Aravind Kannan Rathinasabapathi Date: 23-07-2021
195001022 Ex. No. 2

Exercise 2 – 16-bit arithmetic operations
2A – 16-bit addition

Aim:

To add two 16-bit numbers

Procedure for executing MASM:

1. Mount the local folder in the DOS-BOX using a temp disk name: `mount <disk-name> <folder-location>`
2. Change directory into the mounted disk: `<disk-name>: `
3. Assemble the instructions: `masm <file-name>.asm`
4. Link the object file(s) to produce an executable file(.exe): `link <file-name>.obj; ` Note that removal of semi-colon will make linking process interactive.
5. Debug the executable file to read the memory map and execute the program: `debug <file-name>.exe`. After entering debug mode,
   a. `d <segment:offset> ` - dump(read) memory map from the given location
   b. `e <segment:offset> ` - edit memory values from the given location. Use 'White space' to continue editing and 'new line' to exit editing.
   c. `u ` - unassemble code (with or without <segment:offset>)
   d. `g ` - execute the program
   e. `? ` - display command list
   f. `q` - quit the debugger

Algorithm:

1. Declare and initialize the data segment.
2. Begin code segment, where actual assembler instructions are present.
3. Move the starting address of data segment into ds register.
4. Store the augend in ax and addend in bx.
5. Initialize ch to be *00H*, to process carry.
6. Add ax and bx: ax = ax + bx
7. If carry generated, increment ch, else jump ahead of it.
8. Store ax in sum, ch in carry.
9. Terminate program and code segment.

Program:

Program	Comments
`;Program to add two 16-bit numbers`	Comment after ';'
`assume cs:code,ds:data`	Map CS to code segment and DS to data segment

`data segment`
`    augend dw 0f209H`
`    addend dw 130aH`
`    sum dw 0000H`
`    carry db 00H`
`data ends`
	Initialise data segment
	db = define a byte, dw = define a byte
	Initialise addend = 130A, augend = F209, sum = 0000, carry = 0000

`code segment`
`start:  mov ax, data`
`        mov ds, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds.
	Since in 8086, only code segment register is loaded automatically, the remaining segment register can be assigned using general purpose registers.

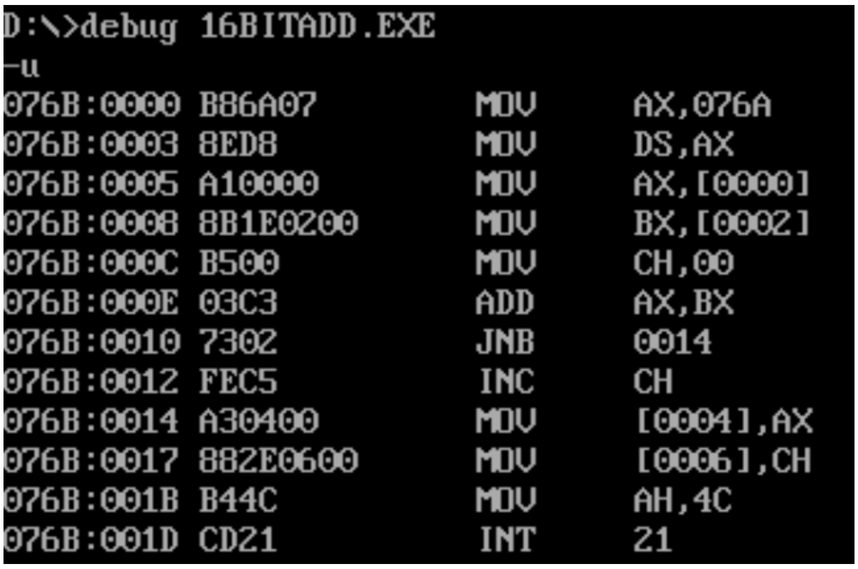
`        mov ax, augend`
`        mov bx, addend`
`        mov ch, 00H`
	Move augend to ax, addend to bx
	
	Initialise ch = 00H using immediate addressing mode

`        add ax, bx`
`        jnc noCarry`
`        inc ch`
	Add ax and bx: ax = ax + bx
	Jump if no carry to '*noCarry*' label
	Increment ch

`noCarry:mov sum, ax`
`        mov carry, ch`
	Move ax to sum
	Move ch to carry

`        mov ah, 4cH`
`        int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Unassembled code:



```
D:\>debug 16BITADD.EXE
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 A10000        MOV     AX,[0000]
076B:0008 8B1E0200      MOV     BX,[0002]
076B:000C B500          MOV     CH,00
076B:000E 03C3          ADD     AX,BX
076B:0010 7302          JNB     0014
076B:0012 FEC5          INC     CH
076B:0014 A30400        MOV     [0004],AX
076B:0017 882E0600      MOV     [0006],CH
076B:001B B44C          MOV     AH,4C
076B:001D CD21          INT     21
```

8086 follows little-endian notation. The lower half(8-bit) of the 16-bit register goes to lower memory address and upper half(8-bit) goes to higher memory address and vice-versa.

