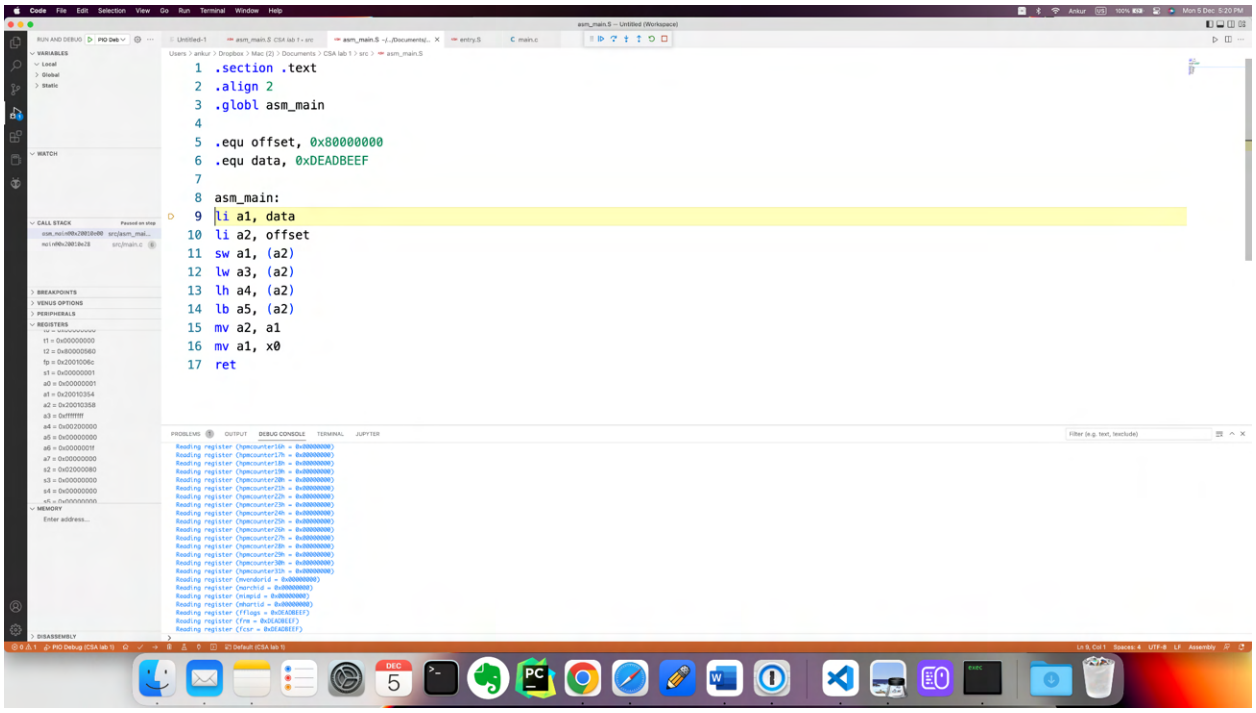
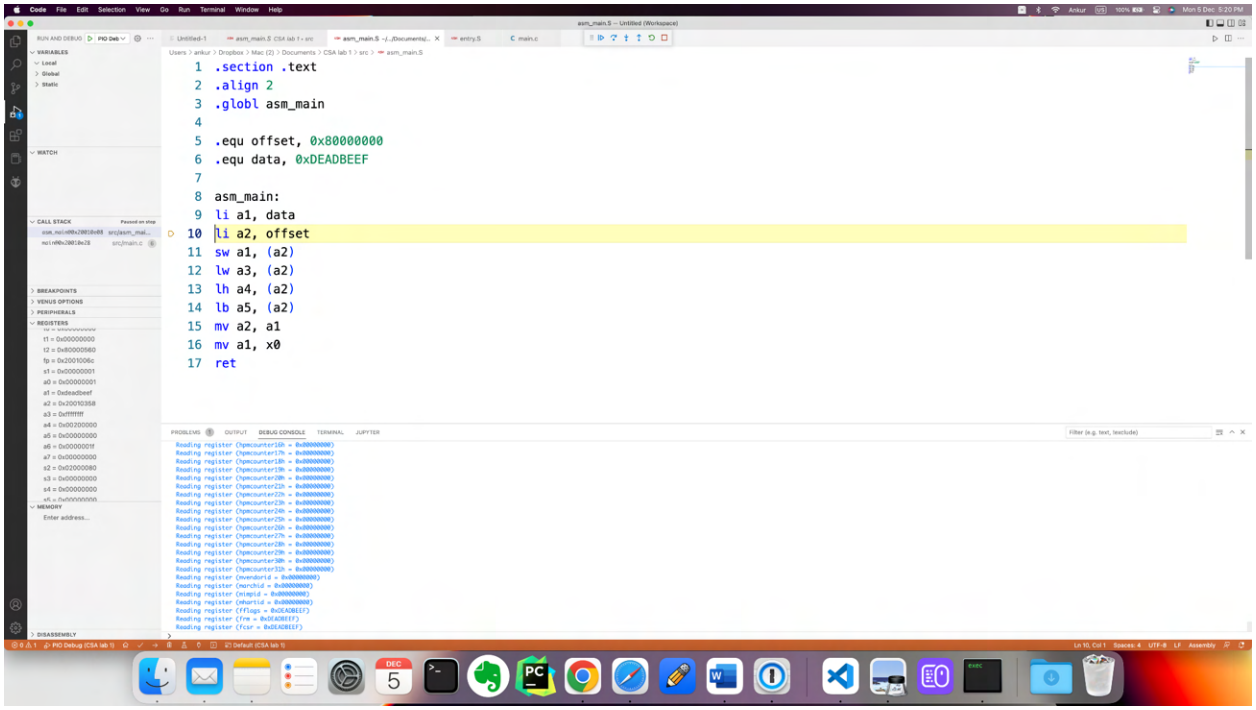
Name: Ankur Aggarwal
Net ID: aa10336

Name: Pratham Mehta
Net ID: pm3483

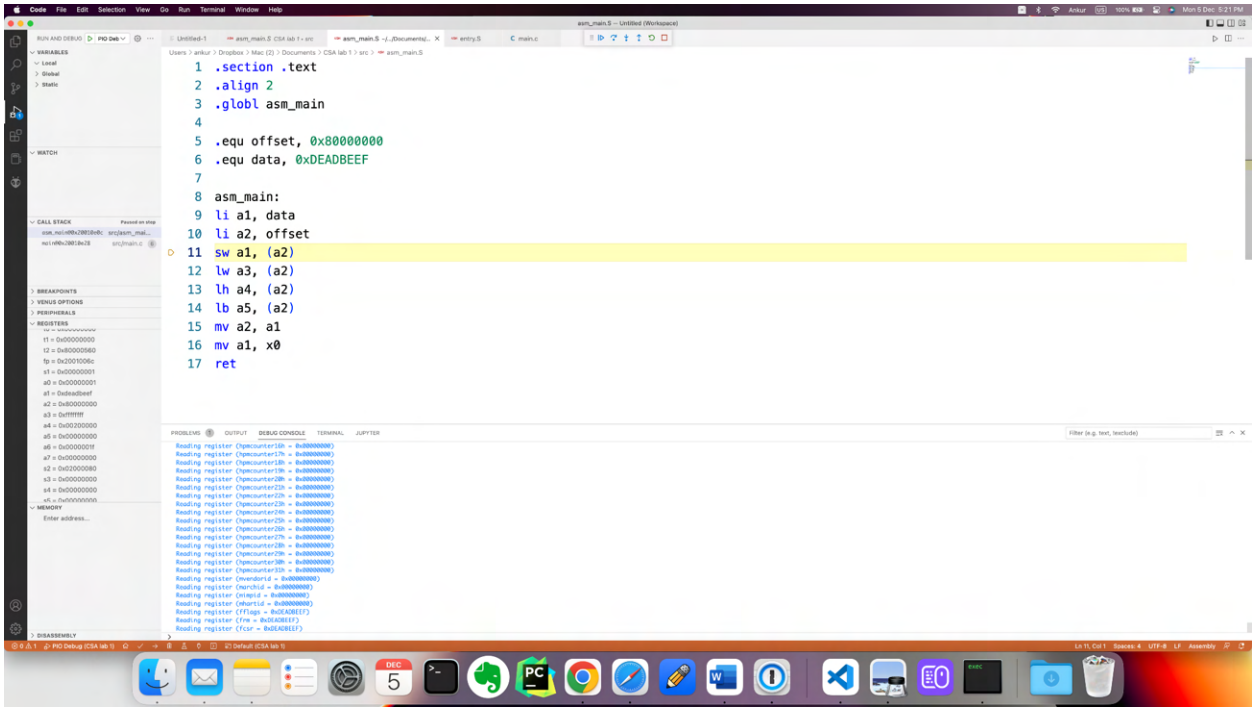
A file CSA project B.zip has been uploaded along with this PDF file

Lab 1
Part 1

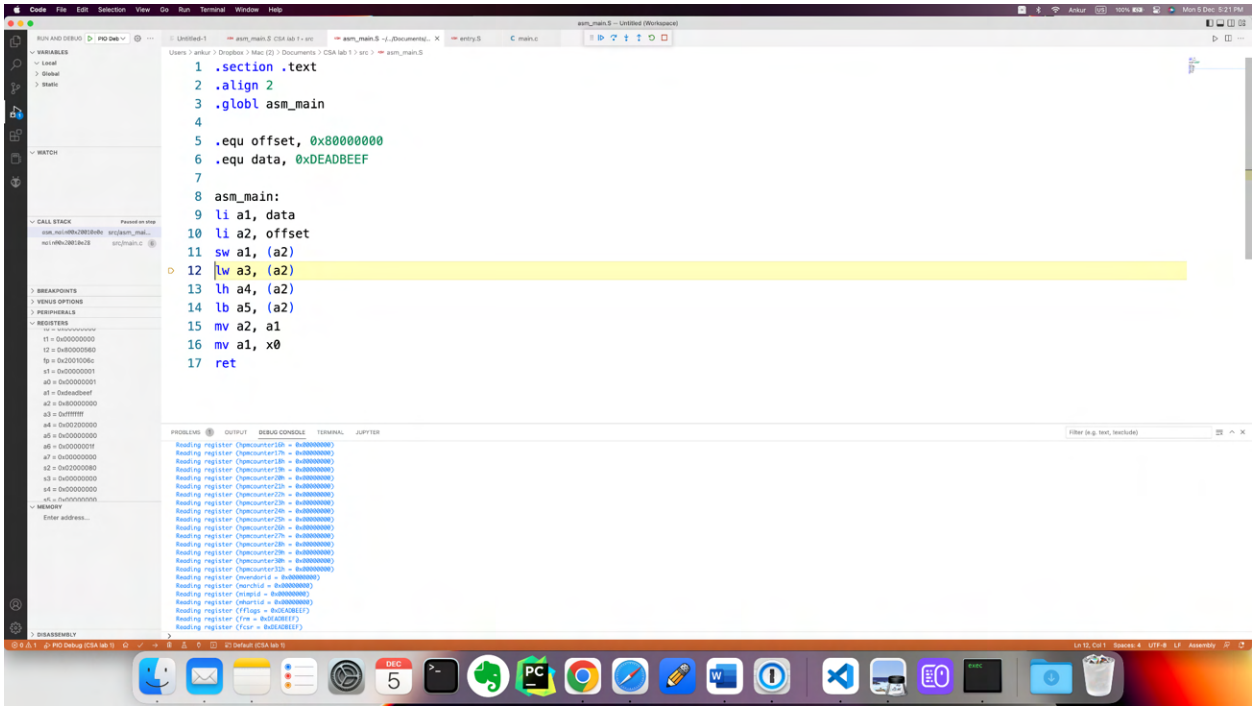


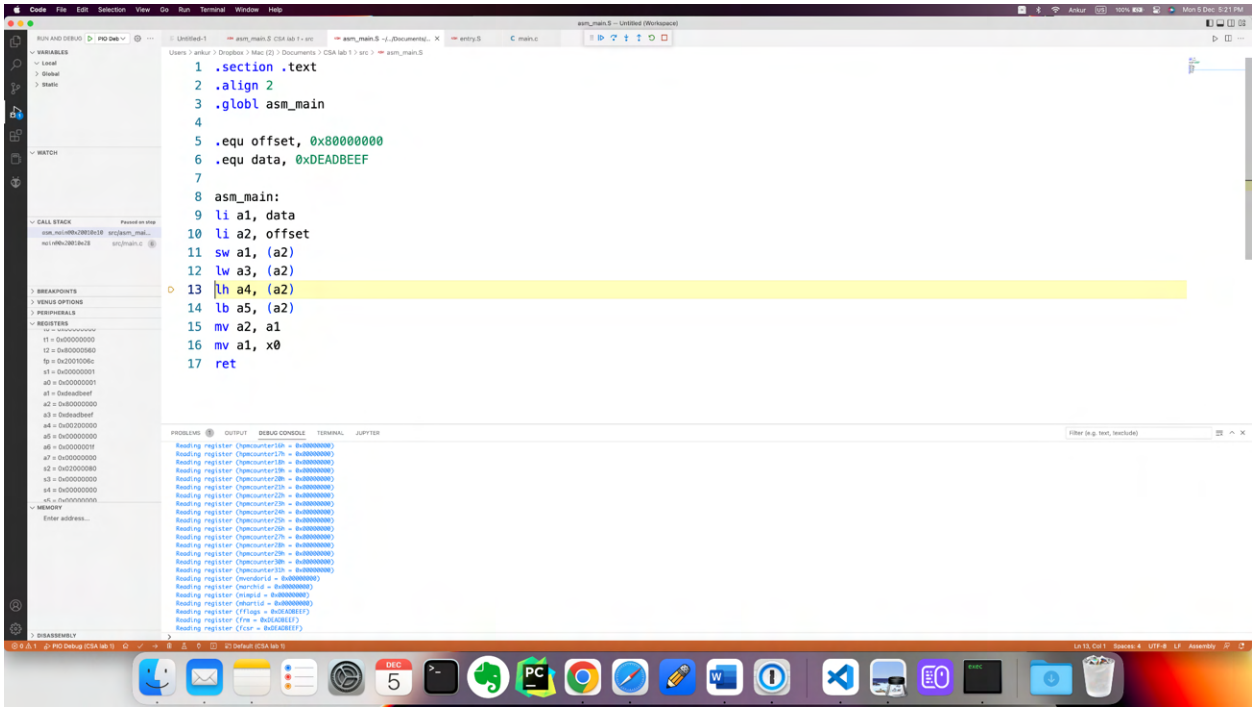


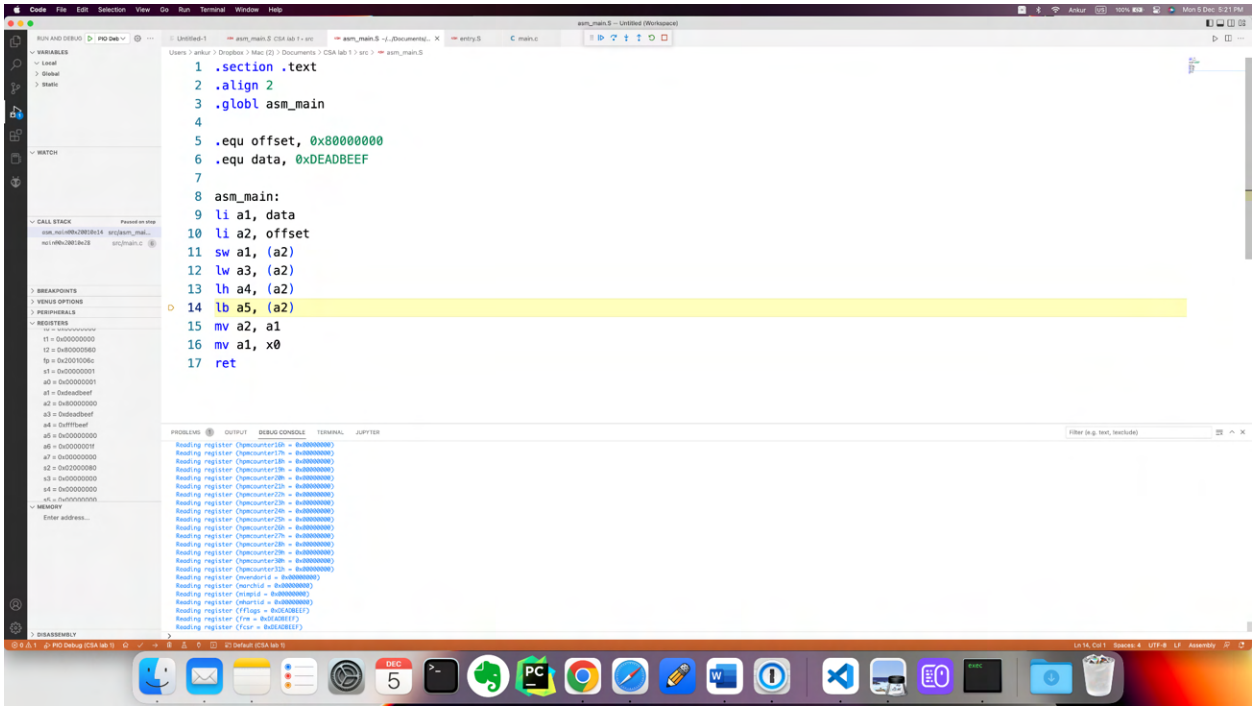
```
.section .text
.align 2
.globl asm_main

.equ offset, 0x80000000
.equ data, 0xDEADBEEF

asm_main:
li a1, data
li a2, offset
sw a1, (a2)
lw a3, (a2)
lh a4, (a2)
lb a5, (a2)
mv a2, a1
mv a1, x0
ret
```



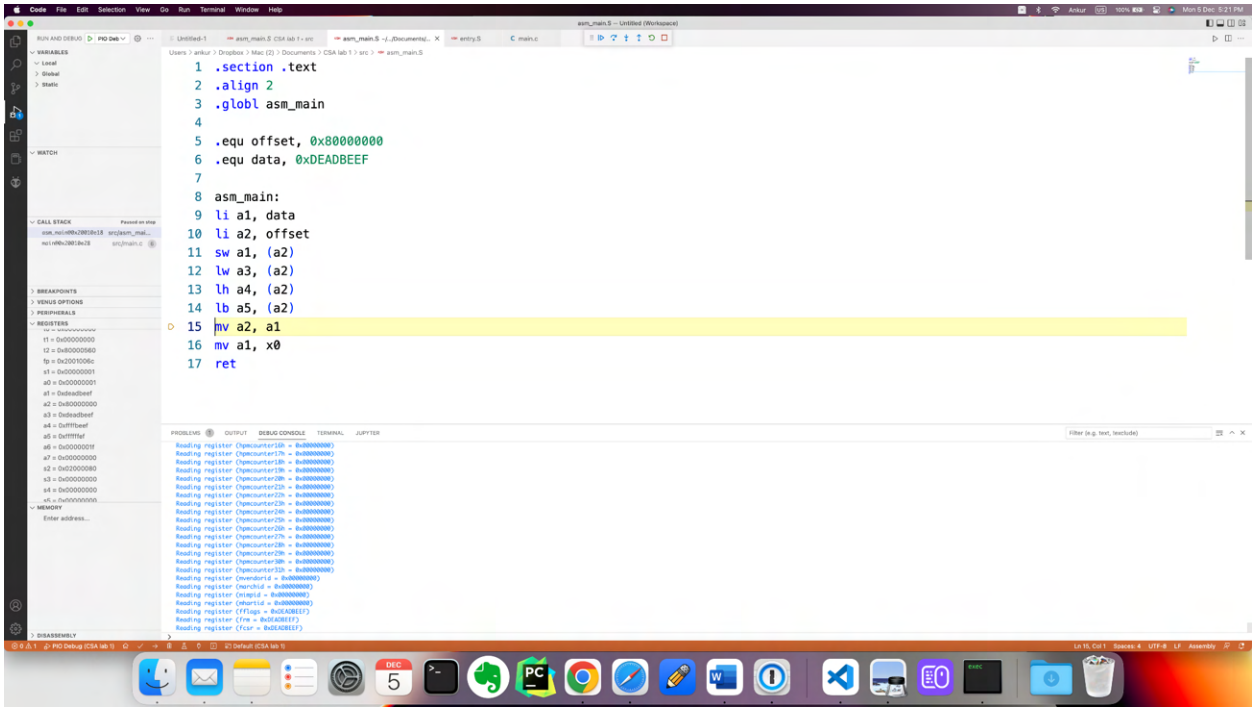
```
.section .text
.align 2
.globl asm_main

.equ offset, 0x80000000
.equ data, 0xDEADBEEF

asm_main:
li a1, data
li a2, offset
sw a1, (a2)
lw a3, (a2)
lh a4, (a2)
lb a5, (a2)
mv a2, a1
mv a1, x0
ret
```


.section .text
.align 2
.globl asm_main
.equ offset, 0x80000000
.equ data, 0xDEADBEEF
asm_main:
li a1, data
li a2, offset
sw a1, (a2)
lw a3, (a2)
lh a4, (a2)
lb a5, (a2)
mv a2, a1
mv a1, x0
ret

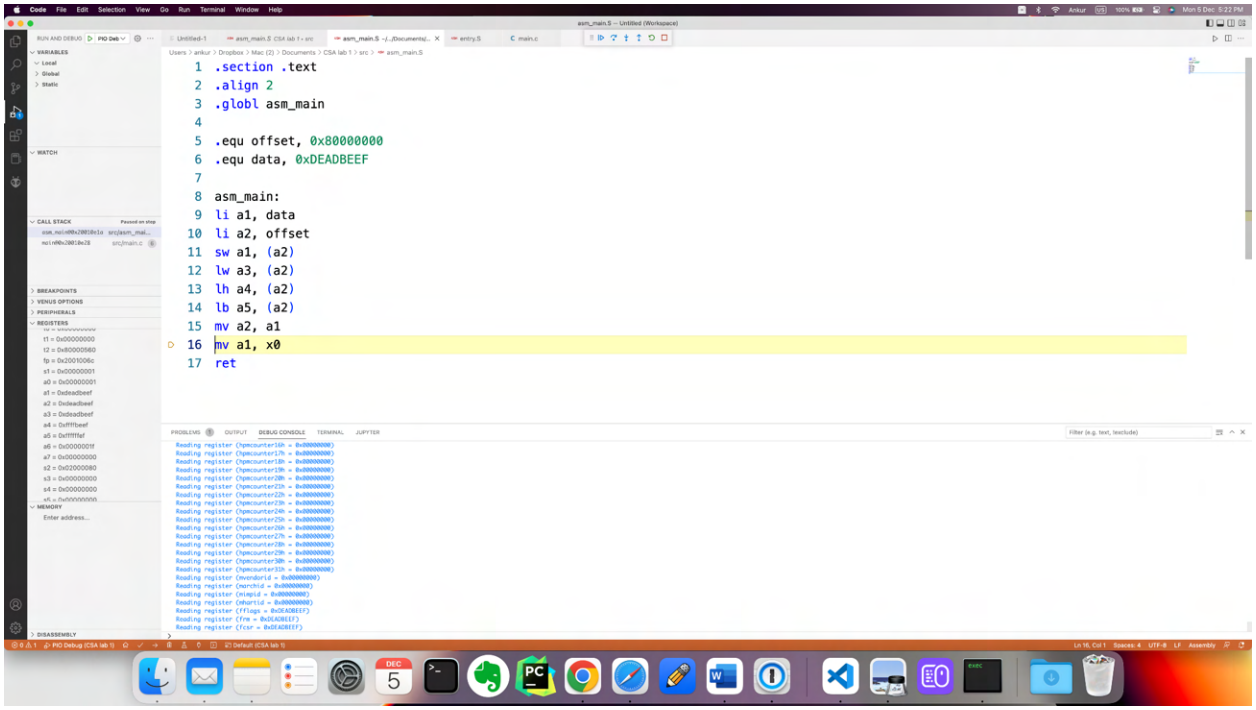

.section .text
.align 2
.globl asm_main
.equ offset, 0x80000000
.equ data, 0xDEADBEEF
asm_main:
li a1, data
li a2, offset
sw a1, (a2)
lw a3, (a2)
lh a4, (a2)
lb a5, (a2)
mv a2, a1
mv a1, x0
ret

