Name, SURNAME and ID ⇒

Middle East Technical University
Department of Computer Engineering

CENG 334

Section 1 and 2
Spring '2007-2008

Midterm

- **Duration: 120** minutes.
- **Exam:**
  - This is a **closed book**, **closed notes** exam. No attempts of cheating will be tolerated. In case such attempts are observed, the students who took part in the act will be prosecuted. The legal code states that students who are found guilty of cheating shall be expelled from the university for **a minimum of one semester**!
- **About the exam questions:**
  - The points assigned for each question are shown in parenthesis next to the question.
  - Whereever available, use the boxes to write down your answers.
- **This booklet consists of *8* pages including this page. Check that you have them all!**
- **GOOD LUCK !**

Question 1	
Question 2	
Question 3	
Question 4	
Question 5	
Question 6	
Total ⇒	

1 (18 pts)

(a) (3 pts) Which of the following is true? Note that incorrect answers would be punished with -3 points. (a) OS kernel is essentially an infinite loop that runs on the CPU in all the remaining times left from application programmes. (b) OS kernel is just a set of interrupt handler that gets triggered by software and hardware interrupts. (c) Both.

(b) (3 pts) You're given the binaries of two programs, and asked which one of them is a kernel code. By looking at the de-assembled binary code, how can you distinguish the two?

(c) (3 pts) Write down one advantage and one disadvantage of a microkernel OS against a monolithic one?

(d) (3 pts) In two sentences, write down how priority inversion happens.

(e) (3 pts) Which of the following pages is faster to page-out: (a) page storing the code segment of the executable, (b) page storing the stack. Incorrect answer would be punished with -3 points.

(f) (3 pts) In which implementation of monitors does the thread that waken up is expected to check the condition that it was sleeping on: Mesa or Hoare? Incorrect answer would be punished with -3 points.

2 (30 pts)

Answer the following questions on synchronization among multiple threads:

(a) (5 pts) Thread A has to wait for Thread B and vice versa. The idea is that two threads rendezvous at a point of execution, and neither is allowed to proceed until both have arrived. In other words, given this code

Thread A	Thread B
`a1;`	`b1;`
`a2;`	`b2;`

we want to guarantee that `a1` happens before `b2`, and `b1` happens before `a2`. Your solution should not enforce too many constraints. For example, we don't care about the order of `a1` and `b1`. In your solution, either order should be possible.

Use the following declarations as a base for your solution:

```
aArrived = Semaphore(0);
bArrived = Semaphore(0);
```

You can use these semaphores by calling two methods, such as `aArrived.up();` or `bArrived.down();` Write the necessary synchronization code for Thread A, and Thread B in the boxes shown below.

(b) (15 pts) Consider a generalized version of the previous problem for n threads, which is called as a *barrier*. Every thread should run the following code:

```
rendezvous;
critical_point;
```

The synchronization requirement is that no thread executes `critical_point` until after all threads have executed `rendezvous`.

You can assume that there are n threads and that this value is stored in a variable, `n`, that is accessible from all threads.

When the first $n-1$ threads arrive they should block until the nth thread arrives, at which point all the threads may proceed.

Use the following declarations as a base for your solution:

```
n = the_number_of_threads;
count = 0;
mutex = Semaphore(1);
barrier = Semaphore(0);
```

`count` keeps track of how many threads have arrived. `mutex` provides exclusive access to `count` so that threads can increment it safely.

`barrier` is locked until all threads arrive; then it should be unlocked.

Write the synchronization code that needs to be inserted in between the `rendezvous` and the `critical_point` below.

(c) (10 pts) Implement a barrier, using monitors and condition variables. After executing the `rendezvous` part of the code, each thread should just call `Barrier.check()`.

Use the following skeleton to write your code. Declare the necessary `int`eger and `condition` variables as needed. Assume that the functions `wait(cv)`, `notify(cv)`, and `notifyAll(cv)`, are available where `cv` is a declared condition variable.

```
monitor Barrier{
int                                                    ;
condition                                              ;


     void check(){









    }
}
```

3 (20 pts)

Three processes get into the ready queue of the scheduler in the order A, B and E. If you use the FCFS policy the scheduling executed is as follows. In the timing diagram shown below for 20 time units, A, B, an E represents the CPU use of these processes whereas b and e denote the I/O's of processes B and E respectively (A has no I/O). The vertical bars indicate transitions of a process from running to the I/O wait or ready queue.

In the diagram below fill in the scheduling executed by Round robin (RR) with a timeslice of 1 unit, Shortest-Job-First (SJF), and Shortest-Remaining-Time-First (SRTF). For SRTF, assume that the scheduler has complete knowledge on the CPU and I/O requirements of the processes. If at a given time slice the CPU is being used by, say A, whereas the I/O requests of B and E are active, then fill in that time slice with Abe **vertically**. Also assume that the I/O device can concurrently handle multiple I/O requests.

	0										1									
Time	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
FCFS	A						B		b		B		b		E	e	E	e	E	e
RR																				
SJF																				
SRTF																				

4 (10 pts)

A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:

Thread	Allocation					Maximum				
T1	1	0	2	1	1	1	1	2	1	3
T2	2	0	1	1	1	2	2	2	1	1
T3	1	1	0	1	0	2	1	3	1	0
T4	1	1	1	1	0	1	1	2	2	1

If the resources currently available in the system are given by `0 0 x 1 1`, then what is the smallest value of `x` for which this is a safe state? Show your work.

5 (10 pts)

A computer has four page frames. The time of loading, time of last access and the R bits of each page are shown as below:

Page	Loaded	Last access	R
0	126	280	1
1	230	265	0
2	140	270	0
3	110	285	1

(a) Which page will FIFO algorithm replace next?

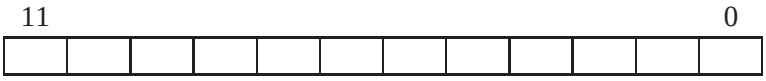
(b) Which page will LRU algorithm replace next?

(c) Assuming that the clock is pointing at page frame 0, which page will the second-chance algorithm replace next?

6 (12 pts)

A system has an 12-bit virtual address space, and a 14-bit physical address space, with a page size of 64 bytes. Page table entries are 4 bytes each and include a valid bit, permission bits (Read, Write and eXecute), and the physical Page Frame Number (PFN) mapped to that entry. A two-level page table scheme is used for address translation.

(a) Show how the virtual address is split into pieces to access the physical memory.



In this system, indicate the result of each of the operations below. Identify the page table entries that are used, and if the translation is successful, give the physical page frame that is accessed. If the translation fails, indicate the type of exception.

(b) Read from virtual address 0xC0D.

(c) Write to virtual address 0x24F.

(d) Load instruction from virtual address 0x287.

Top level page table

Index	V	rwx	PageFrameNumber
0	1	110	0x03
1	1	110	0x08
2	0	110	0x01
3	0	110	0xAB
4	0	110	0x02
5	0	110	0x13
6	1	110	0x07
7	1	110	0x0F

Second level page table
(stored in page frame 0x08)

Index	V	rwx	PageFrameNumber
0	1	101	0x04
1	0	101	0x06
2	1	100	0x0C
3	0	000	0x10
4	1	100	0x09
5	0	000	0x14
6	1	110	0x05
7	1	110	0x1E

Second level page table
(stored in page frame 0x07)

Index	V	rwx	PageFrameNumber
0	1	110	0x11
1	0	110	0x03
2	0	110	0x0C
3	1	110	0x10
4	1	110	0x0B
5	0	110	0x1A
6	1	100	0x1C
7	0	100	0x0A