From memory to registers:

```
AL <- 0000 AH <- 0001
BL <- 0002 BH <- 0003
```

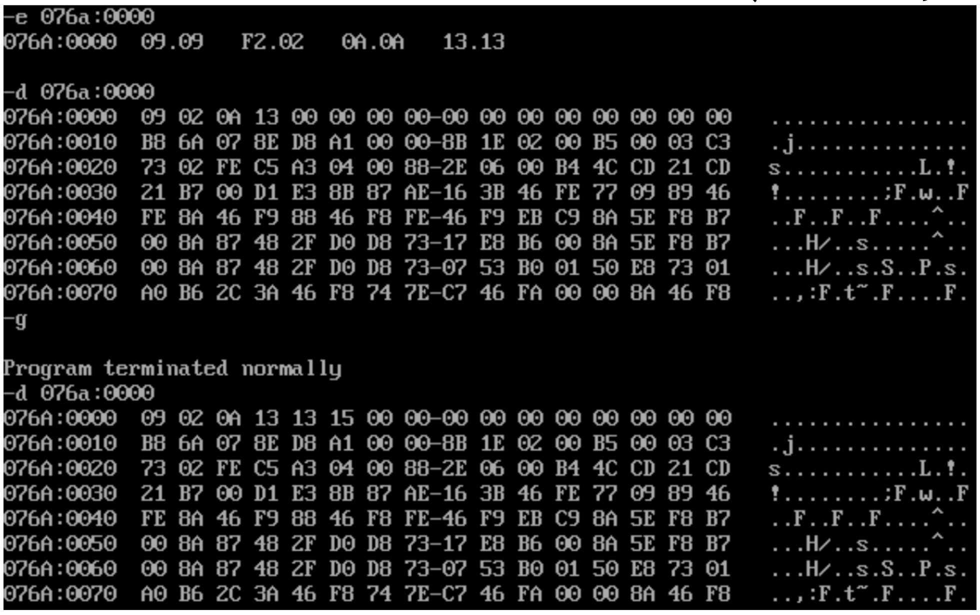
From registers to memory

```
0004 <- AL 0005 <- AH
0006 <- CH
```

Snapshot of sample input and output:

Case i: Without Carry

Hexadecimal addition – 0209 + 130A = 1513 (Sum: 1513, Carry: 00)

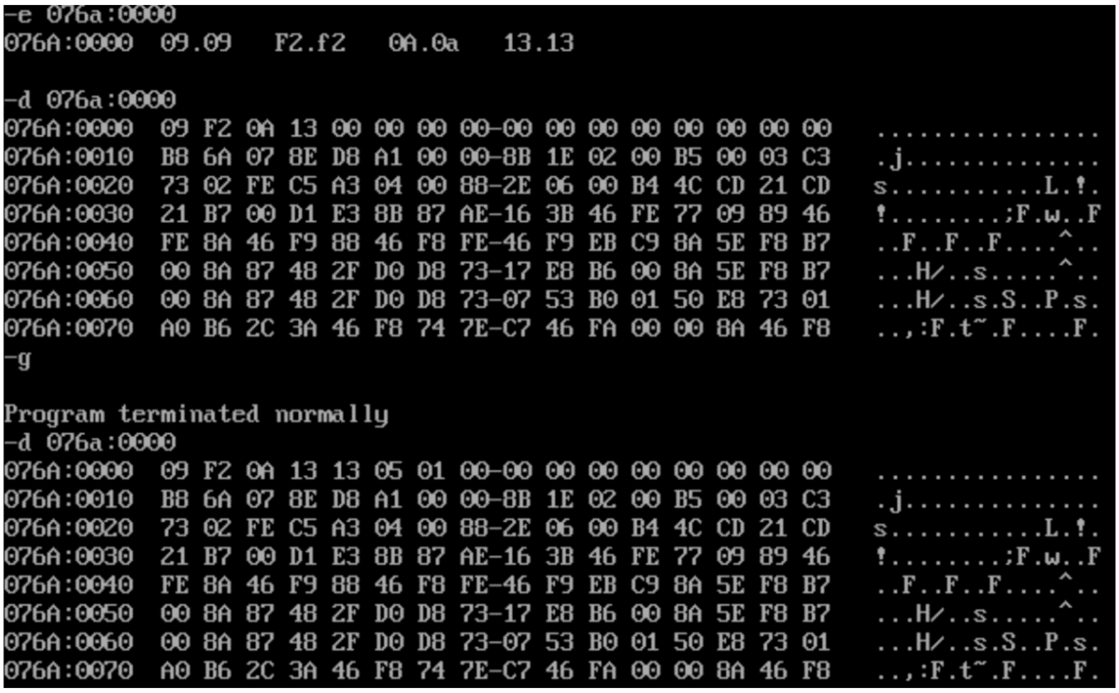


```
-e 076a:0000
076A:0000  09.09   F2.02   0A.0A   13.13

-d 076a:0000
076A:0000  09 02 0A 13 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 03 C3   .j..............
076A:0020  73 02 FE C5 A3 04 00 88-2E 06 00 B4 4C CD 21 CD   s...........L.!.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  09 02 0A 13 13 15 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 03 C3   .j..............
076A:0020  73 02 FE C5 A3 04 00 88-2E 06 00 B4 4C CD 21 CD   s...........L.!.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Case ii: With Carry
Hexadecimal addition - F209 + 130A = 01 0513 (Sum: 0513, Carry: 01)



```
-e 076a:0000
076A:0000  09.09   F2.f2   0A.0a   13.13

-d 076a:0000
076A:0000  09 F2 0A 13 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 03 C3   .j..............
076A:0020  73 02 FE C5 A3 04 00 88-2E 06 00 B4 4C CD 21 CD   s...........L.!.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  09 F2 0A 13 13 05 01 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 03 C3   .j..............
076A:0020  73 02 FE C5 A3 04 00 88-2E 06 00 B4 4C CD 21 CD   s...........L.!.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Result:
Program to add two 16-bit numbers assembled, executed and verified.

