MÄLUMUDELID

Intel 8086 registrid (16-bitised):

Üldregistrid

AH	AL	AX - aritmeetika akkumulaator
BH	BL	BX – baas (indekseerimine)
CH	CL	CX – loendaja (indekseerimine)
DH	DL	DX - andmed

Segmentide adresseerimise reg-d

CS	viit programmi koodi segmendile
DS	viit andmesegmendile
SS	viit stacki segmendile
ES	viit ekstra (andme)segmendile

Eriregistrid

SP	stack pointer
BP	base pointer
SI	source index
DI	destination index

Flags register – info protsessori oleku kohta, näiteks:

S – sign
Z – zero
C – carry

Mälu adresseerimine

Adresseeritav mäluruum – 1M bait, selleks vajalik 20-bitine aadressisiin.

Igal ajahetkel jaguneb programmi poolt adresseeritav mälu neljaks 64K baidiseks mälusegmendiks. Segmentide algusaadressite 16 vanemat bitti paiknevad registrites CS, DS, SS ja ES.

Viitamine konkreetsele mälubaidile segmendis toimub kahe registri abil

segment register : offset register

näiteks DS : BX või SS : SP

20-bitine aadress arvutatakse nii:

VIITMUUTUJATE LIIGID

near 16-bitine aadress, võimaldab adresseerida 64K baiti

far 32-bitine viitmuutuja kujul *segment : offset*

NB! ühele ja samale baidile saab viidata paljude far-aadressitega!

Näiteks:

0x0000 : 0x0125	=>	0x00125L
0x0010 : 0x0025	=>	0x00125L
0x0001 : 0x0115	=>	0x00125L

NB! Võrdlusoperaatorite jaoks on kõik ülaltoodud far-väärtused erinevad!

Järeldus: far-pointerite kasutamisel tuleb vältida viitmuutujate aritmeetikat.

huge "normaliseeritud" far-pointer: *offset* on vahemikus 0000…000F

MÄLUMUDELID

Tiny **Small** **Medium** **Compact** **Large** **Huge**

Andmed	Programmi kood	
	64K	**1M**
64K	Tiny kõik kokku 64K Small 64K + 64K	Medium vähe andmeid, pikk kood
1M	Compact palju andmeid, lühike kood	Large Huge palju andmeid, pikk kood

KATKESTUSTE TÖÖTLEMINE

```
#include  <dos.h>
#define  port  0x61
#define  viide  1000
typedef  unsigned int uint;

void  interrupt  piiks( uint bp, uint di, uint si, uint ds, uint es,
                        uint dx, uint cx, uint bx, uint ax )
{       int k, j;
        char  olek, algolek;
        unsigned char  kordusi = ( ax >> 8 );          // register AH

        algolek = olek = inportb( port )               //  oleku lugemine
        for( k = 0; k < kordusi; k++ )
        {
                outportb( port,  olek & 0xFC )         // kahe madalama 0
                                                       //  = spiiker välja
                for( j = 0; j < viide; j++ )           // viitetsükkel
                        ;
                outportb( port, olek | 2 );            // spiiker sisse

                for( j = 0; j < viide; j++ )           // viitetsükkel
                        ;
        }
        outportb( port, algolek );                     // taastame oleku
}
```

```
int  main (void )
{
        setvect ( 0x0A, piiks );     // katkestust 10 töötleb piiks( )

        _AH = 4;                     // korduste arv

        geninterrupt( 0x0A );        // tekitada katkesus 10

        getch();
}
```