Motto:

"Nu calea conteaza, ci modul cum o parcurgem. " Fer. V. Ghika

Curs 9

Setul de instructiuni 8086
- Proceduri si Intreruperi

Setul de instructiuni de baza 8086/88

INSTRUCTIUNI:

- **Transfer**
- **Aritmetice/logice(2)**
- **Manipulare siruri**
- **Ramificare**
- **Control procesor**
- **Proceduri. Intreruperi**

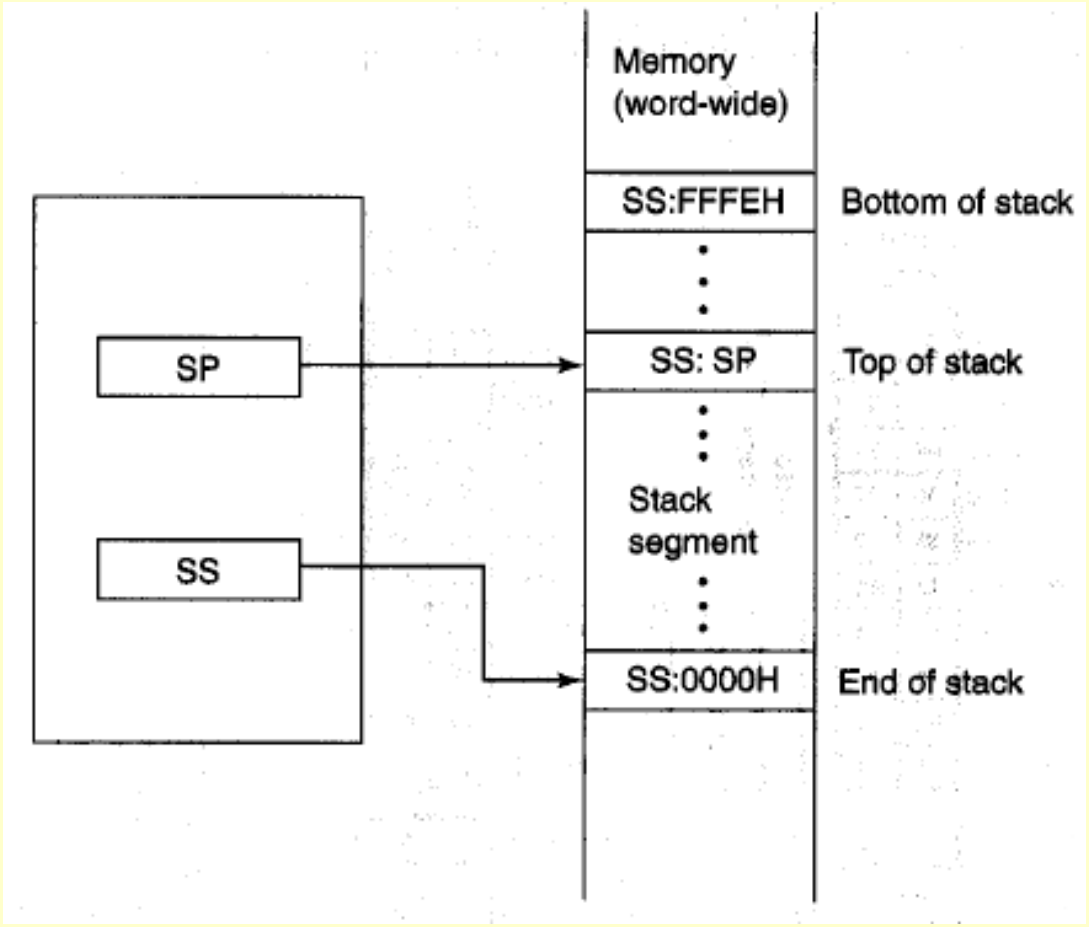
Proceduri. Intreruperi

- Operatii cu stiva (Stack SS:SP)
- Definirea si utilizarea Procedurilor
- Instructiuni pentru subrutine
- Intreruperi

Operatii cu Stiva

- ***De ce stiva?***

- Apelul Procedurilor este o operatie cu stiva
- Stiva se foloseste ptr. a pastra adresa de revenire
- Parametrii si variabilele locale se pun pe stiva cand apelam o subrutina



Operatii cu Stiva

- ***Concepte***

- STIVA este o structura LIFO (last-in first-out)
- STIVA este o zona de memorie care este controlata direct de CPU, folosind 2 registri: *SS si SP (ESP)*
- SS pastreaza un descriptor de segment care nu este modificat de programul utilizator
- SP (ESP) pastreaza un offset de 16 (32) biti in cadrul stivei
- Stivele INTEL “cresc” spre adrese mici (downward)

Operatii cu Stiva

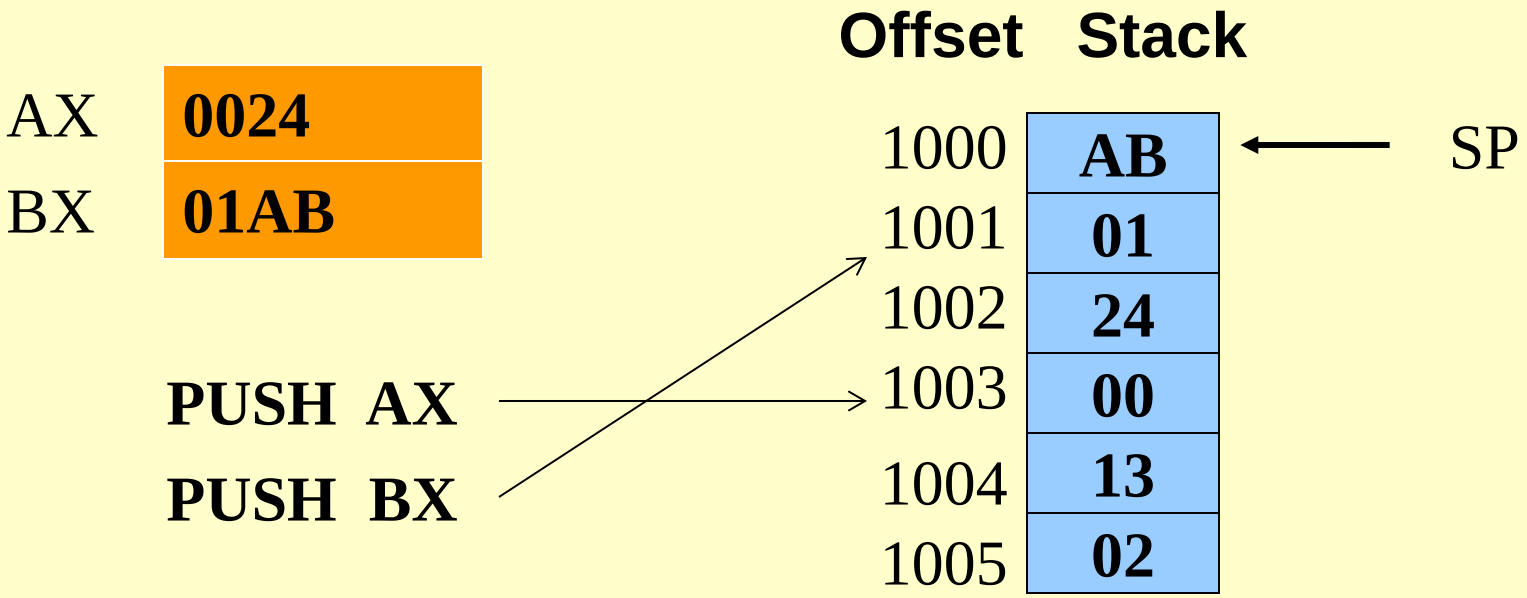
- *Operatii PUSH*
  - Format: *PUSH r/m16/ m32/ im16/im32*

