SSN College of Engineering

Department of Computer Science and Engineering

UCS1512 – Microprocessors Lab

16 BIT ARITHMETIC OPERATIONS

Experiment Number: 2	Name: **S Saikrishnan**
Date: 25th August 2020	Register Number:**185001133**
Semester: V	Batch: 2018-2022

AIM:

To write and execute 8086 programs for arithmetic operations of 16 bit numbers like addition, subtraction, multiplication and division.

PROCEDURE:

- Firstly, write the 8086 program for 16 bit Addition using editor(like notepad) and save it with .asm(16bitadd.asm) extension and move to the MASM folder.
- Now mount the MASM folder in DOSBOX(“mount d e:/masm”) and then enter into the mounted drive(“d:”)
- Now using “edit 16bitadd.asm”,we can edit or create a asm file for execution and then save and exit.
- Assemble the code using “masm 16bitadd.asm” to generate the “16bitadd.obj” file.
- Link the file using “link 16bitadd.obj;” to generate the executable “16bitadd.exe” file.
- Now enter the debug mode using “debug 16bitadd.exe” to execute and analyse memory contents. The various commands used in debug mode are as follows:-
    - U :- To display unassembled code
    - D :- Used as ‘D segment:offset’ to see the content of memory locations starting from segment:offset address.
    - E:- To change the value in memory
    - G:- To execute
    - Q:- To quit

16 BIT ADDITION:

ALGORITHM:

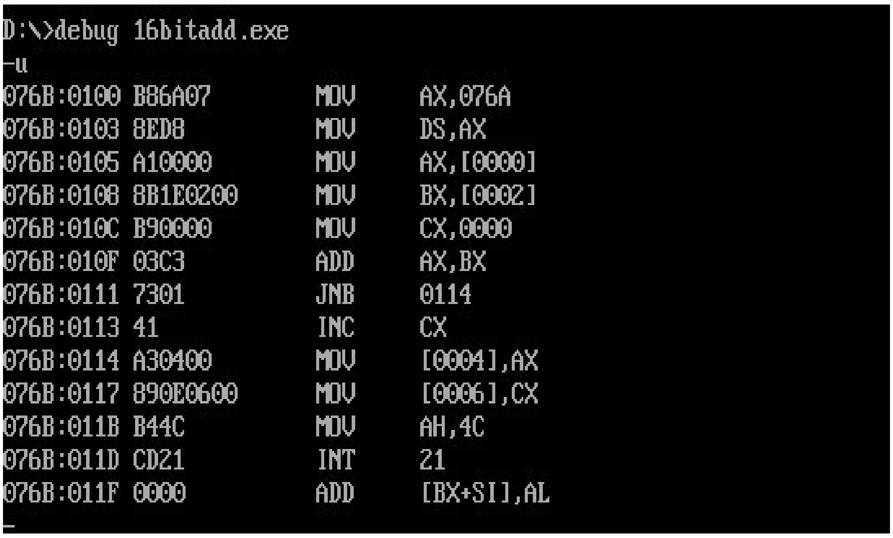
- Move the data segment to the AX register and then move it to the DS register.
- Move the first operand to AX register.
- Move the second operand to the BX register
- Initially set the CX register to 0000h.
- Then add using **ADD AX,BX**.
- Using JNC instruction check for carry and if there is no carry, no need to increment CX.
- Else, increment CX by 1.
- The result and carry stored in AX and CX should be moved to RESULT and CARRY respectively.

PROGRAM:

PROGRAM	**COMMENTS**
mov ax,opr1	Transfers operand 1(opr1's) value to AX register.
mov bx,opr2	Transfers operand 2(opr2's) value to BX register.
mov cx,0000h	Initialises CX register with 0000h.
add ax,bx	AX=AX+BX.
jnc here	Jumps to 'Here' Label if no carry i.e., if carry==0,jump to 'Here'.
inc cx	CX=CX+1(Increments CX by 1).
here: mov result,ax	Transfers AX register's data to RESULT.
mov carry,cx	Transfers CX register's data to CARRY.
mov ah,4ch int 21h	Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).

