



# COS 318: Operating Systems

## Review Session 2



# Outline

---

- ◆ Practice six more sample questions



# Sample Question 9

Attribution: from U Waterloo CS 350 Study Questions

A virtual memory system uses both paging and segmentation. Process virtual addresses consist of a total of 32 bits. The page size is 1024 ( $2^{10}$ ) bytes. A page table may occupy at most one frame, and the size of a page table entry is 4 bytes. A single process may have up to 64 ( $2^6$ ) segments.

- How many levels of page tables are required for virtual address translation in this system?
- What is the maximum size of a virtual memory segment?
- Draw an address translation diagram that shows how a virtual address is translated in this system. Your diagram must show each component of the virtual and physical addresses, and the size of each component. It must show any tables used in translation, and their lengths. It must show how the components of the virtual address are used to index the tables, and how the components of the physical address are determined.

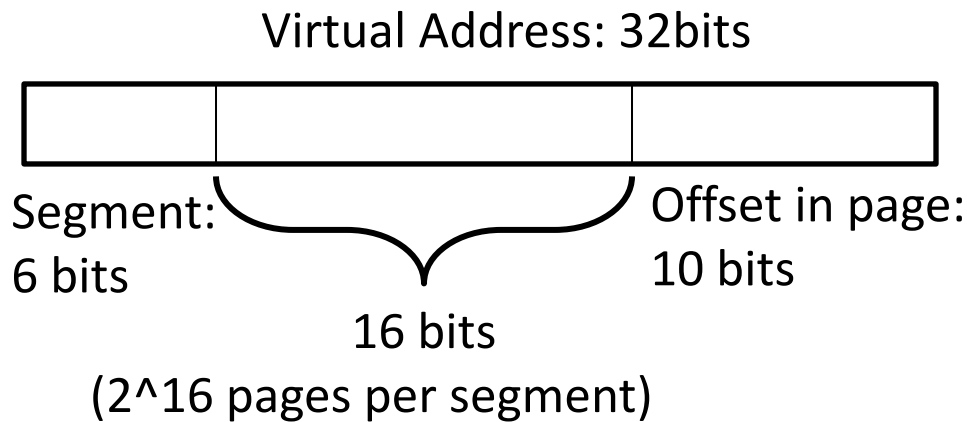
- ◆ Paging + segmentation
- ◆ Virtual address space: 32-bit
- ◆ Page size: 1KB ( $2^{10}$  bytes)
- ◆ PT max 1 frame
- ◆ PTE: 4 bytes ( $2^2$ )
- ◆ Segments per process: Up to 64 ( $2^6$ )



# Sample Question 9.A

Attribution: from U Waterloo CS 350 Study Questions

a. How many levels of page tables are required for virtual address translation in this system?



Each page table fits  
 $2^{10} \div 2^2 = 2^8$   
page table entries.

**A two-level page  
table is needed.**

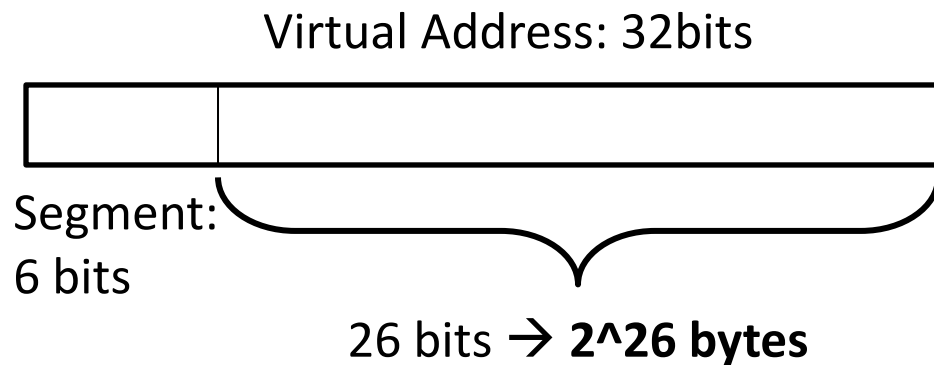
- ◆ Paging + segmentation
- ◆ Virtual address space: 32-bit
- ◆ Page size: 1KB ( $2^{10}$  bytes)
- ◆ PT max 1 frame
- ◆ PTE: 4 bytes ( $2^2$ )
- ◆ Segments per process: Up to 64 ( $2^6$ )



# Sample Question 9.B

Attribution: from U Waterloo CS 350 Study Questions

b. What is the maximum size of a virtual memory segment?



