

Experiment 1

Class: SE Comp

Year: 2020-21

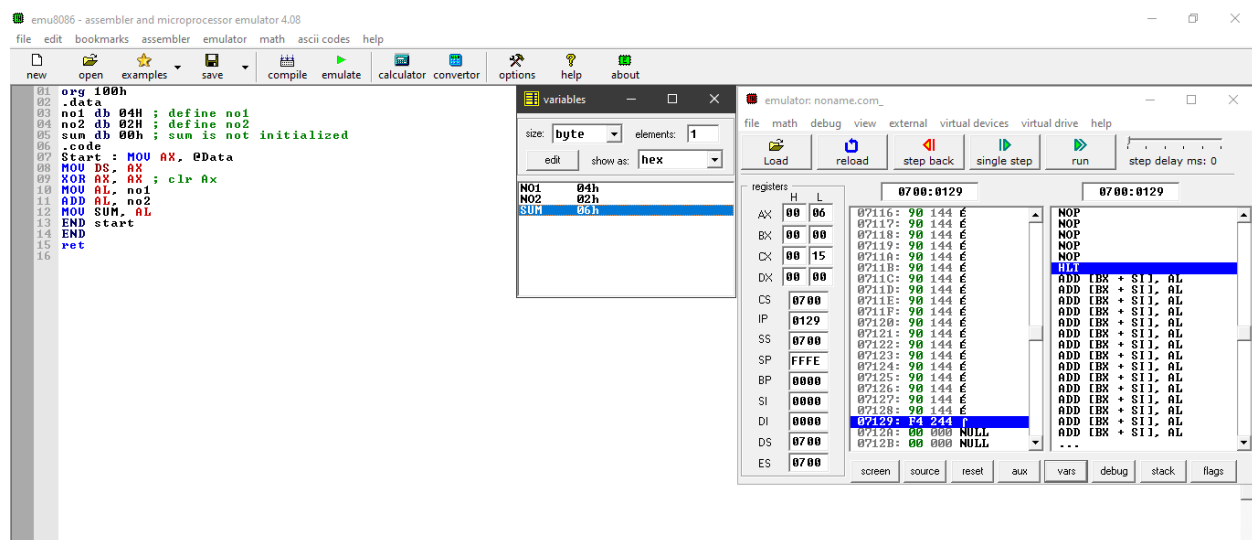
Performed by: Danyl Fernandes, 72

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:

emu8086 - assembler and microprocessor emulator 4.08

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

emu8086: noname.com_

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	36
BX	00	00
CX	00	18
DX	00	00
CS	0700	
IP	012C	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:012C

07118:	90	144	E
07119:	90	144	E
0711A:	90	144	E
0711B:	90	144	E
0711C:	90	144	E
0711D:	90	144	E
0711E:	90	144	E
0711F:	90	144	E
07120:	90	144	E
07121:	90	144	E
07122:	90	144	E
07123:	90	144	E
07124:	90	144	E
07125:	90	144	E
07126:	90	144	E
07127:	90	144	E
07128:	90	144	E
07129:	90	144	E
0712A:	90	144	E
0712B:	90	144	E
0712C:	E4	244	J
0712D:	00	000	NULL

0700:012C

NOP
NOP
NOP
NOP
NOP
HIT
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], 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], AL
ADD [BX + SI], AL
...

