SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	**ITM 512**		**Title**	**Operating System Design**	
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**

1. Consider the following snapshot of a system: initial available = 1520

	Allocation	Max	Available	Need
	A B C D	A B C D	A B C D	A B C D
① P0	0 0 1 2	0 0 1 2	1 5 3 0	0 0 0 0
⑤ P1	1 0 0 0	1 7 5 0	3 14 12 12	0 7 5 0
② P2	1 3 5 4	2 3 5 6	2 8 8 6	1 0 0 2
③ P3	0 6 3 2	0 6 5 2	2 14 11 8	0 0 2 0
④ P4	0 0 1 4	0 6 5 6	2 14 12 12	0 6 4 2

a. Is the system in a safe state? Show the reason of your answer. (2 pts)

Need = Max – Available

Sequence : P1 – P2 – P3 – P4 – P1

Need
A B C D
0 0 0 0
0 7 5 0
1 0 0 2
0 0 2 0
0 6 4 2

b. Can the request be granted immediately, if a request from P0 arrives for (0,3,2,0)? (2 pts)

No | Max < (0, 3, 2, 0)

req (0,3,2,0) < available (1,5,3,0) … ok.
req (0,3,2,0) > need P0 (0,0,0,0) … No.

c. Can the request be granted immediately, if a request from P1 arrives for (0,5,0,0)? (3 pts)

Yes | Max > (0, 5, 0, 0)

req (0,5,0,0) < available (1,5,3,0) … ok
req (0,5,0,0) < need P_1 (0,7,5,0) … ok

initial available = 1020

	allocation	max	need	available
P_0	0012	0012	0000	① 1032
P_1	1500	1750	0250	⑤ 3 14 12 12
P_2	1354	2356	1002	② 2 3 8 6
P_3	0632	0652	0020	③ 2 9 11 8
P_4	0014	0656	0642	④ 2 9 12 12

… ok

SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	**ITM 512**	**Title**	**Operating System Design**		
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**

2. Compare the following three memory allocation methods, contiguous memory allocation, pure segmentation, and pure paging, in terms of the following issues:

a. External fragmentation

unused, scattered
⇒ "hole" ... dynamic storage-allocation problem. { not enough space for new process.

External fragmentation occurs in contiguous memory allocation and pure segmentation because memory is allocated in contiguous blocks, leading to small gaps of unused memory between allocated segments. Pure paging does not suffer from external fragmentation as memory is divided into fixed-size pages that can be allocated non-contiguously.

page : logical memory block of fixed size
non-contiguous allocation allowed

b. Internal fragmentation

allocation. last page < last frame.
fixed-size memory block > requested mem space

Internal fragmentation occurs in pure paging because allocated memory may be slightly larger than requested, leading to wasted space within allocated pages. Contiguous memory allocation and pure segmentation can also suffer from internal fragmentation if allocated segments are larger than needed.

requested < allocated.
fixed-size partition

c. Ability to share codes across processes

Pure segmentation allows easy sharing of code across processes by referencing the same segment in different segment tables. Pure paging also allows code sharing by mapping the same physical pages into different page tables. Contiguous memory allocation is less flexible for sharing code as it requires contiguous blocks of memory.

segmentation divide process address space by segment with logical unit (code, data, stack, ...)
Through GDT (Global Descriptor Table), the processes can share own segments
and access external segments.

physical memory space can be non-contiguous.
code sharing by mapping the same physical frame to different page tables.
read-only.
read-write

each process allocated in a single contiguous section of memory
also, base register & limit register defines process' logical address range.
cannot access other process' address space in ordinary ways.

