Lab #04
Addition and Subtraction Instructions

Objective

- Understand Addition and Subtraction instructions
- Working of ASCII and Decimal adjust instruction

Theory

INC (Increment) Instruction

The INC (Increment) instruction increases the content of register by 1.
Increment instruction format:

INC DESTINATION; Destination = Destination + 1

```
E.g.   MOV  AH, FDH     ; AH = FD H
       INC  AH          ; AH = FE H
       INC  AH          ; AH = FF H
       INC  AH          ; AH = 00 H
       INC  AH          ; AH = 01 H
```

INT interrupt number

. DEC (Decrement) instruction

The DEC (decrement) instruction decreases the content of register by 1.

Decrement instruction format:

DEC DESTINATIION; Destination = Destination - 1

```
E.g.   MOV  AH, FDH     ; AH = 02 H
       DEC  AH          ; AH = 01 H
       DEC  AH          ; AH = 00 H
       DEC  AH          ; AH = FF H
       DEC  AH          ; AH = FE H
```

```
TITLE EXAMPLE_PROG_INC
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX
                ;----------
    MOV AH, 01H ;USER INPUT
    INT 21H     ;----------
    MOV BL, AL  ;SAVING VALUE
                ;----------------
    DEC BL      ;DECREMENT BY ONE
    MOV DL, BL  ;----------------
    MOV AH, 02H ;DISPLAYING VALUE
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

```
TITLE EXAMPLE_PROG_DEC
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX
                ;----------
    MOV AH, 01H ;USER INPUT
    INT 21H     ;----------
    MOV BL, AL  ;SAVING VALUE
                ;----------------
    DEC BL      ;DECREMENT BY ONE
    MOV DL, BL  ;----------------
    MOV AH, 02H ;DISPLAYING VALUE
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

ADD (Addition) Instruction

The ADD instruction adds the value of destination to the source and the **result store in the destination register, where source register remains unchanged**.

ADD Instruction format

ADD Destination, Source ; Destination = Destination + Source

ADC (Add with carry) Instruction

ADC instruction adds the two numbers with carry. When two number are added carry may generated this carry store by making the carry flag 1, if no carry is generated then carry flag becomes 0.

ADC Instruction format

ADC Destination, Source ; Destination = Destination + Source + Carry

E.g.

```
        2792
+       3283
--------------------
        5A15
--------------------
```

```
MOV  AL, 92H      ; AL = 92 H
MOV  BL, 83H      ; BL = 83 H
ADD  AL, BL       ; AL = 92 H + 83 H Result is 15 H and C = 1
MOV  CL, AL       ; Saving result of lower byte
MOV  AL, 27H      ; AL = 27 H
MOV  BL, 32H      ; BL = 32 H
ADC  AL, BL       ; AL = 27 H + 32 H +C (C = 1) Result is 5A and C = 0
MOV  CH, AL       ; Saving result of higher byte
```

The CX Register contains 5A15 (Result)

```
TITLE EXAMPLE_PROG_ADD
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 04H
    MOV AL, 04H

    ADD AH, AL  ;RESULT IN AH

    MOV DL, AH  ;----------------
    MOV AH, 02H ;DISPLAYING VALUE
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

```
TITLE EXAMPLE_PROG_ADC
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 09H
    MOV AL, 04H

    ADC AH, AL  ;RESULT IN AH

    MOV DL, AH  ;----------------
    MOV AH, 02H ;DISPLAYING VALUE
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

SUB (Subtract) Instruction

The SUB instruction subtracts the value of source from the value of destination, **where source register remains unchanged.**

SUB Instruction format

SUB Destination, Source ; Destination = Destination - Source

SBB (Subtract with borrow) Instruction

SBB instruction subtracts the two numbers with carry. When two number are subtracted a borrow may needed when borrow is taken carry flag is set to 1. The SBB instruction execute it subtract two number with carry + flag value.

