Karl Rouhana 260948648

ECSE 324 Lab 3 Report

Part 1: Drawing things with VGA

1. My approach

For VGA_draw_point_ASM, I left shift *x* by 1 and *y* by 10 before adding them to the correct address to get to the right coordinate.
In the VGA_write_char_ASM, I first start by making sure that *x* and *y* are in the correct range. After that, I go to the specified x and y position by moving the pointer to the right coordinates. I add *x*, and then by $y*2^7$. Once there, I store the character that I want to the new location at the right address.
To clear the pixels and the characters, I implemented a common approach. I use 2 loops; one that goes over all the correct *x* coordinates and one that goes over all the *y* coordinates and then calling the right subroutine (write_char or draw_point)

2. Challenges faced

The biggest challenge was to understand how to access and correctly change the pixels and how to clear them. After having read the DE1-SoC manual and the lab manual a couple of times, I had a better idea of what I should do. I understood how I should access the memory to change each coordinate.

3. Possible improvements

I think one area where we can improve on is to check the coordinates are correct and legal, i.e., exception handling.

Part 2: Reading keyboard input

1. My approach

In my read_PS2_data_ASM subroutine, I first start by checking if RVALID is 1 by doing an AND between the number 1000000000000000 and the data of the PS2.
If the result is 1, the input is valid, we return 1 in R0 and store the data of PS2 into the location specified by R0.
If the AND results in a 0, which means that it isn't valid, we return 0 in R0.

2. Challenges faced

This part was simple, so I did not face any major challenges.

3. Possible improvements

Since the driver I had to implement was straight forward, I don't think there's any major improvement to be made. In addition, I'm also using the previous part's code, the same improvements I mentioned prior also apply.

Part 3: Putting everything together: Vexillology

1. My approach

Of course, I based myself on the Texas flag code that was already given, so I could setup for my imaginary flag. I just change the colors of the rectangle and the stars. For my real-life flag, I decided to create the flag of France since the colors needed were already present. I based myself of the ranges of x and y on the VGA screen to divide the latter in 3 equal rectangles. Same went for the height and width of the flag.

2. Challenges faced

This part was fairly simple, so I did not face any major challenges.

3. Possible improvements

Since the driver code was already present, and I could base myself on this, I don't think there's any major improvement to be made. In addition, I'm also using the previous parts' code, the same improvements I mentioned prior also apply.