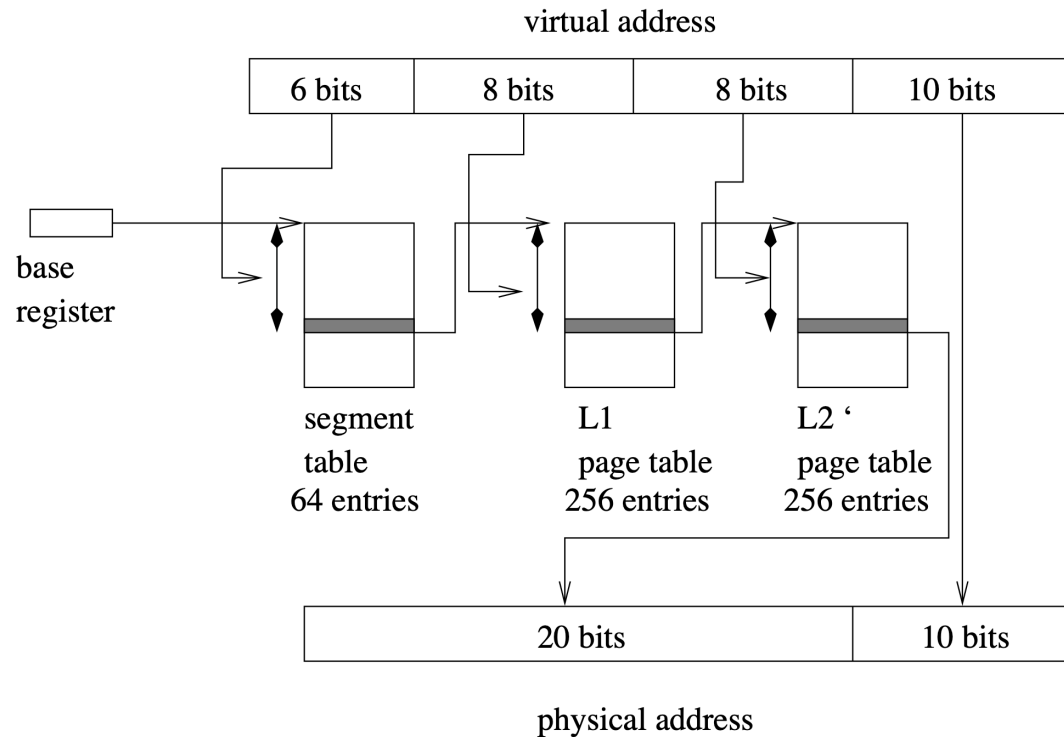
- ◆ Paging + segmentation
- ◆ Virtual address space: 32-bit
- ◆ Page size: 1KB ( $2^{10}$  bytes)
- ◆ PT max 1 frame
- ◆ PTE: 4 bytes ( $2^2$ )
- ◆ Segments per process: Up to 64 ( $2^6$ )



# Sample Question 9.C

Attribution: from U Waterloo CS 350 Study Questions

c. Draw an address translation diagram that shows how a virtual address is translated in this system. Your diagram must show each component of the virtual and physical addresses, and the size of each component. It must show any tables used in translation, and their lengths. It must show how the components of the virtual address are used to index the tables, and how the components of the physical address are determined.



- ◆ Paging + segmentation
- ◆ Virtual address space: 32-bit
- ◆ Page size: 1KB ( $2^{10}$  bytes)
- ◆ PT max 1 frame
- ◆ PTE: 4 bytes ( $2^2$ )
- ◆ Segments per process: Up to 64 ( $2^6$ )

# Sample Question 10

Attribution: from Stanford CS 140 Winter 2007 Final Exam

(16 points) For each of the following protection systems, describe if they more resemble access control lists (ACLs) or capabilities. Be sure to justify your answer.

## Protection Domain

- ◆ Once identity known, provides rules
  - E.g. what is Bob allowed to do?
  - E.g. who can do what to file A?
- ◆ Protection matrix: domains and resources

	File A	Printer B	File C
Domain 1	R	W	RW
Domain 2	RW	W	...
Domain 3	R	...	RW



20

## By Columns: Access Control Lists (ACLs)

- ◆ Each object has a list of <user, privilege> pairs
- ◆ ACL is simple, implemented in most systems
  - Owner, group, world
- ◆ Implementation considerations
  - Stores ACLs in each file
  - Use login authentication to identify
  - Kernel implements ACLs
- ◆ Q: Any issues?



## By Rows: Capabilities

- ◆ For each user, there is a capability list
  - A lists of <object, privilege> pairs
- ◆ Capabilities provide both naming and protection
  - Can only “see” an object if you have a capability
- ◆ Implementation considerations
  - Architecture support
  - Capabilities stored in the kernel
  - Capabilities stored in the user space in encrypted format
- ◆ Q: Issues?



22

## Lecture 16



# Sample Question 10

Attribution: from Stanford CS 140 Winter 2007 Final Exam

A door with a lock that takes a key to unlock.

*Answer: **Capability.** The key is a user-specific capability and the door has no idea who is opening it.*

A door with a badge reader an employee needs to swipe to unlock.

