



## **Experiment No-1**

**AIM:** Program for arithmetic operations (8-bits/16-bits)

### **THEORY:**

**TASM** - This is the assembler. It is used to convert your assembly language file to an object file that contains the machine code that the processor will execute. Register examine or modify the content of internal register of the CPU. The 8086 contains 14 registers. Each register is 16 bits long. The general purpose registers can be "split". AH contains the high byte of AX and AL contains the low byte. "Mov" It is to be shorthand for the word "Move".

**Code Segment** – It contains all the instructions to be executed. A 16-bit Code Segment register or CS register stores the starting address of the code segment.

**Data Segment** – It contains data, constants and work areas. A 16-bit Data Segment register or DS register stores the starting address of the data segment.

**Stack Segment** – It contains data and return addresses of procedures or subroutines. It is implemented as a 'stack' data structure. The Stack Segment register or SS register stores the starting address of the stack.

**Sub** : It is used for subtraction of two numbers

**Add** : It is used for addition of two numbers

**Mul** : It is used for multiplication of two numbers

**Div** : It is used for division of two numbers

**HLT** : HLT (halt) is an assembly language instruction which halts the central processing unit (CPU) until the next external interrupt is fired

### **# Addition Of 8-bit numbers**

assume cs:code, ds:data

data segment



AET's  
**Atharva College of Engineering, Malad(W)**

Approved by AICTE, New Delhi, DTE, Mumbai  
Affiliated to University of Mumbai, ISO certified 9001:2015

Department of Information Technology

Academic Year: 2021-22

data ends

code segment

start:

    mov al,04h

    mov bl,07h

    add al,bl

    HLT

code ends

end start

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

READY

[CPU 80486] File View Run Breakpoints Data Options Window Help

Address	OpCode	Instruction	Registers
cs:0000 B004	MOV	al,04	ax 000B c=0
cs:0002 B307	MOV	bl,07	bx 0007 z=0
cs:0004 02C3	ADD	al,bl	
cs:0006 F4	HLT		dx 0000 o=0
cs:0007>0000	ADD	[bx+si],al	si 0000 p=0
cs:0009 0000	ADD	[bx+si],al	di 0000 a=0
cs:000B 0000	ADD	[bx+si],al	bp 0000 i=1
cs:000D 0000	ADD	[bx+si],al	sp 0000 d=0
cs:000F 0000	ADD	[bx+si],al	ds 449D
cs:0011 0000	ADD	[bx+si],al	es 449D
cs:0013 0000	ADD	[bx+si],al	ss 44AC
cs:0015 0000	ADD	[bx+si],al	cs 44AD
cs:0017 0000	ADD	[bx+si],al	ip 0007

