DIGITAL ASSIGNMENT 2 CALCULATOR(ASSEMBLY LANGUAGE)

MICROPROCESSOR AND INTERFACING(L51-L52)[MRS SHOBHA REKH]

SEPTEMBER 14, 2022

ANIRUDH VADERA
20BCE2940

ANIRUDH VADERA
DIGITAL ASSIGNMENT 2 CALCULATOR(ASSEMBLY LANGUAGE)

QUESTION:

The program should be simulated and verified using DOS BOX/EMU 86 before submission

1) Implement a calculator using 8086

The program should have 8 options (password) to perform 8-bit addition, 8-bit subtraction, 16-bit addition, 16-bit subtraction, 8 & 16-bit multiplication and 16/8 division and 32/16 division

2) The submission should be a pdf file consisting of

a. The algorithm/ flowchart implemented in the code

b. The Inputs/outputs given for testing along with their memory location

c. The screen shots of the code / code written in any editor

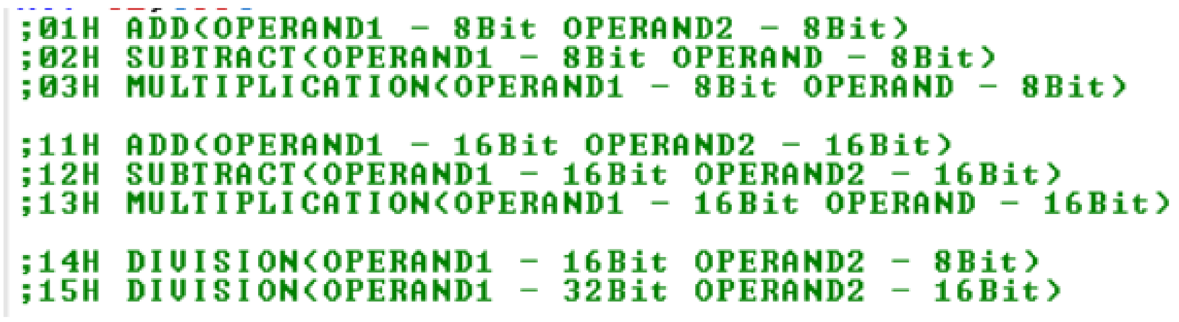
ALGORITHM:

- First we take the operation type from the required location
- Then we decide the type of input needed for the required action
- We take the input dependent on the operation we need to perform
- We then one by one check the value stored for operation type then jump to the particular label to perform the particular operation
- We then Store the result in the required format

We only perform basic operations which are performed by instructions ADD, SUB, MUL, DIV

In Assembly Language (Flowchart Explanation):

1. Start
2. We first take the operation type from the location 2500
3. Following are the meaning of values:

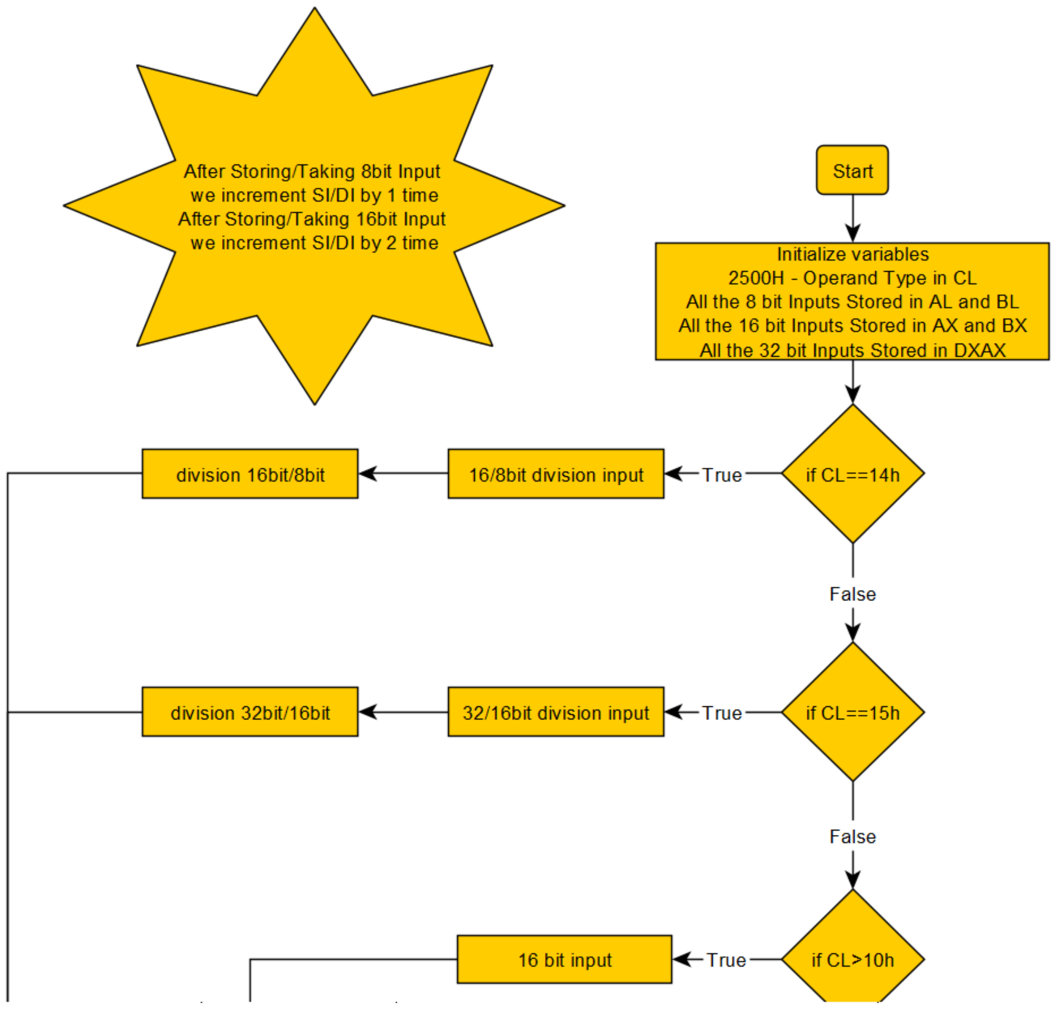


```
;01H ADD(OPERAND1 - 8Bit OPERAND2 - 8Bit)
;02H SUBTRACT(OPERAND1 - 8Bit OPERAND - 8Bit)
;03H MULTIPLICATION(OPERAND1 - 8Bit OPERAND - 8Bit)

;11H ADD(OPERAND1 - 16Bit OPERAND2 - 16Bit)
;12H SUBTRACT(OPERAND1 - 16Bit OPERAND2 - 16Bit)
;13H MULTIPLICATION(OPERAND1 - 16Bit OPERAND - 16Bit)

;14H DIVISION(OPERAND1 - 16Bit OPERAND2 - 8Bit)
;15H DIVISION(OPERAND1 - 32Bit OPERAND2 - 16Bit)
```

