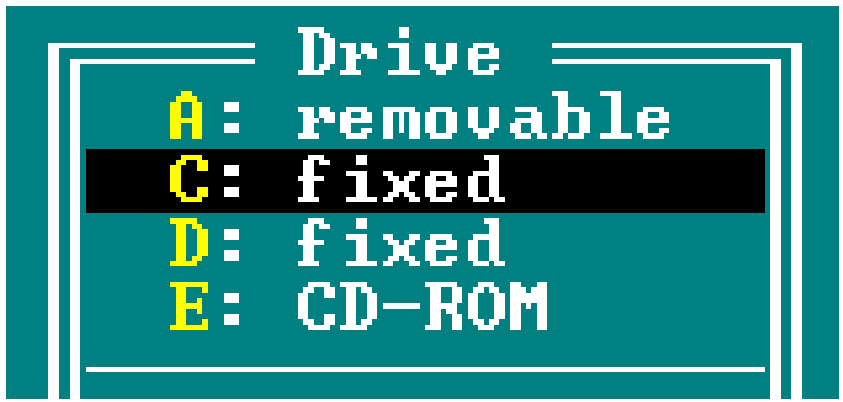
Darbs ar programmu *FAR Manager*

1. ***Alt*** + ***F1***, ***Alt*** + ***F2***: *nomainīt disku* kreisajā (labajā) panelī.

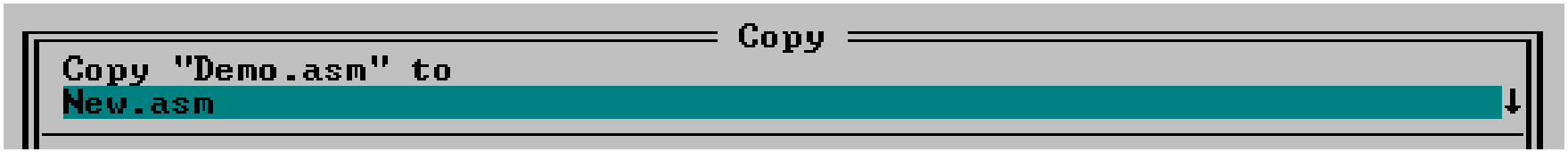


2. ***Shift*** + ***F4***: *izveidot* jaunu teksta failu.

Darbs ar programmu *FAR Manager* *(turpinājums)*

3. ***F4***: rediģēt failu.

4. ***Shift*** + ***F5***: *nokopēt* failu pašreizējā mapē.



5. ***Shift*** + ***F6***: *pārdēvēt* failu pašreizējā mapē.

6. ***F2***: *saglabāt* failu.

Iegultais asamblera kods (C++ programma)

```
unsigned int X, Y;
...
X = 1;
Y = 3;
cout << "X  : " << X << ", Y  : " <<
  Y << "." << endl;     //X  : 1, Y  : 3.
asm {
  Inc X             // X = X+1

  Mov Ax, 2         // Ax = 2
  Mul Y             // Dx:Ax := Ax*Y
  Mov Y, Ax         // Y := Ax
}
cout << "X+1: " << X << ", 2*Y: " <<
  Y << "." << endl; //X+1: 2, 2*Y: 6.
```

Iegultais asamblera kods (Pascal programma)

```
Var X, Y : Word;
...
X := 1;
Y := 3;
WriteLn('X  : ', X, ',
  Y  : ', Y, '.'); {X  : 1, Y  : 3.}
asm
  Inc X           {X := X+1}

  Mov Ax, 2       {Ax := 2}
  Mul Y           {Dx:Ax := Ax*Y}
  Mov Y, Ax       {Y := Ax}
end;
WriteLn('X+1: ', X, ',
  2*Y: ', Y, '.'); {X+1: 2, 2*Y: 6.}
```

Mašīnkoda ekvivalents (Pascal programma)

```
(*
    10 0106  FF 06 0102r     Inc X
    11 010A  B8 0002         Mov Ax, 2
    12 010D  F7 26 0104r     Mul Y
    13 0111  A3 0104r        Mov Y, Ax
*)

inline (
   $FF/$06/X/   {X := X+1}
   $B8/>0002/   {Ax := 2}
   $F7/$26/Y/   {Dx:Ax := Ax*Y}
   $A3/Y        {Y := Ax}
);
```

Elementārā programma (*Demo.asm*)

```
.model tiny   ;atmiņas modelis
.code         ;programmas kods
.startup      ;ieejas punkts
.exit 0       ;izejas punkts
end
```

Pēc programmas palaišanas *nekādu rezultātu nav (**Ctrl + O**)*.

