Thực hành kiến trúc máy tính

Báo cáo thực hành

Bài 13. Lập trình hợp ngữ với ESP32-C3 –

Mô phỏng bằng Wokwi

|  |  |
| --- | --- |
| Họ Tên | Lê Thành An |
| MSSV | 20235631 |

**ASSIGNMENT**

ĐOẠN MÃ :

.global init

.equ GPIO\_ENABLE\_REG, 0x60004020 # Thanh ghi cho phép xuất tín hiệu GPIO

.equ GPIO\_OUT\_REG, 0x60004004 # Thanh ghi thiết lập mức logic đầu ra GPIO

.equ GPIO\_IN\_REG, 0x6000403C # Thanh ghi đọc trạng thái GPIO

.equ IO\_MUX\_GPIO0\_REG, 0x60009004

.equ IO\_MUX\_GPIO1\_REG, 0x60009008

.equ IO\_MUX\_GPIO2\_REG, 0x6000900C

.equ IO\_MUX\_GPIO3\_REG, 0x60009010

.equ IO\_MUX\_GPIO4\_REG, 0x60009014

.equ IO\_MUX\_GPIO5\_REG, 0x60009018

.equ IO\_MUX\_GPIO6\_REG, 0x6000901C

.equ IO\_MUX\_GPIO7\_REG, 0x60009020

.data

led\_patterns: # Mã hiển thị cho LED 7 thanh Anode chung (gfedcba), 0=ON, 1=OFF

.word 0xC0 # Số 0 (a,b,c,d,e,f ON)

.word 0xF9 # Số 1 (b,c ON)

.word 0xA4 # Số 2 (a,b,g,e,d ON)

.word 0xB0 # Số 3 (a,b,g,c,d ON)

.word 0x99 # Số 4 (f,g,b,c ON)

.word 0x92 # Số 5 (a,f,g,c,d ON)

.word 0x82 # Số 6 (a,c,d,e,f,g ON)

.word 0xF8 # Số 7 (a,b,c ON)

.word 0x80 # Số 8 (a,b,c,d,e,f,g ON)

.word 0x90 # Số 9 (a,b,c,f,g ON)

.text

init:

# --- Cấu hình GPIO0-GPIO6 là OUTPUT ---

li a1, GPIO\_ENABLE\_REG

li a2, 0x007F # Cho phép output trên GPIO0-GPIO6 (các bit từ 0 đến 6)

sw a2, 0(a1)

# --- Cấu hình MUX cho GPIO0-GPIO6 sang chức năng GPIO ---

# Giá trị 0x1000 để đặt MCU\_SEL = 1 (chức năng GPIO)

li t0, 0x1000

li a1, IO\_MUX\_GPIO0\_REG

sw t0, 0(a1)

li a1, IO\_MUX\_GPIO1\_REG

sw t0, 0(a1)

li a1, IO\_MUX\_GPIO2\_REG

sw t0, 0(a1)

li a1, IO\_MUX\_GPIO3\_REG

sw t0, 0(a1)

li a1, IO\_MUX\_GPIO4\_REG

sw t0, 0(a1)

li a1, IO\_MUX\_GPIO5\_REG

sw t0, 0(a1)

li a1, IO\_MUX\_GPIO6\_REG

sw t0, 0(a1)

# --- Cấu hình GPIO7 là INPUT ---

# GPIO\_ENABLE\_REG đã đặt bit 7 (GPIO7) là 0 (input) do giá trị 0x007F ở trên.

li a1, IO\_MUX\_GPIO7\_REG

# Đặt MCU\_SEL=1 (0x1000) VÀ FUN\_IE=1 (0x0200 - cho phép input)

li t1, 0x1200

sw t1, 0(a1)

# --- Khởi tạo các thanh ghi cho vòng lặp ---

la s5, led\_patterns # s5 = địa chỉ cơ sở của mảng led\_patterns

li s4, GPIO\_OUT\_REG # s4 = địa chỉ của thanh ghi GPIO\_OUT\_REG

li s3, 0 # s3 = giá trị số đang hiển thị (0-9), khởi tạo là 0

# Hiển thị số 0 ban đầu

lw s6, 0(s5) # s6 = mã pattern cho số 0 (current\_display\_pattern)

sw s6, 0(s4) # Ghi ra cổng GPIO để hiển thị

main\_loop:

# Đọc trạng thái GPIO7

li t0, GPIO\_IN\_REG

lw t1, 0(t0)

andi t2, t1, 0x0080 # Mask để lấy giá trị bit 7 (GPIO7)

beqz t2, skip\_increment # Nếu GPIO7 là 0 (LOW), bỏ qua việc tăng số

# GPIO7 là 1 (HIGH), tiến hành tăng số

addi s3, s3, 1 # current\_digit\_value++

li t3, 10 # So sánh với 10

bne s3, t3, update\_led\_pattern\_register # Nếu chưa bằng 10 thì cập nhật pattern

li s3, 0 # Nếu bằng 10, reset về 0

update\_led\_pattern\_register:

slli t4, s3, 2 # offset = current\_digit\_value \* 4 (vì mỗi word là 4 byte)

add t5, s5, t4 # address\_of\_pattern = led\_patterns\_base + offset

lw s6, 0(t5) # s6 = mã pattern mới để hiển thị (current\_display\_pattern)

skip\_increment:

# Hiển thị pattern trong s6 (có thể là pattern cũ hoặc mới)

sw s6, 0(s4)

call delay\_asm # Gọi hàm delay

j main\_loop

# --- Hàm Delay ---

delay\_asm:

li a3, 0 # Biến đếm

li a4, 5000000 # Thời gian chờ (số lần lặp, điều chỉnh để có tốc độ mong muốn)

loop\_delay:

addi a3, a3, 1

blt a3, a4, loop\_delay

ret