BAŞKENT UNIVERSITY

ENGINEERING FACULTY

ELECTRICAL-ELECTRONICS
ENGINEERING DEPARTMENT

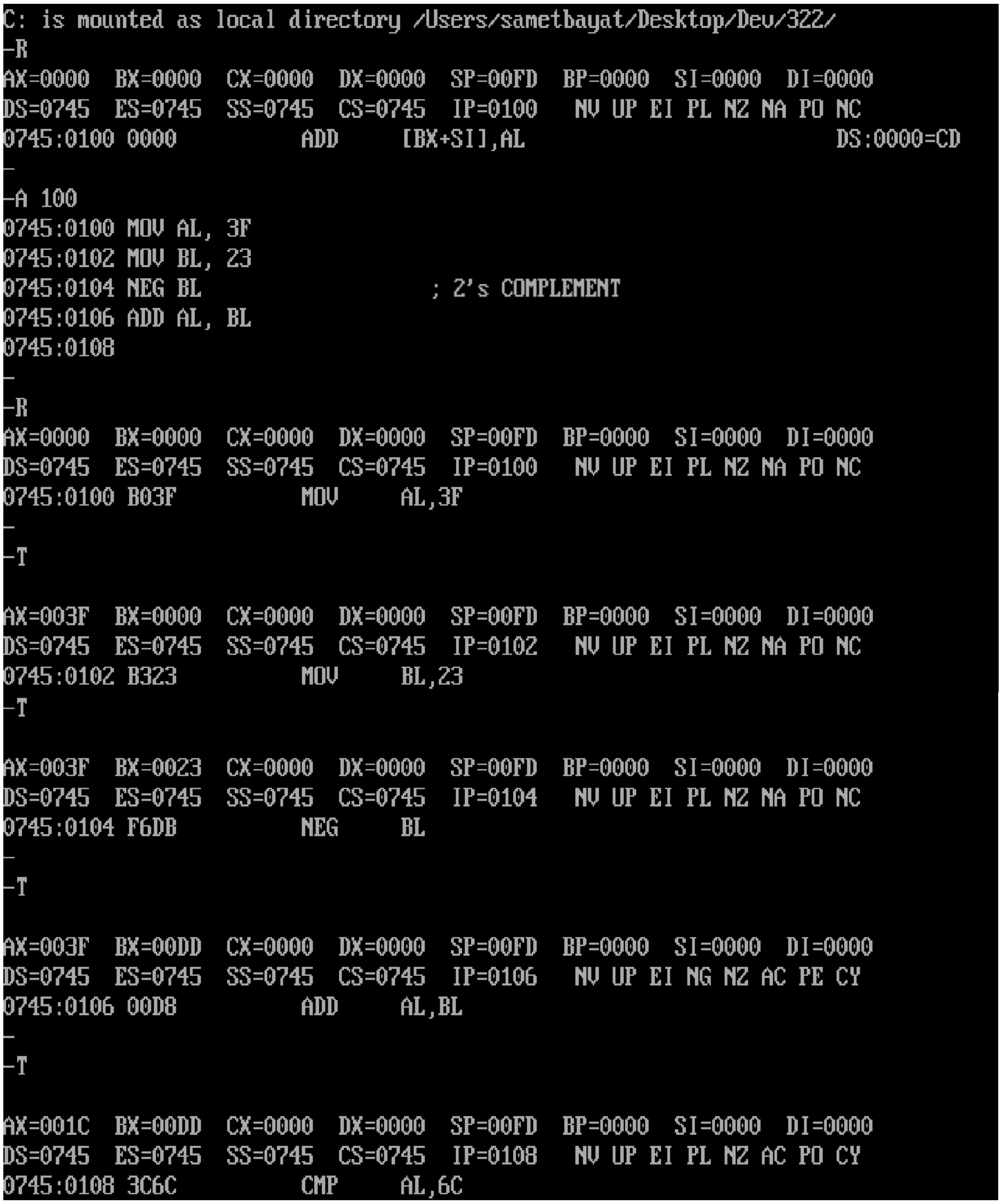
# EEM 322 - MICROPROCESSORS LAB

## EXPERIMENT NO. 04

SAMET BAYAT

22293730

1-) AL=3FH, BL=23H. Perform the AL-BL operation without using the SUB command.



```
C: is mounted as local directory /Users/sametbayat/Desktop/Dev/322/
-R
AX=0000  BX=0000  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:0000=CD
-
-A 100
0745:0100 MOV AL, 3F
0745:0102 MOV BL, 23
0745:0104 NEG BL                  ; 2's COMPLEMENT
0745:0106 ADD AL, BL
0745:0108
-
-R
AX=0000  BX=0000  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 B03F          MOV     AL,3F
-
-T

AX=003F  BX=0000  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 B323          MOV     BL,23
-T

AX=003F  BX=0023  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0104   NV UP EI PL NZ NA PO NC
0745:0104 F6DB          NEG     BL
-
-T

AX=003F  BX=00DD  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0106   NV UP EI NG NZ AC PE CY
0745:0106 00D8          ADD     AL,BL
-
-T

AX=001C  BX=00DD  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0108   NV UP EI PL NZ AC PO CY
0745:0108 3C6C          CMP     AL,6C
```

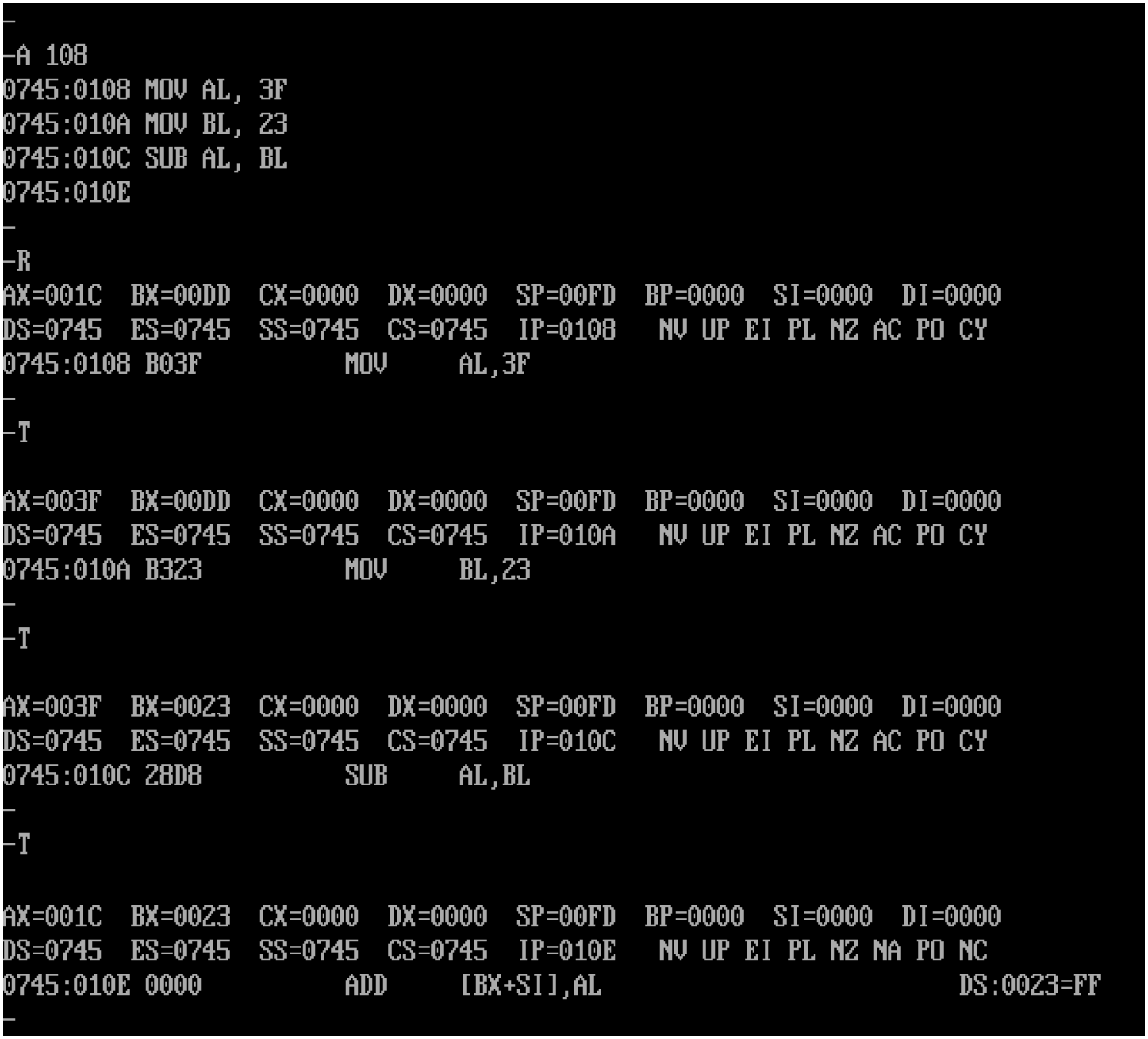
The NEG instruction in x86 assembly language computes the two's complement negation of a value. In other words, it flips all the bits in the binary representation of the value and then adds 1.

```
   ➔ 0010 0011       ;                     .. 23H
   ➔ 1101 1100       ; ALL BITS FLIPPED    .. DC
   ➔ 1101 1101       ; ADD 1               .. DD
                                               SYNTAX: NEG DESIRED_NUMBER
```

