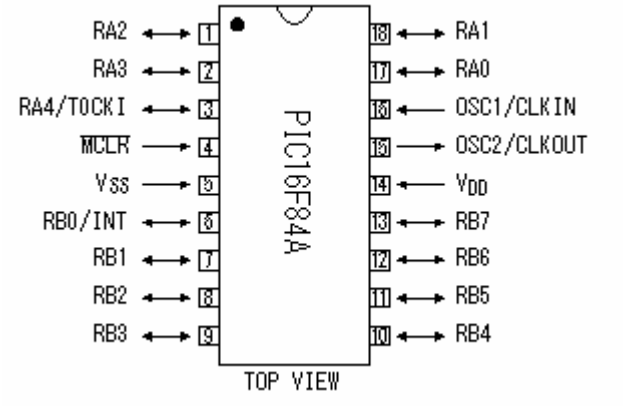
# Bài 1: Làm quen với PIC16F84A

## A. Mạch điện căn bản với PIC16F84A.




Bạn xem hình
PIC16F84A có 18 chân: Khi nhìn vào ic này, Bạn hãy xác định công năng của các chân.

Ở đây, chân 5 cho nối masse để lấy dòng, chân 14 cho nối vào đường nguồn +5V.

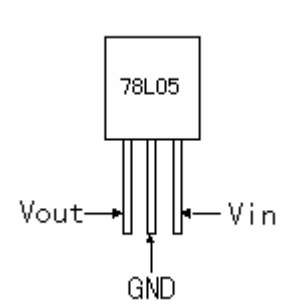
Chân 15, 16 mắc gốm định tần, hay thạch anh.

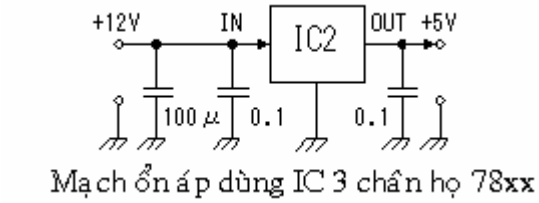
Chân 4 cho nối lên đường nguồn +5V để giữ chân 4 ở mức áp cao.

PortA với các chân 17 (RA0), 18 (RA1), 1(RA2), 2 (RA3), 3 (RA4).

PortB với các chân 6 (RB0), 7 (RB1), 8 (RB2), 9 (RB3), 10 (RB4), 11 (RB5), 12 (RB6), 13 (RB7).

**Khảo sát mạch cấp nguồn**:



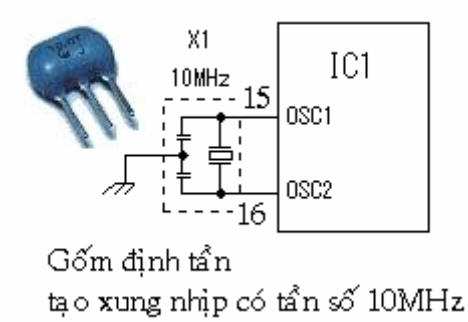



Mạch ổn áp dùng IC 3 chân họ 78xx

Cấp cho mạch với đường nguồn +12V. Mạch dùng IC ổn áp 7805, như vậy điện áp ngả ra là 5V. Mức áp này cấp cho chân 14 của PIC16F84A. Chân số 5 cho nối vào đường masse.

Tụ 100µF dùng để ổn áp, các tụ 0.1µF dùng để lọc bỏ các tín hiệu nhiễu tần số cao. Khi dùng ic ổn áp 7805, chân giữa luôn cho nối masse, chân bên trái lấy điện và chân bên phải là đường nguồn ổn áp.

**Mắc gồm định tần**:

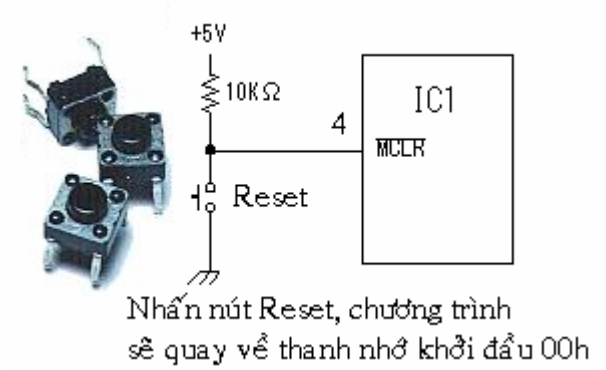


Gốm định tần
tạo xung nhịp có tần số 10MHz

Trong PIC16F84A có mạch dao động dùng để tạo ra xung nhịp, tần số xung nhịp lấy theo tần số của thạch anh hay gốm định tần mắc trên chân 15 (OSC_IN), 16 (OSC_OUT).

Tần số xung nhịp sẽ bằng 10MHz/4.

**Chân MCLR (tác dụng ở mức áp thấp) hay chân RESET**:

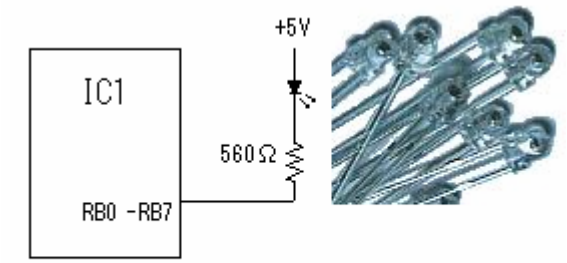


Nhấn nút Reset, chương trình
sẽ quay về thanh nhớ khởi đầu 00h

Trong PIC16F84A có mạch điện Reset (còn gọi là Master Clear). Mạch làm việc với chân 4 nối lên mức áp thấp.

Trong mạch, 10K là điện trở treo chân 4 lên mức áp cao. Khi Bạn muốn chương trình quay lại từ thanh nhớ 0000h, hãy nhấn nút Reset.

**Đặt Led trên các Cảng PortA và PortB**:



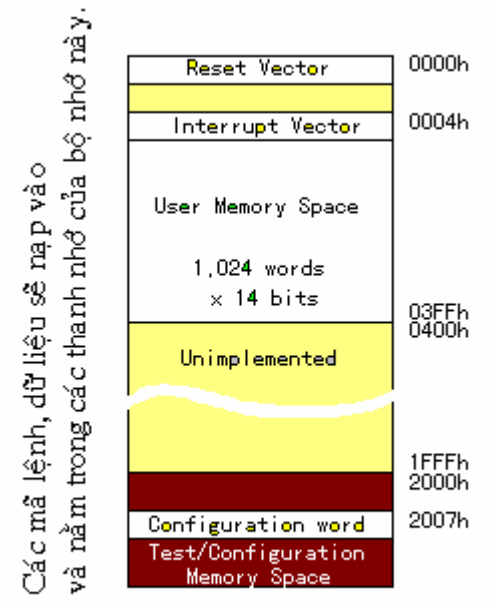
Với cách mắc này,
Led sẽ sáng với bit 0 và tắt với bit 1

Trong mạch, các chân Anode của các Led cho nối vào đường nguồn +5V, điện trở 560 có tác dụng hạn dòng. Với cách mắc này, Led sẽ được cấp dòng khi các ngả ra ở mức áp thấp (bit 0). Khi các chân ngả ra ở mức áp cao (bit 1), Led sẽ tắt.

