University of Waterloo
Midterm Examination
Term: Winter Year: 2010

Student Family Name _____

Student Given Name _____

Student ID Number _____

Section : Circle one (Brecht 9:30) (Brecht 11:30)

Course Abberviation and Number:	CS 350
Course Title:	Operating Systems
Section(s):	2
Instructors:	Tim Brecht

Date of Exam:	February 24, 2010
Time Period	Start time: 7:00 pm End time: 9:00 pm
Duration of Exam:	120 minutes
Number of Exam Pages:	10 (including cover sheet)
	NO CALCULATORS, NO ADDITIONAL MATERIAL

Problem	Mark	Score	Marker's Initials
1	12		
2	24		
3	20		
4	24		
5	16		
6	15		
7	9		
8	25		
Total	145		

Problem 1 (12 marks)

a. **(4 mark(s))** There are only two processes in the system P_A and P_B containing one thread each. Assume that P_A is running and P_B is on the ready to run queue. Describe the steps required to save and restore or manipulate the processes context (including MMU) if P_A performs a **non-blocking** system call and it has not used up all of its quantum. Be sure to explain where the context is saved to and restored from. Provide a general description, you don't need to give specific register names.

b. **(8 mark(s))** There are only two processes in the system P_C and P_D containing one thread each. Assume that P_C is running and P_D is on the ready to run queue. Describe the steps required to save and restore or manipulate the processes context (including MMU) if P_C performs a **blocking** system call and it has not used up all of its quantum. Be sure to explain where the context is saved to and restored from. Provide a general description, you don't need to give specific register names.

Problem 2 (24 marks)
For this question all addresses, virtual page numbers and physical frame numbers are represented in hexidecimal. Consider a machine with *44-bit* virtual addresses, *48-bit* physical addresses, and a page size of 1 MB. During a program execution the TLB contains the following valid entries (in hexidecimal).

Virtual Page Num	Physical Frame Num	Valid	Read-Only / Can't Make Dirty
0x 0	0x 171	0	0
0x 1E	0x 172	1	1
0x 1EF	0x 173	1	1
0x 1EF1	0x 1EF	1	0
0x 1EF17	0x EF1	0	0
0x 1EF170	0x 8132CD	1	0
0x 1EF171	0x 132CD	1	1
0x 1EF172	0x 88132CD	1	0

Examine the following set of instructions and if possible, translate the addresses. If the translation is not possible or an exception would be raised, indicate which exception and why the exception is raised. Show and explain how you derived your answer. Express, in hexidecimal, the required address using **ALL 44-bits for virtual addresses and 48-bits for physical addresses** (i.e., including leading zeros).

a. **(4 mark(s))** A load occurs from virtual address = 0x 000 01EF AC00.
If a translation occurs provide the physical address.

b. **(4 mark(s))** A load occurs from virtual address = 0x 1EF 1729 52CD.
If a translation occurs provide the physical address.

c. **(4 mark(s))** A store occurs to virtual address = 0x 1EF 1729 AC00.
If a translation occurs provide the physical address.

d. **(4 mark(s))** A store occurs to virtual address = 0x 1EF 1709 AC00.
If a translation occurs provide the physical address.

e. **(4 mark(s))** A load occurs from virtual address = 0x 1EF 1739 AC00.
If a translation occurs provide the physical address.

f. **(4 mark(s))** Can a store occur at physical address 0x 0000 1EF1 32CD? Explain your answer. If it can occur, what is the corresponding virtual address?

Problem 3 (20 marks)

a. **(12 mark(s))** We would like multiple threads to be able to call `add` and `sub` (shown below), simultaneously. Add code that uses **locks** (as defined in OS/161) to ensure that all of the code below will execute correctly and so that maximum parallelism is ensured. Assume that these are the only functions that can alter the array and that there is no need to do any error checking.

```
#define MAX      (100)
volatile int array[MAX];









void init() /* Only called by one thread before using other functions */
{
        int i;


        for (i=0; i<MAX; i++) {


            array[i] = 0;


        }


}

void add(int index, int value)
{



        array[index] = array[index] + value;



}

void subtract(int index, int value)
{



        array[index] = array[index] - value;



}
```

b. **(8 mark(s))** Now you would like to add the `sum` function shown below to the library of functions on the previous page. This function will be called relatively infrequently (e.g., once a month) and **IT WILL ONLY EVER BE CALLED BY ONE THREAD**.

