

Assignment 1:

```
.global init
.equ  GPIO_ENABLE_REG, 0x60004020 # Thanh ghi cấu hình vào/ra các chân
GPIO
.equ  GPIO_OUT_W1TS_REG, 0x60004008 # Thanh ghi thiết lập chân GPIO

init:
    # Thiết lập GPIO2 là Output
    li  a1, GPIO_ENABLE_REG
    li  a2, 0x04             # Mặt nạ cho GPIO2 (bit 2)
    sw  a2, 0(a1)

    # Thiết lập GPIO3 là Output
    li  a1, GPIO_ENABLE_REG
    li  a2, 0x08             # Mặt nạ cho GPIO3 (bit 3)
    sw  a2, 0(a1)

    # Thiết lập GPIO4 là Output
    li  a1, GPIO_ENABLE_REG
    li  a2, 0x10             # Mặt nạ cho GPIO4 (bit 4)
    sw  a2, 0(a1)

    # Bật LED trên GPIO2
    li  a1, GPIO_OUT_W1TS_REG
    li  a2, 0x04             # Mặt nạ cho GPIO2
    sw  a2, 0(a1)

    # Bật LED trên GPIO3
    li  a1, GPIO_OUT_W1TS_REG
    li  a2, 0x08             # Mặt nạ cho GPIO3
    sw  a2, 0(a1)

    # Bật LED trên GPIO4
    li  a1, GPIO_OUT_W1TS_REG
    li  a2, 0x10             # Mặt nạ cho GPIO4
    sw  a2, 0(a1)
```

Lưu ý: Tùy với mỗi chân thì sẽ chỉ lưu và chạy GPIO_ENABLE_REG và GPIO_OUT_W1TS_REG của chân đó.

Output:

<p>GPIO 2</p>	 <pre> .global init .equ GPIO_ENABLE_REG, 0x60004020 # Thanh ghi cấu hình vào/ra các chân GPIO .equ GPIO_OUT_WITS_REG, 0x60004008 # Thanh ghi thiết lập chân GPIO init: # Thiết lập GPIO2 là Output li a1, GPIO_ENABLE_REG li a2, 0x04 # Mặt nạ cho GPIO2 (bit 2) sw a2, 0(a1) # Bật LED trên GPIO2 li a1, GPIO_OUT_WITS_REG li a2, 0x04 # Mặt nạ cho GPIO2 sw a2, 0(a1) # Thiết lập GPIO3 là Output li a1, GPIO_ENABLE_REG li a2, 0x08 # Mặt nạ cho GPIO3 (bit 3) sw a2, 0(a1) # Bật LED trên GPIO3 li a1, GPIO_OUT_WITS_REG li a2, 0x08 # Mặt nạ cho GPIO3 sw a2, 0(a1) # Thiết lập GPIO4 là Output li a1, GPIO_ENABLE_REG li a2, 0x10 # Mặt nạ cho GPIO4 (bit 4) sw a2, 0(a1) # Bật LED trên GPIO4 li a1, GPIO_OUT_WITS_REG li a2, 0x10 # Mặt nạ cho GPIO4 sw a2, 0(a1) </pre> <p>mode:DIO, clock div:1 load:0x3fcd6100,len:0x420 load:0x403ce000,len:0x90c load:0x403d0000,len:0x2370 entry 0x403ce000 Hello, ESP32-C3! Hello, ESP32-C3!</p>
<p>GPIO 3</p>	 <pre> 1 .global init 2 .equ GPIO_ENABLE_REG, 0x60004020 # Thanh ghi cấu hình vào/ra các chân GPIO 3 .equ GPIO_OUT_WITS_REG, 0x60004008 # Thanh ghi thiết lập chân GPIO 4 5 init: 6 7 # Thiết lập GPIO3 là Output 8 li a1, GPIO_ENABLE_REG 9 li a2, 0x08 # Mặt nạ cho GPIO3 (bit 3) 10 sw a2, 0(a1) 11 # Bật LED trên GPIO3 12 li a1, GPIO_OUT_WITS_REG 13 li a2, 0x08 # Mặt nạ cho GPIO3 14 sw a2, 0(a1) 15 16 # Thiết lập GPIO4 là Output 17 li a1, GPIO_ENABLE_REG 18 li a2, 0x10 # Mặt nạ cho GPIO4 (bit 4) 19 sw a2, 0(a1) 20 # Bật LED trên GPIO4 21 li a1, GPIO_OUT_WITS_REG 22 li a2, 0x10 # Mặt nạ cho GPIO4 23 sw a2, 0(a1) 24 25 </pre> <p>mode:DIO, clock div:1 load:0x3fcd6100,len:0x420 load:0x403ce000,len:0x90c load:0x403d0000,len:0x2370 entry 0x403ce000 Hello, ESP32-C3! Hello, ESP32-C3!</p>
<p>GPIO 4</p>	 <pre> 1 .global init 2 .equ GPIO_ENABLE_REG, 0x60004020 # Thanh ghi cấu hình vào/ra các chân GPIO 3 .equ GPIO_OUT_WITS_REG, 0x60004008 # Thanh ghi thiết lập chân GPIO 4 5 init: 6 # Thiết lập GPIO2 là Output 7 li a1, GPIO_ENABLE_REG 8 li a2, 0x04 # Mặt nạ cho GPIO2 (bit 2) 9 sw a2, 0(a1) 10 11 # Thiết lập GPIO3 là Output 12 li a1, GPIO_ENABLE_REG 13 li a2, 0x08 # Mặt nạ cho GPIO3 (bit 3) 14 sw a2, 0(a1) 15 16 # thiết lập GPIO4 là output 17 li a1, GPIO_ENABLE_REG 18 li a2, 0x10 # Mặt nạ cho GPIO4 (bit 4) 19 sw a2, 0(a1) 20 21 # Bật LED trên GPIO2 22 li a1, GPIO_OUT_WITS_REG 23 li a2, 0x04 # Mặt nạ cho GPIO2 24 sw a2, 0(a1) 25 26 # Bật LED trên GPIO3 27 li a1, GPIO_OUT_WITS_REG 28 li a2, 0x08 # Mặt nạ cho GPIO3 29 sw a2, 0(a1) 30 31 # Bật LED trên GPIO4 32 li a1, GPIO_OUT_WITS_REG 33 li a2, 0x10 # Mặt nạ cho GPIO4 </pre> <p>mode:DIO, clock div:1 load:0x3fcd6100,len:0x420 load:0x403ce000,len:0x90c load:0x403d0000,len:0x2370 entry 0x403ce000 Hello, ESP32-C3! Hello, ESP32-C3!</p>