*Answer: **ACL.** A badge identifies the person, and the door security system maintains a list of people allowed through the door, so the resource is maintaining a list of (user, privilege) pairs.*





# Sample Question 11

Attribution: from UCSC CMPS 111 Spring 2002 Final Exam

Suppose that you have a UNIX file system where the disk block size is 1KB, and an inode takes 128 bytes. Disk addresses take 32 bits, and the inode contains space for 64 bytes of data (a recent optimization), 8 direct addresses, one indirect, one double-indirect and one triple-indirect (the rest of the space in the inode is taken up with other information such as ownership and protection). An index block is the same size as a disk block. How much space (including overhead) do files that are: One (1) byte long, 1025 bytes long, 65536 (64KB) bytes long, and 1048576 (1MB) bytes long require? Hint: it may help if you draw a picture of how inodes are used to locate the blocks making up a file.

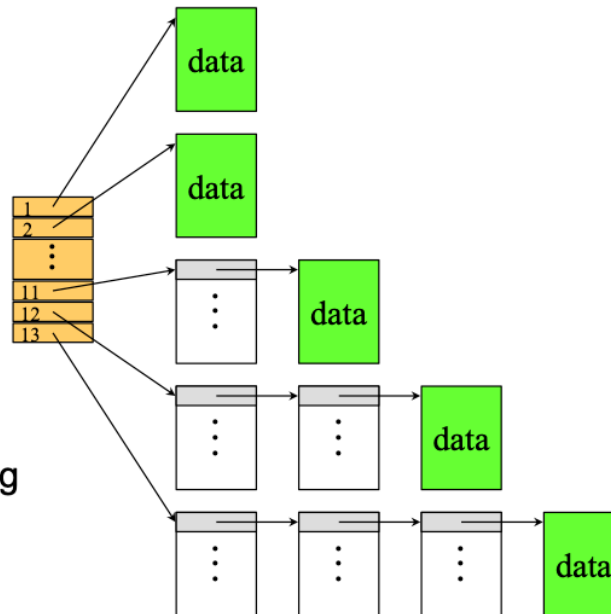
- ◆ Disk block size: 1KB ( $2^{10}$ )
- ◆ inode: 128 bytes
- ◆ 32-bit disk address space (4 bytes)
- ◆ inode: 64B data + 8 direct address + 1 indirect pointer + 1 double-indirect + 1 triple-indirect
- ◆ Index block same size as disk block: 1KB ( $2^{10}$ )



# Reminder: Multi-Level Indexed Files

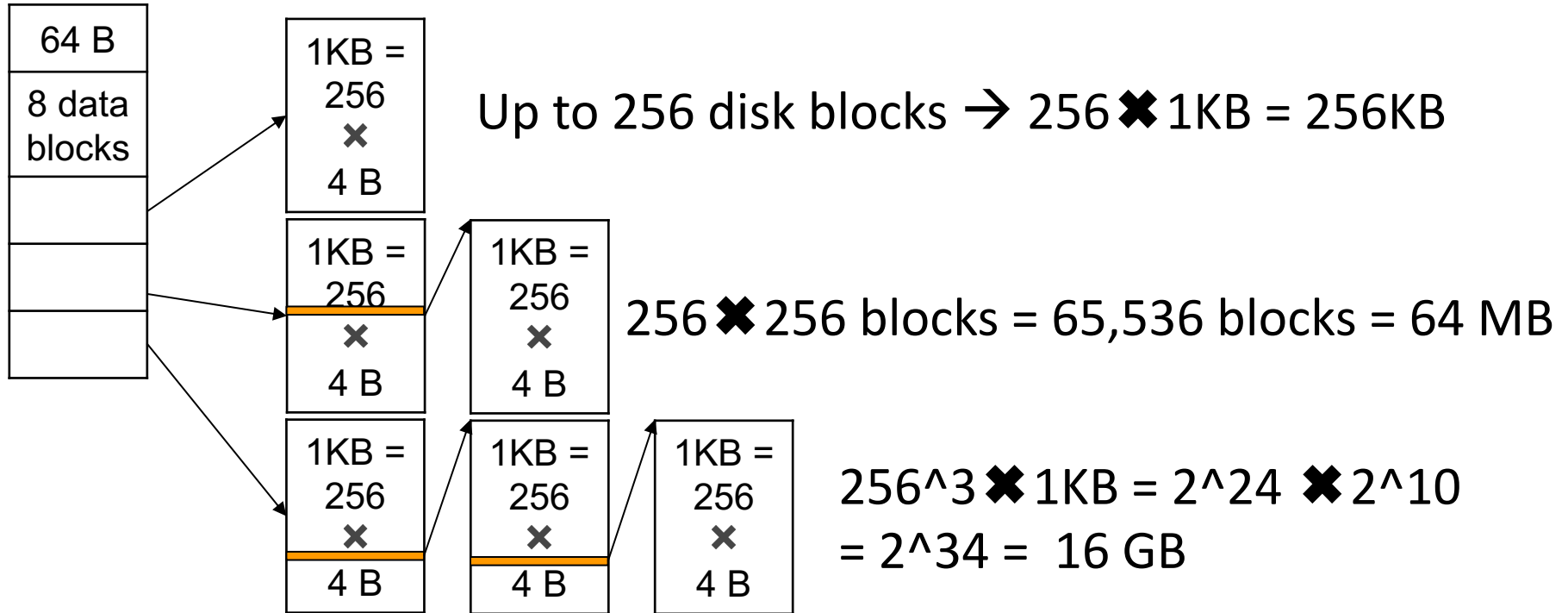
## Hybrid Multi-level Indexed Files (Unix)