Assuming that multiple threads may be executing `add` and `sub` concurrently with the thread that calls `sum`, is synchronization required for the function `sum` if it is only called by a single thread? Explain why or why not. If synchronization is required, add it to the code below so that it integrates with your solution on the previous page (i.e., be sure to do the first part of this question first).

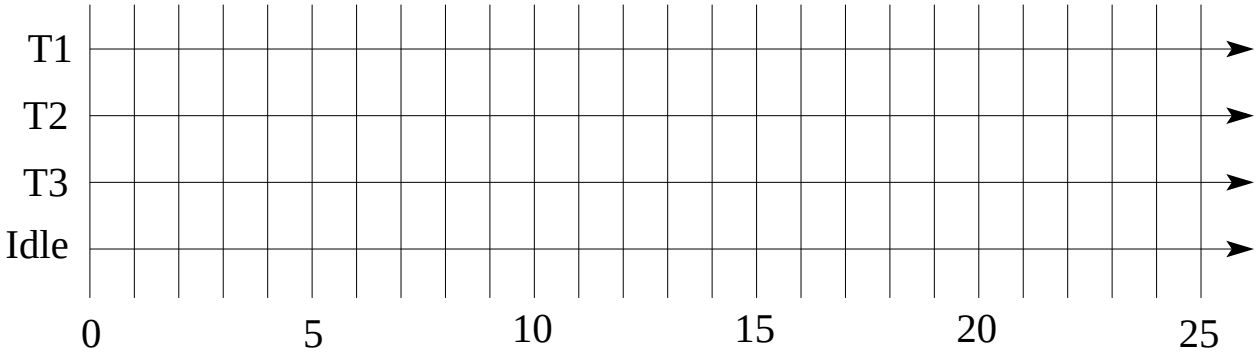
```
int sum()
{
        int i;
        int total = 0;









        for (i=0; i<MAX; i++) {




                total = total + array[i];




        }









        return total;




}
```

Problem 4 (24 marks)
Three threads are in the ready to run queue, in the order, T_1, T_2, and T_3.

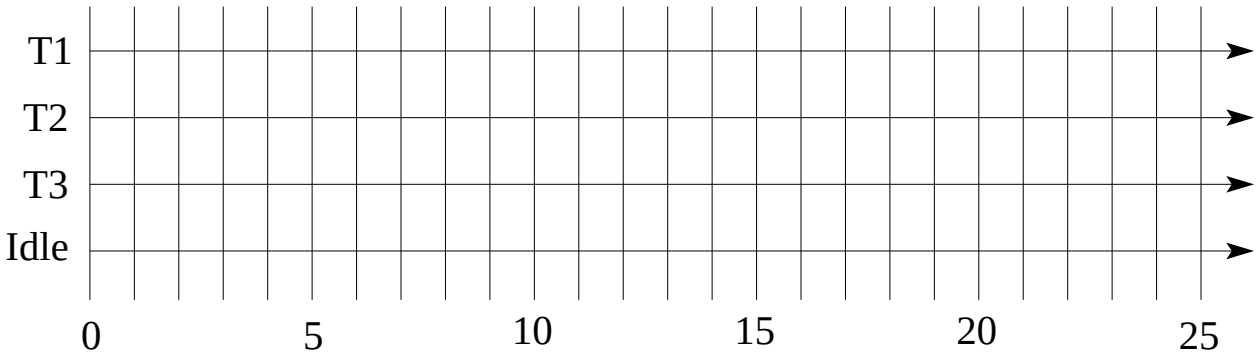
- Thread T_1 runs a loop that executes 4 times. During each loop it runs for 1 unit of time and then makes a system call that blocks for 4 units of time. After looping the program exits.
- Threads T_2, and T_3 run a loop that executes 2 times. During each loop they run for 3 units of time and then call `thread_yield`. After looping the program exits.

