Az utasítások szerkezete

prefixum	*operációs kód*	*címzési mód*	*operandus*
0 - 2 byte	1 byte	0 - 1 byte	0 - 4 byte

***Prefixum*:**
utasítás ismétlés, explicit szegmens megadás vagy **LOCK**

```
MOV     AX, CS:S      ; S nem a DS,
                      ; hanem a CS regiszterrel címzendő
```

***Operációs kód*:** szimbolikus alakját mnemonic-nak nevezzük
***Címzési mód byte*:** hogyan kell az operandust értelmezni
***Operandus*:** mivel kell a műveletet elvégezni

Címzési mód byte

A legtöbb utasítás kód után szerepel. Szerkezete:

7	6	5	4	3	2	1	0
Mód		Regiszter			Reg/Mem		

Ha a műveleti kód legalacsonyabb helyértékű bit-je
0, akkor **byte**-os művelet,
1, akkor **word**-ös (szavas) művelet.

Regiszter			**Reg/Mem jelentése, ha Mód =**			
	byte	word	00	01	10	11
000	AL	AX	BX + SI + DI	„00” + 8 bit displ.	„00” + 16 bit displ.	Regiszter
001	CL	CX				
010	DL	DX	BP + SI + DI			
011	BL	BX				
100	AH	SP	SI DI			
101	CH	BP				
110	DH	SI	16 bit d.	BP+8 bit d.	BP+16 bit d.	
111	BH	DI	BX	„00”+8 bit	„00”+16 bit	

Szimbolikus alakban az operandusok sorrendje, gépi utasítás formájában a gépi utasítás kód mondja meg a regiszter és a memória közti adatátvitel irányát. Pl. az alábbi két utasítás esetén a címzési mód byte megegyezik:

```
MOV     AX, 122H[SI+BX]
        ; hexadecimálisan 8B 80 0122
MOV     122H[SI+BX], AX
        ; hexadecimálisan  89 80 0122
```

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Mód		**Reriszter**			**Reg/Mem**		
+16 bit d.		**AX**			**SI+BX**		

Az általános regiszterek és **SI**, **DI**, **SP**, **BP** korlátlanul használható, a többi (a szegmens regiszterek, **IP** és **STATUS**) csak speciális utasításokkal. Pl.:

```
MOV     DS, ADAT    ; hibás!

MOV     AX, ADAT    ; helyes!
MOV     DS, AX      ; helyes!
```

A „többi” regiszter nem lehet aritmetikai utasítás operandusa, sőt, **IP** és **CS** csak vezérlés átadó utasításokkal módosítható, közvetlenül nem is olvasható.


```
; Assembly főprogram, amely adott szöveget ír a képernyőre
; ========================================================
KOD  SEGMENT  PARA  PUBLIC  'CODE' ; Szegmens kezdet
; KOD: a szegmens neve
; align-type (igazítás típusa): BYTE, WORD, PARA, PAGE
; combine-type: PUBLIC, COMMON, AT <kifejezés>, STACK
; class: 'CODE', 'DATA', ('CONSTANT',) 'STACK', 'MEMORY'
;                ajánlott értelemszerűen
     ASSUME  CS:KOD, DS:ADAT, SS:VEREM, ES:NOTHING
; feltételezett szegmens regiszter értékek.
; A beállításról ez az utasítás nem gondoskodik!
```

```
KIIR    PROC    FAR ; A fő eljárás mindig FAR
                    ; FAR: távoli, NEAR: közeli eljárás
; Az operációs rendszer úgy hívja meg a főprogramokat, hogy
; a CS és IP a program végén lévő END utasításban megadott
; címke szegmens és OFFSET címét tartalmazza, SS és SP a
; a STACK kombinációs típusú szegmens végét mutatja,
; a visszatérés szegmens címe DS-ben van, OFFSET-je pedig 0
        PUSH    DS      ; DS-ben van a visszatérési cím
                        ; SEGMENT része
        XOR     AX, AX  ; AX⇐0, az OFFSET rész = 0
        PUSH    AX      ; Veremben a (FAR) visszatérési cím
        MOV     AX, ADAT ; AX⇐ az ADAT SEGMENT címe
        MOV     DS, AX
; Most már teljesül, amit az ASSUME utasításban írtunk
; Eddig tartott a főprogram előkészületi része
```



```
        MOV     SI, OFFSET SZOVEG
                        ; SI⇐SZÖVEG OFFSET címe
        CLD             ; a SZÖVEGet növekvő címek
                        ; szerint kell olvasni
        CALL    KIIRO   ; Eljárás hívás

        RET             ; Visszatérés az op. rendszerhez
                        ; a veremből visszaolvasott
                        ; szegmens és OFFSET címre

KIIR    ENDP            ; A KIIR eljárás vége
```



```
KIIRO   PROC            ; NEAR eljárás,
                        ; megadása nem kötelező
CIKLUS: LODSB           ; AL⇐a következő karakter
        CMP     AL, 0   ; AL =? 0
        JE      VEGE    ; ugrás a VEGE címkéhez,
                        ; ha AL=0
        MOV     AH, 14  ; BIOS rutin paraméterezése
        INT     10H     ; a 10-es interrupt hívása:
                        ; az AL-ben lévő karaktert kiírja
                        ; a képernyőre
        JMP     CIKLUS  ; ugrás a CIKLUS címkéhez,
                        ; a kiírás folytatása
VEGE:   RET             ; Visszatérés a hívó programhoz
KIIRO   ENDP            ; A KIIRO eljárás vége
KOD     ENDS            ; A KOD szegmens vége
```



```
ADAT    SEGMENT PARA PUBLIC 'DATA'
SZOVEG  DB      'Ezt a szöveget kiírja a képernyőre'
        DB      13, 10, 0   ; 13: a kocsi vissza,
                            ; 10: a soremelés kódja,
                            ;  0: a szöveg vége jel
ADAT    ENDS                ; Az ADAT szegmens vége
; ================================================
VEREM   SEGMENT PARA STACK
        DW      100 DUP (?)     ; Helyfoglalás 100 db
                                ; inicializálatlan szó számára
VEREM   ENDS                    ; A VEREM szegmens vége
; ================================================
        END     KIIR        ; Modul vége,
                            ; a program kezdőcíme: KIIR
```