With using NEG, we convert subtraction to addition. After AL = AL + BL we see the result as 1C (as expected.)

In addition, Carry Flag set as 1. CY

2-) Apply SUB AL,BL operation to the registers using the same numbers.

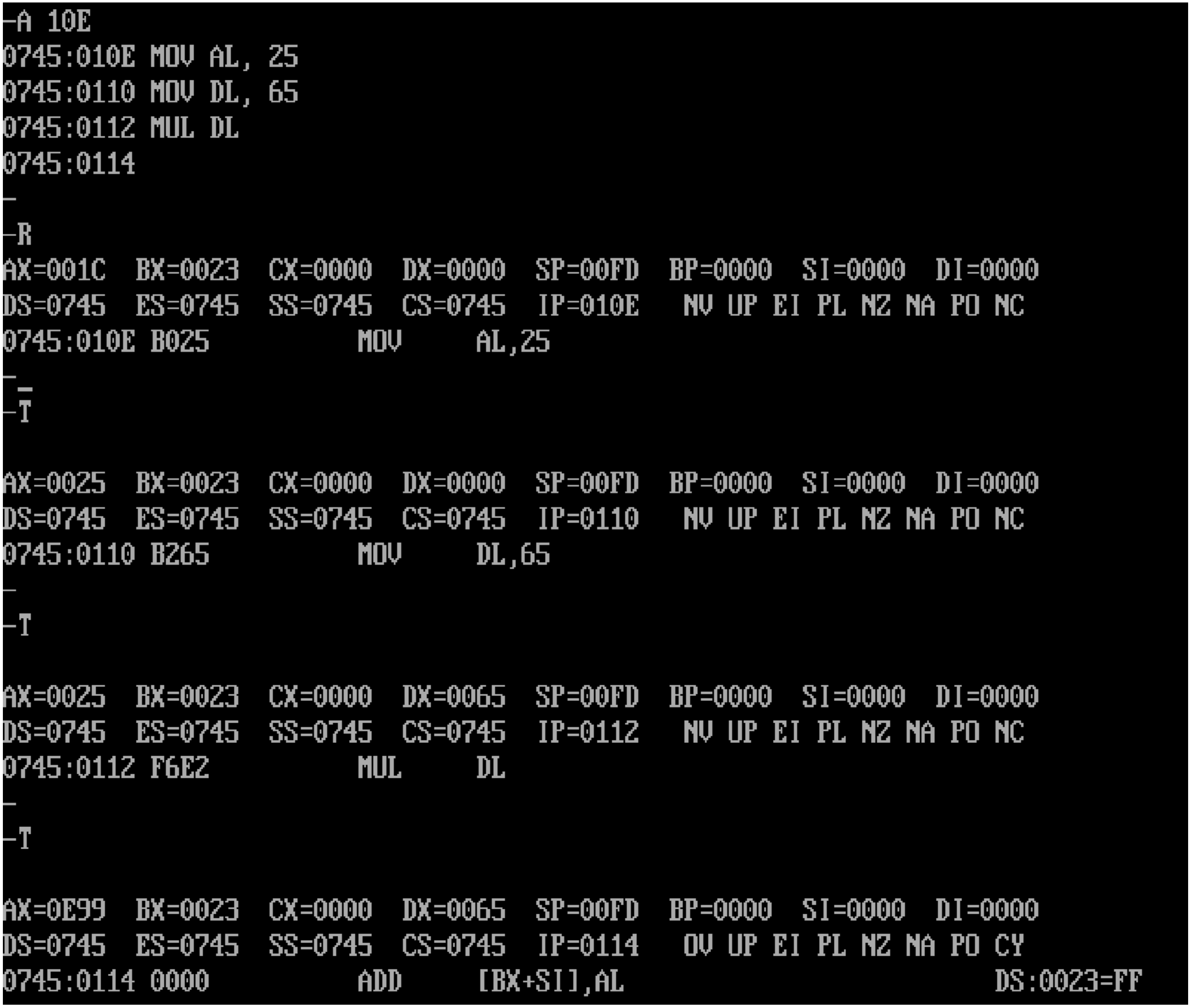


```
-
-A 108
0745:0108 MOV AL, 3F
0745:010A MOV BL, 23
0745:010C SUB AL, BL
0745:010E
-
-R
AX=001C  BX=00DD  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0108   NV UP EI PL NZ AC PO CY
0745:0108 B03F          MOV     AL,3F
-
-T

AX=003F  BX=00DD  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=010A   NV UP EI PL NZ AC PO CY
0745:010A B323          MOV     BL,23
-
-T

AX=003F  BX=0023  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=010C   NV UP EI PL NZ AC PO CY
0745:010C 28D8          SUB     AL,BL
-
-T

AX=001C  BX=0023  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=010E   NV UP EI PL NZ NA PO NC
0745:010E 0000          ADD     [BX+SI],AL                         DS:0023=FF
-
```

The result is again 1C as expected.

The difference from the 1st questions occurred in “Carry Flag”.
Because of the behavior of **SUB** command, the carry flag set to 0. NC

3-) AL=25H, DL=65H. Perform the AL*DL operation.



