



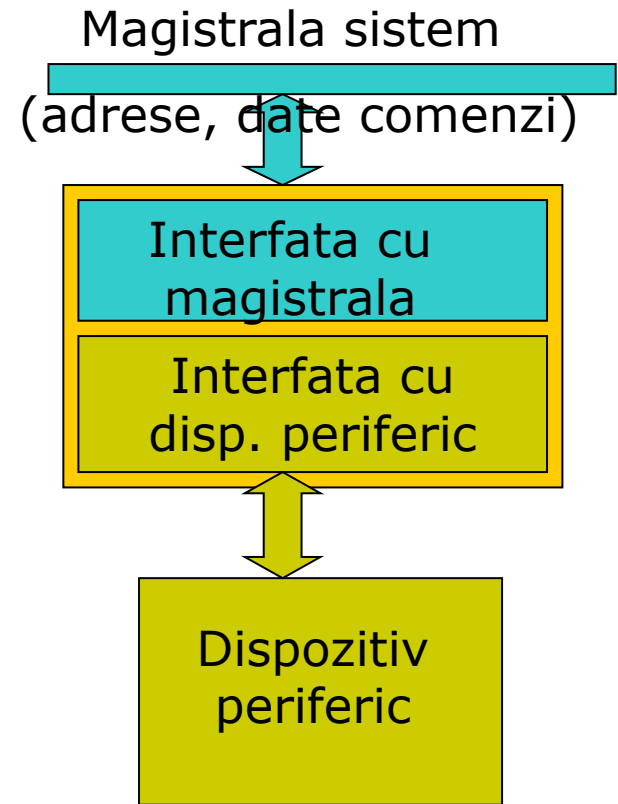
Sisteme cu microprocesoare

Cursul 8

Proiectarea interfetelor de intrare/iesire

Interfete de intrare/iesire

- Rolul: adaptarea particularităților unui/unor dispozitiv(e) periferice de intrare/ieșire la cerințele unui anumit sistem de calcul (adaptarea la o magistrală)
- Ce se adaptează:
 - Semnale
 - Secvența de transmisie și recepție a datelor
 - Ritmul/viteza/frecvența de transfer – interfata este un buffer temporar de date
- Structura: 2 părți:
 - Adaptor de magistrală
 - Adaptor pentru un anumit dispozitiv periferic



