

CS2106 Final Assessment
2023/24 Semester II
Final Assessment

Part 1. MCQ (20 MARKS)

Choose the best option for each of the following questions. Each question has exactly one answer.

1. A logical memory address [2 marks]
 - A. Allows direct access to the physical RAM location.
 - B. Is visible to the programmer.
 - C. Is address after translation by the hardware.
 - D. Is generated by the operating system.
 - E. None of the above A. to D.

Ans. B

2. Memory fragmentation is [2 marks]
 - A. The existence of usable areas in the memory of a computer system.
 - B. The existence of usable areas within a memory partition of a computer system.
 - C. The existence of unreachable area in the memory of computer system.
 - D. The existence of unusable areas in the memory of a computer system.
 - E. None of the above.

Ans. D

3. Consider a buddy allocation system. Which of the following pairs of addresses belongs to buddy blocks of size 1KiB? [2 marks]
 - A. 0xF123, 0xF823
 - B. 0xF423, 0xFC23
 - C. 0xF034, 0xF234
 - D. 0xF234, 0xD234
 - E. None of the above

Ans. B.

1 KiB = 2^{10} => differ 10th bit. Two pairs of buddy block starting address =>
 1111 0000 0000 0000 (0xF000) and 1111 0100 0000 0000 (0xF400)
 1111 1000 0000 0000 (0xF800) and 1111 1100 0000 0000 (0xFC00)

0xF423 and 0xFC23 both belong to the 0xF400 and 0xFC00 buddy block.

4. Following up from the previous question, what are the starting addresses of the buddy blocks to which your selected pairs of addresses belong? [2 marks]
 - A. 0xF000, 0xF800
 - B. 0xF200, 0xF800
 - C. 0xF000, 0xFC00
 - D. 0xF000, 0xF200
 - E. None of the above

Ans. A. (Explained in previous solution)

5. Consider a machine with 64 MiB physical memory and 32-bit virtual address space. If the page size is 4 KiB, the size of direct page table and inverted page table will be? [2 marks]

- A. Direct: 2^{20} , Inverted: 2^{20}
- B. Direct: 2^{20} , Inverted: 2^{14}
- C. Direct: 2^{14} , Inverted: 2^{20}
- D. Direct: 2^{14} , Inverted: 2^{14}
- E. Direct: 2^{20} , Inverted: 2^{13}

Ans: B.

$$\text{Direct page table} = 2^{32} / 2^{12} = 2^{20}$$

$$\text{Inverted page table} = 2^{26} / 2^{12} = 2^{14}$$

6. Consider the following scenario that happens in order: (1) Process A opens two files File1.txt and File2.txt, (2) Process A then forks a child process B, (3) Process B opens File3.txt, (4) Process B forks a child process C, and (5) Process C close File1.txt. What are the values of the reference count for the files, File1.txt, File2.txt, and File3.txt in the system-wide open file table? [2 marks]

- A. File1.txt: 3 ; File2.txt: 3 ; File3.txt: 3
- B. File1.txt: 3 ; File2.txt: 3 ; File3.txt: 2
- C. File1.txt: 2 ; File2.txt: 3 ; File3.txt: 2
- D. File1.txt: 3 ; File2.txt: 2 ; File3.txt: 3
- E. File1.txt: 2 ; File2.txt: 3 ; File3.txt: 3

Ans: C

After step 1; File1. txt and File2.txt -> reference count 1 each

After step 2; File1. txt and File2.txt -> reference count 2 each

After step 3; File1. txt and File2.txt -> reference count 2 each File3.txt -> reference count 1

After step 4; File1. txt and File2.txt -> reference count 3 each , File3.txt -> reference count 2

After step 5; File1. txt -> 2 reference count, File2.txt -> reference count 3 each File3.txt -> reference count 2

7. Consider a combined index allocation scheme implemented for a file system. For simplicity, consider a single disk block can have only 2 entries. Let's assume the following I-node structure that contains 5 disk block pointers: 2 direct block pointers, 1 single indirect data block pointer, 1 double indirect pointer, and 1 triple indirect pointer. What is the maximum size of a file, in terms of the number of blocks that can be stored in this I-node structure? [2 marks]

- A. 2 blocks
- B. 4 blocks
- C. 8 blocks
- D. 16 blocks
- E. None of the above

Ans: D.

