CS 140 Midterm Examination
Spring Quarter, 2010

You have 1.5 hours (90 minutes) for this examination; the number of points for each question indicates roughly how many minutes you should spend on that question. Make sure you print your name and sign the Honor Code below. During this exam you are allowed to consult a single double-sided page of notes that you have prepared ahead of time; other than that, you may not consult books, notes, or your laptop. If there is a trivial detail that you need for one of your answers but cannot recall, you may ask the course staff for help.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination, and I have not consulted any external information other than a single double-sided page of prepared notes.

_____
(Signature)

_____
(Print your name, legibly!)

_____
(email id, for sending score)

Problem	#1	#2	#3	#4	#5	Total
Score						
Max	18	15	9	10	35	87

Problem 1 (18 points)

Indicate whether each of the following statements is True or False, and explain your answer *briefly*.

(a) Reducing the length of time slices could worsen a system's average response time.

(b) In a machine without dynamic address translation, it isn't possible to move a process in memory once it has started executing.

(c) If synchronization is used properly, a system's behavior will be the same regardless of scheduling policy.

(d) In a system with only independent threads, deadlock cannot occur.

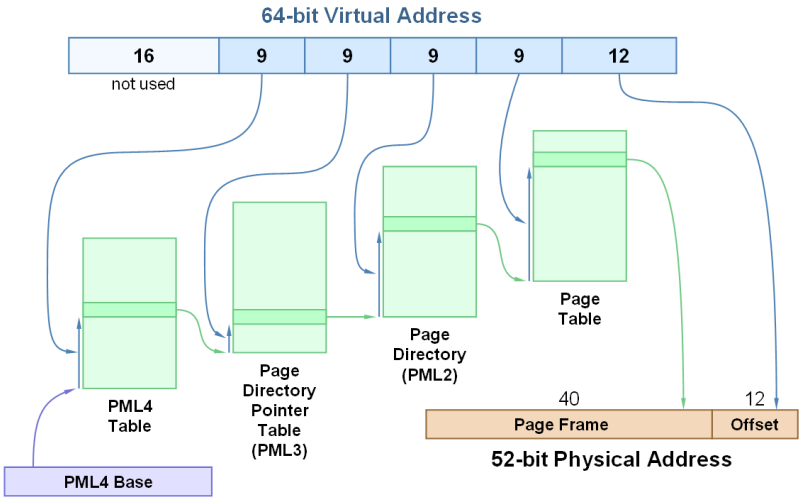
(e) From the standpoint of a developer writing a multi-threaded application using existing primitives such as locks and condition variables, it is harder to write such applications for a multiprocessor than a uniprocessor.

(f) From the standpoint of an operating system programmer, it is harder to implement synchronization primitives (such as locks and condition variables) for a multiprocessor than a uniprocessor.

Problem 2 (15 points)

The following questions concern virtual memory.

(a) The figure below shows the address translation mechanism used for 64-bit mode in Intel x86 processors. How many distinct bytes of virtual address space can be referenced using a single page directory (PML2 table)?



(b) Consider the following hypothetical machine instruction:

```
ADD A, B, C
```

A, B, and C are all memory addresses. The instruction reads the words at the locations given by A and B, adds them, and stores the resulting word at location C. Does this instruction present any challenges for restarting after a page fault? Why or why not?

(c) Consider the following hypothetical machine instruction:

```
POPJUMP R2, A
```

In this instruction A is a memory address and R2 selects register 2. The instruction reads the word whose location is given by the stack pointer register and stores the value of that word in register 2. Then it adds 4 to the stack pointer register and jumps to the location given by A. Does this instruction present any challenges for restarting after a page fault? Why or why not?

Problem 3 (9 points)

For each of the scheduling algorithms listed below, indicate whether or not the algorithm is preemptive. If it is preemptive, describe briefly the conditions under which preemption occurs.

(a) First come first served

(b) Round robin

(c) Shortest time to completion first

Problem 4 (10 points)

Answer each of the following questions in a few sentences.

(a) Why does a linker require two passes over the object files?

(b) What is the difference between internal fragmentation and external fragmentation?

Problem 5 (35 points)

CalTrain has decided to improve its efficiency by automating not just its trains but also its passengers. From now on, passengers will be robots. Each robot and each train is controlled by a thread. You have been hired to write synchronization functions that will guarantee orderly loading of trains. You must define a structure `struct station`, plus several functions described below.

When a train arrives in the station and has opened its doors, it invokes the function

```
station_load_train(struct station *station, int count)
```

where `count` indicates how many seats are available on the train. The function must not return until the train is satisfactorily loaded (all passengers are in their seats, and either the train is full or all waiting passengers have boarded).

When a passenger robot arrives in a station, it first invokes the function

```
station_wait_for_train(struct station *station)
```

This function must not return until a train is in the station (i.e., a call to `station_load_train` is in progress) and there are enough free seats on the train for this passenger to sit down. Once this function returns, the passenger robot will move the passenger on board the train and into a seat (you don't need to worry about how this mechanism works). Once the passenger is seated, it will call the function

```
station_on_board(struct station *station)
```

to let the train know that it's on board.

Use the next pages to write a declaration for `struct station` and the three functions above, plus the function `station_init`, which will be invoked to initialize the station object.

- You must write your solution in C using the Pintos functions for locks and condition variables:

  ```
  lock_init (struct lock *lock)
  lock_acquire(struct lock *lock)
  lock_release(struct lock *lock)
  cond_init(struct condition *cond)
  cond_wait(struct condition *cond, struct lock *lock)
  cond_signal(struct condition *cond, struct lock *lock)
  cond_broadcast(struct condition *cond, struct lock *lock)
  ```

  Use only these functions (e.g., no semaphores or other synchronization primitives).
- You may assume that there is never more than one train in the station at once, and that all trains (and all passengers) are going to the same destination (i.e. any passenger can board any train).
- Your code must allow multiple passengers to board simultaneously.
- There must not be any busy-waiting in your solution.

This page is for your solution to Problem 5.

Additional working space for Problem 5, if needed.