Schema de principiu a unei interfete de intrare/iesire

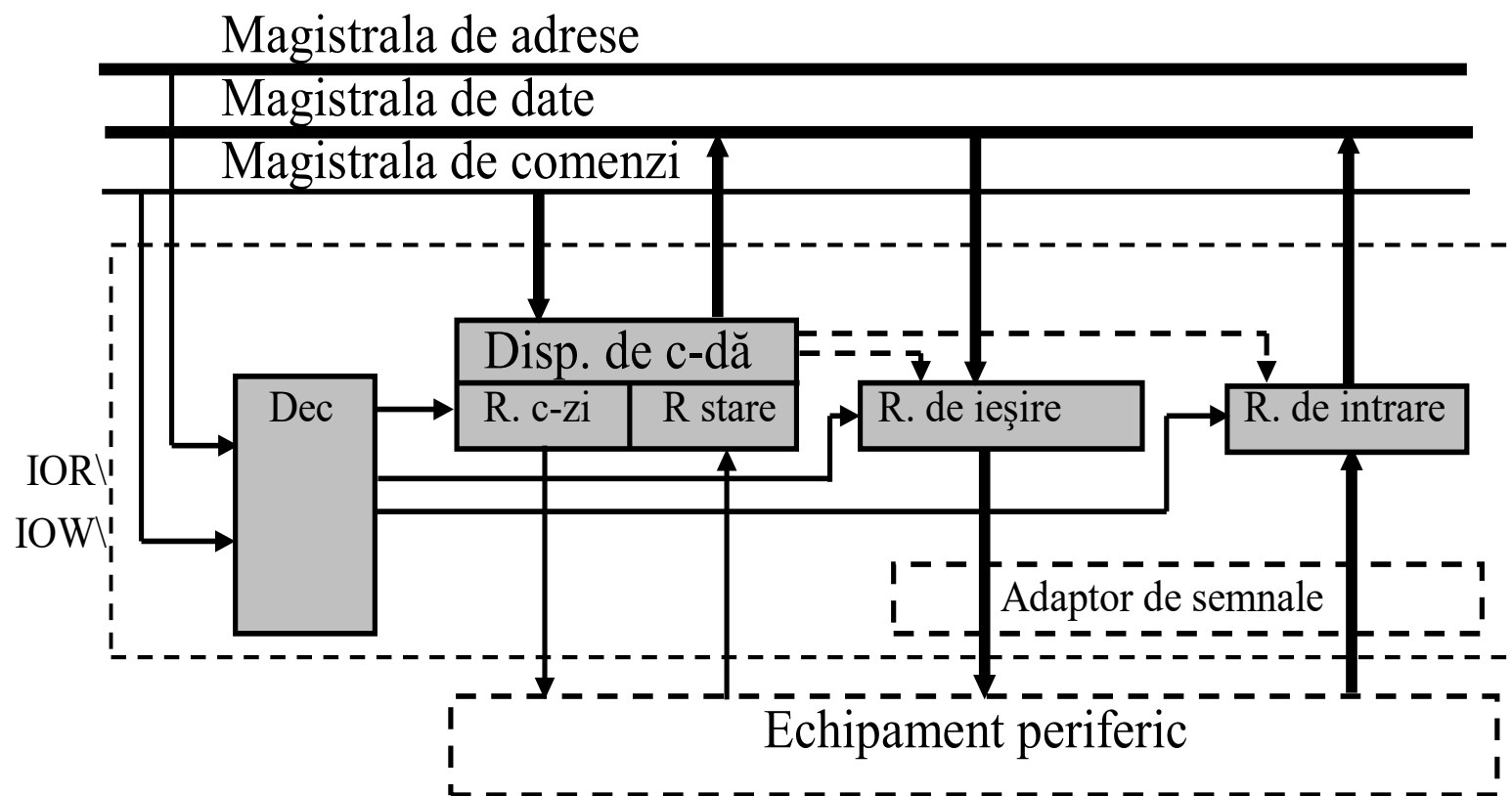


Figura 10-1 Schema de principiu a unei interfete de intrare/ieșire

Componente:

- Interfata de magistrala:
 - registre de date (de intrare și/sau ieșire)
 - registre de comenzi
 - registre de stare
 - bloc de selecție a registrelor (decodificator)
 - dispozitiv de comandă
- Interfata de dispozitiv periferic:
 - Adaptoare/convertoare de semnal
 - Memorie tampon (RAM)
 - Memorie de program (driver)
 - Controlor de dispozitiv (floppy, HDD, video, etc.)
 - Controlor DMA
 - Etc.

Moduri de transfer

- Functia principala a unei interfete: Transferul de date
- Tipuri de transfer:
 - **Transfer prin program**
 - **Transfer prin intreruperi**
 - **Transfer prin acces direct la memorie (DMA)**
 - **Transfer prin procesor de intrare/iesire**
- Diferite grade de implicare a procesorului principal
- Alegerea modului de transfer optim:
 - Viteza de transfer
 - Complexitatea transferului
 - Restrictii de timp si de cost

Fazele unui transfer

- Inicializarea transferului
 - sarcini:
 - directia de transfer,
 - nr. de date transferate,
 - adresa zonei de memorie unde/de unde se transfera
- Transferul propriu-zis
 - Sarcini:
 - Transferul de date
 - Sincronizarea transferului cu disp. Periferic
- Finalizarea transferului
 - Sarcini:
 - Verificarea corectitudinii transferului



Transferul prin program

- Cel mai simplu mod de transfer
 - Hardware simplu
 - Program simplu de transfer
- Procesorul se ocupa de toate fazele transferului, pe baza unui program (rutina, driver)
 - Initializare
 - Transfer:
 - Finalizare:

Exemplu de program – fara semnal de control (de reactie)

```
buf:      DB      100 DUP(?)
lbuf      EQU     $-buf                ; lungimea bufferului
adrport   EQU     300h                 ; adresă port de intrare
.....
; inițializare transfer
        mov si, offset buf            ; inițializare poantor memorie
        mov dx, adrport               ; inițializare adresă port
        mov cx, lbuf                  ; inițializare contor
.....
; transfer de date
et1:     in al, dx                     ; citire port de intrare
        mov [si], al                 ; memorare dată
        inc si                        ; avans poantor
        call delay                    ; apel rutină de întârziere
        loop et1                      ; buclare (testare încheiere transfer)
.....
```


