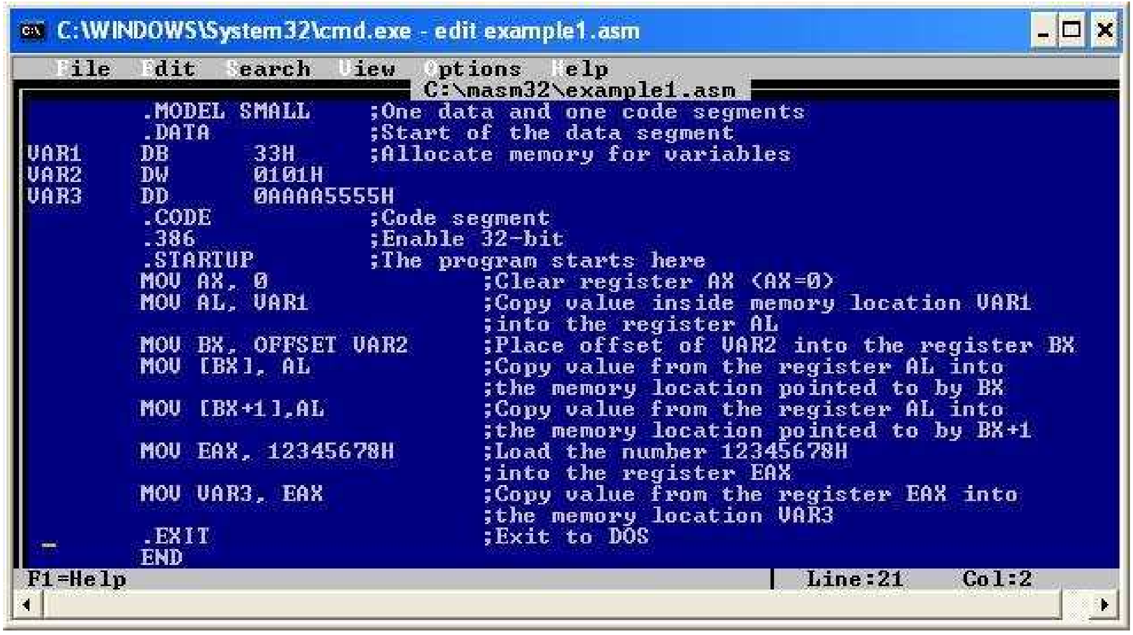
MASM Tutorial

Follow this tutorial step by step:

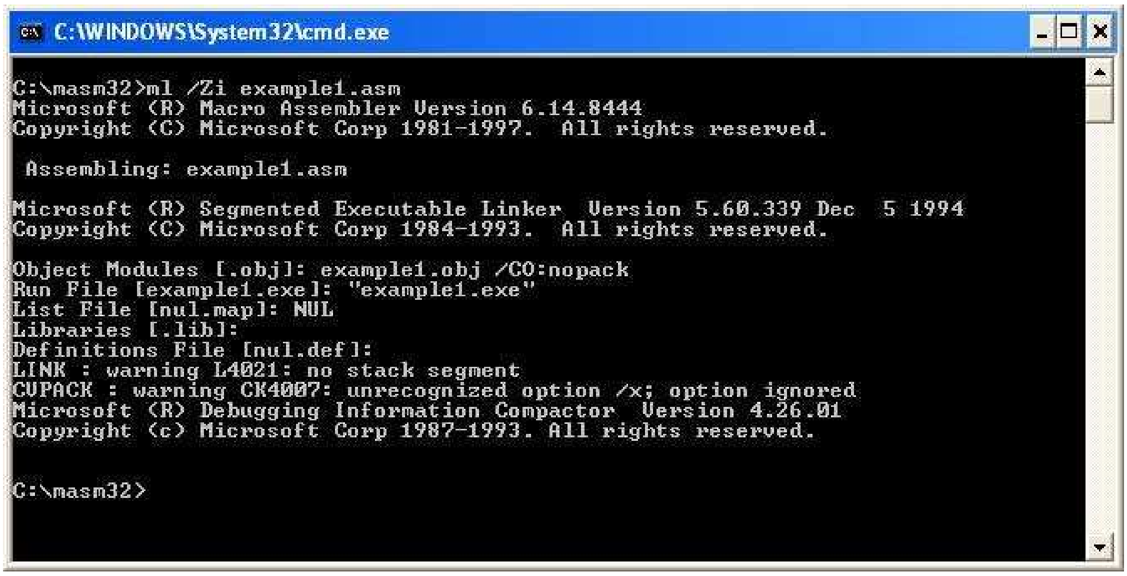
- You can use almost any text editor to create an assembly program. In this example, we will use Microsoft’s EDIT. Type “edit example1.asm” on the command prompt and enter the text of the program.

  Save the file by “Alt-F”,”Alt+S”. Exit “Alt-F”,“Alt-X”


C:\WINDOWS\System32\cmd.exe - edit example1.asm
File Edit Search View Options Help
C:\masm32\example1.asm
F1=Help
Line:21
Col:2

```
        .MODEL SMALL     ;One data and one code segments
        .DATA            ;Start of the data segment
VAR1    DB      33H      ;Allocate memory for variables
VAR2    DW      0101H
VAR3    DD      0AAAA5555H
        .CODE            ;Code segment
        .386             ;Enable 32-bit
        .STARTUP         ;The program starts here
        MOV AX, 0                ;Clear register AX (AX=0)
        MOV AL, VAR1             ;Copy value inside memory location VAR1
                                 ;into the register AL
        MOV BX, OFFSET VAR2      ;Place offset of VAR2 into the register BX
        MOV [BX], AL             ;Copy value from the register AL into
                                 ;the memory location pointed to by BX
        MOV [BX+1],AL            ;Copy value from the register AL into
                                 ;the memory location pointed to by BX+1
        MOV EAX, 12345678H       ;Load the number 12345678H
                                 ;into the register EAX
        MOV VAR3, EAX            ;Copy value from the register EAX into
                                 ;the memory location VAR3
        .EXIT                    ;Exit to DOS
        END
```

- Compile and link the assembly file by issuing “ ml /Zi example1.asm”

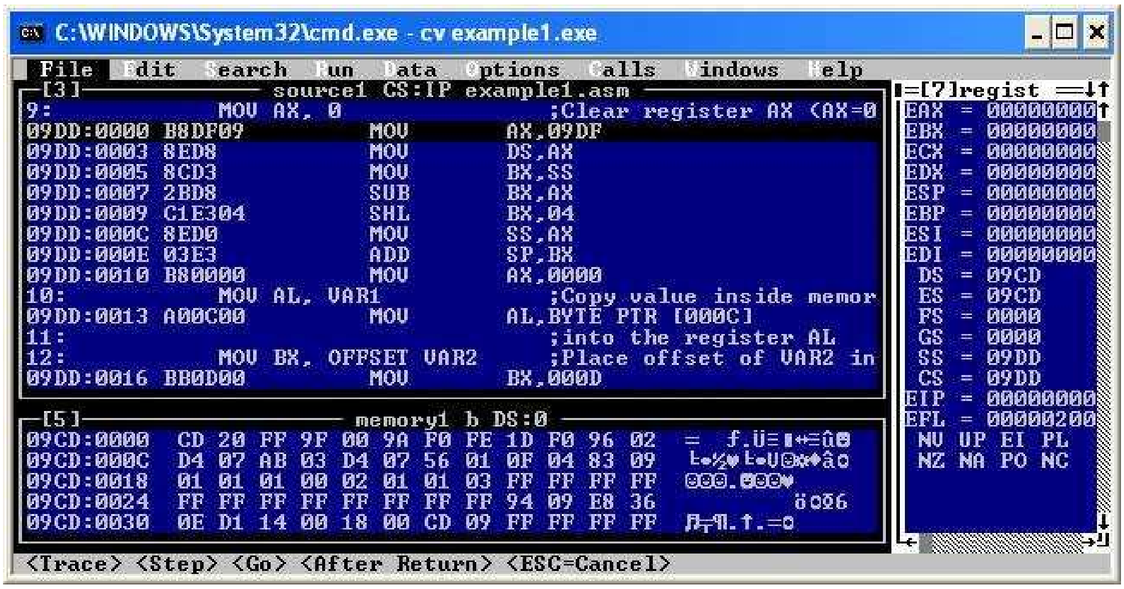


```
C:\masm32>ml /Zi example1.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997.  All rights reserved.

 Assembling: example1.asm

Microsoft (R) Segmented Executable Linker  Version 5.60.339 Dec  5 1994
Copyright (C) Microsoft Corp 1984-1993.  All rights reserved.

Object Modules [.obj]: example1.obj /CO:nopack
Run File [example1.exe]: "example1.exe"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:
LINK : warning L4021: no stack segment
CVPACK : warning CK4007: unrecognized option /x; option ignored
Microsoft (R) Debugging Information Compactor  Version 4.26.01
Copyright (c) Microsoft Corp 1987-1993. All rights reserved.


C:\masm32>
```

- Now let’ s start and configure the Code View debugger.
  - o Type “ cv example1.exe’ at the command prompt.Enter “ Alt-W” and make sure that you have the following windows on the screen:

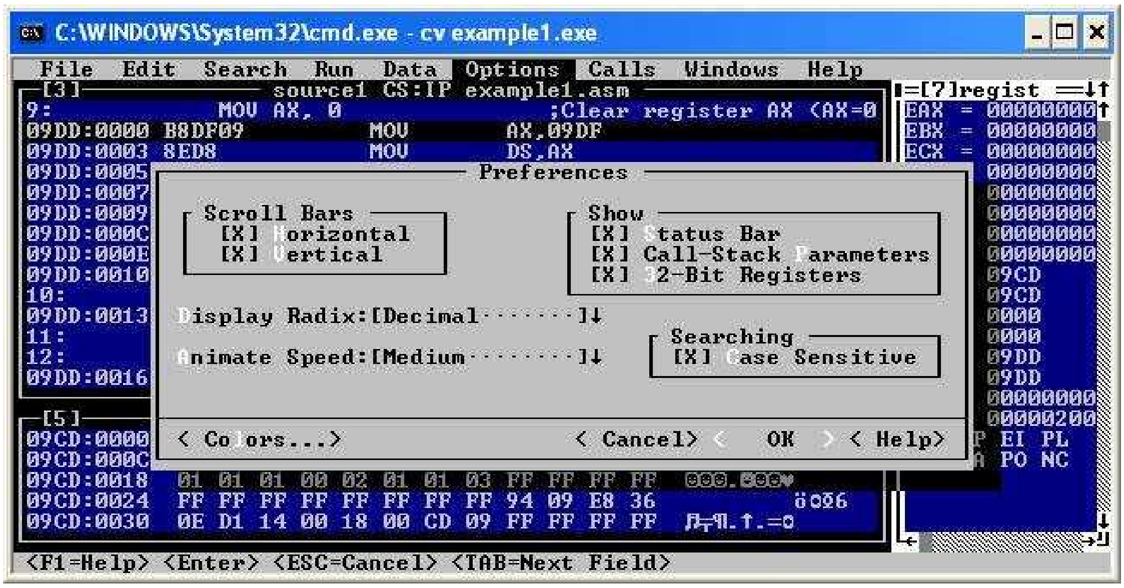
    Code1
    Registers
    Memory 1

Press “ Alt-F5” to arrange the windows on the screen.


