TỔNG KẾT BÀI TẬP HÀNG TUẦN

Họ tên sinh viên	MSSV	Lớp (thứ - tiết)
Hoàng Ngọc Dung	23139006	Thứ 5 – Sáng chiều

MỤC LỤC

Phần 1: Vi điều khiển 8051

Phần 1.1: Lập trình ASM

Tuần 1 – Bài 06 : Tổng quan về dòng 8051 (VDK:89S52)

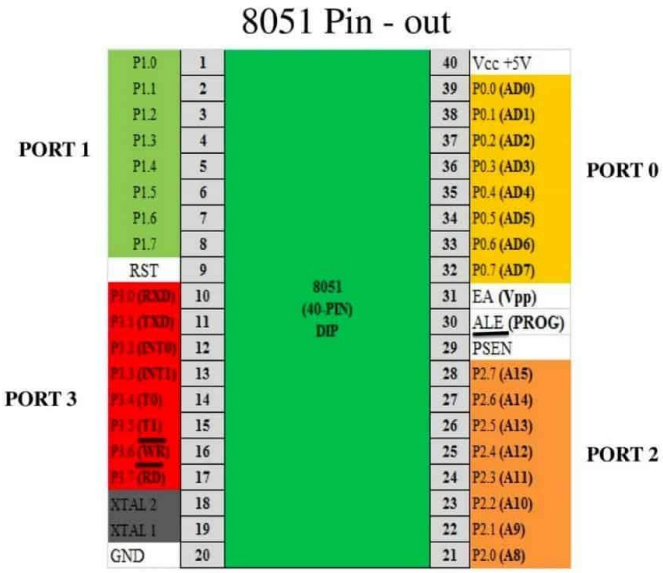
6.1. Bài tập và câu hỏi

1. Trình bày cấu trúc cơ bản của dòng vi điều khiển 8051.

- 8051 là dòng vi điều khiển 8 bit. (Kích thước thanh ghi, ô nhớ, bộ thực hiện phép toán,... là 8 bit.)
- VĐK 8051 sử dụng kiến trúc Harvard (Code lưu trong ROM, Data lưu trong RAM).
- Hệ thống có 3 loại bus chính:
- Control bus: Chứa các tín hiệu điều khiển RD/WR, EA, PSEN, ALE, RST,...
- Data bus (8 bit): Truyền dữ liệu
- Address bus (16 bit): Xác định địa chỉ ô nhớ giao tiếp
- Thanh ghi 16 bit:
    - PC (Program Counter – 16 bit): Vì vậy, VĐK có thể giao tiếp bộ nhớ ROM/RAM lên tới 64 KB (**2^6 ∗ 2^10 = 2^16**).
    - Địa chỉ: 0000h – 0FFFFh
    - DPTR (Data Pointer – 16 bit): Hỗ trợ trong quá trình giao tiếp bộ nhớ.

Vi điều khiển 8051 gồm các khối chức năng:

- Bộ xử lý trung tâm (CPU)
- Bộ nhớ chương trình (ROM 4kB) và bộ nhớ dữ liệu (RAM 128B)
- Các thanh ghi có chức năng đặc biệt (SFR 128B)
- Các cổng vào ra (P0,P1,P2,P3)
- Bộ định thời/bộ đếm (Timer0, Timer1)
- Bộ giao tiếp nối tiếp (Serial)
- Hỗ trợ 5 nguồn ngắt:
    - 2 ngắt ngoài (INT0, INT1)
    - 2 ngắt định thời (Timer 0, Timer 1)
    - 1 ngắt cổng nối tiếp (Serial)



Hình 1: Sơ đồ chân của VDK 8051

2. Liệt kê các ngoại vi cơ bản của dòng vi điều khiển 8051.

- Nguồn xung Clock: cung cấp xung nhịp CPU hoạt động
  - Bộ dao động OSC cần được cấp 1 nguồn xung (Crystal – Thạch anh) từ bên ngoài.
    - Thạch anh được nối vào 2 chân: XTAL1, XTAL2.
    - Cần gắn thêm 2 tụ 33 pF ở 2 chân Crystal xuống đất để giảm nhiễu.
  - VĐK sử dụng Crystal (thạch anh) có tốc độ từ 2 MHz ~ 33 MHz. (Thường dùng: 12 MHz, 11.0592 MHz)
  - 1 chu kỳ máy (Machine Cycle) = 12 chu kỳ xung clock.
  - Mỗi lệnh thực thi tốn 1–3 chu kỳ máy.
- Chân reset (RST)
  - Mặc định chân Reset ở mức cao.
  - Khi nhấn, thì chân Reset kéo xuống mức thấp.
    - VĐK sẽ được khởi động lại, CPU nạp lệnh đầu tiên trong ô nhớ đầu tiên để thực thi.
    - Chương trình sẽ bắt đầu chạy lại từ đầu.
- Chân EA

- chân EA cần nối VCC (vì xài flash nội).
  - EA = 1: VĐK giao tiếp Flash nội..
  - EA = 0: VĐK giao tiếp Flash ngoại.

- Các cổng vào ra (P0,P1,P2,P3)
- Bộ định thời/bộ đếm (Timer0, Timer1)
- Bộ giao tiếp nối tiếp (Serial)

3. Trình bày các vùng bộ nhớ và ý nghĩa của dòng vi điều khiển 8051.

- Bộ nhớ chương trình (Program Memory): Dùng để lưu trữ mã lệnh chương trình.
  - ROM nội (EEPROM/Flash) 4KB
  - Khi chạy, CPU sẽ đọc lệnh từ đây để thực thi.
  - Nếu ROM nội không đủ, có thể mở rộng thêm ROM ngoài đến tối đa 64KB
- Bộ nhớ dữ liệu (Data Memory):
  - Dùng để lưu dữ liệu tạm thời RAM nội Dung lượng: 128 byte
  - Chia thành các vùng nhỏ:
- 00h – 1Fh: 32 byte cho 4 bank thanh ghi (R0–R7).
- 20h – 2Fh: 16 byte RAM định địa chỉ từng bit (128 bit).
- 30h – 7Fh: RAM đa dụng (biến tạm, dữ liệu người dùng).
- Ngoài ra còn có RAM đặc biệt (SFR, 80h – FFh): chứa các thanh ghi chức năng đặc biệt như ACC, B, PSW, SP, DPTR, các thanh ghi điều khiển Timer, Serial, ngắt, các cổng P0–P3…Có thể mở rộng thêm RAM ngoài thêm 64KB

6.2. Chương trình mẫu

4. Tạo Project ASM và chạy mô phỏng Proteus với đoạn chương trình sau.

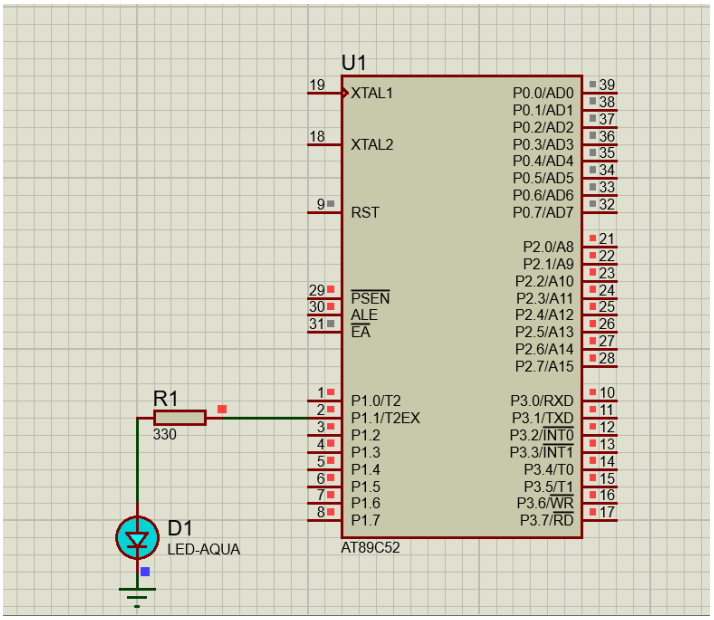
Hỏi: Mục đích của đoạn chương trình là gì? (Chạy mô phỏng xem kết quả).

Chương trình dùng để làm nhấp nháy LED tại chân P1.1 của vi điều khiển 8051, với tần số khoảng 2.5 Hz (một lần bật/tắt mỗi 200 ms).

```
    LED1 EQU P1.1       ; Đặt nhãn LED1 cho chân P1.1 của cổng P1
    ORG 00H             ; Đặt địa chỉ bắt đầu của chương trình tại 0000h
    SETB LED1           ; Đặt bit P1.1 = 1 (LED tắt nếu nối cực dương chung)
LOOP:
    CPL LED1             ; Đảo trạng thái của LED
    CALL DELAY200ms      ; Gọi hàm tạo trễ khoảng 200 ms
```

```
    JMP LOOP            ; Quay lại lặp vô hạn

DELAY200ms:
    MOV R2, #200        ; R2 = 200 (vòng lặp ngoài)
LAP2:
    MOV R1, #200        ; R1 = 200 (vòng lặp trong)
LAP1:
    NOP                 ; Lệnh rỗng, chỉ chiếm thời gian
    NOP
    NOP
    DJNZ R1, LAP1       ; Giảm R1, nếu R1 ≠ 0 thì quay lại LAP1
    DJNZ R2, LAP2       ; Giảm R2, nếu R2 ≠ 0 thì quay lại LAP2
    RET                 ; Kết thúc hàm trễ, quay lại chương trình chính
END
```

Mô phỏng protues



Bảng phân tích:

Thành phần	Chức năng	Vai trò
MOV R2, #200	Gán 200 vào thanh ghi R2	Bộ đếm vòng ngoài
MOV R1, #200	Gán 200 vào thanh ghi R1	Bộ đếm vòng trong
NOP	Lệnh rỗng (tốn thời gian)	Dùng để kéo dài trễ
DJNZ R1, LAP1	Giảm R1, nếu chưa bằng 0 thì lặp lại LAP1	Vòng lặp nhỏ (~200 lần)
DJNZ R2, LAP2	Giảm R2, nếu chưa bằng 0 thì quay lại LAP2	Vòng lặp lớn (200×200 lần)
RET	Trở lại hàm gọi (CALL)	Kết thúc delay

5. Chạy mô phỏng đoạn code sau

Chương trình này làm LED nối tại chân P1.1 nhấp nháy liên tục (bật 200 ms, tắt 200 ms),

```
#include<REGX52.H> // SFR – Special Function Register

sbit LED = P1^1

void delay_ms(unsigned int ms) {
    unsigned int i,j;
    for (i = 0; i < ms ; i++)
            for (j = 0; j < 123; j++);
}
void main(){
    while(1) {
        LED = 1;
        delay_ms(200);
        LED = 0;
        delay_ms(200);
    }
}
```

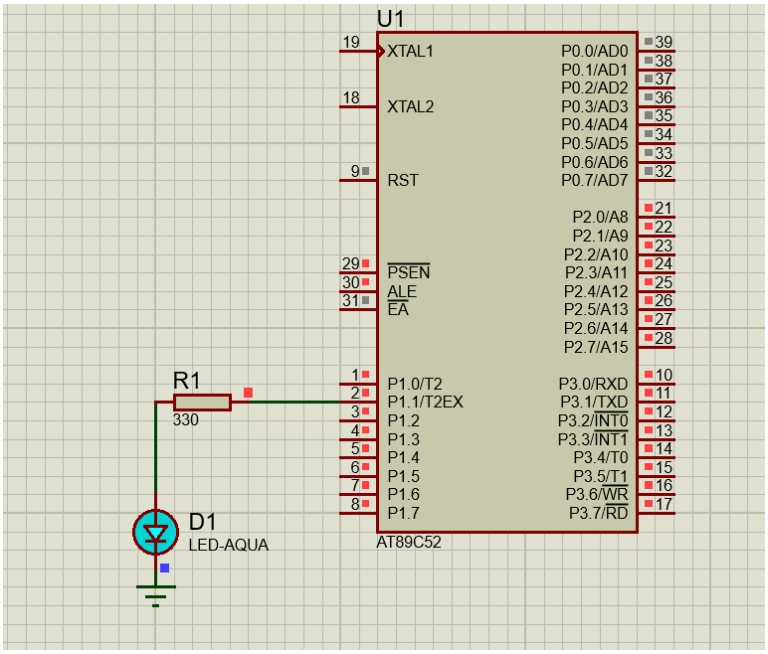
Nếu LED được nối kiểu cực dương chung (Vcc → LED → điện trở → P1.1) thì:

- LED = 1 → LED tắt
- LED = 0 → LED sáng

Với thạch anh 12 MHz, việc lặp 123 lần cho mỗi mili-giây cho ra trễ tương đối đúng:

1 vòng ngoài ≈ 1 ms.

delay_ms(200) ≈ 200 ms.



Tuần 1 – Bài 07 : Tập lệnh ASM của 8051

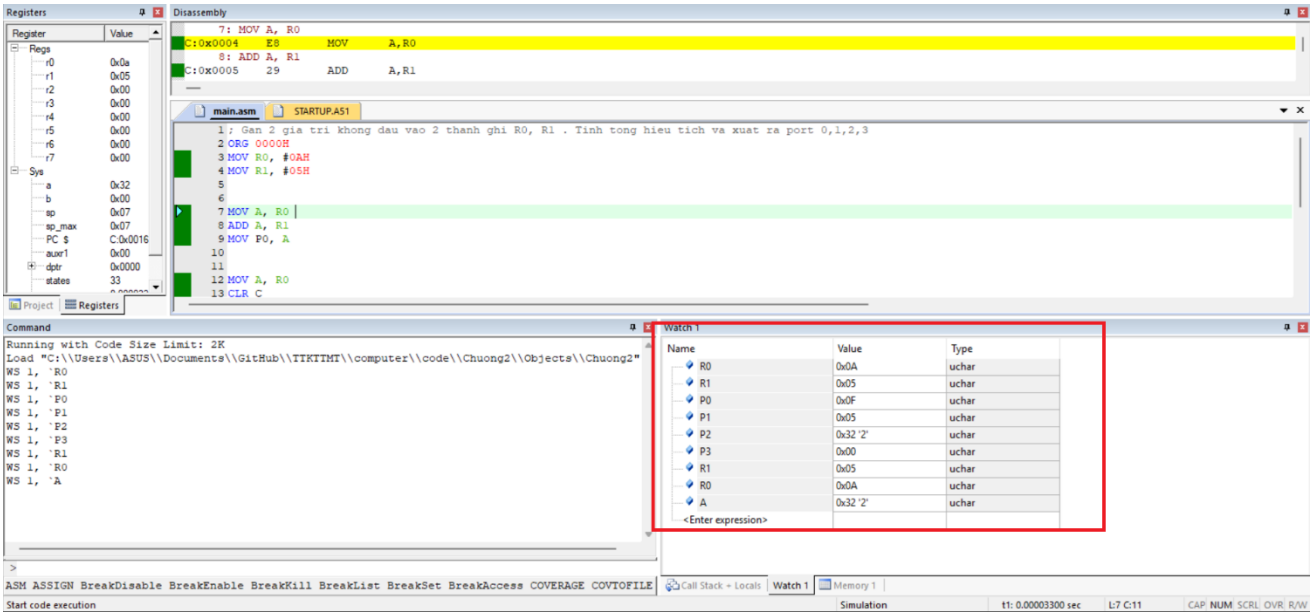
7.1. Chương trình mẫu

1. Chạy lại 5 chương trình mẫu, xem và giải thích toàn bộ chương trình.

Chương trình 1: Gán 2 gia tri khong dau vao 2 thanh ghi R0, R1 . Tinh tong hieu tich va xuat ra port 0,1,2,3

```
ORG 0000H
MOV R0, #0AH
MOV R1, #05H
MOV A, R0
ADD A, R1
MOV P0, A
MOV A, R0
CLR C
SUBB A, R1
MOV P1, A
MOV A, R0
MOV B, R1
MUL AB
MOV P2, A
MOV P3, B

JMP $
END
```

Kết quả chạy debug:

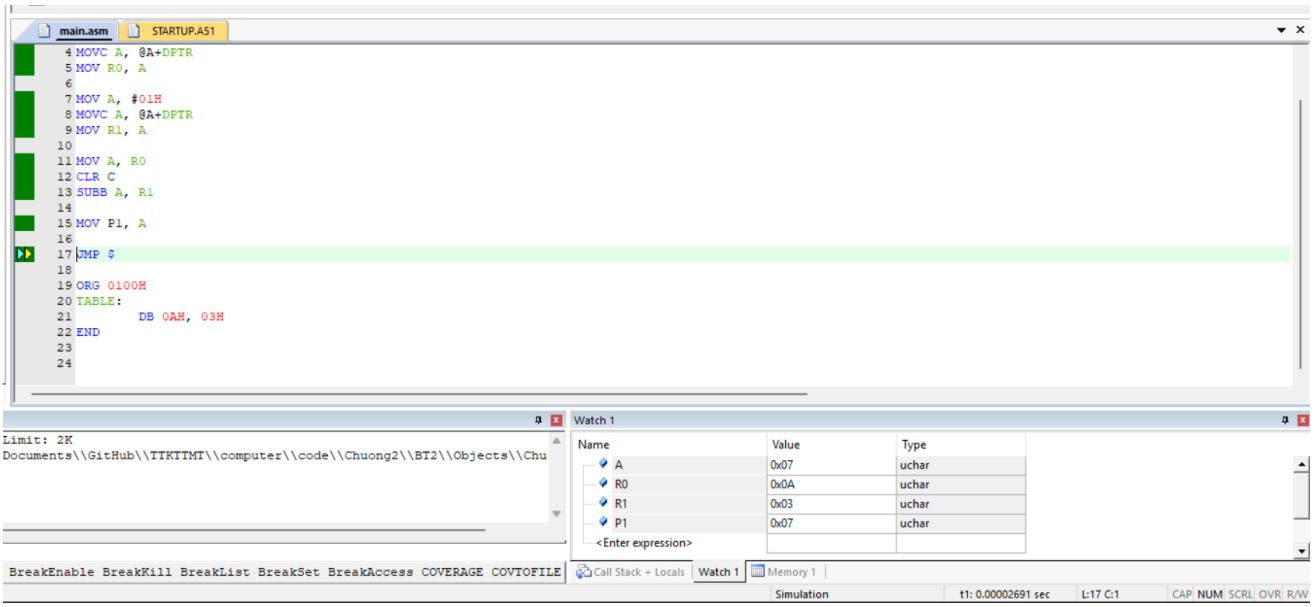


Bảng phân tích:

Thanh ghi	Giá trị hex	Giá trị decimal	Giải thích
P0	0x0F	15	Tổng: 10 + 5 = 15
P1	0x05	5	Hiệu: 10 − 5 = 5
P2	0x32	50	Tích: 10 × 5 = 50 (byte thấp)
P3	0x00	0	Byte cao của tích (vì 50 < 256)

Chương trình 2: Lưu 2 hằng số vào vùng nhớ ROM, đọc và tính hiệu và xuất ra Port 1.

