ریزپردازنده

دانشکده کامپیوتر دانشگاه یزد

نیم‌سال دوم تحصیلی ۹۷-۹۶

ارائه‌دهنده : پریسا استواری

دانشگاه یزد

وقفه‌ها

وقفه

- وقفه‌یک اتفاق داخلی یا خارجی است که کار اصلی میکروکنترلر را متوقف می‌کند تا میکروکنترلر ابتدا به آن وقفه سرویس‌دهی نماید.
  - در این صورت برنامه‌ی در حال اجرا در میکروکنترلر متوقف می‌شود و ابتدا برنامه‌ی مربوط به سرویس وقفه اجرا می‌گردد و سپس باز کنترل به برنامه‌ی متوقف شده باز می‌گردد و اجرای برنامه از جایی که متوقف شده است ادامه می‌یابد.

- بدین ترتیب به نظر می‌رسد CPU چندین کار را همزمان انجام می‌دهد. اما در واقع CPU بین برنامه‌ی اصلی و برنامه‌ی سرویس‌های مورد نیاز دستگاه‌ها مختلف سوییچ می‌کند.

وقفه

- برای اینکه میکروکنترلر بتواند به چندین دستگاه پاسخ دهد از دو روش می‌تواند استفاده کند:

- وقفه (Interrupt)
  - هر زمان یک دستگاهی به سرویس میکروکنترلر نیاز داشته باشد، یک سیگنال وقفه به میکروکنترلر می‌فرستد.
  - هنگامی میکروکنترلر وقفه را دریافت نمود، برنامه در حال اجرا را متوقف می‌نماید و به وقفه سرویس‌دهی می‌کند.
  - برنامه‌ای که برای سرویس‌دهی به وقفه است به نام روتین سرویس وقفه (Interrupt Service Routine) (ISR) یا interrupt handler نامیده می‌شود.

وقفه

- برای اینکه میکروکنترلر بتواند به چندین دستگاه پاسخ دهد از دو روش می‌تواند استفاده کند:

- Polling
  - میکروکنترلر به صورت مکرر وضعیت دستگاه‌ها را چک می‌کند.
  - هر زمان که وضعیت دستگاه نشان‌دهنده‌ی نیاز به سرویس باشد، میکروکنترلر به دستگاه سرویس می‌دهد.
  - باز میکروکنترلر به چک کردن وضعیت دستگاه‌ها می‌پردازد.

- Polling روش مناسبی نمی‌باشد، زیرا باید میکروکنترلر زمان زیادی از وقت CPU را صرف چک کردن وضعیت دستگاه‌هایی بنماید که به سرویس‌دهی نیازی ندارند.
  - مثال : JNB TF1, HERE یا JNB TI, HERE

ISR

- روتین سرویس وقفه (ISR)
  - برای هر وقفه‌یک روتین سرویس وقفه نیاز است.

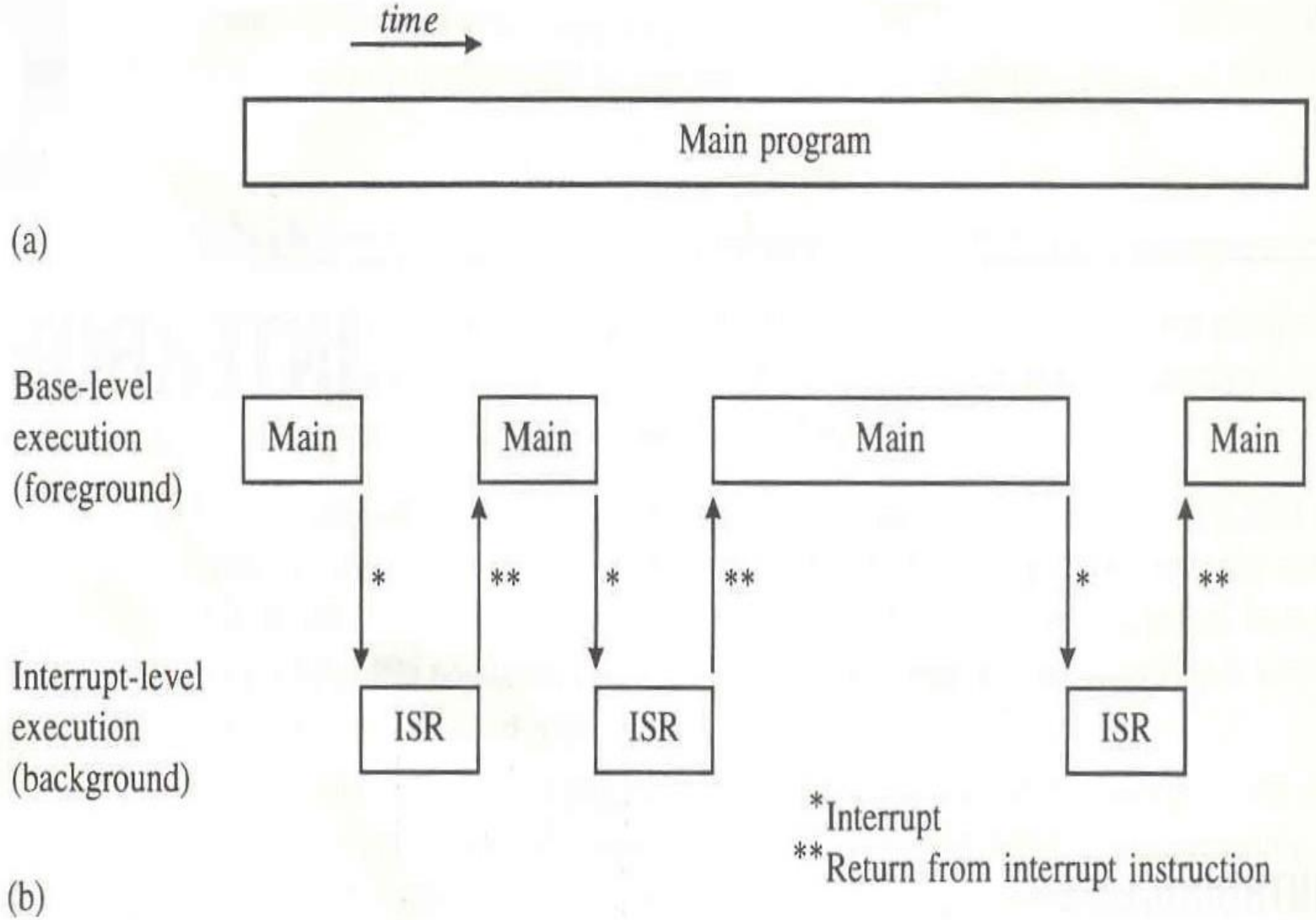
- برنامه ایست که برای اداره کردن دستگاه خاصی نوشته می‌شود.
  - برای مثال در پورت سری، برنامه‌ای که کاراکتر دریافت شده را در جایی ذخیره نماید. یا برنامه‌ای که بعد از اتمام ارسال یک کاراکتر، کاراکتر بعدی را در SBUF بار نماید.
  - برای مثال در تایمرها، برنامه‌ای که پس از رخ دادن سرریز یک تایمر، یک عملی انجام پذیرد.

- در واقع روتین سرویس وقفه همانند یک زیربرنامه است.
  - با این تفاوت که روتین سرویس وقفه هر زمان که یک رویداد اتفاق افتاد فراخوانی می‌شود.
  - اما یک زیربرنامه در جاهای مشخصی از برنامه صدا زده می‌شود و کاملا مشخص است چه زمانی صدا زده خواهد شد.

ISR

- زمانی که یک وقفه پیش می‌آید، برنامه‌ی اصلی متوقف می‌شود، میکروکنترلر روتین سرویس وقفه‌ی فعال شده را اجرا می‌کند و پس از پایان به برنامه‌ی اصلی باز می‌گردد و ادامه‌ی برنامه‌ی اصلی را اجرا می‌کند.

- معمولا گفته می‌شود برنامه‌ی اصلی در سطح پایه (foreground) و برنامه‌ی سرویس وقفه در سطح وقفه (background) اجرا می‌گردد.



