Fall Semester 2021-22

Microprocessors and Interfacing LAB

CSE2006

Slot – L43+L44

Digital Assignment 5

Name: Ishan Sagar Jogalekar

Reg. No: 19BCE2250

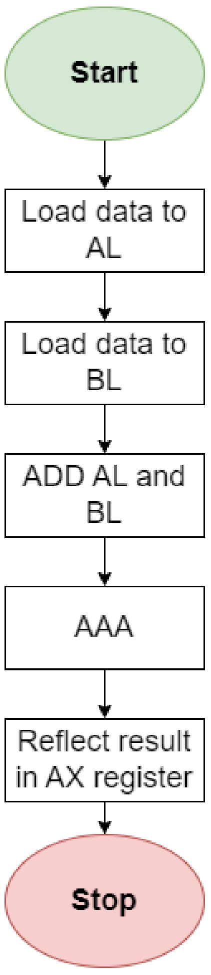
Arithmetic Adjustment Operation –

1. AAA –

Input:

ASCII Adjust after Addition, consider 2 input numbers 8 bit for this program so that result displayed as their arithmetic adjust addition.

Flowchart:

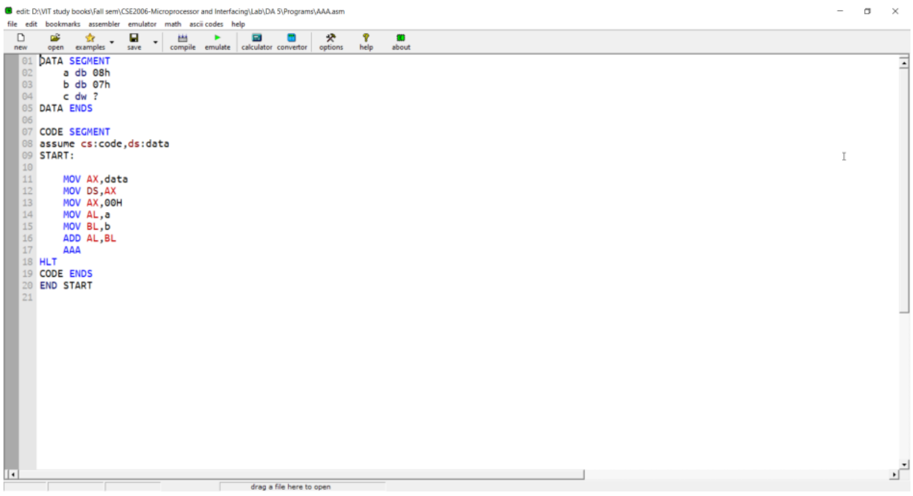


Algorithm:

1. Create variable and load numbers as input in data segment.
2. Load numbers into AL and BL respectively and perform ADD instruction between AL and BL registers.
3. If low nibble of AL > 9 or AF = 1 then:
    a. AL = AL + 6,
    b. AH = AH + 1,
    c. AF = 1,
    d. CF = 1
4. else
    a. AF=0,
    b. CF=0

Program:

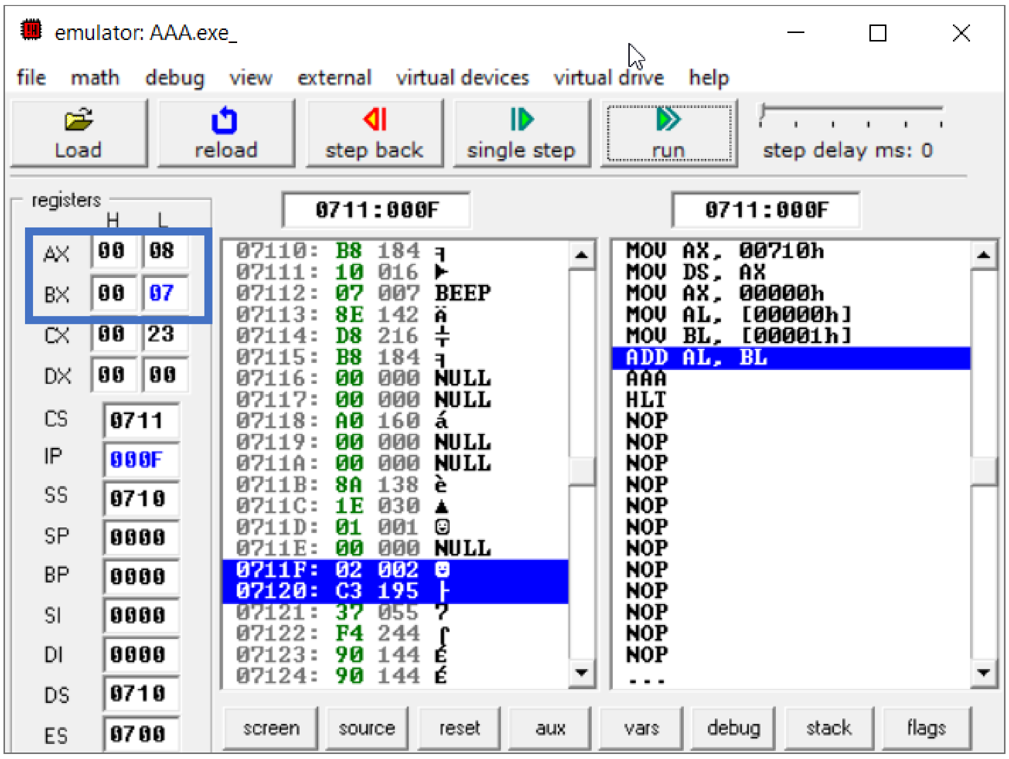
```
DATA SEGMENT
   a db 08h
   b db 07h
   c dw ?
DATA ENDS
CODE SEGMENT
assume cs:code,ds:data
START:
    MOV AX,data
    MOV DS,AX
    MOV AX,00H
    MOV AL,a
    MOV BL,b
    ADD AL,BL
    AAA
HLT
CODE ENDS
END START
```


edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 5\Programs\AAA.asm

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
drag a file here to open

```
DATA SEGMENT
    a db 08h
    b db 07h
    c dw ?
DATA ENDS

CODE SEGMENT
assume cs:code,ds:data
START:

    MOV AX,data
    MOV DS,AX
    MOV AX,00H
    MOV AL,a
    MOV BL,b
    ADD AL,BL
    AAA
HLT
CODE ENDS
END START
```

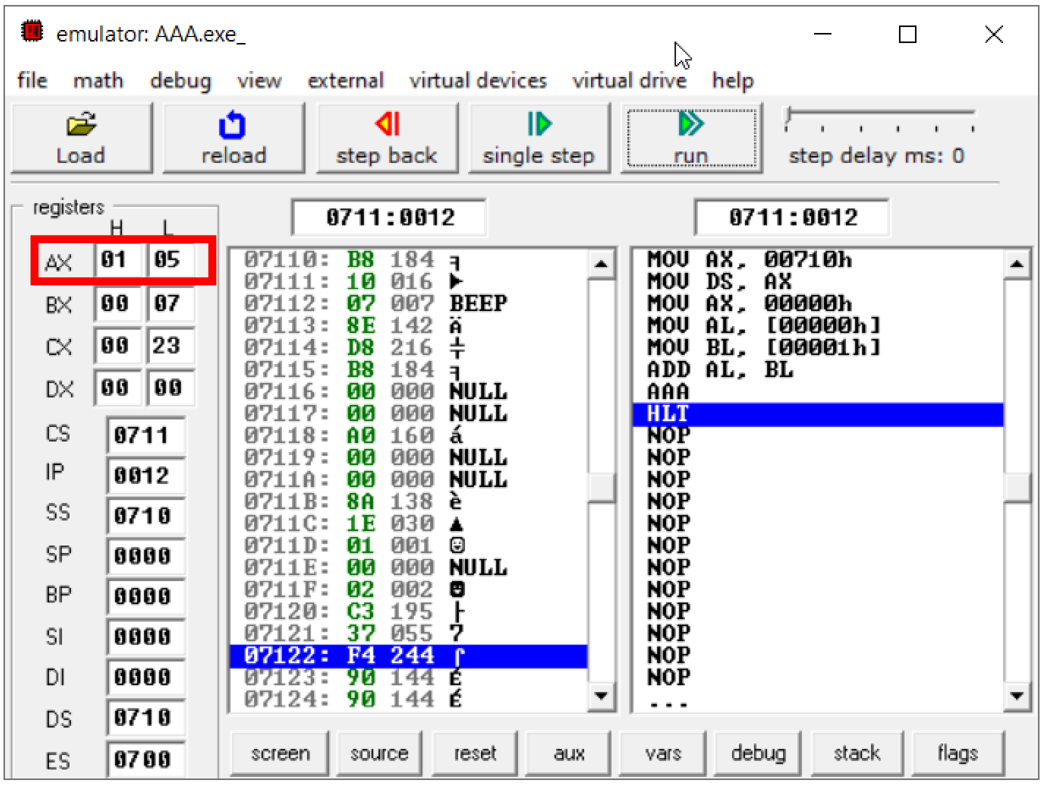
Output:



```
registers
     H   L
AX   00  08
BX   00  07
CX   00  23
DX   00  00
CS   0711
IP   000F
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0710
ES   0700
```

```
0711:000F

07110: B8 184 ╕
07111: 10 016 ►
07112: 07 007 BEEP
07113: 8E 142 Ä
07114: D8 216 ╪
07115: B8 184 ╕
07116: 00 000 NULL
07117: 00 000 NULL
07118: A0 160 á
07119: 00 000 NULL
0711A: 00 000 NULL
0711B: 8A 138 è
0711C: 1E 030 ▲
0711D: 01 001 ☺
0711E: 00 000 NULL
0711F: 02 002 ☻
07120: C3 195 ├
07121: 37 055 7
07122: F4 244 ⌠
07123: 90 144 É
07124: 90 144 É
```

```
0711:000F

MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
ADD AL, BL
AAA
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```



