وقفه ها

هر رویدادی که باعث شود CPU اجرای عادی یک برنامه را قطع کند وقفه نامیده می شود. یک برنامه نویس اسمبلی با صدور وقفه های نرم افزاری می تواند به طور موثری با دستگاه های جانبی ارتباط برقرار کند.

گاهی اوقات جریان عادی اجرای یک برنامه برای پردازش رویدادی که نیاز به پاسخ سریع دارد متوقف می شود. سخت افزار کامپیوتر برای مدیریت این رویدادها مکانیسمی به نام وقفه (interrupt) را دارد .

مثال. وقتی mouse حرکت می کند، سخت افزار mouse برنامه جاری را متوقف می کند تا حرکت mouse گرفته شود(برای حرکت مکان نمای mouse روی صفحه نمایش).

وقتی CPU یک سیگنال وقفه را تشخیص می دهد، فعالیت جاری خود را متوقف می کند و روتین خاصی را فراخوانی می کند که روتین وقفه (interrupt handler) نام دارد. این روتین علت وقوع وقفه را تشخیص می دهد و عکس العمل مناسب را انجام می دهد .

بیشتر روتین های وقفه بعد از پایان یافتن کنترل اجرا را به برنامه متوقف شده بازمی گردانند. آنها کلیه مقادیر ثبات ها را به وضعیت قبل از تولید وقفه بر می گردانند. بنابراین برنامه متوقف شده به گونه ای به اجرا ادامه می دهد که هیچ اتفاقی نیافتاده است به جز این که سیکل های CPU را از دست می دهند .

وقتی دو یا چند وقفه همزمان با هم اتفاق می افتند، CPU از سیستم الویت استفاده می کند و می تواند در طی اجرای بخش بحرانی یک برنامه وقفه ها را غیرفعال کند. وقتی دارد یک روتین وقفه را اجرا می کند کلیه وقفه های با الویت کمتر یا، تا زمان خاتمه اجرای روتین، غیر فعال هستند .

انواع وقفه

۲۵۶ سطح الویت توسط پردازنده های 80x86 پشتیبانی می شود که می توان آنها را به سه گروه کلی تقسیم کرد:

- وقفه های داخلی سخت افزاری
- وقفه های خارجی سخت افزاری
- وقفه های نرم افزاری

وقفه های داخلی سخت افزاری

وقفه های داخلی سخت افزاری (internal hardware-interrupts) بدلیل رخ دادن وضعیت معینی که درحین اجرای یک برنامه پیش آمده تولید می شوند(مانند تقسیم بر صفر).

وقفه هایی که در اثر خطا بوجود می آید تله (trap) هم نامیده می شود. تله باعث سقط برنامه می شوند .

این وقفه ها توسط سخت افزار اداره می شوند و امکان تغییر آنها وجود ندارد. اما با وجودیکه نمی توان آنها را مستقیما مدیریت کرد، این امکان وجود دارد که از اثر آن روی کامپیوتر به نحو مفیدی استفاده شود .

مثال. سخت افزار وقفه شمارنده ساعت کامپیوتر را چندبار در ثانیه فراخوانی می کند تا زمان را نگه دارد. می توان برنامه ای نوشت که مقدار شمارنده ساعت را خوانده آنرا به شکل قابل درک کاربر به صورت ساعت و دقیقه تبدیل کند .

وقفه های خارجی سخت افزاری

وقفه های خارجی سخت افزاری (external hardware-interrupts) خارج از CPU و توسط دستگاه های جانبی ، مانند صفحه کلید، چاپگر، کارت های ارتباطی و یا کمک پردازنده تولید می شوند .

دستگاه های جانبی با ارسال وقفه به CPU خواستار قطع اجرای برنامه فعلی شده و CPU را متوجه خود می کنند. آنها به پایه INTR (maskable interrupts)یا NMI (non maskable interrupts) پردازنده متصل هستند .

