UCS1512 Microprocessors Lab

Experiment 1: 8-bit Arithmetic Operations

Date: 18-08-2020

Name: Nivedhitha D

Register Number: 185001104

Aim

To write programs for performing 8-bit arithmetic operations in an 8086 microprocessor using MASM and DOSBox.

Procedure for executing 8086 programs in MASM assembler

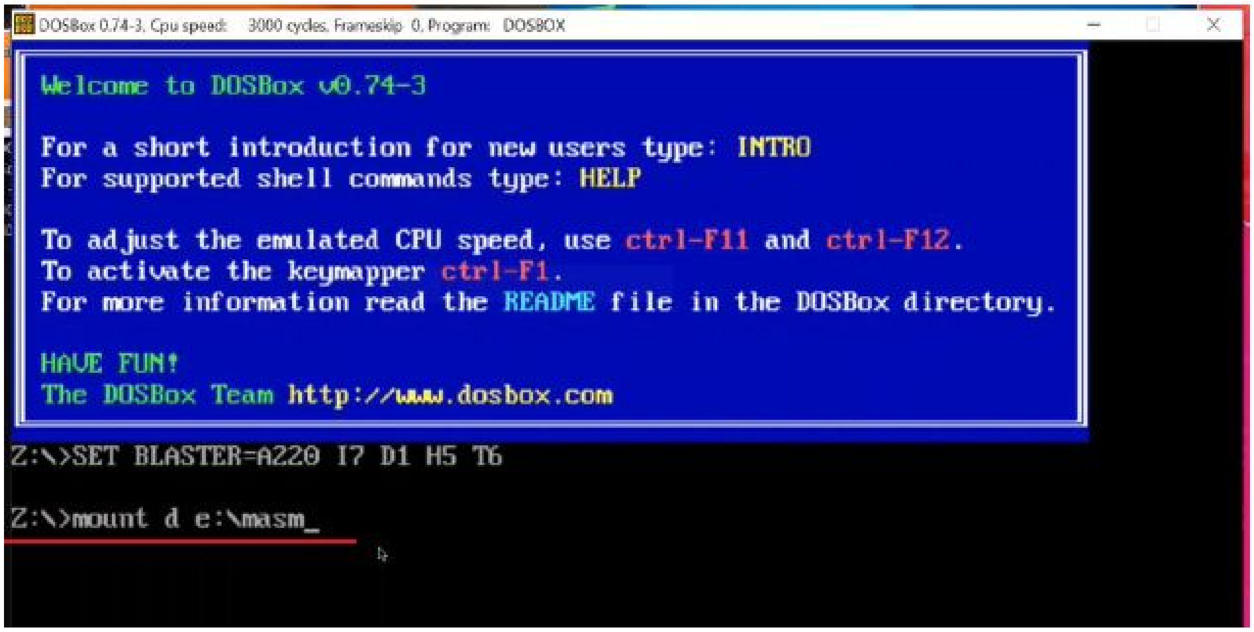
Prerequisite

1. Install the software DOSBox 0.74-3 (Link for DOSBox)

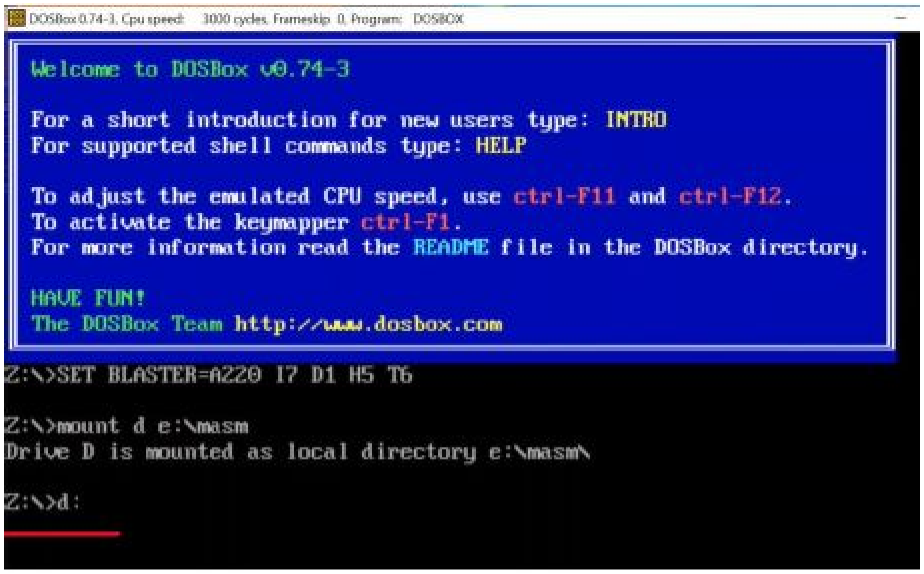
2. Download the masm files required

Using DOSBox

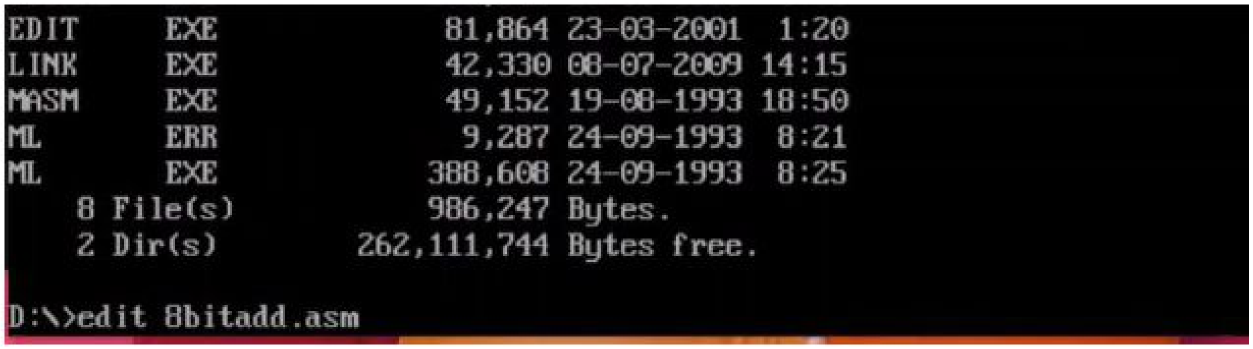
1. Open DosBox
2. Mount the masm folder to a drive on DOSBox using the command, **mount**


```
DOSBox 0.74-3, Cpu speed:    3000 cycles, Frameskip  0, Program:   DOSBOX

Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount d e:\masm_
```

3. Goto the mounted drive (here it is D)

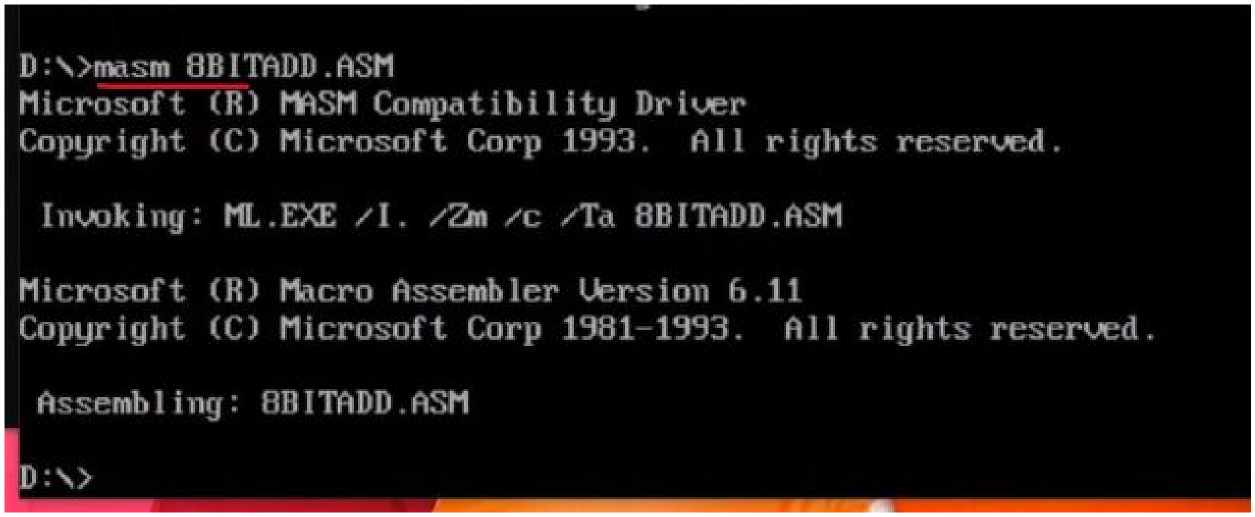

```
DOSBox 0.74-3, Cpu speed:    3000 cycles, Frameskip  0, Program:   DOSBOX

Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount d e:\masm
Drive D is mounted as local directory e:\masm\

Z:\>d:
```