```
registers
     H   L
AX   01  05
BX   00  07
CX   00  23
DX   00  00
CS   0711
IP   0012
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0710
ES   0700
```

```
0711:0012

07110: B8 184 ╕
07111: 10 016 ►
07112: 07 007 BEEP
07113: 8E 142 Ä
07114: D8 216 ╪
07115: B8 184 ╕
07116: 00 000 NULL
07117: 00 000 NULL
07118: A0 160 á
07119: 00 000 NULL
0711A: 00 000 NULL
0711B: 8A 138 è
0711C: 1E 030 ▲
0711D: 01 001 ☺
0711E: 00 000 NULL
0711F: 02 002 ☻
07120: C3 195 ├
07121: 37 055 7
07122: F4 244 ⌠
07123: 90 144 É
07124: 90 144 É
```

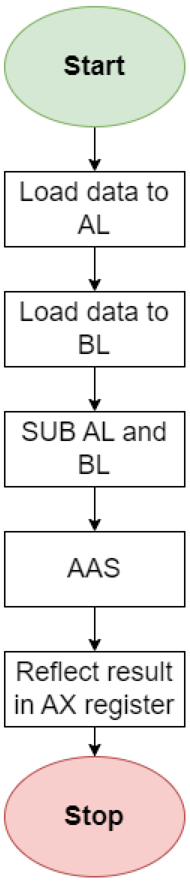
```
0711:0012

MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
ADD AL, BL
AAA
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

2. AAS –

Input:

ASCII Adjust after Subtraction, consider input as two numbers and output should be arithmetic adjusted subtraction of these 2 numbers.

Flowchart:



Algorithm:

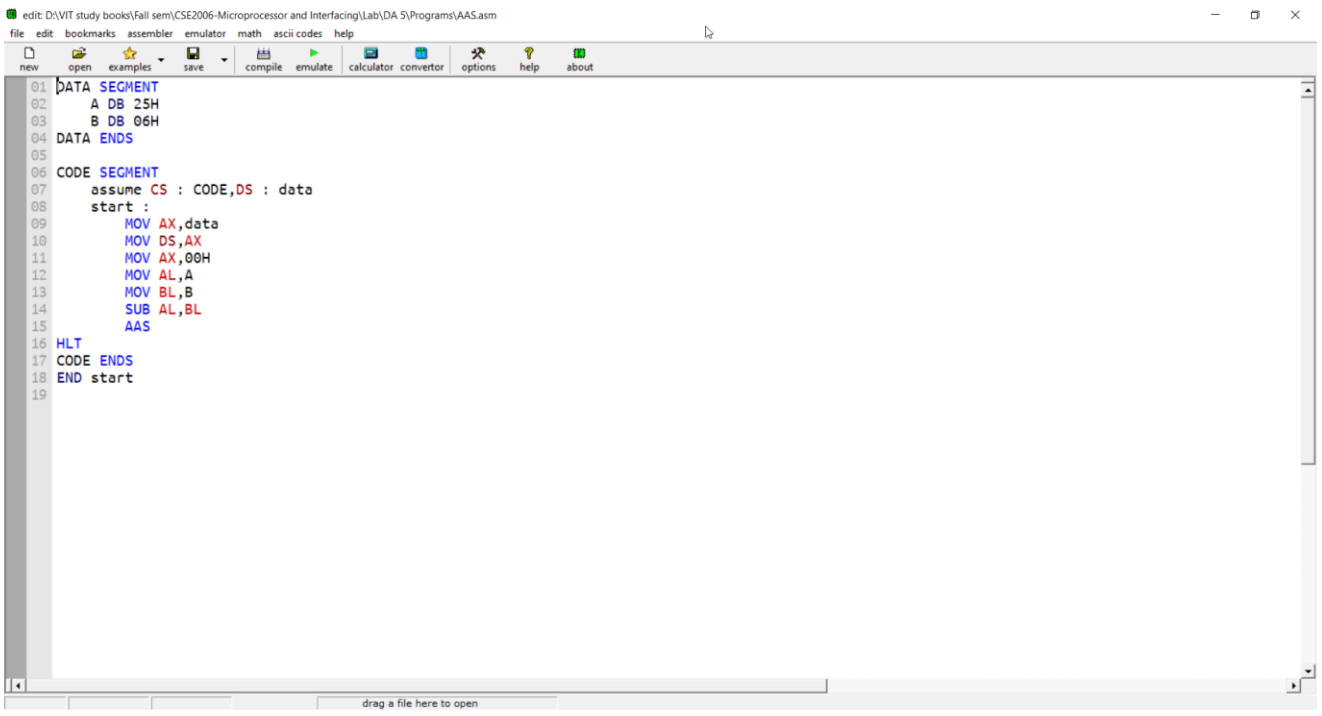
1. Create variable and load numbers as input in data segment.
2. Load numbers into AL and BL respectively and perform SUB instruction between AL and BL registers.
3. If:

   low nibble of AL > 9 or AF = 1 then: AL=AL-6, AH=AH-1, AF=1, CF=1
4. else:

   AF=0, CF=0 In both the cases Clear the Higher Nibble of AL.

Program:

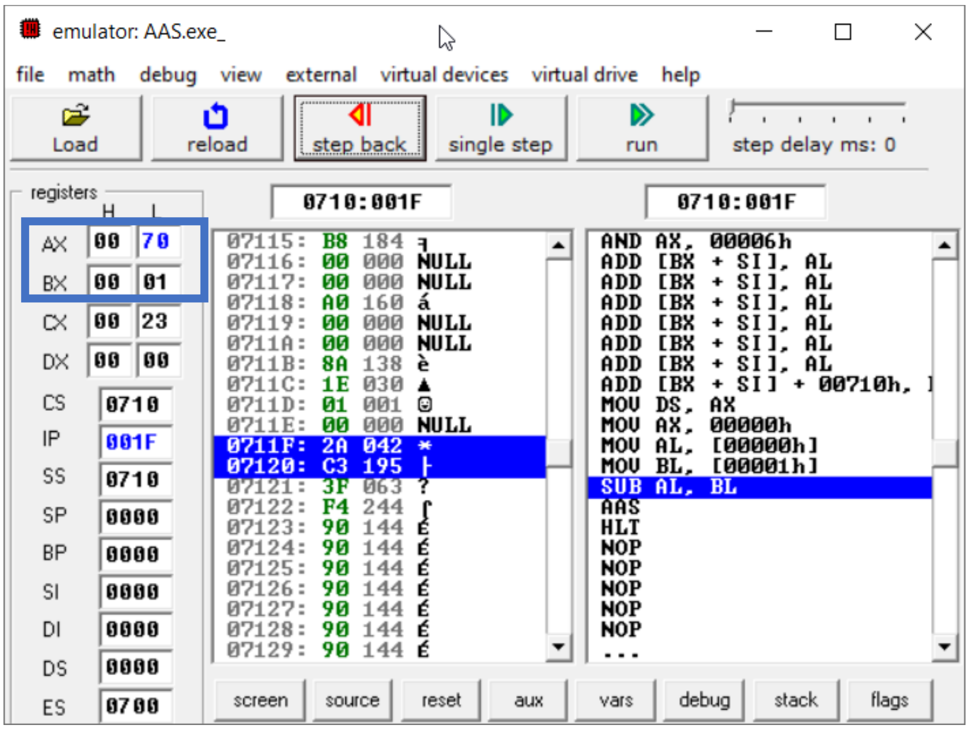
```
DATA SEGMENT
   A DB 25H
   B DB 06H
DATA ENDS
CODE SEGMENT
   assume CS : CODE,DS : data
   start :
           MOV AX,data
           MOV DS,AX
           MOV AX,00H
           MOV AL,A
           MOV BL,B
           SUB AL,BL
           AAS
HLT
CODE ENDS
END start
```


edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 5\Programs\AAS.asm

```
DATA SEGMENT
    A DB 25H
    B DB 06H
DATA ENDS

CODE SEGMENT
    assume CS : CODE,DS : data
    start :
        MOV AX,data
        MOV DS,AX
        MOV AX,00H
        MOV AL,A
        MOV BL,B
        SUB AL,BL
        AAS
HLT
CODE ENDS
END start
```

drag a file here to open

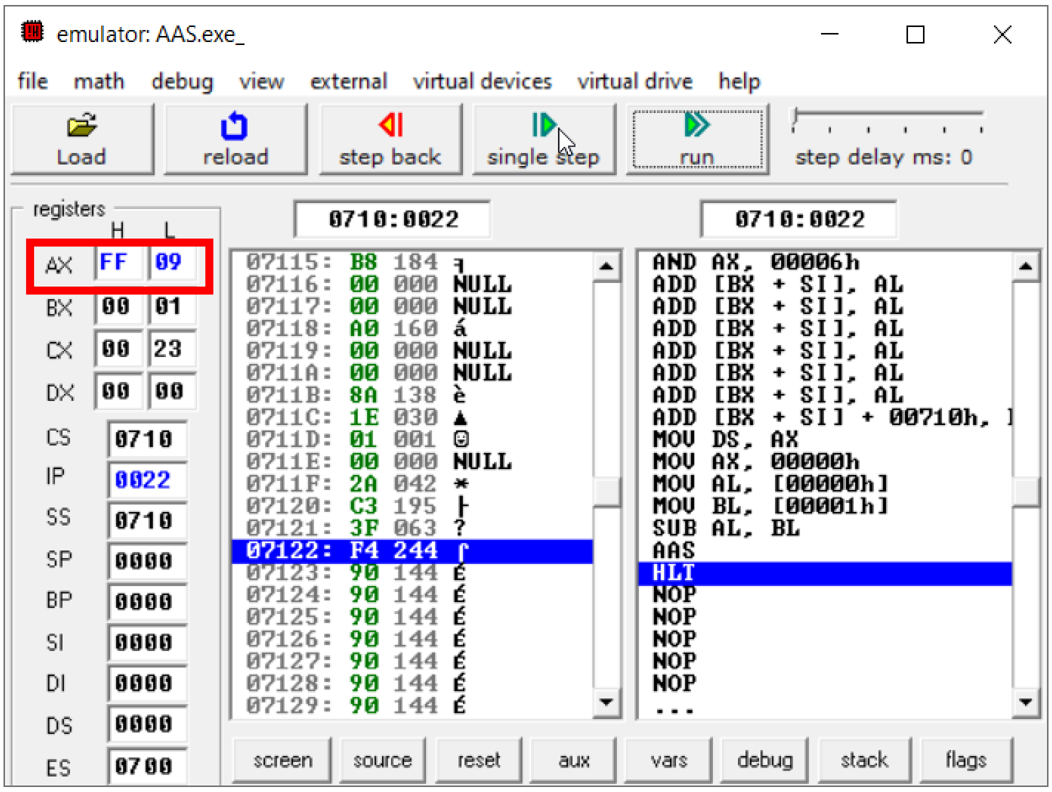
Output:


emulator: AAS.exe_

file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   00  70
BX   00  01
CX   00  23
DX   00  00
CS   0710
IP   001F
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0000
ES   0700
```

```
0710:001F                         0710:001F
07115: B8 184 ╕                   AND AX, 00006h
07116: 00 000 NULL                ADD [BX + SI], AL
07117: 00 000 NULL                ADD [BX + SI], AL
07118: A0 160 á                   ADD [BX + SI], AL
07119: 00 000 NULL                ADD [BX + SI], AL
0711A: 00 000 NULL                ADD [BX + SI], AL
0711B: 8A 138 è                   ADD [BX + SI], AL
0711C: 1E 030 ▲                   ADD [BX + SI] + 00710h, ]
0711D: 01 001 ☺                   MOV DS, AX
0711E: 00 000 NULL                MOV AX, 00000h
0711F: 2A 042 *                   MOV AL, [00000h]
07120: C3 195 ├                   MOV BL, [00001h]
07121: 3F 063 ?                   SUB AL, BL
07122: F4 244 ⌠                   AAS
07123: 90 144 É                   HLT
07124: 90 144 É                   NOP
07125: 90 144 É                   NOP
07126: 90 144 É                   NOP
07127: 90 144 É                   NOP
07128: 90 144 É                   NOP
07129: 90 144 É                   ...
```


emulator: AAS.exe_

file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   FF  09
BX   00  01
CX   00  23
DX   00  00
CS   0710
IP   0022
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0000
ES   0700
```

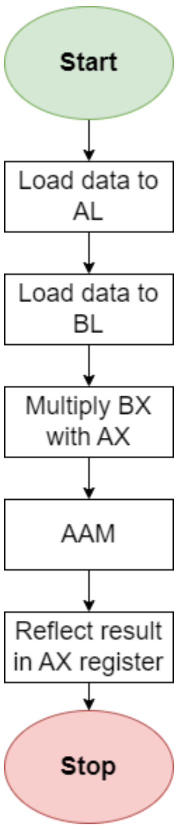
```
0710:0022                         0710:0022
07115: B8 184 ╕                   AND AX, 00006h
07116: 00 000 NULL                ADD [BX + SI], AL
07117: 00 000 NULL                ADD [BX + SI], AL
07118: A0 160 á                   ADD [BX + SI], AL
07119: 00 000 NULL                ADD [BX + SI], AL
0711A: 00 000 NULL                ADD [BX + SI], AL
0711B: 8A 138 è                   ADD [BX + SI], AL
0711C: 1E 030 ▲                   ADD [BX + SI] + 00710h, ]
0711D: 01 001 ☺                   MOV DS, AX
0711E: 00 000 NULL                MOV AX, 00000h
0711F: 2A 042 *                   MOV AL, [00000h]
07120: C3 195 ├                   MOV BL, [00001h]
07121: 3F 063 ?                   SUB AL, BL
07122: F4 244 ⌠                   AAS
07123: 90 144 É                   HLT
07124: 90 144 É                   NOP
07125: 90 144 É                   NOP
07126: 90 144 É                   NOP
07127: 90 144 É                   NOP
07128: 90 144 É                   NOP
07129: 90 144 É                   ...
```

3. AAM –

Input:

ASCII Adjust after Multiplication, consider input as two numbers and output should be arithmetic adjusted multiplication of these 2 numbers.

