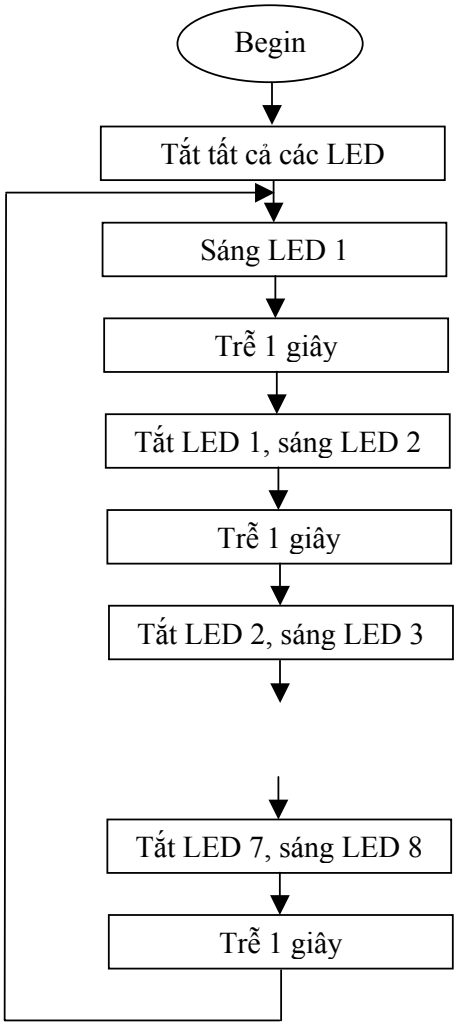
Bài tập 1: *Cho sáng lần lượt từng LED, mỗi LED sáng 1 giây.*



Trên đây là lưu đồ, việc thể hiện bằng lệnh lưu đồ trên có thể bằng cách chân phương (làm lần lượt) hoặc có thể dùng lệnh quay để đưa bit = 0 ra lần lượt các chân cổng làm LED sáng theo lần lượt. Từ bài tập này trở đi, mặc định chương trình đã có khai báo:
#include <sfr51.inc>

Cách chân phương:

```
        org     00h
        ljmp    main
        org     40h
main:
        mov     SP,#5fh
        ;Việc tắt tất cả các LED được tự động làm do khi reset lên, các chân cổng đều = 1
        mov     p1,#11111110b       ;sáng LED 1
        lcall   tre_1s
        mov     p1,#11111101b       ;tắt LED 1, sáng LED 2
        lcall   tre_1s
        mov     p1,#11111011b       ;tắt LED 2, sáng LED 3
```

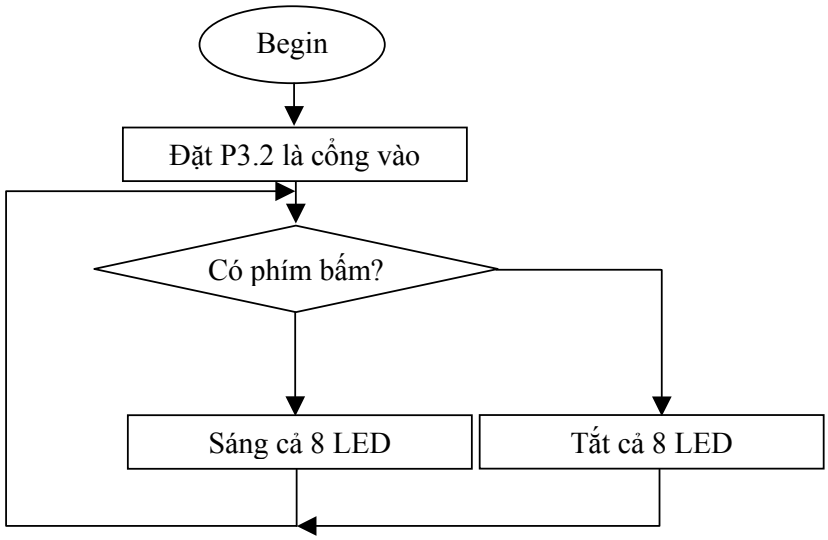
```
        lcall   tre_1s
        ;…
        mov     p1,#01111111b
        lcall   tre_1s
        sjmp    main

tre_1s:
        mov     r1,#10
loop1:
        mov     r2,#100
loop2:
        mov     r3,#100
loop3:
        nop
        ;…<tất cả 8 lệnh Nop>
        nop
        djnz    r3,loop3
        djnz    r2,loop2
        djnz    r1,loop1
        ret
end
```