4. Save the 8086 programs with extension .asm in the same folder using the **edit** command



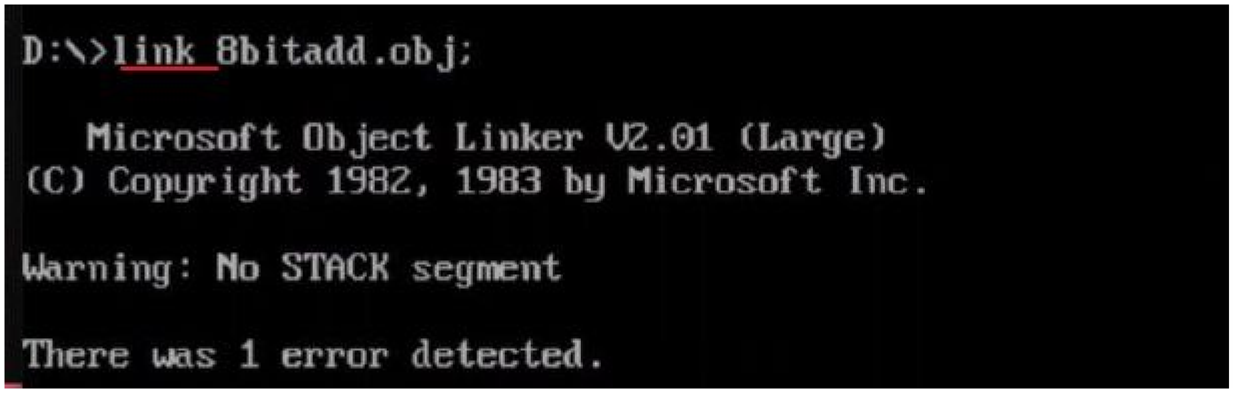
```
EDIT     EXE            81,864 23-03-2001  1:20
LINK     EXE            42,330 08-07-2009 14:15
MASM     EXE            49,152 19-08-1993 18:50
ML       ERR             9,287 24-09-1993  8:21
ML       EXE           388,608 24-09-1993  8:25
    8 File(s)          986,247 Bytes.
    2 Dir(s)       262,111,744 Bytes free.

D:\>edit 8bitadd.asm
```

5. After creating the file, assemble it using the command **masm** *filename.asm*



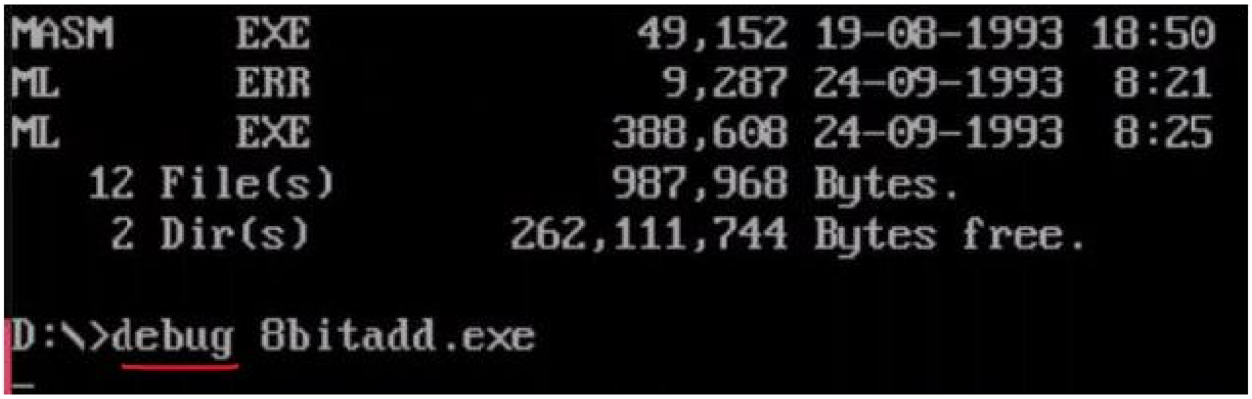
```
D:\>masm 8BITADD.ASM
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993.  All rights reserved.

 Invoking: ML.EXE /I. /Zm /c /Ta 8BITADD.ASM

Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993.  All rights reserved.

 Assembling: 8BITADD.ASM

D:\>
```

6. Link the file using the command **link** *filename.obj*; (Use the semicolon to use the default name for the executable file, unless you want to rename it)



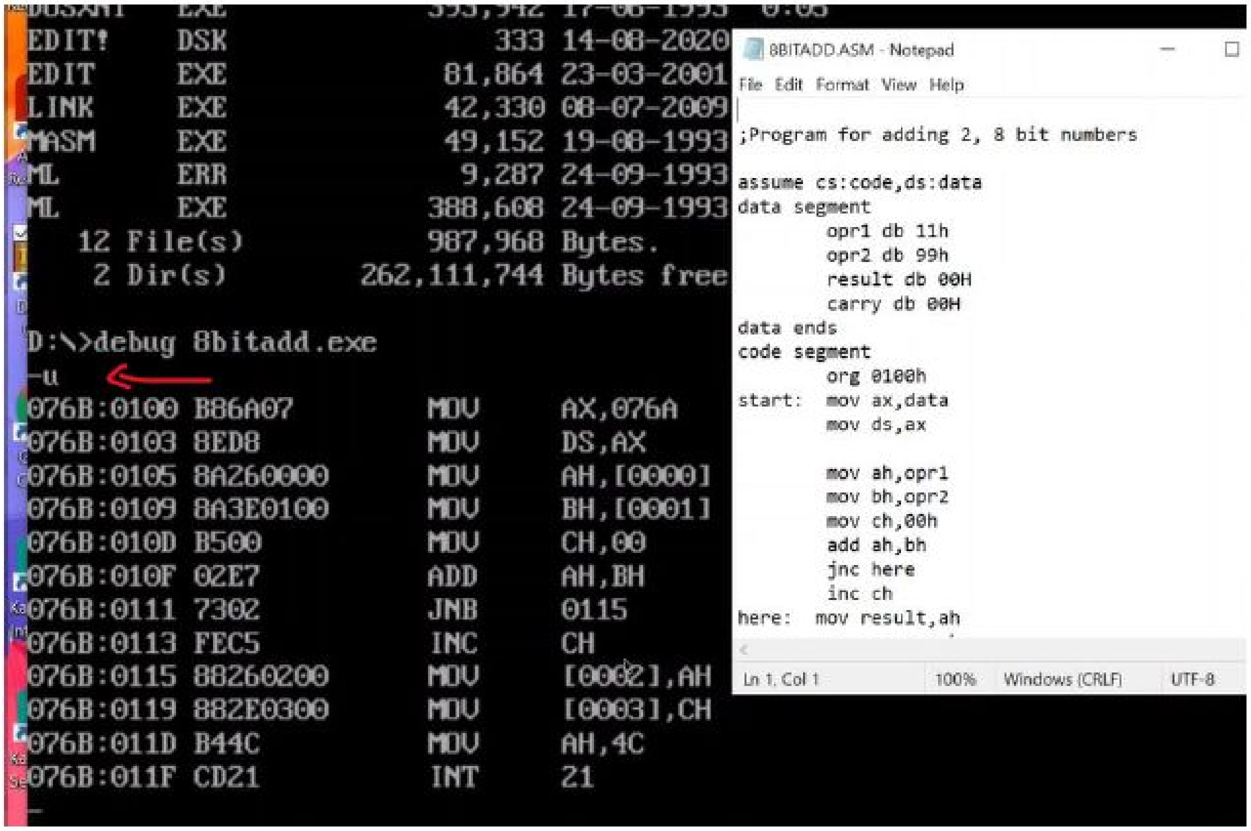
```
D:\>link 8bitadd.obj;

   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.
```

7. This warning and error can be neglected as the stack segment is not being used in this program
8. Use **debug** command with *filename.exe* to execute and analyse the memory contents, **debug** *filename.exe*



```
MASM      EXE           49,152 19-08-1993 18:50
ML        ERR            9,287 24-09-1993  8:21
ML        EXE          388,608 24-09-1993  8:25
   12 File(s)          987,968 Bytes.
    2 Dir(s)       262,111,744 Bytes free.

D:\>debug 8bitadd.exe
-
```