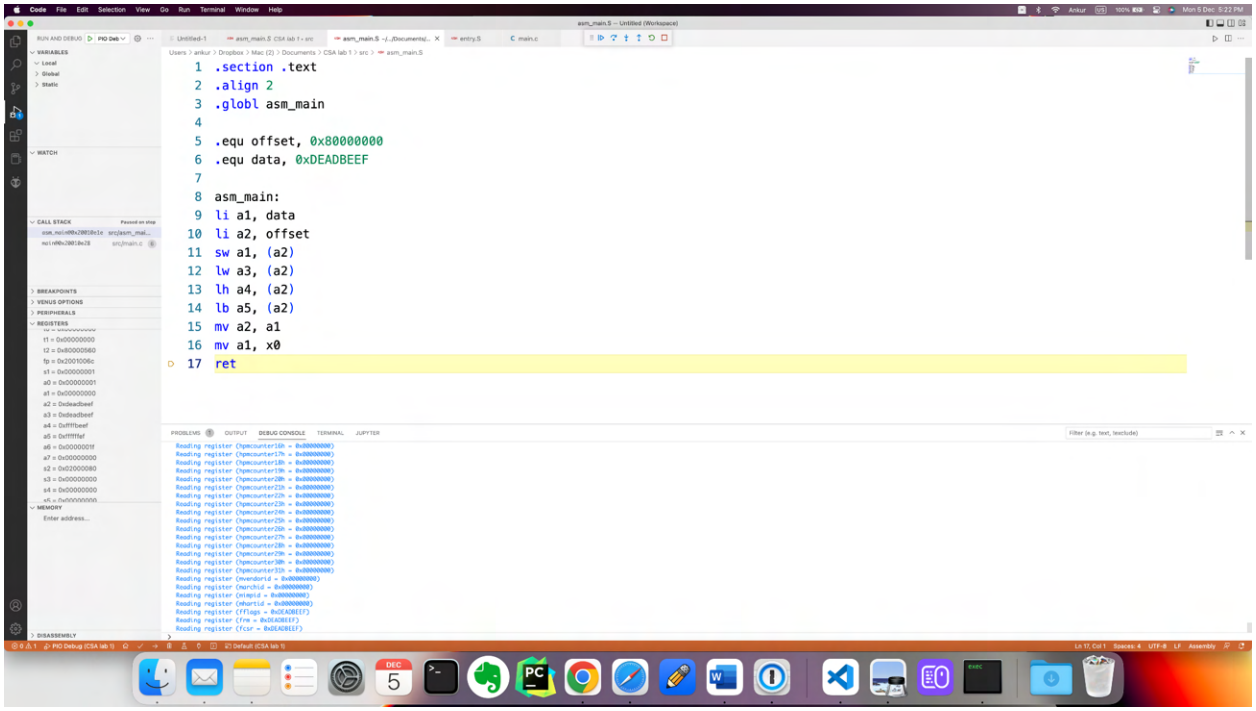


```
.section .text
.align 2
.globl asm_main

.equ offset, 0x80000000
.equ data, 0xDEADBEEF

asm_main:
li a1, data
li a2, offset
sw a1, (a2)
lw a3, (a2)
lh a4, (a2)
lb a5, (a2)
mv a2, a1
mv a1, x0
ret
```

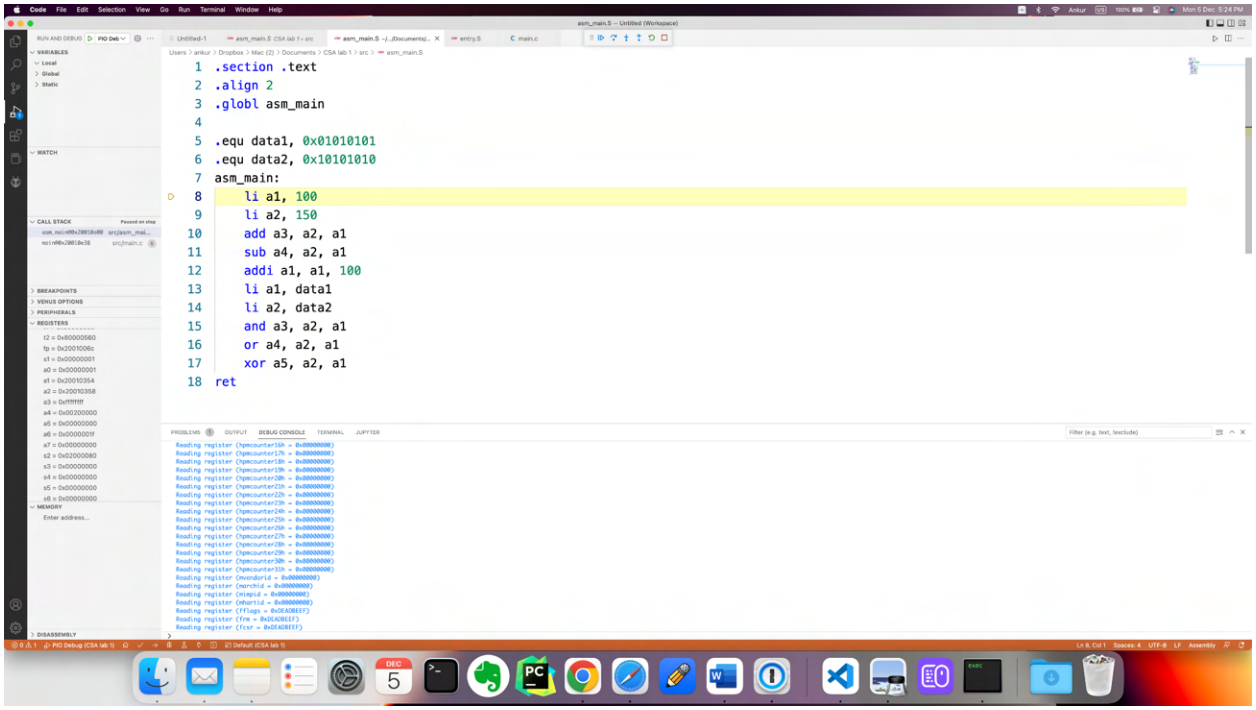


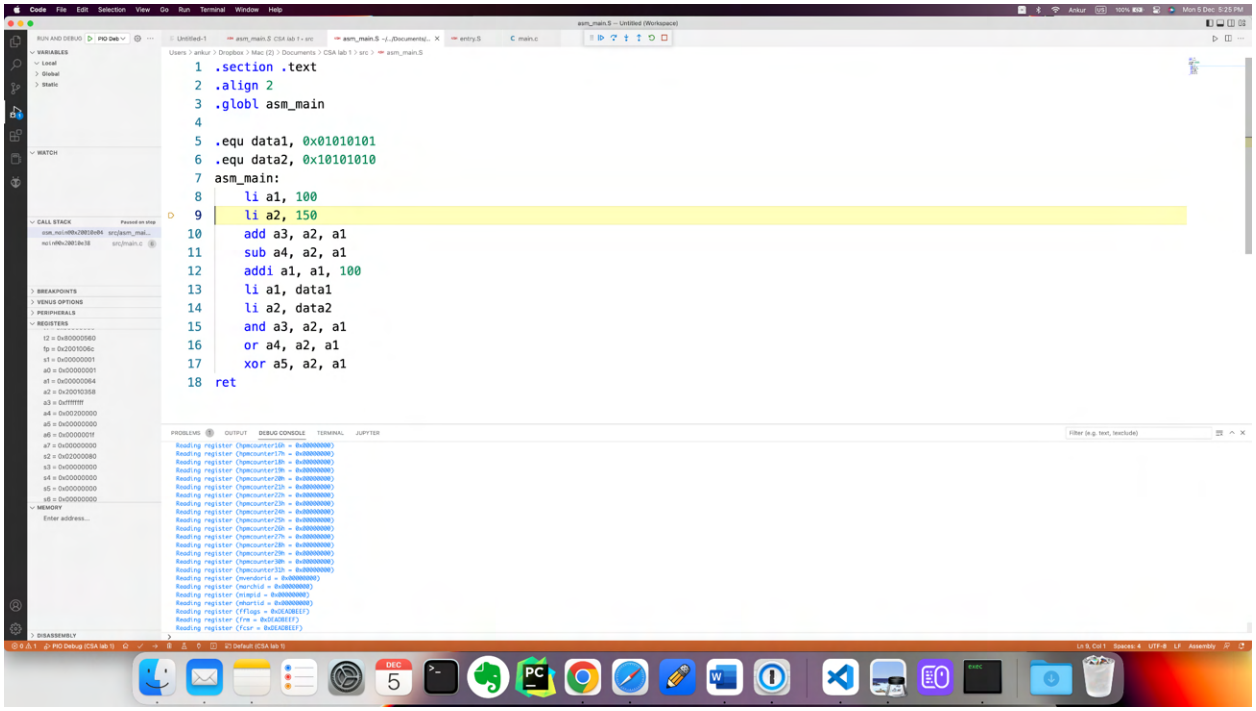
```
.section .text
.align 2
.globl asm_main

.equ offset, 0x80000000
.equ data, 0xDEADBEEF

asm_main:
li a1, data
li a2, offset
sw a1, (a2)
lw a3, (a2)
lh a4, (a2)
lb a5, (a2)
mv a2, a1
mv a1, x0
ret
```

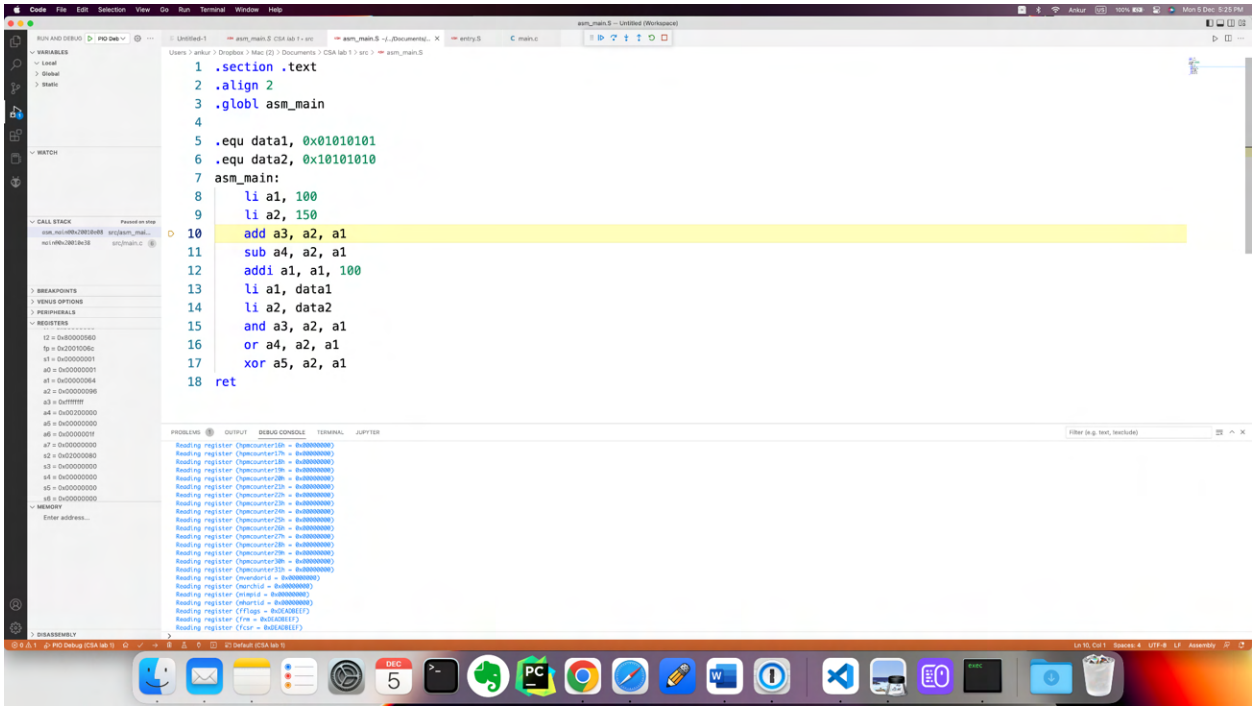

1 .section .text
2 .align 2
3 .globl asm_main
4
5 .equ offset, 0x80000000
6 .equ data, 0xDEADBEEF
7
8 asm_main:
9 li a1, data
10 li a2, offset
11 sw a1, (a2)
12 lw a3, (a2)
13 lh a4, (a2)
14 lb a5, (a2)
15 mv a2, a1
16 mv a1, x0
17 ret

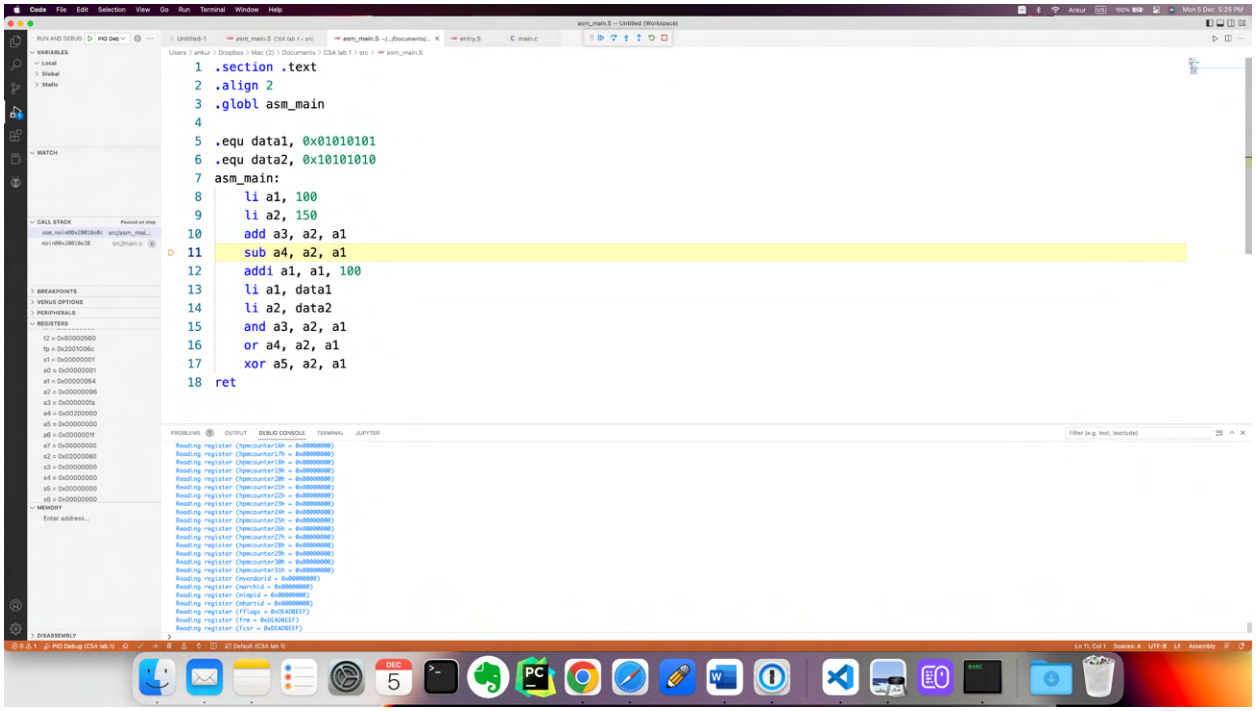

1 .section .text
2 .align 2
3 .globl asm_main
4
5 .equ offset, 0x80000000
6 .equ data, 0xDEADBEEF
7
8 asm_main:
9 li a1, data
10 li a2, offset
11 sw a1, (a2)
12 lw a3, (a2)
13 lh a4, (a2)
14 lb a5, (a2)
15 mv a2, a1
16 mv a1, x0
17 ret

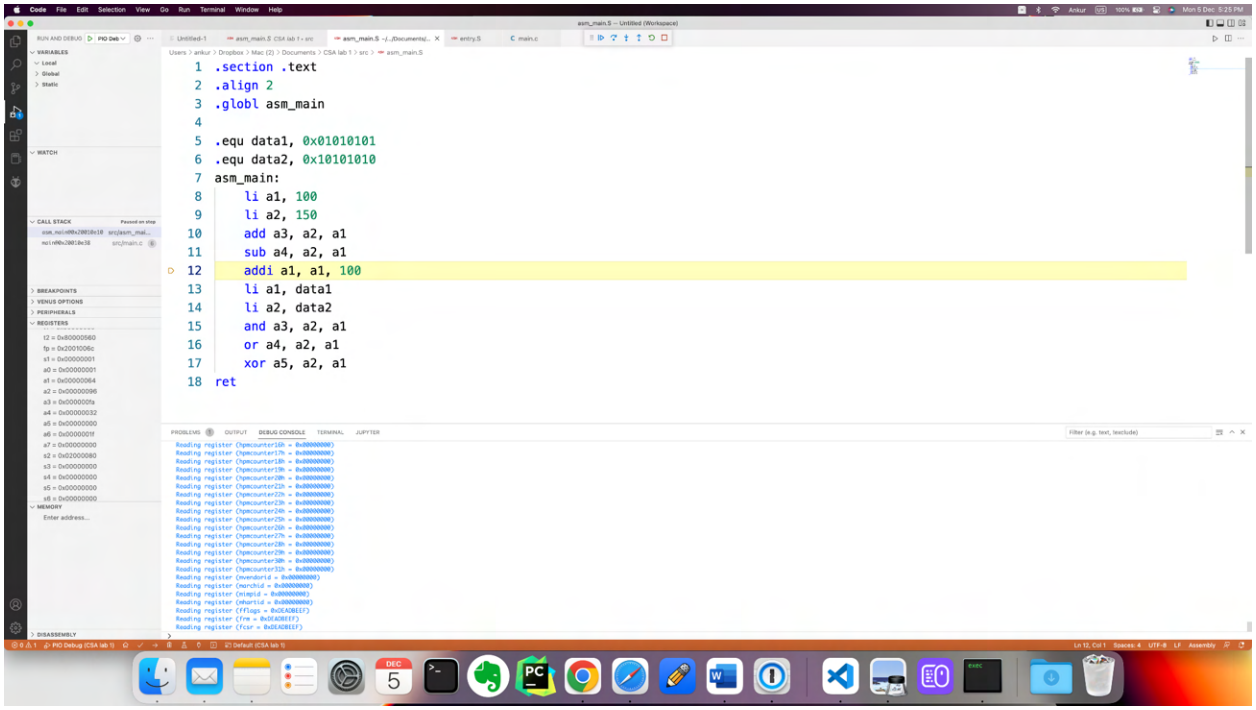
Part 2


.section .text
.align 2
.globl asm_main
.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
li a1, 100
li a2, 150
add a3, a2, a1
sub a4, a2, a1
addi a1, a1, 100
li a1, data1
li a2, data2
and a3, a2, a1
or a4, a2, a1
xor a5, a2, a1
ret

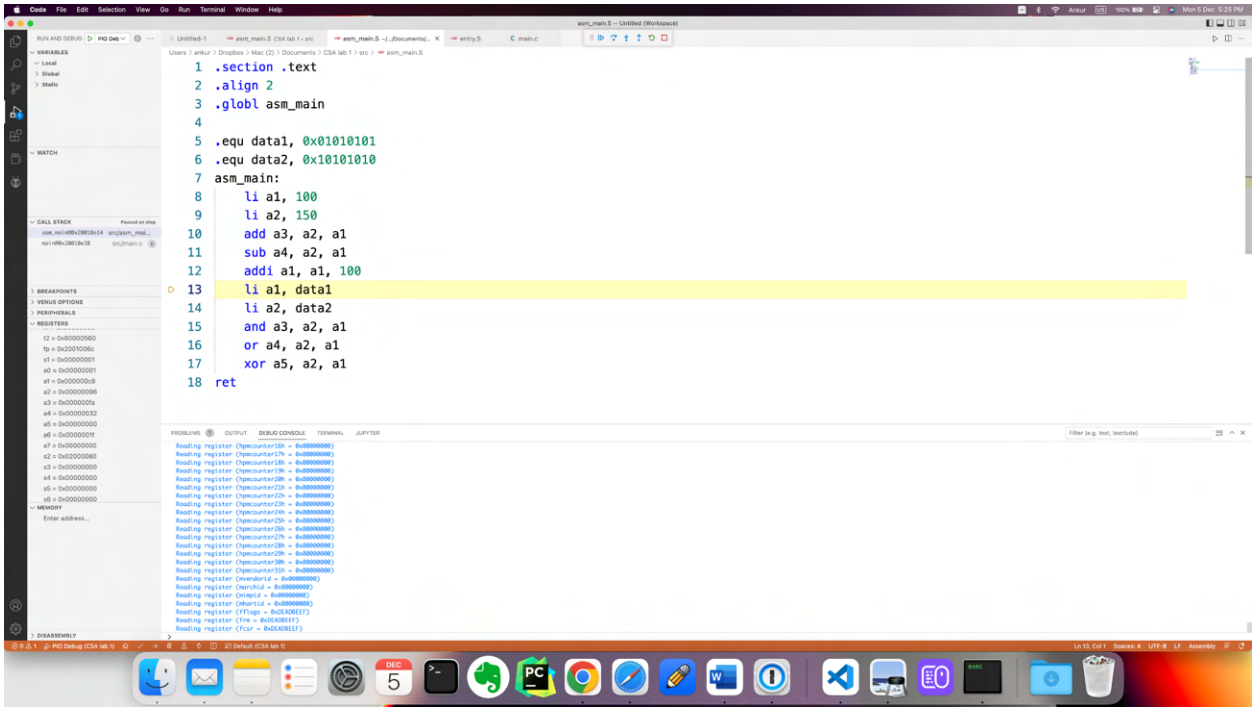

.section .text
.align 2
.globl asm_main
.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
li a1, 100
li a2, 150
add a3, a2, a1
sub a4, a2, a1
addi a1, a1, 100
li a1, data1
li a2, data2
and a3, a2, a1
or a4, a2, a1
xor a5, a2, a1
ret


.section .text
.align 2
.globl asm_main
.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
li a1, 100
li a2, 150
add a3, a2, a1
sub a4, a2, a1
addi a1, a1, 100
li a1, data1
li a2, data2
and a3, a2, a1
or a4, a2, a1
xor a5, a2, a1
ret

