OS Midsem Exam, Sec-B
Monsoon 2024

Total points: 125

Q1- What is the output printed by the following snippet of pseudocode? If more than one answer is possible, list all the answers and the conditions under which each answer is possible. **[5 pts]**

```
int fd[2];
pipe(fd);
int rc = fork();
if(rc == 0) {
close(fd[1]);
printf("child1\n");
read(fd[0], bufc, bufc_size);
printf("child2\n");
}
else {
close(fd[0]);
printf("parent1\n");
write(fd[1], bufp, bufp_size);
wait();
printf("parent2\n");
}
```

Answer - If child scheduled before parent: child1, parent1, child2, parent2.
If parent scheduled before child: parent1, child1, child2, parent2.

Q2 What are the possible outputs printed from the program snippet shown below? You may assume that the program runs on a modern Linux-like OS. You may ignore any output generated from "some executable". You must consider all possible scenarios of the system calls succeeding as well as failing. In your answer, clearly list all the possible scenarios, and the output of the program in each of these scenarios. **[5 pts]**

```
int ret = fork();
if(ret == 0) {
printf("Hello1\n");
exec("some_executable");
printf("Hello2\n");
} else if(ret > 0) {
wait();
printf("Hello3\n");
```

```c
} else {
printf(“Hello4\n”);
}
```

Answer -
Case I: fork and exec succeed. Hello1, Hello3 are printed.
Case II: fork succeeds but exec fails. Hello1, Hello2, Hello3 are printed.
Case III: fork fails. Hello4 is printed.

Q3. Consider a code segment where a parent process forks a child process. The child process is responsible for counting the number of words in a text file and returning the count to the parent process. Fill in the blanks to complete the code. Provide brief explanations.
[15 pts]

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int count_words() {
    return 42;  // Assume this is the word count for simplicity
}

int main() {
    int pipefd[2];
    pipe(___);  // Fill in the blank (1)

