10. Subrutine, întreruperi, şi servicii

10.1 Subrutine

Subrutina este o secvenţă de instrucţiuni scrisă separat, care poate fi apelată din diferite puncte ale unui program. Mecanismul de implementare a subrutinelor este realizat cu ajutorul instrucţiunilor **CALL** şi **RET**. Subrutinele pot fi NEAR (în acelaşi segment cu programul apelant/intrasegment) sau FAR (într-un segment diferit/extrasegment). La apelarea unei subrutine (prin CALL), adresa unde urmează a se face revenirea este salvată pe stivă, iar la revenirea din rutină (prin RET) adresa este refăcută din stivă iar registrul IP (eventual şi CS) se reîncarcă.

Mecanismul de apel a unei subrutine este ilustrat în figura 10.1.

1. program principal
2. apel subrutină
3. execuţie subrutină
4. întoarcere din subrutină
5. reluare program principal

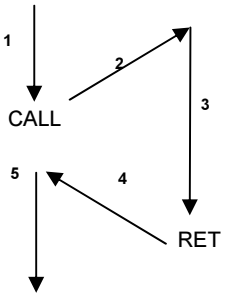


Fig. 10.1. Mecanismul de apel al unei subrutine

Apelurile intrasegment salvează doar offsetul adresei de revenire, iar la RET această valoare este reîncărcată în IP. Apelurile intersegment salvează şi conţinutul registrului CS şi cel al registrului IP, astfel că, la revenire, se vor reface ambele. Pentru cele două cazuri, instrucţiunile CALL şi RET au coduri diferite.
Instrucţiunea RET nu are operanzi; instrucţiunea CALL are un singur operand. Modurile de adresare sunt identice cu cele de la JMP, cu excepţia adresării relative. În cazul în care subrutina alterează (foloseşte) regiştrii a căror valoare este necesară în continuare în programul ppellant, aceşti regiştri trebuie salvaţi pe stivă cu PUSH şi apoi refăcuţi cu POP înainte de revenirea din subrutină.

10.2. Întreruperi

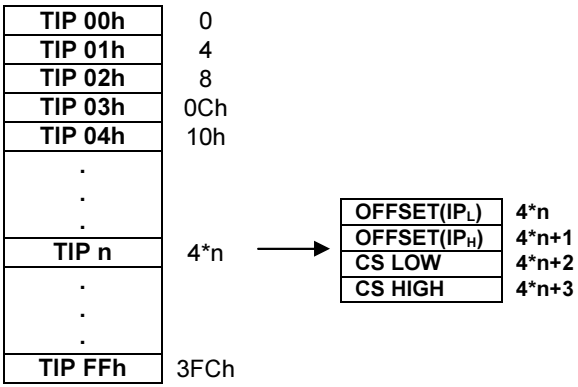
Întreruperea este un semnal transmis sistemului de calcul prin care acesta este anunţat de apariţia unui eveniment care necesită atenţie.

Atunci când evenimentul s-a produs, au loc, în ordine, următoarele acţiuni:
- suspendarea programului în curs de desfăşurare; salvarea pe stivă a adresei de revenire (IP, CS);
- lansarea în execuţie a unei rutine specializate numită rutină de tratare a întreruperii care deserveşte întreruperea;
- reluarea execuţiei programului suspendat prin refacerea de pe stivă a adresei de revenire.

Cauzele acestor evenimente pot fi de două tipuri: interne şi externe. O întrerupere este luată în considerare numai între execuţiile a două instrucţiuni maşină succesive. Dacă apar simultan două întreruperi, circuitele hard ale sistemului de calcul decid care dintre ele va fi servită prima.

La apariţia unei întreruperi, sistemul de calcul trebuie, în această ordine:
- să determine tipul evenimentului care a generat întreruperea (intern, extern),
- să afle care este cauza întreruperii,
- să determine adresa rutinei de tratare a întreruperii (RTI).