Exemplu de program – cu semnal de control (de reacție)

```
buf: DB      100 DUP(?)
lbuf EQU     $-buf                ; lungimea bufferului
adrport EQU   300h                ; adresă port de intrare
adrstare EQU  301h                ; adresă registru de stare
masca EQU     01h                ; mască pentru bitul D0
timeout EQU   100h                ; limită de timp
.....
; inițializare transfer
    mov si, offset buf            ; inițializare poantor memorie
    mov dx, adrport              ; inițializare adresă port
    mov cx, lbuf                 ; inițializare contor
.....
; transfer de date
et1: mov bx, timeout              ; inițializare limită de timp pt.
                                ; așteptare
    inc dx                       ; adresă registru de stare
et2: in al, dx                   ; citire port de intrare
    and al, masca                ; testare bit de stare
    jnz et3                      ; testare bit de stare setat
    dec bx                       ; decrementare contor de timp
    cmp bx, 0                    ; test depășire limită de timp
    jnz et2
    jmp sfîrșit                 ; salt la sfîrșit
et3: dec dx                      ; adresă port de intrare
    in al, dx                    ; citire port
    mov [si], al                 ; memorare dată
    inc si                       ; avans poantor
    loop et1                     ; buclare (testare încheiere transfer)
; verificarea corectitudinii transferului
sfîrșit: cmp cx, 0                ; se verifică dacă transferul s-a încheiat
    jnz eroare                  ; corect (contor=0)
.....
```

Transferul prin program, avantaje si dezavantaje

○ Avantaje:

- Simplu
- Usor de implementat
- Ieftin

○ Dezavantaje:

- Lent
- Procesorul este blocat pe toata durata transferului
- Ineficient

Transfer prin întreruperi

- Caracteristicile transferului:
 - Soluționează problema sincronizării în faza de transfer propriu-zis
 - Procesorul poate efectua alte operații în paralel cu transferul
 - Procesorul se ocupă pe mai departe de inițializare, transfer de date și finalizare
- **Întreruperea**: oprirea temporară a execuției unui program, la comanda unui semnal extern sau a unui eveniment intern
 - Deservirea unei întreruperi: prin execuția unei **rutine de întrerupere**
 - **Tabela de întreruperi**: conține adresele rutinelor de tratare a întreruperilor