Flowchart:

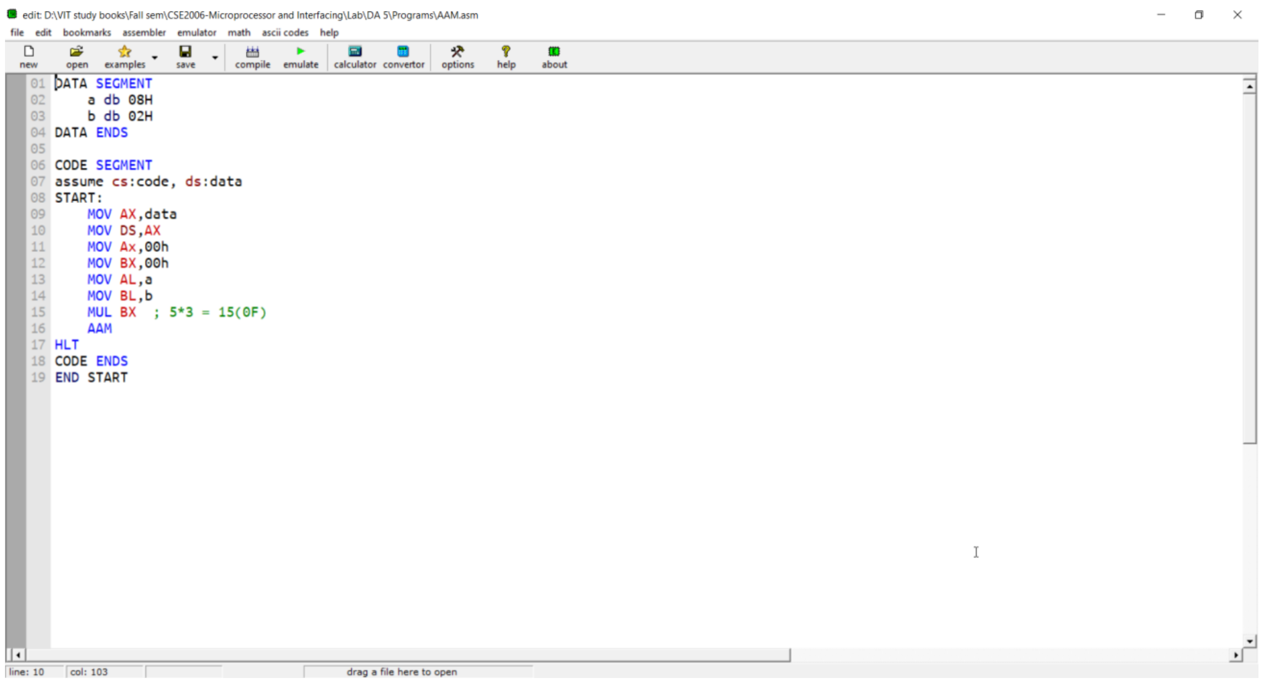


Algorithm:

1. Create variable and load numbers as input in data segment.
2. Load numbers into AL and BL respectively and perform MUL instruction between AL and BL registers.
3. Then for AAM instruction, resultant value as AH = AL / 10 and AL = remainder.
4. HLT the program.

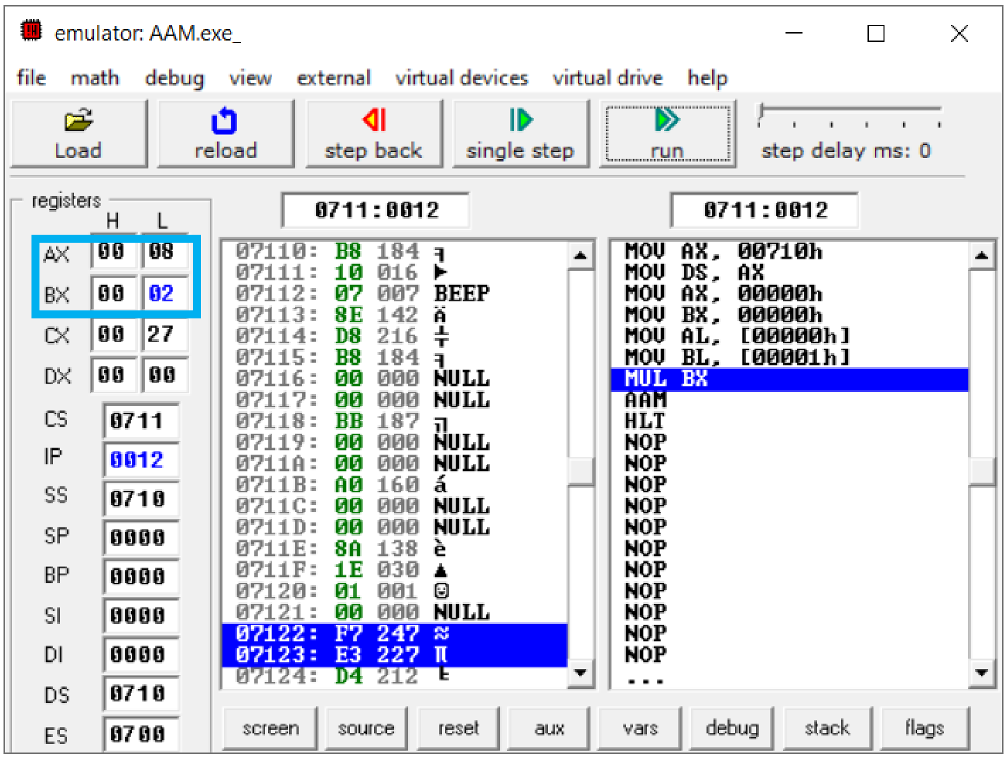
Program:

```
DATA SEGMENT
        a db 08H
        b db 02H
DATA ENDS
CODE SEGMENT
assume cs:code, ds:data
START:
        MOV AX,data
        MOV DS,AX
        MOV Ax,00h
        MOV BX,00h
        MOV AL,a
        MOV BL,b
        MUL BX  ; 5*3 = 15(0F)
        AAM
HLT
CODE ENDS
END START
```


edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 5\Programs\AAM.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
line: 10 col: 103
drag a file here to open

```
DATA SEGMENT
    a db 08H
    b db 02H
DATA ENDS

CODE SEGMENT
assume cs:code, ds:data
START:
    MOV AX,data
    MOV DS,AX
    MOV Ax,00h
    MOV BX,00h
    MOV AL,a
    MOV BL,b
    MUL BX   ; 5*3 = 15(0F)
    AAM
HLT
CODE ENDS
END START
```

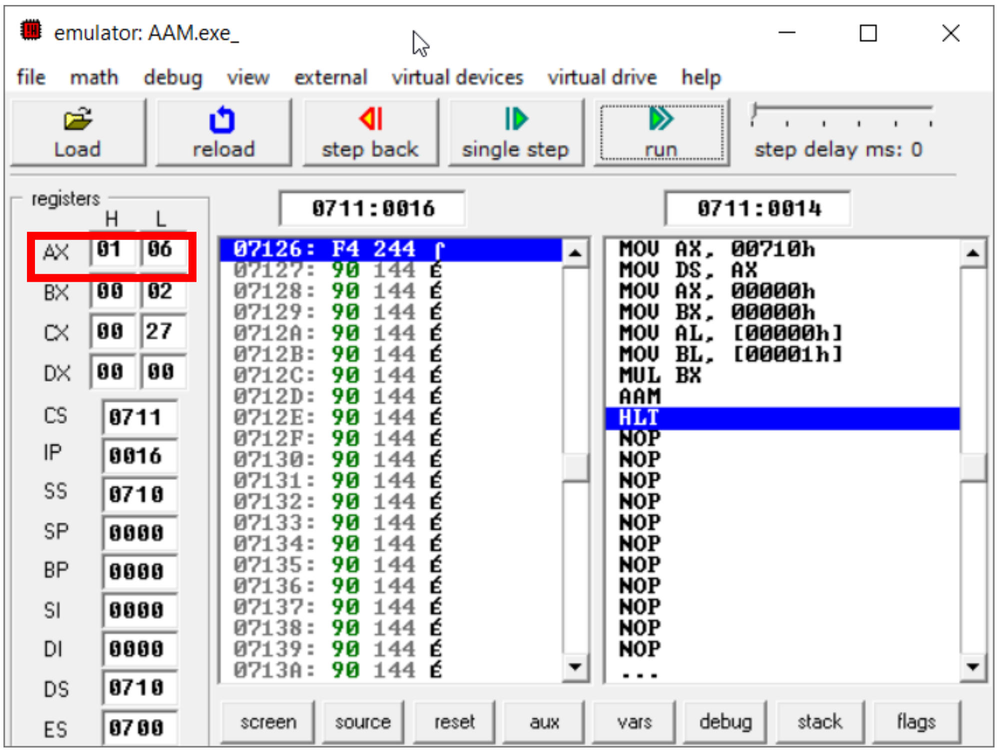
Output:


emulator: AAM.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   00  08
BX   00  02
CX   00  27
DX   00  00
CS   0711
IP   0012
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0710
ES   0700
```