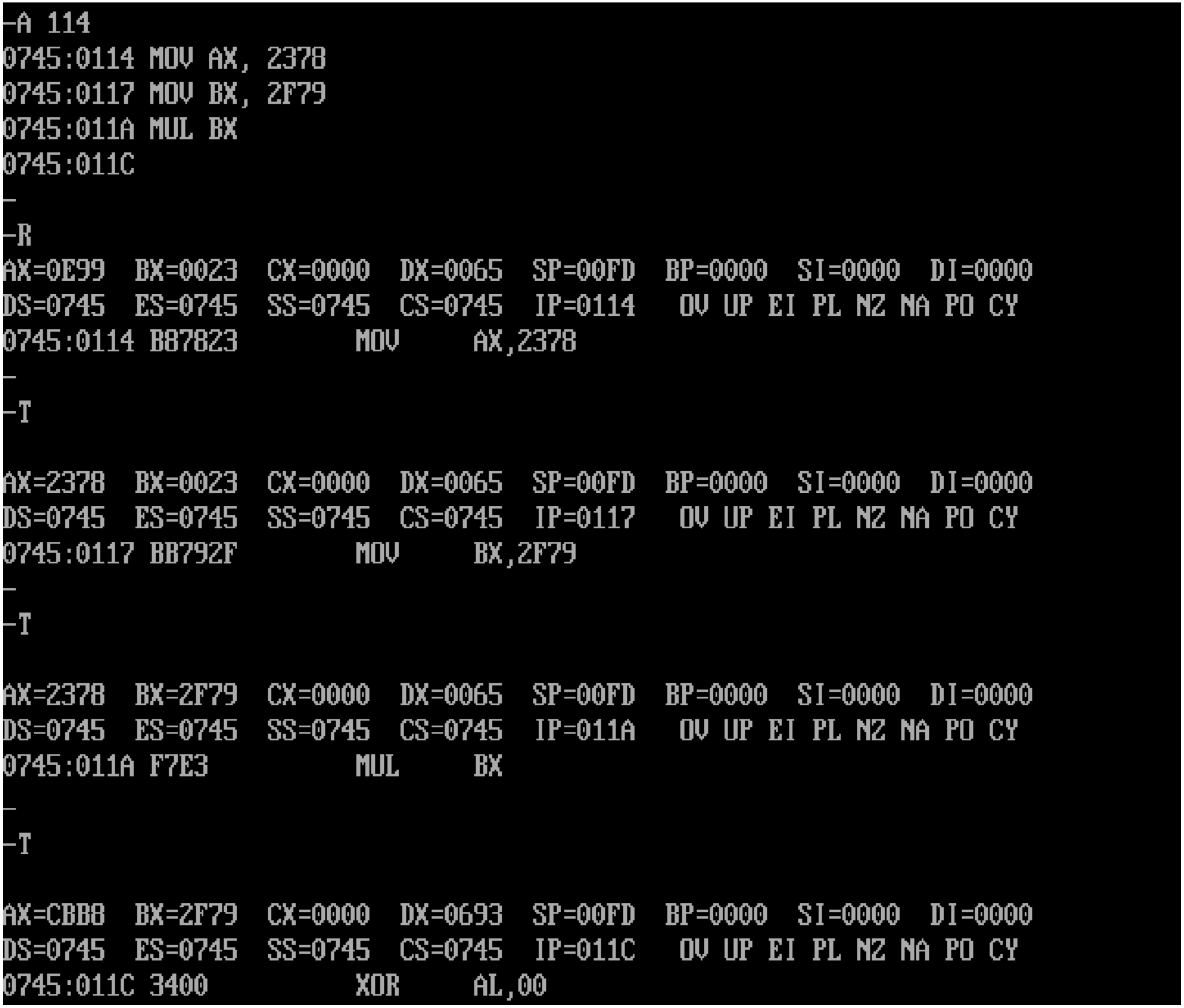
```
-A 10E
0745:010E MOV AL, 25
0745:0110 MOV DL, 65
0745:0112 MUL DL
0745:0114
-
-R
AX=001C  BX=0023  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=010E   NV UP EI PL NZ NA PO NC
0745:010E B025          MOV     AL,25
-_
-T

AX=0025  BX=0023  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0110   NV UP EI PL NZ NA PO NC
0745:0110 B265          MOV     DL,65
-
-T

AX=0025  BX=0023  CX=0000  DX=0065  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0112   NV UP EI PL NZ NA PO NC
0745:0112 F6E2          MUL     DL
-
-T

AX=0E99  BX=0023  CX=0000  DX=0065  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0114   OV UP EI PL NZ NA PO CY
0745:0114 0000          ADD     [BX+SI],AL                         DS:0023=FF
```

For 8 bit numbers, result of multiplication can be max. 16 bit. Therefore, the result of **MUL** saved in default 16-bit default register AX. As we expected the result is **0E99.**

**NOTE:** Numbers are not signed. That’s why we used **MUL**, instead of **IMUL**.

**SYNTAX: MUL NUMBER_OTHER_THAN_AL_AX**

4-) AX=2378H, BX=2F79H. Perform the AX*BX operation.



```
-A 114
0745:0114 MOV AX, 2378
0745:0117 MOV BX, 2F79
0745:011A MUL BX
0745:011C
-
-R
AX=0E99  BX=0023  CX=0000  DX=0065  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0114   OV UP EI PL NZ NA PO CY
0745:0114 B87823        MOV     AX,2378
-
-T

AX=2378  BX=0023  CX=0000  DX=0065  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0117   OV UP EI PL NZ NA PO CY
0745:0117 BB792F        MOV     BX,2F79
-
-T

AX=2378  BX=2F79  CX=0000  DX=0065  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=011A   OV UP EI PL NZ NA PO CY
0745:011A F7E3          MUL     BX
-
-T

AX=CBB8  BX=2F79  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=011C   OV UP EI PL NZ NA PO CY
0745:011C 3400          XOR     AL,00
```

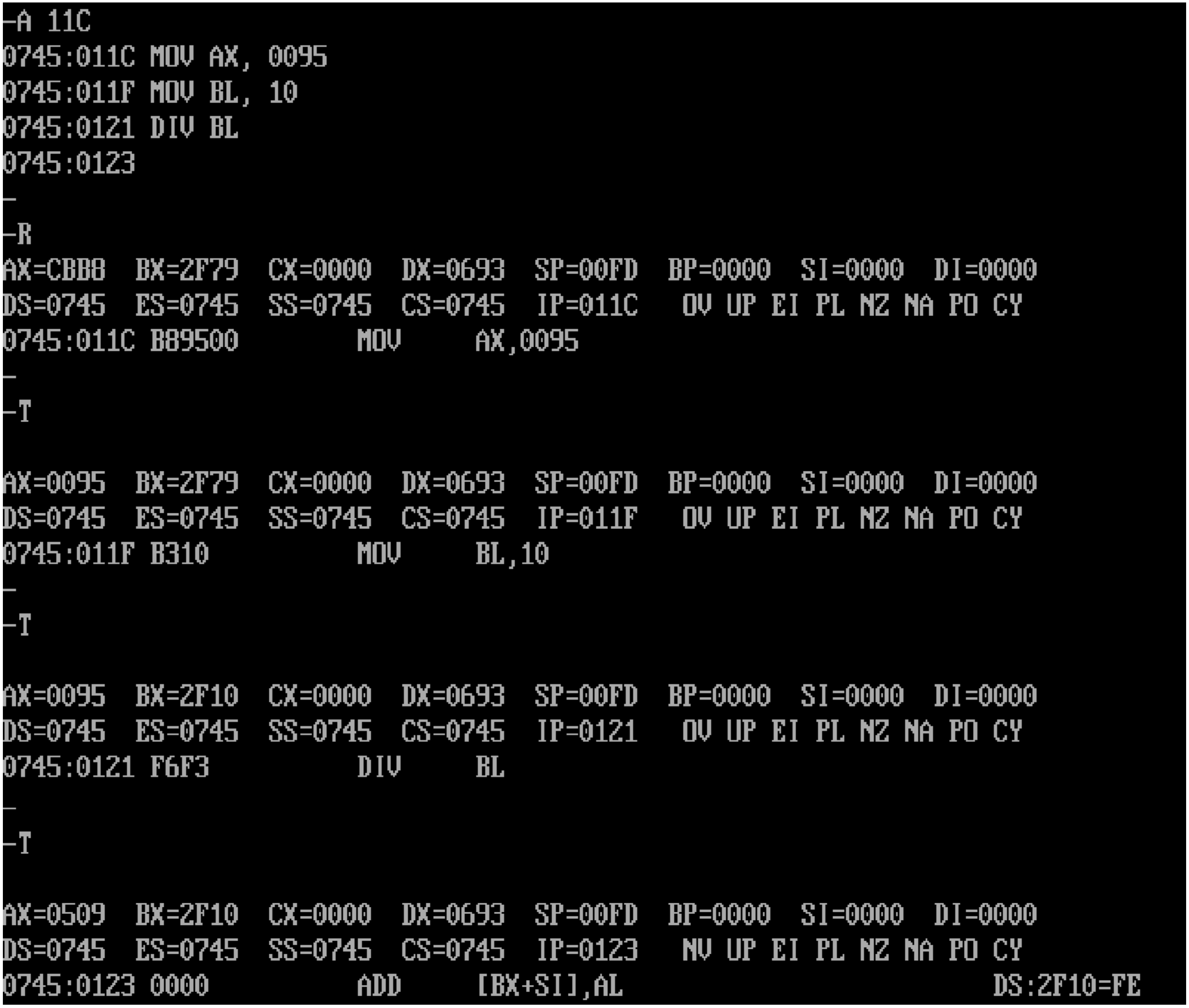
Now again we need to multiply two unsigned numbers. This time numbers are 16-bit. Result of multiplication can be max. 32 bits. As we know well, the x8086 processor is not capable to store 32 bits information in 1 register. At this very point, we need to split the result into two 16-bit numbers. Logically, our result will be **DX:AX**.