Tratarea intreruperilor multiple

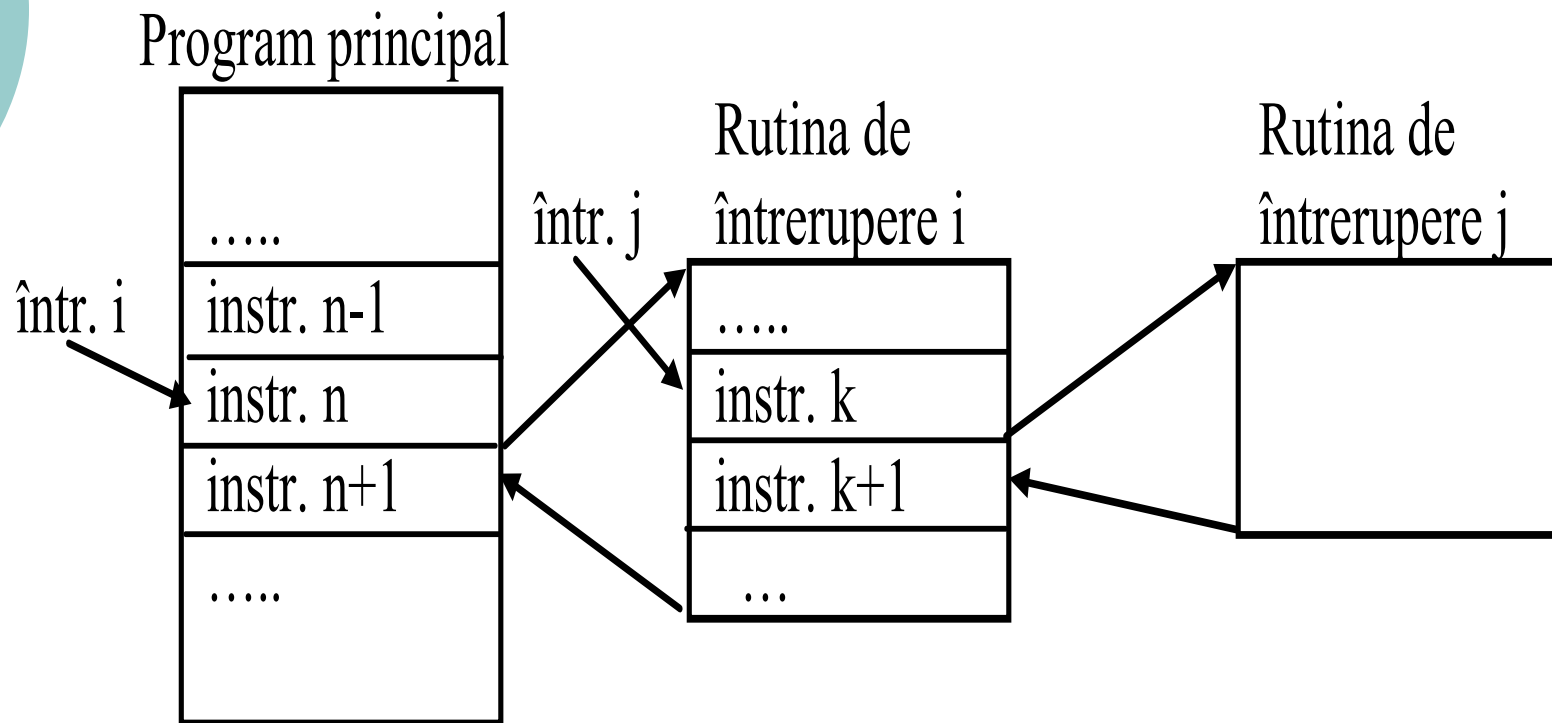


Figura 10-2. Tratarea întreruperilor multiple

Sistemul de întreruperi al familiei de procesoare Intel x86

- 256 de nivele de intrerupere
- Intreruperi:
 - Interne:
 - Divizare cu 0, intreruperi software (INT n), intrerupere de trasare, depasire de capacitate, breakpoint
 - Externe:
 - Nemascabile: NMI
 - Mascabile INTR
- Validarea/invalidarea intreruperilor:
 - STI: set interrupt – IF=1 - validare
 - CLI: clear interrupt – IF=0 – invalidare (mascare)

Deservirea unei cereri de intrerupere de catre microprocesor

1. salvează pe stivă adresa instrucțiunii următoare și conținutul registrului de stare program PSW
2. generează două cicluri INTA (INTerrupt Acknowledge) pentru confirmarea acceptării întreruperii și pentru identificarea sursei de întrerupere; în al doilea ciclu citește vectorul întreruperii de pe magistrala de date
3. execută salt la rutina de tratare a întreruperii; adresa rutinei este citită din tabela de întreruperi, folosind ca index vectorul de întrerupere citit anterior
4. la terminarea execuției rutinei de întrerupere extrage de pe stivă starea programului întrerupt și adresa de revenire
5. se continuă execuția programului întrerupt

Tipuri de intreruperi

- Hardware: generate de un semnal extern
 - Semnale de intrerupere: INTR, NMI
 - Semnal de identificare a intreruperii: INTA/
 - Surse de intrerupere:
 - Interfete de intrare/iesire: mouse, tastatura, HDD, retea
 - Detectoare de defect: eroare de paritate la date
- Software: generate prin instructiuni
 - Simuleaza intreruperile hardware
 - Folosite pentru apelul unor functii ale sistemului de operare: apeluri sistem (INT 21H)
 - Tipuri de instructiuni de intrerupere:
 - INT n
 - INTO
 - INT 3 (breakpoint)