SNAPSHOT:

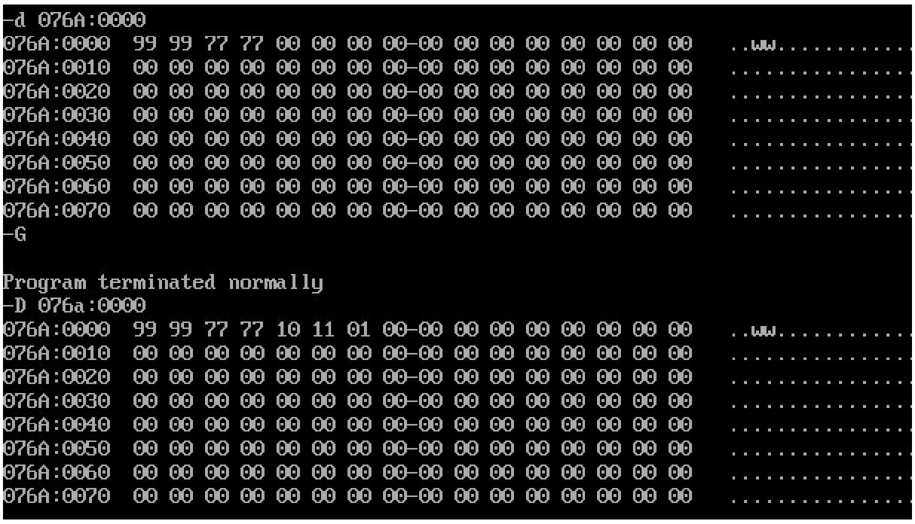
UNASSEMBLED CODE:



```
D:\>debug 16bitadd.exe
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A10000        MOV     AX,[0000]
076B:0108 8B1E0200      MOV     BX,[0002]
076B:010C B90000        MOV     CX,0000
076B:010F 03C3          ADD     AX,BX
076B:0111 7301          JNB     0114
076B:0113 41            INC     CX
076B:0114 A30400        MOV     [0004],AX
076B:0117 890E0600      MOV     [0006],CX
076B:011B B44C          MOV     AH,4C
076B:011D CD21          INT     21
076B:011F 0000          ADD     [BX+SI],AL
-
```

Input : opr1=9999 opr2=7777

Output: Result:1110 Carry: 0001

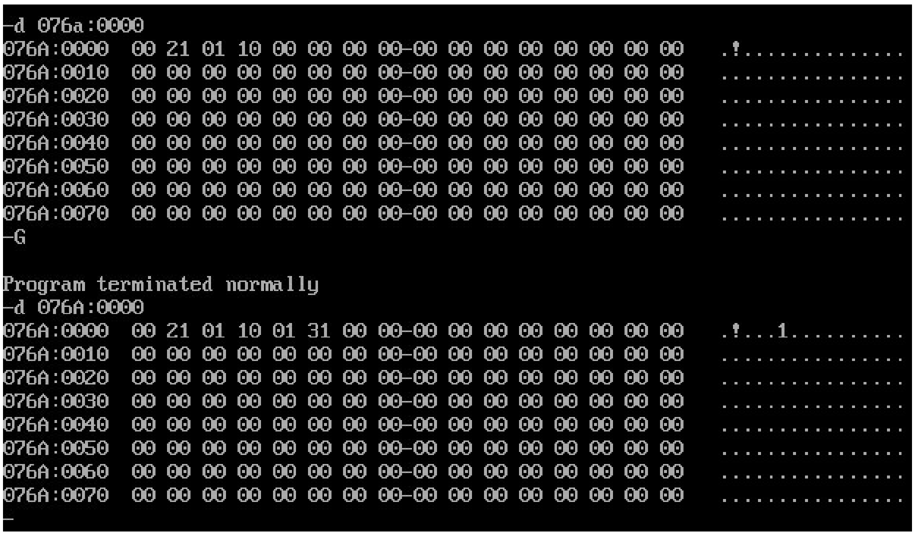


```
-d 076A:0000
076A:0000  99 99 77 77 00 00 00 00-00 00 00 00 00 00 00 00   ..ww............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 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  99 99 77 77 10 11 01 00-00 00 00 00 00 00 00 00   ..ww............
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   ................
```

