GAZI UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING

Experiment #4

Ahmet Emin Karakaya
191110046

First of all, I would like to state that I have explained what I added to the base state of the code.

In Figure 1, since we are using port H in this code and we will do some operations from its base address, I defined the base address of port H.

```
1  ; GPIO base addresses of port D,E and H
2  gpioDbase   EQU     0x4005B000 ; inp
3  gpioEbase   EQU     0x4005C000 ; out
4  gpioHbase   EQU     0x4005F000 ; inp
```

Figure 1

In Figure 2, I made some definitions. Let me explain these definitions as follows. The definition I call delay_time is the definition I wrote to observe in simulation. If I had to give an example of the purpose of this, when I brought the joystick to the up position, the leds would increase by burning binary. I wrote to observe this increase. delay_real is the delay time when I tried it on the card. If game_time, each game will have a duration. So I needed to increment a variable until I got to that duration. I used this definition for him. game_total_time actually specifies the duration of a game. When game_time reaches game_total_time, the game will be over. In order to actually observe the game_total_time value, a value must be given accordingly. Also in the simulation, I used values such as 3 and 5 to observe this value more easily.

```
50  rcgcgpio     EQU     0x400FE608
51  delay_time   EQU     4
52  delay_real   EQU     0X0C0000
53  game_time    EQU     0x0
54  game_total_time EQU 20
55
```

Figure 2

In Figure 3, I assigned the variables I defined to the registers (R4,R5). Then I read from the button on the D port. If he didn't press the button, I sent the game directly to the finish_game branch without starting the game. So I finished the game from scratch. If the button was pressed, I went to the start_game branch to start the game. There is also a code that I bought a comment. I set it to I pressed the button to observe it in the simulation.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;In this branch, a playtime value is assigned to the R5 r
;Then, the button on D port 0 bit was read and if it was
;If the button was pressed, it went to the start_game bra
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

checkSwitch
                LDR R5, =game_total_time
                LDR R4, =game_time
                LDR R1, =gpioDbase
                ADD R1, #0x04
                LDR R0, [R1]                    ; data re
                ;MOV R0, #0x03                   ;Assigne

                CMP R0, #0
                BEQ finish_game
                BNE start_game

```

Figure 3

In Figure 4, I created a most general branch to start the game. Here I compared game_time with game_total_time. If the time played has reached the total time, I went to the finish_game branch to finish the game. If we still have time, I went to the restart_game branch and did a joystick status reading.

```
start_game
                CMP R4,R5
                BEQ finish_game
                BNE restart_game

```

Figure 4

In Figure 5, first of all, I increased the game_time value by 1 and showed that the time has actually started. Later, when I looked at the joystick usage from the datasheet, I saw the following. I found that I should use the H7 bit if I do it right, H6 if I do it left, H5 if I do it up, and H4 if I do it down. First of all, I read H7 and went to the on_all_leds branch to light all the leds if there is data in that bit. If there was no data, I continued the code and did an H6 read. I went to off_all_led branch to turn off all leds if there is data in H6. If the data is not here, I continued the code and read the H5. If there is data here, it went to the increasing_blink branch. If the data is not here, I have read H4. If there is data here I went to the decreasing_blink branch. If it is not here either, the joystick is in an empty state and I went to start_game to count the game time when it is idle. From there, it came back to the restart_game branch and I checked whether the joystick position has changed or not. I also did MOV assignment CMP operations in some parts of the code. The reason for this is to be able to observe in the simulation. (Since the joystick is pull-up, we will see the value 0 when we read data.)

H7 --> right

H6 -->left

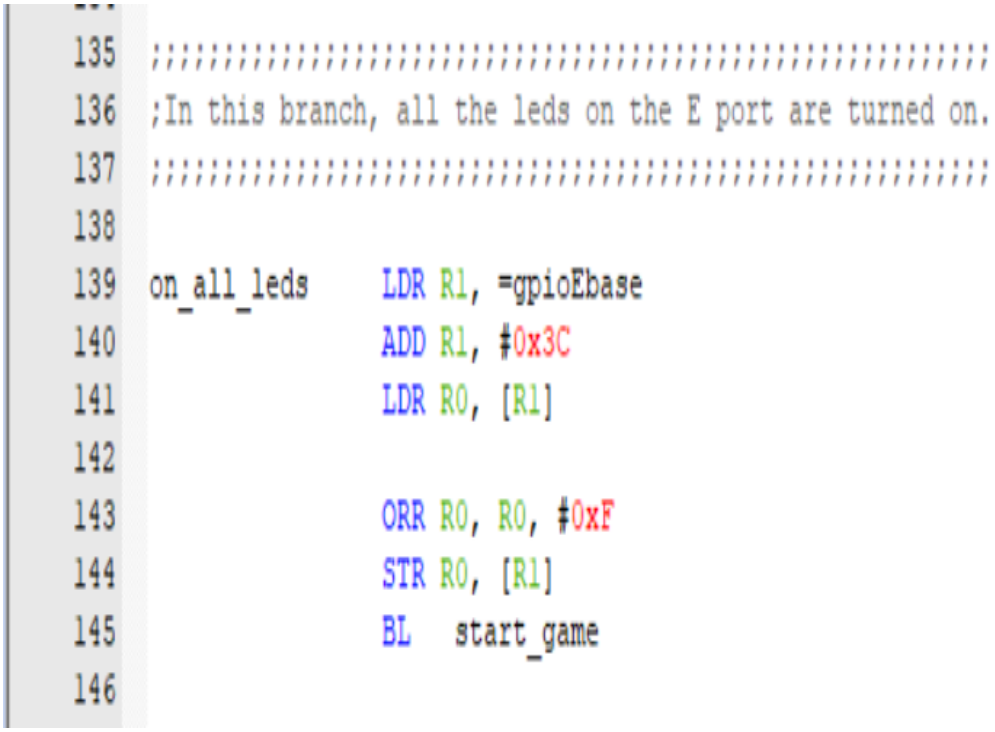
H5 -->up

H4 --> down

```
97  restart_game
98              ADD R4, R4, #1
99
100             LDR R1, =gpioHbase
101             ADD R1, #0x200          ; 7th bit control of H port
102             LDR R0, [R1]
103             ;MOV R11, #1            ;Assigned to debug simulation. In normal code this is not valid.
104
105             ;CMP R11, #1            ;Assigned to debug simulation. In normal code this is not valid.
106             CMP R0, #0              ;I compared it to 0 because the joystick pull-up is a button. If it was pull-down then we need to compare with 1.
107             BEQ on_all_leds
108
109             LDR R1, =gpioHbase
110             ADD R1, #0x100          ;6th bit control of H port
111             LDR R0, [R1]
112
113             ;CMP R11, #2            ;Assigned to debug simulation. In normal code this is not valid.
114             CMP R0, #0              ;I compared it to 0 because the joystick pull-up is a button. If it was pull-down then we need to compare with 1.
115             BEQ off_all_leds
116
117             LDR R1, =gpioHbase
118             ADD R1, #0x80           ;5th bit control of H port
119             LDR R0, [R1]
120
121             ;CMP R11, #3            ;Assigned to debug simulation. In normal code this is not valid.
122             CMP R0, #0              ;I compared it to 0 because the joystick pull-up is a button. If it was pull-down then we need to compare with 1.
123             BEQ increasing_blink
124
125             LDR R1, =gpioHbase
126             ADD R1, #0x40           ;4th bit control of H port
127             LDR R0, [R1]
128
129             ;CMP R11, #4            ;Assigned to debug simulation. In normal code this is not valid.
130             CMP R0, #0              ;I compared it to 0 because the joystick pull-up is a button. If it was pull-down then we need to compare with 1.
131             BEQ decreasing_blink
132
133             BL start_game           ;If the joystick is idle
134
```

Figure 5

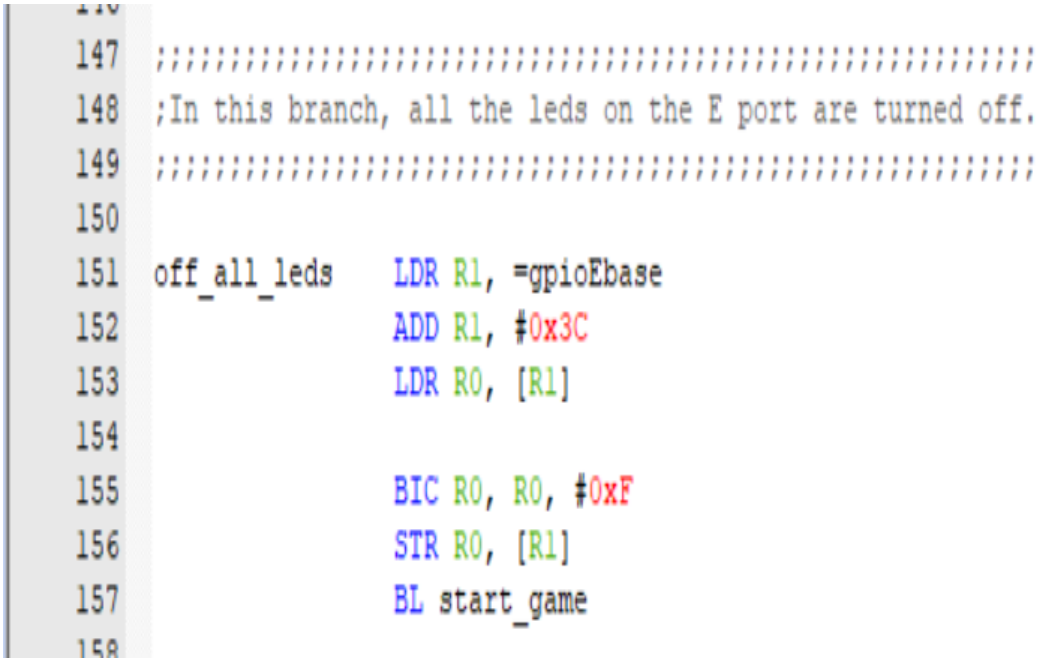
In Figure 6, if the joystick position is right, the code will come here and turn on all the leds as desired. In this branch, I made the process of turning on all the leds.



