LẬP TRÌNH HỆ THỐNG NHÚNG

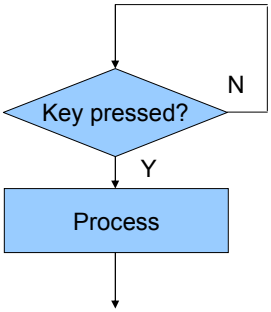
BÙI QUỐC BẢO

Delay sử dụng vòng lặp lệnh

```
For(i=1000;i>0;i--);
```

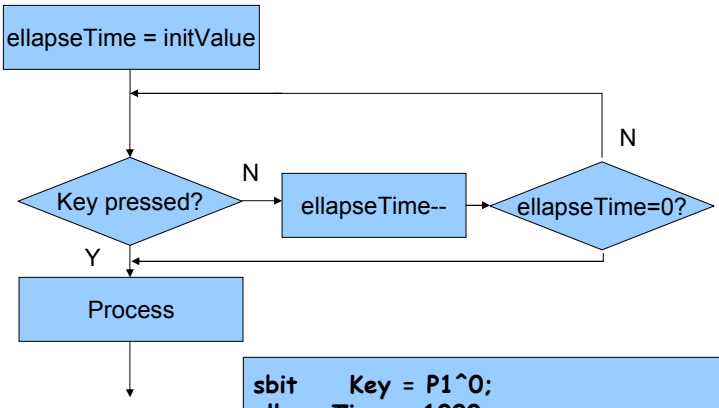
Đơn giản
Không đòi hỏi timer
Không biết chính xác được khoảng delay là bao nhiêu
Được sử dụng khi cần tạo ra các thời gian delay không đòi hỏi sự chính xác lớn.

Cấu trúc Timeout



Nếu phím bị hỏng hay tiếp xúc không tốt, chương trình sẽ bị treo trong quá trình chạy (run-time)

Cấu trúc Timeout




```
sbit    Key = P1^0;
ellapseTime = 1000;
While (Key != 0)   &&  (--ellapseTime  != 0);
```

8051 Timer

Bảng 3.17 Tóm tắt thanh ghi TMOD.

Bit	Tên	Timer	Mô tả
7	GATE	1	Bit mở cổng. Khi bit này 1, timer chỉ chạy trong khi $\overline{INT1}$ ở mức cao.
6	C/$\overline{T}$	1	Bit chọn counter (bộ đếm) hay timer 1 = bộ đếm sự kiện 0 = timer khoảng thời gian
5	M1	1	Bit 1 của chọn chế độ (xem bảng 3.)
4	M0	1	Bit 0 của chọn chế độ (xem bảng 3.)
3	GATE	0	Bit mở cổng cho timer 0.
2	C/$\overline{T}$	0	Bit chọn counter (bộ đếm) hay timer của timer 0
1	M1	0	Bit 1 của chọn chế độ của timer 0
0	M0	0	Bit 0 của chọn chế độ của timer 0

8051 Timer

Bảng 3.19 Tóm tắt thanh ghi TCON.

Bit	Ký hiệu	Địa chỉ bit	Mô tả
TCON.7	TF1	8FH	Cờ báo tràn timer 1. Đặt lên 1 bởi phần cứng khi tràn; được xóa về 0 bởi phần mềm hoặc phần cứng khi bộ xử lý chỉ đến chương trình phục vụ ngắt.
TCON.6	TR1	8EH	Bit điều khiển Timer 1 chạy. Đặt/xóa bằng phần mềm để cho timer chạy/ngưng.
TCON.5	TF0	8DH	Cờ báo tràn Timer 0.
TCON.4	TR0	8CH	Bit điều khiển Timer 0 chạy.
TCON.3	IE1	8BH	Cờ cạnh ngắt 1 bên ngoài. Đặt bởi phần cứng khi phát hiện có cạnh xuống ở $\overline{INT1}$; xóa bằng phần mềm hoặc bằng phần cứng khi CPU chỉ đến chương trình phục vụ ngắt
TCON.2	IT1	8AH	Cờ kiểu ngắt 1 bên ngoài. Đặt/xóa bằng phần mềm để ngắt ngoài tích cực cạnh xuống/mức thấp.
TCON.1	IE0	89H	Cờ cạnh ngắt 0 bên ngoài.
TCON.0	IT0	88H	Cờ kiểu ngắt 0 bên ngoài.