```
0711:0012
07110: B8 184 ╕
07111: 10 016 ►
07112: 07 007 BEEP
07113: 8E 142 Ä
07114: D8 216 ╪
07115: B8 184 ╕
07116: 00 000 NULL
07117: 00 000 NULL
07118: BB 187 ╗
07119: 00 000 NULL
0711A: 00 000 NULL
0711B: A0 160 á
0711C: 00 000 NULL
0711D: 00 000 NULL
0711E: 8A 138 è
0711F: 1E 030 ▲
07120: 01 001 ☺
07121: 00 000 NULL
07122: F7 247 ≈
07123: E3 227 π
07124: D4 212 ╘
```

```
0711:0012
MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
MUL BX
AAM
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```


emulator: AAM.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   01  06
BX   00  02
CX   00  27
DX   00  00
CS   0711
IP   0016
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0710
ES   0700
```

```
0711:0016
07126: F4 244 ⌠
07127: 90 144 É
07128: 90 144 É
07129: 90 144 É
0712A: 90 144 É
0712B: 90 144 É
0712C: 90 144 É
0712D: 90 144 É
0712E: 90 144 É
0712F: 90 144 É
07130: 90 144 É
07131: 90 144 É
07132: 90 144 É
07133: 90 144 É
07134: 90 144 É
07135: 90 144 É
07136: 90 144 É
07137: 90 144 É
07138: 90 144 É
07139: 90 144 É
0713A: 90 144 É
```

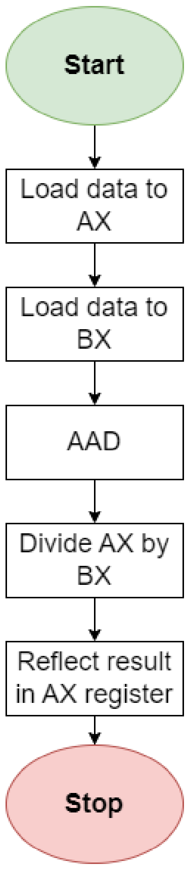
```
0711:0014
MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
MUL BX
AAM
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

4. AAD –

Input:

ASCII Adjust after Division, consider input as two numbers and output should be arithmetic adjusted division of one number to another.

Flowchart:

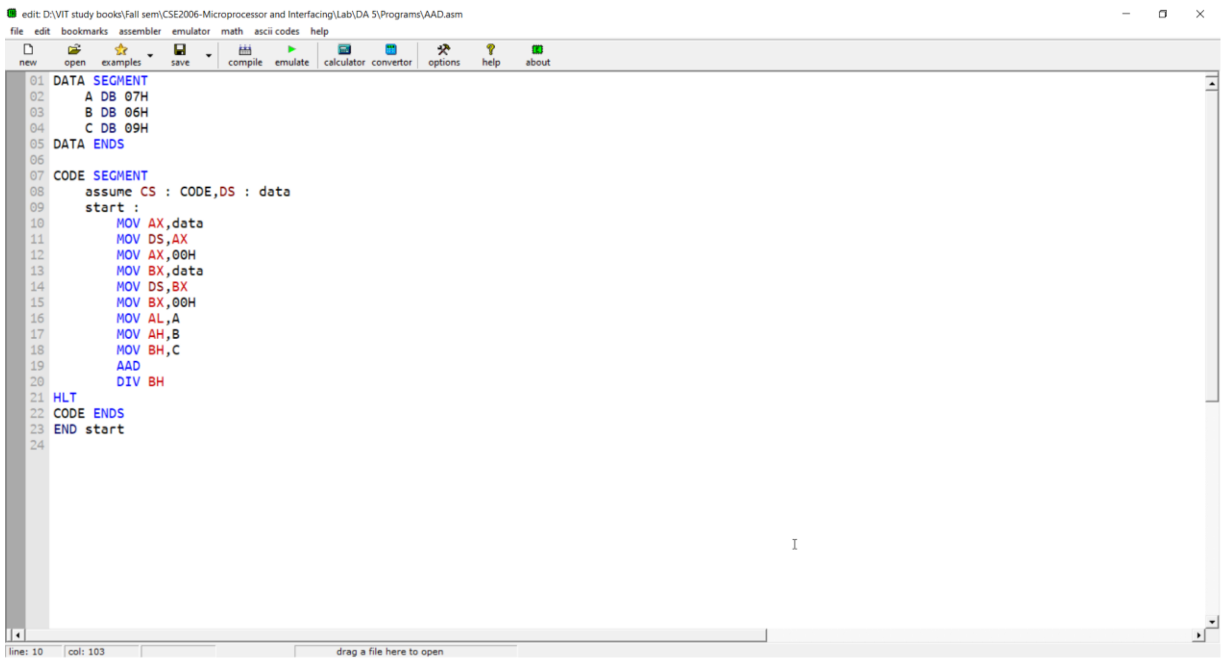


Algorithm:

1. Create variable and load numbers as input in data segment.
2. Load numbers into AX and BX respectively and perform DIV instruction AX by BX.
3. sets the value in the AL register to (AL + (10 * AH)), and then clears the AH register to 00H.
4. The value in the AX register is then equal to the binary equivalent of the original unpacked two-digit (base 10) number in registers AH and AL.
5. HLT the program.

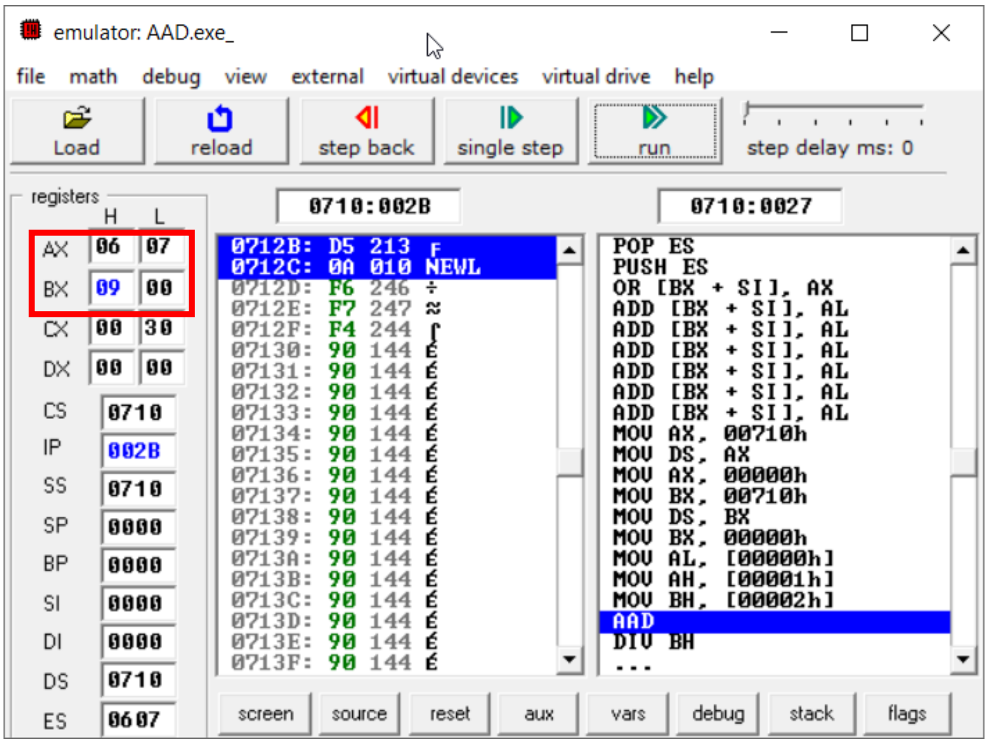
Program:

```
DATA SEGMENT

        A DB 07H
        B DB 06H
        C DB 09H
DATA ENDS
CODE SEGMENT
        assume CS : CODE,DS : data
        start :
                MOV AX,data
                MOV DS,AX
                MOV AX,00H
                MOV BX,data
                MOV DS,BX
                MOV BX,00H
                MOV AL,A
                MOV AH,B
                MOV BH,C
                AAD
                DIV BH
HLT
CODE ENDS
END start
```


edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 5\Programs\AAD.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
line: 10 col: 103 drag a file here to open

```
DATA SEGMENT
    A DB 07H
    B DB 06H
    C DB 09H
DATA ENDS

CODE SEGMENT
    assume CS : CODE,DS : data
    start :
        MOV AX,data
        MOV DS,AX
        MOV AX,00H
        MOV BX,data
        MOV DS,BX
        MOV BX,00H
        MOV AL,A
        MOV AH,B
        MOV BH,C
        AAD
        DIV BH
HLT
CODE ENDS
END start
```

