CSE 421/521 - Operating Systems
Spring 2018

LECTURE - XXV

MIDTERM-2 REVIEW

Tevfik Koşar

University at Buffalo

May 8th, 2018

Midterm-2 Exam

May 10th, Thursday

3:30pm - 4:50pm

Room: NSC 201

Topics included in Midterm-II

- Main Memory Management
- Virtual Memory Management
- File Systems
- Mass Storage and I/O
- Distributed Systems
- Protection and Security

Main Memory Management

- Contiguous Allocation
- Dynamic Allocation Algorithms
- Fragmentation
- Address Binding
- Address Protection
- Paging
- Segmentation

Virtual Memory

- Demand Paging
- Page Faults
- Page Replacement
- Page Replacement Algorithms (FIFO, LRU, LRU-Clock, LFU, MFU, Optimal)
- Performance of Demand Paging

Mass Storage & I/O

- Disk Mechanism & Structure
- Disk Scheduling Algorithms
  - FCFS, SSTF, SCAN, LOOK, C-SCAN, C-LOOK
- Hierarchical Storage Management
- RAID Architectures
  - RAID 0-6, RAID 0+1, RAID 1+0

File Systems

- Directory structure & implementation
- File allocation methods
  - contiguous, linked, indexed
- Free space management
  - bit vectors, linked lists, grouping, counting

Distributed Coordination

- Event Ordering
  - Happened before relationship
- Distributed Mutual Exclusion
  - Centralized & Fully Distributed Approaches
- Distributed Deadlock Prevention
  - Resource Ordering
  - Timestamp Ordering (Wait-die & Wound-wait)
- Distributed Deadlock Detection
  - Centralized & Fully Distributed Approaches

Protection and Security

- Security Violation Categories
- Security Violation Methods
- Cryptography
- Symmetric & Asymmetric Encryption
- Program & Network Threats —> Not included! (Lecture-24: p.20-31)

Exercise Questions

Question 1 (a)

Provide short answers to the following questions:

(a) FCFS disk scheduling tends to favor accesses to innermost/ middle/outermost/none cylinders of the disk.

Question 1 (a)

- Provide short answers to the following questions:

(a) FCFS disk scheduling tends to favor accesses to innermost/ middle/outermost/none cylinders of the disk.

FCFS does not favor accesses to any particular part of the disk

Question 1 (b)

(b) Can the segments that are shared between two or more processes be swapped out to the disk?

Question 1 (b)

(b) Can the segments that are shared between two or more processes be swapped out to the disk?

Yes, sharing does not require locking.

Question 1 (c)

(c) The layout of disk blocks for a file using Unix inodes is always/ sometimes/never contiguous on disk?

Question 1 (c)

(c) The layout of disk blocks for a file using Unix inodes is always/ sometimes/never contiguous on disk?

Sometimes.

Question 1 (d)

(d) A piece of code which is made available to unsuspecting user, that misuses its environment is called:

1. Trojan Horse
2. Stealth Virus
3. Logic Bomb
4. Trap Door

Question 1 (d)

(d) A piece of code which is made available to unsuspecting user, that misuses its environment is called:

1. Trojan Horse
2. Stealth Virus
3. Logic Bomb
4. Trap Door

Trojan Horse.

(PS: Not included in the exam!)

Question 1 (e)

(e) The modified (dirty) bit is used for the purpose of

1. Implementing FIFO page replacement algorithm
2. To reduce the average time required to service page faults
3. Dynamic allocation of memory used by one process to another
4. All of the above

Question 1 (e)

(e) The modified (dirty) bit is used for the purpose of

1. Implementing FIFO page replacement algorithm
2. To reduce the average time required to service page faults
3. Dynamic allocation of memory used by one process to another
4. All of the above

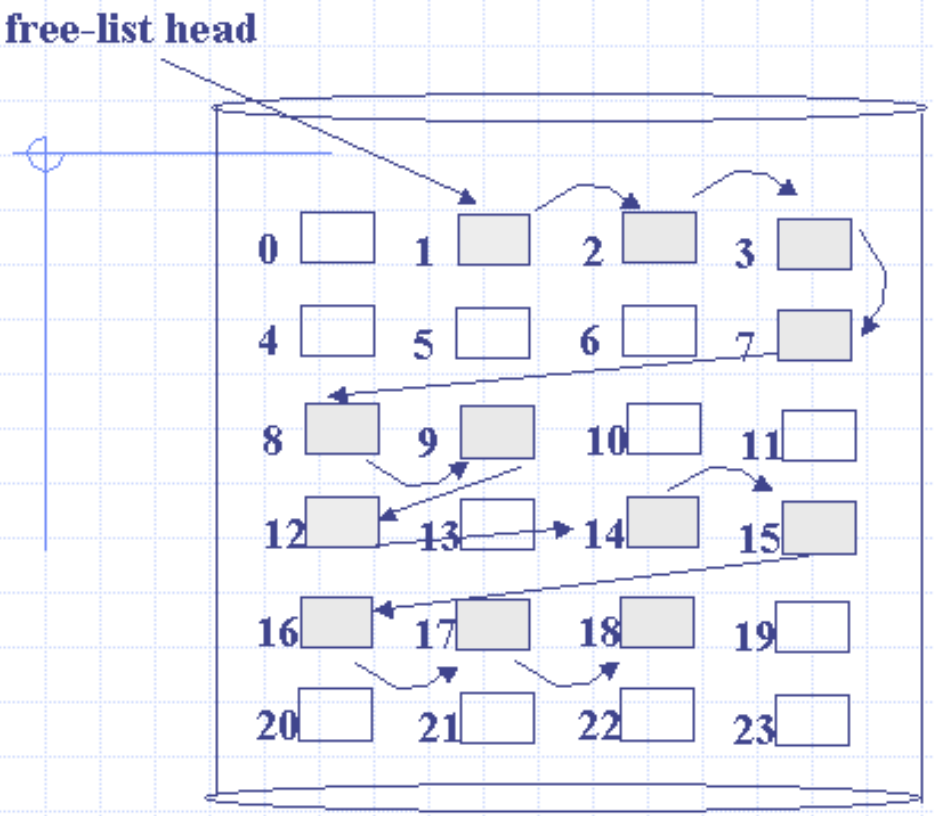
To reduce the average time required to service page fault

