



UNIUNEA EUROPEANĂ



GUVERNUL ROMÂNIEI



Instrumente Structurale
2007-2013



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

Programare în limbaj de asamblare

**39. Componentele unui program TSR
(partea tranzitorie, de instalare și cea rezidentă).**

Componentele unui program TSR

Programele TSR utilizează, după cum am arătat, în general, întreruperile de la tastatură sau de la ceasul de timp real; ele nu trebuie să interfereze cu întreruperile hardware care aparțin de domeniul driverelor de dispozitiv, excepție făcând întreruperile de la tastatură și de la circuitul de ceas.

Circuitul de ceas generează întreruperi hardware la fiecare 1/18,2 secunde (acest interval este dat de faptul că, pe durata unei ore, un registru contor ajunge la valoarea maximă 65535); aceste întreruperi periodice permit actualizarea orei sistemului. Pentru ceasul de timp real există doi vectori de întrerupere, și anume *08h*, respectiv *1Ch*. Întreruperea de pe nivelul *08h*, *INT 08h*, actualizează ora și data sistemului, după care execută instrucțiunea *INT 1Ch*; această rutină constă dintr-o singură instrucțiune *IRET*, tocmai pentru a permite utilizatorului să-și insereze aici rutina de tratare a întreruperii, și nu pe nivelul *08h*, tocmai pentru a nu afecta actualizările de sistem.

Întreruperile de la tastatură, generate atât la apăsarea unei taste, cât și la eliberarea ei, au fost descrise anterior. Întreruperea de la tastatură se află pe nivelul *09h*, *INT 09h*; această rutină de tratare a întreruperii citește codul generat de tastatură și în funcție de starea tastelor de tip „shift“ (CTRL, SHIFT, ALT)- memorată într-un octet de stare la adresa 40h:17h- generează codul tastei apăsate respective. De asemenea, această procedură verifică și anumite combinații de taste cu semnificație deosebită, cum ar fi: CTRL-ALT-DEL (reinițializare), CTRL-BREAK (suspendare), pentru care se execută instrucțiunile *INT 19h*, respectiv *INT 1Bh*. Caracterele de la tastatură sunt depuse într-un buffer circular, de 16 cuvinte; bufferul tastaturii se află la adresa 40h:1Ah și conține un pointer (deplasament) pentru extragere, un pointer (tot de tip cuvânt) pentru introducere și zona de 16 cuvinte unde se depun codurile tastelor.

Aplicațiile utilizator vor folosi instrucțiunea *INT 16h* pentru accesul la bufferul tastaturii, care permite citirea caracterelor din buffer și verificarea stării tastaturii.

Componenta mai simplă a unui program rezident este tocmai cea de inițializare, care îl lasă rezident. În locul terminării normale, ca până acum, a programului cu funcția *4Ch*, *INT 21h*, programul se va termina cu funcția *31h*, cea care păstrează programul în memorie. Când se va executa rutina de inițializare, sistemul va rezerva blocul de memorie unde este rezident programul și va încărca programele următoare în memoria rămasă disponibilă după programul rezident.

Componenta mai delicată din elaborarea unui program rezident este cea care realizează activarea programului după ce acesta a devenit rezident, deoarece nu este un program intern, de sistem, cum sunt de exemplu CLS, COPY, DIR etc. Metoda cea mai utilizată pentru activare este cea de modificare a tabelii vectorilor de întrerupere, astfel încât programul rezident întrerupe toate tastările, acționând la o anumită tastă sau combinație de taste. Partea rezidentă conține:

- una sau mai multe proceduri, dintre care una este noua rutină de tratare a întreruperii (RTI), care apelează și vechea întrerupere; la apariția unei întreruperi corespunzătoare acestei întreruperi, se va activa RTI din partea rezidentă.

- o zonă de date, care conține și spațiul alocat în care se salvează vectorii de întrerupere ce sunt modificați de programul rezident.