Direct – 2 ; single – 2; double – 4; triple – 8 ; total =16

8. Which of the following statements is false regarding the directory structure in a file system? [2 marks]

- A. Represents the smallest unit of storage of data.

- B. Organize files into a hierarchical structure for easier management.
- C. Provide a way to navigate and access files and directories.
- D. Allow for efficient storage and retrieval of file metadata.
- E. All statements are true.

Ans. A

9. Let the time taken to switch between user and kernel modes of execution be t_1 while the time taken to switch between two processes be t_2 . Which of the following is true? [2 marks]

- A. $t_1 = t_2$
- B. $t_1 > t_2$
- C. $t_1 < t_2$
- D. $t_1 \geq t_2$
- E. Cannot say

Ans: C. Since switching between two processes requires switching from user mode to kernel mode, updating the PCB and registers, then switching back from kernel model to user mode.

10. Consider a memory snapshot in a point of time contains five memory partitions of size 75KB (allocated to P1), 60 KB (free), 100 KB (allocated to P2), 200 KB (free), and 50 KB (allocated to P3). Process requests come in the order: 25K, 75K, free P1, 50K, 75K. Which of the following algorithms results in the largest free contiguous memory space, after satisfying all the process requests? Assume that the adjacent free locations can be merged. [2 marks]

- A. First Fit
- B. Worst Fit
- C. Best Fit
- D. All three algorithms result in the same largest contiguous free space.
- E. No contiguous memory space is free after allocation.

Ans. B

FF :	75(P1), 60(F), 100 (P2), 200(F), 50(P3)
25k:	75(P1), 25(O), 35(F), 100 (P2), 200(F), 50(P3)
75K:	75(P1), 25(O), 35(F), 100 (P2), 75(O), 125(F), 50(P3)
Free p1:	75(F), 25(O), 35(F), 100 (P2), 75(O), 125(F), 50(P3)
50K:	50(O), 25(F), 25(O), 35(F), 100 (P2), 75(O), 125(F), 50(P3)
75K:	50(O), 25(F) , 25(O), 35(F) , 100 (P2), 75(O), 75(O), 50(F) , 50(P3)
BF :	75(P1), 60(F), 100 (P2), 200(F), 50(P3)
25k:	75(P1), 25(O), 35(F), 100 (P2), 200(F), 50(P3)
75K:	75(P1), 25(O), 35(F), 100 (P2), 75(O), 125(F), 50(P3)
Free p1:	75(F), 25(O), 35(F), 100 (P2), 75(O), 125(F), 50(P3)
50K:	50(O), 25(F), 25(O), 35(F), 100 (P2), 75(O), 125(F), 50(P3)
75K:	50(O), 25(F) , 25(O), 35(F) , 100 (P2), 75(O), 75(O), 50(F) , 50(P3)
WF :	75(P1), 60(F), 100 (P2), 200(F), 50(P3)
25k:	75(P1), 60(F), 100 (P2), 25(O), 175(F), 50(P3)
75K:	75(P1), 60(F), 100 (P2), 25(O), 75(O), 100(F), 50(P3)
Free p1:	135K [75(F) , 60(F)- merge], 100 (P2), 25(O), 75(O), 100(F), 50(P3)
50K:	50(O), 85(F), 100 (P2), 25(O), 75(O), 100(F), 50(P3)
75K:	50(O), 85(F) , 100 (P2), 25(O), 75(O), 75(O), 25(F) , 50(P3)

Part 2. Short Questions (60 MARKS)**Question 1.** Synchronization (10 MARKS)

- a. (4 marks) We have the following two processes updating a variable x, whose starting value is 2.

Process A	Process B
$x = x + 1$	$x = x - 3$
$x = x / 2$	

How many possible values of x IN TOTAL are there?