SBB Instruction format

SBB Destination, Source ; Destination = Destination - Source - Carry

E.g.

```
        6826
-       2434
--------------------
        43F2
--------------------
```

```
MOV  AL, 26H      ; AL = 26 H
MOV  BL, 34H      ; BL = 34 H
SUB  AL, BL       ; AL = 26 H - 34 H Result is F2 H and C = 1
MOV  CL, AL       ; Saving result of lower byte
MOV  AL, 68H      ; AL = 68 H
MOV  BL, 24H      ; BL = 24 H
SBB  AL, BL       ; AL = 68 H - 24 H - C (C = 1) Result is 43 and C = 0
MOV  CH, AL       ; Saving result of higher byte
```

```
TITLE EXAMPLE_PROG_SUB
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX
                ;----------
    MOV AH, 09H ;----------
    MOV AL, 04H ;SAVING VALUES
                ;-------------
    SUB AH, AL  ;09-04=05
                ;RESULT IN AH
    MOV DL, AH  ;----------------
    MOV AH, 02H ;DISPLAYING VALUE
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

```
TITLE EXAMPLE_PROG_SBB
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX
                ;----------
    MOV AH, 04H ;----------
    MOV AL, 09H ;SAVING VALUES
                ;-------------
    SBB AH, AL  ;09-04=05
                ;RESULT IN AH
    MOV DL, AH  ;----------------
    MOV AH, 02H ;DISPLAYING VALUE
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

The CX Register contains 43F2 (Result)

AAA (ASCII Adjust for Addition) Instruction

The AAA instruction converts the result in hexadecimal format to decimal format present in AL register after adding two unpacked BCD or two ASCII digits. If lower nibble of AL register is greater then $(9)_H$ or AF is set to 1 then AL is increment by $(6)_H$ and AH is increment by $(1)_H$. This instruction always clear upper nibble of AL register.

AAA instruction format

```
AAA             ; ASCII adjust for addition
```

E.g. Consider $(9)_D + (4)_D = (13)_D$, but processor add it in hexadecimal format so its answer will be $(D)_H$, to convert hexadecimal format into decimal format AAA instruction add 6 is in $(D)_H$ which makes $(13)_H$ so $(03)_H$ is store in AL register and $(01)_H$ is added in AH register.

```
MOV   AL, 09D       ; AL = 09H and AH = 00H
ADD   AL, 04D       ; AL = 0DH and AH = 00H
AAA                 ; AH = 01H and AL = 03H
```

AAS (ASCII Adjust for Subtraction) Instruction

The AAS instruction converts the result in hexadecimal format to decimal format present in AL register after subtraction of two unpacked BCD. If lower nibble of AL register is greater then $(9)_H$ or AF is set to 1 then AL is decrement by $(6)_H$ and AH is decrement by 1. This instruction always clear upper nibble of AL register.

AAS instruction format

```
AAS             ; ASCII adjust for addition
```

E.g. Consider $(5)_D - (7)_D = -(8)_D$, but processor add it in hexadecimal format so its answer will be $(FE)_H$, to convert hexadecimal format into decimal format AAS instruction subtract 6 from $(FE)_H$ which makes $(F8)_H$ so $(08)_H$ is store in AL register and $(01)_H$ is subtracted from AH register. Note negative sign will not store so by checking AH register negative answer can determine.

```
MOV   AL, 05H       ; AL = 09H and AH = 00H
SUB   AL, 07H       ; AL = FEH and AH = 00H
AAS                 ; AL = 08H and AH = FFH (00 H – 01H = FFH)
```

```
TITLE EXAMPLE_PROG_AAA
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX
                ;----------
    MOV AH, 05H ;----------
    MOV AL, 05H ;SAVING VALUES
                ;-------------
    ADD AL, AH  ;AL = 05+05=10
    MOV AH, 00H
    AAA         ; ASCII ADJUST
    ; AL = HIGHER DIGIT = 1
    ; AH = LOWER DIGIT = 0
    MOV DL, AH  ;----------------
    MOV AH, 02H ;DISPLAYING AH
    INT 21H     ;----------------
    MOV DL, AL  ;----------------
    MOV AH, 02H ;DISPLAYING AL
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

