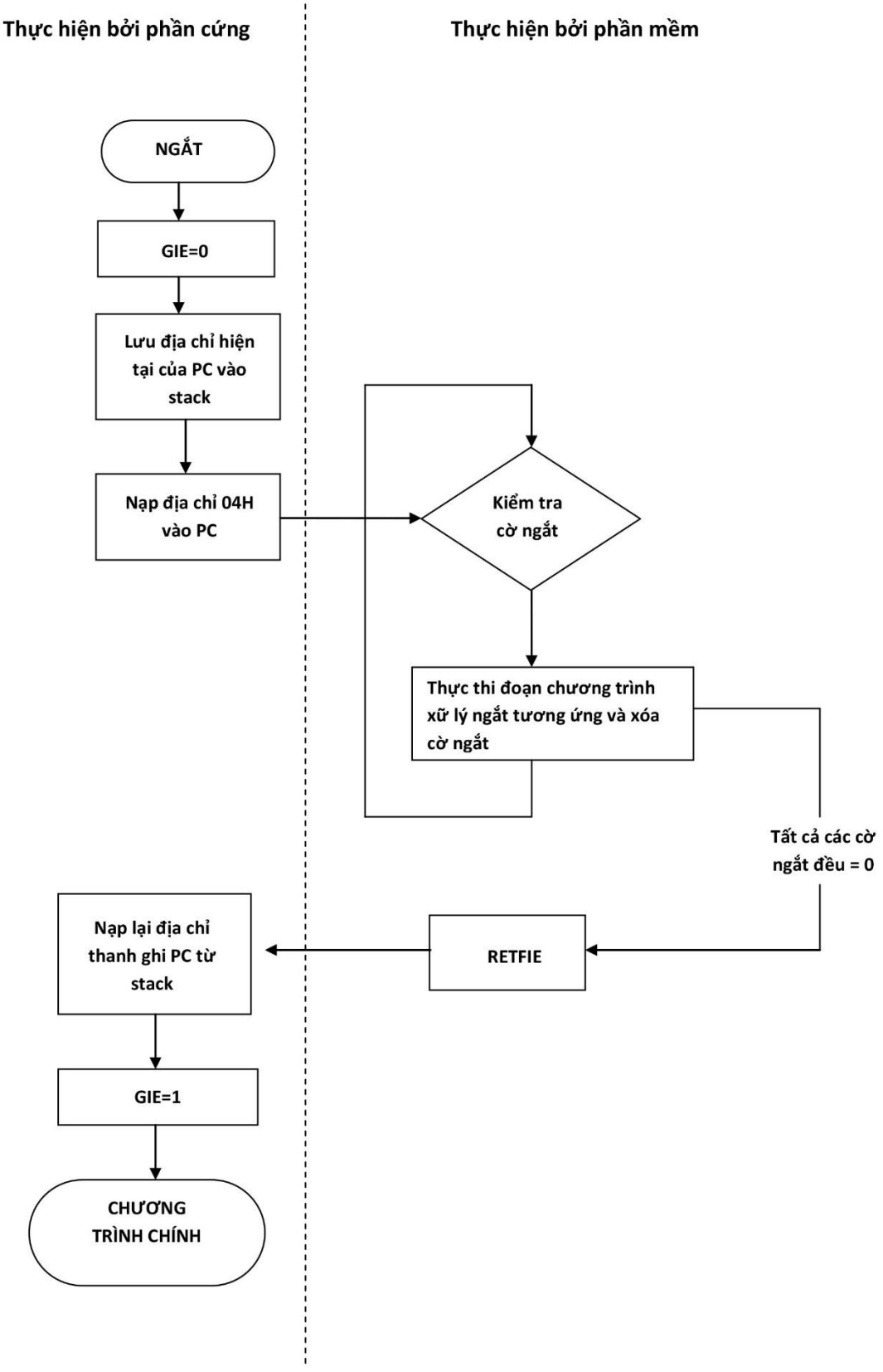
NGẮT VỚI VI ĐIỀU KHIỂN PIC

CHƯƠNG TRÌNH BIÊN DỊCH CCS

Ngắt đến từ nhiều nguồn gây ra sự kiện ngắt bên trong Pic cũng như các sự kiện gây ngắt do bên ngoài. Khi một ngắt xảy ra, trình xử lý ngắt diễn ra như hình bên dưới


