This exam is open book and notes, including ebooks. But no communication with intelligent entities is permitted.

Answer all the questions in part A (63 points) in the bluebook. Answer parts B (24 points) and C (24 points) on the question sheet, using the bluebook if necessary for space.

Incomprehensible answers get zero points! Precise answers that demonstrate clear understanding earn more points than “logically” correct but vague or uninformative answers. All technical words signify their meanings given in this subject.

Part A (62 points)

1. (3 points) What does a 2-year-old, properly trained (programmed) to run a semaphore in day-care, do when she wants to play with a toy but finds the toy box is empty? And who wakes her up?

2. (3 points) The memory management unit (MMU) is affected when the kernel performs a context switch. What is the purpose for changing the data used by the MMU?

3. (3 points) What does the scheduler do during a blocking system call? Say more than “it runs”.

4. (2 points) Inefficient use of a contiguously addressed memory or other contiguous data storage resource is the result of the first kind of “fragmentation”. The second kind of “fragmentation” signifies inefficient use of fixed size allocations of storage.

Which words are used to name (1) the first and (2) the second kind of fragmentation?

5. (4 points) For which processes or threads are the saved contents of the CPU registers and Program Counter (EIP) in the process table entry up-to-date (that is, they have meaningful values)? For which processes or threads are they not up-to-date, i.e., their values have nothing useful to do with the process or thread?

6. (12 points) Every system call has a system call service routine within the kernel. You put in the kernel a new system call service routine during a lab. The address of each service routine is stored in an array indexed by the system call number, so that the kernel can determine which routine to call from the system call number. The system call number was passed in the %eax register.

Simon Simpleton, the architect of the Open Operating Pits System (OOPS) figured that one step of indirection could be omitted by making the system call wrapper routine (of the C library) supply the address (in the kernel’s address space) of the system call service routine.

(a) Why must a syscall, trap, interrupt, etc. type of instruction still be used in the wrapper routine, rather than an ordinary function call? After all, a function call instruction simply uses the address of the function it calls.
(b) Why does Simon’s scheme require even more kernel overhead (more time needed for computing) between the trap, trap handler, etc. and actually calling the system call service routine?

(c) Why can’t the OOPS kernel be upgraded independently of the C library?

7. (6 points) Simon Simpleton seeing semaphores sought a silly solution in C:

```c
typedef unsigned int semaphore_t;
void init(semaphore_t *sem, unsigned int initval) {
    *sem = initval;
}
void take(semaphore_t *sem) /*also known as down()*/ {
    while(*sem == 0) /* wait busily */
        *sem = *sem - 1;
    return;
}
void put(semaphore_t *sem) /*also known as up()*/ {
    *sem = *sem + 1;
    return;
}
```

- Describe a failure due to a race condition, by giving a bad interleaving and demonstrating the resulting failure.
- What should a real semaphore do instead of busily wait, and then, what else should the `put()` method do?

8. (4 points) Java has a variation of monitors and condition variables. Here is an outline of a Java semaphore class. Recall there is one monitor associated with each instance of an class. The synchronized methods of the class called on that instance constitute the monitor’s procedures. Fill in the bodies of the `take` (down) and `put` (up) methods. Remember to write appropriate calls to `wait()` (which waits on the condition variable) and `notify()` (which signals the condition variable).

```java
class Semaphore {
    private int value;
    public Semaphore( int init ) {
        value = init;
    }
    synchronized public take( ) {
        // You code ... (Write answer in the bluebook.)
    }
    synchronized public put( ) {
```
9. (6 points) During normal operation, a multiprogramming (i.e., multitasking) with priority (round-robin for tasks of equal priority) and preemptive scheduling system sometimes changes which task or process it is currently running. Briefly indicate three situations that cause such a task switch to occur, besides a process exiting or getting killed.

10. (4 points) Explain the difference between a dirty page and a used (or referenced) page, and why it is important in virtual memory systems.

The next page is quoted from an Intel Pentium System Programming Manual. Use it to answer a question on the following page. That will be the last of the short answer questions.
To select the various table entries, the linear address is divided into three sections:

- **Page-directory entry**—Bits 22 through 31 provide an offset to an entry in the page directory. The selected entry provides the base physical address of a page table.
- **Page-table entry**—Bits 12 through 21 of the linear address provide an offset to an entry in the selected page table. This entry provides the base physical address of a page in physical memory.
- **Page offset**—Bits 0 through 11 provides an offset to a physical address in the page.

