BAŞKENT UNIVERSITY

ENGINEERING FACULTY

ELECTRICAL-ELECTRONICS
ENGINEERING DEPARTMENT

EEM 322 - MICROPROCESSORS LAB

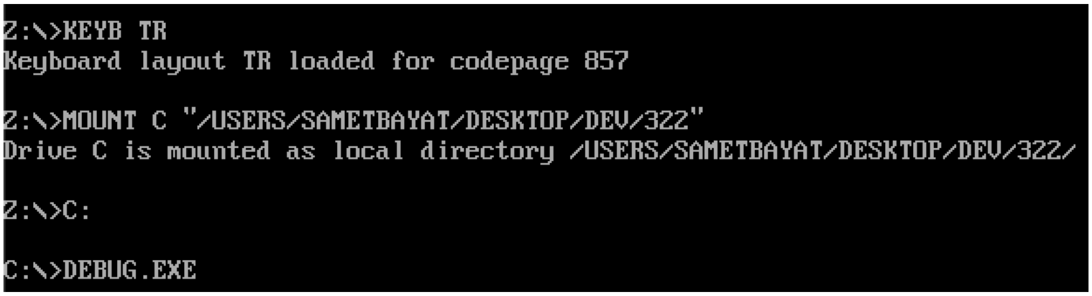
EXPERIMENT NO. 02:

INTRODUCTION TO DEBUG
x86 REGISTERS AND MACHINE COMMANDS

SAMET BAYAT

22293730

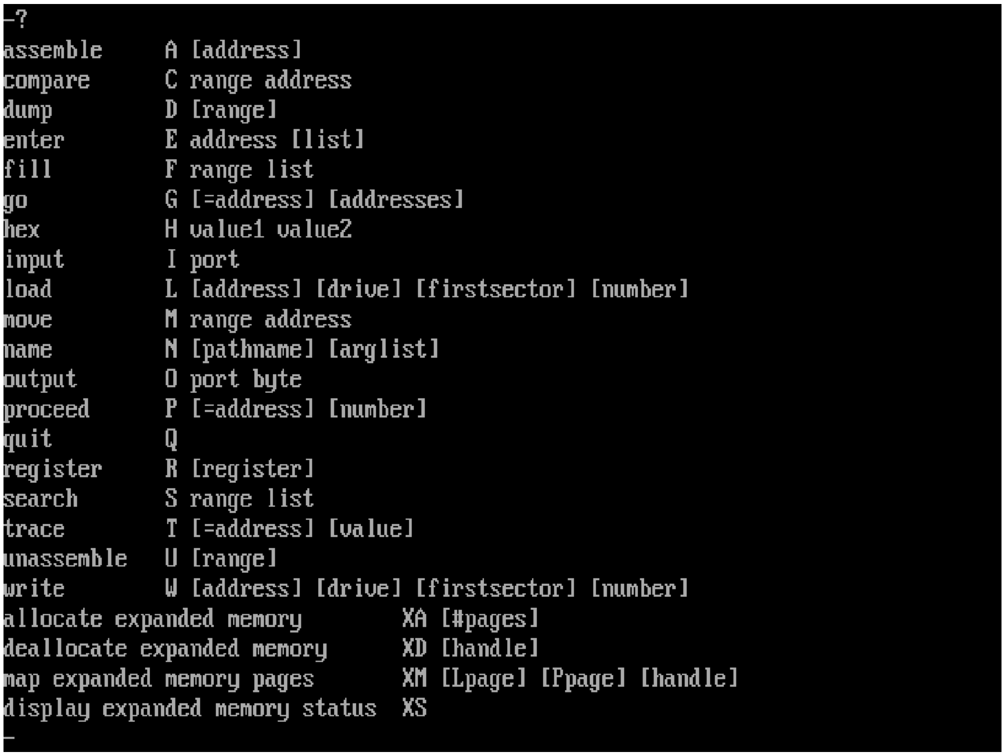
Initially, our first prompt is about we are using keyboard in Turkish language, and we tell the machine where our directory is and enter in it. After than we execute “debug.exe” which will help us while debugging.



```
Z:\>KEYB TR
Keyboard layout TR loaded for codepage 857

Z:\>MOUNT C "/USERS/SAMETBAYAT/DESKTOP/DEV/322"
Drive C is mounted as local directory /USERS/SAMETBAYAT/DESKTOP/DEV/322/

Z:\>C:

C:\>DEBUG.EXE
```

For this very first attempt, I tried various programs and methods (virtual machine, open-source codes, etc.) During trials I faced some bugs therefore you may see default statements might be changed because of using different apps.

* Virtual machine worked well and “Boxer app” is quite user friendly although there are some crashes.

