CSE 421/521 - Operating Systems
Fall 2014

Lecture - XXV

Final Review

Tevfik Koşar

University at Buffalo

December 2nd, 2014

Final Exam

December 4th, Thursday

11:00am - 12:20pm

Room: 110 Knox

Chapters included in Final

- Ch. 3.2-3.4 (Processes)
- Ch. 4.2-4.6 (Threads)
- Ch. 6.2-6.3 (CPU Scheduling)
- Ch. 5.2-5.8 (Synchronization)
- Ch. 7-2-7.7 (Deadlocks)

~ 20%

- Ch. 8.1-8.6 (Main Memory)
- Ch. 9.1-9.6 (Virtual Memory)
- Ch. 10.1-10.7 (Mass Storage & IO)
- Ch. 11.3; 12.1-12.5 (File Systems)
- Ch. 17 & Lect.notes (Distr. Systems)
- Ch. 15.1 - 15.5 (Security)

~ 80%

Processes

- Process Creation & Termination
- Context Switching
- Process Control Block (PCB)
- Process States
- Process Queues & Scheduling
- Interprocess Communication

Threads

- Concurrent Programming
- Threads vs Processes
- Threading Implementation & Multi-threading Models
- Other Threading Issues
  - Thread creation & cancellation
  - Signal handling
  - Thread pools
  - Thread specific data

CPU Scheduling

- Scheduling Criteria & Metrics
- Scheduling Algorithms
  - FCFS, SJF, Priority, Round Robing
  - Preemptive vs Non-preemptive
  - Gantt charts & measurement of different metrics
- Multilevel Feedback Queues
- Estimating CPU bursts

Synchronization

- Race Conditions
- Critical Section Problem
- Mutual Exclusion
- Semaphores
- Monitors
- Classic Problems of Synchronization
  - Bounded Buffer
  - Readers-Writers
  - Dining Philosophers
  - Sleeping Barber

Deadlocks

- Deadlock Characterization
- Deadlock Detection
  - Resource Allocation Graphs
  - Wait-for Graphs
  - Deadlock detection algorithm
- Deadlock Avoidance *(Bankers alg. excluded)*
- Deadlock Recovery

Main Memory

- 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, SC, 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

Security

- Security Violation Categories
- Security Violation Methods
- Program & Network Threats
- Cryptography
- Symmetric & Asymmetric Encryption
- Key distribution

Main Memory

- 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, SC, LFU, MFU, Optimal)
- Performance of Demand Paging