وقفه های دستگاه ها می توانند از طریق مداری به نام PIC 8259A ، که کارش منحصرا سروکار داشتن با این نوع وقفه هاست، به پردازنده ارسال شوند. مدار PIC (programmable interrupt controller) که توسط CPU کنترل می شود سیگنال هایش را روی پایه INTR قرار می دهد و امکان فعال و غیرفعال کردن وقفه ها و تغییر سطح الویت را تحت نظارت یک برنامه می دهد .

دستورات STI و CLI می توانند برای فعال و غیرفعال کردن وقفه هایی که روی پایه INTR ارسال می شوند بکار روند که البته روی وقفه های NMI تاثیری ندارد.

وقفه های نرم افزاری

وقفه های نرم افزاری (software interruptions) در نتیجه دستورالعمل int در یک برنامه درحال اجرا تولید می شوند .

برنامه نویس می تواند با دادن دستور int یک وقفه نرم افزاری تولید کند .بدین طریق بلافاصله اجرای برنامه فعلی را متوقف می کند و CPU را به روتین وقفه هدایت می کند. برنامه نویس از طریق وقفه ها می تواند در برنامه با وسایل جانبی ارتباط برقرار کند. استفاده از وقفه ها باعث کوتاهتر شدن کد برنامه و درک آسانتر و اجرای بهترآن می شود .

روتین های وقفه نرم افزاری بخشی از سیستم عامل هستند. از اینرو وقفه های نرم افزاری را می توان به دو گروه تقسیم کرد؛ وقفه های سیستم عامل DOS و وقفه های BIOS. وقفه های DOS آسانتر استفاده می شوند اما از وقفه های BIOS که قسمتی از سخت افزار هستند کندتر هستند .

DOS این نوع وقفه ها را برای اجرای API (application programming interface) خودش استفاده می کند. بیشتر سیستم عامل های جدید مانند Windows و Unix واسطه C-based را استفاده می کنند .

INT

دستورالعمل int (interrupt) یک روتین وقفه را فراخوانی می کند. فرم کلی آن به صورت زیر است :

```
int n
```

n شماره وقفه موردنظر و مقداری بین 0 تا 255 است که اجازه فراخوانی ۲۵۶ روتین مختلف وقفه را می دهد.

دستورالعمل int یک فراخوانی سیستمی را می سازد و شکل خاصی از دستورالعمل فراخوانی یک زیربرنامه (دستورالعمل call) است.

مشکل دستورالعمل int این است که تنها ۲۵۶ روتین وقفه را می تواند پشتیبانی کند. درحالیکه DOS به تنهایی دارای بیش از ۱۰۰ سرویس مختلف وقفه و BIOS بیش از هزاران سرویس وقفه است. که این تعداد بیش از کلیه وقفه هایی است که توسط اینتل رزرو شده است. برای حل این مشکل از یک شماره وقفه برای هر دسته از سرویس های وقفه و یک شماره تابع برای تعیین سرویس موردنظر استفاده می شود. شماره تابع توسط یکی از ثبات ها (اکثرا AH) هنگام فراخوانی وقفه ارسال به روتین وقفه می شود.

مثال. سیستم عامل DOS شماره وقفه 21h را بکار می گیرد. برای انتخاب یک تابع خاص، قبل از فراخوانی وقفه، کد تابع در ثبات AH قرار می گیرد. برای نمونه تابع 4Ch این وقفه برای خاتمه برنامه و برگشت به محیط DOS فراخوانی می شود.

```
mov AH, 4Ch
int 21h
```

جدول بردار وقفه

هر سطح وقفه یک محل رزرو شده در حافظه دارد که بردار وقفه (interrupt vector) نامیده می شود. همه بردارهای وقفه در جدولی به نام جدول بردار وقفه (interrupt vector table) نگهداری می شوند. این جدول از ابتدای حافظه اصلی یعنی آدرس 0000:0000 ذخیره شده است.