We have AL1, AS1, AL2, AS2, BL, BS

AL1AS1AL2AS2:

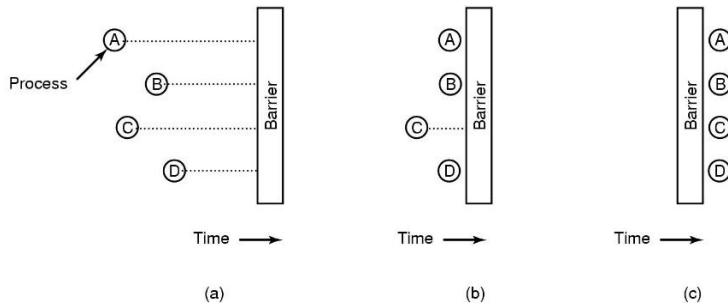
BL has 5 possible positions

In each of these positions, BS has 1, 2, 3, 4, 5 possible positions

Total = $1 + 2 + 3 + 4 + 5$ positions = 15 positions

1	AL1	2	AL1	3	AL1	4	AL1	5	AL1	6	AL1	7	AL1
	AS1		AS1		AS1		AS1		AS1		AS1		BL
	AL2		AL2		AL2		BL		BL		BL		AS1
	AS2		BL		BL		AL2		AL2		BS		AL2
	BL		AS2		BS		AS2		BS		AL2		AS2
	BS		BS		AS2		BS		AS2		AS2		BS
8	AL1	9	AL1	10	AL1	11	BL	12	BL	13	BL	14	BL
	BL		BL		BL		AL1		AL1		AL1		AL1
	AS1		AS1		BS		AS1		AS1		AS1		BS
	AL2		BS		AS1		AL2		AL2		BS		AS2
	BS		AL2		AL2		AS2		BS		AL2		AL2
	AS2		BL2		BL2		BS		AS2		AS2		AS2
15	BL												
	BS												
	AL1												
	AS1												
	AL2												
	AS2												

- b. (6 marks) A barrier is a synchronization mechanism for n processes. Each process will call the barrier. For as long as fewer than n processes have called the barrier, all processes will block. When the n^{th} process calls the barrier, all processes are unblocked simultaneously. The diagram below shows this idea with a barrier for 4 processes.



In figures a. and b., processes call the barrier and are blocked, while in figure c., the 4th process finally calls the barrier and all are unblocked simultaneously.

We are given the following functions that set the number of processes, and clear the number of processes that have called the barrier:

```
// numProcesses = # of processes to call the barrier before
// it unblocks. processesCalled = # of processes that have
// called the barrier.
int numProcesses = 0, processesCalled = 0;

void setBarrierN(int n) {
    numProcesses = n;
}

void clearBarrierCount() {
    processesCalled = 0;
}
```

You are given the following semaphore operations:

```
void initializeSema(TSema *s, int v); // Set semaphore s to initial value v
void downSema(TSema *s); // Down the semaphore
void upSema(TSema *s); // Up the semaphore
```

Complete the following code:

// Declare your semaphores here. Semaphores are of type TSema (1 mark)

TSema mutex, count;

// Initialize your semaphores (2 marks)
void initSema() {

initializeSema(&mutex, 1);

```
initializeSema(&count, 0);

}

// Implement the barrier here (3 marks)
// Rubrics: Give 3 marks if students display this general idea
// Solution DOES NOT have to be perfect – e.g. forgot mutex to
// protect processesCalled, etc.
// 1) when processesCalled < numProcesses:
//     - Increment processesCalled
//     - down(count) – Causes process to block
//     - up(count) – Unblock next process after this process is unblocked.
// 2) When processesCalled >= numProcesses
//     - Up(count) – Unblock the last process to allow it to unblock others.

// 2 marks if idea is mostly there, 1 mark if the person wrote something remotely
// sensible.

void barrier() {
    int n;
    downSema(&mutex);

    if(processesCalled >= numProcesses) {
        upSema(&mutex);
        return;
    }

    if(processesCalled < numProcesses) {
        processesCalled++;
        n = processesCalled;
    }
    upSema(&mutex);
    if n < numProcesses
        downSema(&count);

    upSema(&count);

}
```

