

AIDS MICROPROCESSOR LAB S21 BATCH (2023-24)

Experiment 2(a) Title: Assembly language programming to add two 16 bit hexadecimal numbers using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 29/01/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 29/01/2024

Assembly language code

```
data segment      # Data segment

a dw 0307h      # variable a define first 16 bit number
b dw 0502h      # variable b define second 16 bit number
sum dw ?        #variable sum define result of addition of two 16 bit data
                 Numbers
data ends        # Assemble directives

code segment

assume cs:code, ds:data  # Assemble directives defining cs and ds
start:             # start of code segment

        mov ax,data      # moving data to ax register- initialization process
        mov ds,ax          # moving ax to dx register -- initialization process
        mov ax,a            # moving first number to ax register
        mov bx,b            # moving second number to bx register
        add ax,bx           # Addition of bx number to ax and result stored in ax
        mov sum,ax          # result stored in ax moved to variable sum
        int 3               # breakpoint interrupt

code ends        # Assembler directives to end code
end start         # Assembler directives to end start
```

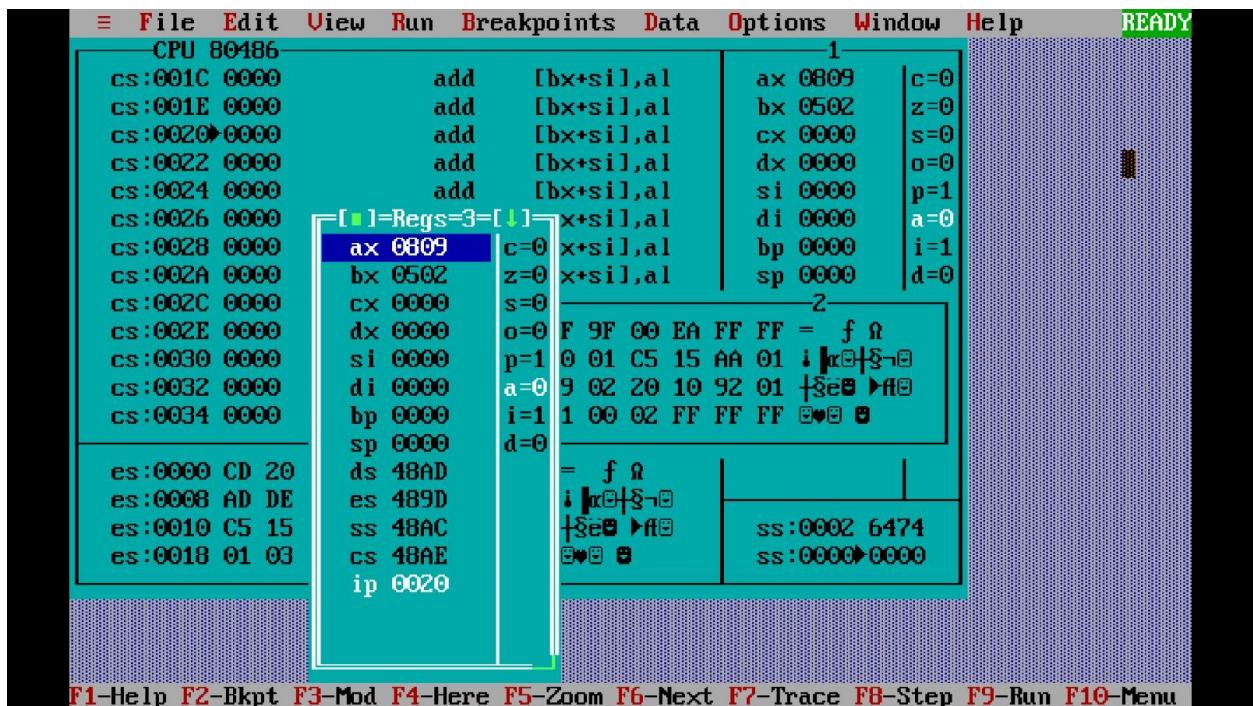
Result of addition:

The screenshot shows an 80486 assembly debugger interface. The CPU window displays assembly code with the instruction `add [bx+si],al` repeated 16 times at address `cs:001C 0000`. The Registers window shows the following register values:

ax	0809	c=0
bx	0502	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	48AD	
es	489D	
ss	48AC	
cs	48AE	
ip	0020	

The Dump window shows memory dump data starting at address `es:0000 CD 20 FF 9F 00 EA FF FF`, which is the result of the addition. The Registers window is labeled "Registers".

This screenshot shows the same debugger interface after expanding the Dump window. The Dump window now displays the full 16 bytes of the addition result, from `es:0000` to `es:0018`. The Registers window is labeled "Registers".



Experiment 2(b) Title: Assembly language programming to add two 16 bit hexadecimal numbers with carry using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 29/01/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 29/01/2024

Assembly language code

data segment

Data segment

a dw 0307h

variable a define first 16 bit number

b dw 0502h

variable b define second 16 bit number

sum dw ?

#variable sum define result of addition of two 16 bit

data Numbers

carry dw ?

Variable 'Carry' defines a flag to indicate if there is a carry during addition

```

data ends          # Assemble directives

code segment

assume cs:code, ds:data    # Assemble directives defining cs and ds
start:                  # start of code segment

        mov ax,data
        mov ds,ax
        mov cx,0000h
        mov ax,a
        mov bx,b
        add ax,bx
        jnc x
        inc cx
        x: mov sum,ax
           mov carry,cx
           int 3

        # moving data to ax register- initialization process
        # moving ax to dx register -- initialization process
        # Initialize cx register to 0 for counting carry
        # moving first number to ax register
        # moving second number to bx register
        # Addition of bx number to ax and result stored in ax
        # Jump to 'x' label if there is no carry
        # Increment cx if there is a carry
        # result stored in ax moved to variable sum
        # Move the carry count in cx to the variable 'carry'
        # breakpoint interrupt

code ends          # Assembler directives to end code
end start         # Assembler directives to end start

```

Result of addition:

The screenshot shows a debugger window with the following details:

- Assembly View:** Displays the assembly code from the previous listing.
- Registers View:** Shows the state of CPU registers. The carry flag (c) is set to 1. Other flags (z, s, o, p, a, i, d) are 0. Registers contain values: ax=1243, bx=8202, cx=0000, dx=0000, si=0000, di=0000, bp=0000, sp=0000, ds=48AD, es=489D, ss=48AC, cs=48AE, ip=0013.
- Stack Dump View:** Shows memory dump starting at address 489D:0000, with value 6474.
- Registers View (Bottom):** Shows the current state of the registers.