Un program rezident obișnuit constă din următoarele componente:

1. O componentă care redefinește locațiile din tabela vectorilor de întrerupere, adică instalarea unui nou vector de întrerupere.
2. O componentă de inițializare, care se execută doar la prima rulare a programului și care realizează:
 - înlocuirea adresei din vectorul de întrerupere cu propria sa adresă (a codului rezident);

- determină dimensiunea părții rezidente a programului (la nivel de paragrafe, 16 octeți);
 - utilizează o întrerupere (31h) care specifică sistemului să termine execuția programului curent și să lase rezidentă în memorie porțiunea specificată de program.
3. Componenta (procedura) care rămâne rezidentă și care va fi activată fie printr-o combinație specifică de taste, fie de ceasul de timp real al sistemului.

Ca efect, rutina de inițializare stabilește toate condițiile pentru a lăsa rezident programul și apoi permite ca ea însăși să fie ștearsă. Organizarea memoriei, acum, apare astfel:

Memoria disponibilă
Porțiunea de inițializare a programului TSR (peste care se poate suprapune următorul program)
Programul rezident (TSR)
COMMAND.COM
IO.SYS și MSDOS.SYS
Tabela vectorilor de întrerupere

Un program rezident poate utiliza două funcții DOS, *INT 21h*, *35h* și respectiv *25h*, pentru accesul la tabela vectorilor de întreruperi, deoarece nu există nici o garanție că adresa acestei tabele va începe întotdeauna de la adresa *0000H*. Aceste funcții primesc ca parametru nivelul întreruperii pe care se dorește instalarea activării codului rezident sau salvarea vechiului vector, deci nu trebuie cunoscută adresa fizică a acestuia.

Instalarea unui nou vector de întrerupere

Partea de instalare a unui program TSR trebuie să realizeze următoarele operații:

- să verifice, în primul rând, dacă nu cumva a fost deja instalat în memorie codul său rezident; dacă programul a fost deja instalat, nu îl va mai instala și programul se va termina cu funcția *4C00h*, *INT 21h*.
- să captureze un vector de întrerupere pe care îl înlocuiește cu unul nou. Înainte de a modifica vectorul de întrerupere, programul rezident (cel ce se instalează), salvează în partea sa de inițializare vectorul curent și apoi îl înscrie pe cel nou, care reprezintă adresa, de tip FAR a procedurii, pentru tratarea întreruperii definite în partea rezidentă;
- să calculeze dimensiunea memoriei necesare pentru codul rezident și să apeleze funcția DOS *31h*, care lasă codul părții rezidente în memorie și termină programul.

Prin capturarea unui vector de întrerupere se înțelege înlocuirea vectorului existent, cu unul nou, care conține adresa noii rutine de tratare a întreruperii, definită în programul TSR. De obicei, noua rutină apelează vechea rutină de tratare a întreruperii.

Înlocuirea unui vector de întrerupere, care constă în salvarea celui vechi și înscrierea unuia nou, se poate implementa direct de utilizator, utilizând instrucțiunile de transfer (mov), dar este mult mai comodă utilizarea funcțiilor DOS (*INT 21h*) *35h* și *25h*.

Funcția DOS *35h* returnează în ES:BX vechiul vector de întrerupere, cel care se dorește a fi înlocuit, al cărui număr a fost transmis în registrul AL, astfel:

```

mov ah, 35h      ; funcția DOS de salvare a vechiului vector
mov al, nr_int   ; nr. Într. pe nivelul căreia se pune TSR-ul
int 21h
mov Vechea_Int, bx      ; salv. vechiului vector de întrerupere
mov Vechea_Int[2], es ; din ES:BX, în zona TSR-rezident

