

ECSE 324: Computer Organization

Lab3 Report

Part 1: Drawing things with VGA
Part 2: Reading keyboard input
Part 3: Vexillology

Grey Yuan (260857954)

Part 1

Approach

In this part, we need to implement a driver which is a set of functions that control the screen. First, we need to implement `VGA_draw_point_ASM` which will draw one point on the screen using specific location and color. We left shift the x coordinate by 1 and y coordinate by 10. Then we load the pixel buffer memory to R3 and add the coordinates to R3 which will be storing the address of the pixel we desired to draw. Second, we want to implement `VGA_clear_pixelbuff_ASM` which is used to clear the pixel on the screen. As suggested by the hint, we use `VGA_draw_point_ASM` implemented previously to set the pixel to zero. To do that, we use two subroutines called `colorForLoop1` and `colorForLoop2` and set each pixel to zero. Then we will go to the subroutine “finish” to end the process. The next two subroutines for characters are very similar to the two for pixel. Third, we need to implement `VGA_write_char_ASM` which will draw a character on the screen using specific location and specific ASCII code. We left shift the y coordinate by 7. Then we load the character buffer memory to R3 and add the coordinates to R3 which will be storing the address that we desired to draw. Last, we want to implement `VGA_clear_charbuff_ASM` which is used to clear the character on the screen. As suggested by the hint, we use `VGA_write_char_ASM` implemented previously to set the char to zero. To do that, we use two subroutines called `charForLoop1` and `charForLoop2` and set each character to zero. Then we will go to the subroutine “finish” to end the process.

Testing

After hit run and compile and click continue, we should see the following in the VGA screen.



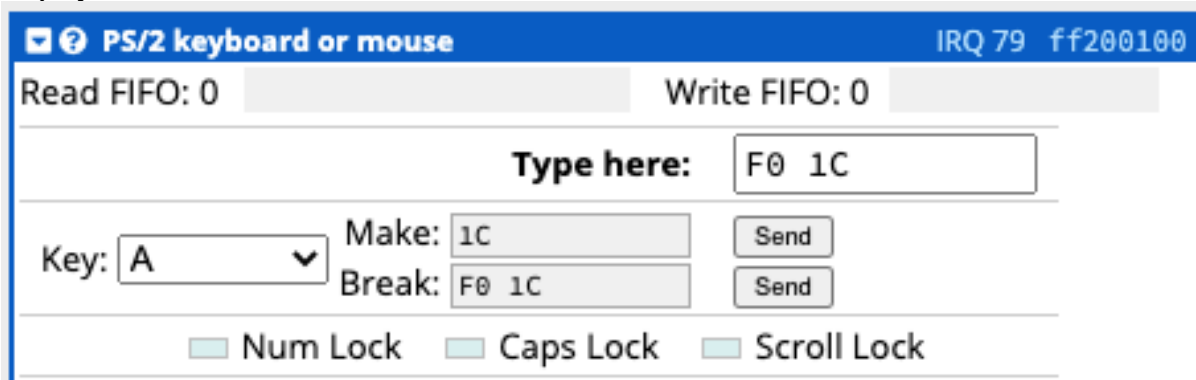
Part 2

Approach

In this part, besides using the code in part 1, we need to implement `read_PS2_data_ASM` which will check the RVALID bit in the PS/2 Data register. We first will load PS2_MEMORY to R1. Then, we load the value in R1 into R2 and compare the value of it after right shifting by 15. By doing that, we can check if RVALID bit is enabled or not. If not will go to subroutine “notValid” and return its value 0. If it is enabled, we will proceed to set R0 to 1 and return the value 1.

Testing

After running the code, we should see just dark in VGA screen. Then, we press random keys on the keyboard. For example, we press A in “Type here” textbox, we will get the following display:



Then, if we go to VGA to check the make code and break code should appear:



We should be able to repeat the step for many times.

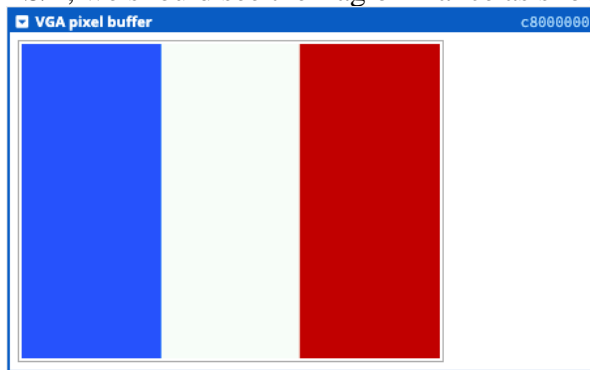
Part 3

Approach

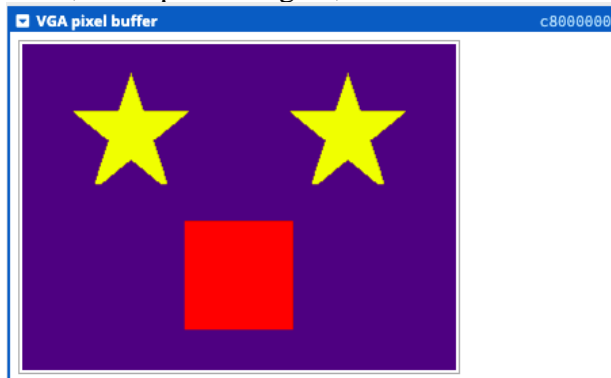
For this part, we need to draw a flag a real country and a imaginary flag. To do that, we first copy the code for the previous two parts. Then, we try to draw the flag of France first. We first draw the blue rectangle by setting the location to (0,0), the height to 240, and the width to 107. Then draw the white rectangle by setting the location to (107,0), the height to 240, and the width to 107. Similarly, draw the red rectangle with similar setting besides the location is at (213,0). Second, we want to draw a flag of my own design. I want to draw a robot face as a flag design, so I first set the background to purple by drawing a rectangle with size 240*320 at (0,0). Then draw two yellow stars as the robot's eyes at location (80,60) and (240,60). Next, draw a red rectangle as the robot's mouth at (120,130) with size 80*80.

Testing

After running the code, we should see the flag of Texas is drawn first. Then, if we press D in PS/2, we should see the flag of France as shown below.



Then, if we press D again, we can see the robot flag:



Also, we should be go back to the previous flag by pressing A.