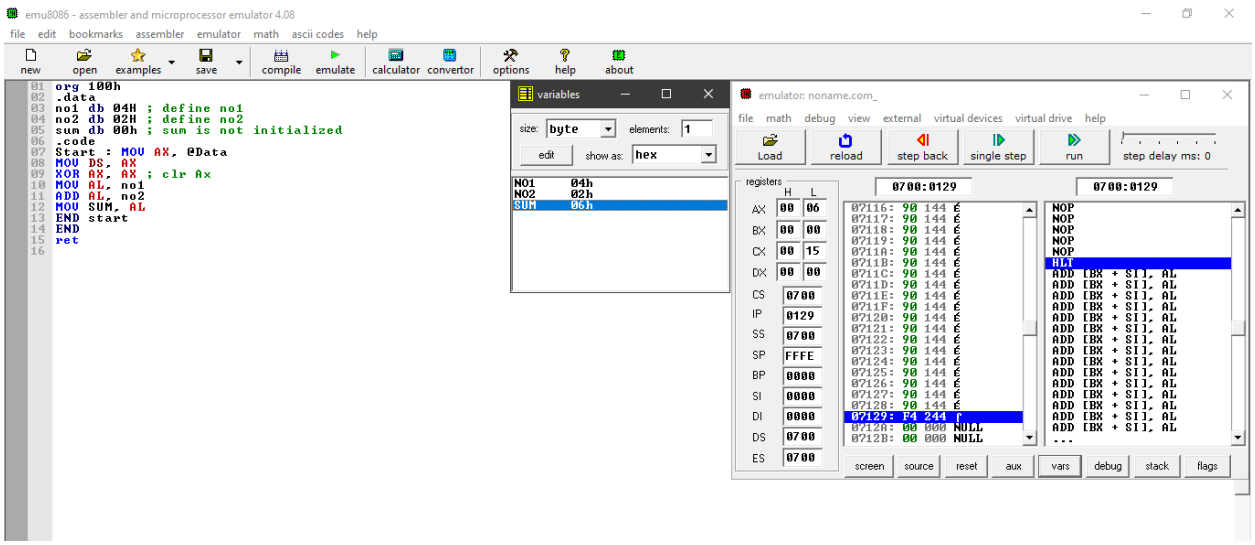
Experiment 1
Class: SE Comp
Year: 2020-21
Performed by: Danyl Fernandes, 72

Addition of two 8-bit numbers:

Code:

```
org 100h
.data
     no1 db 04H ; define no1
     no2 db 02H ; define no2
     sum db 00h ; sum is not initialized
.code
     Start : MOV AX, @Data
     MOV DS, AX
     XOR AX, AX ; clr Ax
     MOV AL, no1
     ADD AL, no2
     MOV SUM, AL
END start
END
ret
```

Output:

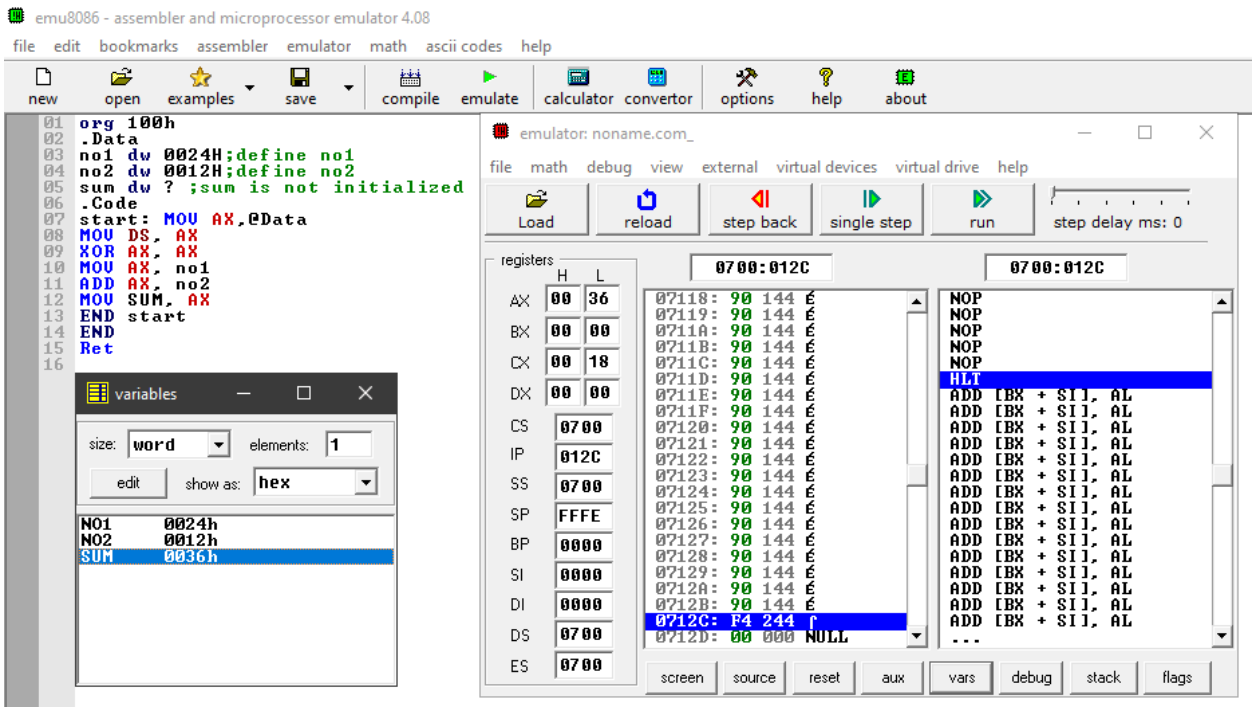


Addition of two 16-bit numbers:

Code:

```
org 100h
.Data
      no1 dw 0024H;define no1
      no2 dw 0012H;define no2
      sum dw ? ;sum is not initialized
.Code
      start: MOV AX,@Data
      MOV DS, AX
      XOR AX, AX
      MOV AX, no1
      ADD AX, no2
      MOV SUM, AX
END start
END
Ret
```

Output:

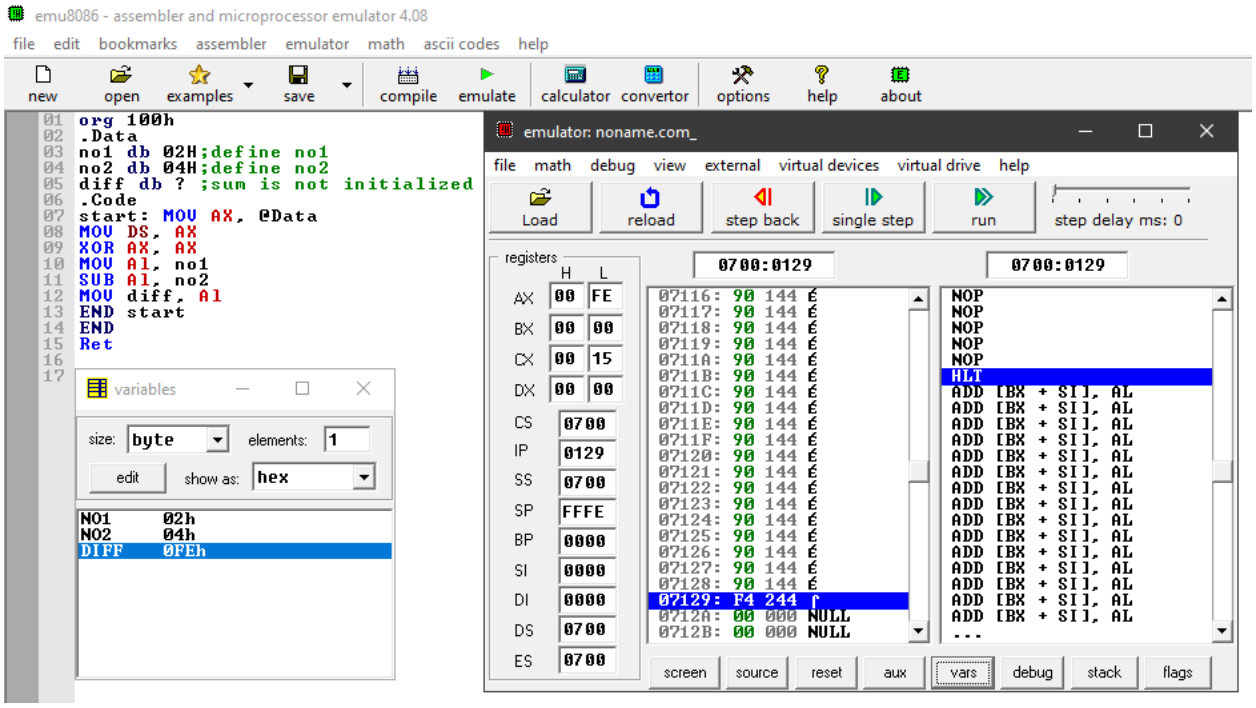


Subtraction of two 8-bit numbers:

Code:

```
org 100h
.Data
     no1 db 02H;define no1
     no2 db 04H;define no2
     diff db ? ;sum is not initialized
.Code
     start: MOV AX, @Data
     MOV DS, AX
     XOR AX, AX
     MOV Al, no1
     SUB Al, no2
     MOV diff, Al
END start
END
Ret
```

Output:

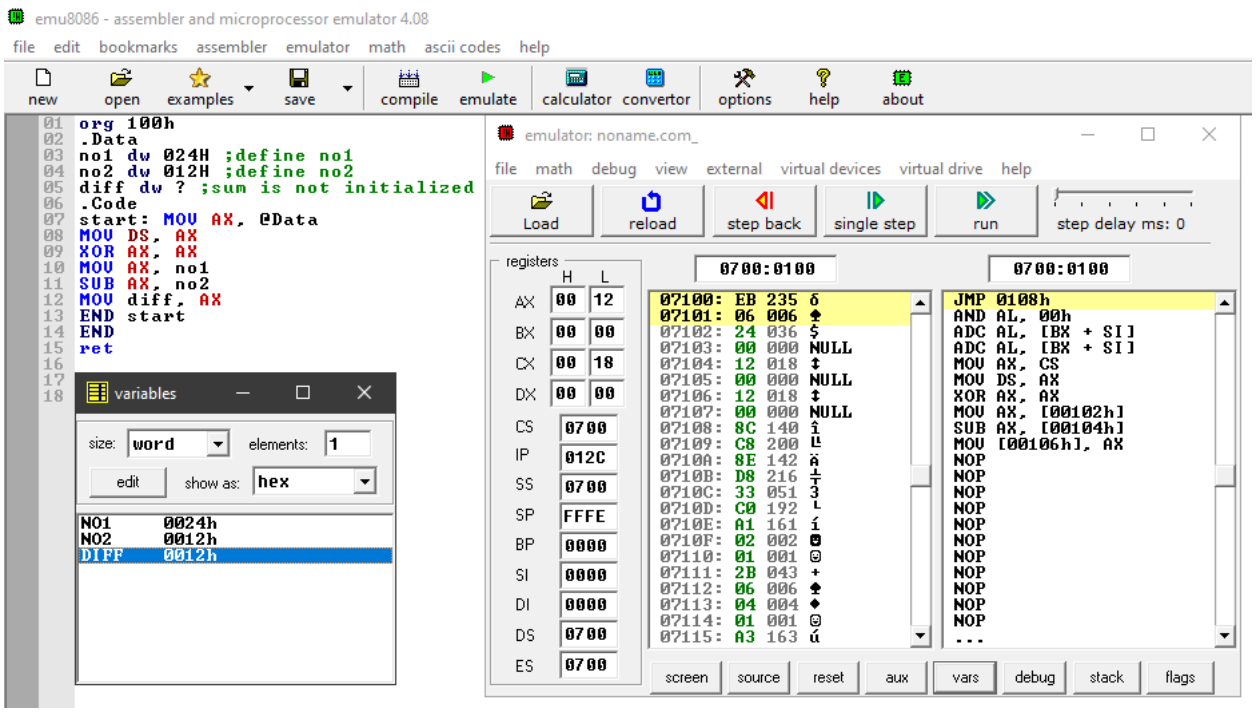


Subtraction of two 16-bit numbers:

Code:

```
org 100h
.Data
     no1 dw 024H ;define no1
     no2 dw 012H ;define no2
     diff dw ? ;sum is not initialized
.Code
     start: MOV AX, @Data
     MOV DS, AX
     XOR AX, AX
     MOV AX, no1
     SUB AX, no2
     MOV diff, AX
END start
END
ret
```

