فصل هفدهم

مقیم کردن برنامه در حافظه یا برنامه‌های TSR

مقدمه

معمولاً برنامه‌های کامپیوتر بر روی دیسک قرار دارند و هر برنامه‌ای که از روی دیسک می‌خواهد اجرا شود، سیستم عامل محلی از حافظه را برای آن تخصیص می‌دهد و سپس برنامه را از دیسک به حافظه بارگذاری می‌کند. بعد از اجرای برنامه، سیستم عامل محلی از حافظه که برنامه‌ی کاربر آن را اشغال کرده بود را آزاد می‌نماید که بتواند در اختیار برنامه‌ی دیگری قرار دهد. در غیر این صورت بعد از اجرای چند برنامه حافظه کاملاً پر می‌شود و کامپیوتر نمی‌تواند برنامه‌ی دیگری را اجرا نماید لذا هر برنامه‌ای که از روی دیسک می‌خواهد اجرا شود به طور موقت وارد حافظه می‌شود و پس از اجرا، حافظه مذکور در اختیار برنامه‌ی دیگری قرار می‌گیرد به عبارتی دیگر مانند این است که برنامه‌ی کاربر از حافظه خارج شده است. فلسفه برنامه‌های TSR یا مقیم کردن برنامه در حافظه، این است که پس از اجرای برنامه، برنامه‌ی مذکور در حافظه باقی بماند به عبارتی دیگر سیستم عامل برنامه‌ی مذکور را در حافظه مقیم کند که در این صورت نیازی نیست برای اجرای برنامه آن را از روی دیسک فراخوانی کرد، چون برنامه در حافظه قرار دارد، لذا بسیار سریع‌تر اجرا می‌گردد. البته اشکال آن این است که برنامه‌ی مقیم در حافظه، قسمتی از حافظه را اشغال می‌نماید و جای کمتری در حافظه برای برنامه‌های دیگر خواهد ماند. لذا هر چه تعداد بیشتری برنامه در حافظه مقیم شوند، جای کمتری برای بقیه برنامه‌های کاربر باقی خواهند ماند. به عنوان نمونه برنامه‌های ساعت گوشه مانیتور یا ماشین حساب کامپیوتر و غیره در حافظه مقیم می‌باشند و بقیه برنامه‌ها، بر روی دیسک قرار دارند. برنامه‌های مقیم یا TSR تا موقعی که کامپیوتر روشن است در حافظه قرار دارند.

روش مقیم کردن برنامه در حافظه

برای مقیم کردن برنامه‌ی کاربر در حافظه کافیست در انتهای برنامه به جای برگشت به سیستم عامل (با سرویس 4CH دستور INT 21H) از سرویس 31H دستور INT 21H استفاده نمود به شرطی که اندازه برنامه کاربر بر حسب تعداد پاراگراف (هر پاراگرف 16 بایت است) قبلاً در ثبات DX قرار داده شود. به عنوان مثال اگر برنامه‌ای دارای 12 پاراگراف باشد، دستورات زیر جهت مقیم کردن آن در حافظه در انتهای برنامه کاربر باید گذارده شود:

```
MOV AH, 31H ;
MOV DX, 12 ;
INT 21H ;
```

فراخوانی یا فعال کردن برنامه‌ی مقیم

روش کلی برای فعال کردن برنامه‌ی مقیم در حافظه این است که آدرس برنامه‌ی مقیم در حافظه را جایگزین آدرس یا بردار روتین وقفه سخت‌افزاری نمود. لذا هر موقع که وقفه مذکور فعال شود، به جای روتین وقفه سخت افزاری، برنامه‌ی مقیم در حافظه اجرا می‌گردد. برای اینکه وقفه‌ی سخت‌افزاری نیز دچار اشکال نشود، آدرس روتین وقفه‌ی سخت‌افزاری در محل دیگری در حافظه ذخیره می‌گردد که در انتهای برنامه‌ی مقیم، به آن مراجعه می‌شود. یکی از وقفه‌های سخت‌افزاری که به کار می‌رود عبارتند از دستور INT 09 که با فشار دادن کلید صفحه کلید فعال می‌شود و یا وقفه تایمر که با دستور INT 08H فعال می-گردد. حال بررسی می‌نمائیم که چطور می‌توان آدرس برنامه‌ی مقیم در حافظه را، جایگزین آدرس روتین وقفه سخت‌افزاری کرد.

