SYSTEM PROGRAMMING FINAL EXAM **January 18, 2010**

2009-2010 FALL **120 minutes**

You are not allowed to ask any questions during the exam. Answer the questions to the best of your understanding. If you need to make any extra assumptions, state them clearly. Explain all your answers.

CRN	No	Name	Signature	Q1	Q2	Q3	Q4	Total
				/25	/25	/25	/25	/100

1. *ANSWER THIS QUESTION ON THE FRONT SIDE OF THE EXTRA PAPER.*

```
struct int_array {
  int *x;
  int n;
};

int sum(struct int_array p);

int main(void) {
  int points[] = { 2, 3, 1, 2, 5 };
  struct int_array a = { points, 5 };
  printf("%d\n", sum(a));
  return 0;
}
```

In the given C program, the *int_array* structure consists of a pointer to an array of integers (x) and the number of elements (n) in that array. The *sum* function returns the sum of the array it receives as parameter. Write the *sum* function in NASM. (*Note*: Assume that an integer is 4-bytes long.)

2. *ANSWER THIS QUESTION BELOW.* In your projects for this course, you have implemented operating system functionalities using three different approaches: by building into the kernel, by developing as a kernel module, and by implementing in the user space. For each of the following projects, state which approach*es* are suitable and which one you would select. Justify your selection.

   (a) adding a new file system

   (b) implementing a new scheduler

   (c) adding support for a network device

3. *ANSWER THIS QUESTION BELOW.* The kernel provides three mechanisms for mutual exclusion: atomic operations, spin locks, semaphores.

   (a) Explain how each of these mechanisms works.

   (b) When should each mechanism be preferred over the others?

4. *ANSWER THIS QUESTION BELOW.* Consider the functions given below which are responsible for handling the *open* and *read* system calls on the device "foo". *foo_u_owner*, *foo_u_count* (initial value 0) and *foo_r_count* (initial value 65) are global module variables. (*Note*: Assume that an integer is 4-bytes long.)

```
int foo_open(inode, filp)
{
  ...
  if ((foo_u_count > 0) &&
      (foo_u_owner != current->uid))
    return -EBUSY; /* device busy */
  if (foo_u_count == 0)
    foo_u_owner = current->uid;
  foo_u_count++;
  ...
}
```

```
int foo_read(filp, buf, count, f_pos)
{
    copy_to_user(buf, &foo_r_count,
                 sizeof(int));
    foo_r_count++;
    return sizeof(int);
}
```

   (a) Explain the restrictions when opening this device.

   (b) Assume that there has been no read operations on the device before. What will be the contents of the read buffer of a user-space process when it tries to read 12 bytes from this device? Explain your answer. (*Hint*: Remember partial reads.)