Assignment 2

Code GPIO2

```

.global init

.equv GPIO_ENABLE_REG, 0x60004020    # Thanh ghi cấu hình vào/ra các
chân GPIO
.equv GPIO_OUT_W1TS_REG, 0x60004008    # Thanh ghi thiết lập chân GPIO
.equv GPIO_OUT_W1TC_REG, 0x6000400C    # Thanh ghi xóa chân GPIO

.text

init:
    # Thiết lập GPIO2 là Output
    li a1, GPIO_ENABLE_REG
    li a2, 0x04                # Mặt nạ cho GPIO2 (bit 2)
    sw a2, 0(a1)

main_loop:
    # Bật LED trên GPIO2
    li a1, GPIO_OUT_W1TS_REG
    li a2, 0x04                # Mặt nạ cho GPIO2
    sw a2, 0(a1)

    # Gọi hàm delay
    call delay_asm

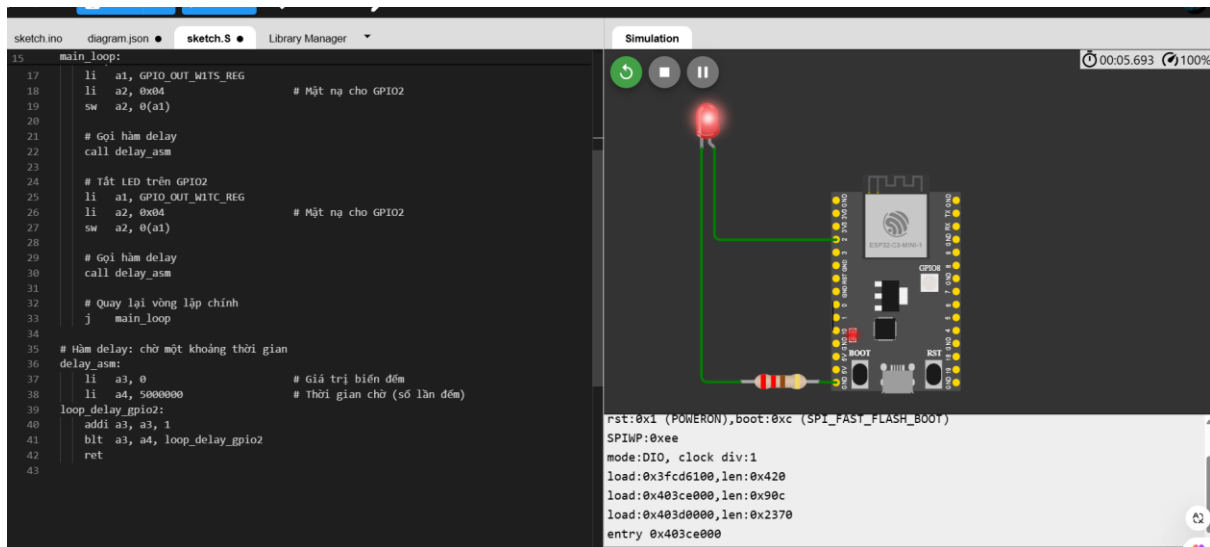
    # Tắt LED trên GPIO2
    li a1, GPIO_OUT_W1TC_REG
    li a2, 0x04                # Mặt nạ cho GPIO2
    sw a2, 0(a1)

    # Gọi hàm delay
    call delay_asm

    # Quay lại vòng lặp chính
    j main_loop

# Hàm delay: chờ một khoảng thời gian
delay_asm:
    li a3, 0                    # Giá trị biến đếm
    li a4, 10000000            # Thời gian chờ (số lần đếm)
loop_delay_gpio2:
    addi a3, a3, 1
    blt a3, a4, loop_delay_gpio2
    ret

```



Code GPIO3

```
.global init
```

```
.eqv GPIO_ENABLE_REG, 0x60004020    # Thanh ghi cấu hình vào/ra các chân GPIO
```

```
.eqv GPIO_OUT_W1TS_REG, 0x60004008    # Thanh ghi thiết lập chân GPIO
```

```
.eqv GPIO_OUT_W1TC_REG, 0x6000400C    # Thanh ghi xóa chân GPIO
```

```
.text
```

```
init:
```

```
    # Thiết lập GPIO3 là Output
```

```
    li a1, GPIO_ENABLE_REG
```

```
    li a2, 0x08           # Mặt nạ cho GPIO3 (bit 3)
```

```
    sw a2, 0(a1)
```

```
main_loop:
```

```
    # Bật LED trên GPIO3
```

```
    li a1, GPIO_OUT_W1TS_REG
```

```
    li a2, 0x08           # Mặt nạ cho GPIO3
```

```
    sw a2, 0(a1)
```

```
    # Gọi hàm delay
```

```
    call delay_asm
```

```
    # Tắt LED trên GPIO3
```

```
    li a1, GPIO_OUT_W1TC_REG
```

```
    li a2, 0x08           # Mặt nạ cho GPIO3
```

```
    sw a2, 0(a1)
```

```
    # Gọi hàm delay
```

```
call delay_asm
```

```
# Quay lại vòng lặp chính  
j    main_loop
```

```
# Hàm delay: chờ một khoảng thời gian
```

```
delay_asm:
```

```
    li a3, 0                # Giá trị biến đếm
```