2B – 16-bit subtraction

Aim:

To subtract two 16-bit numbers

Algorithm:

1. Declare and initialize the data segment.
2. Begin code segment, where actual assembler instructions are present.
3. Move the starting address of data segment into ds register.
4. Store the minuend ax and subtrahend in bx.
5. Initialize ch to be *00H*, to process diff.
6. Subtract ax and bx: ax = ax - bx
7. If carry generated, increment sign to denote negative and take 2's complement of ah, else jump ahead of it.
8. Store ax in diff, ch in sign.
9. Terminate program and code segment.

Aravind Kannan Rathinasabapathi Date: 23-07-2021
195001022 Ex. No. 2

Program:

Program	Comments
`;Program to subtract two 16-bit numbers`	Comment after ‘;’
`assume cs:code,ds:data`	Map CS to code segment and DS to data segment

`data segment`
`    minuend dw 0f209H`
`    subtrahend dw 130aH`
`    diff dw 0000H`
`    sign db 00H`
`data ends`
	Initialise data segment
	db = define a byte, dw = define a word
	Initialise minuend = F209, subtrahend = 130A, diff = 0000, sign = 00

`code segment`
`start:  mov ax, data`
`        mov ds, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds.

`        mov ax, minuend`
`        mov bx, subtrahend`
	Move minuend to ax, subtrahend to bx

`        mov ch, 00H`	Initialise ch = 00H using immediate addressing mode

`        sub ax, bx`
`        jnc noCarry`
	Sub ax and bx: ax = ax - bx
	Jump if no carry to ‘*noCarry*’ label

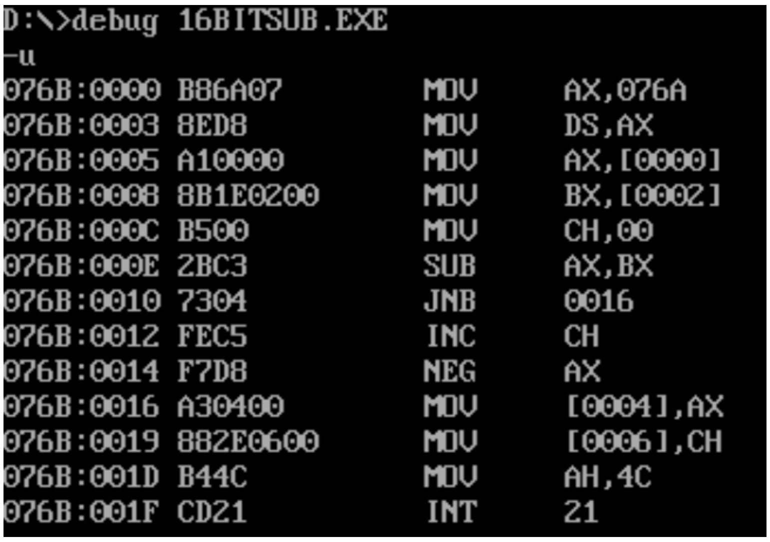
`        inc ch`
`        neg ax`
	Increment ch
	Negate – Take 2’s complement of ax

`noCarry:mov diff, ax`
`        mov sign, ch`
	Move ax to diff
	Move cx to sign

`        mov ah, 4cH`
`        int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Aravind Kannan Rathinasabapathi
195001022
Date: 23-07-2021
Ex. No. 2

Unassembled code:



```
D:\>debug 16BITSUB.EXE
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 A10000        MOV     AX,[0000]
076B:0008 8B1E0200      MOV     BX,[0002]
076B:000C B500          MOV     CH,00
076B:000E 2BC3          SUB     AX,BX
076B:0010 7304          JNB     0016
076B:0012 FEC5          INC     CH
076B:0014 F7D8          NEG     AX
076B:0016 A30400        MOV     [0004],AX
076B:0019 882E0600      MOV     [0006],CH
076B:001D B44C          MOV     AH,4C
076B:001F CD21          INT     21
```

8086 follows little-endian notation. The lower half(8-bit) of the 16-bit register goes to lower memory address and upper half(8-bit) goes to higher memory address and vice-versa.

From memory to registers:

```
      AL <- 0000 AH <- 0001
      BL <- 0002 BH <- 0003
```

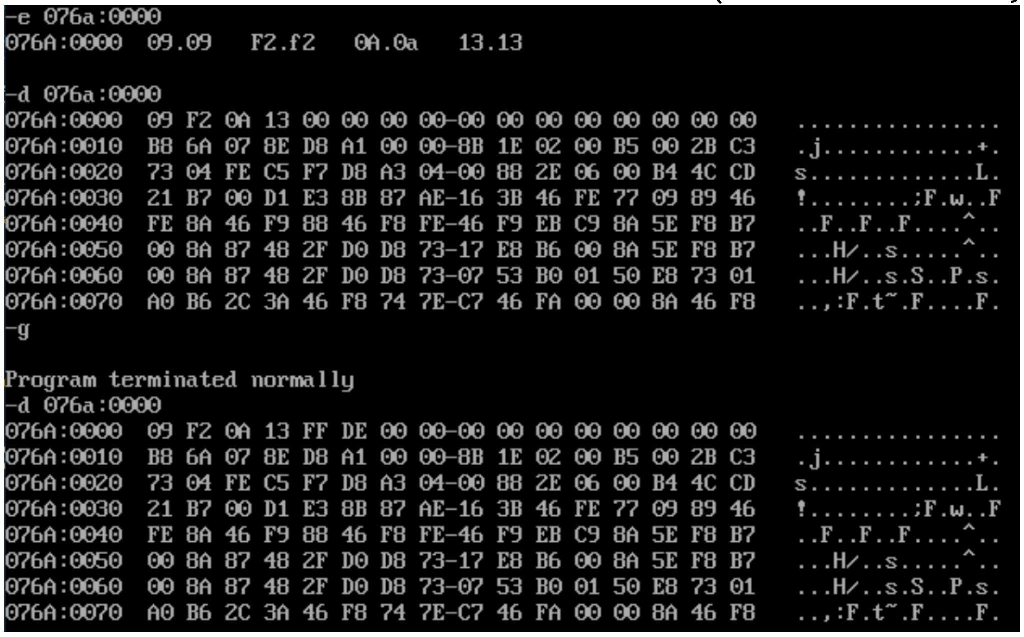
From registers to memory

```
      0004 <- AL 0005 <- AH
      0006 <- CH
```