- Valori imediate :16-biti in mod real si 32-biti in mod protejat
- Un operand pe 16-biti/32-biti cauzeaza decrementarea reg. SP(ESP) cu 2 resp. 4

- *Operatii POP*
  - Format: *POP r/m16/ m32*

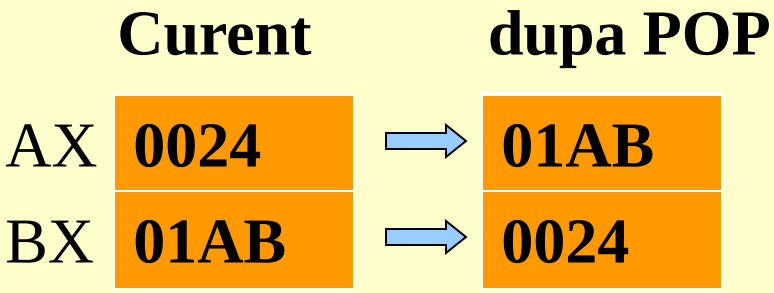
Operatii cu Stiva

- *Exemple cu PUSH/POP*



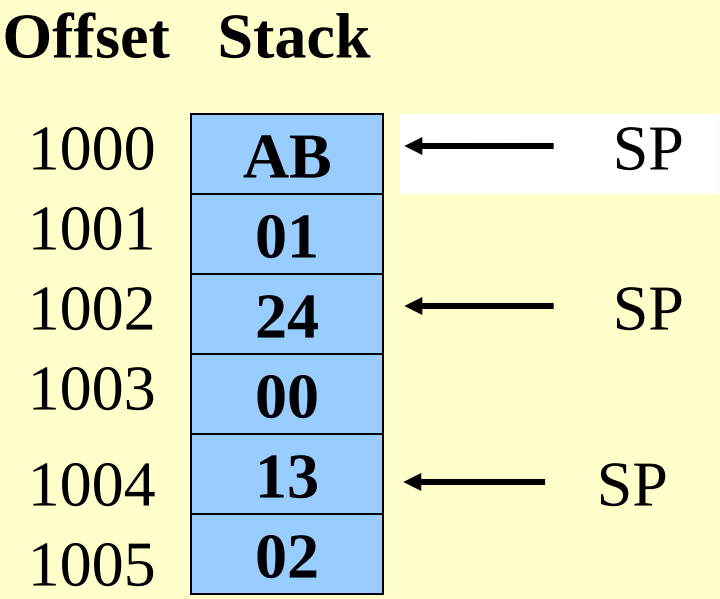
Operatii cu stiva

- *Exemple cu PUSH/POP*



POP AX

POP BX



- Zona stivei deasupra valorii lui SP(ESP) este logic libera si se va rescrie (overwrite)

