GAZI UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING

Experiment #3

Ahmet Emin Karakaya
191110046

In Figure 1, I made some definitions to determine the burning intervals of the leds. Then I made an address definition to keep these values at any address. After that I created code AREA and start for __main

```
fast_delay EQU 0X4C0000      ; Fast_delay time defined
slow_delay EQU 0X0C0000      ; Slow_delay time defined
delay_address EQU 0x20000400 ; Delay_address defined


 AREA MYCODE, CODE
     ENTRY
     EXPORT __main
__main

```

Figure 1

Button control was done in Figure 2. The button on port D has been checked. If the button status is logic 0, it went to the slowBlink branch, if not to the fastBlink branch.

```
checkSwitch     LDR R1, =gpioDbase
                ADD R1, #0x04
                LDR R0, [R1]                ; data read in R0
                ;MOV R0,1

                CMP R0, #0    ; Button status checked
                BEQ slowBlink ; If the button state is logic 0, slowBlink branch has been moved
                BNE fastBlink ; If the button state is not logic 0, fastBlink branch has been moved


```

Figure 2

In Figure 3, we observe the fastBlink branch. Here, the fast_delay value is written to the R2 register and the address where we will write that value is written to the R3 register. Then the value in register R2 is written to register R3. Once these assignments were made it went to the blink branch. In fact, the purpose here is to determine how many seconds the led lights up. This branch was for fastBlink.

```

fastBlink

                LDR R2, =fast_delay     ; Fast_delay time written to register R2
                LDR R3, =delay_address  ; Delay_address written to R3 register
                STR R2, [R3]            ; Fast_delay time is written to delay address
                B blink                 ; Going to blink branch

```

Figure 3

In Figure 4, we observe the slowBlink branch. Here, the slow_delay value is written to the R2 register and the address where we will write that value is written to the R3 register. Then the value in register R2 is written to register R3. Once these assignments were made it went to the blink branch. In fact, the purpose here is to determine how many seconds the led lights up. This branch was for slowBlink.

```
slowBlink

            LDR R2, =slow_delay     ; Slow_delay time written to register R2
            LDR R3, =delay_address  ; Delay_address written to R3 register
            STR R2, [R3]            ; Slow_delay time is written to delay address
            B blink                 ; Going to blink branch

```

Figure 4

Led blink operation is done in Figure 5. We turned the ORR directive leds on and then went to the delay branch. After the delay process, he came back to where he left off. Then we turned off the leds with the BIC directive and went back to the delay branch. After the delay process was finished, the code went to the branch that controls the button status and checked the button status again and it was decided whether it was fast or slow accordingly.

```

blink
            LDR R1, =gpioEbase
            ADD R1, #0x0C               ; R1 stores address of data reg port E (make sure not to change in any bra
            LDR R0, [R1]


            ORR R0, R0, #0x03   ; With the ORR operation, the first two bits of the R0 register are made one and the f
            STR R0, [R1]        ; The value in register R0 has been written to the address indicated by register R1
            BL delay            ; Here the code branches into the delay part

            LDR R0, [R1]

            BIC R0, R0, #0x03   ; With the BIC operation, the first two bits of the R0 register are set to zero and th
            STR R0, [R1]        ; The value in register R0 has been written to the address indicated by register R1
            BL delay            ; Here the code branches into the delay part

            B checkSwitch   ; Button status went to checkSwitch branch to recheck

```

Figure 5

In Figure 6, delay and wait branches are seen. In the delay branch, the waiting time is written to the R8 register as desired. After leaving here, the code went to the wait branch. In this branch, we reduced the waiting time to 0 and the code waited here.

```
delay

            LDR R8, [R3]    ; The value at address R3, which holds the delaying time, is written to register R8

wait

            SUB R8, R8, #1 ; Delay time reduced by 1
            CMP R8, #0     ; Comparing whether the delay time is 0
            BNE wait       ; If not 0 it went back to the beginning of the loop
            BX  LR
```

Figure 6

The general description of the code part was like this. The screenshots that follow are screenshots showing the code working. While running the code, I made some changes to the code. For one of them, I set the fast and slow values to 2 and 5, respectively. In the other, I set the value of the button to 1. In Figure 7, we observed that we made the button state 1 in the R0 register.