Faila *Demo.lst* fragments:

```
1      6     0100   B8 4C00     MOV       AX,04C00h
1      7     0103   CD 21       INT       21h
```

Abas komandas aizvieto makrokomandu *.exit 0*.

Komandas formāts:

Iezīme:	Mnemonika	Operands (-i)	;Komentārs

Komandu *operandi* asamblerā:

- ✓ Reģistrs
- ✓ Atmiņa
- ✓ Tiešais operands
- ✓ Ports

Augsta līmeņa programmēšanas valodas (Pascal, C++):
Viena komanda => *vairākas* mašīnkoda komandas.

Asamblers:
Viena komanda => *viena* mašīnkoda komanda.

Asamblers **nav** pārnesama programmēšanas valoda.

Reģistri

1.*Vispārēja uzdevuma* reģistri

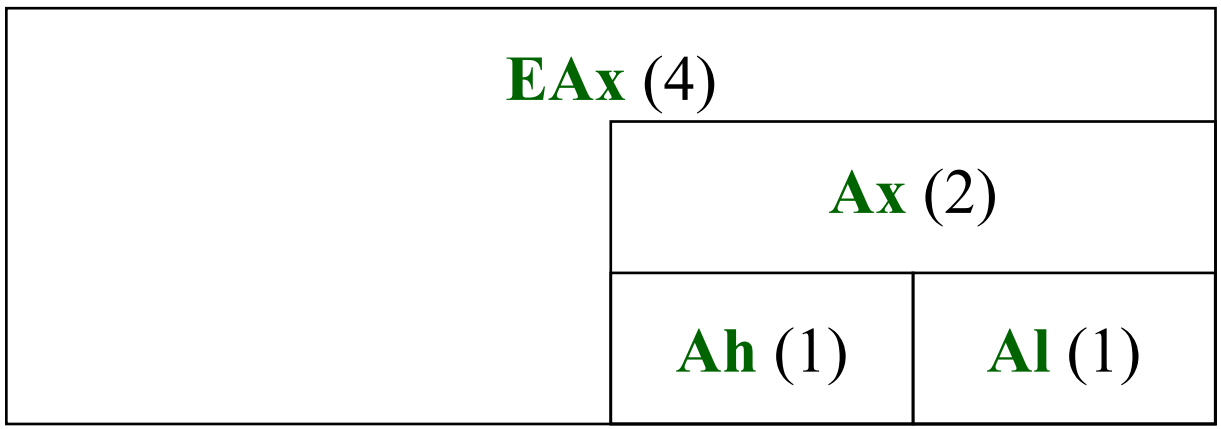
a. *Datu* reģistri (2 baiti)

Ax – Accumulator (akumulators)

Bx – Base (bāze)

Cx – Counter (skaitītājs)

Dx – Data (dati)



Reģistri *(turpinājums)*

b. *Indeksa* reģistri un *rādītāji* (2 baiti)

SI – Source Index (avota indekss)
DI – Destination Index (saņēmēja indekss)
BP – Base Pointer (bāzes rādītājs)
SP – Stack Pointer (steka rādītājs)

---

2. *Segmentu* reģistri (2 baiti)

CS – Code Segment (koda segments)
DS – Data Segment (datu segments)
SS – Stack Segment (steka segments)
ES – Extra Segment (ekstra segments)

Reģistri *(nobeigums)*

3. *Komandu rādītājs* (2 baiti)

IP - Instruction Pointer (komandu rādītājs)

---

4. *Karogu* reģistrs (1 baits)

1 karogs => 1 bits

```
CF - Carry Flag      (pārneses karogs)
OF - Overflow Flag   (pārpildes karogs)
ZF - Zero Flag       (nulles karogs)
SF - Sign Flag       (zīmes karogs)
...
```

Programma *Hello.asm*

```
.model tiny
.code
.startup
    Org 100h
    Jmp Short Start
    Hello DB "Hello, world!", '$'
Start:
    Lea Dx, Hello
    Mov Ah, 9H
    Int 21H
.exit 0
end
```

Programma *Hello.asm* (“vecais” stils)

```
MyCode  Segment Para
        Org 100h
Main    Proc
        Assume      CS:MyCode, DS:MyCode, SS:MyCode

        Jmp Short   Start
Hello   DB          "Hello, world!", '$'
Start:
        Lea         Dx, Hello
        Mov         Ah, 9H
        Int         21H

        Mov         Ax, 4C00h
        Int         21h
Main    EndP
MyCode  EndS
        End         Main
```

Daudzrindu komentārs:

```
Comment &
    Hello, world!
&
```

Elementārā aritmētika

5 + 10 – 3 = 12 ?