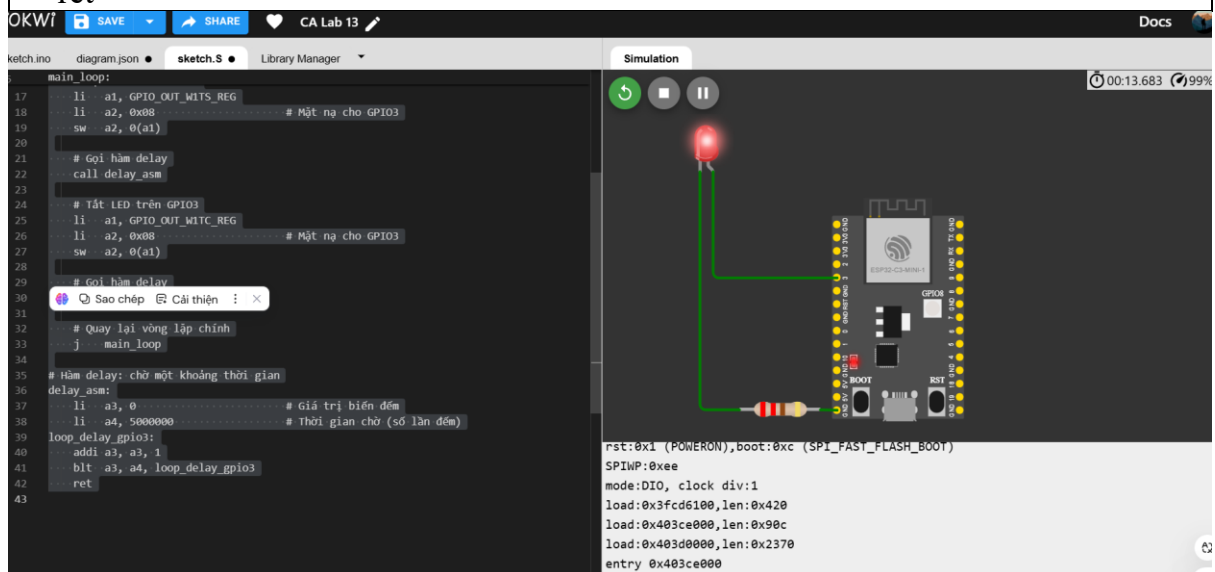
```
    li a4, 5000000          # Thời gian chờ (số lần đếm)
```

```
loop_delay_gpio3:
```

```
    addi a3, a3, 1
```

```
    blt a3, a4, loop_delay_gpio3
```

```
ret
```



Code GPIO4

```
.global init
```

```
.eqv GPIO_ENABLE_REG, 0x60004020    # Thanh ghi cấu hình vào/ra các  
chân GPIO
```

```
.eqv GPIO_OUT_W1TS_REG, 0x60004008    # Thanh ghi thiết lập chân GPIO
```

```
.eqv GPIO_OUT_W1TC_REG, 0x6000400C    # Thanh ghi xóa chân GPIO
```

```
.text
```

```
init:
```

```
    # Thiết lập GPIO4 là Output
```

```
    li a1, GPIO_ENABLE_REG
```

```
    li a2, 0x10                # Mặt nạ cho GPIO4 (bit 4)
```

```
    sw a2, 0(a1)
```

```
main_loop:
```

```
    # Bật LED trên GPIO4
```

```

li a1, GPIO_OUT_W1TS_REG
li a2, 0x10          # Mặt nạ cho GPIO4
sw a2, 0(a1)

# Gọi hàm delay
call delay_asm

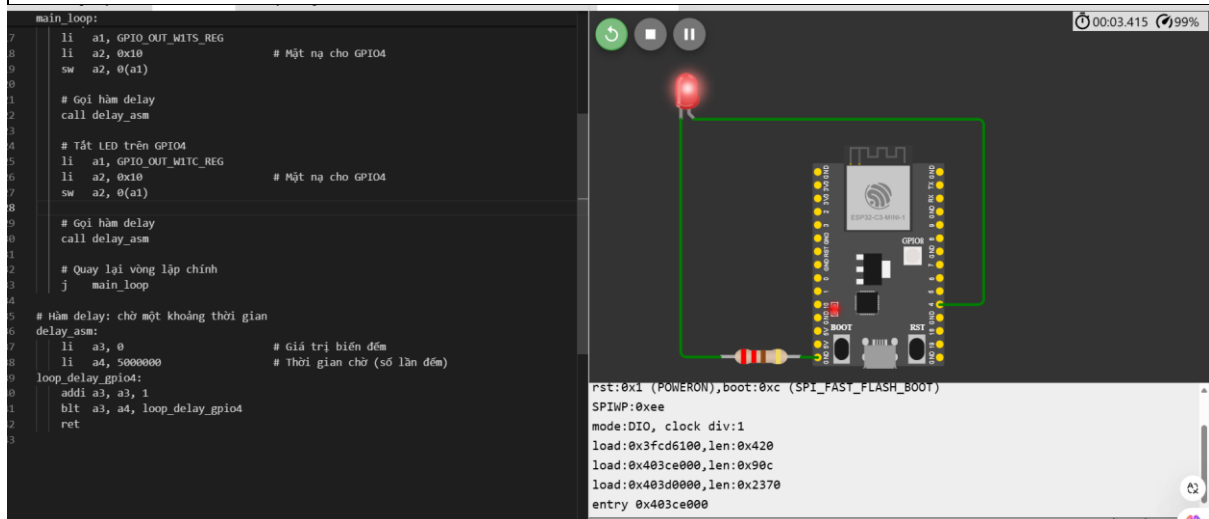
# Tắt LED trên GPIO4
li a1, GPIO_OUT_W1TC_REG
li a2, 0x10          # Mặt nạ cho GPIO4
sw a2, 0(a1)

# Gọi hàm delay
call delay_asm

# Quay lại vòng lặp chính
j main_loop

# Hàm delay: chờ một khoảng thời gian
delay_asm:
li a3, 0              # Giá trị biến đếm
li a4, 5000000        # Thời gian chờ (số lần đếm)
loop_delay_gpio4:
addi a3, a3, 1
blt a3, a4, loop_delay_gpio4
ret

```



Assignment 3:

Code:

```
.global init
```

```
# GPIO Register Definitions
```

```
.eqv GPIO_ENABLE_REG,    0x60004020 # GPIO output enable register
.eqv GPIO_OUT_REG,      0x60004004 # GPIO output register
.eqv IO_MUX_GPIO4_REG,  0x60009014 # GPIO4 function configuration
.eqv IO_MUX_GPIO5_REG,  0x60009018 # GPIO5 function configuration
.eqv IO_MUX_GPIO6_REG,  0x6000901C # GPIO6 function configuration
.eqv IO_MUX_GPIO7_REG,  0x60009020 # GPIO7 function configuration
```

7-segment display patterns for digits 0-9 (Common Anode - active low)

Segment mapping: GPIO0-6 -> segments a-g

.data

digits:

```
.word 0xC0 # 0: 1100 0000 - segments a,b,c,d,e,f on (0 = on)
.word 0xF9 # 1: 1111 1001 - segments b,c on
.word 0xA4 # 2: 1010 0100 - segments a,b,d,e,g on
.word 0xB0 # 3: 1011 0000 - segments a,b,c,d,g on
.word 0x99 # 4: 1001 1001 - segments b,c,f,g on
.word 0x92 # 5: 1001 0010 - segments a,c,d,f,g on
.word 0x82 # 6: 1000 0010 - segments a,c,d,e,f,g on
.word 0xF8 # 7: 1111 1000 - segments a,b,c on
.word 0x80 # 8: 1000 0000 - all segments on
.word 0x90 # 9: 1001 0000 - segments a,b,c,f,g on
```