Question 2

- In terms of reliability and performance, compare bit vector implementation of a free block list with keeping a list of free blocks where the first few bytes of each free block provide the logical sector number of the next free block.

Remember

Bit Map/Linked List/Grouping/Counting



grouping (n=3)

1	2,3,7
7	8,9,12
12	14,15,16
16	17,18,-1

bit map: 011100011100101111100000

counting: (1,3), (7, 3), (12, 1), (14, 5)

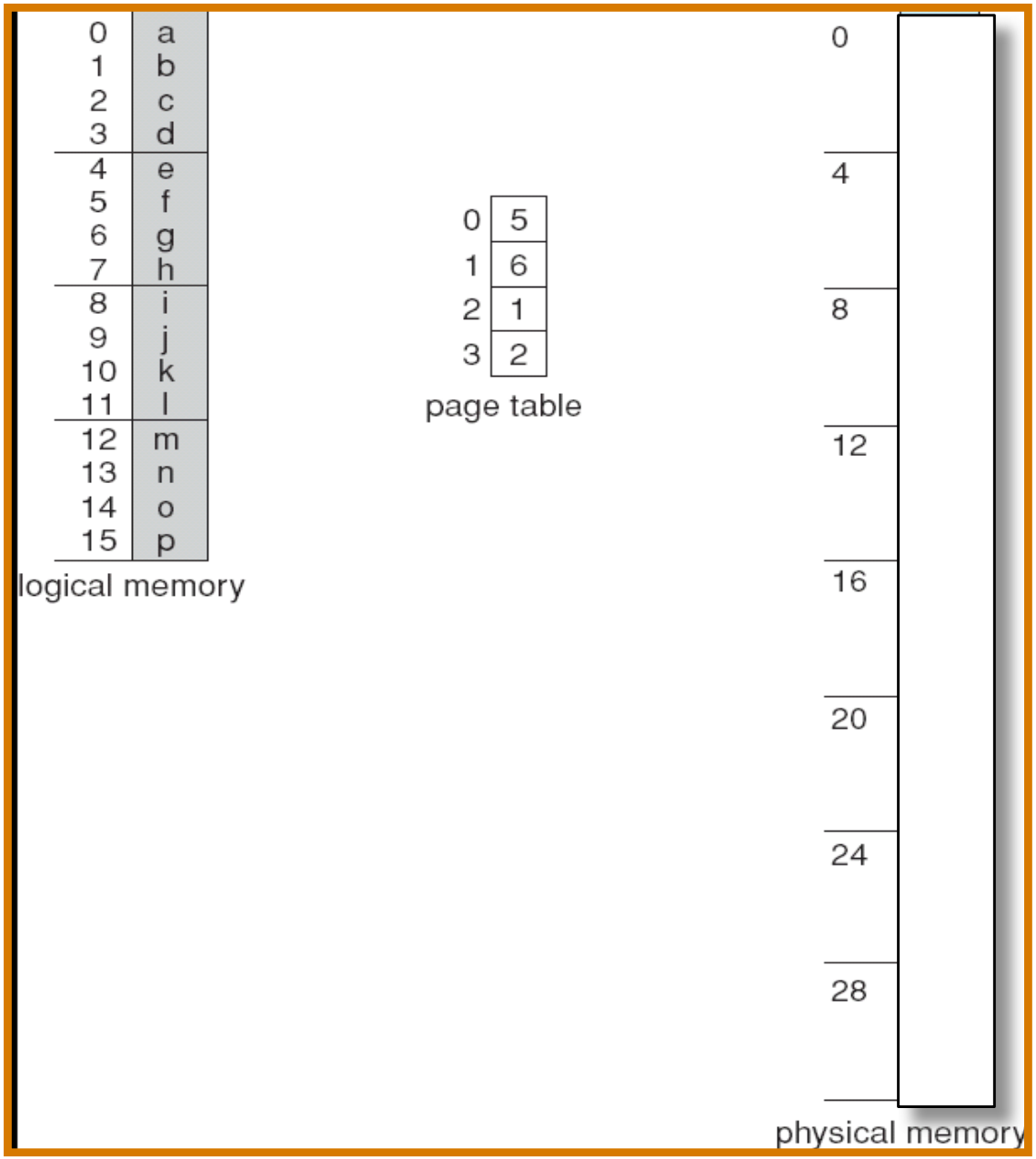
Question 2 - Solution

performance: Bit vector implementation is more efficient since it allows fast and random access to free blocks, linked list approach allows only sequential access, and each access results in a disk read. Bit vector implementation can also find *n* consecutive free blocks much faster. (Assuming bit vector is kept in memory)