At the bottom, a menu bar includes F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, and F10-Menu.

READY

CPU 80486				1	
cs:0000 B8AD48	mov	ax,48AD		ax 1243	c=1
cs:0003 8ED8	mov	ds,ax		bx 8202	z=0
cs:0005 B90000	mov	cx,0000		cx 0000	s=0
cs:0008 A10000	mov	ax,[0000]		dx 0000	o=1
cs:000B 8B1E0200	mov	bx,[0002]		si 0000	p=0
cs:000F 03C3	add	ax,bx		di 0000	a=0
cs:0011 7301	jnb	0014		bp 0000	i=1
cs:0013>41	inc	cx		sp 0000	d=0
cs:0014 A30400	[]=Dump		2=[↑][↓]=		
cs:0017 890E06	es:0000 CD 20 FF 9F 00 EA FF FF	= f Ω			
cs:001B CC	es:0008 AD DE E0 01 C5 15 AA 01	↓ [x] [S] ↴			
cs:001C 0000	es:0010 C5 15 89 02 20 10 92 01	+ [ee] ► [ff]			
cs:001E 0000	es:0018 01 03 01 00 02 FF FF FF	■ [ee] ■			
es:0000 CD 20 FF 9F 00 EA FF FF	= f Ω				
es:0008 AD DE E0 01 C5 15 AA 01	↓ [x] [S] ↴				
es:0010 C5 15 89 02 20 10 92 01	+ [ee] ► [ff]				
es:0018 01 03 01 00 02 FF FF FF	■ [ee] ■				
ss:0002 6474					
ss:0000>0000					

Registers

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

READY

CPU 80486				1	
cs:0000 B8AD48	mov	ax,48AD		ax 1243	c=1
cs:0003 8ED8	mov	ds,ax		bx 8202	z=0
cs:0005 B90000	mov	cx,0000		cx 0000	s=0
cs:0008 A10000	mov	ax,[0000]		dx 0000	o=1
cs:000B 8B1E0200	mov	bx,[0002]		si 0000	p=0
cs:000F 03C3	add	ax,bx		di 0000	a=0
cs:0011 7301				bp 0000	i=1
cs:0013>41				sp 0000	d=0
cs:0014 A30400	[]=Regs=3=[↓]=		2		
cs:0017 890E06	ax 1243	c=1	14		
cs:001B CC	bx 8202	z=0			
cs:001C 0000	cx 0000	s=0			
cs:001E 0000	dx 0000	o=1	F 9F 00 EA FF FF	= f Ω	
	si 0000	p=0	0 01 C5 15 AA 01	↓ [x] [S] ↴	
	di 0000	a=0	9 02 20 10 92 01	+ [ee] ► [ff]	
	bp 0000	i=1	1 00 02 FF FF FF	■ [ee] ■	
	sp 0000	d=0			
es:0000 CD 20	ds 48AD	= f Ω			
es:0008 AD DE	es 489D	↓ [x] [S] ↴			
es:0010 C5 15	ss 48AC	+ [ee] ► [ff]		ss:0002 6474	
es:0018 01 03	cs 48AE	■ [ee] ■		ss:0000>0000	
ip 0013					

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

Experiment 2(c) Title: Assembly language programming to subtract two 16 bit hexadecimal numbers using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 05/02/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 05/02/2024

Assembly language code

```
data segment      # Data segment

a dw 9057h      # variable a define first 16 bit number
b dw 8022h      # variable b define second 16 bit number
diff dw ?        #variable diff define result of subtraction of two 16 bit
                  data Numbers
.
data ends        # Assemble directives

code segment

assume cs:code, ds:data    # Assemble directives defining cs and ds
start:                      # start of code segment

mov ax,data          # moving data to ax register- initialization process
mov ds,ax            # moving ax to dx register -- initialization process
mov ax,a             # moving first number to ax register
mov bx,b             # moving second number to bx register
add ax,bx            # Subtraction of bx number to ax and result stored in .
.
ax
mov diff,ax          # result stored in ax moved to variable diff
int 3                # breakpoint interrupt

code ends            # Assembler directives to end code
end start            # Assembler directives to end start
```

Result of subtraction:

READY

CPU 80486		1	
cs:0000 BBAD48	mov ax,48AD	ax 1035	c=0
cs:0003 8ED8	mov ds,ax	bx 8022	z=0
cs:0005 A10000	mov ax,[0000]	cx 0000	s=0
cs:0008 8B1E0200	mov bx,[0002]	dx 0000	o=0
cs:000C 2BC3	sub ax,bx	si 0000	p=1
cs:000E A30400	mov [0004],ax	di 0000	a=0
cs:0011 CC	int 03	bp 0000	i=1
cs:0012 0000	add [bx+si],al	sp 0000	d=0
cs:0014 0000	[]=Dump	Z=[↑][↓]	
cs:0016 0000	es:0000 CD 20 FF 9F 00 EA FF FF = f ↳		
cs:0018 0000	es:0008 AD DE E0 01 C5 15 AA 01 ↳		
cs:001A 0000	es:0010 C5 15 89 02 20 10 92 01 ↳		
cs:001C 0000	es:0018 01 03 01 00 02 FF FF FF ↳		
		ss:0002 6474	
		ss:0000 0000	

