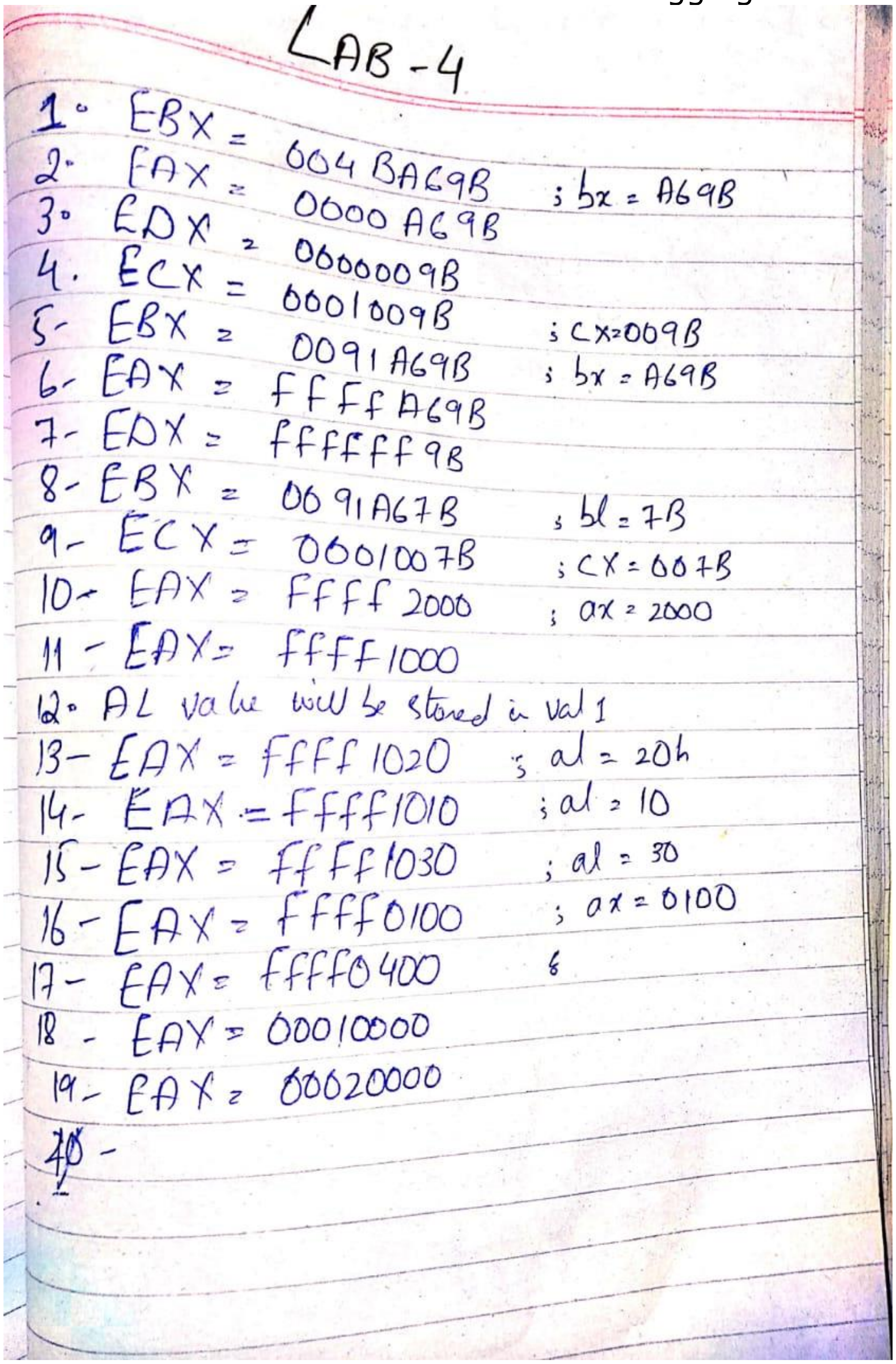
SAAD UR REHMAN
19k-0218
SEC: C

COAL LAB TASK 4

Q1)What are the values of the registers and the variables after each group of instructions in the following program.

Q1 A)

SOL: I have written the values after debugging each step:



LAB-4

1. EBX = 004BA69B ; bx = A69B
2. EAX = 0000A69B
3. EDX = 0000009B
4. ECX = 0001009B
5- EBX = 0091A69B ; CX=009B
6- EAX = FFFFA69B ; bx = A69B
7- EDX = FFFFFF9B
8- EBX = 0091A67B ; bl = 7B
9- ECX = 0001007B ; CX = 007B
10- EAX = FFFF2000 ; ax = 2000
11 - EAX= FFFF1000
12. AL value will be stored in val 1
13- EAX = FFFF1020 ; al = 20h
14- EAX = FFFF1010 ; al = 10
15- EAX = FFFF1030 ; al = 30
16- EAX = FFFF0100 ; ax = 0100
17- EAX= FFFF0400
18 - EAX = 00010000
19- EAX = 00020000
20 -

SAAD UR REHMAN
19k-0218
SEC: C

(b)

```
TITLE Data Transfer Examples (Test.asm)
INCLUDE Irvine32.inc
.data
val1 WORD 2000h
val2 WORD 1000h
arrayB BYTE 20h,10h,30h,40h,50h
arrayW WORD 100h,400h,300h
arrayD DWORD 10000h,20000h
.code
main PROC
mov bx,0A69Bh
    ➔ The value 0A69Bh will be stored in bx Register
movzx eax,bx
    ➔ Bx will will be moved to eax with the extended zeros, eax will store the lower 16
      bit value while other 16 bit will be all zero's
movzx edx,bl
    ➔ Bl value will be moved to edx register with extended zero's
movzx cx,bl
    ➔ Bl value will be moved to cx register with extended zero'
mov bx,0A69Bh
    ➔ Value 0A69B will move to bx Register.
movsx eax,bx
    ➔ Bx value will be moved to eax register with extended one's (EAX=FFFFA69B).
movsx edx,bl
    ➔ Bl value will be moved to edx register with extended one's
mov bl,7Bh
    ➔ 7B value will be moved in bx register.
movsx cx,bl
    ➔ Value of bl which is 7B will store in cs refister with extended ones .
  mov ax,val1
    ➔ Val1 value(2000h) will be moved to ax register
xchg ax,val2
    ➔ In this instruction, value of ax register will be exchanged to val2.
      mov val1,ax
    ➔ Ax register value has been moved to val1 variable.
mov al,arrayB
    ➔ Value of arrayB moved to al register which is 20h.
mov al,[arrayB+1]
    ➔ As the arrayB is of size BYTE, we increment with one to get the next value which
      is 10h to store in al register.
mov al,[arrayB+2]
    ➔ As the arrayB is of size BYTE, we increment with one again to get the next value
      which is 30h to store in al register.
mov ax,arrayW
    ➔ Moved the value of arrayW in ax register.
mov ax,[arrayW+2]
    ➔ As the arrayW is of size WORD, we increment with two to get the next value which
      is 400h to store in ax register.
mov eax,arrayD
    ➔ arrayD has been moved to eax register.

mov eax,[arrayD+4]
```

- ➔ As the arrayD is of size DWORD, we increment with four to get the next value which is 2000h to store in al register.

```
 mov eax,[arrayD+TYPE arrayD]
```

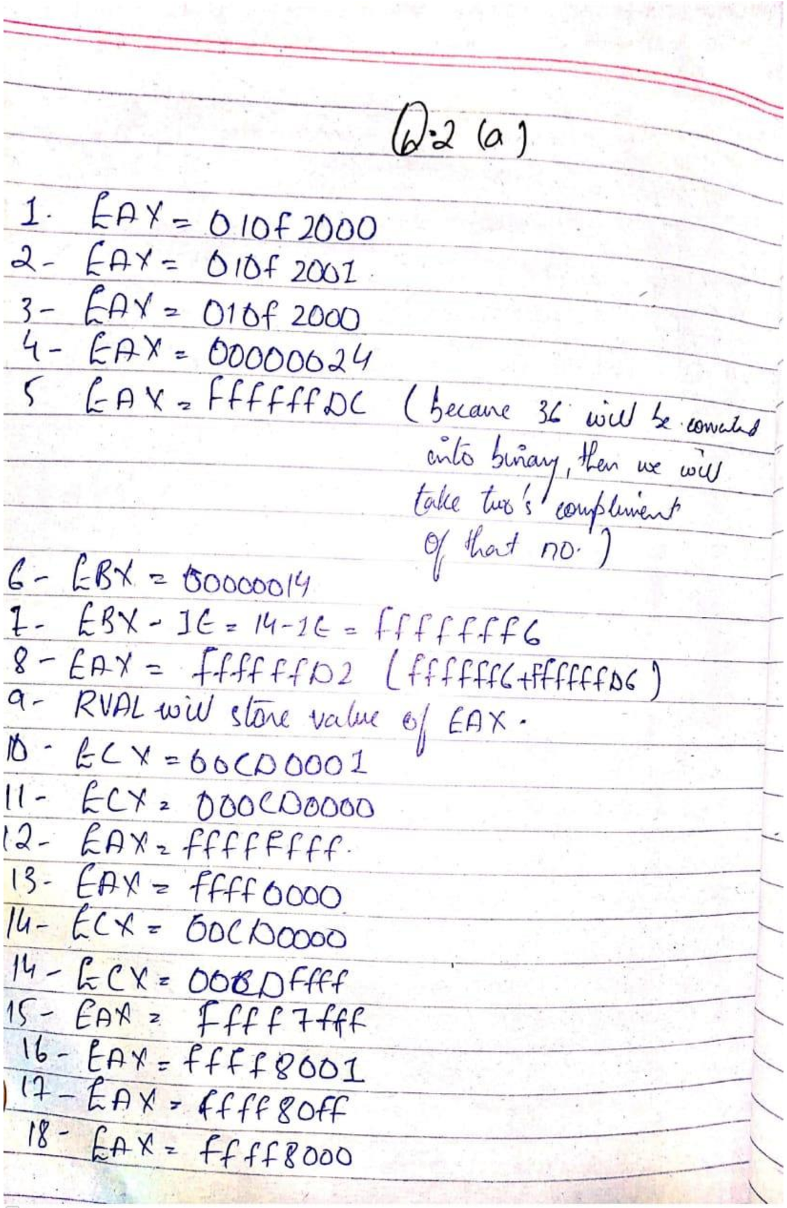
- ➔ Add the value of arrayD with TYPE arrayD and then moved it to eax register.

```
exit
main ENDP
END main
```

Q2) What are the values of the registers and the variables after each group of instructions in the following program .
a)Put the break point and notice the value of register in the register window and Write down the value of output (i.e register value) or attached the snips of each step.

SOL: I have written values after debugging each step.