```

Funcția 25h scrie noul vector de întrerupere, transmis în registrele DS:DX, în tabela vectorilor de întrerupere, în locul vectorului al cărui număr (index din tabelă) este specificat în registrul AL:

```

mov ah, 25h      ; funcția DOS de instalare a noului vector
mov al, nr_int   ; nr. Într. pe nivelul căreia se pune TSR-ul
lea dx, Noua_Int; offsetul RTI, adr. seg. este în DS, întrucât
                  ; toate reg. seg., la programele .COM
                  ; sunt inițializate cu segmentul PSP
int 21h          ; deci nu mai trebuie reinițializat cu
                  ; 'seg Noua_Int'

```

Ca efect, la apariția unei întreruperi pe nivelul specificat în AL, se va executa noua procedură rezidentă (TSR), în locul celei vechi.

Salvarea vechiului vector și instalarea celui nou se poate face și fără funcțiile DOS, direct în tabela vectorilor de întrerupere (TVI); secvența aceasta trebuie însă executată cu întreruperile dezactivate.

Pentru a evita reinstalarea acelui program TSR, se verifică dacă există deja o copie în memorie a codului rezident, deci dacă acesta a fost deja instalat, prin compararea offset-ului noii rutine cu offset-ul vectorului de întrerupere utilizat. În cazul în care cele două offset-uri sunt diferite, înseamnă că programul nu este rezident și, deci, poate fi instalat. Dacă, însă, cele două offseturi sunt identice, întrucât se poate întâmpla ca programe diferite să utilizeze același vector de întrerupere și să aibă același offset, se vor compara identificadorii celor două programe TSR. Două TSR-uri diferite pot avea același offset al adresei de lansare dacă zona de date ce precede începutul codului TSR are aceeași dimensiune. Din acest motiv trebuie înscris în codul rezident un șir de caractere care să identifice în mod unic acel program; acest identificador de program TSR se plasează înaintea programului TSR propriu-zis.

Pentru a lăsa rezidentă în memorie numai partea rezidentă a programului TSR se va utiliza funcția 31h, care are ca parametru de intrare dimensiunea de memorie ocupată de partea rezidentă, exprimată în paragrafe (1 paragraf = 16 octeți), transmisă în registrul DX (adresa de segment este cea conținută în DS, care are același conținut cu toate celelalte registre segment în cazul programelor .COM). Funcția va lăsa rezident în memorie codul de lungime specificată și realizează ieșirea din program, în DOS. Memoria ocupată de partea de inițializare a programului va fi eliberată de DOS.

Deci un program TSR are următoarea structură:

cod_TSR segment

```

; partea rezidentă
assume cs: cod_TSR
org 100H

```

intrare:

```

    jmp start_init      ; exe. la prima intrare
nume_TSR    db    'prog_TSR_1'    ; ID cod rezident
lung_ume    equ    $ - nume_TSR    ; lungime ID