Nếu sử dụng lệnh quay thì chương trình sẽ gọn hơn:

```
;…
main:
        mov     SP,#5fh
        mov     a,#11111110b
main_loop:
        mov     P1,a
        lcall   tre_1s
        rl      a
        sjmp    main_loop
tre_1s:
;…như ở trên….
end
```

Lưu ý là trong các đoạn chương trình trên, ngăn xếp được khởi tạo với giá trị ban đầu của thanh ghi con trỏ ngăn xếp SP là 5fh, tức là các địa chỉ hay dữ liệu cất vào ngăn xếp sẽ bắt đầu từ ô nhớ 60h trở đi. Việc khởi tạo ngăn xếp là một thao tác không thể thiếu khi trong chương trình có sử dụng lệnh gọi chương trình con hoặc các ngắt.

Bài tập 2: *kiểm tra phím bấm nối với P3.2, khi phím được bấm thì sáng cả 8 LED, khi không bấm phím thì tắt cả 8 LED.*



Với lưu đồ thuật toán như trên ta viết được chương trình sau:

```
        …
        org     00h
        ljmp    main
        org     40h
main:
        ;việc đặt cổng P3.2 làm cổng vào được tự động thực hiện khi 8051 reset xong.
        ;các LED cũng tự động được tắt vì khi 8051 reset xong, các cổng đều = 1
        jnb     P3.2,phim_bam
        mov     P1,#0ffh        ;tắt tất cả các LED nếu không có phím bấm
        sjmp    main
phim_bam:
        mov     P1,#0           ;sáng tất cả các LED nếu có phím bấm
        sjmp    main
end
```

Với cùng mạch phần cứng như trên, ta có thể thực hành về ngắt của vi điều khiển. Bài toán thực hành đặt ra có thể là nếu có phím bấm thì đảo trạng thái của 8 LED (đang sáng thì thành tắt và ngược lại). Như vậy nếu ta bấm phím, trạng thái của LED sẽ được đảo lại, mỗi lần bấm phím đảo một lần.

Nếu không sử dụng ngắt, ta có thể viết chương trình gần giống với chương trình trên, chỉ khác là phải thêm thao tác đợi nhả phím ra trước khi quay trở lại quét kiểm tra điện áp tại chân P3.2. Nếu không có thao tác này, chân P3.2 xuống mức 0 sẽ bị gây ra nhiều lần đảo trạng thái LED bởi vì chân P3.2 sẽ được quét liên tục, thấy còn ở mức 0 là lại đảo trạng thái, cho đến khi nhả phím ra thì các LED sẽ cùng sáng hoặc cùng tắt, tùy thuộc vào việc lần đảo trạng thái nào được thực hiện cuối cùng. Như vậy sẽ không đúng với mong muốn là chỉ đảo một lần khi bấm một lần.

```
…
main:
        jb      P3.2,$                  ;nhảy tại chỗ chờ cho đến khi nào P3.2 = 0 (tức là chờ
                                        cho đến khi có phím bấm
```

```
phim_bam:
        mov     a,P1            ;đọc giá trị hiện thời của cổng P1
        cpl     a               ;đảo trạng thái đọc được đi
        mov     P1,a            ;đưa trở lại cổng P1 gây ra đảo trạng thái LED
        jnb     P3.2,$          ;nhảy tại chỗ để đợi cho đến khi nào P3.2 = 1 trở lại
        sjmp    main
end
```