3 Registers

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

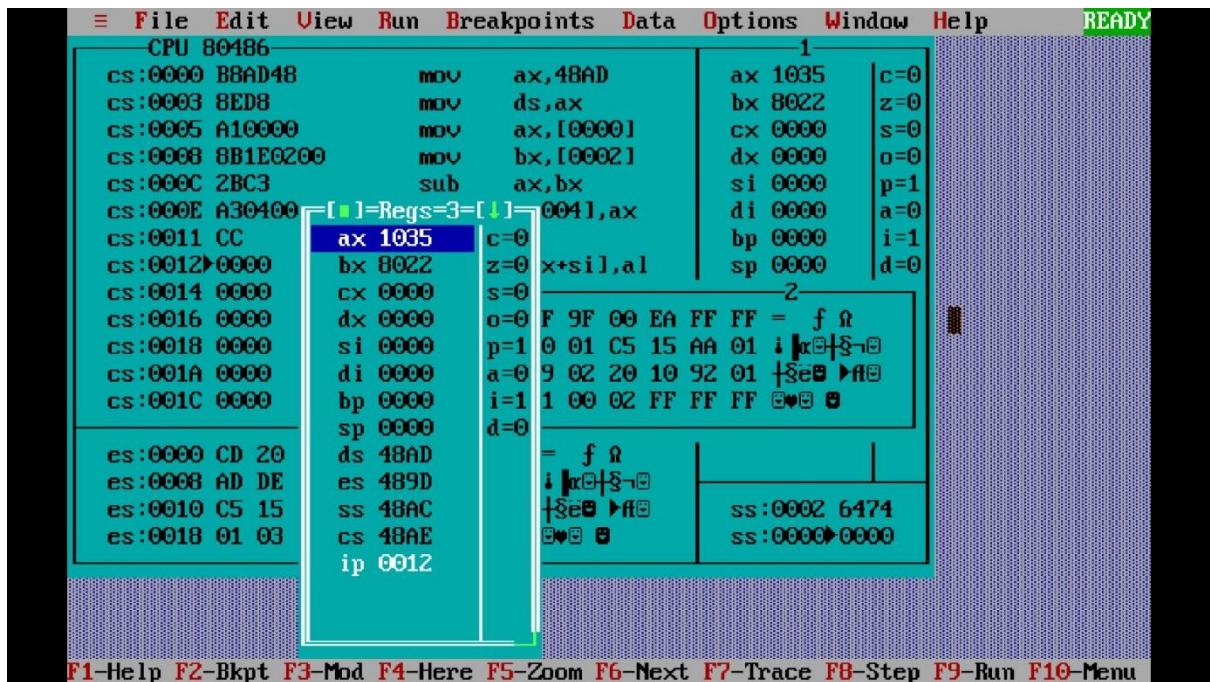
READY

CPU 80486		1=[↑][↓]	
cs:0000 BBAD48	mov ax,48AD	ax 1035	c=0
cs:0003 8ED8	mov ds,ax	bx 8022	z=0
cs:0005 A10000	mov ax,[0000]	cx 0000	s=0
cs:0008 8B1E0200	mov bx,[0002]	dx 0000	o=0
cs:000C 2BC3	sub ax,bx	si 0000	p=1
cs:000E A30400	mov [0004],ax	di 0000	a=0
cs:0011 CC	int 03	bp 0000	i=1
cs:0012 0000	add [bx+si],al	sp 0000	d=0
cs:0014 0000	add [bx+si],al	ds 48AD	
cs:0016 0000	add [bx+si],al	es 489D	
cs:0018 0000	add [bx+si],al	ss 48AC	
cs:001A 0000	add [bx+si],al	cs 48AE	
cs:001C 0000	add [bx+si],al	ip 0012	
		ss:0002 6474	
		ss:0000 0000	

2 Dump 489D:0000

3 Registers

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



Experiment 2(d) Title: Assembly language programming to subtract two 16 bit hexadecimal numbers with carry using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 05/02/2024

Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

Assembly language code

data segment	# Data segment
a dw 9305h	# variable a define first 16 bit number
b dw 6157h	# variable b define second 16 bit number
diff dw ?	#variable diff define result of subtraction of two 16 bit data Numbers
carry dw ?	# Variable 'Carry' defines a flag to indicate if there is a carry during subtraction

```

data ends          # Assemble directives

code segment

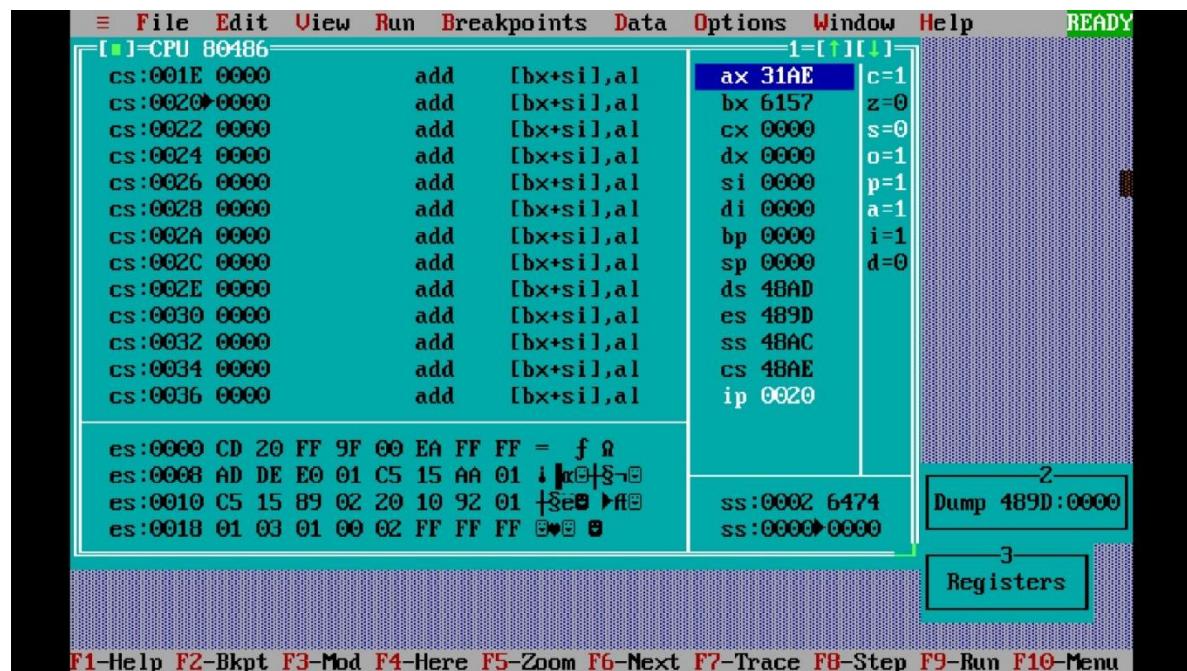
assume cs:code, ds:data    # Assemble directives defining cs and ds
start:                   # start of code segment

    mov ax,data      # moving data to ax register- initialization process
    mov ds,ax        # moving ax to dx register -- initialization process
    mov cx,0000h     # Initialize cx register to 0 for counting carry
    mov ax,a         # moving first number to ax register
    mov bx,b         # moving second number to bx register
    sub ax,bx       # Subtraction of bx number to ax and result stored in
                     .   ax
    jnc x           # Jump to 'x' label if there is no carry
    inc cx           # Increment cx if there is a carry
    x: mov diff,ax  # result stored in ax moved to variable diff
                     mov carry,cx  # Move the carry count in cx to the variable 'carry'
    int 3            # breakpoint interrupt

code ends          # Assembler directives to end code
end start          # Assembler directives to end start

