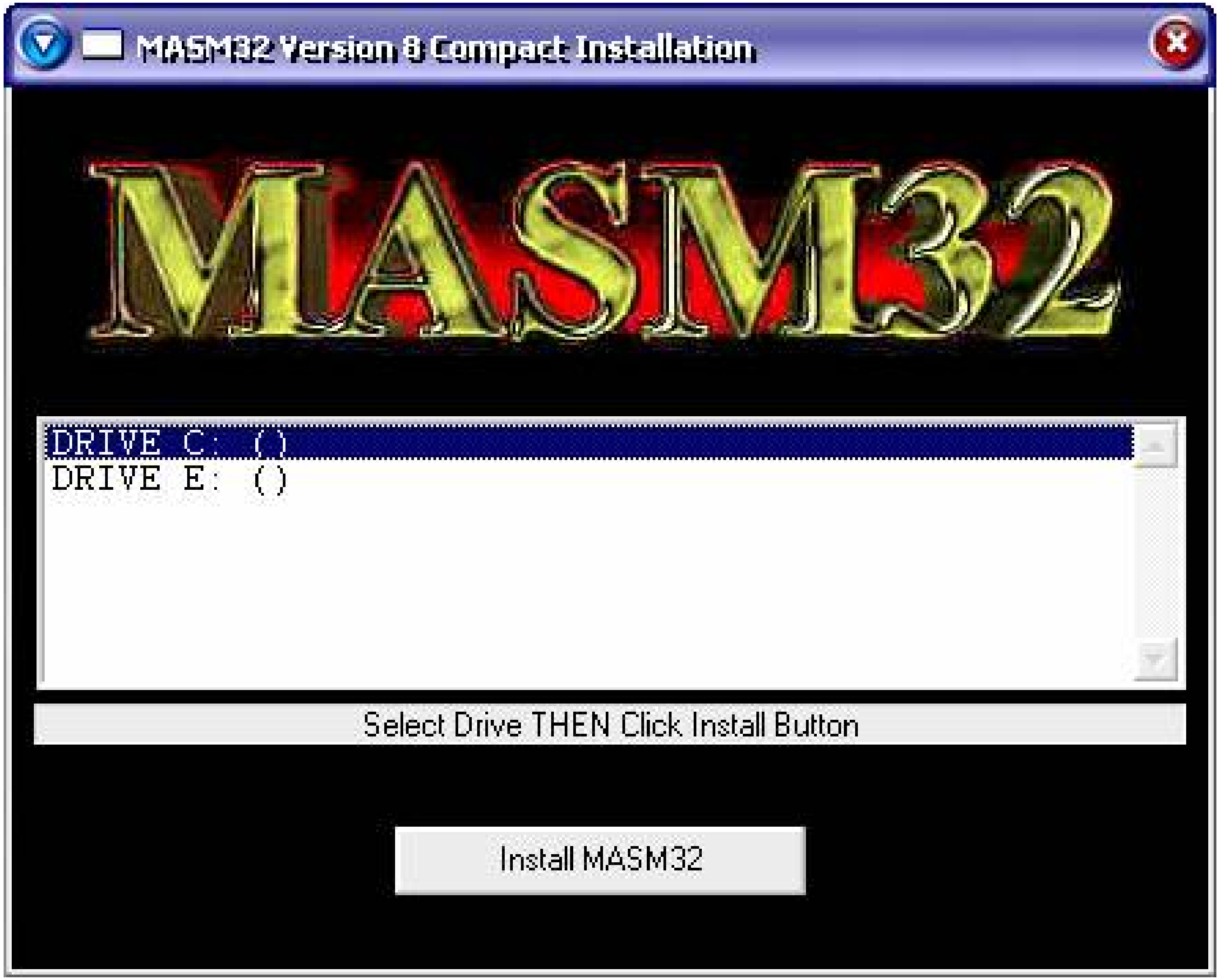
MASM CODEVIEW TUTORIALS

- **1. Download the package MASM 6.14**
- **2. Unzip the package to a folder name MASM32 or MASM 6.14 anywhere you like.**

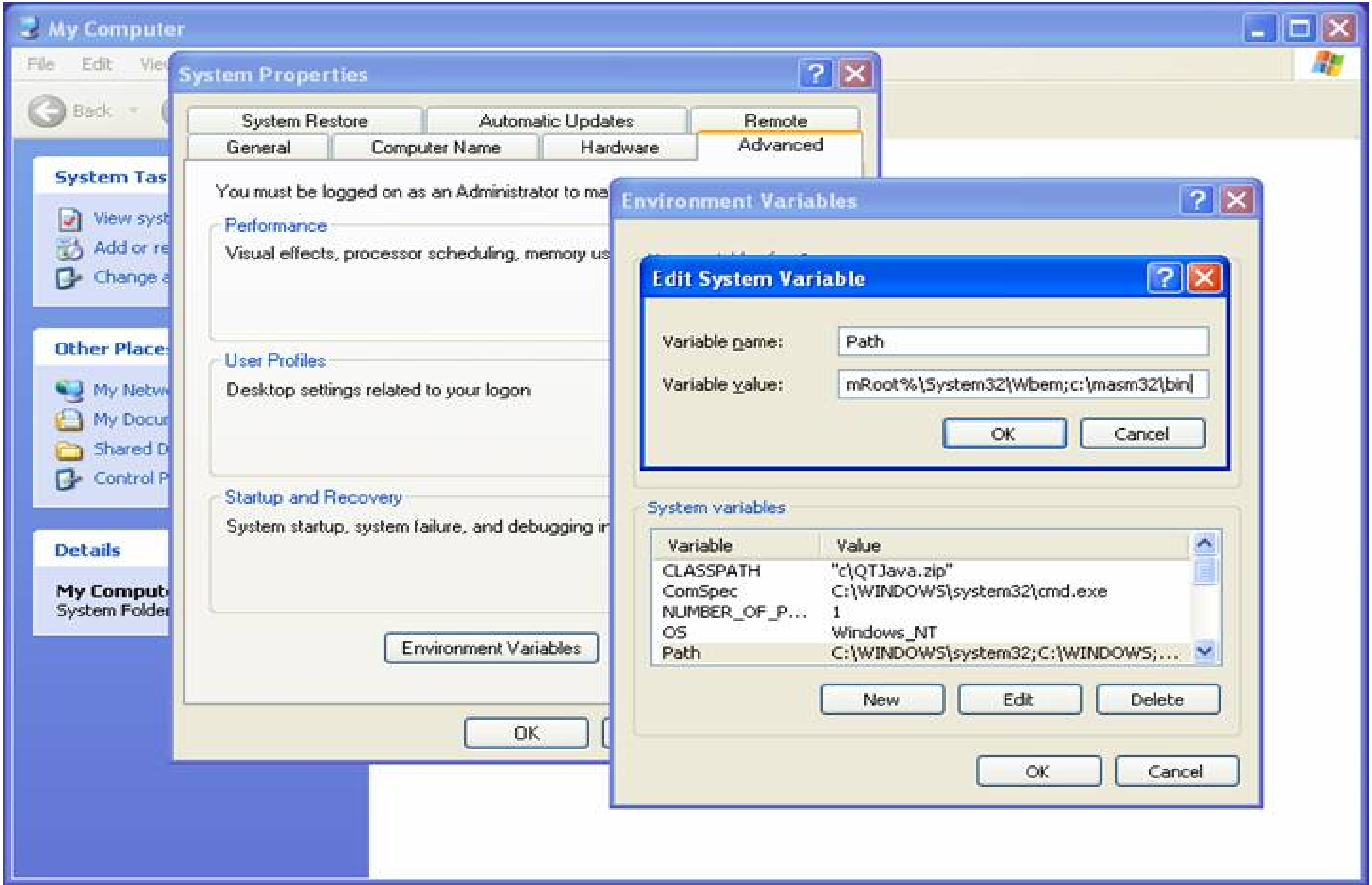


MASM32 Version 8 Compact Installation

DRIVE C: ()
DRIVE E: ()

Select Drive THEN Click Install Button

3. **Set the path to the compiler. Open “My computer”, right click and select “Properties”. Select “Advanced” -> “Environment variables”->”Path”. Click “Edit” and add “;c:\masm32\bin” to the path**



You must be logged on as an Administrator to ma

Visual effects, processor scheduling, memory us

Desktop settings related to your logon

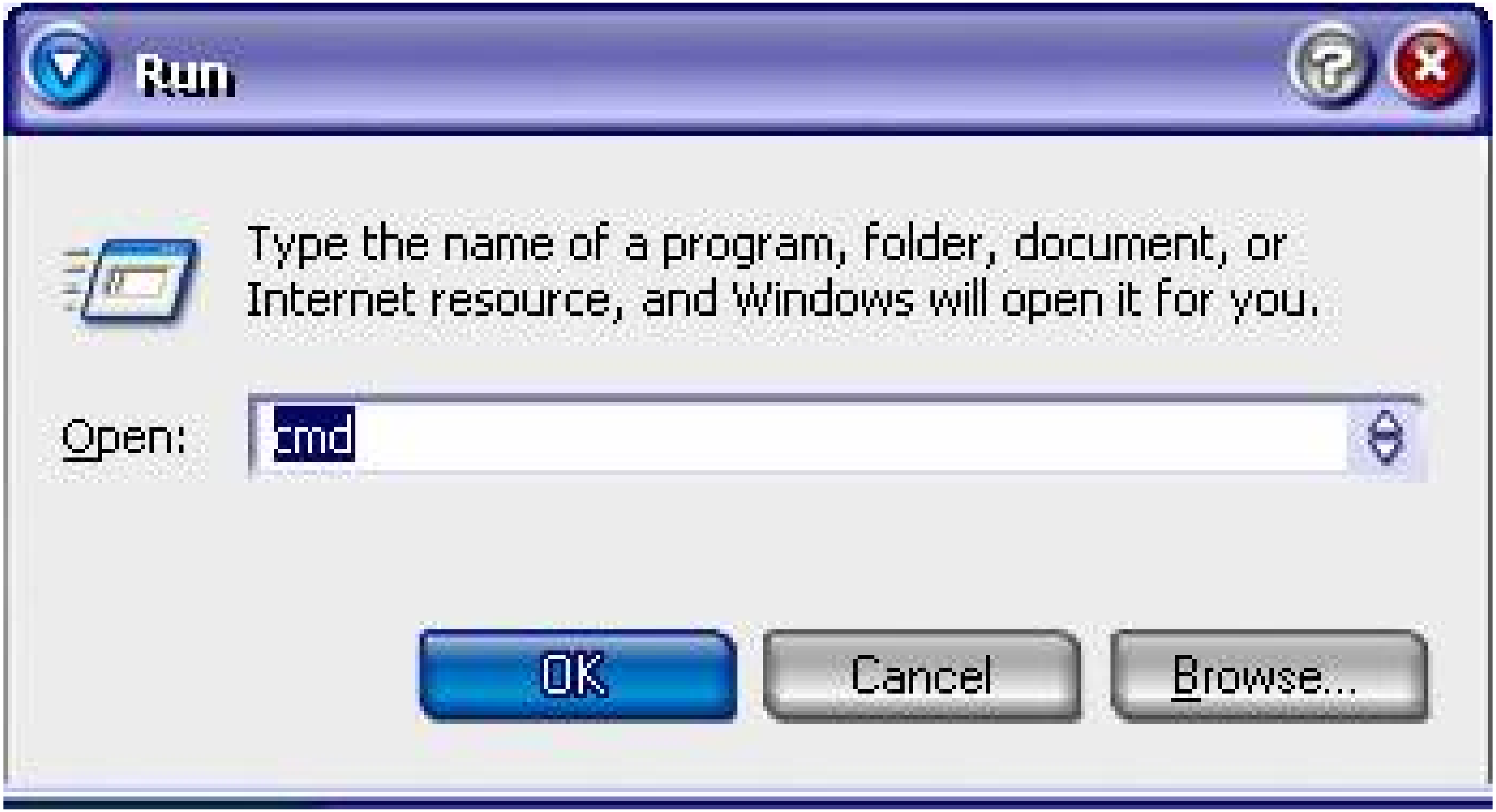
System startup, system failure, and debugging ir

Variable name: Path

Variable value: mRoot%\System32\Wbem;c:\masm32\bin

Variable	Value
CLASSPATH	"c\QTJava.zip"
ComSpec	C:\WINDOWS\system32\cmd.exe
NUMBER_OF_P...	1
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;...