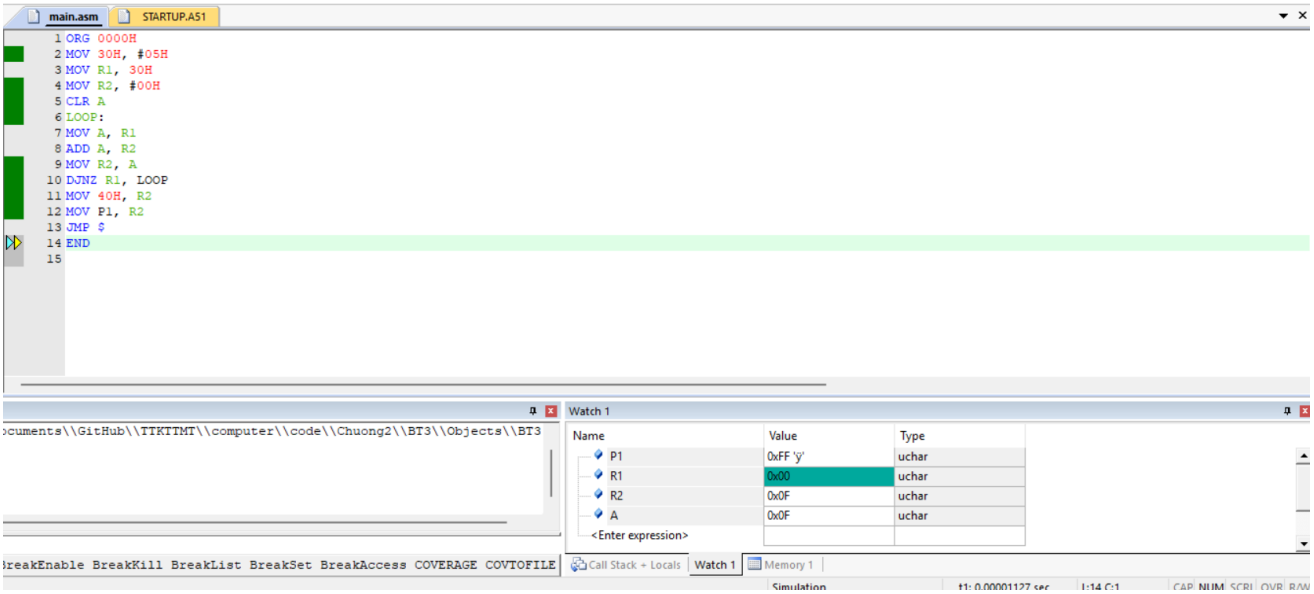
```
ORG 0000H
MOV DPTR, #TABLE
MOV A, #00H
MOVC A, @A+DPTR
MOV R0, A

MOV A, #01H
MOVC A, @A+DPTR
MOV R1, A

MOV A, R0
CLR C
SUBB A, R1

MOV P1, A

JMP $

ORG 0100H
TABLE:
        DB 0AH, 03H
END
```



Tên biến	Giá trị (hex)	Giá trị thập phân	Vai trò
A	0x07	7	Kết quả cuối cùng sau phép trừ R0 - R1
R0	0x0A	10	Hằng số thứ nhất đọc từ ROM (DB 0AH)
R1	0x03	3	Hằng số thứ hai đọc từ ROM (DB 03H)
P1	0x07	7	Giá trị được xuất ra Port 1 (chính là kết quả A)

Chương trình 3: Gán 1 số tự nhiên vào bộ nhớ RAM có địa chỉ 30H. Tính tổng các số nguyên từ 0 đến giá trị ô nhớ có địa chỉ 30H. Lưu kết quả vào ô nhớ có địa 40H và xuất ra port 1

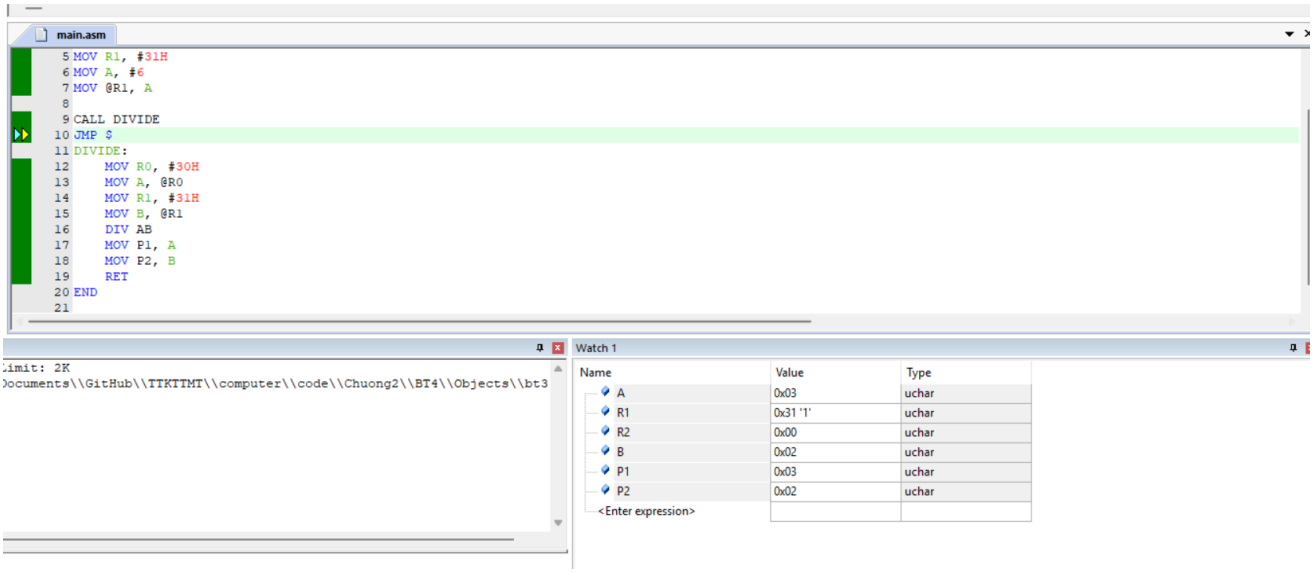
```
ORG 0000H
MOV 30H, #05H
MOV R1, 30H
MOV R2, #00H
CLR A
LOOP:
MOV A, R1
ADD A, R2
MOV R2, A
DJNZ R1, LOOP
MOV 40H, R2
MOV P1, R2
JMP $
END
```



- Đây là chương trình cộng dồn các số từ 1 đến N, trong đó N = 5.
- Kết quả là: 5 + 4 + 3 + 2 + 1 = 15
- Kết quả cuối cùng được xuất ra Port 1 và lưu trong RAM nội tại 40H.

Chương trình 4: Lưu 2 giá trị vào bộ nhớ RAM nội, sử dụng cách truy xuất bằng địa chỉ gián tiếp. Thực hiện phép chia 2 số này và xuất kết quả ra port P2. Viết chương trình con thực hiện phép chia và xuất ra 2 port.

```
ORG 0000H
MOV R0, #30H
MOV A, #20
MOV @R0, A
MOV R1, #31H
MOV A, #6
MOV @R1, A

CALL DIVIDE
JMP $
DIVIDE:
    MOV R0, #30H
    MOV A, @R0
    MOV R1, #31H
    MOV B, @R1
    DIV AB
    MOV P1, A
    MOV P2, B
    RET
END
```



Bảng phân tích:

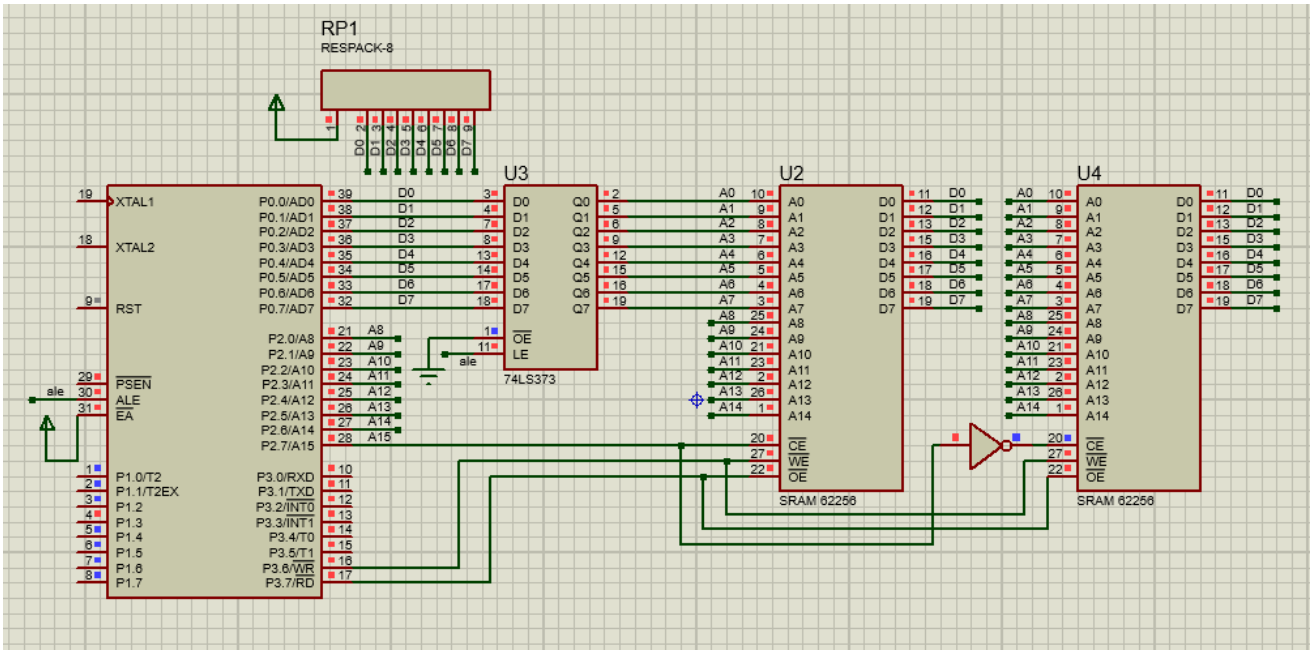
Thanh ghi	Giá trị (hex)	Thập phân	Ý nghĩa
A	03H	3	Thương
B	02H	2	Dư
P1	03H	3	Xuất thương
P2	02H	2	Xuất dư

Chương trình 5: Viết chương trình ghi dữ liệu vào bộ nhớ RAM ngoài, sau đó ra kiểm tra lại xem đúng không

```
ORG 0000H

MAIN:
    MOV DPTR, #07FFAH
    MOV R3, #08H
    MOV A, #01H
REP1:
    MOVX @DPTR, A
    RL A
    INC DPTR
    DJNZ R3, REP1

    MOV DPTR, #07FFAH
    MOV R3, #08H

REP2:
    MOVX A, @DPTR
    MOV P1, A
```

```
    CALL DELAY
    INC DPTR
    DJNZ R3, REP2
    SJMP MAIN
DELAY:
    MOV R1, #200
LAP1:
    MOV R0, #200
LAP:
    NOP
    DJNZ R0, LAP
    DJNZ R1, LAP1
    RET

END
```

Mô phỏng proteus



7.2. Câu hỏi và bài tập

2. Tính chính xác thời gian delay khi gọi đoạn chương trình con DELAY (trong chương trình mẫu số 5).

```
DELAY:
    MOV R1, #200
LAP1:
    MOV R0, #200
LAP:
    NOP
```

```
    DJNZ R0, LAP
    DJNZ R1, LAP1
    RET
```

Vòng lặp LAP : $T1 = 200 \times 3 = 600\ MC$ (200 lần lặp, Mỗi lần gồm NOP (1) + DJNZ (2))

Lần	NOP (1 MC)	DJNZ (2 MC)	Tổng
1 → 199	1	2	3 MC mỗi vòng
Lần 200 (DJNZ không nhảy)	1	2	3 MC

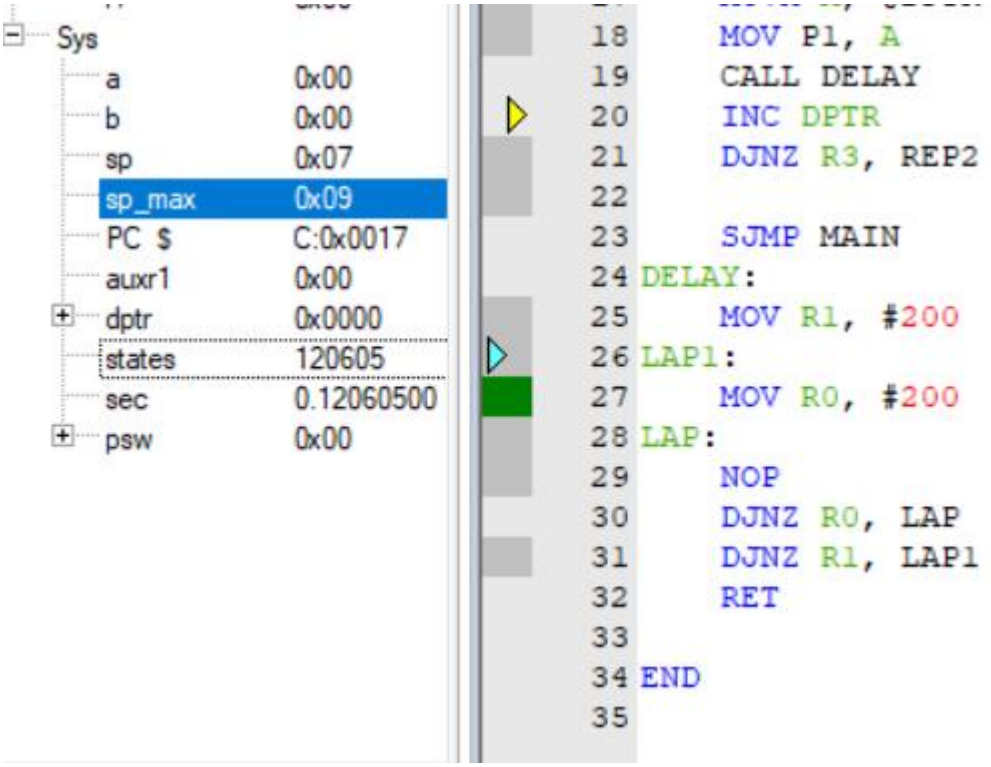
Vòng lặp LAP 1 (603 MC)

MOV R0, #200 ; 1 MC

LAP: ; 600 MC

DJNZ R1, LAP1 ; 2 MC

Tổng 600 + 603 = 1203 MC



Từ hình ảnh : states = 120605 Tổng số chu kỳ máy (machine states) mà CPU đã chạy kể từ đầu mô phỏng với sec = 0.12060500 (Tổng thời gian thực tương ứng (theo f_osc đã cài đặt trong Keil). Cài đặt 8051 dùng 12 xung dao động (oscillator) cho 1 machine cycle, nên:

$$T = \frac{12}{12Mhz} = 1us$$

Do đó:

$$Thời\ gian = 120605 \times 1\mu s = 120.605\ ms$$

Tần số thạch anh	Chu kỳ máy	Thời gian Delay
12.000 MHz	1.000 µs	120.403 ms

3. Viết ra mã lệnh máy (machine code) + kèm địa chỉ ô nhớ + kèm thời gian thực thi từng lệnh cho đoạn chương trình số 01.