Proceduri

- ***Concepte***

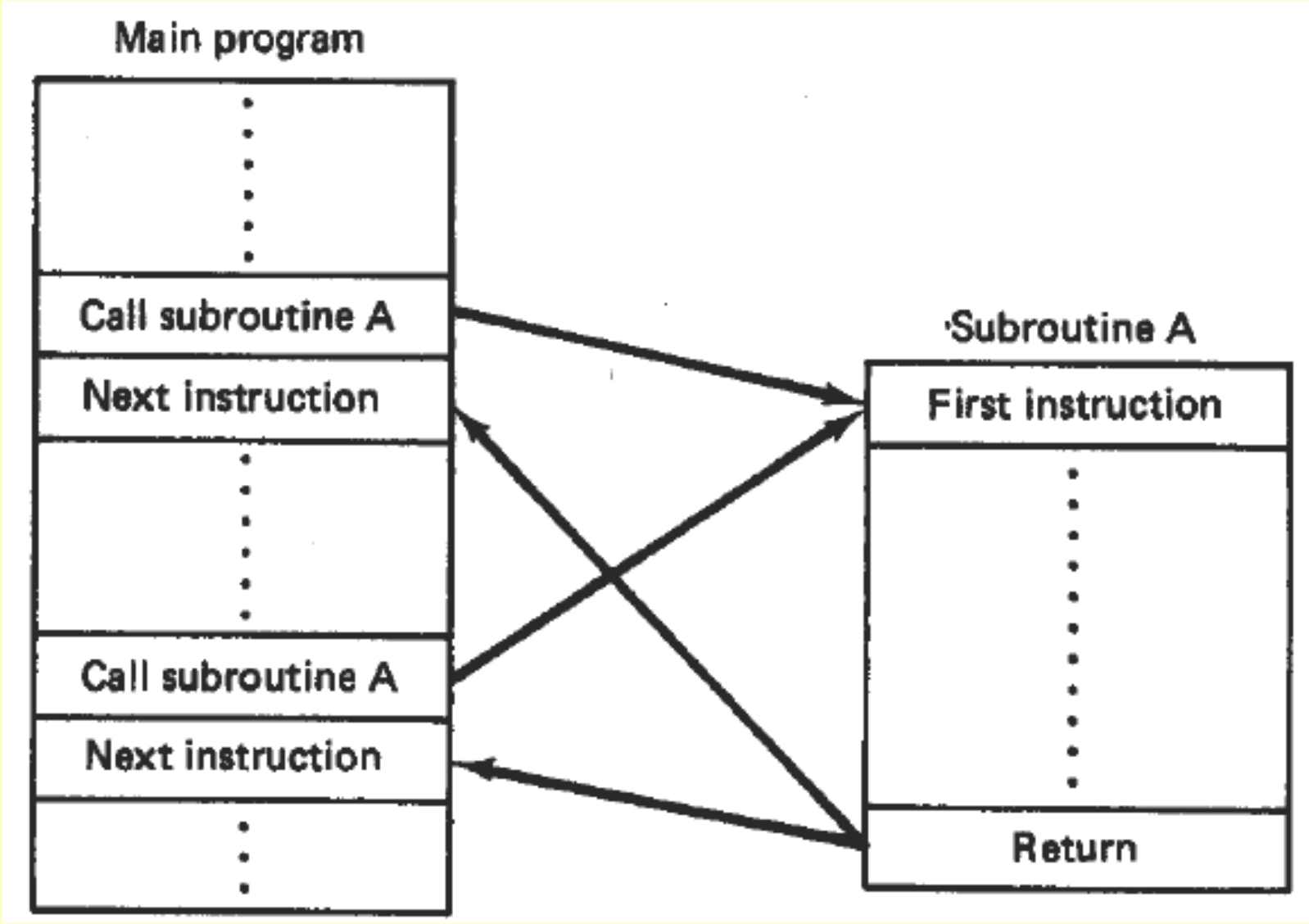
➢ **Procedura** : bloc de cod care poate fi apelat (called) si din care se poate reveni (return)

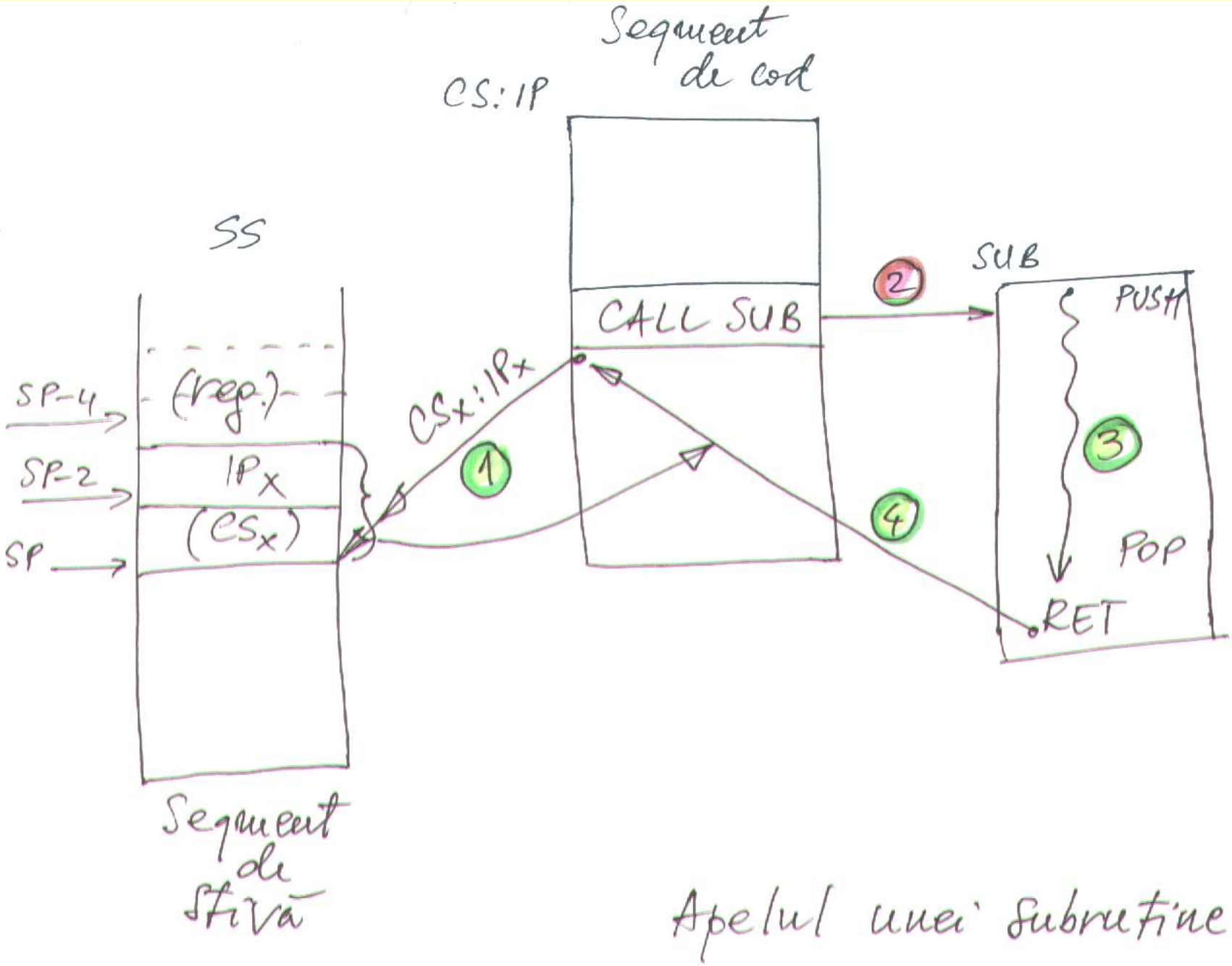
❑ O procedura incepe cu ***nume* proc** si se termina cu ***nume endp***

❑ Pentru a incheia o procedura se foloseste instructiunea **ret** *exceptand procedura principala (*main);

❑ Pentru apelul unei proceduri se foloseste instructiunea : **call *nume***

Proceduri


Main program
Call subroutine A
Next instruction
Call subroutine A
Next instruction
Subroutine A
First instruction
Return



Apelul unei subrutine.