Question 2. Segmented and Disjoint Memory Models (20 MARKS)

In a particular operating system, the text, data, stack and heap segments are laid out next to each other in the process memory space as shown below:

Segment
Code (C)
Data (D)
Stack (S)
Heap (H)

- a. (2 marks) We are given the following limit values for the segments (as well as the base value for the code segment), where the Limit values are the length of the respective segment in bytes, and Base is the starting address of the segment. Given again that the segments are adjacent to each other in the memory, calculate the base addresses of each segment (the base address of the code segment is already given).

Segment	Base	Limit
C	1000	3842
D		286
S		512
H		1024

ANSWER:

Segment	Base	Limit
C	1000	3842
D	4842	286
S	5128	512
H	5640	1024

- b. (2 marks) An array B containing 32-bit integers has its first element at (D, 100). Compute the addresses of the following elements, indicating N/A if it results in an access violation:

Array Element	Address (or NA if access violation)
A[4]	
A[20]	
A[26]	
A[31]	

ANSWER:

Element A[0] is at address (D, 100) = 4842 + 100 = 4942. Largest possible address is 4842 + 286 - 1 = 5127

Array Element	Address (or NA if access violation)
A[4]	4942 + 4 x 4 = 4958
A[20]	4942 + 20 x 4 = 5022
A[41]	4942 + 41 x 4 = 5106
A[50]	N/A 4942 + (50 x 4) = 5142

Our segmented memory system is now implemented on a virtual memory system. Each address that you calculated earlier is now a virtual address that must be translated to a physical address. Our system has 512-byte pages and frames.

- c. (2 marks) In general, operating systems would not place two different segments in the same page. Explain briefly why (hint: This is particularly true with the code and data segments).

ANSWER:

Two segments may need different permissions. For example the code segment is read only, while the data segment is read/write.

- d. (8 marks) Given that two segments cannot be in the same page, we now move each segment to the next adjacent page if they are currently occupying the same page as the previous segment. For example, if both the code and data segments are in page 3, then the data segment will be moved to the start of page 4.

Complete the following table, calculating the base address, starting page and ending page of each segment.

Note: Unlike what is shown in the lecture notes, there is only one page table used for all four segments. Virtual address 0 starts at page 0.

Segment	Base	Limit	Starting Page	Ending Page
C	1000	3842		
D		286		
S		512		
H		1024		

ANSWER:

Page	Starting Address
0	0
1	512
2	1024
3	1536
4	2048
5	2560
6	3072
7	3584
8	4096
9	4608
10	5120
11	5632
12	6144

Segment	Base	Limit	Starting Page	Ending Page
C	1000	3842	1	9
D	5120	286	10	10
S	5632	512	11	11
H	6144	1024	12	13

- e. (8 marks) Compute the internal fragmentation of your layout in part d.

ANSWER:

Segment	Starting Page	Ending Page	# of pages	Total Size (Bytes)	Leftover
C	1	9	9	4608	4608 – 3842 = 766
D	10	10	1	512	512 – 286 = 226
S	11	11	1	512	512 – 512 = 0
H	12	13	2	1024	1024 – 1024 = 0
Total					766+226 = 992

992 bytes.

Question 3. Virtual Memory (20 MARKS)

Consider a byte-addressed virtual memory system with the following set up: (i) virtual addresses are 64 bits long, (ii) physical addresses are 48 bits long, (iii) a page is 8KiByte, (iv) each page table is a page in length, (v) page table entries are 8 bytes, and (vi) it uses 4 levels of page tables.

- a. (3 marks) How many logical pages and physical frames are there?

Ans: Logical pages = $2^{64}/2^{13} = 2^{51}$ pages
 $Frames = 2^{48}/2^{13} = 2^{35}$ frames

- b. (3 marks) How many page entries can a single page table chunk (or tablets) have?