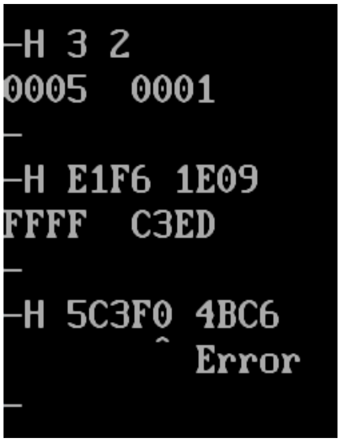


```
-?
assemble     A [address]
compare      C range address
dump         D [range]
enter        E address [list]
fill         F range list
go           G [=address] [addresses]
hex          H value1 value2
input        I port
load         L [address] [drive] [firstsector] [number]
move         M range address
name         N [pathname] [arglist]
output       O port byte
proceed      P [=address] [number]
quit         Q
register     R [register]
search       S range list
trace        T [=address] [value]
unassemble   U [range]
write        W [address] [drive] [firstsector] [number]
allocate expanded memory       XA [#pages]
deallocate expanded memory     XD [handle]
map expanded memory pages      XM [Lpage] [Ppage] [handle]
display expanded memory status  XS
-
```

-?
shows the possible actions
(like “man, help, -help?” on other platforms)

-H

allows us to do hexadecimal operations like adding or subtracting. (! But for 16 bits)



```
-H 3 2
0005  0001
-
-H E1F6 1E09
FFFF  C3ED
-
-H 5C3F0 4BC6
       ^ Error
-
```

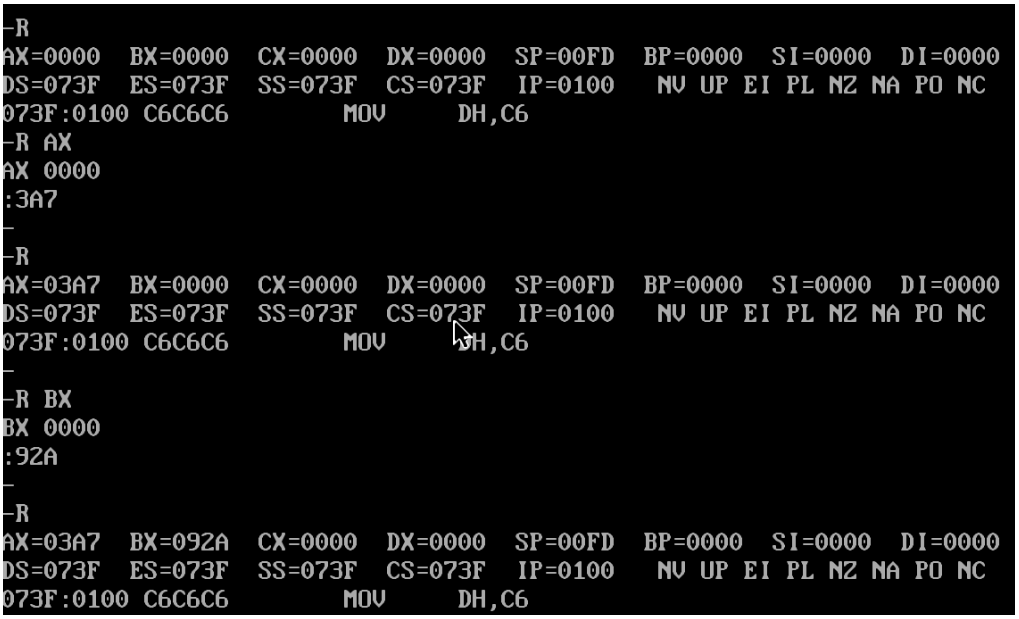
Here we add/subtract 2H to/from 3H.

Here again we do standard hex operations like above also the summation shows the possible highest number.

As we know the max. number which the machine is able to operate is 0xFFFF and '0x5C3F0' is greater than it so the machine could not handle it.

-R

Plain usage shows the status of registers and pending operation, also it can change the value of them.

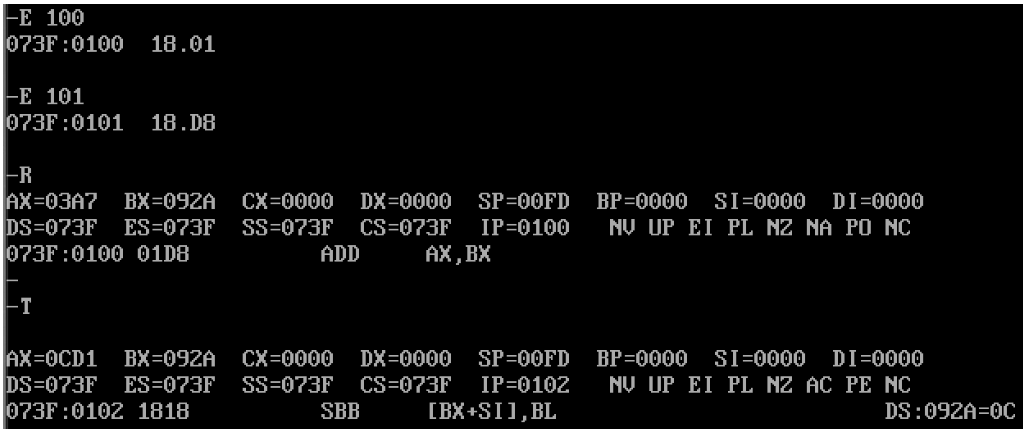


```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 C6C6C6        MOV     DH,C6
-R AX
AX 0000
:3A7
-
-R
AX=03A7  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 C6C6C6        MOV     DH,C6
-
-R BX
BX 0000
:92A
-
-R
AX=03A7  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 C6C6C6        MOV     DH,C6
```

Here, AX and BX = 0000, with command '-R ..' they became 3A7H and 92AH.