**B. Khảo sát tổ chức bộ nhớ trong PIC16F84A.**

Trong PIC16F84A có 3 loại bộ nhớ, đó là bộ nhớ PlashROM, bộ nhớ SRAM và bộ nhớ EEPROM.

**Bộ nhớ FlashROM**:



Bộ nhớ chương trình FlashROM

Bộ nhớ FlashROM của PIC16F84A có 1024 thanh nhớ, mỗi thanh nhớ rộng 14 bit.

Thanh nhớ có địa chỉ 0000h là vị trí của lệnh Reset. Mỗi khi chân số 4 bị kéo xuống mức áp thấp thì chương trình sẽ nhẩy về địa chỉ này.

Thanh nhớ có địa chỉ 0004h là vị trí của các lệnh ngắt (Interrupt). Khi xuất hiện các dấu hiệu ngắt, chương trình sẽ nhẩy ngay đến địa chỉ này.

Thanh nhớ có địa chỉ 2007h là thanh nhớ xác định cấu hình của IC. Ở đây Bạn sẽ cho khai báo trạng thái hoạt động của PIC16F84A. Cấu trúc thanh nhớ cấu hình như sau:

Thanh ghi dùng khai báo cấu hình của PIC16F84A

| CP | CP | CP | CP | CP | CP | CP | CP | CP | CP | PWRTE | WDTE | FOSC1 | FOSC0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

Thanh Configuration Word có 14 bit nằm ở địa chỉ 2007h trong bộ nhớ FlashROM, Bạn có thể chọn định trạng thái làm việc cho PIC16F84A bằng cách đặt bit 0 hay bit 1 vào thanh ghi này.

* Bit FOSC1, FOSC0 (Oscillator Selection bits) dùng khai báo kiểu mạch dao động.
Nếu FOSC1=1, FOSC0=1, kiểu dao động là R-C. Tần số làm việc nhỏ hơn 1MHz.
Nếu FOSC1=1, FOSC0=0, kiểu dao động HS dùng thạch anh hay gốm. Tần số từ 4MHz đến 20MHz.
Nếu FOSC1=0, FOSC0=1, kiểu dao động là thạch anh hay gốm. Tần số làm việc nhỏ hơn 4MHz.
Nếu FOSC1=1, FOSC0=0, kiểu dao động là thạch anh công suất nhỏ. Tần số làm việc nhỏ hơn 200KHz.

* Bit WDTE (Watchdog Timer(WDT) Enable bit).
WDTE=1: Cho dùng đồng hồ Watchdog.
WDTE=0: Tắt đồng hồ Watchdog.

* PWRTE (Power-up Timer Enable bit).
PWRTE=1: Tắt chức năng Power Timer.
PWRTE=0: Mở chức năng Power Timer.

* CP (Code Protection bit).
CP=1: Tắt tính bảo vệ bộ nhớ, cho đọc nội dung có trong bộ nhớ FlashROM.
CP=0: Mở tính bảo vệ bộ nhớ (Chống đọc nội dung có trong bộ nhớ).

**Bộ nhớ SRAM:**

| | Address | Bank 0 | Bank 1 | Address |
|---|---|---|---|---|
| Các thanh nhớ đặc dụng | 00h | INDF | ← | 80h |
| | 01h | TMR0 | OPTION_REG | 81h |
| | 02h | PCL | ← | 82h |
| | 03h | STATUS | ← | 83h |
| | 04h | FSR | ← | 84h |
| | 05h | PORTA | TRISA | 85h |
| | 06h | PORTB | TRISB | 86h |
| | 07h | Unimplemented | ← | 87h |
| | 08h | EEDATA | EECON1 | 88h |
| | 09h | EEADR | EECON2 | 89h |
| | 0Ah | PCLATH | ← | 8Ah |
| | 0Bh | INTCON | ← | 8Bh |
| | 0Ch - 4Fh | GPR | ← | 8Ch - CFh |
| | | Dãy 0 | Dãy 1 | |

Bộ nhớ SRAM, ghi đọc tùy ý

SRAM là bộ nhớ ghi đọc tùy ý. Bộ nhớ SRAM của PIC16F84A có 2 dãy (2 Bank, gọi là Bank 0 và Bank 1).

12 thanh nhớ ở phần đầu là các thanh nhớ đặc dụng (SFR). Từ thanh nhớ có địa chỉ 0ch đến thanh nhớ 4fh (có 68 thanh nhớ) là các thanh nhớ phổ dụng (GPR).

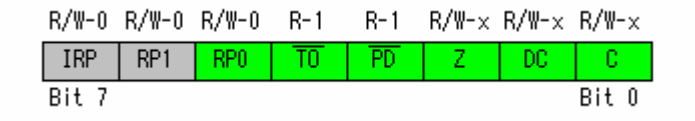
Tìm hiểu các thanh nhớ đặc dụng:

INDF (Data memory contents by indirect addressing): Thanh nhớ chứa nội dung theo địa chỉ gián tiếp có trong thanh nhớ FSR.

TMR0 (Timer counter): Thanh nhớ 8 bit của Timer 0. Thanh nhớ này đếm được 256 xung thì tràn.

PCL (Low order 8 bits of program counter): Thanh nhớ 8 bit thấp của bộ đếm chương trình PC.

STATUS (Flag of calculation result): Thanh nhớ trạng thái. Xác định cờ và chọn bank. SRAM Memory có địa chỉ 03h,83h :




FSR (Indirect data memory address pointer): Thanh nhớ giữ mã địa chỉ dùng truy cập theo bộ nhớ theo mã địa chỉ gián tiếp.

PORTA (PORTA DATA I/O): Thanh nhớ xuất nhập dữ liệu của cảng PortA.

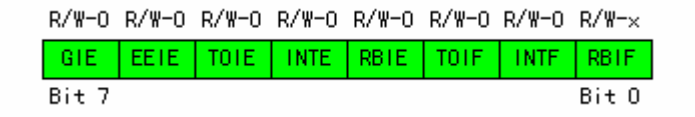
PORTB (PORTB DATA I/O): Thanh nhớ xuất nhập dữ liệu của cảng PortB.

EEDATA (Data for EEPROM): Thanh nhớ cất giữ dữ liệu của bộ nhớ EEPROM.

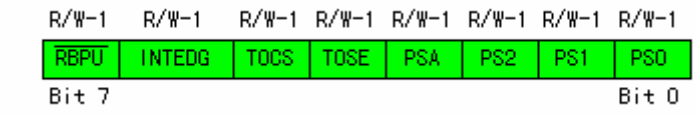
EEADR (Address for EEPROM): Thanh nhớ cất giữ mã địa chỉ dùng cho bộ nhớ EEPROM.

PCLATH (Write buffer for upper 5 bits of the program counter): Thanh nhớ 5 bit cao của bộ đếm chương trình PC. Bộ đếm chương trình PC có 13 bit.

