



Princess Sumaya University for Technology
Computer Engineering Department
22440: Microprocessor Lab

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**" and "**Alt-S**". Exit "**Alt-F**" and "**Alt-X**"

A screenshot of a Windows command prompt window titled "C:\WINDOWS\System32\cmd.exe - edit example1.asm". The window displays the MASM32 editor interface with a menu bar (File, Edit, Search, View, Options, Help) and a toolbar. The main text area shows the assembly code for "example1.asm". The code includes data segment declarations for variables VAR1, VAR2, and VAR3, followed by a code segment starting with a STARTUP label. The code uses MOV instructions to manipulate registers (AX, AL, BX, EAX) and memory locations. The program ends with an EXIT instruction. The status bar at the bottom shows "F1=Help", "Line:21", and "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:\WINDOWS\System32\cmd.exe

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.

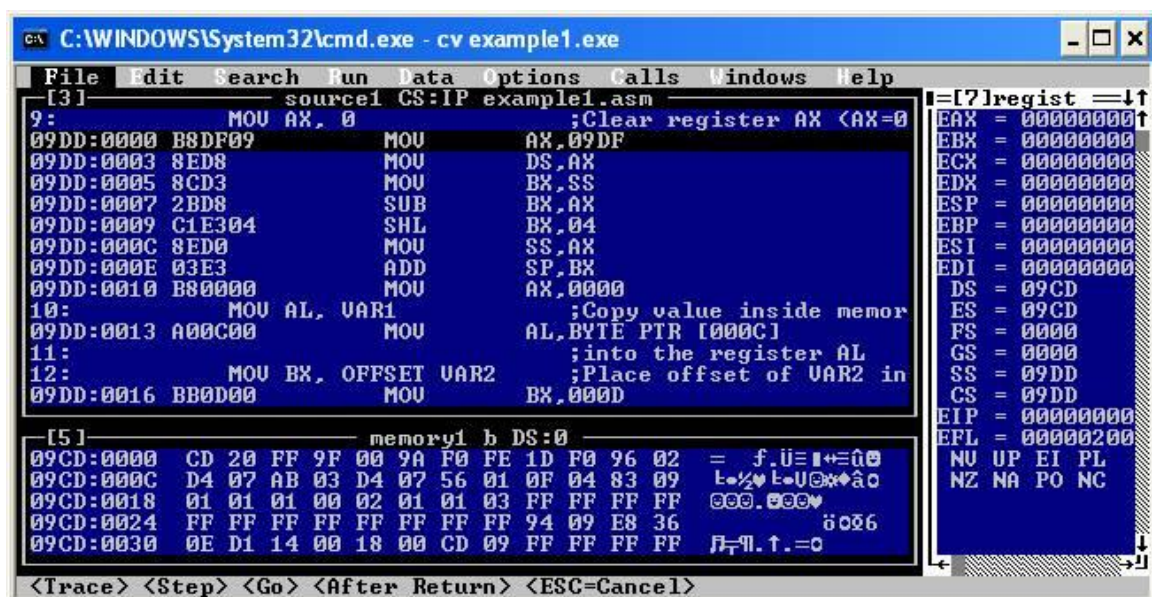
Object Modules [.obj]: example1.obj /CO:nopack
Run File [example1.exe]: "example1.exe"
List File [nul.map]: NUL
Libraries [lib]:
Definitions File [nul.def]:
LINK : warning L4021: no stack segment
CVPACK : warning CK4007: unrecognized option /x; option ignored
Microsoft (R) Debugging Information Compactor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