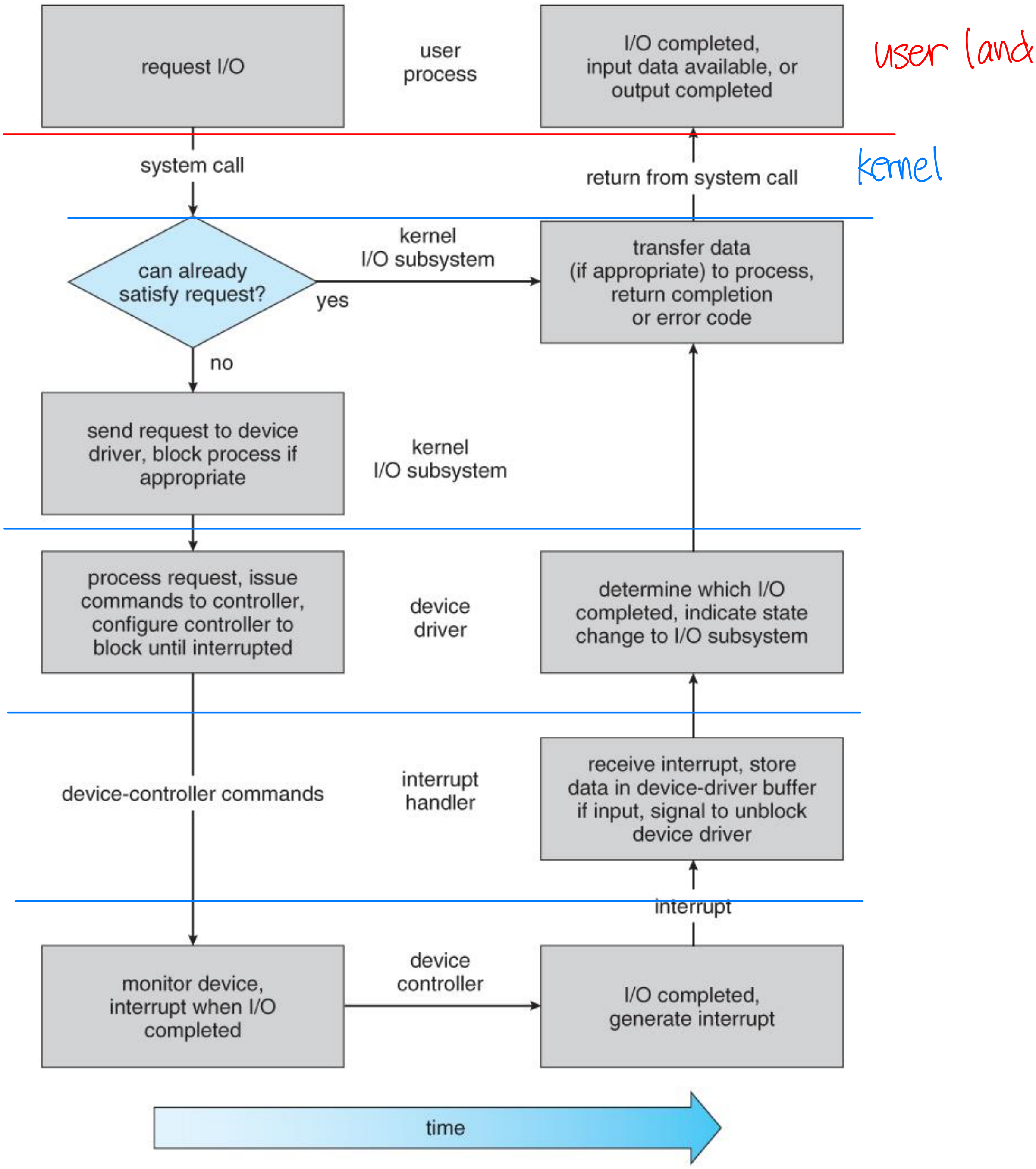
SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	ITM 512		Title	Operating System Design	
Time for Exam	120 min	Questions	14	Weighting	40%

3. The following figure depicts the life cycle of an I/O operation requested by a user process, from the request I/O step to the I/O completed step. Fill the blanks to complete the explanation of the life-cycle. (10pts)



SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	ITM 512		Title		Operating System Design
Time for Exam	120 min	Questions	14	Weighting	40%

4. Why does a threshing occur? (3pts) How can an operating system detect a threshing? (2pts) How does an operating system handle a threshing when it is detected? (2pts)

a. Why does a thrashing occur? (3pts) not enough pages, page fault rate very high.
replace existing frame, quickly need replaced frame back.

Thrashing occurs when a system spends more time swapping pages in and out of memory than executing processes, due to excessive paging.

Low CPU utilization. OS increase the degree of multiprogramming
Another program added.

- process doesn't have enough pages → page fault rate is very high
- page fault → replace frame → quickly need replaced frame back

busy swapping pages in & out.

⇒ low cpu utilization

spend much more time for swapping rather than executing process.

Thrashing occurs when system spends more time swapping in and out of memory than executing process, due to excessive paging.

Process doesn't have enough pages, so page fault rate is so high. If page fault happens, frame should be replaced. But it should be replaced back quickly. Cpu utilization is low.

b. How can an operating system detect a thrashing? (2pts) tracking page-fault frequency, PFF is higher than certain value, then thrashing.

It can be detected by monitoring high paging activity and low CPU utilization.

Page-fault increasing, but CPU utilization decreasing
⇒ thrashing

Monitoring paging activity and check low cpu utilization.