Controlorul de întreruperi I8259A

- Responsabil pentru gestionarea a 8 întreruperi externe

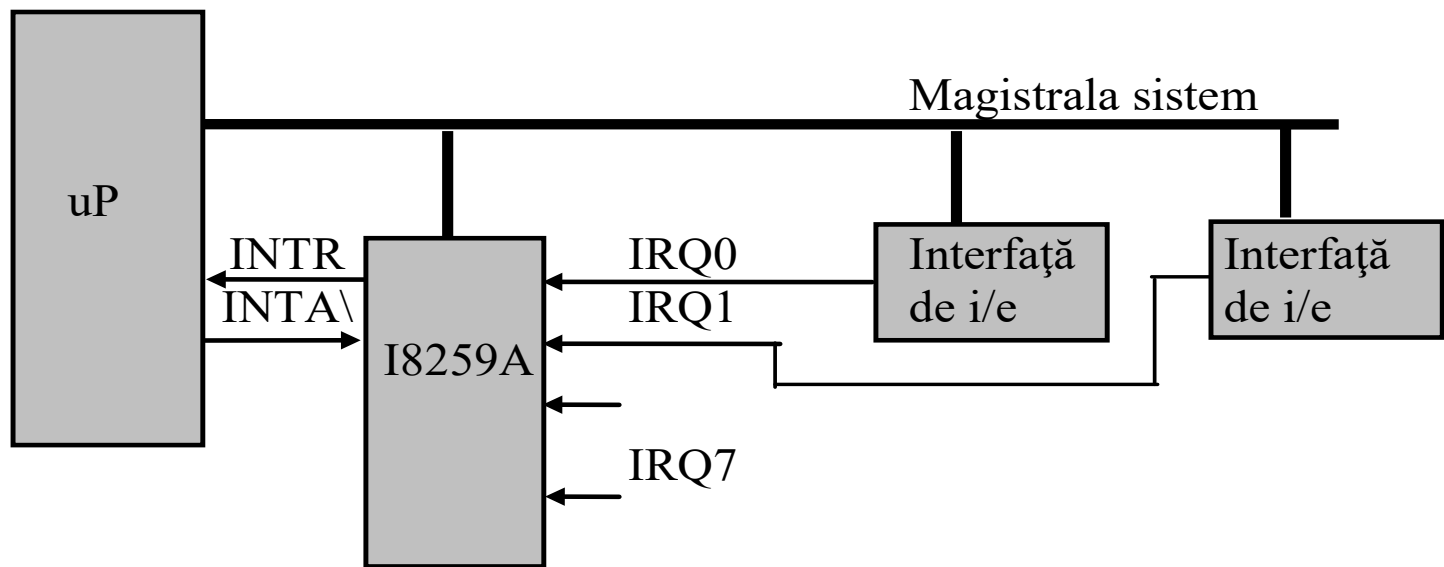


Figura 10-3. Conectarea controlorului I8259A în sistem

Structura internă a controlului de întreruperi

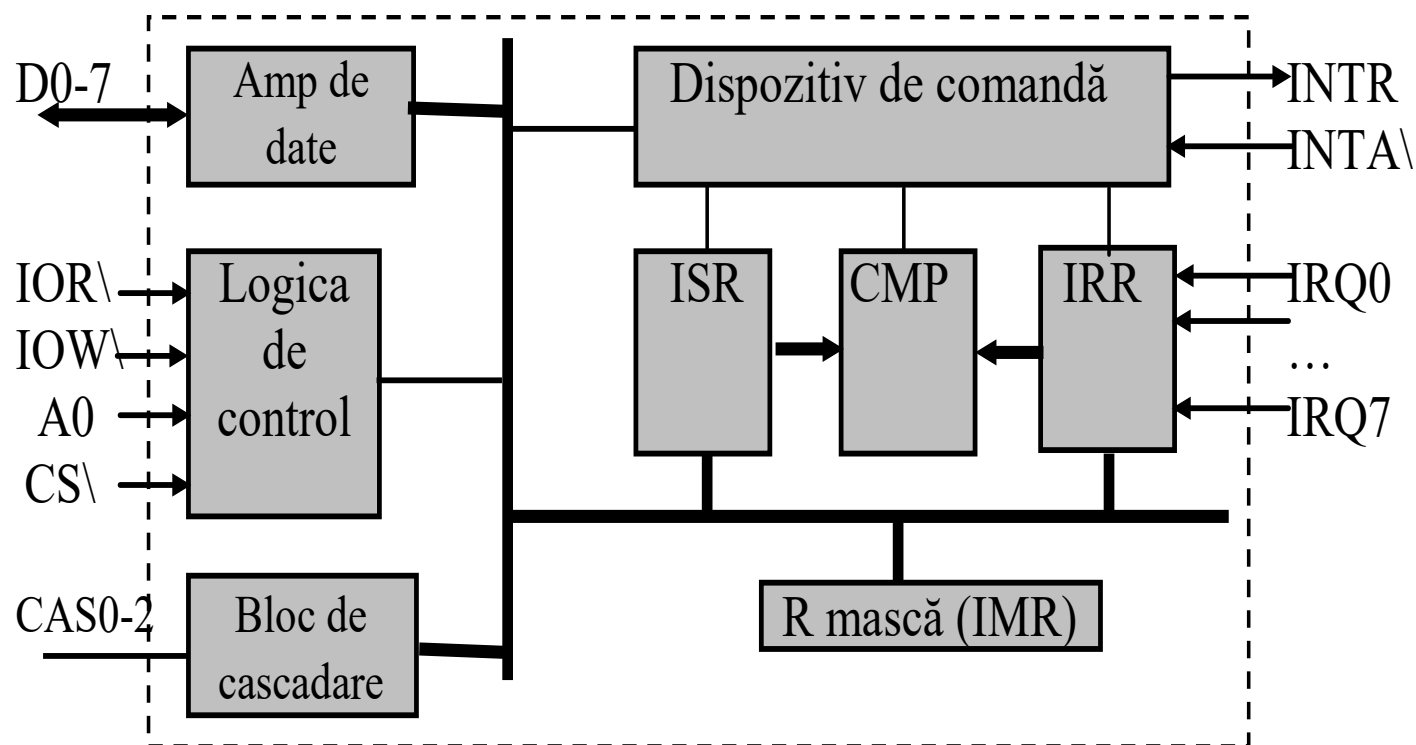


Figura 10-4. Structura internă a controlului de întreruperi I8259A

Programarea controlorului de întreruperi

- cuvinte de initializare (folosite o singură dată la pornirea sistemului) - ICW0-4 (Initialization Command Word)
- cuvinte de operare (folosite pentru modificarea dinamică a modului de lucru) - OCW0-3 (Operation Command Word)
- Prin programare se definesc următorii parametri de funcționare:
 - modul de lucru 8080 sau 8086
 - modul de detecție a întreruperilor: pe front (la tranziția din 0 în 1) sau pe nivel
 - modul de imbricare a întreruperilor: normală sau specială (imbricare=deservirea unei alte întreruperi în timpul execuției unei rutine de întrerupere)
 - tipul de prioritate: fixă sau rotativă
 - poziția întreruperilor deservite în tabela de întreruperi (adresa de bază)

Secventa de deservire a unei intreruperi

1. **Interfata** genereaza o intrerupere prin activarea unui semnal IRQi catre **controlor**
2. **Controlorul** verifica daca intreruperea este permisa si nu este alta mai prioritara in curs de deservire