File Systems

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

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

Distributed Coordination

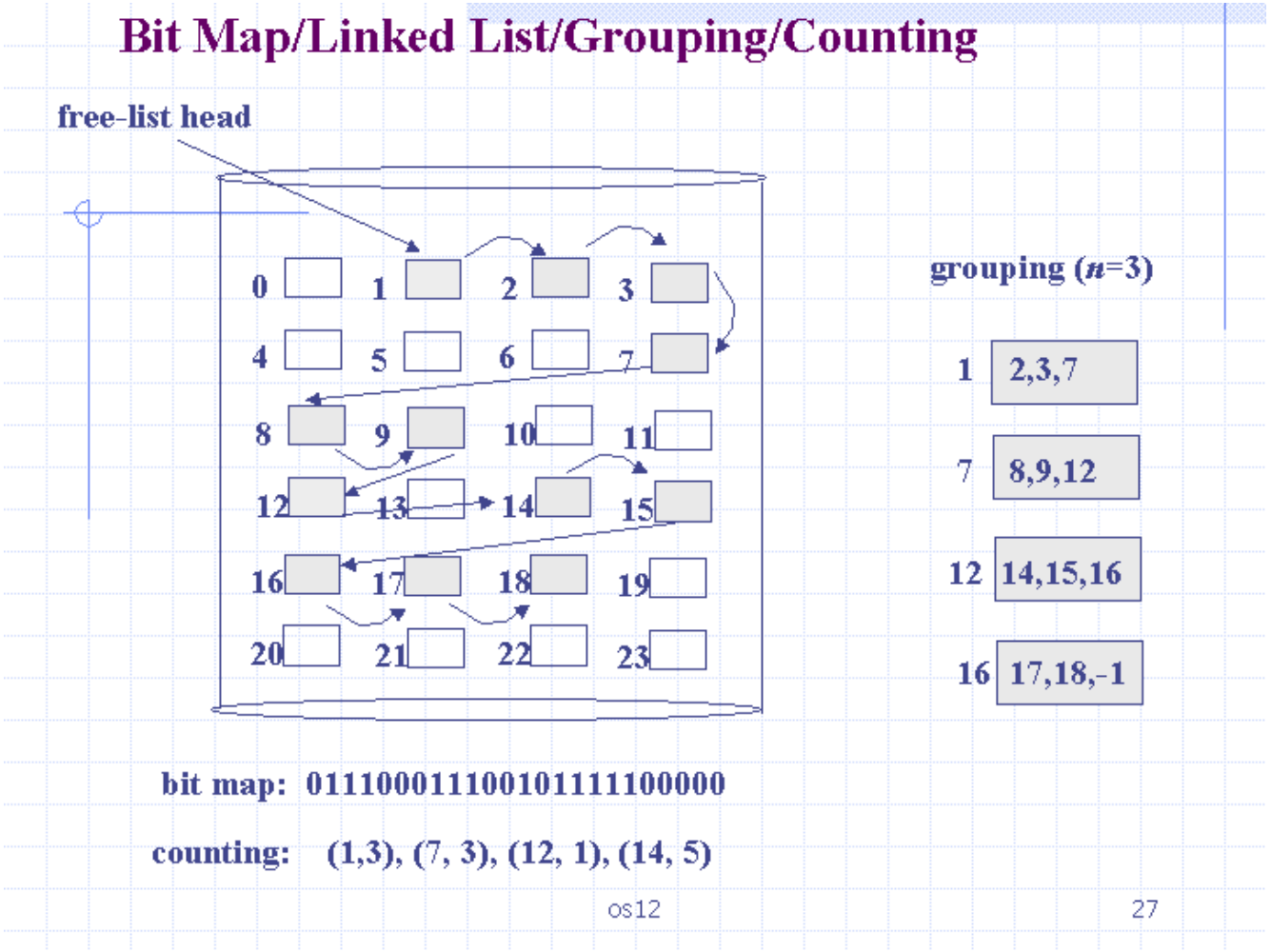
- Event Ordering
  - Happened before relationship
- Distributed Mutual Exclusion
  - Centralized & Fully Distributed Approaches
- Distributed Deadlock Prevention
  - Resource Ordering
  - Timestamp Ordering (Wait-die & Wound-wait)
- Distributed Deadlock Detection
  - Centralized & Fully Distributed Approaches

Exercise Questions

Question 1

- 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
free-list head
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
grouping (n=3)
1
2,3,7
7
8,9,12
12
14,15,16
16
17,18,-1
bit map: 01110001110010111110000
counting: (1,3), (7, 3), (12, 1), (14, 5)
os12
27

Solution 1

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 vectors, 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 2