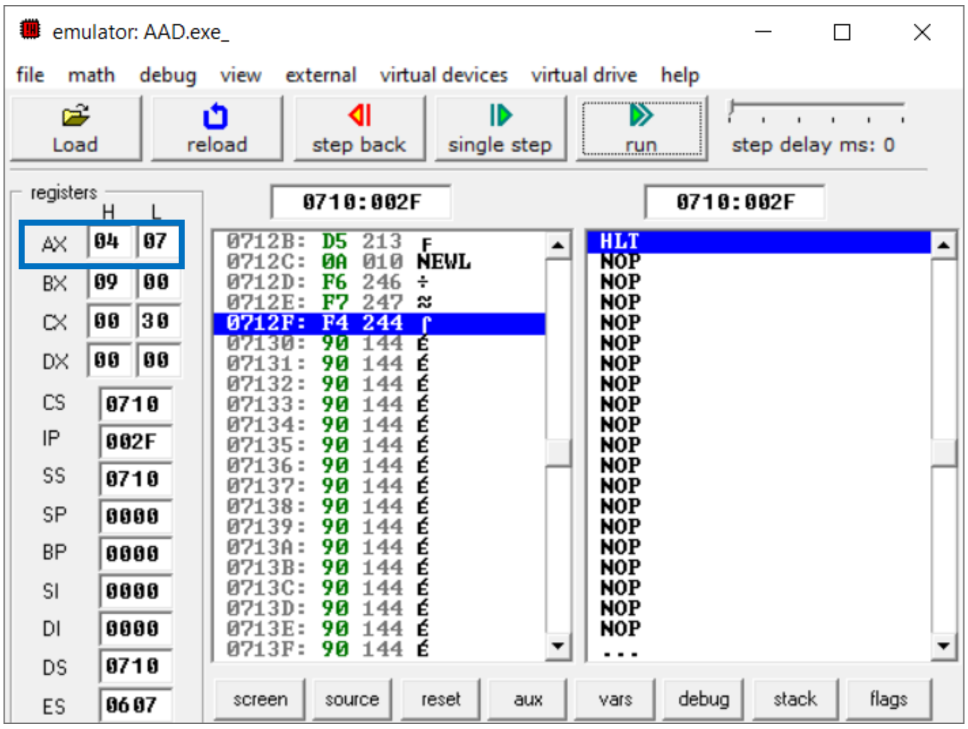
Output:



```
    H  L
AX  06 07
BX  09 00
CX  00 30
DX  00 00
CS  0710
IP  002B
SS  0710
SP  0000
BP  0000
SI  0000
DI  0000
DS  0710
ES  0607
```

```
0710:002B

0712B: D5 213 F
0712C: 0A 010 NEWL
0712D: F6 246 ÷
0712E: F7 247 ≈
0712F: F4 244 ⌠
07130: 90 144 É
07131: 90 144 É
07132: 90 144 É
07133: 90 144 É
07134: 90 144 É
07135: 90 144 É
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
```

```
0710:0027

POP  ES
PUSH ES
OR [BX + SI], AX
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV BX, 00710h
MOV DS, BX
MOV BX, 00000h
MOV AL, [00000h]
MOV AH, [00001h]
MOV BH, [00002h]
AAD
DIV BH
...
```



```
    H  L
AX  04 07
BX  09 00
CX  00 30
DX  00 00
CS  0710
IP  002F
SS  0710
SP  0000
BP  0000
SI  0000
DI  0000
DS  0710
ES  0607
```

```
0710:002F

0712B: D5 213 F
0712C: 0A 010 NEWL
0712D: F6 246 ÷
0712E: F7 247 ≈
0712F: F4 244 ⌠
07130: 90 144 É
07131: 90 144 É
07132: 90 144 É
07133: 90 144 É
07134: 90 144 É
07135: 90 144 É
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
```

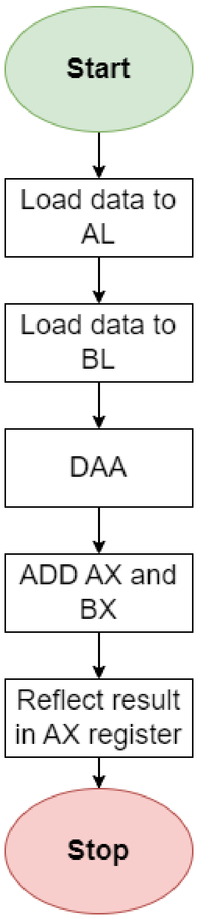
```
0710:002F

HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

5. DAA –

Input:

Decimal Adjust after Addition, consider input as two numbers and output should be decimal adjusted addition of two numbers.

Flowchart:

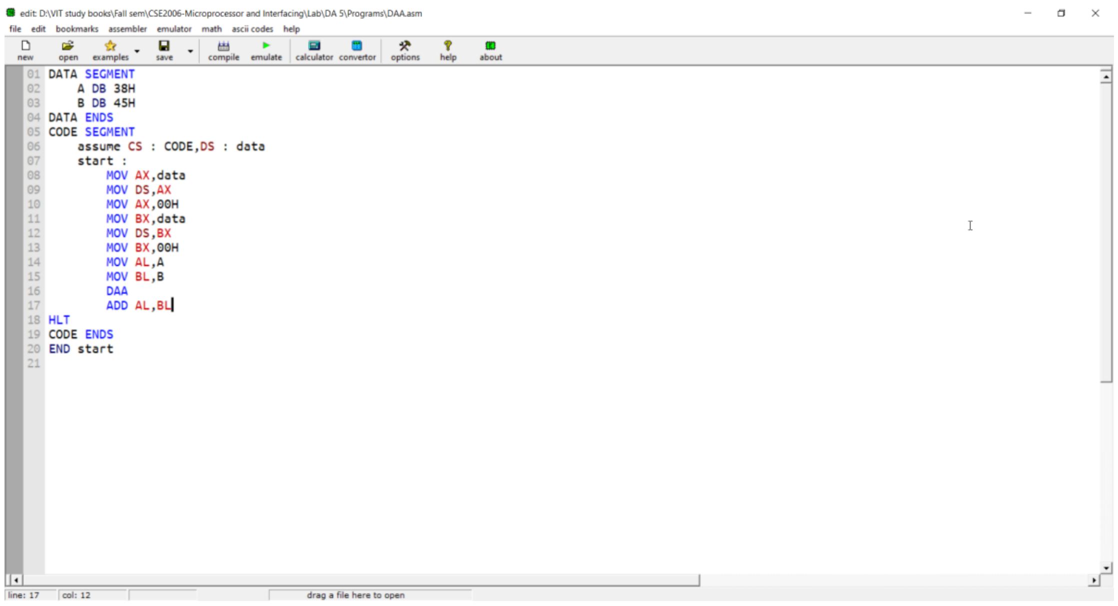


Algorithm:

1. Create variable and load numbers as input in data segment.
2. Load numbers into AL and BL respectively and perform DAA instruction along with ADD instruction.
3. If the digit in the lower four nibbles of AL is greater than 10 (decimal),
4. then subtract 10 and add 1 to the digit in the higher four nibbles of AL.
5. HLT the program.

Program:

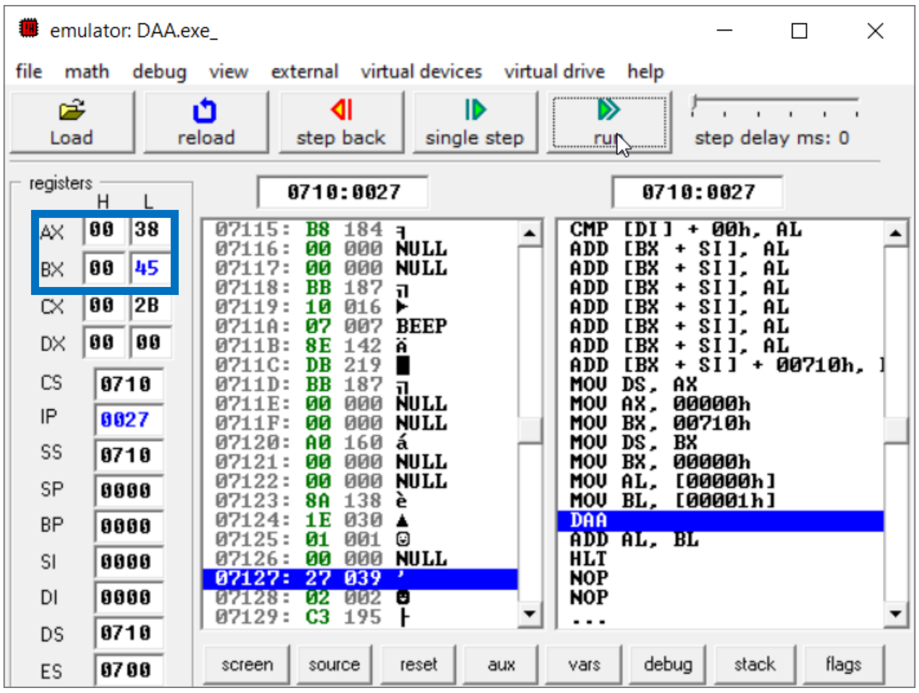
```
DATA SEGMENT
        A DB 38H
        B DB 45H
DATA ENDS
CODE SEGMENT
        assume CS : CODE,DS : data
        start :
                MOV AX,data
                MOV DS,AX
                MOV AX,00H
                MOV BX,data
                MOV DS,BX
                MOV BX,00H
                MOV AL,A
                MOV BL,B
                DAA
                ADD AL,BL
HLT
CODE ENDS
END start
```


edit: D:\VIT study books\Fall sem\CSE2006-Microprocessor and Interfacing\Lab\DA 5\Programs\DAA.asm