جایگزینی آدرس برنامه‌ی مقیم در حافظه با آدرس یا بردار روتین وقفه

برای اینکار از سرویس 35H و 25H استفاده می‌کنیم که در ذیل هر یک از آنها شرح داده می‌شوند

الف: استخراج مقادیرCS:IP یک روتین وقفه از جدول بردار وقفه توسط سرویس 35H دستور INT 21H:

در این حالت با قرار دادن عدد 35H در ثبات AH و شماره وقفه در ثباتAL و اجرای دستور INT 21H، آدرس سگمنت کد CS روتین وقفه در ثبات ES قرار می‌گیرد و مقدار افست آدرس IP نیز در ثبات BX قرار می‌گیرد. یعنی آدرس CS:IP روتین وقفه مورد نظر، در ثبات ES:BX گذارده می شوند. به عنوان مثال اگر بخواهیم مقدار CS:IP روتین وقفه INT 08H را پیدا کنیم و محل‌های حافظه OLDVECT و OLDVECT +2 قرار دهیم دستورات زیر را می‌نویسیم.

```
MOV AH, 35H
MOV AL, 08H
INT 21H
MOV OLDVECT, BX
MOV OLDVECT +2, EX
```

ب: جایگزین کردن آدرس برنامه‌ی مقیم در جدول بردار وقفه به جای آدرس روتین وقفه

برای این منظور از سرویس 25H دستور INT 21H را استفاده می‌کنیم. در این حالت با قرار دادن عدد 25H در ثبات AH و شماره‌ی وقفه در ثبات AL و همچنین گذاردن افست آدرس روتین جدید وقفه در ثبات DX، بالاخره اجرای دستورات INT 21H آدرس روتین وقفه‌ی جدید در محل بردار وقفه مربوطه در جدول بردار وقفه قرار می‌گیرند.

مثال: می‌خواهیم آدرس برنامه یا روتین خودمان به نام NEWISR را در محل آدرس دستور وقفه INT 08H، در جدول بردار وقفه قرار دهیم.

```
MOV   AH,25H              ;   شماره سرویس 25H را در AH قرار بده
MOV   AL,08H              ;   آدرس بُردار وقفه 08H را انتخاب کن
MOV   DX,OFFSET NEWISR    ;   افست آدرس روتین جدید را در ثبات DX قرار بده
INT 21H                   ;   وقفه 21H را فعال کُن
```

```
        PAGE 110,100
TITLE   'tsr_exa.asm'  structure of TSR program
;-----------------------------------------------
;
;               Defining Segment of Program
;               ---------------------------
CODESG  SEGMENT 'CODE'
        ASSUME CS:CODESG
        ORG 100H
MAIN:  JMP LOAD1          ; Jump to instuctions
;-----------------------------------------------
OLDVECT DD ?     ;Four byte to save CS:IP of
;       origin ISR
;
```

```
;                          تعریف بقیه داده‌ها
;
;-----------------------------------------------
;                  PART1:                    قسمت اول
;This part of program reside in memory as new ISR
;
NEWISR   PROC NEAR
         PUSH AX
;
;              بقیه دستورات برنامه مقیم در حافظه
;
         POP AX
         JMP CS:OLDVECT   ;Perform original ISR
NEWISR   ENDP
;-----------------------------------------------
;                  PART2:                    قسمت دوم
;This part run once only, during initialization
LOAD1    PROC NEAR
;
;    1-Get vector of ISR & save on OLDVECT
;     -------------------------------------
         MOV AH,35H       ;Get the
         MOV AL,- -       ; vector
         INT 21H          ; of OLDISR
         MOV WORD PTR OLDVECT,BX     ;Save them
         MOV WORD PTR OLDVECT+2,ES   ; on OLDVECT
;
;    2-Set the offset of new ISR in vector table
;     -------------------------------------------
         MOV AH,25H       ;Set vector of
         MOV AL,- -       ; new ISR in vector table
         MOV DX,OFFSET NEWISR  ;DX=IP,DS=CS (Set by
         INT 21H          ; COM program)
;
;    3-Make resident of PART1 of program (NEWISR)
;     --------------------------------------------
         MOV DX,(OFFSET LOAD1-OFFSET CODESG) ;Find
;                how many byte resident
         ADD DX,15        ;Make it
         MOV CL,4         ; multiple of
         SHR DX,CL        ; 16 byte
;
         MOV AH,31H       ;Make it
         INT 21H          ; resident
LOAD1    ENDP
CODESG   ENDS
         END MAIN
```