(a) اجرای برنامه بدون وقفه
(b) اجرای برنامه با وقفه

وقفه در 8051

- پنج منبع وقفه وجود دارد.
  - دو وقفه خارجی
  - دو پایه P3.2 و P3.3 برای وقفه‌های خارجی INT0 و INT1 است.
- دو وقفه یکی برای تایمر یک و یکی برای تایمر صفر
  - هر زمان تایمر سرریز دهد، یک وقفه رخ می‌دهد.
  - به عبارتی هر زمان TF1 یک شود، وقفه تایمر یک فعال می‌گردد و هر زمان TF0 یک شود، وقفه تایمر صفر فعال می‌شود.
- یک وقفه برای پورت سری
  - هر زمان کاراکتری دریافت یا ارسال گردد، وقفه پورت سری فعال می‌گردد.
  - به عبارتی هر زمان TI یا RI برابر با یک شد، وقفه فعال می‌گردد.
- ریست را هم می‌توان به عنوان وقفه ششم در نظر گرفت.

وقفه در 8051

- وقتی میکروکنترلر ریست می‌شود، تمام وقفه‌ها غیر فعال می‌گردند.
  - هر یک از وقفه‌ها را باید با دستور فعال نمود.

- اگر دو یا چند وقفه همزمان فعال شوند، وقفه‌ها با توجه به اولویتشان سرویس‌دهی می‌گردند.
  - اولویت وقفه‌ها قابل برنامه‌ریزی است.

فعال و غیرفعال کردن وقفه‌ها

- هر کدام از وقفه‌ها از طریق ثبات فعال‌ساز وقفه (Interrupt Enable) (IE) قابل فعال‌سازی اند.
- ثبات IE در خانه شماره A8H حافظه قرار دارد.
  - به صورت بیتی آدرس پذیر است.

شماره بیت	سمبول بیت		آدرس بیت	توضیحات
IE.7	EA	Enable All	AFH	فعال کردن یا غیر فعال کردن کل وقفه‌ها
IE.6	---	---	AEH	تعریف نشده
IE.5	~~ET2~~	~~Enable Timer 2~~	ADH	فعال کردن وقفه تایمر ۲ (در 8052)
IE.4	ES	Enable Serial Port	ACH	فعال کردن وقفه پورت سری
IE.3	ET1	Enable Timer 1	ABH	فعال کردن وقفه تایمر ۱
IE.2	EX1	Enable External 1	AAH	فعال کردن وقفه ۱ خارجی
IE.1	ET0	Enable Timer 0	A9H	فعال کردن وقفه تایمر ۰
IE.0	EX0	Enable External 0	A8H	فعال کردن وقفه ۰ خارجی

فعال و غیرفعال کرده وقفه‌ها

- برای فعال کردن وقفه، دو بیت باید یک شوند.
  - یکی برای فعال کردن وقفه‌ی مورد نظر
  - یکی برای فعال کردن کلیه وقفه‌ها EA

- مثال : فعال کردن وقفه تایمر یک
  - SETB ET1
  - SETB EA
  - یا MOV IE, #10001000B

فعال و غیرفعال کرده وقفه‌ها

- مثال : الف) وقفه پورت سری، تایمر صفر و وقفه‌ی خارجی یک را فعال کنید.

ب) وقفه تایمر صفر را غیر فعال کنید.

ج) تمام وقفه‌ها را با یک دستور غیر فعال کنید.

Solution:

(a)
```
MOV   IE,#10010110B ;enable serial,
                    ;timer 0, EX1
```

Another way to perform the same manipulation is

```
SETB IE.7   ;EA=1, global enable
SETB IE.4   ;enable serial interrupt
SETB IE.1   ;enable Timer 0 interrupt
SETB IE.2   ;enable EX1
```

(b)
```
CLR   IE.1    ;mask (disable) timer 0
              ;interrupt only
```

(c)
```
CLR   IE.7    ;disable all interrupts
```

عملیات وقفه

- هنگامی که یک وقفه رخ می‌دهد، میکروکنترلر وارد مراحل زیر می‌شود.
  - اجرای دستور فعلی تمام می‌شود.
  - محتوای فعلی PC در پشته ذخیره می‌شود. (برای اینکه بتوان بعدا برنامه را از جایی که قطع شده ادامه داد.)
  - وضعیت تمام بیت‌های وقفه ذخیره می‌شوند.
  - به محل ثابتی از حافظه به نام بردار وقفه پرش می‌شود. در این محل آدرس ISR ها نگه‌داری می‌شود.
  - میکروکنترلر آدرس ISR مربوطه را می‌گیرد (آدرس در PC بار می‌شود) و به آن آدرس پرش می‌کند.
  - میکروکنترلر برنامه‌ی ISR را تا انتها اجرا می‌کند. در آخرین دستور ISR دستور RETI وجود دارد.
  - در هنگام اجرای دستور RETI آدرس دستوری که میکروکنترلر در آن وقفه خورده بود در PC بار می‌شود. مقدار بیت‌های وضعیت وقفه نیز باز می‌گردند و برنامه از مکانی که وقفه خورده بود اجرا می‌شود.
    - PC آدرس برگشت را از دو بایت اول پشته بر می‌دارد.

بردارهای وقفه

- هنگامی که وقفه رخ می‌دهد مقداری که در PC بار می‌شود بردار وقفه نام دارد. که آدرس شروع ISR است.

INTERRUPT	FLAG	VECTOR ADDRESS
System reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial port	RI or TI	0023H
Timer 2	TF2 or EXF2	002BH

بردارهای وقفه

- هنگامی که وقفه رخ می‌دهد مقداری که در PC بار می‌شود بردار وقفه نام دارد. که آدرس شروع ISR است.

INTERRUPT	FLAG	VECTOR ADDRESS
System reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial port	RI or TI	0023H
Timer 2	TF2 or EXF2	002BH

Reset را می‌توان شبیه وقفه در نظر گرفت.
هر وقت میکروکنترلر ریست می‌شود مقدار PC 0000H می‌شود که برنامه از ابتدای ROM شروع به اجرا شدن می‌کند.

بردارهای وقفه

- هنگامی که وقفه رخ می‌دهد مقداری که در PC بار می‌شود بردار وقفه نام دارد. که آدرس شروع ISR است.

INTERRUPT	FLAG	VECTOR ADDRESS
System reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial port	RI or TI	0023H
Timer 2	TF2 or EXF2	002BH

برای این چهار وقفه، هنگامی که آدرس روتین وقفه در PC بار می‌شود، بیت پرچم وقفه‌ای که باعث وقفه شده به طور اتوماتیک توسط سخت افزار صفر می‌شود. یعنی TFO یا TF1 یا IE0 یا IE1 هر کدام که وقفه تولید کرده صفر می‌شود.

IE : Interrupt External

بردارهای وقفه

- هنگامی که وقفه رخ می‌دهد مقداری که در PC بار می‌شود بردار وقفه نام دارد. که آدرس شروع ISR است.

INTERRUPT	FLAG	VECTOR ADDRESS
System reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial port	RI or TI	0023H
Timer 2	TF2 or EXF2	002BH

در مورد وقفه‌های پورت سری، TI یا RI به صورت اتوماتیک صفر نمی‌شوند.
چون دو منبع برای یک وقفه وجود دارد، پاک کردن بیت پرچم وقفه توسط CPU عملی نیست.
باید در سرویس روتین وقفه ISR به صورت دستی چک و صفر شوند.

