ECSE 324 LAB 3 REPORT

RAFID SAIF

Part 1: Drawing Things With VGA

Description:

The task for this part was to write drivers for the VGA display to draw points and characters, as well as clear the points and characters on the display.

Approach Taken:

For this part, the functions for both the points and characters are nearly identical. To draw, we first add the offset based on the desired coordinate, so we know which space in memory to write to. In doing so, we shift the first two arguments and add them as specified in the manual. We then write the content of the third argument into the memory space. To clear, we simply create a nested loop. We first loop through x, calling the corresponding draw function with zero as the third argument. At the end of our x loop, we set x back to zero, increment y and then loop back through x. We continue until we reach the end of the y.

Challenges Faced:

Understanding the importance of using STRH and STRB in the draw and write functions. It was also a slight challenge to work with a nested for loop, however our experience from the second part of lab 1 was useful here.

Part 2: Reading Keyboard Input

Description:

The task for this part was to input for the keyboard from the PS/2 bus.

Approach Taken:

The task here is straightforward and described in the lab manual. We first shift the data on the PS/2 data register by 15 bits to the right. If that bit is 1, we read the last 8 bits of the keyboard. Using STRB, we store the content of those bits into the memory address provided in R0.

Challenges Faced:

No challenges faced for this part, all instructions were given in the manual and the remaining part was similar to part 1.

Possible Improvements:

When we check RVALID, we could store the content of the data register just before we perform the check. In some cases, by the time we check for RVALID, we may lose the original data and get new data.

Part 3: Tic-Tac-Toe Game

Description:

The task for this part was to implement a working Tic-Tac-Toe game using the drivers we created in the previous parts.

Approach Taken:

We first wrote the subroutines to display the turns of the players, the result, a square and a plus sign. We also wrote subroutines to fill the display with a solid color and then draw a grid on the colored display. Afterwards, we proceed to write the actual game. First, we use the PS/2 driver to poll for a keypress of the '0' key. When this key is pressed, we start the game with player one. We first perform a logical OR of the two players' marks. Every time we loop, we check if all the spots are occupied. If they are, we call a draw using the result subroutine. We then check the key press against this, to check if the player is trying to mark an occupied spot. If the spot is free, we proceed to draw a square or a plus depending on the player turn, and add the spot to one-hot encoded score for that player. We then check the marks for that player against all possible winning combinations. To do this, we AND the marks with each combination and then compare the result against that winning combination. If a win is detected, we pass the corresponding argument into the result subroutine and call a win. Otherwise, we change the player turn and loop back to see if all the spots are occupied. If not, we continue the game.

Challenges Faced:

We faced a challenge in checking for the winning combination. It is not immediately evident that we need to compare against the result of an AND operation.

Possible Improvements:

When checking for winning combinations, we can create a subroutine and call it for different combinations. We can also reuse this for both players.