شکل (۱۸-۱) ساختار اصولی برنامه اسمبلی مقیم در حافظه

برنامه‌ی نوشته شده شامل قسمت‌های زیر می‌باشد

1. بردار یا آدس سرویس روتین قبلی مربوط به وقفه 08H یا 09H را می‌گیرد و در محل OLDVECT قرار می‌دهد.
2. آدرس روتین وقفه جدید NEWVECT را در محل بردار روتین قبلی قرار می‌دهد.
3. روتین جدید وقفه NEWISR را در حافظه مقیم می‌کند

همان طور که قبلاً اشاره شد برای اینکار از سرویس 31H استفاده می‌شود و اندازه سرویس روتین وقفه جدید، باید برحسب پاراگراف در ثبات DX قرار گیرد. برای این کار آدرس CODSEG و LOAD1 اندازه سرویس روتین وقفه‌ی جدید NEWISR را برحسب بایت تعیین می‌نمایند که این مقدار توسط دستور:

MOV DX,(OFFSET LOAD1-OFFSET CODSEG)

در ثبات DX قرار می‌گیرد ولی باید مقدار را تقسیم بر 16 نمود که بر حسب پاراگراف گردد، لذا توسط دستورات:

MOV CL, 4

SHR DX, CL

محتوای ثبات DX را چهار بار به طرف راست شیفت می‌دهیم که تقسیم بر 16 گردد. اما ممکن است اندازه روتین وقفه جدید، دقیقاً مضارب 16 نباشد و تعداد پاراگراف در DX کمتر از مقدار واقعی گردد لذا برای اطمینان بیشتر قبل از دستورات فوق توسط دستور:

ADD DX, 15

15 بایت به DX اضافه می‌شود که پس از تقسیم DX بر 16 تعداد پاراگراف کمتر از مقدار واقعی مورد نیاز نگردد.

برنامه نمونه:

برنامه‌ی زیر ساختار کلی برنامه‌های مقیم در حافظه می‌باشد که با این روش می‌توان هر برنامه را در حافظه مقیم نمود. مثال زیر برنامه‌ای است که در حافظه مقیم می‌شود و هر 30 ثانیه از بلندگوی داخلی کامپیوتر صدای بوق به صدا درمی‌آید.

توضیح: برای این منظور از وقفه‌ی زمانی INT 08H استفاده می‌نمائیم. همان طوری که قبلا بحث شد وقفه INT 08H در هر 55 میلی ثانیه فعال می‌شود (دقیقاً 54/94 میلی‌ثانیه یا در هر ثانیه 18/2 بار). بنابراین اگر بخواهیم کامپیوتر هر 30 ثانیه بوق بزند باید 550 بار وقفه فعال شود. (میلی‌ثانیه 30000 = 54/94 × 550) برای این کار در برنامه متغیر COUNT را با مقدار اولیه 550 بصورت زیر معرفی می‌کنیم.

```
COUNT DW 550
```

```
        PAGE 110,100
TITLE   'timer_8.asm'  a TSR program with timer 8
;-----------------------------------------------
;
;               Defining Segment of Program
;               ---------------------------
CODESG  SEGMENT 'CODE'
        ASSUME CS:CODESG
        ORG 100H
MAIN:  JMP LOAD1        ;1-Jump to instuctions
;-----------------------------------------------
OLDINT8 DD ?    ;Four byte to save CS:IP of INT 8H
```

```
COUNT   DW 550  ;550 *55 ms=30 seconds
;-----------------------------------------------
;                PART1:
;This part of program reside in memory as new ISR
;
NEWISR  PROC NEAR
        DEC CS:COUNT     ;2-Is time
        JNZ EXIT         ;3- over?
        MOV CS:COUNT,550 ;4-If yes intialize COUNT
        MOV CX,00FFH     ;5- make delay
AGAIN:  MOV AH,0EH       ;6- and beep
        MOV AL,07        ;7- the speaker
        INT 10H          ;8- for
        LOOP AGAIN       ;9- a delay
EXIT:   JMP CS:OLDINT8   ;10- take care INT 08
NEWISR  ENDP
;-------------------------------------------
;                PART2:
;This part run once only, during initialization
LOAD1   PROC NEAR
;
;   1-Get vector of ISR & save on OLDVECT
;    ------------------------------------
        MOV AH,35H       ;11-Get the
        MOV AL,08H       ;12- vector
        INT 21H          ;13- of OLDISR INT 8H
        MOV WORD PTR OLDINT8,BX     ;14-Save them
        MOV WORD PTR OLDINT8+2,ES   ;15- on OLDINT8
;
;   2-Set the offset of new ISR in vector table
;    ---------------------------------------------
        MOV AH,25H       ;16-Set vector of
        MOV AL,08H       ;17- new ISR in vector table
        MOV DX,OFFSET NEWISR  ;18-DX=IP,DS=CS (Set by
        INT 21H          ;19- COM program)
;
;   3-Make resident of PART1 of program (NEWISR)
;    ---------------------------------------------
        MOV DX,(OFFSET LOAD1-OFFSET CODESG) ;20-
;       Find how many byte resident
        ADD DX,15        ;21-Make it
        MOV CL,4         ;22- multiple of
        SHR DX,CL        ;23-  16 byte
;
        MOV AH,31H       ;24-Make it
        INT 21H          ;25- resident
LOAD1   ENDP             ;End of procedure LOAD1
CODESG  ENDS             ;End of CODESG
        END MAIN         ;End of program
```