طراحی برنامه با استفاده از وقفه

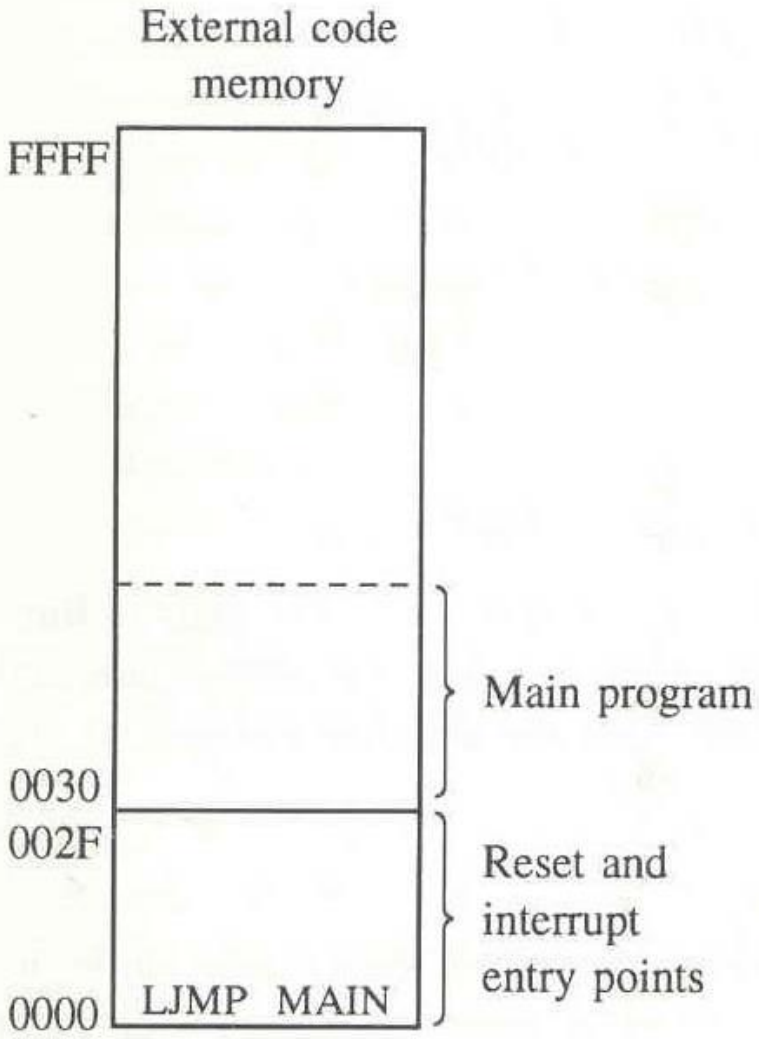
- تمام برنامه‌هایی که در فصل تایمرها و پورت سری نوشته شده بدون استفاده از وقفه بود.
  - در حلقه‌های انتظار بیت‌های پرچم TF، TI یا RI چک می‌شد.
  - مشکل این است تمام وقت CPU صرف انتظار می‌شود. مثلا CPU منتظر دریافت یک کاراکتر در پورت سری است که هیچ وقت قرار نیست دریافت کند!

- در صورت استفاده از وقفه باید ساختار برنامه تغییر یابد.

```
        ORG 0000H           ; reset entry point
        LJMP MAIN           ; ISR entry point
        .....
        ORG 0030H           ; main program entry point
MAIN: .....                 ; program begins
```

طراحی برنامه با استفاده از وقفه

- ابتدای برنامه یک دستور پرش به آدر 30H صورت می‌گیرد.
  - آدرس 00H تا 30H مربوط به روتین‌های سرویس وقفه‌ها است.
  - اولین دستور برنامه‌ی اصلی در آدرس 30H قرار می‌گیرد.



INTERRUPT	FLAG	VECTOR ADDRESS
System reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial port	RI or TI	0023H
Timer 2	TF2 or EXF2	002BH

طراحی برنامه با استفاده از وقفه

- موقعی که میکروکنترلر ریست می‌شود، از آدرس 0H شروع به اجرای برنامه می‌کند. معمولا در این آدرس یک دستور LJMP قرار دارد که به مکان اول برنامه پرش می‌کند.

- سرویس روتین وقفه تایمر صفر از آدرس 000BH شروع می‌شود و در آدرس 0012H پایان می‌یابد. تایمر یک نیز از 1BH تا 22H است.
- و به همین صورت بقیه

- برای هر روتین سرویس وقفه ۸ بایت در نظر گرفته شده است.
- اگر برنامه‌های مربوط به سرویس وقفه کوچک باشد، می‌تواند در همین ۸ نوشته شود.
- اگر برنامه‌های مربوط به سرویس وقفه بزرگتر از ۸ بایت باشد، باید در سرویس روتین وقفه به یک محلی از حافظه پرش کند و برنامه‌ها در آنجا نوشته شود.

روتین سرویس وقفه کوچک و بزرگ

- مثلا برای تایمر صفر با ISR کوچک :

```
        ORG 0H
        LJMP MAIN
        ORG 0BH
T0ISR:  …
        …
        RETI
        ORG 30H
MAIN: …
        …
```

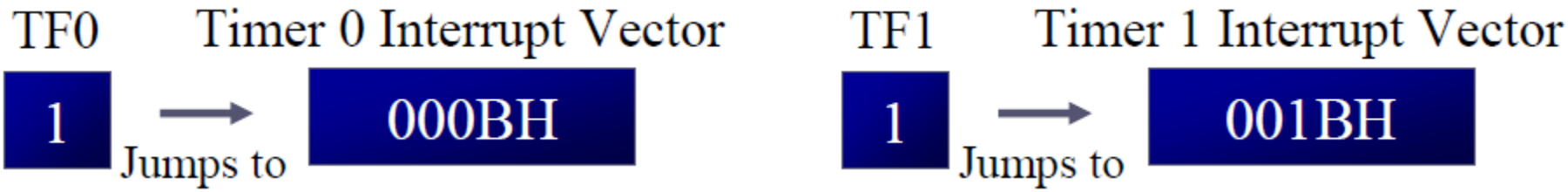
- مثلا برای تایمر صفر با ISR بزرگ :

```
        ORG 0H
        LJMP MAIN
        ORG 0BH
        LJMP T0ISR
        ORG 30H
MAIN: …
        …
T0ISR: …
        …
        RETI
```

وقفه‌ی تایمر

- در حالت Polling، صبر می‌کردیم تا زمانی که TF یک شود.
  - در این زمان دیگر نمی‌توان کار دیگری انجام داد.

- در حالت وقفه‌این مشکل حل می‌شود.
  - اگر وقفه‌ی تایمر در IE فعال باشد، هر زمان که تایمر سرریز داد و TF یک شد، برنامه‌ی میکروکنترلر هر جا که باشد متوقف می‌شود و PC به بردار وقفه‌ی تایمر می‌پرد تا به وقفه‌ی تایمر سرویس‌دهی کند.
  - بدین صورت میکروکنترلر می‌تواند به کارهای دیگر بپردازد و هر زمان که تایمر نیاز به سرویس داشت متوجه شود.



وقفه‌ی تایمر

- مثال : برنامه‌ای بنویسید که به طور مکرر، ۸ بیت داده از پورت P0 بگیرد و به پورت P1 ارسال نماید. و به طور هم زمان موج مربعی با دوره 200µs در P2.1 ایجاد نماید. از تایمر صفر برای ایجاد موج مربعی استفاده شود. فرض کنید فرکانس اسیلاتور 11.0592MHz است.

Solution:

We will use timer 0 in mode 2 (auto reload). TH0 = 100/1.085 us = 92

```
;--upon wake-up go to main, avoid using
;memory allocated to Interrupt Vector Table
     ORG  0000H
     LJMP MAIN  ;by-pass interrupt vector table
;
;--ISR for timer 0 to generate square wave
     ORG  000BH ;Timer 0 interrupt vector table
     CPL  P2.1  ;toggle P2.1 pin
     RETI       ;return from ISR
...
```

وقفه‌ی تایمر

- مثال : برنامه‌ای بنویسید که به طور مکرر، ۸ بیت داده از پورت P0 بگیرد و به پورت P1 ارسال نماید. و به طور هم زمان موج مربعی با دوره 200µs در P2.1 ایجاد نماید. از تایمر صفر برای ایجاد موج مربعی استفاده شود. فرض کنید فرکانس اسیلاتور 11.0592MHz است.

```
;--The main program for initialization
        ORG   0030H     ;after vector table space
MAIN:   MOV   TMOD,#02H ;Timer 0, mode 2
        MOV   P0,#0FFH  ;make P0 an input port
        MOV   TH0,#-92  ;TH0=A4H for -92
        MOV   IE,#82H   ;IE=10000010 (bin) enable
                        ;Timer 0
        SETB  TR0       ;Start Timer 0
BACK:   MOV   A,P0      ;get data from P0
        MOV   P1,A      ;issue it to P1
        SJMP  BACK      ;keep doing it loop
                        ;unless interrupted by TF0
        END
```

مثال

- برنامه‌ی مثال قبل را طوری بازنویسی کنید که موج مربعی به مدت 1085µs یک و به مدت 15µs صفر باشد.