Programmas *Math.asm* fragments:

```
    Org 100h
    Mov Ax, 5
    Add Ax, 10
    Sub Ax, 3
.exit 0
```

Programmas *Math.com* atkļūdošana

1. Programma *Debug.exe*:

`Debug MATH.COM`

```
C:\Work>Debug MATH.COM
_
```

2. Trasēšana (kārtējas komandas izpilde): simbols **t** <Enter>

```
AX=0005  BX=0000  CX=000E  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=142D  ES=142D  SS=142D  CS=142D  IP=0103   NV UP EI PL NZ NA PO NC
142D:0103 050A00        ADD     AX,000A
-t

AX=000F  BX=0000  CX=000E  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=142D  ES=142D  SS=142D  CS=142D  IP=0106   NV UP EI PL NZ NA PE NC
142D:0106 2D0300        SUB     AX,0003
-t

AX=000C  BX=0000  CX=000E  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=142D  ES=142D  SS=142D  CS=142D  IP=0109   NV UP EI PL NZ NA PE NC
142D:0109 B8004C        MOV     AX,4C00
_
```

3. Izeja no programmas: simbols **q** <Enter>

```
AX=000C  BX=0000  CX=000E  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=142D  ES=142D  SS=142D  CS=142D  IP=0109   NV UP EI PL NZ NA PE NC
142D:0109 B8004C        MOV     AX,4C00
-q
```

---

5 + 10 – 24 = -9 ?

Izmaiņas programmā *Math.asm*:

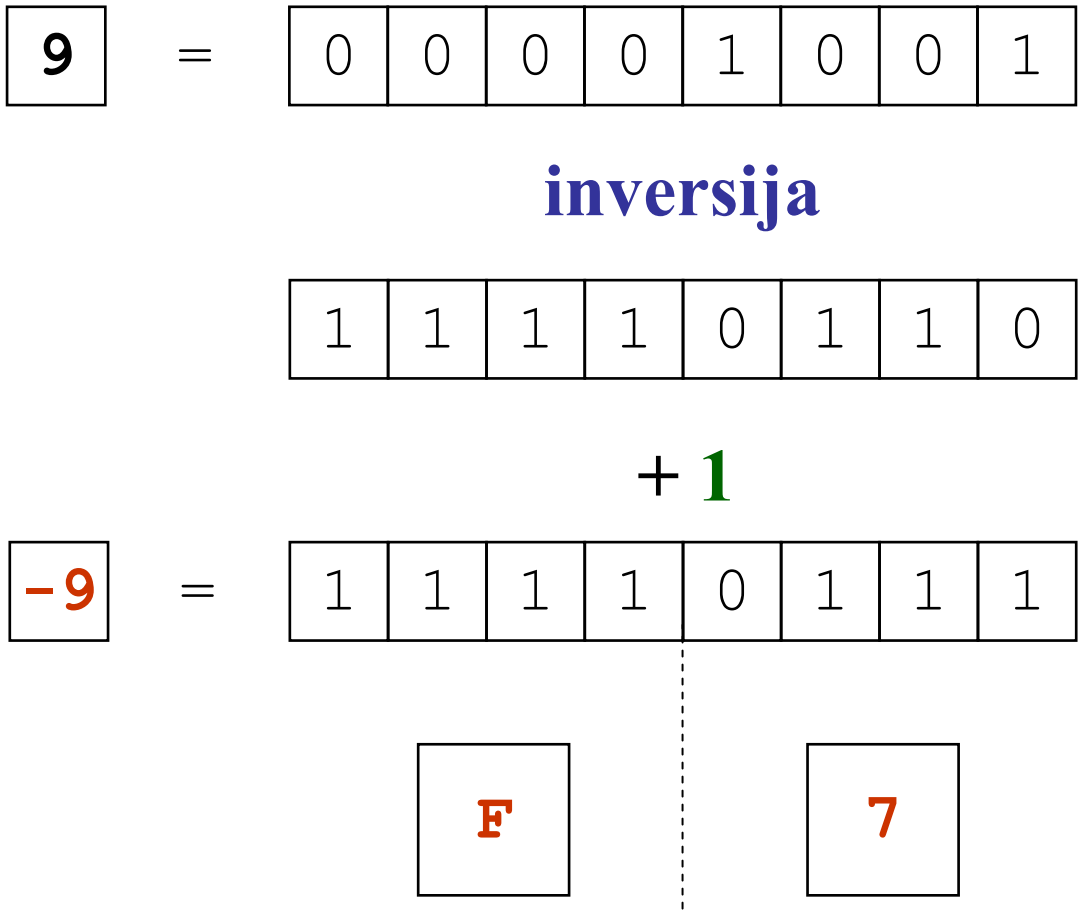
```
Sub Ax, 24
```

Trasēšana: **FFF7 = -9**

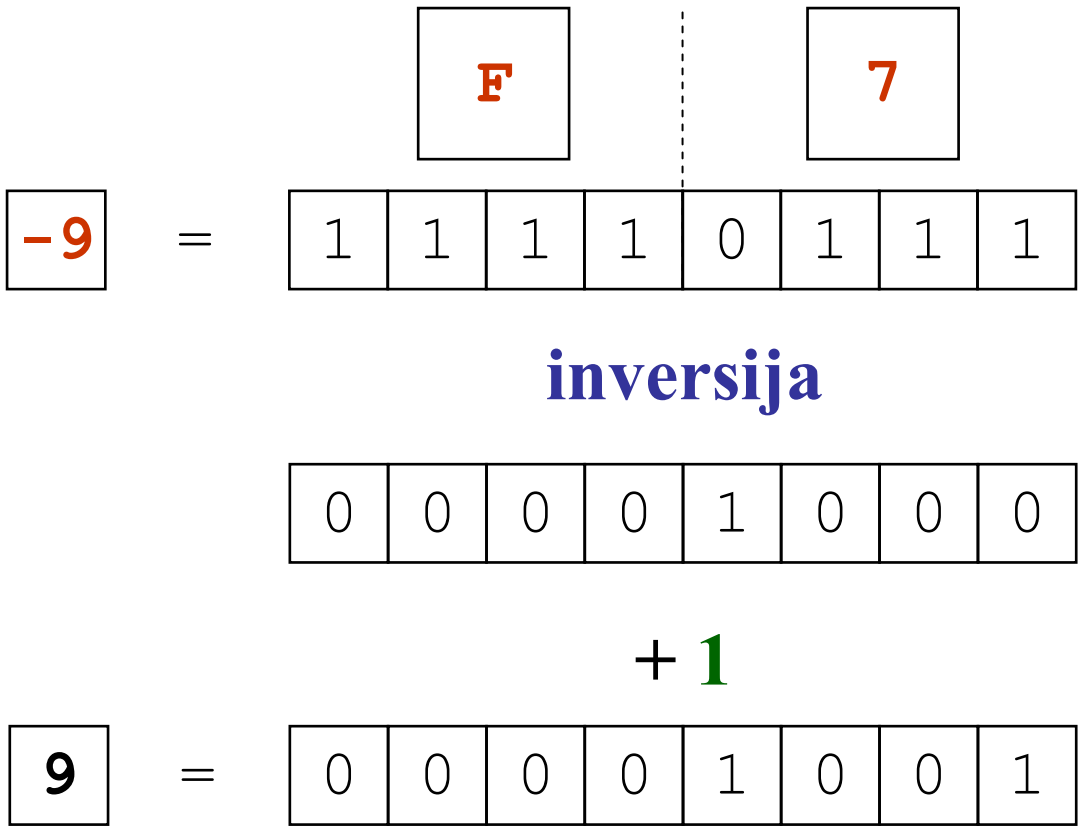
```
AX=000F  BX=0000  CX=000E  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=142D  ES=142D  SS=142D  CS=142D  IP=0106   NV UP EI PL NZ NA PE NC
142D:0106 2D1800        SUB     AX,0018
-t

AX=FFF7  BX=0000  CX=000E  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=142D  ES=142D  SS=142D  CS=142D  IP=0109   NV UP EI NG NZ NA PO CY
142D:0109 B8004C        MOV     AX,4C00
-
```

Papildus kods: inversija + 1



Pretējā operācija: inversija + 1



Darbs ar atkļūdotāju *td.exe:* Alt + F3

1. *F7*: trasēšana.

```
cs:0100►B80500          mov     ax,0005          ax 0000    c=0
cs:0103 050A00          add     ax,000A          bx 0000    z=0
cs:0106 2D1800          sub     ax,0018          cx 0000    s=0
cs:0109 B8004C          mov     ax,4C00          dx 0000    o=0
```

```
cs:0100 B80500          mov     ax,0005          ax 0005    c=0
cs:0103►050A00          add     ax,000A          bx 0000    z=0
cs:0106 2D1800          sub     ax,0018          cx 0000    s=0
cs:0109 B8004C          mov     ax,4C00          dx 0000    o=0
```

```
cs:0100 B80500          mov     ax,0005          ax 000F    c=0
cs:0103 050A00          add     ax,000A          bx 0000    z=0
cs:0106►2D1800          sub     ax,0018          cx 0000    s=0
cs:0109 B8004C          mov     ax,4C00          dx 0000    o=0
```

```
cs:0100 B80500          mov     ax,0005          ax FFF7    c=1
cs:0103 050A00          add     ax,000A          bx 0000    z=0
cs:0106 2D1800          sub     ax,0018          cx 0000    s=1
cs:0109►B8004C          mov     ax,4C00          dx 0000    o=0
```

2. ***Ctrl*** + ***F2***: "*Reset*" (atgriezties programmas sākumā).