```
DATA SEGMENT
    A DB 38H
    B DB 45H
DATA ENDS
CODE SEGMENT
    assume CS : CODE,DS : data
    start :
        MOV AX,data
        MOV DS,AX
        MOV AX,00H
        MOV BX,data
        MOV DS,BX
        MOV BX,00H
        MOV AL,A
        MOV BL,B
        DAA
        ADD AL,BL
HLT
CODE ENDS
END start
```

line: 17 col: 12 drag a file here to open

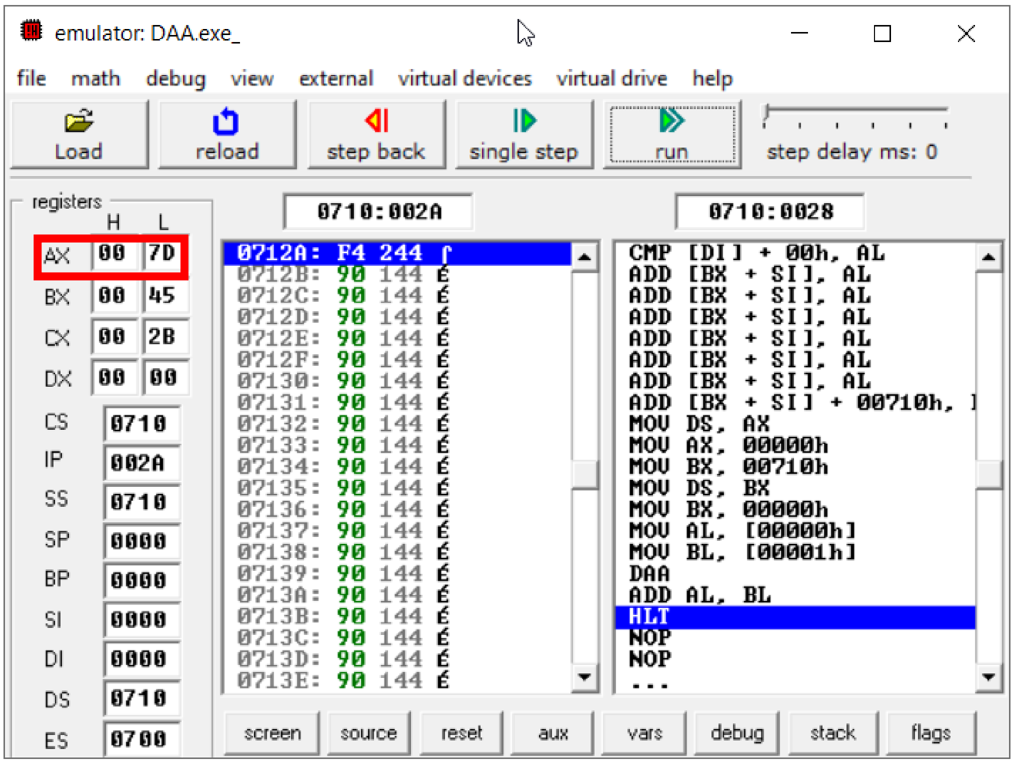
Output:



```
registers
     H  L
AX  00 38
BX  00 45
CX  00 2B
DX  00 00
CS  0710
IP  0027
SS  0710
SP  0000
BP  0000
SI  0000
DI  0000
DS  0710
ES  0700
```

```
0710:0027
07115: B8 184 ╕
07116: 00 000 NULL
07117: 00 000 NULL
07118: BB 187 ╗
07119: 10 016 ►
0711A: 07 007 BEEP
0711B: 8E 142 Ä
0711C: DB 219 █
0711D: BB 187 ╗
0711E: 00 000 NULL
0711F: 00 000 NULL
07120: A0 160 á
07121: 00 000 NULL
07122: 00 000 NULL
07123: 8A 138 è
07124: 1E 030 ▲
07125: 01 001 ☺
07126: 00 000 NULL
07127: 27 039 '
07128: 02 002 ☻
07129: C3 195 ├
```

```
0710:0027
CMP [DI] + 00h, AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI] + 00710h, ]
MOV DS, AX
MOV AX, 00000h
MOV BX, 00710h
MOV DS, BX
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
DAA
ADD AL, BL
HLT
NOP
NOP
...
```



```
registers
     H  L
AX  00 7D
BX  00 45
CX  00 2B
DX  00 00
CS  0710
IP  002A
SS  0710
SP  0000
BP  0000
SI  0000
DI  0000
DS  0710
ES  0700
```