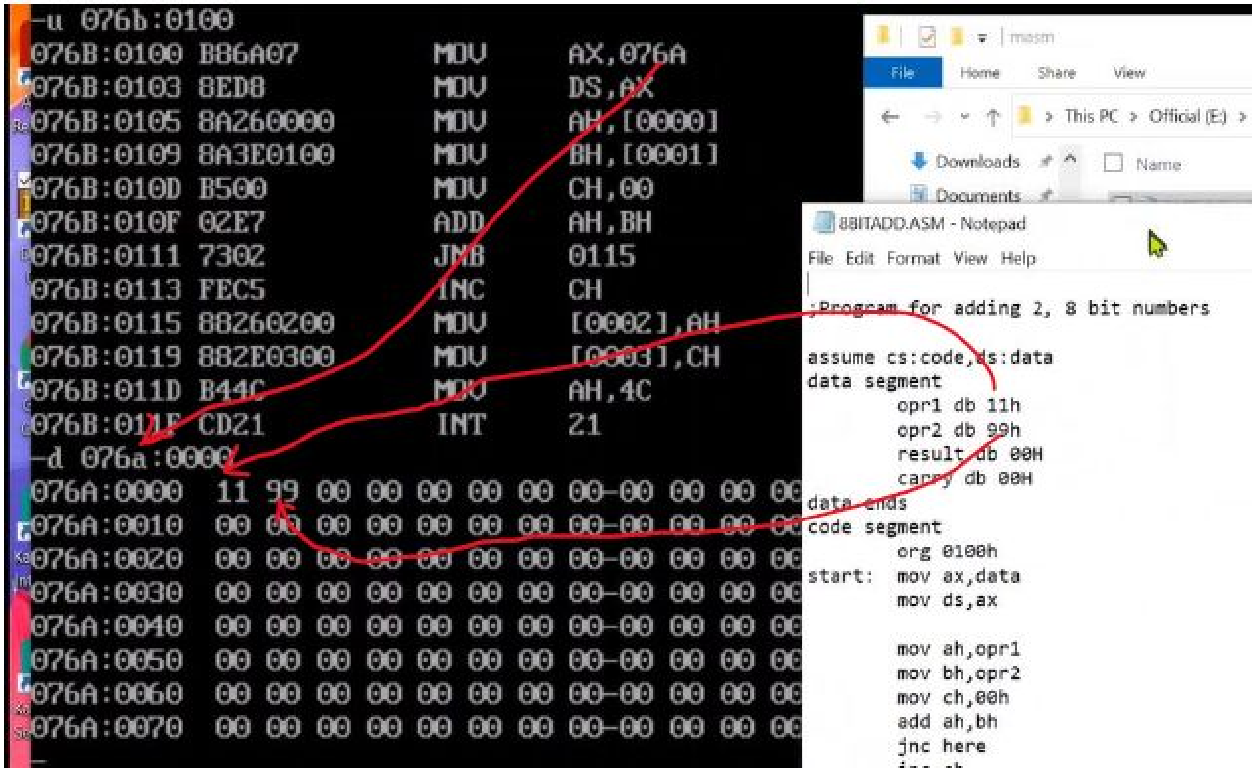
9. In debug, command **u** will display the unassembled code



```
EDIT!     DSK              333 14-08-2020
EDIT      EXE           81,864 23-03-2001
LINK      EXE           42,330 08-07-2009
MASM      EXE           49,152 19-08-1993
ML        ERR            9,287 24-09-1993
ML        EXE          388,608 24-09-1993
   12 File(s)          987,968 Bytes.
    2 Dir(s)       262,111,744 Bytes free

D:\>debug 8bitadd.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8A260000      MOV     AH,[0000]
076B:0109 8A3E0100      MOV     BH,[0001]
076B:010D B500          MOV     CH,00
076B:010F 02E7          ADD     AH,BH
076B:0111 7302          JNB     0115
076B:0113 FEC5          INC     CH
076B:0115 88260200      MOV     [0002],AH
076B:0119 882E0300      MOV     [0003],CH
076B:011D B44C          MOV     AH,4C
076B:011F CD21          INT     21
-
```

```
;Program for adding 2, 8 bit numbers

assume cs:code,ds:data
data segment
        opr1 db 11h
        opr2 db 99h
        result db 00H
        carry db 00H
data ends
code segment
        org 0100h
start:  mov ax,data
        mov ds,ax

        mov ah,opr1
        mov bh,opr2
        mov ch,00h
        add ah,bh
        jnc here
        inc ch
here:   mov result,ah
```

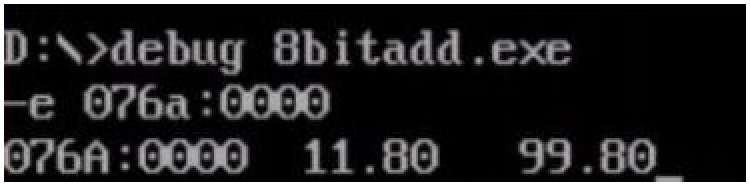
10. Use command **d** *segment:offset* to see the content of memory locations starting from segment:offset address



```
-u 076b:0100
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8A260000      MOV     AH,[0000]
076B:0109 8A3E0100      MOV     BH,[0001]
076B:010D B500          MOV     CH,00
076B:010F 02E7          ADD     AH,BH
076B:0111 7302          JNB     0115
076B:0113 FEC5          INC     CH
076B:0115 88260200      MOV     [0002],AH
076B:0119 882E0300      MOV     [0003],CH
076B:011D B44C          MOV     AH,4C
076B:011F CD21          INT     21
-d 076a:0000
076A:0000  11 99 00 00 00 00 00 00-00 00 00 00
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00
```

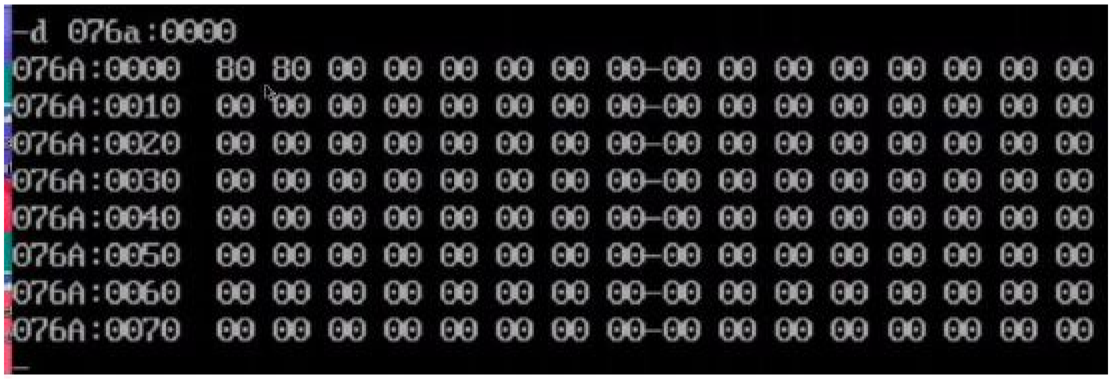
```
;Program for adding 2, 8 bit numbers

assume cs:code,ds:data
data segment
        opr1 db 11h
        opr2 db 99h
        result db 00H
        carry db 00H
data ends
code segment
        org 0100h
start:  mov ax,data
        mov ds,ax

        mov ah,opr1
        mov bh,opr2
        mov ch,00h
        add ah,bh
        jnc here
```

11. To change the value in memory, use the command **e** *segment:offset*
12. Press space if you want to edit the next location and press enter to finish (here 80 is the new value given by the user)



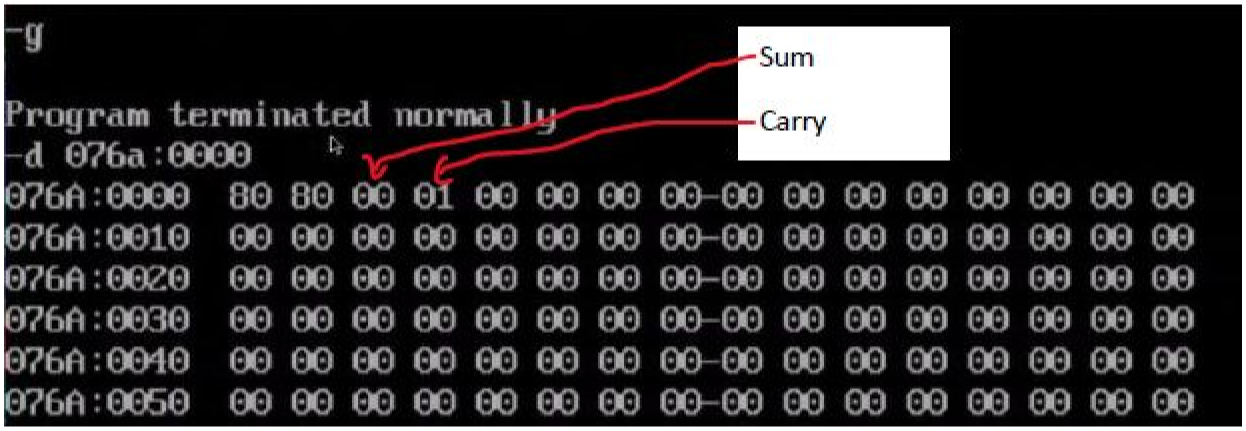
```
D:\>debug 8bitadd.exe
-e 076a:0000
076A:0000  11.80   99.80_
```

13. Verify the memory contents to ensure the updates (using command **d**)



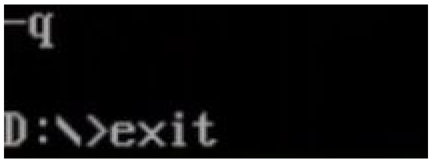
```
-d 076a:0000
076A:0000  80 80 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
-
```

14. Execute using the command **g** and check the output



```
-g

Program terminated normally
-d 076a:0000
076A:0000  80 80 00 01 00 00 00 00-00 00 00 00 00 00 00 00
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

15. Command **q** to exit from the debug mode and command **exit** for closing DOSBox