Code asm	**Machine code**

ORG 0000H
MOV R0, #0AH
MOV R1, #05H
 	
MOV A, R0
ADD A, R1
MOV P0, A
 	
MOV A, R0
CLR C
SUBB A, R1
MOV P1, A
 	
MOV A, R0
MOV B, R1
MUL AB
MOV P2, A
MOV P3, B
JMP $
END
	ORG 0000H
	0111 1000 0000 1010 -> 780A
	0111 1001 0000 0101 -> 7905
	
	1110 1000 -> E8
	0010 1001 -> 29
	1111 0101 1000 0000 -> F5 80
	
	1110 1000 -> E8
	1100 0011 -> C3
	1001 1001 -> 99
	F5 90
	
	1110 1000 -> E8
	1110 1001 -> 89F0
	1010 0100 -> A4
	F5 A0
	1000 0101 1111 0000 1011 -> F5 F0 B0
	80FE
	END

Bảng tổng hợp

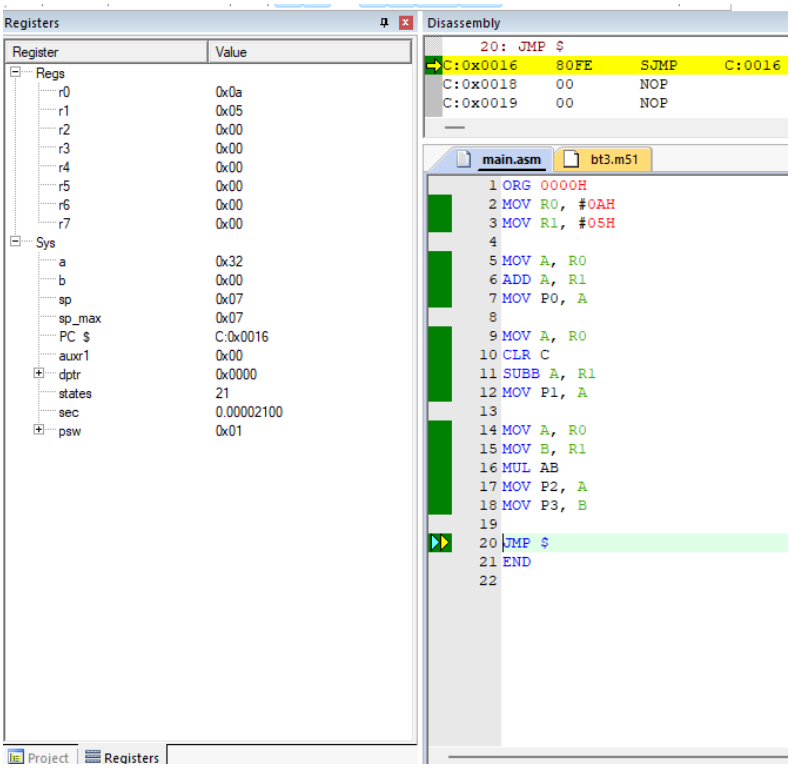
Lệnh Assembly	Size	Chu kỳ máy	Thời gian f = 12 MHz
MOV R0, #0AH	2	1	2 µs
MOV R1, #05H	2	1	2 µs
MOV A, R0	1	1	1 µs
ADD A, R1	1	1	1 µs
MOV P0, A	2	2	2 µs
MOV A, R0	1	1	1 µs

CLR C	1	1	1 µs
SUBB A, R1	1	1	1 µs
MOV P1, A	1	2	2 µs
MOV A, R0	1	1	1 µs
MOV B, R1	1	1	1 µs
MUL AB	1	4	4 µs
MOV P2, A	2	2	2 µs
MOV P3, B	2	2	2 µs
JMP $	2	2	2 µs

Dung lượng chương trình : 23 bytes

Thời gian thực thi 21 µs

Mã máy : 78 0A 79 05 E8 28 F5 80 E8 C3 98 F5 90 E8 C9 A4 F5 A0 85 F0 B0 80 FE

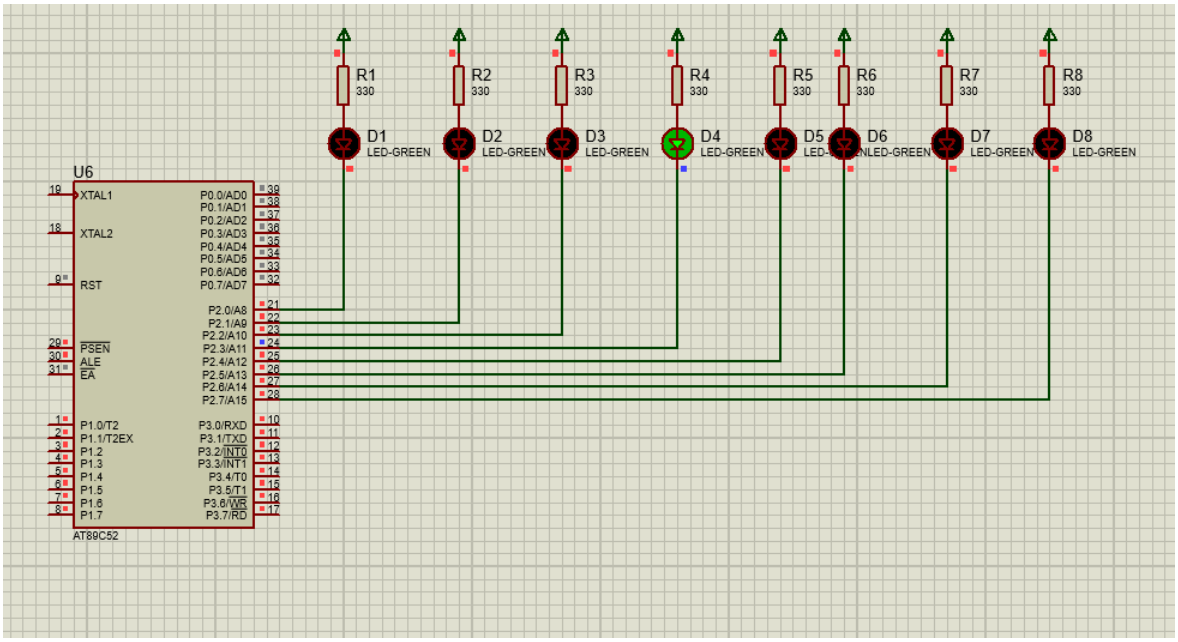


Tuần 2 – Bài 08: Nhập xuất GPIO

8.1. Chương trình mẫu

a) Chớp tắt một led đơn ở pin 2.3 với chu kì 2s

```
ORG 0000H

BEGIN_ENTRY:
    SETB P2.3


MAIN_LOOP:
    CPL P2.3
    LCALL DELAY1S
    SJMP MAIN_LOOP

DELAY1S:
    MOV R3, #5
S1BACK:
    LCALL DELAY200ms
    DJNZ R3, S1BACK
    RET

DELAY200ms:
    MOV R2, #200

LAP2:
    MOV R1, #200
LAP1:
    NOP
    NOP
    NOP
    DJNZ R1, LAP1
    DJNZ R2, LAP2
    RET

END
```



b) Chương trình bật tắt 1 led đơn pin P2.0 bằng một nút nối P3.0

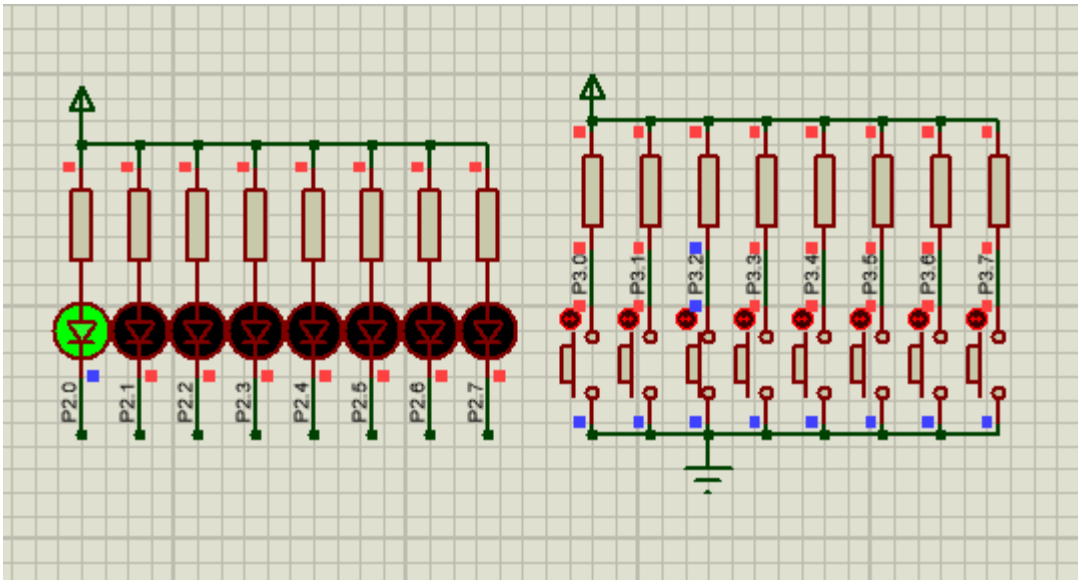
Khi nhấn nút P3.0, thì LED P2.0 sáng.

Khi nhấn nút P3.1, thì LED P2.0 tắt.

```
ORG 00H
START:

        SETB P2.0
        SETB P3.0
        SETB P3.1

MAIN_LOOP:
        JB P3.0, CHECK_OFF
        CLR P2.0
        SJMP MAIN_LOOP

CHECK_OFF:
        JB P3.1, MAIN_LOOP
        SETB P2.0
        SJMP MAIN_LOOP
END
```

Thêm bật tắt 2 đèn bằng 4 nút nhấn

```
ORG 0000H

START:
        SETB P2.0         ; LED1 OFF
        SETB P2.1         ; LED2 OFF

MAIN_LOOP:

        JNB P3.0, LED1_ON
        JNB P3.1, LED1_OFF
        SJMP CHECK_LED2

LED1_ON:
        CLR P2.0
        SJMP CHECK_LED2

LED1_OFF:
        SETB P2.0
        SJMP CHECK_LED2


CHECK_LED2:
        JNB P3.6, LED2_ON
        JNB P3.7, LED2_OFF
        SJMP MAIN_LOOP

LED2_ON:
        CLR P2.1
        SJMP MAIN_LOOP

LED2_OFF:
        SETB P2.1
        SJMP MAIN_LOOP

END
```



8.2. Bài tập và câu hỏi

1. Viết chương trình tạo 5 hiệu ứng led đơn (dịch, sáng dồn, tắt dần, chớp tắt, sáng dần).

Code 5 hiệu ứng led đơn

```
BASE DATA 30H
MASK DATA 31H
PLUS DATA 32H
J DATA 33H

ORG 0000H

MAIN:
    LCALL EFFECT_DICH
    LCALL EFFECT_SANGDON
    LCALL EFFECT_TATDAN
    LCALL EFFECT_CHOPTAT
    LCALL EFFECT_SANGDAN
    SJMP MAIN

EFFECT_DICH:
    MOV A, #0FEH ; 1111 1110 P2.0 sang
DICH_LOOP:
    MOV P2,A
    LCALL DELAY200MS
    RL A
    CJNE A, #7FH, DICH_LOOP ; ss P2.7 = 1 -> quay lai P2.0
    MOV P2, #7FH
    LCALL DELAY200MS
    ;MOV A, #0FEH
```

```
    RET
EFFECT_TATDAN:
    MOV A, #00
TATDAN_LOOP:
    MOV P2, A
    LCALL DELAY200MS

    RL A
    ORL A, #01H ; dua bit 1

    CJNE A, #0FFH, TATDAN_LOOP
    MOV A, #000H
    RET

EFFECT_SANGDAN:
    MOV A, #0FFH
SANGDAN_LOOP:
    MOV P2, A
    RL A
    ANL A, #0FEH ; 1111 1110
    LCALL DELAY200MS
    CJNE A, #00H, SANGDAN_LOOP
    MOV A, #0FFH
    RET

EFFECT_CHOPTAT:
    SETB P2.0
    SETB P2.1
    SETB P2.2
    SETB P2.3 ; LED tat
    SETB P2.4
    SETB P2.5
    SETB P2.6
    SETB P2.7
    MOV R0, #5
CHOPTAT_LOOP:
    CPL P2.0
    CPL P2.1
    CPL P2.2
    CPL P2.3 ; LED dao
    CPL P2.4
    CPL P2.5
    CPL P2.6
    CPL P2.7
    LCALL DELAY200MS
```

```
    DJNZ R0, CHOPTAT_LOOP
    LCALL MAIN
    RET

EFFECT_SANGDON:
    MOV BASE, #0FFH
    MOV MASK, #0FFH
    MOV R4, #0 ;i

FOR_I:
    CJNE R4,#8, OK_I
    RET
    ;SJMP MAIN_LOOP
OK_I:
    MOV A, BASE
    CLR C
    RRC A
    MOV BASE, A
    MOV PLUS, #01H

    MOV A, #9
    CLR C
    SUBB A, R4     ; A = 8 - i
    MOV J , A
FOR_J:
    DJNZ J, OK_J
    SJMP END_J
OK_J:
    MOV A, PLUS
    CPL A ; A = ~plus
    ANL A, MASK ; A = mask & ~plus
    MOV P2, A

    MOV A, PLUS
    RL A
    MOV PLUS, A
    LCALL DELAY200MS
    SJMP FOR_J
END_J:
    MOV MASK, BASE
    INC R4
    SJMP FOR_I

DELAY200MS:
    MOV R2, #200
```

```
D2:
    MOV R1, #200
D1:
    NOP
    NOP
    NOP
    DJNZ R1, D1
    DJNZ R2, D2
    RET

END
```

a) Led dịch sang trái từ (p2.0 đến P2.7)

Xuất port2 (0: sáng , 1 tắt)

Ban đầu: P2.0 = 0 , P2.1 -> P2.7 = 1

A = (A << 1 || A >> 7)

```
ORG 0000H

BEGIN_ENTRY:
    MOV A, #0FEH     ; 1111 1110  LED P2.0 sáng

MAIN_LOOP:
    MOV P2, A
    LCALL DELAY1S

    RL A

    CJNE A, #0FFH, CONT ; ss P2.7 = 1 -> quay lại P2.0
    MOV A, #0FEH

CONT:
    SJMP MAIN_LOOP
DELAY1S:

    MOV R3, #5
L1:
    LCALL DELAY200ms
    DJNZ R3, L1
    RET

DELAY200ms:
    MOV R2, #200
L2:
    MOV R1, #200
```

```
L1_1:
    NOP
    NOP
    NOP
    DJNZ R1, L1_1
    DJNZ R2, L2
    RET

END
```

b) Led tắt dần từ (p2.0 đến P2.7)

A = (A << 1| 0x01)

```
ORG 0000H

BEGIN_ENTRY:
    MOV A, #00H     ; 0000 0000

MAIN_LOOP:

    RL A
    ORL A, #01H

    MOV P2, A
    LCALL DELAY200ms

    CJNE A, #0FFH, CONT
    MOV A, #00H

CONT:
    SJMP MAIN_LOOP

DELAY1S:
    MOV R3, #5
L1:
    LCALL DELAY200ms
    DJNZ R3, L1
    RET

DELAY200ms:
    MOV R2, #200
L2:
    MOV R1, #200
L1_1:
    NOP
    NOP
```

```
    NOP
    DJNZ R1, L1_1
    DJNZ R2, L2
    RET

END
```

c) Chớp tắt led

Đảo trạng thái : CPL

```
ORG 0000H

BEGIN_ENTRY:
    SETB P2.0
    SETB P2.1
    SETB P2.2
    SETB P2.3 ; LED tat
    SETB P2.4
    SETB P2.5
    SETB P2.6
    SETB P2.7

MAIN_LOOP:
    CPL P2.0
    CPL P2.1
    CPL P2.2
    CPL P2.3 ; LED dao
    CPL P2.4
    CPL P2.5
    CPL P2.6
    CPL P2.7
    LCALL DELAY200ms
    SJMP MAIN_LOOP

DELAY1S:
    MOV R3, #5
S1BACK:
    LCALL DELAY200ms
    DJNZ R3, S1BACK
    RET

DELAY200ms:
    MOV R2, #200

LAP2:
    MOV R1, #200
```

```
LAP1:
    NOP
    NOP
    NOP
    DJNZ R1, LAP1
    DJNZ R2, LAP2
    RET

END
```

d) Sáng dần

Led tắt dần bit theo thứ tự

```
ORG 0000H

BEGIN_ENTRY:
    MOV A, #0FFH     ; 0000 0000

MAIN_LOOP:
    MOV P2, A
    LCALL DELAY200ms

    RL A
    ANL A, #0FEH

    CJNE A, #00H, CONT
    MOV P2, #00H
    LCALL DELAY200ms
    MOV A, #0FFH


CONT:
    SJMP MAIN_LOOP
DELAY200ms:
    MOV R2, #200
L2:
    MOV R1, #200
L1_1:
    NOP
    NOP
    NOP
    DJNZ R1, L1_1
    DJNZ R2, L2
    RET

END
```

e) Led sáng dồn

Sáng dần từng led (rồi lặp lại)

Ban đầu 1111 1111

Lần 1 : led chạy

+) 0111 1111

+) 1011 1111

+) 1101 1111

+) 1110 1111

…..