Proceduri

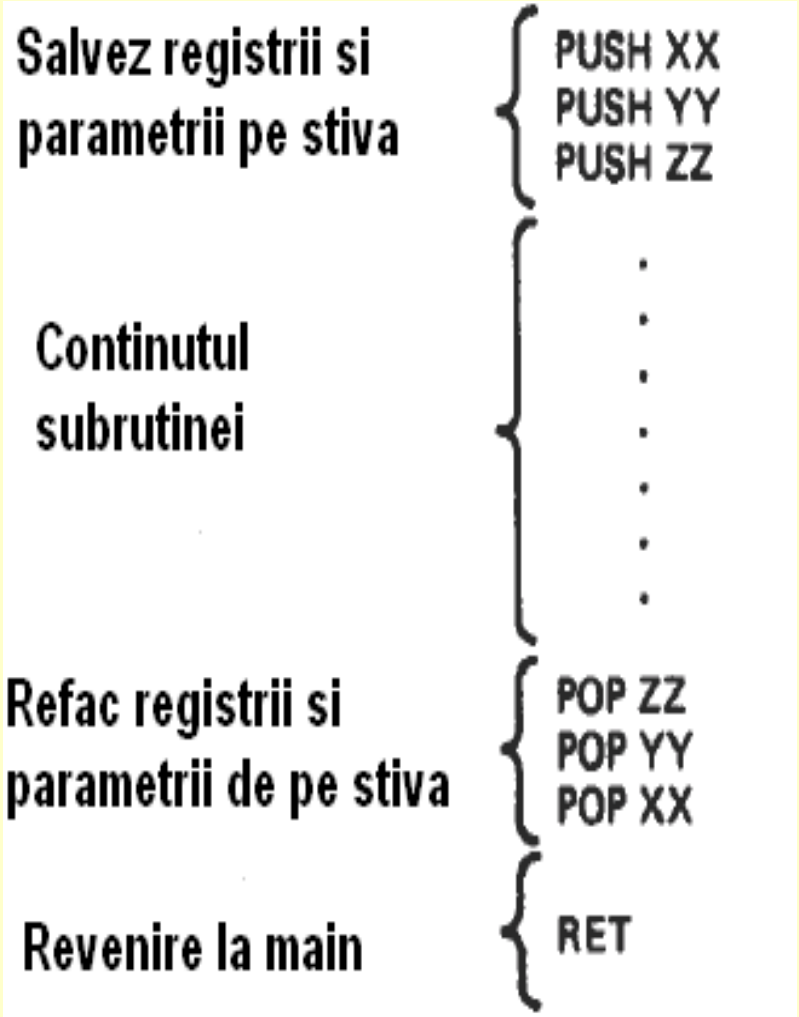
- **OBS.**

1.Este indicat sa se salveze continutul registrelor la intrarea in procedura si sa se refaca inainte de revenire

2. Tipul instr. Call si RET ≡

3. Adr. de revenire sa Nu se altereze

4. SP in momentul executiei instr. RET ≡ cu cea la intrarea in proc.



Proceduri

- ***Instructiunile CALL si RET***

➢ **CALL: redirectioneaza procesorul sa inceapa executia la o noua locatie de memorie**

- ❑ Salveaza adresa de revenire pe stiva
- ❑ Copiaza adresa procedurii apelate in IP(EIP) / IP,CS

Proceduri

Mnemonic	Meaning	Format	Operation	Flags Affected
CALL	Subroutine call	CALL operand	Execution continues from the address of the subroutine specified by the operand. Information required to return back to the main program such as IP and CS are saved on the stack.	None

(b)

Operand
Near-proc
Far-proc
Memptr16
Regptr16
Memptr32

Proceduri

Instructiunea Call - implementeaza 2 tipuri de apel

- **Intra-segment call (IP)**
- **Inter-segment call (IP,CS)**

- Intra-segment call —> adr. de inceput a subrutinei este localizata in segmentul de cod curent (CS nu se modifica)

- Numai IP isi schimba valoarea

Near-proc

1. operand =16-biti codat in instructiune

Ex. CALL 1234H

Operatii:

- **1. IP-ul instructiunii urmatoare salvat pe varful stivei (SS:SP)**
- **2. SP =SP-2**
- **3. IP=1234h; Curent CS:New IP (CS:1234h)**
- **4. Se citeste (fetch) prima instr. a subrutinei …**

Proceduri

2. Operand= regptr16

- **Valoarea pe 16-biti care se incarca in IP este specificata in registru**
- ***Adresare la Registru***

Ex.

- **CALL BX ; (BX) => New IP, IP=BX**
- **Se continua de la CS: BX**

3. Operand= memptr16

- **Valoarea pe 16-biti care se incarca in IP este specificata in memorie**
- ***Adresare memorie***

Exemplu:

- **CALL [BX] ; (DS:BX) => New IP, IP=(DS:BX)**
- **Se continua de la CS: (DS:BX)**

Proceduri

Inter-segment call — adresa de inceput a subrutinei se afla intr-un alt segment de cod

- Valorile reg. CS si IP se schimba - ***far-proc***

1. Operand= val. imediata 32-biti codata in instructiune

Noua adresa se calculeaza ca:

- primii 16 biti >> New IP
- urmatorii 16 biti >> New CS

Subrutina incepe la = (New CS: New IP)

2. Operand=memptr32

- **Valoare pe 32-biti specificata in memorie**
- **Adresare memorie**

Exemplu:

CALL DWORD PTR [DI] ; (DS:DI) >> New IP, (DS:DI +2) >> New CS

- Adresa de start a subrutinei = (New CS:New IP)

Proceduri

- **RET: aduce procesorul inapoi in punctul de program unde procedura a fost apelata**
- Reface adresa de revenire de pe stiva in IP/ IP:CS

Mnemonic	Meaning	Format	Operation	Flags Affected
RET	Return	RET or RET Operand	Return to the main program by restoring IP (and CS for fat-proc). If Operand is present, it is added to the contents of SP.	None

(a)

Operand
None Disp16

Proceduri

- Return
- Fiecare subrutina trebuie sa se incheie cu return
- Initiaza revenire la executia instructiunii din programul principal (main) ce urmeaza instructiunii call a subrutinei

Ex.