4. **Check the installation by opening the command prompt window (Start->Run->cmd)**

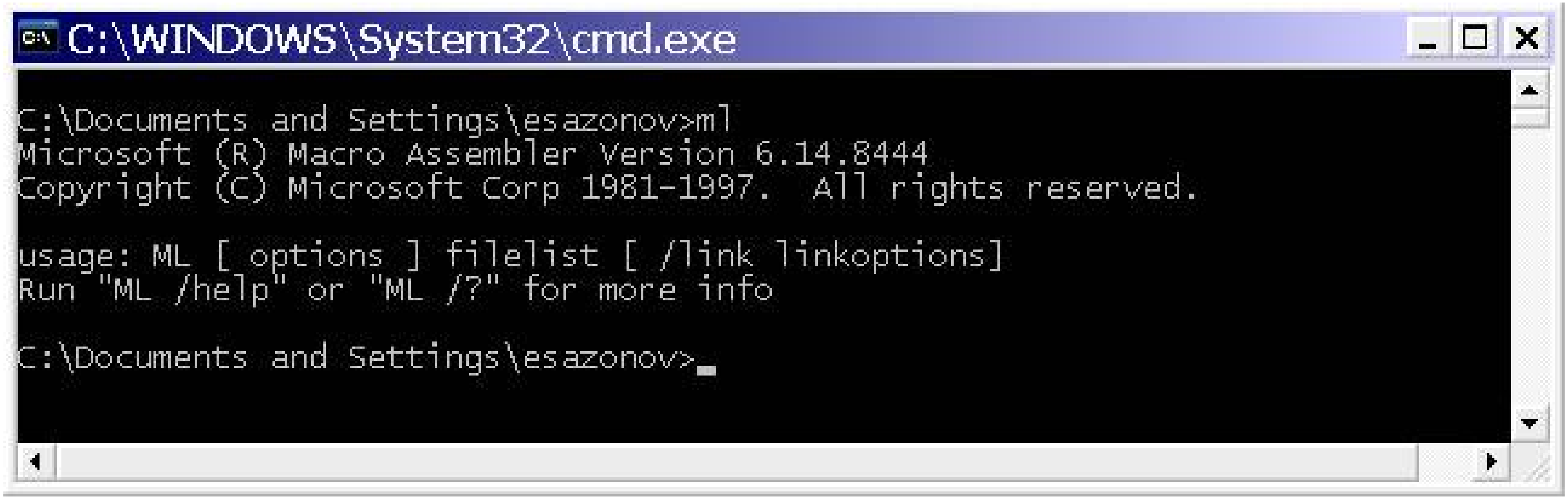


Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: cmd

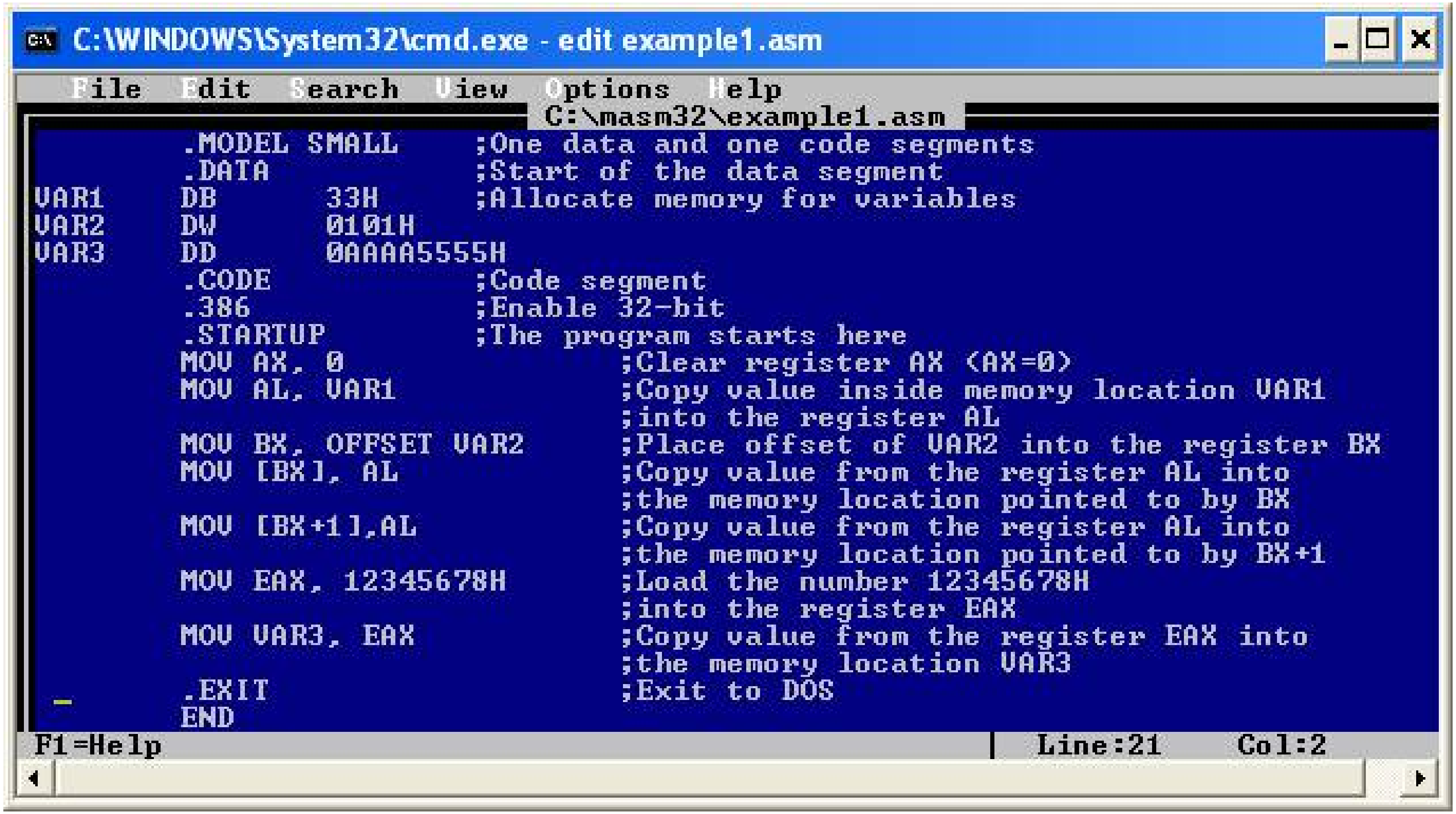
- **and typing ML at the command prompt**

cmd.exe



```
C:\Documents and Settings\esazonov>ml
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997.  All rights reserved.

usage: ML [ options ] filelist [ /link linkoptions]
Run "ML /help" or "ML /?" for more info

C:\Documents and Settings\esazonov>_
```

- Now 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”



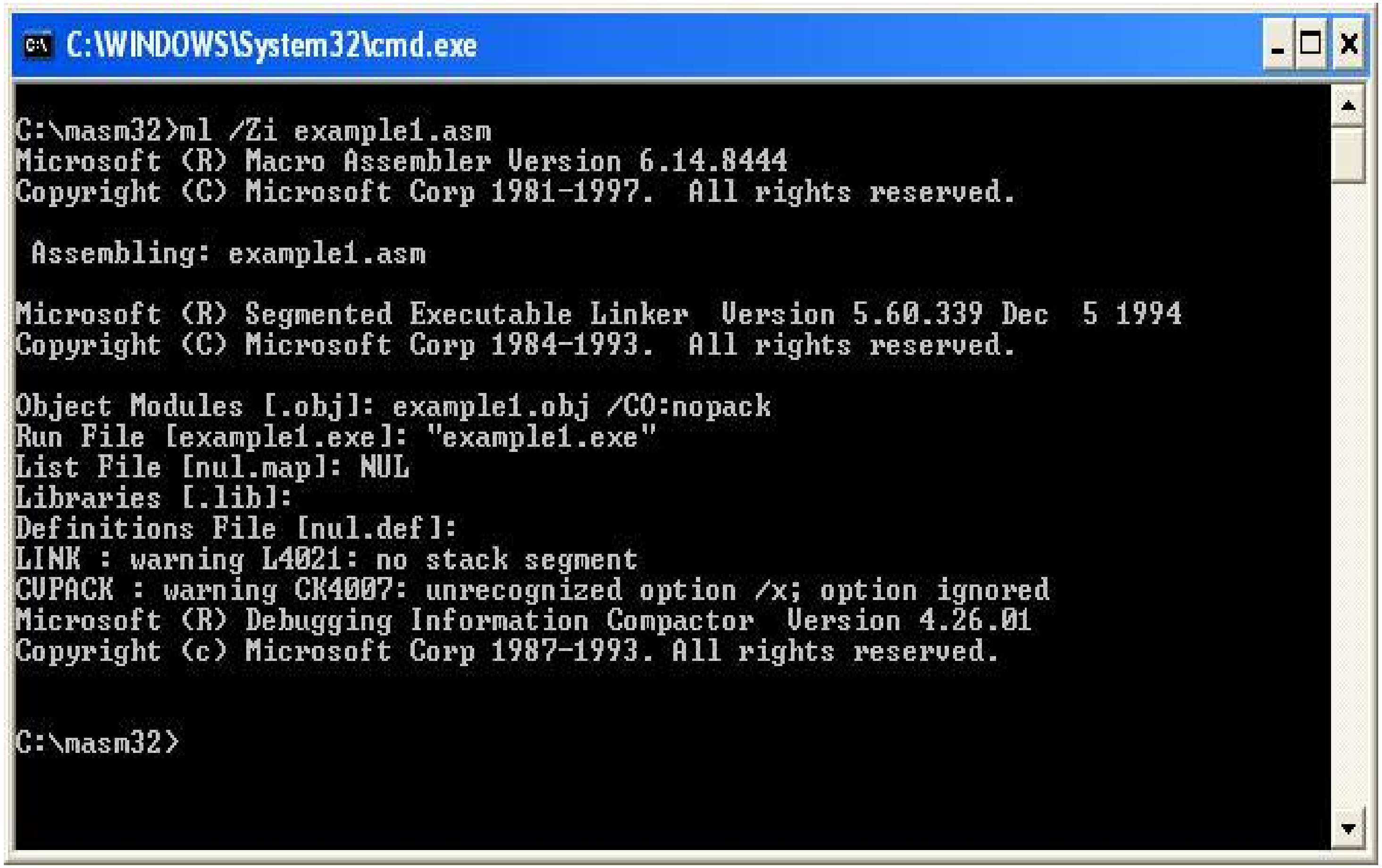
```
        .MODEL SMALL    ;One data and one code segments
        .DATA           ;Start of the data segment
VAR1    DB      33H     ;Allocate memory for variables
VAR2    DW      0101H
VAR3    DD      0AAAA5555H
        .CODE           ;Code segment
        .386            ;Enable 32-bit
        .STARTUP        ;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
```

8. Compile and link the assembly file by issuing "ml /Zi example1.asm"

Notes: The letter Z must be capital.