+) 1111 1110 (lưu lại)

Lần 2: led chạy

+) 0111 1110

+) 1011 1110

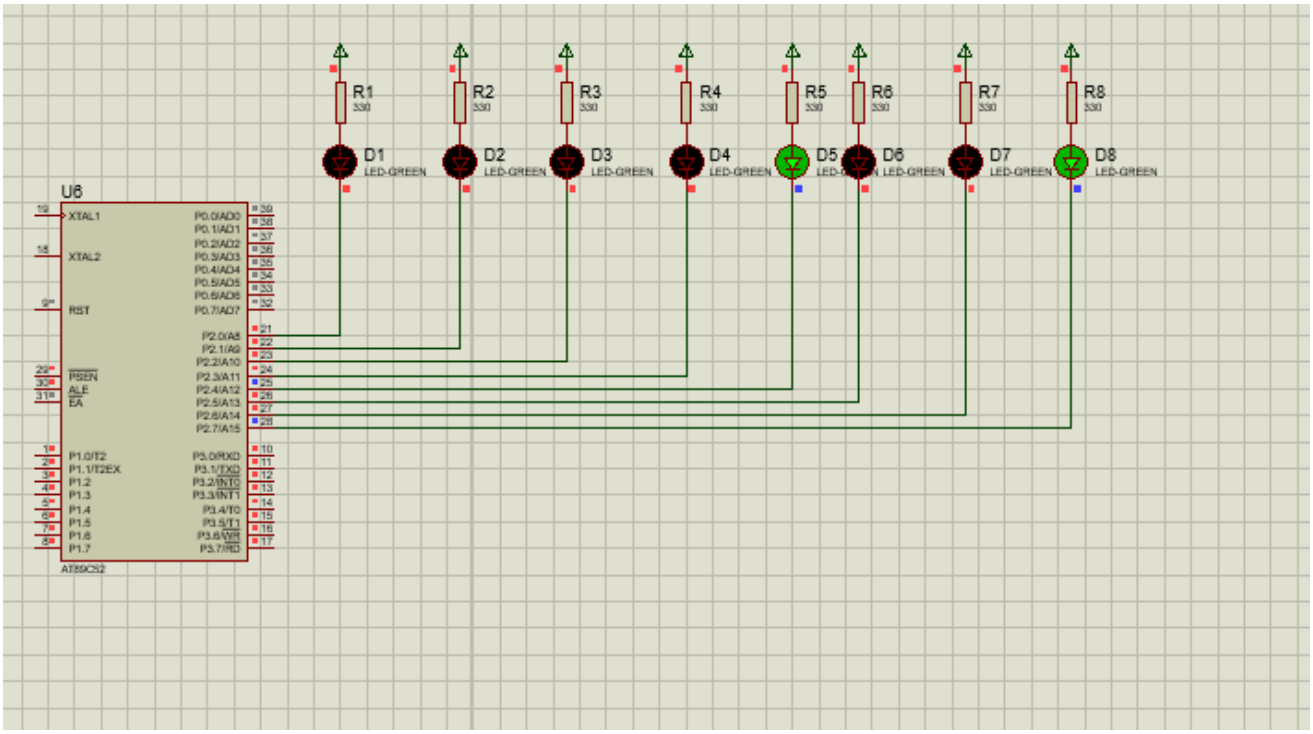
…

+) 1111 1100 (lưu lại)

Cho tới 0000 0000

```
   MASK      DATA 30H
BASE      DATA 31H

ORG 0000H

BEGIN_ENTRY:
    MOV BASE, #0FFH
    MOV MASK, BASE          ; mask = base

MAIN_LOOP:

    MOV R6, #0              ; i = 0

FOR_I:
    CJNE R6, #8, OK_I
    SJMP BEGIN_ENTRY
OK_I:

    ; base >>= 1
    MOV A, BASE
```

```
    RR A
    MOV BASE, A
    MOV MASK, A
    MOV R7, #0

FOR_J:
    MOV A, R7
    ADD A, R6
    CJNE A, #8, OK_J
    SJMP END_J
OK_J:
    MOV A, #1
    MOV B, R7
SHL_J:
    JZ SHL_DONE
    RL A
    DJNZ B, SHL_J
SHL_DONE:

    CPL A                   ; ~(1<<j)
    MOV B, A

    MOV A, MASK
    ANL A, B
    MOV P2, A
    LCALL DELAY200ms
    INC R7
    SJMP FOR_J
END_J:
    INC R6
    SJMP FOR_I
DELAY200ms:
    MOV R2, #200
DL2:
    MOV R1, #200
DL3:
    NOP
    NOP
    NOP
    DJNZ R1, DL3
    DJNZ R2, DL2
    RET
END
```



2. Viết chương trình hiển thị giờ lên 4 led 7 đoạn, dạng [mm:ss].

Chức năng	Dùng PORT nào?	Lý do
Xuất dữ liệu 7 đoạn (a,b,c,d,e,f,g,dp)	P0	P0 có 8 bit liên tục (D0→D7) phù hợp cho dữ liệu 7SEG
Chọn LED nào sáng (digit select)	P2	P2 thường dùng làm cổng điều khiển (control port) vì không cần transistor hỗ trợ
Gửi dữ liệu lớn (8 bit) → 74HC245	P0	Port 0 mạnh về xuất bus, đúng thiết kế ban đầu của 8051
Gửi tín hiệu chọn LED → 74HC138	P2	74HC138 chỉ cần 3 bit A0–A2, P2 dễ dùng hơn

Code - Led sáng 3s rồi tự tắt (Led7seg hiện số 3 rồi đếm lùi về 0 thì led đơn tắt)

```
SEGMENT_PORT EQU P0
VALUE        DATA 30H
LED_CTRL     BIT P2.0

ORG 0000H
    SJMP MAIN
MAIN:
```

```
    MOV VALUE, #03H       ; start
    CLR LED_CTRL

MAIN_LOOP:
    MOV DPTR, #TABLE
    MOV A, VALUE
    MOVC A, @A+DPTR
    MOV SEGMENT_PORT, A
    LCALL DELAY1S
    DJNZ VALUE, CONT
    MOV SEGMENT_PORT, #3Fh ; end = 0
    SETB LED_CTRL
    LCALL DELAY1S
    MOV VALUE, #03H
    CLR LED_CTRL
CONT:
    SJMP MAIN_LOOP

DELAY_200MS:
    MOV R2, #200
D1:
    MOV R1, #200
D2:
    DJNZ R1, D2
    DJNZ R2, D1
    RET
DELAY1S:
    MOV R3, #5
L1:
    LCALL DELAY_200MS
    DJNZ R3, L1
    RET

TABLE:
    DB 03FH
    DB 006H
    DB 05BH
    DB 04FH
    DB 066H
    DB 06DH
    DB 07DH
    DB 007H
    DB 07FH
    DB 06FH
END
```

3. Chạy code mẫu sau, xem kết quả hiển thị lên 8 led 7 đoạn, giải thích nghĩa chương trình:

(hình vẽ sơ đồ LED 7 đoạn và các IC 74HC138, 74HC245)

Chương trình hiển thị chạy số trên hệ thống LED 7 đoạn đa hợp (multiplexing) gồm nhiều digit (tối đa 8 digit). Quét 8 LED 7 đoạn, hiển thị số chạy từ 0 đến 7, và liên tục lặp lại

1. Khởi tạo

LED7SEG = 0 → chọn digit đầu tiên.

VALUE = 0 → hiển thị số 0.

2. Chọn digit

Dịch trái 2 bit để chuyển LED7SEG → bit chọn digit (cho 74HC138).

3. Lấy mã 7 đoạn

DPTR trỏ bảng mã 0–9.

MOVC A,@A+DPTR lấy đúng pattern 7 đoạn.

4. Xuất ra P0

P0 sẽ điều khiển các đoạn của LED.

5. Delay 2 ms

Giữ digit sáng đủ lâu để mắt thấy.

6. Tăng LED7SEG

Chọn digit tiếp theo: 0→1→2→…→7→0.

7. Tăng VALUE

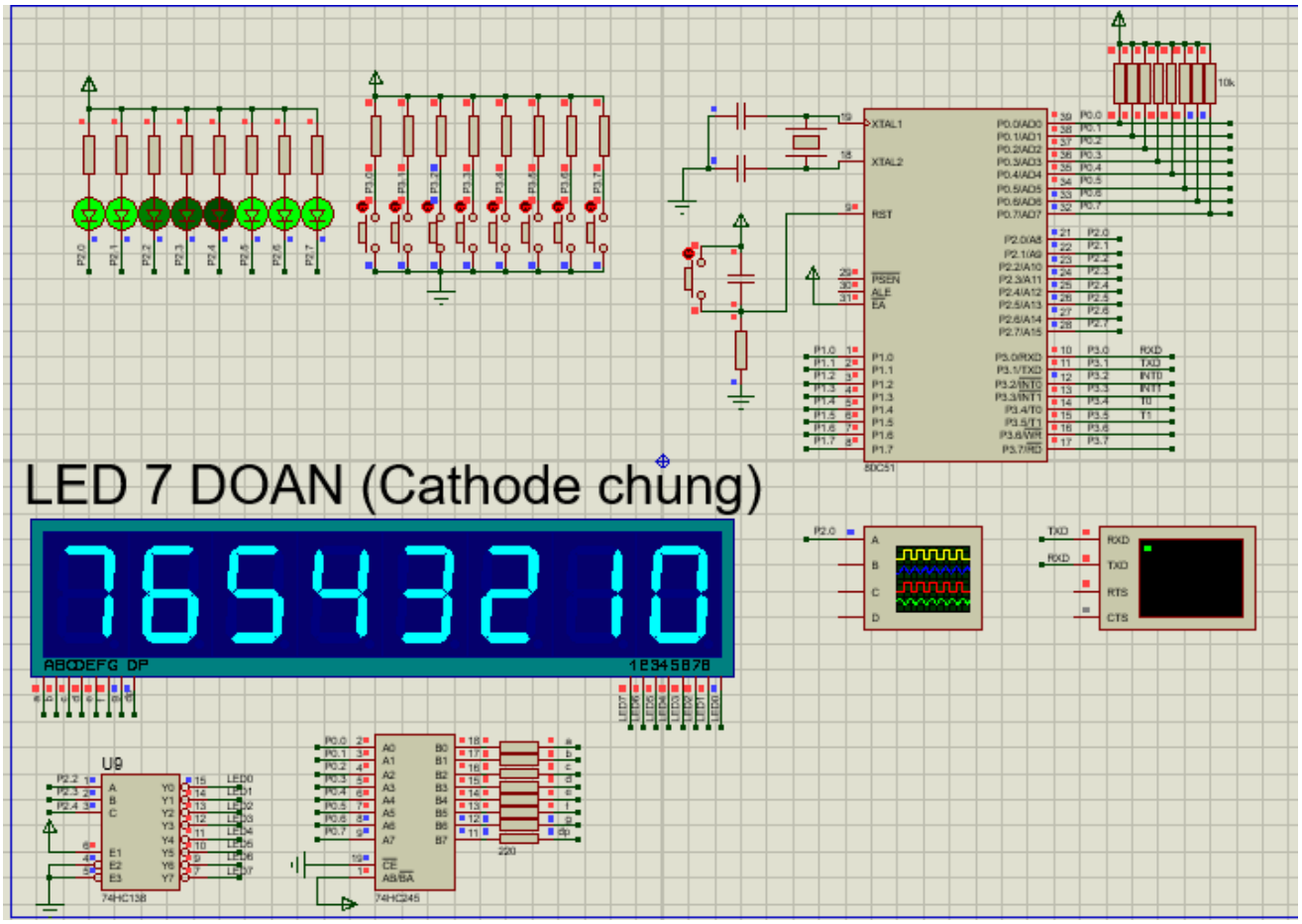
Giá trị hiển thị đổi theo: 0→1→2→…→7→0.

8. Lặp lại

Toàn bộ LED sáng liên tục theo hiệu ứng chạy số.

```
SEGMENT_PORT EQU P0
SELECT_PORT EQU P2
LED7SEG DATA 30H
VALUE DATA 31H

ORG 0000H
JMP MAIN

ORG 0030H
MAIN:
        MOV LED7SEG, #0
```

```
        MOV VALUE, #0
LOOP:
        MOV A, LED7SEG
        RL A
        RL A
        MOV SELECT_PORT, A

        MOV DPTR, #DIGIT_PATTERNS
        MOV A, VALUE
        MOVC A, @A+DPTR
        MOV SEGMENT_PORT, A
        CALL DELAY_2MS

        INC LED7SEG
        MOV A, LED7SEG
        ANL A, #07H
        MOV LED7SEG, A

        INC VALUE
        MOV A, VALUE
        ANL A, #07H
        MOV VALUE, A
        JMP LOOP

DELAY_2MS:
        MOV R2, #4
D2_OUTER:
        MOV R1, #250
D2_INNER:
        DJNZ R1, D2_INNER
        DJNZ R2, D2_OUTER
        RET

DIGIT_PATTERNS:
        DB 03Fh, 006h, 05Bh, 04Fh, 066h, 06Dh, 07Dh, 007h, 07Fh, 06Fh

END
```


LED 7 DOAN (Cathode chung)
ABCDEFG DP
74HC138
74HC245
8051

Tuần 2 – Bài 09: Timer – Counter

9.1. Chương trình mẫu

Đề bài: Viết chương trình chớp/tắt LED ở pin P2.5 với chu kỳ 1s, sử dụng thạch anh 12 MHz (Yêu cầu: sử dụng Timer0 để viết chương trình con delay 1s)

Phân tích đề bài

- Lựa chọn cấu hình: GATE0 = 0
- Tính toán giá trị khởi tạo cho TH0, TL0
- Theo phân tích ở mục 3, có thể chọn Timer hoạt động ở chế độ 16-bit (Mode 1)
- → Timer có thể tạo độ trễ tối đa < 65,5 ms

Để tạo độ trễ 1s, có thể xử lý như sau:

- Tạo độ trễ 10 ms (tạm gọi tên là DELAY_10MS)
- Sau đó đoạn chương trình con gọi chạy 100 lần DELAY_10MS → tạo được độ trễ 1s
- Khi đó, giá trị khởi tạo cho TH0, TL0 được tính toán như sau:

```
Giá trị nạp ban đầu = 65536 – (Thời gian trễ tạo × F_Timer)

Giá trị nạp ban đầu = 65536 – (10 ms × 12 MHz / 12)

Giá trị nạp ban đầu = 65536 – 10000 = 55536 = 0xDBF0

⇒ TH0 = 0xDB, TL0 = 0xF0
(TH0 chứa byte cao, TL0 chứa byte thấp)
```

Chương trình

```
ORG 0000H

MAIN:
    SETB P2.5        ; Tắt LED (LED nối VCC)
LOOP:
    CPL  P2.5        ; Đảo trạng thái LED
    CALL DELAY_1S    ; Gọi hàm trễ 1 giây
    SJMP LOOP        ; Lặp vô hạn
;============================
; Hàm DELAY 1 giây
;============================
DELAY_1S:
    MOV R7, #100     ; Gọi delay 10ms 100 lần
```

```
DELAY_LOOP:
    CALL DELAY_10MS  ; Gọi hàm trễ 10ms
    DJNZ R7, DELAY_LOOP
    RET
;============================
; Hàm DELAY 10ms dùng Timer0
;============================
DELAY_10MS:
    MOV TMOD, #01H   ; GATE0=0, Timer0 Mode 1 (16-bit)
    MOV TH0, #0DBH   ; Nạp byte cao
    MOV TL0, #0E0H   ; Nạp byte thấp
    SETB TR0         ; Bắt đầu chạy Timer0

WAIT:
    JNB TF0, WAIT    ; Đợi Timer tràn (TF0 = 1)
    CLR TR0          ; Dừng Timer
    CLR TF0          ; Xóa cờ tràn
    RET

END
```

9.2. Bài tập và câu hỏi

1. Timer khác Counter như thế nào trong 8051?

Đặc điểm	Timer	Counter
Nguồn xung đếm	Lấy từ bên trong CPU (chu kỳ máy)	Lấy từ bên ngoài thông qua chân T0 (P3.4) hoặc T1 (P3.5)
Ứng dụng	Tạo trễ (delay), đo thời gian	Đếm sự kiện bên ngoài: xung, số lần nhấn, số vật đi qua cảm biến
Cách hoạt động	Tăng 1 sau mỗi chu kỳ máy (1 Machine Cycle = 12 xung oscillator nếu dùng 12MHz → 1µs)	Tăng 1 mỗi khi có xung xuống cạnh (cạnh xuống "falling edge") ở chân T0/T1
Bit chọn chế độ	TMOD: C/T = 0	TMOD: C/T = 1
Độ chính xác phụ thuộc	Tần số thạch anh	Tần số xung ngoài
Ảnh hưởng bởi phần mềm	Không bị ảnh hưởng bởi nhiễu ngoài	Dễ bị nhiễu nếu xung ngoài không ổn định

Timer = bộ đếm thời gian, đếm theo chu kỳ máy của 8051.