```
0710:002A
0712A: F4 244 ⌠
0712B: 90 144 É
0712C: 90 144 É
0712D: 90 144 É
0712E: 90 144 É
0712F: 90 144 É
07130: 90 144 É
07131: 90 144 É
07132: 90 144 É
07133: 90 144 É
07134: 90 144 É
07135: 90 144 É
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
```

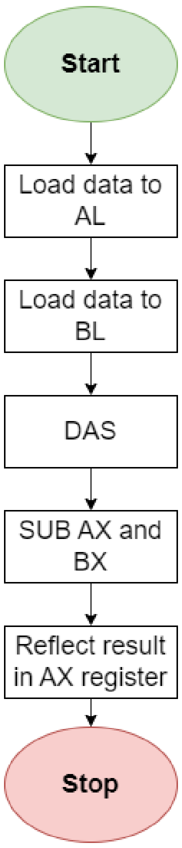
```
0710:0028
CMP [DI] + 00h, AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI] + 00710h, ]
MOV DS, AX
MOV AX, 00000h
MOV BX, 00710h
MOV DS, BX
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
DAA
ADD AL, BL
HLT
NOP
NOP
...
```

6. DAS –

Input:

Decimal Adjust after Subtraction, consider input as two numbers and output should be decimal adjusted subtraction of two numbers.

Flowchart:

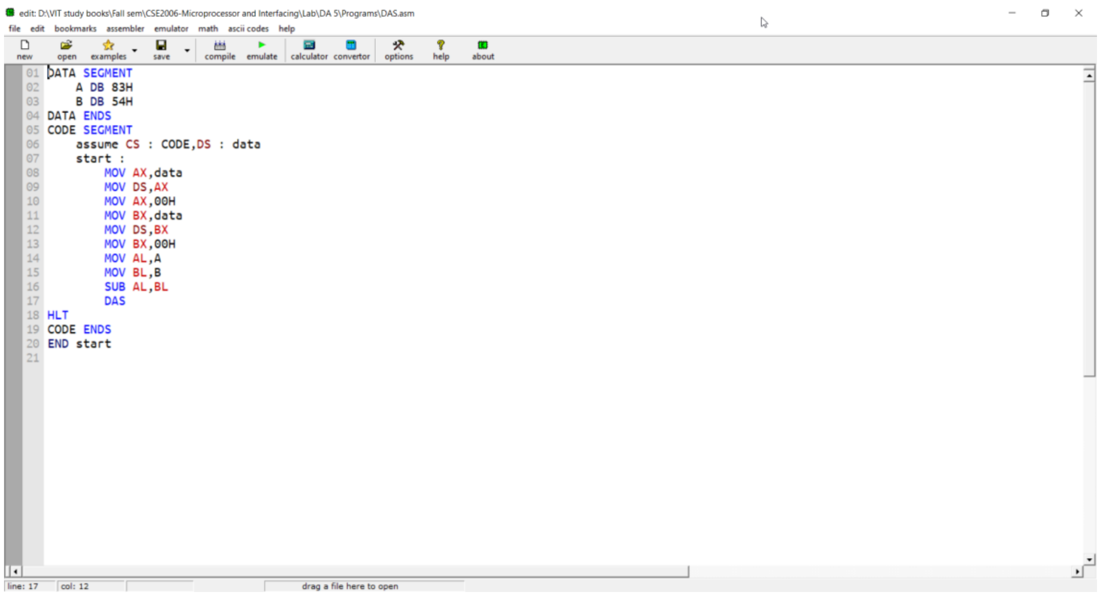


Algorithm:

1. Create variable and load numbers as input in data segment.
2. Load numbers into AL and BL respectively and perform DAS instruction along with SUB instruction.
3. If the lower nibble of AL is higher than the value 9, this instruction will subtract 06 from lower nibble of the AL.
4. If the output of subtraction operation sets the carry flag or if the upper nibble is higher than value 9, it subtracts 60H from the AL.
5. HLT the program.

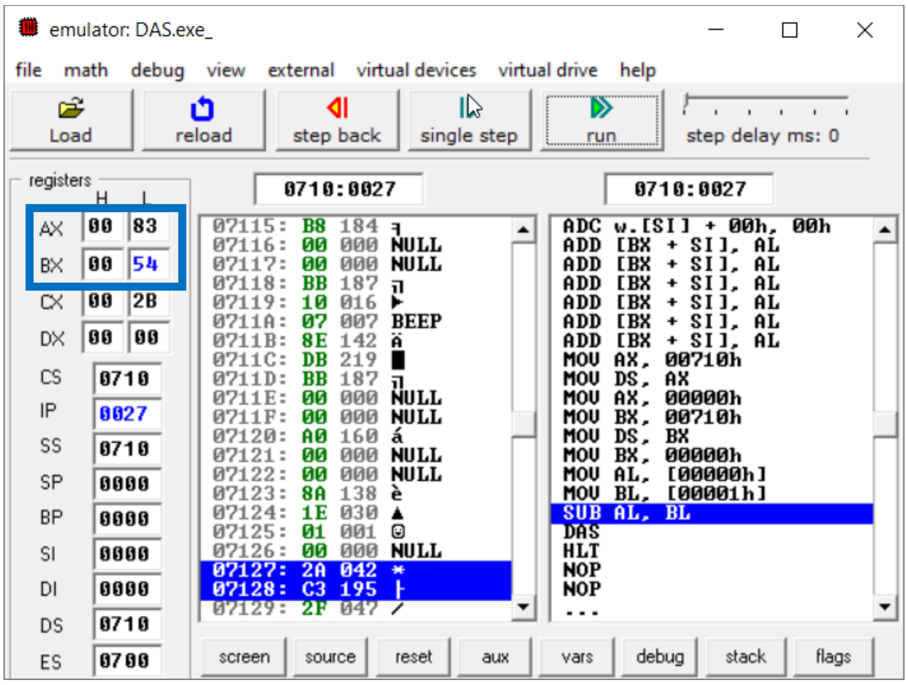
Program:

```
DATA SEGMENT
        A DB 83H
        B DB 54H
DATA ENDS
CODE SEGMENT
        assume CS : CODE,DS : data
        start :
                MOV AX,data
                MOV DS,AX
                MOV AX,00H
                MOV BX,data
                MOV DS,BX
                MOV BX,00H
                MOV AL,A
                MOV BL,B
                SUB AL,BL
                DAS
HLT
CODE ENDS
END start
```



```
DATA SEGMENT
    A DB 83H
    B DB 54H
DATA ENDS
CODE SEGMENT
    assume CS : CODE,DS : data
    start :
        MOV AX,data
        MOV DS,AX
        MOV AX,00H
        MOV BX,data
        MOV DS,BX
        MOV BX,00H
        MOV AL,A
        MOV BL,B
        SUB AL,BL
        DAS
HLT
CODE ENDS
END start
```

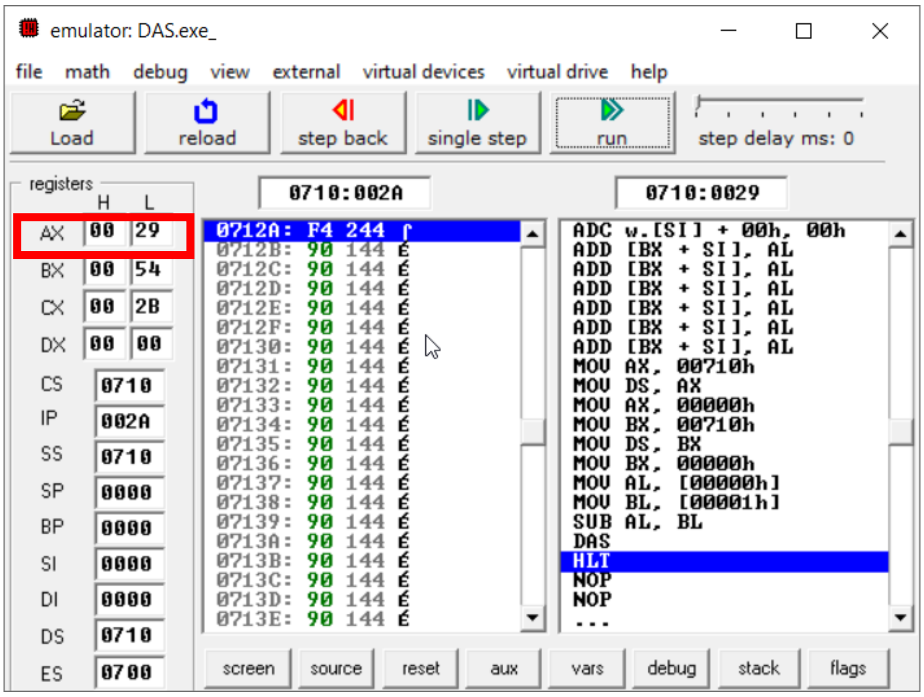
Output:



```
registers
     H  L
AX   00 83
BX   00 54
CX   00 2B
DX   00 00
CS   0710
IP   0027
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0710
ES   0700
```

```
0710:0027
07115: B8 184 ╕
07116: 00 000 NULL
07117: 00 000 NULL
07118: BB 187 ╗
07119: 10 016 ►
0711A: 07 007 BEEP
0711B: 8E 142 Ä
0711C: DB 219 █
0711D: BB 187 ╗
0711E: 00 000 NULL
0711F: 00 000 NULL
07120: A0 160 á
07121: 00 000 NULL
07122: 00 000 NULL
07123: 8A 138 è
07124: 1E 030 ▲
07125: 01 001 ☺
07126: 00 000 NULL
07127: 2A 042 *
07128: C3 195 ├
07129: 2F 047 /
```

```
0710:0027
ADC w.[SI] + 00h, 00h
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV BX, 00710h
MOV DS, BX
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
SUB AL, BL
DAS
HLT
NOP
NOP
...
```



```
registers
     H  L
AX   00 29
BX   00 54
CX   00 2B
DX   00 00
CS   0710
IP   002A
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0710
ES   0700
```

```
0710:002A
0712A: F4 244 ⌠
0712B: 90 144 É
0712C: 90 144 É
0712D: 90 144 É
0712E: 90 144 É
0712F: 90 144 É
07130: 90 144 É
07131: 90 144 É
07132: 90 144 É
07133: 90 144 É
07134: 90 144 É
07135: 90 144 É
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
```

```
0710:0029
ADC w.[SI] + 00h, 00h
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV BX, 00710h
MOV DS, BX
MOV BX, 00000h
MOV AL, [00000h]
MOV BL, [00001h]
SUB AL, BL
DAS
HLT
NOP
NOP
...
```