Solution:

Since 1085 us is 1000 × 1.085 we need to use mode 1 of timer 1.

```
;--upon wake-up go to main, avoid using
;memory allocated to Interrupt Vector Table
        ORG   0000H
        LJMP MAIN       ;by-pass int. vector table
;--ISR for timer 1 to generate square wave
        ORG   001BH     ;Timer 1 int. vector table
        LJMP ISR_T1     ;jump to ISR
...
```

```
;--The main program for initialization
        ORG   0030H      ;after vector table space
MAIN:   MOV   TMOD,#10H ;Timer 1, mode 1
        MOV   P0,#0FFH   ;make P0 an input port
        MOV   TL1,#018H ;TL1=18 low byte of -1000
        MOV   TH1,#0FCH ;TH1=FC high byte of -1000
        MOV   IE,#88H ;10001000 enable Timer 1 int
        SETB  TR1        ;Start Timer 1
BACK:   MOV   A,P0       ;get da
        MOV   P1,A       ;issue
        SJMP  BACK       ;keep d
;Timer 1 ISR. Must be reloaded, not auto-reload
ISR_T1: CLR TR1          ;stop Timer 1
        MOV R2,#4        ;                          2MC
        CLR P2.1        ;P2.1=0, start of low portion
HERE:   DJNZ R2,HERE ;4x2 machine cycle             8MC
        MOV TL1,#18H ;load T1 low byte value        2MC
        MOV TH1,#0FCH;load T1 high byte value       2MC
        SETB TR1         ;starts timer1             1MC
        SETB P2.1        ;P2.1=1,back to high       1MC
        RETI             ;return to main
        END
```

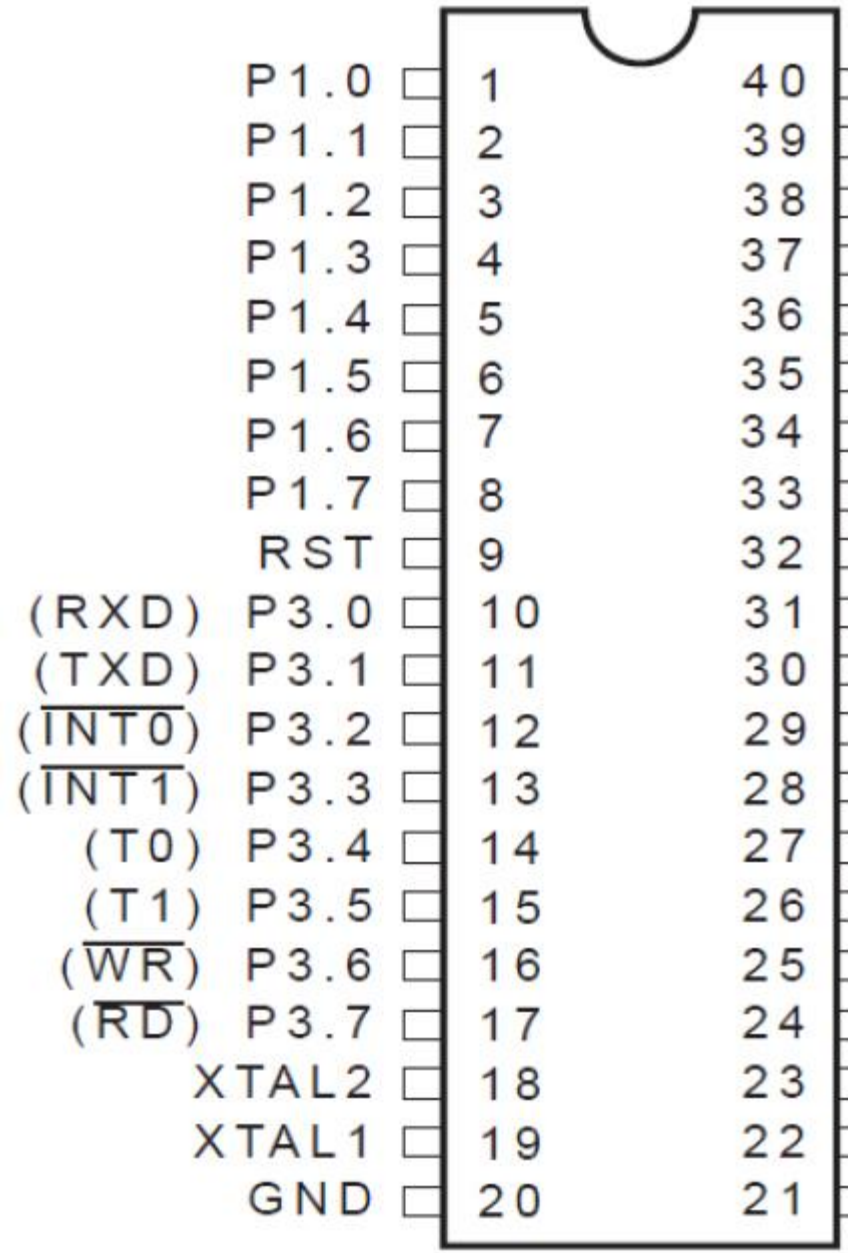
Low portion of the pulse is created by 14 MC
14 x 1.085 us = 15.19 us

مثال

- برنامه‌ای بنویسید که موج مربعی با فرکانس 50Hz در پین P2.1 ایجاد نماید. از وقفه‌ی تایمر صفر استفاده کنید. فرض کنید فرکانس اسیلاتور 11.0592MHz است.

Solution:

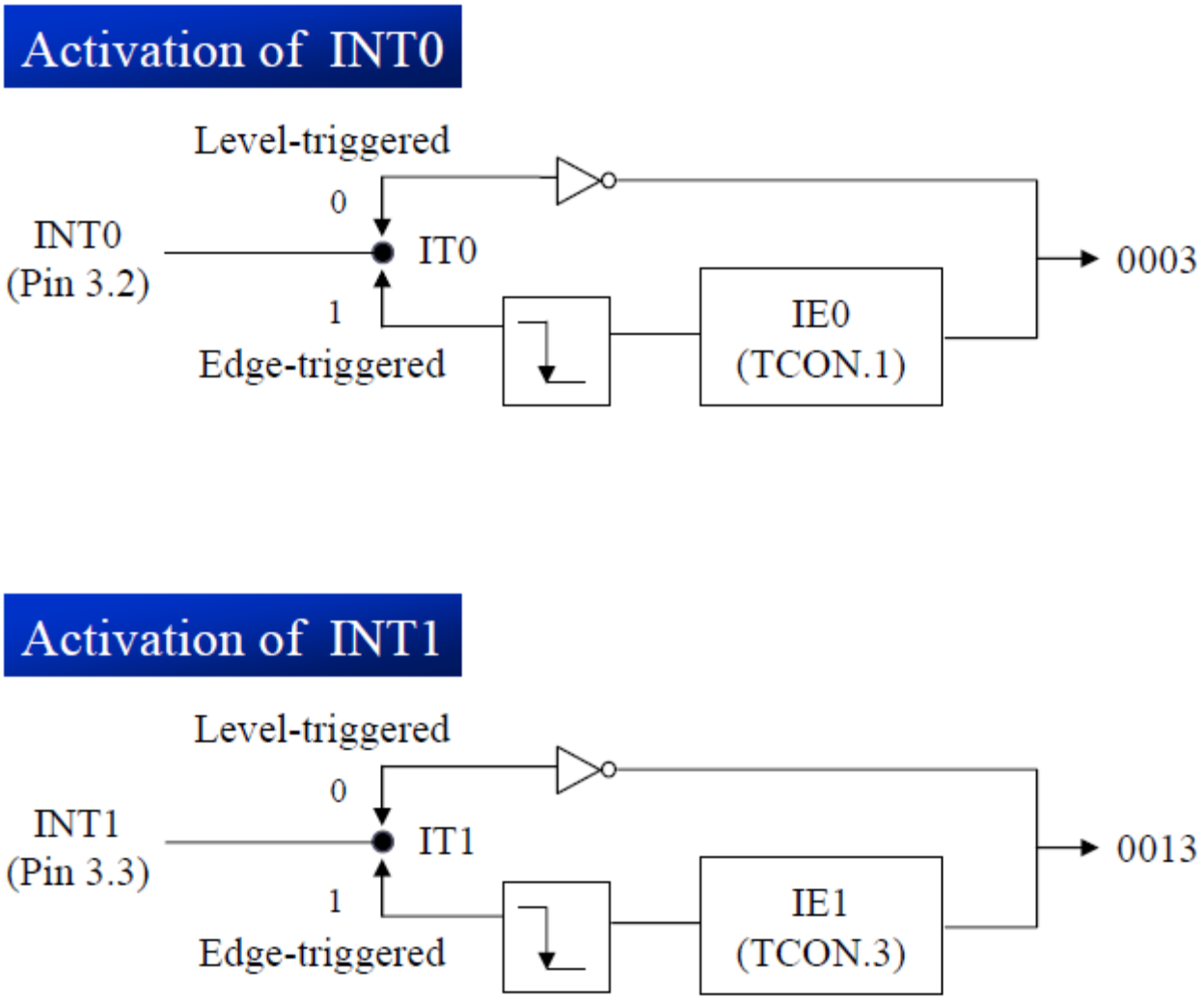
```
      ORG   0
      LJMP MAIN
      ORG   000BH   ;ISR for Timer 0
      CPL   P1.2
      MOV   TL0,#00
      MOV   TH0,#0DCH
      RETI
      ORG   30H
;---------main program for initialization
MAIN:MOV   TM0D,#00000001B  ;Timer 0, Mode 1
      MOV   TL0,#00
      MOV   TH0,#0DCH
      MOV   IE,#82H    ;enable Timer 0 interrupt
      SETB TR0
HERE:SJMP HERE
      END
```