C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help
<Trace> <Step> <Go> <After Return> <ESC=Cancel>

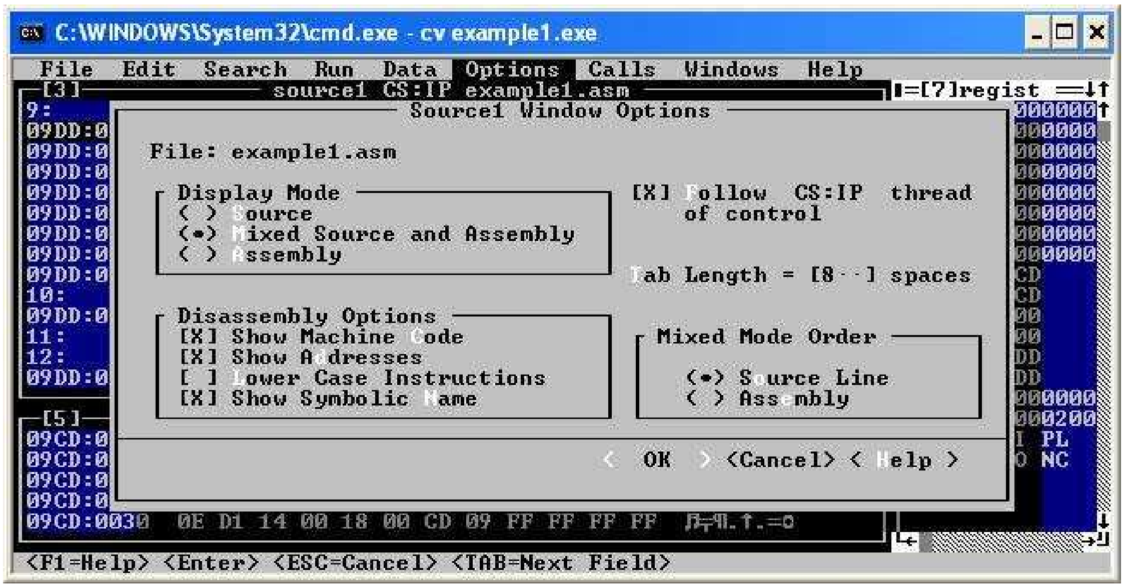
```
[3]                 source1 CS:IP example1.asm
9:              MOV AX, 0                ;Clear register AX (AX=0
09DD:0000 B8DF09            MOV       AX,09DF
09DD:0003 8ED8              MOV       DS,AX
09DD:0005 8CD3              MOV       BX,SS
09DD:0007 2BD8              SUB       BX,AX
09DD:0009 C1E304            SHL       BX,04
09DD:000C 8ED0              MOV       SS,AX
09DD:000E 03E3              ADD       SP,BX
09DD:0010 B80000            MOV       AX,0000
10:             MOV AL, VAR1             ;Copy value inside memor
09DD:0013 A00C00            MOV       AL,BYTE PTR [000C]
11:                                      ;into the register AL
12:             MOV BX, OFFSET VAR2      ;Place offset of VAR2 in
09DD:0016 BB0D00            MOV       BX,000D

[5]                     memory1 b DS:0
09CD:0000  CD 20 FF 9F 00 9A F0 FE 1D F0 96 02
09CD:000C  D4 07 AB 03 D4 07 56 01 0F 04 83 09
09CD:0018  01 01 01 00 02 01 01 03 FF FF FF FF
09CD:0024  FF FF FF FF FF FF FF FF 94 09 E8 36
09CD:0030  0E D1 14 00 18 00 CD 09 FF FF FF FF

[7]regist
EAX = 00000000
EBX = 00000000
ECX = 00000000
EDX = 00000000
ESP = 00000000
EBP = 00000000
ESI = 00000000
EDI = 00000000
 DS = 09CD
 ES = 09CD
 FS = 0000
 GS = 0000
 SS = 09DD
 CS = 09DD
EIP = 00000000
EFL = 00000200
 NV UP EI PL
 NZ NA PO NC
```

- Set the options. “ Alt-O” -> Preferences. Set the options as shown and click “ ok” .


C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help
< Colors...> < Cancel> < OK > < Help>
<F1=Help> <Enter> <ESC=Cancel> <TAB=Next Field>

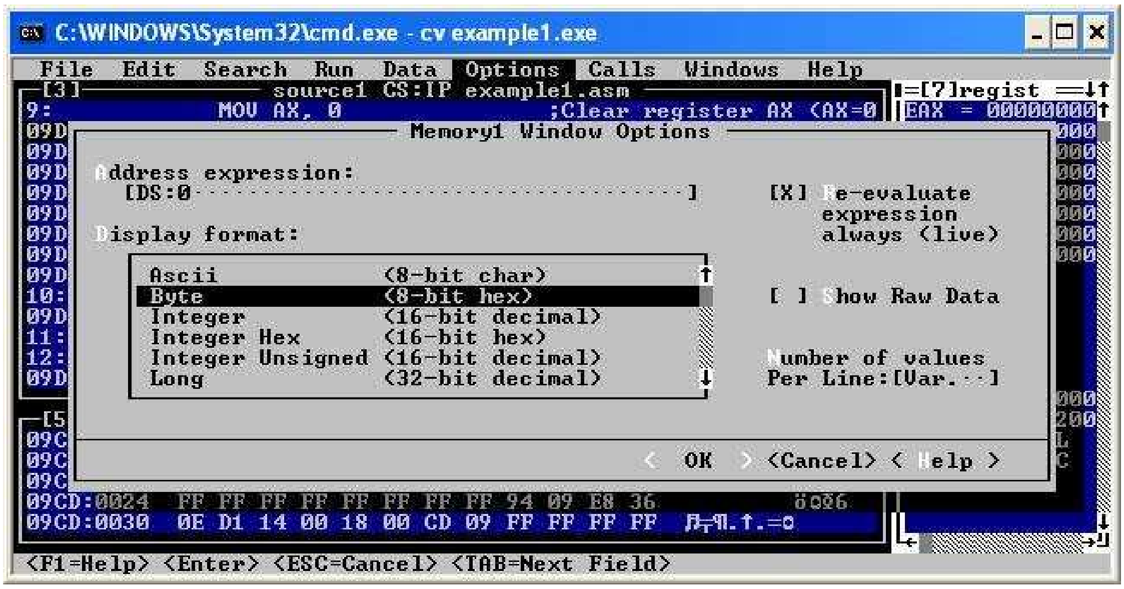
```
Preferences
Scroll Bars
  [X] Horizontal
  [X] Vertical
Show
  [X] Status Bar
  [X] Call-Stack Parameters
  [X] 32-Bit Registers
Display Radix:[Decimal]
Animate Speed:[Medium]
Searching
  [X] Case Sensitive
```

- Again, “ Alt-O” -> “ Source 1 window”



```
C:\WINDOWS\System32\cmd.exe - cv example1.exe
  File  Edit  Search  Run  Data  Options  Calls  Windows  Help
                       Source1 Window Options

  File: example1.asm

   Display Mode                              [X] Follow  CS:IP  thread
   ( ) Source                                    of control
   (•) Mixed Source and Assembly
   ( ) Assembly                              Tab Length = [8··] spaces

   Disassembly Options                        Mixed Mode Order
   [X] Show Machine Code
   [X] Show Addresses                            (•) Source Line
   [ ] Lower Case Instructions                   ( ) Assembly
   [X] Show Symbolic Name

                                         < OK > <Cancel> < Help >
09CD:0030  0E D1 14 00 18 00 CD 09 FF FF FF FF
<F1=Help> <Enter> <ESC=Cancel> <TAB=Next Field>
```

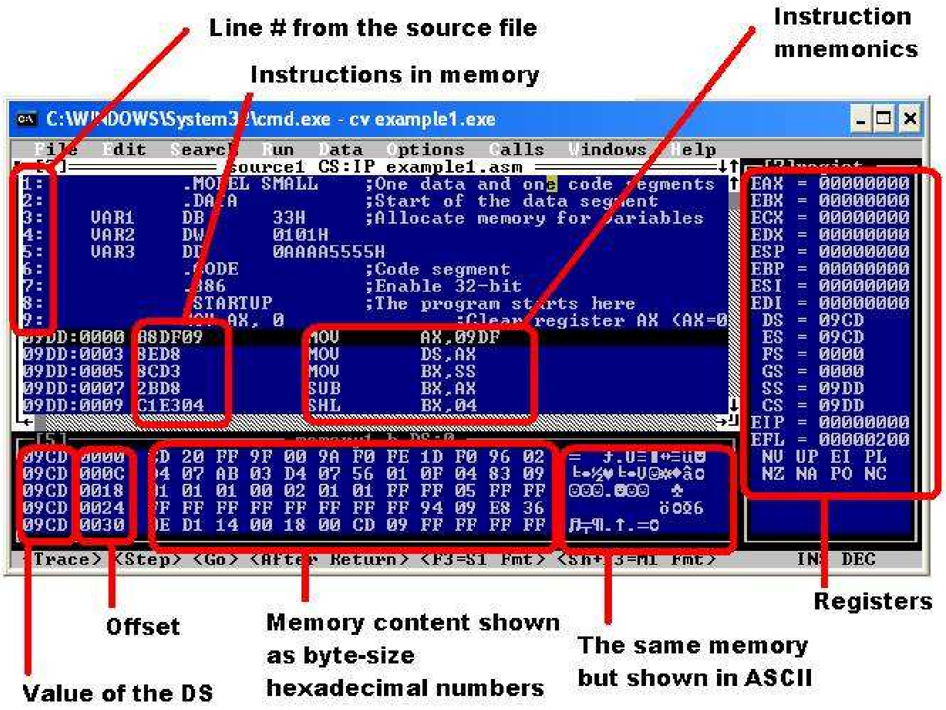
- “ Alt-O” - >” Memory 1 window”



```
C:\WINDOWS\System32\cmd.exe - cv example1.exe
  File  Edit  Search  Run  Data  Options  Calls  Windows  Help
9:            MOV AX, 0              ;Clear register AX (AX=0    EAX = 00000000
                       Memory1 Window Options

 Address expression:
   [DS:0··································]         [X] Re-evaluate
                                                          expression
 Display format:                                          always (live)

   Ascii              (8-bit char)
   Byte               (8-bit hex)                     [ ] Show Raw Data
   Integer            (16-bit decimal)
   Integer Hex        (16-bit hex)
   Integer Unsigned   (16-bit decimal)                Number of values
   Long               (32-bit decimal)                Per Line:[Var.··]

                                         < OK > <Cancel> < Help >
09CD:0024  FF FF FF FF FF FF FF FF 94 09 E8 36
09CD:0030  0E D1 14 00 18 00 CD 09 FF FF FF FF
<F1=Help> <Enter> <ESC=Cancel> <TAB=Next Field>
```

The configuration is now complete.