```

Result of subtraction:



File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

cs:001E 0000	add	[bx+si],al	ax 31AE	c=1	
cs:0020 0000	add	[bx+si],al	bx 6157	z=0	
cs:0022 0000	add	[bx+si],al	cx 0000	s=0	
cs:0024 0000	add	[bx+si],al	dx 0000	o=1	
cs:0026 0000	add	[bx+si],al	si 0000	p=1	
cs:0028 0000	add	[bx+si],al	di 0000	a=1	
cs:002A 0000	add	[bx+si],al	bp 0000	i=1	
cs:002C 0000	add	[bx+si],al	sp 0000	d=0	
cs:002E 0000					
	[1]=Dump		2=[↑][↓]		
es:0000 CD 20 FF 9F 00 EA FF FF = f Ω					
es:0008 AD DE E0 01 C5 15 AA 01 i x S					
es:0010 C5 15 89 02 20 10 92 01 +Se ►Af					
es:0018 01 03 01 00 02 FF FF FF E					
es:0000 CD 20 FF 9F 00 EA FF FF = f Ω					
es:0008 AD DE E0 01 C5 15 AA 01 i x S					
es:0010 C5 15 89 02 20 10 92 01 +Se ►Af					
es:0018 01 03 01 00 02 FF FF FF E					
				ss:0002 6474	
				ss:0000 0000	

Registers

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

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

cs:001E 0000	add	[bx+si],al	ax 31AE	c=1	
cs:0020 0000	add	[bx+si],al	bx 6157	z=0	
cs:0022 0000	add	[bx+si],al	cx 0000	s=0	
cs:0024 0000	add	[bx+si],al	dx 0000	o=1	
cs:0026 0000	add	[bx+si],al	si 0000	p=1	
cs:0028 0000	add	[bx+si],al	di 0000	a=1	
cs:002A 0000	add	[bx+si],al	bp 0000	i=1	
cs:002C 0000	add	[bx+si],al	sp 0000	d=0	
cs:002E 0000					
	[1]=Regs=3=[2]				
ax 31AE	c=1	x+si,al			
bx 6157	z=0	x+si,al			
cx 0000	s=0	x+si,al			
dx 0000	o=1		F 9F 00 EA FF FF = f Ω		
si 0000	p=1		0 01 C5 15 AA 01 i x S		
di 0000	a=1		9 02 20 10 92 01 +Se ►Af		
bp 0000	i=1		1 00 02 FF FF FF E		
sp 0000	d=0				
es:0000 CD 20			= f Ω		
es:0008 AD DE			i x S		
es:0010 C5 15			+Se ►Af		
es:0018 01 03			E		
				ss:0002 6474	
				ss:0000 0000	

Registers

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

Experiment 2(e) Title: Assembly language programming to multiply two 8 bit hexadecimal numbers using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 05/02/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 05/02/2024

Assembly language code

```
data segment      # Data segment
a dB 0Fh          # variable a define first 8 bit number
b dB 09h          # variable b define second 8 bit number
mult dw ?         #variable mult define result of multiplication of two 8 bit .
.                  data Numbers

data ends          # Assemble directives

code segment
assume cs:code, ds:data    # Assemble directives defining cs and ds
start:                 # start of code segment

mov ax,data          # moving data to ax register- initialization process
mov ds,ax            # moving ax to dx register -- initialization process
mov al,a             # moving first number to al register
mov bl,b             # moving second number to bl register
mul bl              # multiplication of two numbers
mov mult,ax          # result stored in ax moved to variable mult
mov ah, 4Ch           # breakpoint interrupt
int 21h

code ends          # Assembler directives to end code
end start           # Assembler directives to end start
```

Result of multiplication:

File Edit View Run Breakpoints Data Options Window Help READY

CPU 80486

			1	
cs:0000	B8AD48	mov	ax,48AD	ax 0087 c=0
cs:0003	8ED8	mov	ds,ax	bx 0009 z=0
cs:0005	A00000	mov	al,[0000]	cx 0000 s=0
cs:0008	8A1E0100	mov	bl,[0001]	dx 0000 o=0
cs:000C	F6E3	mul	bl	si 0000 p=0
cs:000E	89160200	mov	[0002],dx	di 0000 a=0
cs:0012	►B44C	mov	ah,4C	bp 0000 i=1
cs:0014	CD21	int	21	sp 0000 d=0
cs:0016	0000			
cs:0018	0000			
cs:001A	0000			
cs:001C	0000			
cs:001E	0000			