هر بردار وقفه ٤ بایت طول دارد. دوبایت بالای آن آفست و دو بایت پایین آن سگمنت روتین وقفه را دربر می گیرند. چون ۲۵٦ روتین های وقفه وجود دارد بنابراین اندازه جدول بردار وقفه 256×4=1024=1KB است .

شماره وقفه به عنوان اندیسی برای جدول بردار وقفه استفاده می شود. آفست روتین وقفه شماره n در آدرس n×4 و آدرس سگمنت روتین وقفه شماره n در آدرس n×4+2 جدول قرار دارد.

یک ویژگی خوب این سیستم این است که می توان بردارها را برای اشاره به روتین دیگری تغییر داد. که این همان کاری است که برنامه های TSR (Terminate and Stay Resident) انجام می دهند .

در برنامه همیشه توسط شماره وقفه به یک روتین وقفه مراجعه می شود بنابراین برنامه نیازی به دانستن آدرس واقعی در حافظه ندارد و آدرس روتین وقفه هنگام اجرا توسط CPU تعیین می شود .

مثال. آدرس های آفست و سگمنت روتین وقفه شماره 5 برابر با 5×4=0014h و 5×4+2=0016h می باشد.

چند نمونه وقفه متعارف

وقفه 21h

تابع 01h. یک کلید را از صفحه کلید می خواند.
کد کلید خوانده شده در ثبات AL برگردانده می شود.

```
mov AH, 0lh
int 21h
mov AL, character
```

تابع 02h. یک کاراکتر را روی صفحه نمایش نشان می دهد.
کد کاراکتر در ثبات DL باید قرار داده شود.

```
mov AH, 0lh
mov DL,character
int 21h
```

تابع 09h. یک رشته کاراکتری را روی صفحه نمایش نشان می دهد.
آدرس شروع رشته باید در ثبات های DS:DX قرار بگیرد. انتهای رشته توسط کاراکتر ($) باید تعیین شده باشد.

```
mov AH,09h
lea DX, string
int 21h
```

وقفه 10h

تابع 02h. مکان نما را روی صفحه نمایش به سطر و ستون خاصی منتقل می کند.
شماره صفحه در BH، شماره سطر در DH و شماره ستون در DL باید قرار بگیرد. مختصات صفحه از نقطه 0 و 0 از گوشه بالای چپ صفحه نمایش شروع می شود.

```
mov AH, 02h
mov BH, page
mov DH, row
mov DL, column
int 10h
```

تابع 09h. یک کاراکتر را با رنگ معین چندبار نمایش می دهد.
کد کاراکتر در AL، شماره صفحه در BH، خاصیت رنگ کاراکتر در BL و تعداد تکرار کاراکتر در CX باید قرار بگیرد.

```
mov AH, 09h
mov AL, character
mov BH, page
mov BL, attribute
mov CX, number
int 10h
```

برنامه اسمبلی تبدیل کاراکتر ورودی از حرف کوچک به حرف بزرگ

```
   .MODEL small
   .STACK 256
   .DATA
MSG1    DB 'Enter a lower case letter: $'
MSG2    DB 0Dh,0Ah,'In upper case it is: '
CHAR    DB ?,'$'
exCode  DB 0
   .CODE
MAIN:
;initialize ds
   mov    ax,@data    ; Initialize DS to address
   mov    ds,ax       ; of data segment
;print user prompt
   mov    ah,9        ; display string fcn
   lea    dx,MSG1     ; get first message
   int    21h         ; display it
;input a character and convert to upper case
   mov    ah,1        ; read char fcn
   int    21h         ; input char into AL
   sub    al,20h      ; convert to upper case
   mov    CHAR,al     ; and store it
;display on the next line
   mov    dx,offset MSG2 ; get second message
   mov    ah,9        ; display string function
   int    21h         ; display message and upper case
;return to DOS
Exit:
   mov    ah,4Ch      ; DOS function: Exit program
   mov    al,exCode   ; Return exit code value
   int    21h         ; Call DOS. Terminate program
   END    MAIN        ; End of program / entry point
```