Delay using timer

```
void delay50ms(void)
{
    TMOD &= 0xF0;
    TMOD |=  0x01;
    ET = 0;
    TH0 = 0x3C;
    TL0 = 0xB0;
    TR0 = 1;
    While (!TF0);
    TR0 = 0;
    TF0 = 0;
}
```


Delay.h

```
#ifndef _DELAY_H
#define ……..

#define OSC_FREQ       12
#define OSC_PER_INST 12
#define TIME_1ms       1000
#define PRELOAD_1ms    65536 - (TIME_1ms)/(OSC_FREQ/OSC_PER_INST)
#define  PRELOAD_1ms_H          (PRELOAD_1ms / 256)
#define  PRELOAD_1ms_L          (PRELOAD_1ms % 256)

#define  TIME_50ms      ……
#define PRELOAD_50ms   ……

void      delay1ms(void);
void      delay50ms(void);
void      delayMs(unsigned int n);
#endif
```

delay.c

```
#include  <reg51.h>
#include "delay.h"

void      delay1ms(void)
{
}
void      delay50ms(void)
{
}
void      delayMs(unsigned int n)
{
}
```


Access lowbyte and highbyte

```
#define   LOWBYTE(v)      ((unsigned char) (v))
#define   HIGHBYTE(v)     ((unsigned char) (((unsigned int) (v)) >> 8))
```

```
#define   BYTELOW(v)      (*(((unsigned char *) (&v) + 1)))
#define   BYTEHIGH(v)     (*((unsigned char *) (&v)))
```

```
Void main(void) {

    volatile unsigned char i;
    i = LOWBYTE(0x1234);
    i = BYTEHIGH(0x1234);

}
```

Sai chỗ nào

Multi tasking

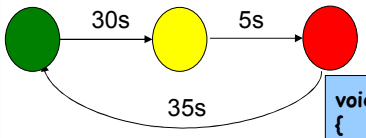
Cấu trúc superloop:

```
int main(void)
{
    while (1) {
        Task_1();
        Task_2();
    }
}
```

Nhược điểm:

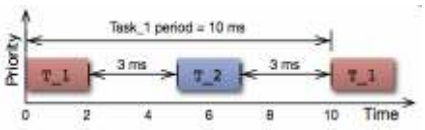
Không dùng cho những ứng dụng đòi hỏi chính xác về mặt timing.
Không có khả năng dừng 1 task đang thực thi để thực hiện task còn lại
(pre-emption)

Cấu trúc superloop với delay

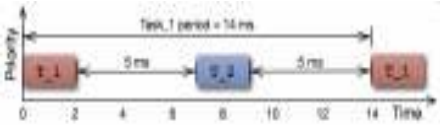



```
void Delay(uint32_t seconds)
{
    // Setup a hardware timer for the given time
    // Loop until the delay has been reached.
}
int main(void)
{
    while (1) {
        Set_Green();
        Delay(30);
        Set_Yellow();
        Delay(5);
        Set_Red();
        Delay(35);
    }
}
```

Cấu trúc superloop với delay



Expected timing



Real timing

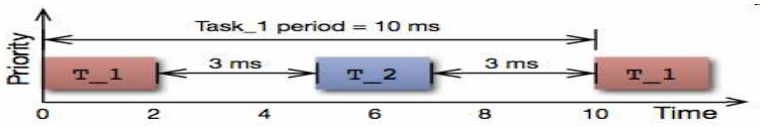
```
void Delay(uint32_t milliseconds)
{
    // Setup a hardware timer for
    // the given time
    // Loop until the delay has
    //been reached.
}

int main(void)
{
    while (1) {
        Task_1();
        Delay(5);

        Task_2();
        Delay(5);
    }
}
```

Nhược điểm:
Cấu trúc này chỉ chạy đúng trong trường hợp các task rất ngắn

Sandwich delay




```
int main(void)
{
    while (1) {
        Start_Timer(5);
        Task_1();
        Wait_For_Timer();

        Start_Timer(5);
        Task_2();
        Wait_For_Timer();
    }
}
```

Nhược điểm:

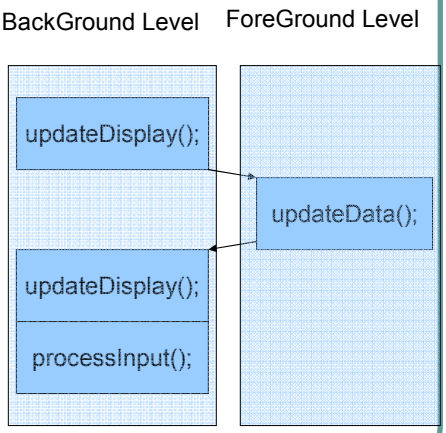
Không cung cấp được mức độ ưu tiên cho các tác vụ