    int pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        return 1;
    } else if (pid == 0) {
        // Child process
        close(___);  // Fill in the blank (2)
        int word_count = count_words();
        write(___, &word_count, sizeof(word_count));  // Fill in the blank (3)
        close(___);  // Fill in the blank (4)
        exit(0);
    } else {
        // Parent process
        close(___);  // Fill in the blank (5)
        int received_count;
        read(___, &received_count, sizeof(received_count));  // Fill in the blank (6)
        close(___);  // Fill in the blank (7)
        wait(NULL);  // Wait for child to finish
        printf("Word count: %d\n", received_count);
    }
    return 0;
}
```

Answer -
1: pipe(pipefd); — Create the pipe.
2: close(pipefd[0]); — Close the unused read end in the child.
3: write(pipefd[1], &word_count, sizeof(word_count)); — Write the word count to the pipe.
4: close(pipefd[1]); — Close the write end after writing.
5: close(pipefd[1]); — Close the unused write end in the parent.
6: read(pipefd[0], &received_count, sizeof(received_count)); — Read the word count in the parent.
7: close(pipefd[0]); — Close the read end after reading.

Q4. Consider a system with a 64 byte virtual address space, and a 16 byte page size. The mapping from virtual page numbers to physical frame numbers of a process is (0,8), (1,3), (2,11), and (3,1). Translate the following virtual addresses to physical addresses. Note that all addresses are in decimal. You may write your answer in decimal or binary.
[8 pts]

(a) 20 (b) 40

Ans: (a) 20 = 01 0100 = 11 0100 = 52 (b) 40 = 10 1000 = 1011 1000 = 184

Q5. Consider a system with several running processes. The system is running a modern OS that uses virtual addresses and demand paging. It has been empirically observed that the memory access times in the system under various conditions are: t1 when the logical memory address is found in TLB cache, t2 when the address is not in TLB but does not cause a page fault, and t3 when the address results in a page fault. This memory access time includes all overheads like page fault servicing and logical-to-physical address translation. It has been observed that, on an average, 10% of the logical address accesses result in a page fault. Further, of the remaining virtual address accesses, two-thirds of them can be translated using the TLB cache, while one-third require walking the page tables. Using the information provided above, calculate the average expected memory access time in the system in terms of t1,t2, and t3. **[10 pts]**

Ans: 0.6*t1 + 0.3*t2 + 0.1*t3

Q6. Consider a system with 8-bit virtual and physical addresses, and 16 byte pages. A process in this system has 4 logical pages, which are mapped to 3 physical pages in the following manner: logical page 0 maps to physical page 6, 1 maps to 3, 2 maps to 11, and logical page 5 is not mapped to any physical page yet. All the other pages in the virtual address space of the process are marked invalid in the page table. The MMU is given a pointer to this page table for address translation. Further, the MMU has a small TLB cache that stores two entries, for logical pages 0 and 2. For each virtual address shown below, describe what happens when that address is accessed by the CPU. Specifically, you must answer what happens at the TLB (hit or miss?), MMU (which page table entry is accessed?), OS (is there a trap of any kind?), and the physical memory (which physical address is accessed?). You may write the translated physical address in binary format. (Note that it is not implied that the accesses below happen one after the other; you must solve each part of the question independently using the information provided above.)

[16 pts]

(a) Virtual address 7 (b) Virtual address 20 (c) Virtual address 70 (d) Virtual address 80

Ans

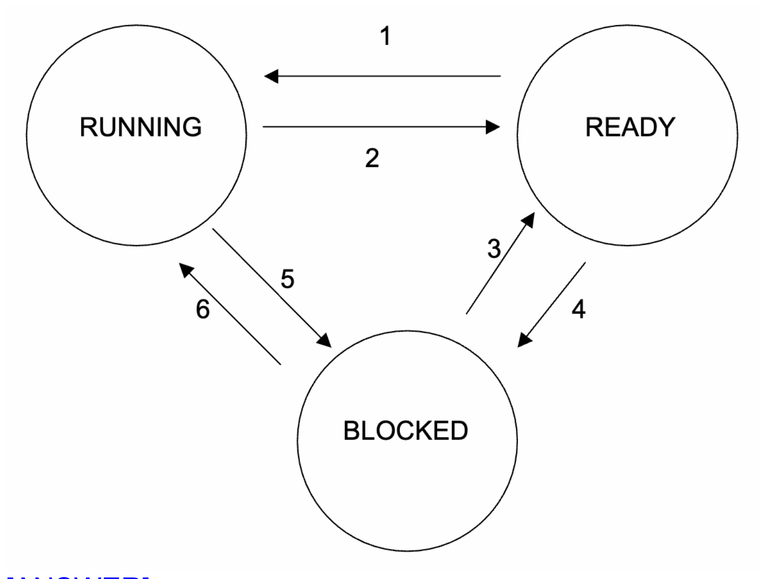
(a) 7 = 0000 (page number) + 0111 (offset) = logical page 0. TLB hit. Page table need not be accessed. No OS trap. Physical address 0110 0111 is accessed.

(b) 20 = 0001 0100 = logical page 1. TLB miss. MMU searches the page table. Physical address 0011 0100

(c) 70 = 0100 0110 = logical page 4. TLB miss. MMU accesses page table and discovers it is an invalid entry. MMU raises trap to OS.

(d) 80 = 0101 0000 = logical page 5. TLB miss. MMU accesses page table and discovers page not present. MMU raises a page fault to the OS.

Q7: Label the conditions that would cause a process to move between the three states. Label it N/A if it doesn't happen.
[6 pts]



- [ANSWER]
    i. Arrow 1: The process is **scheduled** and run by the scheduler.

ii. Arrow 2: Process is **descheduled**, and placed back into the ready queue.
iii. Arrow 3: I/O completes
iv. Arrow 4: N/A
v. Arrow 5: Any **blocking action**. I/O request.
vi. Arrow 6: N/A

Q8. Consider a system with a single CPU core and three processes A, B, C. Process A arrives at t=0, and runs on the CPU for 10 time units before it finishes. Process B arrives at t=6, and requires an initial CPU time of 3 units, after which it blocks to perform I/O for 3 time units. After returning from I/O wait, it executes for a further 5 units before terminating. Process C arrives at t=8, and runs for 2 units of time on the CPU before terminating. For each of the scheduling policies below, calculate the time of completion of each of the three processes. Recall that only the size of the current CPU burst (excluding the time spent for waiting on I/O) is considered as the "job size" in these schedulers. Explain your answer in as much detail as possible. **[15 pts]**