```

```

vect_vechi    dd    ?    ; spațiu pt salvarea vechiului vector
nr_vect      equ    ...    ; vectorul pe care se instalează TSR
              ; definirea noii proceduri de tratare a întreruperii
vect_nou      proc   far    ; noua proc. poate apela și vechea p.
              push  resurse    ; salv. resurse utilizate de TSR
              .....    ; noile prelucrări, descrise de noua
              .....    ; procedură de tratare a întreruperii
              pop   resurse    ; refacerea resurselor utiliz. de TSR
              jmp   cs: vect_vechi    ; vechea într., fie cu salt fie ca apel:
;              pushf    ; apel de tip întrerupere pentru vechea
;              call  cs:vect_vechi ; procedură de tratare întrerupere
;              iret    ; revenire în programul întrerupt
vect_nou      endp
              ; partea de inițializare a codului rezident
start_init:
              mov  al, nr_vect    ; numărul vechiului vector de întrerupere,
              mov  ah, 35h    ; cel înlocuit. Salvare vector vechi.
              int  21h    ; (ES:BX) = vect vechi. Comp. offseturile
              cmp  bx, offset vect_nou    ; celor doi vectori: vechi ÷ nou
              jnz  instalare    ; Este deja rezident ? Dacă nu, se instalează
              mov  si, offset cs: nume_TSR    ; Se compară identificatorii
              mov  di, si    ; offseturile celor două nume_TSR în SI/DI
              mov  ax, cs    ; adresa de segment pentru numele noului
              mov  ds, ax    ; vector
              mov  cx, lung_nume    ; lungimea numelui vectorului nou
              cld    ; direcția de parcurgere, crescătoare
              repe cmpsb    ; se compară cele două nume
              jnz  instalare    ; nume diferite -> se instalează
              lea  dx, mes_deja_inst    ; afișare: 'Program deja instalat'
              mov  ah, 9
              int  21h
; aici se poate insera secvența de dezinstalare a TSR-ului
; se întreabă utilizatorul dacă dorește dezinstalare:
              lea  dx, mes_dezinst
              mov  ah, 9
              int  21h
              mov  ah, 1    ; citire răspuns utilizator
              int  21h
              cmp  al, 'D'    ; dacă nu este DA se termină fără dezinst.
              jnz  iesire
              xor  ax, ax    ; se reface vectorul din TVI, 'vect_vechi'
              mov  ds, ax    ; în codul rezident; se pun în TVI (seg. 0)
              cli
              mov  es, ds:[nr_int*4+2]    ; ES=adr. seg. TSR, deja încărcat,
              mov  ax, word ptr es:vect_vechi    ; care diferă de cea actuală,
              mov  ds:[nr_int*4], ax    ; (se reface în TVI) reface vectorul

```

```

mov ax, word ptr es:vect_vechi[2] ; 'nr_int': offsetul
mov ds:[nr_int*4+2], ax ; și adresa de segment
sti
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 ; afișare mesaj 'Program Dezinstalat'
mov ah, 9
int 21h

```

iesire:

```

mov ax, 4c00h ; se revine în DOS, fără instalarea
int 21h ; codului, care este deja rezident

```

instalare:

```

cli ; dezactivare sistem întreruperi (INTR)
mov word ptr vect_vechi, bx ; se salvează vechiul
mov word ptr vect_vechi[2], es ; vector
mov ah, 25h ; funcția DOS de instalare a noului
mov al, nr_vect ; vector
lea dx, vect_nou ; DS a fost inițializat anterior
int 21h

```

; secvența care lasă codul vect_nou rezident

; 'start_init' - eticheta de început a părții de inițializare,

; reprezintă și eticheta de sfârșit a codului rezident

```

lea dx, start_init ; adr. de sfârșit a părții rezidente, care este
add dx, 15 ; rotunjită la paragraful următor
mov cl, 4 ; pentru a determina
shr dx, cl ; dimensiunea în paragrafe
mov ah, 31h ; funcția DOS, care îl lasă rezident
sti ; reactivare sistem de întreruperi
int 21h

```

```
mes_deja_inst db 0Ah, 0Dh, 'TSR-ul este deja instalat !!!'
```

```
db 0Ah, 0Dh, '$'
```

```
mes_dezinst db 'Doriti dezinstalare (D pentru DA, orice '
```

```
db 'alta tasta pentru NU): $'
```

```
mes_dezi db 0Dh, 0Ah, 'Programul a fost dezinstalat !!!'
```

```
db 0Dh, 0Ah, '$'
```

```
cod_TSR ends
```

```
end intrare
```

- '*nr_vect*' este numărul vectorului de întrerupere ce se înlocuiește, de exemplu *1Ch* - vectorul ceasului de timp real, sau *09h* pentru tastatură.
- Partea de inițializare nu va rămâne în memorie, ci numai codul rezident (*vect_nou*), primele (DX) paragrafe, până la eticheta '*start_init*'.

- Spațiul de memorie ocupat de partea de inițializare a programului TSR, va fi eliberat de DOS.