وقفه سخت‌افزاری خارجی

- در 8051 دو وقفه خارجی داریم.
- پین 12 (P3.2) و پین 13 (P3.3) برای INT0 و INT1 طراحی شده‌اند.
- بردار وقفه‌ها آنها برابر با 03H و 13H است.
- برای فعال شدن وقفه‌ی خارجی دو راه وجود دارد :
  - فعال شدن به سطح
  - فعال شدن به لبه

فعال شدن وقفه خارجی



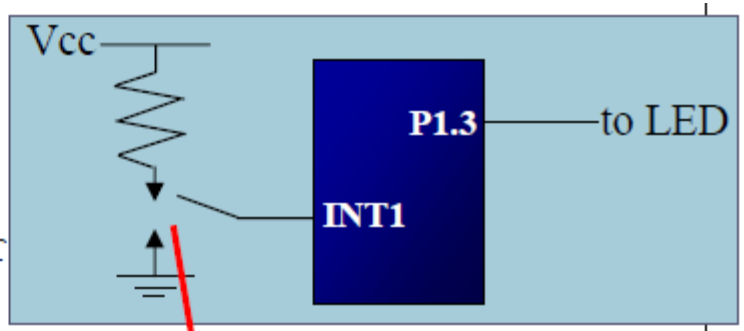
فعال شدن به سطح

- در این حالت باید پین های INT0 یا INT1 در حالت عادی یک باشند.

- اگر یک سیگنال صفر وارد شود، وقفه رخ می‌دهد.
  - در این حالت میکروکنترلر کارش را متوقف می‌کند و به بردار وقفه‌ی خارجی می‌پرد.
  - سیگنال صفری که در INT0 یا INT1 است باید قبل از اجرای آخرین دستور سرویس وقفه‌ی خارجی (RETI) یک شود.
    - در غیر اینصورت باز وقفه‌ی جدیدی اتفاق می‌افتد.

- به این حالت فعال شدن به سطح level-trigged یا level-activated می‌گویند و حالت پیش فرض 8051 این مود است.

- مثال : فرض کنید پین INT1 به یک سوییچ متصل است. این سوییچ همیشه در حالت یک است. وقتی سوییچ به حالت صفر برود، می‌خواهیم یک LED روشن شود. LED به پایه P1.3 متصل است و در حالت عادی خاموش است. اگر سوییچ صفر شود، تا زمانی که سوییچ صفر است LED باید روشن بماند.



Solution:

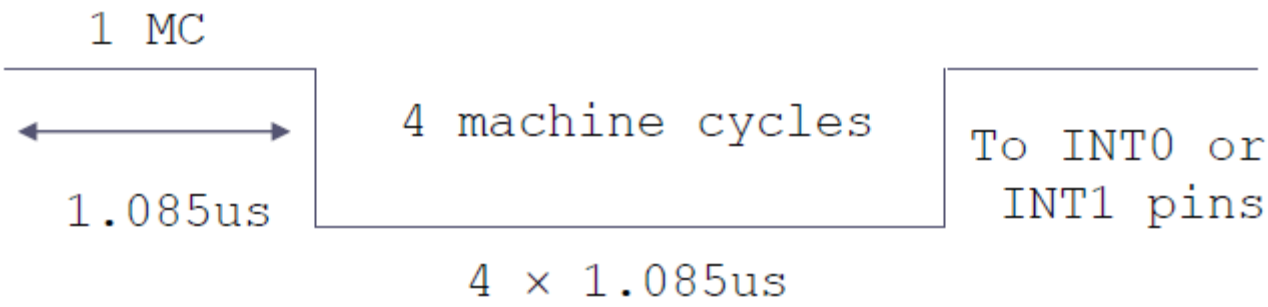
```
        ORG   0000H
        LJMP MAIN ;by-pass inter
                  ;vector table
;--ISR for INT1 to turn on LED
        ORG   0013H      ;INT1 ISR
        SETB P1.3        ;turn on LED
        MOV   R3,#255
BACK:   DJNZ R3,BACK     ;keep LED on for a
        CLR   P1.3       ;turn off the LED
        RETI             ;return from ISR

;--MAIN program for initialization
        ORG   30H
MAIN:   MOV   IE,#10000100B ;enable external INT 1
HERE:   SJMP HERE       ;stay here until get interrupted
        END
```

Pressing the switch will cause the LED to be turned on. If it is kept activated, the LED stays on

نمونه‌برداری از وقفه‌ی خارجی حساس به سطح

- پین‌های P3.2 و P3.3 در حالت عادی به عنوان پین‌های ورودی خروجی معمولی استفاده می‌شوند مگر آنکه بیت‌های INT0 و INT1 در ثبات IE فعال شوند.
- برای اینکه وقفه خارجی رخ دهد :
  - تا زمان شروع اجرای روتین وقفه، باید مقدار پین INTn در حالت صفر نگه داشته شود.
  - اگر مقدار پین INTn قبل از اجرای روتین وقفه به مقدار یک بازگردد دیگر وقفه نخواهیم داشت.
  - اگر مقدار پین INTn پس از اجرای دستور RETI در روتین وقفه همچنان صفر باشد، یک وقفه‌ی دیگر اتفاق می افتد.
- برای اطمینان از وقوع وقفه‌ی خارجی، باید مقدار پین در حدود ۴ سیکل ماشین برابر با صفر باشد و نه بیشتر



وقفه‌ی خارجی حساس به لبه

- برای حساس به لبه کردن وقفه خارجی، باید بیت‌های ثبات TCON را برنامه‌ریزی کنیم.

TCON (timer control) register summary

BIT	SYMBOL	BIT ADDRESS	DESCRIPTION
TCON.7	TF1	8FH	Timer 1 overflow flag. Set by hardware upon overflow; cleared by software, or by hardware when processor vectors to interrupt service routine
TCON.6	TR1	8EH	Timer 1 run-control bit. Set/cleared by software to turn timer on/off
TCON.5	TF0	8DH	Timer 0 overflow flag
TCON.4	TR0	8CH	Timer 0 run-control bit
TCON.3	IE1	8BH	External interrupt 1 edge flag. Set by hardware when a falling edge is detected on $\overline{\text{INT 1}}$; cleared by software, or by hardware when CPU vectors to interrupt service routine
TCON.2	IT1	8AH	External interrupt 1 type flag. Set/cleared by software for falling edge/low-level activated external interrupt
TCON.1	IE0	89H	External interrupt 0 edge flag
TCON.0	IT0	88H	External interrupt 0 type flag

TCON (timer control) register summary

BIT	SYMBOL	BIT ADDRESS	DESCRIPTION
TCON.7	TF1	8FH	Timer 1 overflow flag. Set by hardware upon overflow; cleared by software, or by hardware when processor vectors to interrupt service routine
TCON.6	TR1	8EH	Timer 1 run-control bit. Set/cleared by soft-
TCON.5	TF0		
TCON.4	TR0		
TCON.3	IE1		
			routine
TCON.2	IT1	8AH	External interrupt 1 type flag. Set/cleared by software for falling edge/low-level activated external interrupt
TCON.1	IE0	89H	External interrupt 0 edge flag
TCON.0	IT0	88H	External interrupt 0 type flag

با بیت‌ها IT0 و IT1 می‌توان وقفه‌ی خارجی 0 و وقفه خارجی 1 را حساس به سطح یا لبه کرد.