Snapshot of sample input and output:

Case i: Minuend > Subtrahend = Positive difference

Hexadecimal subtraction – F209 – 130A = (Difference: DEFF, Sign: 00)

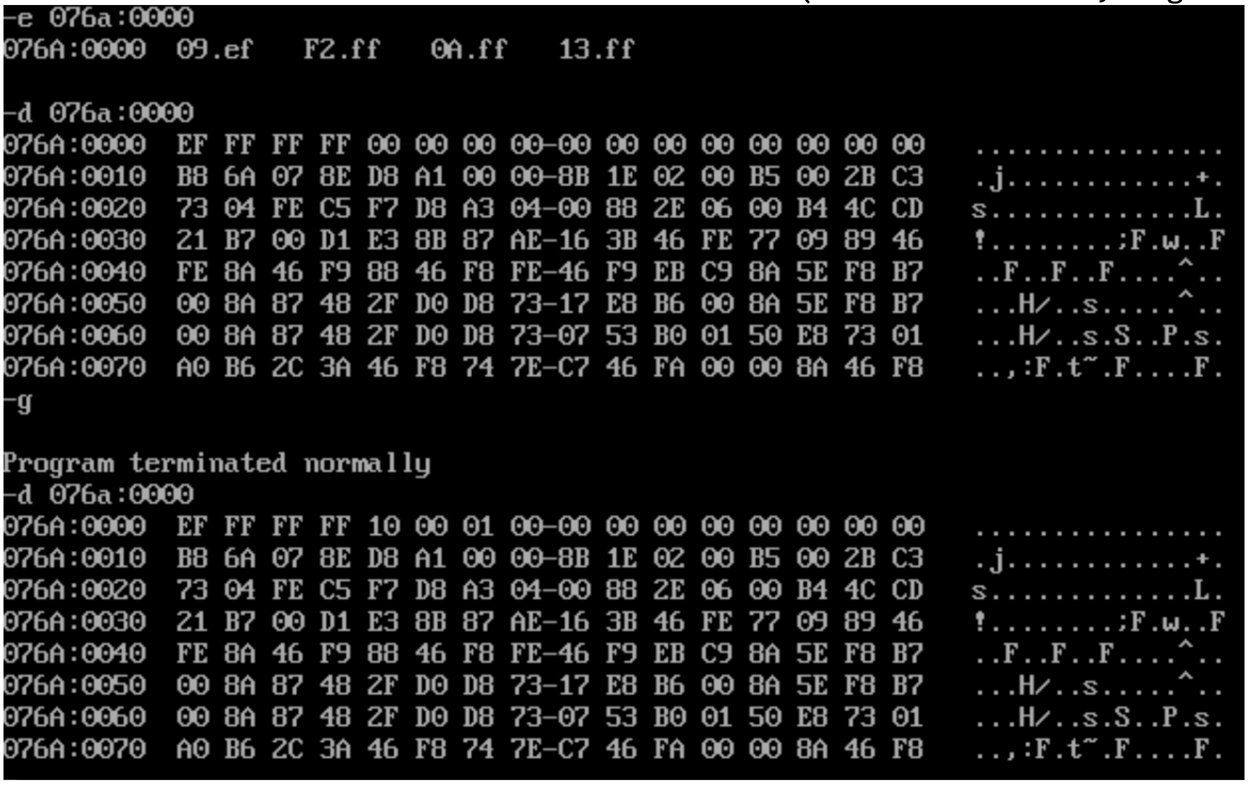


```
-e 076a:0000
076A:0000  09.09   F2.f2   0A.0a   13.13

-d 076a:0000
076A:0000  09 F2 0A 13 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 2B C3   .j............+.
076A:0020  73 04 FE C5 F7 D8 A3 04-00 88 2E 06 00 B4 4C CD   s.............L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  09 F2 0A 13 FF DE 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 2B C3   .j............+.
076A:0020  73 04 FE C5 F7 D8 A3 04-00 88 2E 06 00 B4 4C CD   s.............L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Case ii: Minuend < Subtrahend = Negative difference
Hexadecimal subtraction – FFEF – FFFF = FFF0 (Difference: 0010, Sign: 01)



```
-e 076a:0000
076A:0000  09.ef   F2.ff   0A.ff   13.ff

-d 076a:0000
076A:0000  EF FF FF FF 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 2B C3   .j............+.
076A:0020  73 04 FE C5 F7 D8 A3 04-00 88 2E 06 00 B4 4C CD   s.............L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  EF FF FF FF 10 00 01 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 B5 00 2B C3   .j............+.
076A:0020  73 04 FE C5 F7 D8 A3 04-00 88 2E 06 00 B4 4C CD   s.............L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Result:

Program to subtract two 16-bit numbers assembled, executed and verified.

2C – 16-bit multiplication

Aim:

To multiply two 16-bit numbers

Algorithm:

1. Declare and initialize the data segment.
2. Begin code segment, where actual assembler instructions are present.
3. Move the starting address of data segment into ds register.
4. Store the multiplicand in ax and multiplier in bx.
5. Multiply ax and bx: dx ax = ax * bx
6. Store ax in product_lower and dx in product_higher.
7. Terminate program and code segment.