```
-q

D:\>exit
```

Algorithm

1. Define the values in the data segment and assign the initial values if required
2. Initialize the data segment register with a data segment address
3. Load the data into the appropriate registers and perform addition/ subtraction/ multiplication/ division and store the sum/ difference/product/quotient-remainder to the result address for display

4. Terminate the program

Program for adding two 8-bit numbers

Program Name: 8BITADD.ASM

Program	**Comments**
;Program for adding two 8-bit numbers ASSUME CS:CODE,DS:DATA	Naming the CS and DS for the program

DATA SEGMENT
    OPR1 DB 11H
    OPR2 DB 99H
	Declaring and initializing the values for the two operands in the data segment

RESULT DB 00H CARRY DB 00H DATA ENDS	The carry and result must be set to zero to obtain the correct result and get rid off garbage values
CODE SEGMENT ORG 0100H	Providing an offset value

START:
    MOV AX, DATA
    MOV DS, AX
	Initializing the data segment register with the data segment address

MOV AH, OPR1
    MOV BH, OPR2
    MOV CH, 00H
	Loading the data into the appropriate registers

ADD AH, BH	Performing addition: AH = AH + BH
JNC HERE	If the carry is zero, go-to label, HERE
INC CH	If carry has occurred, increment register, CH

HERE:
    MOV RESULT, AH
    MOV CARRY, CH
	Loading the sum and carry into the appropriate locations for display

MOV AH, 4CH
    INT 21H
    CODE ENDS
END START
	Calling the DOS Function to enter the display screen using interrupt 21H and to terminate the program

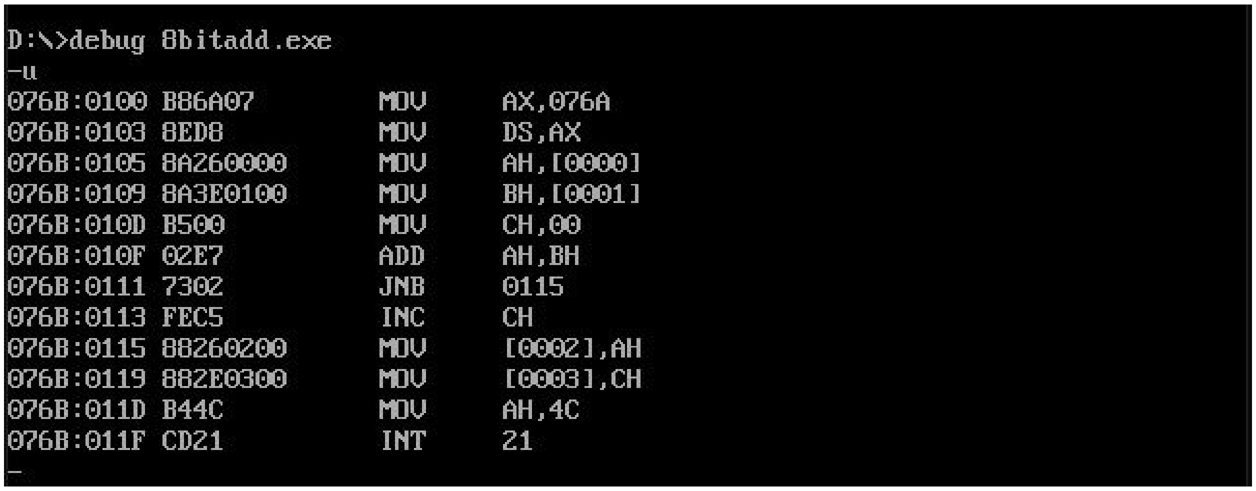
Snapshot of Input and Output

Input

Two 8-bit values

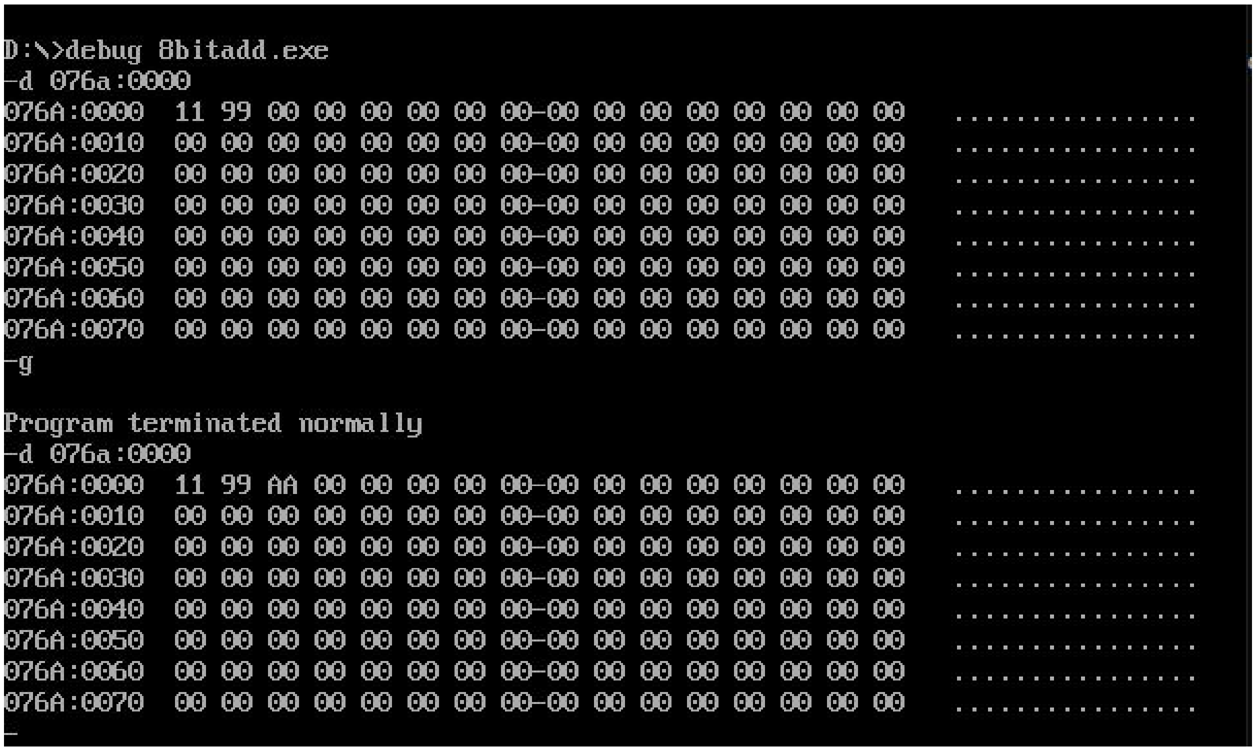
Output

Sum in one memory location and the carry in another



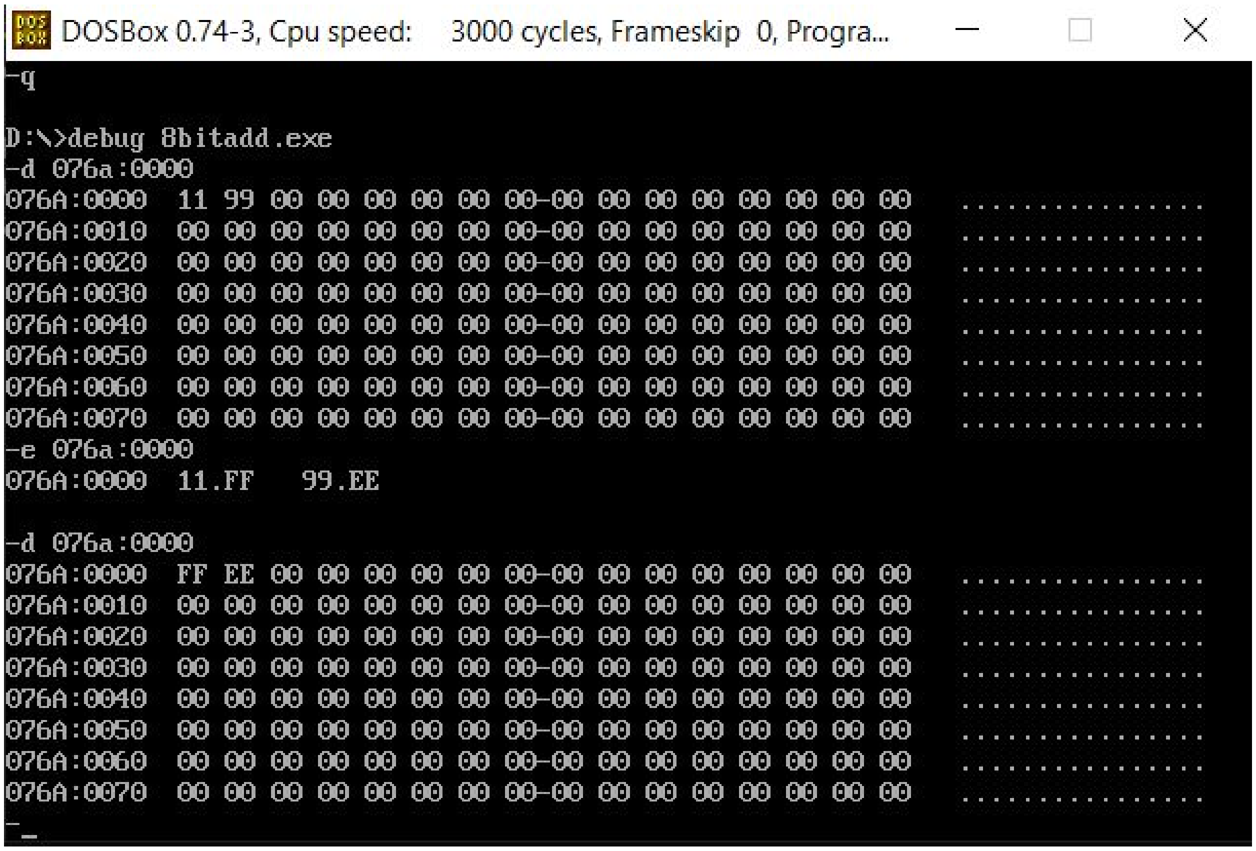
```
D:\>debug 8bitadd.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8A260000      MOV     AH,[0000]
076B:0109 8A3E0100      MOV     BH,[0001]
076B:010D B500          MOV     CH,00
076B:010F 02E7          ADD     AH,BH
076B:0111 7302          JNB     0115
076B:0113 FEC5          INC     CH
076B:0115 88260200      MOV     [0002],AH
076B:0119 882E0300      MOV     [0003],CH
076B:011D B44C          MOV     AH,4C
076B:011F CD21          INT     21
-
```

Without carry



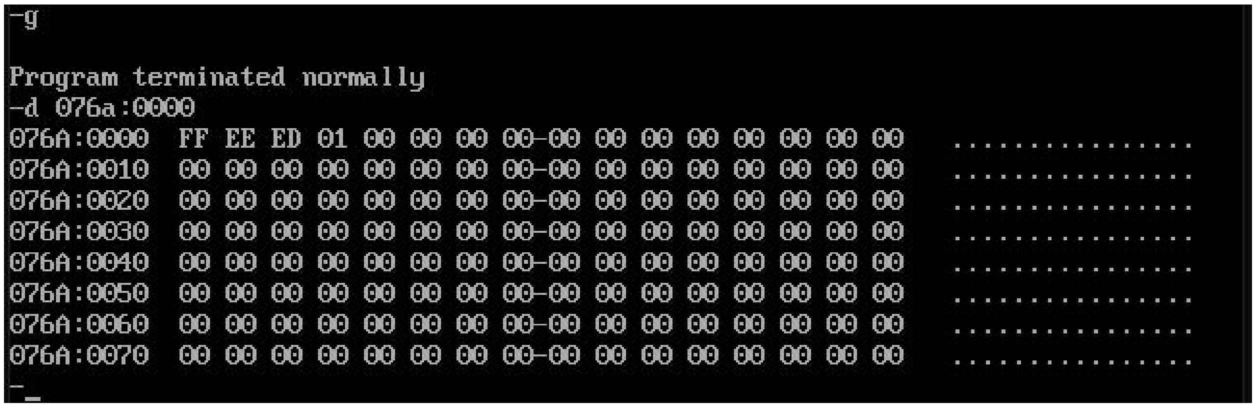
```
D:\>debug 8bitadd.exe
-d 076a:0000
076A:0000  11 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076a:0000
076A:0000  11 99 AA 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

