16 Bit Arithmetic Operations

Expt No:	2	**Name:**	Shivanirudh S G
Date :	28/08/2020	**Reg No:**	185001146

Aim:

To perform arithmetic operations on two 16 bit numbers.

---

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**
assume cs:code,ds:data	Declare code and data segments
data segment	Start of data segment
opr1 db 1111h	Define byte opr1 with hex value 1111
opr2 db 9999h	Define byte opr2 with hex value 9999
result db 0000H	Define byte result with hex value 0000
carry db 0000H	Define byte carry with hex value 0000
data ends	End of data segment
code segment	Start of code segment
org 0100h	Set preferred offset
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ax,opr1	Move contents of opr1 to AX register
mov bx,opr2	Move contents of opr2 to BX register
mov cx,00h	Move hex value 00 to CX register
add ax,bx	AH = AX + BX
jnc here	Jump to the label here, if there is no carry
inc cx	Increment value of CX if there is a carry
here: mov result,ax	Move contents of AX register to result
mov carry,cx	Move contents of CX register to carry
int 21h	Request interrupt routine
code ends	End of code segment
end start	

Unassembled code:

```
0E25:0100 B8240E        MOV     AX,0E24
0E25:0103 8ED8          MOV     DS,AX
0E25:0105 A10000        MOV     AX,[0000]
0E25:0108 8B1E0200      MOV     BX,[0002]
0E25:010C B90000        MOV     CX,0000
0E25:010F 03C3          ADD     AX,BX
0E25:0111 7301          JNB     0114
0E25:0113 41            INC     CX
0E25:0114 A30400        MOV     [0004],AX
0E25:0117 890E0600      MOV     [0006],CX
0E25:011B B44C          MOV     AH,4C
0E25:011D CD21          INT     21
0E25:011F B62C          MOV     DH,2C
-
```

Input and Output:

```
-d 0e24:0000
0E24:0000  11 11 99 99 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 0e24:0000
0E24:0000  11 11 99 99 AA AA 00 00-00 00 00 00 00 00 00 00   ................
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-q
```

Figure 1: **Input:** *opr1:* 1111h, *opr2:* 9999h; **Output:** *Result:* AAAAh, *Carry:* 0000h

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**
assume cs:code,ds:data	Declare code and data segments
data segment	Start of data segment
opr1 db 1111h	Define byte opr1 with hex value 1111
opr2 db 9999h	Define byte opr2 with hex value 9999
result db 0000H	Define byte result with hex value 0000
carry db 0000H	Define byte carry with hex value 0000
data ends	End of data segment
code segment	Start of code segment
org 0100h	Set preferred offset
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ax,opr1	Move contents of opr1 to AX register
mov bx,opr2	Move contents of opr2 to BX register
mov cx,00h	Move hex value 00 to CX register
sub ax,bx	AH = AX + BX
jnc here	Jump to the label here, if there is no carry
inc cx	Increment value of CX if there is a carry
neg ax	Negate value of AX if there is a carry
here: mov result,ax	Move contents of AX register to result
mov carry,cx	Move contents of CX register to carry
int 21h	Request interrupt routine
code ends	End of code segment
end start	

Unassembled code:

```
0E25:0100 B8240E        MOV     AX,0E24
0E25:0103 8ED8          MOV     DS,AX
0E25:0105 A10000        MOV     AX,[0000]
0E25:0108 8B1E0200      MOV     BX,[0002]
0E25:010C B90000        MOV     CX,0000
0E25:010F 2BC3          SUB     AX,BX
0E25:0111 7303          JNB     0116
0E25:0113 41            INC     CX
0E25:0114 F7D8          NEG     AX
0E25:0116 A30400        MOV     [0004],AX
0E25:0119 890E0600      MOV     [0006],CX
0E25:011D B44C          MOV     AH,4C
0E25:011F CD21          INT     21
-_
```