- ◆ 13 Pointers in a header
  - 10 direct pointers
  - 11: 1-level indirect
  - 12: 2-level indirect
  - 13: 3-level indirect
- ◆ Pros & Cons
  - In favor of small files
  - Can grow
  - Limit is 16G
  - Can have lots of seeking



12

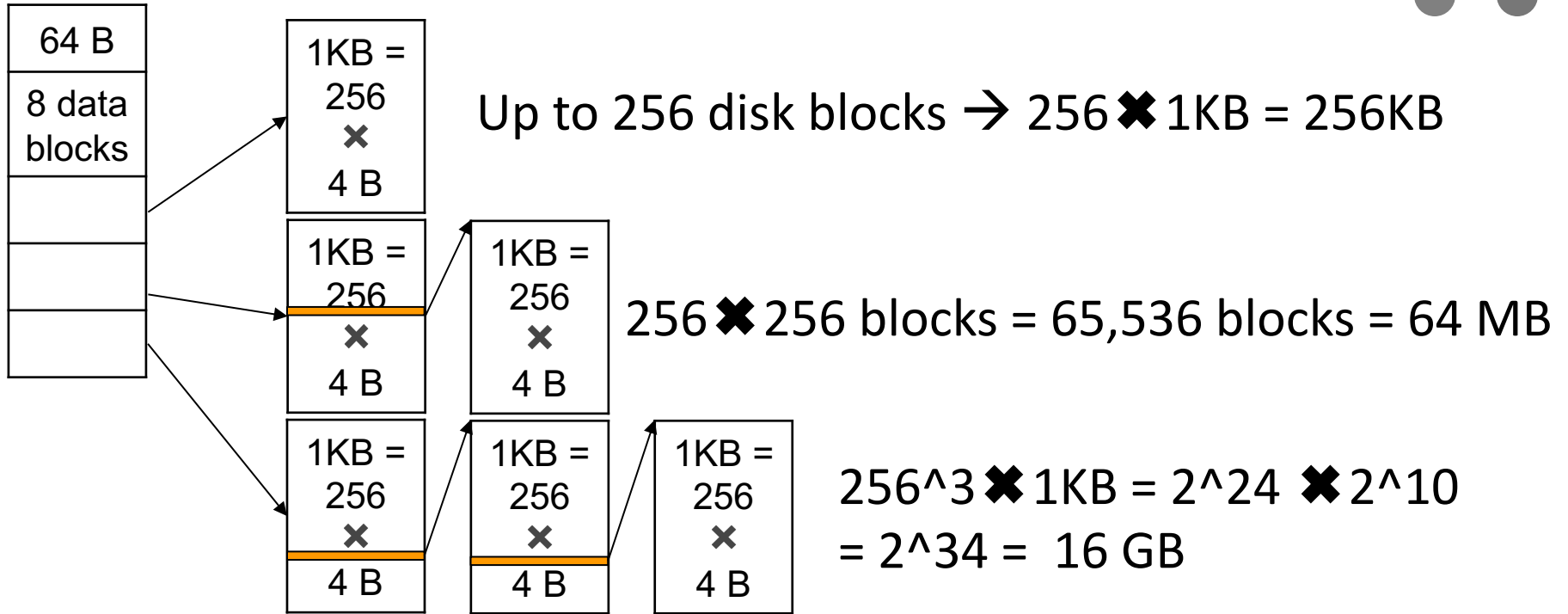
# Sample Question 11 (cont'd)



- ◆ Disk block size: 1KB (2<sup>10</sup>)
- ◆ inode: 128 bytes
- ◆ 32-bit disk address space (4 bytes)
- ◆ inode: 64B data + 8 direct address + 1 indirect + 1 double-indirect + 1 triple-indirect
- ◆ Index block same size as disk block: 1KB (2<sup>10</sup>)