WSS_i = working set of process i
= total # pages referenced in the most recent Δ time

$D = \sum WSS_i > m$ = total available physical frames

c. How does an operating system handle a thrashing when it is detected? (2pts)

To handle thrashing, an operating system can reduce the degree of multiprogramming or use better page replacement algorithms. ⇒ local replacement → acceptable PFF
priority allocation.

↳ some process swap out
suspend.

memory demand < total available memory

Os can reduce degree of multiprogramming or use better page replacement algorithm.

SEOULTECH
Seoul National University of
Science & Technology

Final Exam

Date : 5.31

Code	ITM 512		Title	Operating System Design	
Time for Exam	120 min	Questions	14	Weighting	40%

5. What is buffering? (3pts) Show three reasons why buffering is used. (2pts) What is the difference of the buffering from caching? (2pts)

a. What is buffering? (3pts)

Buffering is the process of storing data in memory while transferring between devices or

Buffering is the process of storing data in memory while transferring between devices.

b. Show three reasons why buffering is used. (2pts)

- cope with device speed mismatch
- cope with transfer size mismatch
- maintain copy semantics

Cope with device speed mismatch

Cope with transfer size mismatch

Maintain copy semantics

c. What is the difference of the buffering from caching? (2pts)

Buffering differs from caching, which stores frequently accessed data to speed up future access.

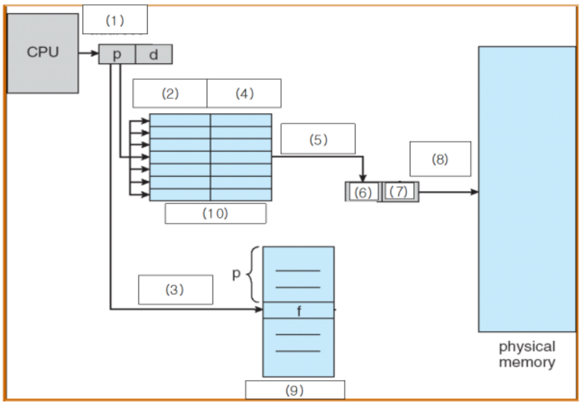
SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	ITM 512		Title	Operating System Design	
Time for Exam	120 min	Questions	14	Weighting	40%

6. The following figure depicts the procedure of accessing memory by CPU in paging hardware with TLB. Fill up (1)~(10). (1pt each)



Logical addr / Page number / TLB miss / Frame number / TLB hit / F / D / Physical addr / Page table

7. Explain blocking I/O (2pt), non-blocking I/O (2pt), and asynchronous I/O (2pt), respectively.

Blocking I/O waits for the operation to complete before returning control to the program.

- process suspended until io completed
- easy to use and understand
- insufficient for some needs

Process suspended until io completed.
Easy to understand and use
Insufficient for some needs

Non-blocking I/O returns immediately, allowing the program to continue while the operation completes.

- io call returns as much as available
- ui, data copy = buffered io
- implement via multithreading
- returns quickly with counts of bytes read / written

Io call returns as much as available
Ui and data copy is buffered io

Final Exam

Date : 5.31

Code	ITM 512	Title	Operating System Design		
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**

Implement via multithreading
Returns quickly with counts of bytes read and written

Asynchronous I/O allows the program to continue and notifies it when the operation is complete.

- process runs while io executes
- difficult
- io subsystem signals process when io completed

