PARSHVANATH CHARITABLE TRUST'S

A.P. SHAH INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering
Data Science

Department of Computer Science Engineering Data Science

Academic Year: 2022-23 **Semester: IV**
Class / Branch: S.E.D.S. **Subject: Microprocessor Lab**

Experiment No. 1

1. Aim: Use of programming tool TASM/Debug to perform basic arithmetic operations on 8 bit data.

ADD,SUB,MUL,DIV

2. Software used: tasm,tlink ,td,dosemu

3. Theory :-

3.1 add(Integer Addition) —

The add instruction adds together its two operands, storing the result in its first operand. Note, whereas both operands may be registers, at most one operand may be a memory location.

- *Syntax*
  add <reg>,<reg>
  add <reg>,<mem>
  add <mem>,<reg>
  add <reg>,<con>
  add <mem>,<con>

- **Example**
  ADD AL,74H ;Add immediate number 74H to content of AL

3.2 sub(Integer Subtraction) —

The sub instruction stores in the value of its first operand the result of subtracting the value of its second operand from the value of its first operand.

sub <reg>,<reg>
sub <reg>,<mem>
sub <mem>,<reg>
sub <reg>,<con>
sub <mem>,<con>

- **Example**

  SUB AX,BX ;AX-BX and store result in AX

3.3 MUL(Unsigned multiply)—

This instruction multiplies an unsigned multiplication of the accumulator by the operand specified by op. The size of op may be a register or memory operand

when operand is a **byte**:

AX = AL * operand.

when operand is a **word**:

(DX AX) = AX * operand.

- ***Syntax***

  MUL REG

- **Example:**

```
MOV AL, 200 ; AL = 0C8h
MOV BL, 4
MUL BL       ; AX = 0320h (800)
RET
```

3.4 DIV(Unsigned divide)—

This instruction is used to divide an Unsigned word by a byte or to divide an unsigned double word by a word. When dividing a word by a byte , the word must be in the AX register. After the division AL will contains an 8- bit result (quotient) and AH willcontain an 8- bit remainder. If an attempt is made to divide by 0 or the quotient is too large to fit in AL (greater than FFH), the 8086 will automatically do a type 0 interrupt

When a double word is divided by a word, the most significant word of the double word must be in DX and the least significant word of the double word must be in AX. After the division AX will contain the 16 –bit result (quotient) and DX will contain a 16 bit remainder. Again , if an attempt is made to divide by zero or quotient is too large to fit in AX (greater than FFFFH) the 8086 will do a type of 0 interrupt

when operand is a **byte**:
AL = AX / operand

AH = remainder (modulus)

when operand is a **word**:
AX = (DX AX) / operand
DX = remainder (modulus)

DIV REG

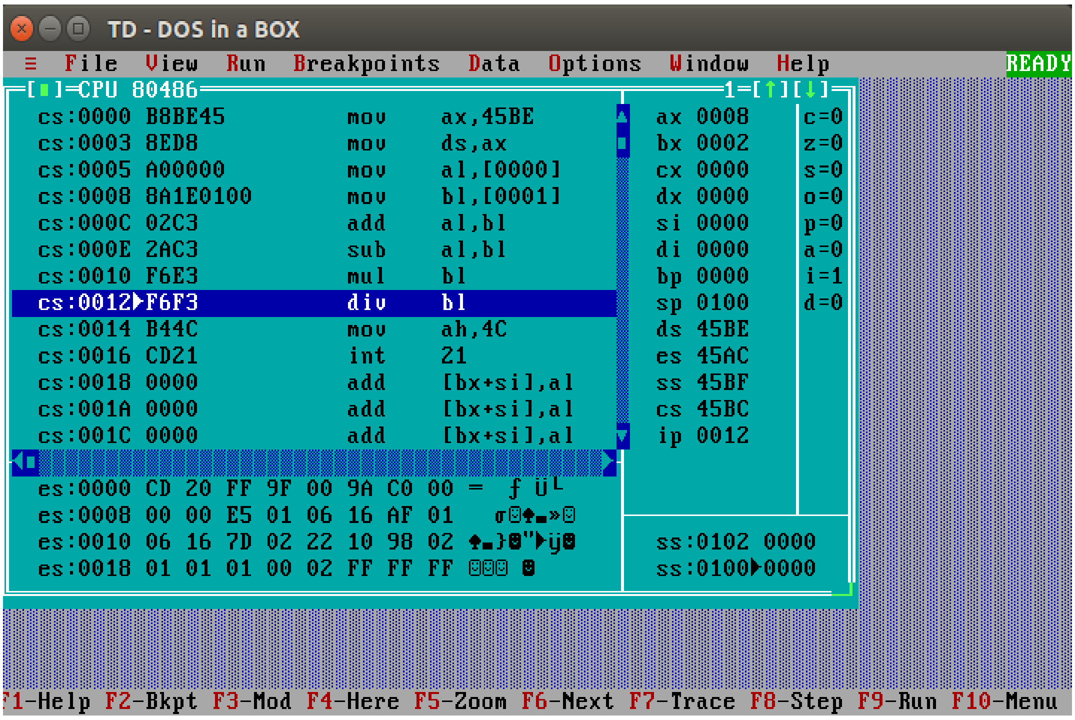
- **Example:**

```
MOV AX, 203 ; AX = 00CBh
MOV BL, 4
DIV BL      ; AL = 50 (32h), AH =3
RET
```

4. Program

```
.model small
.stack 100h
.data
num1 db 04h
num2 db 02h
.code
start:
mov ax,@data
mov ds,ax
mov al,num1
mov bl,num2
add al,bl
sub al,bl
mul bl
div bl
mov ah,4ch
int 21h
end start
```

Program Output:


TD - DOS in a BOX
≡ File View Run Breakpoints Data Options Window Help
READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
[■]=CPU 80486=========================================1=[↑][↓]=
  cs:0000 B8BE45         mov    ax,45BE          ax 0008    c=0
  cs:0003 8ED8           mov    ds,ax            bx 0002    z=0
  cs:0005 A00000         mov    al,[0000]        cx 0000    s=0
  cs:0008 8A1E0100       mov    bl,[0001]        dx 0000    o=0
  cs:000C 02C3           add    al,bl            si 0000    p=0
  cs:000E 2AC3           sub    al,bl            di 0000    a=0
  cs:0010 F6E3           mul    bl               bp 0000    i=1
  cs:0012▶F6F3           div    bl               sp 0100    d=0
  cs:0014 B44C           mov    ah,4C            ds 45BE
  cs:0016 CD21           int    21               es 45AC
  cs:0018 0000           add    [bx+si],al       ss 45BF
  cs:001A 0000           add    [bx+si],al       cs 45BC
  cs:001C 0000           add    [bx+si],al       ip 0012

  es:0000 CD 20 FF 9F 00 9A C0 00 =  ƒ Ü└
  es:0008 00 00 E5 01 06 16 AF 01   σ☺♠▬»☺
  es:0010 06 16 7D 02 22 10 98 02 ♠▬}☻"▶ÿ☻      ss:0102 0000
  es:0018 01 01 01 00 02 FF FF FF ☺☺☺ ☻         ss:0100▶0000
```

5.Conclusion :-