Utasítások

Fontosabb adat mozgató utasítások:

MOV	op1 , op2	(op1 <= op2)
XCHG	op1 , op2	(op1 <=> op2)
XFLAT		(AL <= [BX+AL])
PUSH	op	(SP <= SP-2 ; SS:SP <= op)
PUSHF		(SP <= SP-2 ; SS:SP <= STATUS)
POP	op	(op <= SS:SP ; SP <= SP+2)
POPF		(STATUS <= SS:SP ; SP <= SP+2)
LAHF		(AH <= STATUS alsó 8 bitje)
SAHF		(STATUS alsó 8 bitje <= AH)

Aritmetikai utasítások:

ADD	op1 , op2	(op1 <= op1+op2)
ADC	op1 , op2	(op1 <= op1+op2+C)
INC	op1	(op1 <= op1+1)
SUB	op1 , op2	(op1 <= op1-op2)
CMP	op1 , op2	(op1-op2 szerint állítja be a flag-eket)
SBB	op1 , op2	(op1 <= op1-op2-C)
DEC	op1	(op1 <= op1-1)
NEG	op1	(op1 <= -op1)
MUL	op1	(előjel nélküli: AX <= AL*op1 (8 bit); DX:AX <= AX*op1 (16 bit))
IMUL	op1	(előjeles: AX <= AL*op1 (8 bit); DX:AX <= AX*op1 (16 bit))
DIV	op1	(AL <= AX/op1 hányados ; AH <= AX/op1 maradék ; előjel nélküli)
		(AX <= DX:AX/op1 hányados ; DX <= DX:AX/op1 maradék)
IDIV	op1	(AL <= AX/op1 hányados ; AH <= AX/op1 maradék ; előjeles)
		(AX <= DX:AX/op1 hányados ; DX <= DX:AX/op1 maradék)

Vezérlés átadó utasítások:

CALL	cimke		(eljárás hívás ; közeli (NEAR): push IP , IP <= op
			távoli (FAR): push CS , push IP , CS:IP <= op)
RET			(visszatérés eljárásból ; közeli (NEAR): pop IP)
			távoli (FAR): pop IP , pop CS)
RET	cimke		(mint előbb, csak SP <= SP+op)
JMP	cimke		(feltétel nélküli ugrás)
JCXZ	cimke		(ugrás op-re, ha CX=0000h)
LOOP	cimke		(CX <= CX-1 ; ugrás cimkére, ha CX ≠ 0000h)
LOOPZ		cimke	(CX <= CX-1 ; ugrás cimkére, ha CX ≠ 0000h és Z=1)
LOOPE		cimke	(u.a. mint előbb)
LOOPNZ		cimke	(CX <= CX-1 ; ugrás cimkére, ha CX ≠ 0000h és Z=0)
LOOPNE		cimke	(u.a. mint előbb)

Bitenkénti logikai utasítások (Boole-műveletek)

AND	op1 , op2	(op1 <= op1 AND op2)
OR	op1 , op2	(op1 <= op1 OR op2)
XOR	op1 , op2	(op1 <= op1 XOR op2)
NOT	op1	(op1 <= NOT op1)
TEST	op1 , op2	(flag-eket op1 & op2 szerint)

Korrigáló (bináris-decimális):

AAA	(pakolatlan decimális számok összeadása , ASCII)
AAS	(két ASCII szám kivonása után korrigál)
AAD	(AL <= AH*10+AL ; AH <= 0)
AAM	(AH <= AL/10 ; AL <= maradék)
DAA	(pakolt decimális számok összeadása)
DAS	(két pakolt decimális szám kivonása után korrigál)

CBW (AX <= AL előjel helyesen)
CWD (DX:AX <= AX előjel helyesen)

Bitléptető utasítások
(utasítás regiszter , szam)

SHL – előjeltelen léptetés (shiftelés) balra
SAL – előjeles léptetés balra (ugyanaz, mint SHL)
SHR – előjeltelen léptetés jobbra
SAR – előjeles (aritmetikai) léptetés jobbra
ROL – balra forgatás (rotálás)
RCL – balra forgatás CF-en át
ROR – jobbra forgatás
RCR – jobbra forgatás CF-en át

Feltételes ugrás

Előjeles	Reláció	Előjel nélküli
JZ , JE	=	JZ , JE
JNZ , JNE	≠	JNZ , JNE
JG , JNLE	>	JA , JNBE
JGE	≥	JAE , JNB , JNC
JL , JNGE	<	JB , JNAE , JC
JLE , JNG	≤	JBE , JNA

n! kiszámítása(N=5)

```
        MOV  AX , 1
        MOV  CX , 5
CIMKE:  MUL  CX
        LOOP CIMKE
```

n! másképpen(dx:ax értéke a faktoriális)

```
        mov     cx, 4
        mov     ax, 1       ; dx:ax párosban gyűjtjük
        mov     dx, 0       ; az eredményt
cimke:
        mul     cx          ; DX:AX = AX * CX
        dec     cx          ; számláló csökken
        cmp     cx, 1       ; SR = CX - 1
        jne     cimke       ; ugrik ha nem egyenlő
```

n-ig összeadja a számokat

```
        MOV   AL, 0
        MOV   BL, 1

CIKLUS: CMP   BL, 9          // itt van megmondva, hogy az n=9
        JE    VEGE
        ADD   AL, BL
        INC   BL

VEGE:   MOV   AH, 14
        INT   10H
        RET
```