Counter = bộ đếm sự kiện, đếm xung từ bên ngoài đưa vào chân T0/T1.

2. Trong Mode 1, Timer hoạt động như thế nào? Độ dài đếm là bao nhiêu bit?

Timer Mode 1 hoạt động tương tự như Timer Mode 0, nhưng khác biệt ở việc sử dụng thanh ghi 16-bit (gồm cả 8 bit của 2 thanh ghi THx và TLx). Điều này cho phép timer có thể đếm dài hơn và tạo độ trễ chính xác hơn.

Timer 16-bit (đếm từ 0000H → FFFFH)

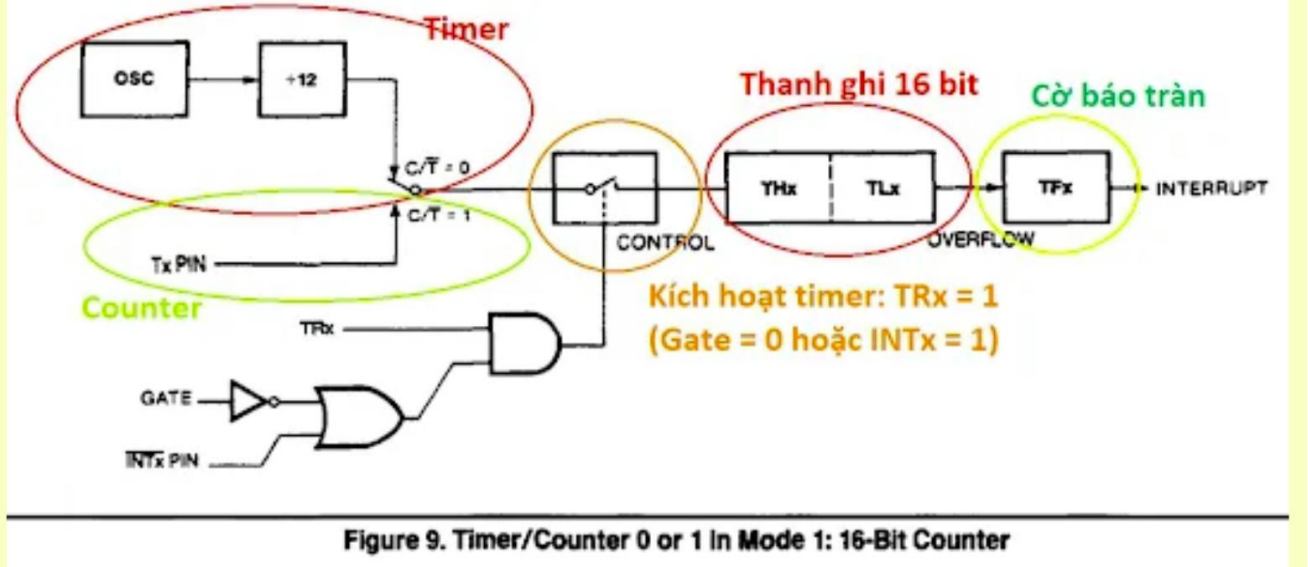


Figure 9. Timer/Counter 0 or 1 in Mode 1: 16-Bit Counter

Trong Mode 1, Timer 0/1 hoạt động ở chế độ đếm 16-bit. Timer sử dụng cặp thanh ghi TH và TL, đếm từ 0000H đến FFFFH (tối đa 65535).

Khi tràn, TF được set = 1 và timer quay về 0000H.

3. Bit nào trong thanh ghi TCON được dùng để khởi động Timer 0?

Trong thanh ghi TCON của 8051, bit được sử dụng để khởi động Timer 0 là bit TR0.

Khi TR0 được đặt lên 1 thì Timer 0 bắt đầu đếm, còn khi TR0 bằng 0 thì Timer 0 dừng lại. Bit này nằm tại vị trí TCON.4, tức là bit thứ 4 của thanh ghi TCON (địa chỉ 88H). Ngoài TR0, bộ Timer 0 còn liên quan đến bit TF0 (TCON.5), dùng để báo tràn Timer, nhưng bit điều khiển chạy/dừng Timer 0 duy nhất là TR0.

4. Tạo trễ 1ms bằng Timer 0 ở chế độ Mode 1, tần số thạch anh 12 MHz.

1 chu kỳ máy = 12 chu kỳ xung = 1 µs

Số đếm tối đa = 65536

Giá trị nạp = 65536 – số xung cần đếm = 65536 – 1000 = 64536 = 0xFC18

Mode 0 → 13 bit (hạn chế)

Mode 1 → 16 bit (chuẩn nhất để tạo delay)

Mode 2 → 8 bit auto reload (dùng cho tốc độ, baud rate)

Mode 3 → tách timer (ít dùng)

```
DELAY_1MS:
    MOV TMOD, #01H      ; Timer0 Mode1 (16-bit)
    MOV TH0, #0FCH      ; nạp giá trị FC18H
    MOV TL0, #018H
    SETB TR0            ; bật Timer0
WAIT_T0:
    JNB TF0, WAIT_T0    ; chờ TF0 = 1 (tràn)

    CLR TR0             ; tắt Timer0
    CLR TF0             ; xóa cờ tràn
    RET
```

Timer 1 Mode 1

```
DELAY_1MS:
    MOV TMOD, #10H      ; Timer1 Mode1 (16-bit)
    MOV TH1, #0FCH      ; nạp giá trị FC18H
    MOV TL1, #018H
    SETB TR1            ; bật Timer0
WAIT_T0:
    JNB TF0, WAIT_T1    ; chờ TF0 = 1 (tràn)

    CLR TR1             ; tắt Timer0
    CLR TF1             ; xóa cờ tràn
```

5. Viết chương trình nhấp nháy LED trên P1.0, với chu kỳ 500ms sử dụng Timer 1.

Fosc = 12 MHz → chu kỳ máy = 1 µs → Timer tăng mỗi 1 µs

500 ms = 500 000 µs -> 500 / 50 = 10 lần 50ms

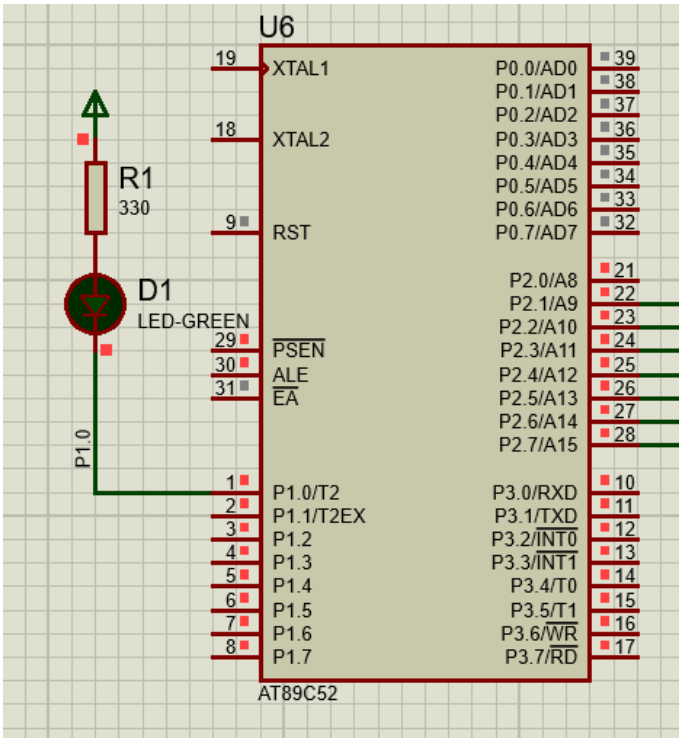
16 bit tối đa 65536 µs = 65.5 ms -> Ta chọn delay = 50 ms mỗi lần

Giá trị nạp : 65536 - 50000 = 15536 = 0x3CB0

```
ORG 0000H
MAIN:
    CLR P1.0            ; LED OFF
LOOP:
    CPL P1.0            ; dao
    LCALL DELAY500MS
    SJMP LOOP

DELAY500MS:
```

```
    MOV R7,#10          ; 10 × 50ms = 500ms

DELAY_LOOP:
    MOV TMOD,#10H       ; Timer1 Mode1 (16-bit)
    MOV TH1,#03CH
    MOV TL1,#0B0H
    SETB TR1            ; bat Timer1

WAIT_T1:
    JNB TF1,WAIT_T1     ; tràn

    CLR TR1             ; tat Timer1
    CLR TF1             ; xóa co tràn

    DJNZ R7,DELAY_LOOP
    RET

END
```

Led chớp theo yêu cầu:



Tuần 3 – Bài 10: Ngắt

10.1. Chương trình mẫu

1. Viết chương trình đảo trạng thái led ở Pin 2.7, sử dụng chân nối ở P3.2 (ngắt ngoài)

Ngắt ngoài:

- Cấu hình kiểu ngắt(cạnh hoặc mức) thông qua bit IT0/IT1 trong TCON
- Cho phép ngắt nút nhất hoạt động EX0, EX1
- Cho phép ngắt toàn cục (EA = 1)
- Viết chương trình xử lý ngắt ISR

```
ORG 0000H
    SJMP MAIN

ORG 0003H
    JMP INT0_ISR

MAIN:
    SETB IT0
    SETB EX0
    SETB EA

    SETB P2.7
LOOP:
    JMP LOOP
INT0_ISR:
    CPL P2.7
    RETI
END
```

2. Chương trình nhấp nhát led P2.1 với tần số 1Hz (chu kì 1s). Yêu cầu sử dụng ngắt timer 0, mod 1. Thời gian trễ 500ms = 10 * 50ms

```
ORG 0000H
    SJMP MAIN

ORG 000BH
    JMP TIMER0_ISR
ORG 0030H
MAIN:
    MOV TMOD, #01H
    MOV TH0, #3CH
    MOV TL0, #0B0H
    MOV R7, #10
```

```
    SETB ET0
    SETB EA
    SETB P2.1
    SETB TR0
LOOP:
    JMP LOOP
TIMER0_ISR:
    MOV TH0, #3CH
    MOV TL0, #0B0H
    DJNZ R7, EXIT_ISR
    MOV R7, #10
    CPL P2.1

EXIT_ISR:
    RETI
END
```

10.2. Bài tập và câu hỏi

1. Ngắt ngoài (External Interrupt) INT0/INT1 hoạt động theo cạnh hay mức?

Cấu hình như thế nào?

- INT0 (P3.2) và INT1 (P3.3) của 8051 có thể hoạt động theo hai chế độ :
- Level-triggered (mức thấp – active low)
- Edge-triggered (cạnh xuống – falling edge)

Tính năng	INT0 (P3.2)	INT1 (P3.3)
Chế độ mức	IT0 = 0	IT1 = 0
Chế độ cạnh xuống	IT0 = 1	IT1 = 1
Bật ngắt	EX0 = 1	EX1 = 1

2. Khi xảy ra ngắt, trình xử lý sẽ làm gì? Lệnh nào giúp trở lại chương trình chính?

- Hoàn thành lệnh đang chạy
- Lưu PC vào stack
- Nhảy đến vector ngắt
- Chạy ISRLệnh RETI để trở lại chương trình chính

3. Bit nào trong thanh ghi IE (Interrupt Enable) và TCON dùng để bật ngắt ngoài INT0?

Để bật được ngắt INT0, cần 2 bit trong IE và 1 bit trong TCON.

Thanh ghi	Bit	Tên	Chức năng

IE	EX0 (IE.0)	Enable external interrupt 0	Bật ngắt INT0
IE	EA (IE.7)	Enable all interrupts	Cho phép toàn bộ ngắt
TCON	IT0 (TCON.0)	Chọn kiểu kích hoạt	Cạnh/mức cho INT0

4. Làm sao để cấu hình và sử dụng ngắt Timer?

- Cấu hình chế độ hoạt động của Timer qua thanh ghi TMOD
- Tính toán và nạp giá trị khởi tạo vào THx và TLx
- Cho phép ngắt Timer qua thanh ghi IE
- Khởi động Timer bằng cách set bit TRx
- Viết chương trình ngắt timer

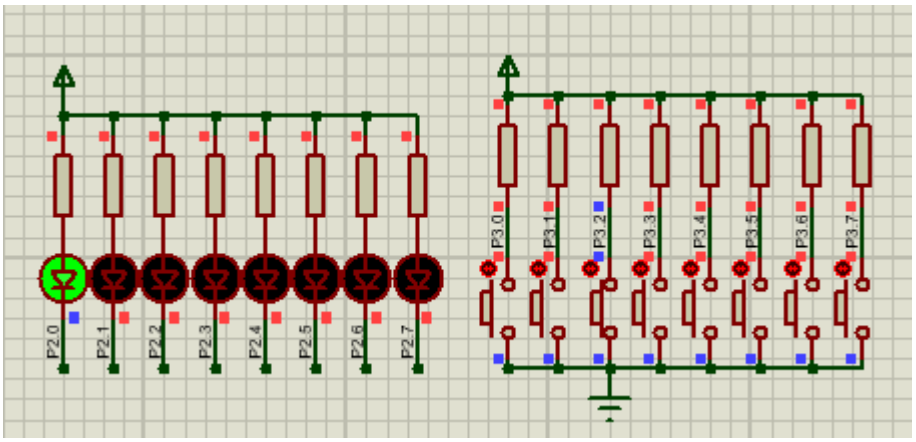
5. Viết chương trình sử dụng cả INT0 và INT1 để điều khiển hai LED:

- INT0 bật LED tại P2.0
- INT1 tắt LED tại P2.0
- INT1 tắt LED tại P2.0

Ngắt ngoài INT0 (P3.2) → bật LED tại P2.0

Ngắt ngoài INT1 (P3.3) → tắt LED tại P2.0

```
ORG 0000H
    SJMP MAIN
ORG 0003H
    LJMP INT0_ISR
ORG 0013H
    LJMP INT1_ISR
MAIN:
    SETB EA
    SETB IT0
    SETB EX0
    SETB IT1
    SETB EX1
    SETB P2.0
MAIN_LOOP:
    SJMP MAIN_LOOP
INT0_ISR:
    CLR P2.0            ; 0 : sang
    RETI
INT1_ISR:
    SETB P2.0           ;1: tat
    RETI
END
```



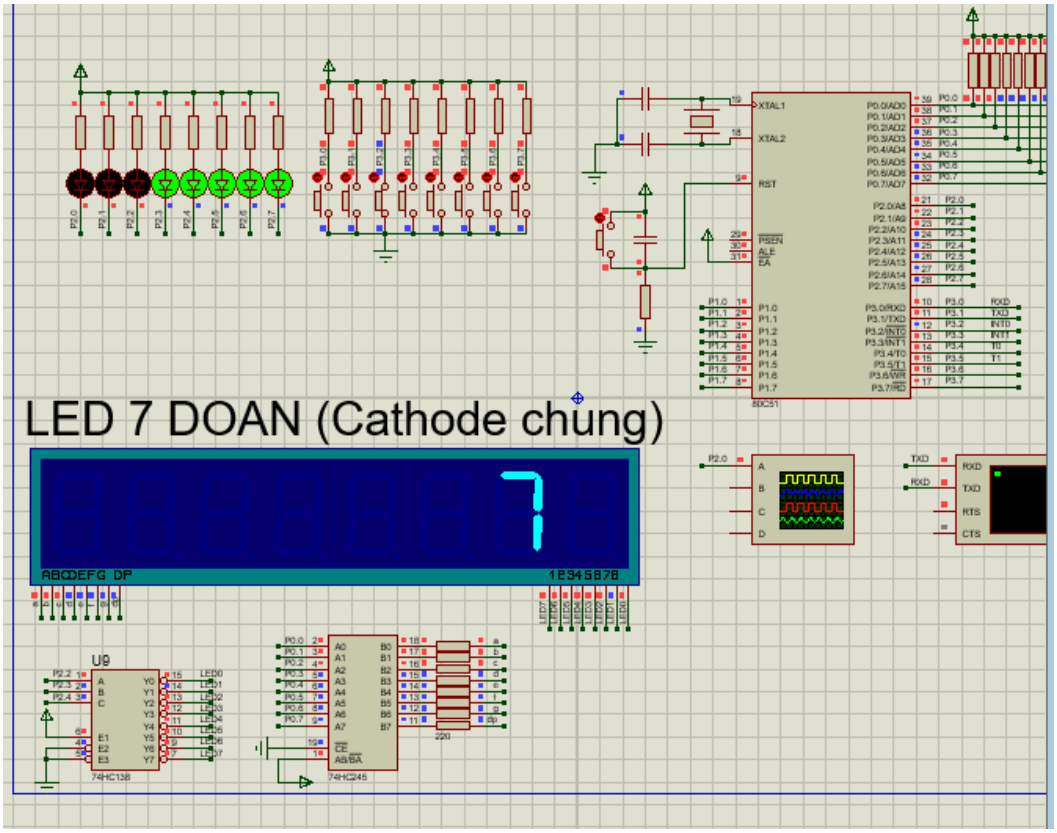
6. Viết chương trình dùng ngắt ngoài INT0 để tăng biến đếm và hiển thị trên port P2.