variables

size: word elements: 1

edit show as: hex

NO1	0024h
NO2	0012h
SUM	0036h

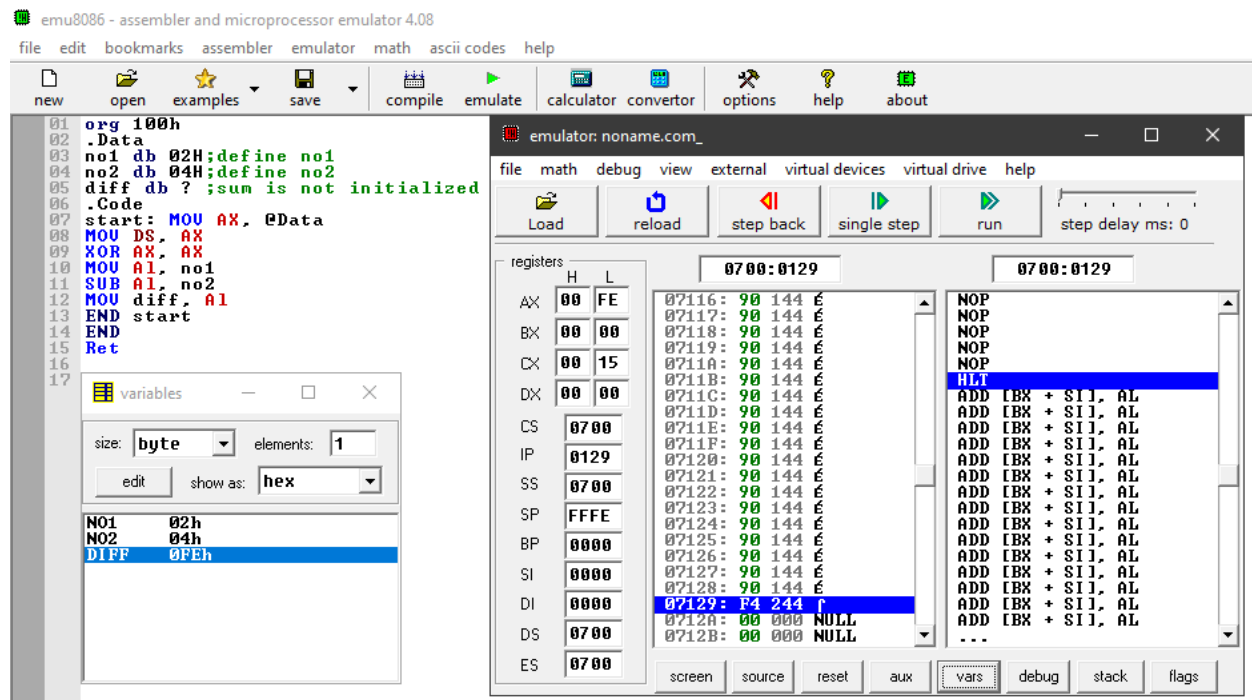
screen source reset aux vars debug stack flags

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:

emu8086 - assembler and microprocessor emulator 4.08

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

emu8086: noname.com_

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	12
BX	00	00
CX	00	18
DX	00	00
CS	0700	
IP	012C	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100 0700:0100

Address	Hex	ASCII	Comment
07100	EB	235	6
07101	06	006	+
07102	24	036	\$
07103	00	000	NULL
07104	12	018	+
07105	00	000	NULL
07106	12	018	+
07107	00	000	NULL
07108	8C	140	i
07109	C8	200	u
0710A	8E	142	ä
0710B	D8	216	±
0710C	33	051	3
0710D	C0	192	i
0710E	A1	161	i
0710F	02	002	0
07110	01	001	0
07111	2B	043	+
07112	06	006	+
07113	04	004	+
07114	01	001	0
07115	A3	163	ü

JMP 0108h

AND AL, 00h

ADC AL, [BX + SI]

ADC AL, [BX + SI]

MOV AX, CS

MOV DS, AX

XOR AX, AX

MOV AX, [00102h]

SUB AX, [00104h]

MOV [00106h], AX

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

...

