COMP 308
Software Systems Lab

Assignment #1

Due: February 3, 2018 on My Courses at 23:30

Our first assignment will not focus on graphics. Instead, it will focus on learning how to interact with the INTEL computer at the assembler level. This will prepare us for the graphics programming we will do later.

Question 1: DEBUG & DOSEMU

(A) On your computer or in the lab TR3070, install DOSBOX (or DOSEMU if that works better on your computer), if it is not already present. Then download the program DEBUG, from our course My Courses Lab1 section, and verify that you can run it. You should have already done this in your lab.

(B) Using DEBUG do the following:

- Write in a text-file, called A1Q1.TXT, the commands in the order that you would type them using DEBUG to switch the video mode from text to any one of the graphics modes that support ASCII text and then back to text again. You should have also done this task in your lab.

- In DEBUG, figure out how to write the letter 'A' using the OS interrupts when the video screen is in text mode and when the video mode is in one of the graphics modes that support ASCII text. Our lecture slides and labs covered this. Add these instructions to the A1Q1.TXT file.

Question 2: Input and Output Functions in INTEL Assembler

(A) If you are working from home you can download the TASM assembler. You can find it on the My Courses Lab2 section.

(B) I/O in assembler is always problematic. Having good I/O libraries is always useful. Create the following I/O subroutines using the INTEL assembler language. Save this in a text file called IO.ASM. For testing purposes IO.ASM will also have a main program.

The four subroutines are: *getche*, *putch*, *gets* and *puts*.

The subroutine *getche* has the following C signature: char *getche*(void)
When *getche* is invoked it reads a single character from the keyboard, echoes that character to the screen, and returns the character to the calling program. You can use the OS interrupts (refer to the third lecture to see how do this).

The subroutine *putch* has the following C signature: void *putch*(char c)
When *putch* is invoked it receives a single character from the calling program. It displays that character on the screen at the current cursor location. It then returns to the calling program. You can use the OS interrupts. The cursor is updated to the next screen location (refer to the third lecture to see how do this).

The subroutine *puts* has the following C signature: void *puts*(char *p)
When *puts* is invoked it receives a pointer to a string. It assumes the string terminates with '\0'. It displays all the characters in the string by invoking *putch()* in a loop. It then terminates.

The subroutine *gets* has the following C signature: void *gets*(char *p)
When *gets* is invoked it receives a pointer to a location in memory that is ready to receive a string. The subroutine *gets()* invokes *getche()* in a loop receiving characters from the user and saving each character within the memory space pointed to by p. When the user presses the enter key *gets()* stops processing. Your program adds A '\0' character to the end of the string, include the enter key in the resultant string.

For all of these programs use proper subroutine invoking techniques. This means that the program calling a subroutine must first push all the arguments onto the run-time stack and then invoke the subroutine. The subroutine must stack all registers it plans to use at the beginning of the subroutine. If the subroutine needs access to local data structures then those need to be created in the run-time stack. At the end of the subroutine all the saved registers are popped back. If the subroutines has a return type, that value is placed in the D register. The program that called the subroutine uses the D register as the returned result (if present).

Submit the following:

- Create a source file called IO.ASM.

- In your source file write the above four subroutines as described.

- In IO.ASM create a main program that tests your subroutines by asking the user to input their name. It uses 'puts' to ask for input and 'gets' to read in the name. The program then prints out the inputted name using 'puts' to confirm that it did it correctly. The program then terminates.

HOW TO HAND IT IN

Submit IO.ASM and A1Q1.TXT to MyCourses. Make sure this is DOSEMU compatible and executable in the Trottier 3rd floor labs.

HOW IT WILL BE GRADED

This assignment is worth a total of 20 points.

- Question 1
  - Video mode +3
  - Character output +3
- Question 2
  - getche +4
  - putch +4
  - gets +2
  - puts +2
  - main program +2

GRADING RULES

- You can submit your solutions with at most two days delay. 15% reduction will be applied to your grade for each day delay.
- Your program must be run to be graded, or you will get zero.
  - If you program runs but does not run correctly, you will get partial marks
  - Partial marks are awarded proportionally, meaning that it is only based on the parts of the program that you actually did complete correctly.
- Essay / Written questions are also graded proportionally. This is defined as points being awarded to only those portions of the question that were answered correctly.
  - If you answered, for example, only half of the questions and they were all correct then you get half of the points.