-E

From now on, our intention is to add 'BX to AX', to manage that we need an intermediary element which helps to execute machine commands on x86 microprocessor memory.

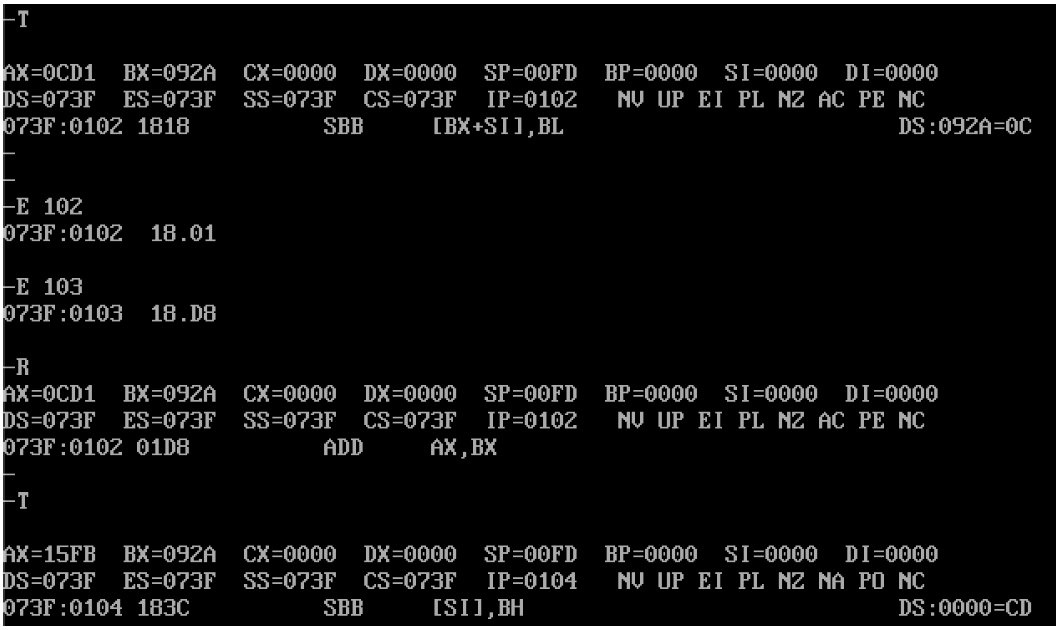


```
-E 100
073F:0100  18.01

-E 101
073F:0101  18.D8

-R
AX=03A7  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 01D8          ADD     AX,BX
-
-T

AX=0CD1  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ AC PE NC
073F:0102 1818          SBB     [BX+SI],BL                         DS:092A=0C
```

Here the number comes after -E is the location of pointer (also you can observe “IP=0100”) and “01D8” is the expression that tells the machine our next operation is **”ADD AX, BX”** (add BX to AX)

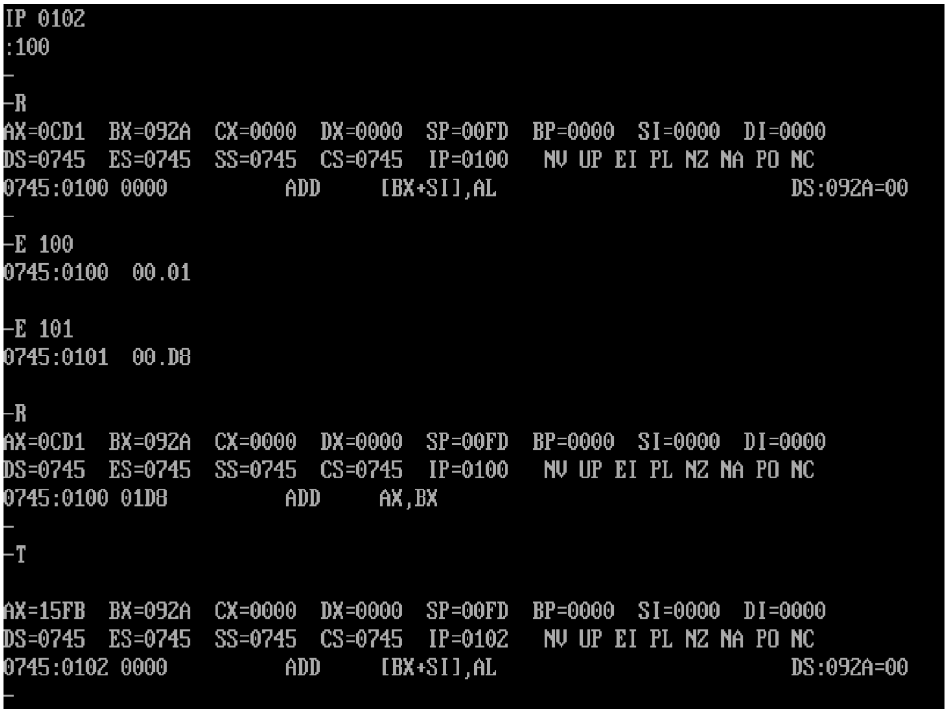
-T
“Trace” runs the machine commands (one for each time).