- Let’ s look at the program.



Line # from the source file

Instructions in memory

Instruction mnemonics

C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help
[3] source1 CS:IP example1.asm
[7]regist
[5] memory1 b DS:0
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt> INS DEC

```
1:              .MODEL SMALL     ;One data and one code segments
2:              .DATA            ;Start of the data segment
3:     VAR1     DB      33H      ;Allocate memory for variables
4:     VAR2     DW      0101H
5:     VAR3     DD      0AAAA5555H
6:              .CODE            ;Code segment
7:              .386             ;Enable 32-bit
8:              .STARTUP         ;The program starts here
9:              MOV AX, 0                 ;Clear register AX (AX=0
09DD:0000 B8DF09         MOV       AX,09DF
09DD:0003 8ED8           MOV       DS,AX
09DD:0005 8CD3           MOV       BX,SS
09DD:0007 2BD8           SUB       BX,AX
09DD:0009 C1E304         SHL       BX,04

09CD 0000  CD 20 FF 9F 00 9A F0 FE 1D F0 96 02
09CD 000C  D4 07 AB 03 D4 07 56 01 0F 04 83 09
09CD 0018  01 01 01 00 02 01 01 FF FF 05 FF FF
09CD 0024  FF FF FF FF FF FF FF FF 94 09 E8 36
09CD 0030  0E D1 14 00 18 00 CD 09 FF FF FF FF

EAX = 00000000
EBX = 00000000
ECX = 00000000
EDX = 00000000
ESP = 00000000
EBP = 00000000
ESI = 00000000
EDI = 00000000
 DS = 09CD
 ES = 09CD
 FS = 0000
 GS = 0000
 SS = 09DD
 CS = 09DD
EIP = 00000000
EFL = 00000200
 NV UP EI PL
 NZ NA PO NC
```

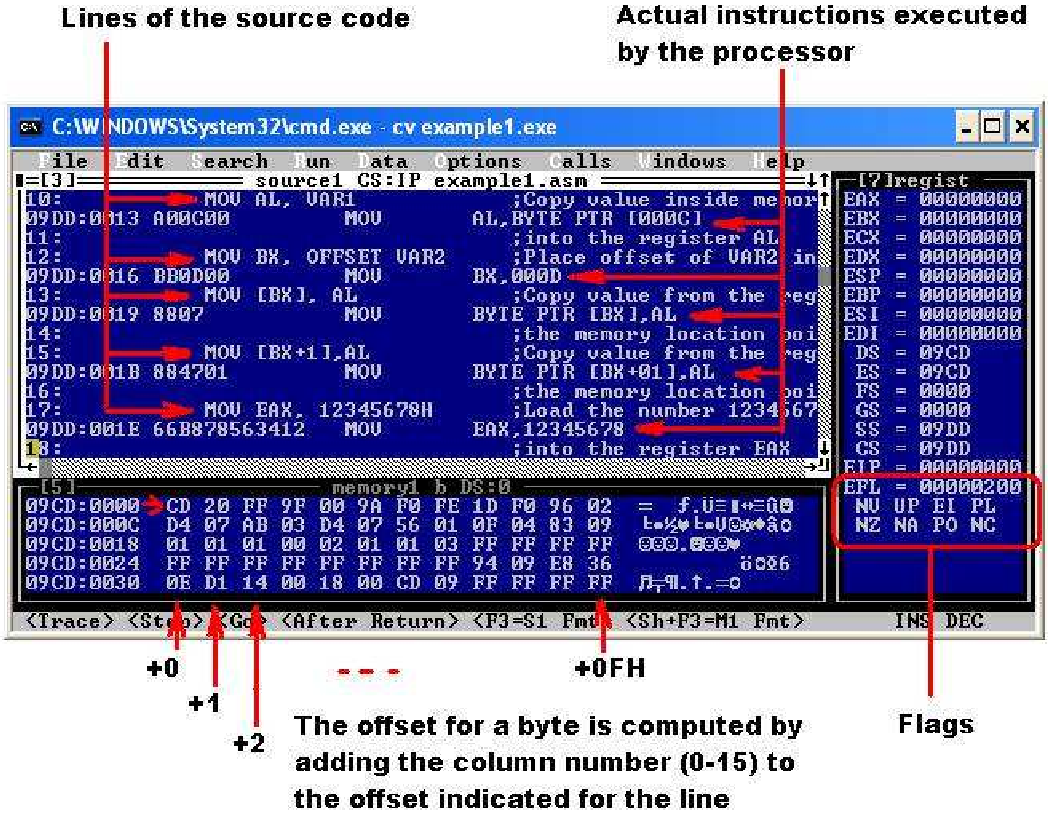
Offset

Memory content shown as byte-size hexadecimal numbers

The same memory but shown in ASCII

Registers

Value of the DS



Lines of the source code

Actual instructions executed by the processor

C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help
[3] source1 CS:IP example1.asm
[7]regist
[5] memory1 b DS:0
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt> INS DEC

```
10:                  MOV AL, VAR1            ;Copy value inside memor
09DD:0013 A00C00          MOV       AL,BYTE PTR [000C]
11:                                          ;into the register AL
12:                  MOV BX, OFFSET VAR2     ;Place offset of VAR2 in
09DD:0016 BB0D00          MOV       BX,000D
13:                  MOV [BX], AL            ;Copy value from the reg
09DD:0019 8807            MOV       BYTE PTR [BX],AL
14:                                          ;the memory location poi
15:                  MOV [BX+1],AL           ;Copy value from the reg
09DD:001B 884701          MOV       BYTE PTR [BX+01],AL
16:                                          ;the memory location poi
17:                  MOV EAX, 12345678H      ;Load the number 1234567
09DD:001E 66B878563412    MOV       EAX,12345678
18:                                          ;into the register EAX

09CD:0000 CD 20 FF 9F 00 9A F0 FE 1D F0 96 02
09CD:000C D4 07 AB 03 D4 07 56 01 0F 04 83 09
09CD:0018 01 01 01 00 02 01 01 03 FF FF FF FF
09CD:0024 FF FF FF FF FF FF FF FF 94 09 E8 36
09CD:0030 0E D1 14 00 18 00 CD 09 FF FF FF FF

EAX = 00000000
EBX = 00000000
ECX = 00000000
EDX = 00000000
ESP = 00000000
EBP = 00000000
ESI = 00000000
EDI = 00000000
 DS = 09CD
 ES = 09CD
 FS = 0000
 GS = 0000
 SS = 09DD
 CS = 09DD
EIP = 00000000
EFL = 00000200
 NV UP EI PL
 NZ NA PO NC
```

+0

+1

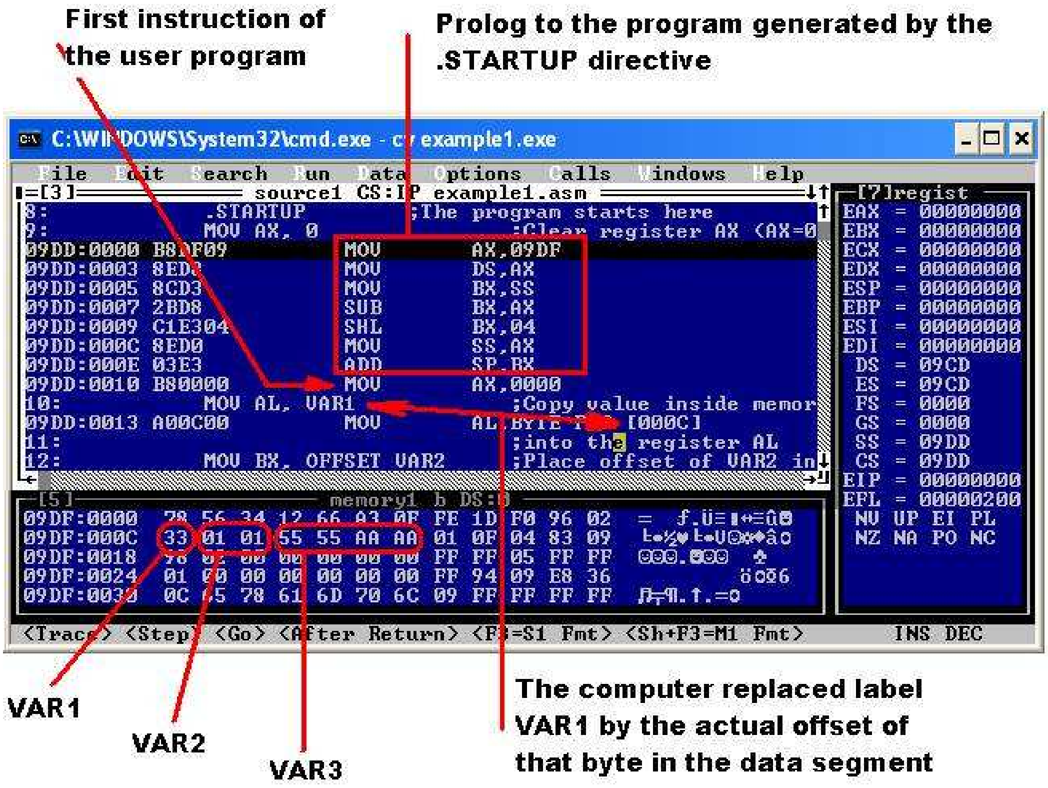
+2

- - -

+0FH

The offset for a byte is computed by adding the column number (0-15) to the offset indicated for the line

Flags



First instruction of the user program

Prolog to the program generated by the .STARTUP directive

```
C:\WINDOWS\System32\cmd.exe - cv example1.exe
 File  Edit  Search  Run  Data  Options  Calls  Windows  Help
=[3]======== source1 CS:IP example1.asm ==========      [7]regist
8:          .STARTUP         ;The program starts here    EAX = 00000000
9:          MOV AX, 0           ;Clear register AX (AX=0 EBX = 00000000
09DD:0000 B8DF09          MOV      AX,09DF               ECX = 00000000
09DD:0003 8ED8            MOV      DS,AX                 EDX = 00000000
09DD:0005 8CD3            MOV      BX,SS                 ESP = 00000000
09DD:0007 2BD8            SUB      BX,AX                 EBP = 00000000
09DD:0009 C1E304          SHL      BX,04                 ESI = 00000000
09DD:000C 8ED0            MOV      SS,AX                 EDI = 00000000
09DD:000E 03E3            ADD      SP,BX                  DS = 09CD
09DD:0010 B80000          MOV      AX,0000                ES = 09CD
10:         MOV AL, VAR1        ;Copy value inside memor  FS = 0000
09DD:0013 A00C00          MOV      AL,BYTE PTR [000C]     GS = 0000
11:                             ;into the register AL     SS = 09DD
12:         MOV BX, OFFSET VAR2 ;Place offset of VAR2 in  CS = 09DD
                                                         EIP = 00000000
-[5]----------- memory1 b DS:0 ---------------------     EFL = 00000200
09DF:0000  78 56 34 12 66 A3 0F FE 1D F0 96 02  =  ƒ.Ü≡▮↔≡ûΘ   NV UP EI PL
09DF:000C  33 01 01 55 55 AA AA 01 0F 04 83 09  └☻½♥└☻U☺☼♦â○   NZ NA PO NC
09DF:0018  98 02 00 00 00 00 00 FF FF 05 FF FF  ΘΘΘ.ΘΘΘ   ♣
09DF:0024  01 00 00 00 00 00 00 FF 94 09 E8 36        ö○Φ6
09DF:0030  0C 65 78 61 6D 70 6C 09 FF FF FF FF  ♫♀¶.↑.=○
 <Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>     INS DEC
```