With carry



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

```
-q

D:\>debug 8bitadd.exe
-d 076a:0000
076A:0000  11 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-e 076a:0000
076A:0000  11.FF   99.EE

-d 076a:0000
076A:0000  FF EE 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```



```
-g

Program terminated normally
-d 076a:0000
076A:0000  FF EE ED 01 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

Program for subtracting two 8-bit numbers

Program Name: 8BITSUB.ASM

Program	**Comments**
;Program for subtracting two 8-bit numbers ASSUME CS:CODE,DS:DATA	Naming the CS and DS for the program

DATA SEGMENT
        OPR1 DB 11H
        OPR2 DB 99H
	Declaring and initializing the values for the two operands in the data segment

DIFF DB 00H SIGN DB 00H DATA ENDS	The difference and sign must be set to zero to obtain the correct result and get rid off garbage values
CODE SEGMENT ORG 0100H	Providing an offset value

START:
        MOV AX, DATA
        MOV DS, AX
	Initializing the data segment register with the data segment address

MOV AH, OPR1
        MOV BH, OPR2
        MOV CH, 00H
	Loading the data into the appropriate registers

SUB AH, BH	Performing subtraction: AH = AH - BH
JNC HERE	If the carry is zero, go-to label, HERE
NEG AH INC CH	If carry has occurred, increment register, CH to denote sign

HERE:
        MOV DIFF, AH
        MOV SIGN, CH
	Loading the difference and sign into the appropriate locations for display

MOV AH, 4CH
        INT 21H
        CODE ENDS
END START
	Calling the DOS Function to enter the display screen using interrupt 21H and to terminate the program

Snapshot of Input and Output

Input

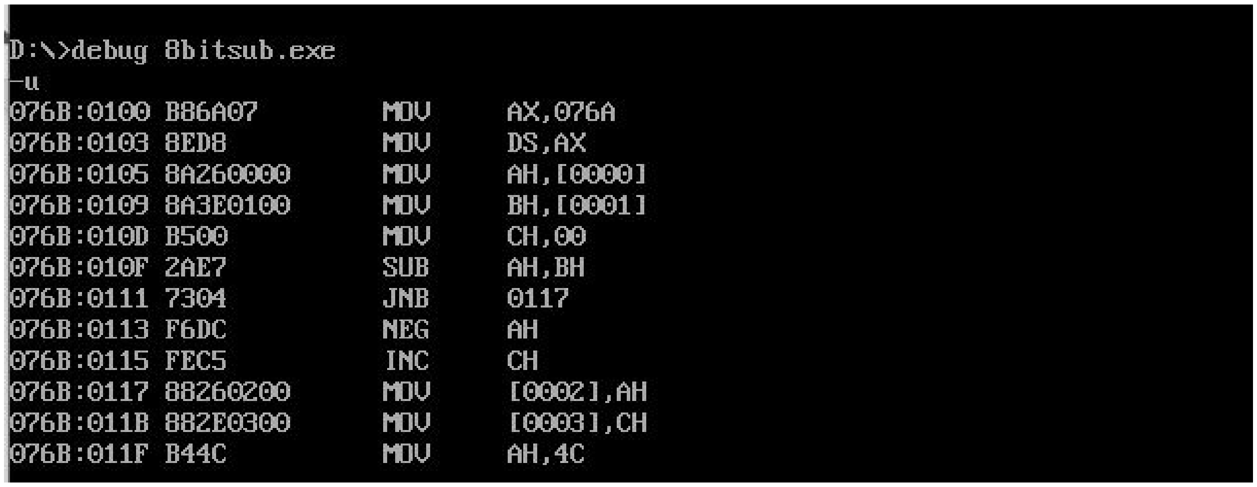
Two 8-bit values

Output

Difference in one memory location and the indication of the sign in another.

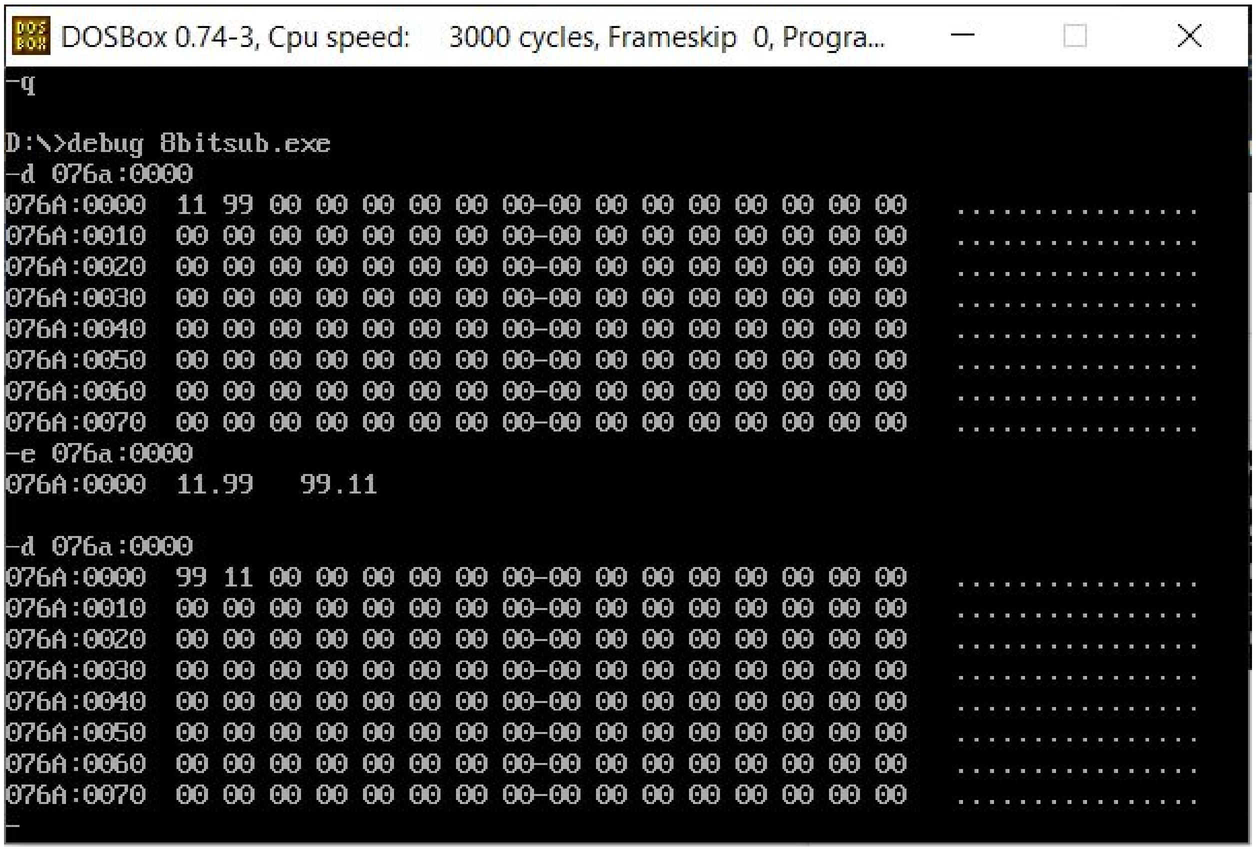
(eg: FF-FE = 01, indication of sign is 00 i.e. positive

FE – FF = 01, indication of sign is 01 i.e. negative)