ds:0000 CD 20 FF 9F 00 EA FF FF = fΩ

ds:0008 AD DE E5 01 00 15 AF 01 ↗ ↘ ↙ ↛

ds:0010 00 15 7D 02 1C 0F 92 01 ↘ ↙ ↛ ↚

ds:0018 01 01 01 00 02 FF FF FF 0000 ↙ ↛

ss:0002 6568

ss:0000>6474

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

### Subtraction of 8-bit numbers

assume cs:code, ds:data

data segment

data ends

code segment

start:

```
    mov al, 25h
```

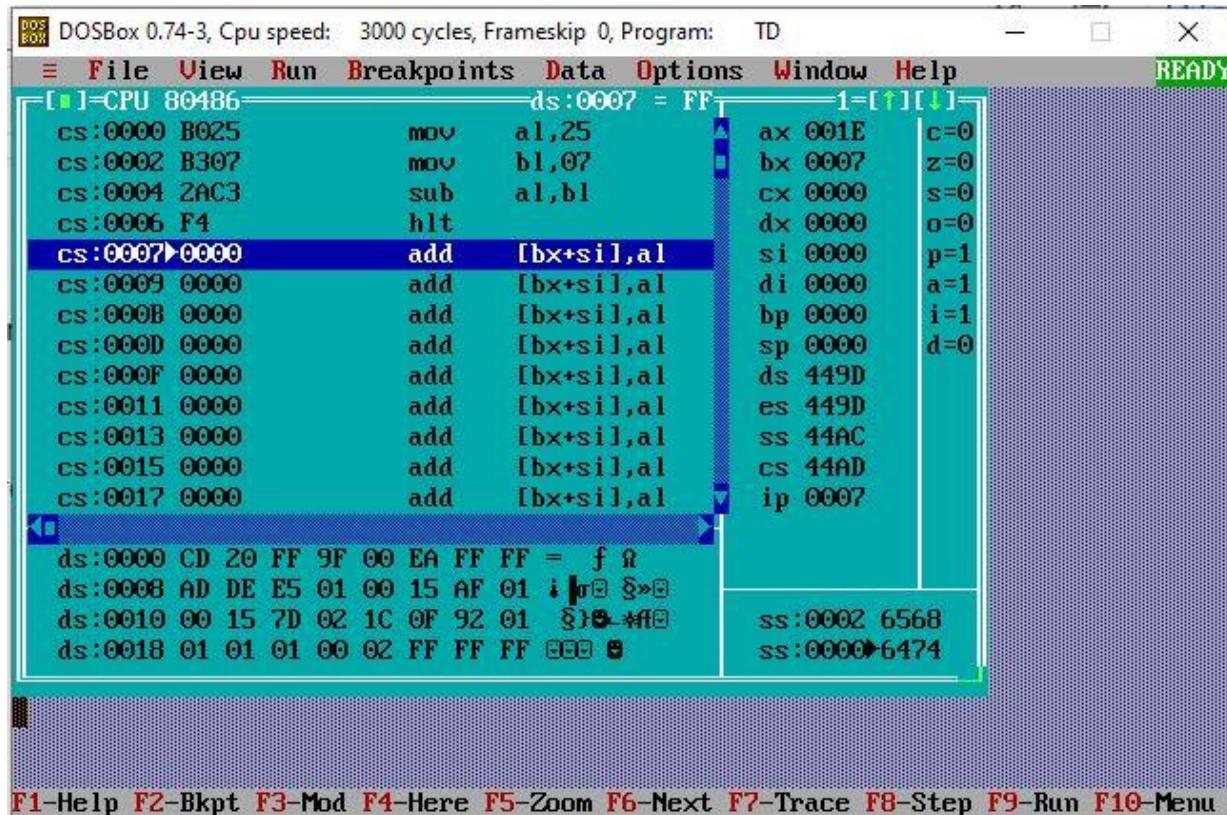
```
    mov bl, 07h
```

```
    sub al,bl
```

```
    HLT
```

```
code ends
```

```
end start
```



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

[CPU 80486] File View Run Breakpoints Data Options Window Help READY

Address	OpCode	Instruction	Registers
cs:0000 B025	mov	al,25	ax 001E c=0
cs:0002 B307	mov	bl,07	bx 0007 z=0
cs:0004 2AC3	sub	al,bl	
cs:0006 F4	hlt		dx 0000 o=0
cs:0007>0000	add	[bx+sil],al	si 0000 p=1
cs:0009 0000	add	[bx+sil],al	di 0000 a=1
cs:000B 0000	add	[bx+sil],al	bp 0000 i=1
cs:000D 0000	add	[bx+sil],al	sp 0000 d=0
cs:000F 0000	add	[bx+sil],al	ds 449D
cs:0011 0000	add	[bx+sil],al	es 449D
cs:0013 0000	add	[bx+sil],al	ss 44AC
cs:0015 0000	add	[bx+sil],al	cs 44AD
cs:0017 0000	add	[bx+sil],al	ip 0007

ds:0000 CD 20 FF 9F 00 EA FF FF = fΩ<sup>1</sup>  
ds:0008 AD DE E5 01 00 15 AF 01 → [pΩ S»Ω  
ds:0010 00 15 7D 02 1C 0F 92 01 S)Ω-#fΩ  
ds:0018 01 01 01 00 02 FF FF FF 0000 0

ss:0002 6568  
ss:0000 6474

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

**Multiplication of 8-bit numbers**

assume cs:code, ds:data



AET's  
**Atharva College of Engineering, Malad(W)**

Approved by AICTE, New Delhi, DTE, Mumbai  
Affiliated to University of Mumbai, ISO certified 9001:2015