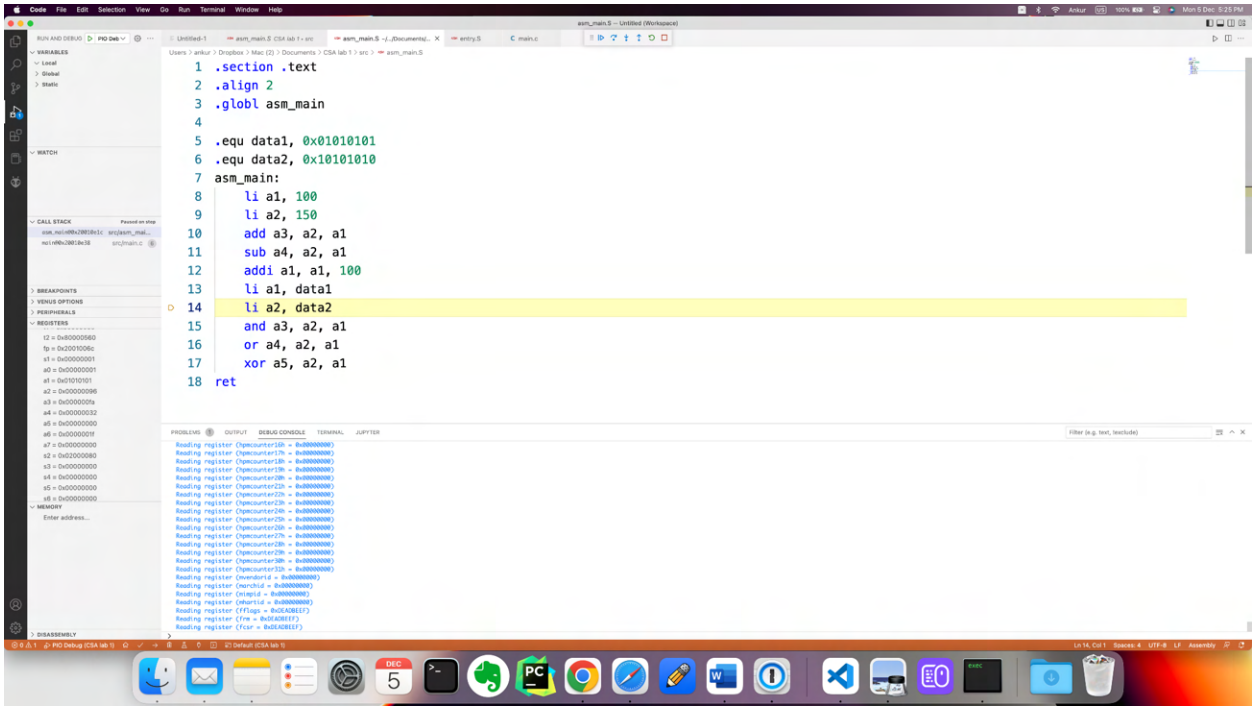

.section .text
.align 2
.globl asm_main
.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
li a1, 100
li a2, 150
add a3, a2, a1
sub a4, a2, a1
addi a1, a1, 100
li a1, data1
li a2, data2
and a3, a2, a1
or a4, a2, a1
xor a5, a2, a1
ret

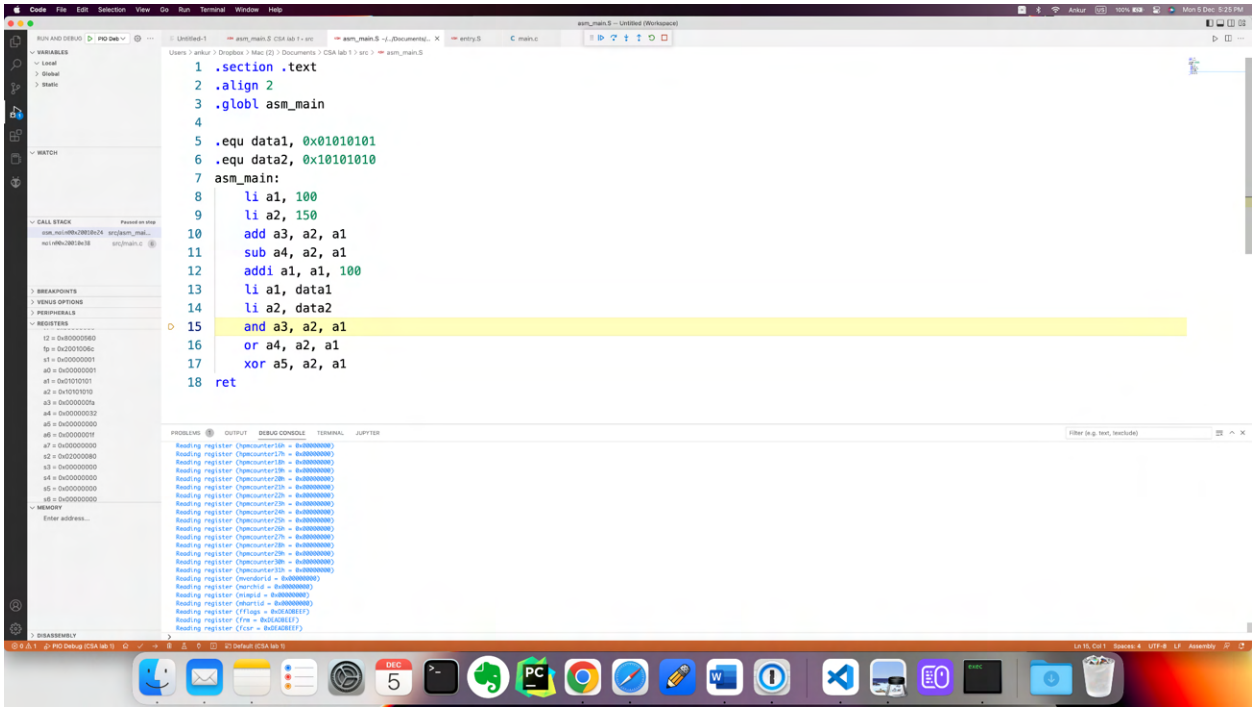


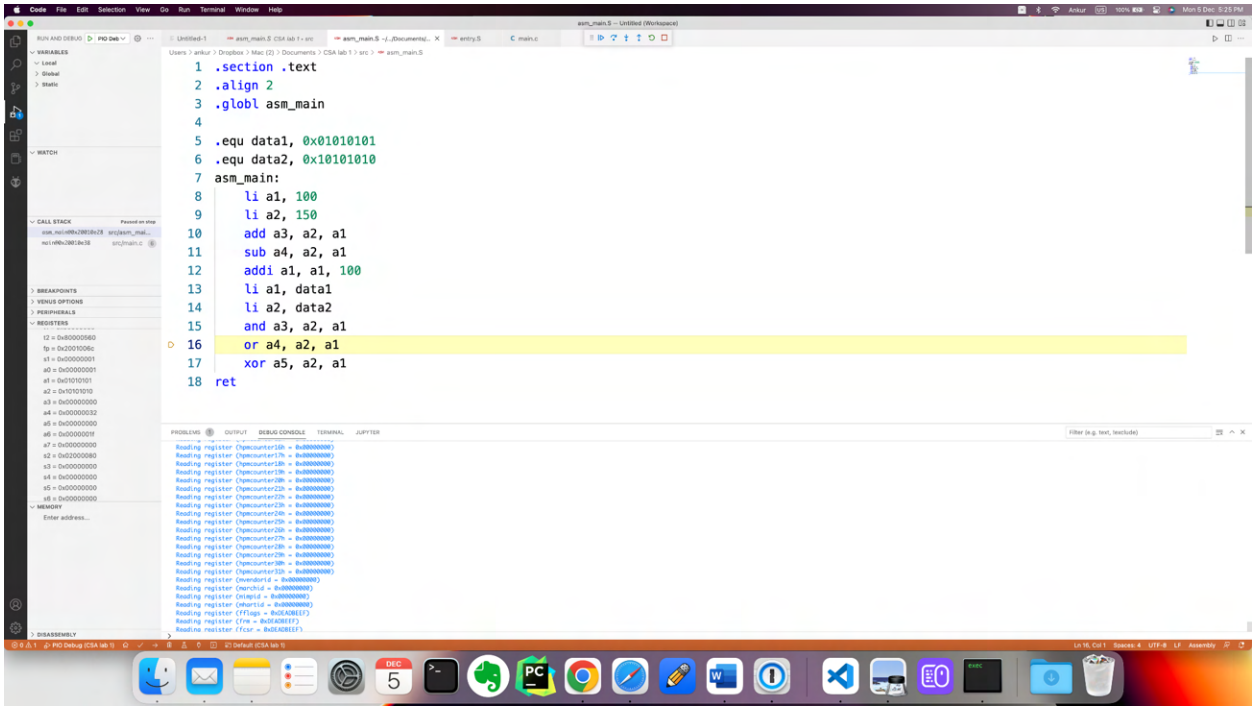
```
.section .text
.align 2
.globl asm_main

.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
    li a1, 100
    li a2, 150
    add a3, a2, a1
    sub a4, a2, a1
    addi a1, a1, 100
    li a1, data1
    li a2, data2
    and a3, a2, a1
    or a4, a2, a1
    xor a5, a2, a1
ret
```



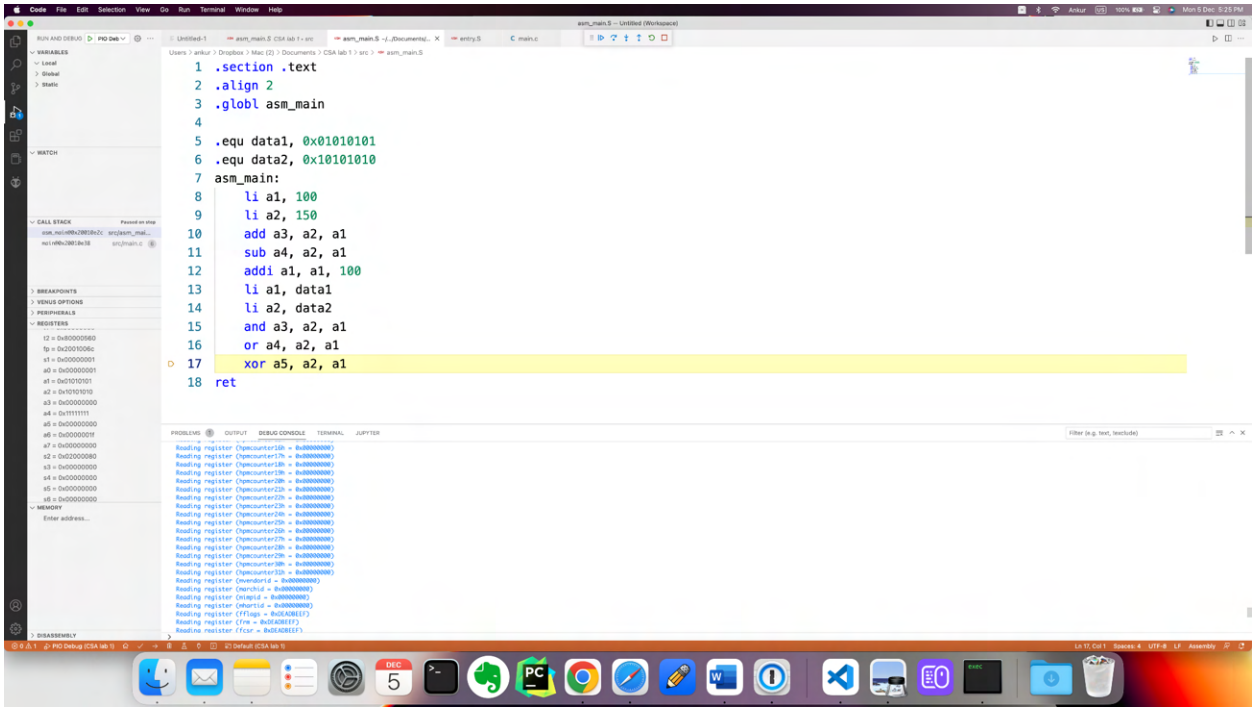
```
.section .text
.align 2
.globl asm_main

.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
    li a1, 100
    li a2, 150
    add a3, a2, a1
    sub a4, a2, a1
    addi a1, a1, 100
    li a1, data1
    li a2, data2
    and a3, a2, a1
    or a4, a2, a1
    xor a5, a2, a1
ret
```


.section .text
.align 2
.globl asm_main
.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
li a1, 100
li a2, 150
add a3, a2, a1
sub a4, a2, a1
addi a1, a1, 100
li a1, data1
li a2, data2
and a3, a2, a1
or a4, a2, a1
xor a5, a2, a1
ret

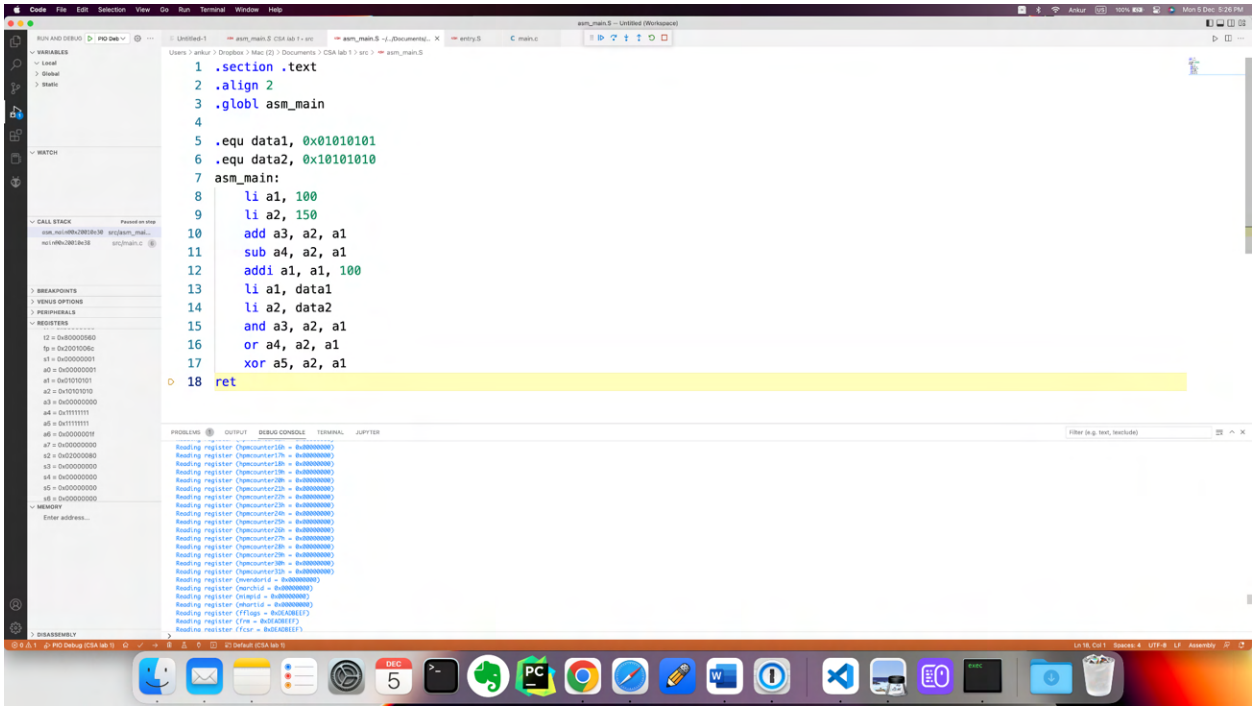

.section .text
.align 2
.globl asm_main
.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
li a1, 100
li a2, 150
add a3, a2, a1
sub a4, a2, a1
addi a1, a1, 100
li a1, data1
li a2, data2
and a3, a2, a1
or a4, a2, a1
xor a5, a2, a1
ret



```
.section .text
.align 2
.globl asm_main

.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
    li a1, 100
    li a2, 150
    add a3, a2, a1
    sub a4, a2, a1
    addi a1, a1, 100
    li a1, data1
    li a2, data2
    and a3, a2, a1
    or a4, a2, a1
    xor a5, a2, a1
ret
```

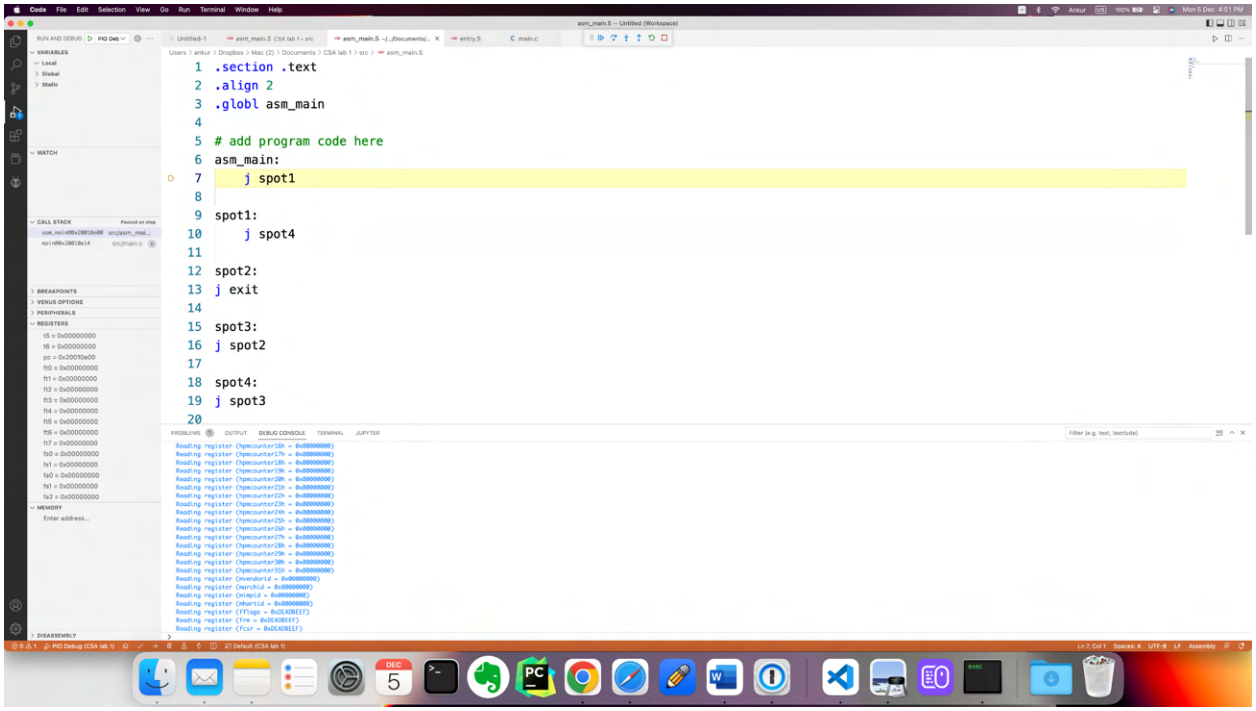


```
.section .text
.align 2
.globl asm_main

.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
    li a1, 100
    li a2, 150
    add a3, a2, a1
    sub a4, a2, a1
    addi a1, a1, 100
    li a1, data1
    li a2, data2
    and a3, a2, a1
    or a4, a2, a1
    xor a5, a2, a1
ret
```

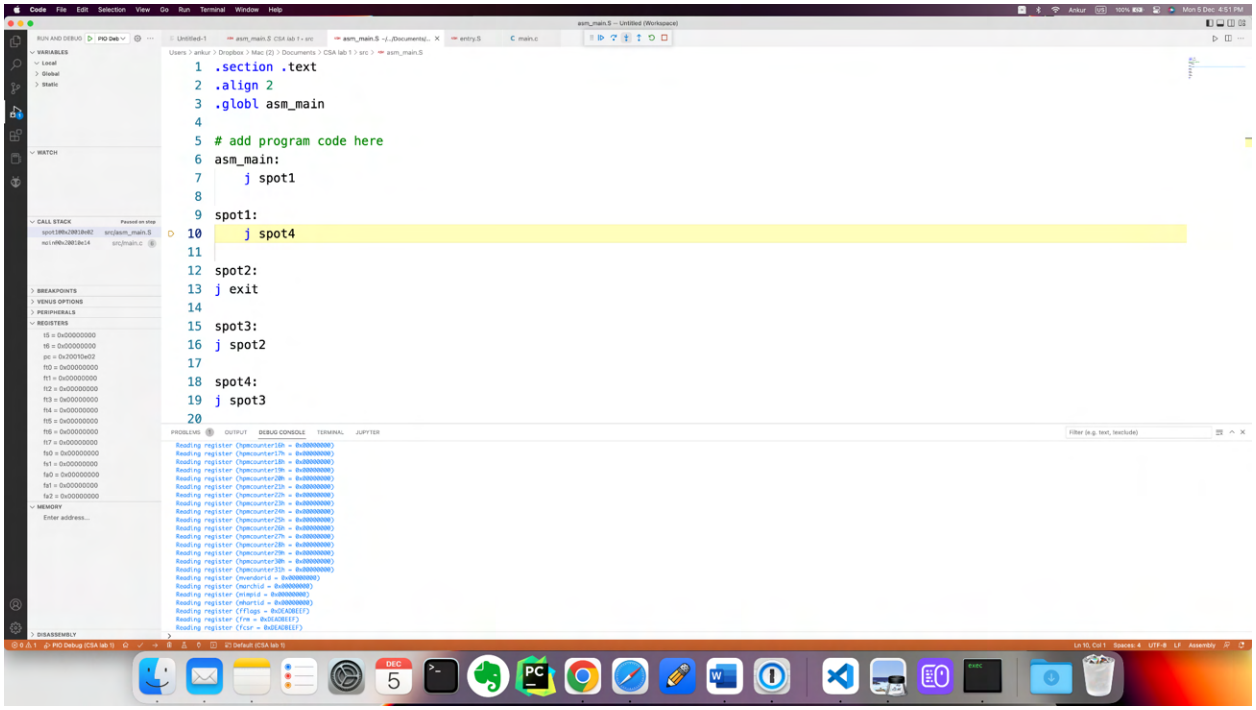


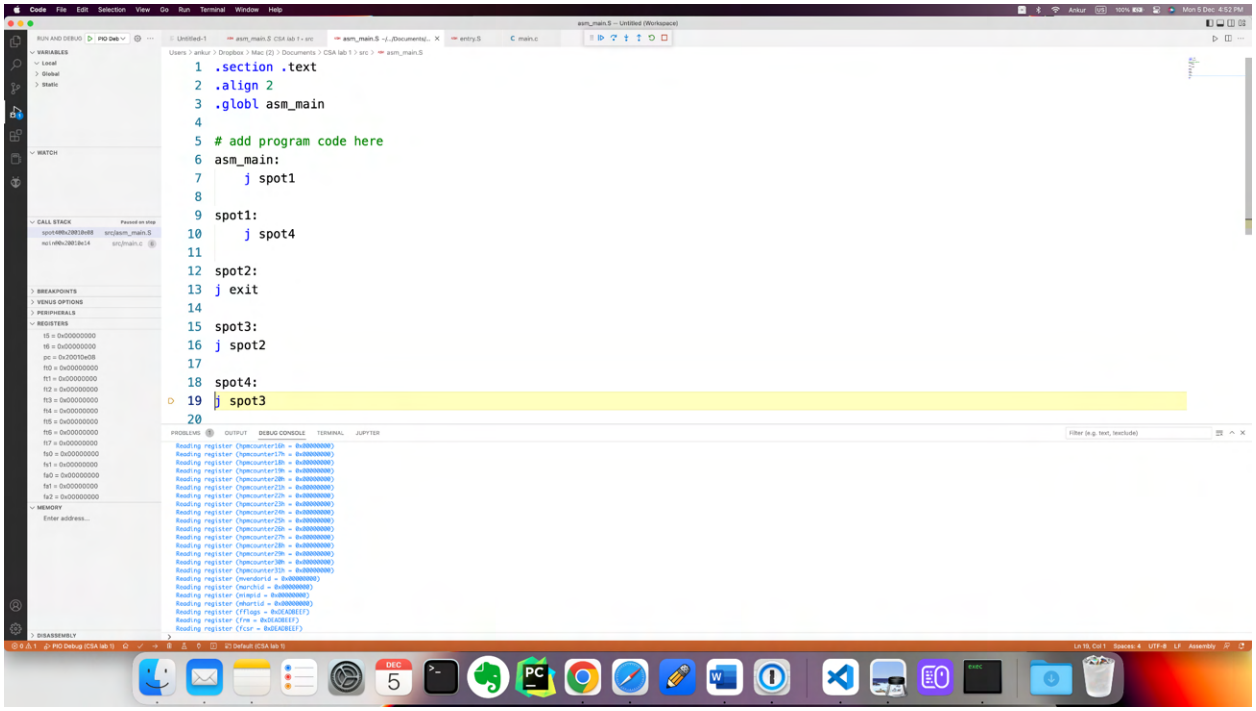
```
.section .text
.align 2
.globl asm_main

.equ data1, 0x01010101
.equ data2, 0x10101010
asm_main:
    li a1, 100
    li a2, 150
    add a3, a2, a1
    sub a4, a2, a1
    addi a1, a1, 100
    li a1, data1
    li a2, data2
    and a3, a2, a1
    or a4, a2, a1
    xor a5, a2, a1
ret
```