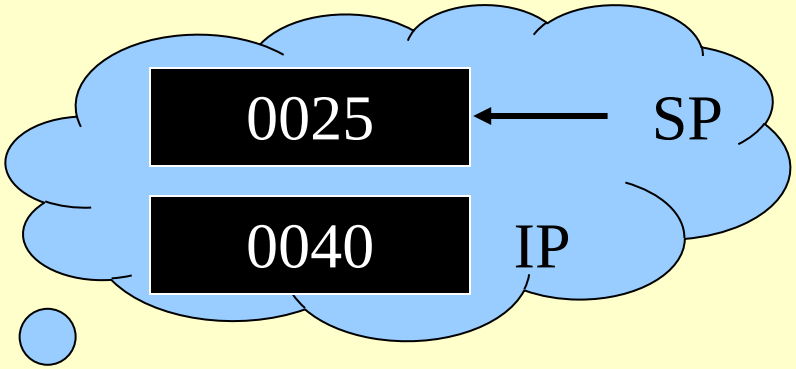
- RET

- Produce incarcarea valorii de pe stiva (SS:SP) in IP (intraseg. return) sau ambele IP si CS (interseg. ret)
- SP=SP+2/4

- RET operand ; SP=SP+operand

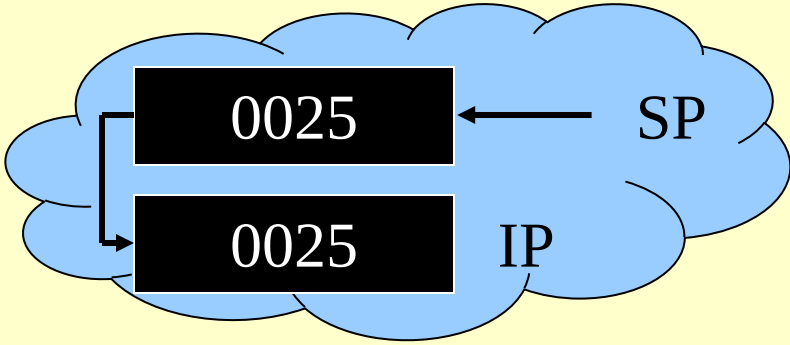
Proceduri

- *Exemplu*




```
main PROC
0020   Call MyProc
0025   MOV eax, ebx
       …

MyProc PROC
0040   MOV eax, edx
       …
RET
MyProc Endp
```



Proceduri

- ***Transmiterea Parametrilor***

- In registre – Cel mai rapid
- In memoria globala – Dificil de refolosit, rar utilizata
- Pe stiva – Utilizata de HLL

Proceduri

- *Exemple* - Calcularea sumei unui sir de intregi
  - Este recomandabil (mai bine) *transmiterea offset-ului sirului* procedurii decat includerea de referiri la anumite nume de variabile specifice in cadrul procedurii

```
.data
array  DWORD  10000h,20000h,30000h,40000h,50000h
theSum DWORD  ?
.code
main PROC
    mov   esi,OFFSET array        ; ESI points to array
    mov   ecx,LENGTHOF array      ; ECX = array count
    call  ArraySum                ; calculate the sum
    mov   theSum,eax              ; returned in EAX
```

```
;-----------------------------------------------------
ArraySum PROC
;
; Calculates the sum of an array of 32-bit integers.
; Receives: ESI = the array offset
;           ECX = number of elements in the array
; Returns:  EAX = sum of the array elements
;-----------------------------------------------------
   push esi                    ; save ESI, ECX
   push ecx
   mov  eax,0                  ; set the sum to zero

L1:
   add  eax,[esi]              ; add each integer to sum
   add  esi,4                  ; point to next integer
   loop L1                     ; repeat for array size

   pop  ecx                    ; restore ECX, ESI
   pop  esi
   ret                         ; sum is in EAX
ArraySum ENDP
```