.text

init:

```
# Configure GPIO0-7 as outputs
li a1, GPIO_ENABLE_REG
li a2, 0xFF          # Enable GPIO0-7 as outputs
sw a2, 0(a1)
```

```
# Configure GPIO4-7 function as GPIO
```

```
li a2, 0x1000        # Set GPIO function
li a1, IO_MUX_GPIO4_REG
sw a2, 0(a1)
li a1, IO_MUX_GPIO5_REG
sw a2, 0(a1)
li a1, IO_MUX_GPIO6_REG
sw a2, 0(a1)
li a1, IO_MUX_GPIO7_REG
sw a2, 0(a1)
```

display_loop:

```
la a3, digits        # Load address of digit patterns
li a4, 0              # Counter for digits 0-9
```

next_digit:

```
# Load and display the current digit pattern
```

```

lw a2, 0(a3)      # Load pattern for current digit
li a1, GPIO_OUT_REG
sw a2, 0(a1)      # Output pattern to GPIO

# Delay
call delay

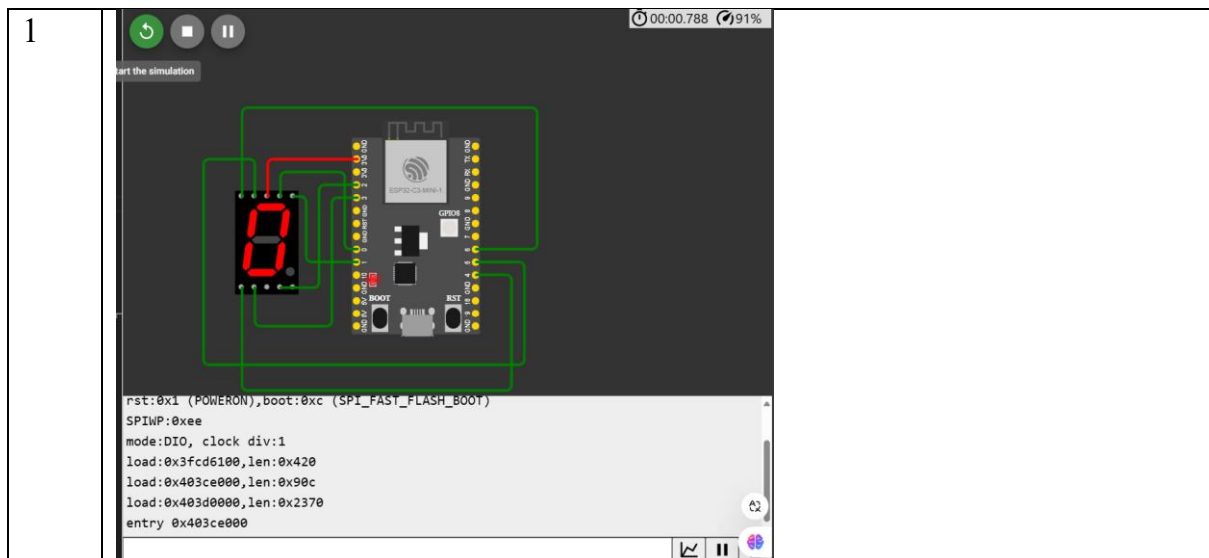
# Move to next digit
addi a3, a3, 4    # Next pattern address
addi a4, a4, 1    # Increment counter
li t0, 10
blt a4, t0, next_digit # If counter < 10, continue

j display_loop    # Repeat from 0

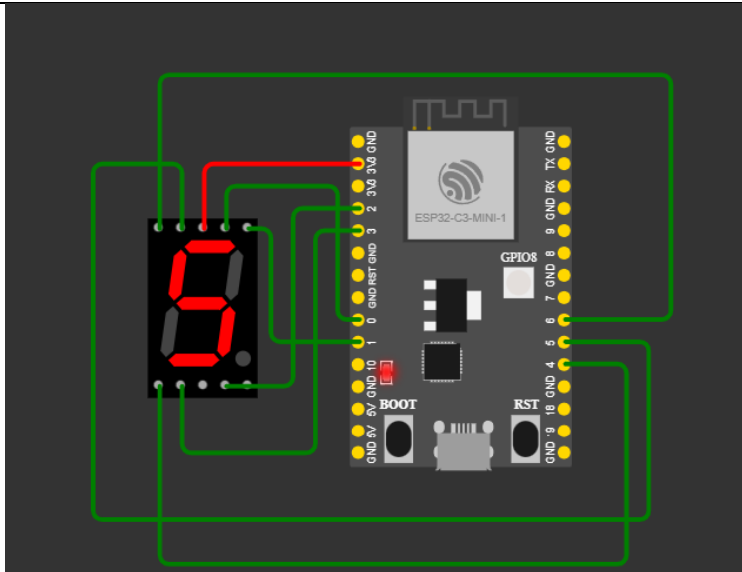
# Delay subroutine
delay:
li t0, 0          # Counter
li t1, 10000000   # Delay duration
delay_loop:
addi t0, t0, 1
blt t0, t1, delay_loop
ret

```

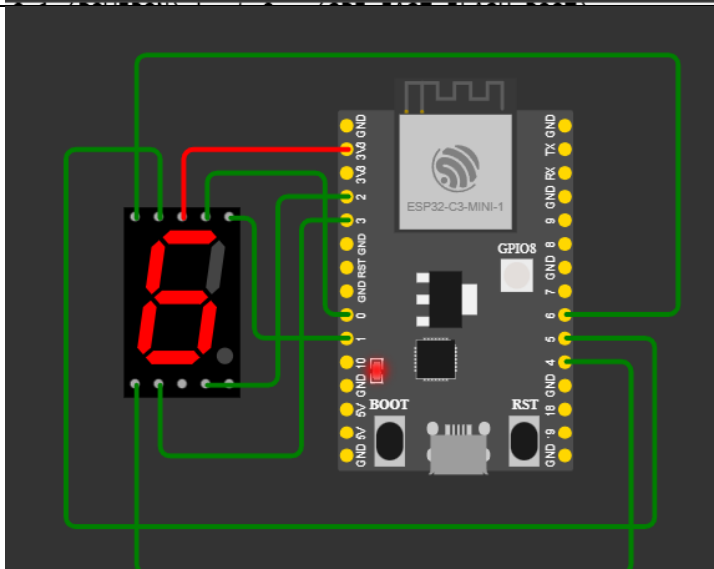
Output:



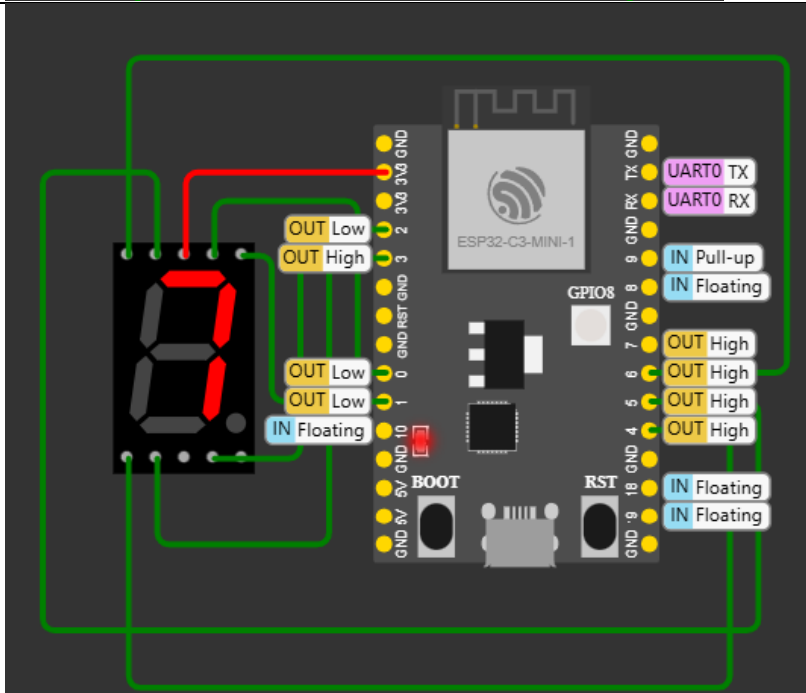
5



6



7




```

# Thiết lập GPIO1 là chân xuất tín hiệu (OUTPUT)
li a1, GPIO_ENABLE_REG    # Nạp địa chỉ thanh ghi GPIO_ENABLE_REG
vào a1
li a2, 0x02                # Giá trị 0x02 tương ứng với việc kích hoạt GPIO1
sw a2, 0(a1)               # Ghi giá trị 0x02 vào thanh ghi GPIO_ENABLE_REG

# Thiết lập GPIO0 là chân nhận tín hiệu (INPUT)
li a1, IO_MUX_GPIO0_REG    # Nạp địa chỉ thanh ghi IO_MUX_GPIO0_REG
vào a1
lw a2, 0(a1)               # Đọc giá trị hiện tại của thanh ghi IO_MUX_GPIO0_REG
ori a2, a2, 0x200          # Thiết lập bit IO_MUX_GPIO0_FUN_IE để cho phép
nhận tín hiệu
sw a2, 0(a1)               # Ghi lại giá trị mới vào IO_MUX_GPIO0_REG

# Thiết lập GPIO2 là chân nhận tín hiệu (INPUT)
#li a1, IO_MUX_GPIO2_REG    # Nạp địa chỉ thanh ghi IO_MUX_GPIO2_REG
vào a1
#lw a2, 0(a1)               # Đọc giá trị hiện tại của thanh ghi
IO_MUX_GPIO2_REG
#ori a2, a2, 0x200          # Thiết lập bit IO_MUX_GPIO2_FUN_IE để cho phép
nhận tín hiệu
#sw a2, 0(a1)               # Ghi lại giá trị mới vào IO_MUX_GPIO2_REG

# Thiết lập GPIO3 là chân nhận tín hiệu (INPUT)
#li a1, IO_MUX_GPIO3_REG    # Nạp địa chỉ thanh ghi IO_MUX_GPIO3_REG
vào a1
#lw a2, 0(a1)               # Đọc giá trị hiện tại của thanh ghi
IO_MUX_GPIO3_REG
#ori a2, a2, 0x200          # Thiết lập bit IO_MUX_GPIO3_FUN_IE để cho phép
nhận tín hiệu
#sw a2, 0(a1)               # Ghi lại giá trị mới vào IO_MUX_GPIO3_REG

# Thiết lập GPIO4 là chân nhận tín hiệu (INPUT)
#li a1, IO_MUX_GPIO4_REG    # Nạp địa chỉ thanh ghi IO_MUX_GPIO4_REG
vào a1
#lw a2, 0(a1)               # Đọc giá trị hiện tại của thanh ghi
IO_MUX_GPIO4_REG
#ori a2, a2, 0x200          # Thiết lập bit IO_MUX_GPIO4_FUN_IE để cho phép
nhận tín hiệu
#sw a2, 0(a1)               # Ghi lại giá trị mới vào IO_MUX_GPIO4_REG

# Vòng lặp chính
loop:
li a1, GPIO_IN_REG          # Nạp địa chỉ thanh ghi GPIO_IN_REG vào a1
lw a2, 0(a1)                # Đọc giá trị tín hiệu từ GPIO_IN_REG

```

```

# Kiểm tra GPIO0
andi a3, a2, 0x01      # Kiểm tra mức tín hiệu GPIO0
bne a3, zero, gpio0_set

# Kiểm tra GPIO2
#andi a3, a2, 0x04      # Kiểm tra bit 2 (GPIO2) trong giá trị đọc được
#bne a3, zero, gpio2_set  # Nếu bit 2 = 1 (GPIO2 HIGH), nhảy tới gpio2_set

# Kiểm tra GPIO3
#andi a3, a2, 0x08      # Kiểm tra bit 3 (GPIO3) trong giá trị đọc được
#bne a3, zero, gpio3_set  # Nếu bit 3 = 1 (GPIO3 HIGH), nhảy tới gpio3_set

# Kiểm tra GPIO4
#andi a3, a2, 0x10      # Kiểm tra bit 4 (GPIO4) trong giá trị đọc được
#bne a3, zero, gpio4_set  # Nếu bit 4 = 1 (GPIO4 HIGH), nhảy tới gpio4_set

clear:
# Tắt LED (GPIO1 = 0)
li a1, GPIO_OUT_W1TC_REG  # Nạp địa chỉ thanh ghi
GPIO_OUT_W1TC_REG
li a2, 0x02                # Giá trị 0x02 tương ứng với việc xóa GPIO1
sw a2, 0(a1)               # Ghi giá trị vào thanh ghi để tắt LED
j loop                    # Quay lại đầu vòng lặp

gpio0_set:
# Bật LED (GPIO1 = 1) nếu GPIO0 = HIGH
li a1, GPIO_OUT_W1TS_REG  # Nạp địa chỉ thanh ghi
GPIO_OUT_W1TS_REG
li a2, 0x02                # Giá trị 0x02 tương ứng với việc thiết lập GPIO1
sw a2, 0(a1)               # Ghi giá trị vào thanh ghi để bật LED
j loop                    # Nhảy tới nhãn loop

gpio2_set:
# Bật LED (GPIO1 = 1) nếu GPIO2 = HIGH
li a1, GPIO_OUT_W1TS_REG
li a2, 0x02
sw a2, 0(a1)
j loop

gpio3_set:
# Bật LED (GPIO1 = 1) nếu GPIO3 = HIGH
li a1, GPIO_OUT_W1TS_REG
li a2, 0x02
sw a2, 0(a1)
j loop