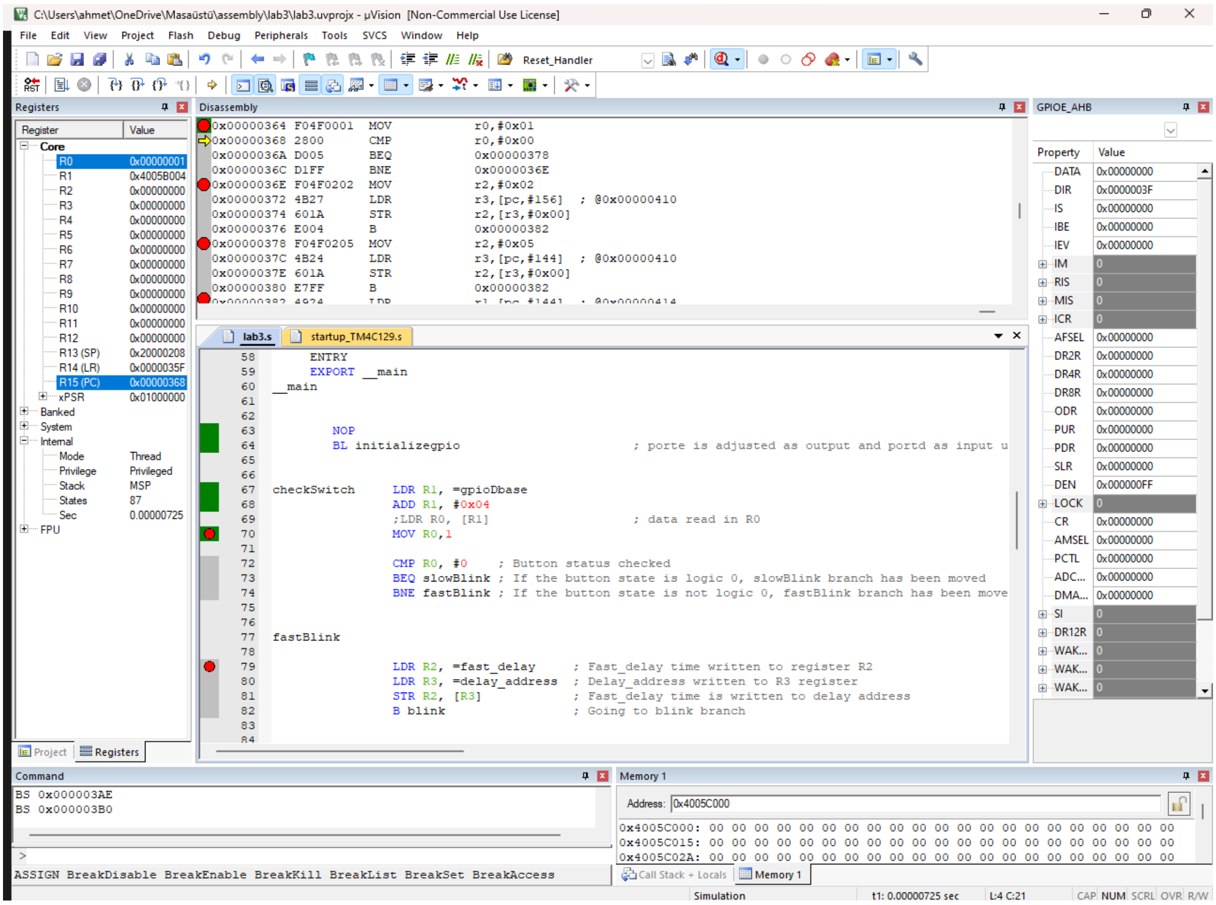

Figure 7

Since I made the button state 1 in Figure 8, the code entered the fastBlink branch. And fast_delay(2) value is written to R2 register.