Partea rezidentă a programului TSR

Partea rezidentă definește o zonă de date și prelucrările respective, reprezentate de una sau mai multe proceduri.

Procedurile dintr-un program rezident devin active ca răspuns la un apel direct al unui alt program sau la apariția unei întreruperi hardware. După lansare un lucru nu poate fi așteptat pentru codul apelant: să inițializeze corespunzător registrele segment. Singurul registru segment cu semnificație pentru codul rezident este registrul *CS*. Pentru acces la variabilele locale -> rutinele vor inițializa *DS* sau alte registre segment. De ex. o rutină care numără de câte ori alte programe o apelează, după ce a devenit rezidentă. Ea ar conține o singură instrucțiune *inc contor*, dar s-ar incrementa variabila de la offsetul *contor* în segmentul referit de *DS*, și deci nu va incrementa variabila *contor*.

Două soluții la această problemă: 1. toate variabilele în segmentul de cod, și să se utilizeze prefixul segment (*inc cs:contor*); utilizată doar când avem puține variabile. 2. salvare registrul segment de date, *DS*, la intrarea în TSR, inițializarea lui cu noua valoare, iar înainte de ieșire din TSR să fie refăcut la valoarea avută.

Activarea unui program TSR presupune întreruperea celui curent executat. În cazul în care codul rezident folosește funcții de sistem, el trebuie să testeze, la activarea sa, dacă programul întrerupt executa o funcție *DOS/BIOS*. Codul rezident nu poate reactiva o funcție de sistem, în timp ce se execută o altă funcție *DOS/BIOS*, deoarece DOS-ul nu este un sistem de operare reentrant (accesul la disc: *13h, 25h, 26h*, nu poate fi întrerupt).

Această problemă este doar pentru TSR-uri active, întrucât cele pasive sunt apelate și se vor executa în cadrul programului apelant, care nu poate executa simultan două apeluri.

Pentru a evita astfel de operații sistemul de operare utilizează două variabile: '*eroare critică*' și '*secțiune*' sau '*stare critică*', pe care le activează corespunzător stării în care se află. Cele două variabile pot fi testate de codul rezident, în momentul în care se utilizează funcții DOS. Adresele celor două variabile, și deci starea lor, sunt returnate de funcțiile *34h* (numită indicator *secțiune critică* sau indicator *inDOS*) și respectiv, *50D6h* (*eroare critică*), în registrele *ES:BX*. Cele două adrese pot fi memorate în zona de date a părții rezidente, de către partea de inițializare a programului TSR, pentru a le testa direct.

Întreruperea multiplexată, INT 2FH

Când se instalează un program TSR care are și componente pasive, trebuie ales un vector de întrerupere pe care să fie instalat, astfel ca alte programe să poată comunica cu rutinele sale, pasive. Se poate alege un vector aproape la întâmplare dar aceasta poate conduce la anumite probleme de compatibilitate. Ce se întâmplă dacă altcineva utilizează deja acest vector de întrerupere ?

În multe situații alegerea este clară, cum ar fi în situația în care codul pasiv al TSR-ului extinde anumite servicii (tastatură, video). În celelalte situații, cum ar fi crearea unui nou driver pentru un nou dispozitiv, care să nu interfereze funcțiile furnizate pentru acest dispozitiv, cu alte întreruperi, alegerea la întâmplare a unui vector de întrerupere este riscantă.

Pentru a rezolva aceste situații DOS-ul furnizează soluția utilizării unei întreruperi multiplexate, *INT 2FH*, care permite instalarea, testarea prezenței și a comunicării cu un program TSR.