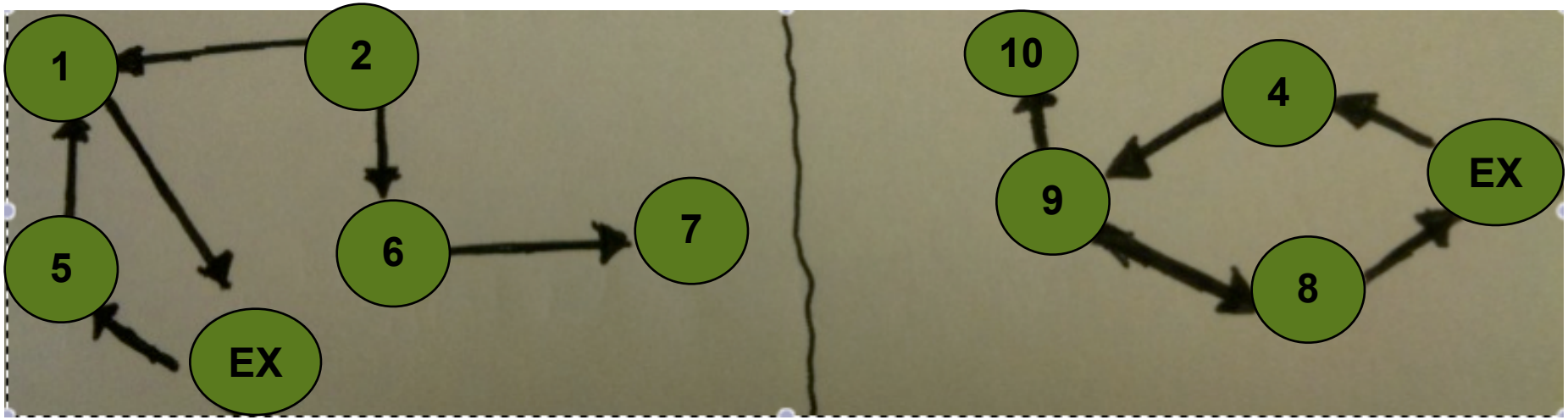
Does SSTF disk scheduling algorithm tend to favor accesses to innermost/middle/outermost cylinders of the disk?

Question 2

Does SSTF disk scheduling algorithm tend to favor accesses to innermost/middle/outermost cylinders of the disk?

Middle cylinders, since these cylinders are, on the average, closest to most other cylinders on the hard drive.