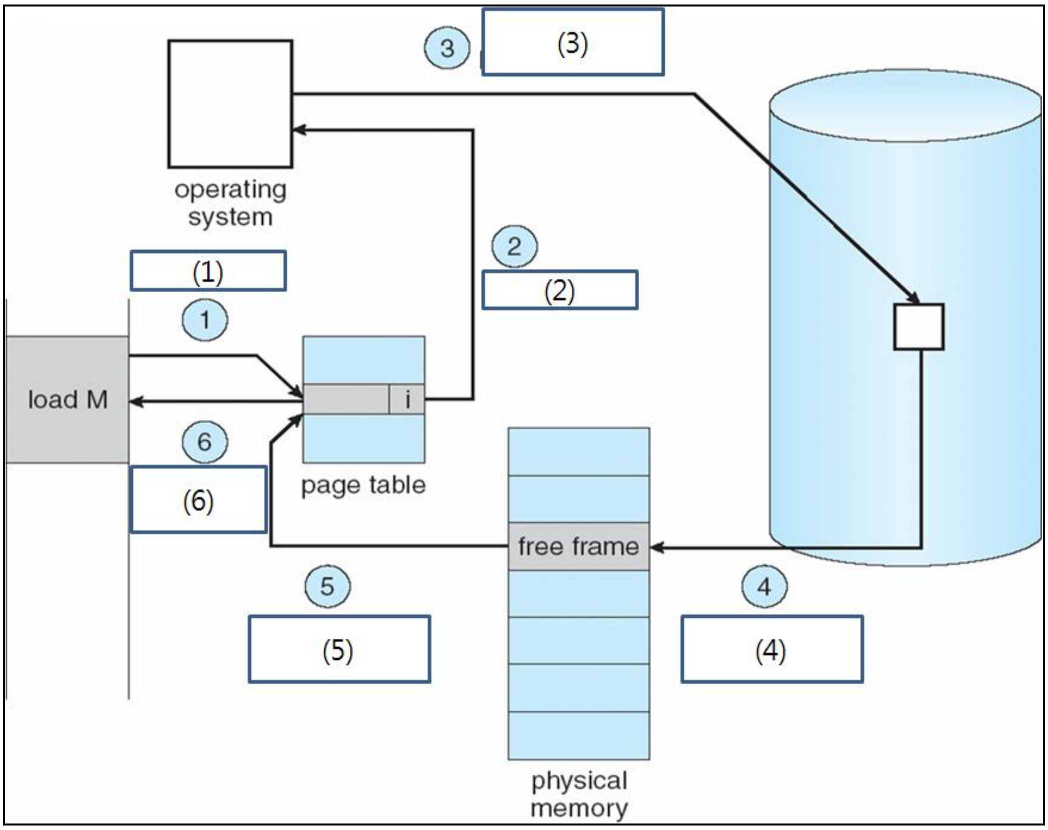
Process runs while io executes
Difficult
Io subsystem signals process when io completed

8. The following figure shows the procedure of handling a page fault event. Fill the blanks. (6pts)

Final Exam

Date : 5.31

Code	**ITM 512**		**Title**	**Operating System Design**	
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**



Reference / Trap / page is on backing store / bring in missing page / reset page table / restart instruction

9. Explain FAT as in detail as possible. What is the advantage of FAT compared to the other methods you learned in the class? (8pts)

The File Allocation Table (FAT) is a simple file system used in various operating systems. It uses a table to keep track of file locations on disk. Each entry corresponds to a cluster and contains the address of the next cluster or an end-of-file marker. Advantages of FAT include simplicity, wide compatibility, and low overhead. It is suitable for smaller disks and embedded systems but lacks advanced features like journaling and access control found in modern file systems.

Fat uses a table to keep track of file locations in disk
Simplicity, wide compatibility, low overhead

10. LRU method works very well as a page replacement algorithm. However, it is not used as it is. Explain why (4pts) and show the solution to handle the problem. (4pts)

a. Why

SEOULTECH
Seoul National University of
Science & Technology

Final Exam

Date : 5.31

Code	**ITM 512**		**Title**	**Operating System Design**	
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**

The Least Recently Used (LRU) page replacement algorithm works well by evicting the least recently accessed pages. However, it is costly to implement due to the need to maintain a precise order of page accesses.

b. Solution

Approximate LRU algorithms, like the Clock algorithm, are used to reduce overhead while providing similar performance.

11. Why does a modern operating system utilize a virtual memory system? Show the reason. (7pts)

- logical > physical
- addr spaces can be shared by several processes
- more efficient process creation
- more process running concurrently
- less I/O to load / swap

Address space can be shared by several processes
More efficient process creation
More process run concurrently
Less io to load or swap

SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	**ITM 512**		**Title**	**Operating System Design**	
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**

12. Consider the page table shown below for a system with 12-bit virtual and physical addresses and 256-byte pages. The list of free page frames is D, E, F (that is, D is at the head of the list, and F is last). Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal. (A dash for a page frame indicates that the page is not in memory.)

Page	Page Frame
0	-
1	A
2	C
3	2
4	-
5	B
6	3
7	-
8	4
9	0

a. 5EF (2pts)

Page Number: 5
Offset: EF
Page Frame: B
Page Frame B in hexadecimal is mapped directly.
Therefore, the physical address is B followed by the offset EF.
Physical Address: BEF

b. 311 (2pts)
Page Number: 3
Offset: 11
Page Frame: 2
Page Frame 2 in hexadecimal is mapped directly. Therefore, the physical address is 2 followed by the offset 11.
Physical Address: 211

c. 013 (2pts)

Page Number: 0
Offset: 13
Page Frame: -
Page Frame for page number 0 is not in memory (indicated by -). Hence, this address is invalid.
Physical Address: Not in memory

d. 4AB (2pts)