For 2nd attempt, again we want to do **”ADD AX, BX”** operation. Unlike the previous operation, now as we see on second to last line pointer value became “0102”. That means either have to follow the pointer or rearrange it with **“-R IP”** command. Both scenarios given in below: (0x0CD1 + 0x092A = 0x15FB)



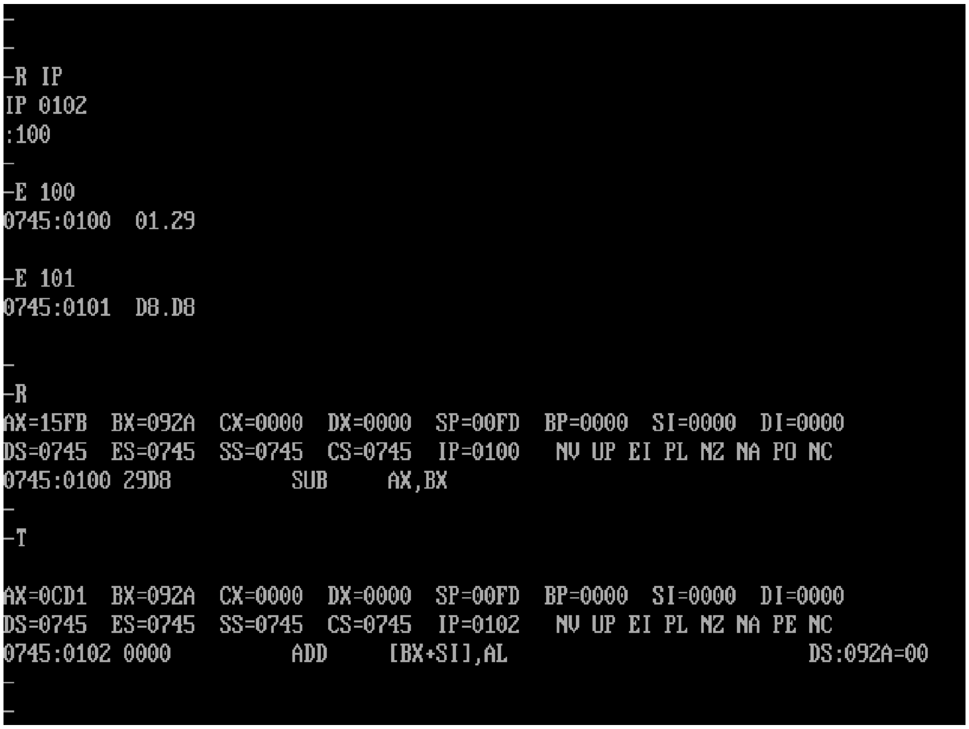
```
-T

AX=0CD1  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ AC PE NC
073F:0102 1818          SBB     [BX+SI],BL                         DS:092A=0C
-
-
-E 102
073F:0102  18.01

-E 103
073F:0103  18.D8

-R
AX=0CD1  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ AC PE NC
073F:0102 01D8          ADD     AX,BX
-
-T

AX=15FB  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0104   NV UP EI PL NZ NA PO NC
073F:0104 183C          SBB     [SI],BH                            DS:0000=CD
```

Scenario 1



```
IP 0102
:100
-
-R
AX=0CD1  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0100   NV UP EI PL NZ NA PO NC
0745:0100 0000          ADD     [BX+SI],AL                         DS:092A=00
-
-E 100
0745:0100  00.01

-E 101
0745:0101  00.D8

-R
AX=0CD1  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0100   NV UP EI PL NZ NA PO NC
0745:0100 01D8          ADD     AX,BX
-
-T

AX=15FB  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0102   NV UP EI PL NZ NA PO NC
0745:0102 0000          ADD     [BX+SI],AL                         DS:092A=00
-
```

Scenario 2



```
-
-
-R IP
IP 0102
:100
-
-E 100
0745:0100  01.29

-E 101
0745:0101  D8.D8

-
-R
AX=15FB  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0100   NV UP EI PL NZ NA PO NC
0745:0100 29D8          SUB     AX,BX
-
-T

AX=0CD1  BX=092A  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0102   NV UP EI PL NZ NA PE NC
0745:0102 0000          ADD     [BX+SI],AL                         DS:092A=00
-
-
```

Here again, the pointer set as “0100” and for subtraction operation “29D8” assigned (4th code block shows the **"SUB AX, BX"** expression) Finally, **-T** runs the machine command. (AX became “0CD1” as we expected. 15FBH-92AH = CD1H)