اگر بیت IT صفر باشد، وقفه‌ی خارجی حساس به سطح صفر است و اگر یک باشد حساس به لبه پایین رونده است.

SETB IT1

CLR IT1

TCON (timer control) register summary

BIT	SYMBOL	BIT	
TCON.7	TF1		
TCON.6	TR1		
TCON.5	TF0	8DH	Timer 0 overflow flag
TCON.4	TR0	8CH	Timer 0 run-control bit
TCON.3	IE1	8BH	External interrupt 1 edge flag. Set by hardware when a falling edge is detected on INT 1; cleared by software, or by hardware when CPU vectors to interrupt service routine
TCON.2	IT1	8AH	External interrupt 1 type flag. Set/cleared by software for falling edge/low-level activated external interrupt
TCON.1	IE0	89H	External interrupt 0 edge flag
TCON.0	IT0	88H	External interrupt 0 type flag

زمانی‌که وقفه خارجی INT1 یا INT0 از ۱ به ۰ تغییر حالت می‌دهد، بیت‌های IE1 یا IE0 متناظر با وقفه‌ی خارجی برابر با یک می‌شوند که نشان‌دهنده‌ی وقوع وقفه است.

زمانی‌که دستور RETI در روتین سرویس وقفه اجرا شد، بیت نظیر وقفه‌ی خارجی به صورت سخت‌افزاری برابر با صفر می‌شود.

دقت شود، این دو بیت پرچم تنها برای وقفه خارجی حساس به لبه فعال می‌شوند. در وقفه‌ی خارجی حساس به سطح، وقفه‌ها ذخیره نمی‌گردند.

وقفه خارجی حساس به لبه

- مثال : فرض کنید پین P3.3 به یک منبع تولید پالس متصل است. برنامه‌ای بنویسید که هر لبه‌ی پایین‌رونده‌ی پالس، یک مقدار یک را به P1.3 بفرستد تا LED متصل به آن روشن و خاموش شود. بدین صورت، چراغ LED با سرعت پالس‌های ورودی روشن و خاموش می‌شود.

زمانی‌که یک لبه‌ی پایین رونده در پین P3.3 رخ دهد، بلافاصله ISR مربوط به وقفه‌ی خارجی ۱ اجرا شده و LED برای لحظه‌ای روشن و خاموش می‌شود.

```
        ORG   0000H
        LJMP MAIN
;--ISR for hardware interrupt INT1 to turn on LED
        ORG   0013H   ;INT1 ISR
        SETB P1.3     ;turn on LED
        MOV   R3,#255
BACK:   DJNZ R3,BACK ;keep the buzzer on for a while
        CLR   P1.3    ;turn off the buzzer
        RETI          ;return from ISR
--MAIN program for initialization
        ORG   30H
        SETB TCON.2 ;make INT1 edge-triggered int.
        MOV   IE,#10000100B ;enable External INT 1
HERE:   SJMP HERE     ;stay here until get interrupted
        END
```

LED برای مدت 500µs روشن می‌ماند و بعد خاموش می‌شود.

نمونه‌برداری از وقفه خارجی حساس به لبه

- در این حالت، ورودی خارجی حداقل برای یک سیکل ماشین باید یک باشد و سپس حداقل برای یک سیکل ماشین باید صفر باشد تا وقفه روی دهد.

- لبه‌ی پایین‌رونده در بیت‌های IE0 یا IE1 ذخیره می‌گردند.
  - زمانی‌که سرویس روتین وقفه‌ی خارجی نظیر پایان یافت، بیت نظیر وقفه‌ی خارجی به صورت سخت‌افزاری صفر می‌شود.

نمونه‌برداری از وقفه خارجی حساس به لبه

- دو نکته :
- زمانی که IRS پایان می‌یابد (پس از اجرای دستور RETI)، بیت‌های IE0 و IE1 نظیر وقفه‌ی خارجی صفر می‌شوند.
  - بدین معنی که میکروکنترلر آماده است وقفه‌ی بعدی را بپذیرد.
  - نیازی به CLR کردن بیت‌های پرچم وقفه‌ی خارجی IEn نیست.

- در زمانی‌که ISR یکی از وقفه‌های خارجی در حال اجراست، از تمامی وقفه‌ها خارجی نظیر که در حال اتفاق افتادن است صرف‌نظر می‌شود.

تفاوت RET و RETI

- هر دوی این دستورات یک عمل را انجام می‌دهند :
  - Pop کردن دو بایت از انتهای پشته و بار کردن آن در PC برای اینکه ادامه‌ی دستورات از مکانی که قطع شده است اجرا شود.

- اما در RETI یک کار اضافی نیز انجام می‌شود :
  - مقدار پرچم وقفه مربوطه را صفر می‌کند. بدین معنا که سرویس‌دهی به وقفه پایان یافته و میکروکنترلر می‌تواند وقفه‌ی جدیدی دریافت کند.
  - اگر به جای RETI از RET استفاده شود، در واقع بعد از وقفه‌ی اول، تمامی وقفه‌ها برای آن وقفه غیر فعال می‌شود. زیرا میکروکنترلر فکر می‌کند هنوز در حال سرویس‌دهی به وقفه‌ی اول است و از مابقی وقفه‌ها صرف‌نظر می‌کند.
  - پرچم‌های TF0، TF1، IE0 و IE1 با اجرای دستور RETI صفر می‌شوند.

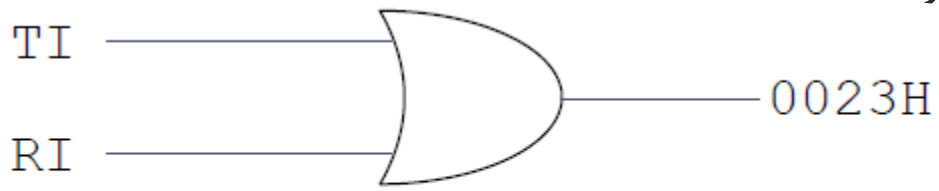
- در یک جا می‌توان به جای دستور RETI از دستور RET استفاده نمود. کجا؟

وقفه پورت سری

- در 8051 تنها یک وقفه برای هم ارسال و هم دریافت پورت سری وجود دارد.

- اگر بیت IE.4 یک باشد (وقفه‌ی پورت سری فعال باشد) هر زمان که TI یا RI یک شد، به میکروکنترلر وقفه وارد می‌شود و به آدرس 0023H حافظه‌ی ROM می‌پرد تا IRS مربوط به وقفه پورت سری را اجرا نماید.

- در ISR پورت سری باید پرچم‌های TI و RI چک شوند تا بفهمیم وقفه برای کدام یک از آنها بوده و بر اساس آن عملیات مورد نظر را انجام دهیم.
  - دقت شود باید در انتهای روتین سرویس وقفه پورت سری، TI یا RI هر کدام که یک شده بود را با دستور CLR صفر نمود.



Serial interrupt is invoked by TI or RI flags

- برنامه‌ای بنویسید که 8051 به طور مکرر ۸ بیت داده از پورت P1 بخواند و در پورت P2 بنویسد، همچنین یک کپی از داده‌ی خوانده شده از P1 را به صورت سریال با سرعت 9600 bps ارسال نماید. فرض کنید فرکانس اسیلاتور 11.0592MHz است.