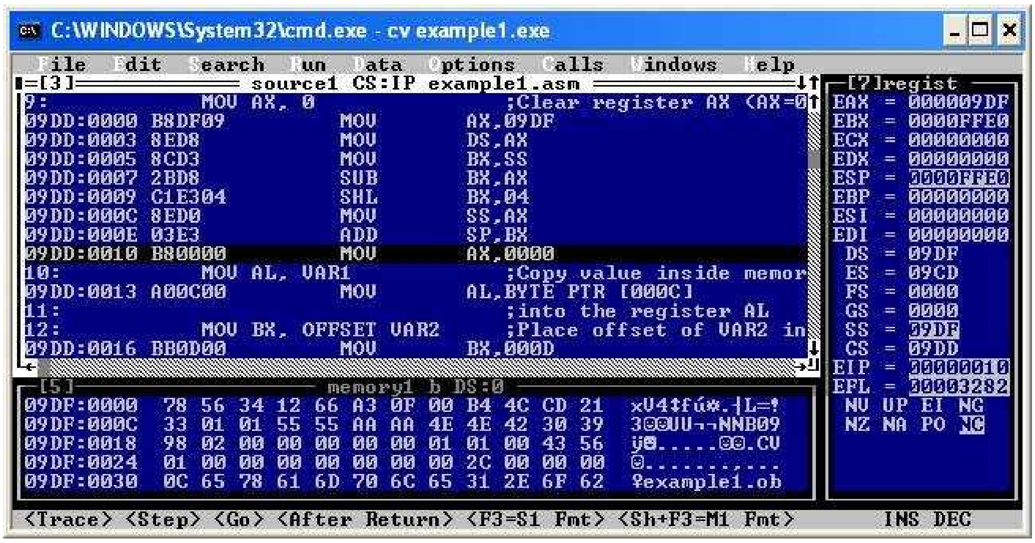
VAR1

VAR2

VAR3

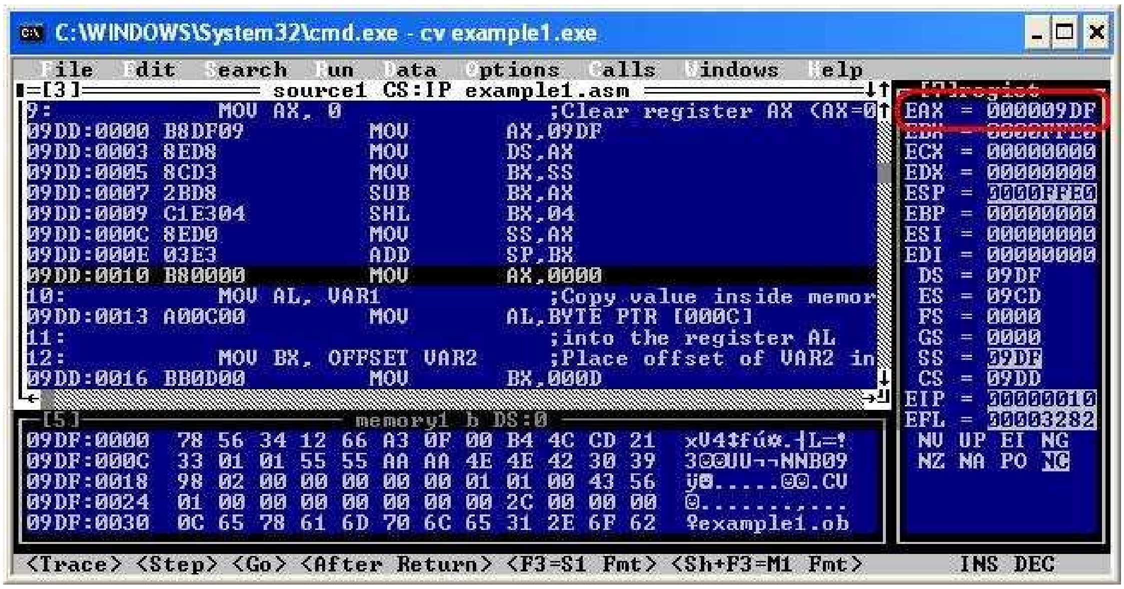
The computer replaced label VAR1 by the actual offset of that byte in the data segment

- step through the program and observe execution of each instruction.
    - o Press “ F10” .
    - o The debugger will show execution of the first line of the prolog.
    - o Press “ F10” until instruction “ MOV AX,0” is highlighted. This is the first instruction of your program.



```
C:\WINDOWS\System32\cmd.exe - cv example1.exe
 File  Edit  Search  Run  Data  Options  Calls  Windows  Help
=[3]======== source1 CS:IP example1.asm ==========      [7]regist
9:          MOV AX, 0           ;Clear register AX (AX=0 EAX = 000009DF
09DD:0000 B8DF09          MOV      AX,09DF               EBX = 0000FFE0
09DD:0003 8ED8            MOV      DS,AX                 ECX = 00000000
09DD:0005 8CD3            MOV      BX,SS                 EDX = 00000000
09DD:0007 2BD8            SUB      BX,AX                 ESP = 0000FFE0
09DD:0009 C1E304          SHL      BX,04                 EBP = 00000000
09DD:000C 8ED0            MOV      SS,AX                 ESI = 00000000
09DD:000E 03E3            ADD      SP,BX                 EDI = 00000000
09DD:0010 B80000          MOV      AX,0000                DS = 09DF
10:         MOV AL, VAR1        ;Copy value inside memor  ES = 09CD
09DD:0013 A00C00          MOV      AL,BYTE PTR [000C]     FS = 0000
11:                             ;into the register AL     GS = 0000
12:         MOV BX, OFFSET VAR2 ;Place offset of VAR2 in  SS = 09DF
09DD:0016 BB0D00          MOV      BX,000D                CS = 09DD
                                                         EIP = 00000010
-[5]----------- memory1 b DS:0 ---------------------     EFL = 00003282
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!   NV UP EI NG
09DF:000C  33 01 01 55 55 AA AA 4E 4E 42 30 39  3☺☺UU¬¬NNB09   NZ NA PO NC
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺.......,...
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob
 <Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>     INS DEC
```

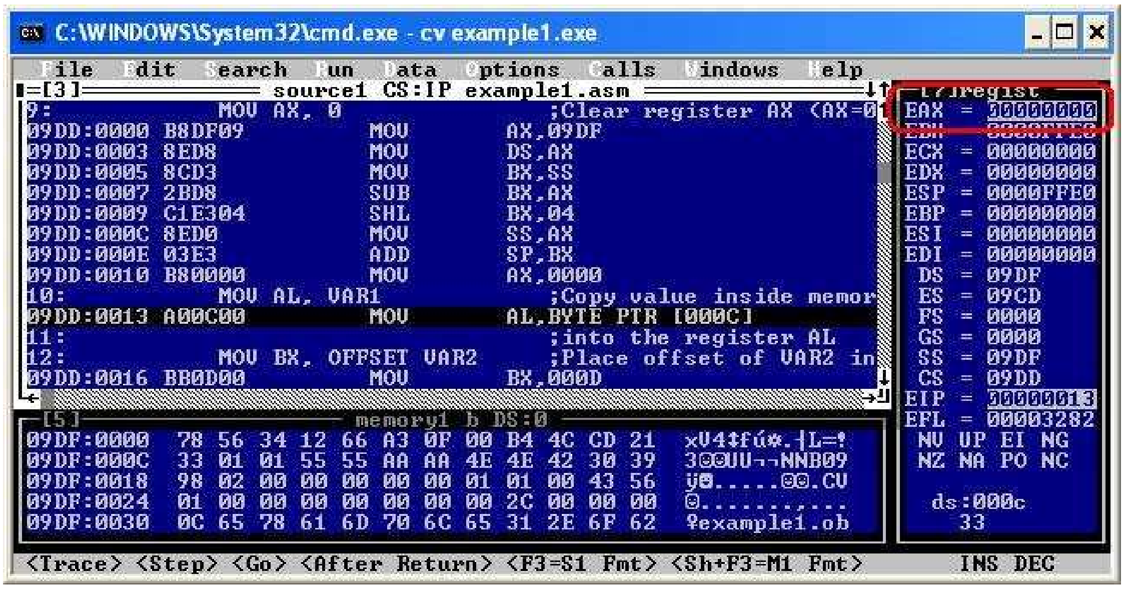
Observe the value in the register EAX. Register AX contains number 09DFH.