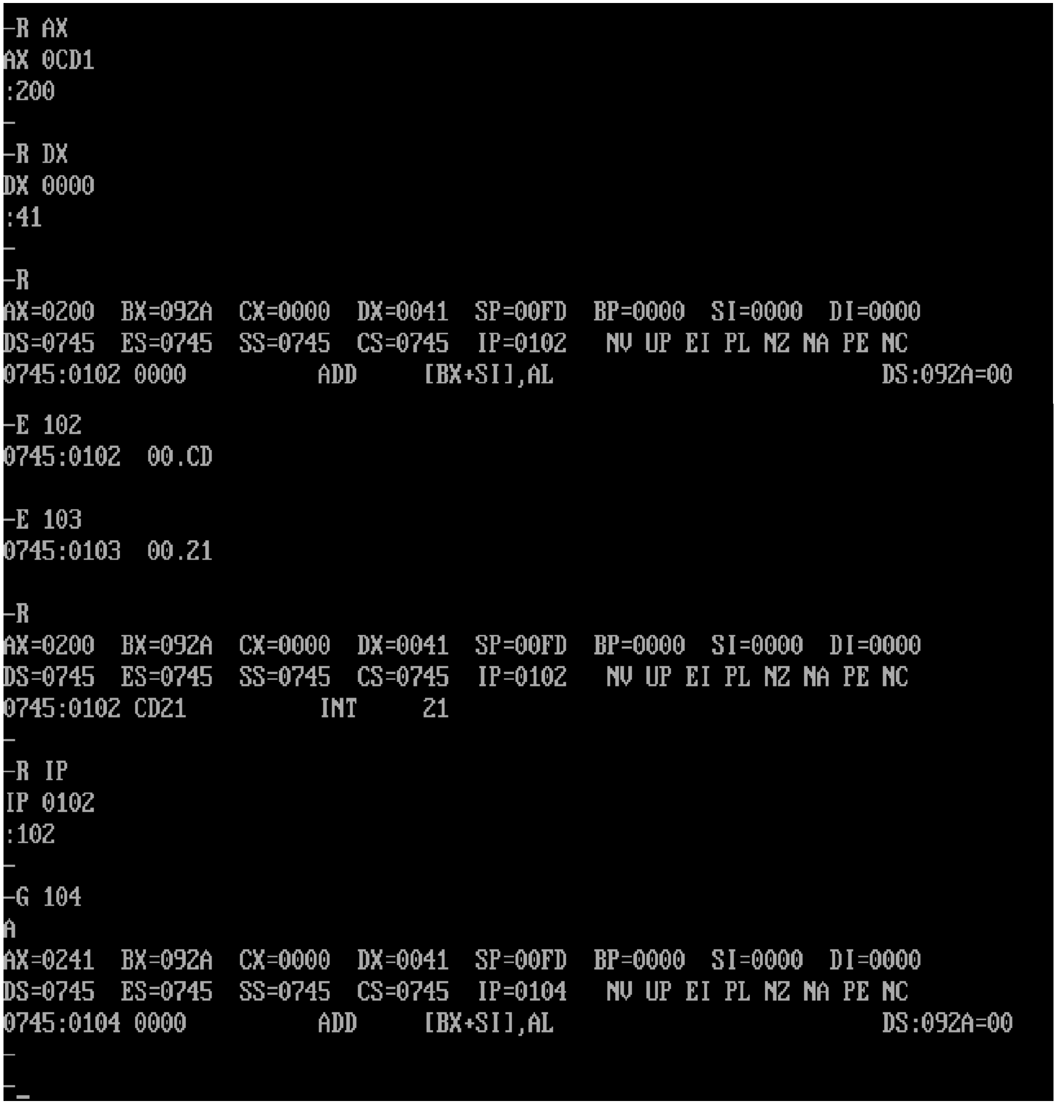
INT

Now in the next part of our duty, we are going to use **INT** instruction which is used by application programs to access services provided by the operating system, such as input/output operations, memory management, and other system services.

The machine knows **INT 21** command as “CD21”.

-G (Go until) provides sequentially execution of code blocks. (In our case, it starts where the pointer is “102” to “104”)

End of all this misery, we see the letter **“A”** :)
(41H is the ASCII equivalent of “A”.)