Foreground/Background scheduling

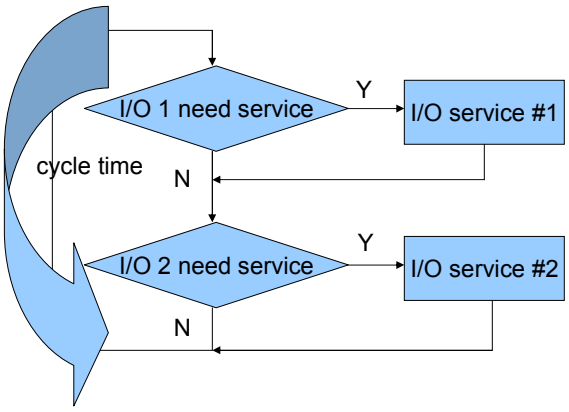
```
// Run by hardware
//every millisecond
void Timer_ISR(void)
{
   updateData();
}

void main(void)
{
   Init_ISR();

   while (1) {
      updateDisplay();
      processInput();   }
}
```



Round-Robin scheduler: Poll and serve



Nhược điểm:

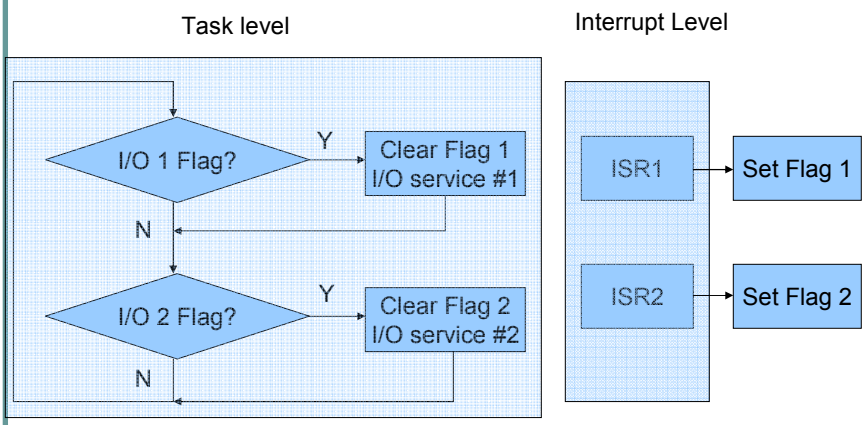
Nếu 1 thiết bị cần thời gian đáp ứng nhanh hơn 1 "cycle time", hệ thống có thể chạy sai.

Nếu có 1 trình phục vụ I/O chạy trong thời gian dài, đáp ứng của hệ thống sẽ bị chậm.

Độ ổn định của hệ thống phụ thuộc nhiều yếu tố.

Khó sửa đổi chương trình

Round-Robin scheduler with interrupt



Nhược điểm:
Tất cả các tác vụ có cùng mức ưu tiên

Solution:
Đưa các code có độ ưu tiên cao vào ISR

8051 interrupt vector table

Interrupt Number	Interrupt Vector Address	Description
0	0003h	EXTERNAL 0
1	000Bh	TIMER/COUNTER 0
2	0013h	EXTERNAL 1
3	001Bh	TIMER/COUNTER 1
4	0023h	SERIAL PORT

Ban đầu, họ 8051 chỉ có 5 nguồn ngắt

Interrupt function

Các nhà sản xuất sau này thêm vào nhiều ngắt cho 8051. Keil C hỗ trợ 32 vector ngắt.

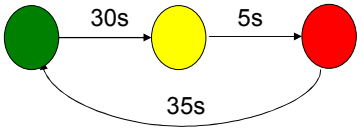
Định nghĩa ngắt:

```
void ISR_name(void) interrupt interrupt_number using bank_number
```

```
unsigned int  int_count;
unsigned char second;

void timer0 (void) interrupt 1 using 2  {
  if (++int_count == 4000)  {       /* count to 4000 */
    second++;                       /* second counter    */
    int_count = 0;                  /* clear interrupt counter */
  }
}
```

Case study



Viết chương trình điều khiển đèn giao thông:

a) Dùng phương pháp sandwich delay
b) Dùng ngắt

Các đèn xanh-vàng-đỏ được điều khiển bởi P1.0, P1.1, P1.2. Đưa chân port lên 1 làm đèn sáng.