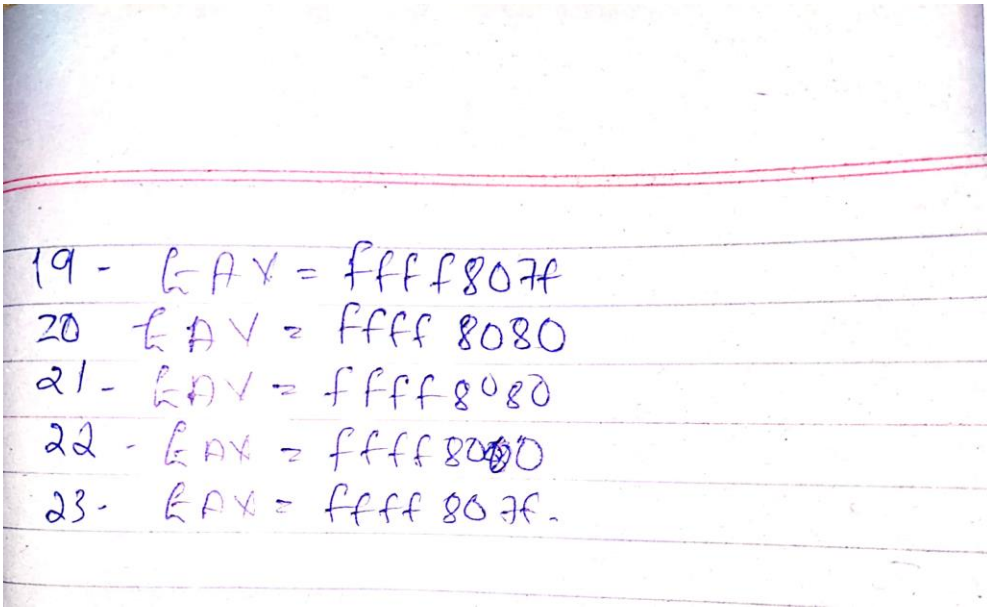
SAAD UR REHMAN
19k-0218
SEC: C



Q.2 (a)

1. EAX = 010f2000
2- EAX = 010f 2001
3- EAX = 010f 2000
4- EAX = 00000024
5 EAX = ffffffDC (because 36 will be converted
into binary, then we will
take two's compliment
of that no.)
6- EBX = 00000014
7- EBX - 1E = 14-1E = fffffff6
8- EAX = ffffffD2 (fffffff6+ffffffD6)
9- RVAL will store value of EAX.
10 - ECX = 00CD0001
11- ECX = 000CD0000
12- EAX = ffffffff
13- EAX = ffff0000
14- ECX = 00CD0000
14- ECX = 00BDffff
15- EAX = ffff7fff
16- EAX = ffff8001
17- EAX = ffff80ff
18- EAX = ffff8000

SAAD UR REHMAN
19k-0218
SEC: C



19 - EAX = ffff807f
20 EAX = ffff 8080
21 - EAX = ffff8080
22 - EAX = ffff8000
23 - EAX = ffff 807f.

(B)

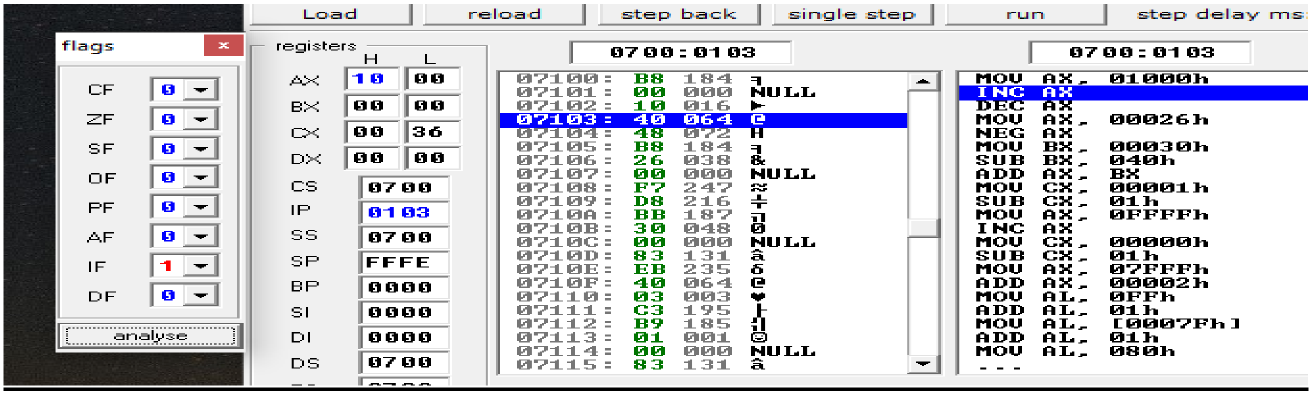
```
TITLE Addition and Subtraction (lab4.asm)
INCLUDE Irvine32.inc
.data
Rval SDWORD ?
Xval SDWORD 36
Yval SDWORD 20
Zval SDWORD 30
.code
main PROC
; INC and DEC
mov ax,2000h → Value 2000h will be mov to ax register
inc ax       → increment in ax register.
dec ax       → decrement in ax register.
mov eax,Xval → we move the value of Xval in eax register.
neg eax      → it will takes the 2’s complement of a number store in eax.
mov ebx,Yval → moving the value of Yval in ebx register.
sub ebx,Zval → In this instruction, we subtact the value of Zval from ebx register.
add eax,ebx  → adding eax register with ebx.
mov Rval,eax → moving the eax value in Rval variable.
mov cx,1     → move 1 in cx register.
sub cx,1     → subtract 1 from cx register.
mov ax,0FFFFh →move 0FFFFh in ax register.
inc ax        →incrementing the ax register.
mov cx,0      →moving 0 in cx register.
sub cx,1      →moving 1 in cx register.
mov ax,7FFFh  →we move 7FFFh in ax register.
add ax,2      →adding 2 in ax register.
mov al,0FFh   →moving 0FFh in al register.
add al,1      →adding 1 in al register.
mov al,+127   →moving positive 127 in al .
add al,1      →adding 1 in al.
mov al,-128   →moving -128 in al.
```

SAAD UR REHMAN
19k-0218
SEC: C

```
sub al,1       → subtracting 1 from al register.
exit
main ENDP
END main
```

Q3 RUN the following program at emu8086 and notice the value of lower byte and higher byte register and status of CPU flag bit. Attached the output of running program. Also write or attached each step of output .

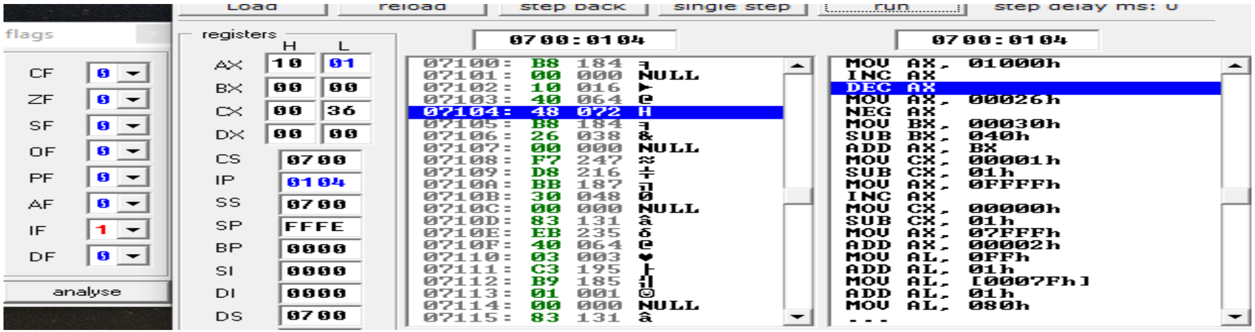
STEP 01:



```
flags          registers  H   L
CF 0           AX        10  00
ZF 0           BX        00  00
SF 0           CX        00  36
OF 0           DX        00  00
PF 0           CS        0700
AF 0           IP        0103
IF 1           SS        0700
DF 0           SP        FFFE
               BP        0000
               SI        0000
               DI        0000
               DS        0700

0700:0103                        0700:0103
07100: B8 184 ╕                  MOV  AX, 01000h
07101: 00 000 NULL               INC  AX
07102: 10 016 ►                  DEC  AX
07103: 40 064 @                  MOV  AX, 00026h
07104: 48 072 H                  NEG  AX
07105: B8 184 ╕                  MOV  BX, 00030h
07106: 26 038 &                  SUB  BX, 040h
07107: 00 000 NULL               ADD  AX, BX
07108: F7 247 ≈                  MOV  CX, 00001h
07109: D8 216 ╪                  SUB  CX, 01h
0710A: BB 187 ╗                  MOV  AX, 0FFFFh
0710B: 30 048 0                  INC  AX
0710C: 00 000 NULL               MOV  CX, 00000h
0710D: 83 131 â                  SUB  CX, 01h
0710E: EB 235 δ                  MOV  AX, 07FFFh
0710F: 40 064 @                  ADD  AX, 00002h
07110: 03 003 ♥                  MOV  AL, 0FFh
07111: C3 195 ├                  ADD  AL, 01h
07112: B9 185 ╣                  MOV  AL, [0007Fh]
07113: 01 001 ☺                  ADD  AL, 01h
07114: 00 000 NULL               MOV  AL, 080h
07115: 83 131 â                  ---
```

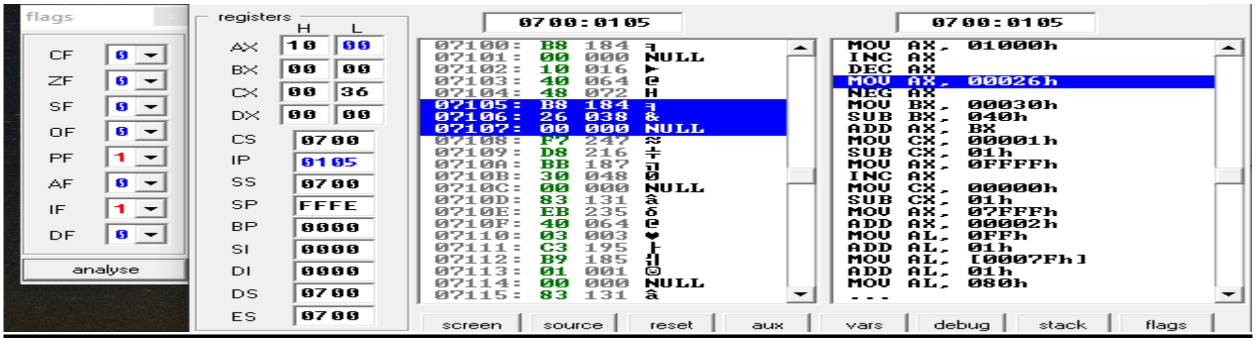
STEP 02:



```
flags          registers  H   L
CF 0           AX        10  01
ZF 0           BX        00  00
SF 0           CX        00  36
OF 0           DX        00  00
PF 0           CS        0700
AF 0           IP        0104
IF 1           SS        0700
DF 0           SP        FFFE
               BP        0000
               SI        0000
               DI        0000
               DS        0700

0700:0104                        0700:0104
07100: B8 184 ╕                  MOV  AX, 01000h
07101: 00 000 NULL               INC  AX
07102: 10 016 ►                  DEC  AX
07103: 40 064 @                  MOV  AX, 00026h
07104: 48 072 H                  NEG  AX
07105: B8 184 ╕                  MOV  BX, 00030h
07106: 26 038 &                  SUB  BX, 040h
07107: 00 000 NULL               ADD  AX, BX
07108: F7 247 ≈                  MOV  CX, 00001h
07109: D8 216 ╪                  SUB  CX, 01h
0710A: BB 187 ╗                  MOV  AX, 0FFFFh
0710B: 30 048 0                  INC  AX
0710C: 00 000 NULL               MOV  CX, 00000h
0710D: 83 131 â                  SUB  CX, 01h
0710E: EB 235 δ                  MOV  AX, 07FFFh
0710F: 40 064 @                  ADD  AX, 00002h
07110: 03 003 ♥                  MOV  AL, 0FFh
07111: C3 195 ├                  ADD  AL, 01h
07112: B9 185 ╣                  MOV  AL, [0007Fh]
07113: 01 001 ☺                  ADD  AL, 01h
07114: 00 000 NULL               MOV  AL, 080h
07115: 83 131 â                  ---
```

STEP 03:

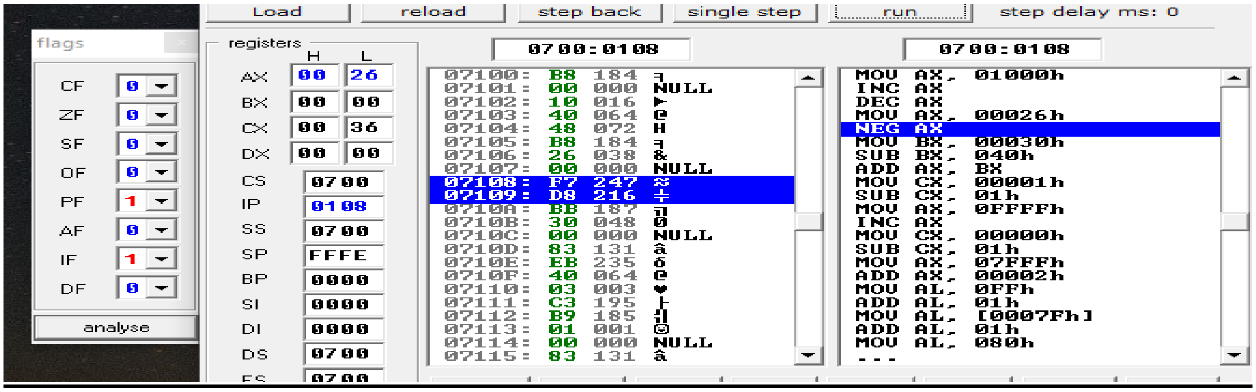


```
flags          registers  H   L
CF 0           AX        10  00
ZF 0           BX        00  00
SF 0           CX        00  36
OF 0           DX        00  00
PF 1           CS        0700
AF 0           IP        0105
IF 1           SS        0700
DF 0           SP        FFFE
               BP        0000
               SI        0000
               DI        0000
               DS        0700
               ES        0700

0700:0105                        0700:0105
07100: B8 184 ╕                  MOV  AX, 01000h
07101: 00 000 NULL               INC  AX
07102: 10 016 ►                  DEC  AX
07103: 40 064 @                  MOV  AX, 00026h
07104: 48 072 H                  NEG  AX
07105: B8 184 ╕                  MOV  BX, 00030h
07106: 26 038 &                  SUB  BX, 040h
07107: 00 000 NULL               ADD  AX, BX
07108: F7 247 ≈                  MOV  CX, 00001h
07109: D8 216 ╪                  SUB  CX, 01h
0710A: BB 187 ╗                  MOV  AX, 0FFFFh
0710B: 30 048 0                  INC  AX
0710C: 00 000 NULL               MOV  CX, 00000h
0710D: 83 131 â                  SUB  CX, 01h
0710E: EB 235 δ                  MOV  AX, 07FFFh
0710F: 40 064 @                  ADD  AX, 00002h
07110: 03 003 ♥                  MOV  AL, 0FFh
07111: C3 195 ├                  ADD  AL, 01h
07112: B9 185 ╣                  MOV  AL, [0007Fh]
07113: 01 001 ☺                  ADD  AL, 01h
07114: 00 000 NULL               MOV  AL, 080h
07115: 83 131 â                  ---
```

STEP 04:

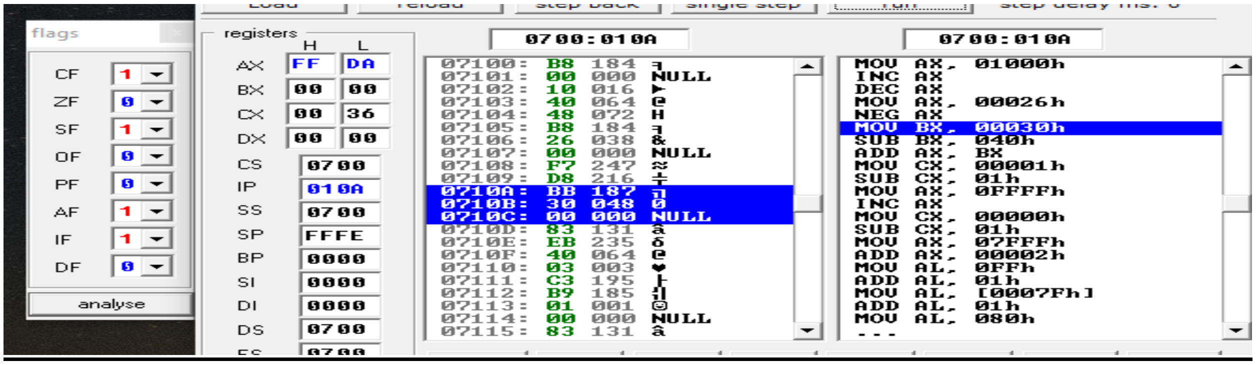
SAAD UR REHMAN
19k-0218
SEC: C



```
flags            registers
                      H   L
CF  0            AX   00  26
ZF  0            BX   00  00
SF  0            CX   00  36
OF  0            DX   00  00
PF  1            CS   0700
AF  0            IP   0108
IF  1            SS   0700
DF  0            SP   FFFE
                 BP   0000
                 SI   0000
                 DI   0000
                 DS   0700
                 ES   0700

0700:0108                    0700:0108
07100: B8 184 ╕              MOV  AX, 01000h
07101: 00 000 NULL           INC  AX
07102: 10 016 ►              DEC  AX
07103: 40 064 @              MOV  AX, 00026h
07104: 48 072 H              NEG  AX
07105: B8 184 ╕              MOV  BX, 00030h
07106: 26 038 &              SUB  BX, 040h
07107: 00 000 NULL           ADD  AX, BX
07108: F7 247 ≈              MOV  CX, 00001h
07109: D8 216 ╪              SUB  CX, 01h
0710A: BB 187 ╗              MOV  AX, 0FFFFh
0710B: 30 048 0              INC  AX
0710C: 00 000 NULL           MOV  CX, 00000h
0710D: 83 131 â              SUB  CX, 01h
0710E: EB 235 δ              MOV  AX, 07FFFh
0710F: 40 064 @              ADD  AX, 00002h
07110: 03 003 ♥              MOV  AL, 0FFh
07111: C3 195 ├              ADD  AL, 01h
07112: B9 185 ╣              MOV  AL, [0007Fh]
07113: 01 001 ☺              ADD  AL, 01h
07114: 00 000 NULL           MOV  AL, 080h
07115: 83 131 â              ...
```

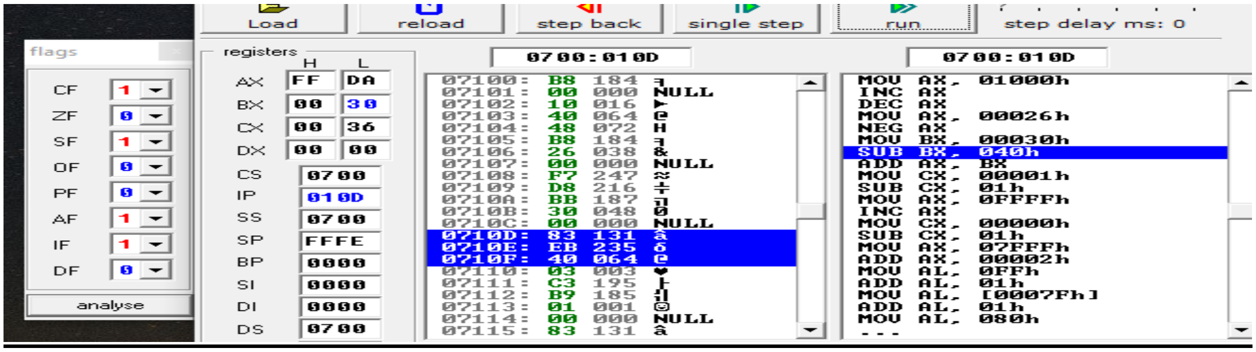
STEP 05:



```
flags            registers
                      H   L
CF  1            AX   FF  DA
ZF  0            BX   00  00
SF  1            CX   00  36
OF  0            DX   00  00
PF  0            CS   0700
AF  1            IP   010A
IF  1            SS   0700
DF  0            SP   FFFE
                 BP   0000
                 SI   0000
                 DI   0000
                 DS   0700
                 ES   0700

0700:010A                    0700:010A
07100: B8 184 ╕              MOV  AX, 01000h
07101: 00 000 NULL           INC  AX
07102: 10 016 ►              DEC  AX
07103: 40 064 @              MOV  AX, 00026h
07104: 48 072 H              NEG  AX
07105: B8 184 ╕              MOV  BX, 00030h
07106: 26 038 &              SUB  BX, 040h
07107: 00 000 NULL           ADD  AX, BX
07108: F7 247 ≈              MOV  CX, 00001h
07109: D8 216 ╪              SUB  CX, 01h
0710A: BB 187 ╗              MOV  AX, 0FFFFh
0710B: 30 048 0              INC  AX
0710C: 00 000 NULL           MOV  CX, 00000h
0710D: 83 131 â              SUB  CX, 01h
0710E: EB 235 δ              MOV  AX, 07FFFh
0710F: 40 064 @              ADD  AX, 00002h
07110: 03 003 ♥              MOV  AL, 0FFh
07111: C3 195 ├              ADD  AL, 01h
07112: B9 185 ╣              MOV  AL, [0007Fh]
07113: 01 001 ☺              ADD  AL, 01h
07114: 00 000 NULL           MOV  AL, 080h
07115: 83 131 â              ...
```

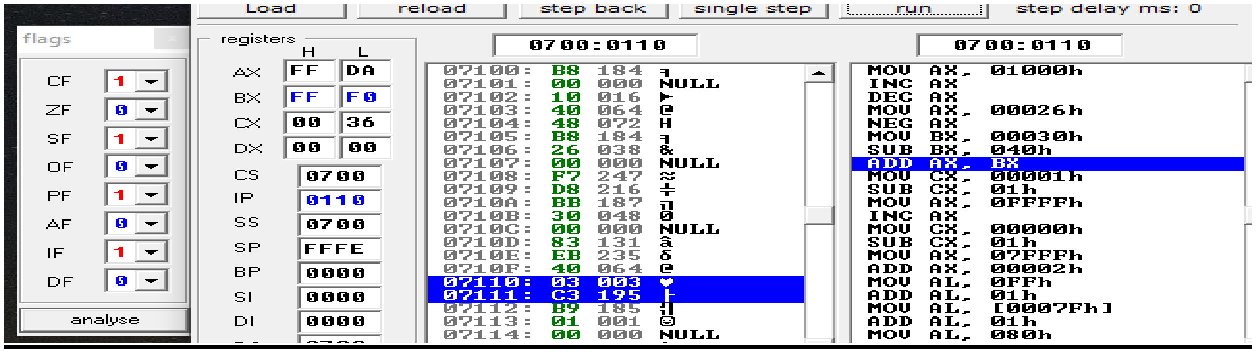
STEP 06:



```
flags            registers
                      H   L
CF  1            AX   FF  DA
ZF  0            BX   00  30
SF  1            CX   00  36
OF  0            DX   00  00
PF  0            CS   0700
AF  1            IP   010D
IF  1            SS   0700
DF  0            SP   FFFE
                 BP   0000
                 SI   0000
                 DI   0000
                 DS   0700

0700:010D                    0700:010D
07100: B8 184 ╕              MOV  AX, 01000h
07101: 00 000 NULL           INC  AX
07102: 10 016 ►              DEC  AX
07103: 40 064 @              MOV  AX, 00026h
07104: 48 072 H              NEG  AX
07105: B8 184 ╕              MOV  BX, 00030h
07106: 26 038 &              SUB  BX, 040h
07107: 00 000 NULL           ADD  AX, BX
07108: F7 247 ≈              MOV  CX, 00001h
07109: D8 216 ╪              SUB  CX, 01h
0710A: BB 187 ╗              MOV  AX, 0FFFFh
0710B: 30 048 0              INC  AX
0710C: 00 000 NULL           MOV  CX, 00000h
0710D: 83 131 â              SUB  CX, 01h
0710E: EB 235 δ              MOV  AX, 07FFFh
0710F: 40 064 @              ADD  AX, 00002h
07110: 03 003 ♥              MOV  AL, 0FFh
07111: C3 195 ├              ADD  AL, 01h
07112: B9 185 ╣              MOV  AL, [0007Fh]
07113: 01 001 ☺              ADD  AL, 01h
07114: 00 000 NULL           MOV  AL, 080h
07115: 83 131 â              ...
```

STEP 07:

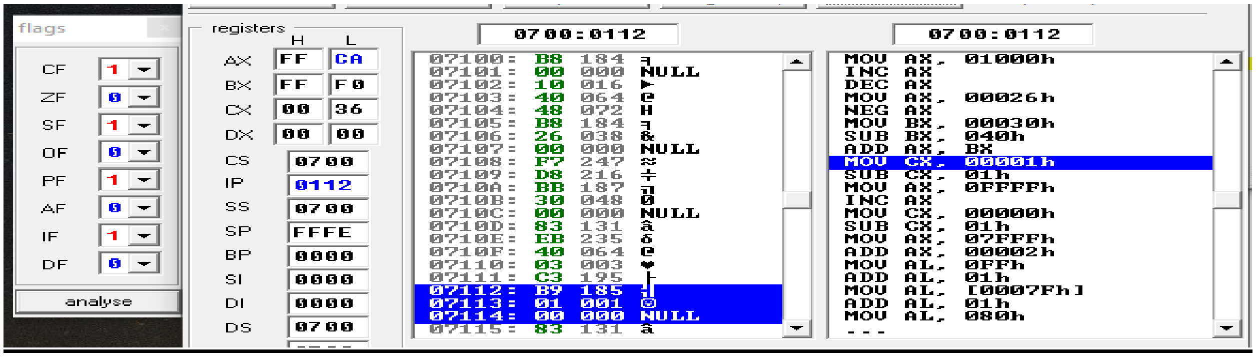


```
flags            registers
                      H   L
CF  1            AX   FF  DA
ZF  0            BX   FF  F0
SF  1            CX   00  36
OF  0            DX   00  00
PF  1            CS   0700
AF  0            IP   0110
IF  1            SS   0700
DF  0            SP   FFFE
                 BP   0000
                 SI   0000
                 DI   0000

0700:0110                    0700:0110
07100: B8 184 ╕              MOV  AX, 01000h
07101: 00 000 NULL           INC  AX
07102: 10 016 ►              DEC  AX
07103: 40 064 @              MOV  AX, 00026h
07104: 48 072 H              NEG  AX
07105: B8 184 ╕              MOV  BX, 00030h
07106: 26 038 &              SUB  BX, 040h
07107: 00 000 NULL           ADD  AX, BX
07108: F7 247 ≈              MOV  CX, 00001h
07109: D8 216 ╪              SUB  CX, 01h
0710A: BB 187 ╗              MOV  AX, 0FFFFh
0710B: 30 048 0              INC  AX
0710C: 00 000 NULL           MOV  CX, 00000h
0710D: 83 131 â              SUB  CX, 01h
0710E: EB 235 δ              MOV  AX, 07FFFh
0710F: 40 064 @              ADD  AX, 00002h
07110: 03 003 ♥              MOV  AL, 0FFh
07111: C3 195 ├              ADD  AL, 01h
07112: B9 185 ╣              MOV  AL, [0007Fh]
07113: 01 001 ☺              ADD  AL, 01h
07114: 00 000 NULL           MOV  AL, 080h
```

SAAD UR REHMAN
19k-0218
SEC: C

STEP 08:



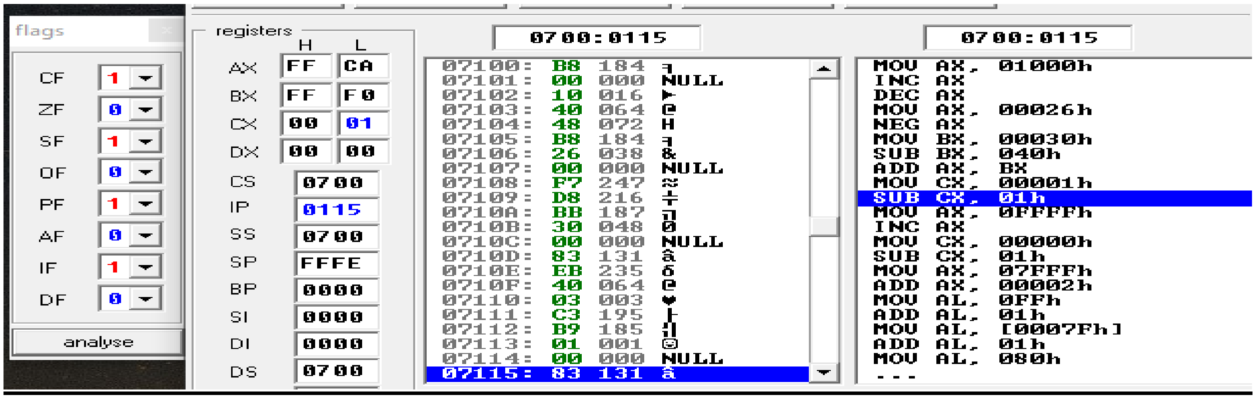
```
flags
CF  1
ZF  0
SF  1
OF  0
PF  1
AF  0
IF  1
DF  0
analyse
```

```
registers
      H    L
AX    FF   CA
BX    FF   F0
CX    00   36
DX    00   00
CS    0700
IP    0112
SS    0700
SP    FFFE
BP    0000
SI    0000
DI    0000
DS    0700
```

```
0700:0112
07100:  B8  184  ╕
07101:  00  000  NULL
07102:  10  016  ►
07103:  40  064  @
07104:  48  072  H
07105:  B8  184  ╕
07106:  26  038  &
07107:  00  000  NULL
07108:  F7  247  ≈
07109:  D8  216  ╪
0710A:  BB  187  ╗
0710B:  30  048  0
0710C:  00  000  NULL
0710D:  83  131  â
0710E:  EB  235  δ
0710F:  40  064  @
07110:  03  003  ♥
07111:  C3  195  ├
07112:  B9  185  ╣
07113:  01  001  ☺
07114:  00  000  NULL
07115:  83  131  â
```

```
0700:0112
MOV  AX,  01000h
INC  AX
DEC  AX
MOV  AX,  00026h
NEG  AX
MOV  BX,  00030h
SUB  BX,  040h
ADD  AX,  BX
MOV  CX,  00001h
SUB  CX,  01h
MOV  AX,  0FFFFh
INC  AX
MOV  CX,  00000h
SUB  CX,  01h
MOV  AX,  07FFFh
ADD  AX,  00002h
MOV  AL,  0FFh
ADD  AL,  01h
MOV  AL,  [0007Fh]
ADD  AL,  01h
MOV  AL,  080h
- - -
```

STEP 09:



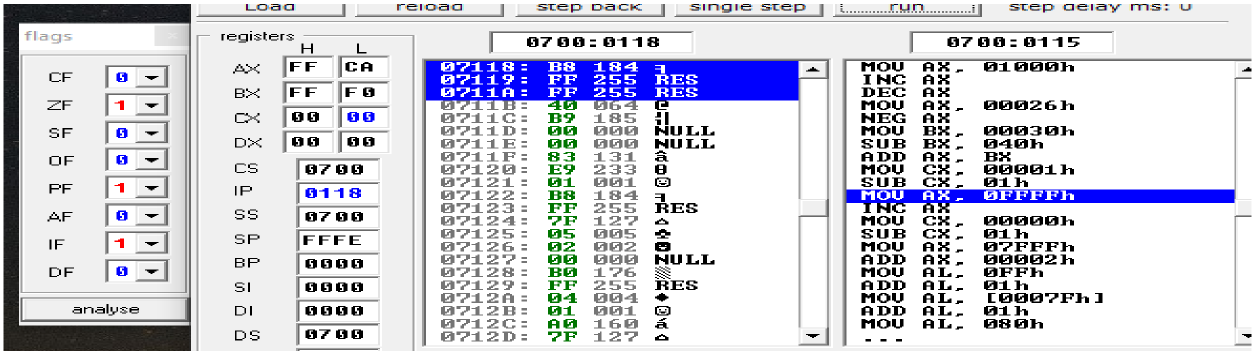
```
flags
CF  1
ZF  0
SF  1
OF  0
PF  1
AF  0
IF  1
DF  0
analyse
```

```
registers
      H    L
AX    FF   CA
BX    FF   F0
CX    00   01
DX    00   00
CS    0700
IP    0115
SS    0700
SP    FFFE
BP    0000
SI    0000
DI    0000
DS    0700
```

```
0700:0115
07100:  B8  184  ╕
07101:  00  000  NULL
07102:  10  016  ►
07103:  40  064  @
07104:  48  072  H
07105:  B8  184  ╕
07106:  26  038  &
07107:  00  000  NULL
07108:  F7  247  ≈
07109:  D8  216  ╪
0710A:  BB  187  ╗
0710B:  30  048  0
0710C:  00  000  NULL
0710D:  83  131  â
0710E:  EB  235  δ
0710F:  40  064  @
07110:  03  003  ♥
07111:  C3  195  ├
07112:  B9  185  ╣
07113:  01  001  ☺
07114:  00  000  NULL
07115:  83  131  â
```

```
0700:0115
MOV  AX,  01000h
INC  AX
DEC  AX
MOV  AX,  00026h
NEG  AX
MOV  BX,  00030h
SUB  BX,  040h
ADD  AX,  BX
MOV  CX,  00001h
SUB  CX,  01h
MOV  AX,  0FFFFh
INC  AX
MOV  CX,  00000h
SUB  CX,  01h
MOV  AX,  07FFFh
ADD  AX,  00002h
MOV  AL,  0FFh
ADD  AL,  01h
MOV  AL,  [0007Fh]
ADD  AL,  01h
MOV  AL,  080h
- - -
```

STEP 10:



```
flags
CF  0
ZF  1
SF  0
OF  0
PF  1
AF  0
IF  1
DF  0
analyse
```

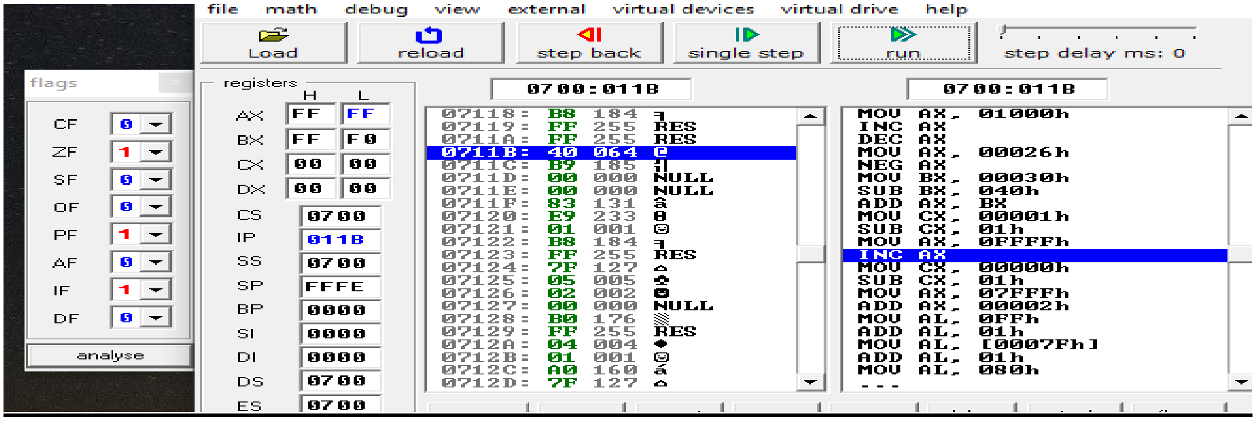
```
registers
      H    L
AX    FF   CA
BX    FF   F0
CX    00   00
DX    00   00
CS    0700
IP    0118
SS    0700
SP    FFFE
BP    0000
SI    0000
DI    0000
DS    0700
```

```
0700:0118
07118:  B8  184  ╕
07119:  FF  255  RES
0711A:  FF  255  RES
0711B:  40  064  @
0711C:  B9  185  ╣
0711D:  00  000  NULL
0711E:  00  000  NULL
0711F:  83  131  â
07120:  E9  233  Θ
07121:  01  001  ☺
07122:  B8  184  ╕
07123:  FF  255  RES
07124:  7F  127  ⌂
07125:  05  005  ♣
07126:  02  002  ☻
07127:  00  000  NULL
07128:  B0  176  ░
07129:  FF  255  RES
0712A:  04  004  ♦
0712B:  01  001  ☺
0712C:  A0  160  á
0712D:  7F  127  ⌂
```

```
0700:0115
MOV  AX,  01000h
INC  AX
DEC  AX
MOV  AX,  00026h
NEG  AX
MOV  BX,  00030h
SUB  BX,  040h
ADD  AX,  BX
MOV  CX,  00001h
SUB  CX,  01h
MOV  AX,  0FFFFh
INC  AX
MOV  CX,  00000h
SUB  CX,  01h
MOV  AX,  07FFFh
ADD  AX,  00002h
MOV  AL,  0FFh
ADD  AL,  01h
MOV  AL,  [0007Fh]
ADD  AL,  01h
MOV  AL,  080h
- - -
```