Program:

Program	Comments
`;Program to multiply two 16-bit numbers`	Comment after ‘;’
`assume cs:code,ds:data`	Map CS to code segment and DS to data segment

`data segment`
`    multiplicand dw 0f209H`
`    multiplier dw 130aH`
`    product_lower dw 0000H`
`    product_upper dw 0000H`
`data ends`
	Initialise data segment
	db = define a byte, dw = define a word
	Initialise multiplicand = F209, multiplier = 130A, product_lower = 0000, product_upper = 0000

`code segment`
`start:  mov ax, data`
`        mov ds, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds.

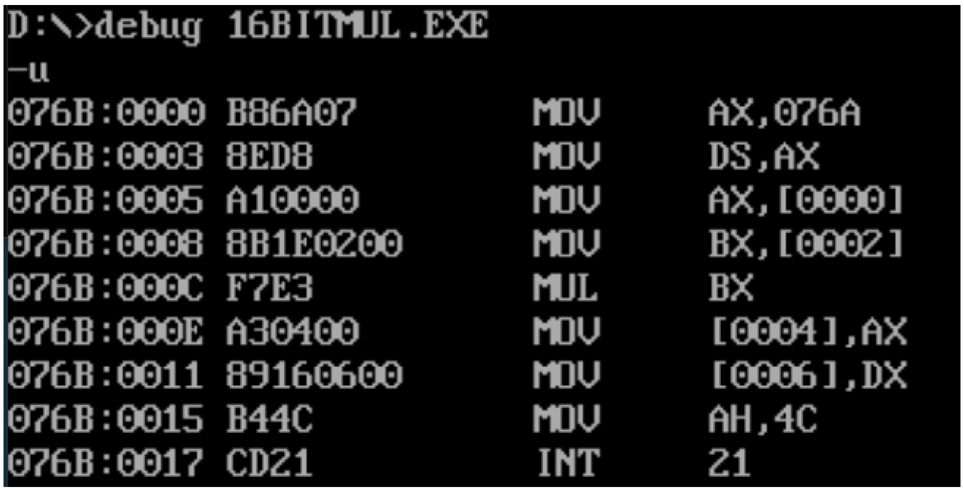
`        mov ax, multiplicand`
`        mov bx, multiplier`
	Move multiplicand to ax, multiplier to bx

`        mul bx`	Multiply ax and bx: ax = al * bl (Fixed instruction)

`        mov product_lower, ax`
`        mov product_upper, dx`
	Move ax as lower 16-bit to product_lower
	Move dx as higher 16-bit to product_upper

`        mov ah, 4cH`
`        int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Unassembled code:

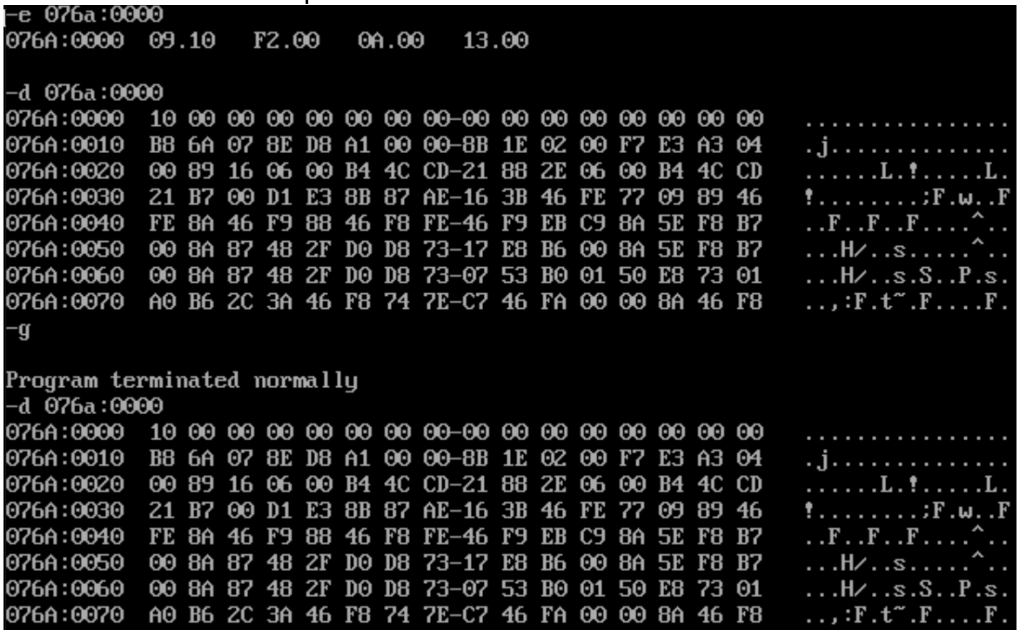


```
D:\>debug 16BITMUL.EXE
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 A10000        MOV     AX,[0000]
076B:0008 8B1E0200      MOV     BX,[0002]
076B:000C F7E3          MUL     BX
076B:000E A30400        MOV     [0004],AX
076B:0011 89160600      MOV     [0006],DX
076B:0015 B44C          MOV     AH,4C
076B:0017 CD21          INT     21
```

8086 follows little-endian notation. The lower half(8-bit) of the 16-bit register goes to lower memory address and upper half(8-bit) goes to higher memory address and vice-versa.
From memory to registers:
AL <- 0000 AH <- 0001
BL <- 0002 BH <- 0003
From registers to memory
0004 <- AL 0005 <- AH
0006 <- DL 0007 <- DH

Snapshot of sample input and output:

Case i: Multiplication with zero
Hexadecimal multiplication - 0010 x 0000 = 0000 0000