reliability: If an item in a linked list is lost, you cannot access the rest of the list. With a bit vector, only those items are lost. Also, its possible to have multiple copies of the bit vector since it is a more compact representation. Although keeping the bit vector in memory seem to be unreliable, you can always keep an extra copy on the disk.

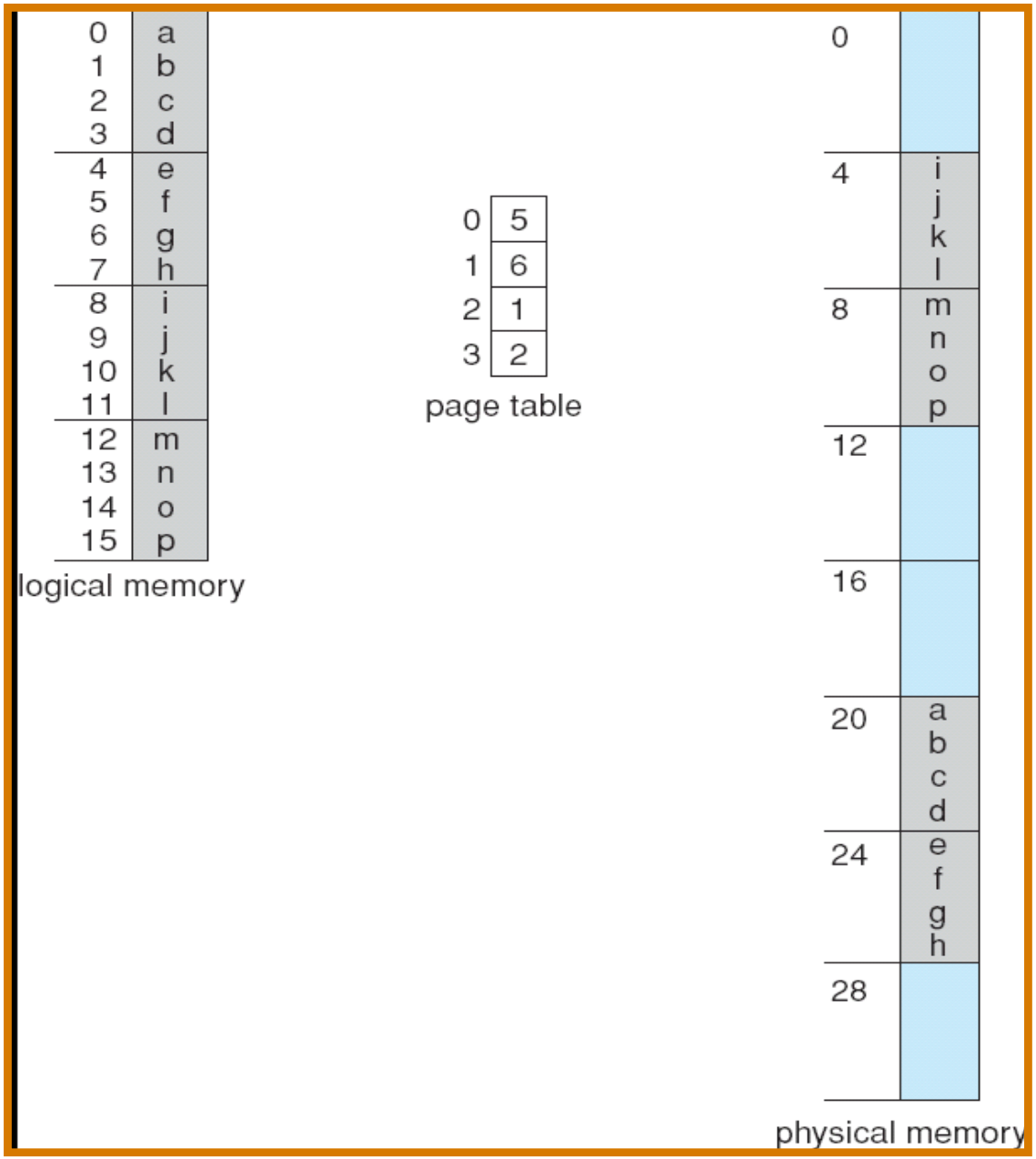
Question 3 (a)

- Consider the paging table on the right. What are the physical addresses of the following logical addresses [p,d] and the words on them :
- a) 0,0
- b) 1,4
- c) 2,3



Question 3 (a) - Solution

- Consider the paging table on the right. What are the physical addresses of the following logical addresses [p,d] and the words on them :
- a) 0,0 --> 20
- b) 1,4 --> illegal
- c) 2,3 --> 7



Question 3 (b)

- Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 1, 100

b. 2, 0

c. 3, 580

Question 3 (b) - Solution

- Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 1, 100

illegal reference (2300+100 is not within segment limits)

b. 2, 0

physical address = 90 + 0 = 90

c. 3, 580

illegal reference (1327 + 580 is not within segment limits)

Question 4 (a)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(a) Install a faster CPU.