```
135 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
136 ;In this branch, all the leds on the E port are turned on.
137 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
138
139 on_all_leds     LDR R1, =gpioEbase
140                 ADD R1, #0x3C
141                 LDR R0, [R1]
142
143                 ORR R0, R0, #0xF
144                 STR R0, [R1]
145                 BL   start_game
146
```

Figure 6

In Figure 7, if the joystick position is left, the code will come here and turn off all the leds as desired. In this branch, I did the process of turning off all the leds.



```
147 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
148 ;In this branch, all the leds on the E port are turned off.
149 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
150
151 off_all_leds    LDR R1, =gpioEbase
152                 ADD R1, #0x3C
153                 LDR R0, [R1]
154
155                 BIC R0, R0, #0xF
156                 STR R0, [R1]
157                 BL start_game
158
```

Figure 7

In Figure 8, if the joystick position is up, the code will come here and turn on the leds by making binary increments as desired. So like 0001->0010->0011->0100->0101 .... At the beginning of the code, I did the following for this situation. Let's say we put the joystick up and all the leds are lit. Then the game time didn't end and the user made it up again. I sent it to the start_game branch if all the leds are on so that the increase does not continue over the last remainder. Then I increased the time. Also, I used the following method while increasing the LEDs binary. I did 0001 at first, I gave it a waiting time to see it with the eyes. Then I checked if all the leds were lit, if they were lit, it went to the start_game branch. If it hasn't turned on yet, it entered the invreasing_blink branch and turned on the other 0010 value.

```
159 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
160 ;In this branch, the LEDs in the E port are turned on using binary increments.
161 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
162
163 increasing_blink
164                 LDR R1, =gpioEbase
165                 ADD R1, #0x3C
166                 LDR R0, [R1]
167
168                 CMP R0, #0x0F
169                 BEQ start_game
170
171                 ADD R0, R0, #0x01
172                 STR R0, [R1]
173                 BL delay
174                 CMP R0, #0x0F
175                 BEQ start_game
176                 BNE increasing_blink
177
```

Figure 8

In Figure 9, if the joystick position is down, the code will come here and will turn off the leds by making binary reduction as desired. Actually, I did the extinguishing process of the event I described in figure 8.

```
178 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
179 ;In this branch, the LEDs in the E port are turned off using binary increments.
180 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
181
182 decreasing_blink
183
184                 LDR R1, =gpioEbase
185                 ADD R1, #0x3C
186                 LDR R0, [R1]
187                 CMP R0, #0x0
188                 BEQ start_game
189
190                 SUB R0, R0, #0x01
191                 STR R0, [R1]
192                 BL delay
193                 CMP R0, #0x0
194                 BEQ start_game
195                 BNE decreasing_blink
196
```

Figure 9

In Figure 10, there were some places in the code where I had to wait. That's why I used these two branches.

```
delay           LDR R3,=delay_real ;The durations between LEDs turn on are assigned to the R3 register.
                ;LDR R3,=delay_time  ;Assigned to debug simulation. In normal code this is not valid.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;During the delay time given in this branch, the code returns here as much as the delay time.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

wait            SUB R3, R3, #1
                CMP R3, #0
                BNE wait
                BX   LR

```

Figure 10

In Figure 11, firstly I reassigned the game_time value, which holds the game counter time, to 0. Because the game is not just a one-time game. I then gave it a cooldown until I started the other game. After the wait process was finished, I cleared all the leds and went to the checkSwitch branch to check the button on the D port.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;This branch will run when the game time is over or if the key on port
;Here, it will wait until the new game starts with the last state of t
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

finish_game
                LDR R4,=game_time
                BL delay
                LDR R1, =gpioEbase
                ADD R1, #0x3C
                LDR R0, [R1]

                BIC R0, R0, #0xF
                STR R0, [R1]
                B   checkSwitch   ;When the game was over, the code we


```

Figure 11

Examining the outputs of the code with simulation

The example I'm going to show is the one when we put the joystick up.

In Figure 12, necessary assignments are made to the registers R4 and R5. I did R0 1 to show that the D button was pressed and it went to the start_game branch.

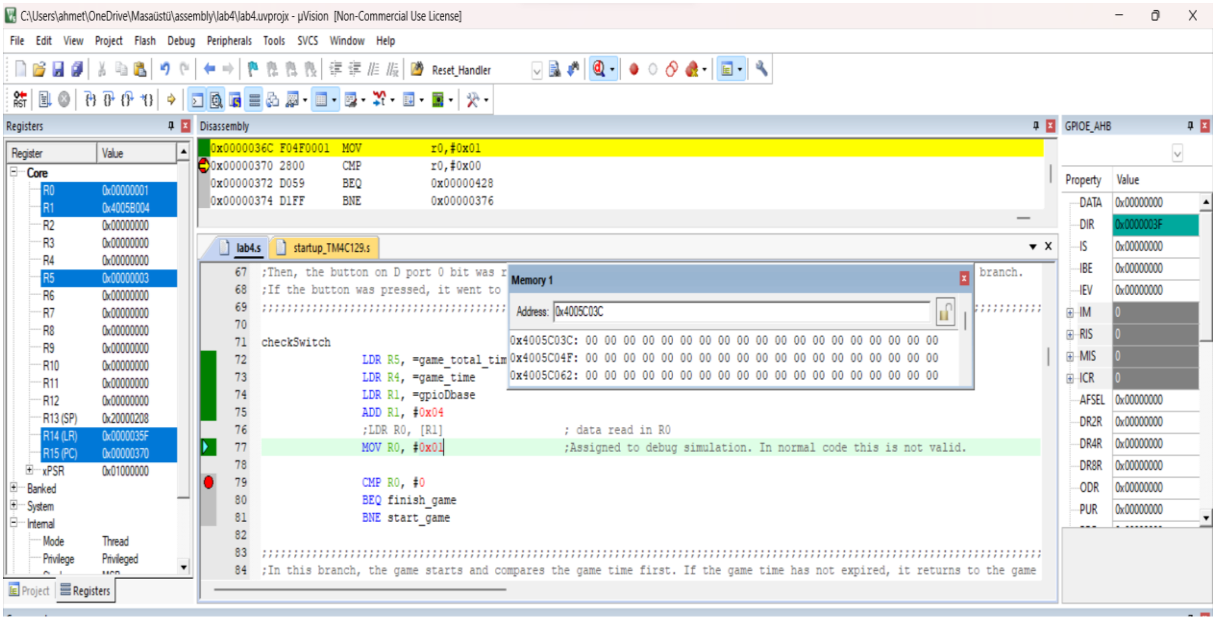


Figure 12

In Figure 13, I made 3 in register R11. So I showed that the joystick went to the up position. Also game_time became 1. (Also, the compared value in line 106 has been changed to 0. It is 0 in the discarded project file. Because I realized later that the joystick is pull-up resistant and fixed it.)