4. We then take the inputs from the location 2000
5. We then compare the value of CL with 10H

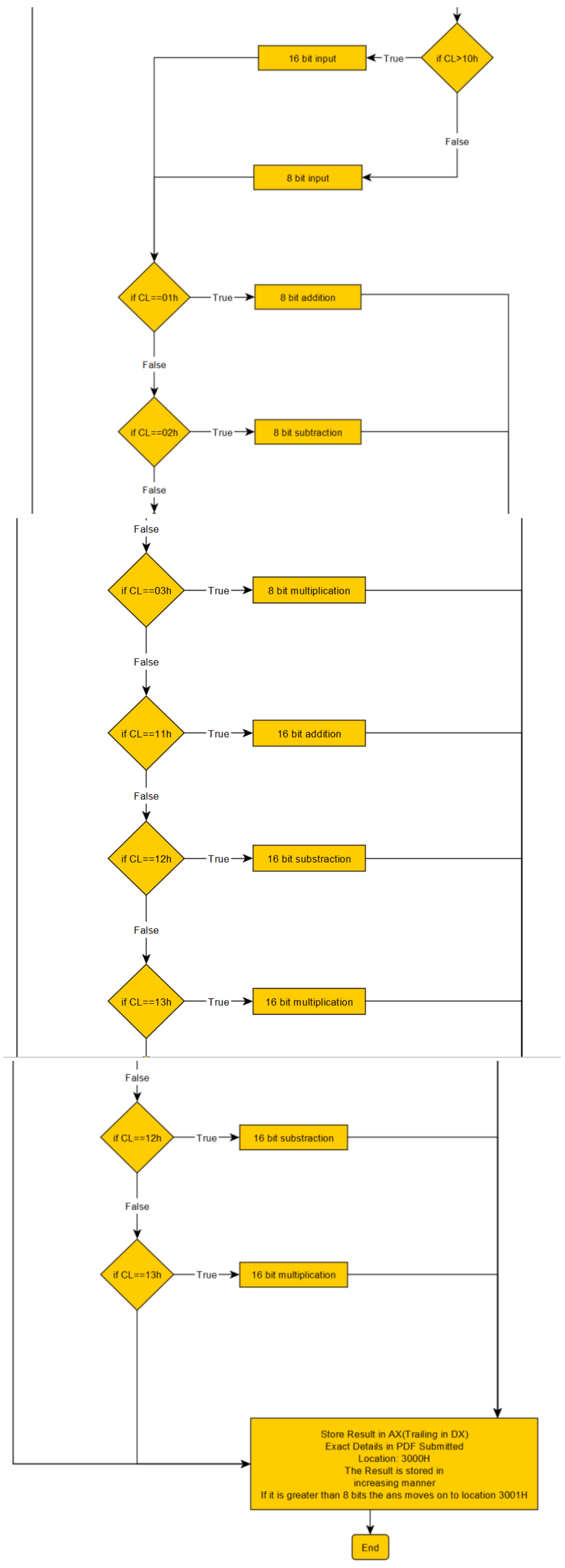
6. If its greater than 10H Then we need to take 16 bit inputs otherwise we need to take 8 bit inputs
7. We take 8 bit inputs in AL,BL
8. We take 16 bit inputs in AX,BX
9. We take 32 bit inputs in DX,AX
10. After taking inputs We check the value of CL in order to know the operation we need to perform
11. We then jump to the particular label accordingly for which we need to perform the operation
12. After jumping to the particular label we perform the operation
13. We then Store the result starting from 3000H
14. The Result is stored from AL or AX
15. In Case of Multiplication of 16 bit we store the result in DXAX starting from 3000
16. In Case of Division of 16bit by 8bit quotient is stored in AL and remainder in AH
17. In Case of Division of 32bit by 16bit quotient is stored in AX and remainder in DX
18. End of Program

FLOWCHART:


After Storing/Taking 8bit Input we increment SI/DI by 1 time After Storing/Taking 16bit Input we increment SI/DI by 2 time
Start
Initialize variables
2500H - Operand Type in CL
All the 8 bit Inputs Stored in AL and BL
All the 16 bit Inputs Stored in AX and BX
All the 32 bit Inputs Stored in DXAX
if CL==14h
True
16/8bit division input
division 16bit/8bit
False
if CL==15h
True
32/16bit division input
division 32bit/16bit
False
if CL>10h
True
16 bit input

ANIRUDH VADERA

DIGITAL ASSIGNMENT 2 CALCULATOR(ASSEMBLY LANGUAGE)



INPUT FORMAT:

First We take which Operation to perform from data location 2500H

We store which operation to perform in CL

Starting Input Location is 2000H

Details:

01H 8 Bit ADD(OPERAND1 - 8Bit OPERAND2 - 8Bit)

02H 8 Bit SUBTRACT(OPERAND1 - 8Bit OPERAND - 8Bit)

03H 8 Bit MULTIPLICATION(OPERAND1 - 8Bit OPERAND - 8Bit)

11H 16 Bit ADD(OPERAND1 - 16Bit OPERAND2 - 16Bit)

12H 16 Bit SUBTRACT(OPERAND1 - 16Bit OPERAND2 - 16Bit)

13H 16 Bit MULTIPLICATION(OPERAND1 - 16Bit OPERAND - 16Bit)

14H 16 Bit/8 Bit DIVISION(OPERAND1 - 16Bit OPERAND2 - 8Bit)