[]=Dump 2=[↑][↓]

es:0000 CD 20 FF 9F 00 EA FF FF = f ↴	
es:0008 AD DE E0 01 C5 15 AA 01 ↴ ↵ ↷ ↶	
es:0010 C5 15 89 02 20 10 92 01 ↴ ↵ ↷ ↶	
es:0018 01 03 01 00 02 FF FF FF ↴ ↵ ↷ ↶	
ss:0002 6474	
ss:0000>0000	

Registers

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

File Edit View Run Breakpoints Data Options Window Help READY

[]=CPU 80486

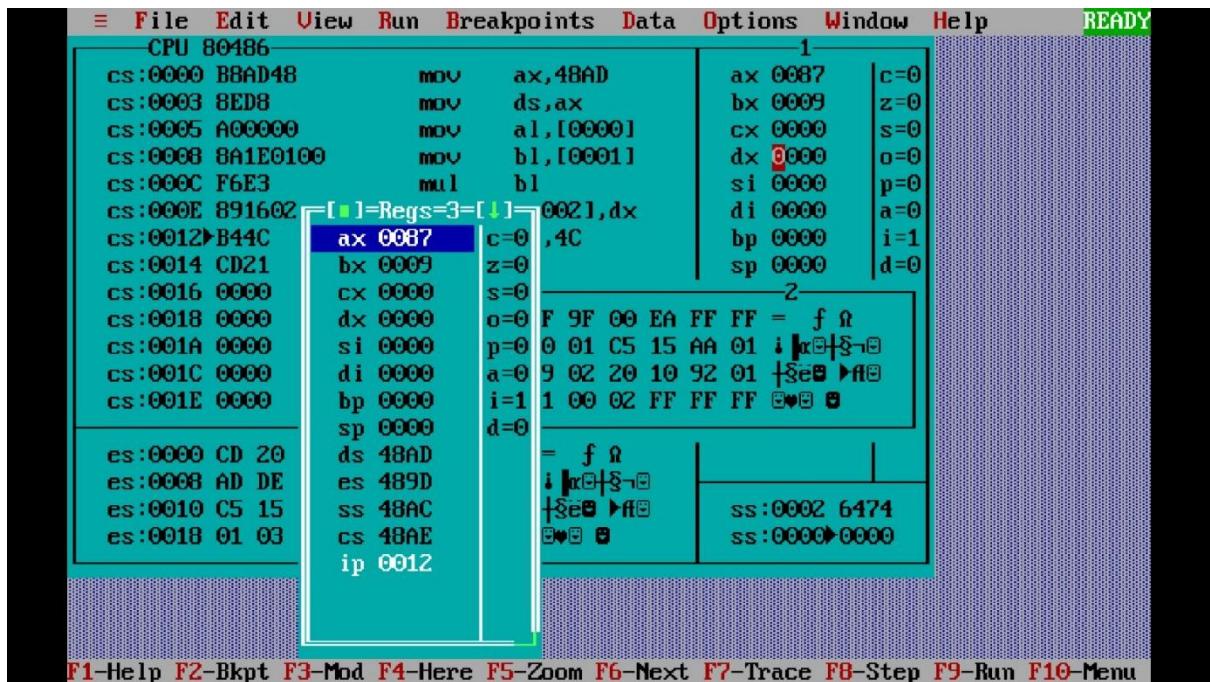
			1=[↑][↓]	
cs:0000	B8AD48	mov	ax,48AD	ax 0087 c=0
cs:0003	8ED8	mov	ds,ax	bx 0009 z=0
cs:0005	A00000	mov	al,[0000]	cx 0000 s=0
cs:0008	8A1E0100	mov	bl,[0001]	dx 0000 o=0
cs:000C	F6E3	mul	bl	si 0000 p=0
cs:000E	89160200	mov	[0002],dx	di 0000 a=0
cs:0012	►B44C	mov	ah,4C	bp 0000 i=1
cs:0014	CD21	int	21	sp 0000 d=0
cs:0016	0000	add	[bx+sil],al	ds 48AD
cs:0018	0000	add	[bx+sil],al	es 489D
cs:001A	0000	add	[bx+sil],al	ss 48AC
cs:001C	0000	add	[bx+sil],al	cs 48AE
cs:001E	0000	add	[bx+sil],al	ip 0012

[]=Dump 2=[↑][↓]

es:0000 CD 20 FF 9F 00 EA FF FF = f ↴	
es:0008 AD DE E0 01 C5 15 AA 01 ↴ ↵ ↷ ↶	
es:0010 C5 15 89 02 20 10 92 01 ↴ ↵ ↷ ↶	
es:0018 01 03 01 00 02 FF FF FF ↴ ↵ ↷ ↶	
ss:0002 6474	
ss:0000>0000	

Registers

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



Experiment 2(f) Title: Assembly language programming to multiply two unsigned 16 bit numbers using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 05/02/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 05/02/2024

Assembly language code

data segment # Data segment

```
a dw 5B6Ah                          # variable a define first 16 bit number
b dw 7809h                          # variable b define second 16 bit number
res_low dw ?
res_high dw ?
```

data ends # Assemble directives

code segment

```
assume cs:code, ds:data      # Assemble directives defining cs and ds
start:                      # start of code segment
```

```
mov ax,data                 # moving data to ax register- initialization process
mov ds,ax                   # moving ax to dx register -- initialization process
mov ax,a                    # moving first number to ax register
mov bx,b                    # moving second number to bx register
mul bx                     # multiplication of two numbers
mov res_low,ax              #move the values to variable
mov res_high,dx
mov ah, 004Ch
int 21h                     # breakpoint interrupt
```