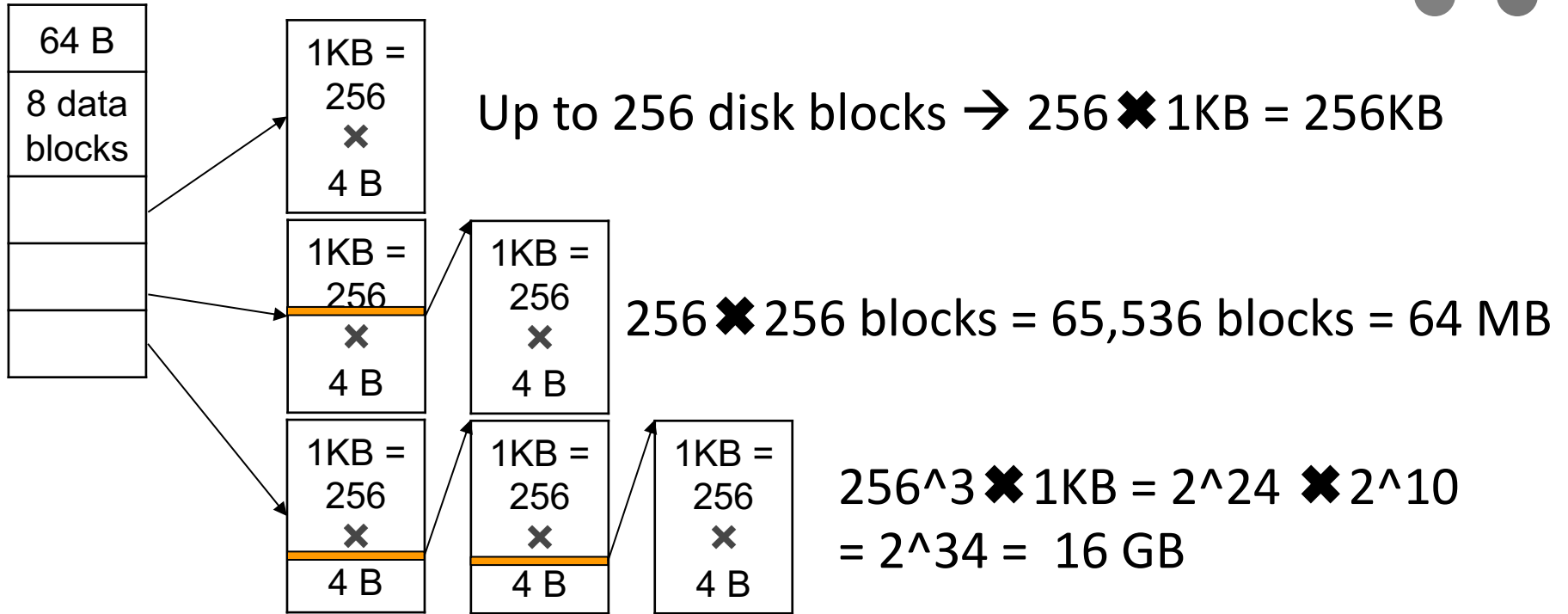
# Sample Question 11 (cont'd)



File Size	Space
1 B	inode can store the data → 128 B



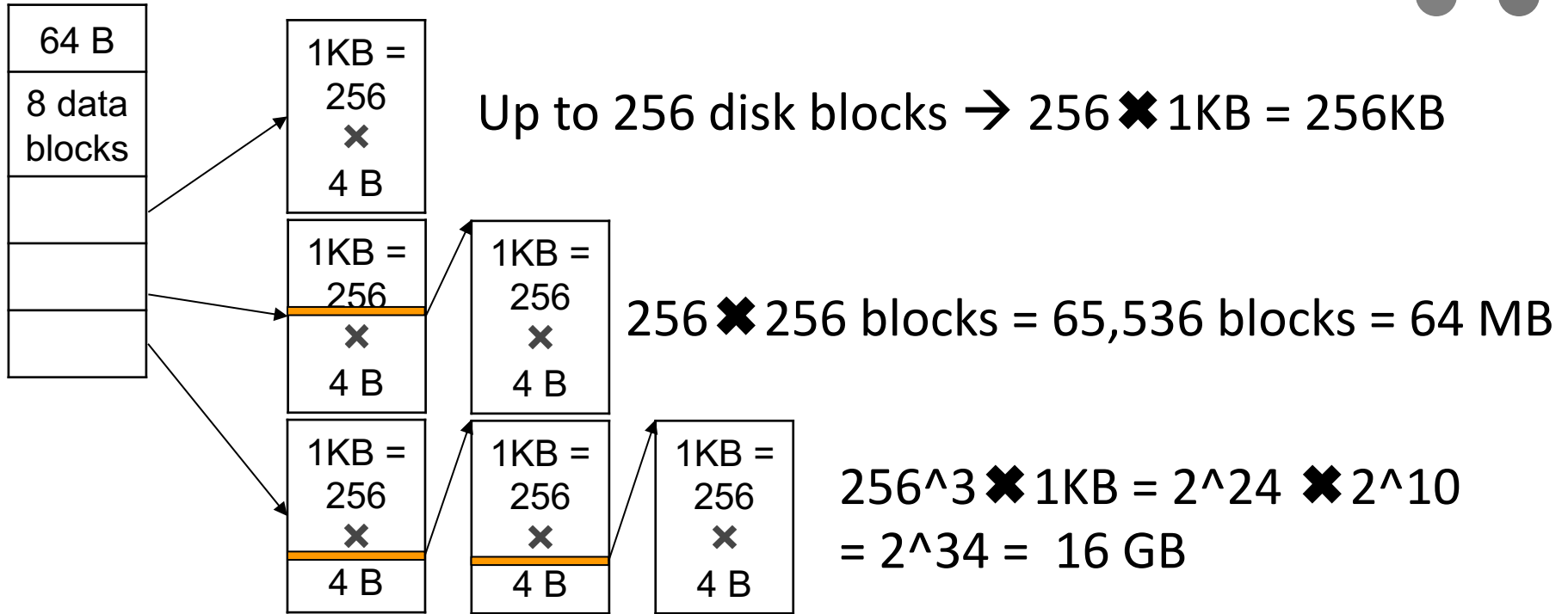
# Sample Question 11 (cont'd)