Pentru fiecare tip de eveniment şi pentru fiecare cauză posibilă se construieşte câte o RTI. Metoda folosită pentru localizarea rapidă a RTI este vectorizarea întreruperilor. Aceasta constă în a asocia pentru fiecare întrerupere o locaţie de memorie cu adresă fixă. În această locaţie se trece adresa RTI corespunzătoare întreruperii. Microprocesorul 8086 dispune de 256 întreruperi numerotate de la 00h la FFh. Vectorizarea acestora se realizează astfel: la începutul memoriei RAM sunt rezervate 256 de dublucuvinte; fiecare dublucuvânt conţine o adresă FAR a unei RTI. Primul dublucuvânt, aflat la adresa *0000:0000*, conţine adresa RTI pentru intreruperea 00h; al doilea, aflat la adresa *0000:0004* conţine adresa RTI pentru întreruperea 01h, etc. Pentru o întrerupere k, dublul cuvânt care conţine adresa RTI se află la adresa *0000:4*k*



Întreruperile se clasifică în întreruperi hard şi soft. Întreruperea hard 00h provine de la microprocesor (internă) şi apare la tentativa de împărţire la zero; întreruperea 08h provine de la circuitul de temporizare şi este folosită pentru contorizarea timpului; întreruperea 02h semnalizează eroare de paritate la accesarea unei locaţii de memorie şi este o întrerupere nemascabilă, adică declanşarea ei nu poate fi controlată prin flagul *Interrupt.* Întreruperile soft oferă accesul la servicii BIOS şi servicii DOS; sunt foarte mult utilizate datorită facilităţii oferite, o bază de programe care poate fi folosită ca o librărie de programe (rutine) gata scrise, care uşurează mult munca programatorului în limbaj de asamblare. Aceste rutine poartă numele de servicii. Se vor exemplifica servicii pentru câteva întreruperi folosite intensiv în scrierea programelor.

Mecanismul de tratare a unei întreruperi este asemănător cu cel al subrutinelor:

- la lansarea unei întreruperi, indiferent de felul acesteia, starea curentă a microprocesorului este salvată pe stivă; se salvează pe stivă şi PSW; alte întreruperi sunt dezactivate;
- microprocesorul identifică adresa unde se află subrutina de tratare a întreruperii; în acest scop, numărul asociat întreruperii (tipul întreruperii) este folosit ca index în tabloul vectorilor de întreruperi;
- se încarcă în CS şi IP adresa subrutinei din poziţia corespunzătoare a tabloului vectorilor de întrerupere; se execută rutina de tratare a întreruperii până la întâlnirea instrucţiunii IRET;
- se revine din întrerupere prin reîncărcarea lui IP, CS şi PSW cu valorile salvate pe stivă la apelare.

10.3 Instrucţiuni specifice întreruperilor

O întrerupere soft poate fi apelată prin instrucţiunea **INT**, cu sintaxa INT n, provocând activarea handler-ului corespunzător întreruperii cu numărul n. Ea realizează patru acţiuni succesive:

- pune în stivă flagurile (PSW);
- pune în stivă adresa FAR de revenire (CS, IP);
- pune 0 în flagurile TF şi IF;
- apelează prin adresare indirectă handlerul asociat întreruperii.

Instrucţiunile **STI** şi **CLI** acţionează asupra flagului de întreruperi IF, indicând procesorului cum să se comporte la apariţia unei întreruperi. După CLI (Clear Interrupt, IF=0), procesorul nu mai acceptă vreo întrerupere. Apare de obicei la începutul unui handler pentru a evita perturbarea activităţii acestuia. STI (Set Interrupt) permite procesorului să accepte întreruperi, IF=1.

OBS. Întreruperile nemascabile nu ţin cont de starea flagului IF!