C:\masm32>

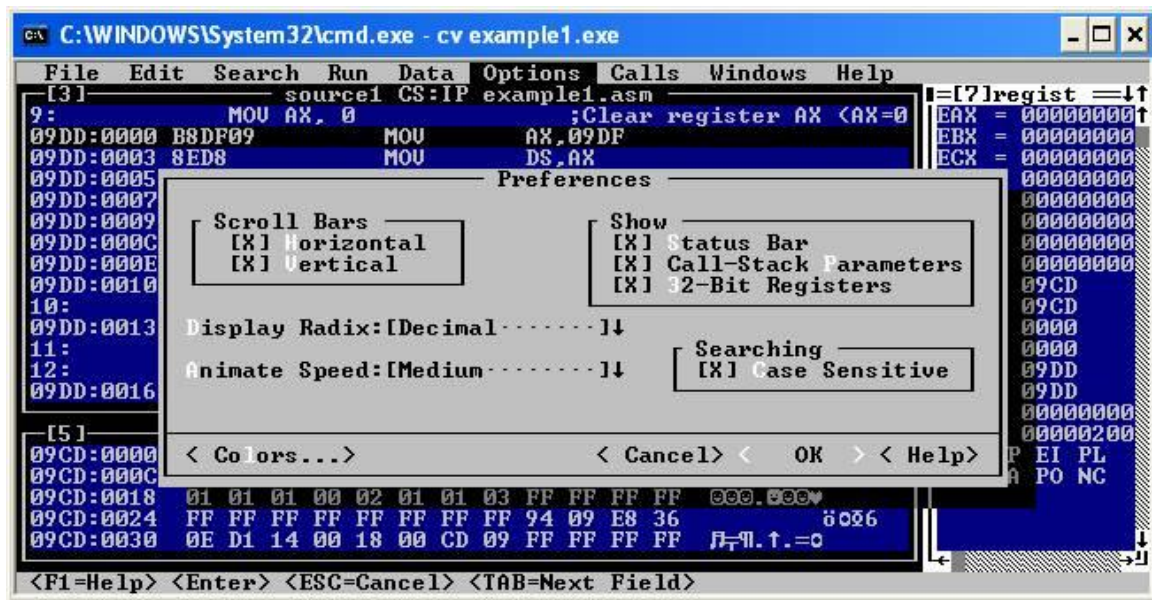
```

- Now let us 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:
 - source1
 - registers
 - memory1

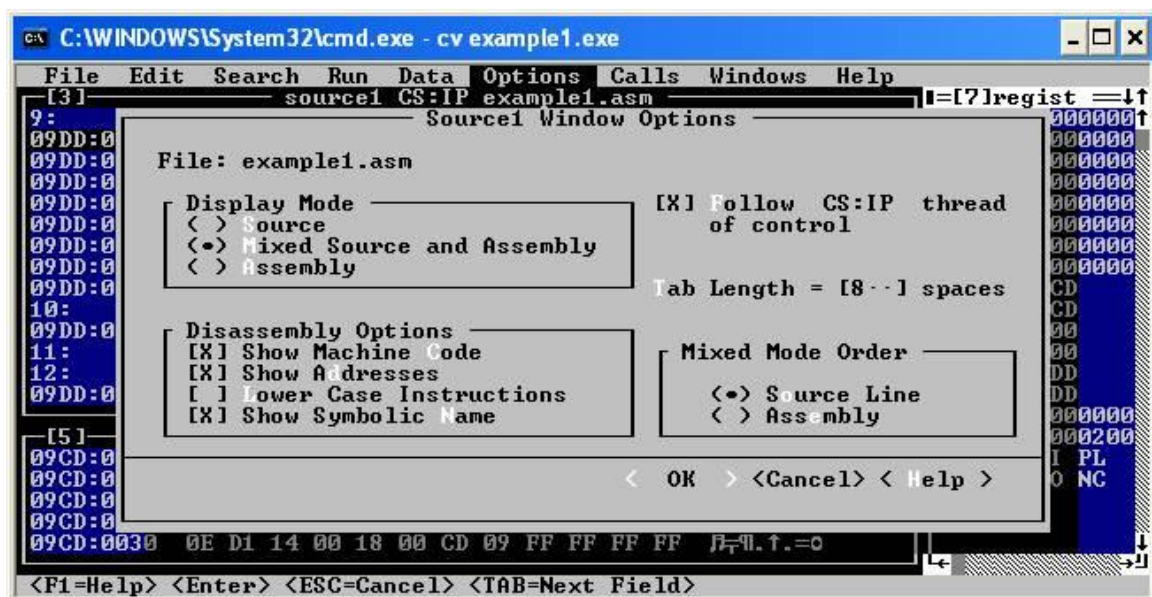
Press “**Alt-F5**” to arrange the windows on the screen.



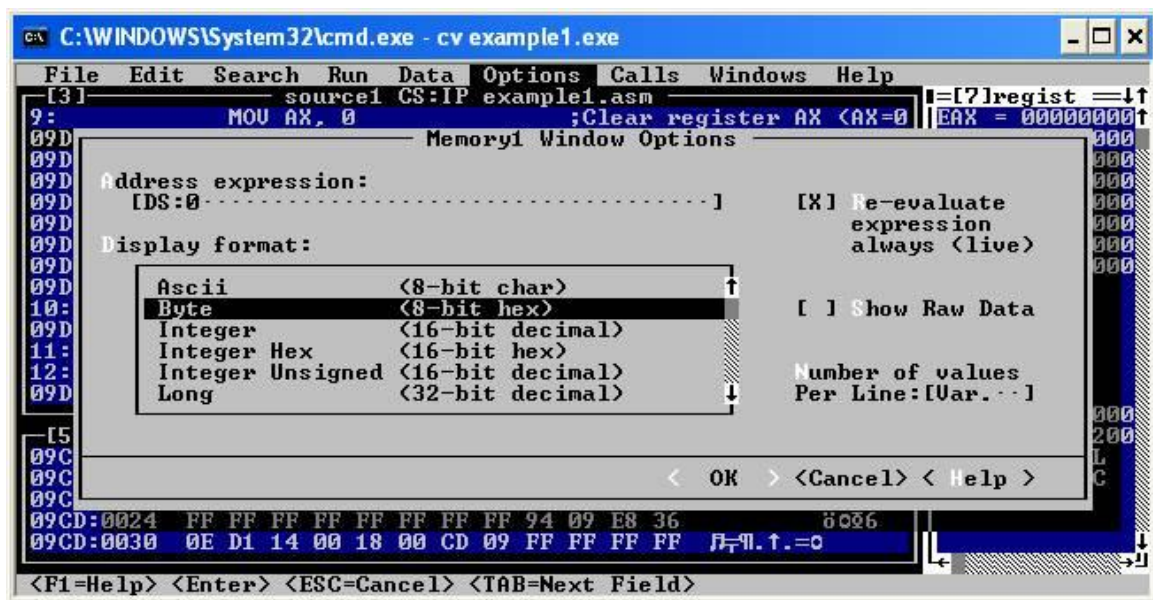
- Set the options through “Alt-O” -> **Preferences**. Set the options as shown below and click “OK”.