```

gpio4_set:

Bật LED (GPIO1 = 1) nếu GPIO4 = HIGH

li a1, GPIO_OUT_W1TS_REG

li a2, 0x02

sw a2, 0(a1)

j loop

Output:

The screenshot displays the Arduino IDE interface. On the left, the 'sketch.ino' file contains a C program for GPIO control. The program initializes GPIO registers (GPIO0, GPIO2, GPIO3, GPIO4) and enters a loop to check the state of GPIO2, GPIO3, and GPIO4, setting GPIO1 if any of them are high. On the right, the 'Simulation' window shows a breadboard circuit with an Arduino Uno and several LEDs connected to the GPIO pins. The simulation status bar at the bottom indicates the program is running with a time of 00:12.700 and 61% completion.

```
14  Init:
21  # Thiết lập GPIO0 là chân nhận tín hiệu (INPUT)
22  #li a1, IO_MUX_GPIO0_REG # Nạp địa chỉ thành ghi IO_MUX_GPIO0_REG vào a1
23  #lw a2, 0(a1) # Đọc giá trị hiện tại của thành ghi IO_MUX_GPIO0_REG
24  #ori a2, a2, 0x200 # Thiết lập bit IO_MUX_GPIO0_FUN_IE để cho phép nhận tín hiệu
25  #sw a2, 0(a1) # Ghi lại giá trị mới vào IO_MUX_GPIO0_REG
26
27  # Thiết lập GPIO2 là chân nhận tín hiệu (INPUT)
28  #li a1, IO_MUX_GPIO2_REG # Nạp địa chỉ thành ghi IO_MUX_GPIO2_REG vào a1
29  #lw a2, 0(a1) # Đọc giá trị hiện tại của thành ghi IO_MUX_GPIO2_REG
30  #ori a2, a2, 0x200 # Thiết lập bit IO_MUX_GPIO2_FUN_IE để cho phép nhận tín hiệu
31  #sw a2, 0(a1) # Ghi lại giá trị mới vào IO_MUX_GPIO2_REG
32
33  # Thiết lập GPIO3 là chân nhận tín hiệu (INPUT)
34  #li a1, IO_MUX_GPIO3_REG # Nạp địa chỉ thành ghi IO_MUX_GPIO3_REG vào a1
35  #lw a2, 0(a1) # Đọc giá trị hiện tại của thành ghi IO_MUX_GPIO3_REG
36  #ori a2, a2, 0x200 # Thiết lập bit IO_MUX_GPIO3_FUN_IE để cho phép nhận tín hiệu
37  #sw a2, 0(a1) # Ghi lại giá trị mới vào IO_MUX_GPIO3_REG
38
39  # Thiết lập GPIO4 là chân nhận tín hiệu (INPUT)
40  #li a1, IO_MUX_GPIO4_REG # Nạp địa chỉ thành ghi IO_MUX_GPIO4_REG vào a1
41  #lw a2, 0(a1) # Đọc giá trị hiện tại của thành ghi IO_MUX_GPIO4_REG
42  #ori a2, a2, 0x200 # Thiết lập bit IO_MUX_GPIO4_FUN_IE để cho phép nhận tín hiệu
43  #sw a2, 0(a1) # Ghi lại giá trị mới vào IO_MUX_GPIO4_REG
44
45  # Vòng lặp chính
46  loop:
47  #li a1, GPIO_IN_REG # Nạp địa chỉ thành ghi GPIO_IN_REG vào a1
48  #lw a2, 0(a1) # Đọc giá trị tín hiệu từ GPIO_IN_REG
49
50  # Kiểm tra GPIO2
51  #andi a3, a2, 0x04 # Kiểm tra bit 2 (GPIO2) trong giá trị đọc được
52  #bne a3, zero, gpio2_set # Nếu bit 2 = 1 (GPIO2 HIGH), nhảy tới 'gpio2_set'
53
54  # Kiểm tra GPIO3
55  #andi a3, a2, 0x08 # Kiểm tra bit 3 (GPIO3) trong giá trị đọc được
56  #bne a3, zero, gpio3_set # Nếu bit 3 = 1 (GPIO3 HIGH), nhảy tới 'gpio3_set'
57
58  # Kiểm tra GPIO4
```

This screenshot is identical to the one above, showing the same Arduino IDE interface with the same C program and simulation. The simulation status bar at the bottom indicates the program is running with a time of 00:18.533 and 57% completion.

Assignment 5

Code:

.global init

GPIO Register Definitions

```
.eqv GPIO_ENABLE_REG,    0x60004020 # GPIO output enable register
.eqv GPIO_OUT_REG,      0x60004004 # GPIO output register
.eqv IO_MUX_GPIO4_REG,  0x60009014 # GPIO4 function configuration
.eqv IO_MUX_GPIO5_REG,  0x60009018 # GPIO5 function configuration
.eqv IO_MUX_GPIO6_REG,  0x6000901C # GPIO6 function configuration
.eqv IO_MUX_GPIO7_REG,  0x60009020 # GPIO7 function configuration
```

7-segment display patterns for digits 0-9 (Common Anode - active low)

Segment mapping: GPIO0-6 -> segments a-g

.data

digits:

```
.word 0xC0 # 0: 1100 0000 - segments a,b,c,d,e,f on (0 = on)
.word 0xF9 # 1: 1111 1001 - segments b,c on
.word 0xA4 # 2: 1010 0100 - segments a,b,d,e,g on
.word 0xB0 # 3: 1011 0000 - segments a,b,c,d,g on
.word 0x99 # 4: 1001 1001 - segments b,c,f,g on
.word 0x92 # 5: 1001 0010 - segments a,c,d,f,g on
.word 0x82 # 6: 1000 0010 - segments a,c,d,e,f,g on
.word 0xF8 # 7: 1111 1000 - segments a,b,c on
.word 0x80 # 8: 1000 0000 - all segments on
.word 0x90 # 9: 1001 0000 - segments a,b,c,f,g on
```