Solution:

```
        ORG   0000H
        LJMP  MAIN
        ORG   23H
        LJMP  SERIAL     ;jump to serial int ISR
        ORG   30H
MAIN:   MOV   P1,#0FFH   ;make P1 an input port
        MOV   TMOD,#20H  ;timer 1, auto reload
        MOV   TH1,#0FDH  ;9600 baud rate
        MOV   SCON,#50H  ;8-bit,1 stop, ren enabled
        MOV   IE,10010000B ;enable serial int.
        SETB  TR1        ;start timer 1
        MOV   A, P1              ; read data from port1
        MOV   SBUF, A            ; send the first data to serial port
BACK:   MOV   A, P1              ; read data from port1
        MOV   P2,A               ; send it to p2
        SJMP  BACK               ; stay in loop indefinitely
```

```
...
;-------------------SERIAL PORT ISR
        ORG 100H
SERIAL: JB  TI,TRANS;jump if TI is high
        MOV A,SBUF  ;otherwise due to receive
        CLR RI      ;clear RI since CPU doesn't
        RETI        ;return from ISR
TRANS:  CLR TI      ;clear TI since CPU doesn't
```

```
        MOV SBUF, A     ; send the next byte to serial port
        RETI            ; clear TI since CPU doesn't
        END
```

- برنامه‌ای بنویسید که 8051 مکررا ۸ بیت داده را از پورت P1 بخواند و در پورت P2 بریزد. همچنین به طور هم زمان داده‌های دریافت شده از پورت سری را در پورت P0 بریزد. فرض کنید فرکانس اسیلاتور 11.0592MHz است. و پورت سری با baud rate 9600 کار می‌کند.

Solution:

```
        ORG   0000H
        LJMP  MAIN
        ORG   23H
        LJMP  SERIAL     ;jump to serial int ISR
        ORG   30H
MAIN:   MOV   P1,#0FFH   ;make P1 an input port
        MOV   TMOD,#20H  ;timer 1, auto reload
        MOV   TH1,#0FDH  ;9600 baud rate
        MOV   SCON,#50H  ;8-bit,1 stop, ren enabled
        MOV   IE,10010000B ;enable serial int.
        SETB  TR1        ;start timer 1
BACK:   MOV   A,P1       ;read data from port 1
        MOV   P2,A       ;send it to P2
        SJMP  BACK       ;stay in loop indefinitely
...
```

- برنامه‌ای بنویسید که 8051 مکررا ۸ بیت داده را از پورت P1 بخواند و در پورت P2 بریزد. همچنین به طور هم زمان داده‌های دریافت شده از پورت سری را در پورت P0 بریزد. فرض کنید فرکانس اسیلاتور 11.0592MHz است. و پورت سری با baud rate 9600 کار می‌کند.

```
...
;------------------SERIAL PORT ISR
        ORG 100H
SERIAL: JB  TI,TRANS;jump if TI is high
        MOV A,SBUF  ;otherwise due to receive
        MOV P0,A    ;send incoming data to P0
        CLR RI      ;clear RI since CPU doesn't
        RETI        ;return from ISR
TRANS:  CLR TI      ;clear TI since CPU doesn't
        RETI        ;return from ISR
        END
```

- برنامه‌ای بنویسید که با استفاده از وقفه کارهای زیر را انجام دهد :
- الف) داده را به صورت سریال دریافت کند و به پورت P0 انتقال دهد.
- ب) داده در پورت P1 را بخواند و به صورت سریالی ارسال نماید همچنین یک کپی از آن را به پورت P2 بفرستد.
- ج) با استفاده از تایمر صفر یک موج مربعی با فرکانس 5KHz بر روی P3.0 ایجاد نماید.
- فرض کنید فرکانس اسیلاتور 11.0592MHz است. و پورت سری با 4800 baud rate کار می‌کند.

```
        ORG   0
        LJMP  MAIN
        ORG   000BH   ;ISR for timer 0
        CPL   P0.1    ;toggle P0.1
        RETI          ;return from ISR
        ORG   23H     ;
        LJMP  SERIAL  ;jump to serial interrupt ISR
        ORG   30H
MAIN:   MOV   P1,#0FFH ;make P1 an input port
        MOV   TMOD,#22H;timer 1,mode 2(auto reload)
        MOV   TH1,#0F6H;4800 baud rate
        MOV   SCON,#50H;8-bit, 1 stop, ren enabled
        MOV   TH0,#-92 ;for 5kHZ wave
...
```

```
...
        MOV   IE,10010010B ;enable serial int.
        SETB  TR1          ;start timer 1
        SETB  TR0          ;start timer 0
        MOV     A, P1              ; read data from port1
        MOV     SBUF, A            ; send the first data to serial port
BACK:   MOV     A, P1              ; read data from port1
        MOV     P2,A               ; send it to p2
        SJMP    BACK               ; stay in loop indefinitely
;-----------------SERIAL PORT ISR
        ORG   100H
SERIAL:JB     TI,TRANS;jump if TI is high
        MOV P0, SBUF     ; otherwise due to receive, send serial
                         ; data to P0
        CLR   RI           ;clear RI since CPU doesn't
        RETI               ;return from ISR
TRANS:  CLR   TI           ;clear TI since CPU doesn't
        MOV SBUF, A      ; send the next byte to serial port
        RETI             ; clear TI since CPU doesn't
        END
```

اولویت وقفه‌ها

- ترتیب اولویت وقفه‌ها به صورت پیش فرض (زمانی‌که میکروکنترلر ریست می‌شود) برابر است با :
  - وقفه 0 خارجی (INT0)
  - تایمر 0
  - وقفه 1 خارجی (INT1)
  - تایمر 1
  - پورت سری

- اولویت وقفه یعنی:
  - یک روتین وقفه (ISR) بوسیله‌ی یک وقفه با اولویت بالاتر می‌تواند قطع شود و روتین وقفه‌ی جدید اجرا گردد.
  - ولی روتین وقفه با اولویت بالاتر، در زمانی‌که وقفه‌ای با اولویت پایین‌تر پیش بیاید، متوقف نمی‌شود.
    - بعد از اتمام روتین وقفه با اولویت بالاتر، روتین وقفه اولویت پایین اجرا می‌گردد.
  - برنامه‌ی اصلی میکروکنترلر که در سطح پایه در حال اجراست با وقوع هر وقفه‌ای متوقف می‌گردد.

اولویت وقفه‌ها

- تمام تقاضاهای وقفه در داخل میکروکنترلر نگهداری می‌شود و به ترتیب اولویت به آنها سرویس داده خواهد شد.

- مثال : اگر وقفه‌های خارجی صفر، تایمر صفر و وقفه خارجی یک با هم فعال شوند چه اتفاقی می افتد؟
- جواب : هر سه وقفه در داخل میکروکنترلر ذخیره می‌گردند. سپس ابتدا وقفه‌ی خارجی صفر سرویس‌دهی می‌شود. بعد از اتمام سرویس آن، تایمر صفر سرویس‌دهی می‌شود و پس از آن وقفه خارجی یک.

اولویت وقفه‌ها

- اولویت وقفه‌ها توسط بیت‌های ثبات اولویت وقفه (Interrupt Priority) (IP) قابل تنظیم است.
- ثبات IP در آدرس B8H قرار دارد.
  - به صورت بیتی نیز قابل دسترسی است.

--	IP.7	Reserved
--	IP.6	Reserved
PT2	IP.5	Timer 2 interrupt priority bit (8052 only)
PS	IP.4	Serial port interrupt priority bit
PT1	IP.3	Timer 1 interrupt priority bit
PX1	IP.2	External interrupt 1 priority bit
PT0	IP.1	Timer 0 interrupt priority bit
PX0	IP.0	External interrupt 0 priority bit

Priority bit=1 assigns high priority
Priority bit=0 assigns low priority

اولویت وقفه‌ها

- می‌توان با برنامه‌ریزی ثبات IP این ترتیب را عوض نمود.

- باید بیت متناظر با وقفه‌ی مورد نظر را در ثبات اولویت وقفه یک کرد تا وقفه در سطح بالاتر قرار گیرد.

- اگر دو یا چند وقفه دارای بیت اولویت آنها یک باشد، به آنها به ترتیب اولویت اولیه وقفه‌ها سرویس‌دهی خواهد شد.

اولویت وقفه‌ها

- مثال : 8051 را به گونه‌ای تنظیم کنید که وقفه‌ی خارجی ۱ دارای بالاترین اولویت باشد.
- جواب : MOV IP, #00000100B یا SETB IP.2
  - در این حالت اگر سه وقفه‌ی INT0 و INT1 و TF0 همزمان فعال شوند، 8051 ابتدا به INT1 سرویس می‌دهد، سپس به INT0 و بعد TF0.

- مثال : فرض کنید اولویت وقفه‌ها بدین صورت تنظیم شده باشند :
  - MOV IP, #00001100B
  - ترتیب سرویس‌دهی به وقفه‌ها را مشخص کنید.