INTCON (Interruption control): Thanh nhớ điều khiển chức năng ngắt. Có mã địa chỉ 0Bh, 8Bh:



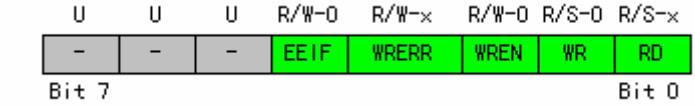

OPTION_REG (Mode set): Thanh nhớ tùy chọn định trạng thái làm việc của PIC16F84A, có địa chỉ 81h:




TRISA (Mode set for PORTA): Thanh nhớ xác định chức năng xuất nhập dữ liệu cho PortA.

TRISB (Mode set for PORTB): Thanh nhớ xác định chức năng xuất nhập dữ liệu cho PortB.

EECON1 (Control Register for EEPROM): Thanh nhớ điều khiển việc ghi đọc bộ nhớ EEPROM, có mã địa chỉ 88h:




EECON2 (Write protection Register for EEPROM ): Thanh nhớ hổ trợ việc ghi đọc bộ nhớ EEPROM.

**C. Tìm hiểu các câu lệnh dùng để chuyển bank trong PIC16F84A.**

Hãy nhìn vào bộ nhớ SRAM của PIC16F84A: Bạn sẽ thấy có 12 thanh nhớ đặc dụng nằm trong Bank 0 từ địa chỉ 00h đến 0bh, và trong Bank 1 cũng có 12 thanh nhớ đặc dụng từ địa chỉ 80h đến địa chỉ 8bh (có 7 thanh nhớ giống nhau nằm trong cả 2 Bank).

| Address | Bank 0 | Bank 1 | Address |
|---|---|---|---|
| 00h | INDF | ← | 80h |
| 01h | TMR0 | OPTION_REG | 81h |
| 02h | PCL | ← | 82h |
| 03h | STATUS | ← | 83h |
| 04h | FSR | ← | 84h |
| 05h | PORTA | TRISA | 85h |
| 06h | PORTB | TRISB | 86h |
| 07h | Unimplemented | ← | 87h |
| 08h | EEDATA | EECON1 | 88h |
| 09h | EEADR | EECON2 | 89h |
| 0Ah | PCLATH | ← | 8Ah |
| 0Bh | INTCON | ← | 8Bh |
| 0Ch - 4Fh | GPR | ← | 8Ch - CFh |

Chỉ từ địa chỉ 0ch đến 4fh là 68 thanh nhớ phổ dụng. Ở 68 thanh nhớ này Bạn có thể cho ghi đọc tùy ý.

Như vậy, khi muốn làm việc với thanh nhớ nào, việc trước tiên Bạn phải biết thanh nhớ đó nằm trong Bank nào và chuyển vào Bank đó mới có thể ghi đọc được. Cách chuyển Bank, Bạn khai báo trong thanh ghi đặc dụng STATUS.

Hãy xem ý nghĩa các bit trong thanh ghi đặc dụng STATUS:

| R/W-0 | R/W-0 | R/W-0 | R-1 | R-1 | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| IRP | RP1 | RP0 | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C |
| Bit 7 | | | | | | | Bit 0 |

*Trong thanh ghi đặc dụng Bạn chú ý 3 bit chọn Bank IRP, RP1, RP0. Với PIC16F84A thì bit IRP và bit RP1 đã luôn cho đặt ở mức áp thấp (bit 0), vậy chỉ còn lại có bit RP0 dùng để chọn Bank.*

## Các câu lệnh chuyển Bank.

Nếu Bạn muốn làm việc với các thanh ghi trong Bank 0 thì đặt bit RP0 ở mức áp thấp (bit 0). Bạn dùng các câu lệnh như sau:

```
BCF    STATUS, RP0        ; Cho RP0=0. Chuyển vào làm việc với các thanh nhớ trong Bank 0.
```

Nếu Bạn muốn làm việc với các thanh ghi trong Bank 1 thì đặt bit RP0 ở mức áp cao (bit 1).

```
BSF    STATUS, RP0        ; Cho RP0=1. Chuyển vào làm việc với các thanh nhớ trong Bank 1.
```

Một thí dụ: Cho đặt PortA và PortB làm các cảng xuất dữ liệu:

```
BSF    STATUS, RP0        ; Vào làm việc với các thanh nhớ trong Bank 1, ở đây có thanh ghi
                          ; TRISA và TRISB.
CLRF   TRISA              ; Đặt vào thanh ghi TRISA trị B'0000 0000', định PortA là ngả ra.
CLRF   TRISB              ; Đặt vào thanh ghi TRISB trị B'0000 0000', định PortB là ngả ra.
BCF    STATUS, RP0        ; Trở lại Bank 0, nơi có các cảng PortA và PortB để cho xuất dữ liệu.
```

Cú pháp của câu lệnh: BSF và BCF như sau:

| BSF | Bit Set f Đặt bit 1 vào một trong tám vị trí của thanh nhớ f |
|---|---|
| Cú pháp | **[label] BSF f, b** |

label là tên nhãn
BSF là lệnh set bit 1
f là mã địa chỉ của thanh nhớ, từ 00 (00h) đến 127 (7fh)
b là trị từ 0 đến 7, dùng chọn định vị trí bit để set 1.

| BCF | Bit Clear f Đặt bit 0 vào một trong tám vị trí của thanh nhớ f |
|---|---|
| Cú pháp | **[label] BCF f, b** |

label là tên nhãn
BCF là lệnh set bit 0
f là mã địa chỉ của thanh nhớ, từ 00 (00h) đến 127 (7fh)
b là trị từ 0 đến 7, dùng chọn định vị trí bit để set 0

| CLRF | Clear f Xóa sạch thanh nhớ f, lúc này trị trong f là b'0000 0000' |
|---|---|
| Cú pháp | **[label] CLRF f** |

label là tên nhãn
CLRF là lệnh xóa thanh nhớ
f là mã địa chỉ của thanh nhớ, có trị từ 00 (00h) đến 127 (7fh)

**Các câu lệnh chuyển trị vào thanh nhớ**.

Để chuyển một trị vô hướng k vào thanh nhớ chúng ta dùng các câu lệnh sau:

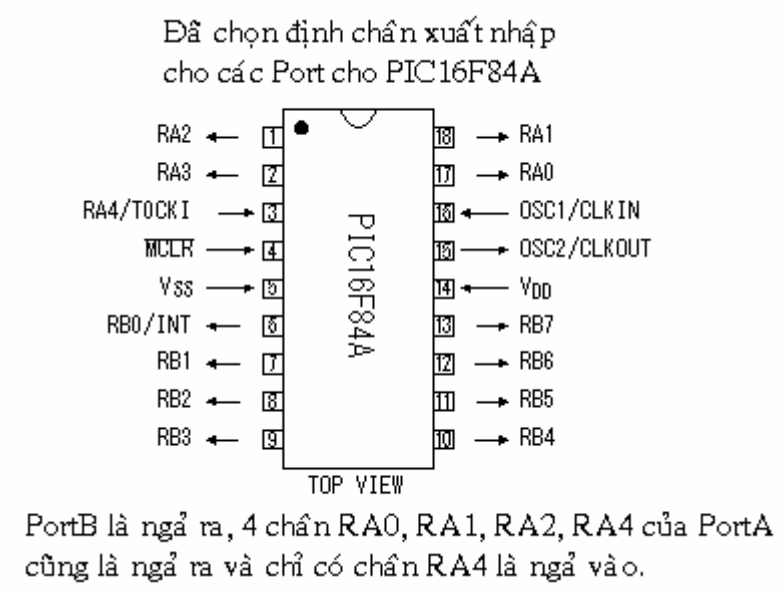
```
MOVLW   H'11110000'  ; Đặt trị 11110000 và thanh ghi W.
MOVWF   TRISA        ; Đặt trị trong thanh ghi W vào thanh ghi TRISA.

MOVLW   H'00000000'  ; Đặt trị 00000000 và thanh ghi W.
MOVWF   TRISB        ; Đặt trị trong thanh ghi W vào thanh ghi TRISB.
```

Với các khai báo này, Bạn đã chọn chân xuất nhập cho các chân của PortB và Port A. Trong khai báo trong các thanh ghi TRISA, TRISB, đặt bit 0 là định ngả ra (Output) và đặt bit 1 là định ngả vào (Input).

| Bank 1 | Address |
|---|---|
| TRISA | 85h |
| TRISB | 86h |

TRISA dùng định hướng xuất nhập dữ liệu cho cảng PortA.
TRISB dùng định hướng xuất nhập dữ liệu cho cảng PortB.

Hai thanh ghi này đều nằm trong Bank 1. TRISA ở địa chỉ 85h và TRISB ở địa chỉ 86h. *Do vậy, Bạn muốn ghi bit vào các thanh ghi này, Bạn phải chuyển vào Bank 1.*




Trên sơ đồ mạch điện các mũi tên cho thấy các đường tín hiệu vào ra trên các chân của PortA và PortB

Việc định ngả vào / ngả ra cho các Port tùy thuộc các khai báo trong thanh ghi TRISA và TRISB.

Cú pháp của câu lệnh: MOVLW và MOVWF như sau:

| **MOVLW** | **Move literal to W** Chuyển một hằng số (k) vào thanh ghi W |
|---|---|
| Cú pháp | **[label] MOVLW k** |

label là tên nhãn
MOVLW là lệnh chuyển trị vào W
k là trị, từ 00 (00h) đến 255 (FFh)

| **MOVWF** | **Move W to f** Chuyển trị có trong W vào thanh ghi f |
|---|---|
| Cú pháp | **[label] MOVWF f** |

label là tên nhãn
MOVWF là lệnh chuyển trị có trong W vào f
f là mã địa chỉ,từ 00 (00h) đến 127 (7Fh)

Hai câu lệnh này rất thông dụng, nó dùng để chuyển một trị (k) vào một thanh nhớ có mã địa chỉ f. Nói khác đi, Bạn không thể chuyển thẳng một trị (k) vào một thanh ghi f, mà trước hết phải chuyển k và W rồi mới chuyển trị trong W vào f.

Thí dụ: Muốn chuyển trị H'32' vào thanh nhớ có mã địa chỉ là 20h. Bạn phải dùng câu lệnh như sau:

```
MOVLW   H'32'  ; Lúc này trị H '32' hay B'00110010' đã có trong thanh ghi W.
MOVWF   H'20'  ; Cho chuyển trị H'32' có trong thanh ghi W vào thanh ghi có mã địa chỉ H'20'
```

**D. Cách viết đoạn chương trình làm trễ.**

Nếu muốn làm chậm một thời gian, Bạn có thể viết đoạn chương trình như sau:

```
CALL  DELAY     ; Lệnh Call cho gọi chương trình con có tên nhãn là DELAY.
```

…..

```
DELAY                           ; Tên nhãn của chương trình làm trễ
        MOVLW     D'10'         ; Nạp trị d'10' vào thanh ghi W
        MOVWF     H'20'         ; Chuyển trị d'10' vào thanh ghi có mã địa chỉ là h'20'
LOOP0
        MOVLW     D'50'         ; Nạp trị d'50' vào thanh ghi W
        MOVWF     H'21'         ; Chuyển trị d'50' vào thanh ghi có mã địa chỉ là h'21'
LOOP1
        MOVLW     D'200'        ; Nạp trị d'200' vào thanh ghi W
        MOVWF     H'22'         ; Chuyển trị d'200' vào thanh ghi có mã địa chỉ là h'22'
LOOP2
        DECFSZ    H'22', 1      ; Cho giảm trị trong thanh ghi h'22' theo bước -1, kết quả cho ghi vào
                                ; h'22',  nếu kết quả chưa bằng 0 thì xuống chấp hành lệnh bên dưới.
        GOTO      LOOP2         ; Nhẩy không điều kiện về tên nhãn LOOP2.
        DECFSZ    H'21', 1      ; Cho giảm trị trong thanh ghi h'21' theo bước -1, kết quả cho ghi vào
                                ; h'21',  nếu kết quả chưa bằng 0 thì xuống chấp hành lệnh bên dưới.
        GOTO      LOOP1         ; Nhẩy không điều kiện về tên nhãn LOOP1.
        DECFSZ    H'20', 1      ; Cho giảm trị trong thanh ghi h'20' theo bước -1, kết quả cho ghi vào
                                ; h'20',  nếu kết quả chưa bằng 0 thì xuống chấp hành lệnh bên dưới.
        GOTO      LOOP0         ; Nhẩy không điều kiện về tên nhãn LOOP0.
        RETURN                  ; Quay lại sau lệnh Call delay.
```

Lệnh trễ này sẽ làm chậm 10x50x200 = 100000 nhịp

Ghi chú: Khi viết đoạn chương trình này cho IC AT889C51, chúng ta viết như sau:

```
DELAY:
        MOV   R7, #10       ; Nạp trị 10 vào thanh ghi R7
V_6:    MOV   R6, #50       ; Nạp trị 50 vào thanh ghi R6
V_5:    MOV   R5, #200      ; Nạp trị 200 vào thanh ghi R5
        DJNZ  R5, $         ; Cho trị trong R5 giảm theo bước -1, chờ R5 bằng 0
        DJNZ  R6, V_5       ; Cho giảm trị trong R6 theo bước -1, chưa bằng 0, quay về tên nhãn V_5
        DJNZ  R7, V_6       ; Cho giảm trị trong R7 theo bước -1, chưa bằng 0, quay về tên nhãn V_6
        RET                 ; Quay lại sau lệnh Call delay.
```