If we use calculator for same operation, the result shown as 0x693CBB8.
If we separate this number into two 16-bit numbers, what we get is:

```
693 |CBB8
 DX | AX
```

It is seen that; the desired output and the program output matches.

5-) AL=95H, BL=10H. Perform the AL/BL operation.



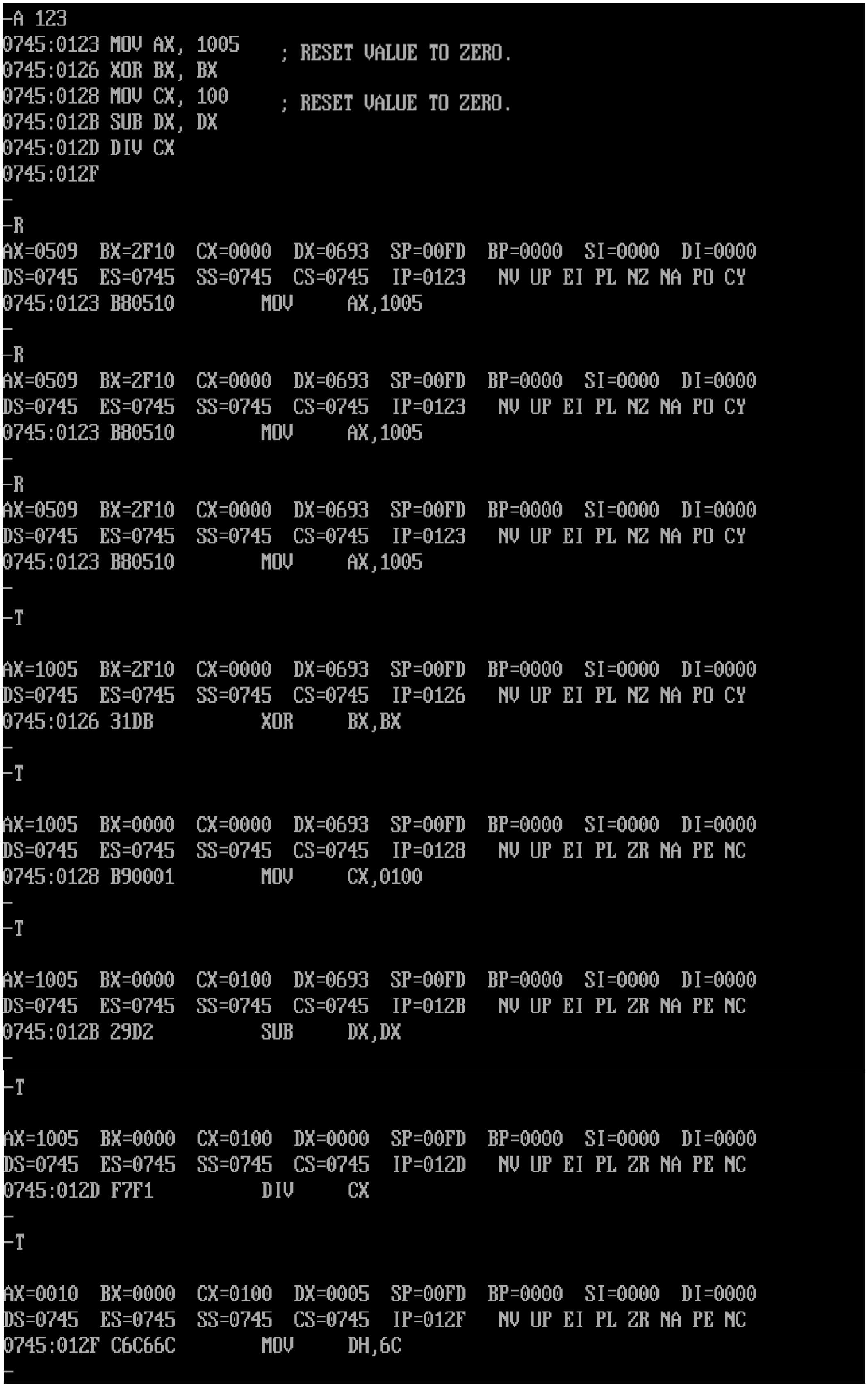
```
-A 11C
0745:011C MOV AX, 0095
0745:011F MOV BL, 10
0745:0121 DIV BL
0745:0123
-
-R
AX=CBB8  BX=2F79  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=011C   OV UP EI PL NZ NA PO CY
0745:011C B89500        MOV     AX,0095
-
-T

AX=0095  BX=2F79  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=011F   OV UP EI PL NZ NA PO CY
0745:011F B310          MOV     BL,10
-
-T

AX=0095  BX=2F10  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0121   OV UP EI PL NZ NA PO CY
0745:0121 F6F3          DIV     BL
-
-T

AX=0509  BX=2F10  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0123   NV UP EI PL NZ NA PO CY
0745:0123 0000          ADD     [BX+SI],AL                         DS:2F10=FE
```

Same for multiplication, 8-bit divisions’ datas stored in register AX.
(AL -> quotient, AH -> reminder)

$95_{16}$ / $10_{16}$ = $9_{16}$ (reminder: $5_{16}$)

AX : (AH = $5_{16}$ AL = $9_{16}$) = 0509 (The application satisfies this result.)