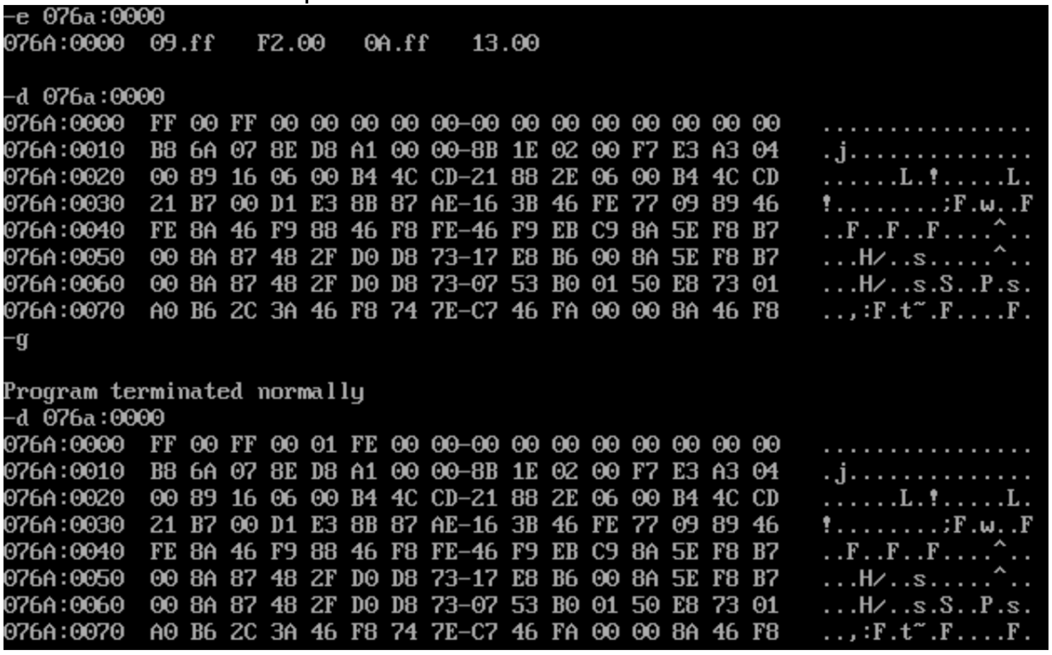


```
-e 076a:0000
076A:0000  09.10   F2.00   0A.00   13.00

-d 076a:0000
076A:0000  10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 F7 E3 A3 04   .j..............
076A:0020  00 89 16 06 00 B4 4C CD-21 88 2E 06 00 B4 4C CD   ......L.!.....L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  10 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 F7 E3 A3 04   .j..............
076A:0020  00 89 16 06 00 B4 4C CD-21 88 2E 06 00 B4 4C CD   ......L.!.....L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Case ii: Multiplication producing only ax
Hexadecimal multiplication – 00FF x 00FF = 0000 FE01

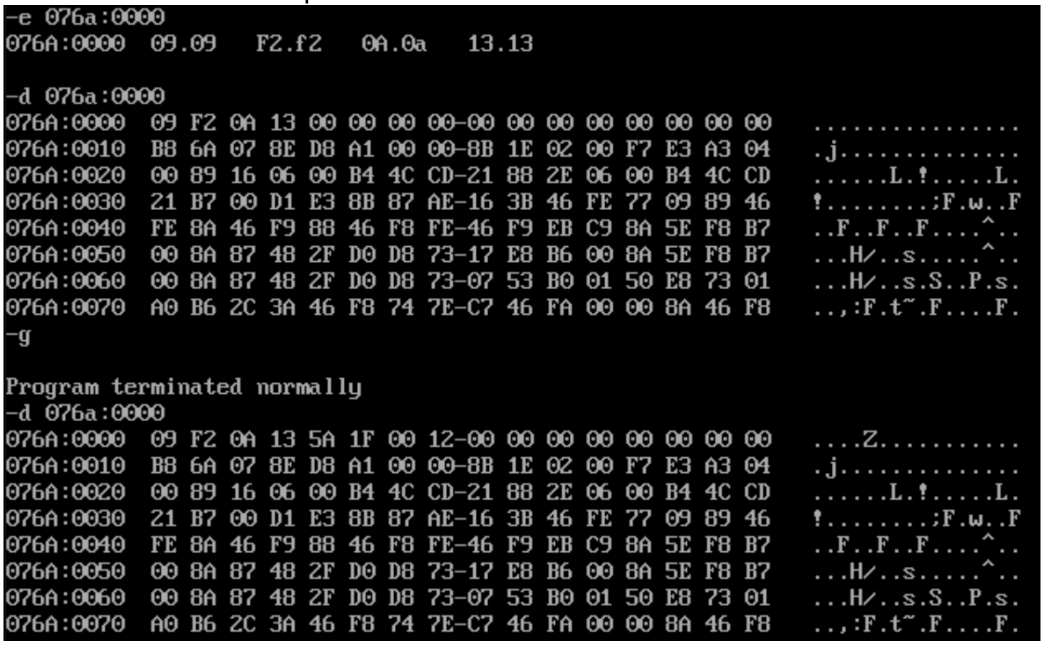