Registrele care returneaza valori nu se salveaza/refac

Se salveaza si se refac registrele care sunt modificate de procedura

Proceduri

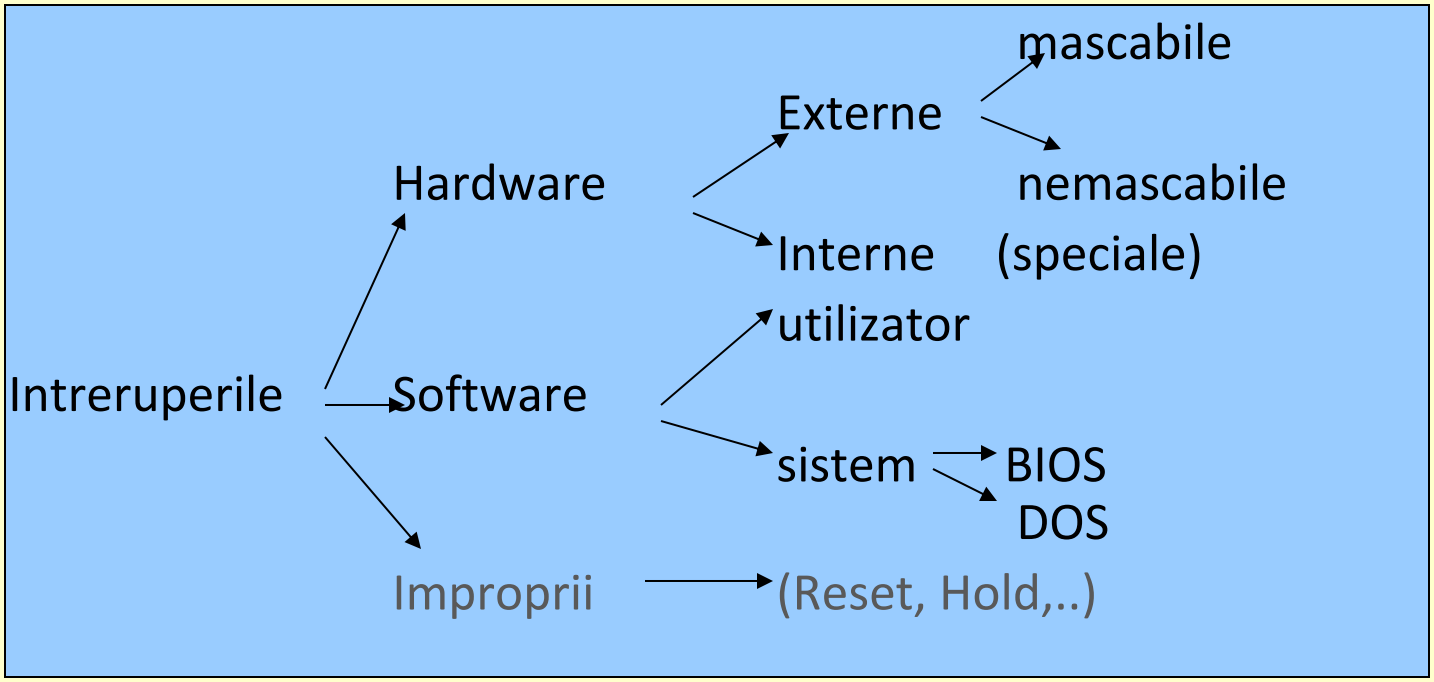
- ***Programarea utilizand Proceduri***

Pasi

- Intelegerea cerintelor programului
- Divizarea specificatiilor in task-uri
- Proiectarea fiecarui task ca sub-procedura
- Proiectarea procedurii main (procedura de start), astfel ca sa apeleze toate sub- procedurile

Intreruperi

- Mecanismul intreruperilor este des si indispensabil la utilizarea microprocesoarelor
- Prin intreruperi se poate sincroniza procesorul cu evenimente si procese aleatoare