Mỗi lần nhấn INT0 (P3.2), biến đếm tăng 1

Giá trị biến đếm xuất ra Port P2

```
SEGMENT_PORT EQU P0
VALUE        DATA 30H

ORG 0000H
    SJMP MAIN
ORG 0003H
    LJMP INT0_ISR

MAIN:
    MOV VALUE, #00H
    SETB IT0
    SETB EX0
    SETB EA

MAIN_LOOP:
    SJMP MAIN_LOOP
INT0_ISR:
    INC VALUE
    MOV A, VALUE
    CJNE A, #0AH, DISPLAY
    MOV VALUE, #00H
DISPLAY:
    MOV DPTR, #TABLE
    MOV A, VALUE
    MOVC A, @A+DPTR
    MOV SEGMENT_PORT, A
```

```
    MOV P2, VALUE
    RETI

TABLE:
    DB 03FH
    DB 006H
    DB 05BH
    DB 04FH
    DB 066H
    DB 06DH
    DB 07DH
    DB 007H
    DB 07FH
    DB 06FH

END
```



7. Viết chương trình dùng ngắt Timer 1 để nhấp nháy LED P2.0 mỗi 3 giây

Timer 1 chajy mode 1 – 16 bit

Chojn 10ms tạo 1 lần ngắt nên -> 3 giây 300 lần

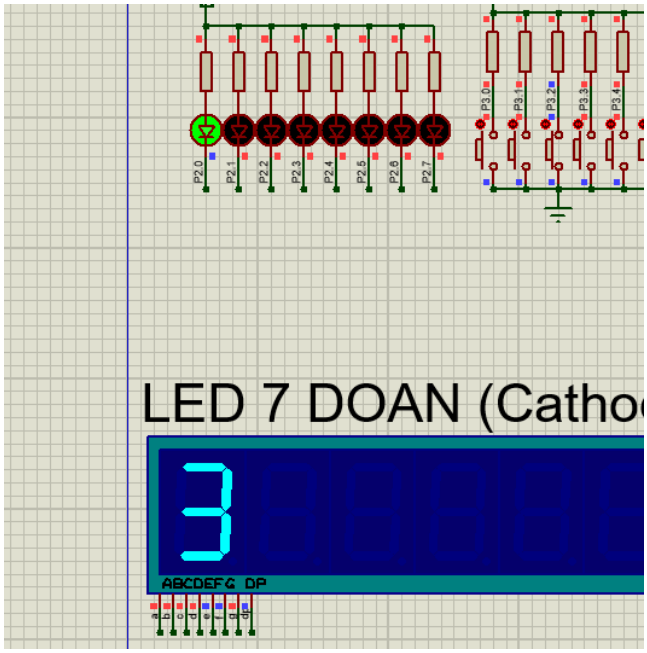
Timer Mode 1 (16-bit): Đếm 0 → 65535

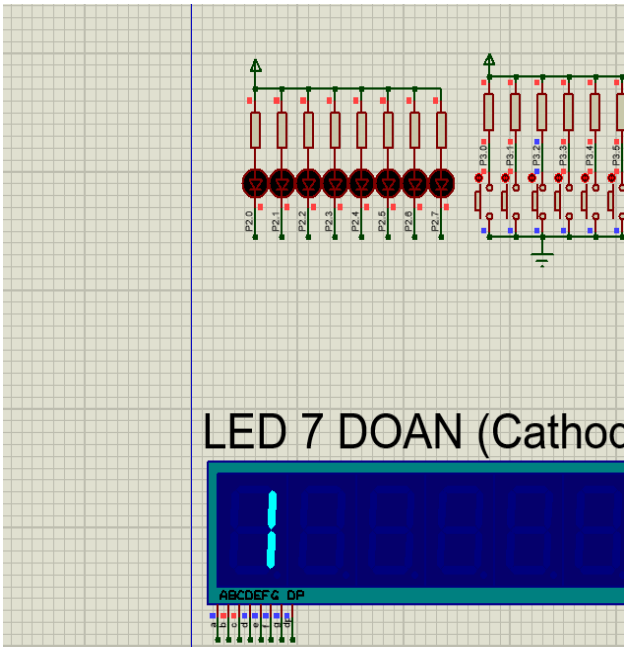
Giá trị nạp 65536 – 10000 = 55536 = D8F0h -> TH1 = D8h , TL1 = F0h

```
SEGMENT_PORT EQU P0
VALUE        DATA 30H
SEC         DATA 31H

ORG 0000H
    SJMP MAIN
ORG 001BH
    LJMP TIMER1_ISR
MAIN:
    MOV VALUE, #00H
    MOV SEC,   #00H

    MOV TMOD, #10H    ; 10ms
    MOV TH1, #0D8H
    MOV TL1, #0F0H

    SETB ET1
    SETB EA
    SETB TR1

    SETB P2.0


MAIN_LOOP:
    SJMP MAIN_LOOP

TIMER1_ISR:
    MOV TH1, #0D8H
    MOV TL1, #0F0H

    INC VALUE
    MOV A, VALUE
    CJNE A, #100, DISPLAY
    MOV VALUE, #00H

    INC SEC
    MOV A, SEC
    CJNE A, #4, DISPLAY
    MOV SEC, #00H

    CPL P2.0
```

```
DISPLAY:
    MOV DPTR, #TABLE
    MOV A, SEC
    MOVC A, @A+DPTR
    MOV SEGMENT_PORT, A
    RETI

TABLE:
    DB 03FH
    DB 006H
    DB 05BH
    DB 04FH
    DB 066H
    DB 06DH
    DB 07DH
    DB 007H
    DB 07FH
    DB 06FH

END
```

Mô phỏng protues





Phần 1.2: Lập trình C

Tuần 4 – Bài 11: Lập trình GPIO sử dụng ngôn ngữ C

11.1. Chương trình mẫu

1. Bật tắt led

```
#include<REGX51.H>

void delay(unsigned int time) {
        unsigned int i , j;
        for (i = 0 ; i < time; i++)
                for (j = 0 ; j < 123; j++);
}
void main(){
    while(1){
        P1 = 0x00; // n=bat led
        delay(500);
        P1 = 0xFF; // tat led
        delay(500);
    }
}
```

2. Chớp tắt led đơn pin P2.0 với chu kỳ 1s

```
#include <REGX51.H>

sbit LED0 = P2^0;

void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 123; j++);
}

void main() {
    while (1) {
        LED0 = 0;
        delay_ms(500);
        LED0 = 1;
        delay_ms(500);
    }
}
```

3. Viết hiệu ứng dịch 1 đốm sáng chạy qua 8 led đơn. Thời gian trễ 200ms.

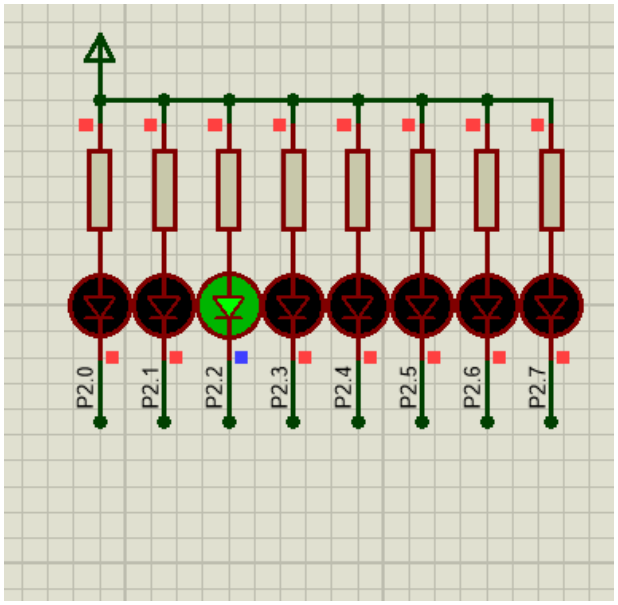
```
#include <REGX51.H>

void delay_ms(unsigned int ms) {
    unsigned int i, j;
      for (i = 0; i < ms; i++)
       for (j = 0; j < 123; j++);
}

void main() {
    while (1) {

        P2 = 0xFE;   // 1111 1110 - LED bit 0 sáng
        delay_ms(200);

        P2 = 0xFD;   // 1111 1101 - LED bit 1 sáng
        delay_ms(200);

        P2 = 0xFB;   // 1111 1011 - LED bit 2 sáng
        delay_ms(200);
        P2 = 0xF7;   // 1111 0111 - LED bit 3 sáng
        delay_ms(200);

        P2 = 0xEF;   // 1110 1111 - LED bit 4 sáng
        delay_ms(200);

        P2 = 0xDF;   // 1101 1111 - LED bit 5 sáng
        delay_ms(200);

        P2 = 0xBF;   // 1011 1111 - LED bit 6 sáng
        delay_ms(200);
        P2 = 0x7F;   // 0111 1111 - LED bit 7 sáng
        delay_ms(200);
    }
}
```

4. Điều khiển độ sáng của LED ở pin P2.2 bằng PWM

```
include <REGX51.H>
#include <intrins.h>
sbit LED = P2^2;

 void delay_us(unsigned int us) {
     while (us--) {
         _nop_();      // mất 1µs nếu chạy ở 12MHz
     }
 }
```

```
void main() {
    unsigned char i, duty_cycle;
    while (1) {
        for (duty_cycle = 0; duty_cycle < 100; duty_cycle++) {

            for (i = 0; i < 100; i++) {
                if (i < duty_cycle)
                    LED = 0;   // Bật LED (active low)
                else
                    LED = 1;   // Tắt LED
            }
            delay_us(10);   // Điều chỉnh độ dài xung PWM (~1ms chung kỳ)
        }
        for (duty_cycle = 100; duty_cycle > 0; duty_cycle--) {

            for (i = 0; i < 100; i++) {
                if (i < duty_cycle)
                    LED = 0;
                else
                    LED = 1;
            }
            delay_us(10);
        }
    }
}
```

Mô phỏng



5. Chương trình bật/tắt led đơn bằng 2 nút nhất (sử dụng phương pháp polling)

```
#include <REGX51.H>

sbit LED     = P2^7;
sbit BTN_ON  = P3^0;
sbit BTN_OFF = P3^1;

void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 123; j++);   // delay 1ms
}

void main() {
    LED = 1;
    BTN_ON  = 1;
    BTN_OFF = 1;
    while (1) {
        if (BTN_ON == 0) {
            delay_ms(20);
            if (BTN_ON == 0) {
                LED = 0;
                while (BTN_ON == 0);
            }
        }

        if (BTN_OFF == 0) {
            delay_ms(20);
            if (BTN_OFF == 0) {
                LED = 1;
                while (BTN_OFF == 0);
            }
        }
    }
}
```

6. Chương trình bật/tắt led đơn bằng 2 nút nhất (sử dụng phương pháp interrupt)

```
#include <REGX51.H>

sbit LED = P2^7;

// INT0 (P3.2) tat
 void external0_isr(void) interrupt 0 {
     LED = 1;
 }
```

```
// INT1 (P3.3) bat
void external1_isr(void) interrupt 2 {
    LED = 0;
}

void main(void) {
    IT0 = 1;
    IT1 = 1;
    EX0 = 1;
    EX1 = 1;
    EA  = 1;

      LED = 1;
        P3_2 = 1;
      P3_3 = 1;
    while (1) {
    }
}
```

11.2. Bài tập và câu hỏi

1. Ngắt là gì? Ưu điểm của việc sử dụng ngắt thay vì polling là gì?

Ưu điểm của phương pháp interrupt:

- Phản ứng ngay lập tức với sự kiện
- Tiết kiệm tài nguyên CPU
- Phù hợp với các hệ thống phức tạp và tiêu thụ điện năng thấp
- Có thể kết hợp với chế độ ngủ của vi điều khiển

Nhược điểm của phương pháp interrupt:

- Phức tạp hơn trong cấu hình và triển khai
- Cần chú ý đến vấn đề chia sẻ dữ liệu giữa ISR và chương trình chính
- Khó debug hơn

So sánh ưu/nhược điểm giữa: Polling và Interrupt

Tiêu chí	Polling	Interrupt
Độ phức tạp	Thấp	Cao
Sử dụng CPU	Cao	Thấp
Thời gian phản hồi	Có thể chậm	Tức thì

Tiêu thụ năng lượng	Cao	Thấp
Phù hợp với	Ứng dụng đơn giản	Ứng dụng phức tạp hoặc tiêu thụ thấp
Xử lý nhiều đầu vào	Khó khăn	Dễ dàng

2. Trong lập trình C cho 8051, hàm ISR (Interrupt Service Routine) được khai báo như thế nào?

Các bước để tạo và xử lý ngắt

- Cấu hình kiểu kích hoạt ngắt (cạnh hoặc mức) thông qua các bit IT0/IT1 trong TCON. (Thông thường, chọn ngắt theo cạnh xuống IT1 = 1)
- Cho phép ngắt nút nhấn hoạt động (EX0, EX1).
- Cho phép ngắt toàn cục (EA = 1).
- Viết chương trình xử lý ngắt (ISR).

Vector ngắn 8051

Nguồn ngắt	**Pin**	**Cờ báo**	**Xóa cờ**	**Vector**
Reset	9	RST	Tự động	—
Ngắt ngoài 0 (INT0)	12 (P3.2)	IE0	Tự động	0
Ngắt Timer 0 (TF0)	—	TF0	Tự động	1
Ngắt ngoài 1 (INT1)	13 (P3.3)	IE1	Tự động	2
Ngắt Timer 1 (TF1)	—	TF1	Tự động	3
Ngắt Serial COM (TI và RI)	—	RI, TI	Bằng phần mềm	4

3. Viết chương trình sử dụng ngắt ngoài INT1 (P3.3) để tăng biến đếm mỗi khi nhấn nút, hiển thị ra cổng P2.

```
#include <REGX51.H>

sbit LED = P2^0;
unsigned char count = 0;

 // INT1 (P3.3) bat
void external1_isr(void) interrupt 2 {
    if (P3^3 == 0) {
        count++;
        P2 = count;
        while (P3^3 == 0);

    }
}
```

```
void main(void) {
    IT1 = 1;
    EX1 = 1;
    EA  = 1;
    P2 = 0;
    while (1) {
    }
}
```

Tuần 5 – Bài 12: Lập trình LED 7 SEG

12.1. Chương trình mẫu

1. Đề bài: Viết chương trình hiển thị số 3 lên LED 7 đoạn thứ 6, thì LED 7 đoạn sẽ tương ứng các chỉ số từ 0 đến 7 (LED thứ 0 nằm bên phải cùng).