File Size	Space
1 B	inode can store the data → 128 B
1025 B	inode + 1 data block: 128 + 1024 = 1152 B



# Sample Question 11 (cont'd)



File Size	Space
1 B	inode can store the data → 128 B
1025 B	inode + 1 data block: 128 + 1024 = 1152 B
65,536 B	Composition: 64B + 8,192B + 57,280B Structure: inode + 1 indirect pointer block Total space: 128B + 8,192B + 1,024B + 56KB(rounded to closest disk block) = <b>66,688B</b>



# Sample Question 12

Consider a disk with the following characteristics:

- Number of surfaces: 8 ( $= 2^3$ )
- Number of tracks / surface: 512 K ( $= 2^{19}$ )
- Number of bytes / track: 8 MB ( $= 2^{23}$  bytes)
- Number of sectors / track: 8 K ( $= 2^{13}$ )
- On-disk cache: 16 MB ( $= 2^{24}$  bytes)

Attribution:

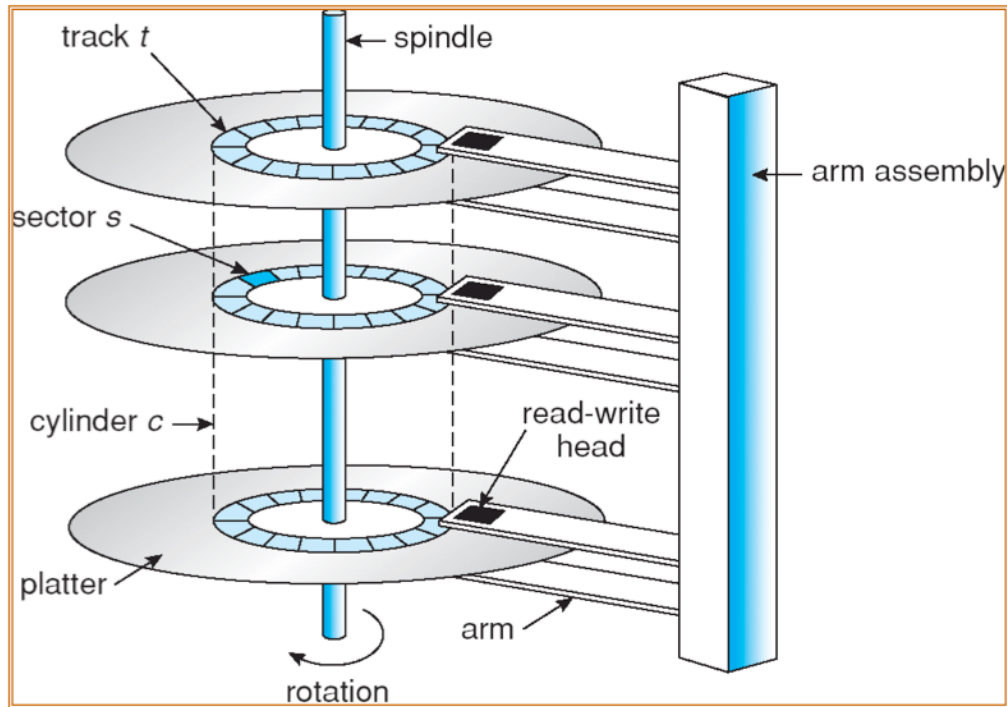
from U Wisc. CS 537 Fall 2015 Midterm Exam

- 1) How many heads does this disk have?
- 2) What is the size of each sector?
- 3) How many bytes per cylinder?
- 4) What is the total capacity of this disk?



# Reminder: HDD Moving Head Mechanism

## Moving-Head Disk Mechanism





# Sample Question 12

Consider a disk with the following characteristics:

- Number of surfaces: 8 ( $= 2^3$ )
- Number of tracks / surface: 512 K ( $= 2^{19}$ )
- Number of bytes / track: 8 MB ( $= 2^{23}$  bytes)
- Number of sectors / track: 8 K ( $= 2^{13}$ )
- On-disk cache: 16 MB ( $= 2^{24}$  bytes)

Attribution:

from U Wisc. CS 537 Fall 2015 Midterm Exam

2) What is the size of each sector?