STEP 11:

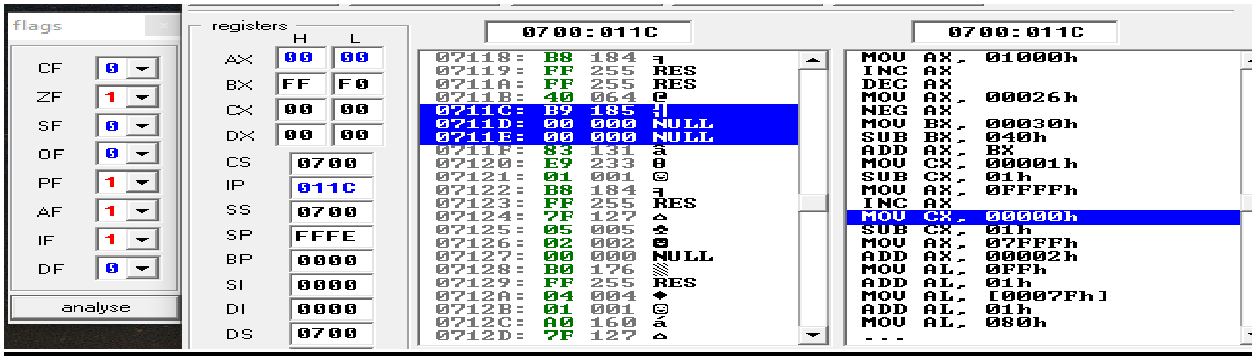
SAAD UR REHMAN
19k-0218
SEC: C



```
flags          registers   H   L
CF  0          AX          FF  FF
ZF  1          BX          FF  F0
SF  0          CX          00  00
OF  0          DX          00  00
PF  1          CS  0700
AF  0          IP  011B
IF  1          SS  0700
DF  0          SP  FFFE
               BP  0000
               SI  0000
               DI  0000
               DS  0700
               ES  0700
```

```
0700:011B                        0700:011B
07118:  B8  184  ╕               MOV  AX,  01000h
07119:  FF  255  RES             INC  AX
0711A:  FF  255  RES             DEC  AX
0711B:  40  064  @               MOV  AX,  00026h
0711C:  B9  185  ╣               NEG  AX
0711D:  00  000  NULL            MOV  BX,  00030h
0711E:  00  000  NULL            SUB  BX,  040h
0711F:  83  131  â               ADD  AX,  BX
07120:  E9  233  Θ               MOV  CX,  00001h
07121:  01  001  ☺               SUB  CX,  01h
07122:  B8  184  ╕               MOV  AX,  0FFFFh
07123:  FF  255  RES             INC  AX
07124:  7F  127  ⌂               MOV  CX,  00000h
07125:  05  005  ♣               SUB  CX,  01h
07126:  02  002  ☻               MOV  AX,  07FFFh
07127:  00  000  NULL            ADD  AX,  00002h
07128:  B0  176  ░               MOV  AL,  0FFh
07129:  FF  255  RES             ADD  AL,  01h
0712A:  04  004  ♦               MOV  AL,  [0007Fh]
0712B:  01  001  ☺               ADD  AL,  01h
0712C:  A0  160  á               MOV  AL,  080h
0712D:  7F  127  ⌂               ---
```

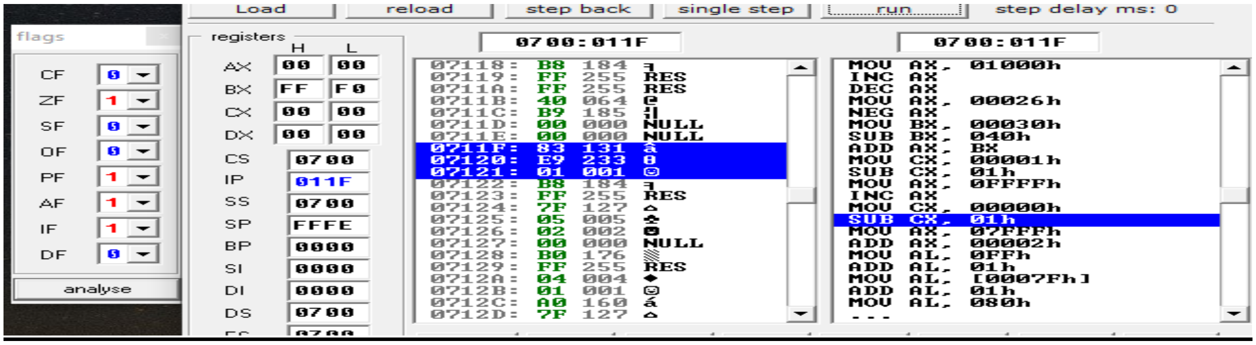
STEP 12:



```
flags          registers   H   L
CF  0          AX          00  00
ZF  1          BX          FF  F0
SF  0          CX          00  00
OF  0          DX          00  00
PF  1          CS  0700
AF  1          IP  011C
IF  1          SS  0700
DF  0          SP  FFFE
               BP  0000
               SI  0000
               DI  0000
               DS  0700
```

```
0700:011C                        0700:011C
07118:  B8  184  ╕               MOV  AX,  01000h
07119:  FF  255  RES             INC  AX
0711A:  FF  255  RES             DEC  AX
0711B:  40  064  @               MOV  AX,  00026h
0711C:  B9  185  ╣               NEG  AX
0711D:  00  000  NULL            MOV  BX,  00030h
0711E:  00  000  NULL            SUB  BX,  040h
0711F:  83  131  â               ADD  AX,  BX
07120:  E9  233  Θ               MOV  CX,  00001h
07121:  01  001  ☺               SUB  CX,  01h
07122:  B8  184  ╕               MOV  AX,  0FFFFh
07123:  FF  255  RES             INC  AX
07124:  7F  127  ⌂               MOV  CX,  00000h
07125:  05  005  ♣               SUB  CX,  01h
07126:  02  002  ☻               MOV  AX,  07FFFh
07127:  00  000  NULL            ADD  AX,  00002h
07128:  B0  176  ░               MOV  AL,  0FFh
07129:  FF  255  RES             ADD  AL,  01h
0712A:  04  004  ♦               MOV  AL,  [0007Fh]
0712B:  01  001  ☺               ADD  AL,  01h
0712C:  A0  160  á               MOV  AL,  080h
0712D:  7F  127  ⌂               ---
```

STEP 13:

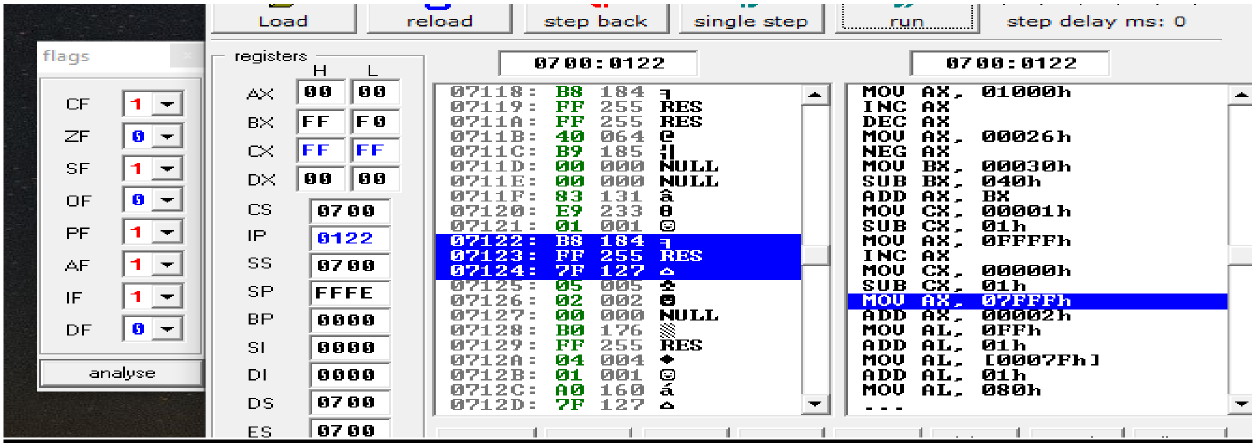


```
flags          registers   H   L
CF  0          AX          00  00
ZF  1          BX          FF  F0
SF  0          CX          00  00
OF  0          DX          00  00
PF  1          CS  0700
AF  1          IP  011F
IF  1          SS  0700
DF  0          SP  FFFE
               BP  0000
               SI  0000
               DI  0000
               DS  0700
               ES  0700
```

```
0700:011F                        0700:011F
07118:  B8  184  ╕               MOV  AX,  01000h
07119:  FF  255  RES             INC  AX
0711A:  FF  255  RES             DEC  AX
0711B:  40  064  @               MOV  AX,  00026h
0711C:  B9  185  ╣               NEG  AX
0711D:  00  000  NULL            MOV  BX,  00030h
0711E:  00  000  NULL            SUB  BX,  040h
0711F:  83  131  â               ADD  AX,  BX
07120:  E9  233  Θ               MOV  CX,  00001h
07121:  01  001  ☺               SUB  CX,  01h
07122:  B8  184  ╕               MOV  AX,  0FFFFh
07123:  FF  255  RES             INC  AX
07124:  7F  127  ⌂               MOV  CX,  00000h
07125:  05  005  ♣               SUB  CX,  01h
07126:  02  002  ☻               MOV  AX,  07FFFh
07127:  00  000  NULL            ADD  AX,  00002h
07128:  B0  176  ░               MOV  AL,  0FFh
07129:  FF  255  RES             ADD  AL,  01h
0712A:  04  004  ♦               MOV  AL,  [0007Fh]
0712B:  01  001  ☺               ADD  AL,  01h
0712C:  A0  160  á               MOV  AL,  080h
0712D:  7F  127  ⌂               ---
```

STEP 14:

SAAD UR REHMAN
19k-0218
SEC: C



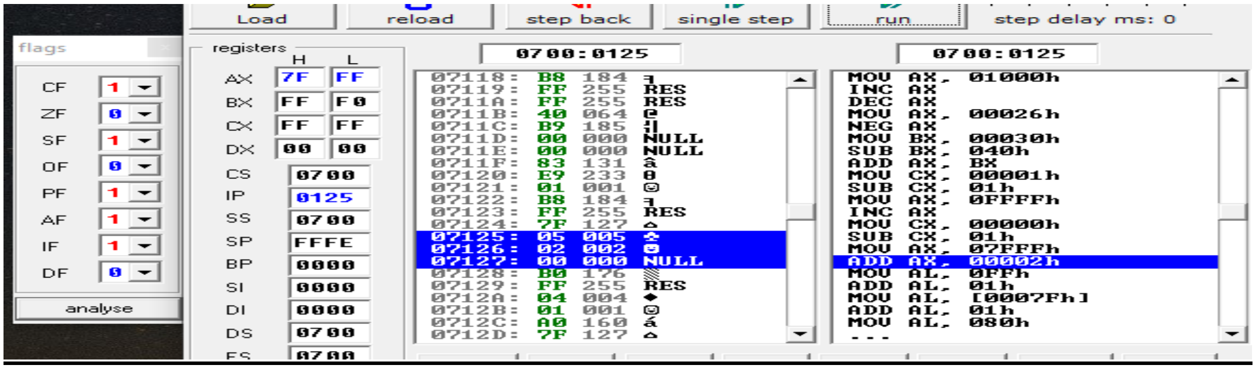
step delay ms: 0

```
flags              registers   H   L
CF  1              AX          00  00
ZF  0              BX          FF  F0
SF  1              CX          FF  FF
OF  0              DX          00  00
PF  1              CS  0700
AF  1              IP  0122
IF  1              SS  0700
DF  0              SP  FFFE
                   BP  0000
                   SI  0000
                   DI  0000
                   DS  0700
                   ES  0700
```

```
0700:0122                     0700:0122

07118: B8 184 ╕               MOV  AX, 01000h
07119: FF 255 RES             INC  AX
0711A: FF 255 RES             DEC  AX
0711B: 40 064 @               MOV  AX, 00026h
0711C: B9 185 ╣               NEG  AX
0711D: 00 000 NULL            MOV  BX, 00030h
0711E: 00 000 NULL            SUB  BX, 040h
0711F: 83 131 â               ADD  AX, BX
07120: E9 233 θ               MOV  CX, 00001h
07121: 01 001 ☺               SUB  CX, 01h
07122: B8 184 ╕               MOV  AX, 0FFFFh
07123: FF 255 RES             INC  AX
07124: 7F 127 ⌂               MOV  CX, 00000h
07125: 05 005 ♣               SUB  CX, 01h
07126: 02 002 ☻               MOV  AX, 07FFFh
07127: 00 000 NULL            ADD  AX, 00002h
07128: B0 176 ░               MOV  AL, 0FFh
07129: FF 255 RES             ADD  AL, 01h
0712A: 04 004 ♦               MOV  AL, [0007Fh]
0712B: 01 001 ☺               ADD  AL, 01h
0712C: A0 160 á               MOV  AL, 080h
0712D: 7F 127 ⌂               ...
```

STEP 15:



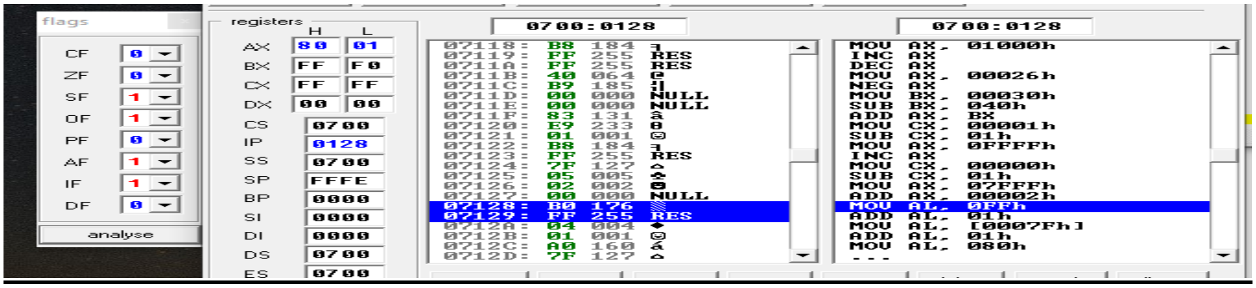
step delay ms: 0

```
flags              registers   H   L
CF  1              AX          7F  FF
ZF  0              BX          FF  F0
SF  1              CX          FF  FF
OF  0              DX          00  00
PF  1              CS  0700
AF  1              IP  0125
IF  1              SS  0700
DF  0              SP  FFFE
                   BP  0000
                   SI  0000
                   DI  0000
                   DS  0700
                   ES  0700
```

```
0700:0125                     0700:0125

07118: B8 184 ╕               MOV  AX, 01000h
07119: FF 255 RES             INC  AX
0711A: FF 255 RES             DEC  AX
0711B: 40 064 @               MOV  AX, 00026h
0711C: B9 185 ╣               NEG  AX
0711D: 00 000 NULL            MOV  BX, 00030h
0711E: 00 000 NULL            SUB  BX, 040h
0711F: 83 131 â               ADD  AX, BX
07120: E9 233 θ               MOV  CX, 00001h
07121: 01 001 ☺               SUB  CX, 01h
07122: B8 184 ╕               MOV  AX, 0FFFFh
07123: FF 255 RES             INC  AX
07124: 7F 127 ⌂               MOV  CX, 00000h
07125: 05 005 ♣               SUB  CX, 01h
07126: 02 002 ☻               MOV  AX, 07FFFh
07127: 00 000 NULL            ADD  AX, 00002h
07128: B0 176 ░               MOV  AL, 0FFh
07129: FF 255 RES             ADD  AL, 01h
0712A: 04 004 ♦               MOV  AL, [0007Fh]
0712B: 01 001 ☺               ADD  AL, 01h
0712C: A0 160 á               MOV  AL, 080h
0712D: 7F 127 ⌂               ...
```

STEP 16:



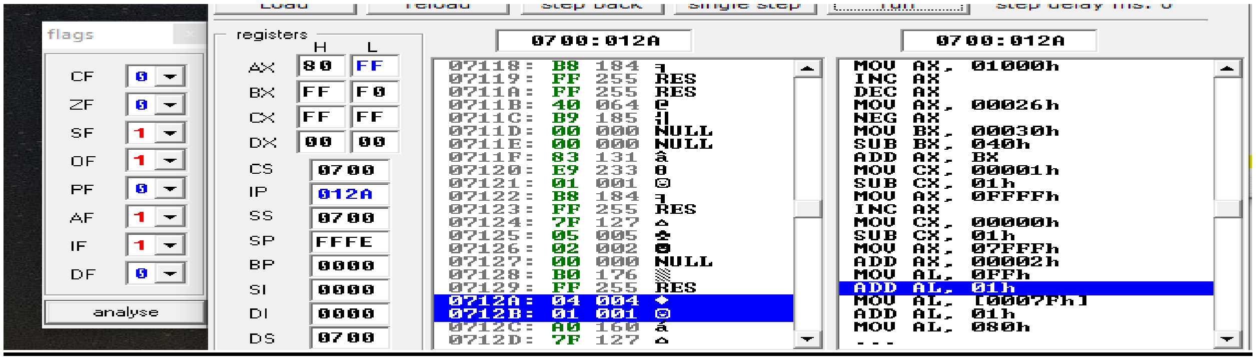
```
flags              registers   H   L
CF  0              AX          80  01
ZF  0              BX          FF  F0
SF  1              CX          FF  FF
OF  1              DX          00  00
PF  0              CS  0700
AF  1              IP  0128
IF  1              SS  0700
DF  0              SP  FFFE
                   BP  0000
                   SI  0000
                   DI  0000
                   DS  0700
                   ES  0700
```

```
0700:0128                     0700:0128

07118: B8 184 ╕               MOV  AX, 01000h
07119: FF 255 RES             INC  AX
0711A: FF 255 RES             DEC  AX
0711B: 40 064 @               MOV  AX, 00026h
0711C: B9 185 ╣               NEG  AX
0711D: 00 000 NULL            MOV  BX, 00030h
0711E: 00 000 NULL            SUB  BX, 040h
0711F: 83 131 â               ADD  AX, BX
07120: E9 233 θ               MOV  CX, 00001h
07121: 01 001 ☺               SUB  CX, 01h
07122: B8 184 ╕               MOV  AX, 0FFFFh
07123: FF 255 RES             INC  AX
07124: 7F 127 ⌂               MOV  CX, 00000h
07125: 05 005 ♣               SUB  CX, 01h
07126: 02 002 ☻               MOV  AX, 07FFFh
07127: 00 000 NULL            ADD  AX, 00002h
07128: B0 176 ░               MOV  AL, 0FFh
07129: FF 255 RES             ADD  AL, 01h
0712A: 04 004 ♦               MOV  AL, [0007Fh]
0712B: 01 001 ☺               ADD  AL, 01h
0712C: A0 160 á               MOV  AL, 080h
0712D: 7F 127 ⌂               ...
```

STEP 17:

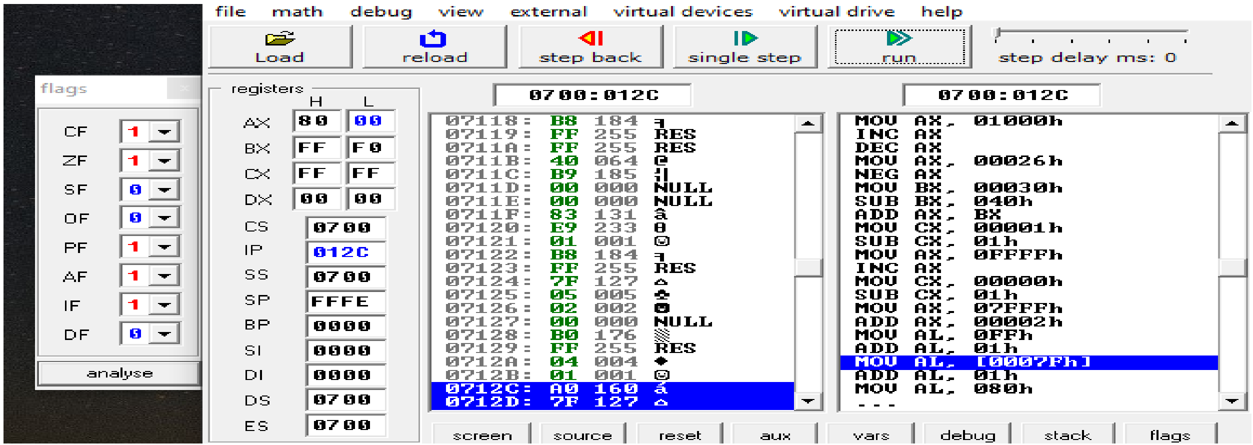
SAAD UR REHMAN
19k-0218
SEC: C



```
flags                 registers   H    L
CF  0                 AX         80   FF
ZF  0                 BX         FF   F0
SF  1                 CX         FF   FF
OF  1                 DX         00   00
PF  0                 CS         0700
AF  1                 IP         012A
IF  1                 SS         0700
DF  0                 SP         FFFE
analyse               BP         0000
                      SI         0000
                      DI         0000
                      DS         0700

0700:012A                        0700:012A
07118:  B8  184  ╕               MOV  AX,  01000h
07119:  FF  255  RES             INC  AX
0711A:  FF  255  RES             DEC  AX
0711B:  40  064  @               MOV  AX,  00026h
0711C:  B9  185  ╣               NEG  AX
0711D:  00  000  NULL            MOV  BX,  00030h
0711E:  00  000  NULL            SUB  BX,  040h
0711F:  83  131  â               ADD  AX,  BX
07120:  E9  233  Θ               MOV  CX,  00001h
07121:  01  001  ☺               SUB  CX,  01h
07122:  B8  184  ╕               MOV  AX,  0FFFFh
07123:  FF  255  RES             INC  AX
07124:  7F  127  ⌂               MOV  CX,  00000h
07125:  05  005  ♣               SUB  CX,  01h
07126:  02  002  ☻               MOV  AX,  07FFFh
07127:  00  000  NULL            ADD  AX,  00002h
07128:  B0  176  ░               MOV  AL,  0FFh
07129:  FF  255  RES             ADD  AL,  01h
0712A:  04  004  ♦               MOV  AL,  [0007Fh]
0712B:  01  001  ☺               ADD  AL,  01h
0712C:  A0  160  á               MOV  AL,  080h
0712D:  7F  127  ⌂               ---
```