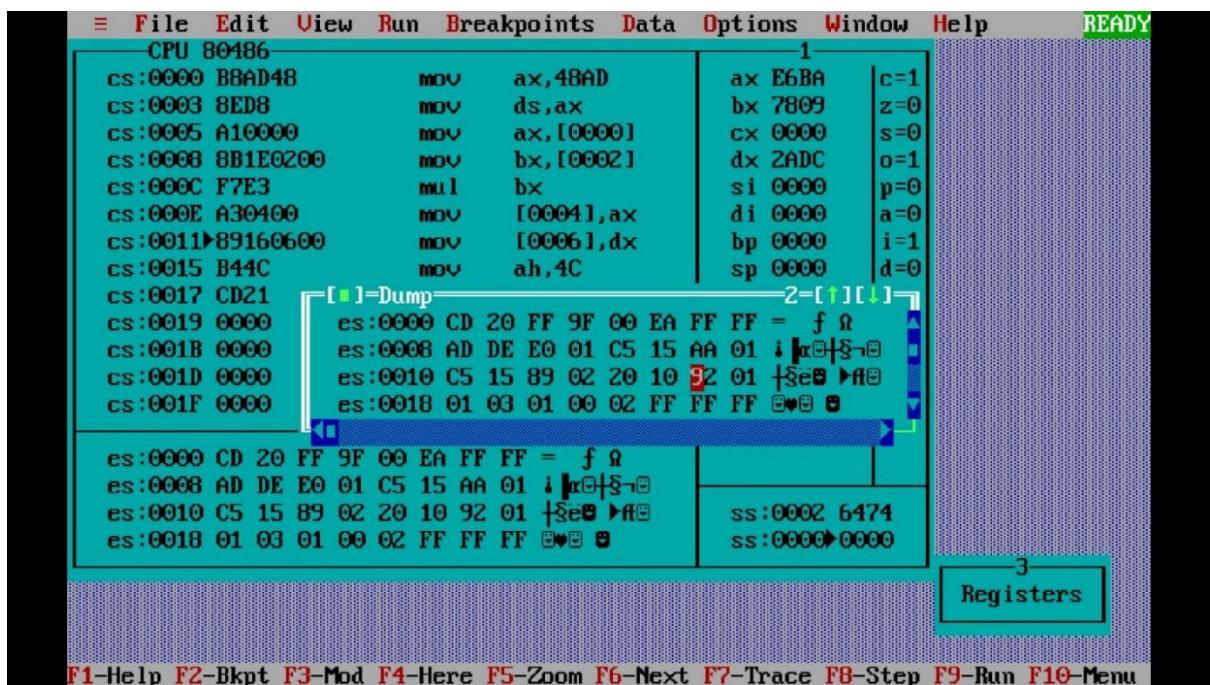
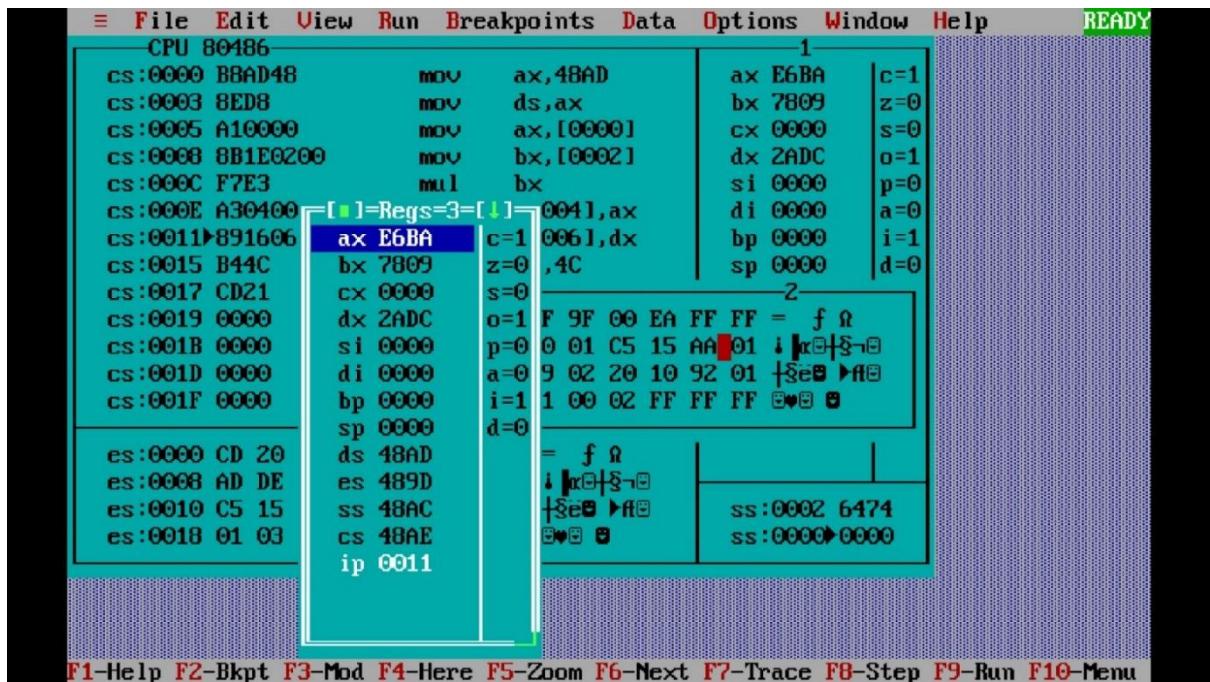
```
code ends                  # Assembler directives to end code
end start                   # Assembler directives to end start
```

Result of multiplication:

The screenshot shows a debugger interface with the following panes:

- Assembly Pane:** Displays assembly code from memory address CS:0000 to CS:001F. The code includes instructions like MOV AX,4BAD, MOV DS,AX, and INT 21.
- Registers Pane:** Shows the CPU register state. The AX register is highlighted in blue and contains the value E6BA. Other registers shown include BX, CX, DX, SI, DI, BP, SP, DS, ES, SS, CS, and IP.
- Stack Dump Pane:** Shows memory dump from SS:0002 to SS:0000. The dump area contains binary data: CD 20 FF 9F 00 EA FF FF, AD DE E0 01 C5 15 AA 01, C5 15 89 02 20 10 92 01, and 01 03 01 00 02 FF FF FF.
- Registers Pane:** Shows a list of registers with their current values: AX E6BA, BX 7809, CX 0000, DX 2ADC, SI 0000, DI 0000, BP 0000, SP 0000, DS 48AD, ES 489D, SS 48AC, CS 48AE, and IP 0011.