Thực hiện bởi phần cứng
Thực hiện bởi phần mềm
NGẮT
GIE=0
Lưu địa chỉ hiện tại của PC vào stack
Nạp địa chỉ 04H vào PC
Kiểm tra cờ ngắt
Thực thi đoạn chương trình xữ lý ngắt tương ứng và xóa cờ ngắt
Tất cả các cờ ngắt đều = 0
RETFIE
Nạp lại địa chỉ thanh ghi PC từ stack
GIE=1
CHƯƠNG TRÌNH CHÍNH

Thực hiện bởi phần cứng: là quá trình thực thi được thực hiện bởi phần cứng của vi xử lý để xử lý ngắt một cách tự động, không phải do phần mềm.

Thực hiện bởi phần mềm: Là chương trình do ta viết để xử lý ngắt đó. Ví dụ như sao lưu các thanh ghi, hỏi vòng cờ ngắt, nhảy đến thực thi chương trình xử lý ngắt tương ứng

Quy trình xử lý ngắt diễn ra như sau:

Khi ngắt xảy ra, phần cứng tự động xóa bit cho phép ngắt toàn cục GIE=0, lưu địa chỉ của con trỏ PC vào ngăn xếp stack, nạp địa chỉ 0x04 vào thanh ghi PC , sao lưu thanh ghi W, STATUS..., sau đó nó mới hỏi vòng xem cờ ngắt nào xảy ra thì nhảy đến hàm phục vụ ngắt đó. Sau khi xử lý ngắt xong, chương trình phục hồi lại các thanh ghi trên, thực thi lệnh RETFIE để nạp lại địa chỉ cho thanh ghi PC từ bộ nhớ stack. Bật bit ngắt toàn cục GIE=1 là tiếp tục thực thi công việc tiếp theo tại thời điểm trước khi xãy ra ngắt.

Điều gì xảy ra nếu chương trình dùng nhiều ngắt và khi có ngắt thì có 2 ngắt trở lên xảy ra đồng thời ? Nghĩa là : 2 ngắt xảy ra cùng lúc , hay khi ngắt A kích hoạt và CCS đang lưu các thanh ghi (chưa tới hỏi vòng cờ ngắt) thì ngắt B xảy ra , dĩ nhiên ngắt B không thể kích vector ngắt nhảy tới 004h vì bit cho phép ngắt toàn cục (GIE) bị khóa tự động khi có ngắt , chỉ có cờ ngắt B bật mà thôi. Sau khi lưu các thanh ghi , chương trình kiểm tra cờ ngắt, rõ ràng là nếu bit nào được kiểm tra trước thì phục vụ trước, dù nó xảy ra sau . Để tránh phục vụ không đúng chỗ, bạn dùng #priority để xác định ưu tiên ngắt . Ngắt ưu tiên nhất sẽ luôn được hỏi vòng trước. Sau khi xác định cờ ngắt cần phục vụ, nó sẽ thực thi hàm ngắt tương ứng. Xong thì xoá cờ ngắt đó và thoát ngắt. Phục vụ ngắt nào xong thì chỉ xoá cờ ngắt đó. Nếu A ưu tiên hơn B thì sau khi làm A, chương trình xoá cờ ngắt A, nhưng cờ B không xoá (vì đâu có phục vụ), nên khi thoát ra ngắt A, nó sẽ lại ngắt tiếp (vì cờ B đã bật), lại hỏi vòng cờ ngắt từ đầu : nếu cờ A chưa bật thì xét B, lúc này B bật nên phục vụ B , xong thì xoá cờ B và thoát ngắt. Một chương trình dùng nhiều ngắt phải lưu ý điều này , tránh trường hợp : ngắt xảy ra liên tục (tràn ngắt), 1 ngắt bị đáp ứng trễ, ngắt không đúng , . . .

