Instrukce CALL

Skok do podprogramu (=volání procedury)

CALL u (u je adresa začátku podprogramu)

Př

deklarace podprogramu:

```
MAX     PROC              ;max (BX,CX) → AX
max1:   MOV     AX,BX
        CMP     AX,CX
        JC      jinak
        RET               ;návrat z podprogramu
jinak:  MOV     AX,CX
        RET               ;návrat z podprogramu
MAX     ENDP              ;konec zápisu
                          ;podprogramu
```

volání podprogramu:

```
        MOV     BX,data1  ;zadání 1. parametru
        MOV     CX,data2  ;zadání 2. parametru
        CALL    MAX       ;volání podprogramu
        MOV     vysl,AX   ;uložení výsledku
```


CALL

Deklarace a volání podle umístění podprogramu vzhledem k místu volání (viz nepodmíněný skok):

	uvnitř segmentu	mezi segmenty
deklarace	PROC NEAR ENDP	PROC FAR ENDP
volání délka instr. zásobník	CALL NEAR PTR 3 B "PUSH IP"	CALL FAR PTR 5 B PUSH CS "PUSH IP"
návrat zásobník	RET "POP IP"	RET "POP IP" "POP CS"

Předávání parametrů: (viz "nahrazení formálních parametrů skutečnými" ve vyšších programovacích jazycích) zajišťuje programátor prostřednictvím:

- registrů
- paměti
- zásobníku

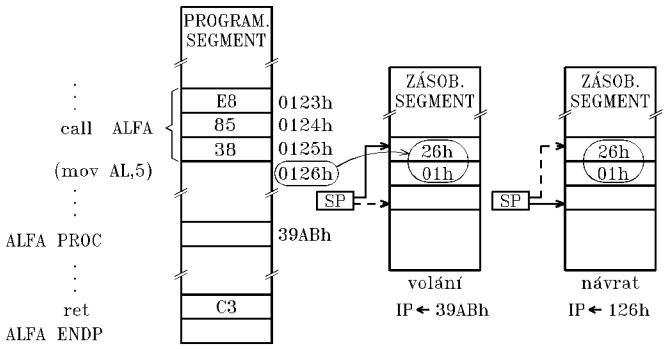
Př.1 (uvnitř segmentu)

Nelze volat z jiného segmentu!

```
39AB    alfa    PROC NEAR   ;pseudoinstrukce
                ...
                RET         ;stroj. kód - C3
        alfa    ENDP        ;pseudoinstrukce
```

0123 CALL NEAR PTR alfa

0126 další instrukce (např. MOV AL,5)

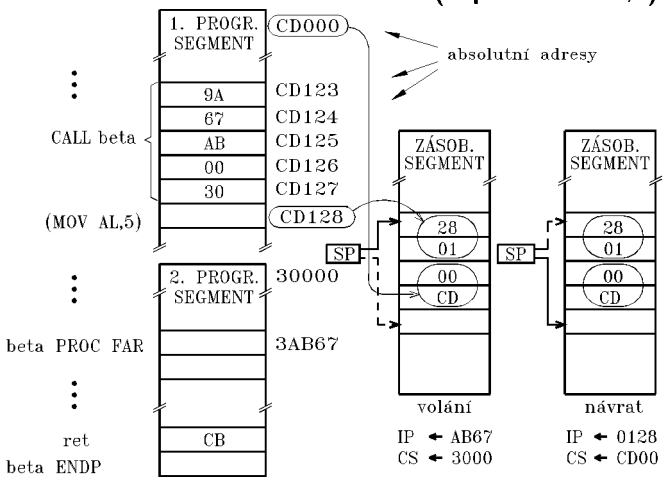


Pozn.: 3885h + 0126h = 39ABh

Př.2 (deklarace je v jiném segmentu než volání)

Při volání vždy použít FAR - i ve stejném segm.!

```
3000:AB67   beta    PROC FAR    ;pseudoinstrukce
                    ...
                    RET         ;stroj. kód - CB
            beta    ENDP        ;pseudoinstrukce
CD00:0123           CALL FAR PTR beta
     0128           další instr. (např. MOV AL,5)
```



Instrukce IN, OUT

obsluha vstupů a výstupů

vstup:

IN Ac, brána Ac ← brána

IN Ac, DX Ac ← [DX]

výstup:

OUT brána, Ac brána ← Ac

OUT DX, Ac [DX] ← Ac

kde:

Ac - AX nebo AL (tzn. 16 nebo 8 bitů)

brána - konstanta z intervalu ⟨ 0, 0FFh ⟩

DX - 16 bitů ⟨ 0, 0FFFFh ⟩

brány [ports] = logické obvody (registry) + pravidla

adresy těchto registrů tvoří nezávislý datový prostor (=vstupní/výstupní adresový prostor), oddělený od paměti pro programy a data, který je přímo adresovatelný (bez segmentových registrů), přístupný pomocí instrukcí IN a OUT

PŘERUŠENÍ

způsobí, že procesor přestane (dočasně) provádět právě probíhající program a místo něj začne provádět jiný program, který přerušení tzv. obslouží (tj. reaguje na jev, který přerušení vyvolal)

vnější - periferie, uživatel, havarijní stavy, ...