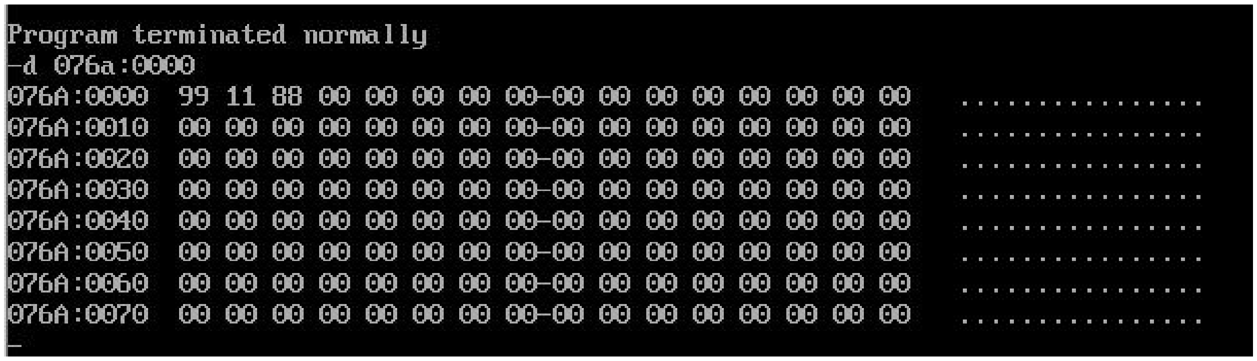
```
D:\>debug 8bitsub.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8A260000      MOV     AH,[0000]
076B:0109 8A3E0100      MOV     BH,[0001]
076B:010D B500          MOV     CH,00
076B:010F 2AE7          SUB     AH,BH
076B:0111 7304          JNB     0117
076B:0113 F6DC          NEG     AH
076B:0115 FEC5          INC     CH
076B:0117 88260200      MOV     [0002],AH
076B:011B 882E0300      MOV     [0003],CH
076B:011F B44C          MOV     AH,4C
```

Positive Sign



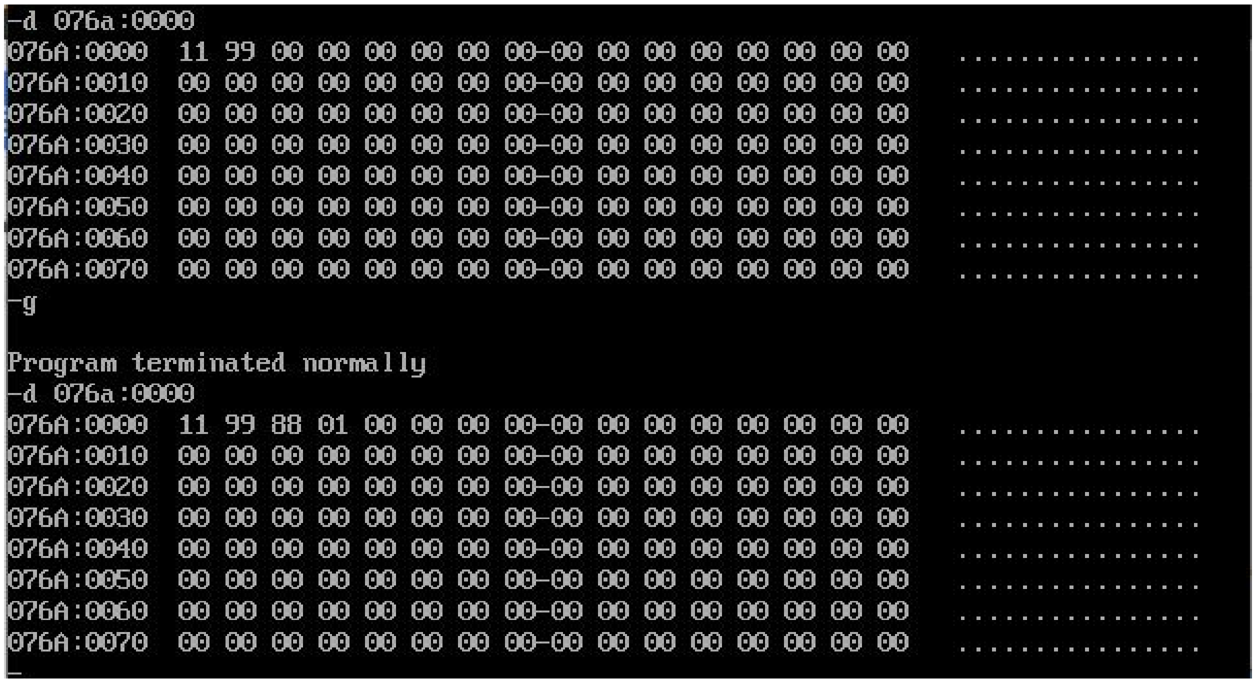
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
-q

D:\>debug 8bitsub.exe
-d 076a:0000
076A:0000  11 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-e 076a:0000
076A:0000  11.99   99.11

-d 076a:0000
076A:0000  99 11 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```



```
Program terminated normally
-d 076a:0000
076A:0000  99 11 88 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Negative Sign



```
-d 076a:0000
076A:0000  11 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076a:0000
076A:0000  11 99 88 01 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Program for multiplying two 8-bit numbers

Program Name: 8BITMUL.ASM

Program	**Comments**
;Program for multiplying two 8-bit numbers ASSUME CS:CODE,DS:DATA	Naming the CS and DS for the program

DATA SEGMENT
        OPR1 DB 11H
        OPR2 DB 44H
	Declaring and initializing the values for the two operands in the data segment

PRODH DB 00H PRODL DB 00H DATA ENDS	The higher and lower order bits of the product must be set to zero to obtain the correct result and get rid off garbage values
CODE SEGMENT ORG 0100H	Providing an offset value

START:
        MOV AX, DATA
        MOV DS, AX
	Initializing the data segment register with the data segment address

MOV AL, OPR1 MOV BL, OPR2	Loading the data into the appropriate registers
MUL BL	Performing multiplication: AX = AL * BL
MOV PRODH, AH MOV PRODL, AL	Since the product is stored in the AX register, the higher and lower order bits are extracted

MOV AH, 4CH
        INT 21H
        CODE ENDS
END START
	Calling the DOS Function to enter the display screen using interrupt 21H and to terminate the program