```
TITLE EXAMPLE_PROG_AAS
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX
                ;----------
    MOV AL, 09H ;----------
    MOV AH, 06H ;SAVING VALUES
                ;-------------
    SUB AL, AH  ;AL = 09-06=03
    MOV AH, 00H
    AAS         ; ASCII ADJUST
    ; AL = HIGHER DIGIT = 3
    ; AH = LOWER DIGIT = 0
    MOV DL, AL  ;----------------
    MOV AH, 02H ;DISPLAYING AL
    INT 21H     ;----------------
    MOV AH, 4CH
    INT 21H
END MAIN
```

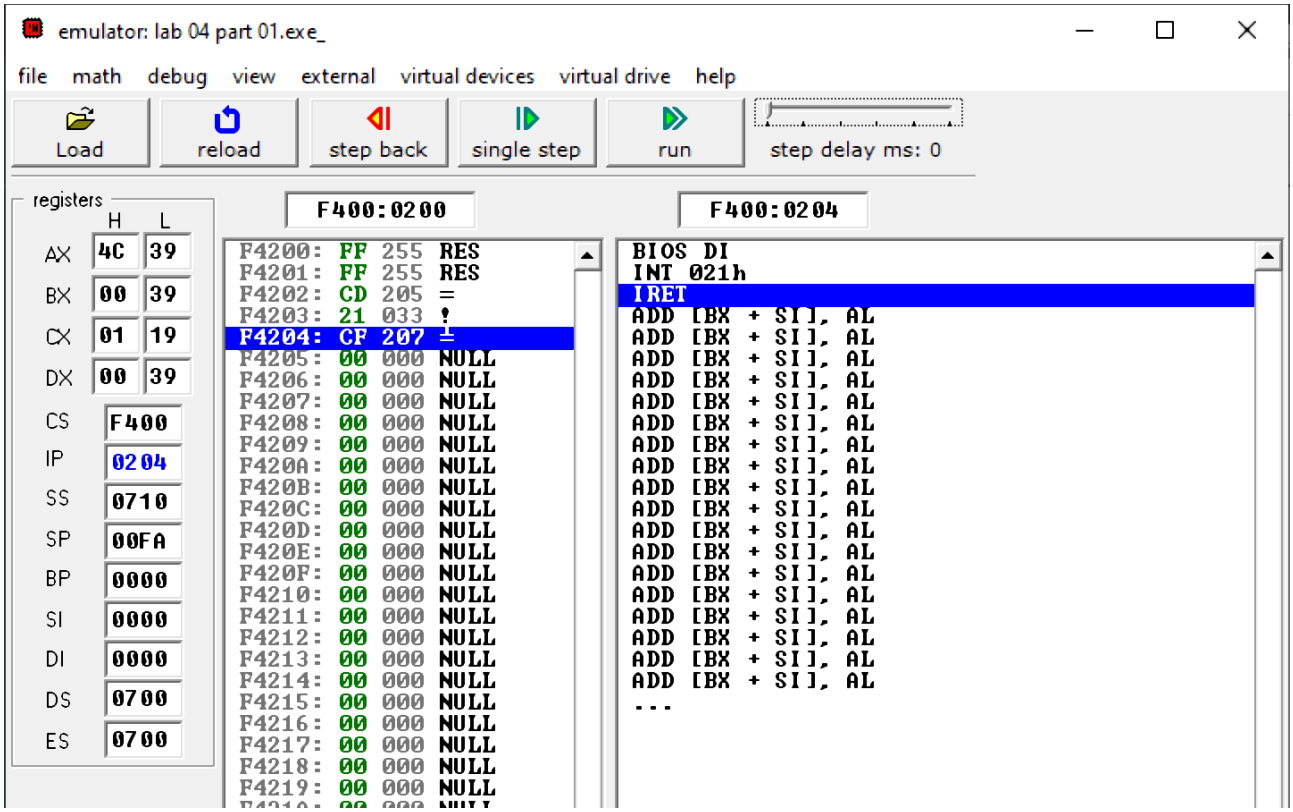
Tasks:

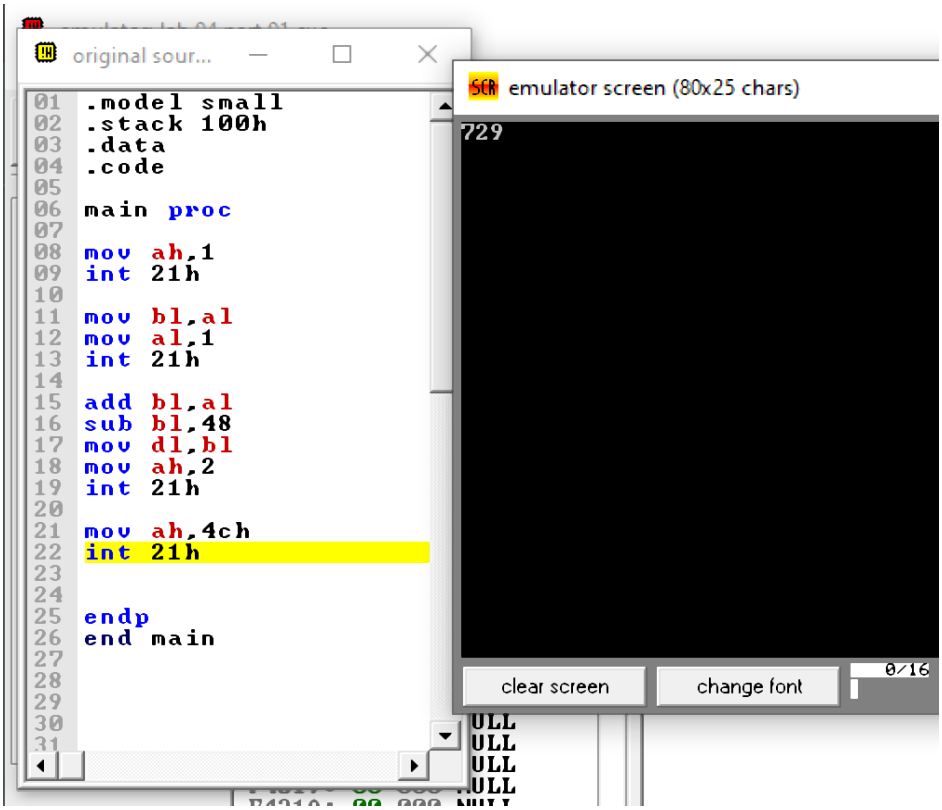
