

EXPERIMENT -1

AIM

1. Design an 8086-assembly language program to multiply two 16-bit numbers and store the 32-bit result. The operands are multiplier, multiplicand and result.
2. Design an 8086-assembly language program to divide a 32-bit no. by 16-bit no. The operands are dividend, divisor, quotient and remainder.

SOFTWARE

- EMU8086 emulator

EXPERIMENTS

1. MULTIPLICATION: -

- **THEORY**

The arithmetic instructions give the microprocessor its computational capabilities. The 8086 can add and subtract 8- and 16-bit numbers in any of the general CPU registers, and using certain dedicated registers, perform multiplication and division of signed or unsigned numbers.

The source operand can be a memory location or a CPU register, but the destination operand must be register AX (and DX for 32-bit results).

OP-CODE: MUL;

OPERAND: source

DESCRIPTION: Unsigned multiplication of the byte or word source operand and the accumulator; word results are stored in AX and double word results are stored DX:AX. CF and OF are set; cleared otherwise, all other flags are undefined.

- **CODE: -**

```
.model small ; DS and CS <= 64KB
.stack      ; occupies 1024 bytes for stack segment

.data
multiplier dw 1357h
multiplicand dw 2468h
result dd ?

.code      ; initialize code segment
multiplication proc ; initializes the procedure name 'multiplication'

.startup ; initialize data segment

mov ax, multiplicand
mul multiplier
mov word ptr [result], ax
mov word ptr [result+2], dx

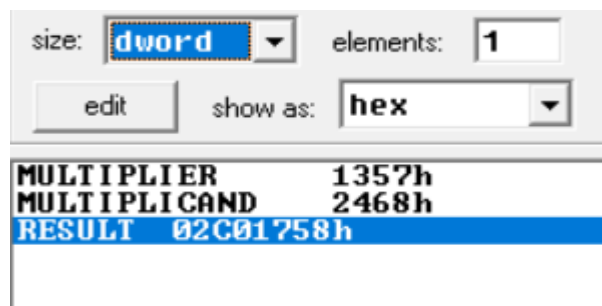
.exit

multiplication endp ; terminates the procedure with procedure name 'multiplication'
end multiplication ; ends the program; imp to give end a procedure name
```

- **EXPLANATION**

In the Data segment (signified at .data) we have defined their variables of which multiplier and multiplicand are 16 bit and result is 32 bit. So, in the code segment we copy multiplicand into the accumulator and then using the MUL operator, operate the multiplication function on multiplier. Now as we know the answers is stored in DX: AX register which is then copied into the 'result' variable which basically represents an address in memory.

- **OUTPUT**



size:	dword	elements:	1
edit		show as:	hex
MULTIPLIER	1357h		
MULTIPLICAND	2468h		
RESULT	02C01758h		

2. DIVISION

- **THEORY**

The arithmetic instructions give the microprocessor its computational capabilities. The 8086 can add and subtract 8- and 16-bit numbers in any of the general CPU registers, and using certain dedicated registers, perform multiplication and division of signed or unsigned numbers.

Division can be performed on the word in AX or the double word in DX: AX. The divisor can be an 8- or a 16- bit memory location on CPU register. Note that a remainder is returned in register AH or DX when the result does not come out even.

OP-CODE: DIV;

OPERAND: source

Description: Unsigned division of the accumulator or accumulator and DX. For byte divisors the result is returned in AL with remainder in AH. For word divisors the result is returned in AX with remainder in DX. If the quotient exceeds the capacity of its destination register a type 0 interrupt is generated. All flags are undefined.

- **CODE**

```

.model small ; DS and CS <= 64KB
.stack      ; occupies 1024 bytes for stack segment

.data
    dividend dd 71C2580Ah;04681234h
    divisor dw 0F6F2h      ;01A57h
    quotient dw ?
    remainder dw ?

.code      ; initialize code segment
    division proc      ; initializes the procedure name 'division'

        .startup      ; initialize data segment

        mov ax, word ptr [dividend]
        mov dx, word ptr [dividend+2]
        div word ptr divisor
        mov word ptr [quotient], ax
        mov word ptr [remainder], dx

        .exit

    division endp ; terminates the procedure with procedure name 'division'
end division      ; ends the program; imp to give end a procedure name

```

- **EXPLANATION**

In the Data segment (signified at .data) we have defined their variables of which the dividend is 32 bit and the divisor, quotient and remainder are 16 bits. So in the code segment we copy dividend into the accumulator and DX register. Then using the DIV operator, operate the division function on dividend. Now as we know the answers is stored in DX: AX register which has the quotient is then copied into the 'quotient' variable which basically represents an address in memory. And the DX register which has the remainder is copied into the 'remainder' variable in the data segment

- **OUTPUT**

size:	dword	elements:	1
edit	show as:	hex	
DIVIDEND	71C2580Ah		
DIVISOR	75EEF6F2h		
QUOTIENT	290E75EEh		
REMAINDER	0000290Eh		

CONCLUSION

In this program we have used the data transfer instructions and then also learnt to perform the multiplication and division operations on variables of different sizes and checked how the result is stored after the manipulation of the registers in calculation of the result and the storage format of the result.