- Set Source 1 window options as shown below through “Alt-O” -> “Source 1 window”

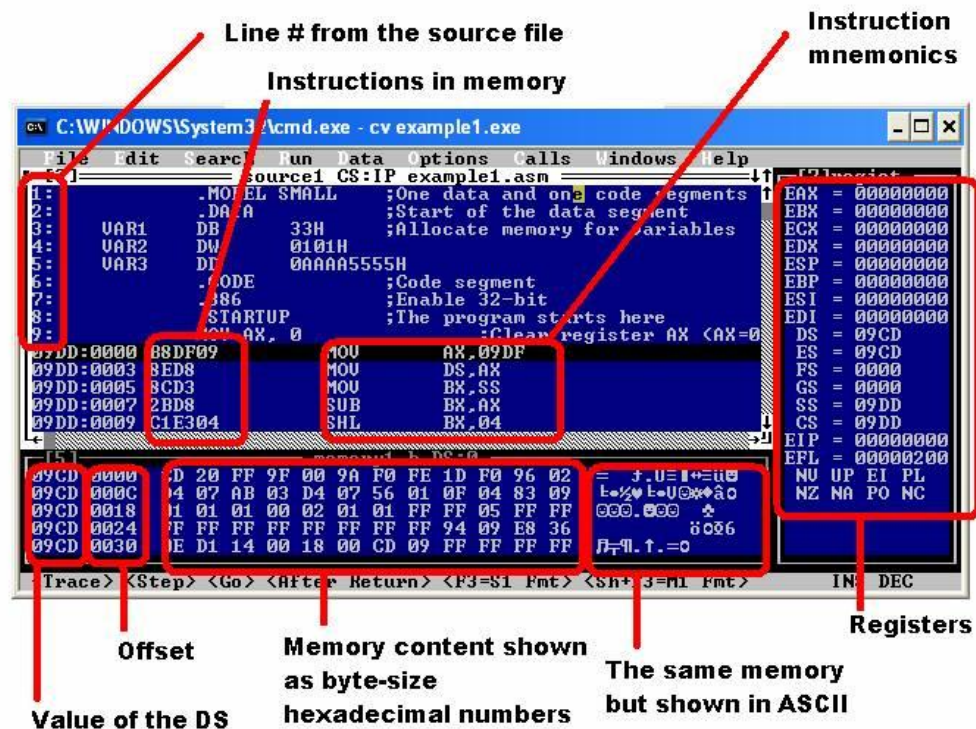


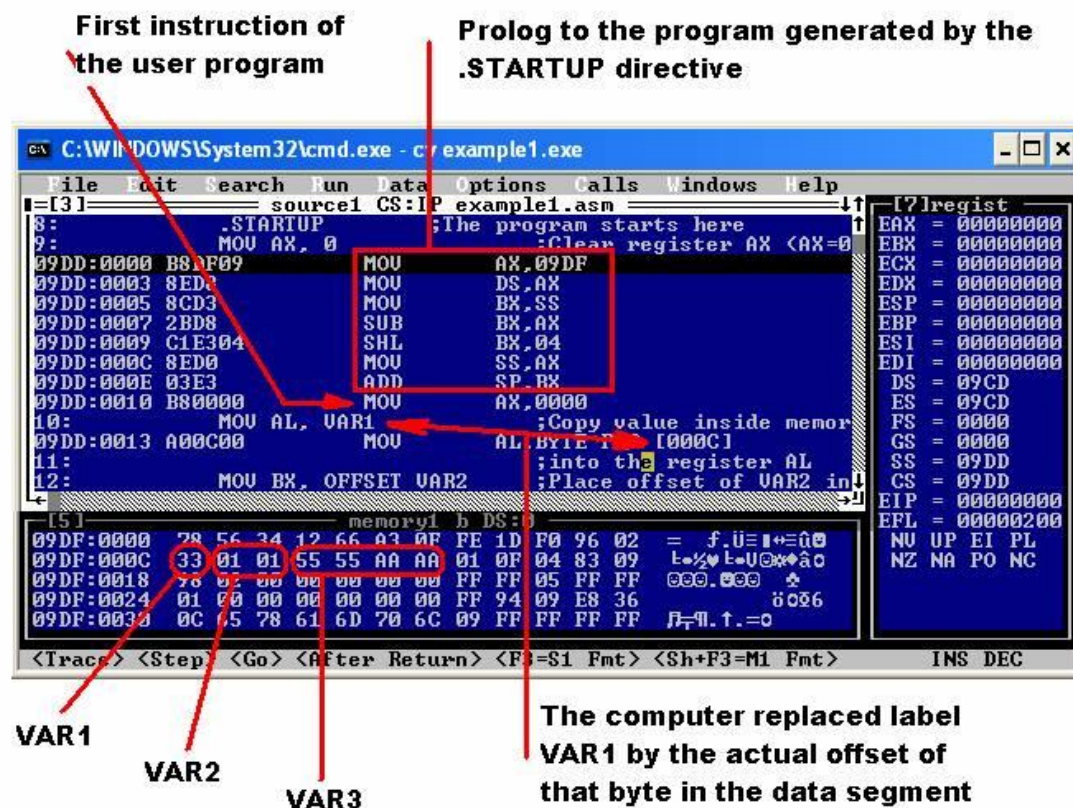
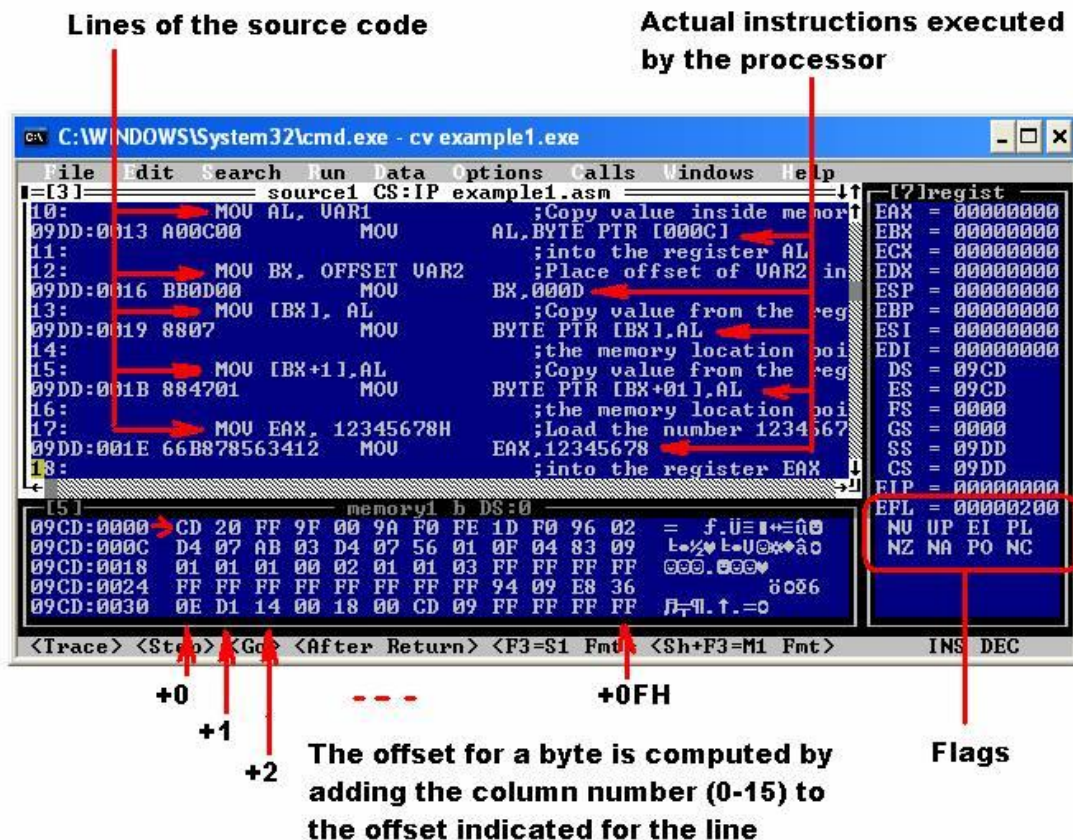
- And the memory window options through “Alt-O” -> “Memory 1 window”



The configuration is now complete.

- Let us examine the **cv** program.





- Step through the program and observe the execution of every instruction.
 - Press “F10”.
 - The debugger shows execution of the first line of the initialization code.
 - Press “F10” multiple times until the 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 = 0000FF00
ECX = 00000000
EDX = 00000000
ESP = 0000FF00
EBP = 00000000
ESI = 00000000
EDI = 00000000
DS = 09DF
ES = 09CD
FS = 0000
GS = 0000
SS = 09DF
CS = 09DD
EIP = 00000010
EFL = 00003282
NU 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 xU4tfú*.!L=?
09DF:000C 33 01 01 55 55 AA AA 4E 4E 42 30 39 300UU--NMB09
09DF:0018 98 02 00 00 00 00 00 01 01 00 43 56 y0.....00.CU
09DF:0024 01 00 00 00 00 00 00 00 2C 00 00 00 0.....
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 the 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
  