a) First Come First Serve b) Shortest Job First c) Shortest Remaining Time First

Ans

(a) First Come First Serve (non-preemptive).
Ans: A=10, B=21, C=15. A finishes at 10 units. First run of B finishes at 13. C completes at 15. B restarts at 16 and finishes at 21.

(b) Shortest Job First (non-preemptive)
Ans: A=10, B=23, C=12. A finishes at 10 units. Note that the arrival of shorter jobs B and C does not preempt A. Next, C finishes at 12. First task of B finishes at 15, B blocks from 15 to 18, and finally completes at 23 units.

(c) Shortest Remaining Time First (preemptive)
Ans: A=15, B=20, C=11. A runs until 6 units. Then the first task of B runs until 9 units. Note that the arrival of C does not preempt B because it has a shorter remaining time. C completes at 11. B is not ready yet, so A runs for another 4 units and completes at 15. Note
that the completion of B's I/O does not preempt A because A's remaining time is shorter. B
finally restarts at 15 and completes at 20.

Q9. Using the least frequently used (LFU) policy for page replacement, find the number of hits, misses and the hit rate for the following pages. The cache/frame size is 3. Show all the steps, including the state of the cache at each step, along with indicating the frequency of the page. Clearly state your assumptions, for example, in the case of a tie. **[20 pts]**

Page sequence: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1

Ans:

Assumptions: In case of a tie, we apply LRU here and choose the page that was least recently used/accessed. Other possibilities include applying FIFO or picking one at random. Answer can change depending on the tie-breaking mechanism used.

Please note that, instead of following any algorithm that looks at history (pages already accessed), if a lookahead mechanism is used, i.e., optimal policy, a heavy penalty will be imposed (student may not get more than 4-5 marks for this question).

1 point for writing the assumptions. If the assumptions are not clearly stated, a heavy penalty will be imposed (student may not get more than 4-5 marks for this question). 14 points for the correct and complete table.

Step	Page	Cache State (after insertion)	Frequency (after insertion)	Hit/Miss
1	7	[7]	{7: 1}	Miss
2	0	[7, 0]	{7: 1, 0: 1}	Miss
3	1	[7, 0, 1]	{7: 1, 0: 1, 1: 1}	Miss
4	2	[0, 1, 2]	{0: 1, 1: 1, 2: 1}	Miss (Replace 7)
5	0	[0, 1, 2]	{0: 2, 1: 1, 2: 1}	Hit
6	3	[0, 2, 3]	{0: 2, 2: 1, 3: 1}	Miss (Replace 1)
7	0	[0, 2, 3]	{0: 3, 2: 1, 3: 1}	Hit
8	4	[0, 3, 4]	{0: 3, 3: 1, 4: 1}	Miss (Replace 2)
9	2	[0, 4, 2]	{0: 3, 2: 2, 4: 1}	Miss (Replace

				3)
10	3	[0, 2, 3]	{0: 3, 2: 2, 3: 2}	Miss (Replace 4)
11	0	[0, 2, 3]	{0: 4, 2: 2, 3: 2}	Hit
12	3	[0, 3, 2]	{0: 4, 3: 3, 2: 2}	Hit
13	2	[0, 3, 2]	{0: 4, 3: 3, 2: 3}	Hit
14	1	[0, 2, 1]	{0: 4, 2: 3, 1: 2}	Miss (Replace 3)

- Hits: 5 (pages: 0, 0, 0, 3, 2) // 1.5 point
- Misses: 9 (pages: 7, 0, 1, 2, 3, 4, 2, 3, 1) // 1.5 points
- Total requests: 14
- Hit Rate = (Number of Hits) / (Total requests) = 5/14 = 0.3571 // 2 points
- Final frequency : {7:1, 0: 4, 1:2, 2:3, 3:3, 4:1}

Q10. Use the clock algorithm that considers reference/use and dirty bits for replacing pages. Find the number of hits, misses and the hit rate for the following pages. The cache/frame size is 4. The reference bit of all the pages is initially set to 1. The dirty bit of pages 1, 2, 3 are set to 1 and is 0 for the rest of the pages. Show all the steps, including the state of the cache at each step, the reference bit and use bit for each page. Clearly state your assumptions, for example, the position of the hand. **[25 pts]**

Page sequence: 1, 2, 3, 4, 5, 1, 6, 2, 7, 8, 1, 2, 9, 10

Assumptions (1 point)

- Clock hand always starts with page 1 in the cache.

// correct table is 20 points

Step	Page	Cache state after insertion - (page, use bit, dirty bit)	Hit/Miss	Comments
1	1	(1,1,1)	Miss	

2	2	((1,1,1), (2,1,1))	Miss	
3	3	((1,1,1), (2,1,1), (3,1,1))	Miss	
4	4	((1,1,1), (2,1,1), (3,1,1), (4,1,0))	Miss	
5	5	((1,0,1), (2,0,1), (3,0,1), (5,1,0))	Miss	Replace 4 in the second pass since that is the only page with dirty bit 0.
6	1	((1,1,1), (2,0,1), (3,0,1), (5,1,0))	Hit	
7	6	((1,0,1), (2,0,1), (3,0,1), (6,1,0))	Miss	Replace 5 in the second pass since that is the only page with dirty bit 0.
8	2	((1,0,1), (2,1,1), (3,0,1), (6,1,0))	Hit	
9	7	((1,0,1), (2,0,1), (3,0,1), (7,1,0))	Miss	Replace 6 in the second pass since that is the only page with dirty bit 0.
10	8	((1,0,1), (2,0,1), (3,0,1), (8,1,0))	Miss	Replace 7 in the second pass since that is the only page with dirty bit 0.
11	1	((1,1,1), (2,0,1), (3,0,1), (8,1,0))	Hit	
12	2	((1,1,1), (2,1,1), (3,0,1), (8,1,0))	Hit	
13	9	((1,0,1), (2,0,1), (3,0,1), (9,1,0))	Miss	Replace 8 in the second pass since that is the only page with dirty bit 0.
14	10	((1,0,1), (2,0,1), (3,0,1), (10,1,0))	Miss	Replace 9 in the second pass since that is the only page with dirty bit 0.

- Hits: 4 // 1 point
- Total requests: 14
- Hit Rate = (Number of Hits) / (Total requests) = 0.286 // 3 points