.text

init:

```
# Configure GPIO0-7 as outputs
li a1, GPIO_ENABLE_REG
li a2, 0xFF          # Enable GPIO0-7 as outputs
sw a2, 0(a1)
```

```
# Configure GPIO4-7 function as GPIO
```

```
li a2, 0x1000        # Set GPIO function
li a1, IO_MUX_GPIO4_REG
sw a2, 0(a1)
li a1, IO_MUX_GPIO5_REG
sw a2, 0(a1)
li a1, IO_MUX_GPIO6_REG
sw a2, 0(a1)
li a1, IO_MUX_GPIO7_REG
sw a2, 0(a1)
```

display_loop:

```
la a3, digits        # Load address of digit patterns
li a4, 0              # Counter for digits 0-9
```

next_digit:

```
# Load and display the current digit pattern
```

```

lw a2, 0(a3)      # Load pattern for current digit
li a1, GPIO_OUT_REG
sw a2, 0(a1)      # Output pattern to GPIO

# Delay
call delay

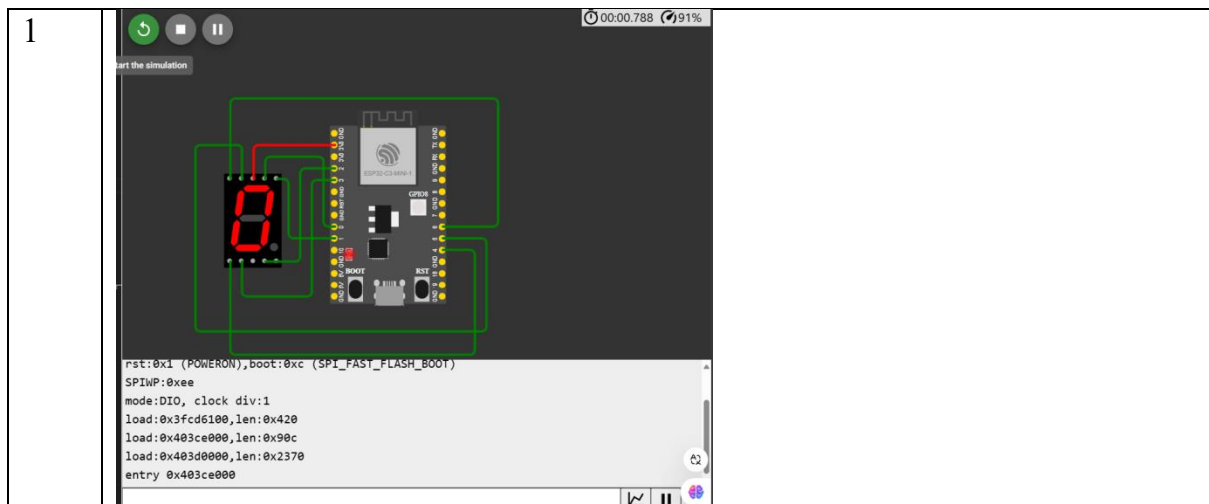
# Move to next digit
addi a3, a3, 4    # Next pattern address
addi a4, a4, 1    # Increment counter
li t0, 10
blt a4, t0, next_digit # If counter < 10, continue

j display_loop    # Repeat from 0

# Delay subroutine
delay:
    li t0, 0      # Counter
    li t1, 10000000 # Delay duration
delay_loop:
    addi t0, t0, 1
    blt t0, t1, delay_loop
ret

```

Output:



The image displays three sequential screenshots from a Tinkercad simulation, illustrating the setup and operation of an ESP32-C3-MINI-1 microcontroller connected to a 7-segment display. The display shows the numbers 8, 0, and 4 in sequence, corresponding to steps 2, 3, and 4 of the tutorial.

Step 2: The simulation shows the ESP32-C3-MINI-1 microcontroller connected to the 7-segment display. The display shows the number 8. The connections are as follows: VCC (pin 1) to 5V, GND (pin 10) to GND, and the remaining pins (2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100) are connected to GND. The display is connected to the microcontroller pins 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100.

Step 3: The simulation shows the ESP32-C3-MINI-1 microcontroller connected to the 7-segment display. The display shows the number 0. The connections are as follows: VCC (pin 1) to 5V, GND (pin 10) to GND, and the remaining pins (2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100) are connected to GND. The display is connected to the microcontroller pins 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100.

Step 4: The simulation shows the ESP32-C3-MINI-1 microcontroller connected to the 7-segment display. The display shows the number 4. The connections are as follows: VCC (pin 1) to 5V, GND (pin 10) to GND, and the remaining pins (2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100) are connected to GND. The display is connected to the microcontroller pins 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100.

The diagram illustrates the setup of an ESP32-C3-MINI-1 microcontroller connected to a 7-segment display. The connections are as follows:

- Power:** The display's VCC is connected to the 5V pin of the ESP32. The display's GND is connected to the GND pin of the ESP32.
- Data:** The display's data pins are connected to the ESP32's GPIO pins:
 - Display Pin 1 (Top Left) to ESP32 GPIO 4
 - Display Pin 2 (Top Right) to ESP32 GPIO 5
 - Display Pin 3 (Bottom Left) to ESP32 GPIO 6
 - Display Pin 4 (Bottom Right) to ESP32 GPIO 7

The sequence of images shows the display showing the numbers 8, 0, and 4, which are the hexadecimal values stored in the memory locations defined in the code.

The diagram illustrates the setup of an ESP32-C3-MINI-1 microcontroller connected to a 7-segment display. The connections are as follows:

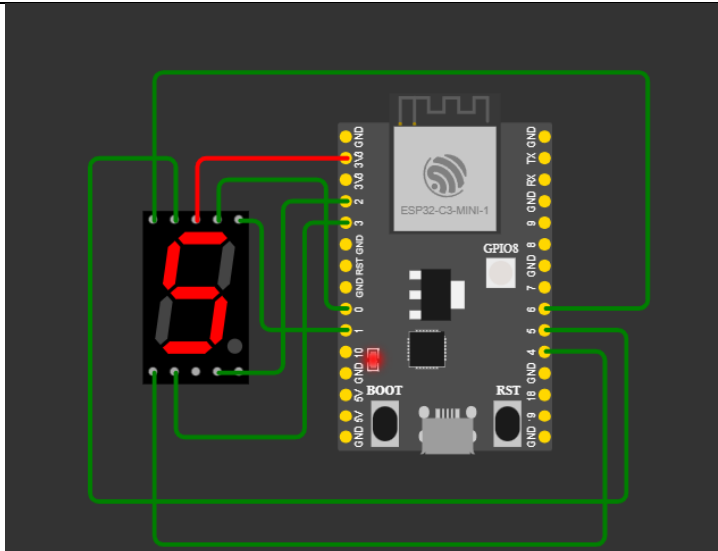
- Power:** The display's VCC pin is connected to the 5V pin on the ESP32. The display's GND pin is connected to a GND pin on the ESP32.
- Data:** The display's DATA pin is connected to GPIO4 on the ESP32.
- Control:** The display's CLK pin is connected to GPIO5 on the ESP32. The display's EN pin is connected to GPIO18 on the ESP32.