Different way: masm/zi **example1**; ***Compiling***

link/co **example1**; ***Linking***



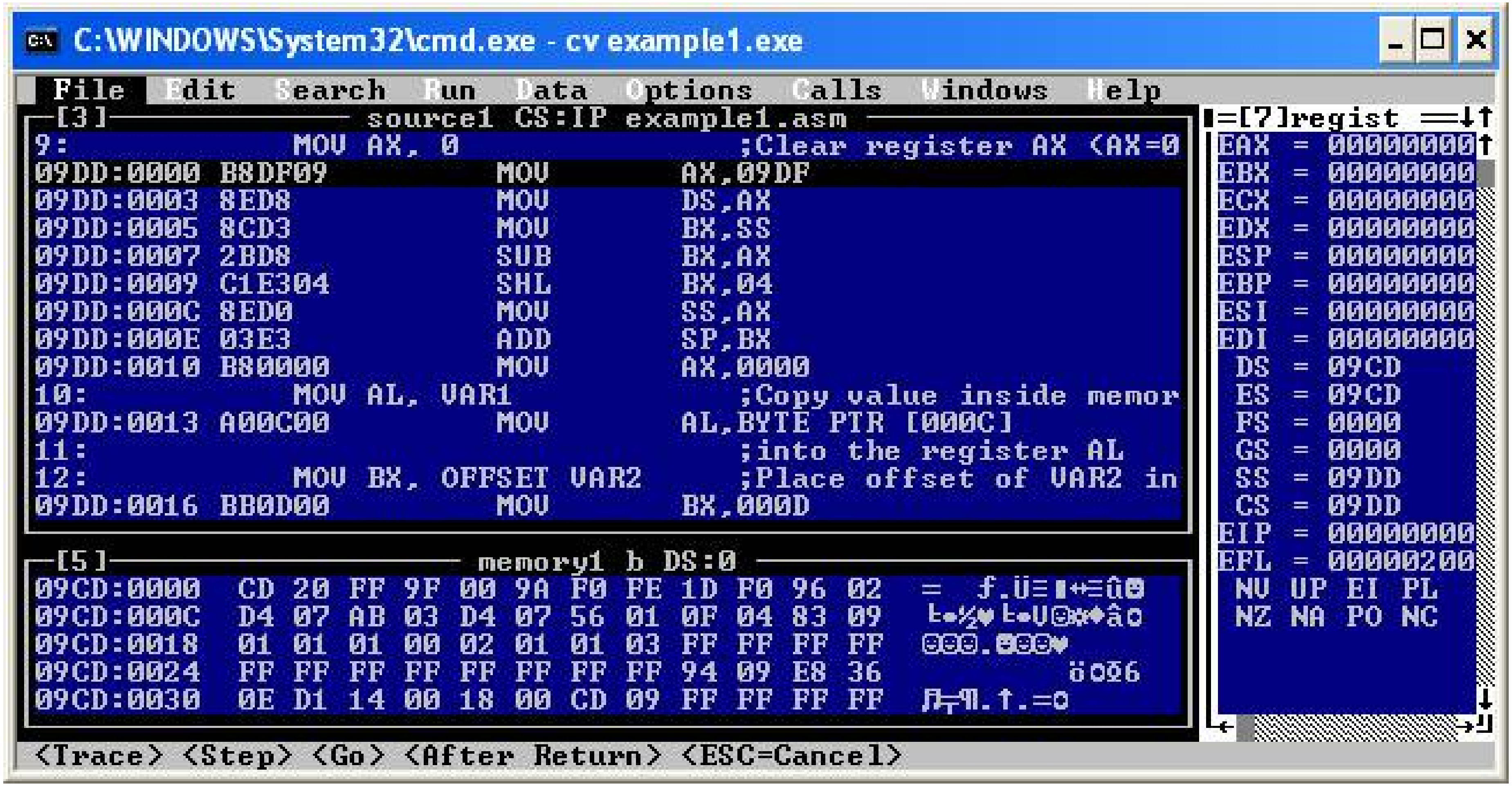
```
C:\masm32>ml /Zi example1.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997.  All rights reserved.

 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>
```

Linking is the final stage of compilation. It takes one or more object files or libraries as input and combines them to produce a single (usually executable) file.

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

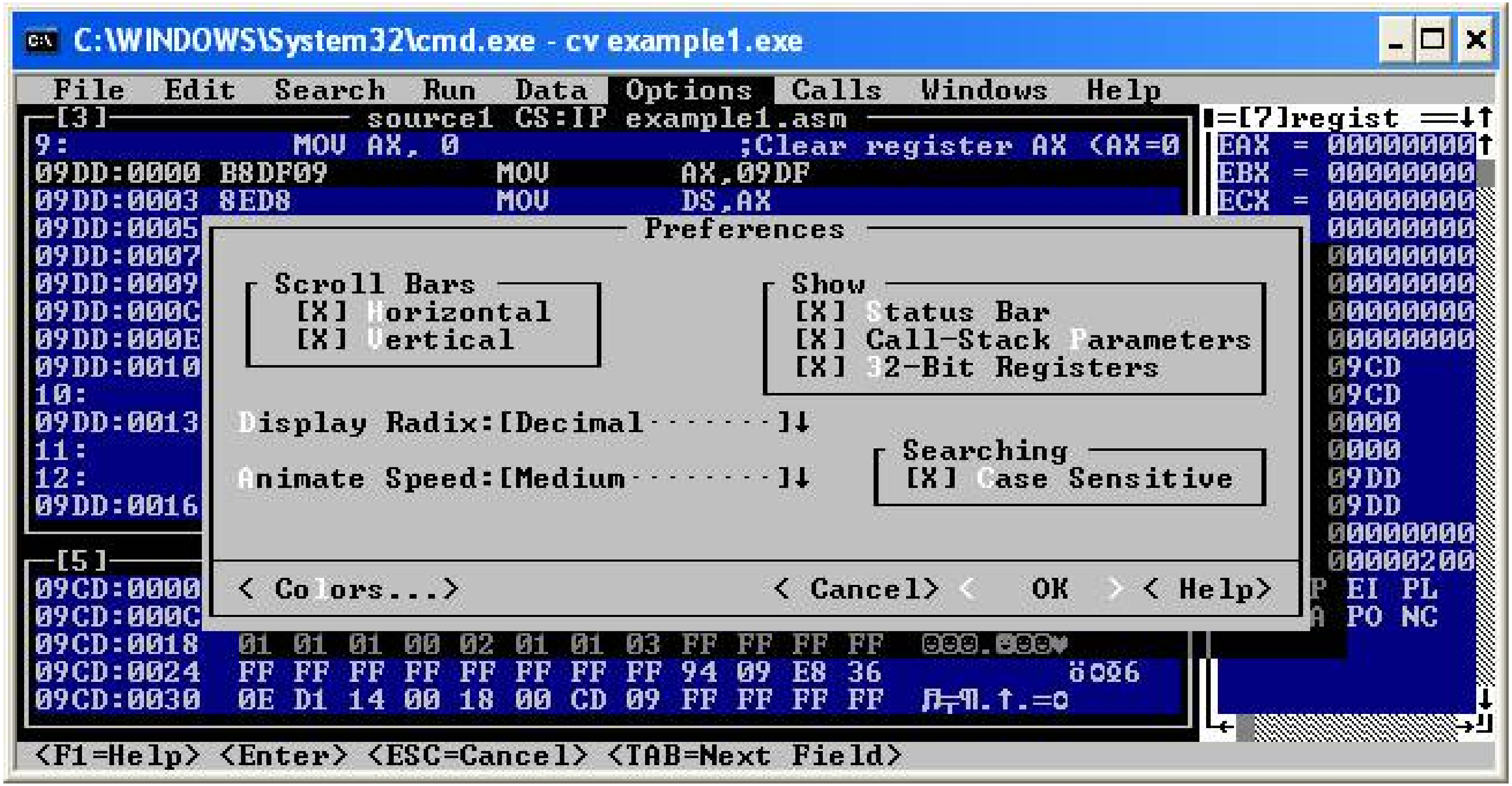
  - Code 1   - Registers   - Memory 1

- 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  =  ƒ.Ü≡▮↔≡û☻
09CD:000C  D4 07 AB 03 D4 07 56 01 0F 04 83 09  ╘•½♥╘•V☺☼♦â○
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          ö○Φ6
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
```

- Now lets 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
-[3]--------------- source1 CS:IP example1.asm ----------  =[7]regist ==↓↑
9:               MOV  AX, 0              ;Clear register AX (AX=0    EAX = 00000000
09DD:0000 B8DF09          MOV         AX,09DF                        EBX = 00000000
09DD:0003 8ED8            MOV         DS,AX                          ECX = 00000000
09DD:0005                                                                  00000000
09DD:0007                                                                  00000000
09DD:0009                                                                  00000000
09DD:000C                                                                  00000000
09DD:000E                                                                  00000000
09DD:0010                                                                  09CD
10:                                                                        09CD
09DD:0013                                                                  0000
11:                                                                        0000
12:                                                                        09DD
09DD:0016                                                                  09DD
                                                                           00000000
-[5]-                                                                      00000200
09CD:0000                                                                P EI PL
09CD:000C                                                                A PO NC
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          ö○Φ6
09CD:0030   0E D1 14 00 18 00 CD 09 FF FF FF FF   ♫╤¶.↑.=○
<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

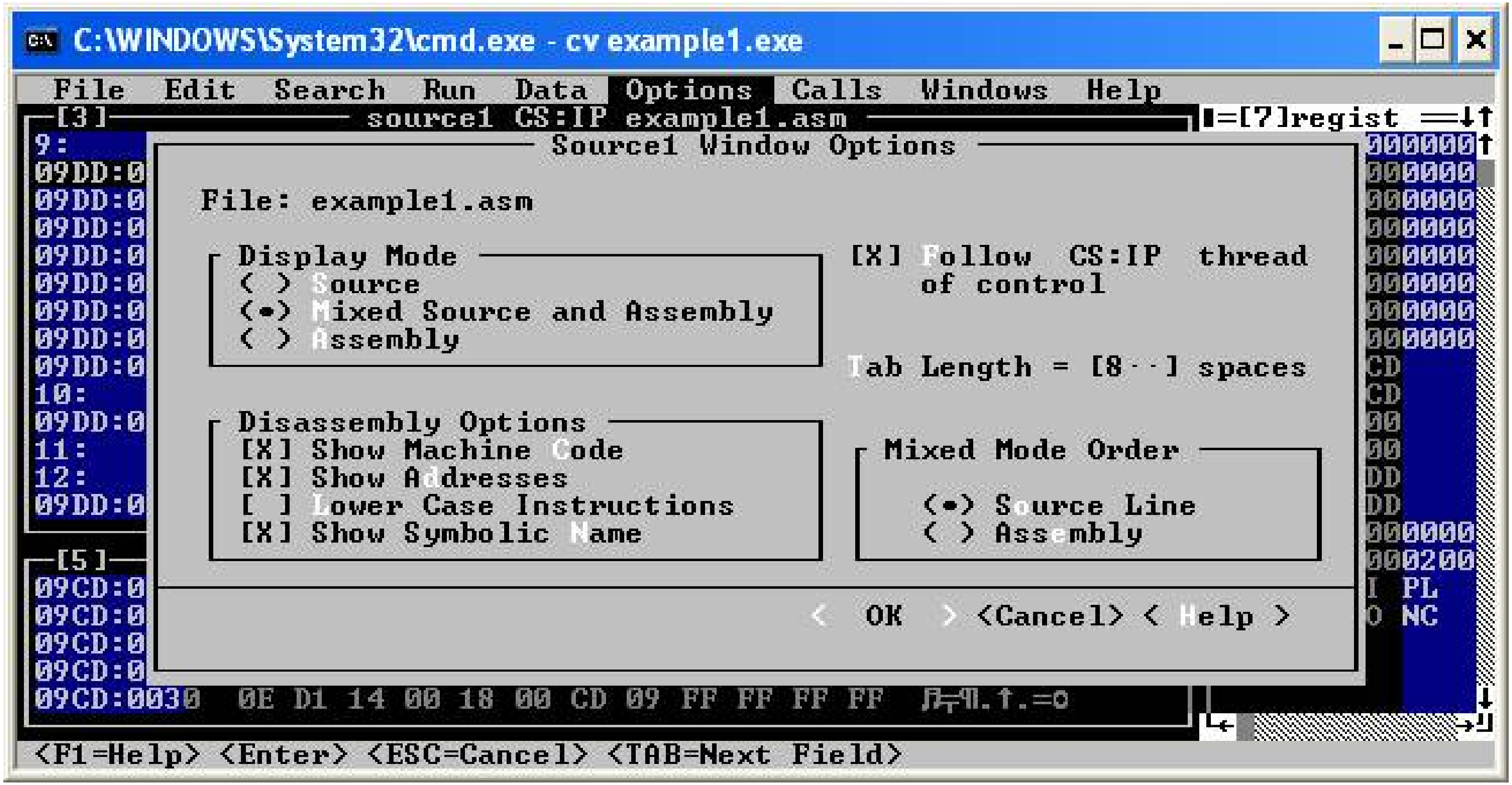
Display Radix:[Decimal·······]↓

Animate Speed:[Medium········]↓

Searching
[X] Case Sensitive

< Colors...>   < Cancel> < OK > < Help>

- **Again, “Alt-O” -> “Source 1 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
[5]
OK
<Cancel>
< Help >

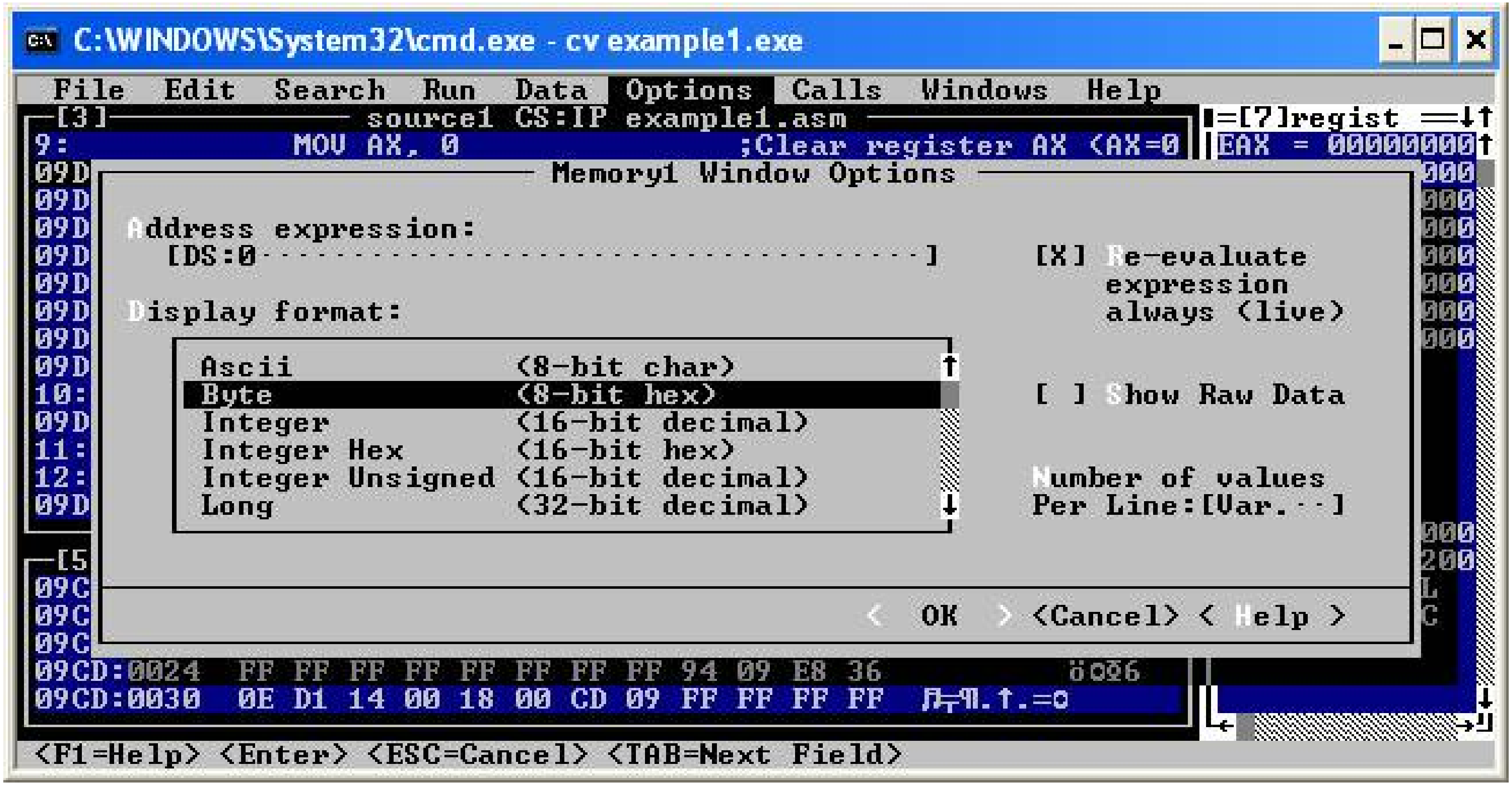
```
9:
09DD:0
09DD:0
09DD:0
09DD:0
09DD:0
09DD:0
09DD:0
09DD:0
09DD:0
10:
09DD:0
11:
12:
09DD:0