Trình tự xử lý bài toán

- Bước 1: Lựa chọn LED 7-seg muốn hiển thị.
- Bước 2: Xuất mã tương ứng ra các chân a, b, c, d, e, f, g.

```
#include <REGX51.H>
#define SEGMENT_PORT P0
#define SELECT_PORT  P2
unsigned char digit_patterns[] = {
    0x3F,   // 0
    0x06,   // 1
    0x5B,   // 2
    0x4F,   // 3
    0x66,   // 4
    0x6D,   // 5
    0x7D,   // 6
    0x07,   // 7
    0x7F,   // 8
    0x6F    // 9
};

void main(void)
{
    unsigned char led_index = 8;
    SEGMENT_PORT = 0x00;
        //SELECT_PORT = (led_index) << 2;
    SELECT_PORT = (led_index & 0x07) << 2;
    SEGMENT_PORT = digit_patterns[5];
    while (1){
    }
}
```

2. Đề bài: Viết chương trình xuất các số từ 0 đến 7 lên 8 led 7 đoạn (dùng phương pháp quét led). Thời gian quét là 2 ms

```
#include <REGX51.H>
#define SEGMENT_PORT P0
#define SELECT_PORT  P2
```

```
unsigned char digit_patterns[] = {
    0x3F,   // 0
    0x06,   // 1
    0x5B,   // 2
    0x4F,   // 3
    0x66,   // 4
    0x6D,   // 5
    0x7D,   // 6
    0x07,   // 7
    0x7F,   // 8
    0x6F    // 9
};
void main(void)
{
    unsigned char led_index = 8;
    SEGMENT_PORT = 0x00;
        //SELECT_PORT = (led_index) << 2;
    SELECT_PORT = (led_index & 0x07) << 2;
    SEGMENT_PORT = digit_patterns[5];
    while (1) {
    }
}
```

12.2. Bài tập và câu hỏi

1. Viết mã giả (pseudocode) trình bày quá trình hiển thị số 1 → 8 lên 8 led 7 đoạn bằng phương pháp dịch.

```
Bắt đầu

Khai báo:
    SEG_CODE[8]  // Mảng mã LED 7 đoạn cho các số 1 → 8
    i, bit       // Biến đếm
    DATA_PIN     // Chân dữ liệu (ví dụ P1.0)
    CLK_PIN      // Chân xung clock (ví dụ P1.1)
    LATCH_PIN    // Chân chốt dữ liệu (ví dụ P1.2)

Khởi tạo:
    Cấu hình DATA_PIN, CLK_PIN, LATCH_PIN là OUTPUT
    Gán mã hiển thị cho SEG_CODE[1] → SEG_CODE[8]

Vòng lặp vô hạn:
    Cho i chạy từ 1 đến 8:
        Lấy mã hiển thị = SEG_CODE[i]

        Cho bit chạy từ 0 đến 7:
```

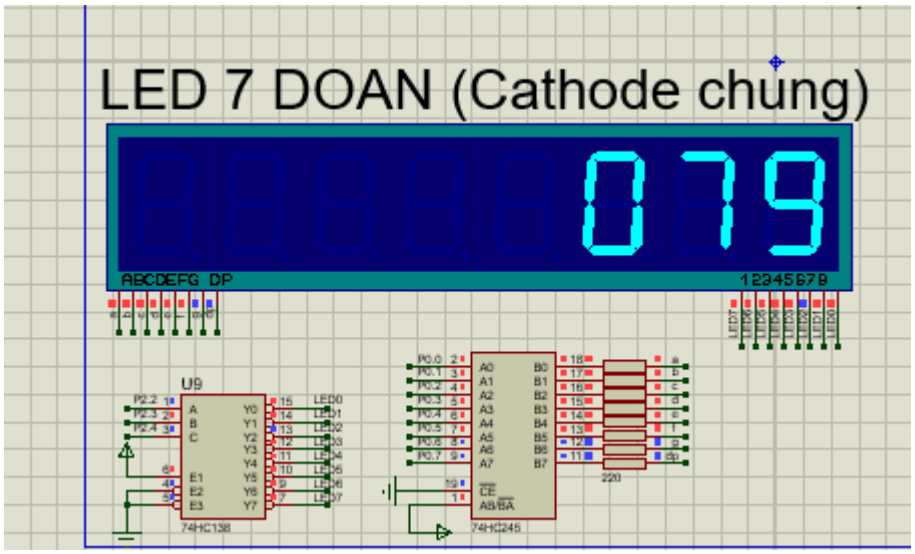
```
            Xuất bit hiện tại của mã hiển thị ra DATA_PIN
            Tạo xung CLK_PIN (0 → 1 → 0) để dịch bit
        Tạo xung LATCH_PIN để chốt dữ liệu ra LED 7 đoạn
        Tạo độ trễ để quan sát LED
    Kết thúc vòng for
Quay lại vòng lặp vô hạn
Kết thúc
```

2. Viết chương trình đếm từ 000 đến 999 và lặp lại, hiển thị lên 3 led 7 đoạn.

```
#include <REGX51.H>

#define SEGMENT_PORT P0
#define SELECT_PORT  P2

void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 123; j++);
}
unsigned char digit_patterns[] = {
    0x3F,   // 0
    0x06,   // 1
    0x5B,   // 2
    0x4F,   // 3
    0x66,   // 4
    0x6D,   // 5
    0x7D,   // 6
    0x07,   // 7
    0x7F,   // 8
    0x6F    // 9
};

unsigned char value1 =  0x00;
unsigned char value2 = 0x00;
unsigned char value3 = 0x00;
void main(void)
{
    SEGMENT_PORT = 0x00;
    while (1)
    {
                SELECT_PORT = (0 << 2);
                SEGMENT_PORT = digit_patterns[value1];
                delay_ms(2);
                SELECT_PORT = (1 << 2);
                SEGMENT_PORT = digit_patterns[value2];
                delay_ms(2);
```

```
                SELECT_PORT = (2 << 2);
                SEGMENT_PORT = digit_patterns[value3];
                delay_ms(2);
                value1++;
        if(value1 == 0x09)
        {
            value1 = 0x00;
            value2++;
            if(value2 == 0x09)
                        {
                            value2 = 0x00;
                            value3++;
                            if(value3 == 0x00)
                                value3 = 0x09;
                            }
        }
        delay_ms(20);
    }
}
```

Mô phỏng proteus:



3. Bài kiểm tra

3 chế độ led: sáng dần trái phải x8, tắt dần phải trái x8, nháy x8

1 nút chuyển chế độ:

- mode 0: lặp lại 3 cái liên tục
- mode 1,2,3: cho từng chế độ riêng

2 led 7 đoạn hiển thị mode và số led sáng

```
#include <REGX51.H>

#define SEGMENT_PORT P0
#define SELECT_PORT  P2
unsigned char cnt = 0;

void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 123; j++);
}

unsigned char digit_patterns[] = {
    0x3F,   // 0
    0x06,   // 1
    0x5B,   // 2
    0x4F,   // 3
    0x66,   // 4
    0x6D,   // 5
    0x7D,   // 6
    0x07,   // 7
    0x7F,   // 8
    0x6F    // 9
};

unsigned char pattern_led[] = {
    0xFF,0xFE,0xFC,0xF8,0xF0,0xE0,0xC0,0x80, 0x00
};
void display_2digit(unsigned char d1, unsigned char d2)
{
    SELECT_PORT = (0 << 2);
    SEGMENT_PORT = digit_patterns[d1];
    delay_ms(1);

    SELECT_PORT = (7 << 2);
    SEGMENT_PORT = digit_patterns[d2];
    delay_ms(1);
}

void effect_sangdan()
{
    unsigned char i;
    for(i=0;i<9;i++){
        P1 = pattern_led[i];
                display_2digit(i,cnt);
```

```
        delay_ms(200);
    }
}

void effect_tatdan()
{
    char i;
    for(i=8; i >= 0; i--){
        P1 = pattern_led[i];
        display_2digit(i,cnt);
        delay_ms(200);
    }
}

void effect_choptat()
{
    unsigned char i;
    for(i=0;i<8;i++){
        P1 = 0x00;
        display_2digit(0,cnt);
        delay_ms(200);

        P1 = 0xFF;
        display_2digit(8,cnt);
        delay_ms(200);
    }
}
void scan_led()
{
    effect_sangdan();
    effect_tatdan();
    effect_choptat();
}

// P3^2
void external0_isr(void) interrupt 0 {
     cnt++;
     if (cnt == 4) cnt = 0;
}

void main()
{
    IT0 = 1;
    EX0 = 1;
    EA  = 1;
```

```
    while (1)
    {
                if(cnt == 0) scan_led();
                if(cnt == 1) effect_sangdan();
                if(cnt == 2) effect_tatdan();
                if(cnt == 3) effect_choptat();
    }
}
```

Tuần 5 – Bài xx: Timer/Counter

Đề bài: Viết chương trình chớp/tắt LED ở pin P2.5 với chu kỳ 1s, sử dụng thạch anh 12 MHz (Yêu cầu: sử dụng Timer0 để viết chương trình con delay 1s)

```
#include <REGX51.H>

sbit LED = P2^5;
void delay_10ms(void)
{
    TMOD = 0x01;
    TH0  = 0xDB;
    TL0  = 0xE0;

    TR0 = 1;
    while (TF0 == 0);

    TR0 = 0;
    TF0 = 0;
}
void delay_1s(void)
{
    unsigned char i;
    for (i = 0; i < 100; i++)
    {
        delay_10ms();
    }
}
void main(void)
{
    LED = 1;
    while (1)
    {
        LED = ~LED;
        delay_1s();
    }
}
```

Đề 2: Hiển thị giá trị đếm 0–999 trên LED 7 đoạn bằng phương pháp quét (multiplexing) sử dụng Timer0 và ngắt trên vi điều khiển 8051

```
#include <REGX51.H>
// Led7seg definition
#define SEGMENT_PORT P0   // Connect Led7seg
#define SELECT_PORT  P2   // Choosing Led7seg

// Led7seg(Cathode chung)
unsigned char digit_patterns[] = {
    0x3F, // 0
    0x06, // 1
    0x5B, // 2
    0x4F, // 3
    0x66, // 4
    0x6D, // 5
    0x7D, // 6
    0x07, // 7
    0x7F, // 8
    0x6F  // 9
};

// 8 values of led7seg
unsigned char led7seg[8] = {0,1,2,3,4,5,6,7};
unsigned char index   = 0;

void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 123; j++);  // delay 1ms
}

void ISR_Timer0() interrupt 1
{
        unsigned char display_code;

    TH0 = 0xF8;
    TL0 = 0x30;
// ------------------- Update LED
//      SELECT_PORT = (index << 2);
//      SEGMENT_PORT = digit_patterns[led7seg[index]];
//
//      index++;
//      index &= 0x07;
```

```
// -------------------- Update theo so luong
//      SEGMENT_PORT = 0x00;
//    if (index < 3) {
//        SELECT_PORT = (index << 2);
//        SEGMENT_PORT = digit_patterns[led7seg[index]];
//        index++;
//    } else {
//        index = 0;
//    }

// -------------------- Update co dot

        SEGMENT_PORT = 0x00;
    if (index < 3) {
        SELECT_PORT = (index << 2);

        display_code = digit_patterns[led7seg[index]];

            if (index == 1) {
            display_code |= 0x80;
        }

        SEGMENT_PORT = display_code;
        index++;
    } else {
        index = 0;
    }
}

void setup_led7seg_timer0_2ms(void)
{

      SEGMENT_PORT = 0x00;
        index = 0;

    TMOD = 0x01;
    TH0 = 0xF8;
    TL0 = 0x30;
    IE  = 0x82;
    TR0 = 1;
}

unsigned int counter = 0;
void main(void)
{
```

```
    setup_led7seg_timer0_2ms();

    while (1)
    {
        led7seg[0] = counter % 10;
        led7seg[1] = (counter / 10) % 10;
        led7seg[2] = (counter / 100) % 10;
        led7seg[3] = 0;
        led7seg[4] = 0;
        led7seg[5] = 0;
        led7seg[6] = 0;
        led7seg[7] = 0;


        counter++;
        if (counter > 999) counter = 0;

        delay_ms(300);
    }
}
```

Tuần 6 – Bài 13: Lập trình UART

Trên 8051 truyền thống:

- Pin P3.0 (RXD): chân nhận dữ liệu
- Pin P3.1 (TXD): chân truyền dữ liệu

Không bật interrupt cho timer 1

Quá trình truyền dữ liệu:

- B1. Viết dữ liệu (8 bit data) cần truyền vào thanh ghi SBUF
- B2. Vi điều khiển tự động thêm bit start, (parity nếu có) và bit stop
- B3. Dữ liệu được truyền tuần tự qua chân TXD
- B4. Khi truyền xong, cờ TI được thiết lập = 1
- B5. Phải xóa cờ TI về 0 (thủ công) để chuẩn bị cho lần truyền tiếp theo

Quá trình nhận dữ liệu:

- B1. Thiết lập bit REN = 1 để cho phép nhận dữ liệu
- B2. Khi có dữ liệu đến, vi điều khiển tự động tách bit start và bit stop
- B3. Dữ liệu (8 bit data) được lưu vào thanh ghi SBUF
- B4. Cờ RI được thiết lập = 1 thông báo đã nhận xong dữ liệu
- B5. Đọc dữ liệu từ SBUF
- B6. Phải xóa cờ RI về 0 (thủ công)

Xử lý sự kiện ngắt:

B1. Bật 2 bit cho phép ngắt:

- EA (ngắt toàn cục)
- ES (ngắt Serial)

B2. Viết đoạn chương trình xử lý ngắt:

```
void UART_ISR(void) interrupt 4 {
   ...
}
```

B3. Hàm xử lý ngắt tự động được gọi khi 1 trong 2 tình huống xảy ra:

- Truyền xong dữ liệu (cờ TI tự động lên 1)
- Nhận xong dữ liệu (cờ RI tự động lên 1)

B4. Khi cờ ngắt lên 1, bạn cần chủ động xóa về 0 bằng phần mềm.

13.1. Chương trình mẫu

1. Chương trình số 01 (không sử dụng ngắt)

Đề bài: Viết chương trình UART đơn giản cho 8051 sử dụng ngôn ngữ C. Tạo một terminal echo: mỗi ký tự nhận được từ máy tính sẽ được gửi lại kèm theo một thông báo.

Lưu ý:

- Khi sử dụng Serial, phải dùng thạch anh 11.0592 MHz (bắt buộc để Baudrate chính xác).
- SMOD = 0 mặc định, nếu muốn nhân đôi tốc độ thì đặt SMOD = 1.
- Sơ đồ kết nối:

RX ----------> RX

TX <---------- TX

GND ---------- GND

```c
#include <REGX51.H>
#include <string.h>

// (fosc = 11.0592 MHz)
void UART_Init(void) {
    SCON = 0x50;   // Mode 1, 8-bit, REN=1 (cho phép nh?n)
    TMOD &= 0x0F;  // Xóa các bit ch? d? Timer 1
    TMOD |= 0x20;  // Thi?t l?p Timer 1 ? mode 2 (auto reload)
    TH1 = 0xFD;
    TR1 = 1;       // Kích ho?t Timer 1
    TI = 1;
}
void UART_SendChar(char c) {
    while (!TI);
    TI = 0;
    SBUF = c;
}
char UART_GetChar(void) {
    while (!RI);
    RI = 0;
    return SBUF;
}
void UART_SendString(char *str) {
    while (*str) {
        UART_SendChar(*str);
        str++;
    }
```

```
}
void main(void) {
    char received_char;

    UART_Init();

    UART_SendString("8051 UART Echo Demo\r\n");
    UART_SendString("Type 1 character and Enter:\r\n");

    while (1) {
        received_char = UART_GetChar();

        UART_SendString("\rReceived Character: ");

        UART_SendChar(received_char);
        UART_SendChar('\r');
    }
}
```

5.2. Chương trình số 02 (Có sử dụng ngắt nhận)

Đề bài:

Viết chương trình UART đơn giản cho 8051 sử dụng ngôn ngữ C.