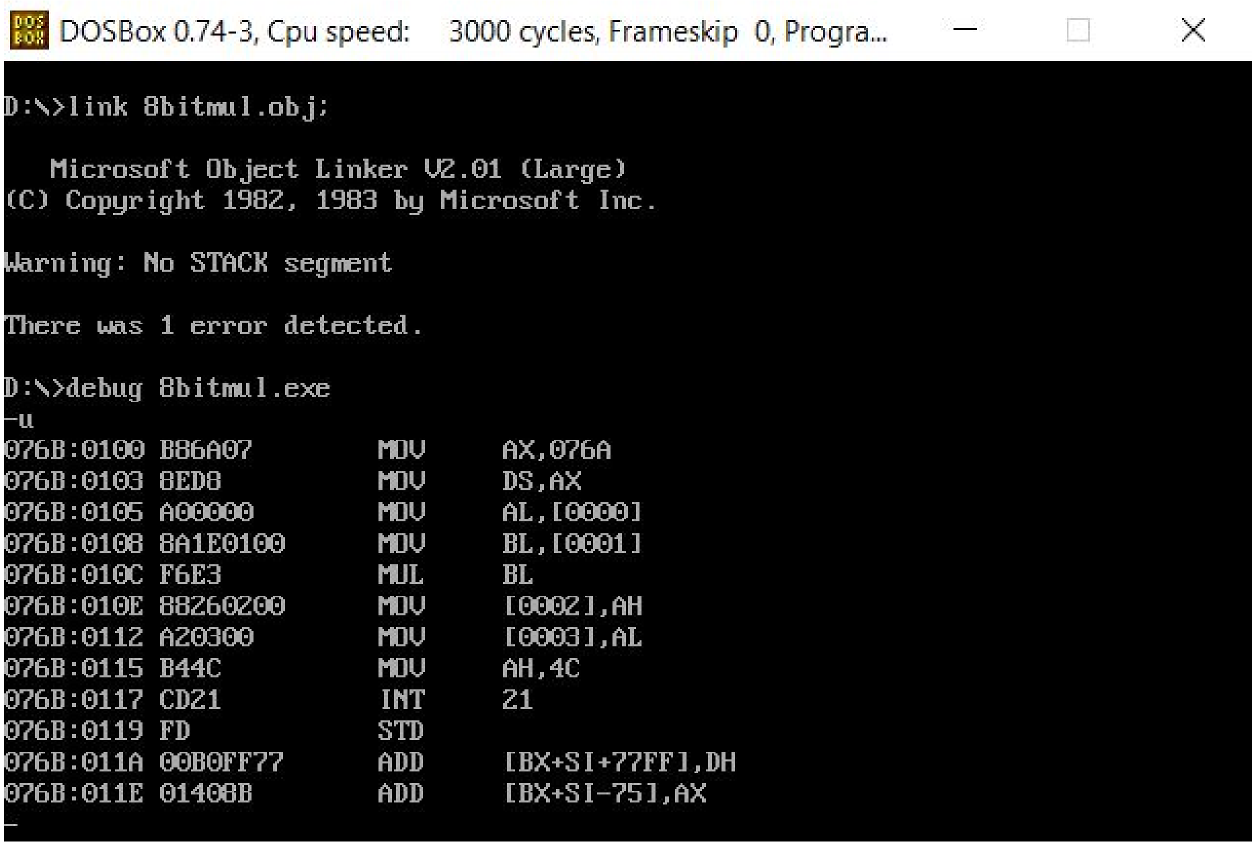
Snapshot of Input and Output

Input

Two 8-bit values

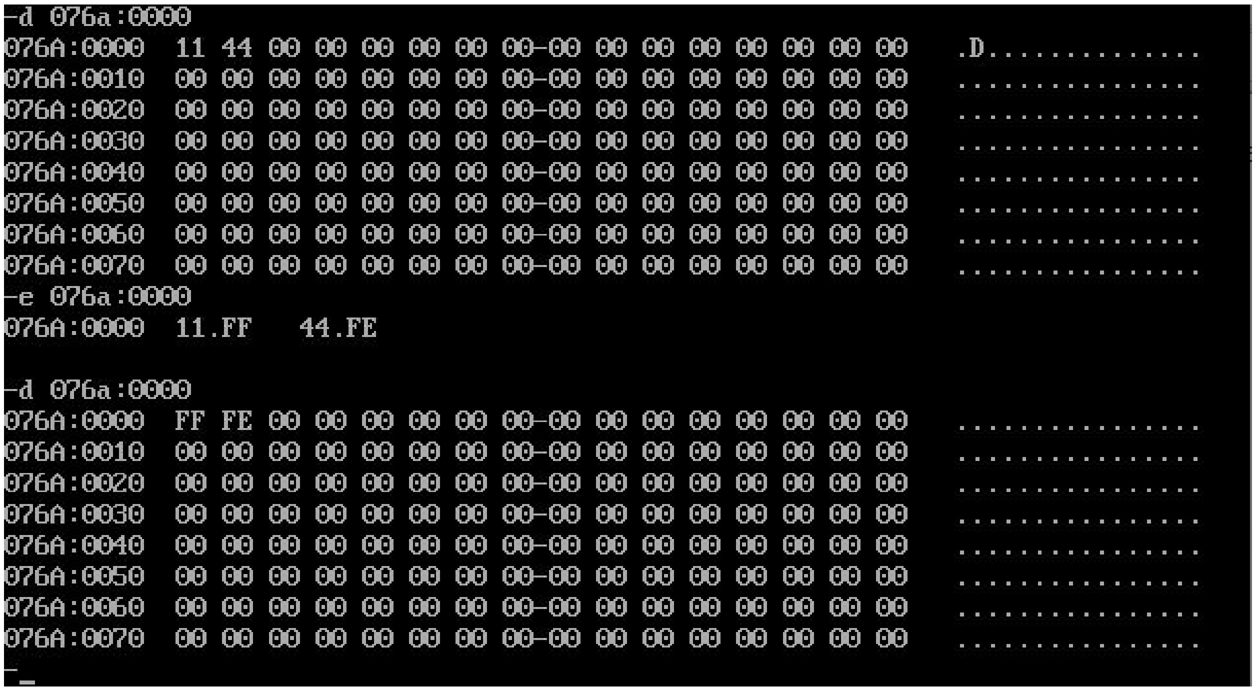
Output

Product in 16 bits

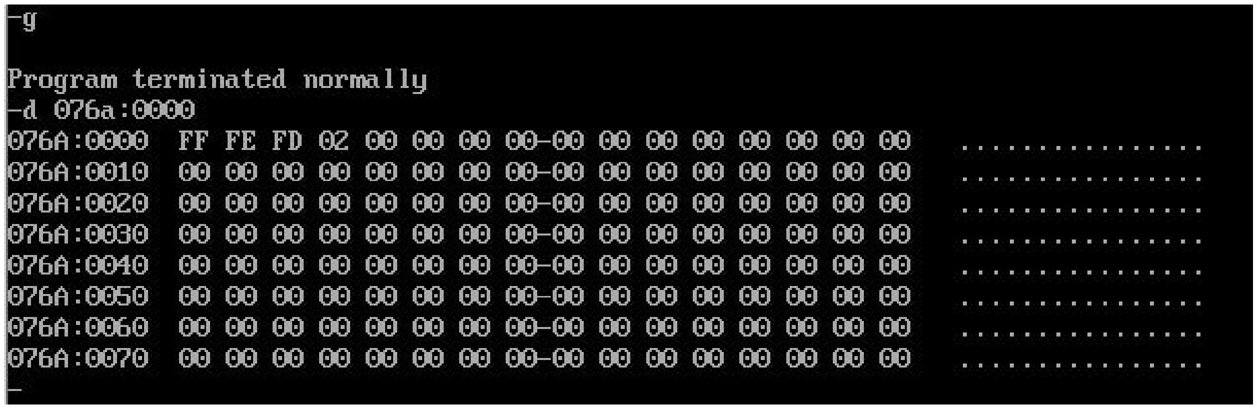


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
D:\>link 8bitmul.obj;

   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

D:\>debug 8bitmul.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A00000        MOV     AL,[0000]
076B:0108 8A1E0100      MOV     BL,[0001]
076B:010C F6E3          MUL     BL
076B:010E 88260200      MOV     [0002],AH
076B:0112 A20300        MOV     [0003],AL
076B:0115 B44C          MOV     AH,4C
076B:0117 CD21          INT     21
076B:0119 FD            STD
076B:011A 00B0FF77      ADD     [BX+SI+77FF],DH
076B:011E 01408B        ADD     [BX+SI-75],AX
-
```



```
-d 076a:0000
076A:0000  11 44 00 00 00 00 00 00-00 00 00 00 00 00 00 00   .D..............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-e 076a:0000
076A:0000  11.FF   44.FE