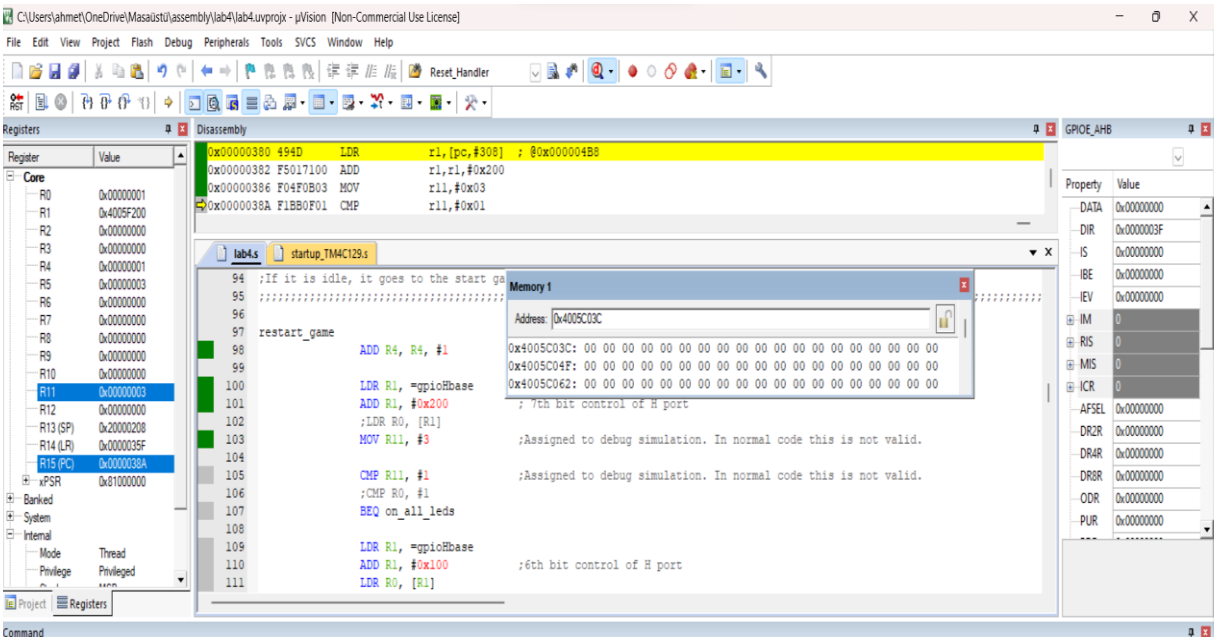


Figure 13

In Figure 14, it entered the increasing_blink branch and checked for 0xf. Since there is no 0XF at the moment, it wrote 0x01 to that address.

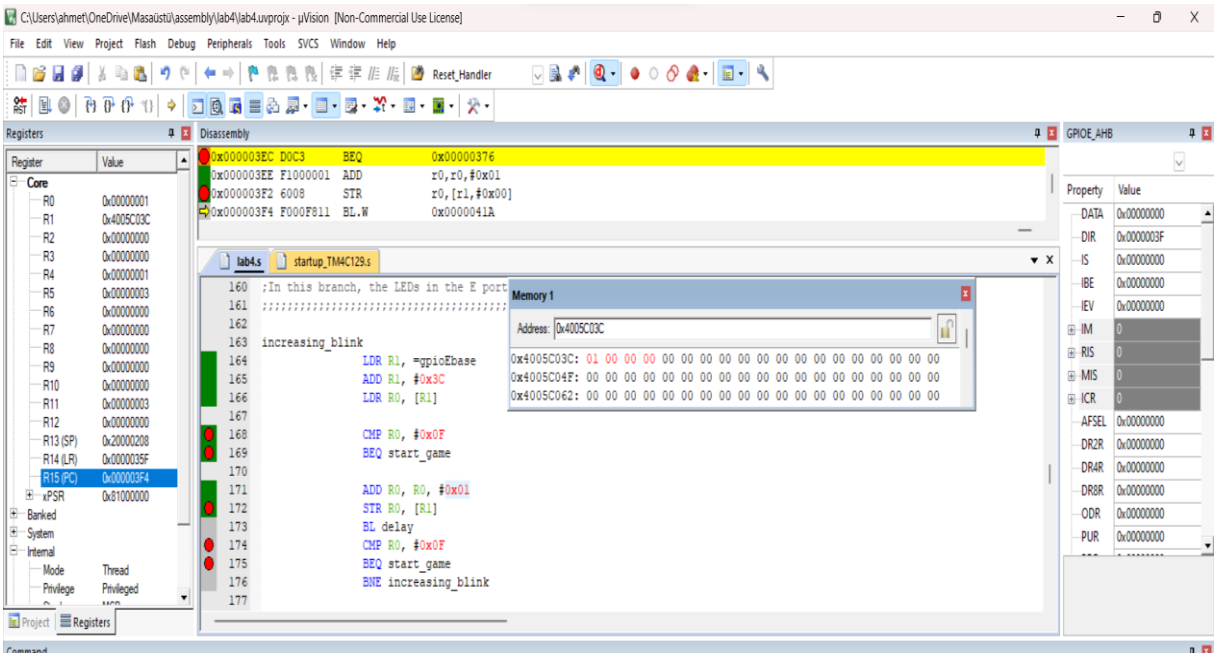


Figure 14

In figure 15, the code went to delay. The code waited until R3 became 4 and 0. Then re-entered increasing_blink.

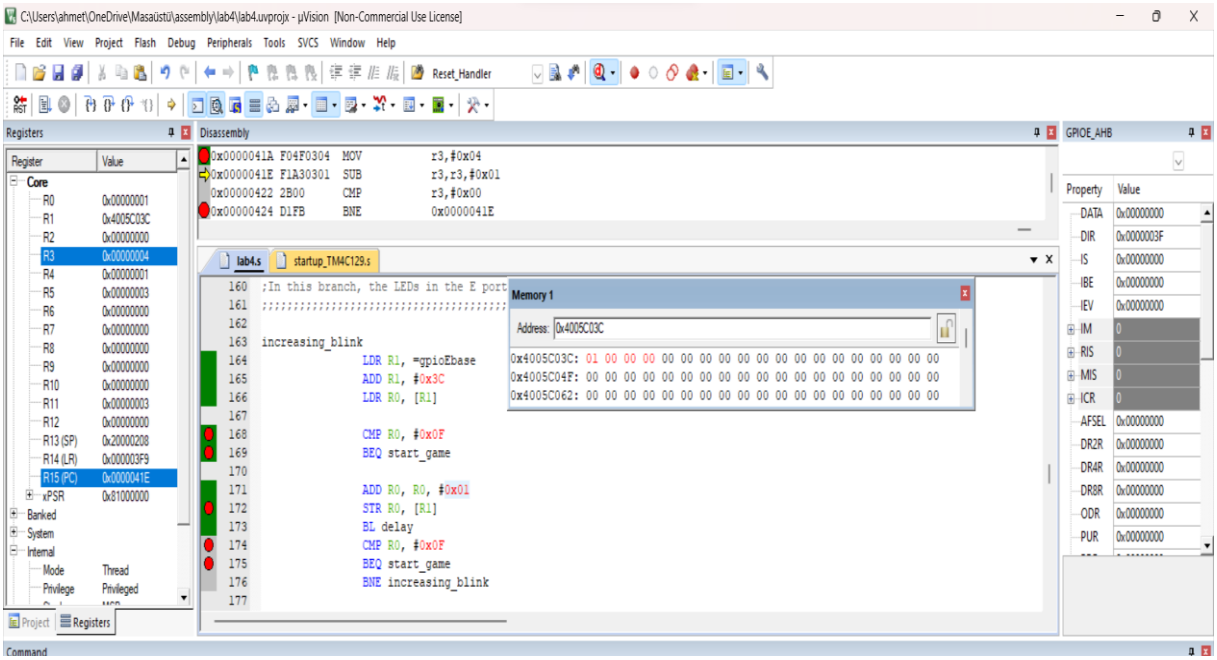


Figure 15

In Figure 16, the code came to increasing_blink. R0 did 2 and wrote it.