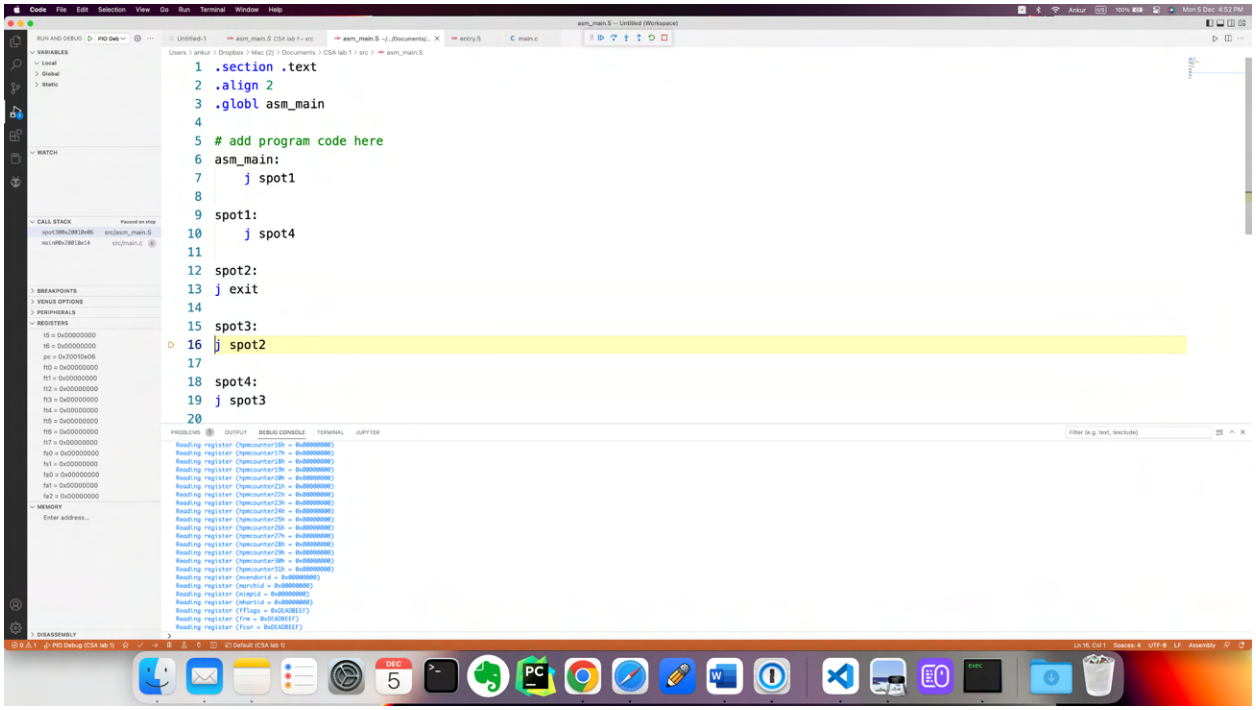
Part 3

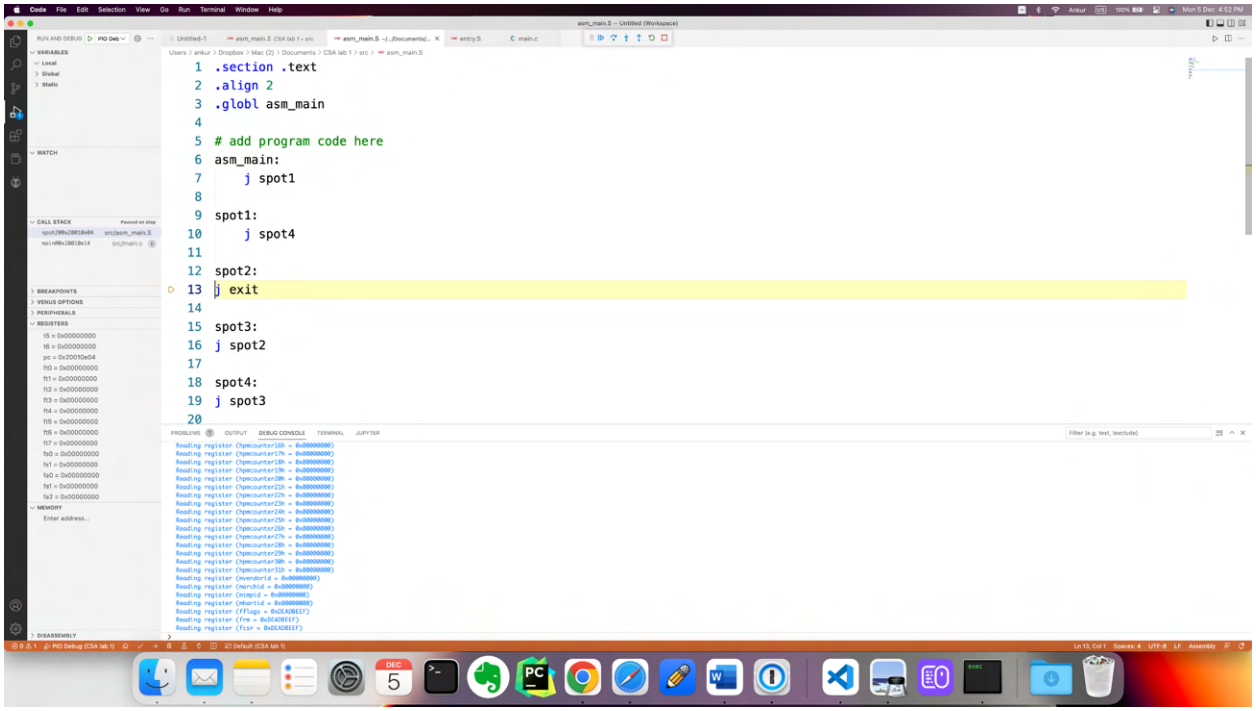


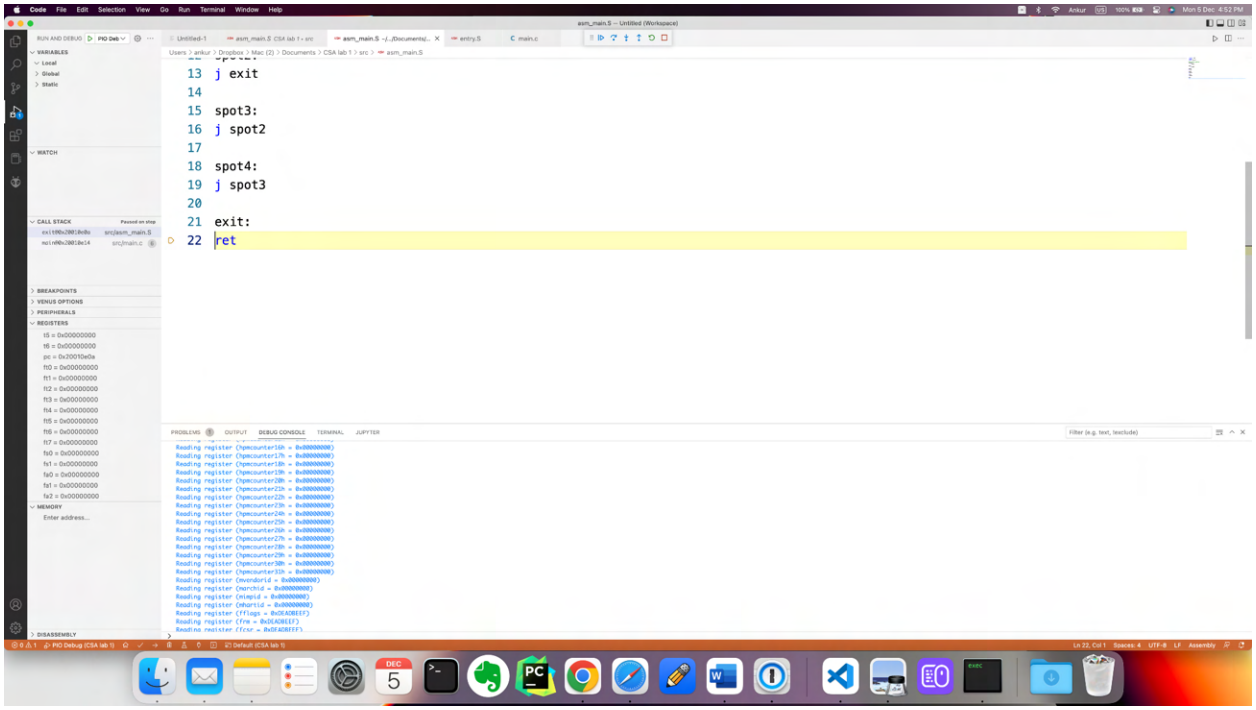
```
.section .text
.align 2
.globl asm_main

# add program code here
asm_main:
    j spot1

spot1:
    j spot4

spot2:
j exit

spot3:
j spot2

spot4:
j spot3
```


.section .text
.align 2
.globl asm_main
add program code here
asm_main:
j spot1
spot1:
j spot4
spot2:
j exit
spot3:
j spot2
spot4:
j spot3

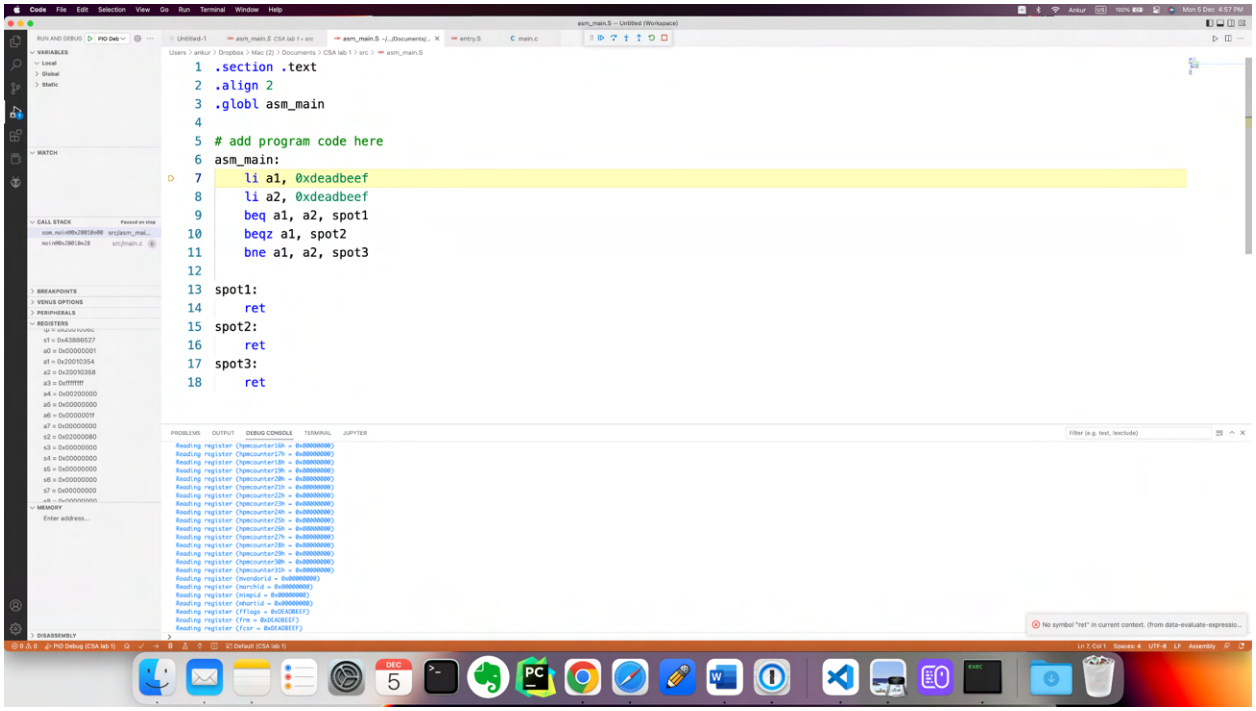

.section .text
.align 2
.globl asm_main
add program code here
asm_main:
j spot1
spot1:
j spot4
spot2:
j exit
spot3:
j spot2
spot4:
j spot3

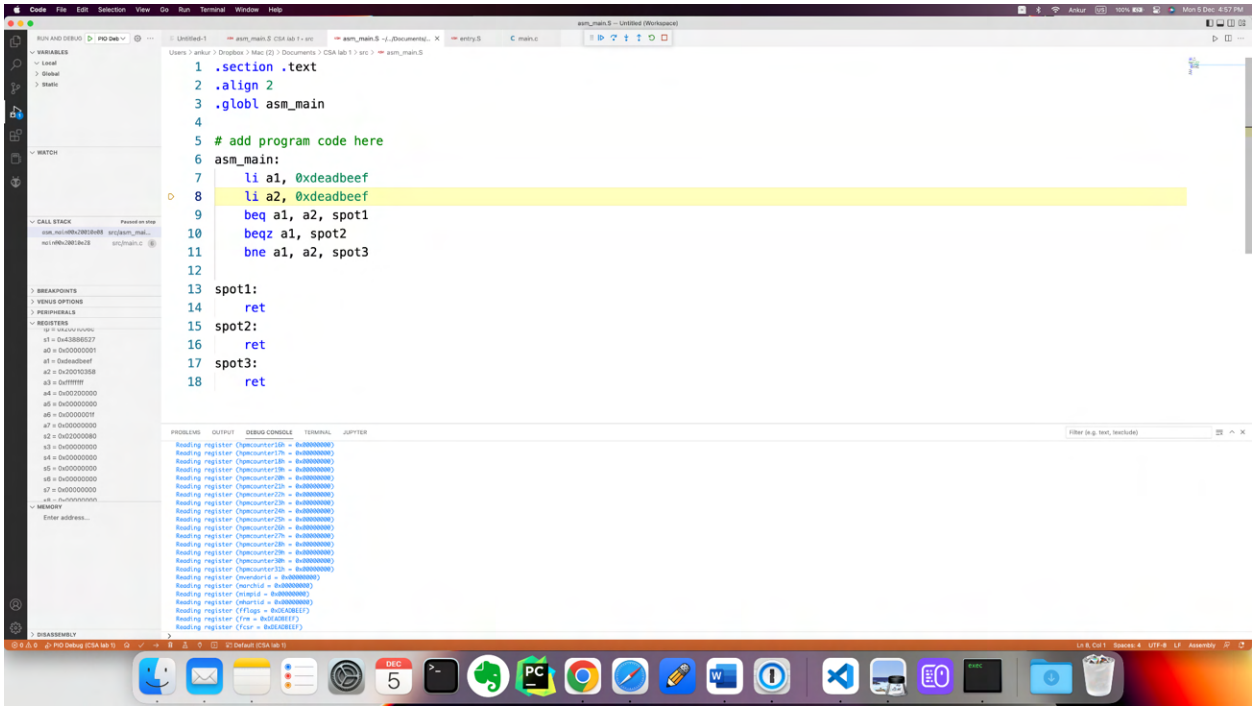

.section .text
.align 2
.globl asm_main
add program code here
asm_main:
j spot1
spot1:
j spot4
spot2:
j exit
spot3:
j spot2
spot4:
j spot3

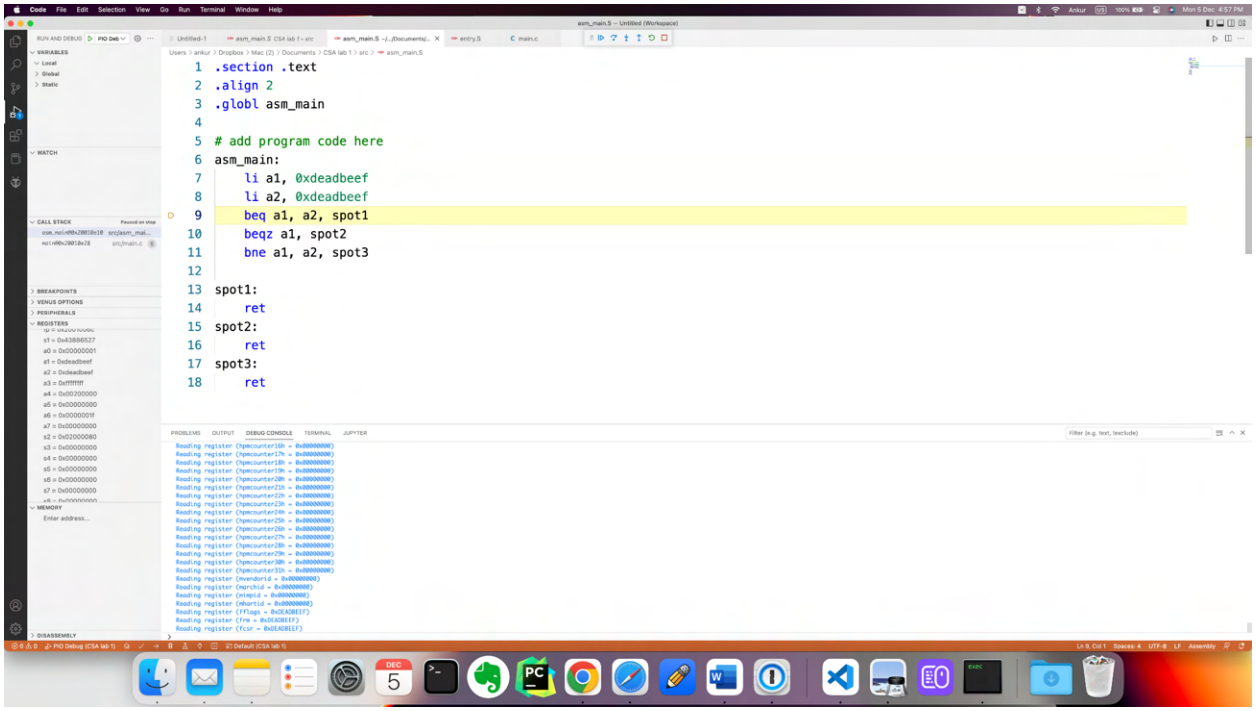

.section .text
.align 2
.globl asm_main
add program code here
asm_main:
j spot1
spot1:
j spot4
spot2:
j exit
spot3:
j spot2
spot4:
j spot3

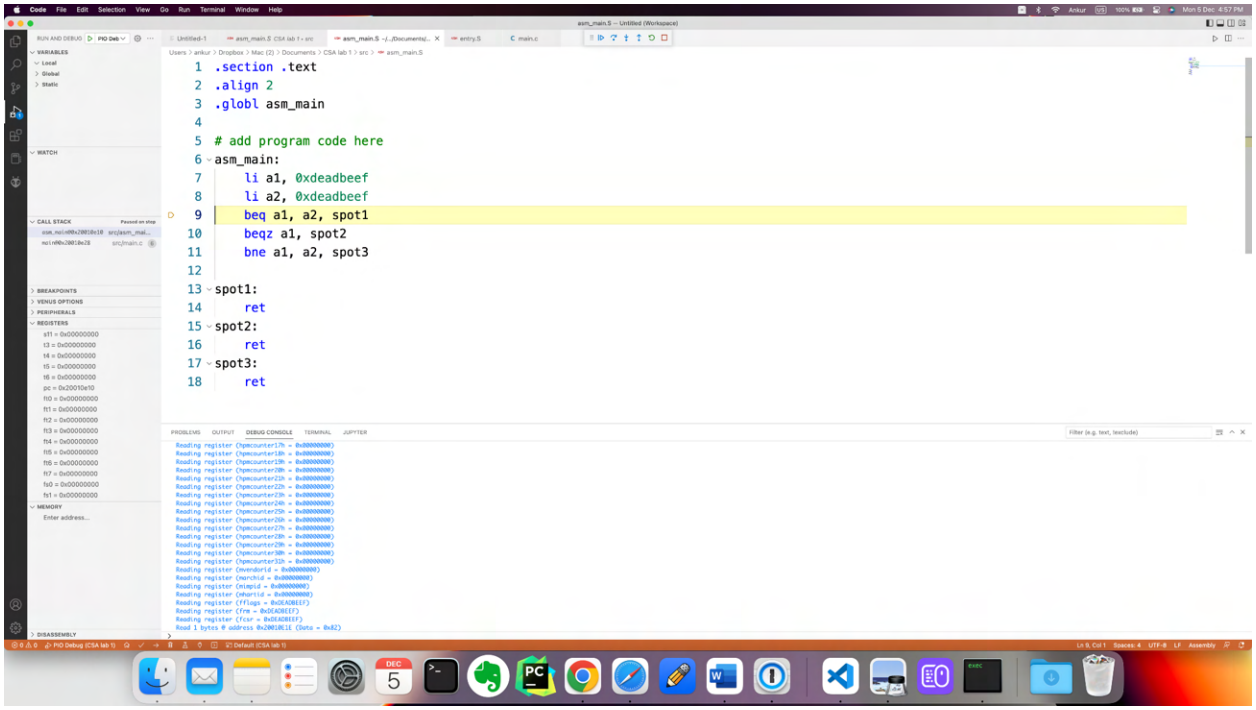

j exit
spot3:
j spot2
spot4:
j spot3
exit:
ret

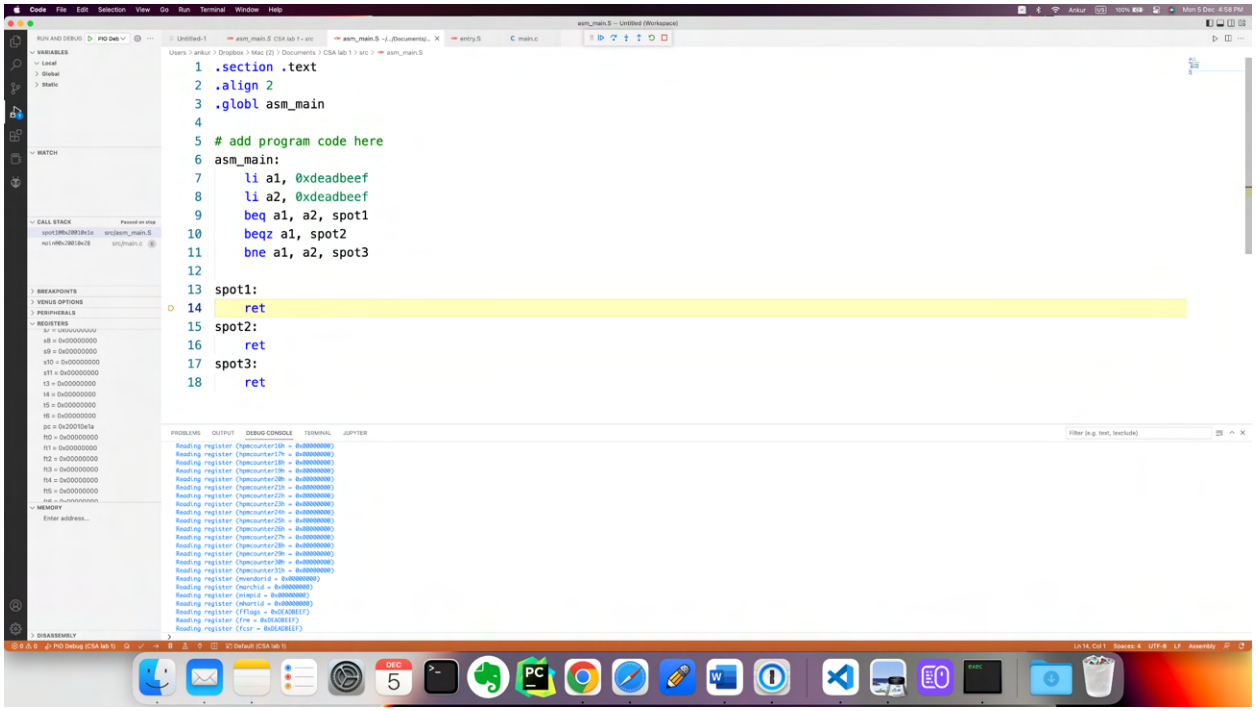
Part 4


.section .text
.align 2
.globl asm_main
add program code here
asm_main:
li a1, 0xdeadbeef
li a2, 0xdeadbeef
beq a1, a2, spot1
beqz a1, spot2
bne a1, a2, spot3
spot1:
ret
spot2:
ret
spot3:
ret

