COMP 308
Software Systems Lab

Assignment #2

Due: Saturday February 28th, 2017 on myCourses at 23:30

Question 1: Expand the IO.ASM File from Assignment #1

In assignment #1 you created four ASCII based I/O functions to read/write characters and strings. I am asking you to expand your IO.ASM file to include two additional I/O functions to read/write integer numbers. Both these functions must follow the C calling convention.

You will submit the IO.ASM file containing all the functions from assignment #1 plus these two additional integer-based I/O functions for assignment #2.

Write a main program to test the two new integer-based I/O functions. Your main program will ask the user to enter three single digit integer numbers. The first number, called A, will be read in using the getche() function from assignment #1. In other words it will read the number as an ASCII characters. The next two numbers, called B and C, will be read in using the getInt() function from this assignment. To validate your inputs calculate and display the following operations: A-B and B-C. If A, B and C are all the same digit, let us say the number 7. Then, B-C should display the number zero, while A-B should not. You do not need to store the numbers as A, B and C. I used these letters to help explain what I want you to do. You may store them as you wish.

Hint: Our lecture slides have some partial examples of reading and writing integer-based numbers.

(A) Create a read integer function, following the C calling convention, called:

AX getInt(void)

It will return the integer number, not the ASCII number, input by the user to the AX register. It takes no arguments.

(B) Create a print integer function, following the C calling convention, called:

void printInt(int)

This function will be passed an integer number through the stack. This will be a true binary integer number. The function will need to convert the number into ASCII and use the putch() function, from assignment #1, to print out each digit.

Question 2: Graphics Card Information

As we saw in lecture #3 and 5 you can ask your graphics card for information about itself. Write an assembler program that does this within DosBox or DosEMU and see what it tells you. Give me an interpretation of the results you received. Googling will help you.

This question asks you to write an new ASM program. To do this copy your IO.ASM program from question 1 from this assignment into a new file called A2Q2.ASM. Change the main program to do the following:

1. Write a program that asks the graphics card for its information (see class notes). This includes all the information in SVGA_INFO and SVGA_ModeInfo.
2. Display this information on the screen using your IO.ASM routines. Display the first 4 fields of information of SVGA_INFO and the following subset from SVGA_ModeInfo, specifically: Xresolution, Yresolution, XcharSize, YcharSize, BitsPerPixel, NumberOfBanks, and MemoryModel. Make the output easy to read.
3. Provide a one or two line interpretation for all the results displayed. For example, are the values returned valid, why/why not.
4. Using the command line redirection operators redirect the output from step 2 to a text file. Then edit that file adding your interpretation. A DOS redirect should be able to do it. Call this text file A2Q2.txt.

Question 3: Graphics Library (part 1)

Let us begin to create a graphics library file called GRAPHICS.ASM. Create the following:

- void drawPixel(int x, int y, int colour)
  Following the C calling convention, this function expects three integers as stacked arguments: the first two arguments are the coordinates on the screen where a single pixel will be turned on. The last argument is the colour the pixel will be set to. The function does not return anything. Use the slides from class to help you construct a general dot program. This does not include palette and it does not include planes.

- A main program.
  The main program will be used to test the drawPixel function. Select an intermediate graphics mode that runs in the Trottier lab and have this main program activate that mode. Then using the drawPixel function multiple times plot a horizontal line 50 pixels long and only 1 pixel wide. The line can be anywhere on the screen. The graphics mode you select must be intermediate or lower to be able to see such a narrow line. Using the interrupts (or your IO.ASM library). The user must press enter to continue after the line was drawn. This will let us see your line before the main program switches back into text mode and terminates.

WHAT TO HAND IN

Submit IO.ASM, A2Q2.ASM, A2Q2.txt and GRAPHICS.ASM to My Courses. Also include the compiled version of your programs. Make sure this is DOSBOX / DOSEMU compatible and executable in the Trottier Windows lab. Make sure to have a readme file to indicate the program`s compatibility with either DOSBOX or DOSEMU.

HOW IT WILL BE GRADED

This assignment is worth a total of 20 points.

- Question 1 (7 points)
  - Integer read +3
  - Write integer +3
  - Main program test +1
- Question 2 (7 points)
  - The assembler program +4
  - The interpretation +3
- Question 3 (6 points)
  - The drawPixel function +3
  - The main program +3
  -

GRADING RULES

- Late penalty policy: each day 15% deduction up to two days, afterward zero.
- Your program must run to be graded, or you will get zero.
  - If you program runs but does not run correctly, you will get part marks
  - Part 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.