09CD:0
09CD:0
09CD:0
09CD:0
09CD:0030  0E D1 14 00 18 00 CD 09 FF FF FF FF  ╒╤¶.↑.=☼
```

```
                    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 ───────────────┐
 │ [X] Show Machine Code              │   ┌ Mixed Mode Order ──────┐
 │ [X] Show Addresses                 │   │                        │
 │ [ ] Lower Case Instructions        │   │   (•) Source Line      │
 │ [X] Show Symbolic Name             │   │   ( ) Assembly         │
 └────────────────────────────────────┘   └────────────────────────┘

                                    <  OK  > <Cancel> < Help >
```

```
000000
000000
000000
000000
000000
000000
000000
000000
CD
CD
00
00
DD
DD
000000
000200
I PL
O NC
```

```
<F1=Help> <Enter> <ESC=Cancel> <TAB=Next Field>
```

- “Alt-O” - >”Memory 1 window”
- Make sure to mark X over Re-evaluate expression always field.


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
EAX = 00000000
[5
OK
<Cancel>
< Help >
<F1=Help> <Enter> <ESC=Cancel> <TAB=Next Field>

```
9:            MOV  AX, 0                      ;Clear register AX (AX=0
```

Memory1 Window Options

Address expression:
[DS:0------------------------------------]

[X] Re-evaluate expression always (live)

Display format:

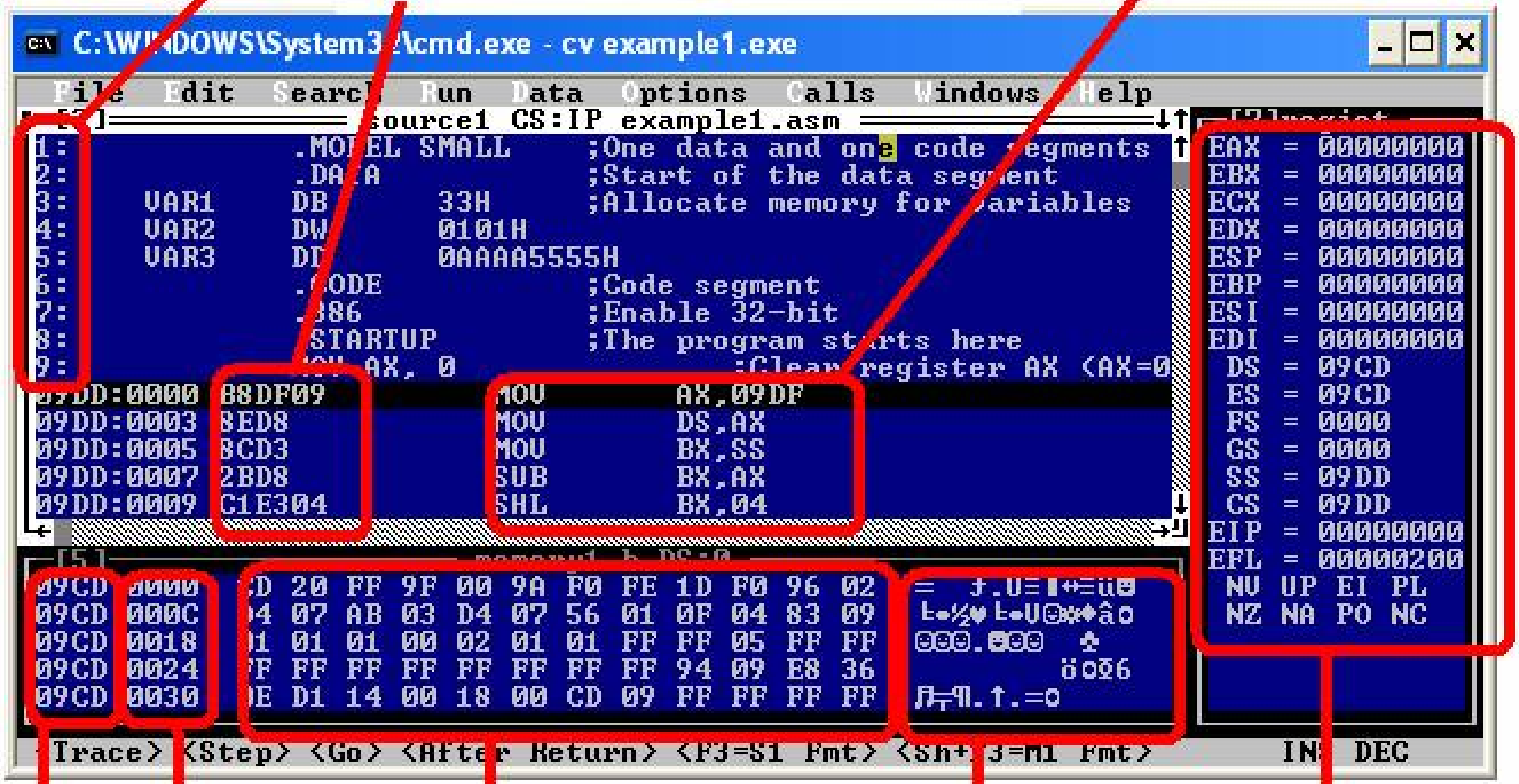
```
Ascii             (8-bit char)
Byte              (8-bit hex)
Integer           (16-bit decimal)
Integer Hex       (16-bit hex)
Integer Unsigned  (16-bit decimal)
Long              (32-bit decimal)
```

[ ] Show Raw Data

Number of values Per Line:[Var.··]

```
09CD:0024  FF FF FF FF FF FF FF FF 94 09 E8 36         ö◘Φ6
09CD:0030  0E D1 14 00 18 00 CD 09 FF FF FF FF     ♫╤¶.↑.=○
```

10. Lets look at the 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] register
[5] memory1 b DS:0

```
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
```

```
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
```

```
09CD 0000  CD 20 FF 9F 00 9A F0 FE 1D F0 96 02  =  ƒ.U=▮↔=ü☻
09CD 000C  04 07 AB 03 D4 07 56 01 0F 04 83 09  ╘•½♥╘•V☺☼♦â○
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          ö○Φ6
09CD 0030  0E D1 14 00 18 00 CD 09 FF FF FF FF  ♫╤¶.↑.=○
```

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

Line # from the source file

Instructions in memory

Instruction mnemonics

Registers

Value of the DS

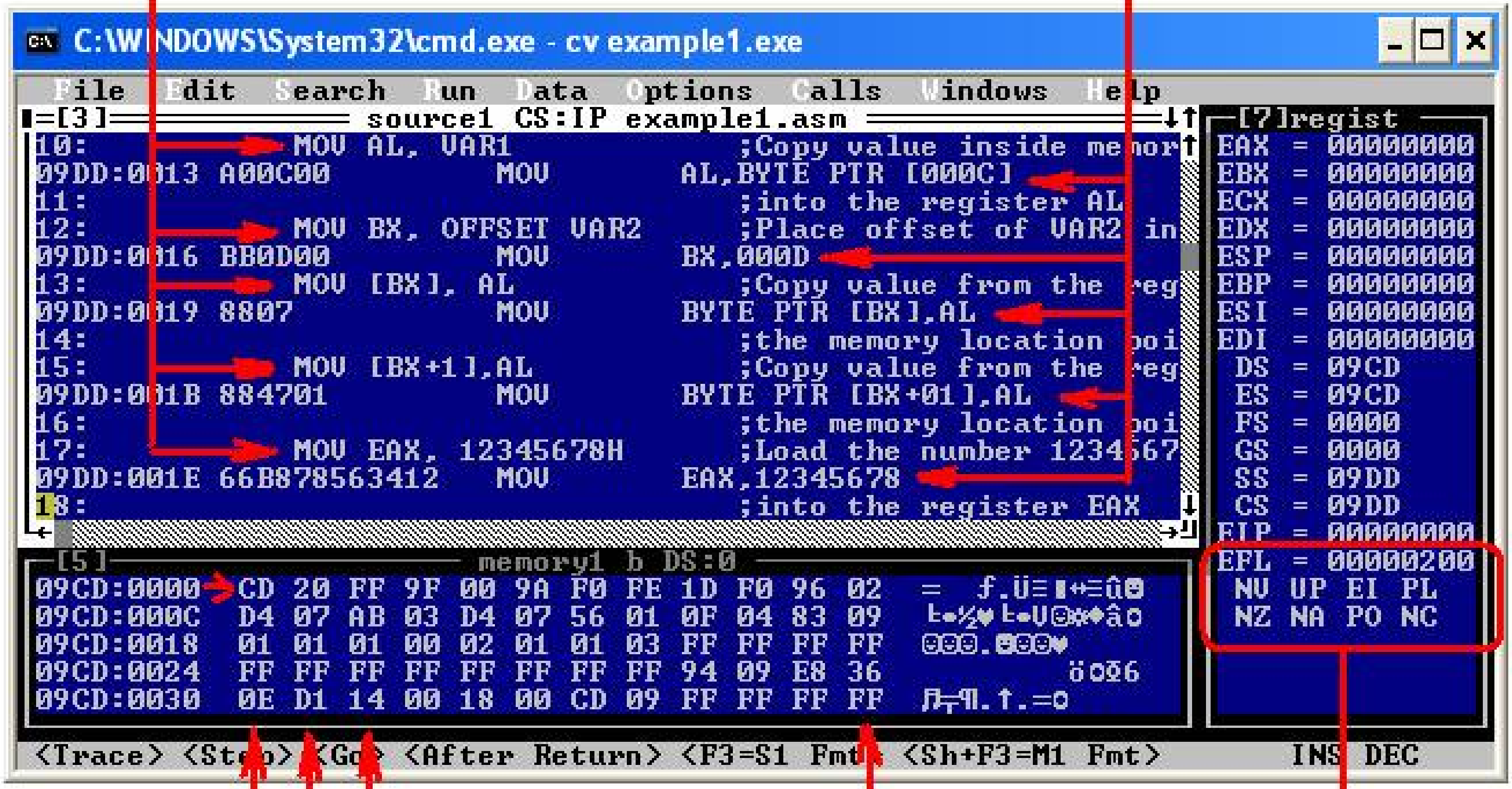
Offset

Memory content shown as byte-size hexadecimal numbers

The same memory but shown in ASCII

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
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
```

```
[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
```

```
[5]                memory1 b DS:0
09CD:0000  CD 20 FF 9F 00 9A F0 FE 1D F0 96 02   =  ƒ.Ü≡▮↔≡û☻
09CD:000C  D4 07 AB 03 D4 07 56 01 0F 04 83 09   ╘•½♥╘•V☺☼♦â○
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           ö○Φ6
09CD:0030  0E D1 14 00 18 00 CD 09 FF FF FF FF   ♫╤¶.↑.═○
```