Question 4 (a)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(a) Install a faster CPU.

NO. a faster CPU reduces the CPU utilization further since the CPU will spend more time waiting for a process to enter in the ready queue.

Question 4 (b)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(b) Install a bigger paging disk.

Question 4 (b)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(b) Install a bigger paging disk.

NO. the size of the paging disk does not affect the amount of memory that is needed to reduce the page faults.

Question 4 (c)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(c) Decrease the degree of multiprogramming.

Question 4 (c)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(c) Decrease the degree of multiprogramming.

YES. by suspending some of the processes, the other processes will have more frames in order to bring their pages in them, hence reducing the page faults.

Question 4 (d)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(d) Install more main memory.

Question 4 (d)

- Consider a demand-paging system with the following time-measured utilization:

  CPU utilization 18%
  Paging disk 96%
  Other I/O devices 6%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(d) Install more main memory.

Likely. more pages can remain resident and do not require paging to or from the disks (i.e. would depend on the page replacement algorithm you are using and the page reference sequence).

Question 5 (a)

Consider a demand paging system where 40% of the page table is stored in the registers, and the rest needs to be addressed from the memory. Assume any reference to the memory takes 100 nanoseconds, and 20% of the time the page that is being replaced has its dirty bit set to 1. Swapping a page in takes 1000 microseconds and swapping out a page takes 2000 microseconds.

(a) How long does a paged memory reference take?

Question 5 (a)

Consider a demand paging system where 40% of the page table is stored in the registers, and the rest needs to be addressed from the memory. Assume any reference to the memory takes 100 nanoseconds, and 20% of the time the page that is being replaced has its dirty bit set to 1. Swapping a page in takes 1000 microseconds and swapping out a page takes 2000 microseconds.

(a) How long does a paged memory reference take?

A paged memory reference would take (60/100) x 100 + 100 = 160 ns

Question 5 (b)

(b) (5 pts) What is the Effective Access Time (EAT) if the page fault ratio is 0.001?

Question 5 (b)

(b) (5 pts) What is the Effective Access Time (EAT) if the page fault ratio is 0.001?

$$EAT = (1-p) \times 0.160 + p \times ((80/100)\times 1000 + (20/100)\times 3000 + 0.160)$$

$$= 0.160 + 1400p$$

$$= 0.160 + 1400 \times 1/1000$$

$$= 1.56 \text{ microseconds}$$

Question 5 (c)

What should be the maximum acceptable page-fault rate if we only want 50% performance degradation?

Question 5 (c)

What should be the maximum acceptable page-fault rate if we only want 50% performance degradation?

0.240 ≥ 0.160 + 1400p

0.240 ≥ 0.160 + 1400p

0.080 ≥ 1400p

0.080 / 1400 ≥ p
// can leave it here.. approx. 0.000057

Question 6 (a)

Consider a virtual memory system with 34-bit addresses. The first 23 bits are used as a page number, and the last 11 bits is the offset. (Note that $2^{10} = 1K$, $2^{20} = 1M$.)

(a) How many words does a single page frame have?

Question 6 (a)

Consider a virtual memory system with 34-bit addresses. The first 23 bits are used as a page number, and the last 11 bits is the offset. (Note that $2^{10} = 1K$, $2^{20} = 1M$.)

(a) How many words does a single page frame have?

Answer: Each frame has $2^{11} = 2048$ words

Question 6 (b)

Assuming there are 2^{10} frames in the physical memory, how many bits are needed to address the physical memory?

Question 6 (b)

Assuming there are 2^{10} frames in the physical memory, how many bits are needed to address the physical memory?

Answer: Physical memory size = $2^{10} * 2^{11} = 2^{21}$ words

21 bits are needed to address the physical memory

Question 6 (c)

Assuming a word is 32 bytes, what is the total size of the logical address space?

Question 6 (c)

Assuming a word is 32 bytes, what is the total size of the logical address space?

Answer: $2^{23} * 2^{11} * 2^{5} = 2^{39}$ bytes = 512 GB

Question 6 (d)

(d) Assuming the single-level paging and each page-table entry to be 4 bytes, how many Megabytes of memory is needed to store the page table?