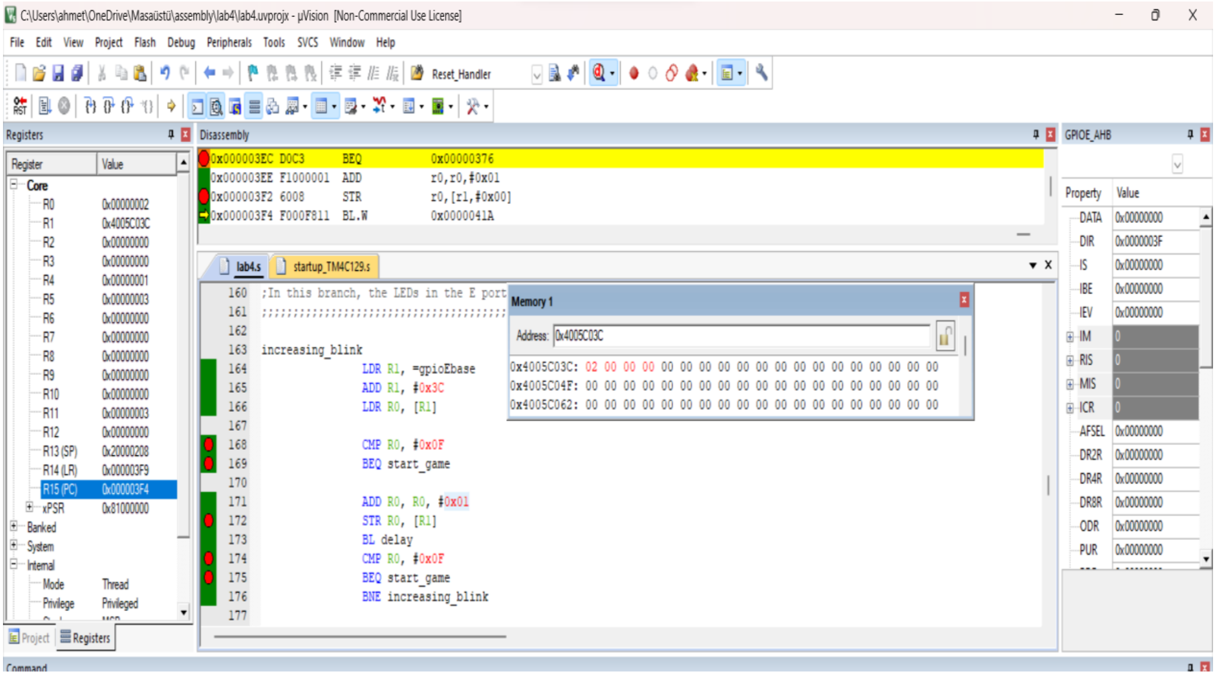


Figure 16

The code wrote in figure 17, then 3. In other words, the LEDs in the form of 0011 turned on.

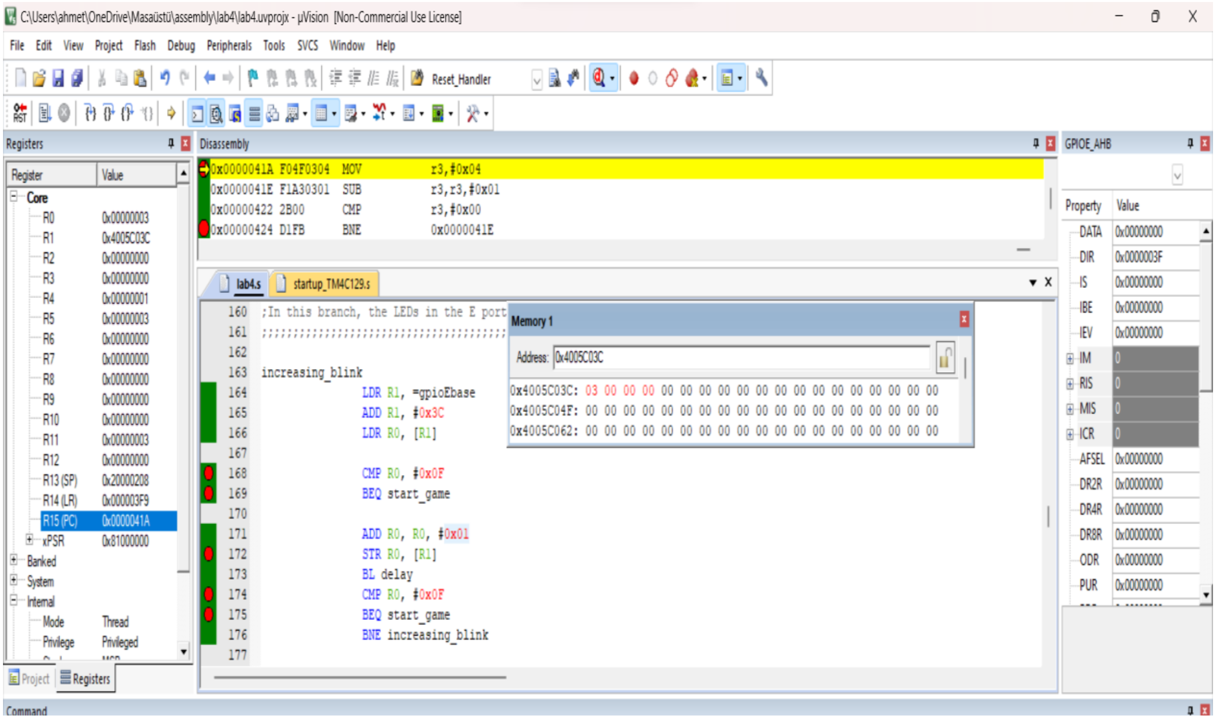


Figure 17

The code wrote in figure 18, then 4. In other words, the LEDs in the form of 0100 turned on.

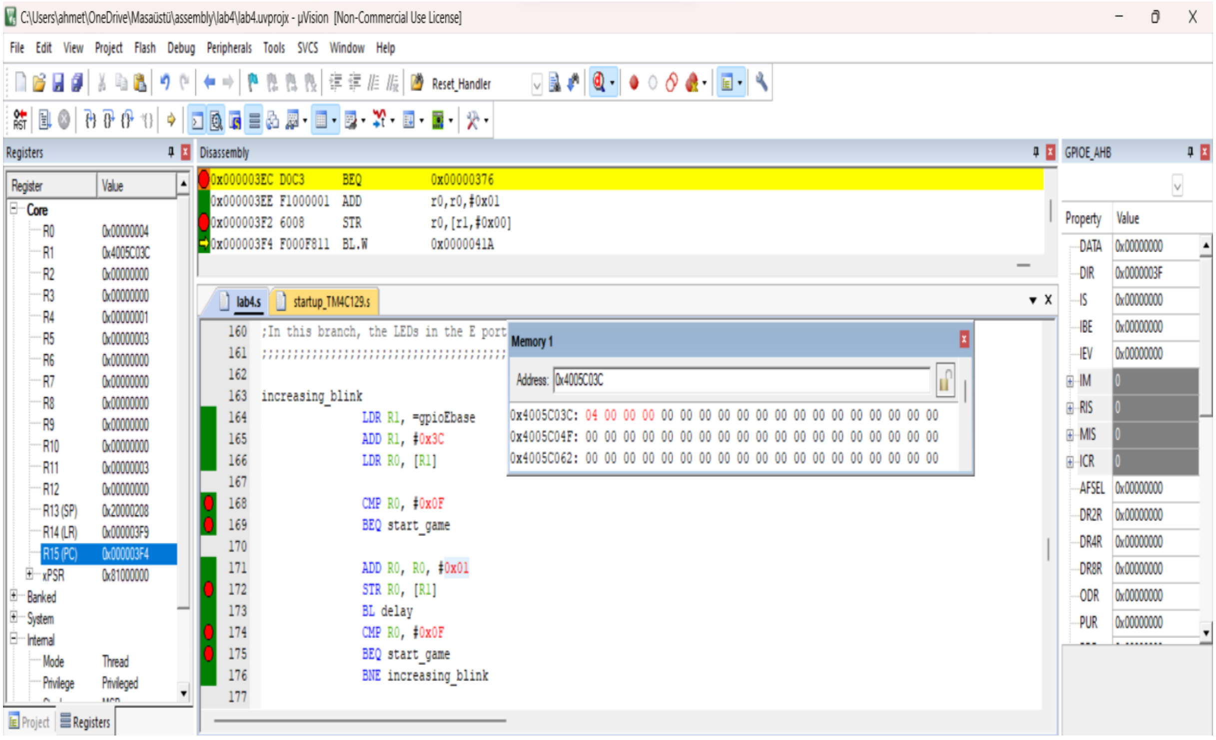


Figure 18

In Figure 19, the code increased in binary like this to 0x0F.