3. Daca intreruperea este permisa se genereaza o intrerupere catre **procesor** prin activarea semnalului INTR
4. Dupa terminarea instructiunii curente, **Procesorul** genereaza doua cicluri INTA pt. identificarea sursei de intrerupere
5. Pe al doilea ciclu **controlorul** pune pe liniile de date ale magistralei vectorul intreruperii curente
6. **Procesorul** foloseste vectorul ca index in Tabela de intreruperi pentru a citi adresa Rutinei de intrerupere
7. **Procesorul** salveaza pe stiva adresa de revenire si starea dupa care face un salt la Rutina de intrerupere
8. La sfarsitul rutinei de intrerupere **procesorul** da o comanda catre **controlor** pentru incheierea intreruperii curente (EOI)
9. La executia instructiuni IRET (revenire din rutina de intrerupere) **procesorul** ia de pe stiva adresa de revenire si face salt la aceasta adresa, continuind programul intrerupt

Sistemul de întreruperi al calculatoarelor personale compatibile IBM PC AT

- 2 controloare de intrerupere
- Intreruperi hardware:
 - Timer de ceas
 - Tastatura
 - Canale seriale
 - Interfata de imprimanta
 - Interfata FDD/HDD
 - Interfata de retea
- Intreruperi software:
 - Intreruperi BIOS: tastatura, interfata video, canal serial, interfata de imprimanta, interfata floppy
 - Apeluri sistem (INT 21h): citire canal standard de intrare, scriere canal de iesire, citire/scriere disc, etc.