C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
[3]              source1 CS:IP example1.asm
9:              MOV AX, 0                  ;Clear register AX (AX=0
09DD:0000 B8DF09          MOV       AX,09DF
09DD:0003 8ED8            MOV       DS,AX
09DD:0005 8CD3            MOV       BX,SS
09DD:0007 2BD8            SUB       BX,AX
09DD:0009 C1E304          SHL       BX,04
09DD:000C 8ED0            MOV       SS,AX
09DD:000E 03E3            ADD       SP,BX
09DD:0010 B80000          MOV       AX,0000
10:             MOV AL, VAR1               ;Copy value inside memor
09DD:0013 A00C00          MOV       AL,BYTE PTR [000C]
11:                                        ;into the register AL
12:             MOV BX, OFFSET VAR2        ;Place offset of VAR2 in
09DD:0016 BB0D00          MOV       BX,000D

[5]              memory1 b DS:0
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!
09DF:000C  33 01 01 55 55 AA AA 4E 4E 42 30 39  3☺☺UU¬¬NNB09
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺.......,...
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob

[7]regist
EAX = 000009DF
EBX = 0000FFE0
ECX = 00000000
EDX = 00000000
ESP = 0000FFE0
EBP = 00000000
ESI = 00000000
EDI = 00000000
 DS = 09DF
 ES = 09CD
 FS = 0000
 GS = 0000
 SS = 09DF
 CS = 09DD
EIP = 00000010
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC

<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>        INS DEC
```

Now press “ F10” . The debugger will execute the highlighted instruction. Note the change in the content of EAX and the fact that the register has been highlighted by the debugger, indicating the change.



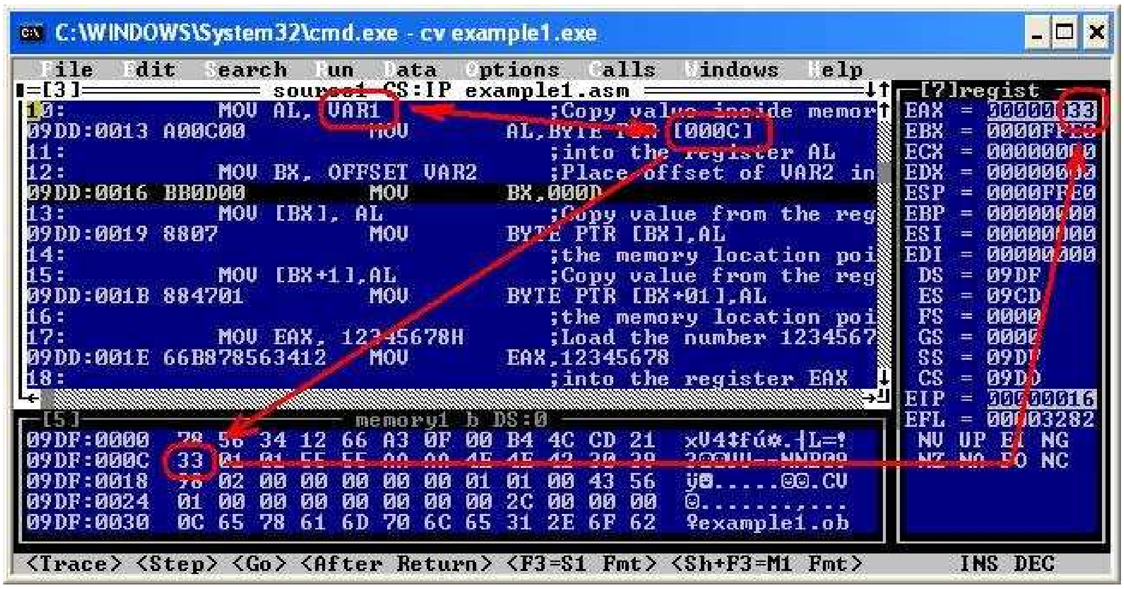
C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
[3]              source1 CS:IP example1.asm
9:              MOV AX, 0                  ;Clear register AX (AX=0
09DD:0000 B8DF09          MOV       AX,09DF
09DD:0003 8ED8            MOV       DS,AX
09DD:0005 8CD3            MOV       BX,SS
09DD:0007 2BD8            SUB       BX,AX
09DD:0009 C1E304          SHL       BX,04
09DD:000C 8ED0            MOV       SS,AX
09DD:000E 03E3            ADD       SP,BX
09DD:0010 B80000          MOV       AX,0000
10:             MOV AL, VAR1               ;Copy value inside memor
09DD:0013 A00C00          MOV       AL,BYTE PTR [000C]
11:                                        ;into the register AL
12:             MOV BX, OFFSET VAR2        ;Place offset of VAR2 in
09DD:0016 BB0D00          MOV       BX,000D

[5]              memory1 b DS:0
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!
09DF:000C  33 01 01 55 55 AA AA 4E 4E 42 30 39  3☺☺UU¬¬NNB09
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺.......,...
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob

[7]regist
EAX = 00000000
EBX = 0000FFE0
ECX = 00000000
EDX = 00000000
ESP = 0000FFE0
EBP = 00000000
ESI = 00000000
EDI = 00000000
 DS = 09DF
 ES = 09CD
 FS = 0000
 GS = 0000
 SS = 09DF
 CS = 09DD
EIP = 00000013
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC

  ds:000c
   33

<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>        INS DEC
```

The highlighting the code window moved to the next instruction.

Note that the line of the source code “ MOV AL, VAR1” became “ MOV AL, [000C] where 000CH is the actual offset of VAR1 in the data segment. You can check that this is true by checking the content of memory location DS:000CH in the data window.

Now execute this instruction by pressing “ F10” . Content of the register AL changed, taking the value from the VAR1.



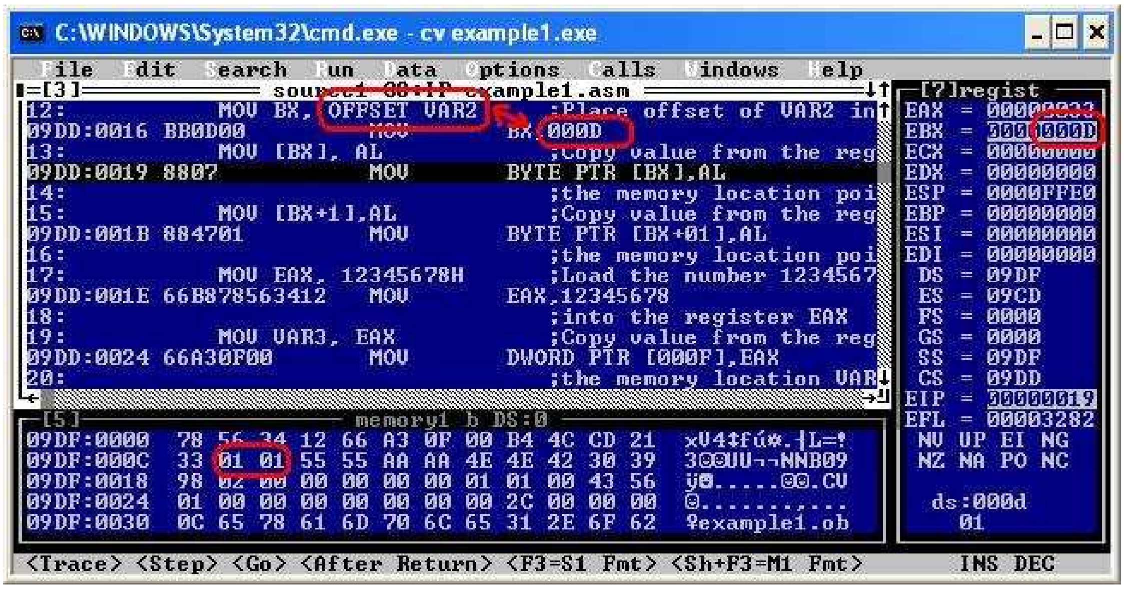
C:\WINDOWS\System32\cmd.exe - cv example1.exe

```
[3]  source1 CS:IP example1.asm
10:              MOV AL, VAR1            ;Copy value inside memor
09DD:0013 A00C00           MOV        AL,BYTE PTR [000C]
11:                                      ;into the register AL
12:              MOV BX, OFFSET VAR2     ;Place offset of VAR2 in
09DD:0016 BB0D00           MOV        BX,000D
13:              MOV [BX], AL            ;Copy value from the reg
09DD:0019 8807             MOV        BYTE PTR [BX],AL
14:                                      ;the memory location poi
15:              MOV [BX+1],AL           ;Copy value from the reg
09DD:001B 884701           MOV        BYTE PTR [BX+01],AL
16:                                      ;the memory location poi
17:              MOV EAX, 12345678H      ;Load the number 1234567
09DD:001E 66B878563412     MOV        EAX,12345678
18:                                      ;into the register EAX
```

```
[5]  memory1 b DS:0
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!
09DF:000C  33 01 01 55 55 AA AA 4E 4E 42 30 39  3☺☺UU¬¬NNB09
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺...........
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob
```

```
[7]regist
EAX = 00000033
EBX = 0000FFE0
ECX = 00000000
EDX = 00000000
ESP = 0000FFE0
EBP = 00000000
ESI = 00000000
EDI = 00000000
 DS = 09DF
 ES = 09CD
 FS = 0000
 GS = 0000
 SS = 09DF
 CS = 09DD
EIP = 00000016
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC
```

The next instruction is “ MOV BX, OFFSET VAR2” . VAR2 follows VAR1 in memory and has offset of 000DH. This is the value that will be placed into the BX upon execution of this instruction. Press “ F10” to execute.



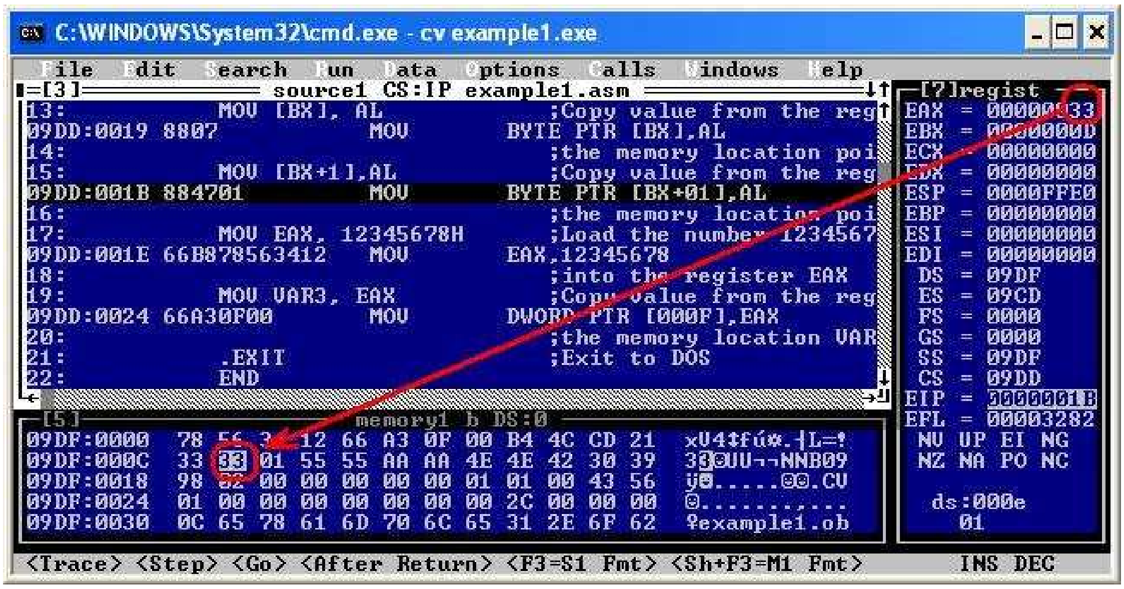
C:\WINDOWS\System32\cmd.exe - cv example1.exe

```
[3]  source1 CS:IP example1.asm
12:              MOV BX, OFFSET VAR2     ;Place offset of VAR2 in
09DD:0016 BB0D00           MOV        BX,000D
13:              MOV [BX], AL            ;Copy value from the reg
09DD:0019 8807             MOV        BYTE PTR [BX],AL
14:                                      ;the memory location poi
15:              MOV [BX+1],AL           ;Copy value from the reg
09DD:001B 884701           MOV        BYTE PTR [BX+01],AL
16:                                      ;the memory location poi
17:              MOV EAX, 12345678H      ;Load the number 1234567
09DD:001E 66B878563412     MOV        EAX,12345678
18:                                      ;into the register EAX
19:              MOV VAR3, EAX           ;Copy value from the reg
09DD:0024 66A30F00         MOV        DWORD PTR [000F],EAX
20:                                      ;the memory location VAR
```

```
[5]  memory1 b DS:0
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!
09DF:000C  33 01 01 55 55 AA AA 4E 4E 42 30 39  3☺☺UU¬¬NNB09
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺...........
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob
```