Sau đây là một thí dụ về cách viết các câu lệnh làm trễ với nhiều hạn định:

```
;*******************************************************************************
;       Timer  Subroutine for 10MHz clock (Khai báo tần số thạch anh là 10MHz)
;
;*************  1msec Timer Subroutine  (Chương trình trễ 1ms) ******************************
        T1M                     ; Tên nhãn
        MOVLW   D'2'            ;(1)    Đặt trị thập phân 2 vào thanh ghi W
        MOVWF   CNT1M           ;(1)    Đặt trị có trong W vào địa chỉ đã đặt tên là CNT1M
        TM1LP1
```

```
        MOVLW   D'249'          ;(1)*2    Đặt trị thập phân 249 vào thanh ghi W
        MOVWF   CNT500U         ;(1)*2    Đặt trị có trong W vào địa chỉ đã đặt tên là CNT500U
        TM1LP2                  ; Tên nhãn
        NOP                     ;(1)*249*2                  Lệnh trống dùng tăng nhịp đếm.
        NOP                     ;(1)*249*2                  Lệnh trống dùng tăng nhịp đếm.
        DECFSZ  CNT500U, F      ;(1)*249*2 cnt500u-1=0 ?    Trị trong CNT500U-1 bằng không chưa ?
        GOTO    TM1LP2          ;(2)*248*2 No, continue     Nếu chưa bằng 0, Quay lại TM1LP2
        DECFSZ  CNT1M, F        ;(1)*2    cnt1m-1=0 ?       Trị trong CNT1M-1 bằng không chưa ?
        GOTO    TM1LP1          ;(2)      No. Continue      Nếu chưa bằng 0, Quay lại TM1LP1
        RETURN                  ;(2)      Yes. Cnt end      Quay lại sau lệnh CALL DELAY
;
;       Total 2501*0.4usec=1msec (Tổng số nhịp của chương trình trên là khoảng 1ms)
;
;*************  100msec Timer Subroutine  ***********************************************
        T100M                   ; Tên nhãn
        MOVLW   D'100'          ;Set loop counter   Đặt trị thập phân 100 vào thanh ghi W
        MOVWF   CNT100M         ;Save loop counterĐặt trị có trong W vào địa chỉ đã đặt tên là CNT100M
        TM2LP                   ; Tên nhãn
        CALL    T1M             ;1msec subroutine     Lệnh gọi chương trình con có tên nhãn là T1M
        DECFSZ  CNT100M,F       ;cnt100m - 1 = 0 ?    Trị trong CNT100M-1 bằng không chưa ?
        GOTO    TM2LP           ;No. Continue         Nếu chưa bằng 0, quay lại TM2LP
        RETURN                  ;Yes. Count end       Quay lại sau lệnh CALL DELAY

;*************  500msec Timer Subroutine  ***********************************************
        T500M                   ; Tên nhãn
        MOVLW   D'5'            ;Set loop counter     Đặt trị thập phân 5 vào thanh ghi W
        MOVWF   CNT500M         ;Save loop counter    Đặt trị có trong W vào địa chỉ đã đặt tên là CNT500M
        TM3LP                   ; Tên nhãn
        CALL    T100M           ;100msec subroutine   Lệnh gọi chương trình con có tên nhãn là T100M
        DECFSZ  CNT500M,F       ;cnt500m - 1 = 0 ?    Trị trong CNT500M-1 bằng không chưa ?
        GOTO    TM3LP           ;No. Continue         Nếu chưa bằng 0, quay lại TM3LP
        RETURN                  ;Yes. Count end       Quay lại sau lệnh CALL DELAY

;*************  1sec Timer Subroutine  **************************************************
        T1S                     ; Tên nhãn
        MOVLW   D'2'            ;Set loop counter     Đặt trị thập phân 2 vào thanh ghi W
        MOVWF   CNT1S           ;Save loop counter    Đặt trị có trong W vào địa chỉ đã đặt tên là CNT1S
        TM4LP                   ; Tên nhãn
        CALL    T500M           ;500msec subroutine Lệnh gọi chương trình con có tên nhãn là T500M
        DECFSZ  CNT1S,F         ;cnt1s - 1 = 0 ?      Trị trong CNT1S-1 bằng không chưa ?
        GOTO    TM4LP           ;No. Continue         Nếu chưa bằng 0, quay lại TM4LP
        RETURN                  ;Yes. Count end       Quay lại sau lệnh CALL DELAY
```

**Tìm hiểu câu lệnh DECFSZ dùng trong chương trình làm trễ.**

Cú pháp của câu lệnh DECFSZ:

| **DECFSZ** | **Decrement f, Skip if 0** Giảm -1 trong f, xét trị trong f bằng 0 chưa? |
|---|---|
| Cú pháp | **[label] DECFSZ f, d** |

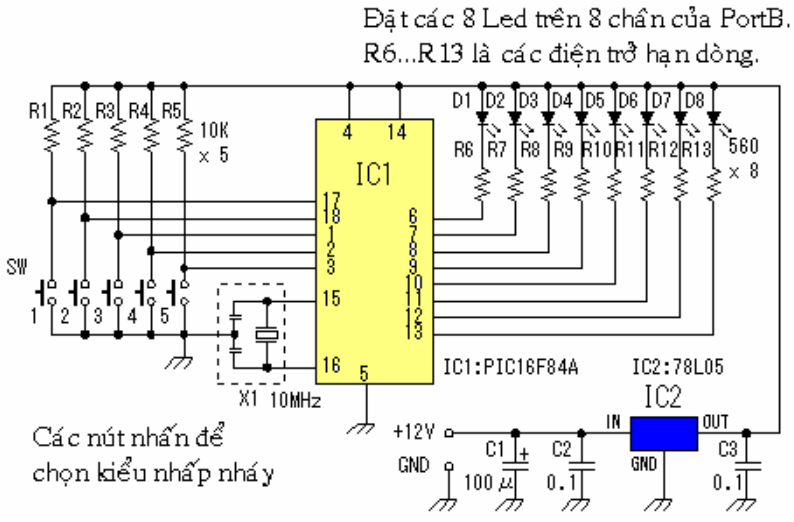
label là tên nhãn
DECFSZ là lệnh cho giảm theo bước -1 và nhẩy qua một câu lệnh khi thanh ghi đã bằng 0.
Nếu chưa bằng không thì xuống thực hiện câu lệnh bên dưới
f là mã địa chỉ có trị từ 00 (00h) đến 127 (7Fh)
d là bit định hướng.
Nếu d=1 (hay f), kết quả cho ghi vào f
Nếu d=0 (hay W), kết quả cho ghi vào W

Một câu lệnh tương tự INCFSZ là cho tăng trị trong f theo bước +1, xét trị trong f băng không chưa? Nếu đã bằng 0 thì nhẩy qua một câu lệnh:

| **INCFSZ** | **Increment f, Skip if 0** Tăng +1 trong f, xét trị trong f bằng 0 chưa? |
|---|---|
| Cú pháp | **[label] INCFSZ f, d** |

label là tên nhãn
INCFSZ là lệnh cho tăng theo bước +1 và nhẩy qua một câu lệnh khi thanh ghi đã bằng 0.
Nếu chưa bằng không thì xuống thực hiện câu lệnh bên dưới
f là mã địa chỉ có trị từ 00 (00h) đến 127 (7Fh)
d là bit định hướng.
Nếu d=1 (hay f), kết quả cho ghi vào f
Nếu d=0 (hay W), kết quả cho ghi vào W

**E. Viết chương trình điều khiển sự tắt sáng của các dãy Led trên PortB của PIC16F84A.**




Sơ đồ mạch điện cho thấy: Chúng ta gắn 8 Led trên PortB và sẽ viết chương trình tạo ra nhiều dạng nhấp nháy trên 8 Led này. Mạch làm việc với mức áp nguồn +5V. Tần số dao động định tần theo gốm X1 10MHz. Chân MCLR cho nối vào đường nguồn +5V.