screen source reset aux vars debug stack flags

variables

size: word elements: 1

edit show as: hex

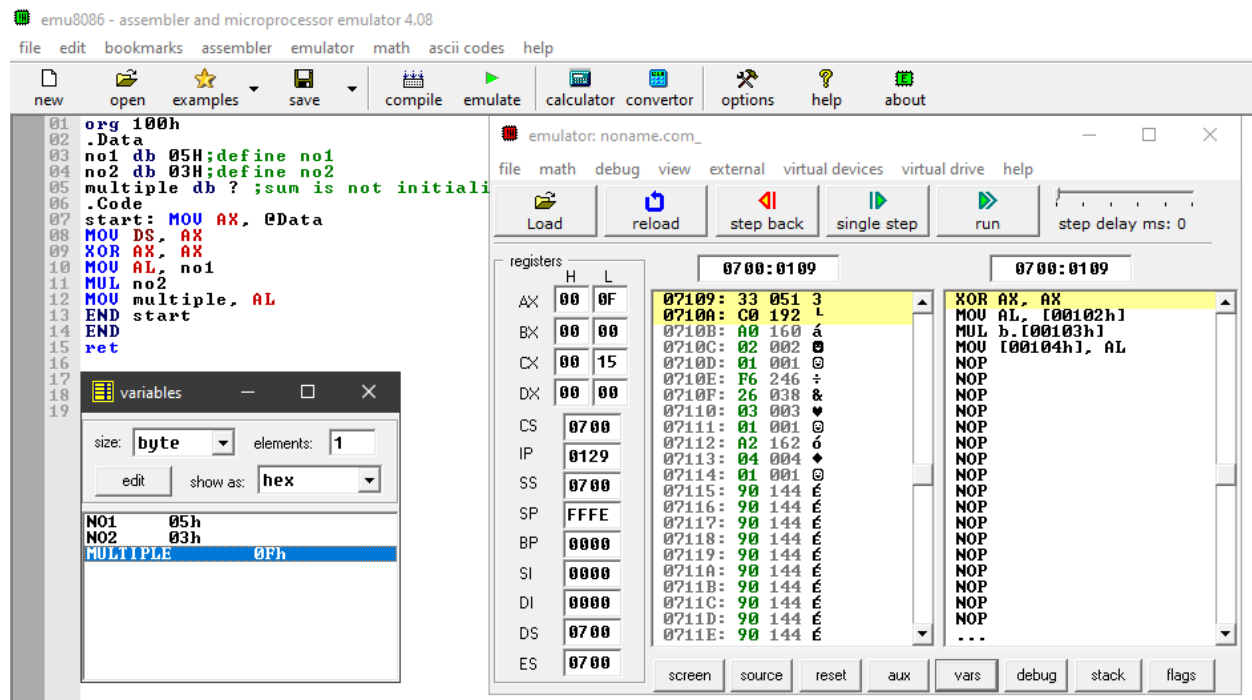
Variable	Value
NO1	0024h
NO2	0012h
DIFF	0012h

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:

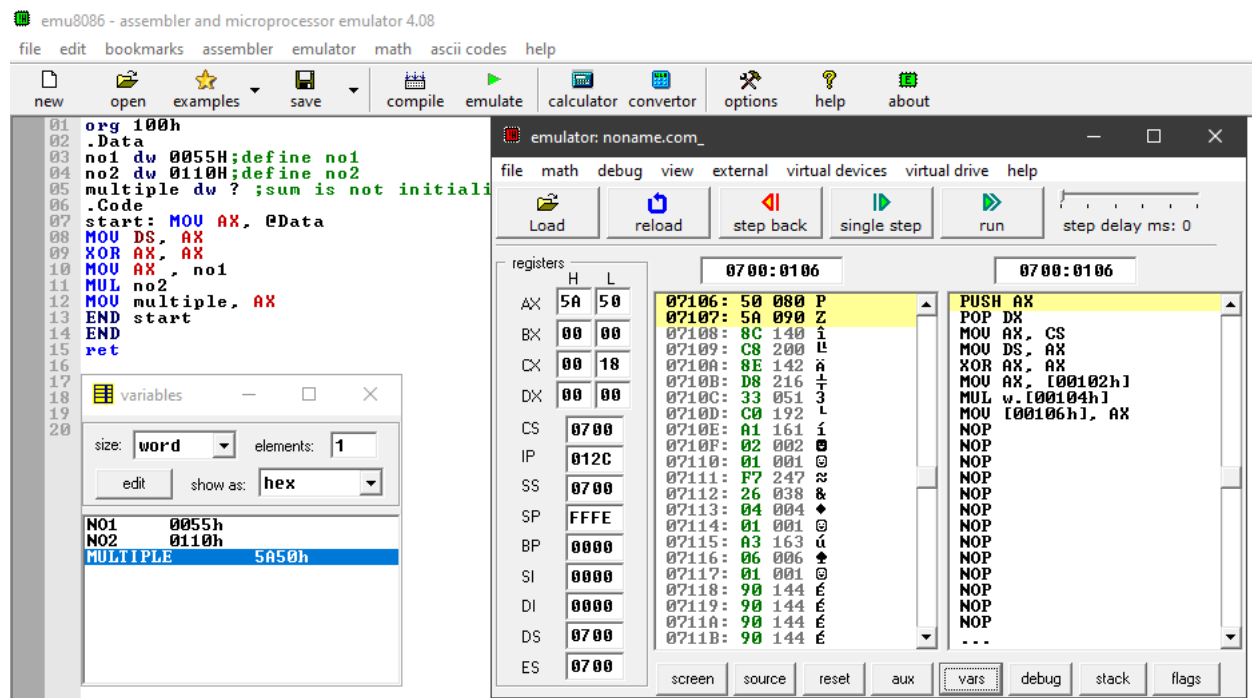


Multiplication of two 16-bit numbers:

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:

emu8086 - assembler and microprocessor emulator 4.08

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

01 org 100h
02 .Data
03 no1 db 08H;define no1
04 no2 db 02H;define no2
05 divide db ? ;sum is not initialize
06 .Code
07 start: MOV AX, @Data
08 MOV DS, AX
09 XOR AX, AX
10 MOV AL, no1
11 DIV no2
12 MOV divide, AL
13 END start
14 END
15 ret

variables

size: byte elements: 1
edit show as: hex

NO1	08h
NO2	02h
DIVIDE	04h

emulator: noname.com

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	04
BX	00	00
CX	00	15
DX	00	00
CS	0700	
IP	0129	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0104

07104: 04 004	ADD AL, 08Ch
07105: 8C 140	ENTER 0D88Eh, 033h
07106: C8 200	SHL b.[BX + SI] + 00102h,
07107: 8E 142	SS:
07108: D8 216	ADD AX, [BX + DI]
07109: 33 051	MOV [00104h], AL
0710A: C0 192	NOP
0710B: A0 160	NOP
0710C: 02 002	NOP
0710D: 01 001	NOP
0710E: F6 246	NOP
0710F: 36 054	NOP
07110: 03 003	NOP
07111: 01 001	NOP
07112: A2 162	NOP
07113: 04 004	NOP
07114: 01 001	NOP
07115: 70 144	NOP
07116: 70 144	NOP
07117: 70 144	NOP
07118: 70 144	NOP
07119: 70 144	NOP
	...