Output:

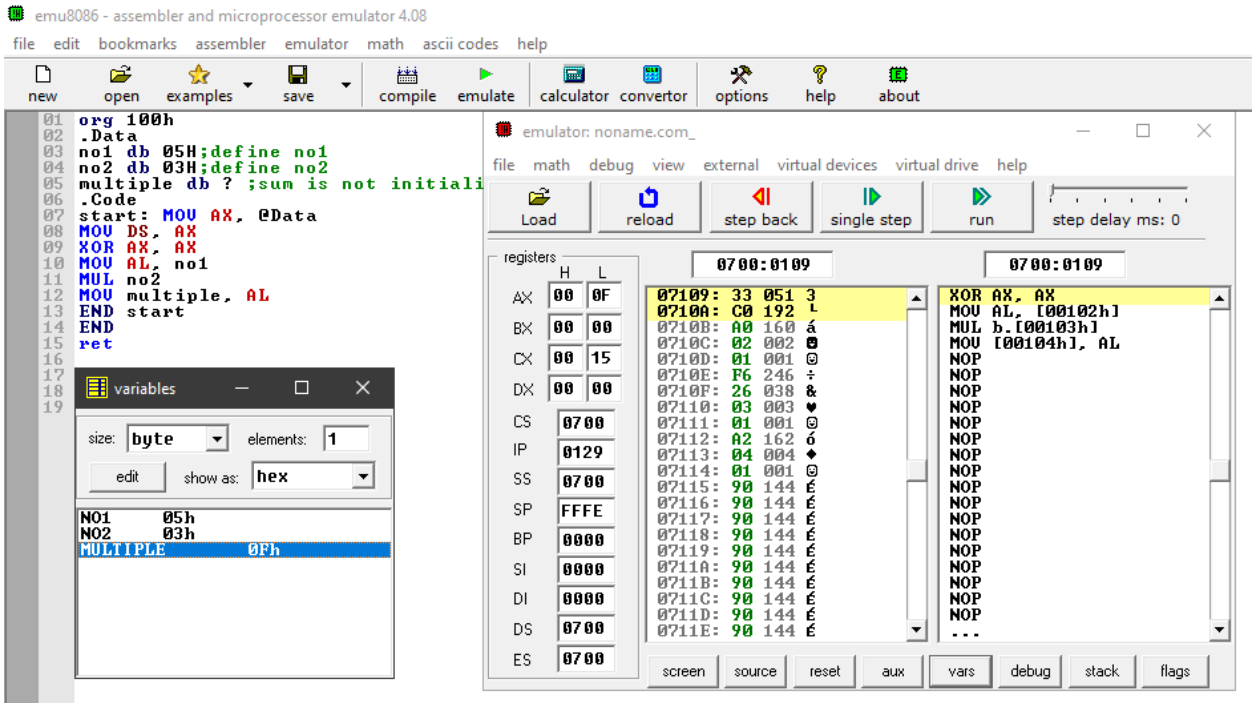


Multiplication of two 8-bit numbers:

Code:

```
org 100h
.Data
     no1 db 05H;define no1
     no2 db 03H;define no2
     multiple db ? ;sum is not initialized
.Code
     start: MOV AX, @Data
     MOV DS, AX
     XOR AX, AX
     MOV AL, no1
     MUL no2
     MOV multiple, AL
END start
END
ret
```

Output:

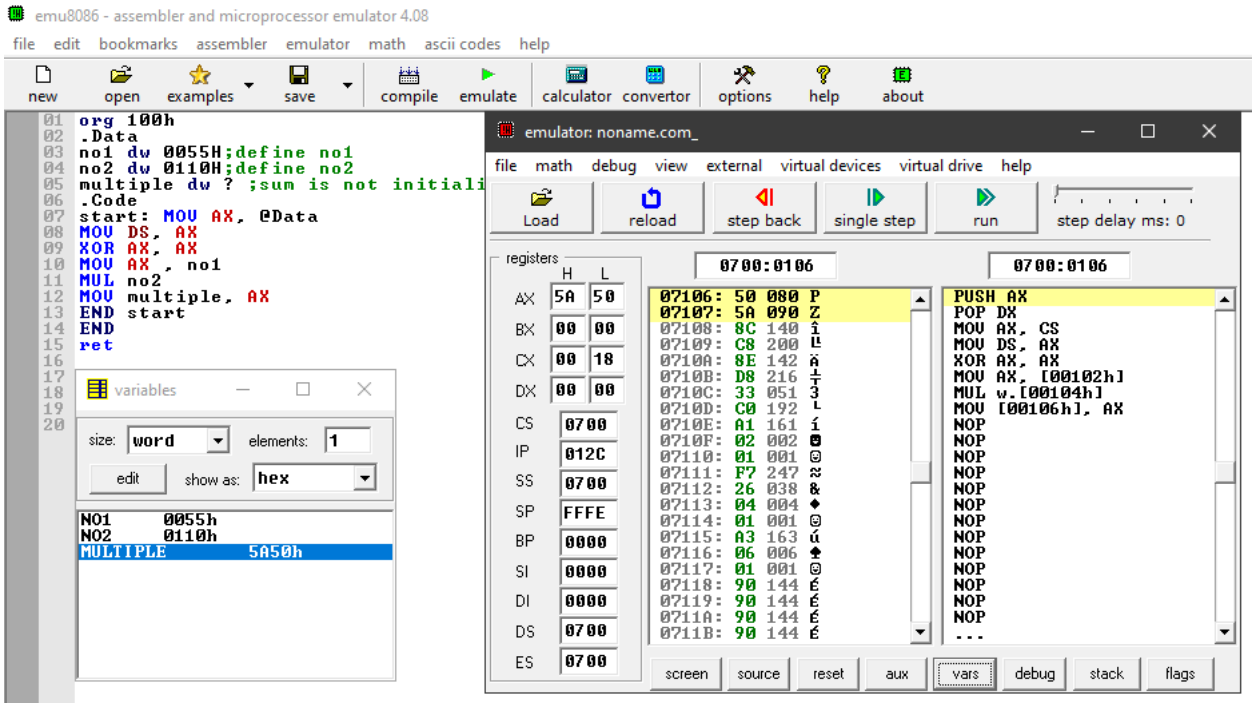


Multiplication of two 16-bit numbers:

Code:

```
org 100h
.Data
     no1 dw 0055H;define no1
     no2 dw 0110H;define no2
     multiple dw ? ;sum is not initialized
.Code
     start: MOV AX, @Data
     MOV DS, AX
     XOR AX, AX
     MOV AX , no1
     MUL no2
     MOV multiple, AX
END start
END
ret
```

Output:

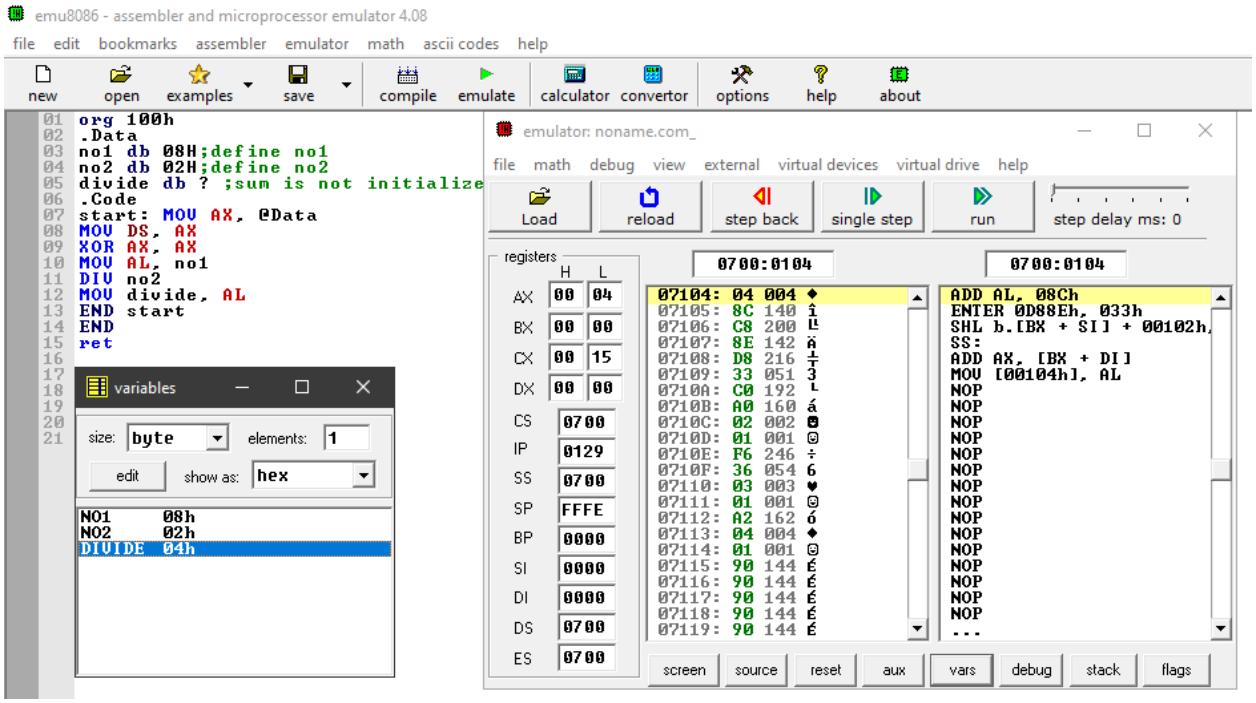


Division of two 8-bit numbers:

Code:

```
org 100h
.Data
     no1 db 08H;define no1
     no2 db 02H;define no2
     divide db ? ;sum is not initialized
.Code
     start: MOV AX, @Data
     MOV DS, AX
     XOR AX, AX
     MOV AL, no1
     DIV no2
     MOV divide, AL
END start
END
ret
```

Output:

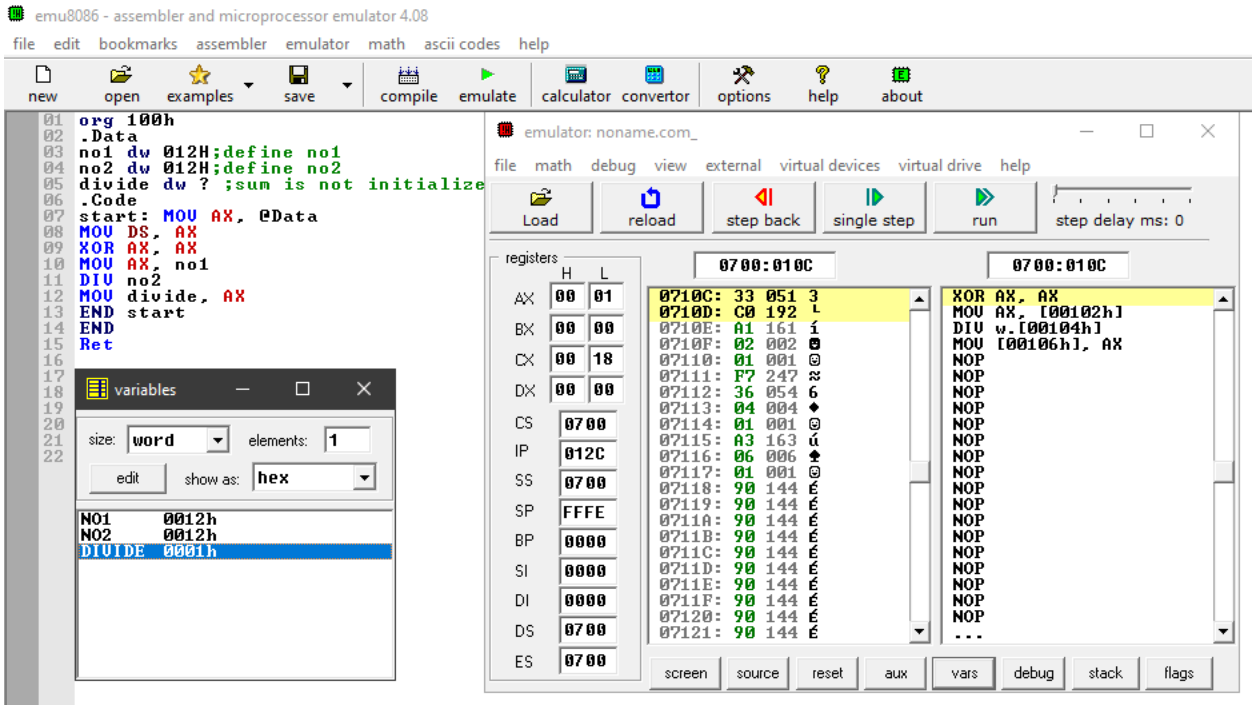


Division of two 16-bit numbers:

Code:

```
org 100h
.Data
     no1 dw 012H;define no1
     no2 dw 012H;define no2
     divide dw ? ;sum is not initialized
.Code
     start: MOV AX, @Data
     MOV DS, AX
     XOR AX, AX
     MOV AX, no1
     DIV no2
     MOV divide, AX
END start
END
Ret
```

Output:



Conclusion:

We successfully implemented 8 and 16 bit addition, subtraction, multiplication and division using assembly language programs

Exp 01

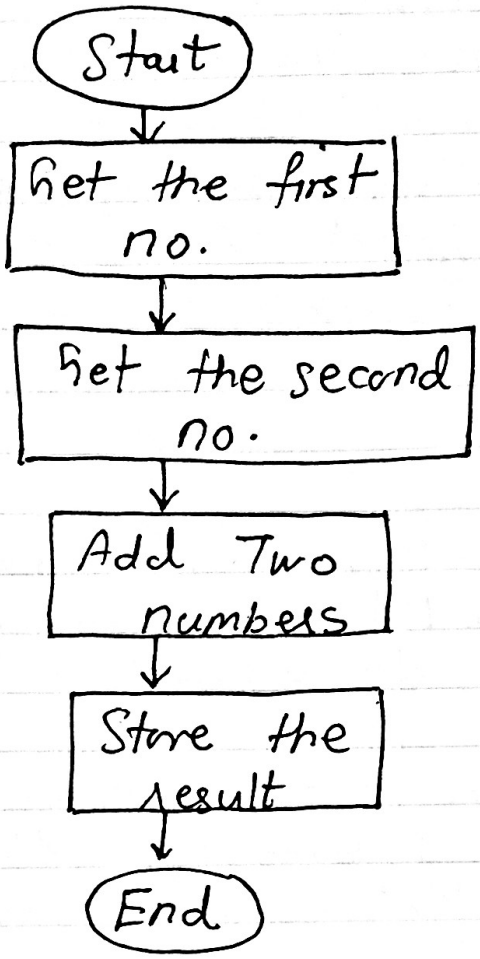
a) 8-bit addition & 16-bit addition

Aim: To write an assembly language program to perform 8-bit & 16-bit addition, substration multiplication & division

Algorithm:

- In the data segment the value of both the 8 & 16 bit numbers are given
- For 8-bit db is used & for 16-bit dw is used
- Assign terms for the answers
- The code is written is the code segment
- For 8-bit 'AL' & 'BL' is used & for 16-bit 'AX' & 'BX' is used
- The value of both the number is moved in the respective register
- The BX is added to AX & answer is stored in the AX or BX then moved to the term assign for the answer
- This value of AX is then removed to term assign for the answer
- Execute the program

Flow Chart :



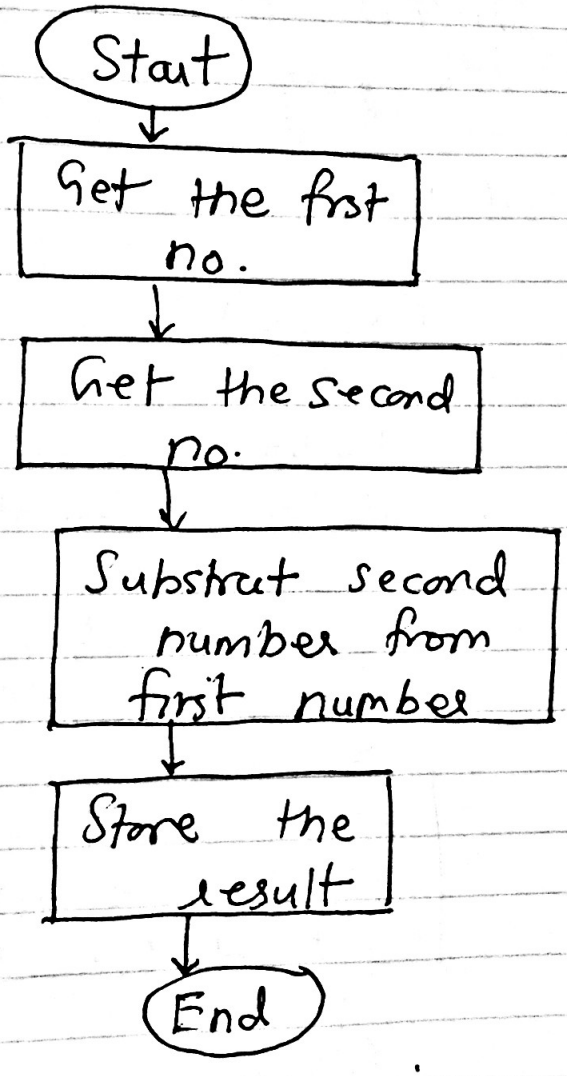
b) 8 bit subtraction & 16 bit subtraction