Page Number: 4
Offset: AB
Page Frame: -
Page Frame for page number 4 is not in memory (indicated by -). Hence, this address is invalid.
Physical Address: Not in memory

SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	**ITM 512**		**Title**	**Operating System Design**	
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**

13. Suppose the stack-based LRU is utilized and the number of frame pages is three, which is numbered 1~3, respectively. On the following sequence of the page references, show the state of the stack at every page reference including the page # that is assigned to each page frame. (7pts)

* Page reference sequence: 1 3 2 6 4 5 2 1 5 4 1 2 3 1 3

1: [1]
3: [3, 1]
2: [2, 3, 1]
6: [6, 2, 3]
4: [4, 6, 2]
5: [5, 4, 6]
2: [2, 5, 4]
1: [1, 2, 5]
5: [5, 1, 2]
4: [4, 5, 1]
1: [1, 4, 5]
2: [2, 1, 4]
3: [3, 2, 1]
1: [1, 3, 2]
3: [3, 1, 2]

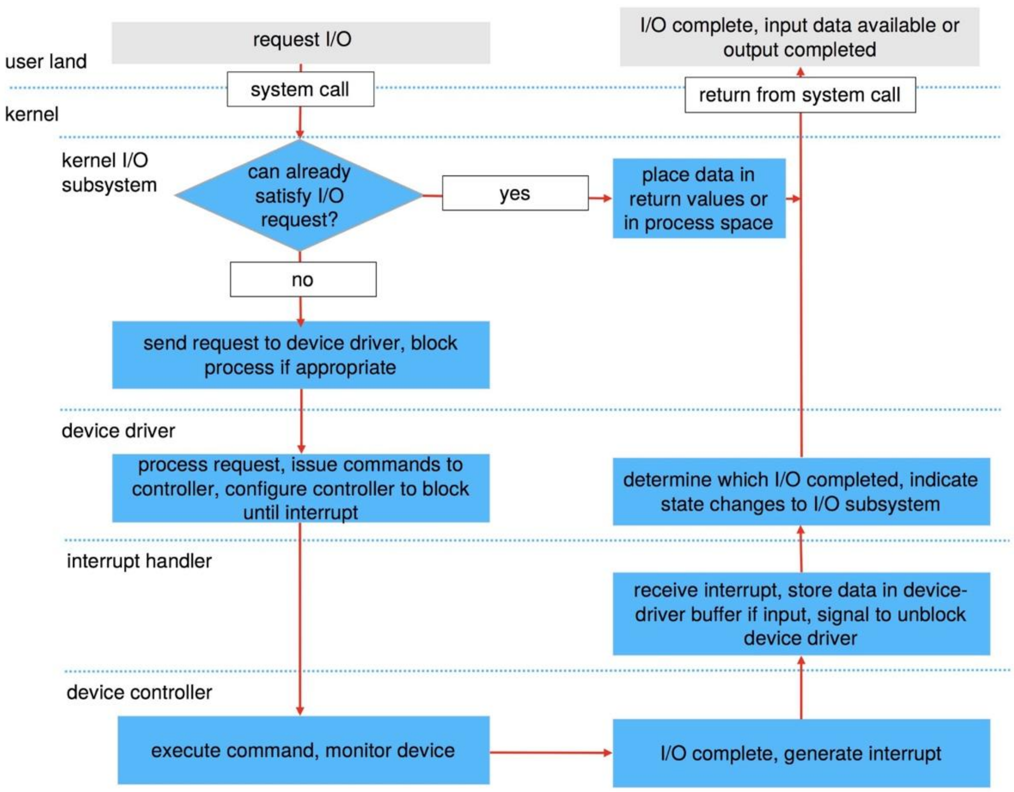
SEOULTECH
Seoul National University of Science & Technology

Final Exam

Date : 5.31

Code	**ITM 512**		**Title**	**Operating System Design**	
Time for Exam	**120 min**	**Questions**	**14**	**Weighting**	**40%**

14. The following figure depicts the life cycle of an I/O operation requested by a user process, from the request I/O step to the I/O completed step. Fill the blanks to complete the explanation of the life-cycle. (7 pts)



Io completed, input available or output completed
Place data in return values or in process space
Determine which io completed, indicate state changes to io subsystem
Receive interrupt, store data in device drive buffer if input, signal to unblock device driver

Determine which io completed, indicate state changes to io subsystem
Receive interrupt, store data in device driver buffer if input, signal to unblock device driver

Determine which io is completed, indicate state changes to io subsystem

Indicate state changes to io subsystem