Ans: Single page size = 2^{13}
 $PTE = 2^3$
 $Each\ page\ table\ chunks\ has = 2^{13}/2^3 = 2^{10}$ entries

- c. (4 marks) How many bits in the logical address remain unused? [2 marks]

Ans: 13 bits = offset
 $Each\ page\ tablet\ requires\ 10\ bit\ and\ there\ are\ 4\ levels\ including\ the\ PD \Rightarrow 40\ bits$
 $Total = 53\ bits.$
 $Unused: 64 - 53 = 11\ bits$

- d. (4 marks) Suppose a process has 1 frame allocated to it. What is total memory space required by the page tables. Do not include the space required by actual frames. Answer in KiB unit.

Ans.

Each page tablet can hold 2^{10} entries.
No of page tablet in level 4 $\Rightarrow 1$
No of page tablet in level 3 $\Rightarrow 1$
No of page tablet in level 2 $\Rightarrow 1$
PD $\Rightarrow 1$ entry
 $1\ PD + 3\ tablet \Rightarrow 4 * 8 \Rightarrow 32\ KiB$

- e. (6 marks) Suppose a process has 2097152 (2^{21}) frames allocated to it. What is total memory space required by the page tables. Do not include the space required by actual frames. Answer in KiB unit.

Ans:

Each page tablet can hold 2^{10} entries.
No of page tablet in level 4 $\Rightarrow 2^{21}/2^{10} = 2^{11}$
No of page tablet in level 3 $\Rightarrow 2^{11}/2^{10} = 2$
No of page tablet in level 2 $\Rightarrow 1$
PD $\Rightarrow 1$ entry
 $Total\ PT\ space = 8\ K(1\ PD) + 8K(lev\ 2) + 2*8K(lev\ 3) + 2^{11}*8K(lev\ 4)$
 $= 8K + 8K + 16K + 16584\ KB = 16416KB$

Question 4. File Systems (10 MARKS)

We consider an inode file system with 8 direct pointers, 4 single-indirect pointers, 2 double-indirect pointers, and 1 triple-indirect pointer. Each data block is 1024 bytes, and each block pointer is 8 bytes.

- a. (2 marks) What is the maximum file size possible on this file system? Express your answer in GB ($1 \text{ GB} = 1024 \text{ MB} = 1024 \times 1024 \text{ KB} = 1024 \times 1024 \times 1024 \text{ bytes}$) correct to 2 decimal places. Assume that there is sufficient space available.

$$\text{Each data block} = 1024 / 8 = 128 \text{ block pointers.}$$

$$\# \text{ of direct blocks} = 8$$

$$\# \text{ of single indirect blocks} = 4 \times 128$$

$$\# \text{ of double indirect blocks} = 2 \times 128^2$$

$$\# \text{ of triple indirect blocks} = 128^3$$

$$\begin{aligned} \text{Total number of blocks} &= 8 + 4 \times 128 + 2 \times 128^2 + 128^3 = 8 + 512 + 32768 + 2097152 \\ \text{blocks} &= 2130440 \text{ blocks} \\ &= 2130440 \times 1024 \text{ bytes} \\ &= 2.03 \text{ GB} \end{aligned}$$

- b. (8 marks) Assuming that only the inode has been loaded into memory, and none of the lower level index blocks have been loaded yet, what is the TOTAL number of blocks that must be loaded from disk when reading data from the following byte offsets (relative to the start of the file):

Answer:

Indirection Level	Starting Block Number	Ending Block Number
0	0	7
1	8	519
2	520	33287
3	33288	2130439

Offset (bytes from start)	Block #	Indirection Level	Total Blocks Read
67	0	0	1
8100	7	0	1
534550	522	2	3
34082815	33283	2	3

In general the # of blocks read = indirection level + data block = indirection level + 1.

Offset (bytes from start)	TOTAL # of block read
67	1
8100	1
534550	3
34082815	3

~~ END OF ASSESSMENT ~~