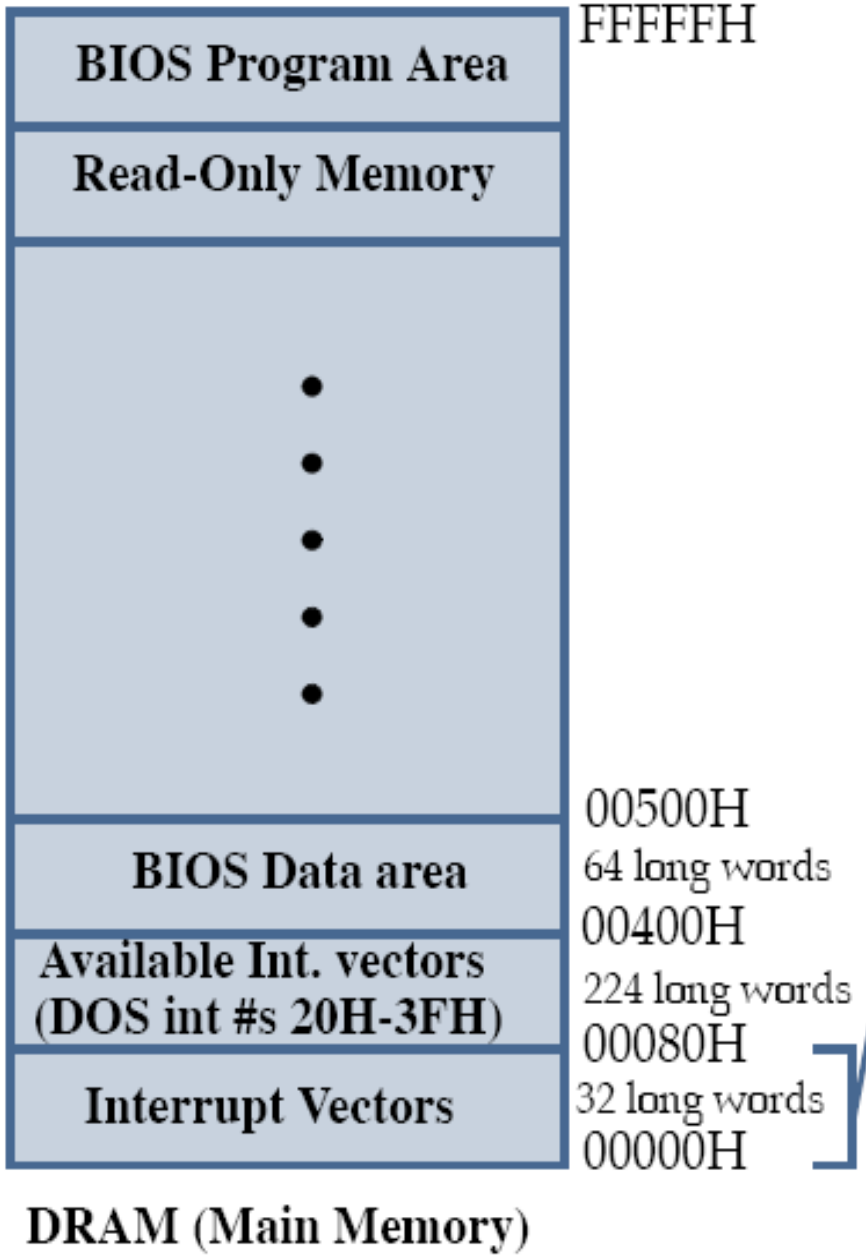
Intreruperi

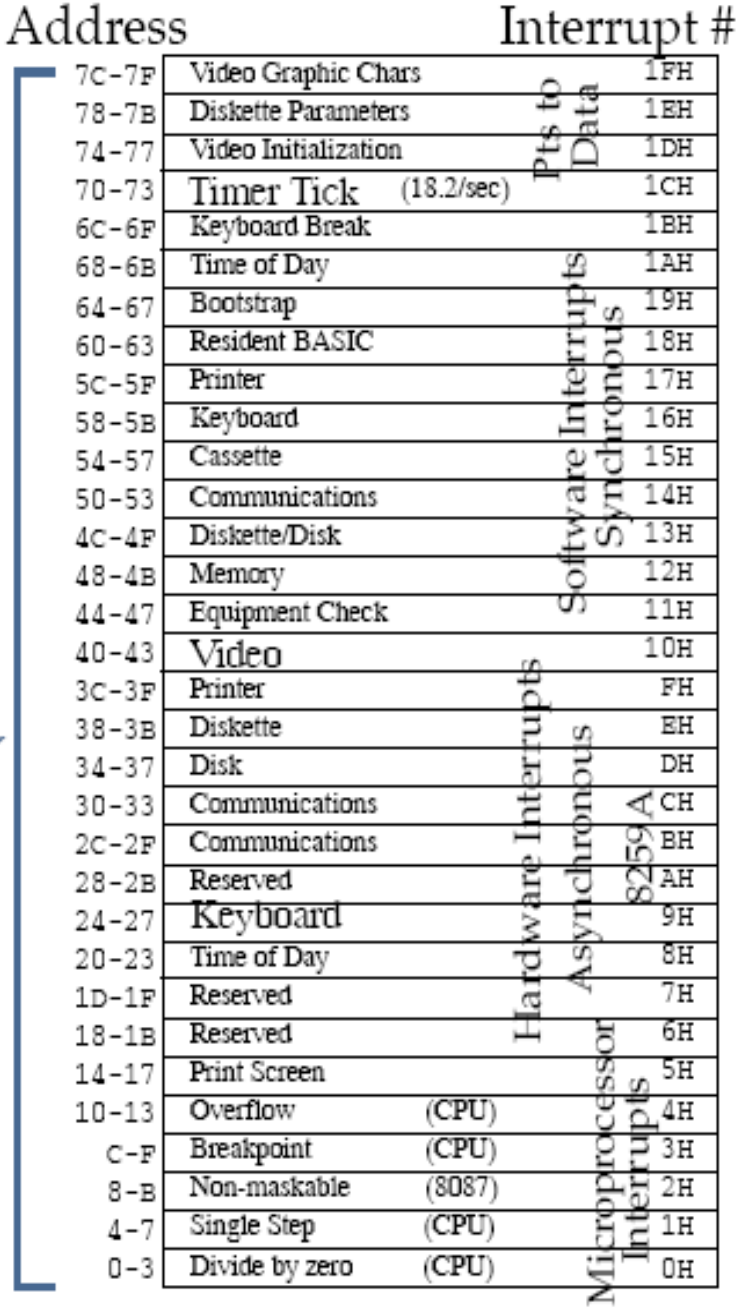
- Fiecarui eveniment “x” i se asociaza o rutina de tratare
- Ptr. o tratare sistematica intreruperile sunt vectorizate , fiecarei intreruperi >> tip (byte)
- In momentul aparitiei intreruperii procesorul trebuie sa primeasca tipul ptr. a sti modul in care sa o trateze
- Adresele rutinelor se afla intr-o tabela plasata la inceputul memoriei (0:0) TVI – tabela vectorilor de intreruperi
- Adresa rutinei (tip x) se afla la (CS=0: IP=4*tipx)
- Pentru asigurarea portabilitatii aplicatiilor si compatibiliatatea software exista un set de intreruperi sistem, predefinite care corespund anumitor tipuri >> se executa aceiasi functie chiar daca adresa din tabela difera
- Prioritatea intreruperilor depinde de tip – 0 este c.m.p.

Intreruperi

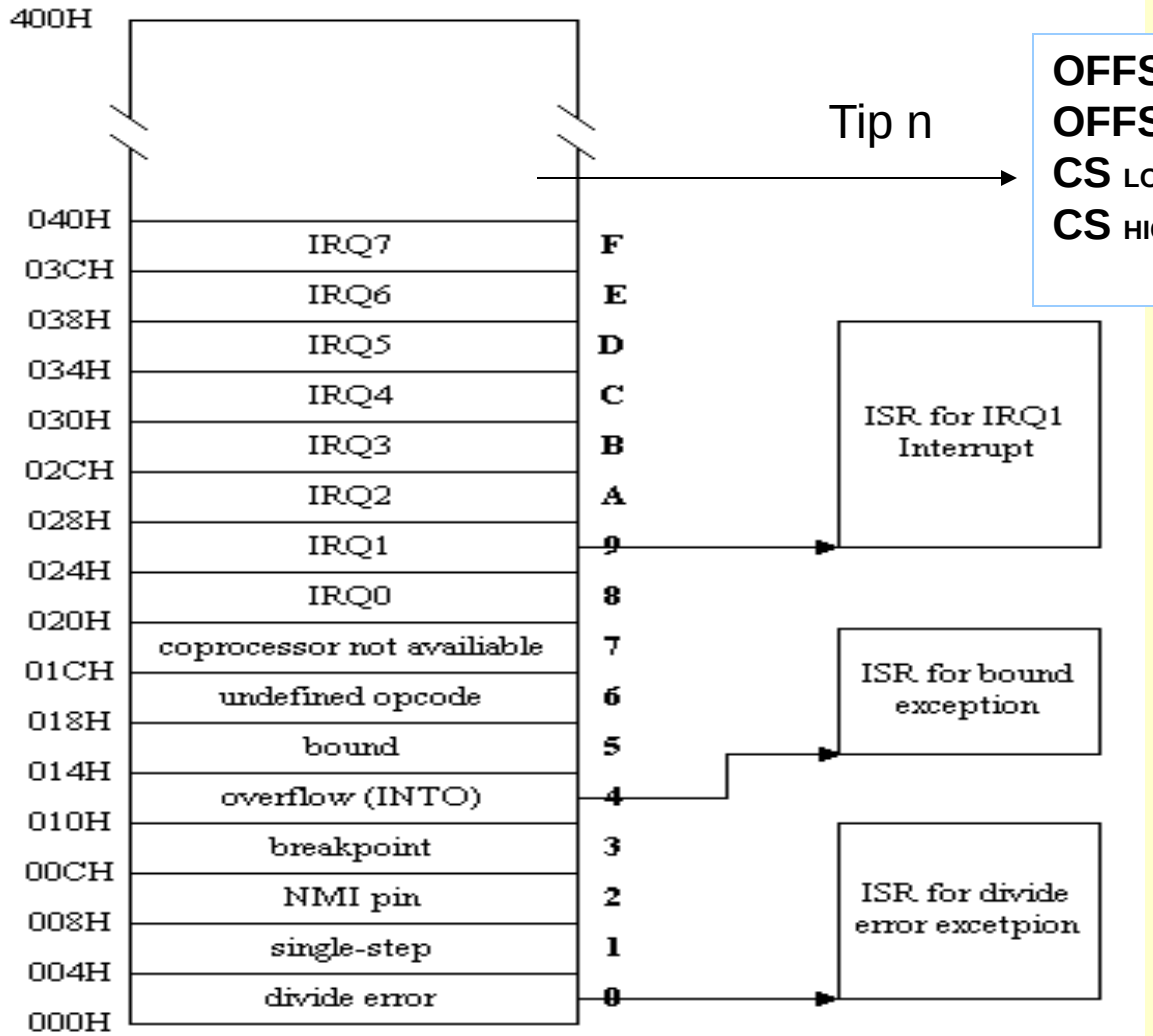
Interrupt	Address	Type	Description
00h	0000:0000h	Processor	Divide by zero
01h	0000:0004h	Processor	Single step
02h	0000:0008h	Processor	Non maskable interrupt (NMI)
03h	0000:000Ch	Processor	Breakpoint
04h	0000:0010h	Processor	Arithmetic overflow
05h	0000:0014h	Software	Print screen
06h	0000:0018h	Processor	Invalid op code
07h	0000:001Ch	Processor	Coprocessor not available
08h	0000:0020h	Hardware	System timer service routine
09h	0000:0024h	Hardware	Keyboard device service routine
0Ah	0000:0028h	Hardware	Cascade from 2nd PIC
0Bh	0000:002Ch	Hardware	Serial port service - COM port 2
0Ch	0000:0030h	Hardware	Serial port service - COM port 1
0Dh	0000:0034h	Hardware	Parallel printer service - LPT 2
0Eh	0000:0038h	Hardware	Floppy disk service
0Fh	0000:003Ch	Hardware	Parallel printer service - LPT 1
10h-1Fh	0000:0040h	Software	BIOS service routine
20h-3Fh	0000:0080f -	Software	DOS interrupts
40h-6Fh	0000:0100h -	Software	Not used
70h-77h	0000:01c0h -	Hardware	Real time clock
78h-FFh	0000:03FCh -	Software	Not used