+0

+1

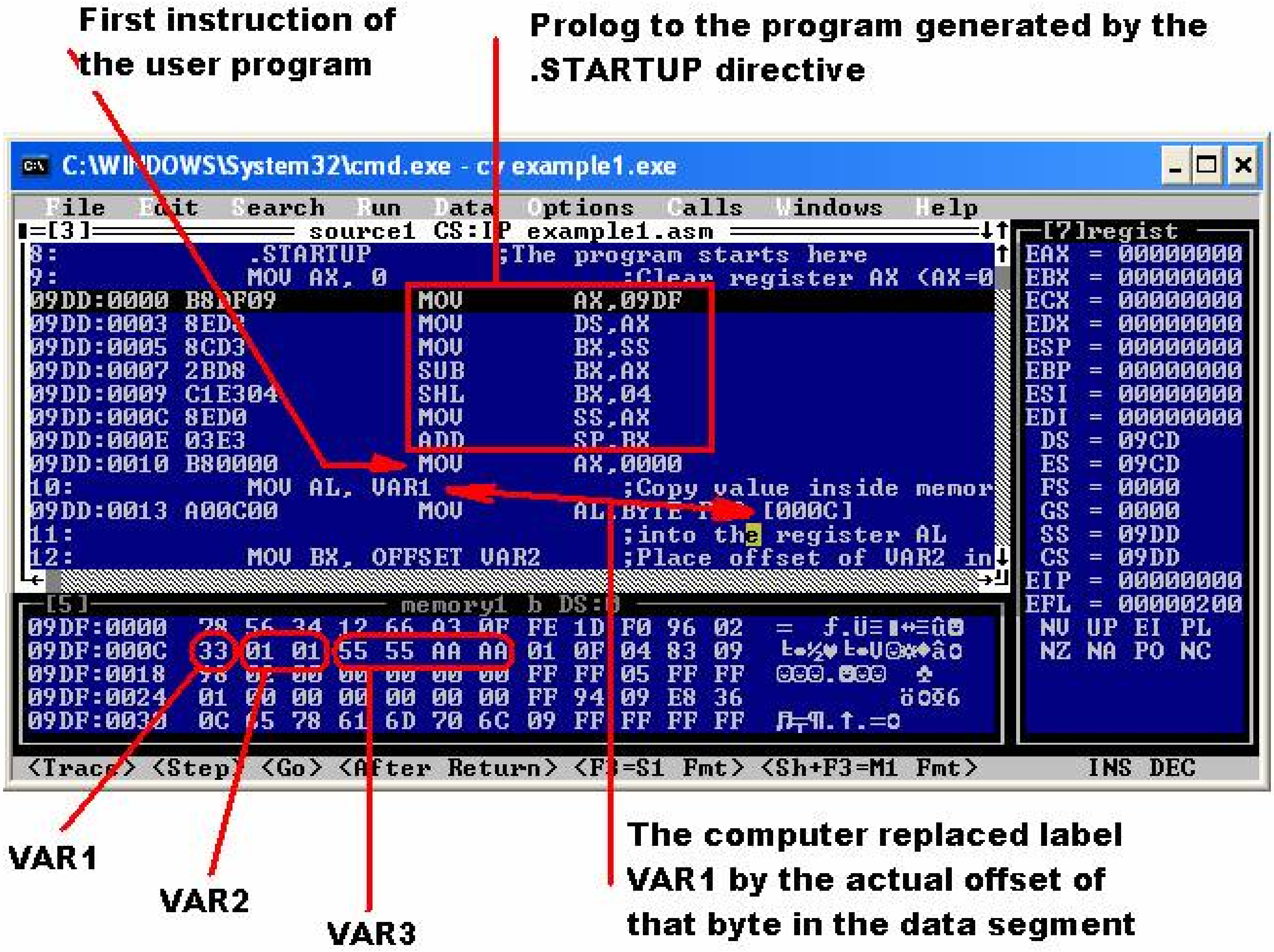
+2

- - -

+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
[5] memory1 b DS:0
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>
INS DEC

```
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
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
```

```
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
```

```
09DF:0000  78 56 34 12 66 A3 0F FE 1D F0 96 02
09DF:000C  33 01 01 55 55 AA AA 01 0F 04 83 09
09DF:0018  78 02 00 00 00 00 00 FF FF 05 FF FF
09DF:0024  01 00 00 00 00 00 00 FF 94 09 E8 36
09DF:0030  0C 65 78 61 6D 70 6C 09 FF FF FF FF
```

VAR1

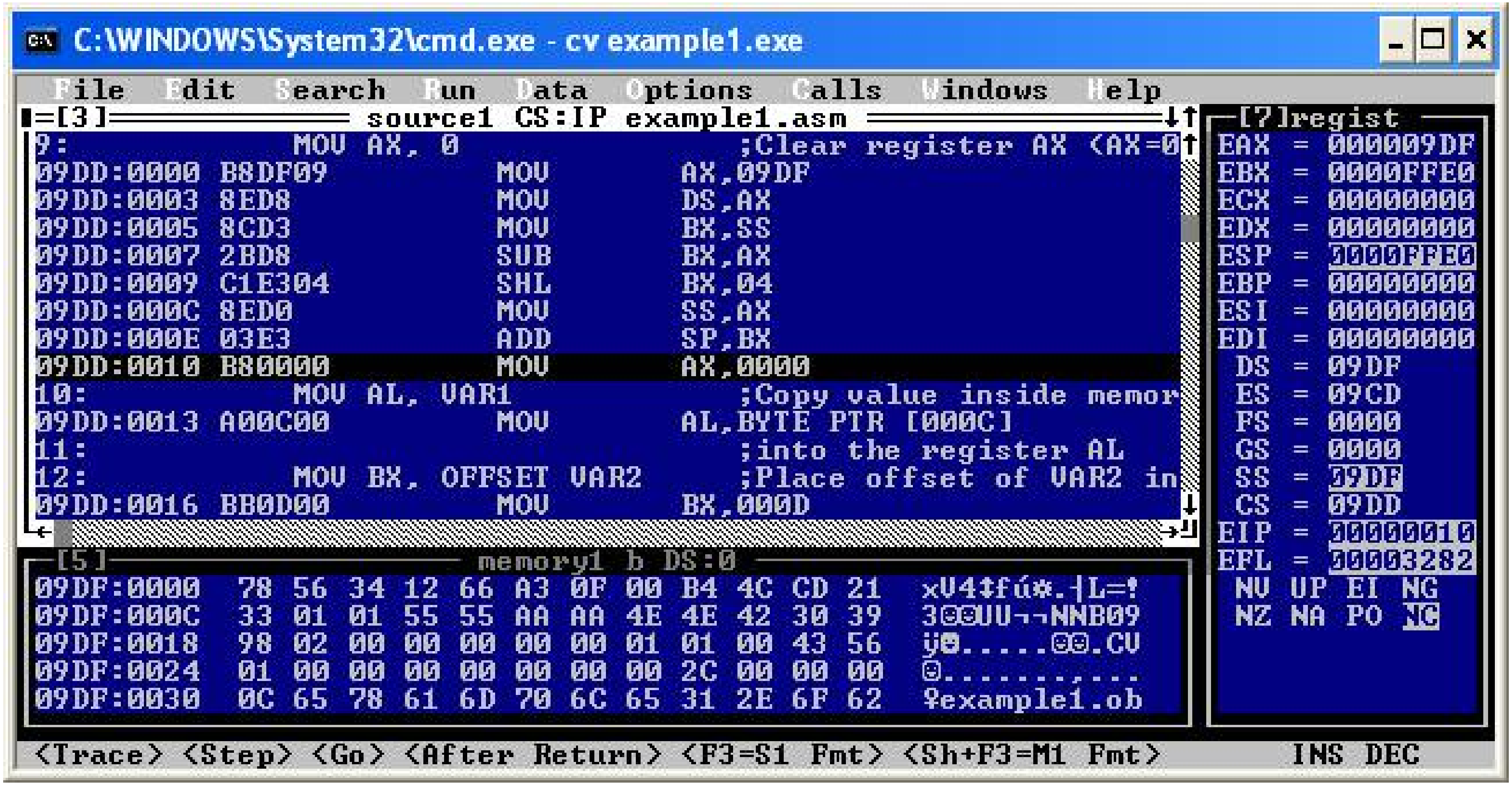
VAR2

VAR3

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

Now lets step through the program and observe execution of each instruction.

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



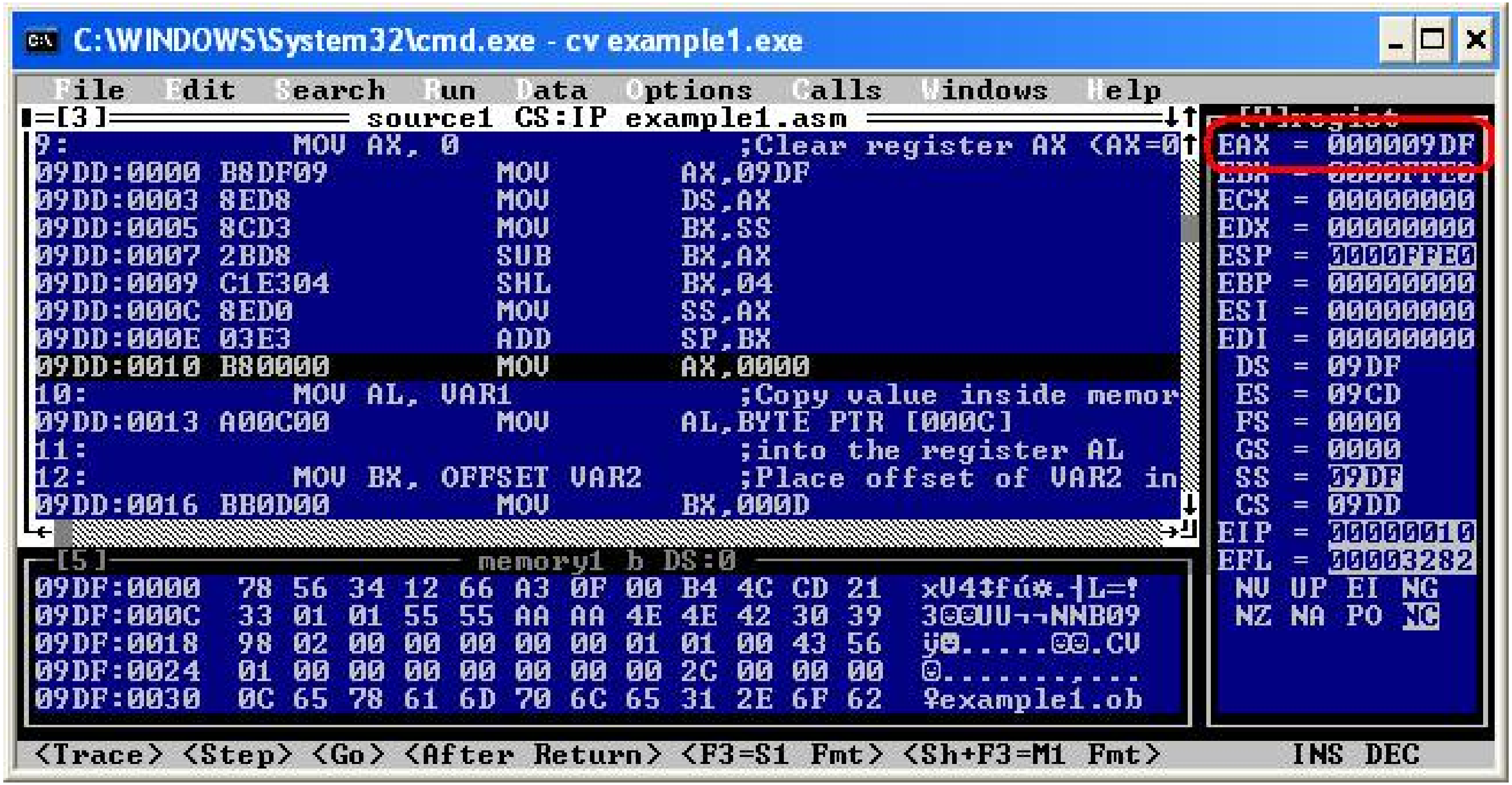
C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
[3] source1 CS:IP example1.asm
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
```