Input : opr1=2100 opr2=1001

Output: Result:3101 Carry: 0000



```
-d 076a:0000
076A:0000  00 21 01 10 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  00 21 01 10 01 31 00 00-00 00 00 00 00 00 00 00   .!...1..........
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   ................
-
```

16 BIT SUBTRACTION:

ALGORITHM:

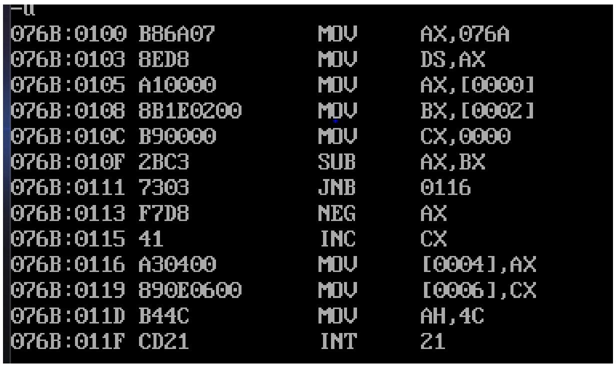
- Move the data segment to the AX register and then move it to the DS register.
- Move the first operand to AX register.
- Move the second operand to the BX register.
- Initially set the CX register to 0000h.
- Then subtract using **SUB AX,BX**.
- Check for carry using JNC instruction. If no carry then it means AX > BX and hence no need to increment CX and no need to complement AX.
- Else, AX<BX. Hence we have to take 2’s complement of AX using NEG AX and also increment CX by 1 using INC CX.
- The result and carry stored in AX and CX should be moved to RESULT and CARRY respectively.

PROGRAM:

PROGRAM	COMMENTS
mov ax,opr1 mov bx,opr2 mov cx,0000h sub ax,bx jnc here neg ax inc cx here: mov result,ax mov carry,cx	Transfers operand 1(opr1’s) value to AX register. Transfers operand 2(opr2’s) value to BX register. Initialises CX register with 0000h. AX=AX-BX. Jumps to ‘Here’ Label if no carry i.e., if carry==0,jump to ‘Here’. AX=2’s complement(AX) CX=CX+1(Increments CX by 1). Transfers AX register’s data to RESULT. Transfers CX register’s data to CARRY.
mov ah,4ch int 21h	Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).

SNAPSHOT:

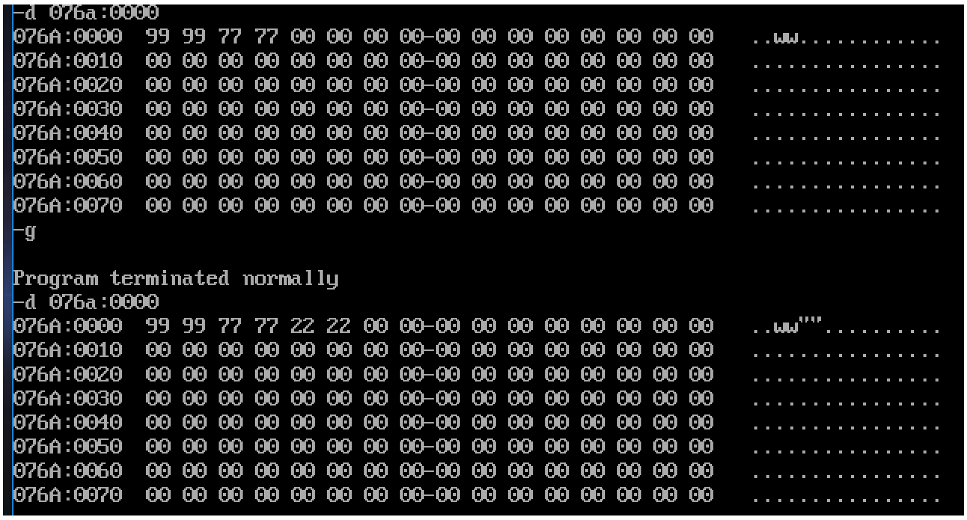
UNASSEMBLED CODE:



```
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A10000        MOV     AX,[0000]
076B:0108 8B1E0200      MOV     BX,[0002]
076B:010C B90000        MOV     CX,0000
076B:010F 2BC3          SUB     AX,BX
076B:0111 7303          JNB     0116
076B:0113 F7D8          NEG     AX
076B:0115 41            INC     CX
076B:0116 A30400        MOV     [0004],AX
076B:0119 890E0600      MOV     [0006],CX
076B:011D B44C          MOV     AH,4C
076B:011F CD21          INT     21
```