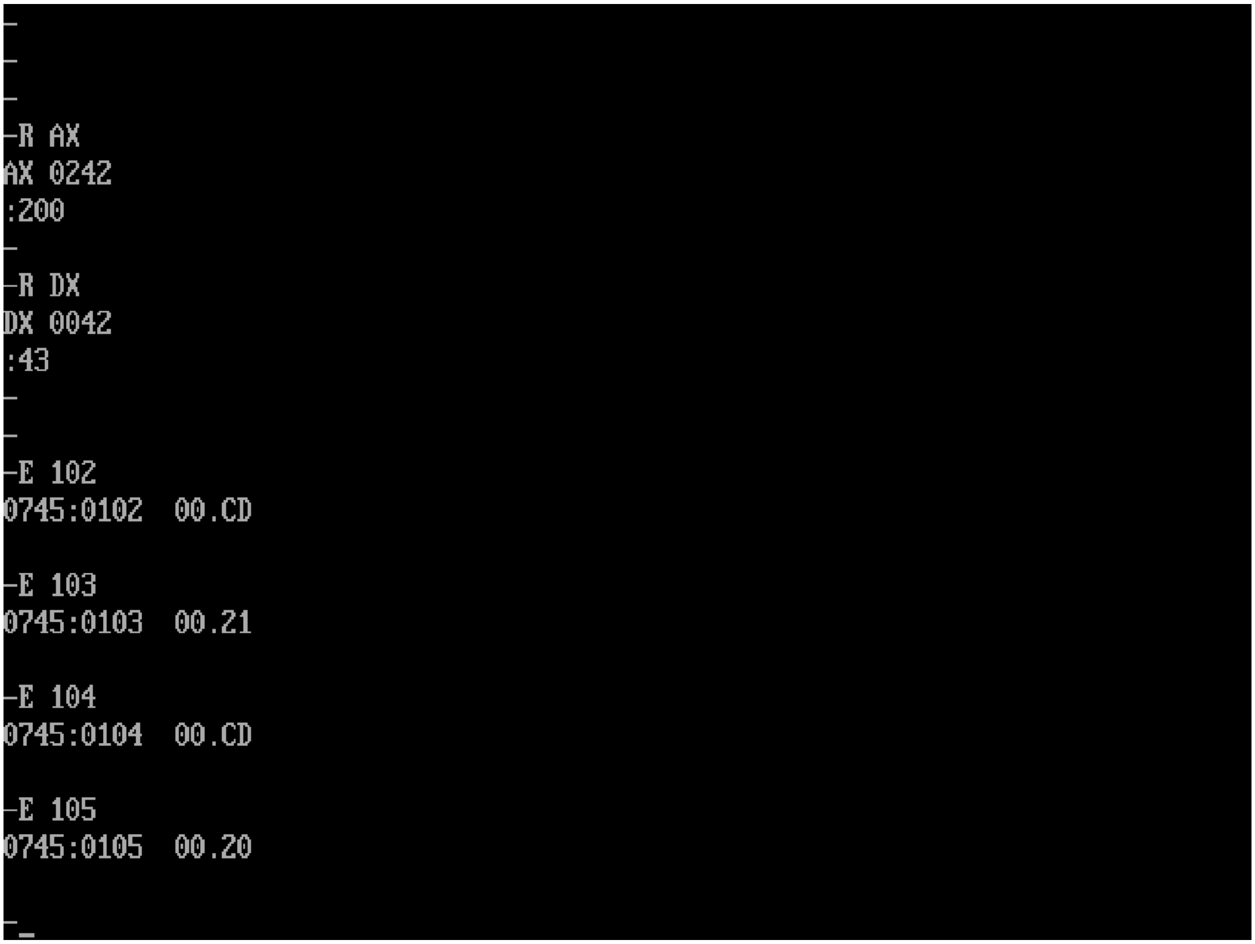


```
-R AX
AX 0CD1
:200
-
-R DX
DX 0000
:41
-
-R
AX=0200  BX=092A  CX=0000  DX=0041  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0102   NV UP EI PL NZ NA PE NC
0745:0102 0000          ADD     [BX+SI],AL                         DS:092A=00

-E 102
0745:0102  00.CD

-E 103
0745:0103  00.21

-R
AX=0200  BX=092A  CX=0000  DX=0041  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0102   NV UP EI PL NZ NA PE NC
0745:0102 CD21          INT     21
-
-R IP
IP 0102
:102
-
-G 104
A
AX=0241  BX=092A  CX=0000  DX=0041  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0104   NV UP EI PL NZ NA PE NC
0745:0104 0000          ADD     [BX+SI],AL                         DS:092A=00
-
-_
```

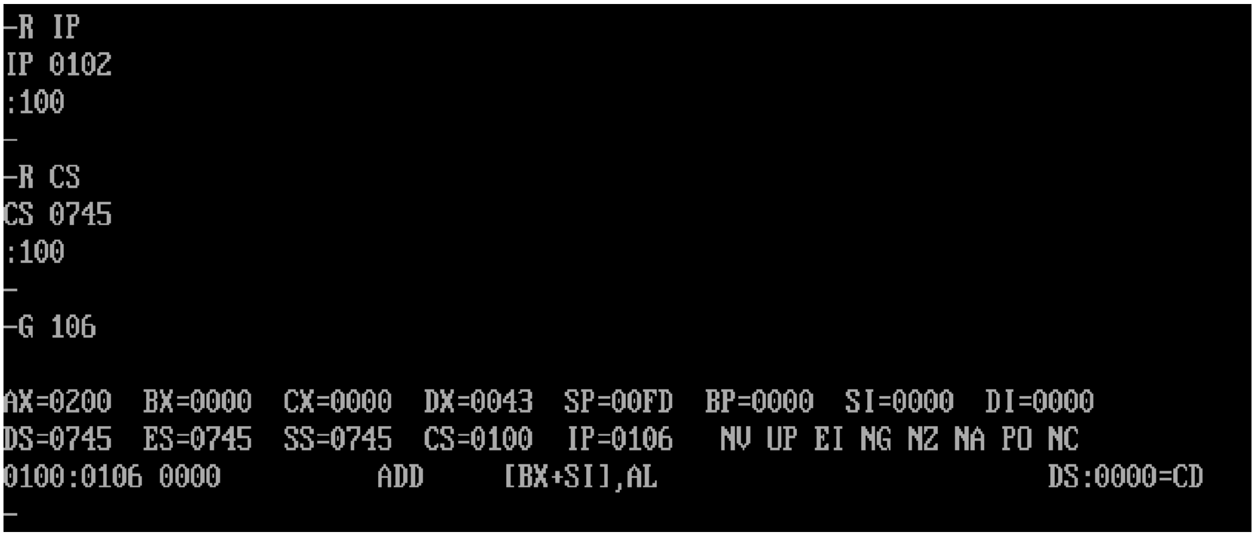
After reaching our destination we need to end the program. This procedure is mandatory for assembler languages, **INT 21** command satisfies that. For executing a program like:

```
INT 20
INT 21
```

In debug, to achieve this we need to save “CD20 and CD21” to the memory, rearrange the IP and CS. Finally, -G will execute the block. Additionally, in some cases we might want to undo (unassemble)the improvement.