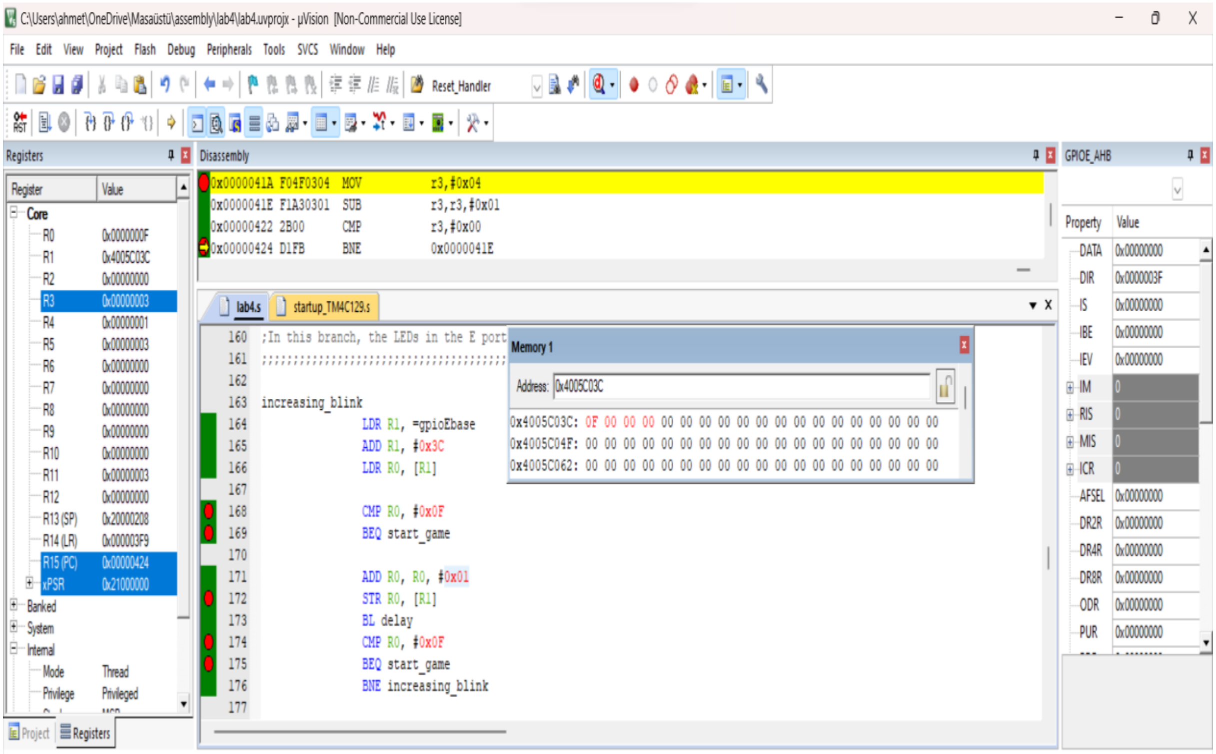


Figure 19

In figure 20, the code exited increasing_blink and came to the start_game branch. The code made the necessary checks and game_time is 2.

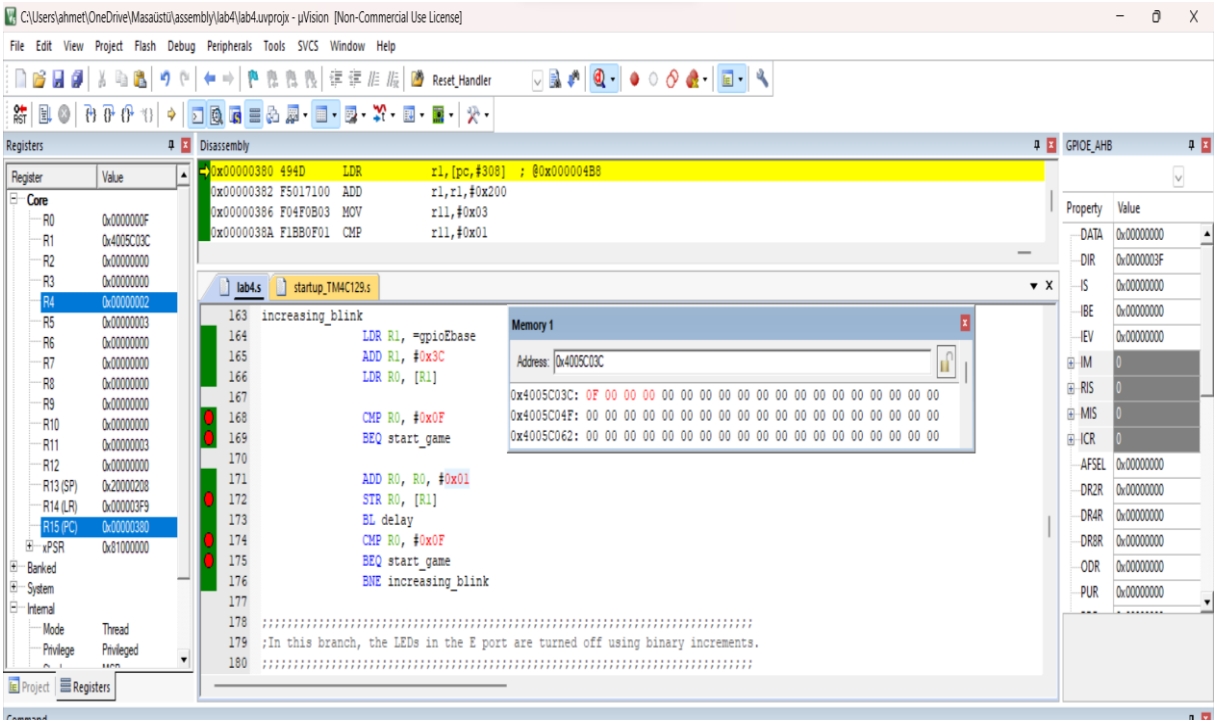


Figure 20

In Figure 21, the code made the first led check as all leds were lit and start_game was gone immediately without the increment. (Also, the compared value in lines 106 and 114 has been changed to 0. It is 0 in the discarded project file. Because I realized later that the joystick is pull-up resistant and fixed it.)

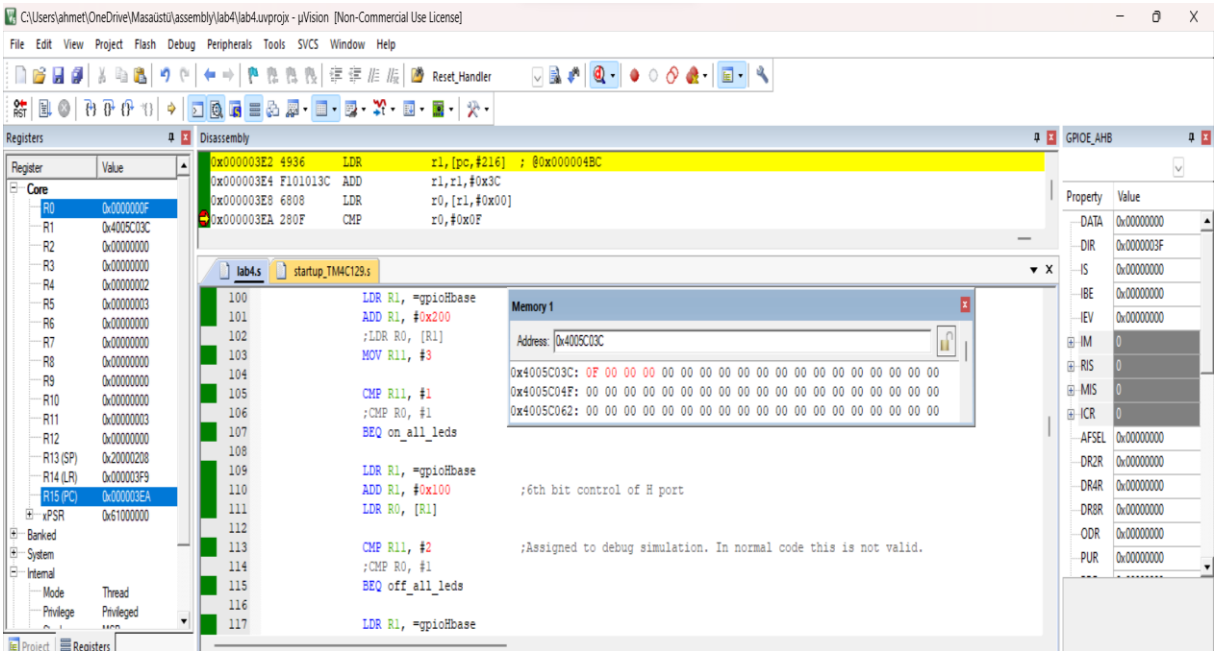


Figure 21

In Figure 22, the code became game_time 3 and the same action was taken. Also, the compared value in lines 106 and 114 has been changed to 0. It is 0 in the discarded project file. Because I realized later that the joystick is pull-up resistant and fixed it.