On the timeline below show when each thread runs and when the processor is idle by shading the appropriate thread or idle line. **If two events happen at the same time, assume that the higher numbered thread goes first (i.e., its event occurs first).**

a. **(10 mark(s))** Assume a non preemptive FIFO scheduler and the starting point is as described at the top of this question.



b. **(14 mark(s))** Assume a preemptive round-robin scheduler with a quantum of 2 and that the running time for the thread is reset to zero every time the thread executes (i.e., unused quantums do **not** carry over to the next time the thread runs). Use the starting point as described at the top of the question.



Problem 5 (16 marks)

a. **(2 mark(s))** After a thread has called `thread_yield` what steps are required for it to run again?

b. **(2 mark(s))** Under what circumstances might an executing thread call `thread_yield` and yet be the next thread to run.

c. **(2 mark(s))** After a thread has called `thread_sleep` what steps are required for it to run again?

d. **(6 mark(s))** Describe one pro (advantage) and two cons (disadvantages) of disabling interrupts as a means for implementing synchronization.

e. **(4 mark(s))** Describe one pro (advantage) and one con (disadvantage) of using an atomic instruction like Test-and-Set as a means for implementing synchronization.

Problem 6 (15 marks)
WRONG ANSWERS WILL RECEIVE NEGATIVE MARKS, SO YOU MAY NOT WANT TO GUESSS IF YOU ARE NOT SURE OF THE ANSWER

a. **(3 mark(s))** In OS/161 the only instruction that can be executed in user-mode that can cause the processor to switch to priveledged (or kernel) mode is the `syscall` instruction. True or False?

b. **(3 mark(s))** The MIPS uses a software loaded TLB. True or False?

c. **(3 mark(s))** In OS/161 interrupts can be disabled while in user-mode. True or False?

d. **(3 mark(s))** In OS/161 the number of bytes in the virtual address space of a program is always equal to the number bytes in the executable file True or False?

e. **(3 mark(s))**

The original implementation of semphores provided by OS/161 ensures that starvation can not occur. True or False?

Problem 7 (9 marks)
Study the code below and answer the question that follows.

```
/* SHOULD ALWAYS BE BETWEEN 0 AND 100. INCLUDING 0 AND 100 */
volatile int count = 0;

adjust_count()
{
    int spl;
    int doreset = FALSE;

    if (count == 100) {
       doreset = TRUE;
    }

    spl = splhigh();

    if (doreset == TRUE) {
        count = 0;
    } else {
        count++;
    }

    splx(spl);
}
```

a. **(9 mark(s))** If multiple threads are executing and calling `adjust_count`, is the value for the variable `count` guaranteed to be between 0 and 100 (including 0 and 100)? Explain your answer. Use examples if it helps to clarify your answer.

Problem 8 (25 marks)

This question uses the following notation (as used in the course notes) to describe resource allocation in a computer system :

- D_i: demand vector for process P_i
- A_i: current allocation vector for process P_i
- U: unallocated (available) resource vector

Given the scenario below, fill in the details for the resource allocation graph, **indicate if the system is deadlocked, and justify your answer (in one sentence).**

PLEASE USE SOLID LINES WITH ARROWS FOR ALLOCATION EDGES AND DASHED OR DOTTED LINES WITH ARROWS FOR REQUEST EDGES.

$U = (1, 0, 0, 0)$

$D_1 = (1, 0, 1, 0)$, $D_2 = (0, 1, 1, 0)$, $D_3 = (0, 0, 1, 1)$, $D_4 = (0, 0, 0, 1)$

$A_1 = (1, 0, 0, 1)$, $A_2 = (1, 1, 0, 1)$, $A_3 = (1, 0, 0, 1)$, $A_4 = (0, 2, 1, 1)$.