Cách khai báo các kiểu nhấp nháy cho 8 Led trên PortB. Hãy dùng định nghĩa EQU để xác định trạng thái cho các chân trên PortB:

```
K00    EQU   B'11111110'   ; Kiểu 1.   Ghi chú: Bit 0 là lúc Led sáng, bit 1 là Led tắt.
K01    EQU   B'11111101'
K02    EQU   B'11111011'
K03    EQU   B'11110111'
K04    EQU   B'11101111'
K05    EQU   B'11011111'
K06    EQU   B'10111111'
K07    EQU   B'01111111'

K10    EQU   B'11111110'   ; Kiểu 2   Ghi chú: Bit 0 là lúc Led sáng, bit 1 là Led tắt.
K11    EQU   B'11111100'
K12    EQU   B'11111000'
K13    EQU   B'11110000'
K14    EQU   B'11100000'
K15    EQU   B'11000000'
K16    EQU   B'10000000'
K17    EQU   B'00000000'

K20    EQU   B'00000000'   ; Kiểu 3  Ghi chú: Bit 0 là lúc Led sáng, bit 1 là Led tắt.
K21    EQU   B'10000000'
K22    EQU   B'11000000'
K23    EQU   B'11100000'
K24    EQU   B'11110000'
K25    EQU   B'11111000'
K26    EQU   B'11111100'
K27    EQU   B'11111110'

K30    EQU   B'00000000'   ; Kiểu 4  Ghi chú: Bit 0 là lúc Led sáng, bit 1 là Led tắt.
K31    EQU   B'10000001'
K32    EQU   B'11000011'
K33    EQU   B'11100111'
K34    EQU   B'11100111'
K35    EQU   B'11000011'
K36    EQU   B'10000001'
K37    EQU   B'00000000'

K40    EQU   B'11111111'   ; Kiểu 5  Ghi chú: Bit 0 là lúc Led sáng, bit 1 là Led tắt.
K41    EQU   B'00000000'
K42    EQU   B'11111111'
K43    EQU   B'00000000'
K44    EQU   B'11111111'
K45    EQU   B'00000000'
K46    EQU   B'11111111'
K47    EQU   B'00000000'
```

Để đưa các trạng thái đã định nghĩa trên cho xuất ra trên PortB, Bạn dùng các câu lệnh như sau:

```
K0                              ; Tên nhãn của kiểu nháy K0
        MOVLW   K00             ; Nạp mã B'11111110' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        MOVLW   K01             ; Nạp mã B'11111101' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        MOVLW   K02             ; Nạp mã B'11111011' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        MOVLW   K03             ; Nạp mã B'11110111' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        MOVLW   K04             ; Nạp mã B'11101111' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        MOVLW   K05             ; Nạp mã B'11011111' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        MOVLW   K06             ; Nạp mã B'10111111' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        MOVLW   K07             ; Nạp mã B'01111111' vào thanh ghi W.
        MOVWF   PORTB           ; Chuyển trị trong W ra PortB.
        CALL    TM1S            ; Gọi chương trình làm chậm 1s.
        RETURN                  ; Quay lại sau câu lệnh CALL K0
```

Bạn sẽ viết tương tự như phần trên cho các kiểu khác, đặt tên nhãn là K1, K2, K3, K4. Các chương trình con này sẽ được gọi là bằng lệnh CALL, như:

```
CALL K0
CALL K1
CALL K2
CALL K3
CALL K4
```

Đến đây, Bạn đã có thể viết một chương trình nguồn đơn giản (gọi là file source có họ là .asm). Chương trình sẽ khiến cho 8 Led trên PortB sáng lan dần lên, chương trình mô tả như sau:

```
;-------------------------------------------------------------------------------------------------------
;       Mach test cac den LED_1 o PORTB
;       00/00/2008
;       Dung PIC16F877A
;       LED giao tiep voi PORTB
;       Chan K cua LED noi GND
;       Chon RB0 - RB7 la cac chan OUTPUT
;=======================================================================
        TITLE       "Mach chop tat LED_1"
        LIST        P=PIC16F877A
        INCLUDE     P16F877A.INC
        __CONFIG  _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &_HS_OSC & _WRT_OFF & _LVP_OFF & _CPD_OFF
;=======================================================================
CNT500U     EQU     H'20'           ; Đặt tên thanh nhớ có địa chỉ h'20' là CNT500U.
CNT1M       EQU     H'21'           ; Đặt tên thanh nhớ có địa chỉ h'21' là CNT1M.
 CNT100M    EQU     H'22'           ; Đặt tên thanh nhớ có địa chỉ h'22' là CNT100M.
CNT500M     EQU     H'23'           ; Đặt tên thanh nhớ có địa chỉ h'23' là CNT500M.
CNT1S       EQU     H'24'           ; Đặt tên thanh nhớ có địa chỉ h'24' là CNT1S.
;-------------------------------------------------------------------------------------------------------
P00    EQU    B'11111110'          ; Led trên chân số 6, sáng.
P01    EQU    B'11111100'          ; Led trên 2 chân số 6, 7, sáng.
P02    EQU    B'11111000'          ; Led trên 3 chân số 6, 7, 8, sáng.
P03    EQU    B'11110000'          ; Led trên 4 chân số 6, 7, 8, 9, sáng.
P04    EQU    B'11100000'          ; Led trên 5 chân số 6, 7, 8, 9, 10, sáng.
P05    EQU    B'11000000'          ; Led trên 6 chân số 6, 7, 8, 9, 10, 11, sáng.
P06    EQU    B'10000000'          ; Led trên 7 chân số 6, 7, 8, 9, 10, 11, 12, sáng.
P07    EQU    B'00000000'          ; Led trên 8 chân số 6, 7, 8, 9, 10, 11, 12, 13, sáng.
;=======================================================================
ORG   0x0000
        GOTO        MAIN                ; Nhẩy đến tên nhãn MAIN
ORG   0x0004
MAIN                                    ; Tên nhãn
        BSF         STATUS, RP0         ; Cho   RP0=1 để vào Bank 1.
        BCF         STATUS, RP1         ; Cho   RP1=0 để vào Bank 1. (có thể không dùng câu lệnh này)
        MOVLW       H'00'               ; Đặt trị B'00000000' vào thanh ghi W
        MOVWF       TRISB               ; Chuyển trị có trong W vào thanh ghi định hướng TRISB.
        BCF         STATUS, RP1         ; Cho   RP1=0 để trở lại Bank 0.
        BCF         STATUS, RP0         ; Cho   RP0=0 để trở lại Bank 0.
LOOP
        MOVLW       P00         ; Đặt trị P00 vào thanh ghi W.
        MOVWF       PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL        T100M       ; Làm chậm 100ms
        MOVLW       P01         ; Đặt trị P01 vào thanh ghi W.
        MOVWF       PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL        T100M       ; Làm chậm 100ms
        MOVLW       P02         ; Đặt trị P02 vào thanh ghi W.
```