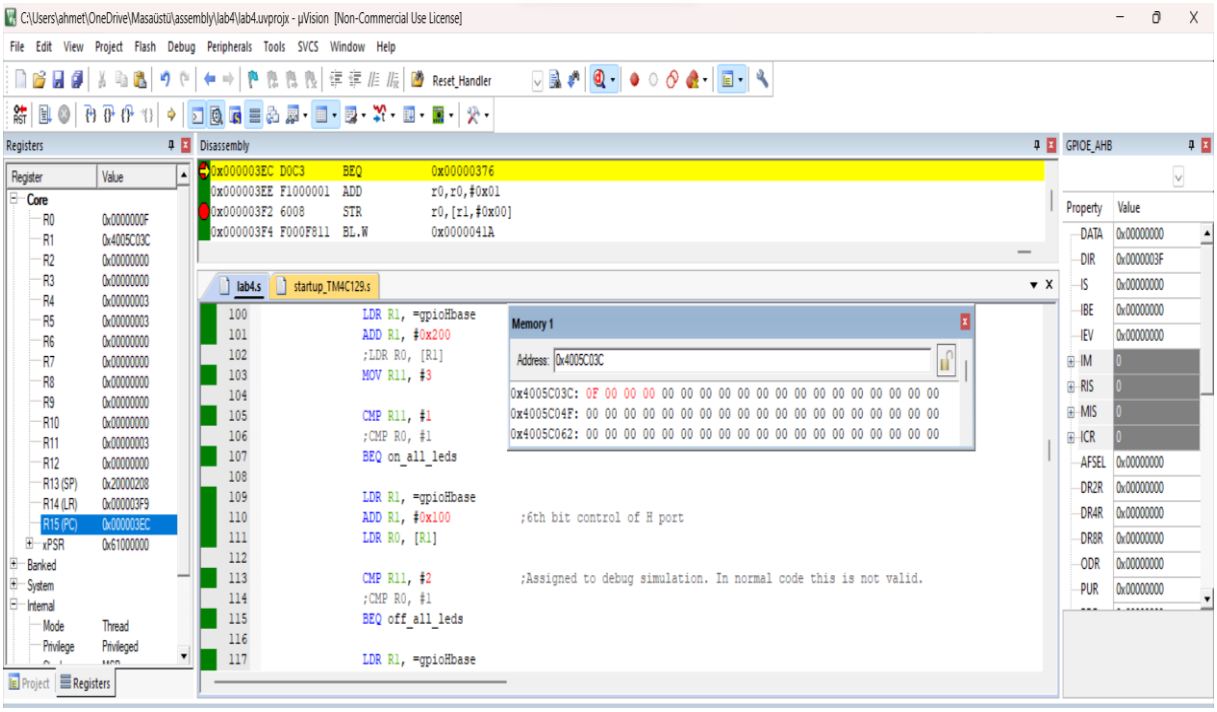


Figure 22

In Figure 23, the game time is over and went to the finish_game branch. game_time is 0.

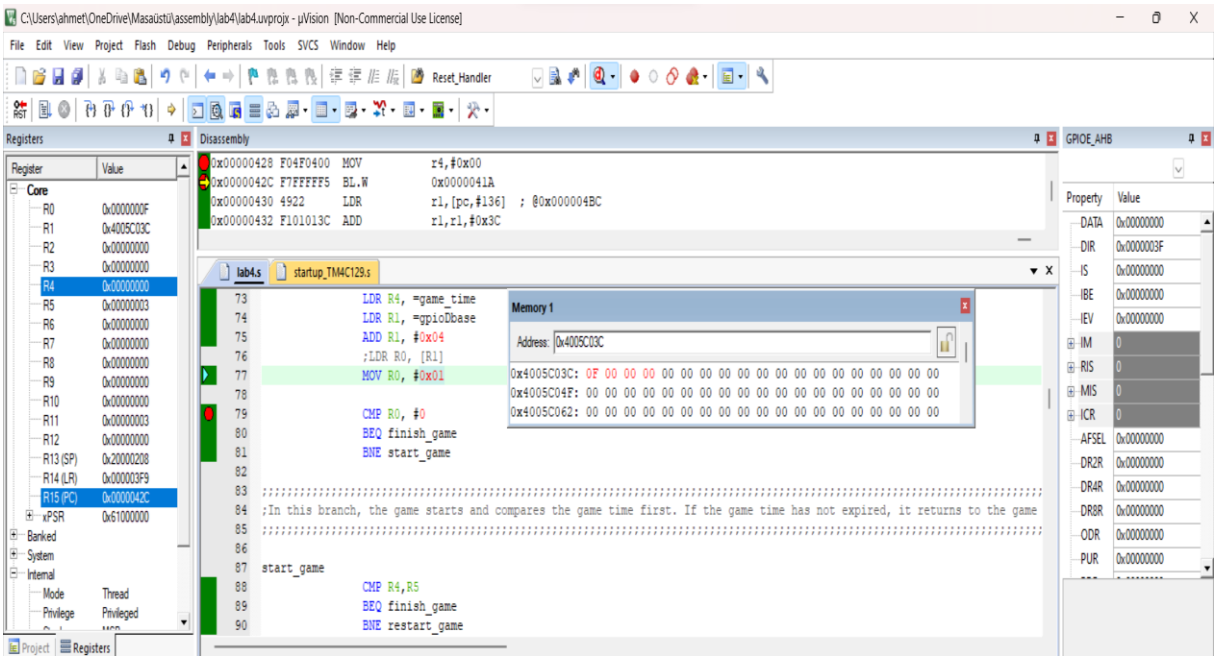


Figure 23

In figure 24, the code waited in the delay branch.

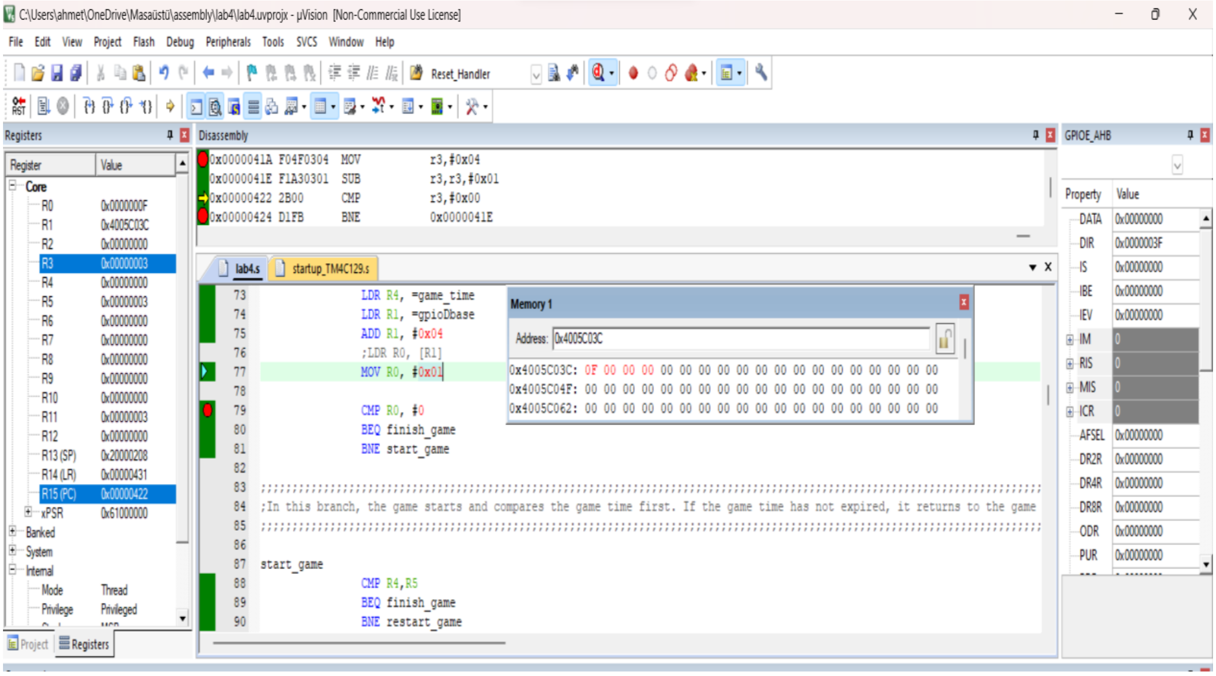


Figure 24

In Figure 25, I cleared the leds after the code came from the wait branch.