```
-e 076a:0000
076A:0000  09.ff   F2.00   0A.ff   13.00

-d 076a:0000
076A:0000  FF 00 FF 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 F7 E3 A3 04   .j..............
076A:0020  00 89 16 06 00 B4 4C CD-21 88 2E 06 00 B4 4C CD   ......L.!.....L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  FF 00 FF 00 01 FE 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 F7 E3 A3 04   .j..............
076A:0020  00 89 16 06 00 B4 4C CD-21 88 2E 06 00 B4 4C CD   ......L.!.....L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Case iii: Multiplication producing ax and dx
Hexadecimal multiplication – F209 x 130A = 1200 1F5A



```
-e 076a:0000
076A:0000  09.09   F2.f2   0A.0a   13.13

-d 076a:0000
076A:0000  09 F2 0A 13 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 F7 E3 A3 04   .j..............
076A:0020  00 89 16 06 00 B4 4C CD-21 88 2E 06 00 B4 4C CD   ......L.!.....L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  09 F2 0A 13 5A 1F 00 12-00 00 00 00 00 00 00 00   ....Z...........
076A:0010  B8 6A 07 8E D8 A1 00 00-8B 1E 02 00 F7 E3 A3 04   .j..............
076A:0020  00 89 16 06 00 B4 4C CD-21 88 2E 06 00 B4 4C CD   ......L.!.....L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Result:

Program to multiply two 16-bit numbers assembled, executed and verified.

2D – 16-bit division

Aim:

To divide two 16-bit numbers

Algorithm:

1. Declare and initialize the data segment.
2. Begin code segment, where actual assembler instructions are present.
3. Move the starting address of data segment into ds register.
4. Store the dividend in ax and divisor in bx.
5. Divide ax and bx: dx ax = ax / bx
6. Store quotient generated at ax and remainder at dx, by default property of the instruction.
7. Terminate program and code segment.

Program:

Program	Comments
`;Program to divide two 16-bit numbers`	Comment after ‘;’
`assume cs:code,ds:data`	Map CS to code segment and DS to data segment

`data segment`
`    dividend dw 0f209H`
`    divisor dw 130aH`
`    quotient dw 0000H`
`    remainder dw 0000H`
`data ends`
	Initialise data segment
	db = define a byte
	Initialise dividend = F209, divisor = 130A, quotient = 0000, remainder = 0000

`code segment`
`start:  mov ax, data`
`        mov ds, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds.

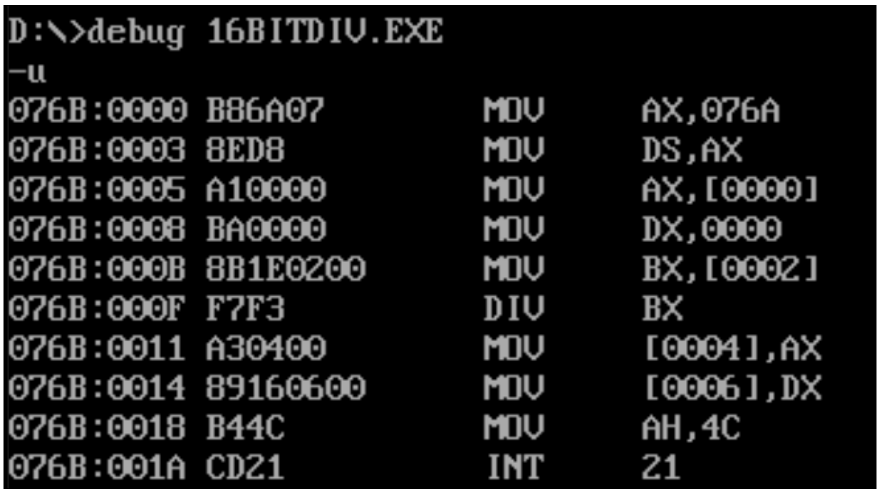
`        mov ax, dividend`
`        mov dx, 0000H`
`        mov bx, divisor`
	Move dividend to al, divisor to bl

`        div bx`	Divide ax and bx: dx ax = ax / bx (Fixed instruction)

`        mov quotient, ax`
`        mov remainder, dx`
	Move ax to quotient
	Move dx to remainder

`        mov ah, 4cH`
`        int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Unassembled code:



```
D:\>debug 16BITDIV.EXE
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 A10000        MOV     AX,[0000]
076B:0008 BA0000        MOV     DX,0000
076B:000B 8B1E0200      MOV     BX,[0002]
076B:000F F7F3          DIV     BX
076B:0011 A30400        MOV     [0004],AX
076B:0014 89160600      MOV     [0006],DX
076B:0018 B44C          MOV     AH,4C
076B:001A CD21          INT     21
```

8086 follows little-endian notation. The lower half(8-bit) of the 16-bit register goes to lower memory address and upper half(8-bit) goes to higher memory address and vice-versa.

From memory to registers:

```
      AL <- 0000 AH <- 0001
      BL <- 0002 BH <- 0003
```

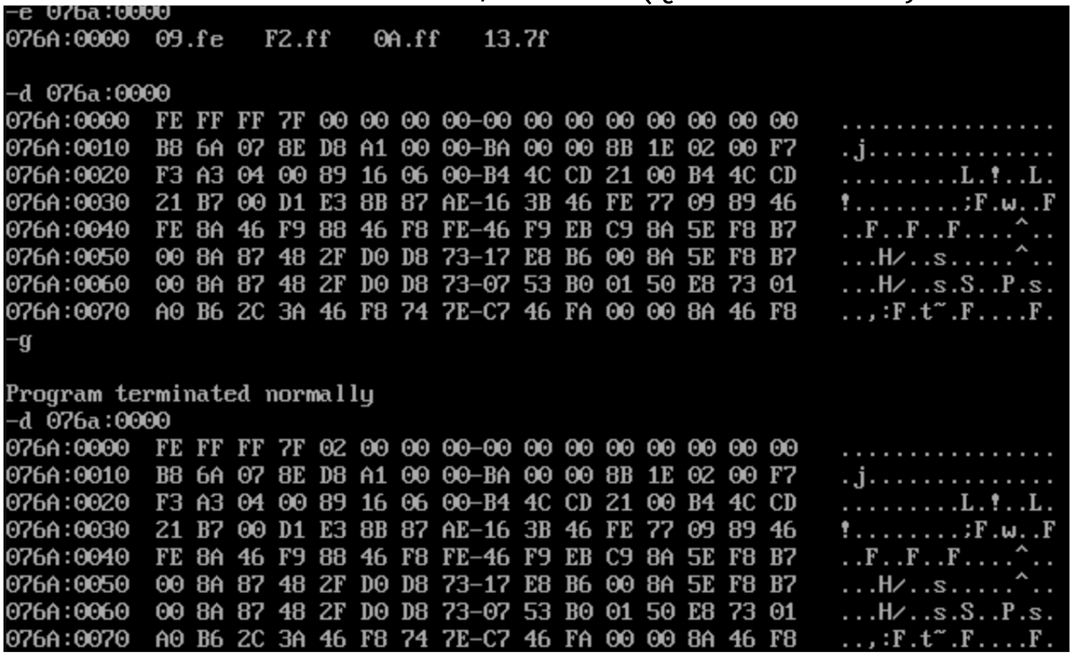
From registers to memory