At the bottom, a menu bar lists keyboard shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, and F10-Menu.



Experiment 2(g) Title: Assembly language programming to multiply two signed 16 bit numbers using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 05/02/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 05/02/2024

Assembly language code

```
data segment                # Data segment
A dB 7A5Bh                 # define 16 bit numbers
B dB 7A5Bh
RES_LOW dw 0
RES_HIGH dw 0

data ends                  # Assemble directives code segment
assume CS: code, DS: data  # Assemble directives defining cs and
.                           ds

start:                     # start of code segment
    mov ax, data
    mov ds, ax

    MOV AL, A                 # move to values to register
    MOV BL, B
    MUL BL                   # multiply the two numbers
    MOV RES_LOW, AX           # move the value to variable
    MOV RES_HIGH, DX
    MOV AH, 4CH
    INT 21H                  # interrupt handle
```

```

code ends          # Assembler directives to end code
end start         # Assembler directives to end start

```

Result of multiplication:

The screenshot shows a debugger window with the following details:

- Registers:**

ax	00F2	c=0
bx	000F	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	48AD	
es	489D	
ss	48AC	
cs	48AE	
ip	000C	
- Stack:** ss:0002 6474
ss:0000 0000
- Memory Dump:** es:0000 CD 20 FF 9F 00 EA FF FF = f ??
es:0008 AD DE E0 01 C5 15 AA 01 i ??+??
es:0010 C5 15 89 02 20 10 92 01 +\$e? ??
es:0018 01 03 01 00 02 FF FF FF ?? ??
- Assembly View:** The assembly code shows the multiplication instruction at address CS:000C F7E3.

Experiment 2(h) Title: Assembly language programming to divide 16 bit hexadecimal number with a 8 bit hexadecimal number using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 05/02/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 05/02/2024

Assembly language code

```
data segment          # Data segment
a dw 0008h           # variable a define 16 bit number
b db 02h             # variable b define 8 bit number
ans dw ?             #variable diff define result of division
.
data ends            # Assemble directives

code segment
assume cs:code, ds:data    # Assemble directives defining cs and ds
start:                  # start of code segment

mov ax,data           # moving data to ax register- initialization process
mov ds,ax              # moving ax to dx register -- initialization process
mov ax,a               # moving first number to ax register
mov bl,b               # moving second number to bx register
div bl                # Division and result stored in ax
mov ans,ax             # result stored in ax moved to variable ans
int 3                 # breakpoint interrupt

code ends             # Assembler directives to end code
end start              # Assembler directives to end start
```

Result of division:

READY

CPU 80486				1
cs:001C 0000	add	[bx+si],al	ax 0004	c=0
cs:001E 0000	add	[bx+si],al	bx 0002	z=0
cs:0020 0000	add	[bx+si],al	cx 0000	s=0
cs:0022 0000	add	[bx+si],al	dx 0000	o=0
cs:0024 0000	add	[bx+si],al	si 0000	p=0
cs:0026 0000	add	[bx+si],al	di 0000	a=0
cs:0028 0000	add	[bx+si],al	bp 0000	i=1
cs:002A 0000	add	[bx+si],al	sp 0000	d=0
cs:002C 0000				
es:0000 CD 20 FF 9F 00 EA FF FF	= f	Ω		
es:0008 AD DE E0 01 C5 15 AA 01	↓	x⊕ S-⊖		
es:0010 C5 15 89 02 20 10 92 01	+Se⊕	►fl⊖		
es:0018 01 03 01 00 02 FF FF FF	⊖v⊕ ⊕			
ss:0002 6474				
es:0034 0000				
ss:0000>0000				

[1]=Dump [2=↑][↓] [3 Registers]

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

READY

CPU 80486				1
cs:0000 B8AD48	mov	ax,48AD	ax 0008	c=0
cs:0003 8ED8	mov	ds,ax	bx 0002	z=0
cs:0005 A10000	mov	ax,[0000]	cx 0000	s=0
cs:0008 8A1E0200	mov	bl,[0002]	dx 0000	o=0
cs:000C>F6F3	div	bl	si 0000	p=0
cs:000E A30300		= [1]=Regs=3=[2]=0031,ax	di 0000	a=0
cs:0011 CC	ax 0008	c=0	bp 0000	i=1
cs:0012 0000	bx 0002	z=0 x+si],al	sp 0000	d=0
cs:0014 0000	cx 0000	s=0		
cs:0016 0000	dx 0000	o=0 F 9F 00 EA FF FF	= f	Ω
cs:0018 0000	si 0000	p=0 0 01 C5 15 AA 01	↓	x⊕ S-⊖
cs:001A 0000	di 0000	a=0 9 02 20 10 92 01	+Se⊕	►fl⊖
cs:001C 0000	bp 0000	i=1 1 00 02 FF FF FF	⊖v⊕ ⊕	
sp 0000	ip 000C	d=0		
ds 48AD		= f		
es:0000 CD 20		↓		
es:0008 AD DE		x⊕ S-⊖		
es:0010 C5 15		+Se⊕		
es:0018 01 03		⊖v⊕ ⊕	ss:0002 6474	
cs 48AE				ss:0000>0000

[1]=Dump [2=↑][↓] [3 Registers]

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

READY

CPU 80486		1	
cs:0000 BBAD48	mov ax,48AD	ax 0004	c=0
cs:0003 8ED8	mov ds,ax	bx 0002	z=0
cs:0005 A10000	mov ax,[0000]	cx 0000	s=0
cs:0008 8A1E0200	mov bl,[0002]	dx 0000	o=0
cs:000C F6F3	div bl	si 0000	p=0
cs:000E A30300	[1]=Regs=3=[2]=0031,ax	di 0000	a=0
cs:0011 CC	ax 0004	bp 0000	i=1
cs:0012 0000	bx 0002	sp 0000	d=0
cs:0014 0000	cx 0000		
cs:0016 0000	dx 0000		
cs:0018 0000	si 0000		
cs:001A 0000	di 0000		
cs:001C 0000	bp 0000		
	sp 0000		
es:0000 CD 20	ds 48AD	= f ↴	
es:0008 AD DE	es 489D	↓ [x] [S] ↴	
es:0010 C5 15	ss 48AC	+Se ↴ fl ↴	ss:0002 6474
es:0018 01 03	cs 48AE	SS []	ss:0000>0000
	ip 000E		