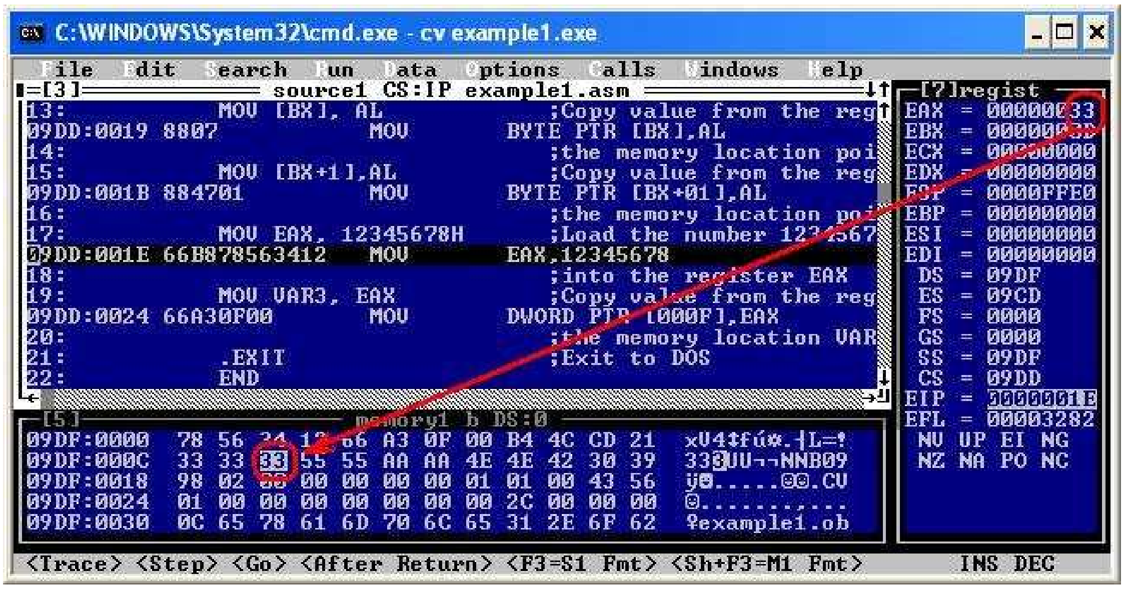
```
[7]regist
EAX = 00000033
EBX = 0000000D
ECX = 00000000
EDX = 00000000
ESP = 0000FFE0
EBP = 00000000
ESI = 00000000
EDI = 00000000
 DS = 09DF
 ES = 09CD
 FS = 0000
 GS = 0000
 SS = 09DF
 CS = 09DD
EIP = 00000019
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC

  ds:000d
   01
```

The following instruction “ MOV [BX], AL” will copy the content of AL into the memory location pointed by BX within the data segment. After the previous instruction BX contains the offset of the first byte of VAR2 or 000DH. That is where the data from AL will appear. Press “ F10” to execute. Note the debugger also highlighted changes in the data window.



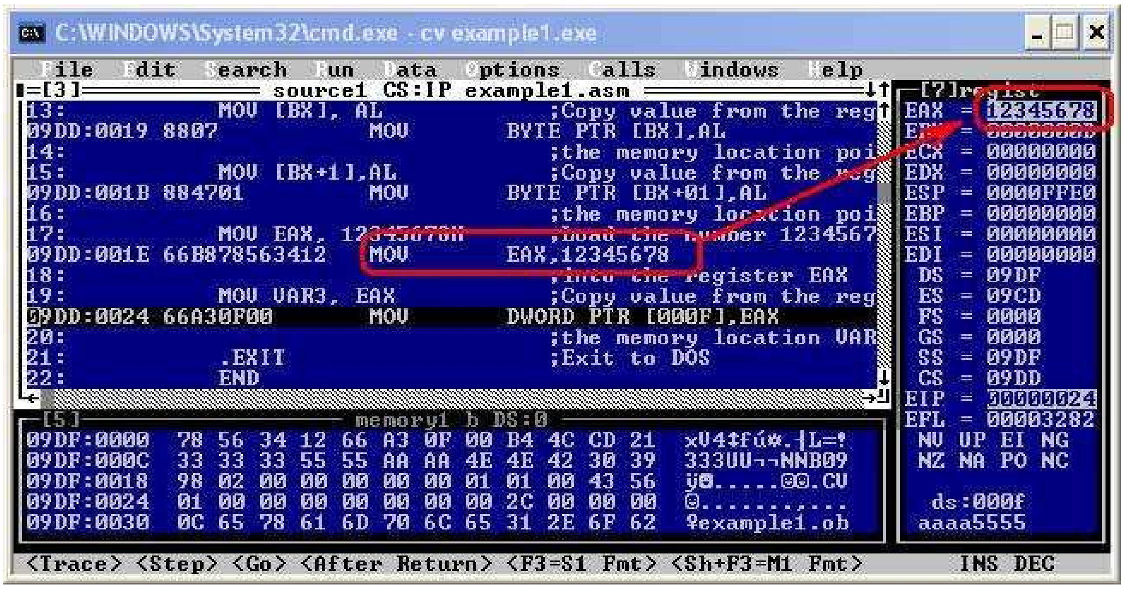
```
C:\WINDOWS\System32\cmd.exe - cv example1.exe
 File  Edit  Search  Run  Data  Options  Calls  Windows  Help
=[3]============= source1 CS:IP example1.asm ==============  -[7]regist
13:            MOV [BX], AL              ;Copy value from the reg  EAX = 00000033
09DD:0019 8807        MOV      BYTE PTR [BX],AL                    EBX = 0000000D
14:                                      ;the memory location poi  ECX = 00000000
15:            MOV [BX+1],AL             ;Copy value from the reg  EDX = 00000000
09DD:001B 884701      MOV      BYTE PTR [BX+01],AL                 ESP = 0000FFE0
16:                                      ;the memory location poi  EBP = 00000000
17:            MOV EAX, 12345678H        ;Load the number 1234567  ESI = 00000000
09DD:001E 66B878563412  MOV    EAX,12345678                        EDI = 00000000
18:                                      ;into the register EAX     DS = 09DF
19:            MOV VAR3, EAX             ;Copy value from the reg   ES = 09CD
09DD:0024 66A30F00    MOV      DWORD PTR [000F],EAX                 FS = 0000
20:                                      ;the memory location VAR   GS = 0000
21:             .EXIT                    ;Exit to DOS               SS = 09DF
22:            END                                                  CS = 09DD
                                                                   EIP = 0000001B
-[5]----------------- memory1 b DS:0 ---------------------------   EFL = 00003282
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!         NV UP EI NG
09DF:000C  33 33 01 55 55 AA AA 4E 4E 42 30 39  33☺UU¬¬NNB09         NZ NA PO NC
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺...........          ds:000e
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob            01
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>        INS DEC
```

Instruction “ MOV [BX+1], AL” will copy the content of the register AL into the memory location with offset equal whatever the number is in BX plus 1. In our case BX=000DH, then the offset is 000DH+0001H=000EH. That is the second byte of the VAR2. Press “ F10” to execute. Note the change in the memory content.



```
C:\WINDOWS\System32\cmd.exe - cv example1.exe
 File  Edit  Search  Run  Data  Options  Calls  Windows  Help
=[3]============ source1 CS:IP example1.asm ===========    [7]regist
13:             MOV [BX], AL             ;Copy value from the reg   EAX = 00000033
09DD:0019 8807           MOV     BYTE PTR [BX],AL                   EBX = 000000[illegible]
14:                                      ;the memory location poi   ECX = 00000000
15:             MOV [BX+1],AL            ;Copy value from the reg   EDX = 00000000
09DD:001B 884701         MOV     BYTE PTR [BX+01],AL                ESP = 0000FFE0
16:                                      ;the memory location poi   EBP = 00000000
17:             MOV EAX, 12345678H       ;Load the number 1234567   ESI = 00000000
09DD:001E 66B878563412   MOV     EAX,12345678                       EDI = 00000000
18:                                      ;into the register EAX      DS = 09DF
19:             MOV VAR3, EAX            ;Copy value from the reg    ES = 09CD
09DD:0024 66A30F00       MOV     DWORD PTR [000F],EAX                FS = 0000
20:                                      ;the memory location VAR    GS = 0000
21:             .EXIT                    ;Exit to DOS                SS = 09DF
22:             END                                                  CS = 09DD
                                                                    EIP = 0000001E
-[5]---------------- memory1 b DS:0 ---------------                 EFL = 00003282
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!         NV UP EI NG
09DF:000C  33 33 33 55 55 AA AA 4E 4E 42 30 39  333UU¬¬NNB09         NZ NA PO NC
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺...........
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>        INS DEC
```