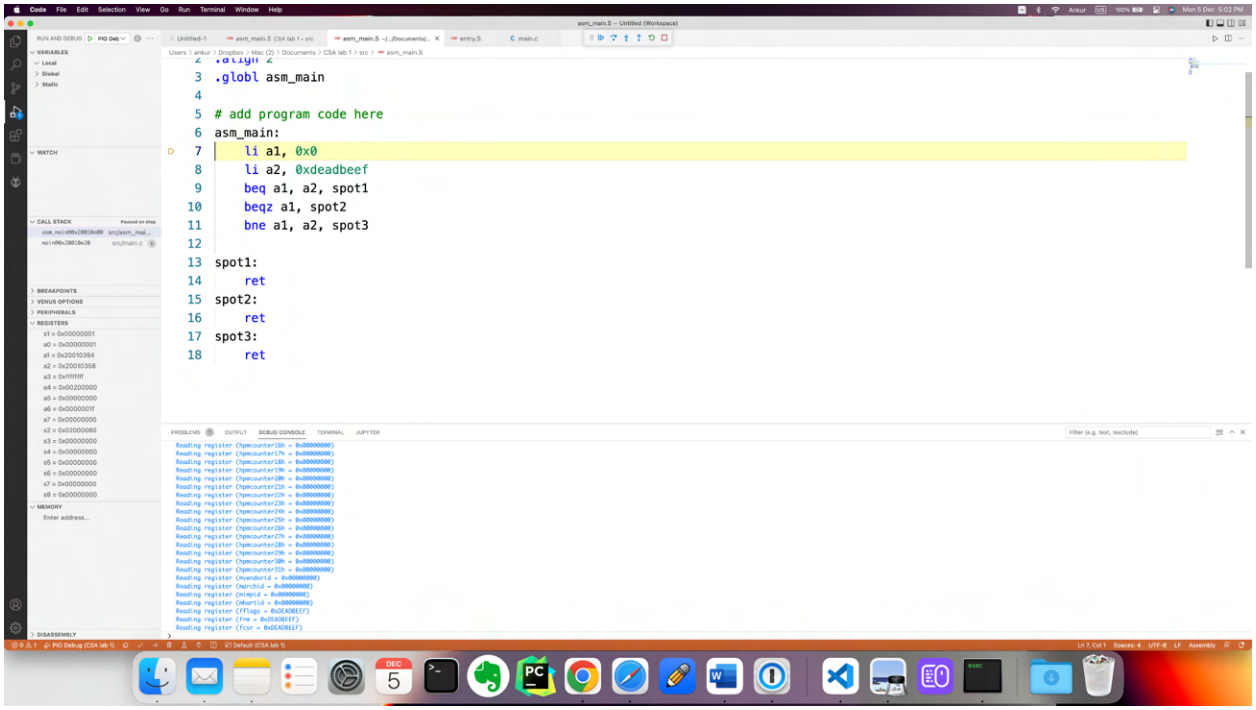

.section .text
.align 2
.globl asm_main
add program code here
asm_main:
li a1, 0xdeadbeef
li a2, 0xdeadbeef
beq a1, a2, spot1
beqz a1, spot2
bne a1, a2, spot3
spot1:
ret
spot2:
ret
spot3:
ret

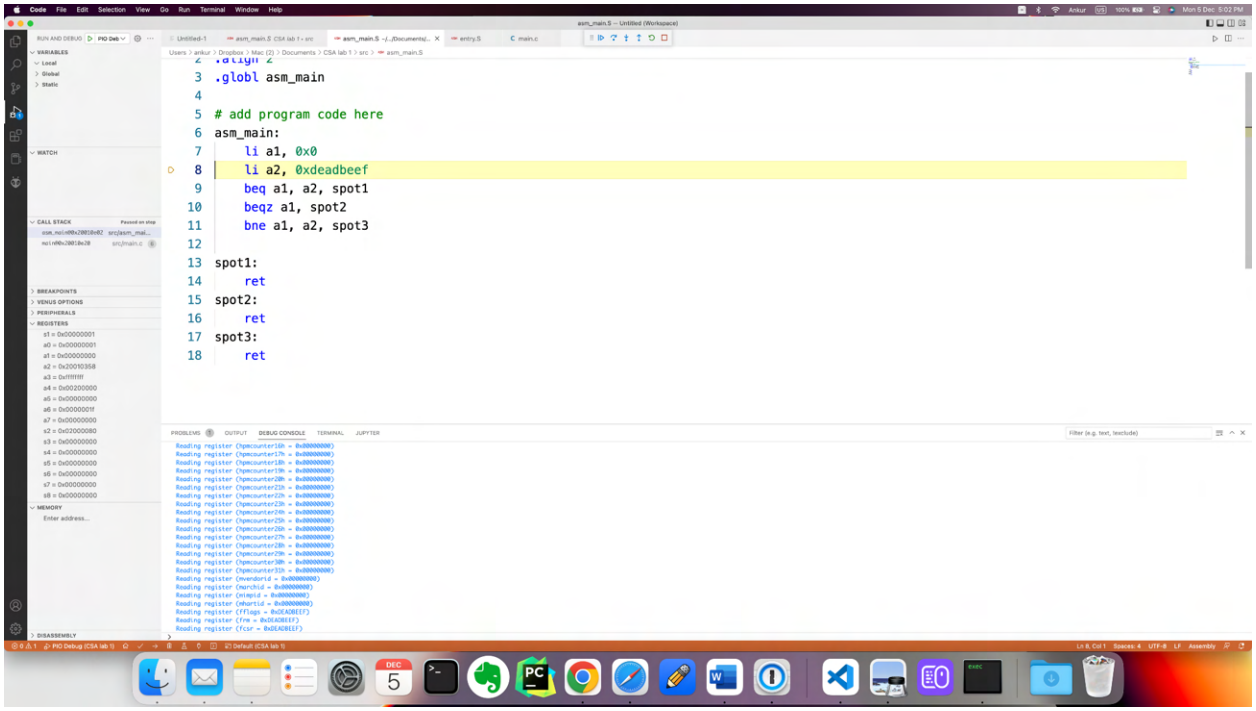

.section .text
.align 2
.globl asm_main
add program code here
asm_main:
li a1, 0xdeadbeef
li a2, 0xdeadbeef
beq a1, a2, spot1
beqz a1, spot2
bne a1, a2, spot3
spot1:
ret
spot2:
ret
spot3:
ret

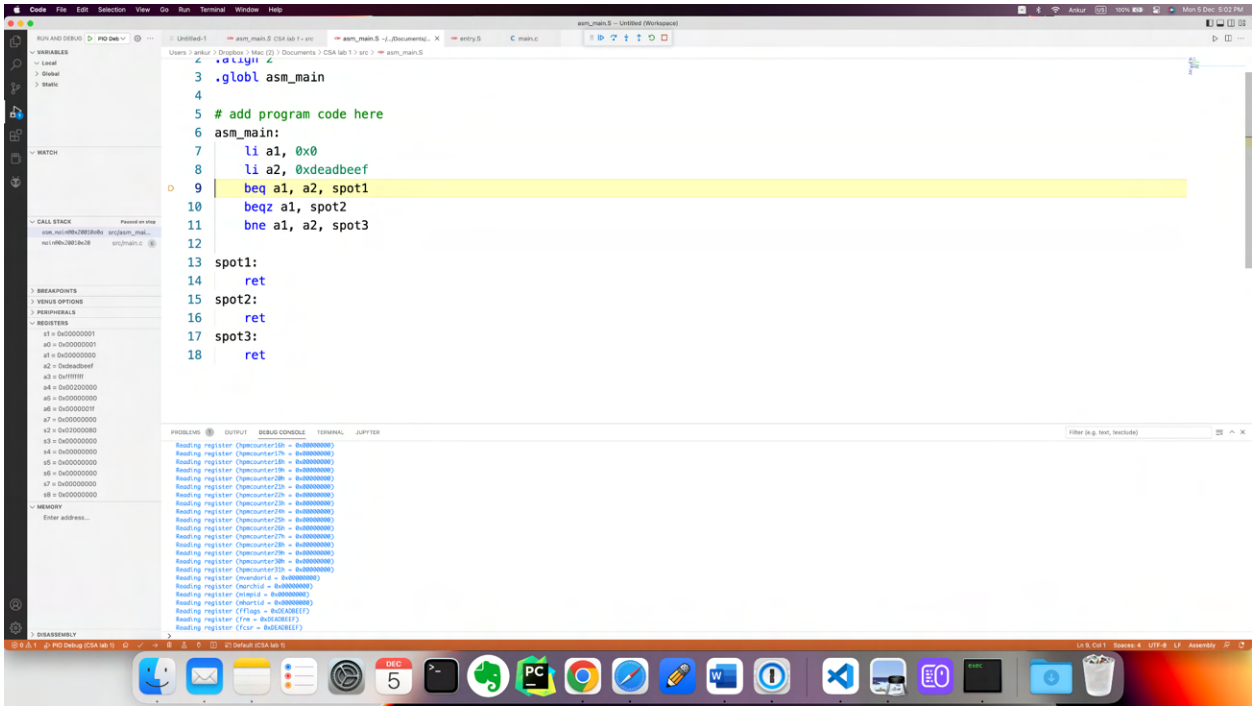




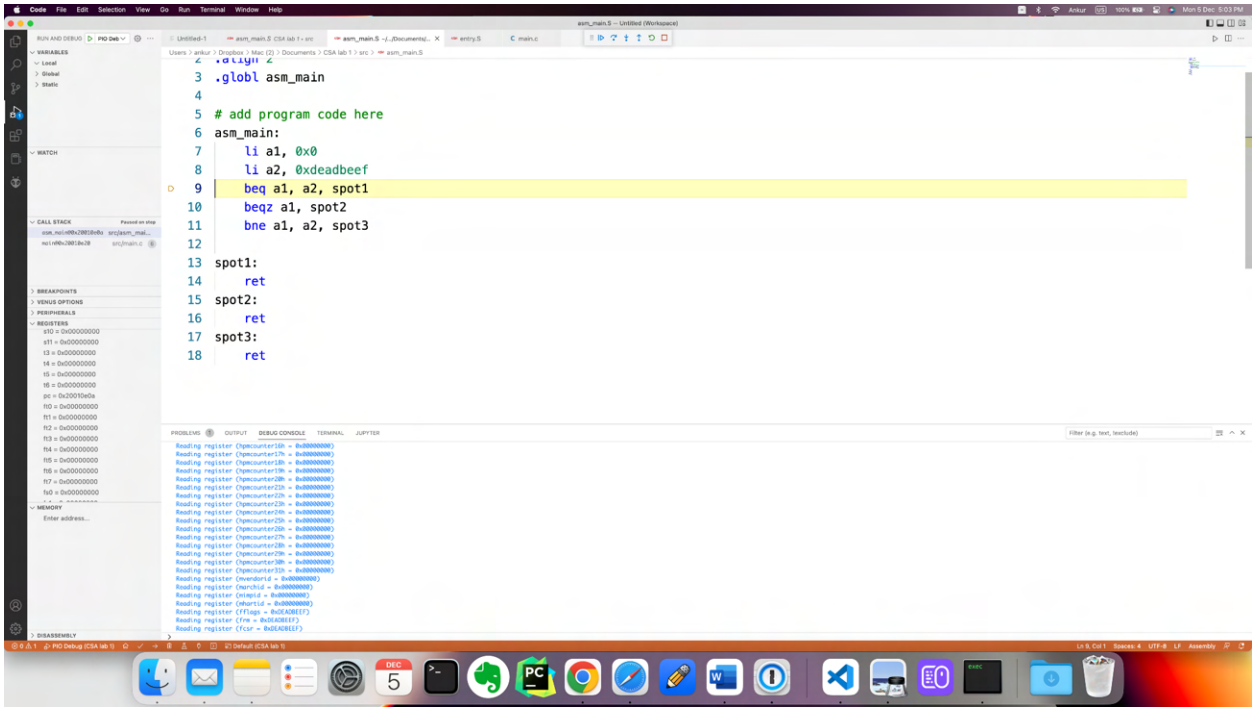
We now change the value of a1 to 0x0 as required


.globl asm_main
add program code here
asm_main:
li a1, 0x0
li a2, 0xdeadbeef
beq a1, a2, spot1
beqz a1, spot2
bne a1, a2, spot3
spot1:
ret
spot2:
ret
spot3:
ret

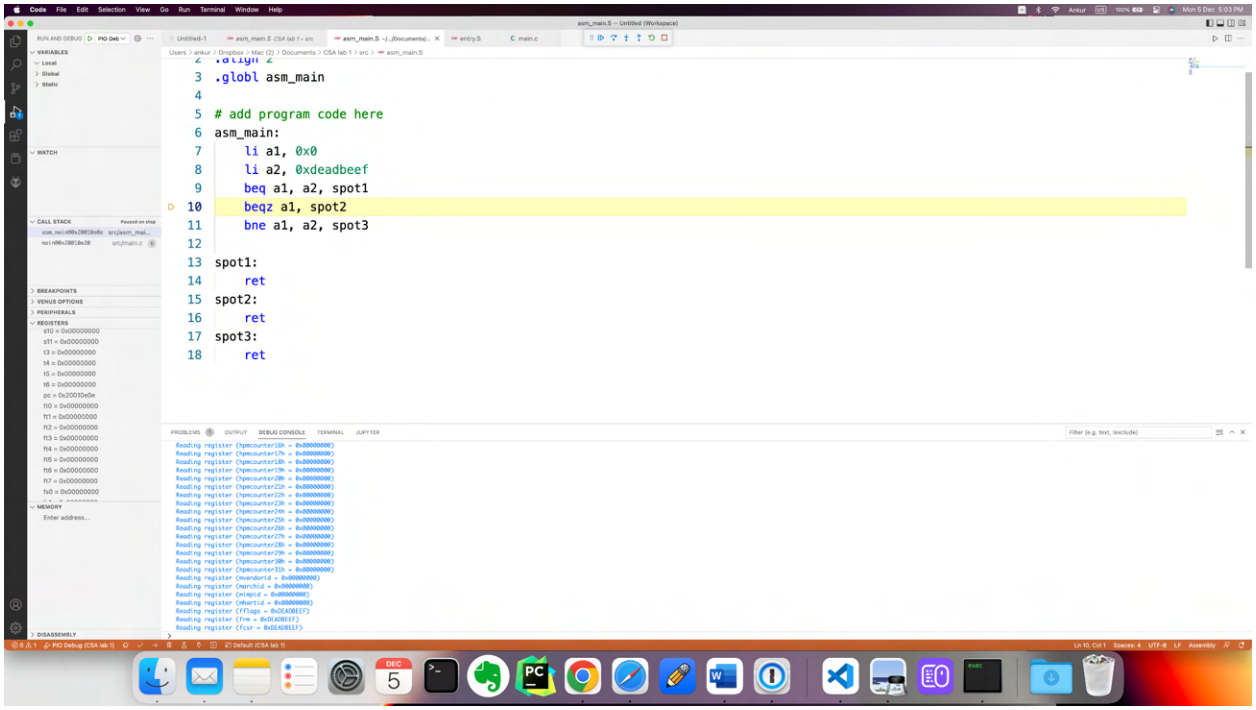

.globl asm_main
add program code here
asm_main:
li a1, 0x0
li a2, 0xdeadbeef
beq a1, a2, spot1
beqz a1, spot2
bne a1, a2, spot3
spot1:
ret
spot2:
ret
spot3:
ret

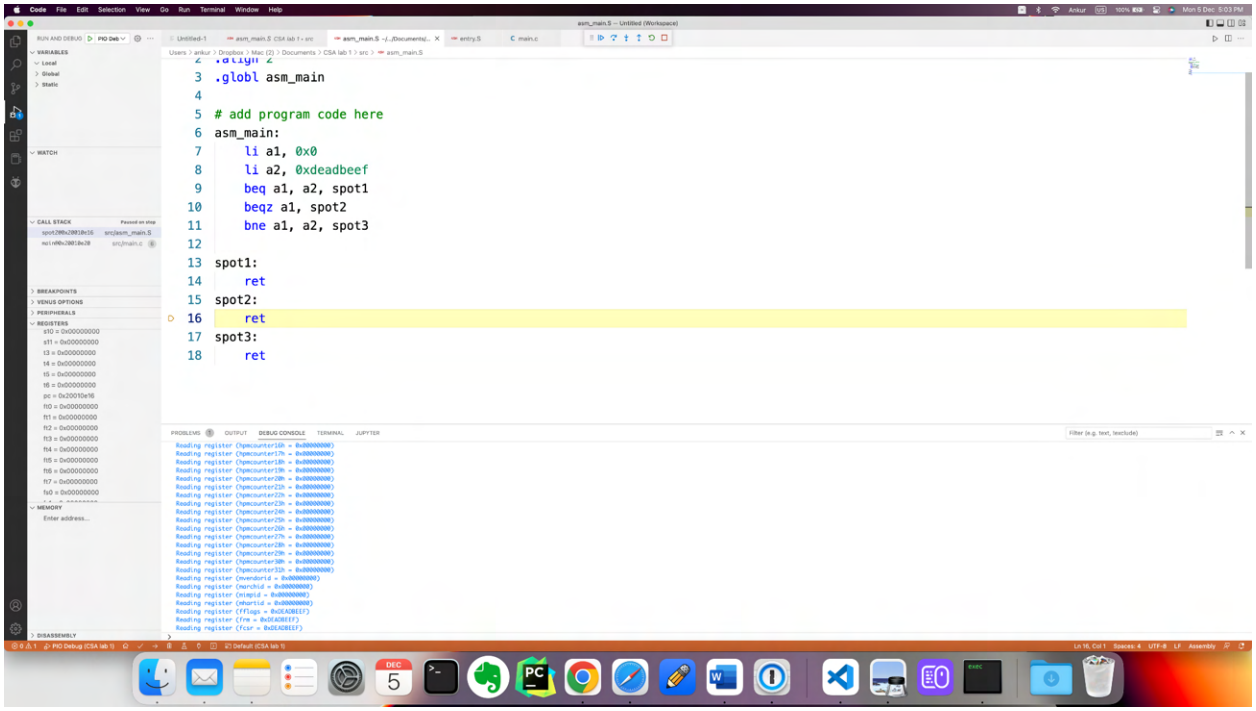


```
.globl asm_main

# add program code here
asm_main:
    li a1, 0x0
    li a2, 0xdeadbeef
    beq a1, a2, spot1
    beqz a1, spot2
    bne a1, a2, spot3

spot1:
    ret
spot2:
    ret
spot3:
    ret
```

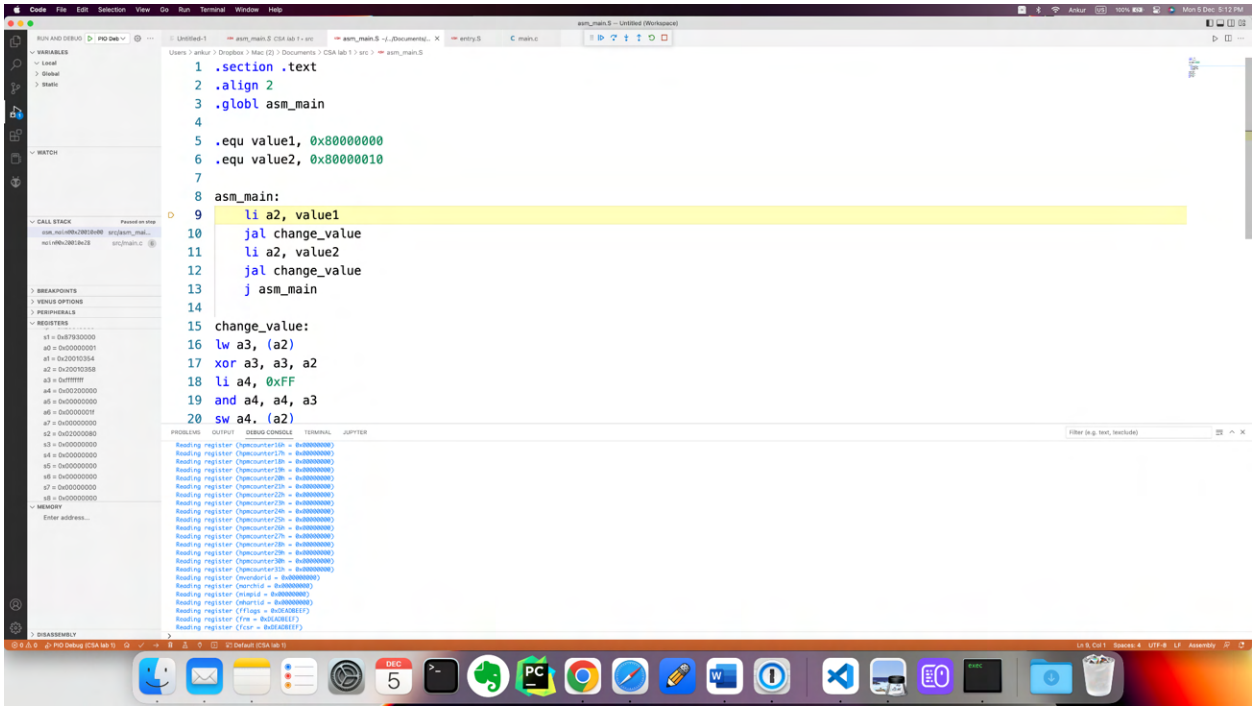


```
.globl asm_main

# add program code here
asm_main:
    li a1, 0x0
    li a2, 0xdeadbeef
    beq a1, a2, spot1
    beqz a1, spot2
    bne a1, a2, spot3

spot1:
    ret
spot2:
    ret
spot3:
    ret
```