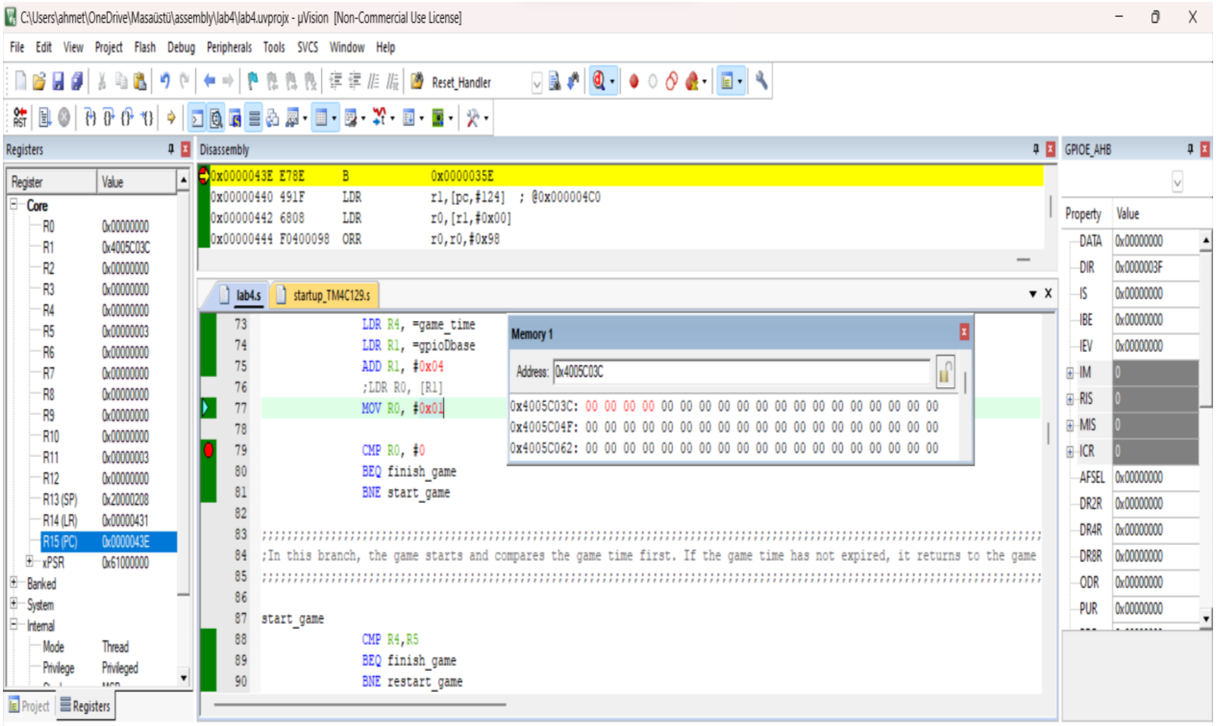


Figure 25

Now, I made the joystick right and observed in the simulation that all the leds were on.

In Figure 26, register R11 became 1, which means there is data in bit H7. This means that the code will go to the on_all_leds branch. (Also, the compared value in line 130 has been changed to 0. It is 0 in the discarded project file. Because I realized later that the joystick is pull-up resistant and fixed it.)

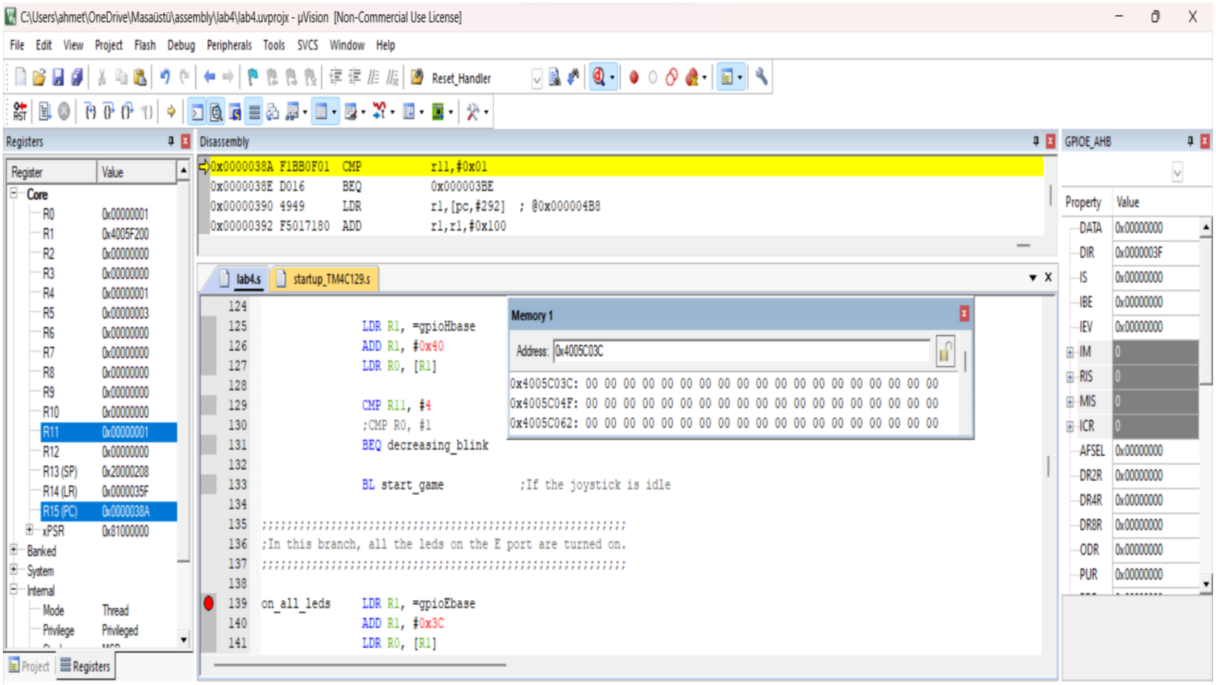


Figure 26

In figure 27, the code went to the on_all_leds branch and turned on all the leds.

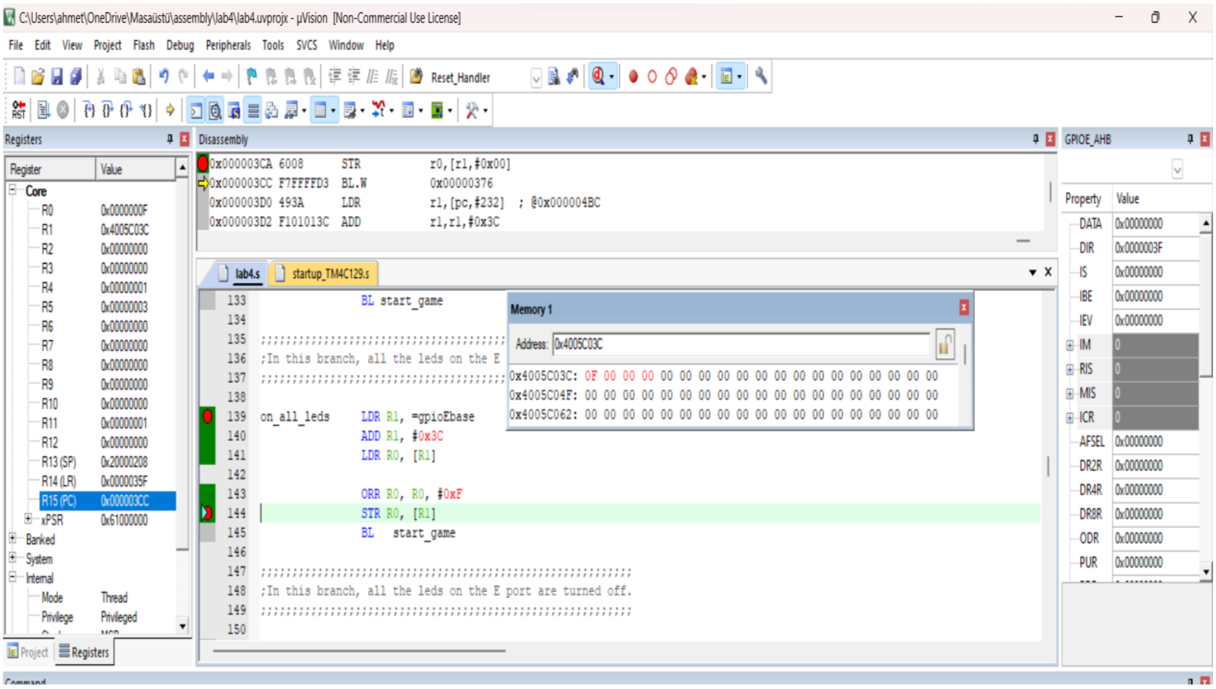


Figure 27

In figure 28, game_time became 2.

