Name: _____ **Student ID #:** _____

Lab #8: COMP 3000B (Operating Systems)
March 14 & 16, 2006

Please answer the following questions. All programs given have assicated `build.bat` files which can be used to build them. **Please shut down your computer when you are done.**

This lab focuses on system calls and hardware interrupts. We will be using FreeDOS, an open source clone of MS-DOS. Although some features of FreeDOS are very different from Linux, because of hardware constraints, system calls and hardware interrupts work very similar across the two operating systems. The only difference is that memory management in Linux makes things slightly more complex. Linux also passes parameters to the kernel via the stack as opposed to in registers.

In FreeDOS, system calls are activated through signaling an interrupt 0x21. When 0x21 is signalled, the operating system determines what functionality is required by examining the AH register. Some common system calls are listed below:

AH	**System Call**
0x00	Terminate Program
0x02	Write Character to Standard Output
0x06	Direct Console Output
0x07	Direct Character Input with Echo
0x09	Write String to Standard Output
0x20	Set Interrupt Vector
0x31	Terminate and Stay Resident
0x35	Get Interrupt Vector
0x38	Get Country-Specific Information
0x40	Write to File or Device
0x43	Get/Set File Attributes
0x4A	Resize Memory Block

After a system call is complete, the results are returned in the registers for the program to deal with. As an example, for 0x35, the interrupt vector requested is returned in ES:BX.

The display can be accessed by system calls to FreeDOS, but can also be accessed through system calls to the BIOS (interrupt 0x10) and directly by writing to video RAM.

1. (2 pt) In the *comp3000/tsr* directory is a terminate and stay resident (TSR) program. A TSR installs code into the FreeDOS kernel and then exits. The installed code in this case intercepts all FreeDOS system calls and prints out the AX register associated with the call. It also calls a C function. Compile and run the program. You will notice the AX register being printed in the lower right corner of the screen. System functions are differentiated by the AH value. Next, run the edit program. What are two calls being made by edit, and how did you determine this?

2. (1 pt) The TSR works by changing the interrupt vector for 0x21. It runs any specific code applicable to the TSR and then calls the original 0x21 interrupt service routine (which is saved somewhere in our TSR). If another program was installed, it would take over the interrupt vector and then call our TSR when it was finished processing. Given this setup, what is one difficulty with unloading our TSR?

3. (1 pt) Because of the speed in which system calls are fulfilled, not all system calls are visible. Modify the doInt21 C function to impose a delay on system calls and then run the dir command. What system call listed above is being used to print to the screen? (You will need to reboot in order to get rid of the installed TSR kernel extension).

4. (1 pt) Given that calling FreeDOS to write AX to the screen during processing a system call would cause an infinate loop, how does the main.asm code write to the screen without calling FreeDOS? (You will need to examine the code to answer this question).

5. (1 pt) If the C function wanted to access any data, how would space for that data need to be allocated?

6. (2 pt) In the *comp3000/timer* directory is a program which hooks into the timer hardware interrupt. By examining the source and running the program, we can see that it performs multithreading under FreeDOS. Video output is done by directly accessing video hardware. Because FreeDOS is not a multithreaded OS, what is one problem that would occur if trying to access any FreeDOS functions in the threads? How could this problem be solved?

7. (1 pt) Can a hardware generated interrupt return a value in a register like a software generated interrupt (system call)? Why or why not?

8. (1 pt) Is it easier or harder to debug a hardware interrupt service routine compared to a normal program? Why?