Lệnh tiền xử lý **#priority** để thiết lập thứ tự ưu tiên ngắt:

Ví dụ:

```
#priority rtcc, rb, tmr0, portb //Ngắt nào đứng trước sẽ được ưu tiên hỏi vòng kiểm tra cờ ngắt trước
```

Ví dụ một chương trình ngắt nhận dữ liệu cổng nối tiếp trong CCS như sau:

#INT_GLOBE : Ta sử dụng chỉ thị này để tự mình quản lý xử lý ngắt theo cách riêng. Khi có chỉ thị này chương trình không tự động tạo code để lưu các thanh ghi khi có ngắt, phục hồi các thanh ghi khi kết thúc ngắt bị vô hiệu hóa, do vậy ta tự viết code riêng để xữ lý.

#INT_DEFAULT: Hàm theo sau chỉ thị này sẽ được gọi nếu Pic trigger một ngắt nhưng không có cờ ngắt nào được set. nếu cờ ngắt được thiết lập nhưng không có trigger ngắt hàm này cũng sẽ được gọi.

```
#int_default

default_isr()

{

  printf("Unexplained interrupt ");

}
```

#INT_xxxx:

với xxxx là các loại ngắt, đi sau chỉ thị tiền xữ lý này là hàm xữ lý ngắt cho loại ngắt đó ví dụ:

Ví dụ:

```
#int_rda
rs232_handler() //chương trình xữ lý ngắt nhận cổng nối tiếp
{
b=getch(); //load character
Buffer[Buff+1]=b; //store character
Buff++; //increment pointer
}
main()
{
enable_interrupts(INT_RDA); // cho phép ngắt nhận
enable_interrupts(GLOBAL); // cho phép ngắt toàn cục
do { while(True); }
}
```

Các chỉ thị ngắt thông dụng

- #INT_AD : chuyển đổi A /D đã hoàn tất , thường thì không nên dùng
- #INT_ADOF : I don't know
- #INT_BUSCOL : xung đột bus
- #INT_BUTTON : nút nhấn (không biết hoạt động thế nào)
- #INT_CCP1 : có Capture hay compare trên CCP1
- #INT_CCP2 : có Capture hay compare trên CCP2
- #INT_COMP : kiểm tra bằng nhau trên Comparator
- #INT_EEPROM : hoàn thành ghi EEPROM
- #INT_EXT : ngắt ngoài
- #INT_EXT1 : ngắt ngoài 1
- #INT_EXT2 : ngắt ngoài 2
- #INT_I2C : có hoạt động I 2C
- #INT_LCD : có hoạt động LCD
- #INT_LOWVOLT : phát hiện áp thấp
- #INT_PSP : có data vào cổng Parallel slave
- #INT_RB : bất kỳ thay đổi nào trên chân B4 đến B7
- #INT_RC : bất kỳ thay đổi nào trên chân C4 đến C7

- #INT_RDA : data nhận từ RS 232 sẵn sàng
- #INT_RTCC : tràn Timer 0
- #INT_SSP : có hoạt động SPI hay I 2C
- #INT_TBE : bộ đệm chuyển RS 232 trống
- #INT_TIMER0 : một tên khác của #INT_RTCC
- #INT_TIMER1 : tràn Timer 1
- #INT_TIMER2 : tràn Timer 2
- #INT_TIMER3 : tràn Timer 3
- #INT_TIMER5 : tràn Timer 5
- #INT_OSCF : lỗi OSC
- #INT_PWMTB : ngắt cuả PWM time base
- #INT_IC3DR : ngắt đổi hướng (direct) của IC 3
- #INT_IC2QEI : ngắt của QEI
- #INT_IC1 : ngắt IC 1

- Hàm đi kèm phục vụ ngắt không cần tham số vì không có tác dụng .
- Sử dụng NOCLEAR sau #int_xxx để CCS không xoá cờ ngắt của hàm đó .
- Để cho phép ngắt đó hoạt động phải dùng lệnh enable_interrupts (int_xxxx) và enable_interrupts (global) .
- Khoá FAST theo sau #int_xxxx để cho ngắt đó là ưu tiên cao, chỉ được 1 ngắt thôi , chỉ có ở PIC 18 và dsPIC .
  VD : #int_timer0 FAST NOCLEAR