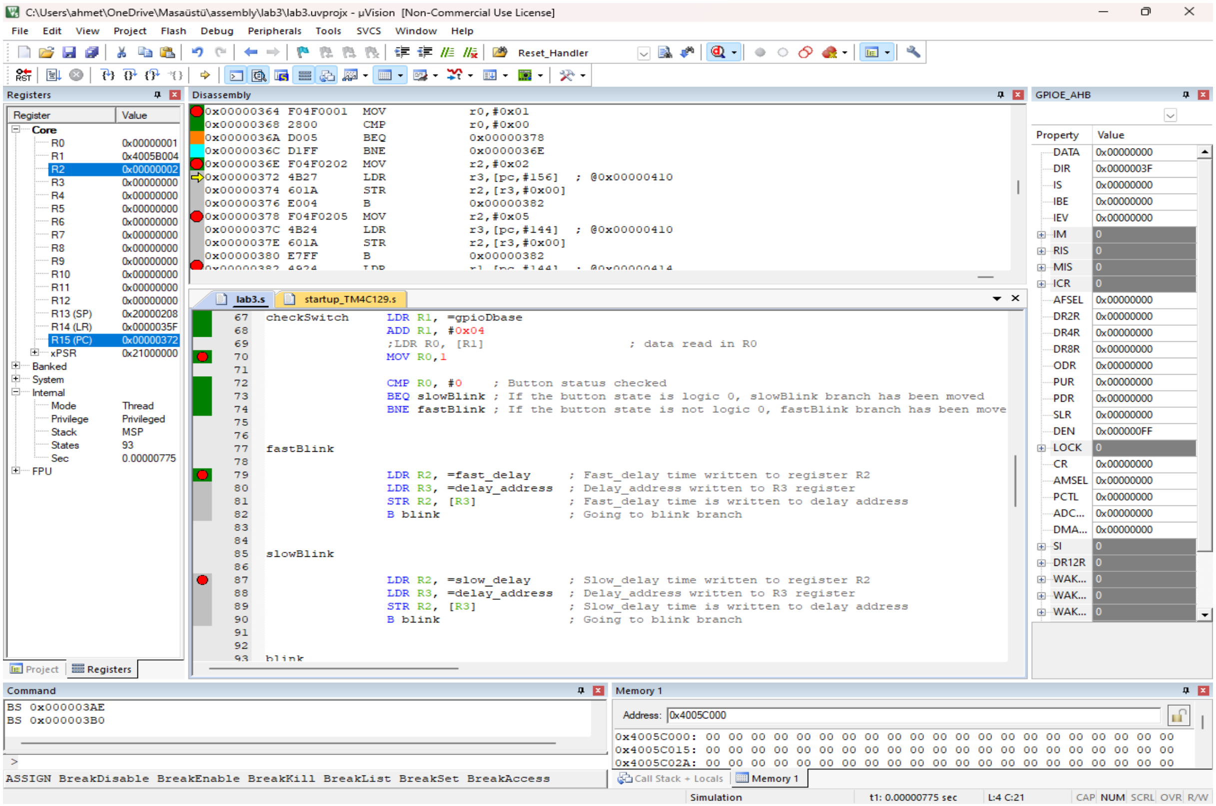

Figure 8

In Figure 9, we put the leds on and went to the delay branch. After the operations were done there, the wait branch was entered and the waiting began there.