Interrupt Vectors (DOS PC)



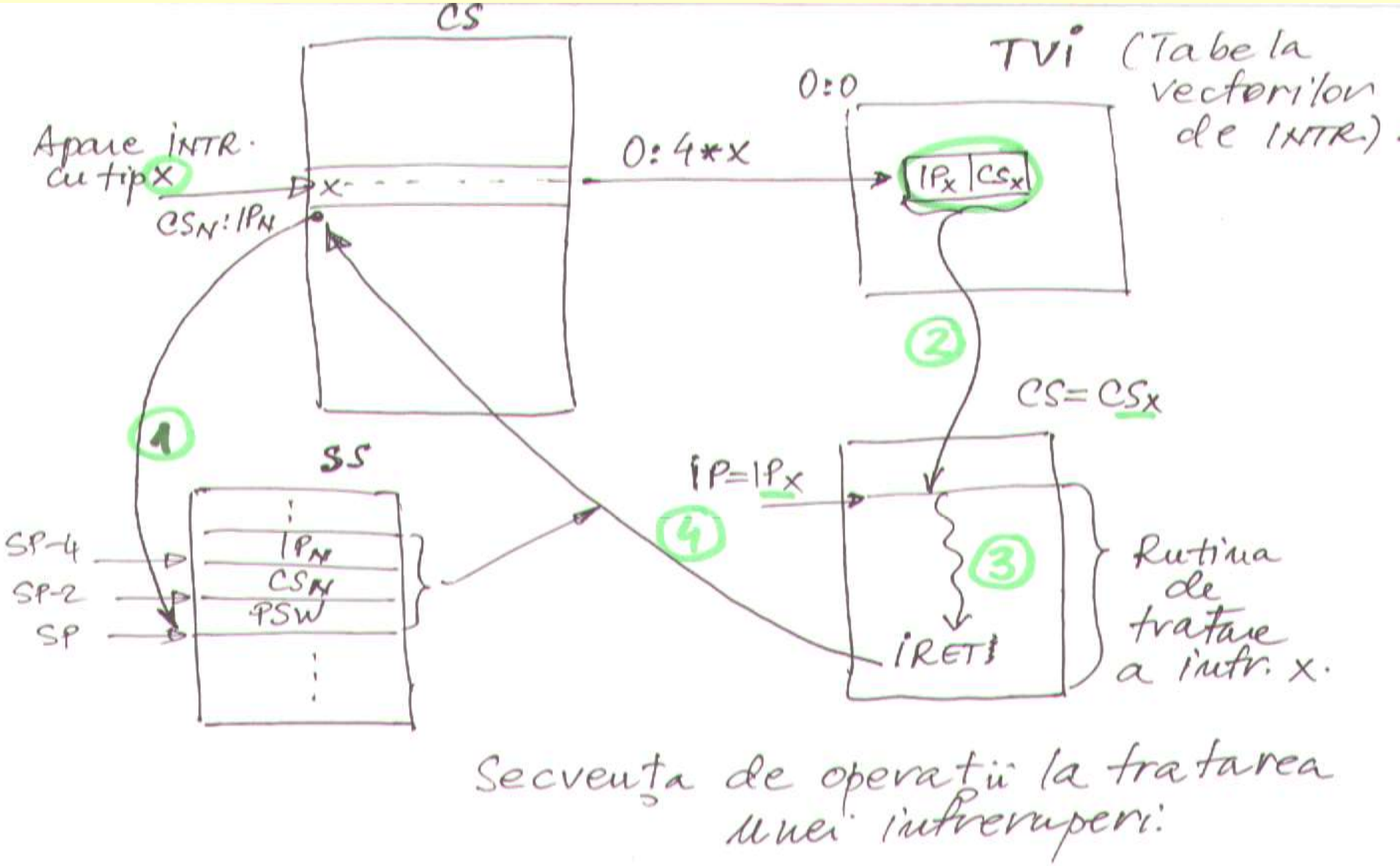


Intreruperi



Intel 80x86 Interrupt Vector Table in Real Mode

Intreruperi



Secvența de operații la tratarea unei întreruperi:

Intreruperi

Instructiuni specifice intreruperilor

INT tip

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

IRET

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

INT N

SP ← SP-2
[SP] ← PSW
I ← 0
SP ← SP-2
[SP] ← CS
SP ← SP-2
[SP] ← IP
IP ← [4*N, 4*N+1]
CS ← [4*N+2, 4*N+3]

IRET

IP ← [SP]
SP ← SP+2
CS ← [SP]
SP ← SP+2
PSW ← [SP]
SP ← SP+2