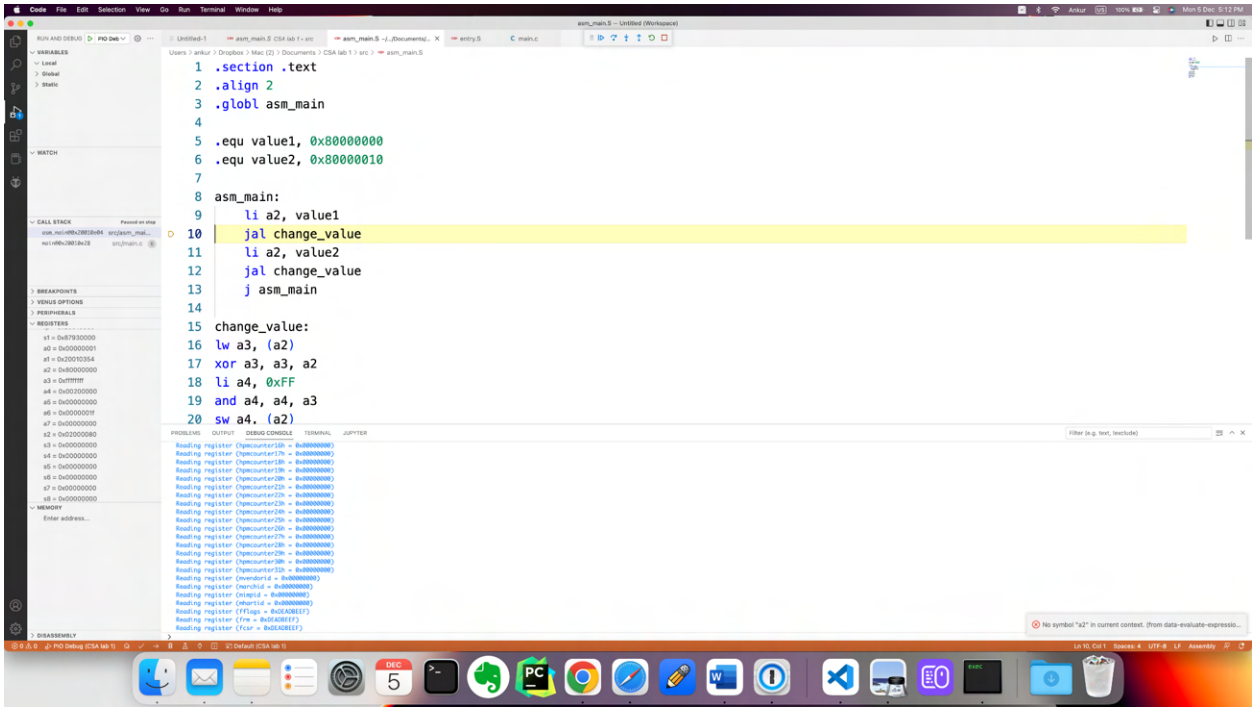

.globl asm_main
add program code here
asm_main:
li a1, 0x0
li a2, 0xdeadbeef
beq a1, a2, spot1
beqz a1, spot2
bne a1, a2, spot3
spot1:
ret
spot2:
ret
spot3:
ret


.globl asm_main
add program code here
asm_main:
li a1, 0x0
li a2, 0xdeadbeef
beq a1, a2, spot1
beqz a1, spot2
bne a1, a2, spot3
spot1:
ret
spot2:
ret
spot3:
ret

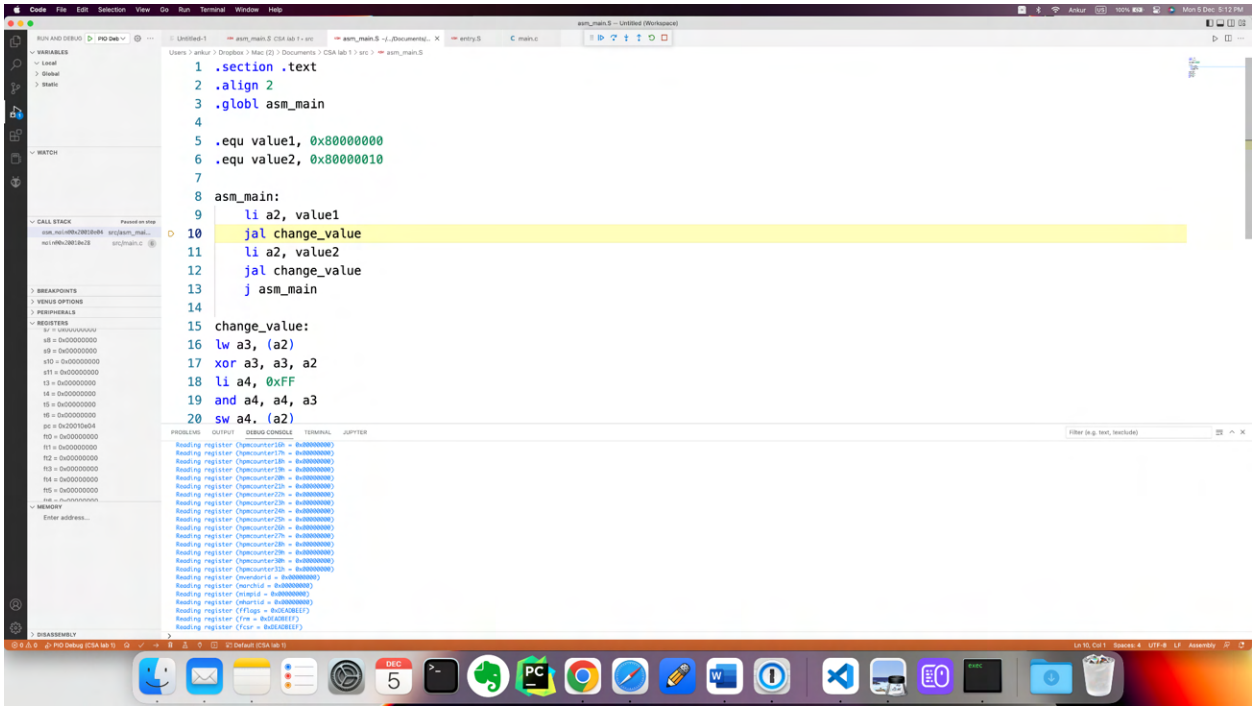
Part 5

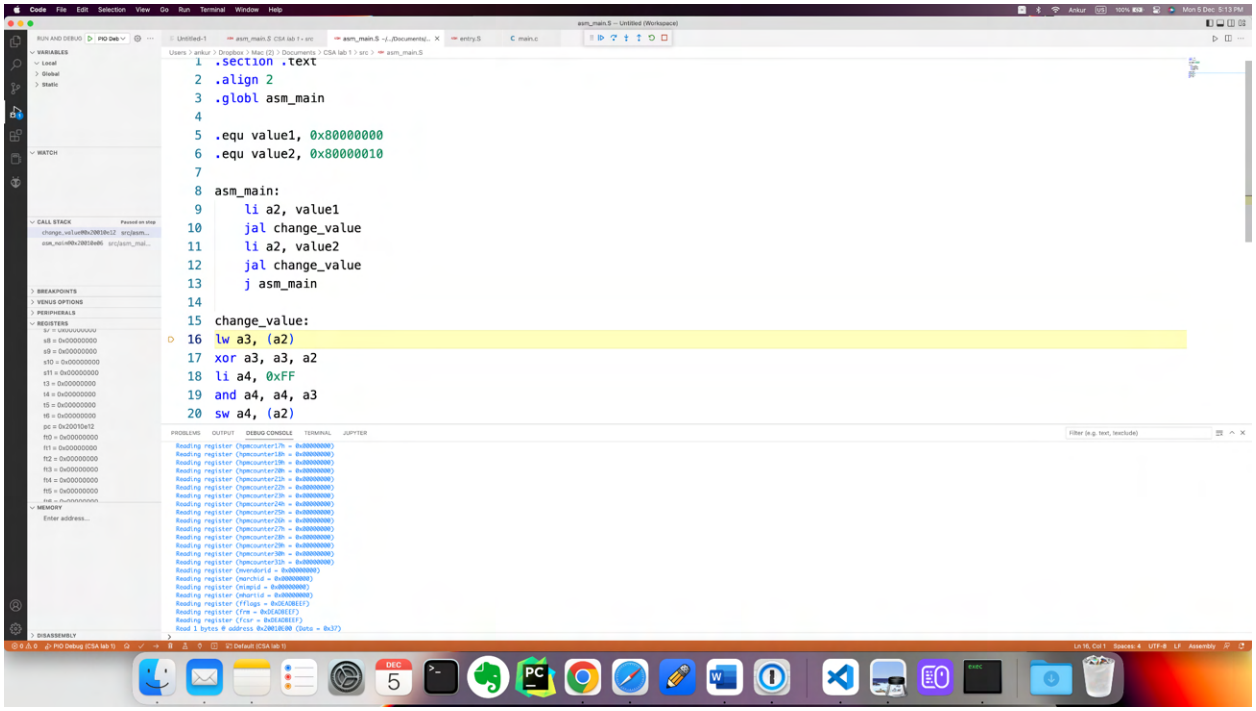


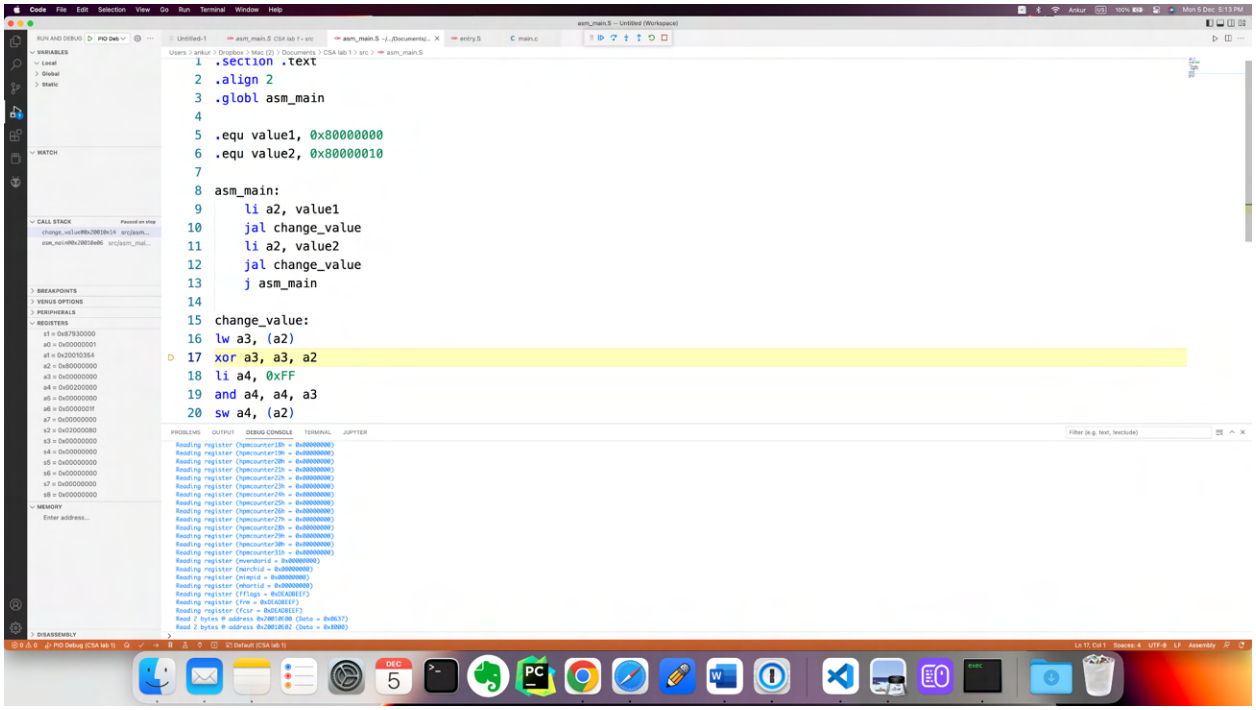
```
.section .text
.align 2
.globl asm_main

.equ value1, 0x80000000
.equ value2, 0x80000010

asm_main:
    li a2, value1
    jal change_value
    li a2, value2
    jal change_value
    j asm_main

change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
```



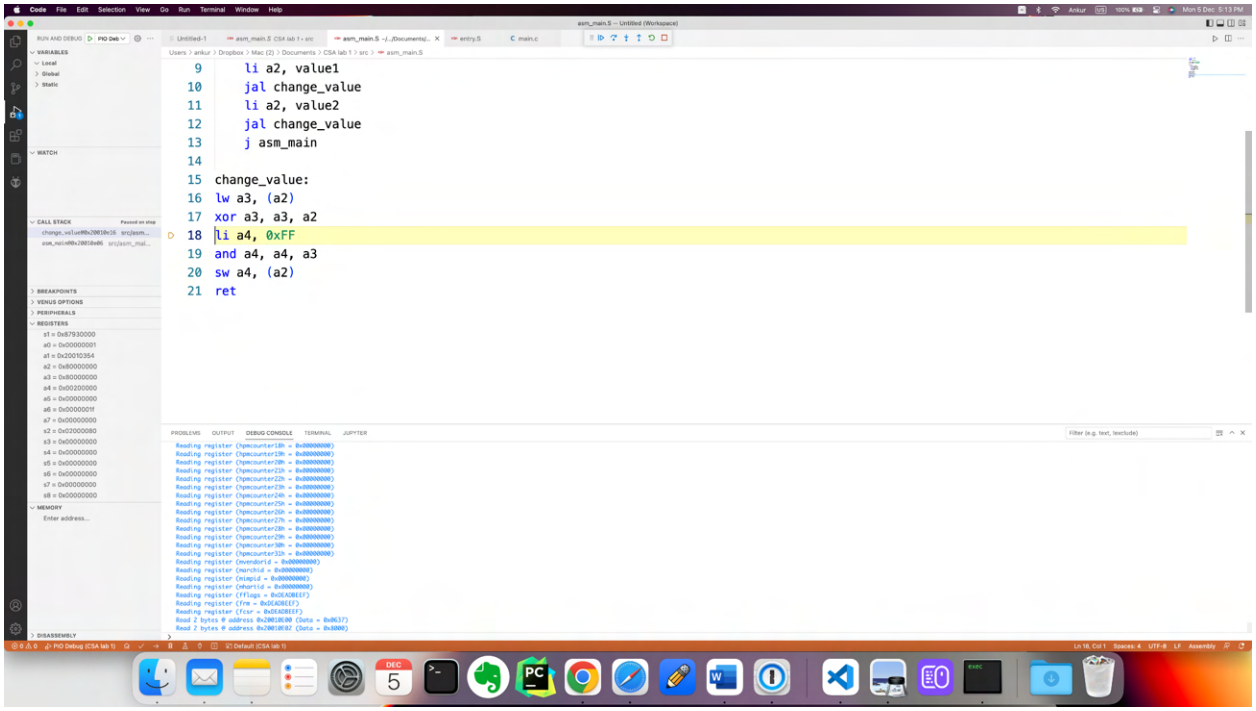
```
.section .text
.align 2
.globl asm_main

.equ value1, 0x80000000
.equ value2, 0x80000010

asm_main:
    li a2, value1
    jal change_value
    li a2, value2
    jal change_value
    j asm_main

change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
```


.section .text
.align 2
.globl asm_main
.equ value1, 0x80000000
.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)

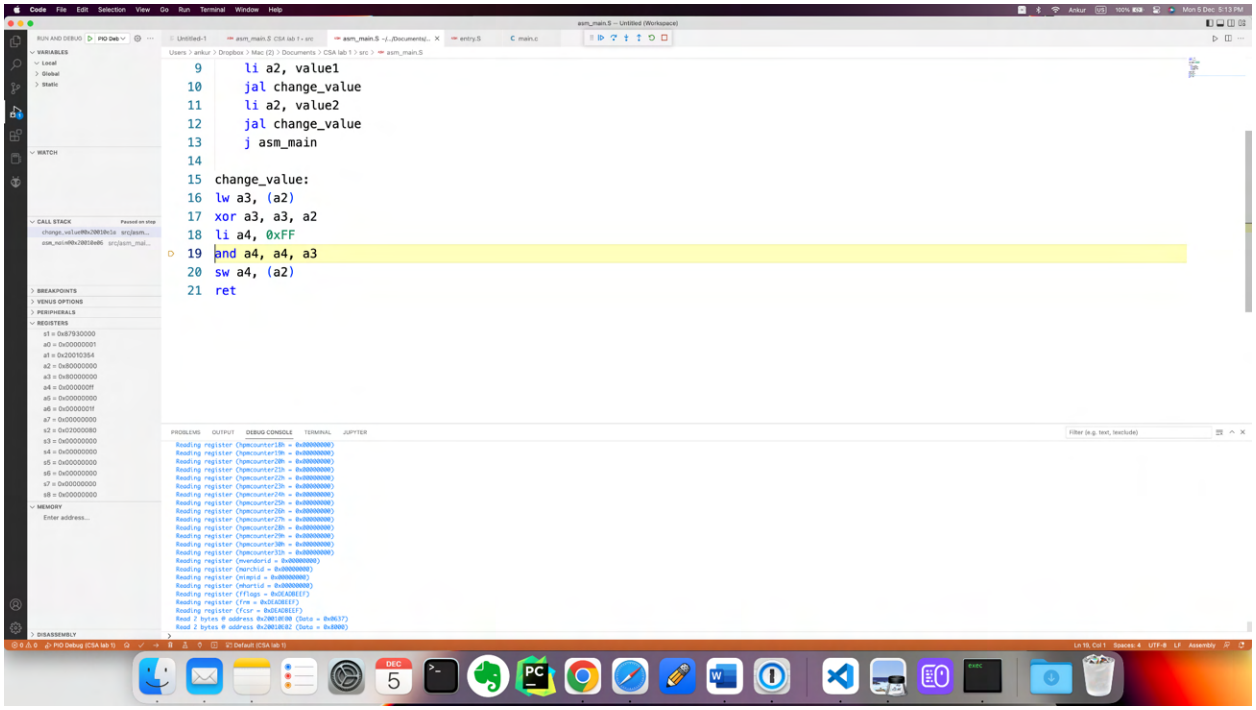

.section .text
.align 2
.globl asm_main
.equ value1, 0x80000000
.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)

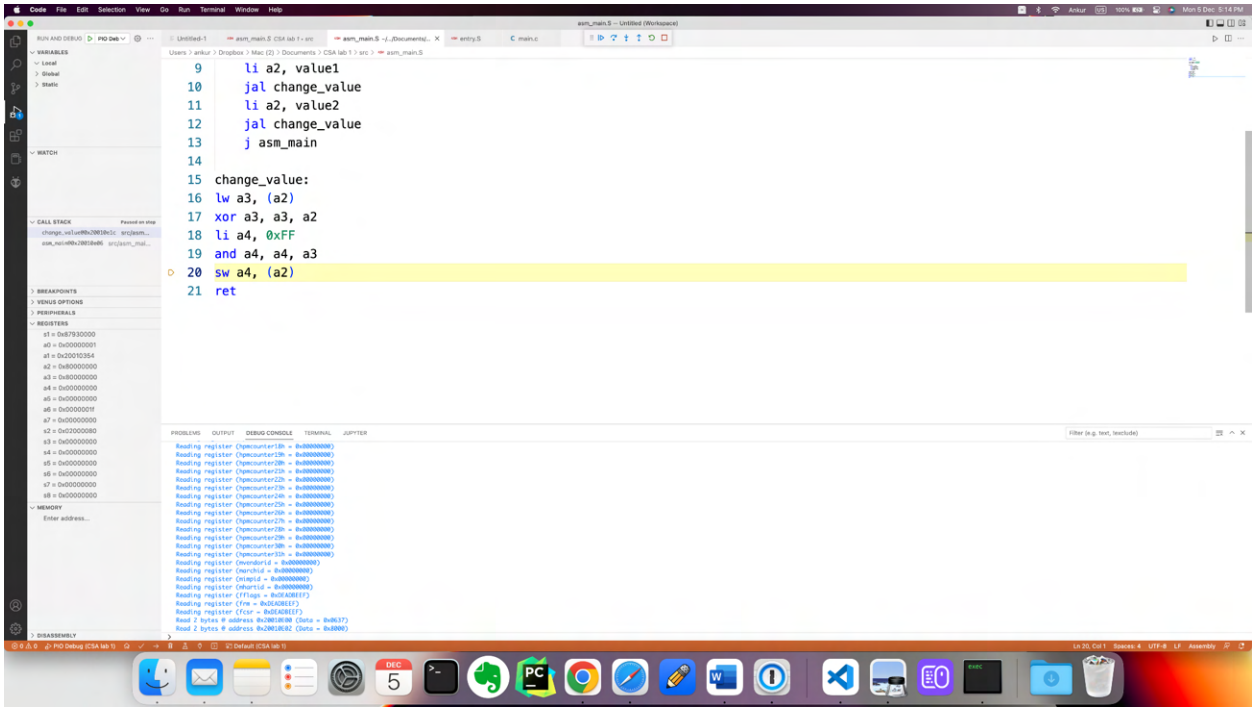


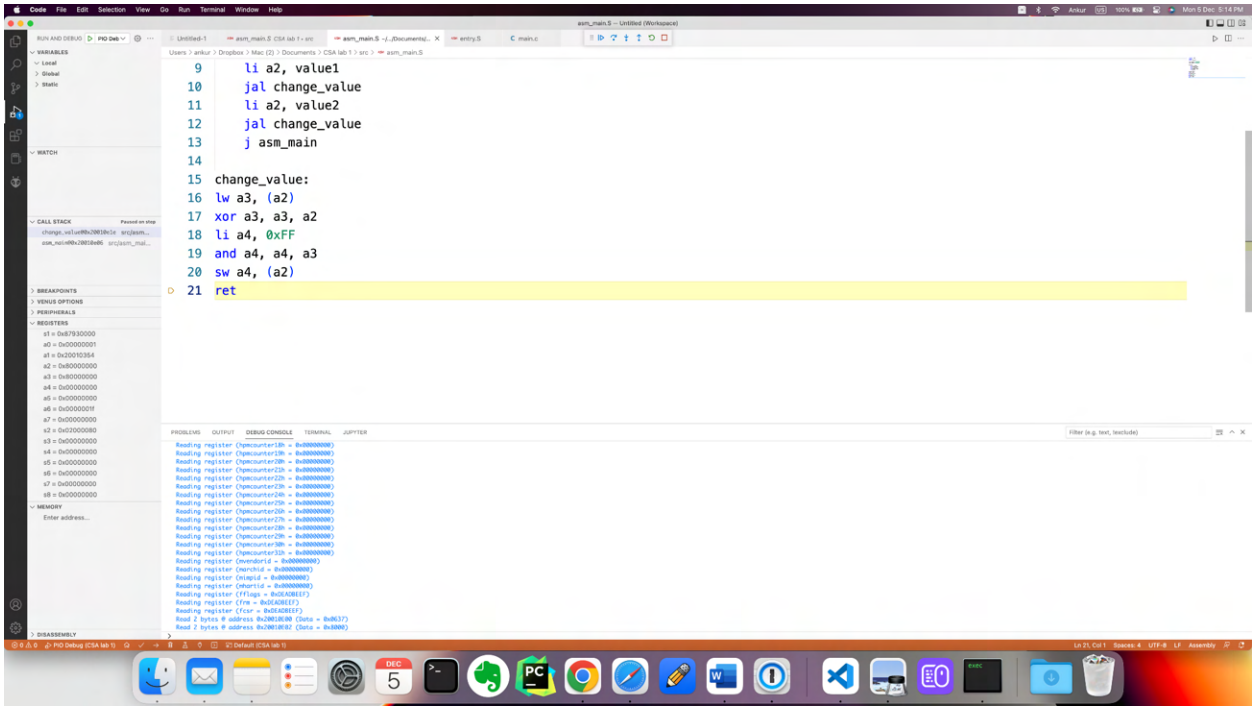
```
.section .text
.align 2
.globl asm_main

.equ value1, 0x80000000
.equ value2, 0x80000010

asm_main:
    li a2, value1
    jal change_value
    li a2, value2
    jal change_value
    j asm_main

change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
```