```

[7]regist

```

EAX = 000009DF
EBX = 0000FF00
ECX = 00000000
EDX = 00000000
ESP = 0000FF00
EBP = 00000000
ESI = 00000000
EDI = 00000000
DS = 09DF
ES = 09CD
FS = 0000
GS = 0000
SS = 09DF
CS = 09DD
EIP = 00000010
EFL = 00003282
NU 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 xU4tfú*.!L=?
09DF:000C 33 01 01 55 55 AA AA 4E 4E 42 30 39 300UU--NMB09
09DF:0018 98 02 00 00 00 00 00 01 01 00 43 56 y0.....00.CU
09DF:0024 01 00 00 00 00 00 00 00 2C 00 00 00 0.....
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

- 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.

The screenshot shows a debugger window titled "C:\WINDOWS\System32\cmd.exe - cv example1.exe". The main window displays assembly code with the following instructions:

```

09:      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 memory
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

```

The register window on the right shows the following values:

EAX	= 00000000
EBX	= 0000FFEE
ECX	= 00000000
EDX	= 00000000
ESP	= 0000FFEE
EBP	= 00000000
ESI	= 00000000
EDI	= 00000000
DS	= 09DF
ES	= 09CD
FS	= 0000
GS	= 0000
SS	= 09DF
CS	= 09DD
EIP	= 00000013
EFL	= 00003282
NU	UP EI NG
NZ	NA PO NC
ds:000c	33

The memory window at the bottom shows the contents of memory locations from 09DF:0000 to 09DF:0030.

The highlighting of 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 contents of memory location DS:000CH in the memory window.

- Now execute this instruction by pressing “F10”. The content of the register AL changes, taking the value from VAR1.

The screenshot shows the same debugger window after executing the instruction “MOV AL, [000C]”. The source code window now highlights instruction 10: “MOV AL, [000C]”. The register window on the right shows the following values:

EAX	= 00000033
EBX	= 0000FFEE
ECX	= 00000000
EDX	= 00000000
ESP	= 0000FFEE
EBP	= 00000000
ESI	= 00000000
EDI	= 00000000
DS	= 09DF
ES	= 09CD
FS	= 0000
GS	= 0000
SS	= 09DF
CS	= 09DD
EIP	= 00000016
EFL	= 00003282
NU	UP EI NG
NZ	NA PO NC

The memory window at the bottom shows the contents of memory locations from 09DF:0000 to 09DF:0030, with the value at 000C being 33.

- The next instruction is “MOV BX, OFFSET VAR2”. VAR2 follows VAR1 in memory and has offset 000DH. This is the value that will be placed into the BX register 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
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 xU4tfu*.!L=?
09DF:000C 33 01 01 55 55 AA AA 4E 4E 42 30 39 30UU--NMB09
09DF:0018 98 02 00 00 00 00 00 01 01 00 43 56 y0.....00.CU
09DF:0024 01 00 00 00 00 00 00 00 2C 00 00 00 0.....
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 = 0000FFEE
EBP = 00000000
ESI = 00000000
EDI = 00000000
DS = 0040
ES = 0040
FS = 0000
GS = 0000
SS = 0040
CS = 0040
EIP = 00401000
EFL = 00003282
NU UP EI NG
NZ NA PO NC
ds:000d
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 to 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
[5] memory1 b DS:0
09DF:0000 78 56 34 12 66 A3 0F 00 B4 4C CD 21 xU4tfu*.!L=?
09DF:000C 33 01 01 55 55 AA AA 4E 4E 42 30 39 30UU--NMB09
09DF:0018 98 02 00 00 00 00 00 01 01 00 43 56 y0.....00.CU
09DF:0024 01 00 00 00 00 00 00 00 2C 00 00 00 0.....
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 = 0000FFEE
EBP = 00000000
ESI = 00000000
EDI = 00000000
DS = 0040
ES = 0040
FS = 0000
GS = 0000
SS = 0040
CS = 0040
EIP = 00401000
EFL = 00003282
NU UP EI NG
NZ NA PO NC
ds:000e
01
<Trace> <Step> <Go> <After Return> <F3=S1 Fmt> <Sh+F3=M1 Fmt> INS DEC