Memory management software has the option of using one page directory for all programs and tasks, one page directory for each task, or some combination of the two.

### 3.7.2. Linear Address Translation (4-MByte Pages)

Figure 3-12 shows how a page directory can be used to map linear addresses to 4-MByte pages. The entries in the page directory point to 4-MByte pages in physical memory. This paging method can be used to map up to 1024 pages into a 4-GByte linear address space.
11. (6 points) Refer to the diagram of Linear Address Translation from the Pentium III System Programming Manual. Suppose a linear address $0xABCD1234$ is translated to a physical address. Express your answers in hexadecimal:

(a) What numeric index into the Page Directory is used to obtain the relevant Directory Entry?

(b) What numeric index into the relevant Page Table is used to obtain relevant Page-Table Entry?

(c) Assuming the relevant Page-Table Entry specifies a legal page frame, and the page base physical address stored in that Page-Table Entry equals $0x$BEF80000, what is physical address corresponding to the given linear address?
Part B (24 points) Page Replacement Algorithms

Assume that the memory size is 4 page frames.

Demonstrate each of these 3 page replacement algorithms:
(1) FIFO. (2) LRU. (3) 2nd chance or “Clock”
on the following reference string of page numbers below. For each algorithm, mark
with a “P” those references that cause page faults. For each algorithm and for each
reference, show the page numbers of the (up to 4) pages that are in memory after any
replacement is performed. Thus the answer for the first 3 lines for all algorithms is (0),
(0 8), (0 8 2).

For all the algorithms list the in-memory page numbers in the order they are kept in
the data structure that the algorithm uses.
Show your work either on this sheet or in the bluebook, or both. Neatness counts!

Your Name, if you will hand this sheet in ___________________________
The objective of this problem is to plot on a sequence diagram what happens in a single CPU system when it runs three jobs (or processes). All times are in units of milliseconds (ms). Make the following assumptions:

- All disk operations require 2.5ms to complete. A job that requests a disk operation blocks until the operation is complete. Concurrent disk operations do not interfere with each other because they use different disks “DiskB” and “DiskC”. Of course, disk operations of one job are performed in parallel with the CPU usage of other jobs.
- Assume the time for scheduling and context switching is negligible.
- Jobs A, B, and C enter the system at approximately the same time, with A first, B second and C third.
- Each column in the table below specifies how time would be used if the corresponding job were run in a dedicated system. Note that the CPU is idle for the indicated periods of disk usage when jobs B or C are run on a dedicated system.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 ms CPU</td>
<td>1 ms CPU</td>
<td>1 ms CPU</td>
</tr>
<tr>
<td></td>
<td>2.5 ms DiskB</td>
<td>2.5 ms DiskC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 ms CPU</td>
<td>1 ms CPU</td>
<td>2.5 ms DiskC</td>
</tr>
<tr>
<td></td>
<td>2.5 ms DiskB</td>
<td></td>
<td>2.5 ms DiskC</td>
</tr>
<tr>
<td></td>
<td>3 ms CPU</td>
<td></td>
<td>1 ms CPU</td>
</tr>
</tbody>
</table>

- When a job first enters the system, or it becomes ready after it has been blocked, it is put in the ready list queue at the end. So, the currently running job is not preempted when a new job is started or becomes ready. (Preemption only occurs when the job’s quantum expires.)

Draw the sequence diagrams on the printed grids. Clarity counts! Ambiguity will be counted against you.

**Question 1:** Draw the sequence diagram assuming that First Come First Serve scheduling is used. (When a process becomes ready after being blocked, it is treated as if it just arrived.) It would help to draw additional “tracks” to represent disk activity.

**Question 2:** Draw the sequence diagram assuming that Preemptive Round Robin scheduling is used with a fixed quantum of 1ms.

**Question 3:** Draw the sequence diagram assuming that at any time, the job runs is the one that will spend the least time using the CPU until it either blocks or exits. (This minimizes the average turnaround time, but generally cannot be determined before running the jobs.)