6-) AX=1005H, BX=0000H, CX=100H, DX=0000. Perform the AX/CX operation and compare the contents of the registers with the 5th question and comment.



```
-A 123
0745:0123 MOV AX, 1005
0745:0126 XOR BX, BX      ; RESET VALUE TO ZERO.
0745:0128 MOV CX, 100
0745:012B SUB DX, DX      ; RESET VALUE TO ZERO.
0745:012D DIV CX
0745:012F
-
-R
AX=0509  BX=2F10  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0123   NV UP EI PL NZ NA PO CY
0745:0123 B80510        MOV     AX,1005
-
-R
AX=0509  BX=2F10  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0123   NV UP EI PL NZ NA PO CY
0745:0123 B80510        MOV     AX,1005
-
-R
AX=0509  BX=2F10  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0123   NV UP EI PL NZ NA PO CY
0745:0123 B80510        MOV     AX,1005
-
-T

AX=1005  BX=2F10  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0126   NV UP EI PL NZ NA PO CY
0745:0126 31DB          XOR     BX,BX
-
-T

AX=1005  BX=0000  CX=0000  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0128   NV UP EI PL ZR NA PE NC
0745:0128 B90001        MOV     CX,0100
-
-T

AX=1005  BX=0000  CX=0100  DX=0693  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=012B   NV UP EI PL ZR NA PE NC
0745:012B 29D2          SUB     DX,DX
-
-T

AX=1005  BX=0000  CX=0100  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=012D   NV UP EI PL ZR NA PE NC
0745:012D F7F1          DIV     CX
-
-T

AX=0010  BX=0000  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=012F   NV UP EI PL ZR NA PE NC
0745:012F C6C66C        MOV     DH,6C
-
```

There are many ways to set a register value as 0. Here, two of them demonstrated.

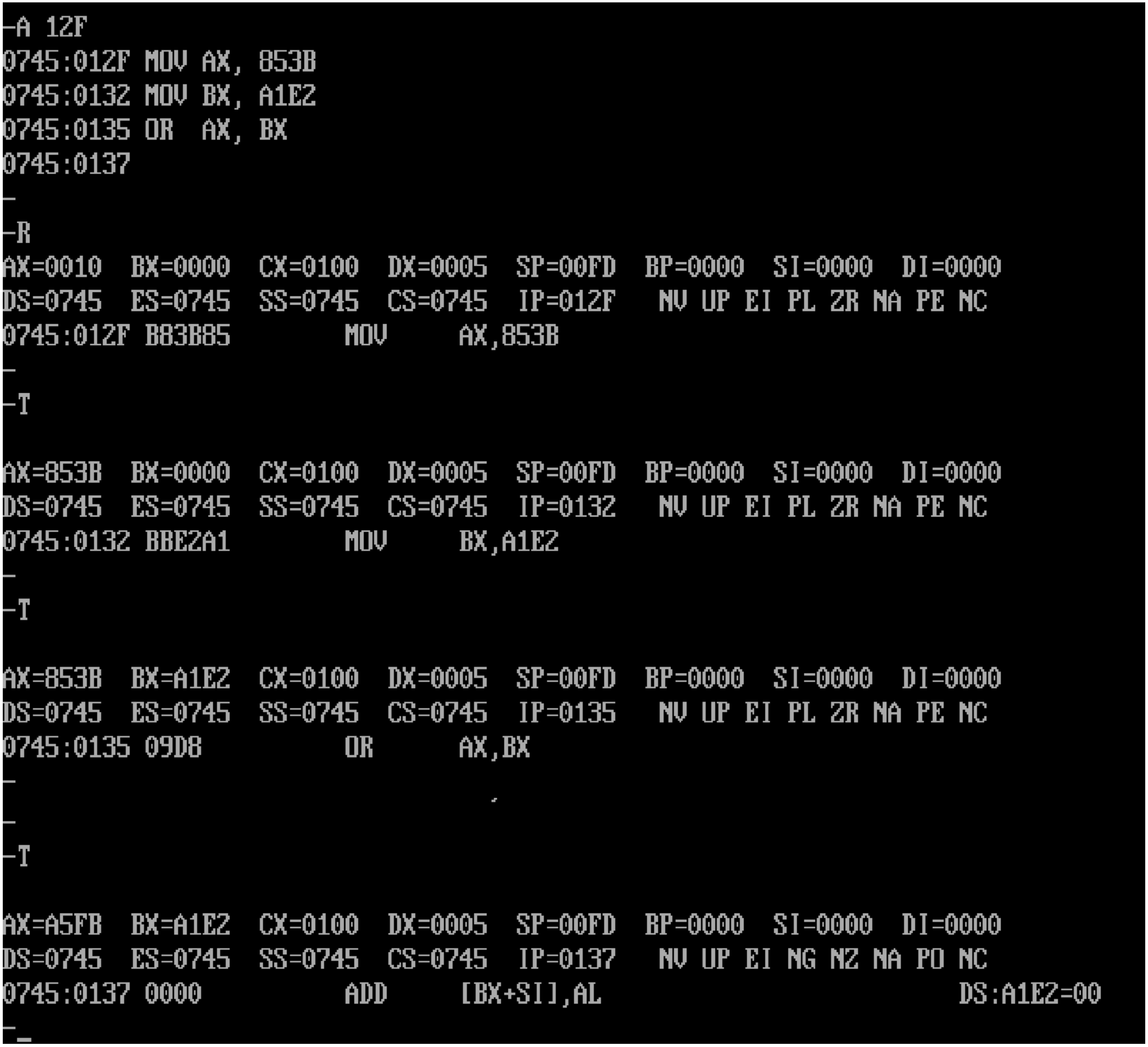
XOR operates on two binary values, often denoted as A and B. The result of the XOR operation between A and B is 1 if either A or B is 1, but not both. If both A and B are either 0 or 1, the result is 0

And simply if we subtract the number from itself, it gives the result as 0.

For division, since we dealt with 16-bit number, we see the results in separate registers.
(AX -> quotient)
(DX -> reminder)