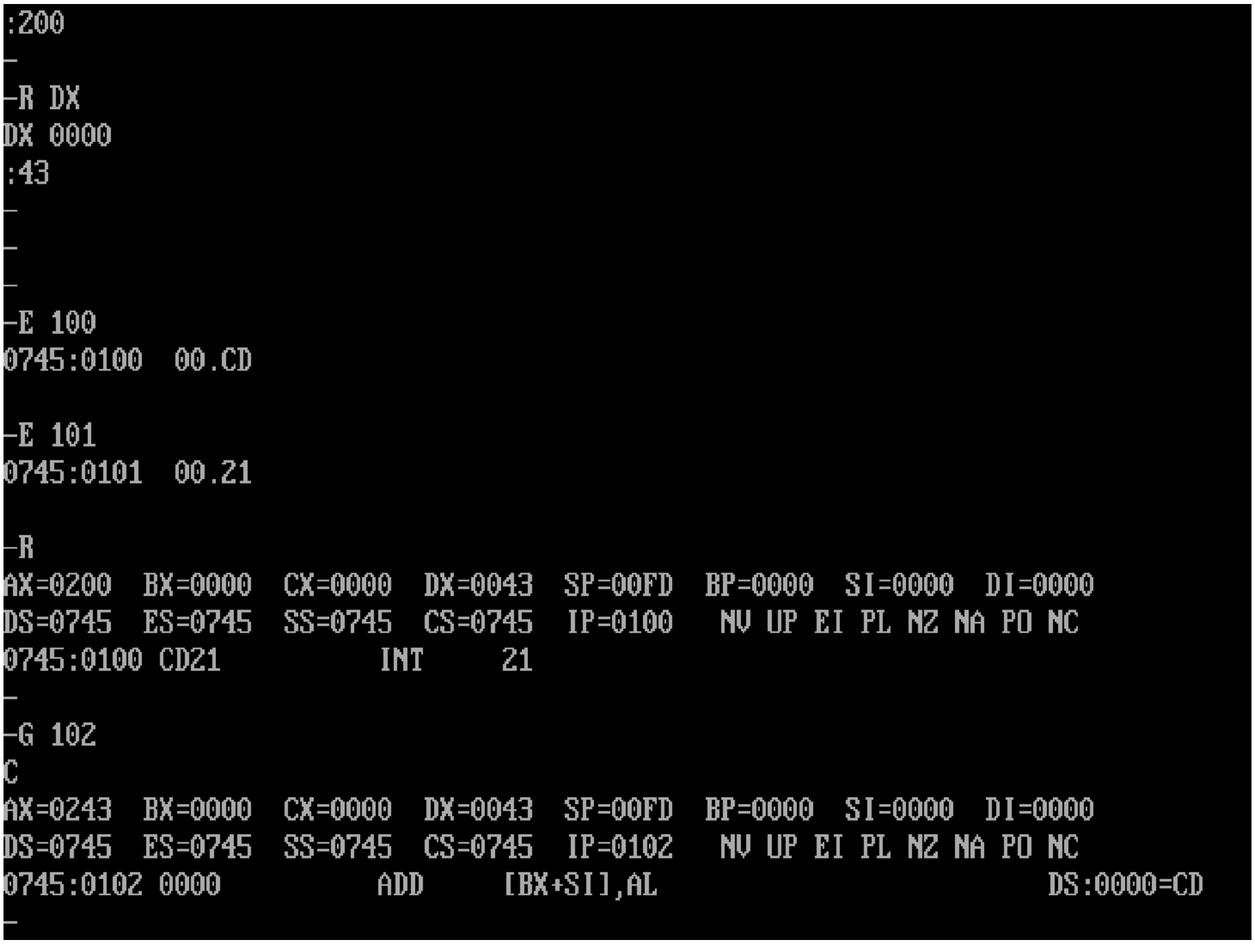
```
-
-
-
-R AX
AX 0242
:200
-
-R DX
DX 0042
:43
-
-
-E 102
0745:0102  00.CD

-E 103
0745:0103  00.21

-E 104
0745:0104  00.CD

-E 105
0745:0105  00.20


-_
```