```
[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
```

```
[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
```

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

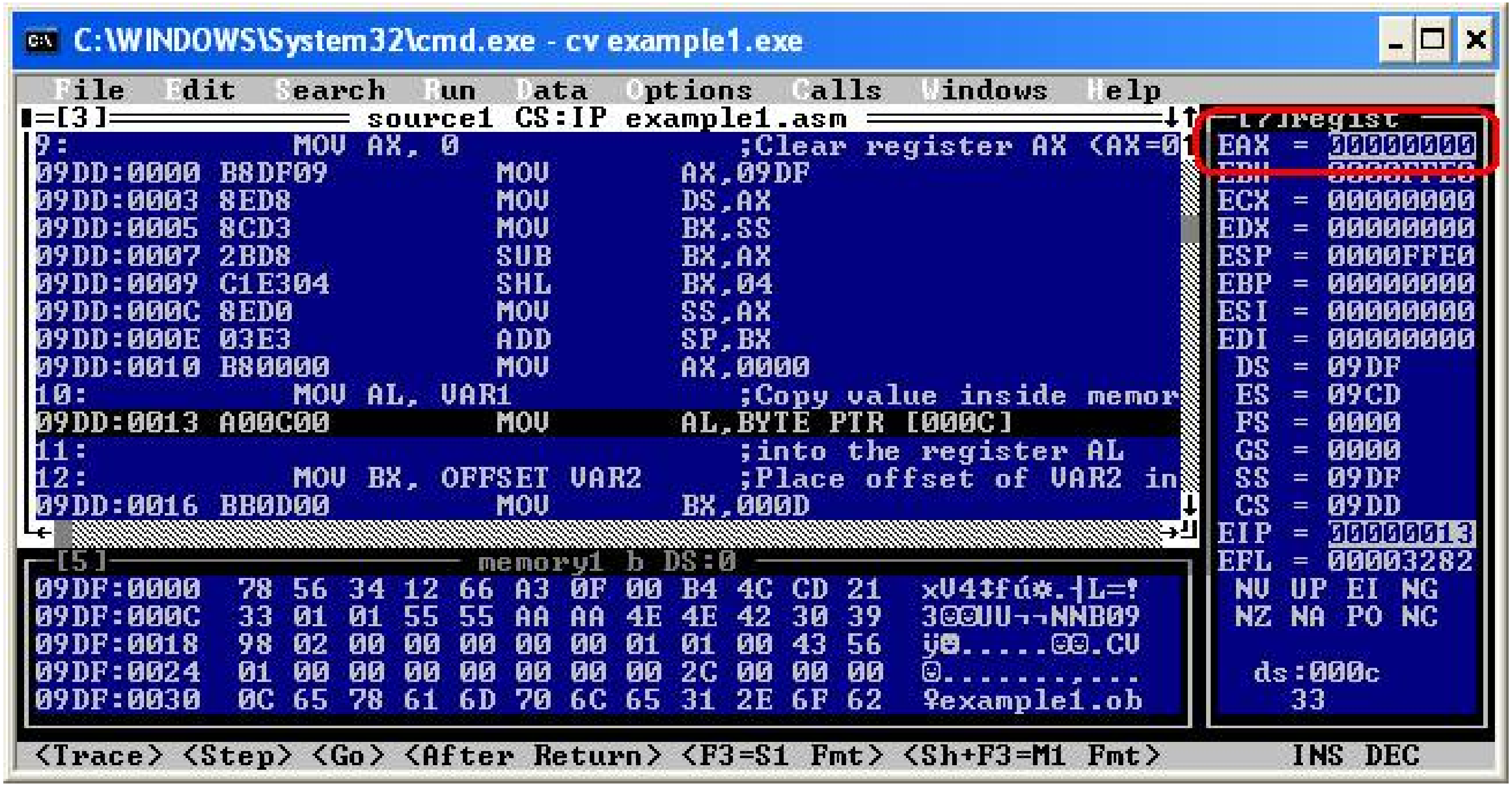

C:\WINDOWS\System32\cmd.exe - cv example1.exe
File
Edit
Search
Run
Data
Options
Calls
Windows
Help
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>
INS DEC

```
[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
```

```
[7]register
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
```

```
[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
```

- 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.



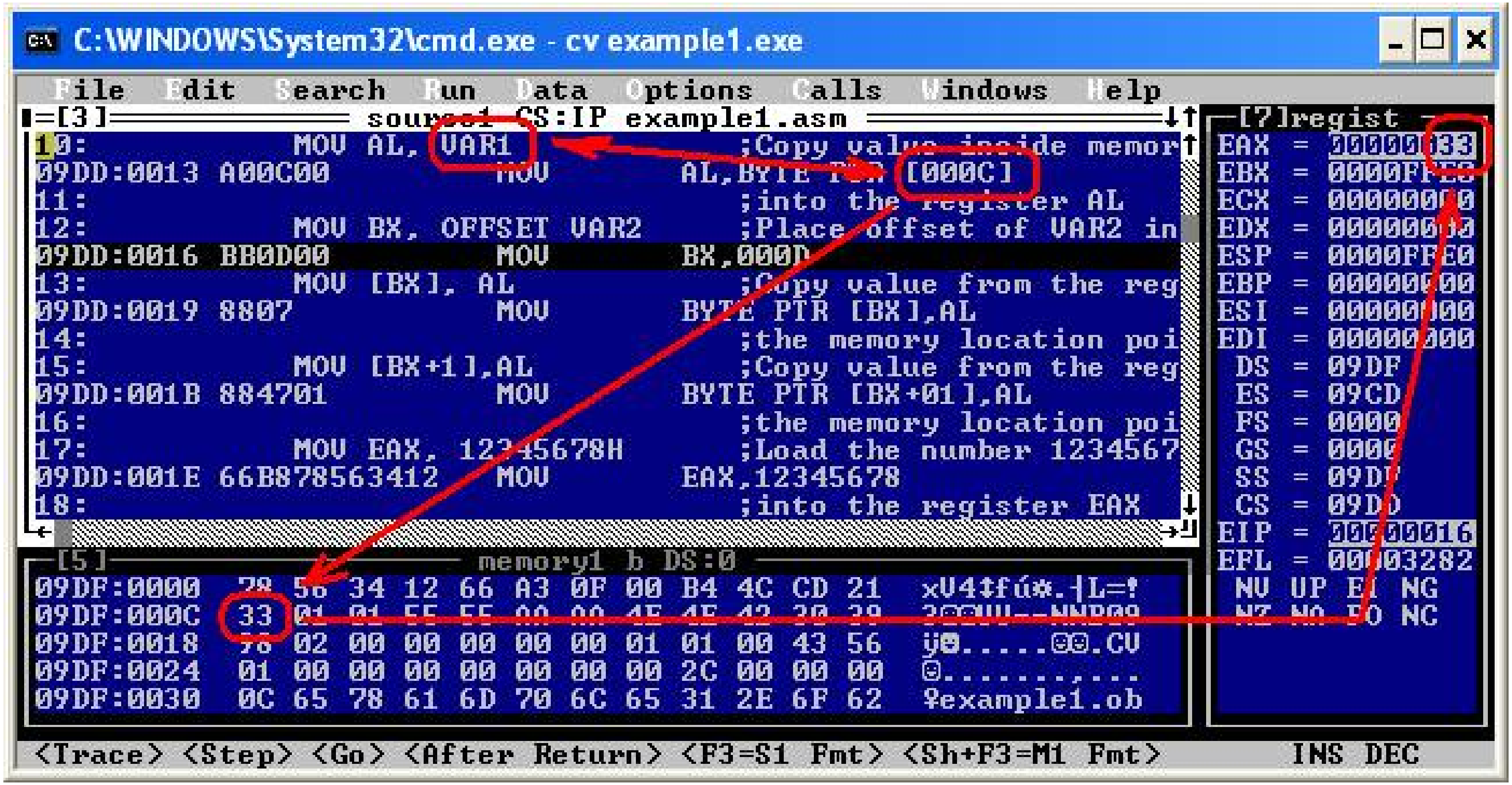
C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
[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
```

```
[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
```

```
[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
```

- 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.