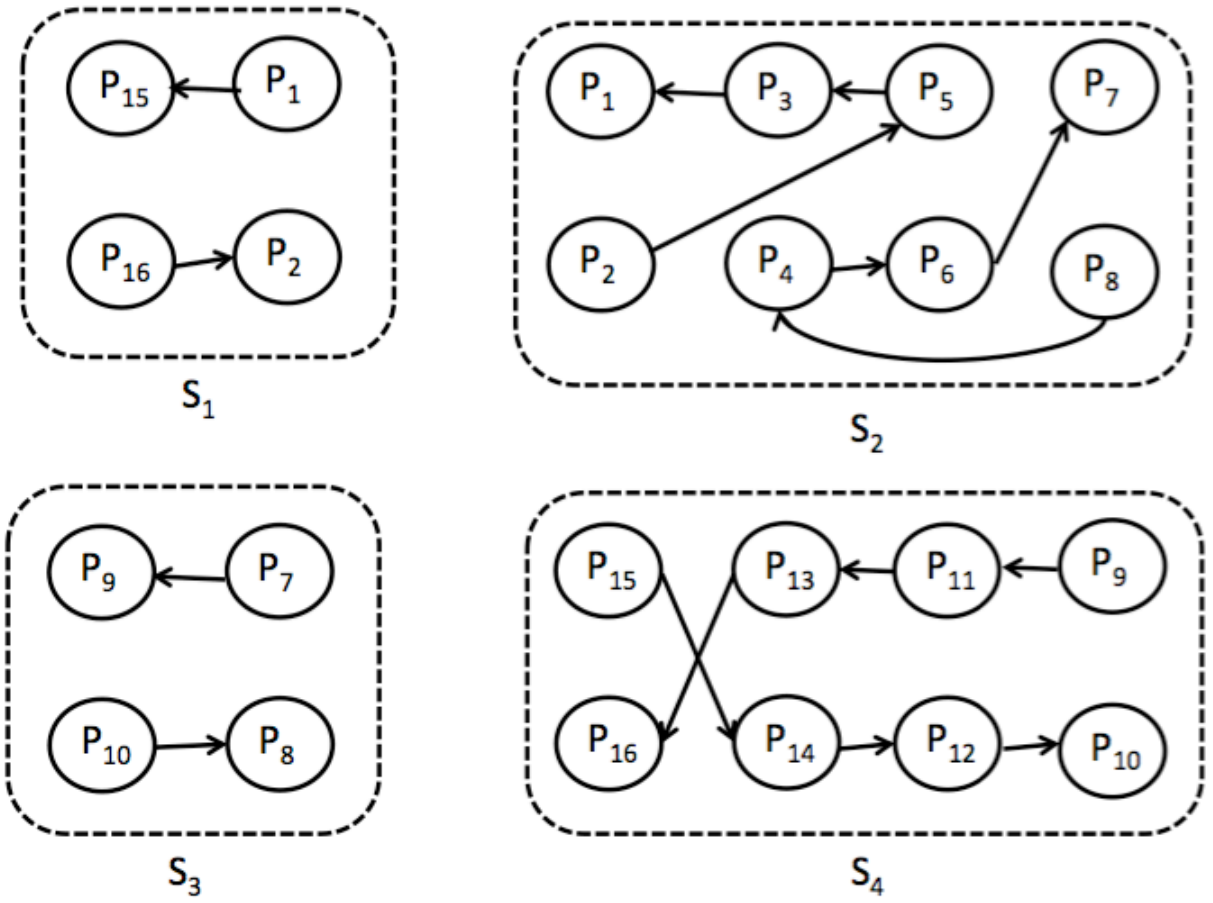
Question 3



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 4

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



Question 4 (a)

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

Question 4 (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.

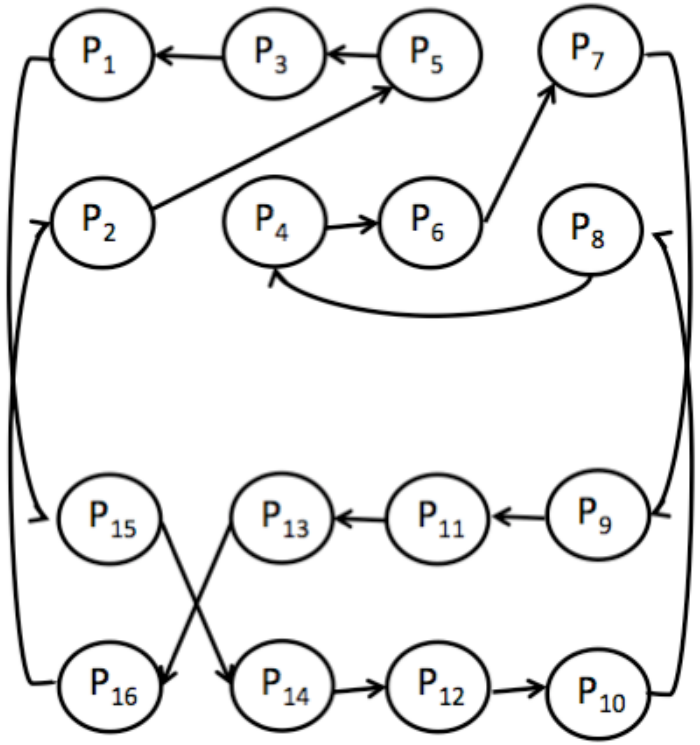
Question 4 (b)

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

Question 4 (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 5 (a)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

a) CPU utilization 86 percent; disk utilization 4 percent.

Question 5 (a)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

a) CPU utilization 86 percent; disk utilization 4 percent.

Answer: CPU utilization is sufficiently high to leave things alone (there are already sufficient processes running to keep the CPU busy); increasing the degree of multiprogramming may decrease the CPU utilization.

Question 5 (b)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

b) CPU utilization 10 percent; disk utilization 95 percent.

Question 5 (b)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

b) CPU utilization 10 percent; disk utilization 95 percent.

Answer: thrashing is occurring. We cannot increase the CPU utilization

Question 5 (c)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

c) CPU utilization 12 percent; disk utilization 2 percent.

Question 5 (c)

Consider a demand-paged computer system where the degree of multi-programming is currently fixed at four. The system was recently measured to determine utilization of CPU and the paging disk. The results are one of the following alternatives. For each case, what is happening (in one phrase)? Can you increase the degree of multiprogramming to increase the CPU utilization?

c) CPU utilization 12 percent; disk utilization 2 percent.

Answer: both CPU and disk utilization are low, and CPU is obviously underutilized. We should increase the degree of multiprogramming to increase CPU utilization.