Answer:

$$\text{bytes/sector} = (\text{bytes/track}) / (\text{sectors/track})$$

$$\text{bytes/sector} = 2^{23} / 2^{13} = 2^{10} = \mathbf{1KB}$$

4) What is the total capacity of this disk?

Answer:

$$\text{total bytes} = (\text{bytes/cylinder}) * \text{cylinders}$$

$$\text{total bytes} = 2^{26} * 2^{19} = 2^{45} = \mathbf{32TB}$$

1) How many heads does this disk have?

Answer:

*8 surfaces, 1 head per surface needed → 8 heads*

3) How many bytes per cylinder?

Answer:

$$\text{bytes/cylinder} = (\text{bytes/track}) * (\text{tracks/cylinder})$$

$$\text{bytes/cylinder} = 2^{23} * 8 = 2^{26} = \mathbf{64MB}$$



# Sample Question 13

Attribution: from Cornell CS 4410 Spring 2007 Midterm Exam

2. (24 points total) CPU Scheduling. Here is a table of processes and their associated arrival and running times.

Process ID	Arrival Time	Expected CPU Running Time
Process 1	0	5
Process 2	1	5
Process 3	5	3
Process 4	6	2

- a. (12 points) Show the scheduling order for these processes under First-In-First-Out (FIFO), Shortest-Job First (SJF), and Round-Robin (RR) with a quantum = 1 time unit. *Assume that the context switch overhead is 0 and new processes are added to the **head** of the queue except for FIFO.*



# Sample Question 13 (cont'd)

Attribution: from Cornell CS 4410 Spring 2007 Midterm Exam

Process ID	Arrival Time	Expected CPU Running Time
Process 1	0	5
Process 2	1	5
Process 3	5	3
Process 4	6	2

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FIFO	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4



# Sample Question 13 (cont'd)

Attribution: from Cornell CS 4410 Spring 2007 Midterm Exam

Process ID	Arrival Time	Expected CPU Running Time
Process 1	0	5
Process 2	1	5
Process 3	5	3
Process 4	6	2

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FIFO	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4
SJF	1	1	1	1	1	3	3	3	4	4	2	2	2	2	2



# Sample Question 13 (cont'd)

Attribution: from Cornell CS 4410 Spring 2007 Midterm Exam

Process ID	Arrival Time	Expected CPU Running Time
Process 1	0	5
Process 2	1	5
Process 3	5	3
Process 4	6	2

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FIFO	1	1	1	1	1	2	2	2	2	2	3	3	3	4	4
SJF	1	1	1	1	1	3	3	3	4	4	2	2	2	2	2
RR	1	2	1	2	1	3	4	2	1	3	4	2	1	3	2

The second part of the question asks about wait times.



# Sample Question 14

Attribution: from UC Berkeley CS 162 Fall 2006 Midterm Exam

For each of the following techniques for synchronization, assume that there are two threads competing to execute a critical section. Further, assume that:

1. A critical section is “protected” if only one thread can enter the critical section at a time.
2. The synchronization is “fair” if, when each thread attempts to acquire the critical section repeatedly, then each thread will enter the critical section about the same number of times.

*Note: Assume that all flags start out “false”. Also assume that store is atomic.*

**Synchronization technique #1:** Suppose each thread does the following:

1. `while (flag == true)`
2.     `do nothing;`
3. `flag = true;`
4. `Execute Critical Section;`
5. `flag = false;`

**Problem 3a[2pts]:** Will this protect the critical section? If “yes”, explain why. If “no”, give an example interleaving that will fail to protect the critical section.

**Answer: No**

*Example: Thread A runs line 1, determines flag is false, and gets context-switched.*

*Then, thread B runs line 1, and it also determines flag is false.*

*Now threads A and B can both access critical section.*



# Sample Question 14 (cont'd)

Attribution: from UC Berkeley CS 162 Fall 2006 Midterm Exam

For each of the following techniques for synchronization, assume that there are two threads competing to execute a critical section. Further, assume that:

1. A critical section is “protected” if only one thread can enter the critical section at a time.
2. The synchronization is “fair” if, when each thread attempts to acquire the critical section repeatedly, then each thread will enter the critical section about the same number of times.

*Note: Assume that all flags start out “false”. Also assume that store is atomic.*

**Synchronization technique #1:** Suppose each thread does the following:

1. `while (flag == true)`
2.     `do nothing;`
3. `flag = true;`
4. *Execute Critical Section;*
5. `flag = false;`

**Problem 3b[2pts]:** Assume this code protects the critical section. Is this code “fair”? Explain.

Answer: Yes

*This code is symmetric and thus each thread has an equal chance.*



# Sample Question 14 (cont'd)

Attribution: from UC Berkeley CS 162 Fall 2006 Midterm Exam

For each of the following techniques for synchronization, assume that there are two threads competing to execute a critical section. Further, assume that:

1. A critical section is “protected” if only one thread can enter the critical section at a time.
2. The synchronization is “fair” if, when each thread attempts to acquire the critical section repeatedly, then each thread will enter the critical section about the same number of times.

