Implementation report

MIPS CPU

Lidor Cohen 305782732

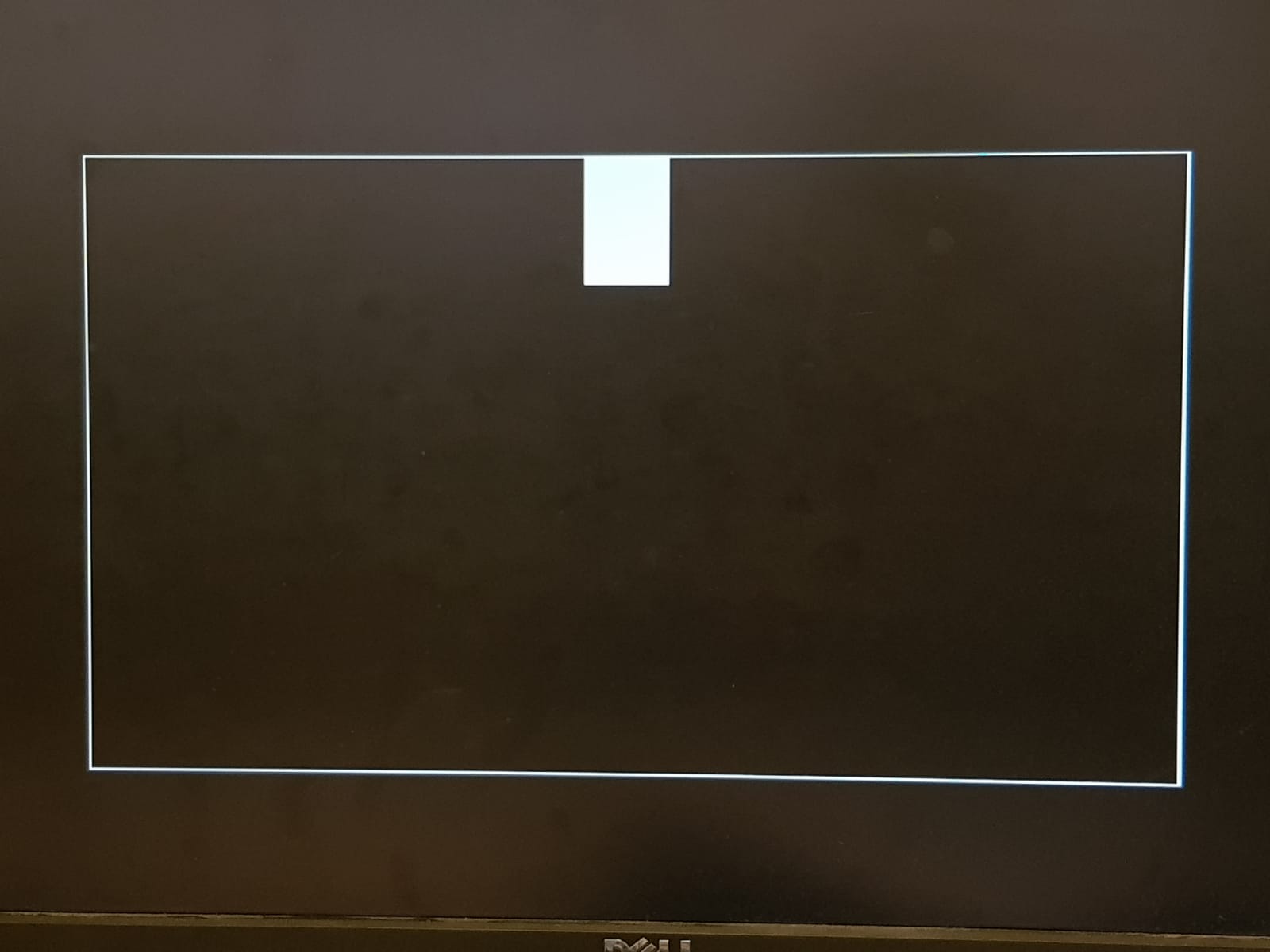
Otabek Sherman 326868775

4.1)

The register $2 starts with 2000h then we multiply it by 2^16, it means we make shift left 16 times and the value of $2 changes to 0x20000000. Next we add 18h to register $2, and current of $2 is 0x20000018

4.2)

Between the cycles 120 and 130 the program drawing the white vertical rectangle in the top middle part of the black screen. Drawing running while the value of the register $1 not equal zero.



The printed rectangle

4.3)

When we press the RUN button the program prints the rectangle on the screen.

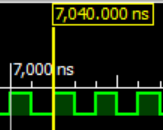
4.4)

The explanation of the HW5\_rect4 program:

1. The register $1 that holds the number of lines, initialized with value 64.
2. The register $2 that holds a pointer to pixel of the screen, initializes with value 0x2000h
3. The register $3 holds a color value of the pixel which is the 32 bit value, initialized with value 0xFFFFFFFF
4. Program entering to the loop that prints the value of the register $3 to the address $2, and running while the value of the $1 is not equal zero. In every iteration we decrement the value of the $2 by 52 and continue to draw the next line.

4.5)

In this program we have 7 commands – sw, addi, addi, nop, nop, bne, nop. Every command includes the IF, ID, EX, MEM and WB, which takes 5 clocks.



Every clock runs for 40ns.

As a result, in total the drawing of the square of size 32x32 will take 7 \* 40 \* 32 = 8960ns

4.6)

We can short the loop by decrease the numbers of commands from 7 to 6 by keeping 2 nop commands from bne command of the last addi command. We will remove one addi command by adding the check in bne command.

…

Subi $3, $2, 832

Loop:

Sw $3, 0($2)

Addi $2, $2, 52

Nop

Nop

Bne $2, $3, drawlp

Nop

4.7)

To draw the square faster in the same loop we can use sw commands in the loop:

Subi $3, $2, 832

Loop:

Sw $3, 0($2)

Sw $3, 52($2)

Addi $2, $2, 52 \* 2

Nop

Nop

Bne $2, $3, drawlp

nop

We also can use another way to draw the square faster. We can map the area of the screen to the memory region and create the square, by writing one time the same region. We will use a single sw operation.

Addi $3, $0, -1

Sw $3, 0($2)