Pentru a utiliza această întrerupere, o aplicație plasează o valoare de identificare în *AH* și un număr de funcție în *AL*, și apoi execută *INT 2FH*. Fiecare TSR din lanțul *INT 2FH* va compara valoarea din *AH* cu propria sa valoare unică de identificare. Dacă valoarea sa coincide cu cea din *AH*, atunci TSR-ul respectiv va prelucra comanda specificată de valoarea din registrul *AL*. Dacă nu se identifică, prin valoarea din *AH*, atunci TSR-ul respectiv transmite controlul la următorul manipulator din lanțul *INT 2FH*. Aceasta simplifică cumva problema, dar nu o elimină. De obicei, trebuie ales numărul întreruperii înainte de proiectarea TSR-ului. Cum, însă, o aplicație va ști ce valoare să încarce în *AH*, dacă se asignează dinamic această valoare, când TSR-ul este rezident ?

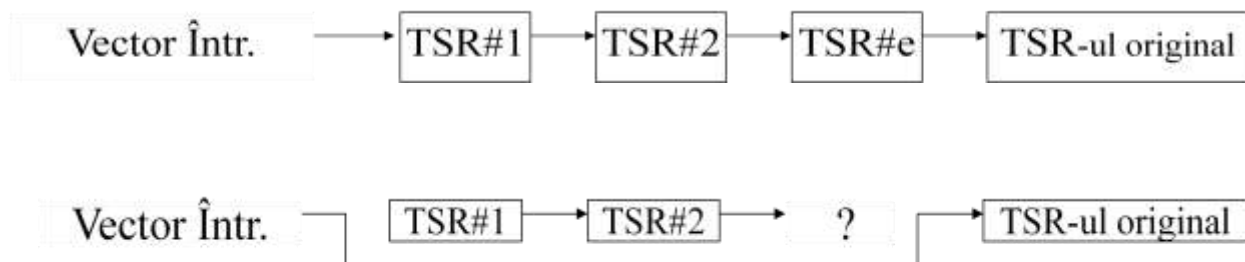
Se permite oricărei aplicații interesate să determine identificatorul (ID) TSR-ului. Prin convenție, funcția zero (*AL=0*) este un apel ce determină prezența TSR-ului. Acest apel se va realiza înainte de solicitarea oricărui serviciu TSR, pentru a determina dacă programul TSR este prezent în memorie.

Dezinstalarea unui program TSR

La dezinstalarea unui TSR trebuie făcute trei lucruri:

- *oprirea/ suspendarea* oricăror activități nerezolvate sau în așteptare (adică, un TSR poate avea anumiți indicatori inițializați pentru a lansa anumite activități ulterioare);
- *refacerea* tuturor vectorilor de întrerupere la valorile lor inițiale;
- *eliberarea* spațiului de memorie ocupat, adică cedarea spațiului de memorie rezervat înapoi către DOS, astfel ca alte aplicații să-l poată utiliza;

Dacă dezinstalarea TSR-ului constă doar în refacerea valorilor vectorilor de întrerupere, se va crea o problemă, dacă există alte TSR-uri care utilizează aceiași vectori de întrerupere. Să presupunem că avem următorul lanț de TSR-uri, pe un anumit nivel de întrerupere:



Aceasta va dezactiva efectiv TSR-urile înlănțuite din program, iar aceste TSR-uri vor avea efecte neașteptate, în aceste condiții. O soluție ar fi tipărirea unui mesaj de eroare, care să informeze utilizatorul că nu se poate dezinstala acest TSR, până nu se dezinstalează toate TSR-urile instalate anterior acestuia.

Problema va fi în a determina TSR-urile înlănțuite în întrerupere. Pentru aceasta știm că vectorii de întrerupere ar trebui încă să refere rutina noastră, dacă acesta este ultimul TSR care se execută. Atunci, trebuie să comparăm vectorii de întrerupere instalați cu adresele rutinelor de servire a întreruperii respective. Dacă toate acestea se potrivesc, atunci se poate elimina în siguranță TSR-ul din memorie. Dacă, însă, doar unul dintre ei nu se potrivește, atunci nu putem elimina TSR-ul din memorie.