Highest Priority	External Interrupt 1	(INT1)
	Timer Interrupt 1	(TF1)
	External Interrupt 0	(INT0)
	Timer Interrupt 0	(TF0)
Lowest Priority	Serial Communication	(RI+TI)

اولویت وقفه‌ها

- در 8051 یک وقفه با اولویت پایین می‌تواند توسط یک وقفه با اولویت بالا قطع شود. اما یک وقفه با اولویت بالا توسط یک وقفه با اولویت پایین قطع نمی‌شود.

- تمام وقفه‌ها در 8051 ذخیره می‌شوند. یک وقفه با اولویت پایین باید تا پایان اجرای سرویس وقفه با اولویت بالاتر صبر کند.

مثال پورت سری

در انتقال اطلاعات به صورت سری فرض کرده‌ایم بیت نهم دریافتی نشان‌دهنده صحت اطلاعات یک بایت باشد. اگر بیت اول بایت دریافتی برابر با صفر باشد، بیت نهم حاصل XOR بیت سوم و پنجم است و اگر بیت اول بایت دریافتی یک باشد، بیت نهم حاصل AND بیت چهارم و هفتم خواهد بود.

برنامه‌ای بنویسید که مرتبا یک بایت از پورت سری دریافت کند و با بررسی بیت نهم صحت دریافت اطلاعات را چک کند. اگر بایت دریافتی صحیح باشد آن را از طریق پورت صفر ارسال کند و اگر بایت دریافتی مخدوش شده باشد عبارت "Error!" را به صورت چشمک زن به مدت ۳ ثانیه در LCD که به پورت یک متصل است نشان دهد.

در زمان چشمک زدن، نباید بایتی از پورت سری دریافت شود.

برای نمایش چشمک زن کافی است ابتدا عبارت به پورت یک ارسال شود، پس از ۰/۵ ثانیه عدد صفر به پورت یک ارسال شود و باز بعد از ۰/۵ ثانیه عبارت ارسال گردد.

پورت سری به صورت UART ۹ بیتی با سرعت ۹۶۰۰ baud کار می‌کند.

فرکانس اسیلاتور 12MHZ است.

مثال پورت سری

راه حل :

هر بایت که از طریق پورت سری دریافت می شود را با تابع CHECK چک می کنیم که آیا بایت درست دریافت شده است یا نه.

اگر درست دریافت نشده بود، وقفه پورت سری را غیرفعال کرده و تایمر۰ را روی 50000µs تنظیم کرده و استارت می کنیم.

باید تایمر۰ ده بار سرریز دهد تا زمان 500000µs یا 0.5s فرا رسد. از R7 برای شمارش تعداد بار رخ دادن سرریز استفاده می کنیم.

وقتی R7 ده شد یعنی 0.5s گذشت حال باید یا LCD را روشن و یا خاموش کنیم. بیت 50H را به عنوان پرچم در نظر می گیرم. اگر این بیت یک بود LCD را پاک و اگر صفر بود عبارت را بر روی LCD می نویسیم.

همچنین باید عملیات نوشتن و پاک کردن LCD ۶ بار صورت گیرد تا ۳ ثانیه بشود. از R1 برای شمارش این تعداد بار استفاده می کنیم. وقتی R1 ۶ بشود، تایمر۰ را متوقف و وقفه پورت سری را فعال می کنیم تا بایت بعدی از طریق پورت سری دریافت شود.

مثال پورت سری

```
ORG 0
LJMP MAIN
ORG 0BH
LJMP T0ISR
ORG 23H
LJMP SPISR

ORG 30H
MAIN: MOV SCON, #0D0H
MOV TMOD, #21H
MOV TH1, #-3
MOV IE, #10010010B
SETB TR1

HERE: SJMP HERE
```

```
ORG 100H
SPISR:JB TI, TRANS
RECEIVE: MOV A, SBUF
MOV C, RB8
MOV 40H, C
CALL CHECK
CLR RI
RETI
TRANS: CLR TI
RETI

ORG 200H
T0ISR:
INC R7
CJNE R7, #10, EXITT0ISR
LCALL TIMEREACHED
EXITT0ISR:
MOV TH0, #3CH
MOV TL0, #0B0H
RETI
```

```
        ORG 300H
CHECK:  JB ACC.1, CHECK_AND
CHECK_XOR: CALL CALC_XOR
        SJMP RESULT
CHECK_AND: CALL CALC_AND
RESULT: JC BIT9ONE
        JNC BIT9ZERO
BIT9ONE: JB 40H, CORRECT
        JNB 40H, N_CORRECT
BIT9ZERO: JNB 40H, CORRECT
        JB 40H, N_CORRECT
N_CORRECT: CLR ESP
        MOV R1, #0
        CLR 50H
        MOV TH0, #3CH
        MOV TL0, #0B0H
        MOV R7, #0
        SETB TR0
        RET
CORRECT: MOV P0, A
        RET

CALC_AND: MOV C, ACC.4
        ANL C, ACC.7
        RET
CALC_XOR: JB ACC.3, L1
        MOV C, ACC.5
        SJMP L2
L1:     MOV C, ACC.5
        CPL C
L2:     RET
```

```
ORG 400H
TIMEREACHED: CLR TR0
INC R1
CJNE R1, #6, CONTINUE
MOV P1, #0
SETB ESP
CLR TR0
RET
CONTINUE: JB 50H, CLEARLCD
SHOWLCD:
MOV DPTR, #DATA
CLR A
AGAIN: MOVC A, @A+DPTR
JZ EXITTIME
MOV P1, A
INC A
SJMP AGAIN
CLEARLCD: MOV P1, #0
EXITTIME: CPL 50H
SETB TR0
RET
ORG 500H
DATA: DB "ERROR!",0
```

مثال Stopwatch

می‌خواهیم با استفاده از میکروکنترلر 8051 یک stopwatch طراحی کنیم. بدین صورت که کلید وصل و قطع stopwatch به پایه INT0 میکروکنترلر متصل است و در زمان فشرده شدن کلید stopwatch یک لبه پایین‌رونده در پایه INT0 بوجود می‌آید. می‌خواهیم زمان متصل بودن کلید stopwatch را اندازه‌گیری کرده و بر روی صفحه نمایش نشان دهیم.

دقت اندازه‌گیری زمان در stopwatch یک میلی ثانیه است، یعنی زمان نمایش داده شده بر روی صفحه نمایش هر یک میلی ثانیه بروزرسانی می‌شود.

همچنین این قابلیت وجود دارد که با وصل شدن مجدد کلید، زمان از ادامه شمارش شود.

فرض می‌کنیم صفحه نمایش به پورت یک متصل است. برای نمایش اعداد بر روی صفحه نمایش کافی است کد اسکی عدد را از کوچکترین رقم به بزرگترین رقم به ترتیب به پورت یک ارسال کنیم. همچنین صفحه نمایش توانایی نمایش نهایتا ۱۰ کاراکتر را دارد.

Stopwatch مثال

```
ORG 0
LJMP MAIN
ORG 03H
LJMP EX0ISR
ORG 0BH
LJMP T0ISR

ORG 30H
MAIN:
MOV R0, #30H
MOV R7, #10
L1: MOV @R0, #0
INC R0
DJNZ R7, L1
CLR 20H

SETB IT0
SETB P3.2
MOV IE, #10000011B
HERE: SJMP HERE
```

```
ORG 100H
EX0ISR:
JB 20H, STOPTIMER
STARTTIMER:
SETB 20H
MOV TH0, #0FCH
MOV TL0, #18H
SETB TR0
RETI
STOPTIMER:
CLR TR0
CLR 20H
RETI
```

```
ORG 200
T0ISR: CLR TR0
LCALL INCTIME
LCALL SHOWTIME
MOV TH0, #0FCH
MOV TL0, #18H
SETB TR0
RETI
```

```
ORG 300H
INCTIME:
MOV R0, #30H
AGAIN: INC @R0
CJNE @R0, #10, EXITINC
MOV @R0, #0
INC R0
SJMP AGAIN
EXITINC: RET

ORG 400H
SHOWTIME:
MOV R0, #30H
MOV R7, #10
L2: MOV A, @R0
ORL A, #30H
MOV P1, A
INC R0
DJNZ R7, L2
RET
```