-d 076a:0000
076A:0000  FF FE 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```



```
-g

Program terminated normally
-d 076a:0000
076A:0000  FF FE FD 02 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Program for dividing two 8-bit numbers

Program Name: 8BITDIV.ASM

Program	Comments
;Program for dividing a 16-bit number by an ;8-bit number ASSUME CS:CODE,DS:DATA	Naming the CS and DS for the program

DATA SEGMENT
        OPR1 DB 11H
        OPR2 DB 99H
	Declaring and initializing the values for the two operands in the data segment

QUOTIENT DB 00H REMINDER DB 00H DATA ENDS	The quotient and remainder must be set to zero to obtain the correct result and get rid off garbage values
CODE SEGMENT ORG 0100H	Providing an offset value

START:
        MOV AX, DATA
        MOV DS, AX
	Initializing the data segment register with the data segment address

MOV BL, OPR1 MOV AL, OPR2	Loading the data into the appropriate registers
MOV AH, 00H	Setting the higher order bits to zero
DIV BL	Performing division: AX = AX / BL
MOV REMINDER, AH MOV QUOTIENT, AL	Since the quotient and remainder are stored in the AX register, the higher and lower order bits are extracted

MOV AH, 4CH
        INT 21H
        CODE ENDS
END START
	Calling the DOS Function to enter the display screen using interrupt 21H

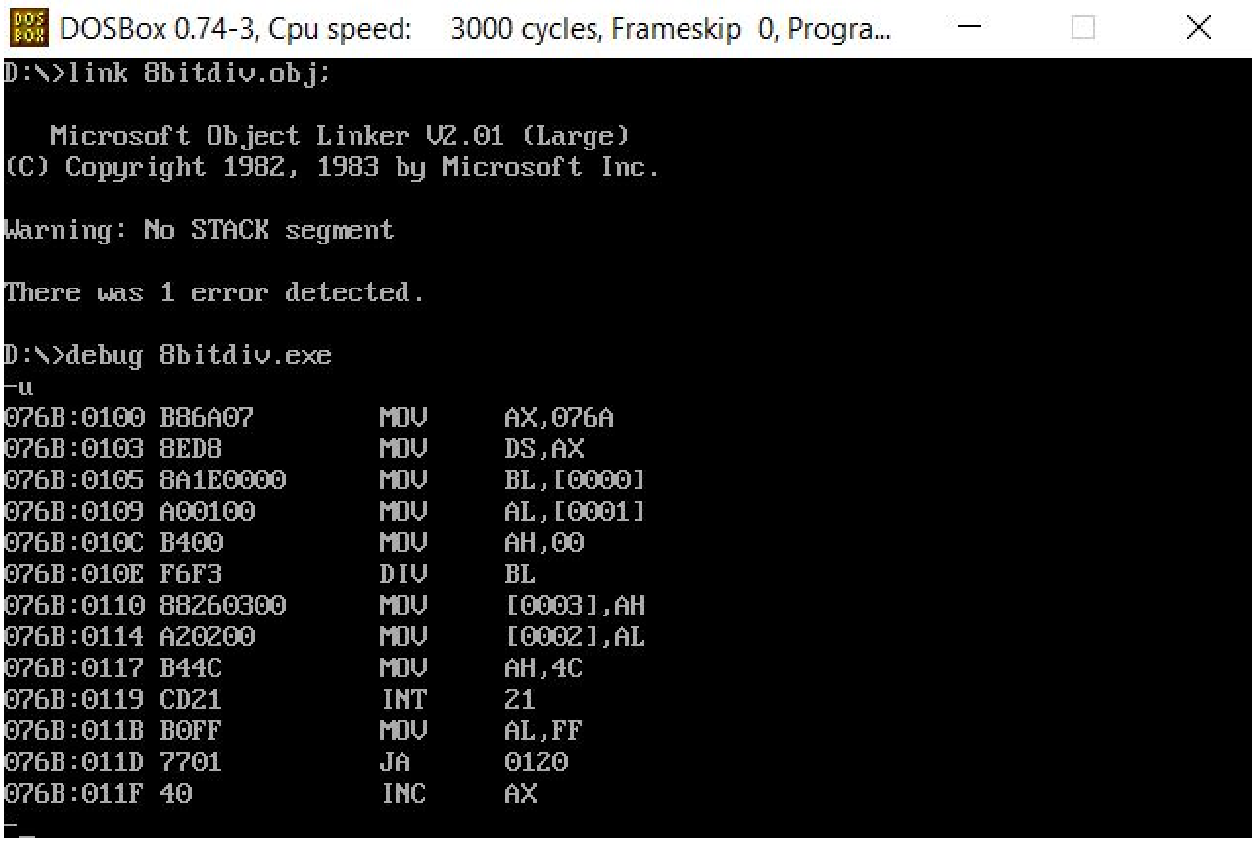
Snapshot of Input and Output

Input

Two eight bit values(**Note:** No dedicated instruction available in 8086 to perform 8 bit / 8 bit)

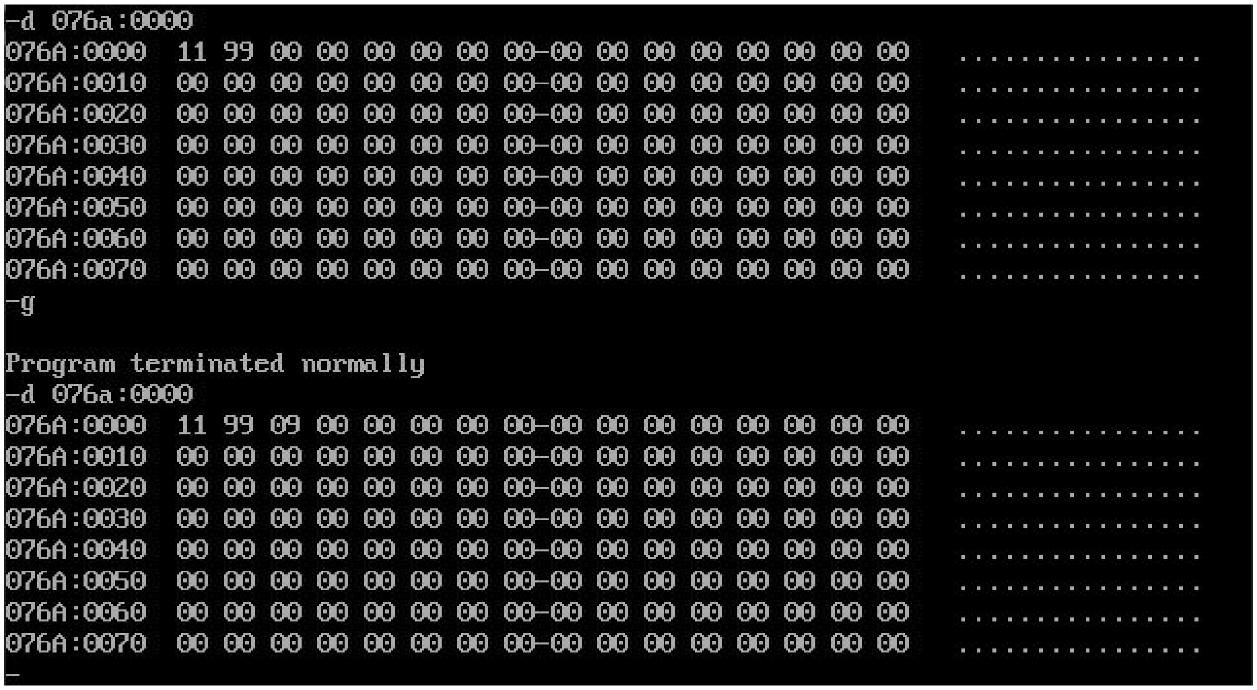
Output

Quotient in one memory location and the remainder in another


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

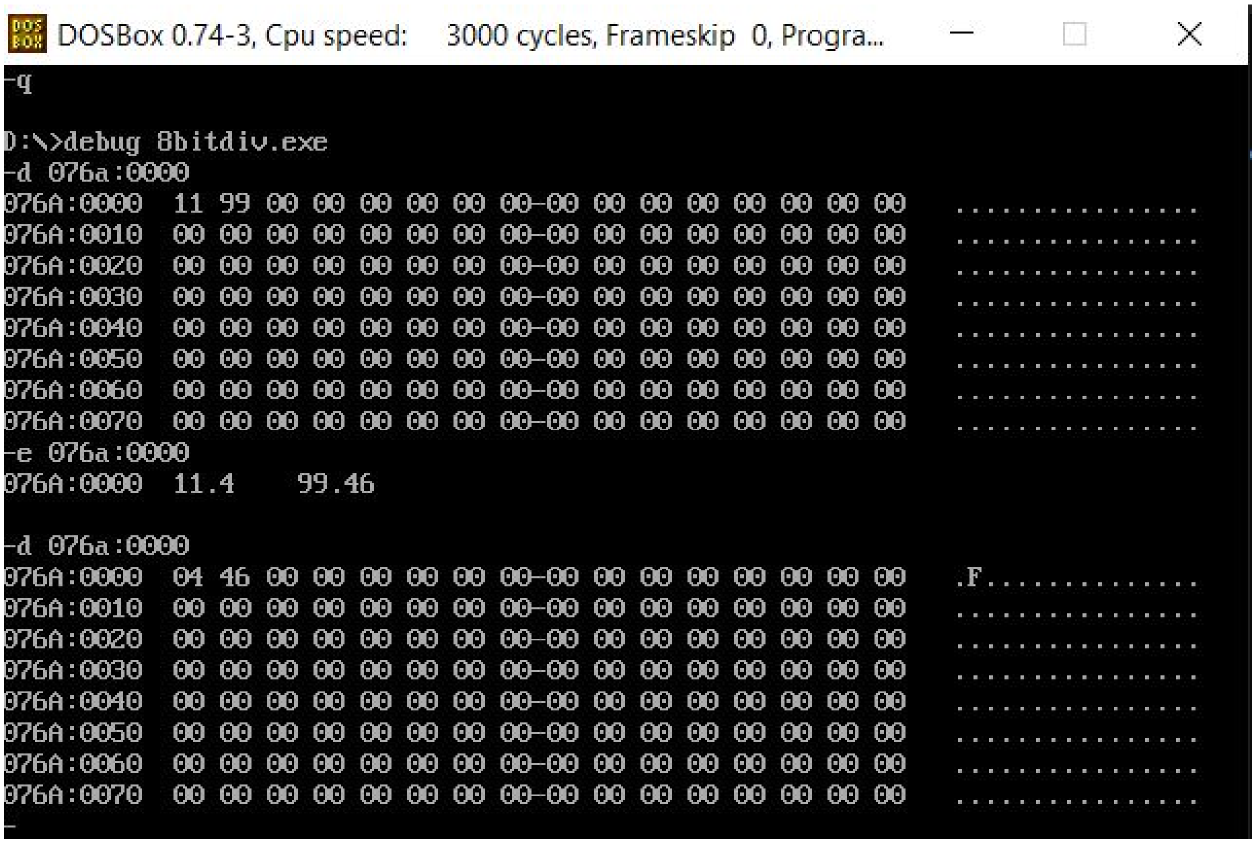
```
D:\>link 8bitdiv.obj;

   Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

D:\>debug 8bitdiv.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 8A1E0000      MOV     BL,[0000]
076B:0109 A00100        MOV     AL,[0001]
076B:010C B400          MOV     AH,00
076B:010E F6F3          DIV     BL
076B:0110 88260300      MOV     [0003],AH
076B:0114 A20200        MOV     [0002],AL
076B:0117 B44C          MOV     AH,4C
076B:0119 CD21          INT     21
076B:011B B0FF          MOV     AL,FF
076B:011D 7701          JA      0120
076B:011F 40            INC     AX
-_
```

Without reminder

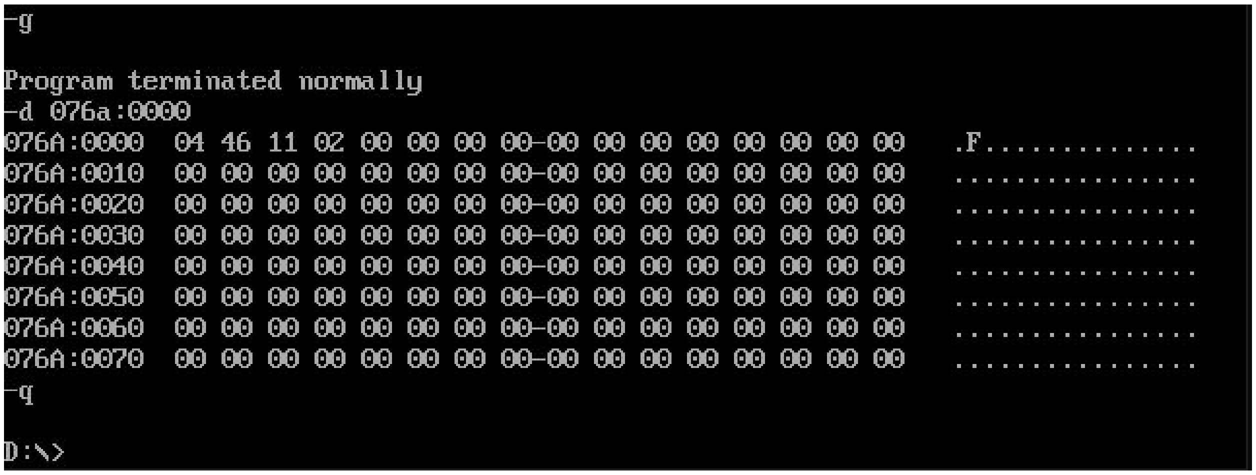


```
-d 076a:0000
076A:0000  11 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076a:0000
076A:0000  11 99 09 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

With reminder


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

```
-q

D:\>debug 8bitdiv.exe
-d 076a:0000
076A:0000  11 99 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-e 076a:0000
076A:0000  11.4    99.46

-d 076a:0000
076A:0000  04 46 00 00 00 00 00 00-00 00 00 00 00 00 00 00   .F..............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```



```
-g

Program terminated normally
-d 076a:0000
076A:0000  04 46 11 02 00 00 00 00-00 00 00 00 00 00 00 00   .F..............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-q

D:\>
```

Result

Programs for performing 8-bit arithmetic operations in an 8086 microprocessor using MASM and DOSBox were implemented and the outputs were verified.