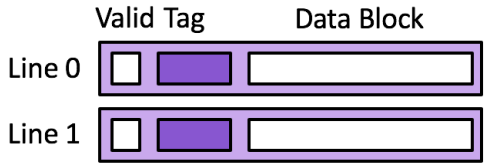
How well does this take advantage of spacial locality?
How well does this take advantage of temporal locality?

Exercise 4: Direct-mapped Cache

Memory

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

Cache



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

Assume 8 byte data blocks

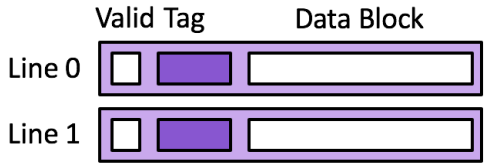
Line 0				Line 1			
0	0000	47	48	0	0000	47	48

Exercise 4: Direct-mapped Cache

Memory

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

Cache



Access	tag	idx	off	h/m
rd 0x00	0000	0	000	m
rd 0x04	0000	0	100	h
rd 0x14	0001	0	100	m
rd 0x00	0000	0	000	m
rd 0x04	0000	0	100	h
rd 0x14	0001	0	100	m

Assume 8 byte data blocks

Line 0				Line 1			
0	0000	47	48	0	0000	47	48
1	0000	13	14				
1	0001	17	18				
1	0000	13	14				
1	0001	17	18				

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

Exercise 5: Feedback

1. Rate how well you think this recorded lecture worked
    1. Better than an in-person class
    2. About as well as an in-person class
    3. Less well than an in-person class, but you still learned something
    4. Total waste of time, you didn't learn anything
2. How much time did you spend on this video lecture (including time spent on exercises)?
3. Do you have any questions that you would like me to address in this week's problem session?
4. Do you have any other comments or feedback?