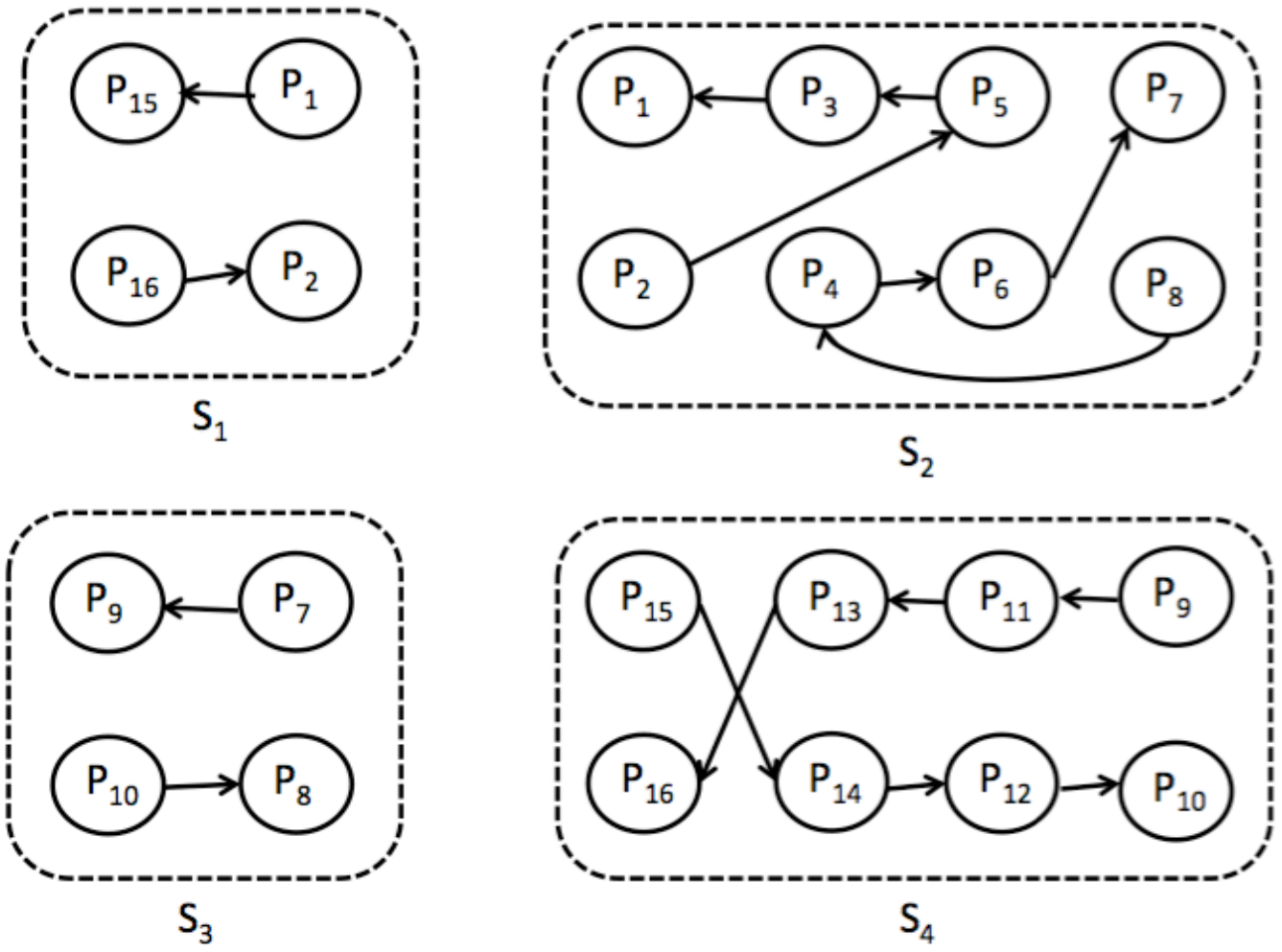
Question 6 (d)

(d) Assuming the single-level paging and each page-table entry to be 4 bytes, how many Megabytes of memory is needed to store the page table?

Answer: 2^{23} entries * 4 bytes = 2^{25} bytes = 32 MB

Question 7

Consider a distributed system which consists of 4 sites given below.



Question 7 (a)

(a) Would you use a centralized vs distributed deadlock detection algorithm in this case? How would that algorithm work, shortly describe.

Question 7 (a)

(a) Would you use a centralized vs distributed deadlock detection algorithm in this case? How would that algorithm work, shortly describe.

We would use a centralized algorithm, since each site only keeps the wait-for graph for their local resources [No EX nodes required by the distributed algorithm]. Whenever there is a need to run the deadlock detection algorithm, each site will send their local graphs to the central coordinator, which in turn will generate a single global wait-for graph, and then will run the deadlock detection algorithm on this graph.