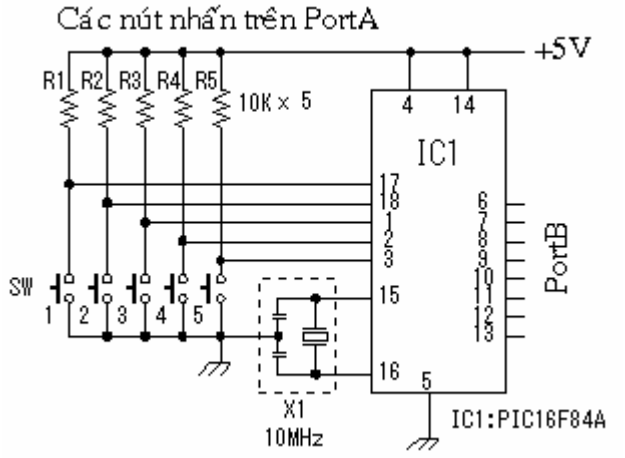
```
        MOVWF   PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL    T100M       ; Làm chậm 100ms
        MOVLW   P03         ; Đặt trị P03 vào thanh ghi W.
        MOVWF   PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL    T100M       ; Làm chậm 100ms
        MOVLW   P04         ; Đặt trị P04 vào thanh ghi W.
        MOVWF   PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL    T100M       ; Làm chậm 100ms
        MOVLW   P05         ; Đặt trị P05 vào thanh ghi W.
        MOVWF   PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL    T100M       ; Làm chậm 100ms
        MOVLW   P06         ; Đặt trị P06 vào thanh ghi W.
        MOVWF   PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL    T100M       ; Làm chậm 100ms
        MOVLW   P07         ; Đặt trị P07 vào thanh ghi W.
        MOVWF   PORTB       ; Xuất trị có trong W ra PortB để làm sáng các Led trên cảng này.
        CALL    T100M       ; Làm chậm 100ms
        MOVLW   H'FF'       ; Đặt trị H'FF' vào thanh ghi W.
        MOVWF   PORTB       ; Xuất trị có trong W ra PortB để tắt tất cả các Led trên cảng này.
        CALL    T100M       ; Làm chậm 100ms
        GOTO    LOOP        ; Quay lại tên nhãn LOOP để tiếp tục.
;*****************************************************************************
;       Timer  Subroutine for 10MHz clock
;************** 1msec Timer Subroutine ***************************************
T1M
        MOVLW   D'2'                ;(1)     Đặt trị thập phân D'2' vào thanh ghi W.
        MOVWF   CNT1M               ;(1)     Chuyển trị có trong W vào CNT1M (địa chỉ 21H)
TM1LP1
        MOVLW   D'249'              ;(1)*2   Đặt trị thập phân D'249' vào thanh ghi W.
        MOVWF   CNT500U             ;(1)*2   Chuyển trị có trong W vào CNT500UM (địa chỉ 20H)
TM1LP2
        NOP                         ;(1)*249*2     Tăng nhịp đếm khi qua lệnh trống NOP
        NOP                         ;(1)*249*2     Tăng nhịp đếm khi qua lệnh trống NOP
        DECFSZ  CNT500U,F           ;(1)*249*2 CNT500U-1=0 ?  Lệnh nhẩy có điều kiện.
        GOTO    TM1LP2              ;(2)*248*2 NO,            Tiếp tục giảm -1 trong CNT500U
        DECFSZ  CNT1M,F             ;(1)*2    CNT1M-1=0 ?     Lệnh nhẩy có điều kiện.
        GOTO    TM1LP1              ;(2)                      Tiếp tục
        RETURN                      ;(2)                      Quay lai sau thời gian làm chậm.
                                    ;TOTAL 2501*0.4USEC=1MSEC
;************** 100MSEC TIMER SUBROUTINE *************************************
T100M
        MOVLW   D'100'              ; Đặt trị thập phân D'100' vào thanh ghi W.
        MOVWF   CNT100M             ; Chuyển trị có trong W vào CNT1M (địa chỉ 22H)
TM2LP
        CALL    T1M                 ; Gọi chương trình con T1M
        DECFSZ  CNT100M,F           ; CNT100M - 1 = 0 ?  Lệnh nhẩy có điều kiện.
```

```
        GOTO        TM2LP           ; Lệnh nhẩy không điều kiện, để tiếp tục đếm.
        RETURN                      ; Quay lai sau thời gian làm chậm.

;*************  500MSEC TIMER SUBROUTINE  ************************************
T500M
        MOVLW       D'5'            ; Đặt trị thập phân D'5' vào thanh ghi W.
        MOVWF       CNT500M         ; Chuyển trị có trong W vào CNT1M (địa chỉ 23H)
TM3LP
        CALL        T100M           ; Gọi chương trình con T100M
        DECFSZ      CNT500M,F       ; CNT500M - 1 = 0 ?  Lệnh nhẩy có điều kiện
        GOTO        TM3LP           ; Lệnh nhẩy không điều kiện, để tiếp tục đếm.
        RETURN                      ; Quay lai sau thời gian làm chậm.
;**************  1SEC TIMER SUBROUTINE  *************************************
T1S
        MOVLW       D'2'            ; Đặt trị thập phân D'2' vào thanh ghi W.
        MOVWF       CNT1S           ; Chuyển trị có trong W vào CNT1M (địa chỉ 24H)
TM4LP
        CALL        T500M           ; Gọi chương trình con T500M
        DECFSZ      CNT1S,F         ; CNT1S - 1 = 0 ?       Lệnh nhẩy có điều kiện
        GOTO        TM4LP           ; Lệnh nhẩy không điều kiện, để tiếp tục đếm.
        RETURN                      ; Quay lai sau thời gian làm chậm.
;*******************************************************************************
;       END of LED flash control processing
;*******************************************************************************
END
```

Sau khi đã viết xong chương trình nguồn trên (lấy họ là .ASM), Bạn dùng trình MPLAB để biên dịch ra các file: file có họ .LST dùng để kiểm tra lỗi, và có file họ .HEX, Bạn dùng trình WINPIC800 để cho nạp file này vào bộ nhớ FlashROM của PIC16F84A. Nếu mọi việc đều tốt đẹp, như vậy công việc của Bạn đã thanh công mỹ mãn.

**F. Viết chương trình cho các nút nhấn**.

Nếu trên board thực hành Bạn có dùng đến các nút nhấn, như hình sau:




Trong mạch, các nút nhấn đặt ở PortA, vậy PortA phải chọn định là ngả vào. Các điện trở 10K dùng treo các chân của PortA lên mức áp cao. Mỗi khi Bạn nhấn nút nhấn xuống, chân đó sẽ bị kéo xuống mức áp thấp. Với dấu hiệu này, chúng ta sẽ cho nó nhẩy đến chương trình đã gắn với phím.