Nếu sử dụng ngắt, trước khi sử dụng ta phải đặt chế độ và cho phép ngắt xảy ra. Phím được nối với chân P3.2 là chân ngắt ngoài số 0 (INT0). Ngắt này có 02 chế độ là ngắt theo mức và ngắt theo sườn. Ngắt theo mức sẽ liên tục gây ra ngắt CPU chừng nào mức điện áp của chân P3.2 còn ở mức thấp. Như vậy nếu CPU thực hiện xong chương trình phục vụ ngắt mà chân P3.2 vẫn ở mức thấp thì CPU sẽ lại thực hiện tiếp chương trình phục vụ ngắt đó. Ngắt theo sườn thì ngược lại, chỉ gây ra ngắt khi có sườn tín hiệu (trong trường hợp 8051 là sườn xuống), còn sau đó chân tín hiệu ngắt dù là mức thấp hay cao hay có sườn lên thì cũng không gây ra ngắt nữa. Việc đặt chế độ cho ngắt ngoài 0 được thực hiện bằng bit IT0 trong thanh ghi TCON. Nếu bit này = 1 thì ngắt theo sườn và ngược lại. Thực hành với hai trường hợp IT0 = 0 (ứng với chế độ ngắt theo mức) và IT0 = 1 (ứng với chế độ ngắt theo sườn xuống), ta sẽ thấy được sự khác biệt giữa chúng. Trường hợp ngắt theo sườn sẽ cho ta kết quả giống như giải pháp không dùng ngắt mà không thêm thao tác đợi chân P3.2 trở về mức 1. Hiện tượng sẽ là giả sử LED đang sáng, khi ta bấm phím và nhả tay ra sau đó, có thể LED sẽ vẫn sáng chứ không tắt!

Ngắt của 8051 được cho phép bởi 2 cấp, thứ nhất là cấp dành riêng cho mỗi ngắt, cấp thứ hai là cấp chung cho tất cả các ngắt. Như vậy để cho phép một ngắt xảy ra thì phải thỏa mãn cả 2 cấp đó: cho phép riêng và cho phép chung. Các bit quy định các nấc này đều nằm trong thanh ghi IE. Ngoài ra để sử dụng một ngắt ta phải viết chương trình phục vụ ngắt đó và đặt một lệnh nhảy tới chương trình đã viết vào địa chỉ của vector ngắt tương ứng. Đó là do khi xảy ra ngắt, CPU sẽ tự động (xin nhớ là tự động!) thực hiện lệnh đặt tại địa chỉ của vector ngắt tương ứng. Trong trường hợp này (trường hợp ngắt ngoài số 0) thì địa chỉ vector ngắt là 03h trong bộ nhớ chương trình.

Chương trình giải quyết bài toán trên sẽ như sau:

```
        …
        org     00h
        ljmp    main
        org     03h
        ljmp    ngat_0
        org     40h
main:
        mov     SP,#5fh         ;khởi tạo ngăn xếp vì có dùng ngắt
        setb    it0             ;đặt ngắt chế độ ngắt theo sườn cho ngắt ngoài 0, nếu muốn
                                ngắt theo mức thì không cần có lệnh này vì khi 8051 reset
                                xong, tự động mặc định là ngắt theo mức (IT0 = 0)
        setb    ex0             ;cho phép ngắt ngoài 0 cấp riêng
        setb    ea              ;cho phép ngắt cấp chung
        ;khác với chương trình trước, bây giờ sử dụng ngắt nên CPU không cần phải thăm dò
        mức điện áp của chân P3.2 mà sẽ có thời gian làm các việc khác, khi có sự kiện phím
        bấm xảy ra, cấu trúc ngắt sẽ tự phát hiện và thông báo cho CPU để xử lý và đáp ứng
main_loop:
        ;…có thể làm gì tùy thích ở đây…
        sjmp    loop
```

```
ngat_0:
        mov     a,P1
        cpl     a
        mov     P1,a
        reti
end
```

Qua bài tập này ta thấy được ưu điểm của ngắt là CPU được tự do làm các công việc khác, khi nào có sự kiện xảy ra thì ngắt sẽ tự báo cho CPU để dừng công việc đang làm lại và đáp ứng xử lý.