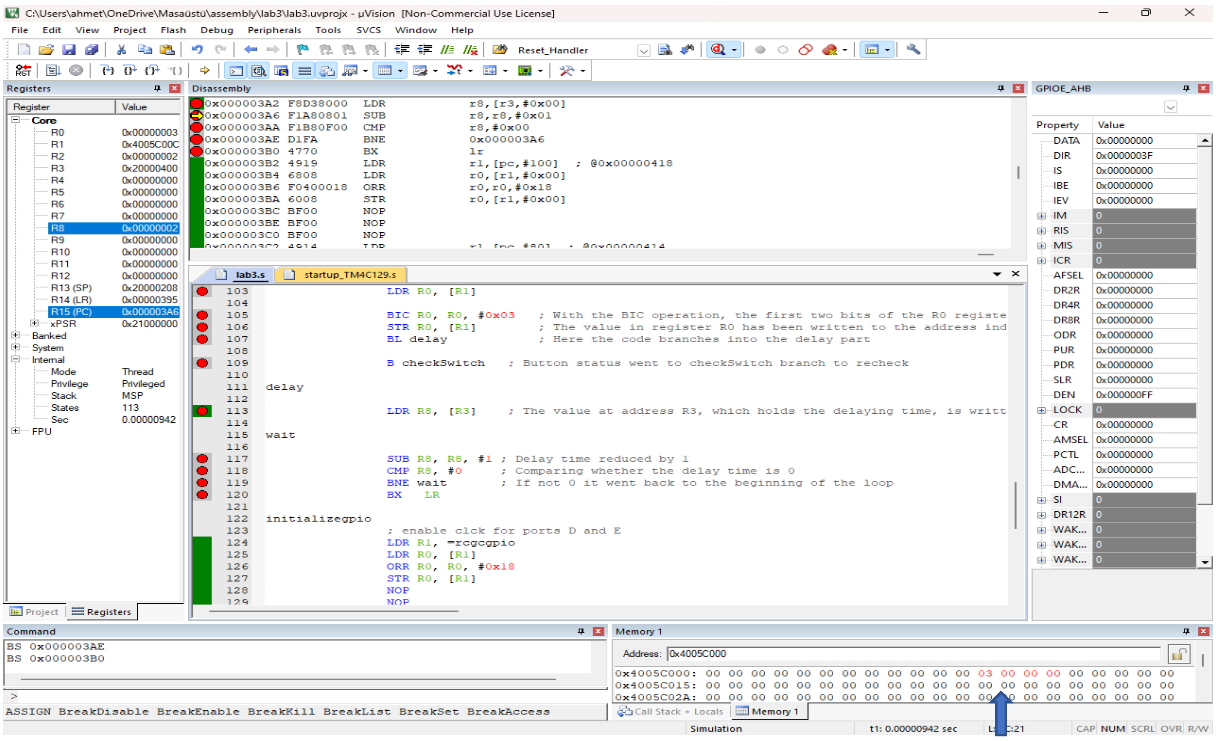

Figure 9

Figure 10 is the screenshot showing the wait branch spinning inside. In Figure 10 we reduced the time by 1 and register R8 became 1.