1) Write a program which asks user to enter number and apply addition.

edit bookmarks assembler emulator math ascii codes help

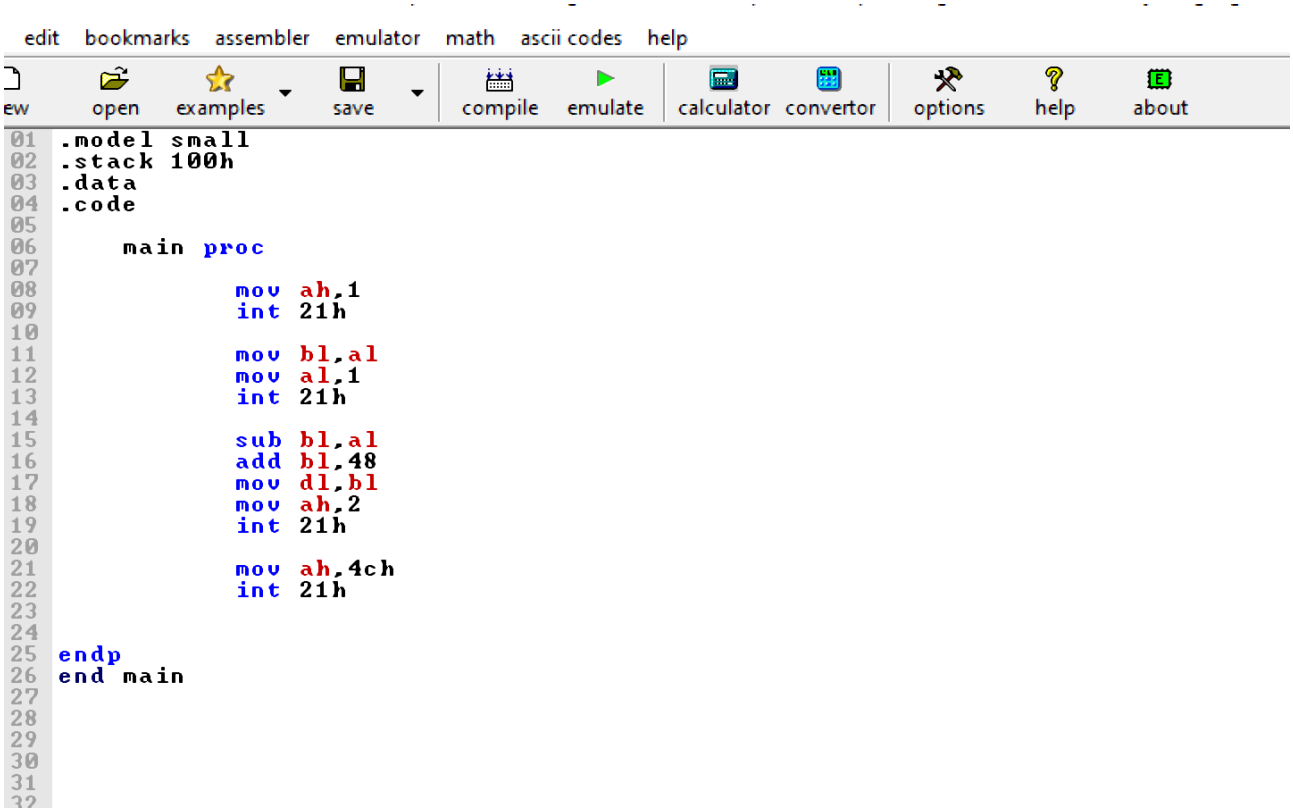
new open examples save compile emulate calculator convertor options help about

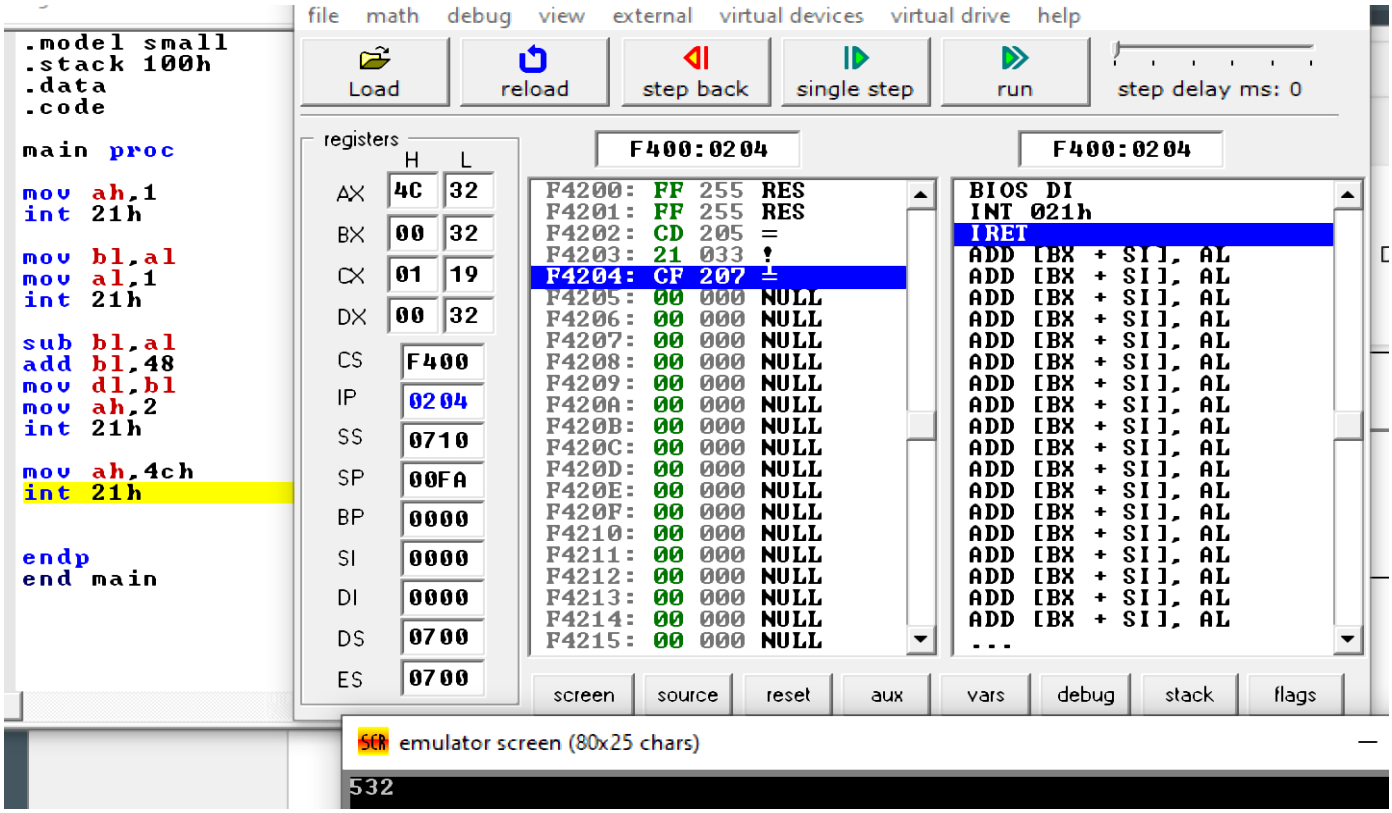
```
.model small
.stack 100h
.data
.code