Üres Assembly program

```
KOD    SEGMENT   PARA PUBLIC 'CODE'
ASSUME    CS:KOD, DS:ADAT, SS:VEREM, ES:NOTHING

START:          PUSH    DS              ;visszatérés segmensének mentése
                XOR     AX, AX          ;offset mindig 0
                PUSH    AX              ;visszatérés offsetjének mentése

                MOV     AX, ADAT        ;állítsuk DS a sajat adatszegm.-re
                MOV     DS, AX          ;mert nincsen 'mov ds, adat'

        ;ide kerül az érdemi rész

VEGE:           RETF                    ;visszatérés a hívó programhoz
KOD             ENDS

;adat szegmens
ADAT    SEGMENT   PARA PUBLIC 'DATA'
ADAT    ENDS

;verem szegmens
VEREM   SEGMENT   PARA STACK
        DW      64 DUP (0)              ;helyfoglalás
VEREM   ENDS

END    START
```

Betű kiíratása

```
kod segment para public 'code'
assume cs:kod, ds:nothing, ss:verem, es:nothing
kiir    proc far
        push ds
        xor ax,ax
        push ax

        mov ah,14
        mov al,'k'
        int 10h

        ret

kiir    endp

kod     ends

verem   segment para stack
        dw 100 dup(5)
verem   ends
end     kiir
```

Sor kiíratása:

```
kod segment para public 'code'
assume cs:kod,ds:adat,ss:verem,es:nothing
kiir    proc far
        push ds
        xor ax,ax
        push ax
        mov ax,adat
        mov ds,ax


        cld
        mov si,offset szoveg_sor1
        call kiiro
        mov si,offset szoveg_sor2
        call kiiro

        ret

kiir    endp

kiiro   proc

ciklus: lodsb
        cmp al,0
        je vege
        mov ah,14
        int 10h
        jmp ciklus
vege:   ret

kiiro   endp
kod     ends

adat segment para public 'data'
        szoveg_sor1     db 'Ez az első sor',10,13
        szoveg_sor2     db 'Ez a második',10,13,0
adat ends

verem   segment para stack
        dw 100 dup(5)
verem   ends
end     kiir
```

Skalár szorzat kiszámítása

```
kod  segment para public 'CODE'
assume cs:kod, ds:adat, ss:verem, es:nothing
        mov     ax, adat
        mov     ds, ax

        mov     dx, 0                   ;részösszeg
        xor     ax, ax
        mov     al, offset v1
        mov     si, ax
        mov     al, offset v2
        mov     di, ax
        mov     cl, [len]
ciklus:
        mov     al, [si]                ;első vektor
        mul     byte ptr [di]           ;szorozva a másodikkal
        add     dx, ax                  ;hozzáadjuk a részösszeghez
        inc     si                      ;vektor köv. eleme
        inc     di
        dec     cx
        cmp     cx, 0
        jne     ciklus
kod  ends

;az adatszegmens
adat segment para public 'DATA'
        v1   db 1,2,3,4
        v2   db 2,4,3,1
        len  db 4
adat ends

;verem szegmens
verem segment para stack
        dw 100 dup(?)
verem ends
end
```

Legnagyobb karakter

```
kod segment para public 'code'
assume cs:kod,ds:nothing,ss:verem,es:nothing
kiir    proc far
        push ds
        xor ax,ax
        push ax


        mov ax,adat
        mov ds,ax


        cld
        mov si,offset szoveg_sor1
        call keres

        ret

kiir    endp

;-----------------------------------------------
keres   proc

        mov bl,48       ;minimalis karakter
ciklus: lodsb           ;karakter betoltese
        cmp al,0        ;ha vege a sztringnek
        je kiiro        ;ugras vegre

        cmp al,bl       ;ha bl kissebb mint al
        ja novel
        jmp ciklus

novel:  mov bl,al
        jmp ciklus

kiiro:  mov al,bl       ; a maximalis karakter
        mov ah,14
        int 10h
        ret



keres   endp
;-----------------------------------------------
kod     ends

adat segment para public 'data'
        szoveg_sor1     db 'Minden van benne',10,13
adat ends

verem   segment para stack
        dw 100 dup(5)
verem   ends
end     kiir
```