Input : opr1=9999 opr2=7777

Output: Result:2222 Carry:0000

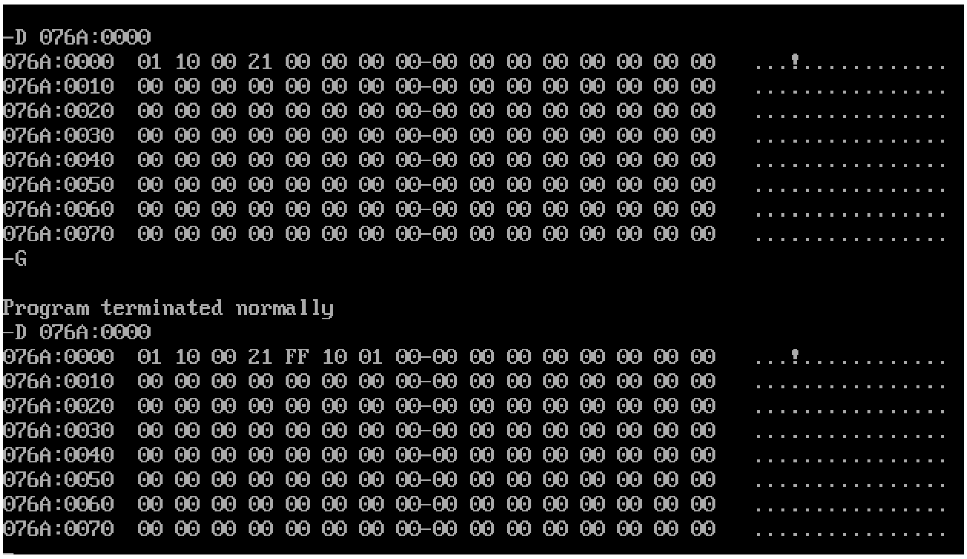


```
-d 076a:0000
076A:0000  99 99 77 77 00 00 00 00-00 00 00 00 00 00 00 00   ..ww............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 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  99 99 77 77 22 22 00 00-00 00 00 00 00 00 00 00   ..ww""..........
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   ................
```

Input : opr1=1001 opr2=2100

Output: Result: 10FF Carry: 0001



```
-D 076A:0000
076A:0000  01 10 00 21 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  01 10 00 21 FF 10 01 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   ................
```

16 BIT MULTIPLICATION:

ALGORITHM:

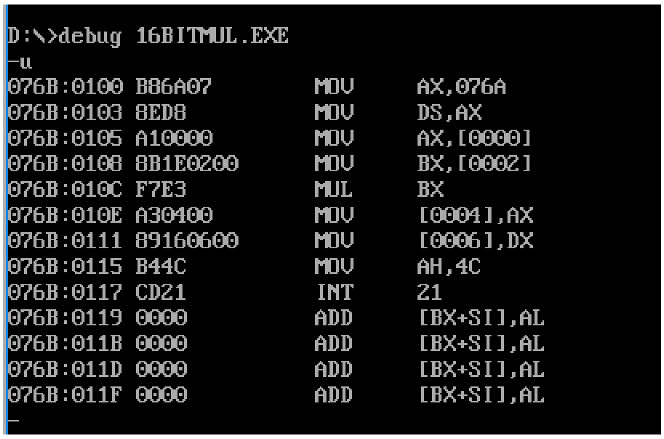
- Move the data segment to the AX register and then move it to the DS register.
- Move the first operand to AX register.
- Move the second operand to the BX register.
- Then multiply using **MUL BX.**(Since AX is default operand register for MUL instruction we only need to specify the other operand register.)
- The lower order and higher order result bits stored in AX and DX are now to be transferred to RESULT1 & RESULT2 respectively.