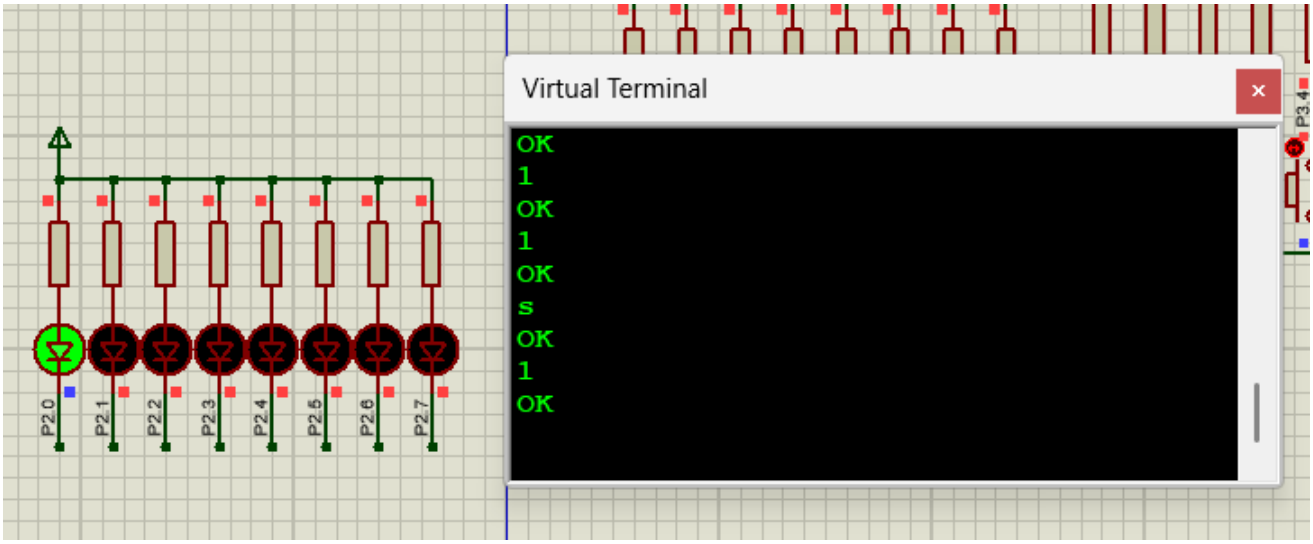
Gửi ký tự điều khiển từ Terminal (máy tính) xuống cho 8051:

- Nếu là ký tự '1' → bật Led
- Nếu là ký tự khác → tắt Led

```
#include <REGX51.H>
#include <stdio.h>
#include <string.h>

sbit LED = P2^0;

volatile bit flag = 0;
volatile unsigned char recvData; // D? li?u nh?n du?c

void UART_Init(void) {
    SCON = 0x50;   /
    TMOD &= 0x0F;  // Xóa các bit ch? d? Timer 1
    TMOD |= 0x20;
    TH1  = 0xFD;   // baud 9600, SMOD=0
    TR1  = 1;      // Kích hoat Timer 1
    TI   = 1;
}
void UART_SendChar(char c) {
```

```
    while (!TI);
    TI = 0;
    SBUF = c;
}
void UART_SendString(char *str) {
    while (*str) {
        UART_SendChar(*str);
        str++;
    }
}
void UART_EnableInterrupt(void) {
    ES = 1;
    EA = 1;
}
void UART_ISR(void) interrupt 4 {
    if (RI) {
        RI = 0;
        recvData = SBUF;
        flag = 1;
    }
}
void main(void)
{
    LED = 1;
    UART_Init();
    UART_EnableInterrupt();

    while(1)
    {
        if(flag == 1)
        {
            if(recvData == '1')
                LED = 0;
            else
                LED = 1;

            UART_SendString("\rOK\r");
            flag = 0;
        }

    }
}
```

Mô phỏng trên protues



13.2. Bài tập và câu hỏi

1. UART là gì? Tại sao cần UART trong hệ thống nhúng?

UART là chuẩn giao tiếp nối tiếp không đồng bộ gồm 2 đường TX/RX. Nó giúp vi điều khiển truyền/nhận dữ liệu với máy tính và các module ngoại vi. UART quan trọng vì đơn giản, tiết kiệm chân, dễ dùng và rất phổ biến trong hệ thống nhúng.).

UART làm nhiệm vụ chuyển đổi dữ liệu song song ↔ nối tiếp, giúp vi điều khiển giao tiếp với các thiết bị khác.

2. UART trên 8051 sử dụng Timer nào để tạo baud rate?

Trong chế độ 1, tốc độ baud được tính theo công thức:

Tốc độ baud = (2^SMOD * fosc) / (32 * 12 * (256 – TH1))

Suy ra: TH1 = 256 – (fosc * 2^SMOD) / (384 * tốc độ baud)

TH1 = 256 – (11059200 * 2) / (384 * 9600)

= 256 – 6

= 250

= 0xFA

Bảng tổng hợp tốc độ baud thông dụng cho 8051

Baud Rate	TH1, TL1 (Hex)
9600	FD
4800	FA

2400	F4
1200	E8

* Lưu ý quan trọng:

- Bắt buộc phải sử dụng Fosc = 11.0592 MHz khi dùng UART, vì chỉ tần số này giúp tính chính xác tốc độ baud.
- Để tăng tốc độ baud gấp đôi (x2), bạn set SMOD = 1 (mặc định thì SMOD = 0).

3. Các thanh ghi liên quan đến UART trong 8051 là gì? Chức năng từng thanh ghi?

- SCON (Serial Control): Cấu hình và điều khiển UART, xác định chế độ truyền (mode 0–3), bật nhận và kiểm tra cờ ngắt truyền/nhận.
- SBUF (Serial Buffer): Thanh ghi đệm dữ liệu, ghi vào để gửi qua TX, đọc ra để nhận từ RX.
- PCON (Power Control): Điều khiển nguồn và tốc độ Baud, bit SMOD dùng để nhân đôi Baud Rate.
- TMOD (Timer Mode): Cấu hình chế độ hoạt động của Timer1, thường chọn Mode 2 để tạo Baud Rate ổn định.
- TH1 / TL1: Thanh ghi của Timer1, chứa giá trị nạp ban đầu tạo tần số Baud Rate.
- TCON (Timer Control): Điều khiển bật/tắt và theo dõi cờ tràn của Timer1 (TR1, TF1).
- IE (Interrupt Enable): Cho phép ngắt UART, bit ES bật ngắt nối tiếp và EA bật ngắt toàn cục.

4. Viết chương trình tạo menu hiện lên Terminal để điều khiển 4 led đơn.

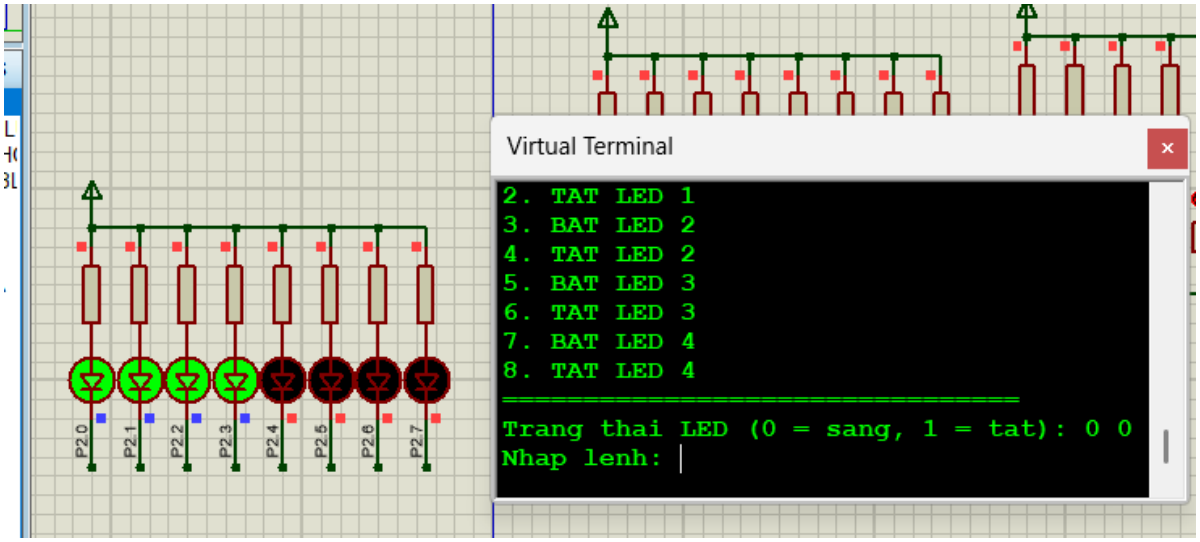
- Viết ứng dụng gửi ký tự xuống để điều khiển bật/tắt 4 led.
- Menu cập nhật lại trạng thái 4 led mỗi lần điều khiển.

```
#include <REGX51.H>
#include <stdio.h>
sbit LED1 = P2^0;
sbit LED2 = P2^1;
sbit LED3 = P2^2;
sbit LED4 = P2^3;
// Khoi tao UART, Baud 9600bps, thach anh 11.0592MHz
void UART_Init(void) {
    TMOD = 0x20;    // Timer1, mode 2 (8-bit auto reload)
    TH1  = 0xFD;    // Toc do baud 9600
    SCON = 0x50;    // UART mode 1, 8-bit, REN=1
    TR1  = 1;
```

```
        //
}
void UART_TxChar(char c) {
    SBUF = c;
    while (TI == 0);
    TI = 0;
}
void UART_SendString(char *s) {
    while (*s) {
        UART_TxChar(*s++);
    }
}
void Display_Menu(void) {
    UART_SendString("\r\n===== MENU DIEU KHIEN LED =====\r\n");
    UART_SendString("1. BAT LED 1\r\n");
    UART_SendString("2. TAT LED 1\r\n");
    UART_SendString("3. BAT LED 2\r\n");
    UART_SendString("4. TAT LED 2\r\n");
    UART_SendString("5. BAT LED 3\r\n");
    UART_SendString("6. TAT LED 3\r\n");
    UART_SendString("7. BAT LED 4\r\n");
    UART_SendString("8. TAT LED 4\r\n");
    UART_SendString("================================\r\n");
    UART_SendString("Trang thai LED (0 = sang, 1 = tat): ");
    UART_TxChar(LED1 ? '1' : '0');
    UART_TxChar(' ');
    UART_TxChar(LED2 ? '1' : '0');
    UART_TxChar(' ');
    UART_TxChar(LED3 ? '1' : '0');
    UART_TxChar(' ');
    UART_TxChar(LED4 ? '1' : '0');
    UART_SendString("\r\nNhap lenh: ");
}
void main(void) {
    char ch;
    UART_Init();
    LED1 = LED2 = LED3 = LED4 = 1;
    Display_Menu();
    while (1) {
        while (RI == 0);
        ch = SBUF;
        RI = 0;
        switch (ch) {
            case '1': LED1 = 0; break;
            case '2': LED1 = 1; break;
```

```
            case '3': LED2 = 0; break;
            case '4': LED2 = 1; break;
            case '5': LED3 = 0; break;
            case '6': LED3 = 1; break;
            case '7': LED4 = 0; break;
            case '8': LED4 = 1; break;
            default:
                UART_SendString("\r\nLenh khong hop le!\r\n");
                break;
        }
        Display_Menu();
    }
}
```

Mô phỏng proteus



5. Bài kiểm tra

```
// 1 Led blink 100ms
// Button 1 : Bật sáng trong T s -> quay về ban đầu (T = 1)
// Button 2: Cấu hình T = T + 0,5 (T = 0,5 -> 2s)
// Hiển thị T lên led7seg (ms)
#include <REGX51.H>

#define SEGMENT_PORT P0
#define SELECT_PORT  P2
sbit LED1 = P1^0;

volatile unsigned int period1 = 100;
volatile unsigned int period2 = 1000;
unsigned int c1 = 0;
unsigned int val;
```

```
void small_delay()
{
    unsigned int i;
    for(i = 0; i < 200; i++);
}
void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 123; j++);
}
void timer0_isr(void) interrupt 1
{
    TH0 = 0xFC;
    TL0 = 0x66;   // 1ms tick
    c1++;
}

void timer0_init()
{
    TMOD |= 0x01;  // Mode 1
    TH0 = 0xFC; TL0 = 0x66;
    ET0 = 1;
    TR0 = 1;
}
unsigned char digit_patterns[] = {
    0x3F,   // 0
    0x06,   // 1
    0x5B,   // 2
    0x4F,   // 3
    0x66,   // 4
    0x6D,   // 5
    0x7D,   // 6
    0x07,   // 7
    0x7F,   // 8
    0x6F    // 9
};
void display_digit(unsigned char index, unsigned char value, unsigned char dot)
{
    if (dot == 0){
      SELECT_PORT = (index << 2);
      SEGMENT_PORT = digit_patterns[value];
      delay_ms(1);
    }
    else
```

```
    {
      SELECT_PORT = (index << 2);
      SEGMENT_PORT = digit_patterns[value] | 0x80;
      delay_ms(1);
    }

}

void display4_digit(unsigned char val)
{
    val = period2;
    SELECT_PORT = (0 << 2);
    SEGMENT_PORT = digit_patterns[val%10];
    delay_ms(2);
    val = val/10;
    SELECT_PORT = (1 << 2);
    SEGMENT_PORT = digit_patterns[val%10];
    delay_ms(2);
    val = val/10;
    SELECT_PORT = (2 << 2);
    SEGMENT_PORT = digit_patterns[val%10];
    delay_ms(2);
    val = val/10;
    SELECT_PORT = (1 << 2);
    SEGMENT_PORT = digit_patterns[val];
    delay_ms(2);
}
void external0_isr(void) interrupt 0
{
    val = period2;
    display4_digit(val);
    LED1 = 0;
    delay_ms(period2);
}

void external1_isr(void) interrupt 2
{
    period2 += 500;
    if(period2 > 2000) period2 = 500;
}



void main()
{
    LED1 = 1;
```

```
    IT0 = 1;
    IT1 = 1;
    EX0 = 1;
    EX1 = 1;
    EA  = 1;
    timer0_init();
    while(1)
    {
        small_delay();
        c1++;
        // LED1
        if(c1 >= period1)
        {
            LED1 = !LED1;
            c1 = 0;
        }
        val = period2;
        display_digit(0,val%10,0);
        val = val/10;
        display_digit(1,val%10,0);
        val = val / 10;
        display_digit(2,val %10,0);
        val = val / 10;
        display_digit(3,val,0);
    }
}
```

Phần 2: Vi xử lý 8086

1. 8086 là gì? Khác gì 8051?

Bộ vi xử lý Intel 8086 là một vi xử lý (CPU/MPU) 16-bit, không phải vi điều khiển, vì nó chỉ chứa khối xử lý trung tâm mà không tích hợp sẵn bộ nhớ hay ngoại vi I/O trên chip.

Về kiến trúc, 8086 gồm ba phần chính là ALU (thực hiện các phép toán số học và logic), CU (điều khiển hoạt động của CPU theo chu trình lấy lệnh – giải mã – thực thi) và hệ thanh ghi dùng để lưu trữ dữ liệu, địa chỉ và trạng thái trong quá trình xử lý. Thanh ghi là vùng nhớ tốc độ rất cao bên trong CPU, có chức năng nạp lệnh từ bộ nhớ, theo dõi chu kỳ lệnh và tích lũy dữ liệu tạm thời. Do không có bộ nhớ nội, 8086 không thể hoạt động độc lập mà phải kết nối với RAM/ROM và các thiết bị I/O bên ngoài thông qua system bus; CPU sẽ nạp lệnh và dữ liệu từ RAM, xử lý trong ALU rồi ghi kết quả trở lại thanh ghi hoặc bộ nhớ.

Với kiến trúc 16-bit, 8086 xử lý tốt các phép cộng/trừ trên số 16-bit (tối đa 65.535) và có thể xử lý các phép nhân/chia với kết quả trung gian hoặc số bị chia lên đến 32-bit. Chức năng chính của 8086 là đọc và thực thi chương trình từ bộ nhớ, xử lý dữ liệu, quản lý truy cập bộ nhớ và giao tiếp với các thiết bị vào/ra của hệ thống.

So sánh 8086 vs 8051

Tiêu chí	8086	8051
Loại	Vi xử lý (Microprocessor)	Vi điều khiển (Microcontroller)
Độ rộng dữ liệu	16-bit	8-bit
Bộ nhớ trên chip	Không	Có
Ngoại vi tích hợp	Không	Timer, UART, GPIO
Bus địa chỉ	20-bit (1 MB)	Nhỏ, phục vụ nội bộ
Lập trình	Assembly x86	Assembly 8051 / C
Ứng dụng	PC, hệ thống tính toán	Điều khiển nhúng

2. Viết chương trình thực hiện: Cộng/trừ/nhân/chia 2 số 8 bit.

- Tạo 2 biến a,b: chứa sẵn giá trị cần tính.
- Kq lưu vào các biến: sum, sub, mul, div.

```
.MODEL SMALL
.STACK 100H

.DATA
    a   DB 20         ; số a (8-bit)
    b   DB 5          ; số b (8-bit)

    sum DB ?          ; a + b
    sub DB ?          ; a - b
    mul DW ?          ; a * b (16-bit)
    div DB ?          ; a / b (thương)

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; --------- CỘNG ---------
    MOV AL, a
    ADD AL, b
    MOV sum, AL

    ; --------- TRỪ ----------
    MOV AL, a
    SUB AL, b
    MOV sub, AL

    ; --------- NHÂN ---------
    MOV AL, a
    MOV BL, b
    MUL BL            ; AX = AL * BL
    MOV mul, AX

    ; --------- CHIA ---------
    MOV AL, a
    MOV BL, b
    XOR AH, AH        ; AH = 0 trước khi chia
    DIV BL            ; AL = a / b
    MOV div, AL

    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

3. Viết chương trình tính tổng: S = 1 + 2 + ... + n.

- Tạo biến n: chứa sẵn giá trị cần tính.
- KQ lưu vào biến: s.

```
.MODEL SMALL
.STACK 100H

.DATA
    n   DB 10          ; n cho sẵn (ví dụ n = 10)
    s   DW 0           ; tổng S (dùng 16-bit)

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    MOV CL, n          ; CL = n (biến đếm)
    MOV AX, 0          ; AX = 0 (tổng)
    MOV BL, 1          ; BL = i = 1

SUM_LOOP:
    ADD AX, BX         ; AX = AX + i
    INC BL             ; i = i + 1
    LOOP SUM_LOOP      ; lặp n lần

    MOV s, AX          ; lưu kết quả vào s

    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```