In HEX,
1005 / 100 = 10 -> AX
(Reminder: 5)   -> DX

7-) AX=853B, BX=A1E2H OR AX and BX and print the result to AX

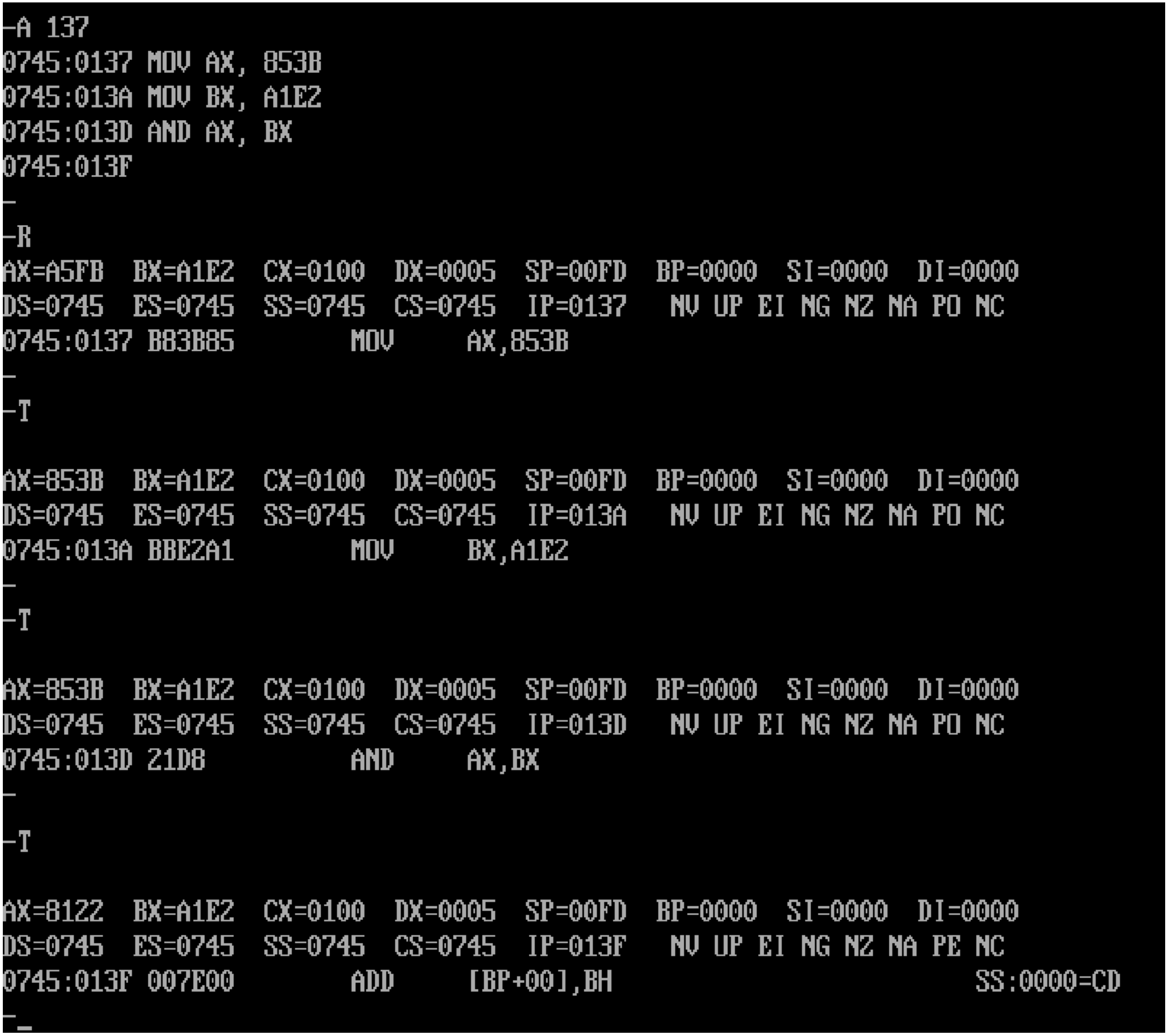


```
-A 12F
0745:012F MOV AX, 853B
0745:0132 MOV BX, A1E2
0745:0135 OR  AX, BX
0745:0137
-
-R
AX=0010  BX=0000  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=012F   NV UP EI PL ZR NA PE NC
0745:012F B83B85        MOV     AX,853B
-
-T

AX=853B  BX=0000  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0132   NV UP EI PL ZR NA PE NC
0745:0132 BBE2A1        MOV     BX,A1E2
-
-T

AX=853B  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0135   NV UP EI PL ZR NA PE NC
0745:0135 09D8          OR      AX,BX
-
-
-T

AX=A5FB  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0137   NV UP EI NG NZ NA PO NC
0745:0137 0000          ADD     [BX+SI],AL                         DS:A1E2=00
-_
```

In assembly language, the OR instruction performs a bitwise OR operation between two operands. The OR instruction takes two operands and performs a logical OR operation between each bit of the operands. The result of the OR operation is stored in the destination operand.

```
1000 0101 0011 1011
1010 0001 1110 0010
-------------------
1010 0101 1111 1011
```

8-) AND operation AX=853B, BX=A1E2H AX and BX and print the result to AX

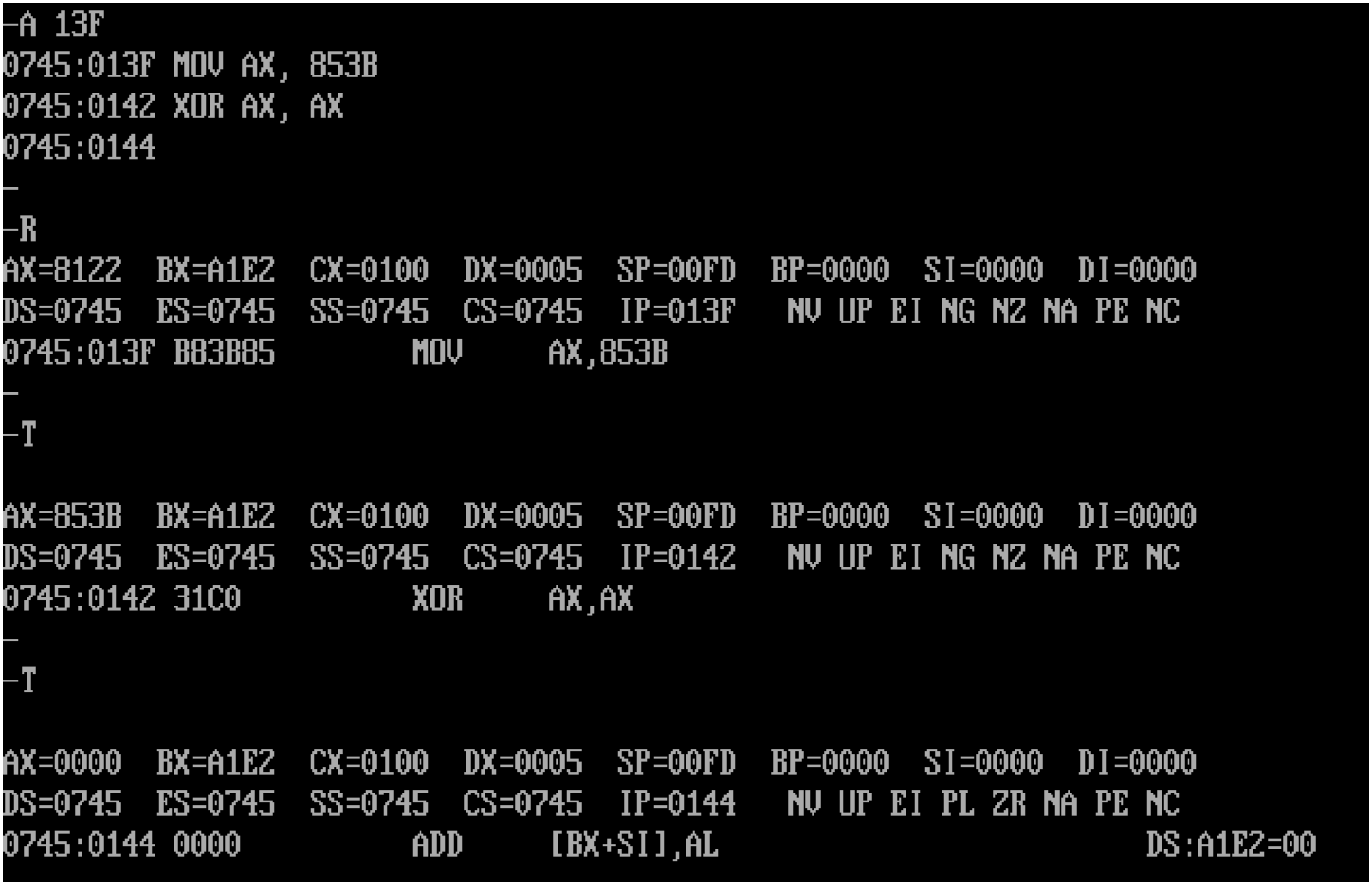


```
-A 137
0745:0137 MOV AX, 853B
0745:013A MOV BX, A1E2
0745:013D AND AX, BX
0745:013F
-
-R
AX=A5FB  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0137   NV UP EI NG NZ NA PO NC
0745:0137 B83B85        MOV     AX,853B
-
-T

AX=853B  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=013A   NV UP EI NG NZ NA PO NC
0745:013A BBE2A1        MOV     BX,A1E2
-
-T

AX=853B  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=013D   NV UP EI NG NZ NA PO NC
0745:013D 21D8          AND     AX,BX
-
-T

AX=8122  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=013F   NV UP EI NG NZ NA PE NC
0745:013F 007E00        ADD     [BP+00],BH                         SS:0000=CD
-_
```