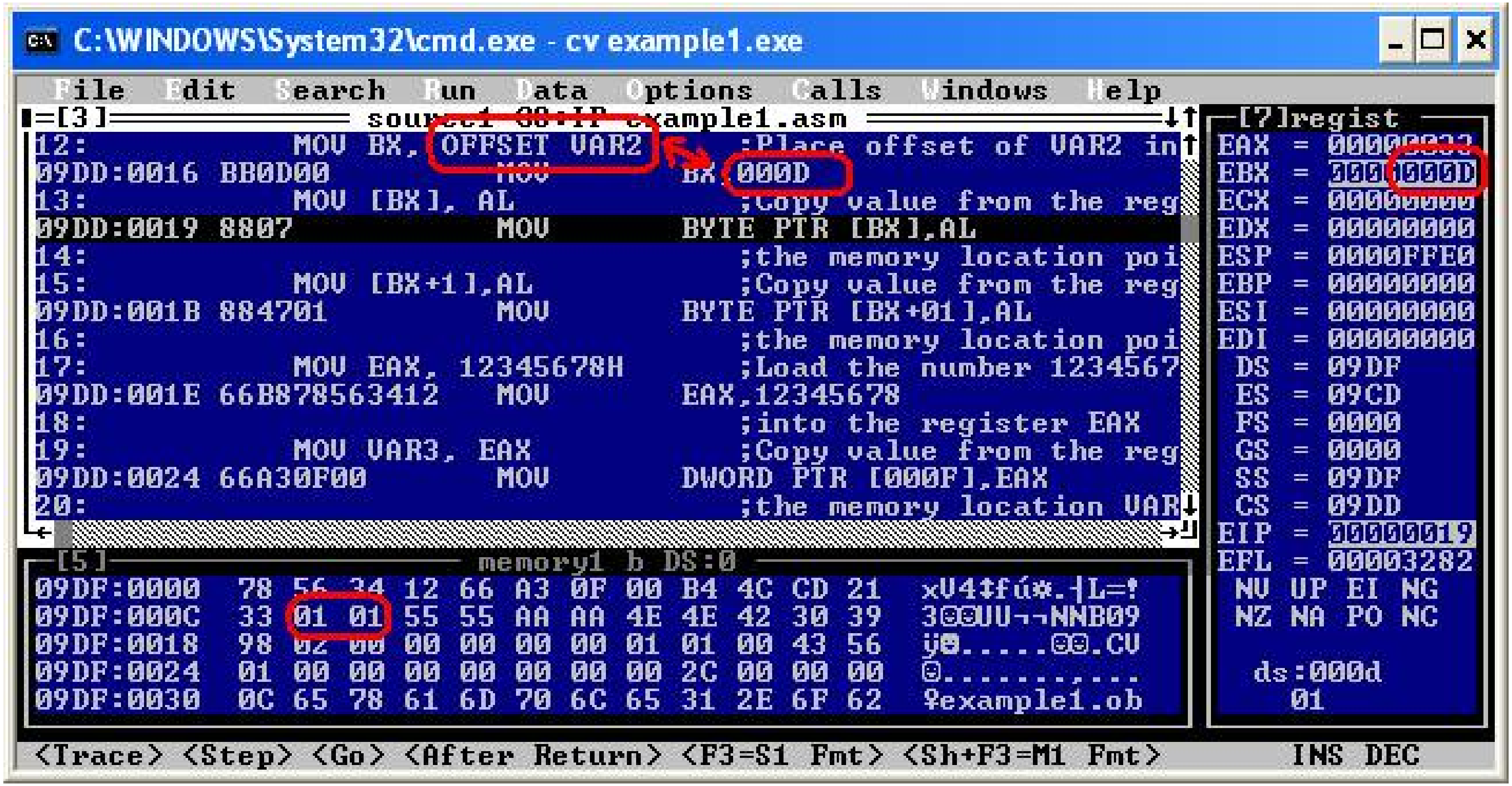

C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
=[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 FF FF 00 00 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
```

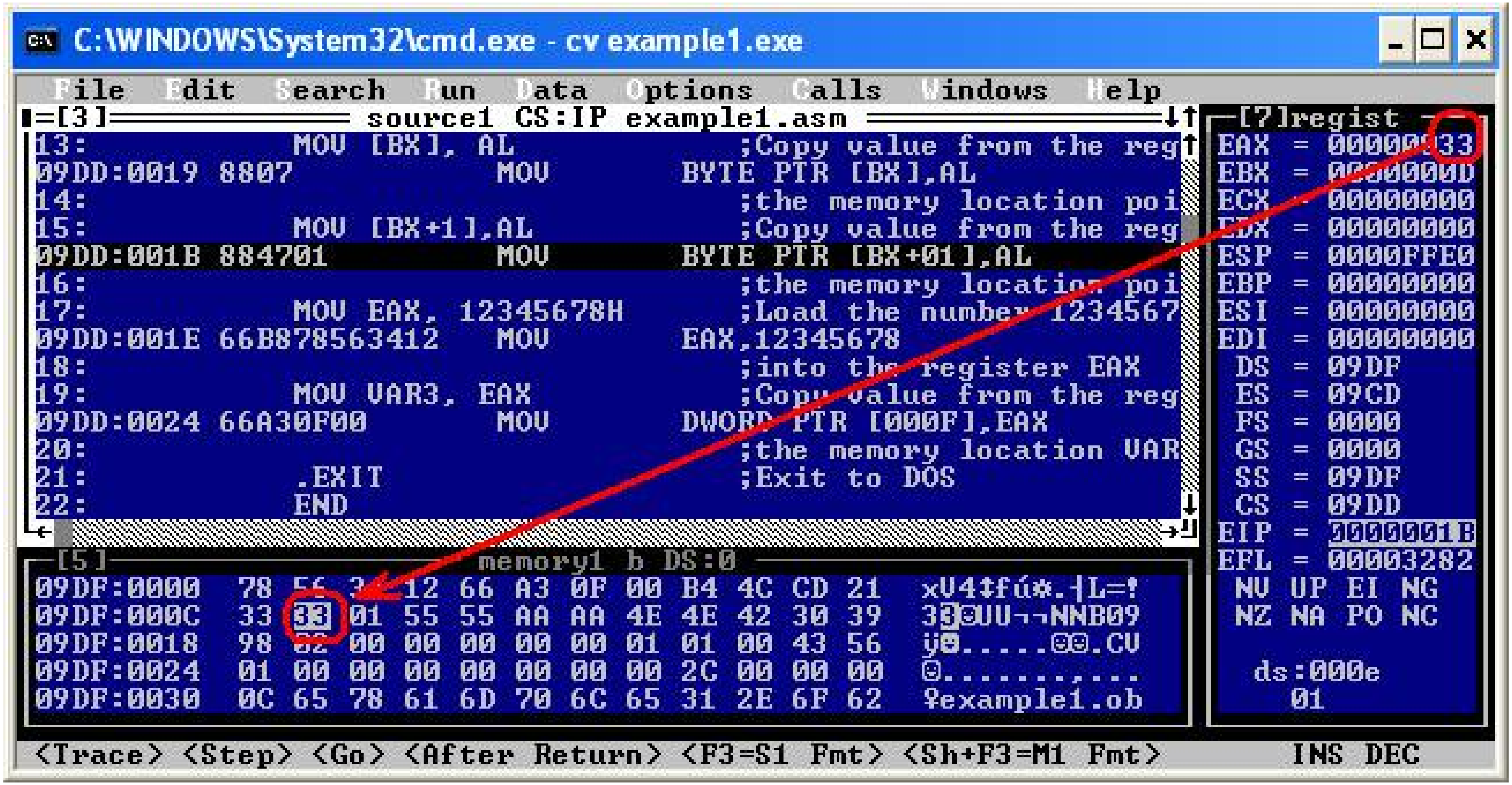
- The next instruction is "MOV BX, OFFSET VAR2". VAR2 follows VAR1 in memory and has offset of 000DH. This is the value that will be placed into the BX upon execution of this instruction. Press "F10" to execute.



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

```
=[3]========== source1 CS:IP example1.asm ============   -[7]regist ----
12:            MOV BX, OFFSET VAR2      ;Place offset of VAR2 in   EAX = 0000[illegible]
09DD:0016 BB0D00         MOV         BX,000D                       EBX = 0000000D
13:            MOV [BX], AL             ;Copy value from the reg   ECX = 00000000
09DD:0019 8807           MOV         BYTE PTR [BX],AL              EDX = 00000000
14:                                     ;the memory location poi   ESP = 0000FFE0
15:            MOV [BX+1],AL            ;Copy value from the reg   EBP = 00000000
09DD:001B 884701         MOV         BYTE PTR [BX+01],AL           ESI = 00000000
16:                                     ;the memory location poi   EDI = 00000000
17:            MOV EAX, 12345678H       ;Load the number 1234567    DS = 09DF
09DD:001E 66B878563412   MOV         EAX,12345678                   ES = 09CD
18:                                     ;into the register EAX      FS = 0000
19:            MOV VAR3, EAX            ;Copy value from the reg    GS = 0000
09DD:0024 66A30F00       MOV         DWORD PTR [000F],EAX           SS = 09DF
20:                                     ;the memory location VAR    CS = 09DD
                                                                   EIP = 00000019
-[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  ☺.......,...        ds:000d
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
```

- 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.



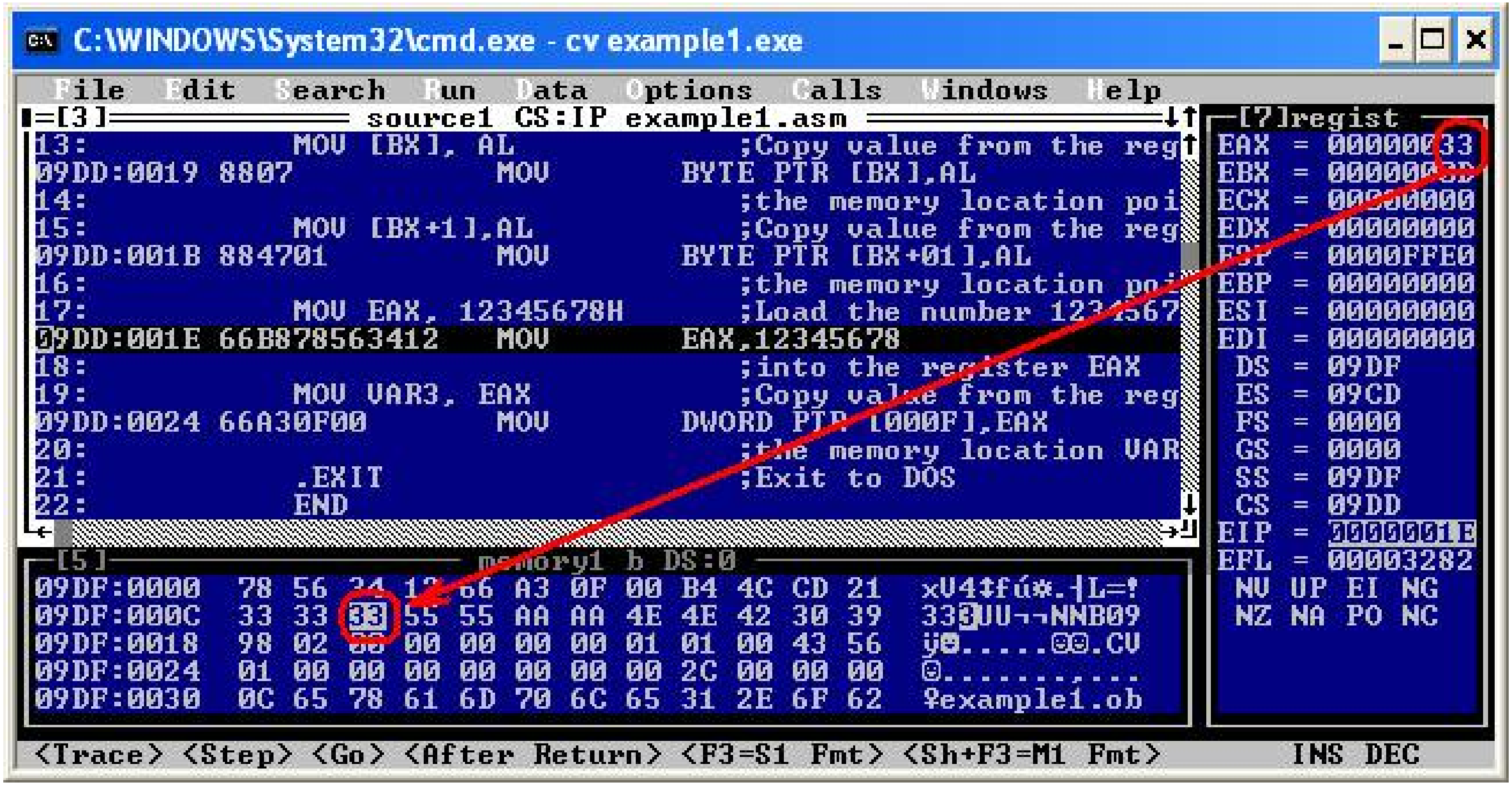
C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
=[3]=========== source1 CS:IP example1.asm ============
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
21:           .EXIT                     ;Exit to DOS
22:           END
```

```
-[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 = 0000001B
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC

 ds:000e
  01
```

```
-[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 33 01 55 55 AA AA 4E 4E 42 30 39  33☺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
```

- 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.

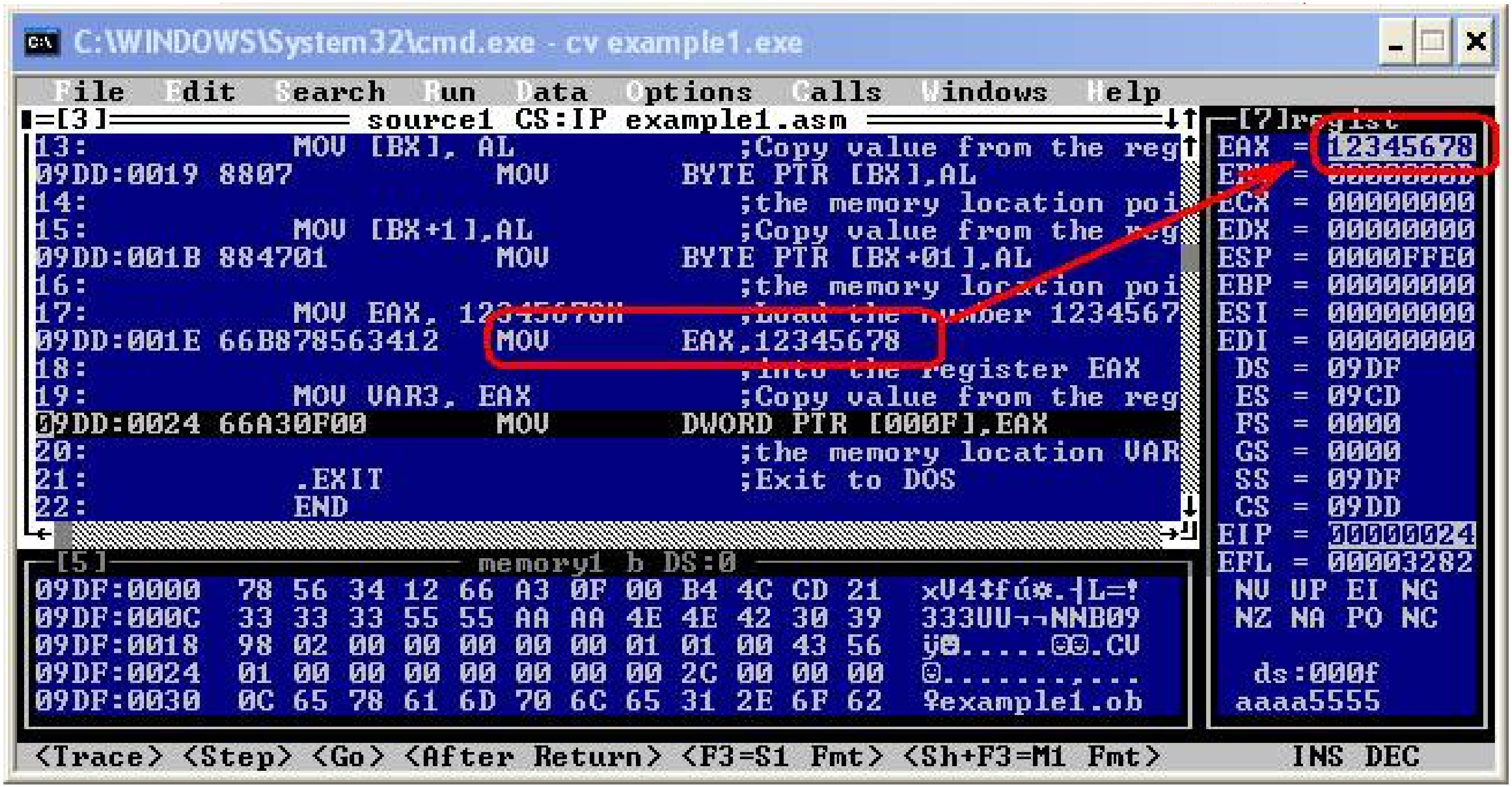

C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt>
INS DEC

```
[3] source1 CS:IP example1.asm
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
21:             .EXIT                     ;Exit to DOS
22:             END
```

```
[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 33 33 55 55 AA AA 4E 4E 42 30 39  333UU¬¬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 = 0000001E
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC
```