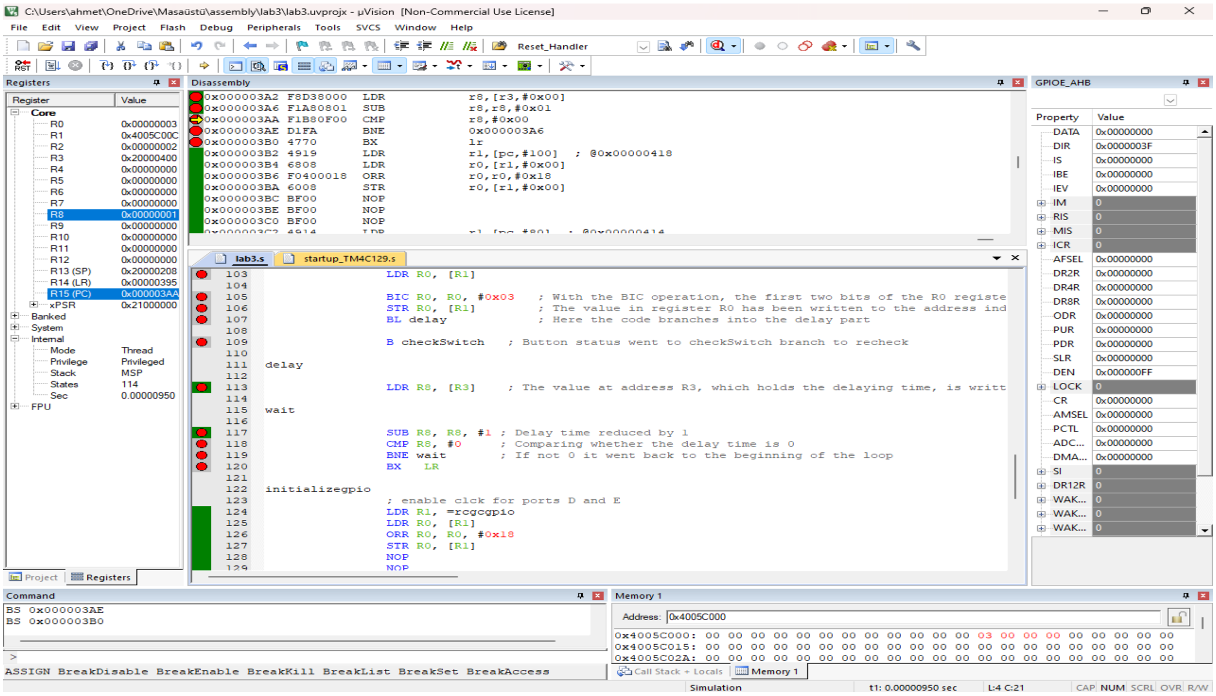

Figure 10

In figure 11 it became 0 and compared with the CMP command and then the code continued where it left off in the blink branch. As a result, while the led is on, it waited until the value of 2 became 0.

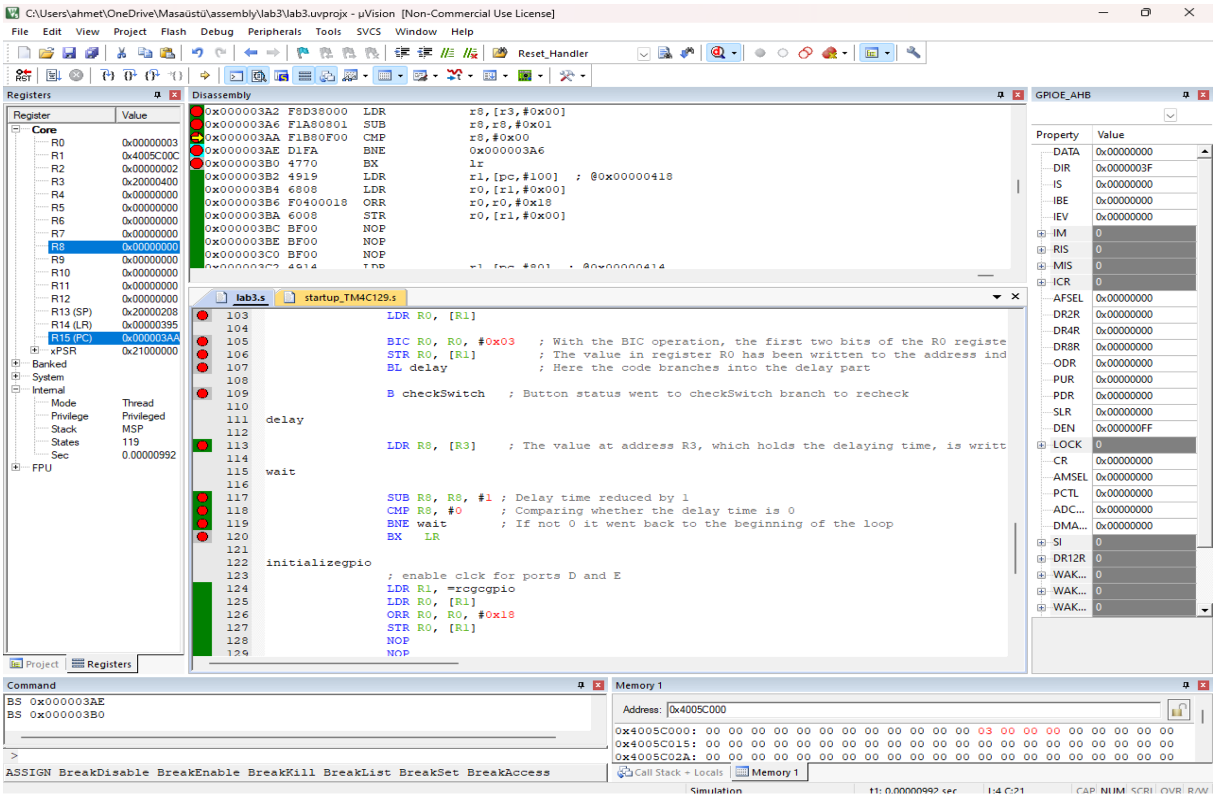

Figure 11

In Figure 12, the wait branch was exited and the leds were turned off. After that process was done, it went to the checkSwitch branch to check the button status again and the same operations continued.