PROGRAM:

PROGRAM	COMMENTS
mov ax,opr1 mov bx,opr2 mul bx mov result1,ax mov result2,dx	Transfers operand 1(opr1's) value to AX register. Transfers operand 2(opr2's) value to BX register. BX=BX*AX. Transfers AX register's data to RESULT1. Transfers DX register's data to RESULT2.
mov ah,4ch int 21h	Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).

SNAPSHOT:

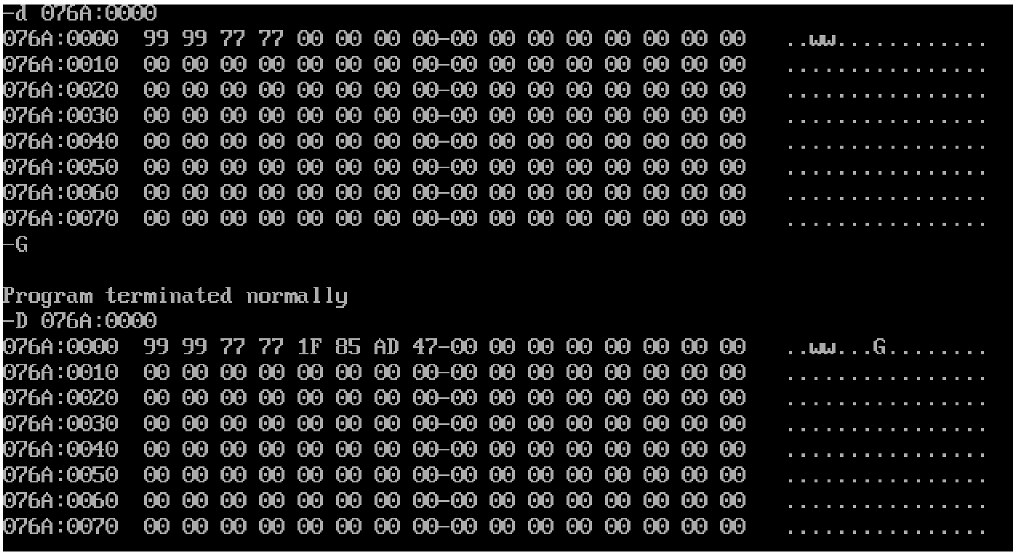
UNASSEMBLED CODE:



```
D:\>debug 16BITMUL.EXE
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 A10000        MOV     AX,[0000]
076B:0108 8B1E0200      MOV     BX,[0002]
076B:010C F7E3          MUL     BX
076B:010E A30400        MOV     [0004],AX
076B:0111 89160600      MOV     [0006],DX
076B:0115 B44C          MOV     AH,4C
076B:0117 CD21          INT     21
076B:0119 0000          ADD     [BX+SI],AL
076B:011B 0000          ADD     [BX+SI],AL
076B:011D 0000          ADD     [BX+SI],AL
076B:011F 0000          ADD     [BX+SI],AL
-
```

Input : opr1=9999 opr2=7777

Output: Result: 47AD851F[Result2: 47AD Result1: 851F]

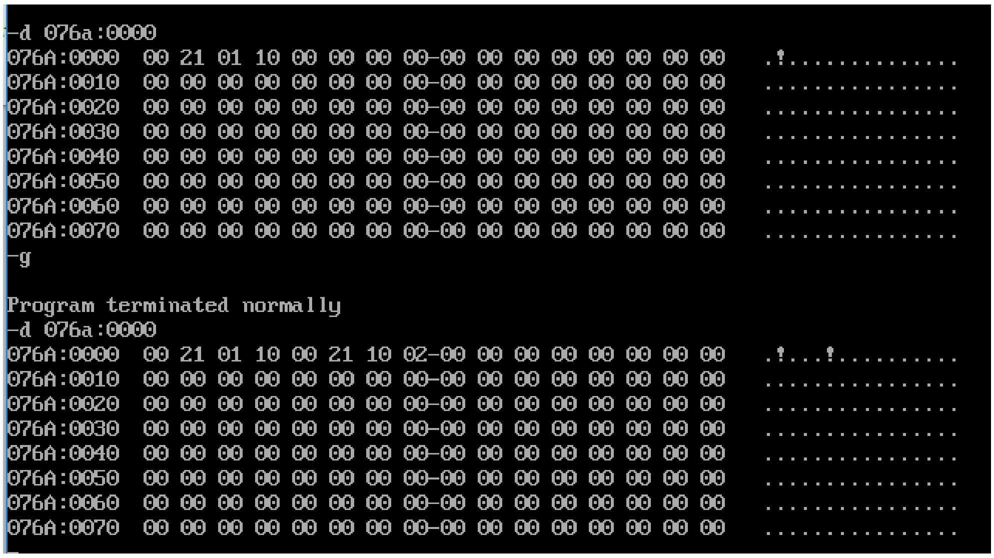


```
-d 076A:0000
076A:0000  99 99 77 77 00 00 00 00-00 00 00 00 00 00 00 00   ..ww............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 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  99 99 77 77 1F 85 AD 47-00 00 00 00 00 00 00 00   ..ww...G........
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   ................
```