AND operation is a binary bitwise operation that takes two binary numbers as operands and performs a logical AND operation on each pair of corresponding bits, producing a new binary number as a result.

```
 1000 0101 0011 1011
 1010 0001 1110 0010
---------------------
 1000 0001 0010 0010
```

9-) Perform the operation that XORs the number AX=853B with itself.



```
-A 13F
0745:013F MOV AX, 853B
0745:0142 XOR AX, AX
0745:0144
-
-R
AX=8122  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=013F   NV UP EI NG NZ NA PE NC
0745:013F B83B85        MOV     AX,853B
-
-T

AX=853B  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0142   NV UP EI NG NZ NA PE NC
0745:0142 31C0          XOR     AX,AX
-
-T

AX=0000  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0144   NV UP EI PL ZR NA PE NC
0745:0144 0000          ADD     [BX+SI],AL                         DS:A1E2=00
```

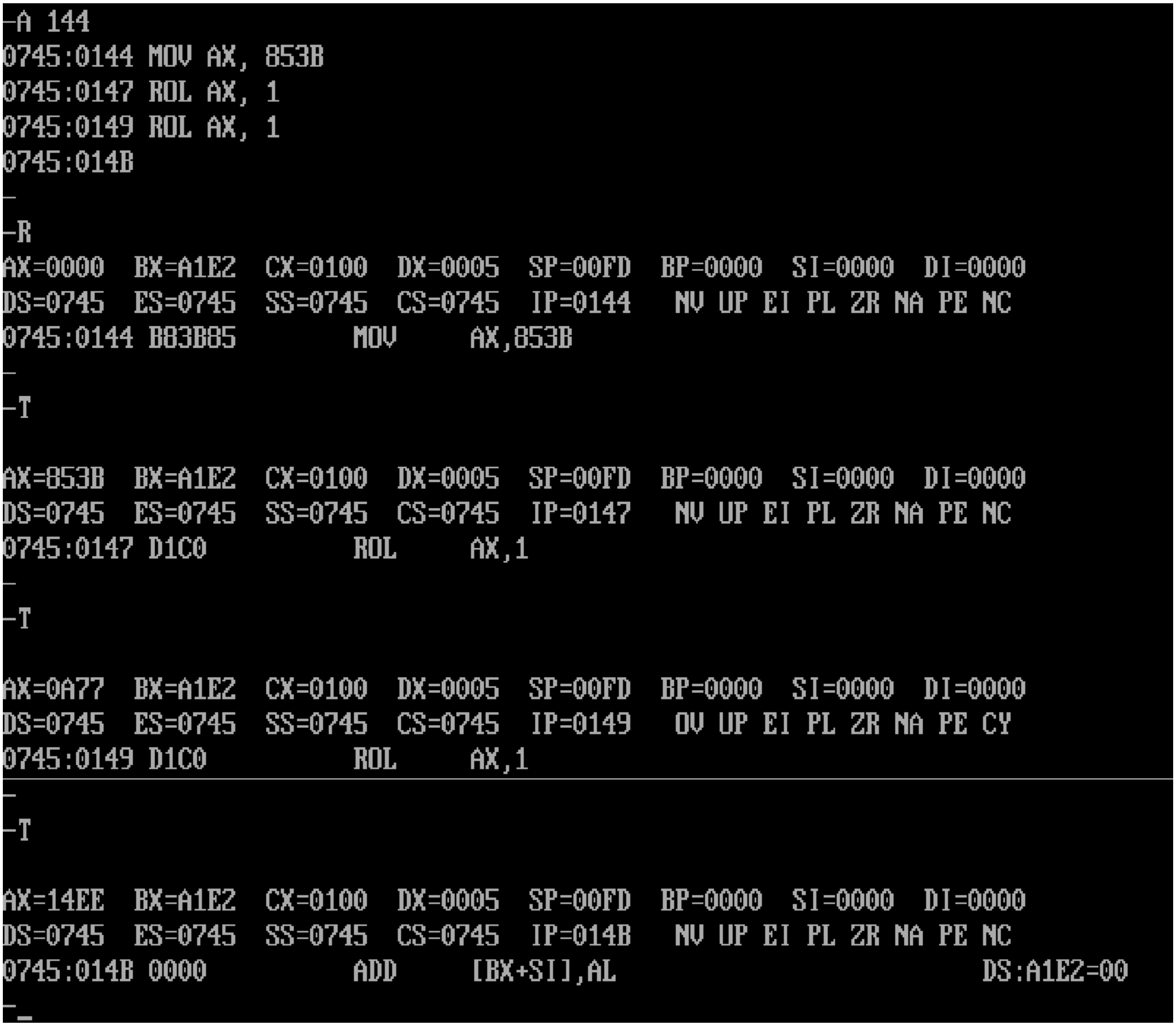
As I mentioned in Q6, XOR (exclusive OR) operation is a binary bitwise operation that takes two binary numbers as operands and performs a logical XOR operation on each pair of corresponding bits, producing a new binary number as a result.

| A | B | Out |
|---|---|---|
| 0 | 0 | **0** |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | **0** |

For our case, since the operation covers XOR operation with same number, the result for each bit became 0.

We can observe this in both AX register and “Zero Flag” (ZR).