15H 32 Bit/16 Bit DIVISION(OPERAND1 - 32Bit OPERAND2 - 16Bit)

ADDITION 8 BIT:

First Operand in location 2000(Move into AL)

Second Operand in location 2001(Move into BL)

ADDITION 16 BIT:

First Operand in location 2000(Move into AX)

Second Operand in location 2002(Move into BX)

SUBTRACTION 8 BIT:

First Operand in location 2000(Move into AL)

Second Operand in location 2001(Move into BL)

SUBTRACTION 16 BIT:

First Operand in location 2000(Move into AX)

Second Operand in location 2002(Move into BX)

MULTIPLICATOIN 8 BIT:

First Operand in location 2000(Move into AL)

Second Operand in location 2001(Move into BL)

MULTIPLICATION 16 BIT:

First Operand in location 2000(Move into AX)

Second Operand in location 2002(Move into BX)

DIVISION 16 BIT/8 BIT:

First Operand in location 2000(Move into AX)

Second Operand in location 2002(Move into BL)

DIVISION 32 BIT/16 BIT:

First Operand in location 2000(DX) and 2002(AX)(Move into DXAX)

Second Operand in location 2004(Move into BX)

OUTPUT FORMAT:

Output is Stored at location 3000

ADDITION 8 BIT:

Result is stored at location 3000(Moved from AL)

ADDITION 16 BIT:

Result is stored at location 3000(Moved from AX)

SUBTRACTION 8 BIT:

Result is stored at location 3000(Moved from AL)

SUBTRACTION 16 BIT:

Result is stored at location 3000(Moved from AX)

MULTIPLICATOIN 8 BIT:

Result is stored at location 3000(Moved from AX)

MULTIPLICATION 16 BIT:

Result is stored at location 3000(DX) 3002(AX)(Moved from DXAX)

DIVISION 16 BIT/8 BIT:

Result is stored at location 3000(Moved from AX)

Quotient: AL

Remainder: AH

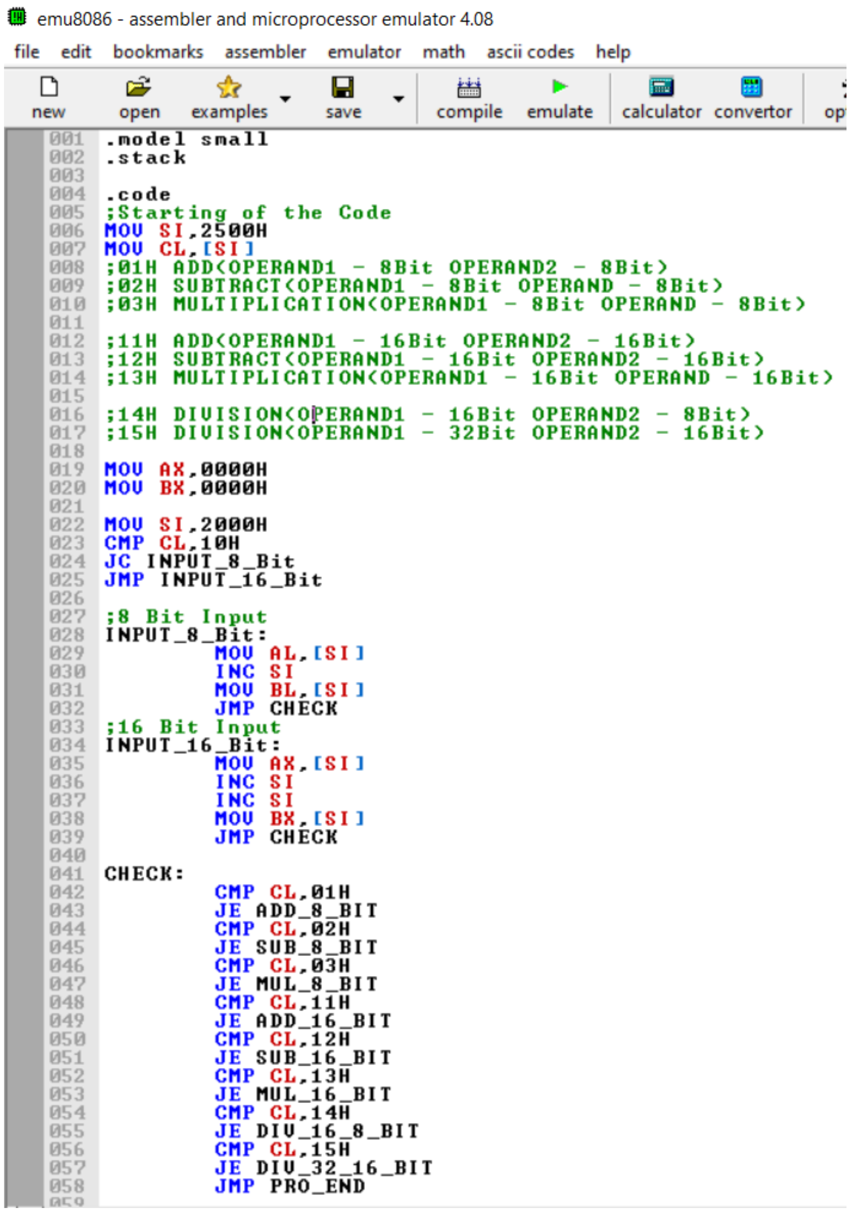
DIVISION 32 BIT/16 BIT:

Result is stored at location 3000(DX) and 3002(AX)(Moved from DXAX)

