Operating Systems

Final Exam, Spring 2015

Date: June 11, 2015 **Marks:** 60 **Time:** 3 hrs

Question 1 (5+5 marks)

a) Consider a CPU scheduler (say scheduler one) that computes process priority using the following formula:

priority = recent CPU usage (in milliseconds)

There is an inverse relationship between the priority and its numeric value, that is, higher the numeric value the lower the priority (zero is the highest priority).

Now consider another scheduler (say scheduler two) that uses the following formula instead:

priority = 1 / (recent CPU usage)

Which scheduler is good? Give reason for your answer.

b) Can a multithreaded solution perform better on a single-processor system? Give reason for your answer.

Question 2 (8+2 marks)

a) Show execution of the LRU page replacement algorithm on the following page reference string. Assume only three frames are available. If the LRU algorithm is unable to guide you at some point then use FIFO.

1 2 3 1 4 5 6 4 7

b) Give the total number of page faults occurred in part a.

Question 3 (10 marks)

Consider a computer system that uses XYZ processor. This processor supports virtual memory with a page size of 16 bytes and has instructions of size 4 bytes. The address bus of XYZ processor is 16 bits, while the data bus is 32 bits (i.e., all memory reads and writes are in 4 bytes).

A process P1 running on this computer system executes the following code:

```
Load  R1  [0x001C]   /* Load the value at address 0x001C into Register R1 */
Load  R2  [0x0074]   /* Load the value at address 0x0074 into Register R2 */
Add   R1  R2         /* Add the values of Registers R1 and R2 and store the result in R1 */
Store R1  [0x0038]   /* Store the value in register R1 at address 0x0038 */
```

Figure 3(a) shows the page table of process P1 and Figure 3(b) shows the state of the physical memory before execution of the above code. What would be the state of the Physical memory after execution of the above code. You need to show the state by drawing the Figure 3(c) with (only) new values in your answer book.

	Frame No.	Valid/Invalid
0	0	Valid
1	3	Valid
2	0	Invalid
3	1	Valid
4	0	Invalid
5	0	Invalid
6	0	Invalid
7	2	Valid

(a)

Physical Address	
0x0000	Process P1 Code
0x0004	
0x0008	
0x000C	
0x0010	0x90000018
0x0014	0x09000055
0x0018	0x00000024
0x001C	0x00000000
0x0020	0x00800098
0x0024	0x20000005
0x0028	0x00002004
0x002C	0x00000007
0x0030	0x00030045
0x0034	0x00000054
0x0038	0x00000036
0x003C	0x00000004

(b)

Physical Address	
0x0000	Process P1 Code
0x0004	
0x0008	
0x000C	
0x0010	
0x0014	
0x0018	
0x001C	
0x0020	
0x0024	
0x0028	
0x002C	
0x0030	
0x0034	
0x0038	
0x003C	

(c)

Figure 3:(a) Page Table of process P1. (b) Memory state before execution of the above code. (c) Memory state after the execution of the above code (you need to fill-in the memory state).

Operating Systems
Final Exam, Spring 2015

Date: June 11, 2015 **Marks:** 60 **Time:** 3 hrs

Question 4 (15 marks)

Consider a signal where vehicles move in four directions: north, south, east and west. At a time, the vehicles move either north/south or east/west; turns are not allowed. Following are four functions: one for each direction. The vehicles (threads) moving north execute the function moveNorth, the vehicles moving south execute the function moveSouth,and so on. Synchronize the threads using semaphore(s). Do not worry about starvation; assume breaks come in the traffic regularly after some time.

// Type 1	// Type 2	//Type 3	//Type 4

void moveNorth() {
...
// cross signal
...
}
	void moveSouth() {
	...
	// cross signal
	...
	}
		void moveEast() {
		...
		// cross signal
		...
		}
			void moveWest() {
			...
			// cross signal
			...
			}

Question 5 (15 marks)

Consider a file system using a variation of the Indexed Allocation:The first n-1 pointers of the first index block point directly to the data blocks. The last pointer, however, points to another index block (second index). Similarly the second index contains n-1 direct pointers, and the last pointer points to the next index block (third index), and so on.

Assuming a block size of one KB and a pointer size of four bytes, write a C/C++ function to compute the physical-block-number for a given logical-block-number of a file. Following is the function prototype:

```
int find(int index_block, long logical_block)
```

Here, index_block holds the address of the first index block of the file, and the logical_block contains the given logical-block-number. The function returns the corresponding physical block number after computation.

Assume we have a function available to read a disk block as follows:

```
bool read(long block_num, char* buffer)
```

Here 'block_num' is the address of the block to read, and 'buffer' is the target memory buffer/array.