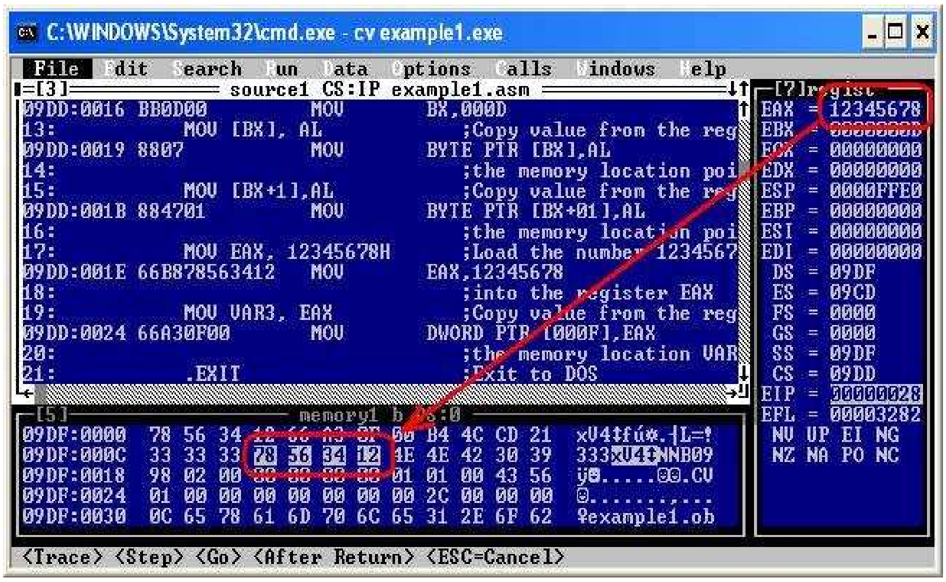
Instruction “ MOV EAX, 12345678H” will place number 12345678H into the register EAX. Press “ F10” to execute.



```
C:\WINDOWS\System32\cmd.exe - cv example1.exe
 File  Edit  Search  Run  Data  Options  Calls  Windows  Help
=[3]============ source1 CS:IP example1.asm ===========    [7]regist
13:             MOV [BX], AL             ;Copy value from the reg   EAX = 12345678
09DD:0019 8807           MOV     BYTE PTR [BX],AL                   EBX = [illegible]
14:                                      ;the memory location poi   ECX = 00000000
15:             MOV [BX+1],AL            ;Copy value from the reg   EDX = 00000000
09DD:001B 884701         MOV     BYTE PTR [BX+01],AL                ESP = 0000FFE0
16:                                      ;the memory location poi   EBP = 00000000
17:             MOV EAX, 12345678H       ;Load the number 1234567   ESI = 00000000
09DD:001E 66B878563412   MOV     EAX,12345678                       EDI = 00000000
18:                                      ;into the register EAX      DS = 09DF
19:             MOV VAR3, EAX            ;Copy value from the reg    ES = 09CD
09DD:0024 66A30F00       MOV     DWORD PTR [000F],EAX                FS = 0000
20:                                      ;the memory location VAR    GS = 0000
21:             .EXIT                    ;Exit to DOS                SS = 09DF
22:             END                                                  CS = 09DD
                                                                    EIP = 00000024
-[5]---------------- memory1 b DS:0 ---------------                 EFL = 00003282
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!         NV UP EI NG
09DF:000C  33 33 33 55 55 AA AA 4E 4E 42 30 39  333UU¬¬NNB09         NZ NA PO NC
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺...........          ds:000f
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob          aaaa5555
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>        INS DEC
```

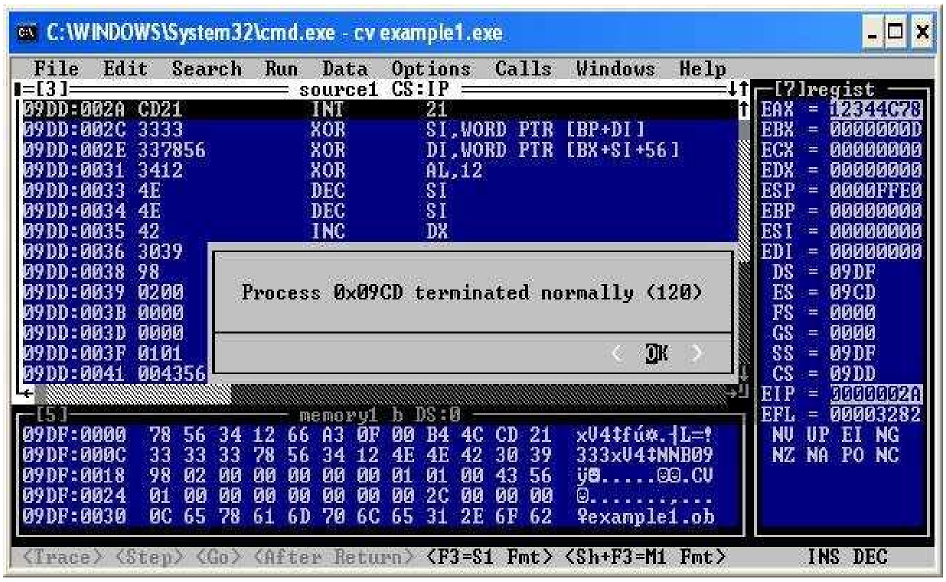
The instruction “ MOV VAR3, EAX” became “ MOV DWORD PTR [000F], EAX ” .

VAR3 has been replaced by the actual offset (000FH) of VAR3 in the data memory. This instruction will take the content of the EAX and place into the four consecutive bytes of memory (a 32-bit variable) starting with the offset 000FH. Press “ F10” to execute.


C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help
<Trace> <Step> <Go> <After Return> <ESC=Cancel>

```
=[3]======== source1 CS:IP example1.asm ========     [7]regist
09DD:0016 BB0D00         MOV      BX,000D               EAX = 12345678
13:          MOV [BX], AL          ;Copy value from the reg  EBX = 0000000D
09DD:0019 8807           MOV      BYTE PTR [BX],AL      ECX = 00000000
14:                                ;the memory location poi  EDX = 00000000
15:          MOV [BX+1],AL         ;Copy value from the reg  ESP = 0000FFE0
09DD:001B 884701         MOV      BYTE PTR [BX+01],AL   EBP = 00000000
16:                                ;the memory location poi  ESI = 00000000
17:          MOV EAX, 12345678H    ;Load the number 1234567  EDI = 00000000
09DD:001E 66B878563412   MOV      EAX,12345678           DS = 09DF
18:                                ;into the register EAX     ES = 09CD
19:          MOV VAR3, EAX         ;Copy value from the reg   FS = 0000
09DD:0024 66A30F00       MOV      DWORD PTR [000F],EAX    GS = 0000
20:                                ;the memory location VAR   SS = 09DF
21:          .EXIT                 ;Exit to DOS               CS = 09DD
                                                          EIP = 00000028
[5]------------ memory1 b DS:0 ------------               EFL = 00003282
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!   NV UP EI NG
09DF:000C  33 33 33 78 56 34 12 4E 4E 42 30 39  333xV4‡NNB09   NZ NA PO NC
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺...........
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob
```

That was the last instruction of the user program. The remaining instructions are generated by the .EXIT directive and serve to terminate the program. Press “ F10” until the process terminates.


C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help
OK
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>   INS DEC

```
=[3]============ source1 CS:IP ============          [7]regist
09DD:002A CD21           INT      21                    EAX = 12344C78
09DD:002C 3333           XOR      SI,WORD PTR [BP+DI]   EBX = 0000000D
09DD:002E 337856         XOR      DI,WORD PTR [BX+SI+56] ECX = 00000000
09DD:0031 3412           XOR      AL,12                 EDX = 00000000
09DD:0033 4E             DEC      SI                    ESP = 0000FFE0
09DD:0034 4E             DEC      SI                    EBP = 00000000
09DD:0035 42             INC      DX                    ESI = 00000000
09DD:0036 3039                                          EDI = 00000000
09DD:0038 98                                             DS = 09DF
09DD:0039 0200     Process 0x09CD terminated normally (120)  ES = 09CD
09DD:003B 0000                                           FS = 0000
09DD:003D 0000                                           GS = 0000
09DD:003F 0101                                           SS = 09DF
09DD:0041 004356                                         CS = 09DD
                                                        EIP = 0000002A
[5]------------ memory1 b DS:0 ------------             EFL = 00003282
09DF:0000  78 56 34 12 66 A3 0F 00 B4 4C CD 21  xV4‡fú☼.┤L=!   NV UP EI NG
09DF:000C  33 33 33 78 56 34 12 4E 4E 42 30 39  333xV4‡NNB09   NZ NA PO NC
09DF:0018  98 02 00 00 00 00 00 01 01 00 43 56  ÿ☻.....☺☺.CV
09DF:0024  01 00 00 00 00 00 00 00 2C 00 00 00  ☺...........
09DF:0030  0C 65 78 61 6D 70 6C 65 31 2E 6F 62  ♀example1.ob
```

Experiment

1

Flight 8086 Training Board

Objective

The aim of this lab experiment is to familiarize the students with Flight 8086 training board.

Equipment

Flight 8086 training board, PC with Flight86 software, download cable.

Tasks to be Performed

Connecting the 8086 training board to PC (using COM1 port)

Study of different commands provided by the training board

Program Entry, Execution and Debugging

Assembling and disassembling of a program

Displaying the contents of registers and memory locations

Modifying the registers and memory contents

Single-step execution and Breakpoint insertion

Downloading & uploading a program file.

Running simple programs to perform

1. Arithmetic operations
2. Finding the smallest/largest number from a given list of numbers
3. Searching for a given number in a list of numbers.

1.1 Background

The FLIGHT 86 Trainer System is designed to simplify the teaching of the 8086 CPU and some of its commonly used peripherals. It can be linked to most PCs with a simple serial line, so that code may be assembled and debugged in a supportive software environment before being downloaded into the RAM on the board. The board itself may then be linked to other peripheral devices. A block diagram of this mode of operation is shown in Figure 1.1.

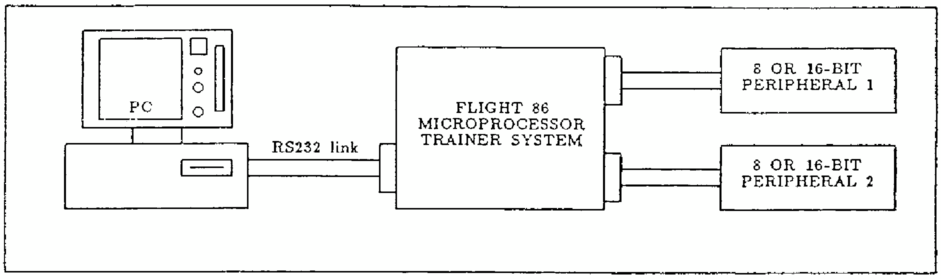


Figure 1.1: Block Diagram of the FLIGHT-86 Trainer System

Once downloaded, the code may be executed and examined in a system which is accessible to the user. Data may be manipulated on the board and the effects viewed on the PC. The software which handles this two-way transfer is supplied with the board, in the form of a monitor program resident on the board in EPROM, and a disk containing the "host" software for the PC.

1.2 Connecting the Training Board to PC

Figure 1.2 shows the FLIGHT-86 Trainer Board layout. The first step is to connect the *serial socket* (P3) on the training board to COM1 in the PC using RS323 cable. Next, connect the power cable to the *power supply connector* (P6). Finally, load the program F86GO.BAT on the PC. This should run and report the amount of RAM and EPROM on the FLIGHT-86 board, before returning the prompt as shown in Figure1.3.

1.3 Commands Provided by Flight-86

A '-' prompt on the screen means that the host is ready to accept a command. Table1.1 gives a summary of the commands that will be used during this experiment.

