Assignment 6

Due October 28 (?), 1996

This assignment asks you to write a program that includes a simple interrupt service routine for the PC keyboard.

Question 1

Write an 8088 assembly-language program that does the following:

- prints your name and student number.
- initializes a flag variable to zero.
- saves the previous value of the keyboard interrupt vector by PUSHing the old CS and IP values on the stack (not by using MOV instructions as in the example in the lecture notes).
- sets up the keyboard interrupt vector to point to an ISR that you'll also write (see below).
- loops continuously until the flag variable becomes non-zero. The value of the flag should not be changed by instructions in the loop.
- restores the previous interrupt vector and returns control to DOS using interrupt 20H.

Write an interrupt service routine for the keyboard interrupt that does the following:

- saves and initializes any registers that need to be saved. If a stack is used it should be a local stack, not the stack of the called program.
- reads a byte from I/O port address 60H. This is the keyboard data port.
- if the value read is 81H (the scan code generated by releasing the ESCape key) the flag variable described above should be set to 1.
- gives an EOI command to the PIC, restores all registers that were saved on entry to the ISR and returns from the interrupt.

Note that your program *must* use an ISR that reads the keyboard port and tests the scan code directly. *Do not* use BIOS or DOS routines to read the keyboard.

Assemble the main program and the ISR into an executable .COM file and test it by running it and then pressing various keys including the ESC key. Your program should terminate only when the ESC key is released.

Submit your assembly language and executable files electronically. Use the key `6asm` for the assembly language file and `6com` for the executable file. To transfer the files from DOS to the Unix systems you can use the ftp command if both machines are networked or the mcopy utility if both machines have 3.5 inch floppy disks. You need not convert the text file to Unix format.

Note that debugging ISRs can be difficult because often: (1) the exact flow of control is unclear, (2) errors cause the computer to crash, and (3) debuggers can't trace into the ISR. You may find it more efficient to start with a simple program and gradually add functionality.