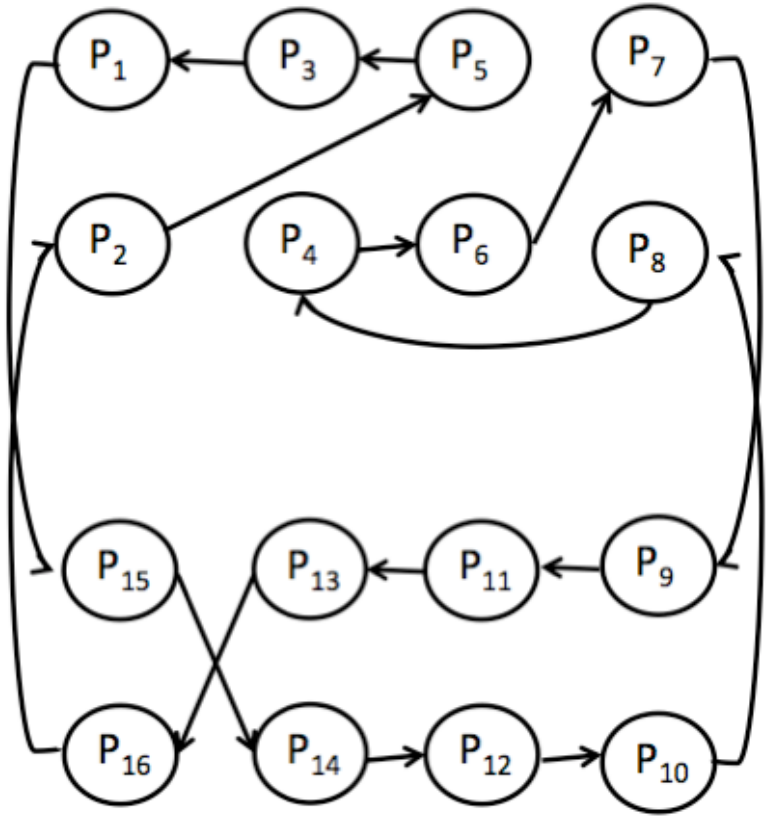
Question 7 (b)

(b) Is the distributed system in a deadlock? If so, please show the cycle.

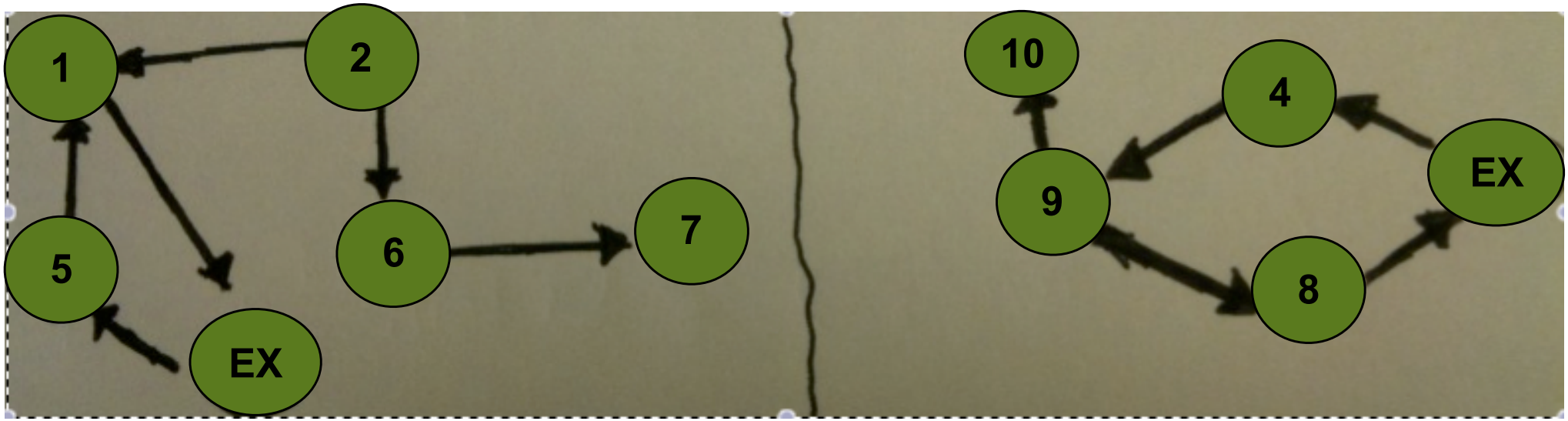
Question 7 (b)

(b) Is the distributed system in a deadlock? If so, please show the cycle.

The global wait-for graph generated by the coordinator is shown below. As it can be seen, there is cycle involving all nodes. Which means all processes are involved in a deadlock



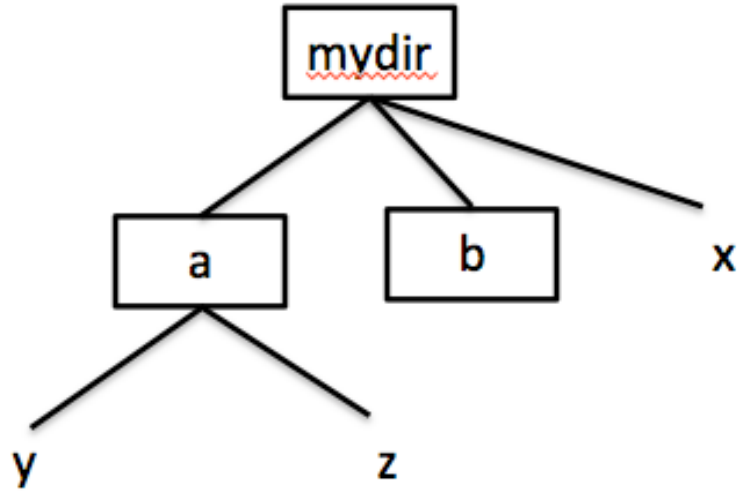
Question 8



Considering the above local wait-for graphs at sites S1 and S2, is the system D in a deadlocked state? If so, which processes are involved in the deadlock? Show how you would check the existence of a deadlock.