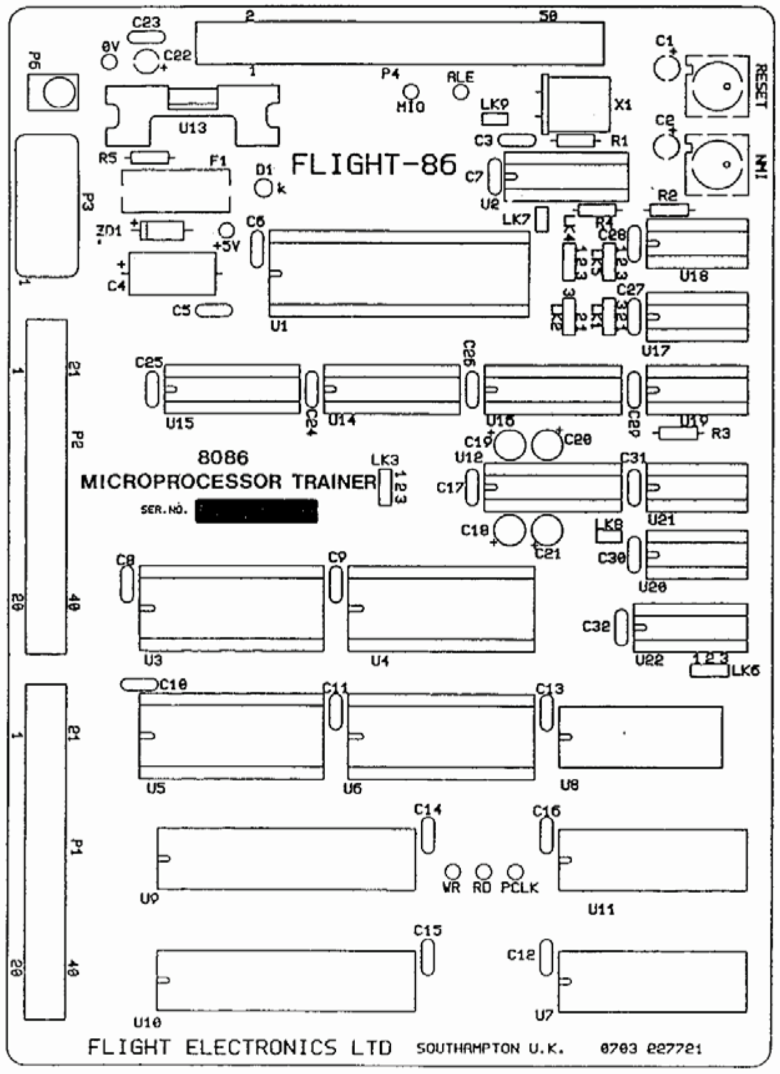


Figure 1.2: Layout of the FLIGHT-86 Training Board

```
Loading FLIGHT86 host program, please wait...
FLIGHT86 Controller Board, Host Program Version 2.1
Press ? and Enter for help - waiting for controller board response...
ROM found at F000:C000 to F000:FFFF Flight Monitor ROM version 2.0
RAM found at 0000:0000 to 0000:FFFF
-
```

Figure 1.3: Starting Message of the FLIGHT-86 Training Board

Table 1.1: Summary of some commands provided by FLIGHT-86

KEY	PARAMETER	DESCRIPTION
ESC		Press the Escape button to stop the current command
X		Resets the training board
Q		Terminates running of the board software and returns control to the operating system
?	[*command letter*]	Help
R	[*register*]	Allows the user to display or change the content of a register
M	[W][*segment*:] *address*1 [*address*2]	Allows the user to display or change one or more memory locations
A	[[*segment*:] *address*]	Allows the user to write 8086 assembly code directly into the training board
Z	[[V] [*segment*:] *address*1 [*address*2]]	
G	[[*segment*:] *address*]	Allows the user to execute code that has been downloaded into RAM
B	? | R | S [*segment*:] *address*	Allows the user to Display/Clear/Set break points inside his code
S	[R][[*segment*:] *address*]	Allows the user to step through code one instruction at a time
:	[*drive*:\path] *filename*	Loads an Extended Intel Hex file from disk into the memory of the training board

1.4 The First Program

Assembling a Program (Command A)

The assemble command (A [*segment*:] *address*) allows you to type in 8086 assembly code, and this code will be assembled and stored into the board memory. The following example shows you how to write a simple program using this command

Example 1.1: Using the assemble command, write a program that will add the content of two memory locations (words) and store the result in a third memory location.

1. Start the line assembler at the desired address by entering **A 0050:0100** (Note that the origin address for user RAM on the FLIGHT-86 is 0050:0100)

2. The FLIGHT-86 responds by echoing the address **0050:0100**

3. Now enter the assembly code one instruction at a time hitting ENTER after each instruction

4. Each time, the FLIGHT-86 responds by echoing the next address

5. When you are done exit from the line assembler by pressing ESC button

The screen will now look like

```
A 0050:0100
0050:0100  DW  0002
0050:0102  DW  0003
0050:0104  DW  0000
0050:0106  MOV AX,[0100]
0050:0109  ADD AX,[0102]
0050:010D  MOV [0104], AX
0050:0111  INT 5
0050:0113
-
```

Disassembling a Program (Command Z)

You can examine what you have entered using the disassemble command. If you type **Z 0050:0100 0111**, then the content of the memory locations between the addresses 0050:0100 and 0050:0111 will be disassembled as follows:

```
0050:0100  02  00          ADD AL, [BX+SI]
0050:0102  03  00          ADD AX, [BX+SI]
0050:0104  00  00          ADD [BX+SI], AL
0050:0106  A1  01  00      MOV AX,[0100]
0050:0109  03  06  02  01  ADD AX,[0102]
0050:010D  89  06  04  01  MOV [0104], AX
0050:0111  CD  05          INT 5
```

The HEX numbers between the addresses and the instructions represent the opcodes of the disassembled instructions. Notice that memory words entered as DW directives have been disassembled as ADD instructions with different parameters. This because the values of these memory words are equivalent to the opcode of the ADD instruction with the shown parameters.

Running a Program (Command G)

To run the above program enter **G 0050:0100** and press the ENTER key. The program will now run, load the word at address 0050:0100 into AX, add the content of the word at address 0050:0102 to the content of AX, store the result into the word at address 0050:0104, and terminate. Note that the instruction **INT 5** is responsible for terminating the program.

Displaying/Modifying Memory Locations (Command M)

To test the result of the above program enter **M W 0050:0104** and press the Enter key. This will display the memory word at address 0050:0104 where the result of the above program is stored. Exit from this command by pressing the ESC key.

Lets now change the content of the memory words stored at addresses 0050:0100 and 0050:0102. At the command prompt '-', enter M W 0050:0100 and press the Enter key. The content of the memory word at address 0050:0100 is displayed. To change the content of this memory location, enter a HEX number (say 0005) and press the Enter key. The content of the next memory location is displayed. Enter another HEX number (say 0007) and press the Enter key. When the content of the next memory location is displayed, press the ESC key to go back to the command prompt. These steps are shown below:

```
-M W 0050:0100
0050:0100 0002  0005
0050:0102 0003  0007
0050:0104 0005
-
```

Now run the program again and test the content of the memory word at address 0050:0104.

Breakpoint Insertion (Command B)

This command is intended for debugging user code. A breakpoint is an **INT 3** instruction inserted at an opcode position. The original opcode at this address is saved. When the code is executed it runs normally, at full speed, until it reaches this location. Then, original opcode is restored and the registers, address and first opcode byte are displayed. The user may set another break point and continue with a G instruction.

As an example, enter the command **B S 0050:010D** and press the Enter key. This will set a breakpoint at address 0050:010D in the previous program (i.e. a breakpoint is set at the instruction MOV [0104], AX). Now, run the program using the command **G 0050:0100**. Notice that the program terminates and the message "Monitor breakpoint at 0050:010D" is displayed. This means that the execution of the program stopped at location 0050:010D. You can resume the execution of the program using the command G, but let us first modify the content of register AX. At the command prompt '-', enter the command **R AX** and press the Enter key. This will display the content of AX which is 000D (i.e. 0005+0007). Modify this value by entering 0001 next to 000D and press the Enter key then ESC to go back to the command prompt. Now, continue the execution of the program from address 0050:010D using the command **G 0050:010D**. Check the content of memory word at address 0050:0104.

The previous steps are shown below:

```
-B S 0050:010D
-G 0050:0100
Monitor Breakpoint at 0050:010D
-R AX
AX 000C  0001
BX 0000
-G 0050:010D
User Break at 0050:0111
-M W 0050:0104
0050:0104  0001
0050:0106  00A1
-
```

Single-Step Execution (Command S)

This command is provided to allow the user to step through code one instruction at a time for debugging purposes. The display will be the next instruction address and opcode byte with, optionally, registers content. Once the command has started, pressing the Enter key will execute the next instruction. As an example, enter the command **S R 0050:0100** and press the Enter key. This will execute the first instruction and terminate with registers content shown on the screen. When you press Enter again, the next instruction is executed. Continue pressing the Enter key until all instructions in the program get executed, or press the ESC key to terminate the command.

1.5 Writing a Program Using Assembler on a PC

In the pervious section, we have used the assemble command to write and load simple assembly instructions into the board memory. However, for more sophisticated applications, you need to write and assemble programs on a PC before downloading them into the board memory. For this purpose, you need the following programs:

MASM: as the assembler and linker

EXE2BIN: to convert from and executable file into a binary file

BIN2HEX: to convert the binary file into an INTEL HEX file for download to the FLIGHT-86

Example 1.2: Write a program to search for a given number in a list of numbers. You should define the list as a sequence of memory bytes labeled with the letter A. The number to be searched is passed through register DL. When the program terminate, BX should contain the index of the number in the list if the number is in the list.

```
COMSEG SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:COMSEG, DS:COMSEG, ES:COMSEG, SS:COMSEG
ORG 0100h
start:
      MOV   AX,   CS
      MOV   DS,   AX ; Set the data segment

      MOV   BX,   0  ; Set BX to index of the 1st element in
                     ; the list
L1:   CMP   BX,   8  ; if BX exceeds the indices of the list
      JZ    L2       ; then end the search

      CMP   DL,   A[BX] ; if the number is found in the list
      JZ    L2          ; then end the search
      INC   BX          ; else increment BX
      JMP   L1
L2:   INT   5           ; terminate the program
A     DB    4
      DB    2
      DB    7
      DB    6
      DB    3
      DB    5
      DB    1
      DB    8
COMSEG ENDS
END start
```

Using any text editor on the PC enter the previous code. Notice that the code shown in Bold is required for every program using **MASM** and can be thought of as a template. Now, save this file as **SEARCH.ASM**. Using the Assembler, i.e. MASM, assemble and link this file to produce **SEARCH.EXE**, and using **EXE2BIN** create the binary code file **SEARCH.BIN**. Now, using **BIN2HEX**, convert this binary file to the Intel hex format file **SEARCH.HEX**. Finally load the HEX file into the board memory using the command "**:SEARCH.HEX**". Note, you may need to specify the drive and the path of the file if it is not in the same directory as the F86GO software (e.g. :C:\MyProjects\Search.hex).

To run this program, first load the required number into DX using the command **R DX**. Next, run the program using the command **G 0000**. Finally, use the command **RX BX** to check result of the search (i.e. the value of BX represents the index of the given number in the list). These steps are shown below.

```
-R DX
DX 0000  0003
SP 0500
-G 0050:0100
User Break at 0050:011A
-R BX
BX 0004
-
```