هر بار که وقفه 08 فعال شود، یک واحد از مقدار متغیر COUNT کم می‌گردد، تا COUNT صفر شود که در این دستور مدت 30 ثانیه گذشته است. برای فعال کردن برنامه‌ی مقیم در حافظه از امکانات وقفه 08H استفاده شده است یعنی آدرس برنامه‌ی مقیم، جانشین آدرس وقفه می‌شود. لذا بعد از دستور 1، متغیر ? OLDINT8 DD برای ذخیره آدرس وقفه 08H تعریف می‌شود.

قسمت اول برنامه

روتین جدید وقفه NEWISR: از دستورات 2 تا 10 می‌باشد. دستورات 2 و 3 یک واحد از مقدارCOUNT کم می‌کند. اگر COUNT صفر نشده بود به دستور 10 می‌رود و کار وقفه 08H ادامه می‌یابد. و با فعال شدن وقفه 08H، دوباره دستورات 2 و 3 و 10 اجرا می‌شوند و این حلقه آنقدر اجرا می‌گردد تا COUNT صفر شود، یعنی وقتی که 550 بار وقفه 08H اجرا شد که معادل 30 ثانیه است، دستور4 اجرا می‌گردد. دستور4 مقدار اولیه به متغییر COUNT می‌دهد و دستورات 5 تا 9 دستورات تولید بوق هستند. در دستور10 دوباره کنترل به وقفه 08H برمی‌گردد و کار عادی وقفه مذکور ادامه می‌یابد.

قسمت دوم برنامه:

1. همانطوری که در ساختار اصولی برنامه‌های مقیم در حافظه بحث شد در این قسمت عملیات زیر انجام می‌شود
2. توسط دستورات 11 تا 15 آدرس روتین وقفه 08H با استفاده از سرویس 35H دستور INT 21H گرفته می‌شود و در محل OLDINT8 ذخیره می‌گردد.
3. توسط دستورات 16 تا 19 افست آدرس روتین جدید وقفه NEWISR در محل آدرس روتین وقفه 08 قرار می‌گیرد.
4. توسط دستورات 20 تا 25 روتین وقفه NEWISR در حافظه مقیم می‌گردد.

با ایجاد فایل com برنامه نوشته شده و اجرای آن، برنامه‌ی مذکور در حافظه مقیم می‌شود و هر30 ثانیه یک بار بوق شنیده می‌شود. البته به جای دستورات زدن بوق کامپیوتر (دستورات 5 تا 9) می‌توان هر تعداد و هر نوع دستوراتی را برای برنامه‌ی مقیم در حافظه نوشت، که هر 30 ثانیه برنامه مذکور اجرا گردد.