Algorithm :

- In the data segment the value of both the 8 & 16 bit numbers are given
- For 8 bit dB is used & for 16 bit dW is used.
- The code is written in the code segment
- For 8 bit 'AL' & 'BL' is used and for 16-bit 'AX' & 'BX' is used.
- The value of both the number is moved in the respective register
- The BX is added to AX & answer

is stored is the AX or BX then moved to the term assign for the answer
- This value of AX is then removed to term assign for the answer
- Execute the program.

Flow Chart:



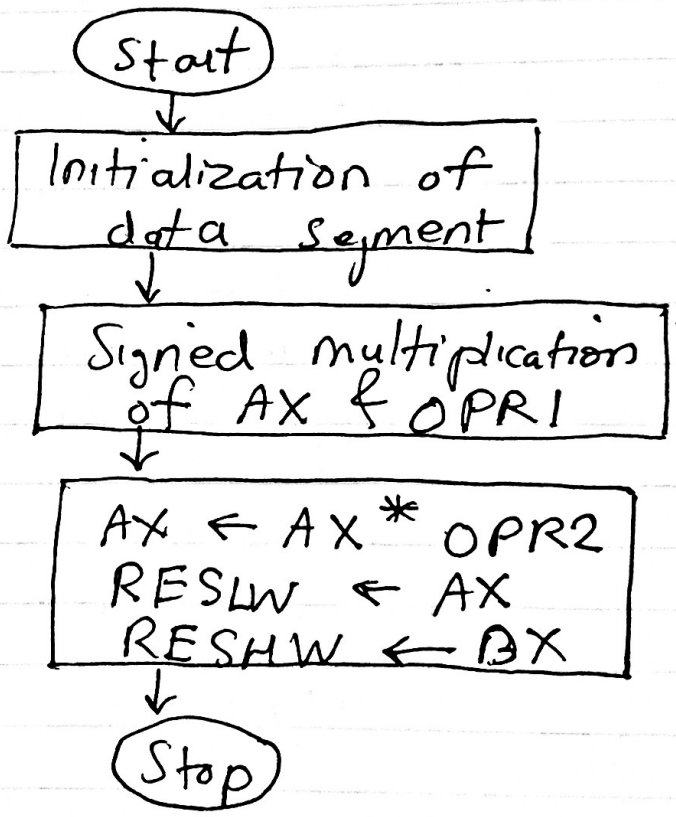
c) 8 bit mul & 16- bit multiplication

Algorithm:

- In the data segment the value of both the 8 & 16 bit numbers are given
- Assign terms of the answers

- The code is written in the code segment values of first & second number are inseted in AX & BX
- Now in multiplication we only write BL/BX in the code which implies we are multiplying BL/BX with AL/AX
- The answer is stored in AX which is then shifted to the term assigned
- Execute the program.

Flow chart:



d) 8-bit ~~multiplication~~ dinsion & 16-bit dinsion

- In the data segment the value of both the 8 & 16 bit numbers are given
- Assign terms for the the answers
- The code is written in the code

segment values of frist & second number are inserted in AX & BX
- Now in multiplication we only write BL / BX in the code which implies we are multiplying BL / BX with AL / AX
- The answer is stored in AX which is then shifted the term assigned
- Execute the program

FlowChart :