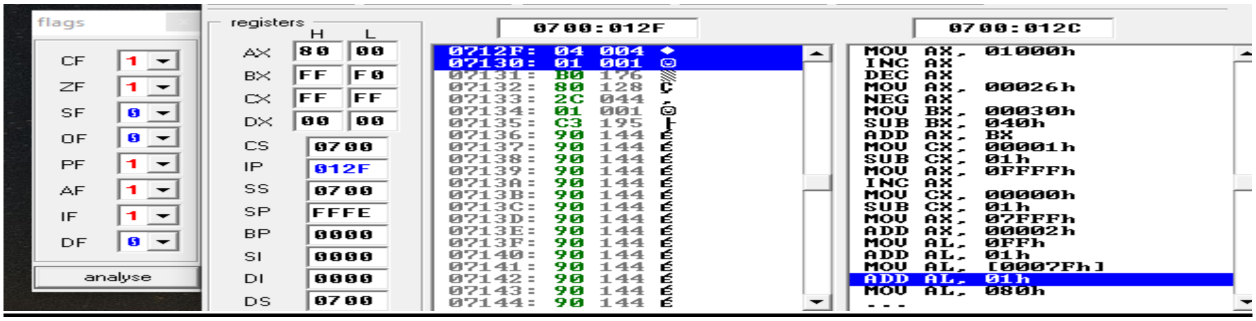
STEP 18:



```
flags                 registers   H    L
CF  1                 AX         80   00
ZF  1                 BX         FF   F0
SF  0                 CX         FF   FF
OF  0                 DX         00   00
PF  1                 CS         0700
AF  1                 IP         012C
IF  1                 SS         0700
DF  0                 SP         FFFE
analyse               BP         0000
                      SI         0000
                      DI         0000
                      DS         0700
                      ES         0700

0700:012C                        0700:012C
07118:  B8  184  ╕               MOV  AX,  01000h
07119:  FF  255  RES             INC  AX
0711A:  FF  255  RES             DEC  AX
0711B:  40  064  @               MOV  AX,  00026h
0711C:  B9  185  ╣               NEG  AX
0711D:  00  000  NULL            MOV  BX,  00030h
0711E:  00  000  NULL            SUB  BX,  040h
0711F:  83  131  â               ADD  AX,  BX
07120:  E9  233  Θ               MOV  CX,  00001h
07121:  01  001  ☺               SUB  CX,  01h
07122:  B8  184  ╕               MOV  AX,  0FFFFh
07123:  FF  255  RES             INC  AX
07124:  7F  127  ⌂               MOV  CX,  00000h
07125:  05  005  ♣               SUB  CX,  01h
07126:  02  002  ☻               MOV  AX,  07FFFh
07127:  00  000  NULL            ADD  AX,  00002h
07128:  B0  176  ░               MOV  AL,  0FFh
07129:  FF  255  RES             ADD  AL,  01h
0712A:  04  004  ♦               MOV  AL,  [0007Fh]
0712B:  01  001  ☺               ADD  AL,  01h
0712C:  A0  160  á               MOV  AL,  080h
0712D:  7F  127  ⌂               ---
```

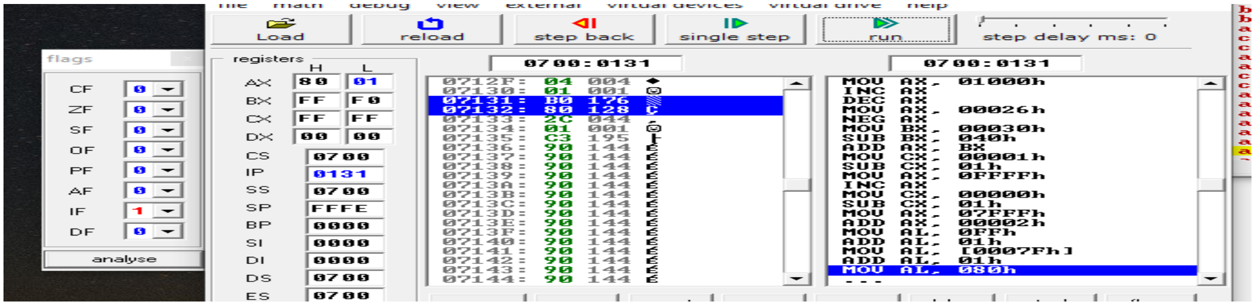
STEP 19:



```
flags                 registers   H    L
CF  1                 AX         80   00
ZF  1                 BX         FF   F0
SF  0                 CX         FF   FF
OF  0                 DX         00   00
PF  1                 CS         0700
AF  1                 IP         012F
IF  1                 SS         0700
DF  0                 SP         FFFE
analyse               BP         0000
                      SI         0000
                      DI         0000
                      DS         0700

0700:012F                        0700:012C
0712F:  04  004  ♦               MOV  AX,  01000h
07130:  01  001  ☺               INC  AX
07131:  B0  176  ░               DEC  AX
07132:  80  128  Ç               MOV  AX,  00026h
07133:  2C  044  ,               NEG  AX
07134:  01  001  ☺               MOV  BX,  00030h
07135:  C3  195  ├               SUB  BX,  040h
07136:  90  144  É               ADD  AX,  BX
07137:  90  144  É               MOV  CX,  00001h
07138:  90  144  É               SUB  CX,  01h
07139:  90  144  É               MOV  AX,  0FFFFh
0713A:  90  144  É               INC  AX
0713B:  90  144  É               MOV  CX,  00000h
0713C:  90  144  É               SUB  CX,  01h
0713D:  90  144  É               MOV  AX,  07FFFh
0713E:  90  144  É               ADD  AX,  00002h
0713F:  90  144  É               MOV  AL,  0FFh
07140:  90  144  É               ADD  AL,  01h
07141:  90  144  É               MOV  AL,  [0007Fh]
07142:  90  144  É               ADD  AL,  01h
07143:  90  144  É               MOV  AL,  080h
07144:  90  144  É               ---
```

STEP 20:

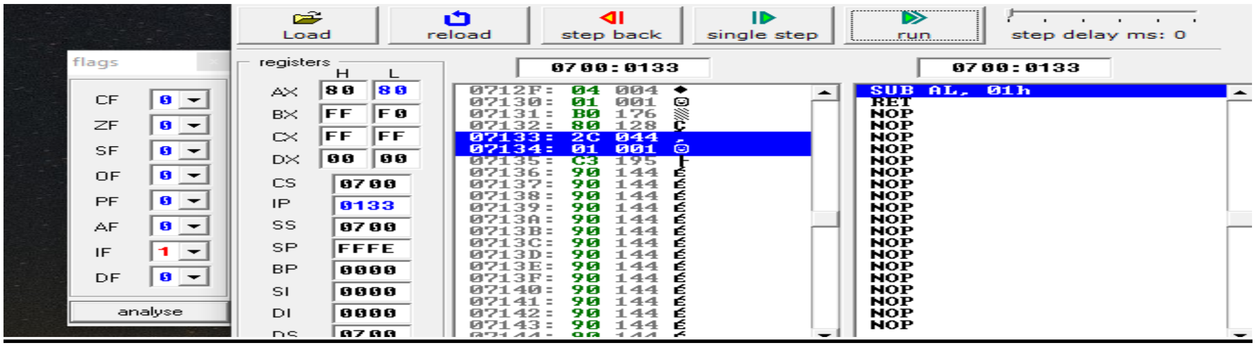


```
flags                 registers   H    L
CF  0                 AX         80   01
ZF  0                 BX         FF   F0
SF  0                 CX         FF   FF
OF  0                 DX         00   00
PF  0                 CS         0700
AF  0                 IP         0131
IF  1                 SS         0700
DF  0                 SP         FFFE
analyse               BP         0000
                      SI         0000
                      DI         0000
                      DS         0700
                      ES         0700

0700:0131                        0700:0131
0712F:  04  004  ♦               MOV  AX,  01000h
07130:  01  001  ☺               INC  AX
07131:  B0  176  ░               DEC  AX
07132:  80  128  Ç               MOV  AX,  00026h
07133:  2C  044  ,               NEG  AX
07134:  01  001  ☺               MOV  BX,  00030h
07135:  C3  195  ├               SUB  BX,  040h
07136:  90  144  É               ADD  AX,  BX
07137:  90  144  É               MOV  CX,  00001h
07138:  90  144  É               SUB  CX,  01h
07139:  90  144  É               MOV  AX,  0FFFFh
0713A:  90  144  É               INC  AX
0713B:  90  144  É               MOV  CX,  00000h
0713C:  90  144  É               SUB  CX,  01h
0713D:  90  144  É               MOV  AX,  07FFFh
0713E:  90  144  É               ADD  AX,  00002h
0713F:  90  144  É               MOV  AL,  0FFh
07140:  90  144  É               ADD  AL,  01h
07141:  90  144  É               MOV  AL,  [0007Fh]
07142:  90  144  É               ADD  AL,  01h
07143:  90  144  É               MOV  AL,  080h
07144:  90  144  É               ---
```

STEP 21:

SAAD UR REHMAN
19k-0218
SEC: C

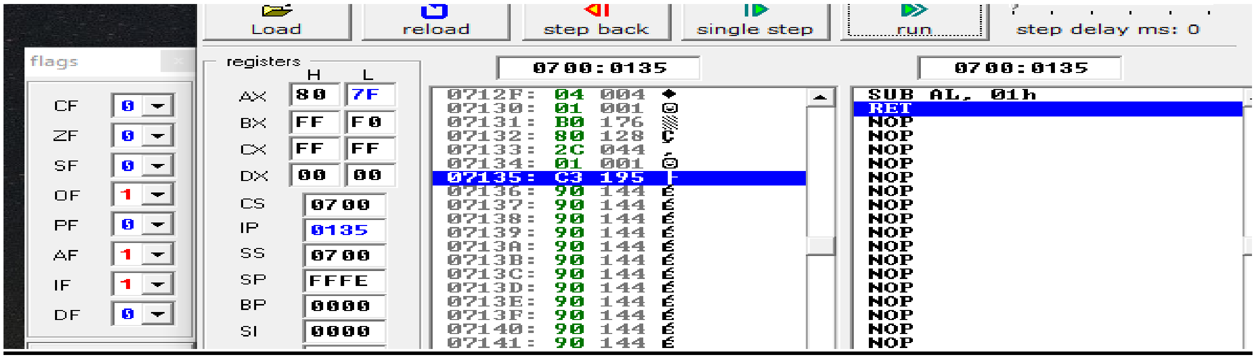


```
flags        registers
                 H    L
CF  0        AX  80   80
ZF  0        BX  FF   F0
SF  0        CX  FF   FF
OF  0        DX  00   00
PF  0        CS  0700
AF  0        IP  0133
IF  1        SS  0700
DF  0        SP  FFFE
             BP  0000
             SI  0000
             DI  0000
             DS  0700
```

```
0700:0133
0712F:  04  004  ◆
07130:  01  001  ☺
07131:  B0  176  ░
07132:  80  128  Ç
07133:  2C  044  ,
07134:  01  001  ☺
07135:  C3  195  ├
07136:  90  144  É
07137:  90  144  É
07138:  90  144  É
07139:  90  144  É
0713A:  90  144  É
0713B:  90  144  É
0713C:  90  144  É
0713D:  90  144  É
0713E:  90  144  É
0713F:  90  144  É
07140:  90  144  É
07141:  90  144  É
07142:  90  144  É
07143:  90  144  É
```

```
0700:0133
SUB AL, 01h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
```

STEP 22:



```
flags        registers
                 H    L
CF  0        AX  80   7F
ZF  0        BX  FF   F0
SF  0        CX  FF   FF
OF  1        DX  00   00
PF  0        CS  0700
AF  1        IP  0135
IF  1        SS  0700
DF  0        SP  FFFE
             BP  0000
             SI  0000
```