Instrucţiunea **IRET** provoacă revenirea dintr-o întrerupere. Ea este ultima instrucţiune executată într-un handler, având efect complementar instrucţiunii INT:

- reface flagurile din stivă;
- revine la instrucţiunea a cărei adresă FAR se află în vârful stivei

```
INT N                          IRET
SP ← Sp-2                      IP ← [SP]
[SP] ← PSW                     SP ← SP+2
I ← 0                          CS ← [SP]
SP ← SP-2                      SP ← SP+2
[SP] ← CS                      PSW ← [SP]
SP ← SP-2                      SP ← SP+2
[SP] ← IP
IP ← [4*N+1 4*N]
CS ← [4*N+3 4*N+2]
```

10.4 Întreruperi şi servicii BIOS şi DOS

BIOS-ul răspunde de gestionarea echipamentelor de intrare/ieşire. Deci, în BIOS sunt scrise o serie întreagă de subrutine legate de aceste echipamente. Apelarea lor într-o aplicaţie se face prin întreruperi; în cadrul fiecărei întreruperi se pot executa mai multe servicii, selecţia serviciului dorit fiind realizată prin încărcarea în registrul AH a unui număr specific serviciului, înainte de apelarea întreruperii. Parametrii de apel ai serviciului se încarcă în anumiţi regiştri, după caz.

Câteva **întreruperi BIOS** sunt prezentate mai jos:

INT 10 h - servicii de ecran / video
INT 13 h - servicii de disc
INT 14 h - servicii de comunicaţii seriale
INT 16 h - servicii de tastatură
INT 20 h - terminare program

Servicii video - INT 10 h

Funcţia 00 - setarea modului video
AH = 00
AL = codul modului video (vezi Anexa 3)
Funcţia 02 - setarea poziţiei cursorului
AH = 02
BH = numărul paginii video (0 pentru modul grafic)
DH = rândul
DL = coloana
Funcţia 09 - scrierea caracterului la cursor (fără deplasarea cursorului)
AH = 09

AL = codul ASCII al caracterului de scris
BH = pagina video
BL = atribut de culoare

Funcţia 0ch - scrierea unui pixel grafic la coordonate
AH = 0ch
AL = culoarea
BH = pagina video
CX = coloana
DX = rândul

Funcţia 0eh - scrierea unui caracter în mod teletype (cu deplasarea cursorului)
AH = 0eh
AL = codul ASCII al caracterului de scris
BH = pagina video

Servicii tastatură - INT 16h

Funcţia 00 - aşteaptă o tastă şi citeşte caracterul tastat
AH = 00
AL = (returnat) codul ASCII al caracterului tastat

Funcţia 01 - citeşte starea tastaturii
AH = 01
Z = 0 - s-a apăsat o tastă; Z = 1 - nu s-a apăsat tastă

Funcţii DOS

Principala întrerupere DOS este **21h**. Sarcinile unora dintre întreruperile amintite mai sus au fost preluate şi uneori extinse de către unele din funcţiile acestei întreruperi.

Funcţia 01 - citire caracter de la tastatură
AH=01
AL = (returnat) caracterul citit

Funcţia 09 - scrierea unui şir de caractere terminat cu "$"
AH = 09
DS:DX - pointer la un şir ce se termină cu caracterul "$"

Funcţia 4ch - terminarea procesului cu cod de retur
AH = 4ch
AL = cod de retur
metodă de terminare a unui program; nu este suportată de versiuni DOS sub 2.x.

Redirectarea unei întreruperi

Rutina de tratare a unei întreruperi oarecare poate fi rescrisă de către utilizator, respectând anumite reguli, legate în primul rând de structura rutinei respective, apoi de manipularea adreselor acestor rutine. În scopul înlocuirii unei RTI cu o rutină scrisă de utlizator, se va înlocui la adresa

corespunzătoare rutinei în TVI cu adresa noii rutine. Ca o măsură firească de precauţie, cel care modifică adresa unui handler trebuie să păstreze adresa veche şi să o refacă atunci când nu mai doreşte folosirea handlerului propriu.
Funcţia 35 a întreruperii 21h permite citirea adresei RTI din TVI pentru o anumită întrerupere dorită de utilizator. Se pune în AH codul funcţiei (35h) iar în AL tipul întreruperii. După apelul INT 21h în regiştrii ES:BX se obţine adresa FAR a handlerului.
Funcţia 25h a întreruperii 21h permite modificarea adresei RTI în TVI pentru o anumită întrerupere dorită. În AH se pune codul funcţiei (25h), în AL se pune tipul întreruperii dorite, iar în DS:DX se pune adresa FAR a noului handler.