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

READY

[1]-CPU 80486		1=[↑][↓]	
cs:001C 0000	add [bx+s1l],al	ax 0004	c=0
cs:001E 0000	add [bx+s1l],al	bx 0002	z=0
cs:0020 0000	add [bx+s1l],al	cx 0000	s=0
cs:0022 0000	add [bx+s1l],al	dx 0000	o=0
cs:0024 0000	add [bx+s1l],al	si 0000	p=0
cs:0026 0000	add [bx+s1l],al	di 0000	a=0
cs:0028 0000	add [bx+s1l],al	bp 0000	i=1
cs:002A 0000	add [bx+s1l],al	sp 0000	d=0
cs:002C 0000	add [bx+s1l],al	ds 48AD	
cs:002E 0000	add [bx+s1l],al	es 489D	
cs:0030 0000	add [bx+s1l],al	ss 48AC	
cs:0032 0000	add [bx+s1l],al	cs 48AE	
cs:0034 0000	add [bx+s1l],al	ip 0026	
es:0000 CD 20 FF 9F 00 EA FF FF = f ↴			
es:0008 AD DE E0 01 C5 15 AA 01 ↓ [x] [S] ↴			
es:0010 C5 15 89 02 20 10 92 01 +Se ↴ fl ↴			
es:0018 01 03 01 00 02 FF FF FF SS []			
		ss:0002 6474	
		ss:0000>0000	

2 Dump 489D:0000

3 Registers

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

Experiment 2(i) Title: Assembly language programming to divide two signed 8 bit hexadecimal numbers using software tool TASM 1.4

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 05/02/2024

Batch: S2-1

Timing: 3:00-5:00

Date of Submission: 05/02/2024

Assembly language code

```
DATA SEGMENT          # Data segment
    NUM1 DB 34H      # define 8 bit numbers
    NUM2 DB 69H
    QUOTIENT DB ?
    REMAINDER DB ?
DATA ENDS            # Assemble directives

CODE SEGMENT          # Assemble directives defining cs and ds
ASSUME CS:CODE, DS:DATA
START:               # Assemble directives defining cs and ds
                    # start of code segment

    MOV AX, DATA
    MOV DS, AX
    MOV AX, 0000H      # move to values to register
    MOVAL, NUM1
    MOV BL, NUM2
    DIV BL             # divide the two numbers
    MOV REMAINDER, AH
    MOV QUOTIENT, AL
    INT 21H
CODE ENDS            # Assembler directives to end code
END START            # Assembler directives to end start
```

Result of division:

The screenshot shows the Immunity Debugger interface with the CPU window active. The assembly pane displays the following code:

```
cs:0000 B8AD48    mov    ax,48AD
cs:0003 8ED8    mov    ds,ax
cs:0005 B80000    mov    ax,0000
cs:0008 A00000    mov    al,[0000]
cs:000B 8A1E0100  mov    bl,[0001]
cs:000F F6F3    div    bl
cs:0011 88260300  mov    [0003],ah
cs:0015 A20200    mov    [0002],al
cs:0018 CD21    int    21
cs:001A 0000    add    [bx+si],al
cs:001C 0000    add    [bx+si],al
cs:001E 0000    add    [bx+si],al
cs:0020 0000    add    [bx+si],al
```

The registers pane shows the following register values:

Register	Value	Description
ax	3700	c=0
bx	0085	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	48AD	
es	489D	
ss	48AC	
cs	48AE	
ip	0018	

The stack pane shows the current stack state:

```
ss:0002 6474
ss:0000→0000
```

Experiment 2(j) Title: Assembly language programming to divide two signed 16 bit hexadecimal numbers using software tool TASM 1.4

Name of student: Meet Raut *Class Roll Number: 2201084*

Date of Performance: 05/02/2024

Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

Assembly language code

DATA SEGMENT # *Data segment*

NUM1 DB 1254H # define 16 bit numbers

NUM2 DB 1223H

QUOTIENT DB ?

REMAINDER DB

TA ENDS

CODE SEGMENT # Assemble directives defining cs and ds ASSUME

CS:CODE, DS:DATA # Assemble directives defining cs and ds

START: # start of code segment

MOV AX, DATA

MOV DS, AX

MOV AX, 0000H # move to values to register

MOVAL, NUM1

MOV BL, NUM2

DIV BL # divide the two numbers

MOV REMAINDER, AH # move to values to respective register

MOV QUOTIENT, AL

INT 21H

CODE ENDS # Assembler directives to end code

END START # Assembler directives to end start

Result of division :

The screenshot shows a debugger interface with the following details:

- Registers:** ax=0001, bx=1223, cx=0000, dx=0031, si=0000, di=0000, bp=0000, sp=0000, ds=48AD, es=489D, ss=48AC, cs=48AE, ip=0015.
- Stack:** ss:0002 6474, ss:0000 0000.
- Memory Dump:** Shows memory starting at address CS:0000 CD 20 FF 9F 00 EA FF FF = f ?
- Assembly View:** Displays the assembly code:

```
CS:0000 B8AD48    mov    ax,4BAD
CS:0003 8ED8    mov    ds,ax
CS:0005 A10000    mov    ax,[0000]
CS:0008 8B1E0200    mov    bx,[0002]
CS:000C F7F3    div    bx
CS:000E A30600    mov    [0006],ax
CS:0011 89160400    mov    [0004],dx
CS:0015 CD21    int    21
CS:0017 0000    add    [bx+si],al
CS:0019 0000    add    [bx+si],al
CS:001B 0000    add    [bx+si],al
CS:001D 0000    add    [bx+si],al
CS:001F 0000    add    [bx+si],al
```
- Status Bar:** F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

CONCLUSION: LO 1, LO 2, LO 3 mapped.