Department of Information Technology

Academic Year: 2021-22

data segment

data ends

code segment

start:

```
    mov al,07h
```

```
    mov bl,03h
```

```
    mul bl
```

```
    HLT
```

```
    code ends
```

```
end start
```

The screenshot shows the DOSBox interface with the title bar "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD". The menu bar includes File, View, Run, Breakpoints, Data, Options, Window, Help, and READY. The CPU window displays assembly code and registers. The assembly code is:

```
[CPU] CPU 80486 ds:0003 = 9F1=1-[↑][↓]-  
cs:0000 B007      mov   al,07  
cs:0002 B303      mov   bl,03  
cs:0004 F6E3      mul   bl  
cs:0006 F4        hlt  
cs:0007>0000      add   [bx+si],al  
cs:0009 0000      add   [bx+si],al  
cs:000B 0000      add   [bx+si],al  
cs:000D 0000      add   [bx+si],al  
cs:000F 0000      add   [bx+si],al  
cs:0011 0000      add   [bx+si],al  
cs:0013 0000      add   [bx+si],al  
cs:0015 0000      add   [bx+si],al  
cs:0017 0000      add   [bx+si],al
```

The registers window shows:

ax 0015	c=0
bx 0003	z=0
cx 0000	s=0
dx 0000	o=0
si 0000	p=0
di 0000	a=0
bp 0000	i=1
sp 0000	d=0
ds 449D	
es 449D	
ss 44AC	
cs 44AD	
ip 0007	

The stack window shows:

ss:0002 6568
ss:0000>6474

The status bar at the bottom shows keyboard shortcuts: F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu.



AET's  
**Atharva College of Engineering, Malad(W)**

Approved by AICTE, New Delhi, DTE, Mumbai  
Affiliated to University of Mumbai, ISO certified 9001:2015

Department of Information Technology

Academic Year: 2021-22

## Division of 8-bit numbers

assume cs:code, ds:data

data segment

data ends

code segment

start:

```
    mov al,20 h
```

```
    mov bl,05h
```

```
    div bl
```

```
    HLT
```

```
    code ends
```

```
    end start
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

= File View Run Breakpoints Data Options Window Help

[CPU 80486]

Address	OpCode	Instruction	Registers
cs:0000 B020	mov	al,20	ax 0206 c=0
cs:0002 B305	mov	bl,05	bx 0005 z=0
cs:0004 F6F3	div	bl	cx 0000 s=0
cs:0006 F4	HLT		dx 0000 o=0
cs:0007 0000	add	[bx+si],al	si 0000 p=0
cs:0009 0000	add	[bx+si],al	di 0000 a=0
cs:000B 0000	add	[bx+si],al	bp 0000 i=1
cs:000D 0000	add	[bx+si],al	sp 0000 d=0
cs:000F 0000	add	[bx+si],al	ds 449D
cs:0011 0000	add	[bx+si],al	es 449D
cs:0013 0000	add	[bx+si],al	ss 44AC
cs:0015 0000	add	[bx+si],al	cs 44AD
cs:0017 0000	add	[bx+si],al	ip 0006

ds:0000 CD 20 FF 9F 00 EA FF FF = fΩ<sup>1</sup>  
ds:0008 AD DE E5 01 00 15 AF 01 i [r]Ω<sup>2</sup>  
ds:0010 00 15 7D 02 1C 0F 92 01 Ω<sup>3</sup>•\*1Ω<sup>4</sup>  
ds:0018 01 01 01 00 02 FF FF FF 0000Ω<sup>5</sup>

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

## # Addition of 16-bit numbers

assume cs:code, ds:data

data segment

a dw 0204h

b dw 0404h

c dw 00h

data ends

code segment

start:

```
    mov ax,data
```

```
    mov ds,ax
```

```
    mov ax,0000h
```

```
    mov bx,0000h
```

```
    mov ax,a
```

```
    mov bx,b
```

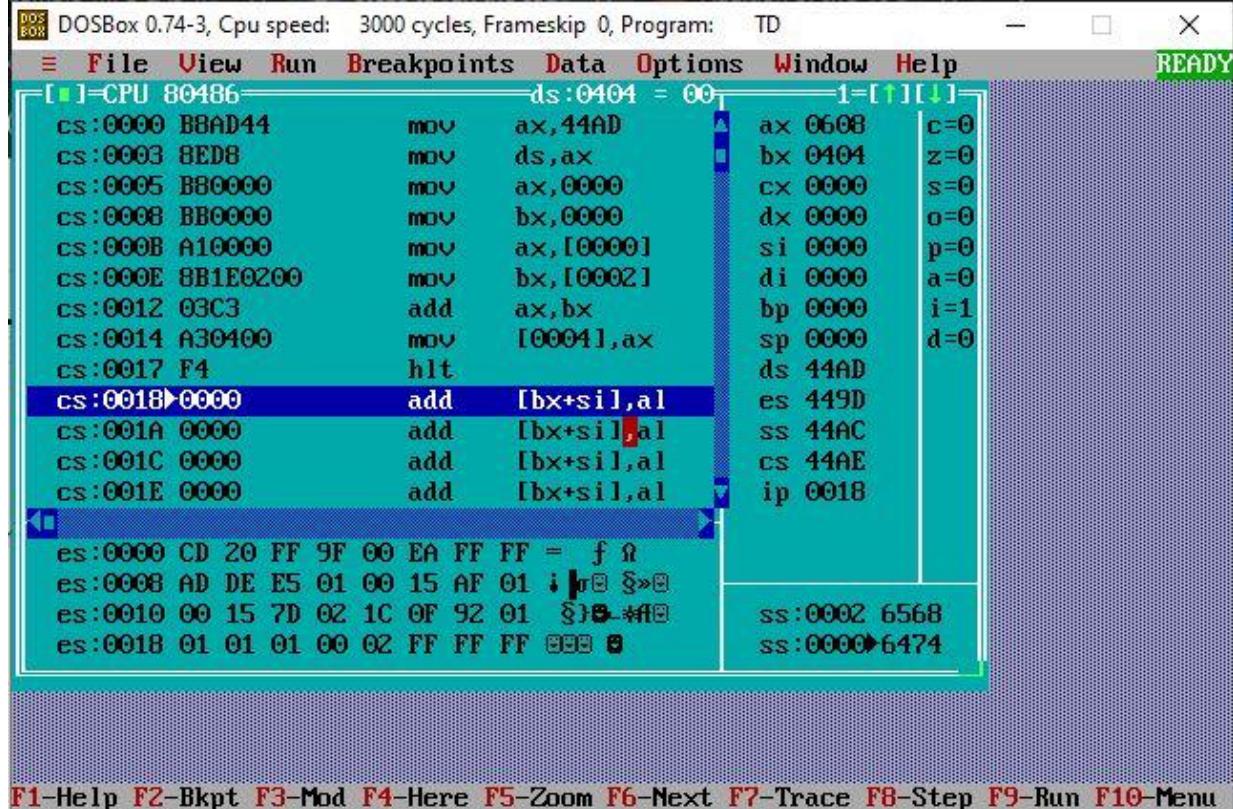
```
    add ax,bx
```

```
    mov c,ax
```

```
    HLT
```

```
code ends
```

```
end start
```



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

READY

Register	Value	Description
ax	0608	c=0
bx	0404	z=0
cx	0000	s=0
dx	0000	o=0
si	0000	p=0
di	0000	a=0
bp	0000	i=1
sp	0000	d=0
ds	44AD	
es	449D	
ss	44AC	
cs	44AE	
ip	0018	

Registers (hex):

- ax: 0608
- bx: 0404
- cx: 0000
- dx: 0000
- si: 0000
- di: 0000
- bp: 0000
- sp: 0000
- ds: 44AD
- es: 449D
- ss: 44AC
- cs: 44AE
- ip: 0018

Stack dump:

```

es:0000 CD 20 FF 9F 00 EA FF FF = f 
es:0008 AD DE E5 01 00 15 AF 01 i 
es:0010 00 15 7D 02 1C 0F 92 01 S 
es:0018 01 01 01 00 02 FF FF FF 0000 
ss:0002 6568 
ss:0000 6474

```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

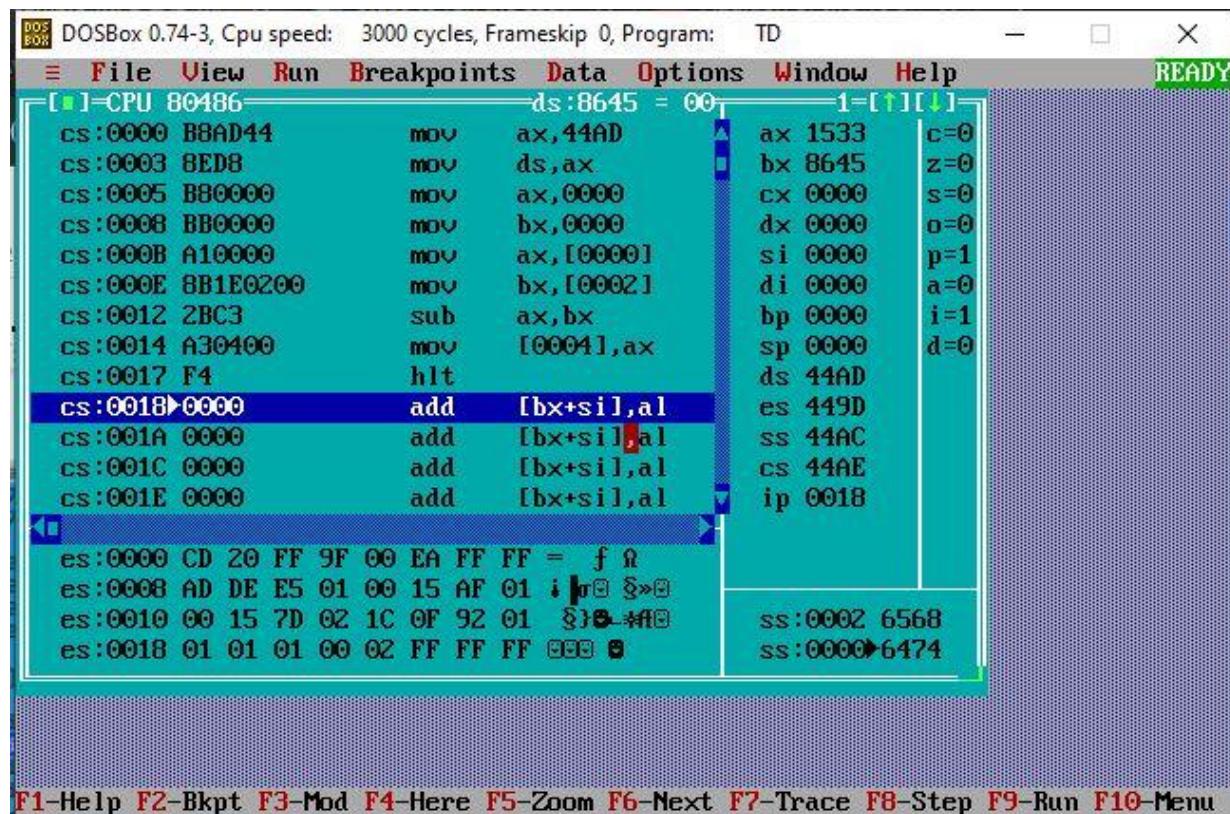
## # Subtraction of 16-bit numbers

assume cs:code, ds:data

data segment

a dw 9B78h