- **Instruction "MOV EAX, 12345678H" will place number 12345678H into the register EAX. Press "F10" to execute.**



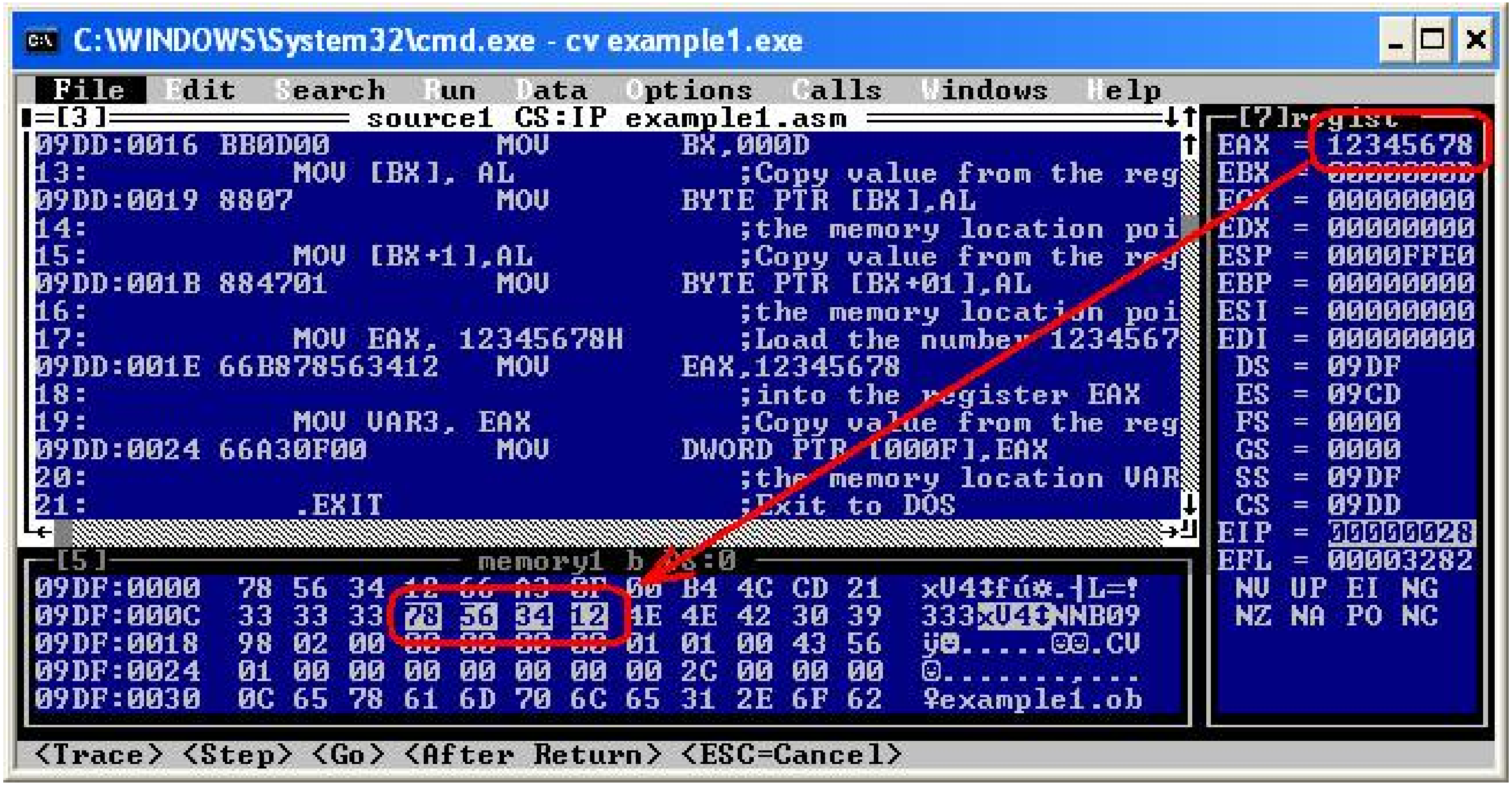
C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
[3]  source1 CS:IP example1.asm
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
21:              .EXIT                          ;Exit to DOS
22:              END
```

```
[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 33 33 55 55 AA AA 4E 4E 42 30 39  333UU¬¬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]register
EAX = 12345678
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 = 00000024
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC

  ds:000f
 aaaa5555
```

- 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.



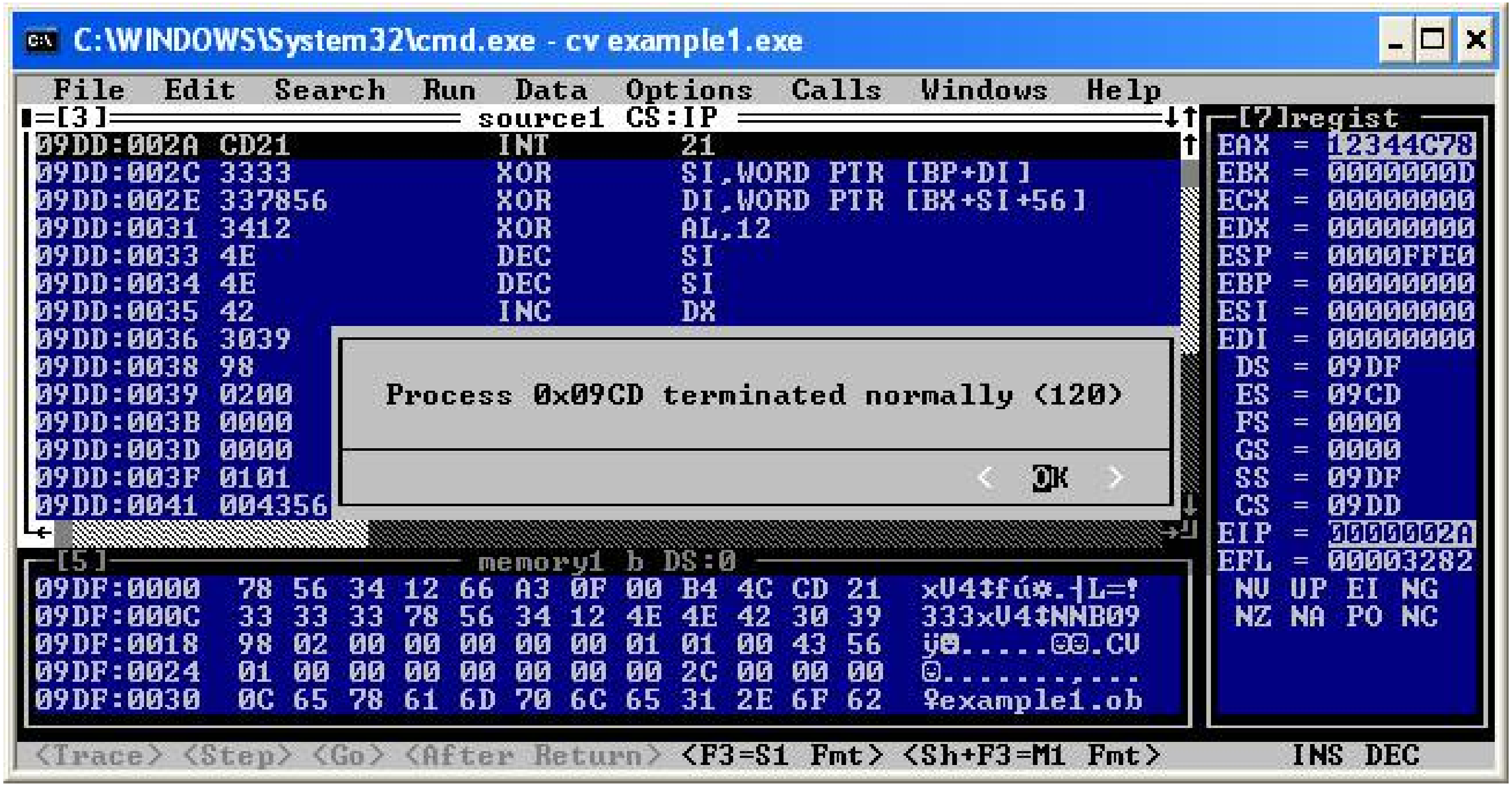
C:\WINDOWS\System32\cmd.exe - cv example1.exe
File Edit Search Run Data Options Calls Windows Help

```
[3] source1 CS:IP example1.asm
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
21:           .EXIT                 ;Exit to DOS
```

```
[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 33 33 78 56 34 12 4E 4E 42 30 39  333xV4‡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] register
EAX = 12345678
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 = 00000028
EFL = 00003282
 NV UP EI NG
 NZ NA PO NC
```

- 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
```

Examples

```
.Model small
.data
var1     db  33h
var2     dw  0101h
var3     dd  0AAAA5555h

.code
.386
.startup
      mov           ax,0
    mov  al,var1
    mov  bx,offset var2
    mov  [bx],al
    mov  [bx+1],al
    mov  eax,12345678h
    mov  var3,eax
    .exit
END
```

Examples

```
.Model small
.data
array DW 20 DUP(?)
.code
        Mov Ax,@data
        Mov DS,AX
        Mov ES,AX
        mov DI,OFFSET array
        Mov Bx,05H
        Mov CX,20
   L1:  Mov  [DI],Bx
        ADD DI,2
        loop L1
.exit
END
```

NOTE:

If the .startup directive is used (MASM version 6.x), the *Mov Ax,@data* followed by *Mov DS,AX* statement can be eliminated.

Examples

```
.Model small
.data
array DW 20 DUP(?)
.code
.startup
        mov DI,OFFSET array
        Mov Bx,05H
        Mov CX,20
   L1:  Mov  [DI],Bx
        ADD DI,2
        loop L1
.exit
END
```

NOTE: In new versions of MASM,
the assembly program can be complied
successfully without using .startup directive.

Examples

```
DATA_SEG SEGMENT 'DATA'
    array DW 20 DUP(?)
DATA_SEG      ENDS
```

*'DATA' is an **optional** field: gives important Information to the assembler for organizing The segment, but is not required.*

```
CODE_SEG      SEGMENT       'CODE'
MOV AX, DATA_SEG
MOV ES, AX
MOV DS, AX
 mov DI,OFFSET array
       Mov Bx,05H
       Mov CX,20
 L1:   Mov  [DI],Bx
       ADD DI,2
       loop L1
CODE_SEG      ENDS
END
```

NOTE: In new versions of MASM, the assembly program can be complied successfully without including the following instructions:

MOV AX, DATA_SEG
MOV ES, AX
MOV DS, AX

Examples

```
DATA_SEG SEGMENT    'DATA'
       array DW 20 DUP(?)
DATA_SEG  ENDS

CODE_SEG  SEGMENT   'CODE'
ASSUME CS:CODE_SEG, DS:DATA_SEG
MAIN          PROC          FAR
MOV AX, DATA_SEG
MOV ES, AX
MOV DS, AX
       mov DI,OFFSET array
       Mov Bx,05H
       Mov CX,20
L1:    Mov  [DI],Bx
       ADD DI,2
       loop L1
MAIN          ENDP
CODE_SEG  ENDS
END  MAIN   ; or just END
```

Notes:

1. The ASSUME statement is needed because a given assembly language program can have several code segments, one or more data segments, and more than a stack segment, but only one of each can be addressed by the CPU at a given time since there is only one of each of the segment registers available inside the CPU. Therefore, ASSUME tells the assembler which of the segments defined by the SEGMENT directives should be used.
2. Procedures:

- FAR corresponds to the term *global* which denotes for a procedure that can be used by any program.
- *NEAR* corresponds to the term *Local* which defines a procedure that is only used by the current program.

- **Still not clear how to work with the CodeView debugger?**
- **Here is additional tutorials you can go through.**

CodeView tutorial

- **http://www.nuvisionmiami.com/books/asm/cv/index.htm**

Debugging

- **http://www.math.uaa.alaska.edu/~afkjm/cs221/handouts/debugging.pdf**