Question 6 (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 6 (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 6 (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 6 (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 6 (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 6 (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 6 (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 6 (d)

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

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

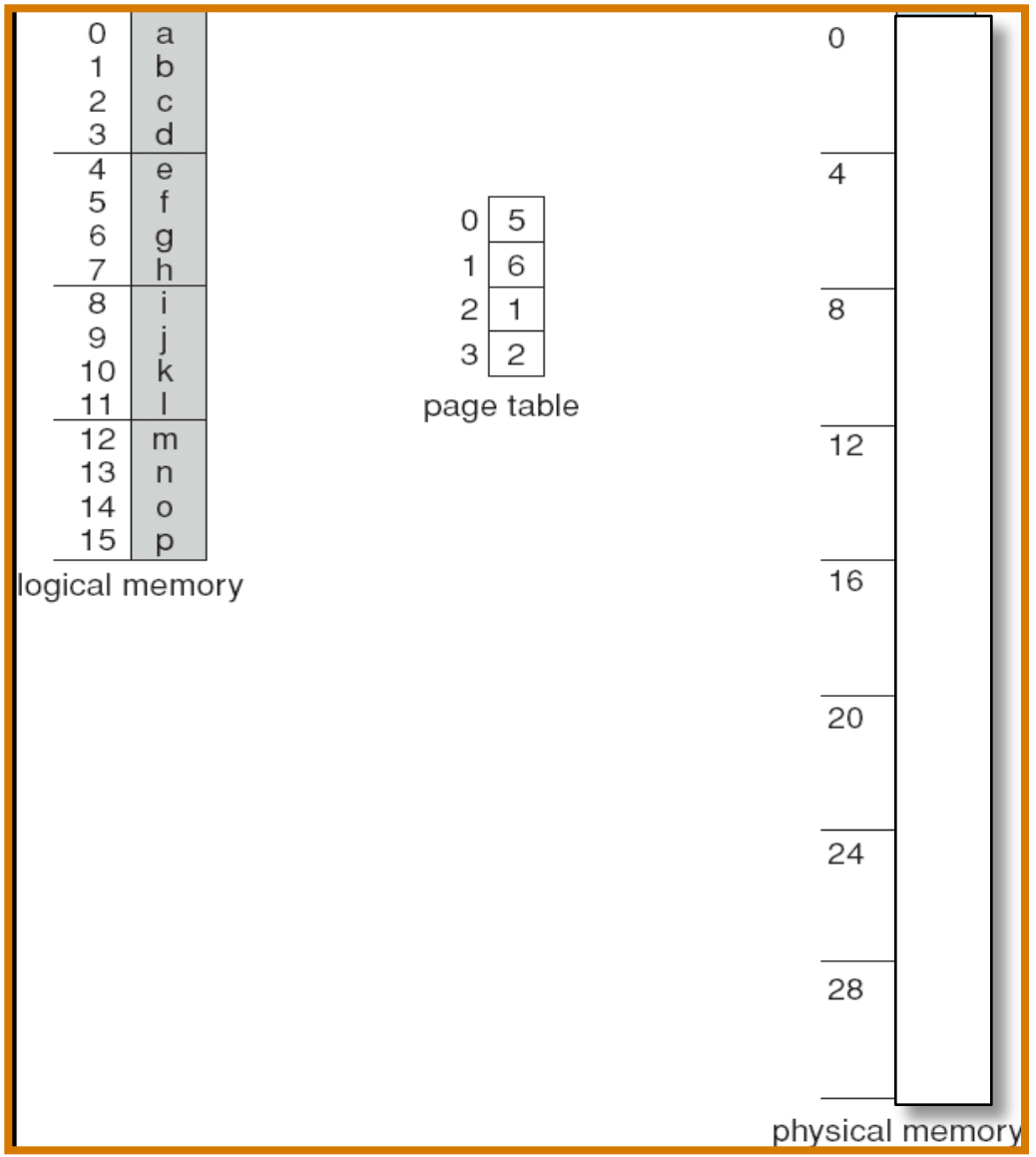
For each of the following, say whether it will (or is likely to) improve CPU utilization. Answer with YES or NO or LIKELY, and justify your answers.