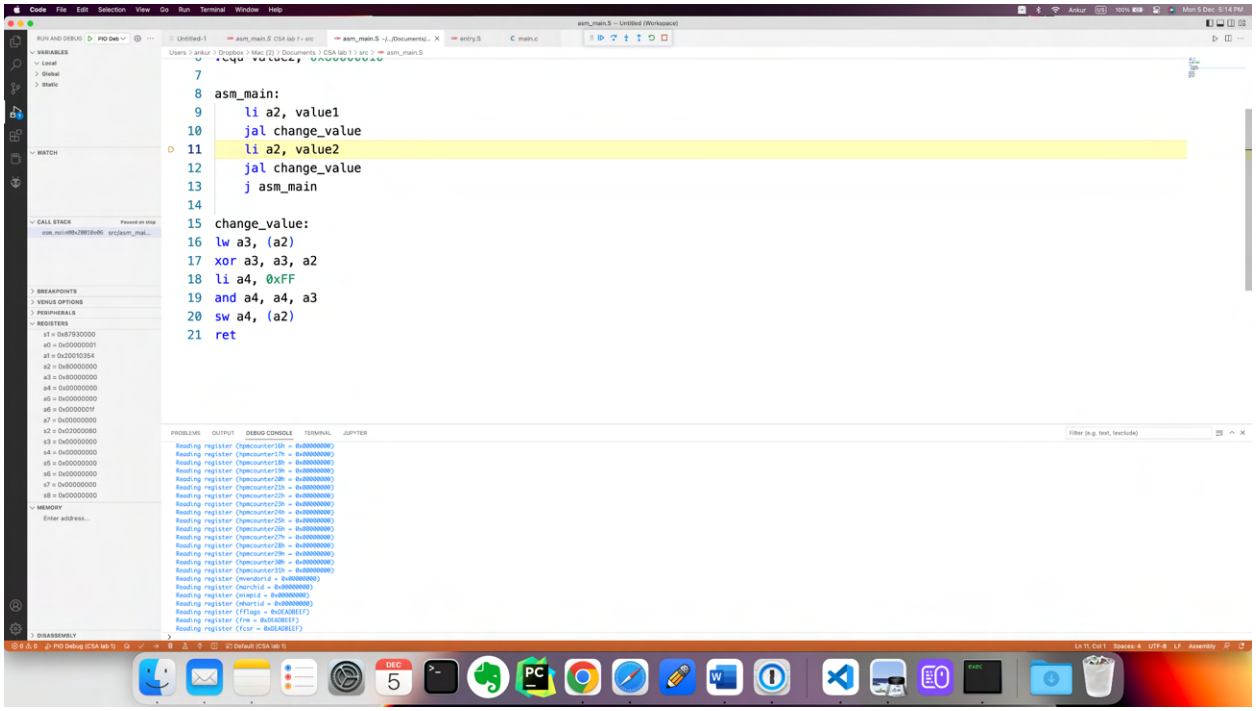


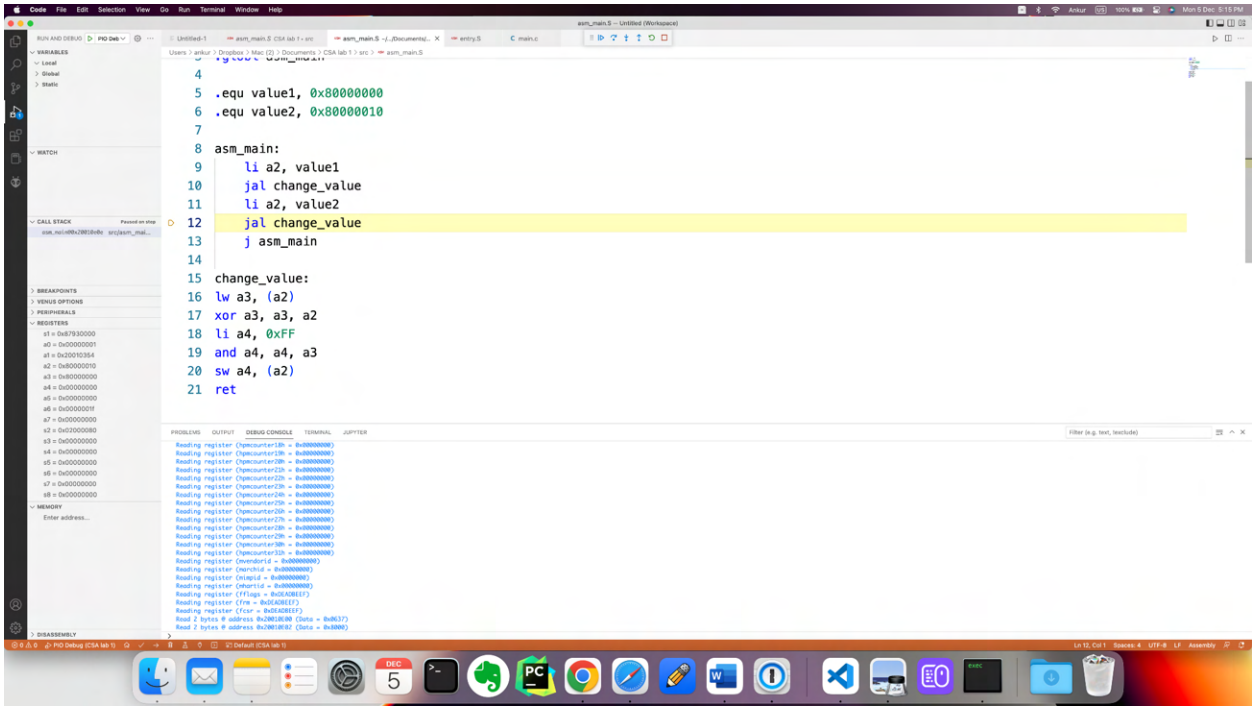
```
    li a2, value1
    jal change_value
    li a2, value2
    jal change_value
    j asm_main

change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret
```


li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret

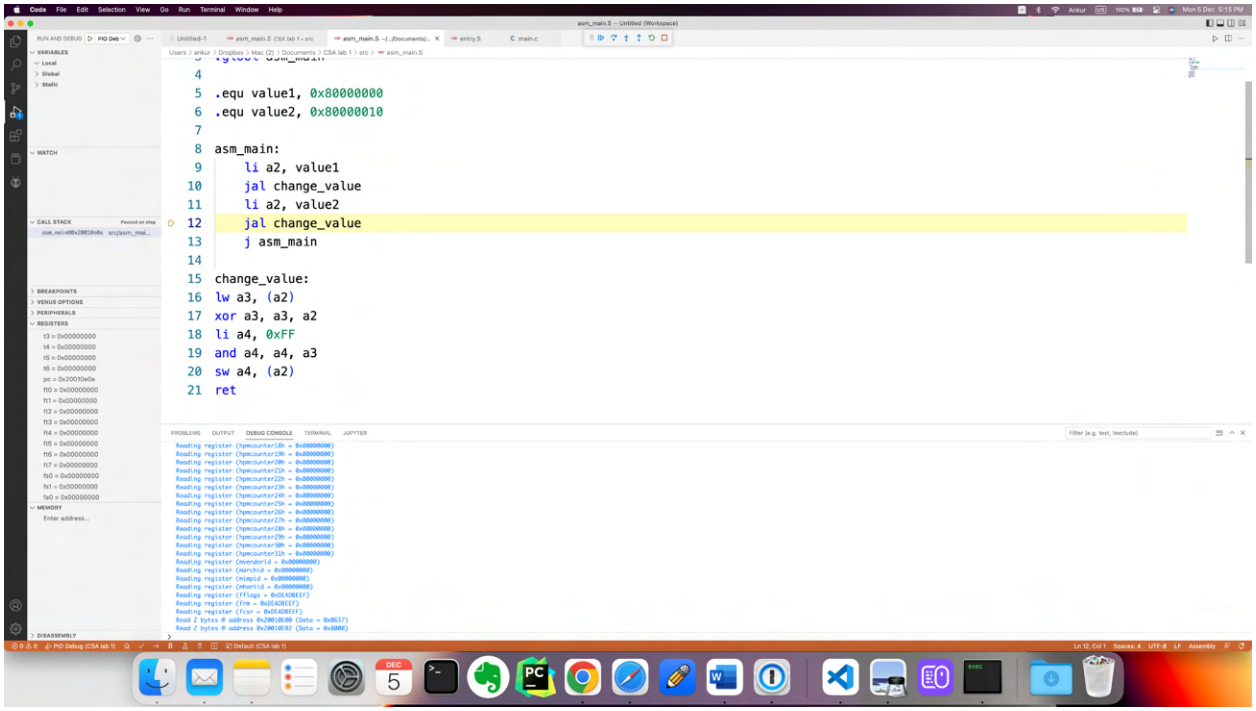

li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret


9 li a2, value1
10 jal change_value
11 li a2, value2
12 jal change_value
13 j asm_main
14
15 change_value:
16 lw a3, (a2)
17 xor a3, a3, a2
18 li a4, 0xFF
19 and a4, a4, a3
20 sw a4, (a2)
21 ret

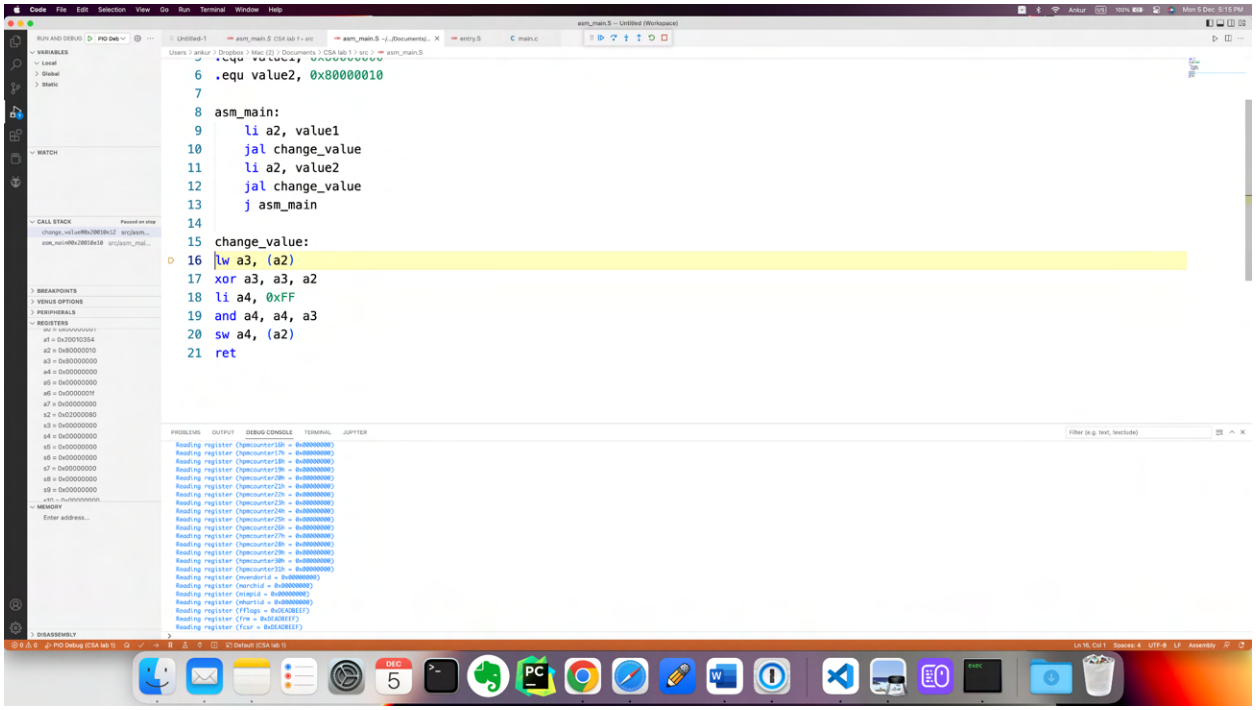

8 asm_main:
9 li a2, value1
10 jal change_value
11 li a2, value2
12 jal change_value
13 j asm_main
14
15 change_value:
16 lw a3, (a2)
17 xor a3, a3, a2
18 li a4, 0xFF
19 and a4, a4, a3
20 sw a4, (a2)
21 ret

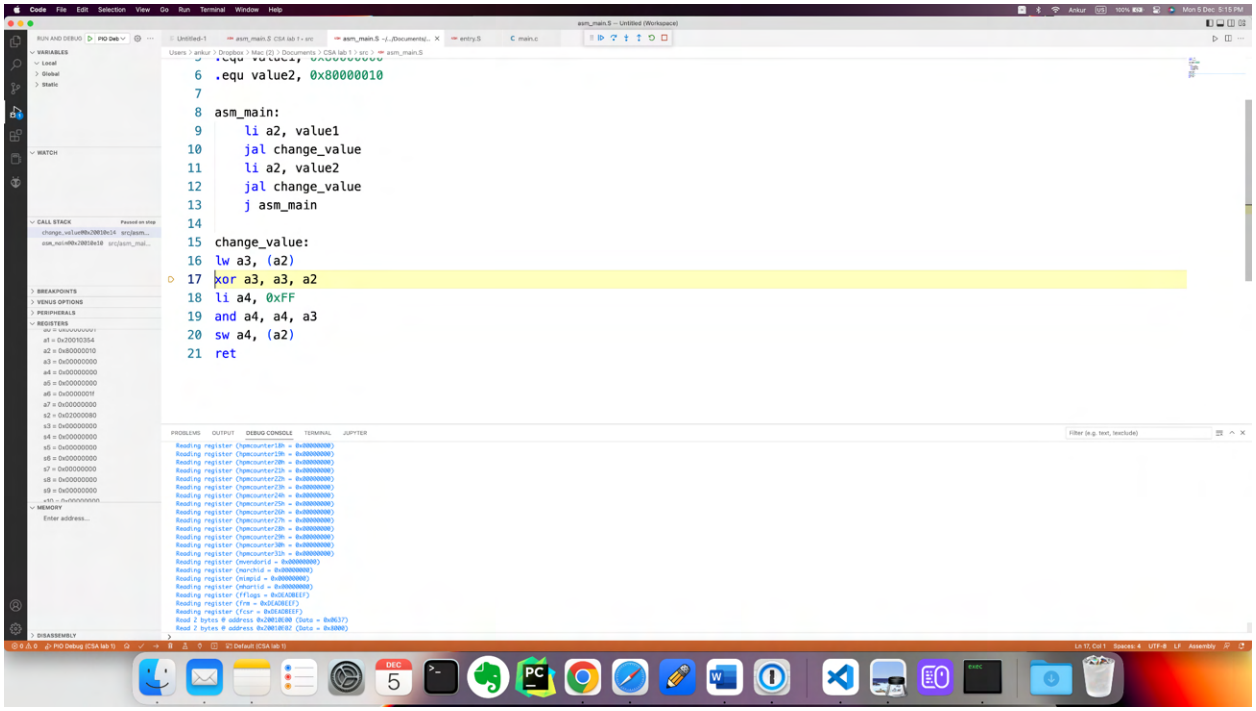


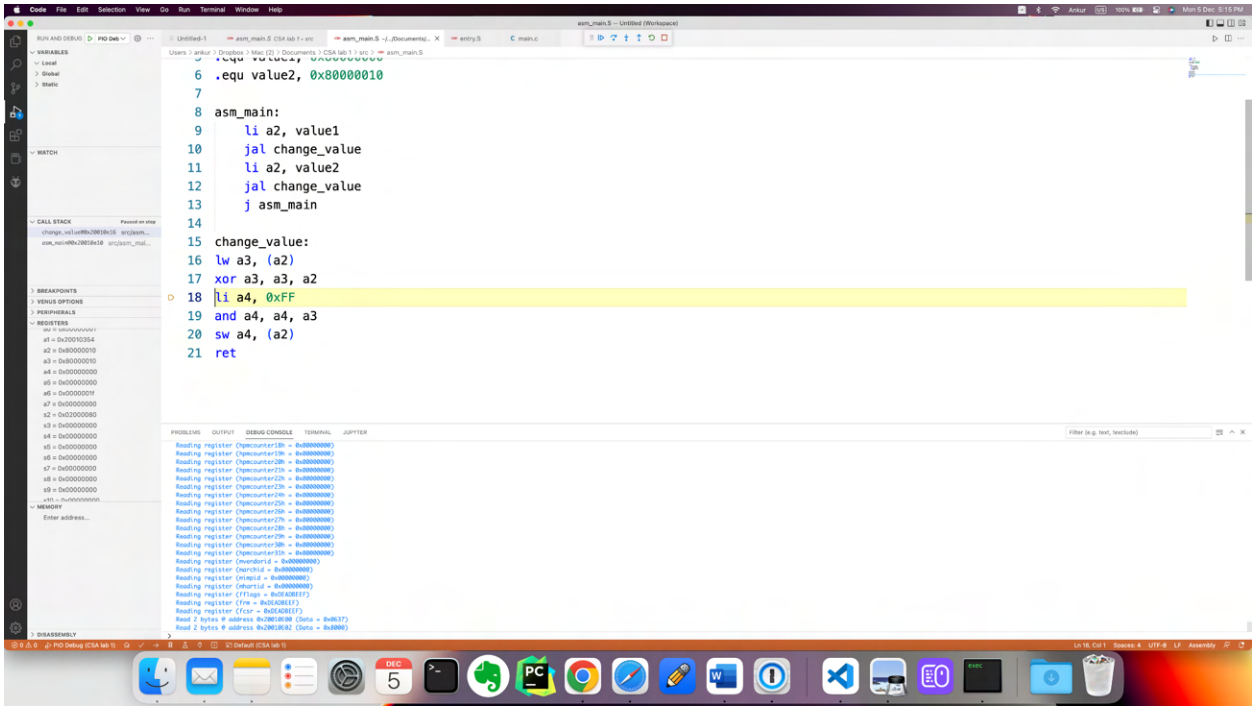
```
.equ value1, 0x80000000
.equ value2, 0x80000010

asm_main:
    li a2, value1
    jal change_value
    li a2, value2
    jal change_value
    j asm_main

change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret
```

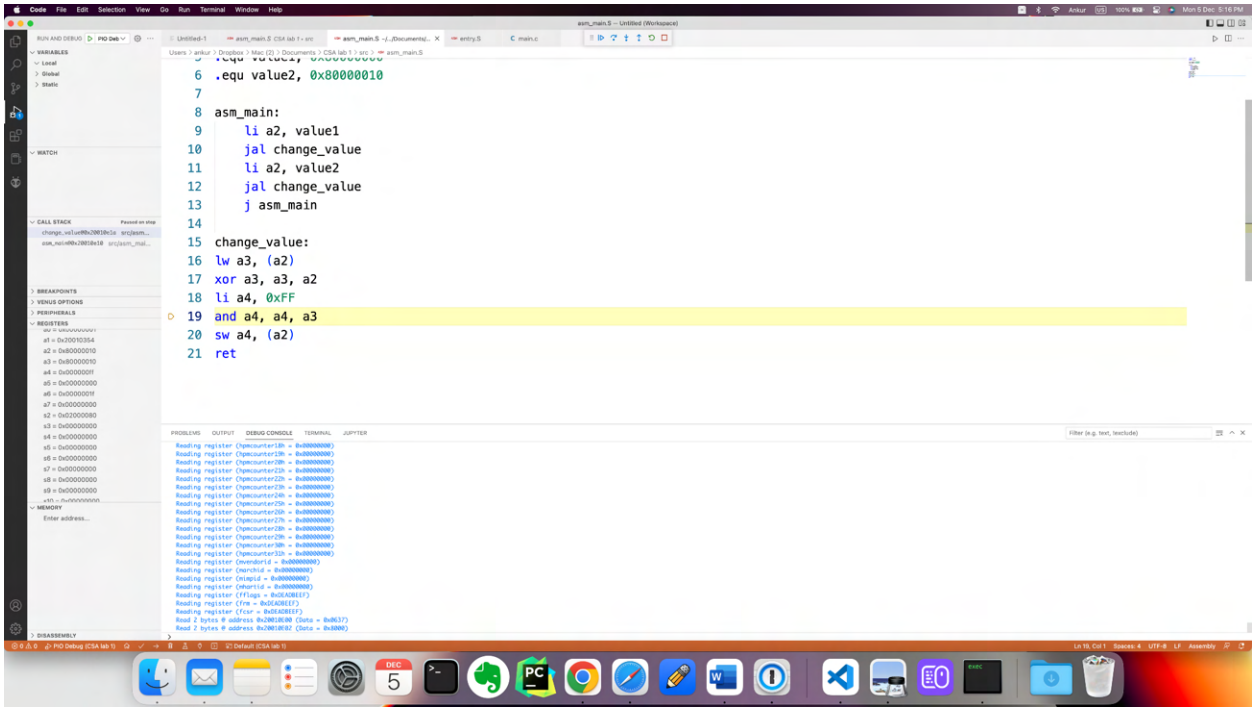


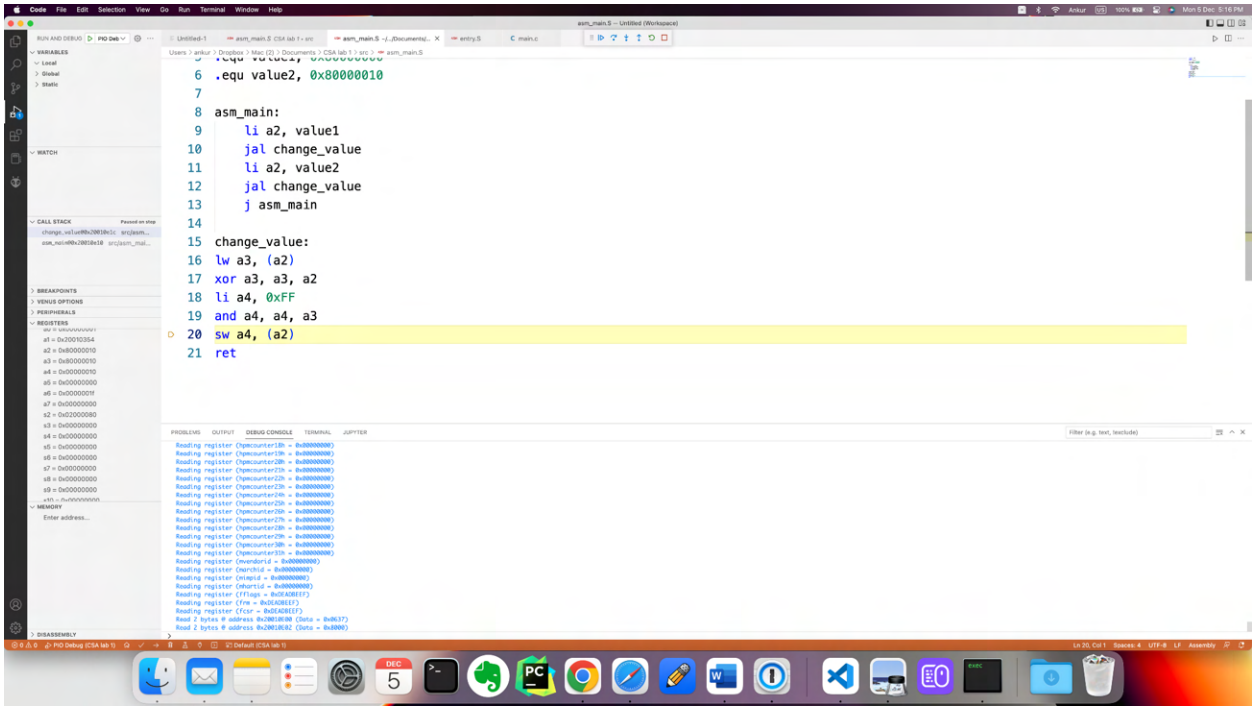
```
.equ value1, 0x80000000
.equ value2, 0x80000010

asm_main:
    li a2, value1
    jal change_value
    li a2, value2
    jal change_value
    j asm_main

change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret
```