```


- Instruction “MOV [BX+1], AL” copies the contents of 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. This is the second byte of VAR2. Press “F10” to execute. Note the change in the memory contents.

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							
09DD:0019 8807	MOV			BYTE PTR [BX],AL				
14:								
15:	MOV [BX+1],AL							
09DD:001B 884701	MOV			BYTE PTR [BX+01],AL				
16:								
17:	MOV EAX, 12345678H							
09DD:001E 66B878563412	MOV			EAX,12345678				
18:								
19:	MOV VAR3, EAX							
09DD:0024 66A30F00	MOV			DWORD PTR [000F],EAX				
20:								
21:	.EXIT							
22:	END							

[5] memory1 b DS:0															
09DF:0000	78	56	34	12	66	A3	0F	00	B4	4C	CD	21	xU4tfú*.!L=!		
09DF:000C	33	33	33	55	55	AA	AA	4E	4E	42	30	39	333UU~NMB09		
09DF:0018	98	02	00	00	00	00	00	01	01	00	43	56	U0.....00.CU		
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	=	00000030													
ECX	=	00000000													
EDX	=	00000000													
ESP	=	0000FF00													
EBP	=	00000000													
ESI	=	00000000													
EDI	=	00000000													
DS	=	09DF													
ES	=	09CD													
FS	=	0000													
GS	=	0000													
SS	=	09DF													
CS	=	09DD													
EIP	=	0000001E													
EFL	=	00003282													
NU	UP	EI	NG												
NZ	NA	PO	NC												

<Trace> <Step> <Go> <After Return> <F3=\$1 Fmt> <Sh+F3=M1 Fmt> INS DEC

- Instruction “MOV EAX, 12345678H” places the number 12345678H into 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							
09DD:0019 8807	MOV			BYTE PTR [BX],AL				
14:								
15:	MOV [BX+1],AL							
09DD:001B 884701	MOV			BYTE PTR [BX+01],AL				
16:								
17:	MOV EAX, 12345678H							
09DD:001E 66B878563412	MOV			EAX,12345678				
18:								
19:	MOV VAR3, EAX							
09DD:0024 66A30F00	MOV			DWORD PTR [000F],EAX				
20:								
21:	.EXIT							
22:	END							

[5] memory1 b DS:0															
09DF:0000	78	56	34	12	66	A3	0F	00	B4	4C	CD	21	xU4tfú*.!L=!		
09DF:000C	33	33	33	55	55	AA	AA	4E	4E	42	30	39	333UU~NMB09		
09DF:0018	98	02	00	00	00	00	00	01	01	00	43	56	U0.....00.CU		
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	=	12345678													
EBX	=	00000030													
ECX	=	00000000													
EDX	=	00000000													
ESP	=	0000FF00													
EBP	=	00000000													
ESI	=	00000000													
EDI	=	00000000													
DS	=	09DF													
ES	=	09CD													
FS	=	0000													
GS	=	0000													
SS	=	09DF													
CS	=	09DD													
EIP	=	00000024													
EFL	=	00003282													
NU	UP	EI	NG												
NZ	NA	PO	NC												
			ds:000f												
			aaaa5555												

<Trace> <Step> <Go> <After Return> <F3=\$1 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 takes the contents of EAX and places them in the four consecutive bytes of memory (a 32-bit variable) starting at the offset 000FH. Press “F10” to execute.