10.5 Exerciţii şi teme

1. Studiaţi programul `afisare.asm`; programul face afişarea textului "TASM" în diagonală pe ecran. Modificaţi atributele de culoare pentru afişarea textului.

```
.model small
.stack 200h
.data
        text db 'TASM',0
.code

linie   proc    near
        mov     cx,1
        mov     dx,0
linie_iar:
        lea     si,text
iar:
        mov     ah,2
        int     10h
        mov     al,[si]
        cmp     al,0
        jz      endtext
        mov     ah,9
        int     10h
        inc     dl
        inc     si
        cmp     dl,80
        jnz     iar
endtext:
        inc     dh
        cmp     dh,25
        jc      linie_iar
        ret

linie   endp
```

```
main label

        mov     ax,@data
        mov     ds,ax
        mov     ah,0
        mov     al,2
        int     10h
        mov     bh,0
        mov     bl,1fh
        call    linie
        mov     ax,4c00h
        int     21h

end main
```

2. Scrieţi un program care afişează un mesaj de test definit în segmentul de date, folosind pentru afişare funcţii ale întreruperii 10h şi 21h; Nu omiteţi încheierea programului! (int 21h, serviciul 4ch sau prin ret)

```
org 100h
.data
    msg db "Hello, World", 24h

.code
    mov ax, @data
    mov ds, ax

    mov dx, offset msg
    mov ah, 9
    int 21h
.exit
```

3. Completaţi programul de adunare a două numere prin afişarea acestora şi a sumei pe ecran. În acest scop trebuie realizată conversia valorilor din binar în ASCII, pentru a putea fi afişate folosind funcţiile prezentate.

4. Considerăm următorul program în limbaj de asamblare:

```
        org     100h
        mov     si,0
        mov     dl,30h
        mov     ah,2
lop:    int     21h
        inc     dl
        inc     si
        cmp     si,8h
        jne     lop
        int     20h
```

a. Ce conţin regiştrii SI şi DL în momentul în care programul iese din buclă?

b. Ce se va afişa pe ecranul PC-ului la terminarea programului?

```
SI conţine 08H
DI conţine 38H
```

Pe ecran va fi scris: `01234567`

5. Scrieţi un program care începe să citească caractere de la tastatură şi să stocheze codul lor ASCII în memorie începând de la adresa 2000h, până la apăsarea tastei Enter (0Dh).

```
        org     100h
        mov     si,0
        mov     ah,1
lop:    int     21h
        cmp     al,0dh
        je      out1
        mov     2000h[si],al
        inc     si
        jmp     lop
out1:   int     20h
```

6. Comentaţi linie cu linie următorul program şi apoi explicaţi ce se va afişa pe ecranul PC-ului. Pentru verificare rulaţi programul cu emu8086:

```
name "colors"
org     100h

        mov     ax, 3
        int     10h

        mov     ax, 1003h
        mov     bx, 0
        int     10h

        mov     dl, 0
        mov     dh, 0

        mov     bl, 0

        jmp     next_char

next_row:
        inc     dh
        cmp     dh, 16
        je      stop_print
        mov     dl, 0
```

```
next_char:
        mov     ah, 02h
        int     10h

        mov     al, 'a'
        mov     bh, 0
        mov     cx, 1
        mov     ah, 09h
        int     10h

        inc     bl

        inc     dl
        cmp     dl, 16
        je      next_row
        jmp     next_char

stop_print:

        mov     dl, 10
        mov     dh, 5
        mov     ah, 02h
        int     10h

        mov     al, 'x'
        mov     ah, 0eh
        int     10h

        mov ah, 0
        int 16h

ret
```