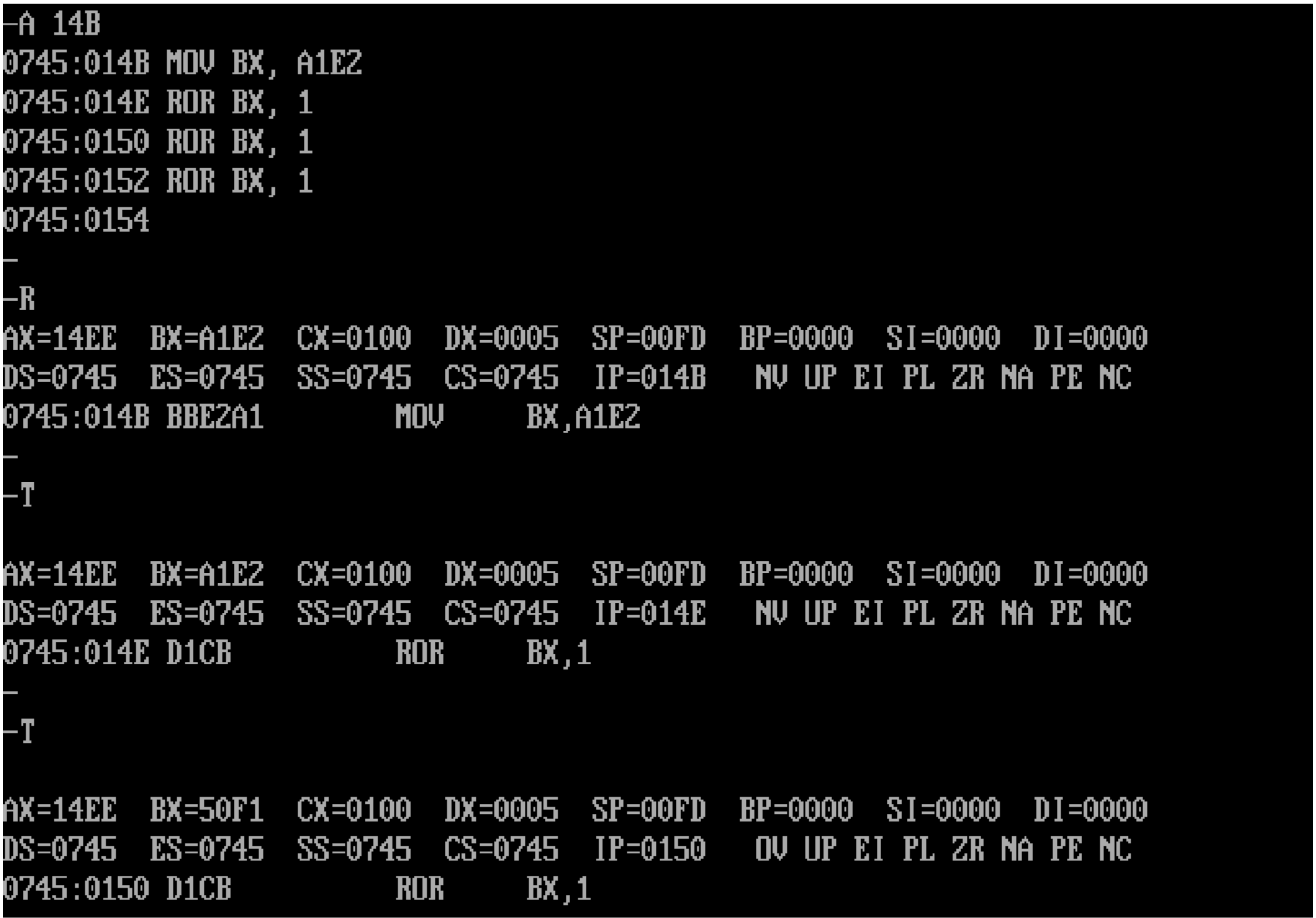
10-) Perform the operations that rotate the number AX=853B to the left 2 times and the number of BX=A1E2H to the right 3 times.



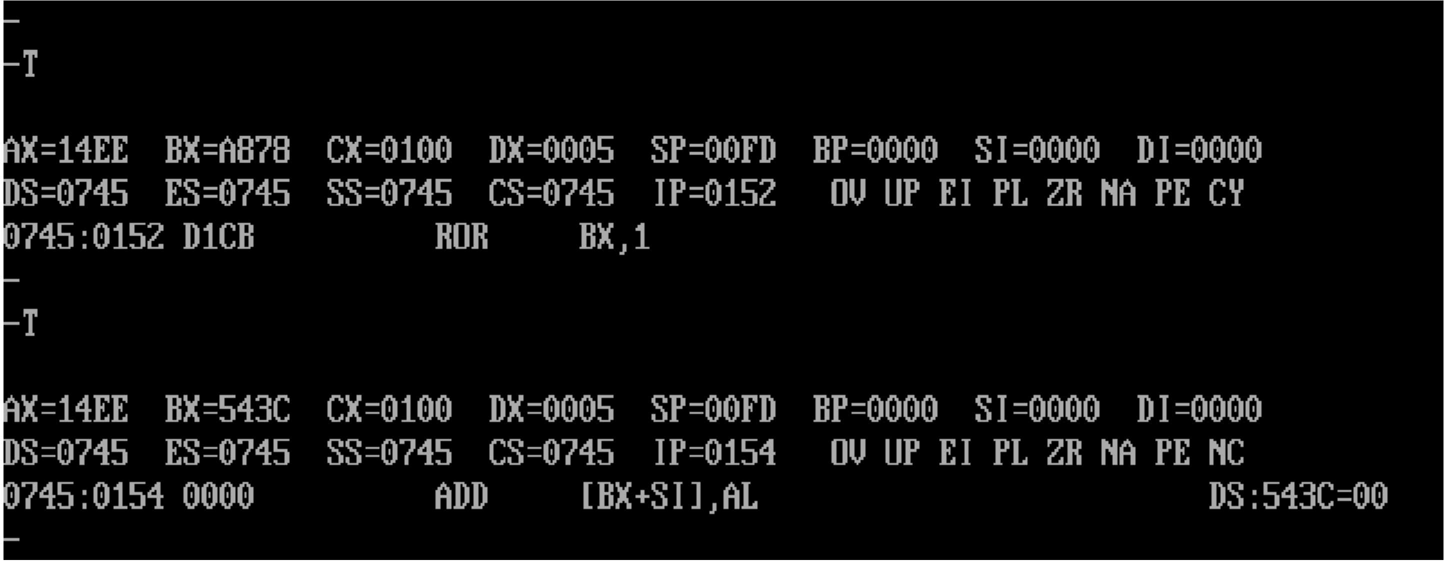
```
-A 144
0745:0144 MOV AX, 853B
0745:0147 ROL AX, 1
0745:0149 ROL AX, 1
0745:014B
-
-R
AX=0000  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0144   NV UP EI PL ZR NA PE NC
0745:0144 B83B85        MOV     AX,853B
-
-T

AX=853B  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0147   NV UP EI PL ZR NA PE NC
0745:0147 D1C0          ROL     AX,1
-
-T

AX=0A77  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0149   OV UP EI PL ZR NA PE CY
0745:0149 D1C0          ROL     AX,1
-
-T

AX=14EE  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=014B   NV UP EI PL ZR NA PE NC
0745:014B 0000          ADD     [BX+SI],AL                         DS:A1E2=00
-_
```

ROL (rotate left) and ROR (rotate right) are bitwise shift operations in computer programming.
In a bitwise ROL operation, the bits of a binary number are shifted to the left by a certain number of positions, and the bits that fall off the left end are added to the right end. This effectively "rotates" the bits to the left.

```
                     < 1000 0101 0011 1011
( CY ) CARRY = 1 < 0000 1010 0111 0110
( CN ) CARRY = 0 < 0001 0100 1110 1110
```



```
-A 14B
0745:014B MOV BX, A1E2
0745:014E ROR BX, 1
0745:0150 ROR BX, 1
0745:0152 ROR BX, 1
0745:0154
-
-R
AX=14EE  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=014B   NV UP EI PL ZR NA PE NC
0745:014B BBE2A1        MOV     BX,A1E2
-
-T

AX=14EE  BX=A1E2  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=014E   NV UP EI PL ZR NA PE NC
0745:014E D1CB          ROR     BX,1
-
-T

AX=14EE  BX=50F1  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0150   OV UP EI PL ZR NA PE NC
0745:0150 D1CB          ROR     BX,1
```



```
-
-T

AX=14EE  BX=A878  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0152   OV UP EI PL ZR NA PE CY
0745:0152 D1CB          ROR     BX,1
-
-T

AX=14EE  BX=543C  CX=0100  DX=0005  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0745  ES=0745  SS=0745  CS=0745  IP=0154   OV UP EI PL ZR NA PE NC
0745:0154 0000          ADD     [BX+SI],AL                         DS:543C=00
-
```

In a bitwise ROR operation, the bits of a binary number are shifted to the right by a certain number of positions, and the bits that fall off the right end are added to the left end. This effectively "rotates" the bits to the right.

```
1010 0001 1110 0010 >
0101 0000 1111 0001 > CARRY = 0 NC
1010 1000 0111 1000 > CARRY = 1 CY
0001 0100 1110 1110 > CARRY = 0 NC
```