- nemaskovatelné vstup NMI
- maskovatelné (z řadiče přerušení) INTR

vnitřní

- chyby operandů, výsledku, krokování, ...,
- instrukce INT n (n je 8bitová konstanta)

- před obsluhou přerušení se uloží na zásobník informace o tom, jaký program se právě prováděl (FLAGS, CS, IP)
- zakáže se další přerušení (CLI)
- zjistí se, jak daný typ přerušení obsloužit - nastaví se nové CS a IP

- při návratu z přerušení je třeba obnovit informace o původním programu (instrukce IRET - ze zásobníku se vyzvednou IP, CS, FLAGS)

maskování přerušení - zákaz/povolení přerušení pomocí příznaku IF (jen maskovatelná přerušení)

instrukce: STI povolení přerušení IF:=1

CLI zákaz přerušení IF:=0

vektor přerušení - dvojité slovo (32 bitů) obsahující nové CS a IP (určení začátku podprogramu pro zpracování přerušení)

adresa vektoru přerušení se získá vynásobením tzv. typu (čísla) přerušení 4:

Př. INT 10h → na adresách 00040h až 00043h je uložen IP a CS pro zpracování tohoto typu přerušení

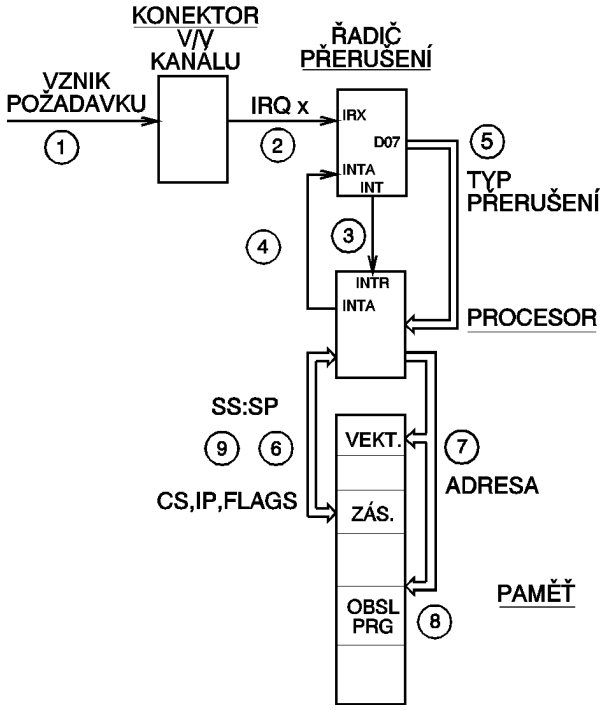
V paměti na nejnižších adresách je vyhražen prostor 0 ÷ 3FFh, kde je 256 vektorů - 32bitových adres podprogramů pro zpracování daného typu přerušení.

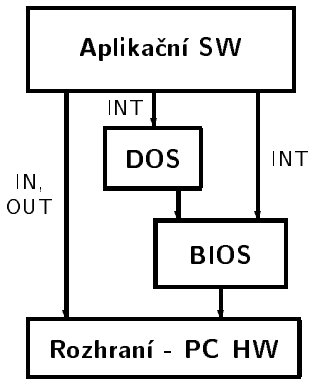
řadič přerušení - logický obvod (IO 8259A), který přijímá signály od V/V identifikuje požadavky na přerušení podle jejich priority [interrupt request - IRQ] generuje přerušovací signál (INT → INTR)

POSLOUPNOST ČINNOSTÍ PŘI OBSLUZE ŽÁDOSTI O VNĚJŠÍ PŘERUŠENÍ U PC (viz UPS4-9)

HW	1-3	vznik žádosti o přerušení
	4	rozhodnutí o obsluze (je-li IF=1 a INTA)
	5	identifikace příčiny přerušení (podle čísla - typu)
	6	uložení stavové informace (FLAGS), segmentového registru (CS) a čítače instrukcí (IP) na zásobník
	7	nalezení adresy začátku programu pro obsluhu daného typu přerušení pomocí vektoru přerušení - nové CS, IP
SW	8	provedení programu obsluhy přerušení
	9	návrat do přerušeného programu (IRET) obnovení IP, CS, FLAGS ze zásobníku

POSLOUPNOST ČINNOSTÍ PŘI OBSLUZE ŽÁDOSTI O PŘERUŠENÍ U PC





BIOS - v paměti ROM, nejnižší úroveň SW v PC (Basic Input/Output System)
(obsluha obrazovky, diskových jednotek, tiskáren, klávesnice, získání informací o konfiguraci systému, velikosti paměti, ...)

DOS - v paměti RAM po natažení s disku (diskety) (Disk Operating System)

Přerušení způsobené instrukcí INT:

V podstatě volání podprogramů, tzn. včetně předávání parametrů (před instrukcí INT) - zde přes registry

Př.1.

využití služeb BIOSu (Basic Input/Output System) pro zobrazení znaku na obrazovku na okamžitou pozici kurzoru

INT 10h

```
MOV   AH, 09h      ;číslo funkce do reg. AH
MOV   AL, znak     ;zobrazovaný znak do reg. AL
MOV   BH, strana   ;číslo stránky do reg. BH
MOV   BL, atrib    ;atribut do reg. BL
MOV   CX, opak     ;kolikrát má byt znak opakován
INT   10h          ;volání BIOSu
```

Poznámka:
zobrazení jednoho znaku (pro zobrazení více znaků je třeba použít cyklu), neposouvá se kurzor

Př.2.

Využití služeb jádra operačního systému (volání funkcí jádra DOSu) pro zobrazení řetězce znaků na obrazovku

INT 21h

```
text  DB     'toto je text na obrazovce$'
      .
      .
      MOV    DX,offset text   ;adresa řetězce
      MOV    AH, 09           ;číslo funkce
      INT    21h              přerušení DOS
```

Poznámka:
řetězec musí být ukončen symbolem $

Rozdíl INT 10h a INT 21h ?!
(mohu/nemohu, nebo musím/nemusím ovládat vše)