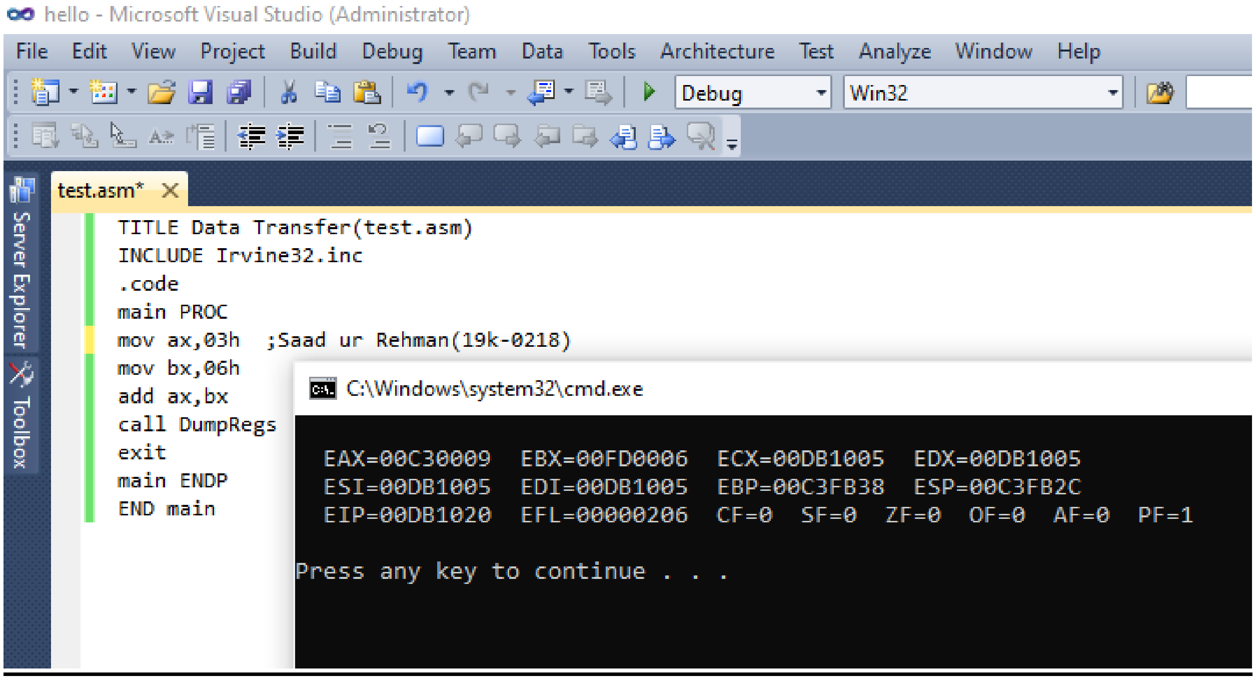
```
0700:0135
0712F:  04  004  ◆
07130:  01  001  ☺
07131:  B0  176  ░
07132:  80  128  Ç
07133:  2C  044  ,
07134:  01  001  ☺
07135:  C3  195  ├
07136:  90  144  É
07137:  90  144  É
07138:  90  144  É
07139:  90  144  É
0713A:  90  144  É
0713B:  90  144  É
0713C:  90  144  É
0713D:  90  144  É
0713E:  90  144  É
0713F:  90  144  É
07140:  90  144  É
07141:  90  144  É
```

```
0700:0135
SUB AL, 01h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
```

Q4)Indicate whether or not each of the following instructions is valid OR not.Run the each instruction in .code segment and attached the snip. Register assume to any value. For example you may take ax=2 or any other integer.

a. add ax,bx

SAAD UR REHMAN
19k-0218
SEC: C


hello - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
Win32
test.asm*
Server Explorer
Toolbox
C:\Windows\system32\cmd.exe

```
TITLE Data Transfer(test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov ax,03h  ;Saad ur Rehman(19k-0218)
mov bx,06h
add ax,bx
call DumpRegs
exit
main ENDP
END main
```

```
  EAX=00C30009  EBX=00FD0006  ECX=00DB1005  EDX=00DB1005
  ESI=00DB1005  EDI=00DB1005  EBP=00C3FB38  ESP=00C3FB2C
  EIP=00DB1020  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1

Press any key to continue . . .
```

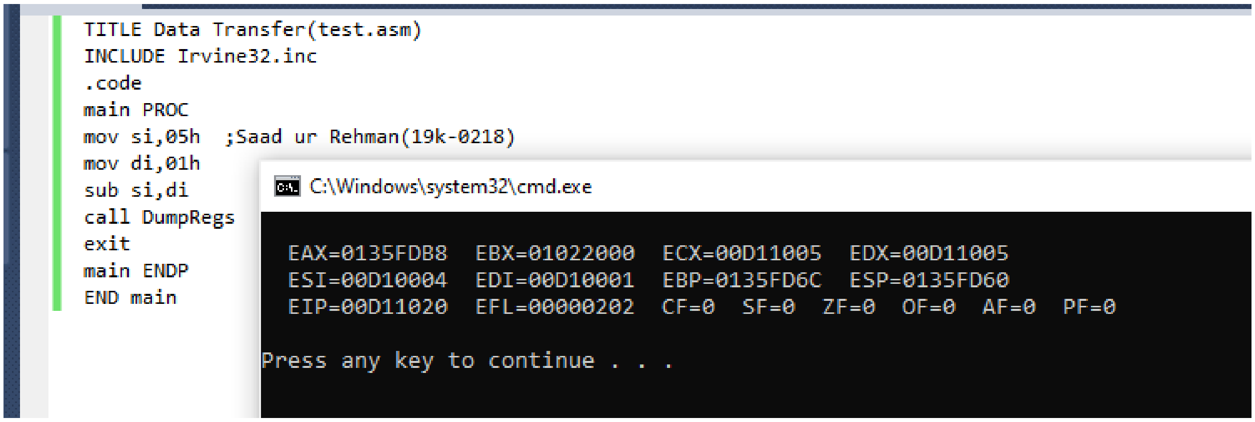
b. add dx,bl

This statement is invalid because dx is 16 bit register and bl is 8 bit register we can add these both different registers

c. add ecx,dx

This statement is invalid because size of destination which is 32 bit and size of source register which is 16 bit is different

d. sub si,di



```
TITLE Data Transfer(test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov si,05h  ;Saad ur Rehman(19k-0218)
mov di,01h
sub si,di
call DumpRegs
exit
main ENDP
END main
```

```
  EAX=0135FDB8  EBX=01022000  ECX=00D11005  EDX=00D11005
  ESI=00D10004  EDI=00D10001  EBP=0135FD6C  ESP=0135FD60
  EIP=00D11020  EFL=00000202  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=0

Press any key to continue . . .
```

SAAD UR REHMAN
19k-0218
SEC: C

```
e. add bx,90000
```

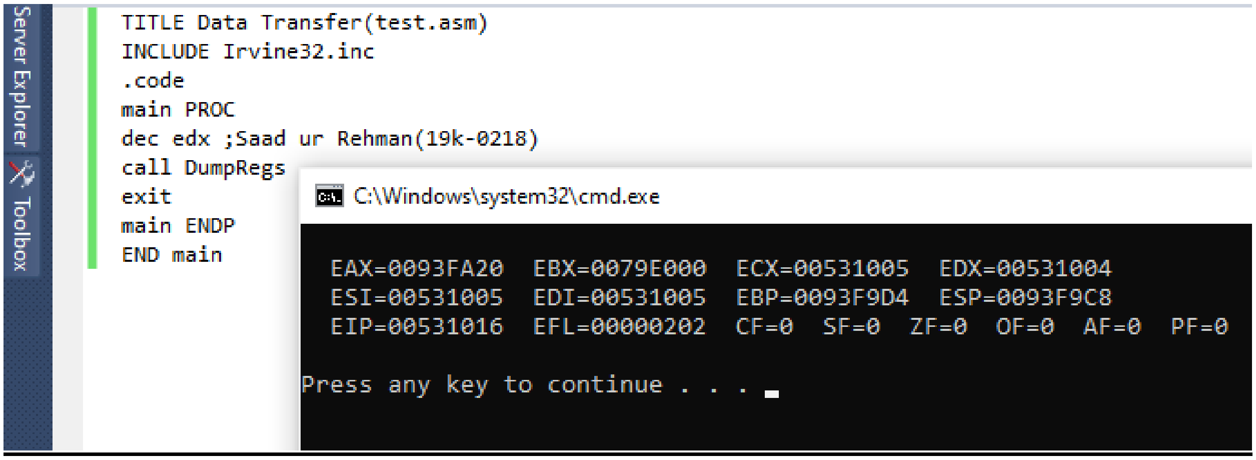
<u>This statement is invalid because 90000 value is too large to fir in bx register.</u>

```
f. sub ds,1
```

```
g. dec ip
```

<u>This instruction is invalid because ip is not defined.</u>

```
h. dec edx
```

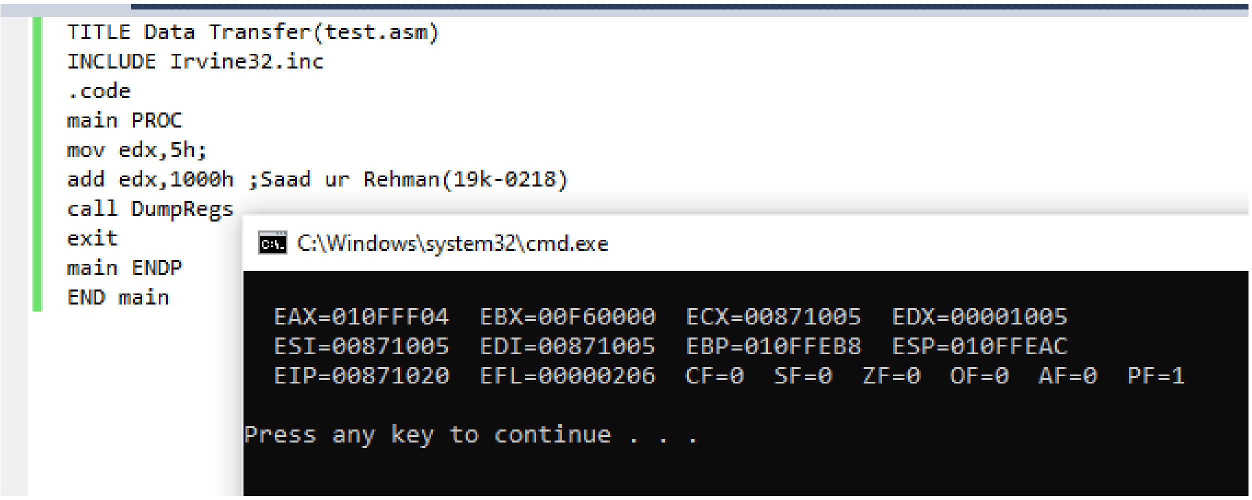


```
TITLE Data Transfer(test.asm)
INCLUDE Irvine32.inc
.code
main PROC
dec edx ;Saad ur Rehman(19k-0218)
call DumpRegs
exit
main ENDP
END main
```

```
  EAX=0093FA20  EBX=0079E000  ECX=00531005  EDX=00531004
  ESI=00531005  EDI=00531005  EBP=0093F9D4  ESP=0093F9C8
  EIP=00531016  EFL=00000202  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=0

Press any key to continue . . .
```

```
i.    add edx,1000h
```



```
TITLE Data Transfer(test.asm)
INCLUDE Irvine32.inc
.code
main PROC
mov edx,5h;
add edx,1000h ;Saad ur Rehman(19k-0218)
call DumpRegs
exit
main ENDP
END main
```

```
  EAX=010FFF04  EBX=00F60000  ECX=00871005  EDX=00001005
  ESI=00871005  EDI=00871005  EBP=010FFEB8  ESP=010FFEAC
  EIP=00871020  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1

Press any key to continue . . .
```

```
j. sub ah,126h
```

This statement is invalid because ah can store 2 byte only while 126h(0001 0010 0110) is exceeding 2 byte

k. sub al,256h

This statement is invalid because ah can store 2 byte only while 126h(0001 0010 0110) is exceeding 2 byte

l. inc ax,1

This statement is invalid because inc can have only one operand.Like inc ax