Az I8086/8088 utasítás rendszere

Jelölések

⇐ : értékadás

⇔ : felcserélés

op, op1, op2: tetszőlegesen választható operandus (közvetlen, memória vagy regiszter).

op1 és **op2** közül az egyik regiszter kell legyen!

reg: általános, bázis vagy index regiszter

mem: memória operandus

ipr: (8 bites) **IP** relatív cím

port: port cím (8 bites eltolás vagy **DX**)

[op]: az **op** által mutatott cím tartalma

Adat mozgató utasítások

Nem módosítják a flag-eket (kivéve **POPF** és **SAHF**)

```
MOV   op1, op2   ; op1 ⇐ op2 (MOVe)
XCHG  op1, op2   ; op1 ⇔ op2 (eXCHanGe), op2 sem
                 ; lehet közvetlen operandus
XLAT             ; AL ⇐ [BX+AL] (trans(X)LATe), a
                 ; BX által címzett maximum 256 byte-
                 ; os tartomány AL-edik byte-jának
                 ; tartalma lesz AL új tartalma
LDS   reg, mem   ; reg ⇐ mem, mem+1
                 ; DS ⇐ mem+2, mem+3 (Load DS)
LES   reg, mem   ; reg ⇐ mem, mem+1
                 ; ES ⇐ mem+2, mem+3 (Load ES)
LEA   reg, mem   ; reg ⇐ mem effektív (logikai) címe
                 ; (Load Effective Address)
```

A veremmel (stack-kel) kapcsolatos adat mozgató utasítások:

```
PUSH    op   ; SP ⇐ SP-2; (SS:SP) ⇐ op
PUSHF        ; (PUSH Flags)
             ; SP ⇐ SP-2; (SS:SP) ⇐ STATUS
POP     op   ; op ⇐ (SS:SP); SP ⇐ SP+2
POPF         ; (POP Flags)
             ; STATUS ⇐ (SS:SP); SP ⇐ SP+2
```

Az Intel 8080-nal való kompatibilitást célozza az alábbi két utasítás:

```
SAHF         ; STATUS alsó 8 bitje ⇐ AH
LAHF         ; AH ⇐ STATUS alsó 8 bitje
```


Aritmetikai utasítások

```
ADD   op1, op2  ; op1 ⇐ op1 + op2   (ADD)
```

Pl.: előjeles/előjel nélküli számok összeadása

```
    MOV     AX, -1     ; AX=-1 (=0FFFFH)
    ADD     AX, 2      ; AX=1, C=1, O=0

ADC   op1, op2  ; op1 ⇐ op1 + op2 + C
                ; (ADD with Carry)
```

Pl.: két szavas összeadás (előjeles/előjel nélküli)

```
    ADD     AX, BX
    ADC     DX, CX   ; (DX:AX) = (DX:AX) + (CX:BX)

INC   op    ; op ⇐ op + 1, C változatlan! (INCrement)
```



```
SUB   op1, op2  ; op1 ⇐ op1 - op2 (SUBtraction)
CMP   op1, op2  ; flag-ek op1 - op2 szerint (CoMPare)
SBB   op1, op2  ; op1 ⇐ op1 - op2 – C:
                ; a több szavas kivonást segíti.
DEC   op        ; op ⇐ op - 1, C változatlan
                ; (DECrement)
NEG   op        ; op ⇐ -op (NEGate)
```