Input   : opr1=2100 opr2=1011

Output: Result: 02102100 [Result2: 0210 Result1: 2100]



```
-d 076a:0000
076A:0000  00 21 01 10 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  00 21 01 10 00 21 10 02-00 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   ................
```

16 BIT DIVISION:

ALGORITHM:

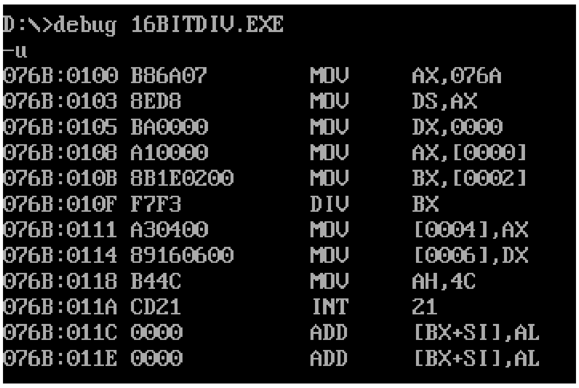
- Move the data segment to the AX register and then move it to the DS register.
- Now, set DX register to 0000h and move first operand to AX register.(Since we can’t directly divide a 16 bit number by 16 bit number in 8086, we now make our dividend 32 bit by storing 0000h in DX register and the 16-bit operand 1 in AX register).
- Move the second operand to the BX register.
- Now divide using **DIV BX.** (It will perform DXAX / BX. Because DX is 0000h, what actually happens is the division of a 32 bit number by a 16 bit number.)
- The quotient and remainder stored in AX and DX should be moved to QUOTIENT and REMAINDER respectively.

PROGRAM:

PROGRAM	**COMMENTS**
mov dx,0000h mov ax,opr1 mov bx,opr2 div bx mov quotient,ax mov remainder,dx	Move the value 0000h to DX register. Transfers operand 1(opr1’s) value to AX register. Transfers operand 2(opr2’s) value to BX register. Performs DXAX/BX. Transfers AX register’s data to QUOTIENT. Transfers DX register’s data to REMAINDER,.
mov ah,4ch int 21h	Moves the hexadecimal value 4c to ah. When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).

SNAPSHOT:

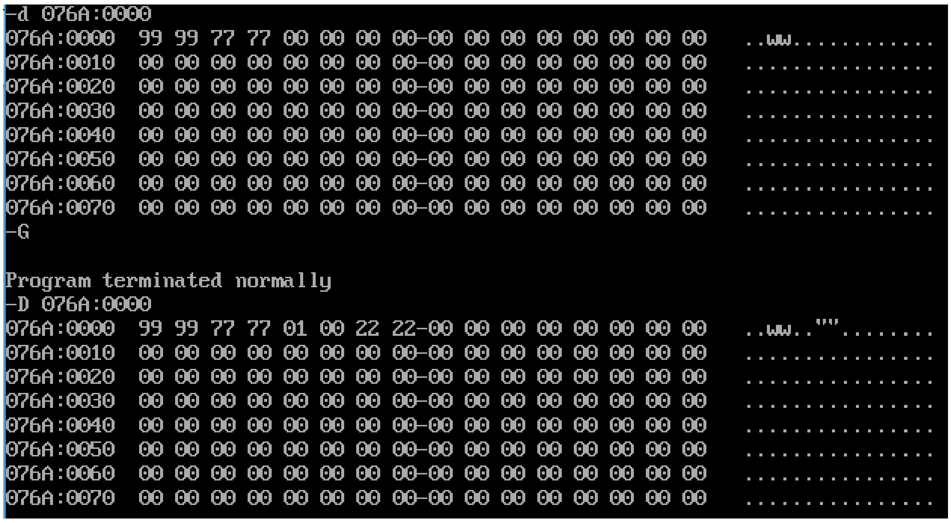
UNASSEMBLED CODE:



```
D:\>debug 16BITDIV.EXE
-u
076B:0100 B86A07        MOV     AX,076A
076B:0103 8ED8          MOV     DS,AX
076B:0105 BA0000        MOV     DX,0000
076B:0108 A10000        MOV     AX,[0000]
076B:010B 8B1E0200      MOV     BX,[0002]
076B:010F F7F3          DIV     BX
076B:0111 A30400        MOV     [0004],AX
076B:0114 89160600      MOV     [0006],DX
076B:0118 B44C          MOV     AH,4C
076B:011A CD21          INT     21
076B:011C 0000          ADD     [BX+SI],AL
076B:011E 0000          ADD     [BX+SI],AL
```

Input : opr1=9999 opr2=7777

Output: Quotient:0001 Remainder:2222

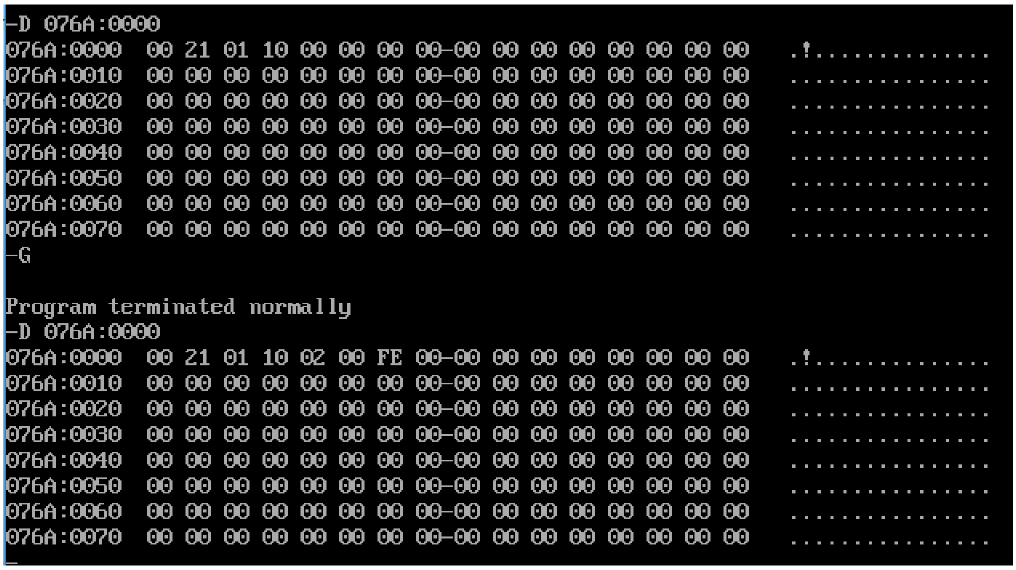


```
-d 076A:0000
076A:0000  99 99 77 77 00 00 00 00-00 00 00 00 00 00 00 00   ..ww............
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 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  99 99 77 77 01 00 22 22-00 00 00 00 00 00 00 00   ..ww..""........
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   ................
```

Input : opr1=1001 opr2=2100

Output: Quotient:0002 Remainder:00FE



```
-D 076A:0000
076A:0000  00 21 01 10 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  00 21 01 10 02 00 FE 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   ................
-
```

RESULT:

Thus,8086 programs for arithmetic operations of 16 bit numbers like addition, subtraction, multiplication and division have been executed successfully using MS - DOSBox.