Quotient: AX

Remainder: DX

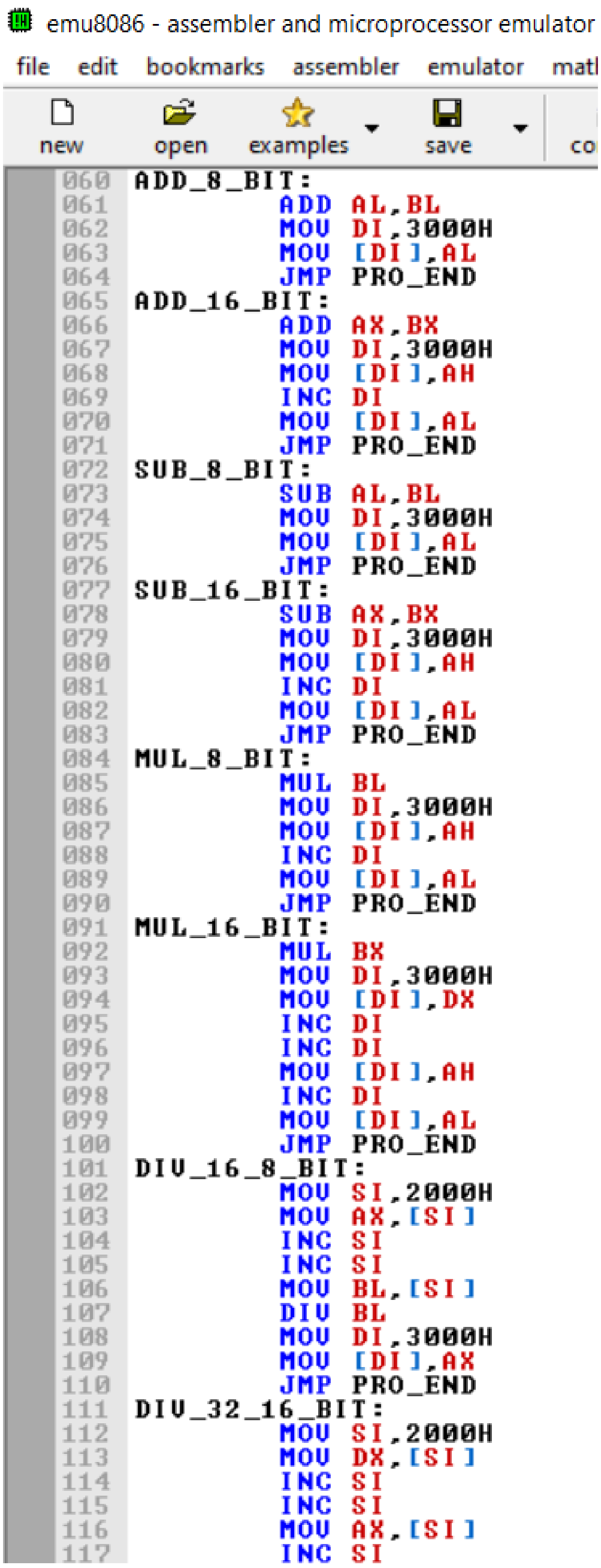
CODE SCREENSHOT:


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor

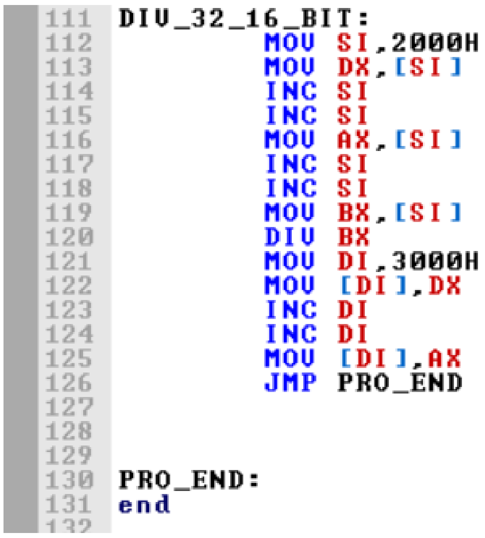
```
.model small
.stack

.code
;Starting of the Code
MOV SI,2500H
MOV CL,[SI]
;01H ADD(OPERAND1 - 8Bit OPERAND2 - 8Bit)
;02H SUBTRACT(OPERAND1 - 8Bit OPERAND - 8Bit)
;03H MULTIPLICATION(OPERAND1 - 8Bit OPERAND - 8Bit)

;11H ADD(OPERAND1 - 16Bit OPERAND2 - 16Bit)
;12H SUBTRACT(OPERAND1 - 16Bit OPERAND2 - 16Bit)
;13H MULTIPLICATION(OPERAND1 - 16Bit OPERAND - 16Bit)

;14H DIVISION(OPERAND1 - 16Bit OPERAND2 - 8Bit)
;15H DIVISION(OPERAND1 - 32Bit OPERAND2 - 16Bit)

MOV AX,0000H
MOV BX,0000H

MOV SI,2000H
CMP CL,10H
JC INPUT_8_Bit
JMP INPUT_16_Bit

;8 Bit Input
INPUT_8_Bit:
        MOV AL,[SI]
        INC SI
        MOV BL,[SI]
        JMP CHECK
;16 Bit Input
INPUT_16_Bit:
        MOV AX,[SI]
        INC SI
        INC SI
        MOV BX,[SI]
        JMP CHECK

CHECK:
        CMP CL,01H
        JE ADD_8_BIT
        CMP CL,02H
        JE SUB_8_BIT
        CMP CL,03H
        JE MUL_8_BIT
        CMP CL,11H
        JE ADD_16_BIT
        CMP CL,12H
        JE SUB_16_BIT
        CMP CL,13H
        JE MUL_16_BIT
        CMP CL,14H
        JE DIV_16_8_BIT
        CMP CL,15H
        JE DIV_32_16_BIT
        JMP PRO_END
```

ANIRUDH VADERA

DIGITAL ASSIGNMENT 2 CALCULATOR(ASSEMBLY LANGUAGE)


emu8086 - assembler and microprocessor emulator
file edit bookmarks assembler emulator matl
new
open
examples
save
co

```
ADD_8_BIT:
        ADD AL,BL
        MOV DI,3000H
        MOV [DI],AL
        JMP PRO_END
ADD_16_BIT:
        ADD AX,BX
        MOV DI,3000H
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
SUB_8_BIT:
        SUB AL,BL
        MOV DI,3000H
        MOV [DI],AL
        JMP PRO_END
SUB_16_BIT:
        SUB AX,BX
        MOV DI,3000H
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
MUL_8_BIT:
        MUL BL
        MOV DI,3000H
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
MUL_16_BIT:
        MUL BX
        MOV DI,3000H
        MOV [DI],DX
        INC DI
        INC DI
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
DIV_16_8_BIT:
        MOV SI,2000H
        MOV AX,[SI]
        INC SI
        INC SI
        MOV BL,[SI]
        DIV BL
        MOV DI,3000H
        MOV [DI],AX
        JMP PRO_END
DIV_32_16_BIT:
        MOV SI,2000H
        MOV DX,[SI]
        INC SI
        INC SI
        MOV AX,[SI]
        INC SI
```