Question 9 (a)

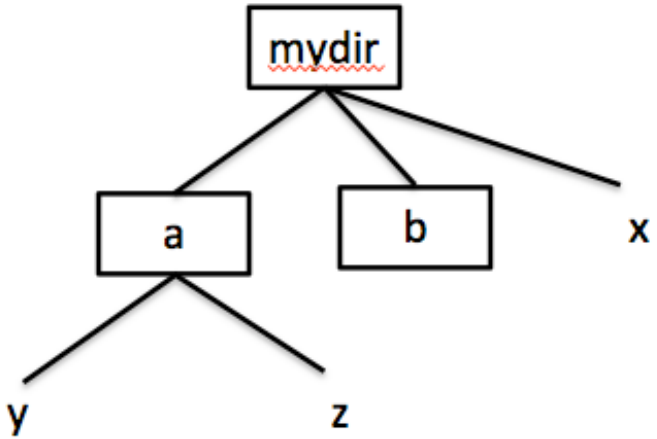
Consider the following directory structure (user view):



Assume **mydir** (10), **a** (20), and **b** (30) are directories and **x** (40), **y** (50), and **z** (60) are files with inode numbers given in parenthesis. The inode number for mydir's parent directory is 1.

(a) Please show the system representation (system view) of this directory tree.

Assume **mydir** (10), **a** (20), and **b** (30) are directories and **x** (40), **y** (50), and **z** (60) are files with inode numbers given in parenthesis. The inode number for mydir's parent directory is 1. Please show the system representation (system view) of this directory tree.

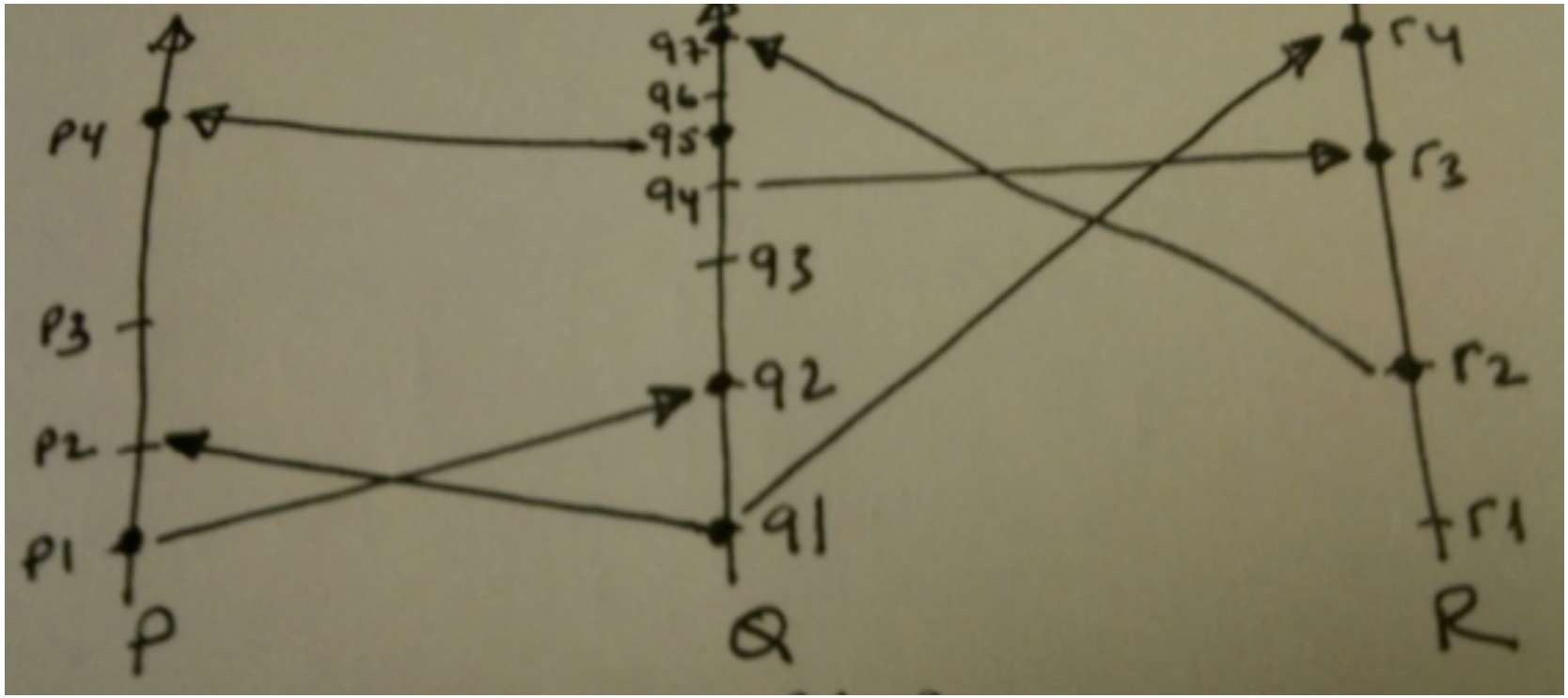


Question 9 (b)