    main proc

            mov ah,1
            int 21h

            mov bl,al
            mov al,1
            int 21h

            add bl,al
            sub bl,48
            mov dl,bl
            mov ah,2
            int 21h

            mov ah,4ch
            int 21h


endp
end main
```





2) Repeat Task 01 with subtraction



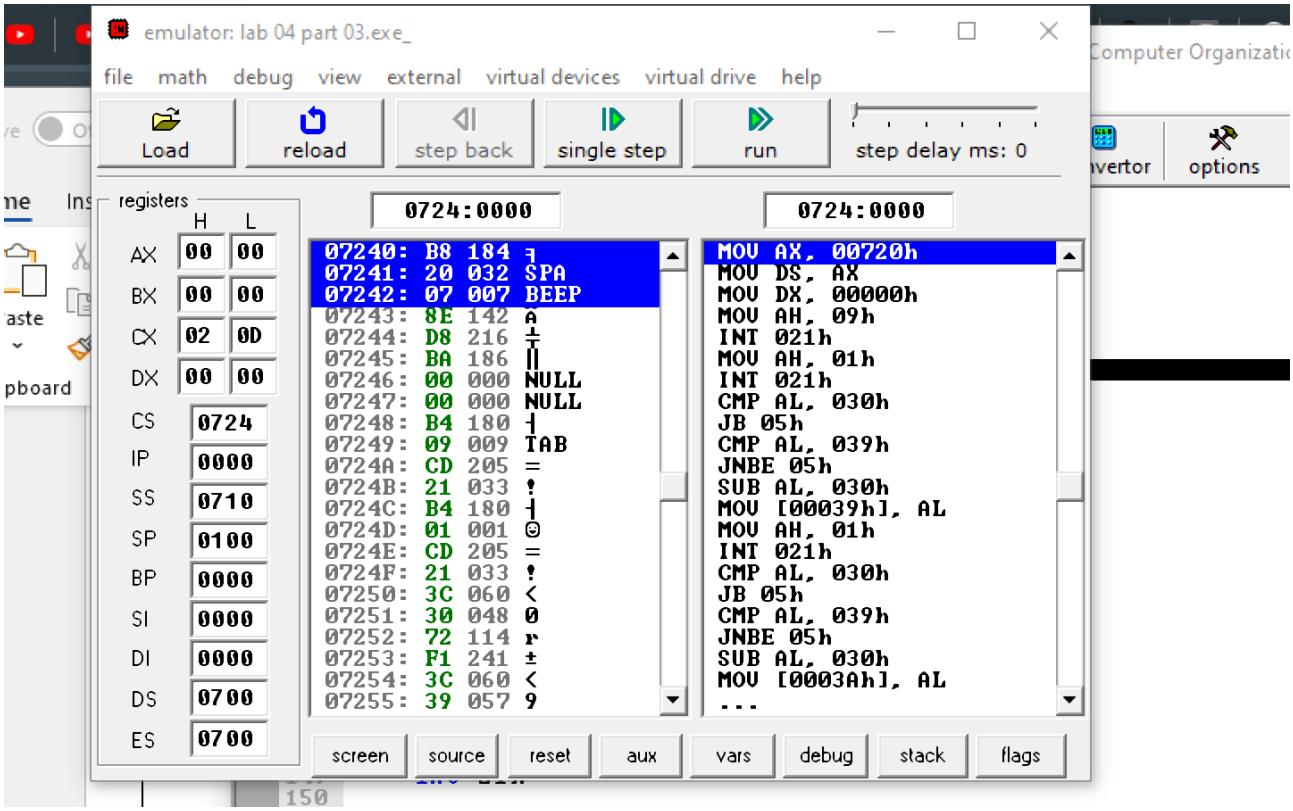


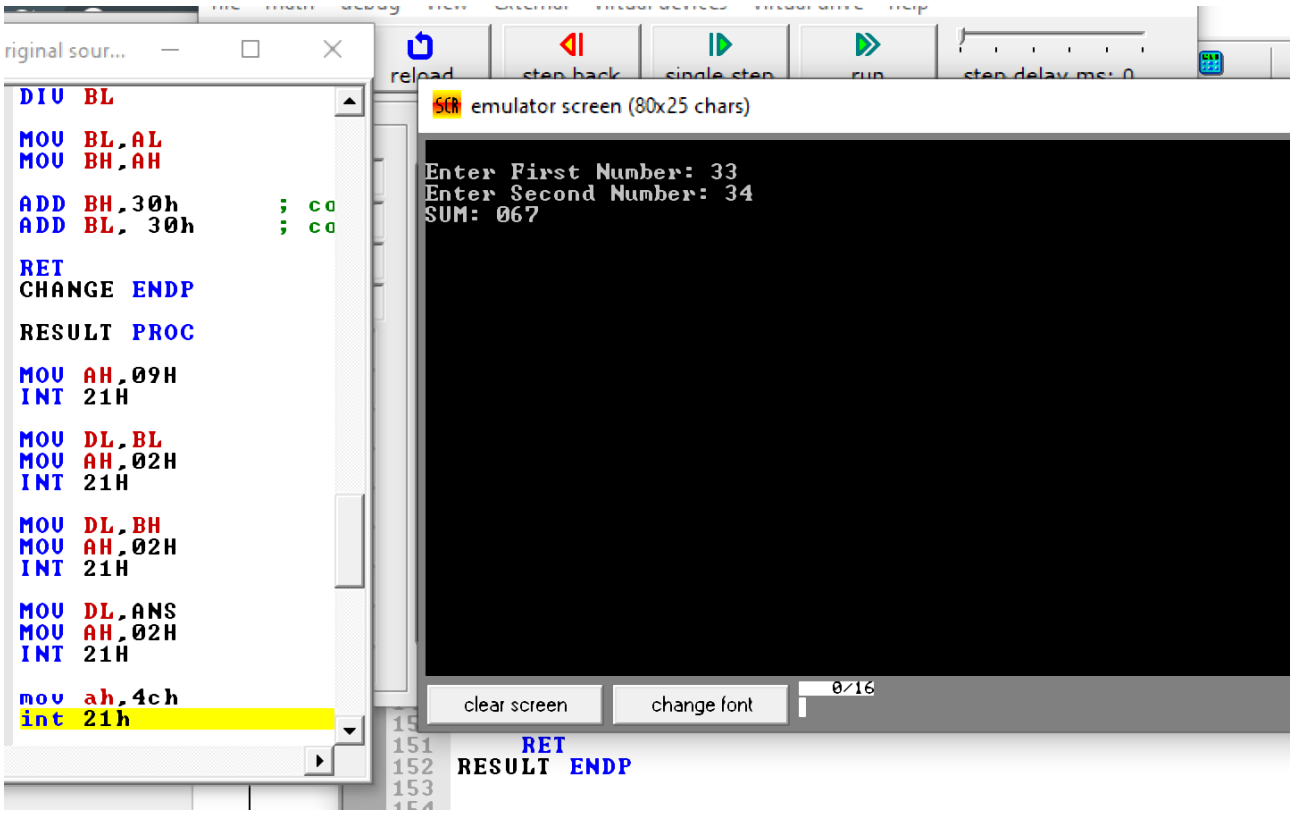
3) Write a program, in which, ask the user to enter two numbers of 2 digits each add it and display its output.