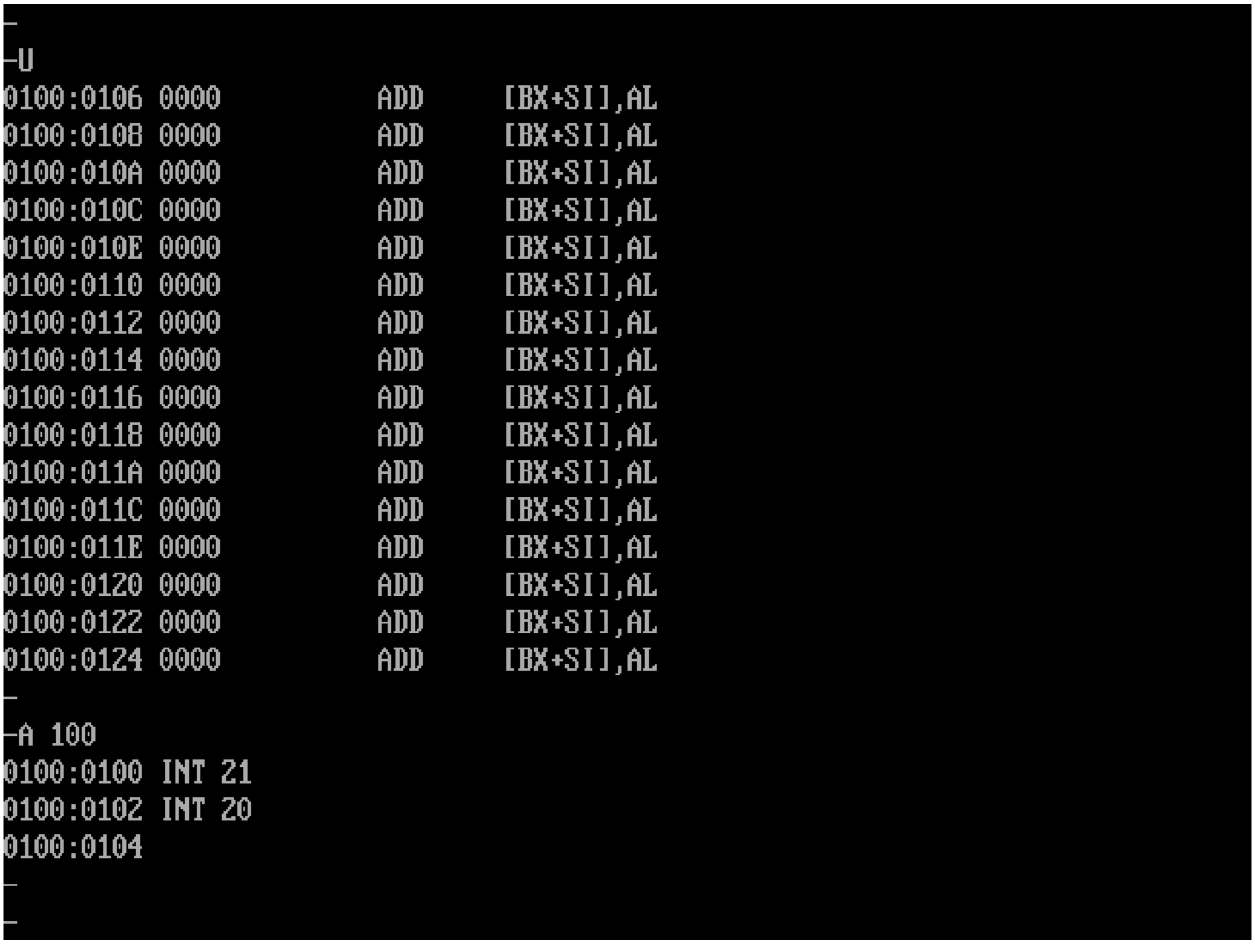
```
-R IP
IP 0102
:100
-
-R CS
CS 0745
:100
-
-G 106

AX=0200  BX=0000  CX=0000  DX=0043  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0100  IP=0106   NV UP EI NG NZ NA PO NC
0100:0106 0000          ADD     [BX+SI],AL                         DS:0000=CD
-
```

I get errors and faced with crashes couple of times during execution of -G command like in our tutorial, but after all we know the 43H is the ASCII equivalent of “C”.

Like below:



```
:200
-
-R DX
DX 0000
:43
-
-
-
-E 100
0745:0100  00.CD

-E 101
0745:0101  00.21

-R
AX=0200  BX=0000  CX=0000  DX=0043  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0100   NV UP EI PL NZ NA PO NC
0745:0100 CD21          INT     21
-
-G 102
C
AX=0243  BX=0000  CX=0000  DX=0043  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0102   NV UP EI PL NZ NA PO NC
0745:0102 0000          ADD     [BX+SI],AL                         DS:0000=CD
-
```



```
-
-U
0100:0106 0000          ADD     [BX+SI],AL
0100:0108 0000          ADD     [BX+SI],AL
0100:010A 0000          ADD     [BX+SI],AL
0100:010C 0000          ADD     [BX+SI],AL
0100:010E 0000          ADD     [BX+SI],AL
0100:0110 0000          ADD     [BX+SI],AL
0100:0112 0000          ADD     [BX+SI],AL
0100:0114 0000          ADD     [BX+SI],AL
0100:0116 0000          ADD     [BX+SI],AL
0100:0118 0000          ADD     [BX+SI],AL
0100:011A 0000          ADD     [BX+SI],AL
0100:011C 0000          ADD     [BX+SI],AL
0100:011E 0000          ADD     [BX+SI],AL
0100:0120 0000          ADD     [BX+SI],AL
0100:0122 0000          ADD     [BX+SI],AL
0100:0124 0000          ADD     [BX+SI],AL
-
-A 100
0100:0100 INT 21
0100:0102 INT 20
0100:0104
-
-
```

At last piece of code, **"A"** allows us to access assemble commands. What we did on the last 4 line is the end block that we struggle with much more lines above.