UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI
Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Programare în limbaj de asamblare

38. Programe rezidente (TSR).

Programe rezidente în memorie (TSR)

TSR este prescurtarea de la „Terminate and Stay Resident“. Deci, este vorba despre programe care rămân în memoria RAM după ce s-a terminat execuția lor. Aceste programe pot fi invocate fie utilizând anumite taste, fie de către alte programe, care lansează o întrerupere software adecvată. De obicei aceste programe sunt de tip .COM.

Să considerăm ce s-ar întâmpla dacă, în timp ce un program rulează, el ar fi copiat adresa sa „top of memory“ în pointerul special, ce memorează locația de memorie unde DOS încarcă programele pentru execuție. În acest caz, DOS va fi păcălit prin încărcarea fiecărui program ulterior în zona de memorie situată imediat deasupra programului original. În consecință, programul original ar rămâne în memorie pe durata executării tuturor programelor ulterioare. Exact la fel se întâmplă într-un program TSR. Acesta înlocuiește pointerul de început al programului cu pointerul „top of memory“. Fiecare program executat după acest program TSR este încărcat în zona de memorie superioară acestuia şi deci nu se suprapune peste memoria utilizată de programul TSR. Când se încarcă mai multe programe rezidente, fiecare dintre ele utilizează această tehnică.

Un program TSR conține două părți: o parte de inițializare (tranzitorie) şi o parte rezidentă. Partea tranzitorie este responsabilă de încărcarea părții rezidente în memoria RAM şi de instalare a unei rutine de întrerupere care să determine cum este invocat (apelat) programul TSR. Dacă programul TSR este invocat de o întrerupere software, porțiunea plasează adresa părții rezidente a codului în vectorul de întrerupere corespunzător. Dacă TSR-ul va fi invocat printr-o tastă (hotkey) porțiunea de inițializare trebuie să modifice manipularea întreruperilor DOS, pentru tastatură (de fapt pentru tasta respectivă).

Când porțiunea tranzitorie este executată, la sfârşitul acesteia ea invocă o funcție DOS care permite ca o porțiune din fişierul executabil să rămână rezidentă în memoria RAM după încheierea execuției acestuia, de unde şi expresia „Termină şi Stai Rezident“. Porțiunea de inițializare a TSR-ului cunoaşte dimensiunea porțiunii rezidente, precum şi locația în memorie a porțiunii rezidente, şi transmite această informație sistemului DOS. Ulterior, DOS nu va afecta blocul de memorie specificat, dar este liber să încarce programe în zona de memorie neprotejată.

Deoarece sistemul de operare DOS este un sistem monoacces, monoprogramare şi nereentrant, apar o serie de restricții si limitări privind operațiile ce pot fi efectuate de codul rezident al programelor din această categorie. De aceea este necesar să cunoaştem structura sistemului de operare, anumite module care afectează un program rezident, precum şi modul de lucru al unor dispozitive (ca de exemplu tastatura).

Programele TSR pot fi de două tipuri, după modul de activare: *active* şi *pasive*.

Un program TSR pasiv se execută ca parte normală a fluxului unui program; el este executat în urma unui apel explicit dintr-o altă aplicație. TSR-urile pasive sunt aproape întotdeauna rutine de servire a întreruperii sau extind apeluri DOS sau BIOS.

Un program TSR activ este executat constant, la anumite intervale de timp, sau la apăsarea unei anumite combinații de taste care nu are semnificație pentru aplicația curentă. Un program TSR activ, de obicei, va suspenda execuția programului curent. TSR-urile active îndeplinesc una din două funcții: fie îndeplinesc direct o întrerupere hardware, fie sunt activate periodic, fără un apel explicit de la o aplicație.

Un TSR poate conține atât componente active, cât şi componente pasive. Astfel, anumite rutine pot fi activate de o întrerupere hardware, iar altele să fie apelate explicit de alte aplicații.

Următorul program este un exemplu de TSR care furnizează atât o rutină activă (*MyInt9h*), cât şi una pasivă (*MyInt16h*). Programul utilizează vectorii de întrerupere ai tastaturii:

09h şi *16h*. De câte ori apare o întrerupere pentru tastatură, rutina *MyInt9h*, activă pe acest nivel (*INT 09h*), va incrementa un contor. Deoarece tastatura generează, de obicei, două întreruperi la o tastare (una la apăsarea şi alta la revenirea tastei), valoarea va fi împărțită la 2, pentru a aproxima numărul de apăsări ale tastelor. Rutina pasivă pe nivelul 16h, *MyInt16h*, returnează numărul de apăsări ale tastelor către programul apelant. Codul următor furnizează cele două programe, TSR-ul (*tsr_tast.asm*) şi aplicația (*tsr_tapl.asm*) care afişează numărul de taste apăsate de când s-a lansat TSR-ul.

```
.model tiny
; tsr_tast.asm - TSR activ, numără într. tastatură, după activare
; pt. a det. nr. de taste apăsate între două apeluri succesive
; prog. permite şi dezinstalarea lui, la cererea utilizatorului
; Se va genera program de tipul .COM: tlink/t tsr_tast
.code
        org 100h
start:  jmp test_Instalare
ID_tsr db 'TSR_numara_taste_apasate_INT_9h_&_16h'
lung_ID equ $-ID_tsr
KeyCont dw 0 ; contor întreruperi de la tastatură
OldInt9h dd ?          ; salvarea vechilor vectori de întrerupere
OldInt16h dd ?
MyInt9h proc far
; comp. activa, rutina de într. tastatură, apelată de sistem la
; fiecare apăsare de tastă, incrementează contorul KeyCont, şi
; transferă controlul la întreruperea originală, Int 9.
        inc cs:KeyCont      ; incr. contor de întreruperi tastatură
        jmp cs:OldInt9h     ; apelul vechii într., salvată la OldInt9h
MyInt9h endp
MyInt16h proc far
; componenta pasivă a TSR-ului apelată prin apelul INT 16h.
; Dacă registrul AH=0FFh, întoarce nr-ul de întreruperi
; de la tastatură, în AX. Dacă AH conține altă valoare, atunci
; rutina transmite controlul la întreruperea originală INT 16h.
        cmp ah, 0FFh; se solicită valoarea contorului ?
        je ReturnCont       ; dacă da, se returnează valoarea lui
        jmp cs:OldInt16h    ; altfel apelează întreruperea originală
ReturnCont:
        mov ax, cs:KeyCont
        mov cs:KeyCont, 0 ; reinițializare contor la fiecare apel
        iret
MyInt16h endp
test_Instalare:
        push es             ; salvare ES, DS
        push ds
        push cs
        pop ds          ; init. DS cu CS, şi poate lipsi la .COM
```

```
        mov ax, 0
        mov es, ax              ; init. ES cu adr. Seg. a TVI (0000H)
; se verifică dacă nu este deja instalat acest TSR:
; se verifică dacă au fost instalate cele două întreruperi în TVI
        mov ax, es:[9h*4]       ; se comp. offseturile pt. vectorii din tabelă
        cmp ax, offset MyInt9h  ; dacă nu sunt găsiți cei doi vectori
        jne inst                ; se vor instala
        mov ax, es:[16h*4]
        cmp ax, offset MyInt16h
        jne inst
; aici se ştie că sunt instalați 2 noi vectori de întrerupere
; pe nivelurile 9h şi 16h, cu offseturile noastre, dar pt a fi siguri
; că sunt chiar aceştia vom comp. şi id. TSR-ului curent
; cu cel din memorie (din TVI)
        mov ax, es:[9h*4+2]
        mov es, ax
        mov si, offset cs: ID_tsr   ; se compară identificatorii
        mov di, si              ; offset celor două 'nume_TSR' în SI/DI
        mov cx, lung_ID         ; lungimea numelui vectorului nou
        cld                     ; direcția de parcurgere, crescătoare
        repe cmpsb      ; se compară cele doua nume
        jnz inst                ; nume diferite -> se instalează
        lea dx, mes_inst        ; afişare mesaj program deja instalat
        mov ah, 9               ; şi solicitare utilizator pentru dezinstalare
        int 21h
        mov ah, 1
        int 21h         ; cit. răsp. utilizator, dacă doreşte dezinst
        push ax                 ; salvare răspuns
        mov ah, 9               ; avans pe linie nouă pe ecran
        lea dx, CRLF
        int 21h
        pop ax          ; refacere răspuns utilizator
        and al, 0DFH ; se transformă caracterul în literă mare
        cmp al, 'D'             ; se compară cu caracterul 'D'
        je dezinst              ; daca este 'D' se va dezinstala programul
        pop es          ; refacere ES, DS
        pop ds
        mov ax, 4c00h           ; dacă nu, revine în DOS, fără instalarea
        int 21h         ; codului, care este deja instalat (rezident)
dezinst:
        xor ax, ax              ; se refac vectorii inițiali salvați în codul
        mov ds, ax              ; rezident; se pun în TVI (adr. seg. 0)
        mov es, ds:[9*4+2] ; ES = adr. seg. prg. TSR, încărcat ant.,
        mov ax, word ptr es:[OldInt9h]      ; diferă de cel actual (TVI)
        mov ds:[9*4], ax        ; reface vectorul 9: offsetul
        mov ax, word ptr es:[OldInt9h+2]
```

```
        mov ds:[9*4+2], ax    ; şi adresa de segment
        mov ax, word ptr es:[OldInt16h] ; reface vectorul 16h:
        mov ds:[16h*4], ax              ; offsetul
        mov ax, word ptr es:[OldInt16h+2]
        mov ds:[16h*4+2], ax          ; şi adresa de segment
        pop ds          ; refacere ES, DS
        pop es
        push cs                 ; dezinstalarea se face complet, cu
        pop es          ; eliberarea spațiului de memorie
        mov ah, 49h   ; alocat programului TSR
        int 21h
        push cs
        pop ds
        lea dx, mes_dezi        ; afişare mesaj 'Program Dezinstalat'
        mov ah, 9
        int 21h
        mov ax, 4C00h
        int 21h
; se memorează vechile valori pentru într. INT 9 şi INT 16
; direct în variabilele OldInt9h, şi respectiv OldInt16h
inst:
        cli                     ; dezactivare întreruperi
                                ; secțiune critică: modificare vect. într.
        mov ax, es:[9*4]        ; citire offset rutină pentru Int 9h şi
        mov word ptr OldInt9h, ax ; salvare la OldInt9h
        mov ax, es:[9*4 + 2]  ; cit. Seg. rutină pentru Int9h şi
        mov word ptr OldInt9h[2], ax        ; salvare la OldInt9h+2
        mov word ptr es:[9*4], offset MyInt9h      ; înlocuire vector
        mov word ptr es:[9*4+2], cs ; vechi cu cel nou
        mov ax, es:[16h*4]    ; aceleaşi operații pentru Int 16h
        mov word ptr OldInt16h, ax
        mov ax, es:[16h*4 + 2]
        mov word ptr OldInt16h[2], ax
        mov word ptr es:[16h*4], offset MyInt16h
        mov word ptr es:[16h*4+2], cs
        sti                     ; reactivare întreruperi
; se termină partea de instalare cu lăsarea rezidentă a codului
        mov ah, 9               ; tipărire mesaj de instalare cod TSR
        lea dx, mesaj
        int 21h
        pop ds          ; refacere DS, ES
        pop es
        mov dx, offset test_instalare + 15 ; calcul spațiul de memorie
        push cx                 ; salvare CX
        mov cl, 4               ; ocupat de codul rezident, ca nr paragrafe
        shr dx, cl
```

```
        pop cx          ; refacere CX
        mov ax, 3100h       ; apelează funcţia ce lasă rezident codul
        int 21h         ; din seg. curent, până la 'test_instalare'
mesaj  db      'S-a instalat programul ce numara ''Numarul de                    taste''
apasate.'
CRLF db      0Dh, 0Ah, '$'
mes_inst db   'Programul este deja instalat in memorie!!!',
                    0Dh, 0Ah
        db      'Doriti dezinstalarea lui (D/d-DA, orice alta tasta pentru NU): $'
mes_dezi db   'Programul a fost dezinstalat !!!', 0Dh, 0Ah, '$'
end start
```