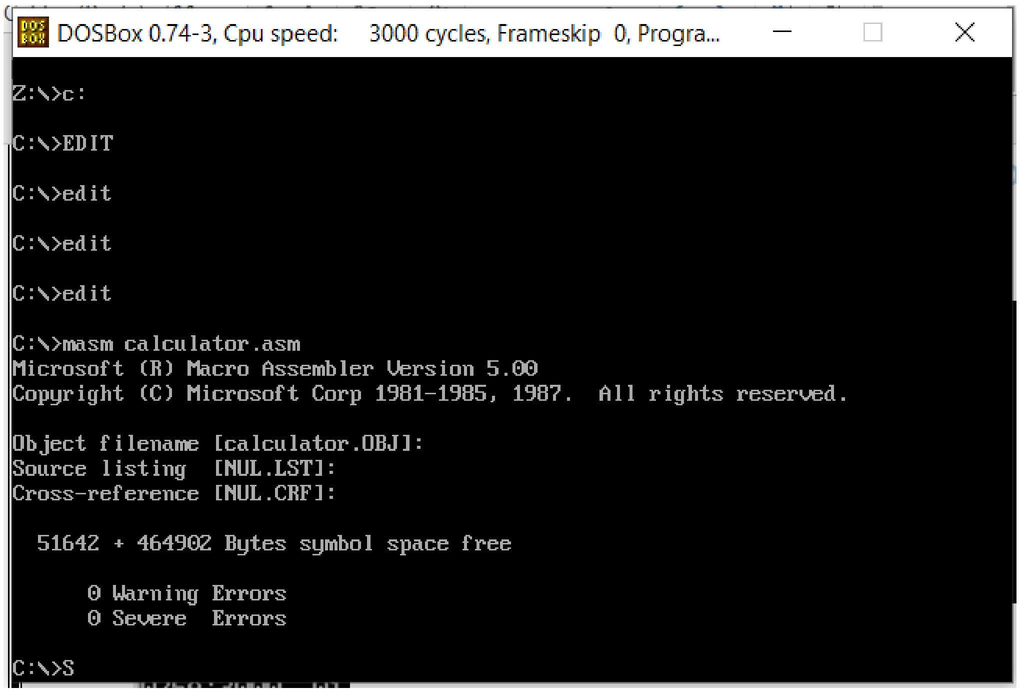
```
DIV_32_16_BIT:
        MOV SI,2000H
        MOV DX,[SI]
        INC SI
        INC SI
        MOV AX,[SI]
        INC SI
        INC SI
        MOV BX,[SI]
        DIV BX
        MOV DI,3000H
        MOV [DI],DX
        INC DI
        INC DI
        MOV [DI],AX
        JMP PRO_END



PRO_END:
end
```

ASSEMBLY LANGUAGE CODE:

```
.model small

.stack


.code

;Starting of the Code

MOV SI,2500H

MOV CL,[SI]

;01H ADD(OPERAND1 - 8Bit OPERAND2 - 8Bit)

;02H SUBTRACT(OPERAND1 - 8Bit OPERAND - 8Bit)

;03H MULTIPLICATION(OPERAND1 - 8Bit OPERAND - 8Bit)


;11H ADD(OPERAND1 - 16Bit OPERAND2 - 16Bit)

;12H SUBTRACT(OPERAND1 - 16Bit OPERAND2 - 16Bit)

;13H MULTIPLICATION(OPERAND1 - 16Bit OPERAND - 16Bit)


;14H DIVISION(OPERAND1 - 16Bit OPERAND2 - 8Bit)

;15H DIVISION(OPERAND1 - 32Bit OPERAND2 - 16Bit)


MOV AX,0000H

MOV BX,0000H


MOV SI,2000H

CMP CL,10H

JC INPUT_8_Bit

JMP INPUT_16_Bit


;8 Bit Input

INPUT_8_Bit:
        MOV AL,[SI]
        INC SI
        MOV BL,[SI]
        JMP CHECK
;16 Bit Input
INPUT_16_Bit:
        MOV AX,[SI]
        INC SI
        INC SI
        MOV BX,[SI]
        JMP CHECK

CHECK:
        CMP CL,01H
        JE ADD_8_BIT
        CMP CL,02H
        JE SUB_8_BIT
        CMP CL,03H
        JE MUL_8_BIT
        CMP CL,11H
        JE ADD_16_BIT
        CMP CL,12H
        JE SUB_16_BIT
        CMP CL,13H
        JE MUL_16_BIT
        CMP CL,14H
        JE DIV_16_8_BIT
        CMP CL,15H
        JE DIV_32_16_BIT
        JMP PRO_END


ADD_8_BIT:
        ADD AL,BL
        MOV DI,3000H
        MOV [DI],AL
        JMP PRO_END
ADD_16_BIT:
        ADD AX,BX
        MOV DI,3000H
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
SUB_8_BIT:
        SUB AL,BL
        MOV DI,3000H
        MOV [DI],AL
        JMP PRO_END
SUB_16_BIT:
        SUB AX,BX
        MOV DI,3000H
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
MUL_8_BIT:
        MUL BL
        MOV DI,3000H
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
MUL_16_BIT:
        MUL BX
        MOV DI,3000H
        MOV [DI],DX
        INC DI
        INC DI
        MOV [DI],AH
        INC DI
        MOV [DI],AL
        JMP PRO_END
DIV_16_8_BIT:
        MOV SI,2000H
        MOV AX,[SI]
        INC SI
        INC SI
        MOV BL,[SI]
        DIV BL
        MOV DI,3000H
        MOV [DI],AX
        JMP PRO_END
DIV_32_16_BIT:
        MOV SI,2000H
        MOV DX,[SI]
        INC SI
        INC SI
        MOV AX,[SI]

        INC SI

        INC SI

        MOV BX,[SI]

        DIV BX

        MOV DI,3000H

        MOV [DI],DX

        INC DI

        INC DI

        MOV [DI],AX

        JMP PRO_END




PRO_END:

end
```