```
      0004 <- AL 0005 <- AH
      0006 <- DL 0007 <- DH
```

Snapshot of sample input and output:

Case i: Without remainder

Hexadecimal Division: FFFE / 7FFF = (Quotient: 0002, Remainder: 0000)

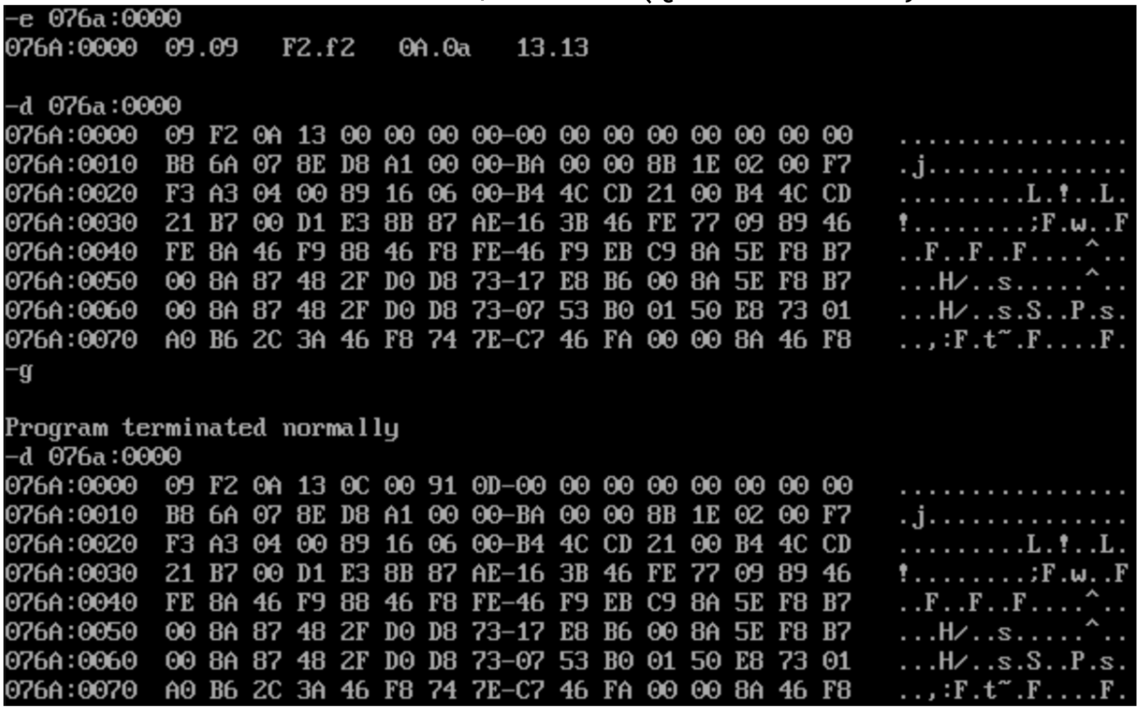


```
-e 076a:0000
076A:0000  09.fe   F2.ff   0A.ff   13.7f

-d 076a:0000
076A:0000  FE FF FF 7F 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-BA 00 00 8B 1E 02 00 F7   .j..............
076A:0020  F3 A3 04 00 89 16 06 00-B4 4C CD 21 00 B4 4C CD   .........L.!..L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  FE FF FF 7F 02 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-BA 00 00 8B 1E 02 00 F7   .j..............
076A:0020  F3 A3 04 00 89 16 06 00-B4 4C CD 21 00 B4 4C CD   .........L.!..L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Case ii: With remainder
Hexadecimal Division: F209 / 130A = (Quotient: 000C, Remainder: 0D91)



```
-e 076a:0000
076A:0000  09.09   F2.f2   0A.0a   13.13

-d 076a:0000
076A:0000  09 F2 0A 13 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-BA 00 00 8B 1E 02 00 F7   .j..............
076A:0020  F3 A3 04 00 89 16 06 00-B4 4C CD 21 00 B4 4C CD   .........L.!..L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-g

Program terminated normally
-d 076a:0000
076A:0000  09 F2 0A 13 0C 00 91 0D-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 A1 00 00-BA 00 00 8B 1E 02 00 F7   .j..............
076A:0020  F3 A3 04 00 89 16 06 00-B4 4C CD 21 00 B4 4C CD   .........L.!..L.
076A:0030  21 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   !........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Result:

Program to divide two 16-bit numbers assembled, executed and verified.