The three screenshots show the display showing the numbers 8, 0, and 4, which are the hexadecimal values stored in the memory locations 0x10, 0x11, and 0x12, respectively. The code snippet at the bottom of the third screenshot shows the initialization of these memory locations:

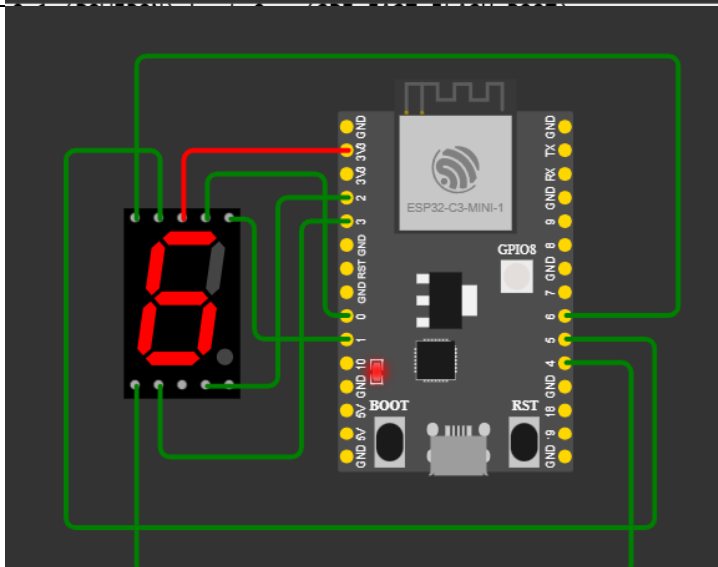
```

0x10 = 0x08; //POWERON
0x11 = 0x00; //SP
0x12 = 0x04; //FLASH_BOOT
  
```

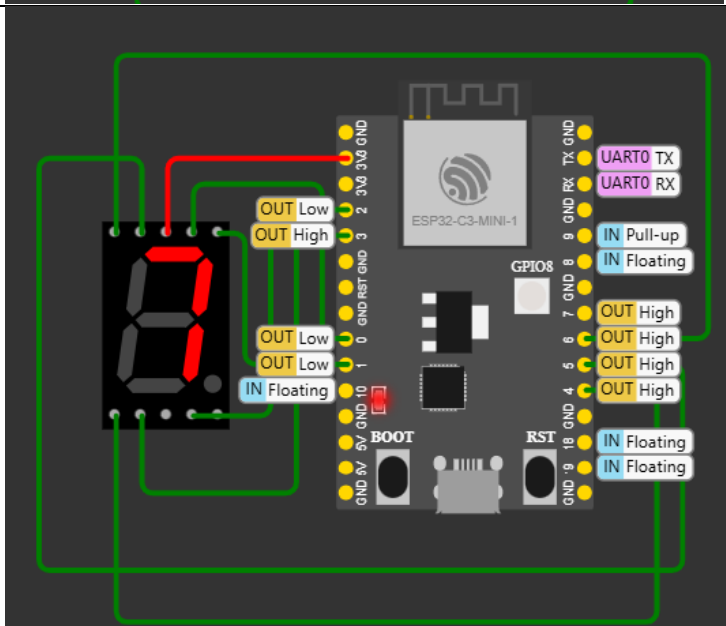
5

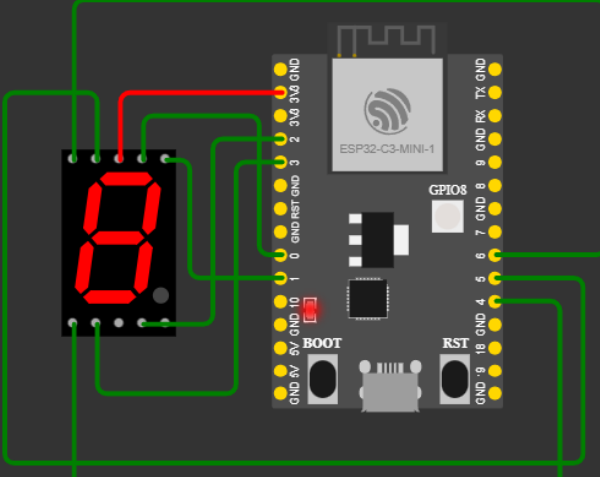
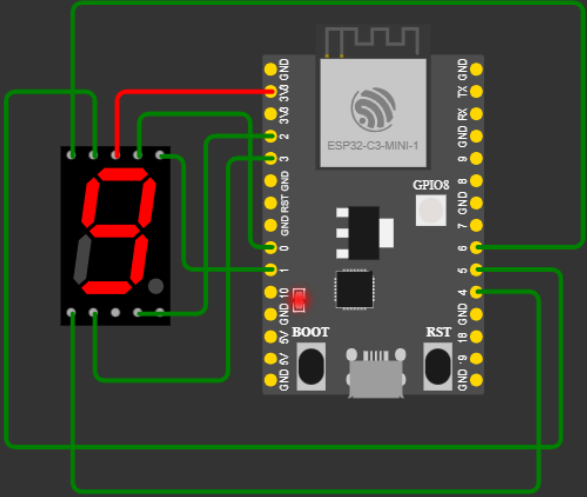


6



7



8		
9		

1. Khởi tạo (init):

- Cấu hình các chân GPIO0-7 làm đầu ra (output).
- Cấu hình chức năng GPIO cho các chân GPIO4-7.

2. Vòng lặp hiển thị (display_loop):

- Tải các giá trị số từ bảng digits.
- Lặp qua các giá trị số (0-9) và hiển thị lên các chân GPIO.
- Sau mỗi lần hiển thị, gọi hàm **delay** để tạo độ trễ.

3. Chuyển sang số tiếp theo (next_digit):

- Cập nhật địa chỉ và chỉ số số tiếp theo.
- Nếu đã hiển thị đủ 10 số (0-9), quay lại hiển thị từ số 0.

4. Hàm delay (delay):

- Tạo một độ trễ bằng cách sử dụng một vòng lặp không làm gì ngoài việc tăng biến đếm.