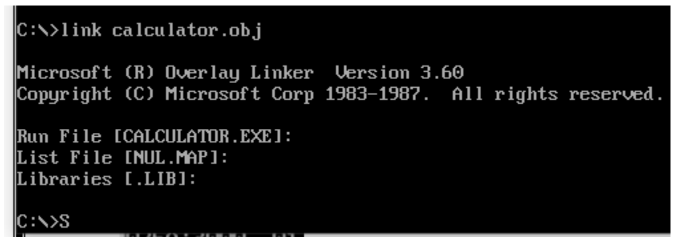
RESULTS:

Command: masm calculator.asm


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

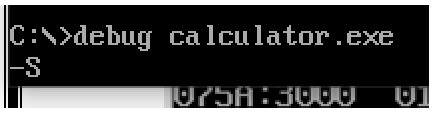
```
Z:\>c:

C:\>EDIT

C:\>edit

C:\>edit

C:\>edit

C:\>masm calculator.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [calculator.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51642 + 464902 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>S
```

Command: link calculator.obj



```
C:\>link calculator.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [CALCULATOR.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S
```

Command: debug calculator.exe



```
C:\>debug calculator.exe
-S
```

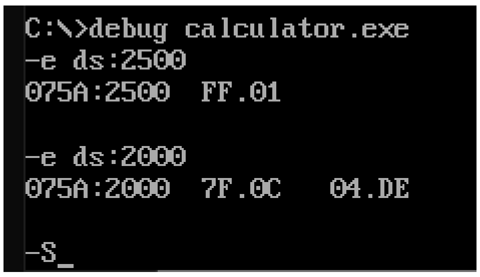
8 Bit Addition:

INPUT:

At Location 2500H: **01H (Operand Type (Addition 8 Bit))**

Giving Inputs from location 2000H

Given Input: 0CH + DEH

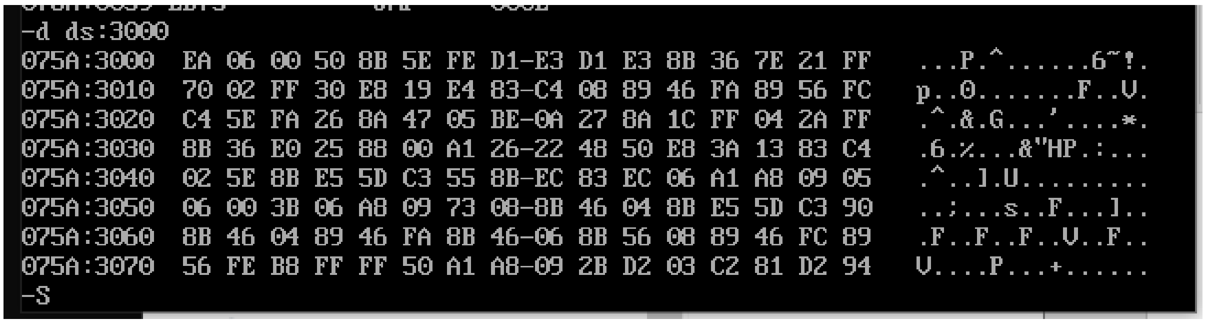


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  FF.01

-e ds:2000
075A:2000  7F.0C   04.DE

-S_
```

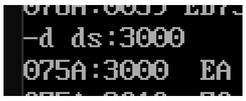
Expected Output: EAH

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  EA 06 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S
```



```
-d ds:3000
075A:3000  EA
```

Hence the Output is **EAH** hence addition 8 Bit is verified

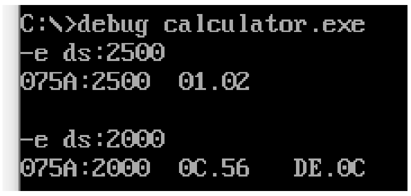
8 Bit Subtraction:

INPUT:

At Location 2500H: **02H (Operand Type (Subtraction 8 Bit))**

Giving Inputs from location 2000H

Given Input: 56H – 0CH

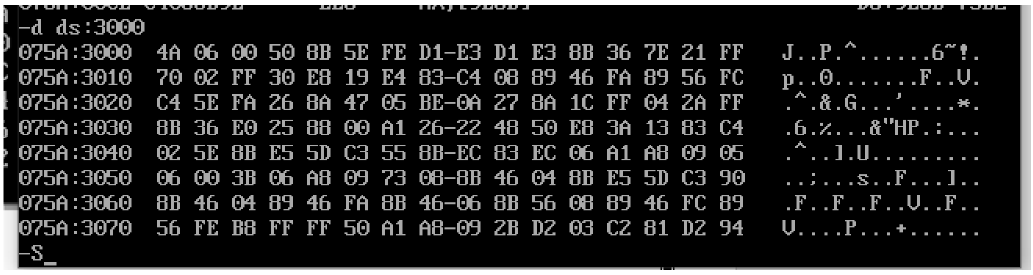


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  01.02

-e ds:2000
075A:2000  0C.56    DE.0C
```

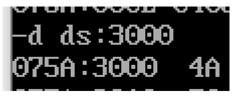
Expected Output: 4AH

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  4A 06 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   J..P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S_
```



```
-d ds:3000
075A:3000  4A
```

Hence the Output is **4AH** hence substraction 8 Bit is verified

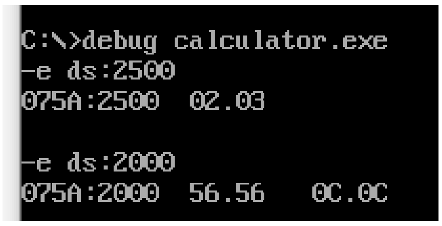
8 Bit Multiplication:

INPUT:

At Location 2500H: **03H (Operand Type (Multiplication 8 Bit))**

Giving Inputs from location 2000H

Given Input: 56H * 0CH

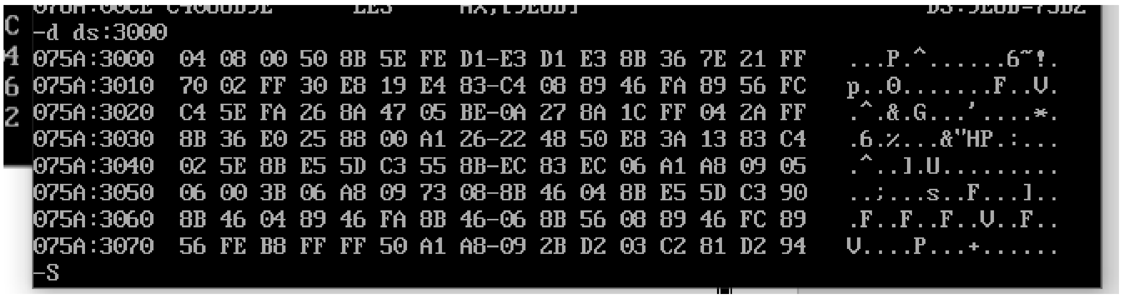


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  02.03

-e ds:2000
075A:2000  56.56    0C.0C
```

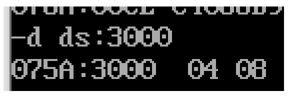
Expected Output: 0408H

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  04 08 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S
```



```
-d ds:3000
075A:3000  04 08
```

Hence the Output is **0408H** hence multiplication 8 Bit is verified

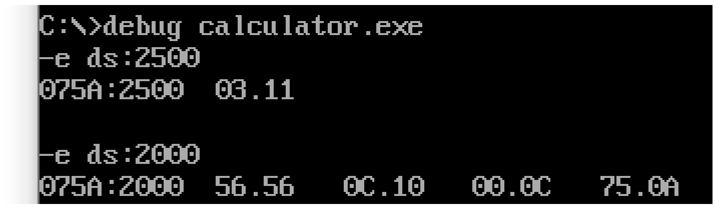
16 Bit Addition:

INPUT:

At Location 2500H: **11H (Operand Type (Addition 16 Bit))**

Giving Inputs from location 2000H

Given Input: 1056H + 0A0CH

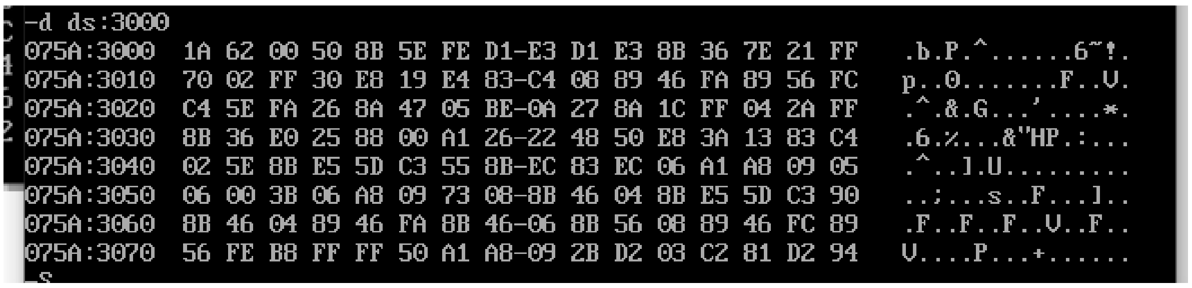


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  03.11

-e ds:2000
075A:2000  56.56   0C.10   00.0C   75.0A
```

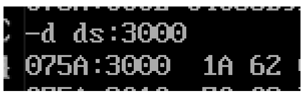
Expected Output: 1A62H

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  1A 62 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   .b.P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
```



```
-d ds:3000
075A:3000  1A 62
```

Hence the Output is **1A62H** hence addition 16 Bit is verified

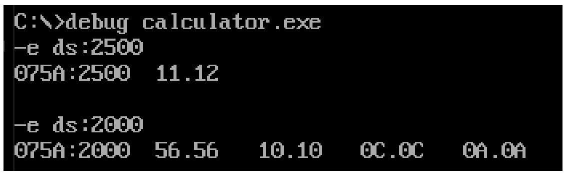
16 Bit Subtraction:

INPUT:

At Location 2500H: **12H (Operand Type (Subtraction 16 Bit))**

Giving Inputs from location 2000H

Given Input: 1056H – 0A0CH

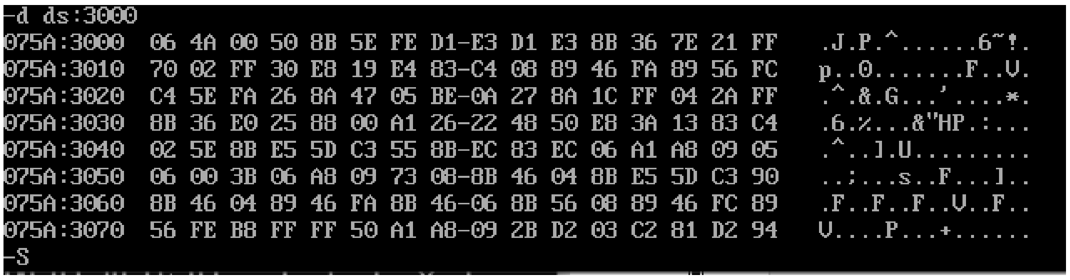


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  11.12

-e ds:2000
075A:2000  56.56   10.10   0C.0C   0A.0A
```

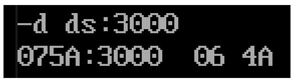
Expected Output: 064AH

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  06 4A 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   .J.P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S
```



```
-d ds:3000
075A:3000  06 4A
```

Hence the Output is **064AH** hence substraction 16 Bit is verified

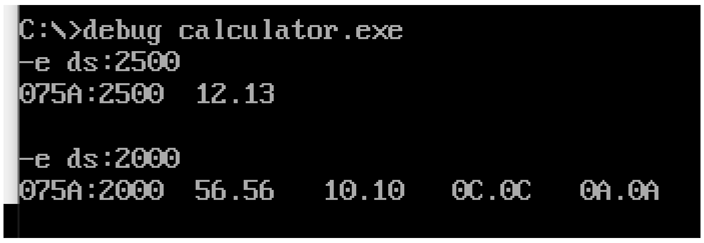
16 Bit Multiplication:

INPUT:

At Location 2500H: **13H (Operand Type (Multiplication 16 Bit))**

Giving Inputs from location 2000H

Given Input: 1056H * 0A0CH

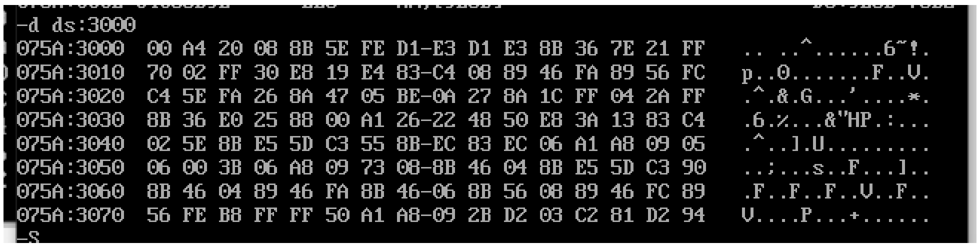


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  12.13

-e ds:2000
075A:2000  56.56   10.10   0C.0C   0A.0A
```

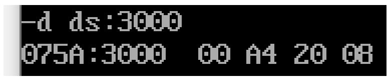
Expected Output: 00A42008H

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  00 A4 20 08 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   .. ..^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S
```



```
-d ds:3000
075A:3000  00 A4 20 08
```

Hence the Output is **00A42008H** hence multiplication 16 Bit is verified

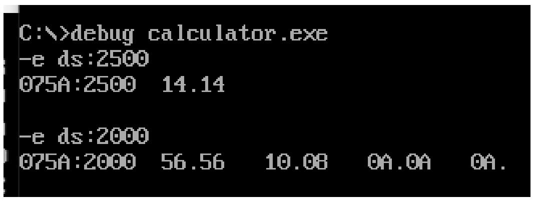
16Bit / 8 Bit Division:

INPUT:

At Location 2500H: **14H (Operand Type (Division 16 bit / 8 Bit))**

Giving Inputs from location 2000H

Given Input: 0856H * 0AH

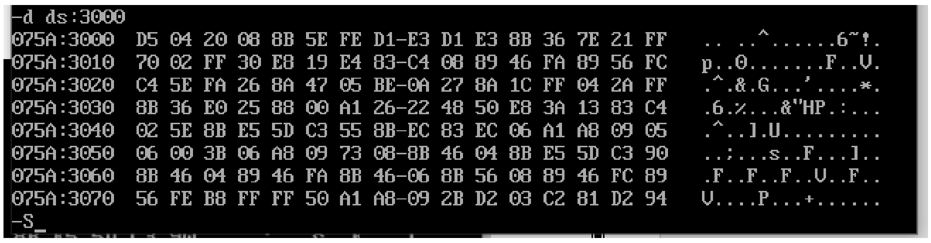


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  14.14

-e ds:2000
075A:2000  56.56   10.08   0A.0A   0A.
```

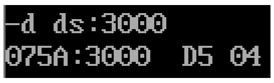
Expected Output: D5H(Quotient) 04H(Remainder)

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  D5 04 20 08 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   .. ..^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S_
```



```
-d ds:3000
075A:3000  D5 04
```

Hence AL(Quotient) = D5

AH(Remainder) = 04

Hence division 16 Bit/8 Bit is verified

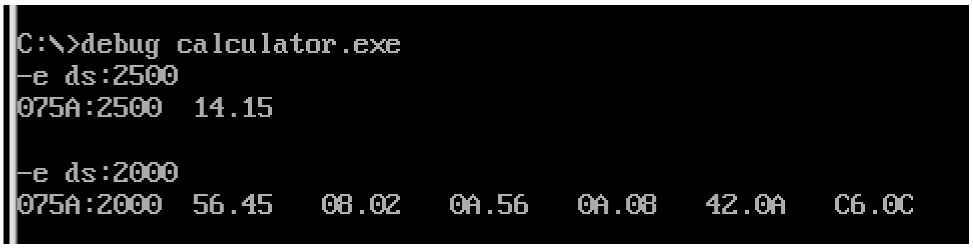
32Bit / 16 Bit Division:

INPUT:

At Location 2500H: **15H (Operand Type (Division 32 bit / 16 Bit))**

Giving Inputs from location 2000H

Given Input: 02450856H * 0C0AH

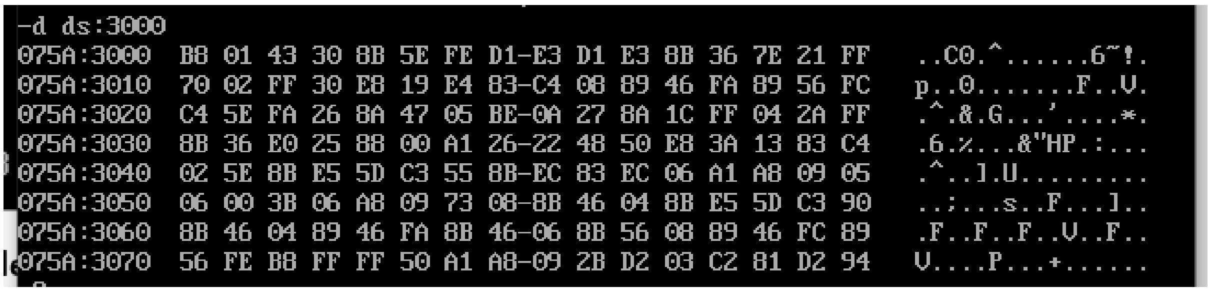


```
C:\>debug calculator.exe
-e ds:2500
075A:2500  14.15

-e ds:2000
075A:2000  56.45   08.02   0A.56   0A.08   42.0A   C6.0C
```

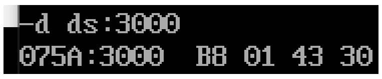
Expected Output: 3043H(Quotient) 018BH(Remainder)

OUTPUT:

Output Stored at location 3000H



```
-d ds:3000
075A:3000  B8 01 43 30 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ..C0.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
```



```
-d ds:3000
075A:3000  B8 01 43 30
```

Hence AX(Quotient) = 3043

DX(Remainder) = 01B8

Hence division 32 Bit/16 Bit is verified