```
new  open  examples  save | compile  emulate | calculator  convertor | options  help  about
001 .MODEL SMALL
002 .STACK
003 .DATA
004     MSG1    DB  10,13,  'Enter First Number: $'
005     MSG2    DB  10,13,  'Enter Second Number: $'
006     MSG3    DB  10,13,  'SUM: $'
007
008     NUM1    DB  0
009     NUM2    DB  0
010     DIG1    DB  0
011     DIG2    DB  0
012     ANS     DB  0
013
014 .CODE
015
016 MAIN PROC
017     MOV AX,@DATA
018     MOV DS,AX
019
020 ENT1:
021     MOV DX,OFFSET MSG1  ;display prompt for first number
022     MOV AH,09H
023     int 21h
024
025     MOV AH,01H      ;input first number
026     INT 21H
027
028     CMP AL,'0'      ;check if it is in range from 0 - 9
029     JB ENT1
030     CMP AL,'9'
031     JA ENT1
032
033     SUB AL,30H      ;convert to real number entered
034     MOV DIG1,AL
035
036     MOV AH,01H      ;input first number
```

```
037      INT 21H
038
039      CMP AL,'0'        ;check if it is in range from 0 - 9
040      JB ENT1
041      CMP AL,'9'
042      JA ENT1
043
044      SUB AL,30H        ;convert to real number entered
045      MOV DIG2,AL
046
047      MOV AL,DIG1       ;convert 1st digit to tens place
048      MOV BL,10
049      MUL BL
050
051      MOV NUM1,AL       ;add 1st digit to 2nd digit
052      MOV AL,DIG2
053      ADD NUM1,AL
054
055
056  ENT2:
057      MOV DX,OFFSET MSG2  ;display prompt for second number
058      MOV AH,09H
059      int 21h
060
061      MOV AH,01H        ;input second number
062      INT 21H
063
064      CMP AL,'0'        ;check if it is in range from 0 - 9
065      JB ENT2
066      CMP AL,'9'
067      JA ENT2
068
069      SUB AL,30H        ; convert to real number entered
070      MOV DIG1,AL
071
072      MOV AH,01H        ;input second number
073      INT 21H
074
075      CMP AL,'0'        ;check if it is in range from 0 - 9
076      JB ENT2
077      CMP AL,'9'
078      JA ENT2
079
080      SUB AL,30H        ;convert to real number entered
081      MOV DIG2,AL
082
083      MOV AL,DIG1
084      MOV BL,10
085      MUL BL
086
087      MOV NUM2,AL
088      MOV AL,DIG2
089      ADD NUM2,AL
090
091
092  ADDITION:
093      MOV BL,NUM1
094      ADD BL,NUM2
095
096      CALL CHANGE
097
098      MOV DX,OFFSET MSG3
099      CALL RESULT
100
101
102  MAIN ENDP
103
104  CHANGE PROC
105      MOV AH,0
106      MOV AL,BL
107
108      MOV BL,10
109      DIV BL
110
```

```
    MOV BL,AL
    MOV BH,AH

    ADD BH,30H        ; convert to ascii code
    MOV ANS,BH

    MOV AH,0
    MOV AL,BL
    MOV BL,10
    DIV BL

    MOV BL,AL
    MOV BH,AH

    ADD BH,30h        ; convert to ascii code
    ADD BL, 30h       ; covert to ascii code

    RET
CHANGE ENDP

RESULT PROC

    MOV AH,09H
    INT 21H

    MOV DL,BL
    MOV AH,02H
    INT 21H

    MOV DL,BH
    MOV AH,02H
    INT 21H

    MOV DL,ANS
    MOV AH,02H
    INT 21H

    mov ah,4ch
    int 21h

    RET
RESULT ENDP

END
```


emulator: lab 04 part 03.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 00 00
BX 00 00
CX 02 0D
DX 00 00
CS 0724
IP 0000
SS 0710
SP 0100
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0724:0000
0724:0000
07240: B8 184 ¸
07241: 20 032 SPA
07242: 07 007 BEEP
07243: 8E 142 Ä
07244: D8 216 ╪
07245: BA 186 ║
07246: 00 000 NULL
07247: 00 000 NULL
07248: B4 180 ┤
07249: 09 009 TAB
0724A: CD 205 ═
0724B: 21 033 !
0724C: B4 180 ┤
0724D: 01 001 ☺
0724E: CD 205 ═
0724F: 21 033 !
07250: 3C 060 <
07251: 30 048 0
07252: 72 114 r
07253: F1 241 ±
07254: 3C 060 <
07255: 39 057 9
MOV AX, 00720h
MOV DS, AX
MOV DX, 00000h
MOV AH, 09h
INT 021h
MOV AH, 01h
INT 021h
CMP AL, 030h
JB 05h
CMP AL, 039h
JNBE 05h
SUB AL, 030h
MOV [00039h], AL
MOV AH, 01h
INT 021h
CMP AL, 030h
JB 05h
CMP AL, 039h
JNBE 05h
SUB AL, 030h
MOV [0003Ah], AL
...
screen
source
reset
aux
vars
debug
stack
flags


DIV BL
MOV BL,AL
MOV BH,AH
ADD BH,30h
ADD BL, 30h
RET
CHANGE ENDP
RESULT PROC
MOV AH,09H
INT 21H
MOV DL,BL
MOV AH,02H
INT 21H
MOV DL,BH
MOV AH,02H
INT 21H
MOV DL,ANS
MOV AH,02H
INT 21H
mov ah,4ch
int 21h
emulator screen (80x25 chars)
Enter First Number: 33
Enter Second Number: 34
SUM: 067
clear screen
change font
0716
RET
RESULT ENDP