(b) Show the system representation (system view) after executing all of the following commands:

```
$ rm mydir/x
$ cp mydir/a/z mydir
$ ln mydir/a/y mydir/b/ylink
$ mv mydir/b mydir/a
```

Question 10



a) r2 happens before p4 :

b) p1 happens before r3 :

c) p2 happens before r4 :

d) p1 and r4 are concurrent processes :

e) r1 and p4 are concurrent processes :

Question 11 (a)

a) Calculate the Hemming Code (7,4) for this data byte: **1011 0001**

Remember

- Linear error correcting code (named after its inventor - Richard Hamming)
- Hamming (7,4) -> 4 data bits, 3 parity bits
  - Can find and correct 1-bit errors
  - Can find but not correct 2-bit errors
- Hamming bits table:
  - p1 = d1 ⊕ d2 ⊕ d4
  - p2 = d1 ⊕ d3 ⊕ d4
  - p3 = d2 ⊕ d3 ⊕ d4

Bit position		1	2	3	4	5	6	7	...
Encoded data bits		p1	p2	d1	p3	d2	d3	d4	
Parity bit coverage	p1	X		X		X		X	
	p2		X	X			X	X	
	p3				X	X	X	X	

Question 11 (a)

a) Calculate the Hemming Code (7,4) for this data byte: **1011 0001**

1. Divide the data byte into two 4-bit blocks: 1011 and 0001.
2. For each calculate the parity bits, i.e., for 1011:

   p1 = d1 ⊕ d2 ⊕ d4 = 1 ⊕ 0 ⊕ 1 = 0

   p2 = d1 ⊕ d3 ⊕ d4 = 1 ⊕ 1 ⊕ 1 = 1

   p3 = d2 ⊕ d3 ⊕ d4 = 0 ⊕ 1 ⊕ 1 = 0
3. Arrange the parity and data bits as follows:

   p1 p2 d1 p3 d2 d3 d4 = 0110011
4. Repeat the same for the second block, i.e., 0001, and get:

   p1 p2 d1 p3 d2 d3 d4 = 1101001

Question 11 (b)

b) If the parity bits p2 and p3 give an error, which data bit is corrupted, and how can you correct it?

Question 11 (b)

b) If the parity bits p2 and p3 give an error, which data bit is corrupted, and how can you correct it?

if p2 and p3 give an error, that means d3 is corrupted.

In order to correct, you the corresponding parity bit and the other data bits used to compute that parity bit, i.e., for 1011:

$d3 = p2 \oplus d1 \oplus d4 = 1 \oplus 1 \oplus 1 = 1$

Question 12

Consider the asymmetric encryption algorithm. You are given two prime numbers:

$$p = 5 , \; q = 7$$

and assume the public key is given for you: Public key, $ke = 5$

Suppose we want to send the message, M=27 to you over the network.

a) How do we calculate the encrypted message (cyphertext)?

Remember

- Select k_e and k_d, where k_e satisfies $k_e k_d \bmod (p-1)(q-1) = 1$
- Calculate N = p * q
- Encryption algorithm is $E(k_e, N)(m) = m^{k_e} \bmod N$,
- Decryption algorithm is then $D(k_d, N)(c) = c^{k_d} \bmod N$

b) How would you calculate your private key?

c) How do you calculate the decrypted message (cleartext) from the cyphertext?

Question 13

Assume a disk with 500 cylinders is accessing and serving request on cylinder 100 right now. Prior cylinder 100, the disk head accessed cylinder 101. Further assume that the FIFO queue of pending requests is 102, 20, 450, 60, 80, 220, 330, 250, 101, 190. What order will the pending requests be satisfied using the following scheduling algorithms?

(a) Circular Scan disk-scheduling policy?

(b) SSTF disk-scheduling policy?

(c) Which of the above algorithms is more efficient in this particular case, and why?

Question 14 (a)

Consider the following page-reference string:

1, 2, 3, 4, 5, 6, 7, 8, 5, 6, 2, 3, 6, 2, 3, 7, 8, 3, 2, 1, 5, 6, 2, 4

(a) Show the page assignments to frames assuming Second Chance – "Clock" algorithm is used. Please fill in all frames. Consider the following rules:

1. When a page is brought to the memory the first time, initialize reference bit to 0.
2. Advance the next victim pointer only if you need to find a victim page to replace, and when you bring a new page in.

Question 14 (b)

(b) Calculate the following for the above page assignments:

- Number of page faults:
- Number of page hits:
- Number of page replacements:

Question 16

- Given the following memory partitions (in kilobytes): 200, 600, 500, 800, 400, 300 (in order); how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 292, 522, 138, 770, 162, 418 (in order).
- Which algorithm makes the most efficient usage of memory?