The screenshot shows a debugger window titled "C:\WINDOWS\System32\cmd.exe - cv example1.exe". The assembly window displays the following code:

```

09DD:0016 BB0D00      MOV     BX,000D
13:              ;Copy value from the reg
09DD:0019 8807      MOV     BYTE PTR [BX],AL
14:              ;the memory location poi
15:              ;Copy value from the reg
09DD:001B 884701      MOV     BYTE PTR [BX+01],AL
16:              ;the memory location poi
17:              ;Load the number 1234567
09DD:001E 66B878563412  MOV     EAX,12345678
18:              ;into the register EAX
19:              ;Copy value from the reg
09DD:0024 66A30F00      MOV     DWORD PTR [000F],EAX
20:              ;the memory location VAR
21:              ;Exit to DOS
.EXIT

```

The registers window on the right shows the following values:

```

EAX = 12345678
EBX = 0000000D
ECX = 00000000
EDX = 00000000
ESP = 0000FF00
EBP = 00000000
ESI = 00000000
EDI = 00000000
DS = 09DF
ES = 09CD
FS = 0000
GS = 0000
SS = 09DF
CS = 09DD
EIP = 00000028
EFL = 00003282
NU UP EI NG
NZ NA PO NC

```

The memory window at the bottom shows the following data:

```

09DF:0000 78 56 34 12 66 A3 0F 00 B4 4C CD 21 xU4fú*.!L=?
09DF:000C 33 33 33 78 56 34 12 4E 4E 42 30 39 333xU4$NMB09
09DF:0018 98 02 00 00 00 00 00 01 01 00 43 56 y0.....00.CU
09DF:0024 01 00 00 00 00 00 00 00 2C 00 00 00 0.....
09DF:0030 0C 65 78 61 6D 70 6C 65 31 2E 6F 62 %example1.ob

```

- This 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” multiple times until the process terminates.

The screenshot shows the same debugger window. The assembly window displays the following code:

```

09DD:002A CD21      INT     21
09DD:002C 3333      XOR     SI,WORD PTR [BP+DI]
09DD:002E 337856    XOR     DI,WORD PTR [BX+SI+56]
09DD:0031 3412      XOR     AL,12
09DD:0033 4E        DEC     SI
09DD:0034 4E        DEC     SI
09DD:0035 42        INC     DX
09DD:0036 3039      MOV     AL,BYTE PTR [SI]
09DD:0038 98        MOV     AL,8
09DD:0039 0200      MOV     AL,0
09DD:003B 0000      MOV     AL,0
09DD:003D 0000      MOV     AL,0
09DD:003F 0101      MOV     AL,1
09DD:0041 004356    MOV     AL,56

```

The registers window on the right shows the following values:

```

EAX = 12344C78
EBX = 0000000D
ECX = 00000000
EDX = 00000000
ESP = 0000FF00
EBP = 00000000
ESI = 00000000
EDI = 00000000
DS = 09DF
ES = 09CD
FS = 0000
GS = 0000
SS = 09DF
CS = 09DD
EIP = 0000002A
EFL = 00003282
NU UP EI NG
NZ NA PO NC

```

The memory window at the bottom shows the following data:

```

09DF:0000 78 56 34 12 66 A3 0F 00 B4 4C CD 21 xU4fú*.!L=?
09DF:000C 33 33 33 78 56 34 12 4E 4E 42 30 39 333xU4$NMB09
09DF:0018 98 02 00 00 00 00 00 01 01 00 43 56 y0.....00.CU
09DF:0024 01 00 00 00 00 00 00 00 2C 00 00 00 0.....
09DF:0030 0C 65 78 61 6D 70 6C 65 31 2E 6F 62 %example1.ob

```

A message box is displayed in the center of the screen with the text: "Process 0x09CD terminated normally <120>".

Reference

<http://www.intelligent-systems.info/classes/ee360/tutorial.htm>