*Note: Assume that all flags start out “false”. Also assume that store is atomic.*

**Synchronization technique #2:** Suppose we have different code for each thread:

<u>THREAD A</u>	<u>THREAD B</u>
A1. <code>flag_A = true;</code>	B1. <code>flag_B = true;</code>
A2. <code>while (flag_B == true)</code>	B2. <code>if (flag_A == false)</code>
A3. <code>do nothing;</code>	B3. <code>Execute Critical Section;</code>
A4. <code>Execute Critical Section;</code>	B4. <code>flag_B = false;</code>
A5. <code>flag_A = false;</code>	

**Problem 3c[2pts]:** Will this protect the critical section? If “yes”, explain why. If “no”, give an example interleaving that will fail to protect the critical section.

Answer: **Yes**

*Thread A only enters the critical section when `flag_B` is false (this is satisfied only while thread B is executing before B1 or after B4).*

*Thread B only enters the critical section when `flag_A` is false (this is satisfied only while thread A is executing before A1 or after A5).*





# Sample Question 14 (cont'd)

Attribution: from UC Berkeley CS 162 Fall 2006 Midterm Exam

For each of the following techniques for synchronization, assume that there are two threads competing to execute a critical section. Further, assume that:

1. A critical section is “protected” if only one thread can enter the critical section at a time.
2. The synchronization is “fair” if, when each thread attempts to acquire the critical section repeatedly, then each thread will enter the critical section about the same number of times.

*Note: Assume that all flags start out “false”. Also assume that store is atomic.*

**Synchronization technique #2:** Suppose we have different code for each thread:

<u>THREAD A</u>	<u>THREAD B</u>
A1. flag_A = true;	B1. flag_B = true;
A2. while (flag_B == true)	B2. if (flag_A == false)
A3.     do nothing;	B3.     Execute Critical Section;
A4. Execute Critical Section;	B4. flag_B = false;
A5. flag_A = false;	

**Problem 3d[2pts]:** Assume this code protects the critical section. Is this code “fair”? Explain.

Answer: No

Thread A always gets a chance to run the critical section during an execution of A1-A5. On the other hand, Thread B will be prevented from running the critical section whenever Thread A is in that region. If A and B are running in a tight loop, Thread B only gets to run the critical section if it is lucky enough to execute B2 between the execution of A5 and the next execution of A1.



# Sample Question 14 (cont'd)

Attribution: from UC Berkeley CS 162 Fall 2006 Midterm Exam

For each of the following techniques for synchronization, assume that there are two threads competing to execute a critical section. Further, assume that:

1. A critical section is “protected” if only one thread can enter the critical section at a time.
2. The synchronization is “fair” if, when each thread attempts to acquire the critical section repeatedly, then each thread will enter the critical section about the same number of times.

*Note: Assume that all flags start out “false”. Also assume that store is atomic.*

**Synchronization technique #3:** Suppose each thread does the following:

1. `while (TestAndSet(flag) == false)`
2.     `do nothing;`
3.     `Execute Critical Section;`
4.     `flag = false;`

**Problem 3e[3pts]:** Will this protect the critical section? If “yes”, explain why. If “no”, explain and explain how to fix it.

Answer: No

Since TAS sets a memory location to 1 (true), the locked condition is indicated by the value of `flag == true`. Hence, the above code doesn't wait when the lock is already taken. Hence, it doesn't protect the critical section. To fix the code, we should replace `TAS(flag) == false` with `TAS(flag) == true`.



# Sample Question 14 (cont'd)

Attribution: from UC Berkeley CS 162 Fall 2006 Midterm Exam

For each of the following techniques for synchronization, assume that there are two threads competing to execute a critical section. Further, assume that:

1. A critical section is “protected” if only one thread can enter the critical section at a time.
2. The synchronization is “fair” if, when each thread attempts to acquire the critical section repeatedly, then each thread will enter the critical section about the same number of times.

*Note: Assume that all flags start out “false”. Also assume that store is atomic.*

**Synchronization technique #3:** Suppose each thread does the following:

1. `while (TestAndSet(flag) == false)`
2.     `do nothing;`
3.     *Execute Critical Section;*
4.     `flag = false;`


**Problem 3f[2pts]:** Assume the above code (or your fixed version). Will this code be “fair”? Explain.

Answer: **Yes** – *The code is symmetric*



# Summary

---

- ◆ We all made it through this crazy semester! 
- ◆ Cover all lectures, study MOS reading assignments, and make sure you are fully familiar with course projects.
- ◆ Please make sure to plan for Project 6 ahead of the time.
- ◆ Reach out to us with your questions.