```
b dw 8645h
c dw 00h
data ends
code segment
start:
    mov ax,data
    mov ds,ax
    mov ax,00h
    mov bx,00h
    mov ax,a
    mov bx,b
    sub ax,bx
    mov c,ax
    HLT
code ends
end start
```



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

READY

Register	Value	Description
ax	1533	c=0
bx	8645	z=0
cx	0000	s=0
dx	0000	o=0
si	0000	p=1
di	0000	a=0
bp	0000	i=1
sp	0000	d=0
ds	44AD	
es	449D	
ss	44AC	
cs	44AE	
ip	0018	

Stack dump:

```
ss:0002 6568
ss:0000 6474
```

Memory dump:

```
es:0000 CD 20 FF 9F 00 EA FF FF = f 
es:0008 AD DE E5 01 00 15 AF 01 i |r| S>S
es:0010 00 15 7D 02 1C 0F 92 01 S>S->S
es:0018 01 01 01 00 02 FF FF FF BBB S
```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

assume cs:code, ds:data

data segment

a dw 2571h

b dw 0204h

c dw 00h

data ends

code segment

start:

    mov ax,data

    mov ds,ax

    mov ax,0000h

    mov bx,0000h

    mov ax,a

    mov bx,b

    mul bx

    mov c,ax

    HLT

code ends

end start

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

READY

ds:0204 = 00	1=[↑][↓]		
cs:0000 B8AD44	mov ax,44AD	ax 77C4	c=1
cs:0003 8ED8	mov ds,ax	bx 0204	z=0
cs:0005 B80000	mov ax,0000	cx 0000	s=0
cs:0008 BB0000	mov bx,0000	dx 004B	o=1
cs:000B A10000	mov ax,[0000]	si 0000	p=0
cs:000E 8B1E0200	mov bx,[0002]	di 0000	a=0
cs:0012 F7E3	mul bx	bp 0000	i=1
cs:0014 A30400	mov [0004],ax	sp 0000	d=0
cs:0017 F4	hlt	ds 44AD	
cs:0018>0000	add [bx+si],al	es 449D	
cs:001A 0000	add [bx+si],al	ss 44AC	
cs:001C 0000	add [bx+si],al	cs 44AE	
cs:001E 0000	add [bx+si],al	ip 0018	
es:0000 CD 20 FF 9F 00 EA FF FF = f ↳			
es:0008 AD DE E5 01 00 15 AF 01 i ↳ \$>			
es:0010 00 15 7D 02 1C 0F 92 01 \$)8-#18			
es:0018 01 01 01 00 02 FF FF FF BBB ↳		ss:0002 6568	
		ss:0000>6474	

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

## # Division of 16-bit numbers

assume cs:code, ds:data

data segment

a dw 4648h

b dw 0004h

c dw 00h

data ends

code segment

start:

```
    mov ax,data
```

```
    mov ds,ax
```

```
    mov ax,0000h
```

```
    mov bx,0000h
```

```
    mov ax,a
```

```
    mov bx,b
```

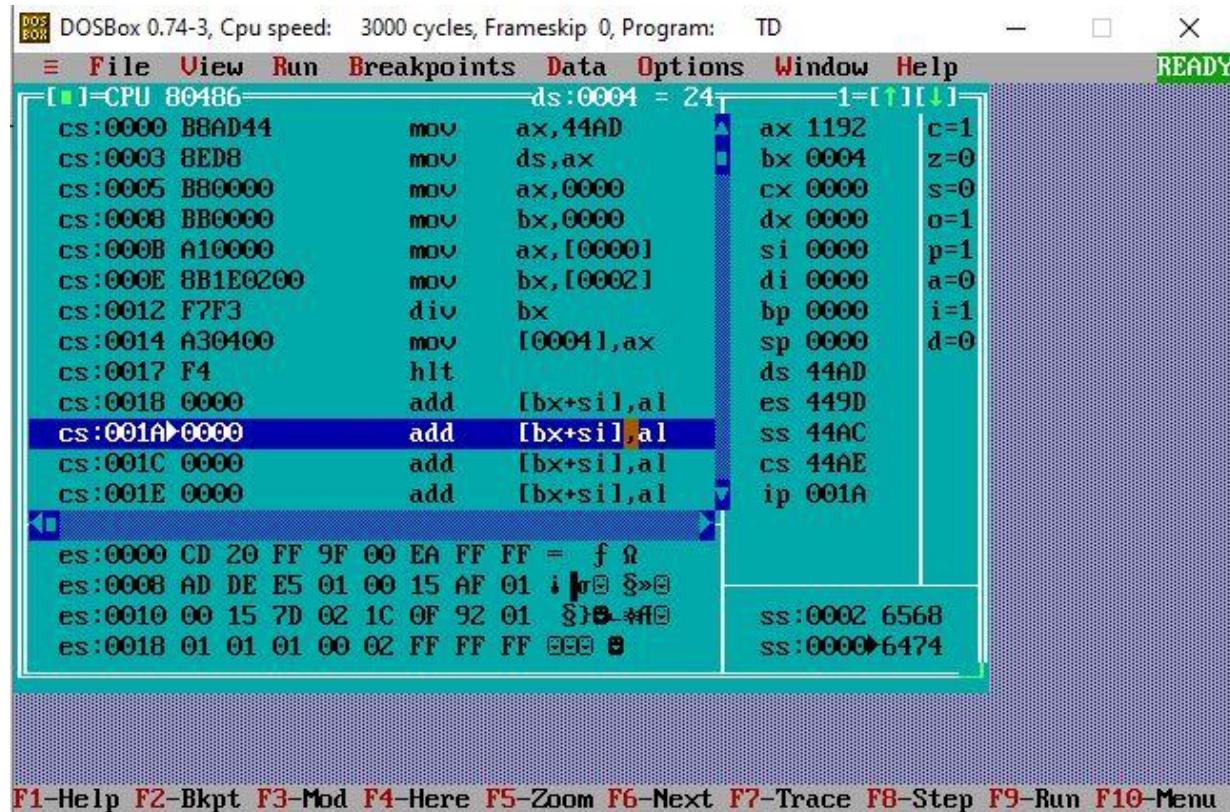
```
    div bx
```

```
    mov c,ax
```

```
    HLT
```

code ends

end start



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TD

READY

Register	Value	Description
ax	1192	c=1
bx	0004	z=0
cx	0000	s=0
dx	0000	o=1
si	0000	p=1
di	0000	a=0
bp	0000	i=1
sp	0000	d=0
ds	44AD	
es	449D	
ss	44AC	
cs	44AE	
ip	001A	
ss	0002 6568	
ss	0000 6474	

Assembly code:

```
[CPU 80486] ds:0004 = 24 1=[↑][↓]
cs:0000 B8AD44      mov    ax,44AD
cs:0003 8ED8        mov    ds,ax
cs:0005 B80000      mov    ax,0000
cs:0008 BB0000      mov    bx,0000
cs:000B A10000      mov    ax,[0000]
cs:000E 8B1E0200    mov    bx,[0002]
cs:0012 F7F3        div    bx
cs:0014 A30400      mov    [0004],ax
cs:0017 F4          hlt
cs:0018 0000        add    [bx+sil],al
cs:001A 0000        add    [bx+sil],al
cs:001C 0000        add    [bx+sil],al
cs:001E 0000        add    [bx+sil],al
es:0000 CD 20 FF 9F 00 EA FF FF = f 
es:0008 AD DE E5 01 00 15 AF 01 i 
es:0010 00 15 7D 02 1C 0F 92 01 S 
es:0018 01 01 01 00 02 FF FF FF 0000 S
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

**CONCLUSION:** Thus using Doc and GUI we have successfully implemented the different mathematical concept i.e Learnt about the mov instruction and assembler directives such db, segment, assume, register, etc and implemented it using TASM and DosBox



AET's  
**Atharva College of Engineering, Malad(W)**

Approved by AICTE, New Delhi, DTE, Mumbai  
Affiliated to University of Mumbai, ISO certified 9001:2015

**Department of Information Technology**  
**Academic Year: 2021-22**