Pe lângă problemele legate de apeluri reentrante ale TSR-ului către DOS/ BIOS, se poate ca apelurile DOS ale TSR-ului să afecteze structuri de date utilizate de alte aplicaţii curente: stiva aplicaţiei, zona DTA, etc. Când se execută un TSR el operează în contextul (mediul de lucru) al aplicaţiei principale. De ex., adresa returnată de TSR şi orice valori, pe care TSR-ul le salvează pe stivă, sunt puse pe stiva aplicaţiei. Dacă TSR foloseşte intens stiva, datorită unor apeluri recursive, sau alocării de spaţiu pe stivă pentru variabile locale, atunci TSR-ul trebuie să salveze stiva aplicaţiei (adică SS şi SP) şi să comute pe o stivă locală (proprie).

Dacă TSR-ul execută apelul funcţiei DOS ce returnează adresa PSP, se va returna adresa PSP-ului aplicaţiei principale, şi nu cel al TSR-ului. Dacă nu se comută de la PSP-ul aplicaţiei principale la cel al TSR-ului, şi apare o excepţie (manipulare disc, sau *ctrl-break*), atunci se va executa manipulatorul asociat cu aplicaţia principală (în locul celui asociat TSR-ului). De aceea, când se execută un apel DOS, care poate rezulta într-una din acele condiţii, trebuie comutat PSP-ul. Astfel când TSR-ul returnează controlul la aplicaţia principală, trebuie refăcut PSP-ul. Funcţia *50h* iniţializează adresa PSP a programului curent la valoarea din registrul BX, iar funcţia *51h* determină adresa PSP a programului curent şi o returnează în registrul BX.