(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).

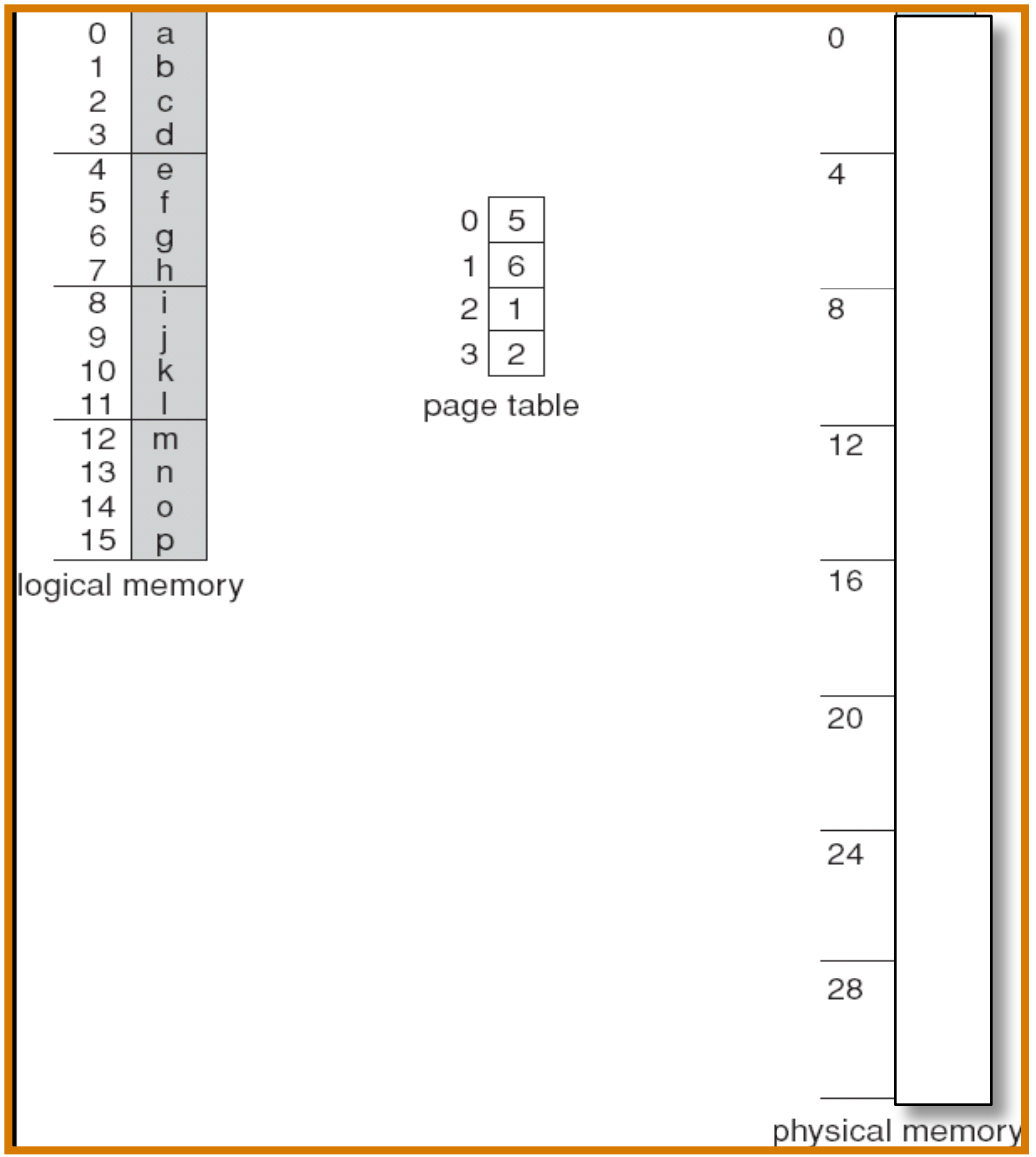
Question 7

- 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 7

- 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

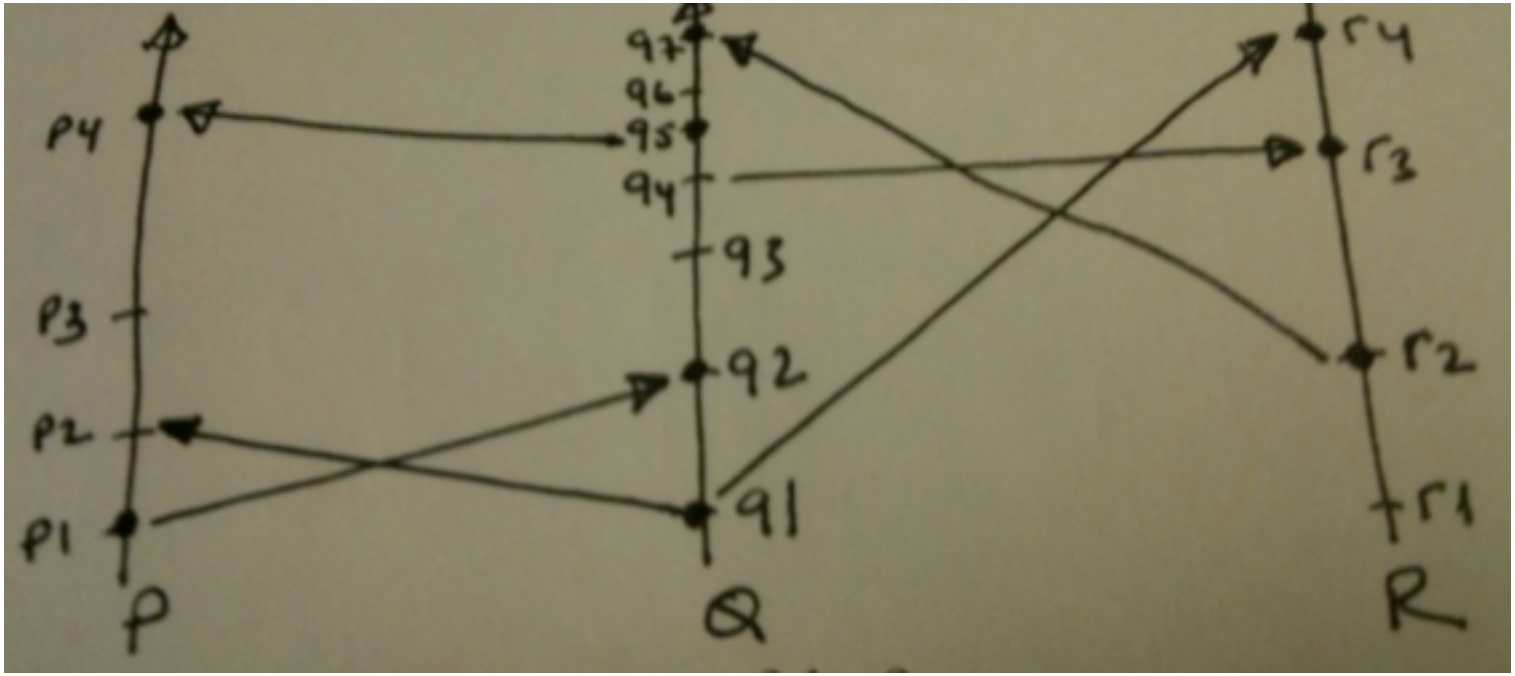


Question 8

Assume we have a demand-paged memory. It takes 100 ms to service a page fault if an empty page is available or the replaced page is not modified, and 300 ms if the replaced page is modified. Memory access time is 1 ms. Assume that the page to be replaced is modified 80 percent of the time.

What is the maximum acceptable page-fault rate for an effective access time of no more than 200% penalty?

Question 9



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 10

- 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?

Question 11

Assume a disk with 500 cylinders is accessing 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 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)?

b) How would you calculate your private key?

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