Exemplu de redirectare a unei intreruperi software

```
VECT  DW      2 DUP(?)      ; aici se salvează vechiul vector de
; întrerupere
;initializare
.....
MOV   AH, 35h                ; 35h-funcție sistem pentru citirea
;vectorului de întrerupere
MOV   AL, n                  ; n – nivelul de întrerupere redirectat
INT 21h                      ;apelul funcției sistem; funcția returnează
;în ES:BX adresa rutinei de la nivelul n
MOV   VECT, BX              ; salvare adresă offset
MOV   BX, ES
MOV   VECT+2, BX            ; salvare adresă segment
MOV   AX, SEG RUT_INT
MOV   DS, AX                ;DS <- adresa de segment a noii rutine de
; tratare a întreruperii
MOV   DX, OFFSET RUT_INT    ; DX <- adresa de offset a noii
; rutine de întrerupere
MOV   AH, 25h                ;25h - funcția de scriere vector; în
;DS:DX se pune adresa rutinei de întrer.
MOV   AL,n                  ;n – nivel de întrerupere
INT 21h                      ;apelul funcției sistem
.....
; program
.....
INT n
.....
; sfârșit program
MOV   AX, VECT+2            ; refacerea adresei
MOV   DS, AX
MOV   DX, VECT
MOV   AH, 25h                ; funcția de scriere vector
MOV   AL,n                  ; n – nivel întrerupere
INT 21h                      ; înscrierea vechiului vector în
;tabela de intreruperi
.....
```

Continuare

```
;rutina de tratare a întreruperii
RUT_INT PROC FAR ; noua rutina de tratare a
    intreruperii
    PUSH r ; salvarea registrelor utilizate în
    cadrul
; rutinei (r = AX, BX, ....)
    STI ; validare întrerupere
    .....
; corpul rutinei
    .....
; sfîrșitul rutinei
    POP r ; refacere registre salvate
    IRET
RUT_INT ENDP
```

Exemplul 2 – programarea unei intreruperi hardware

```
INTA00 EQU    20H          ; adresa portului 0 din controlorul de întreruperi
INTA01 EQU    21H          ; adresa portului 1 din controlorul de întreruperi
EOI EQU       20h          ; comanda de incheiere intrerupere
MASCA EQU     11011111B ; mască pentru validarea intrării nr. 5
MASCA1 EQU    00100000B    ; mască pentru invalidarea intrării nr. 5

.....
; initializare întrerupere
    CLI                                ; invalidarea întreruperilor mascabile
MOV  AX, SEG RUT_INT
MOV  DS, AX
MOV  DX, OFFSET RUT_INT
MOV  AH, 25h    ; 25h - funcția de scriere vector
MOV  AL, 5+8    ; 5+8= nivel de întrerupere programat
; adresa de bază a primului controlor de întreruperi este 8
INT21h          ; apel de funcție sistem
MOV  DX, INTA01
IN   AL, DX      ; citire registru de mască
AND  AL, MASCA   ; șterge bit mască pt. întreruperea 5
OUT  DX, AL      ; validare intrare de întrerupere
STI                                ; validare întreruperi mascabile

.....
; program
.....
```

Continuare

- ; sfîrșit program
CLI
MOV DX, INTA01
IN AL,DX ; citire registru de mască
OR AL, MASCA1 ; setează bitul mască pt. întreruperea 5
OUT DX,AL ; invalidarea intrării de întrerupere
STI
.....
;rutina de tratare a întreruperii
RUT_INT PROC FAR ; noua rutina de tratare a intreruperii
PUSH r ; salvarea registrelor utilizate in cadrul rutinei
STI ; validare întreruperi
.....
; corpul rutinei
.....
;sfîrșitul rutinei
MOV DX, INTA00
MOV AL,EOI
OUT DX, AL ; comandă de sfîrșit întrerupere
POP r ; refacere registre salvate
IRET
RUT_INT ENDP