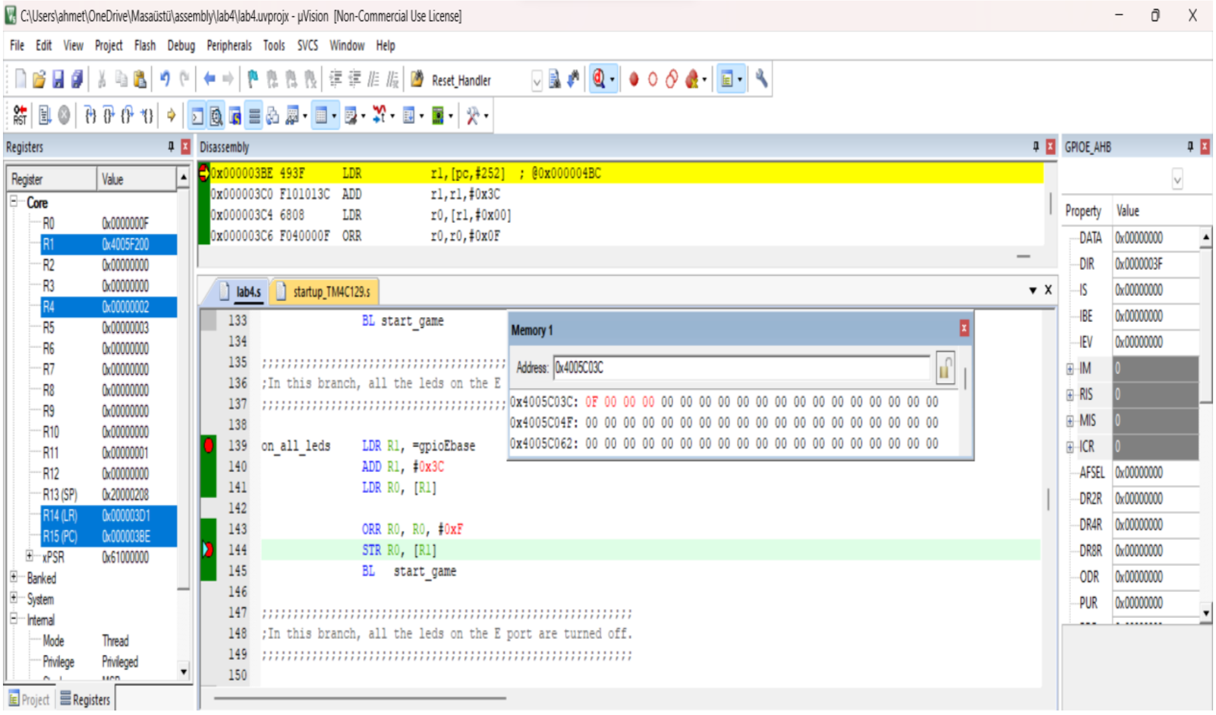


Figure 28

In figure 29, game_time became 3.

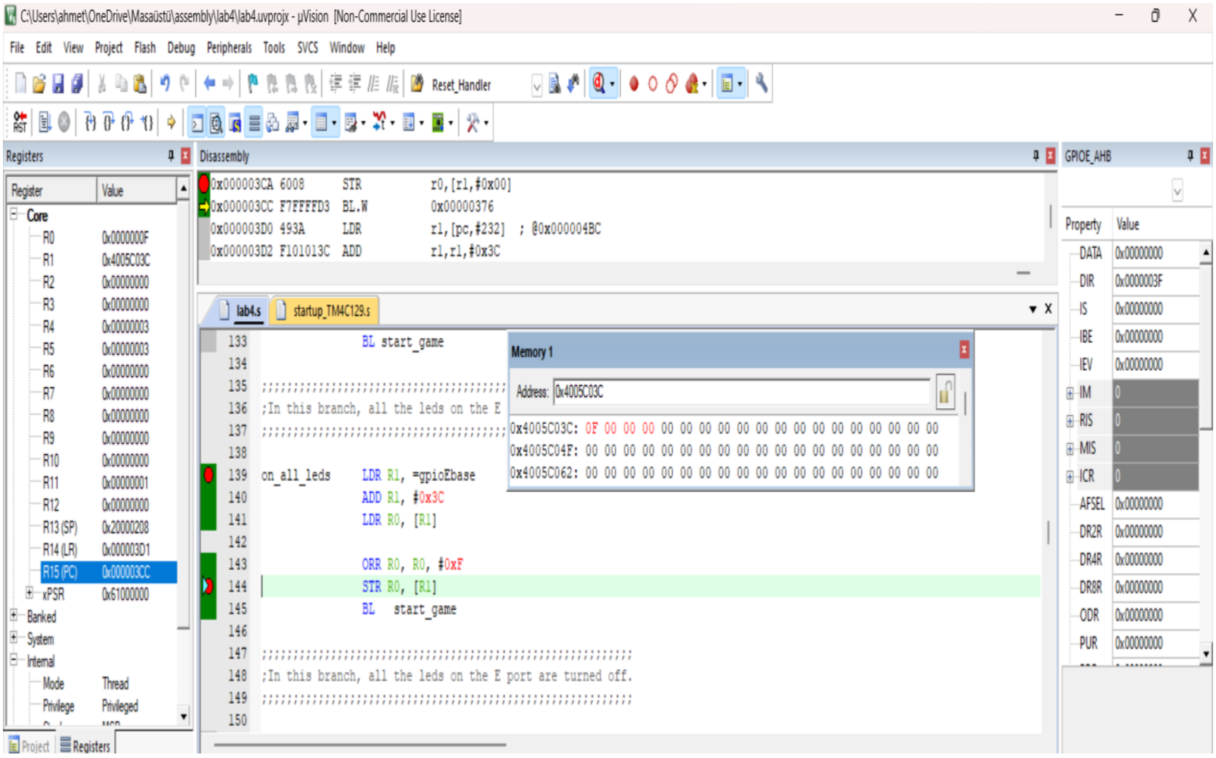


Figure 29

In figure 30, the game time is over and went to the finish_game branch. The code turned off all the leds.

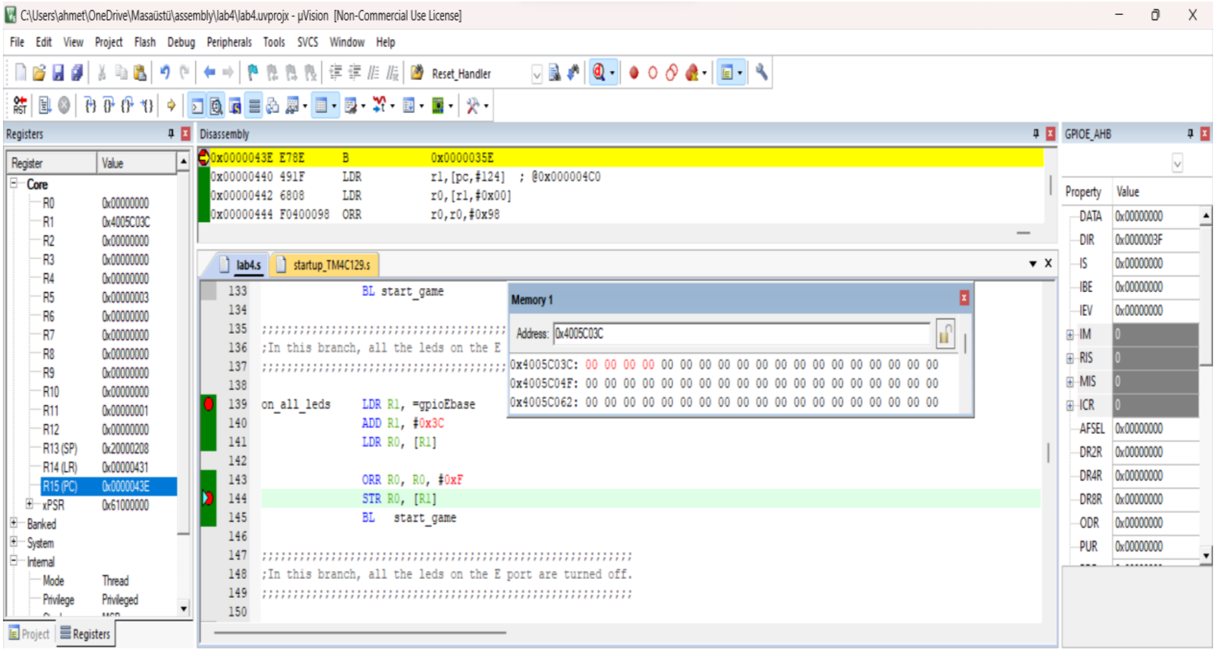


Figure 30