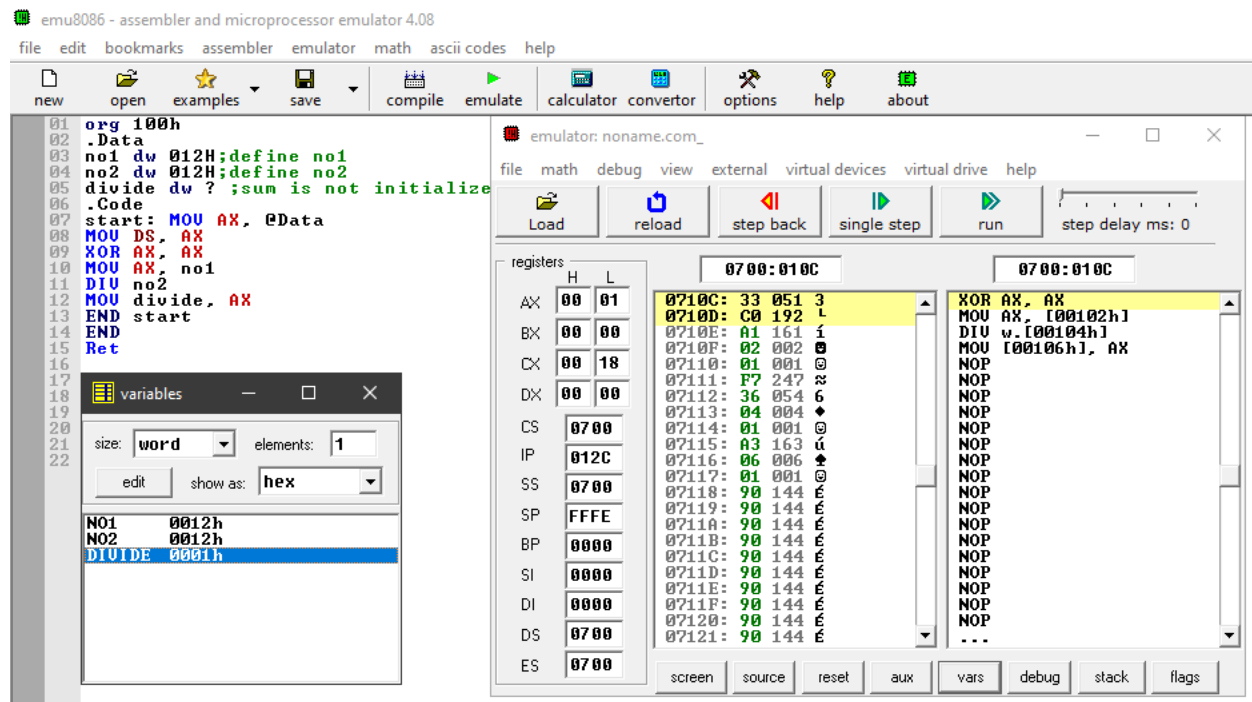
screen source reset aux vars debug stack flags

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

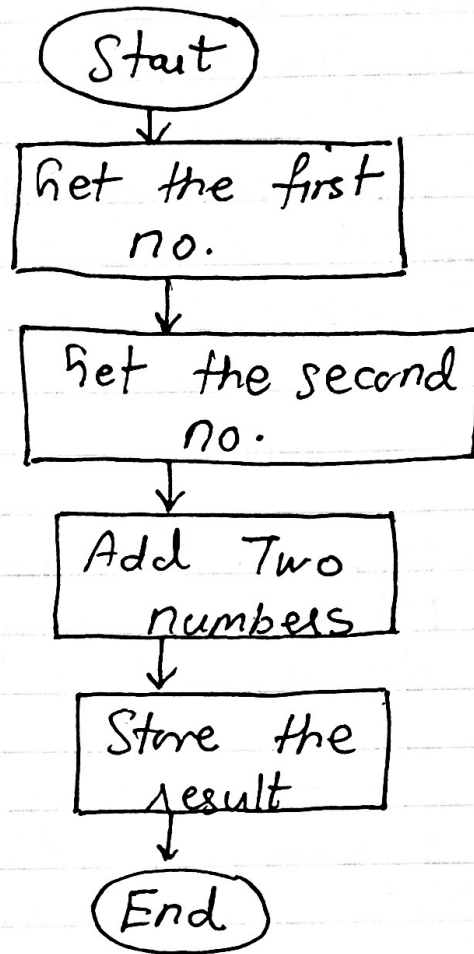
a) 8-bit addition & 16-bit addition

Aim: To write an assembly language program to perform 8-bit & 16-bit addition, subtraction, 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 in the code segment
- For 8-bit 'AL' & 'BL' is used & for 16-bit 'AX' & 'BX' is used
- The value of both the numbers is moved in the respective registers
- The BX is added to AX & answer is stored in the AX or BX then moved to the term assigned for the answer.
- This value of AX is then removed to term assigned for the answer
- Execute the program