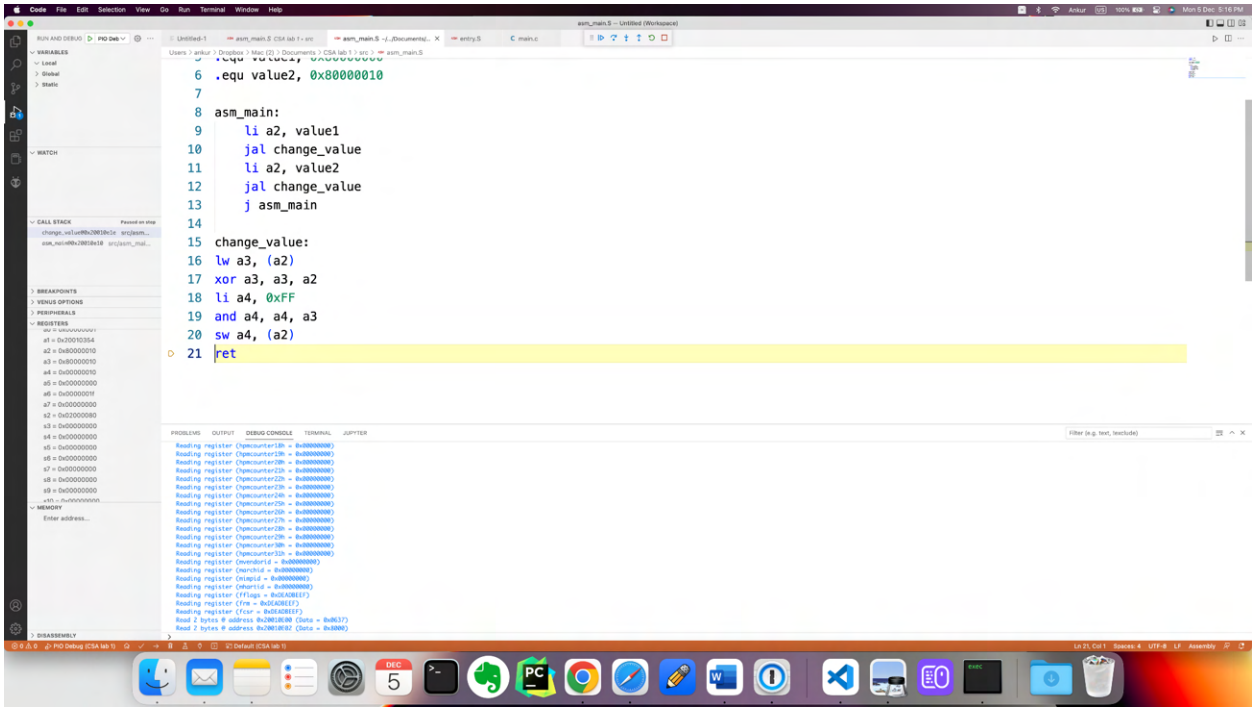

.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret

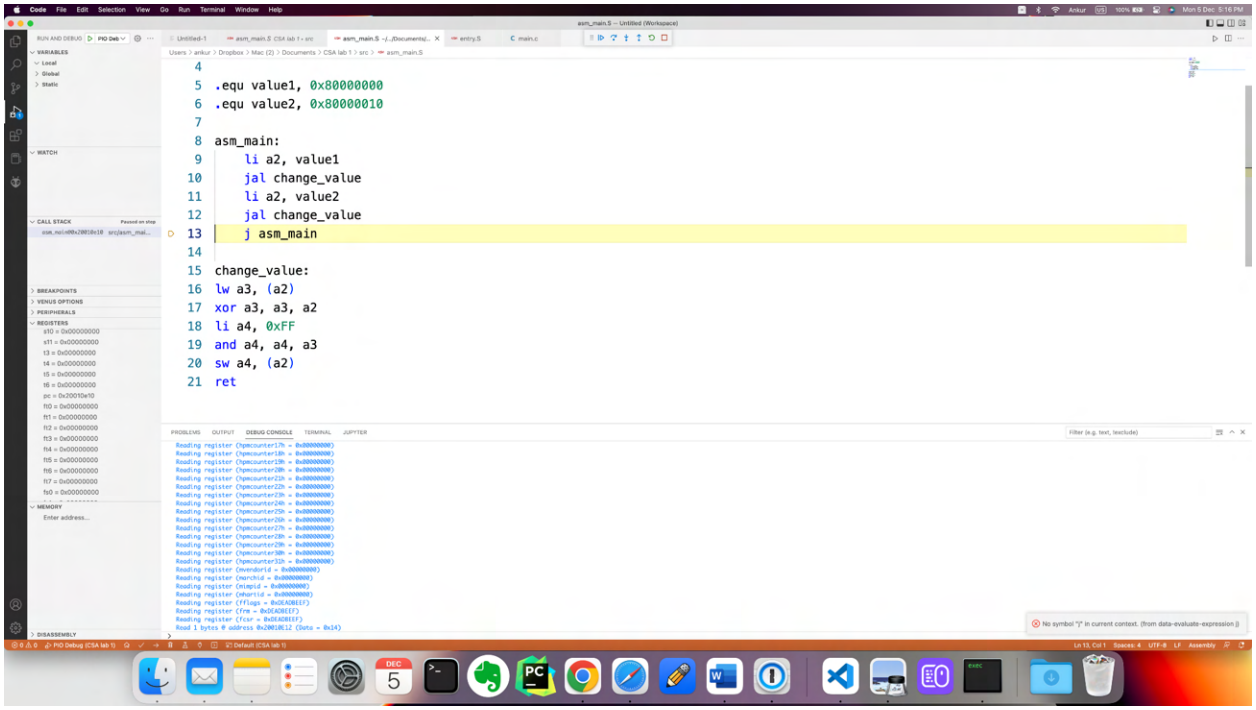

.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret


.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret


.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret


.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret


.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret


.equ value1, 0x80000000
.equ value2, 0x80000010
asm_main:
li a2, value1
jal change_value
li a2, value2
jal change_value
j asm_main
change_value:
lw a3, (a2)
xor a3, a3, a2
li a4, 0xFF
and a4, a4, a3
sw a4, (a2)
ret

Lab 2
Part 1
Q1. Modify main.c code to alternate between red, green, and blue LED using the colors array. Submit the modified file and a short video showing the blinking of LEDs in the required pattern.

Solution 1

```
#include <stdio.h>
#include <header1.h>
int main()
{
    int error = 0;
    int ledNum = 0;
    int colors[NUM_LEDS] = {GREEN_LED, BLUE_LED, RED_LED};
    setupGPIO();

    while (!error)
    {
        setLED(RED_LED, ON);
        delay(DELAY);
        setLED(RED_LED, OFF);

        setLED(GREEN_LED, ON);
        delay(DELAY);
        setLED(GREEN_LED, OFF);

        setLED(BLUE_LED, ON);
        delay(DELAY);
        error = setLED(BLUE_LED, OFF);
        if (ledNum >= NUM_LEDS)
            ledNum = 0;
    }
    return 0;
}
```

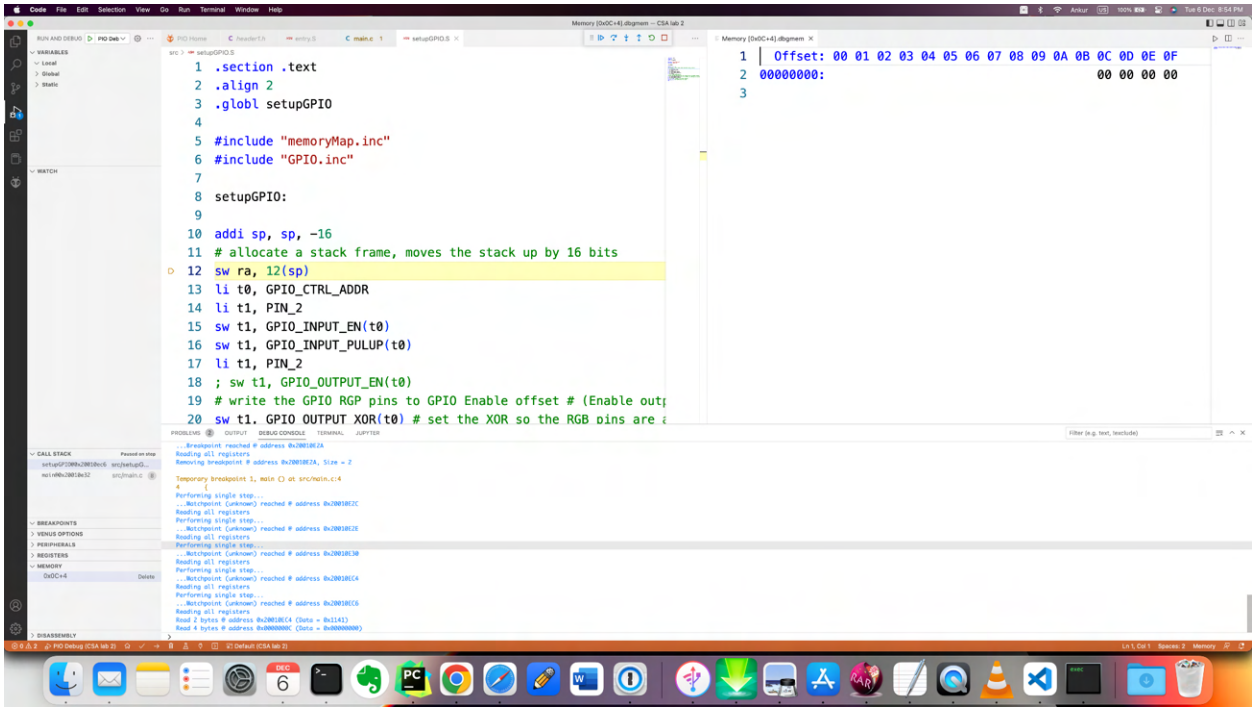
The video has been attached with file name: video1.mov
The code has been attached in file: lab_2,_part_1,_sol1_main.c

Question 2

Record the value of the output_val register at 0x0C for each LED. To do so debugger must be started and the memory location at 0x0C must be observed (use offset of 4 bytes for lookahead) and provide a screenshot of the same.

Solution 2

Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: 00 00 00 00



Question 3

The LED blink rate is controlled by the value in the equate DELAY in header1.h which is used to set how long the time between on and off will be. Run the original code and measure the blink rate by counting the number of times the LED blink in a given interval (Hint: Use the stopwatch feature on a phone to time 30 or 60 seconds). Change increase the DELAY by 50%. Measure the blink rate. Change increase the DELAY by another 50%. Measure the blink rate. Submit the modified file for each of the changes.

Solution 3

Blink rate with the original code
30 seconds = 72 times
60 seconds= 144 times

```
#define DELAY 200
```

```
#define ON 0x01
#define OFF 0x00
#define NUM_LEDS 0x03
#define RED_LED 0x400000
#define BLUE_LED 0x200000
#define GREEN_LED 0x080000
void setupGPIO();
int setLED(int color, int state);
void delay(int milliseconds);
```

The blink rate has now been increased by 50 percent
Number of blinks every 30 seconds = 54 blinks
Number of blinks every 60 seconds = 108 blinks

```
#define DELAY 300
#define ON 0x01
#define OFF 0x00
#define NUM_LEDS 0x03
#define RED_LED 0x400000
#define BLUE_LED 0x200000
#define GREEN_LED 0x080000
void setupGPIO();
int setLED(int color, int state);
void delay(int milliseconds);
```

The blink rate has now been increased by another 50 percent
Number of blinks every 30 seconds = 36 blinks
Number of blinks every 60 seconds = 72 blinks

```
#define DELAY 450
#define ON 0x01
#define OFF 0x00
#define NUM_LEDS 0x03
#define RED_LED 0x400000
#define BLUE_LED 0x200000
#define GREEN_LED 0x080000
void setupGPIO();
```

```
int setLED(int color, int state);
void delay(int milliseconds);
```

The code has been attached in file: lab2,part-1,-sol-3

Question 4
Modify the code to turn on an external LED connected to one of the GPIO ports. You may use pin 2 with bitfield offset of 0x40000, if any other pin is desired refer to FE310 manual. Submit the modified files and a short video showing the blinking of external LED in the required pattern.

Solution 4
The directory lab2,part1,ques4 contains the complete code. Two files, header1.h and main.c have been included below.

A video has been attached with file name: video2.mov. We have used a blue color LED.

header1.h

```
#define DELAY 200
#define ON 0x01
#define OFF 0x00
#define NUM_LEDS 0x03
#define RED_LED 0x400000
#define BLUE_LED 0x200000
#define GREEN_LED 0x080000
#define Blue_PIN_2 0x40000
void setupGPIO();
int setLED(int color, int state);
void delay(int milliseconds);
int checkBot();
```

main.c

```
#include <stdio.h>
#include <header1.h>
int main()
{
 int error = 0;
 int ledNum = 0;
```

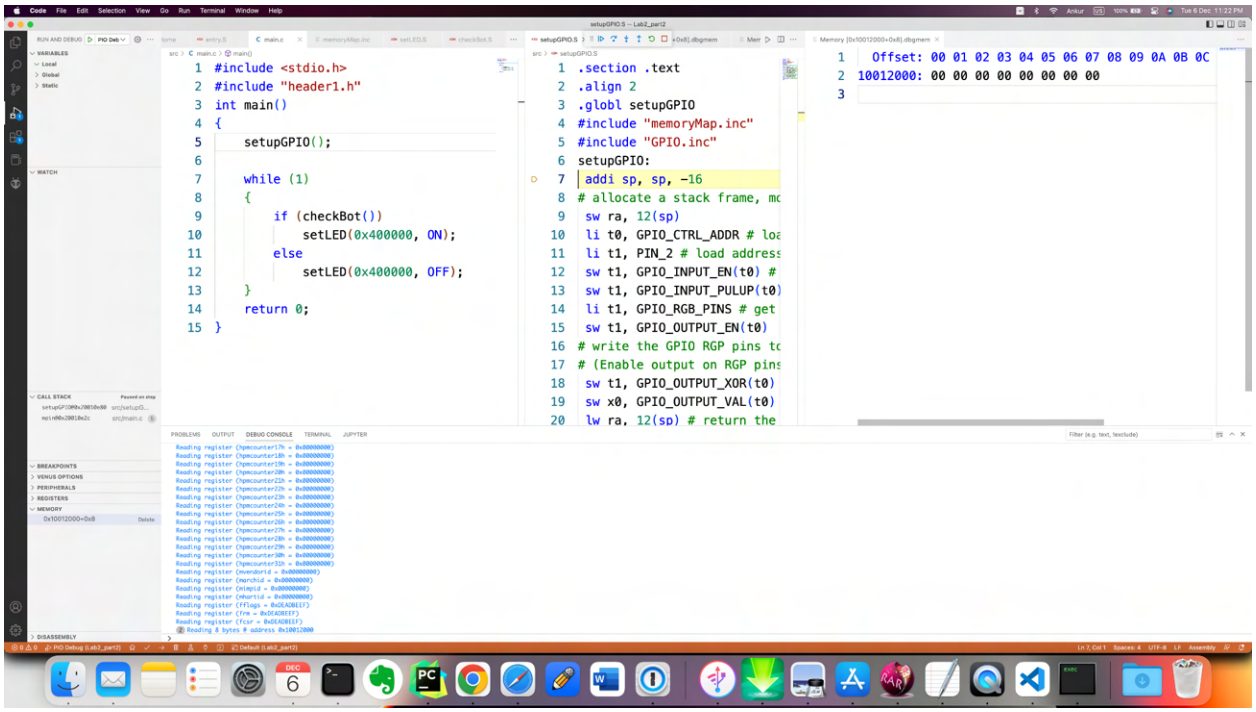
```
 int colors[NUM_LEDS] = {GREEN_LED, BLUE_LED, RED_LED};
 setupGPIO();
 while(!error)
 {
    ledNum++;
    if(ledNum >= NUM_LEDS)
        ledNum = 0;

    setLED(Blue_PIN_2, ON);
    delay(DELAY);
    error = setLED(Blue_PIN_2, ON);
    delay(DELAY);
 }
return 0;
}
```

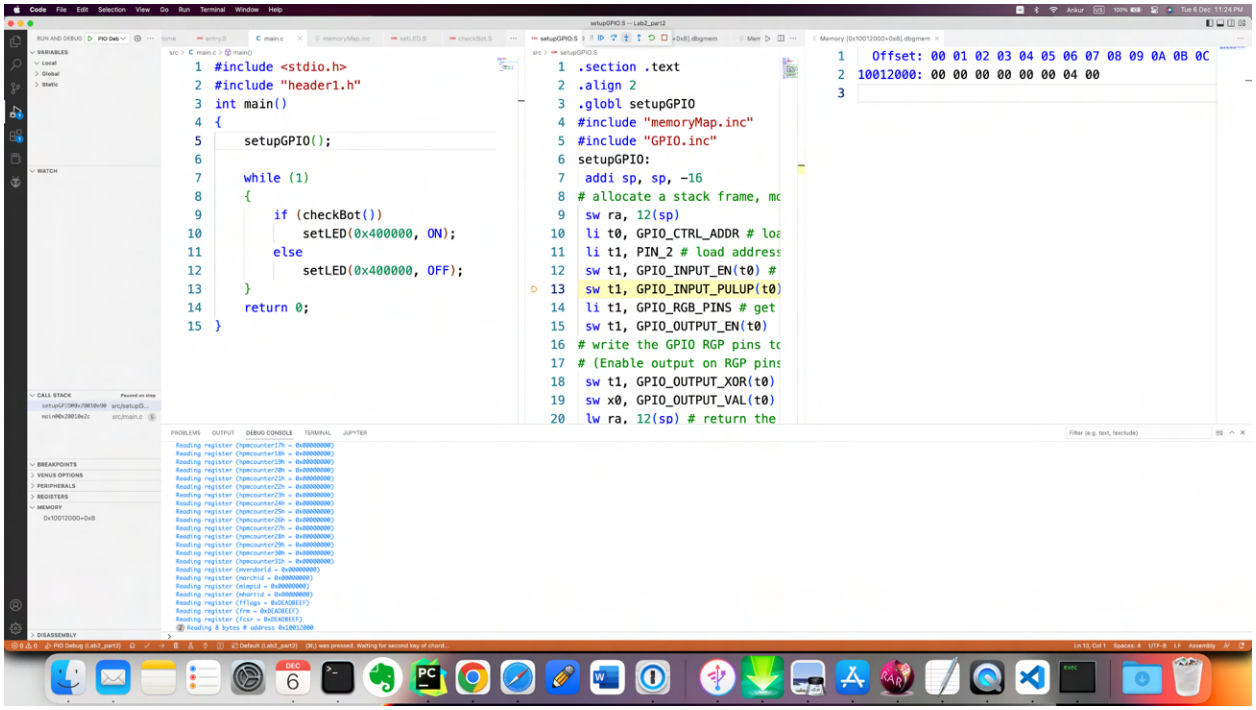
Part 2: GPIO's as Inputs

Question 1

Record the value of at the GPIO input memory location with and without the button press. (Remember that the button must be held or released during debug session)

Solution 1





Question 2
Explain what GPIO_INPUT_PULUP address does and how can the pull up be avoided.

Answer 2
The GPIO_INPUT_PULLUP address for a HiFive version microprocessor gives a logic input with an internal pullup. HiFive version of microprocessors are multicore processors and are accomplished with Linux operating system.

The GPIO_INPUT_PULLUP address is given at the input of a certain pin of the microprocessor which is configured for the purpose. This is done while enabling the internal pullup resistor of the microprocessor. A pullup condition occurs when a pin of the microprocessor is not pulled to a logical high or low level (i.e. 1 or 0). Instead, it floats between the high and low levels. This causes a problem since the processor may assume it to be a high or low level signal.

This problem can be avoided by connecting pullup resistors (resistors of fixed values) between the supply voltage (Vcc for the microprocessor) and the required pin. In absence of an input signal, the pin is pulled up to a logical high level.

For a supply voltage (Vcc) of 5V, generally the value of the pull up resistor used is kilo ohms, which is used before a switch.

However, for circuits requiring a low power consumption, it may be required to use higher values of pull up resistors. Higher values of pullup resistors (such as 10 kilo ohms to 100 kilo ohms or upto 1 mega ohms) may be used to limit the input to smaller values. Thereby, it can reduce the unnecessary high power consumption as well as reduce heating of the device. Hence, pull up conditions in microprocessors can be successfully avoided by using pull up resistors of suitable value depending on the application. These resistors eliminate the misinterpretation of the microprocessor whether a certain pin is in the logical high or low level and also ensures a low amount of current consumption.

Lab 3

The video has been attached with file name: video3.mov
The code has been attached in file: lab3.zip