Input and Output:

```
-d 0e24:0000
0E24:0000  11 11 99 99 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 0e24:0000
0E24:0000  11 11 99 99 88 88 01 00-00 00 00 00 00 00 00 00   ................
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Figure 2: **Input:** *opr1:* 1111h, *opr2:* 9999h; **Output:** *Result:* 8888h, *Sign:* 0001h

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 AL is default operand register for MUL instruction we only need to specify the other operand register.)
- The result stored in (DX)(AX) register(32 bit- because multiplication of two 16 bit numbers yields a 32 bit number) should now be moved to RESULT.

Program:

Program	**Comments**
assume cs:code,ds:data	Declare code and data segments
data segment	Start of data segment
opr1 db 2222h	Define byte opr1 with hex value 2222
opr2 db 3333h	Define byte opr2 with hex value 3333
resulth db 0000H	Define byte resulth with hex value 0000
resultl db 0000H	Define byte resultl with hex value 0000
data ends	End of data segment
code segment	Start of code segment
org 0100h	Set preferred offset
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ax,opr1	Move contents of opr1 to AX register
mov dx, 0000H	Move hex value 0000 to DX register
mov bx,opr2	Move contents of opr2 to BX register
mul bx	(DX)(AX) = AX * BX
here: mov resulth,dx	Move contents of DX register to resulth
mov resultl,ax	Move contents of AX register to resultl
int 21h	Request interrupt routine
code ends	End of code segment
end start	

Unassembled code:

```
-u
0E25:0100 B8240E        MOV     AX,0E24
0E25:0103 8ED8          MOV     DS,AX
0E25:0105 A10000        MOV     AX,[0000]
0E25:0108 8B1E0200      MOV     BX,[0002]
0E25:010C F7E3          MUL     BX
0E25:010E 89160400      MOV     [0004],DX
0E25:0112 A30600        MOV     [0006],AX
0E25:0115 B44C          MOV     AH,4C
0E25:0117 CD21          INT     21
0E25:0119 2A01          SUB     AL,[BX+DI]
0E25:011B E8C6FA        CALL    FBE4
0E25:011E A2B62C        MOV     [2CB6],AL
-
```

Input and Output:

```
-d 0e24:0000
0E24:0000  22 22 33 33 00 00 00 00-00 00 00 00 00 00 00 00   ""33............
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 0e24:0000
0E24:0000  22 22 33 33 D3 06 C6 92-00 00 00 00 00 00 00 00   ""33............
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

Figure 3: **Input:** *opr1:* 2222h, *opr2:* 3333h; **Output:** *Result:* 92C6 06D3h

16 Bit Division

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.
- Move value in DX register
- Then divide using DIV BX .(Since AL is default operand register for MUL instruction we only need to specify the other operand register.)
- The result stored in (DX) for quotient, (AX) for remainder should now be moved to RESULTQ and RESULTR respectively.

Program:

Program	**Comments**
assume cs:code,ds:data	Declare code and data segments
data segment	Start of data segment
opr1 db 6666h	Define byte opr1 with hex value 6666
opr2 db 3333h	Define byte opr2 with hex value 3333
resulth db 0000H	Define byte resulth with hex value 0000
resultl db 0000H	Define byte resultl with hex value 0000
data ends	End of data segment
code segment	Start of code segment
org 0100h	Set preferred offset
start: mov ax,data	Move data segment contents to AX register
mov ds,ax	Move data in AX register to DS register
mov ax,opr1	Move contents of opr1 to AX register
mov dx, 0001H	Move hex value 0001 to DX register
mov bx,opr2	Move contents of opr2 to BX register
div bx	(DX) = (DX)(AX) / BX; (AX) = (DX)(AX) % BX
here: mov resultq,dx	Move contents of DX register to resultq
mov resultr,ax	Move contents of AX register to resultr
int 21h	Request interrupt routine
code ends	End of code segment
end start	

Unassembled code:

```
-u
0E25:0100 B8240E        MOV     AX,0E24
0E25:0103 8ED8          MOV     DS,AX
0E25:0105 A10000        MOV     AX,[0000]
0E25:0108 BA0100        MOV     DX,0001
0E25:010B 8B1E0200      MOV     BX,[0002]
0E25:010F F7F3          DIV     BX
0E25:0111 A30400        MOV     [0004],AX
0E25:0114 89160600      MOV     [0006],DX
0E25:0118 B44C          MOV     AH,4C
0E25:011A CD21          INT     21
0E25:011C C6FAA2        MOV     DL,A2
0E25:011F B62C          MOV     DH,2C
-_
```

Input and Output:

```
0E25:011C C6FAA2        MOV     DL,A2
0E25:011F B62C          MOV     DH,2C
-d 0e24:0000
0E24:0000  66 66 33 33 00 00 00 00-00 00 00 00 00 00 00 00   ff33............
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 0e24:0000
0E24:0000  66 66 33 33 07 00 01 00-00 00 00 00 00 00 00 00   ff33............
0E24:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0E24:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Figure 4: **Input:** *opr1:* 6666h, *opr2:* 3333h; **Output:** *Quotient:* 0007h, *Remainder:* 0001h

---

Result:

The 8086 programs were written to perform 16-bit arithmetic operations, and the results observed.