Flow chart :



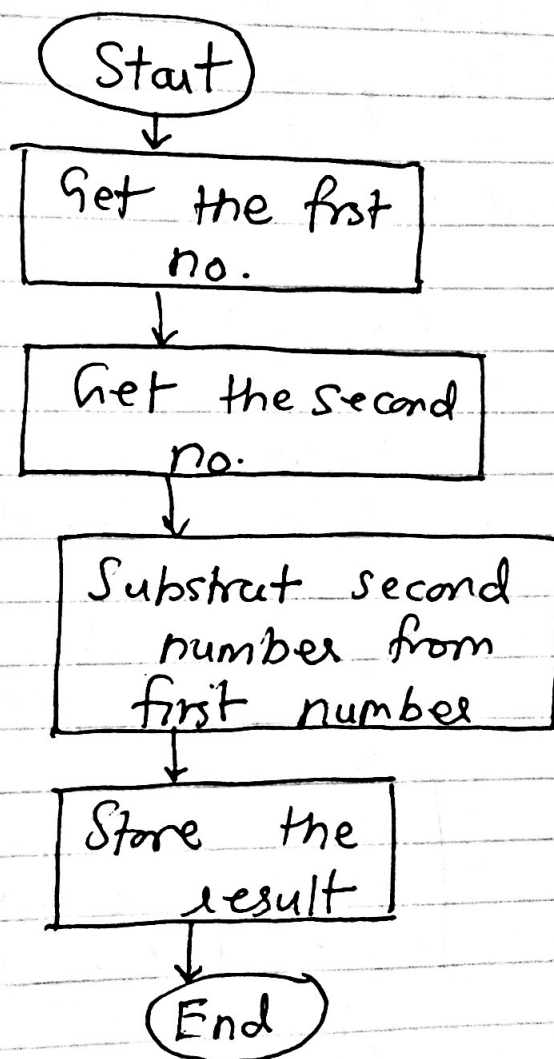
b) 8 bit subtraction & 16 bit subtraction

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 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:



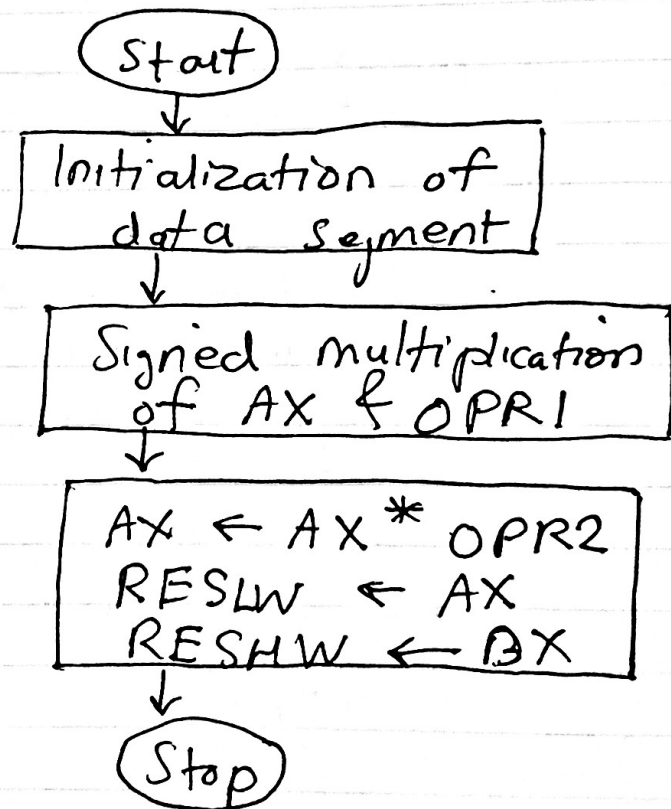
c) 8 bit mul & 16-bit multiplication

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 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 to the term assigned
- Execute the program.

Flow chart :



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

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

- segment values of first & 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 :