---

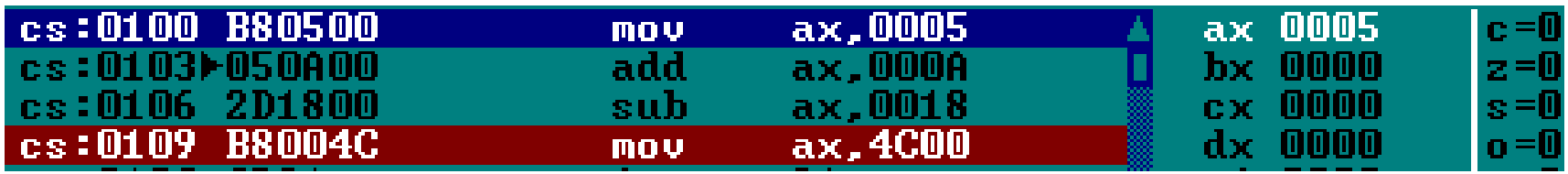
3. ***F2***: "*Breakpoint*" (pārtraukumpunkts).

---

4. ***F9***: "*Run to Breakpoint*" (pārtraukumpunkts).

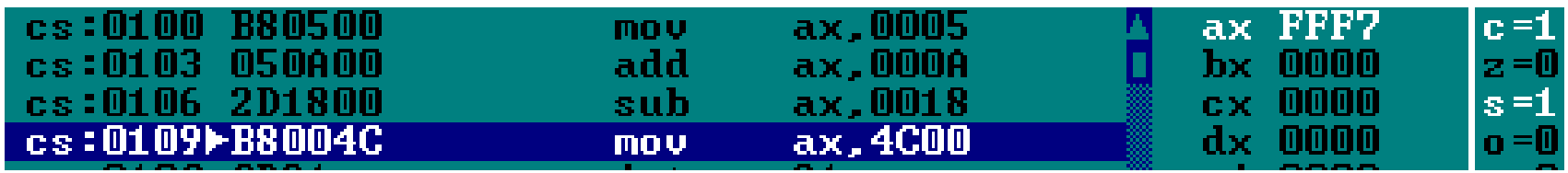
---

a. Rindiņā `mov ax,4C00` bija nospiests ***F2***.



b. ***Ctrl*** + ***F2***.

c. ***F9***.



Reizināšana

```
Mul  <reizināmais>
; Multiplication
; 1. operands atrodas reģistrā Al vai Ax
; bez zīmju reizināšana

IMul <reizināmais>
; sIgned Multiplication
; 1. operands atrodas reģistrā Al vai Ax
; zīmju reizināšana
```

---

1. <Baits> * <Baits> = <Vārds> **Ax**
2. <Vārds> * <Vārds> = <Dubultvārds> **Dx:Ax**

Reizināšanas piemērs *(Mul.asm)*

```
.model tiny
.code
.startup
    Org  100h
    Mov  Al, 5
    Mov  Bl, -2
    IMul Bl       ;Ax = FFF6 (-10)
; informācija reģistrā Ah bija pazaudēta
.exit 0
end
```

Dalīšana

```
Div  <dalītājs>
; Division
; Dalāmais atrodas reģistrā Ax vai Dx:Ax
; bez zīmju dalīšana

IDiv <dalītājs>
; sIgned Division
; Dalāmais atrodas reģistrā Ax vai Dx:Ax
; zīmju dalīšana
```

---

1. <Vārds> / <Baits> = {<Baits>, <Baits>} **Ah, Al**
2. <Dubultvārds> / <Vārds> = {<Vārds>, <Vārds>} **Dx, Ax**

Dalīšanas piemērs *(Div.asm)*

```
.model tiny
.code
.startup
    Org 100h
    Mov  Ax, 17
    Mov  Bl, -3
    IDiv Bl      ;Ax = 02FB (Ah = 02, Al = FB)
.exit 0
end
```