Cách viết đoạn chương trình gắn với 5 nút nhấn đặt trên PortA như sau:

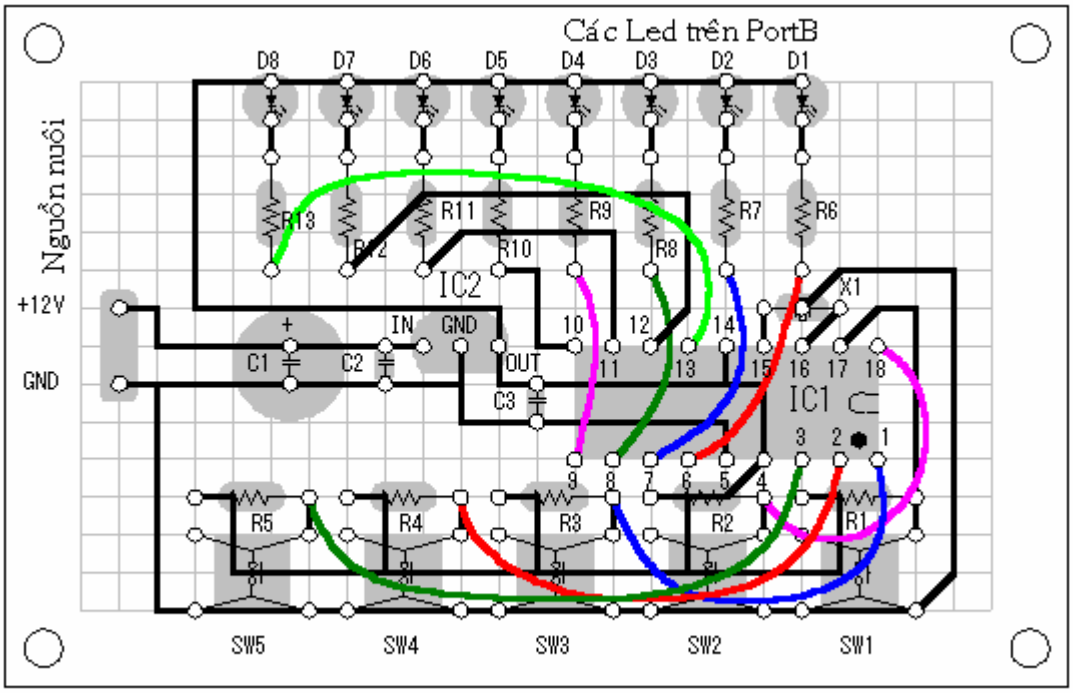
```
;***************                 Program         Start
```

```
**********************
ORG    0                    ; Địa chỉ ứng với chức năng Reset (Reset Vector)
       GOTO   INIT          ; Nhẩy không điều kiện đến tên nhãn INIT.
ORG    4                    ; Địa chỉ ứng với các lệnh ngắt (Interrupt Vector)
       GOTO   INIT
;***************  Initial Process  ********************
ORG    5
INIT
       BSF          STATUS,RP0        ; Chuyển qua Bank 1.
       MOVLW        H'ff'             ; Đặt trị H'FF' hay B'11111111' vào thanh ghi W.
       MOVWF        TRISA             ; Chuyển trị có trong W vào TRISA để định PortA là ngả ra.
       CLRF         TRISB             ; Đặt trị B'00000000' vào TRISB để định PortB là ngả vào.
       BCF          STATUS,RP0        ; Về lại Bank 0.
       MOVLW        H'ff'             ; Đặt trị H'FF' hay B'11111111' vào thanh ghi W.
       MOVWF        PORTB             ; Cho xuất trị có trong W ra PortB.
;***************  Key Scan Process  ********************
KEYSCAN
       BTFSS   PORTA,RA0              ; Nếu chân RA0 ở mức áp cao thì nhẩy qua lệnh CALL PTN0.
       CALL    PTN0                   ; Nếu chân RA0 ở mức áp thấp thì gọi ra chương trình con PTN0.
       BTFSS   PORTA,RA1              ; Nếu chân RA1 ở mức áp cao thì nhẩy qua lệnh CALL PTN1.
       CALL    PTN1                   ; Nếu chân RA1 ở mức áp thấp thì gọi ra chương trình con PTN1.
       BTFSS   PORTA,RA2              ; Nếu chân RA2 ở mức áp cao thì nhẩy qua lệnh CALL PTN2.
       CALL    PTN2                   ; Nếu chân RA2 ở mức áp thấp thì gọi ra chương trình con PTN2.
       BTFSS   PORTA,RA3              ; Nếu chân RA3 ở mức áp cao thì nhẩy qua lệnh CALL PTN3.
       CALL    PTN3                   ; Nếu chân RA3 ở mức áp thấp thì gọi ra chương trình con PTN3.
       BTFSS   PORTA,RA4              ; Nếu chân RA4 ở mức áp cao thì nhẩy qua lệnh CALL PTN4.
       CALL    PTN4                   ; Nếu chân RA4 ở mức áp thấp thì gọi ra chương trình con PTN4.
       GOTO    KEYSCAN                ; Quay lại tên nhãn KEYSCAN để chờ nhấn phím.
```

Tóm lại, đoạn chương trình này sẽ lần lượt cho quét qua 5 phím và chờ có phím nhấn, khi có một phím nhấn xuống, chân đó sẽ xuống mức áp thấp, chương trình sẽ cho chạy ngay lệnh CALL đã gắn với phím nhấn đó.

## G. Tìm hiểu Board thực hành dùng cho bài 1.

Để hiểu rõ hơn cách dùng PIC16F84A trong các ứng dụng, trước hết Bạn hãy làm bảng mạch in và ráp theo sơ đồ mạch điện như hình sau:



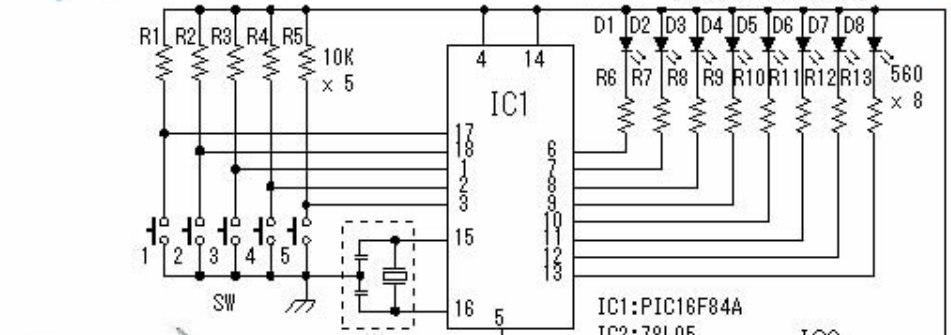

Hình vẽ bố trí các linh kiện trên bảng mạch in

PIC16F84A

Đế cắm 18 chân

Led chiếu sáng

Board thực hành




Điện trở

Nút nhấn

Gốm định tần

Tụ nhỏ

Tụ hóa

Sơ đồ mạch điện dùng thực hành với PIC16F84A

File nguồn .ASM và file .HEX: led_source.zip và led_hex.zip