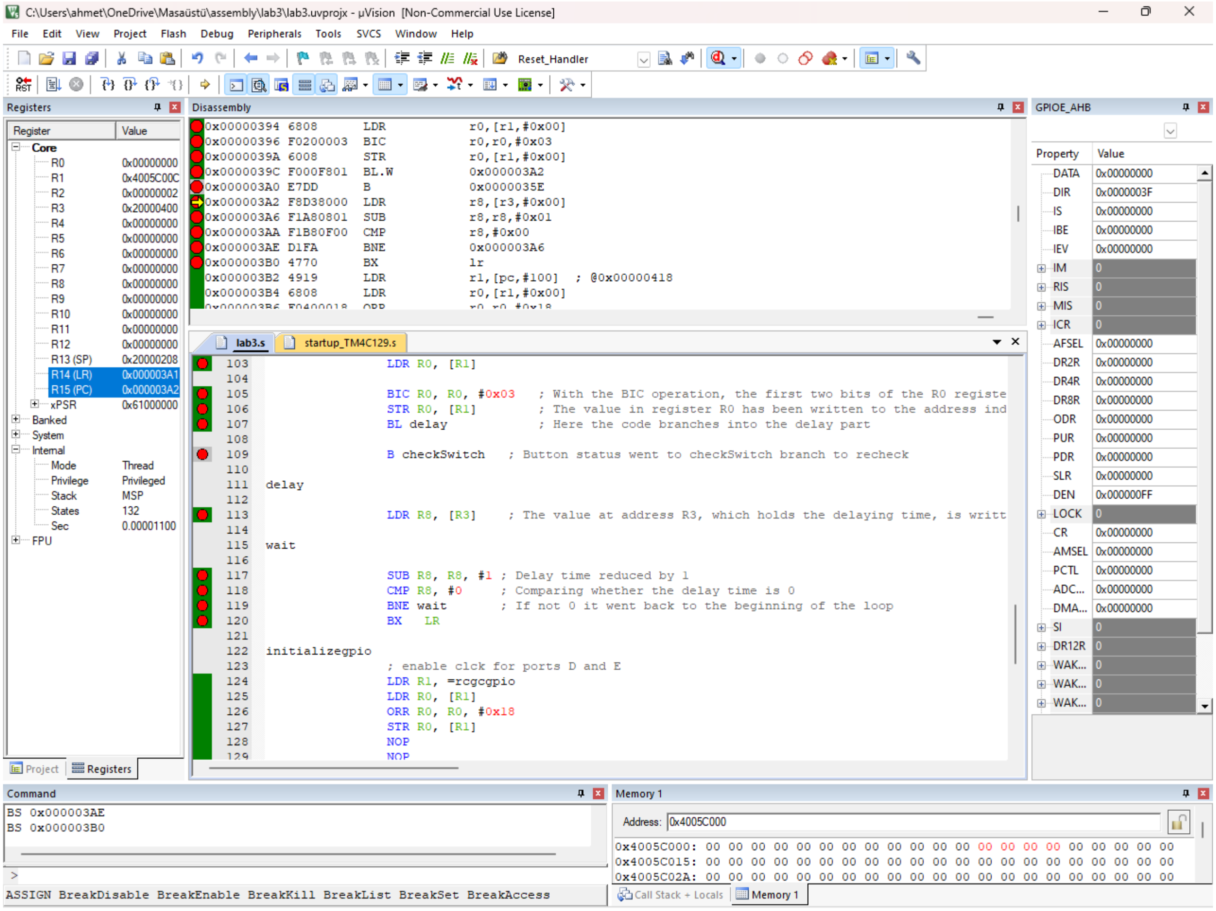

Figure 12

BLOCK DIAGRAM