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

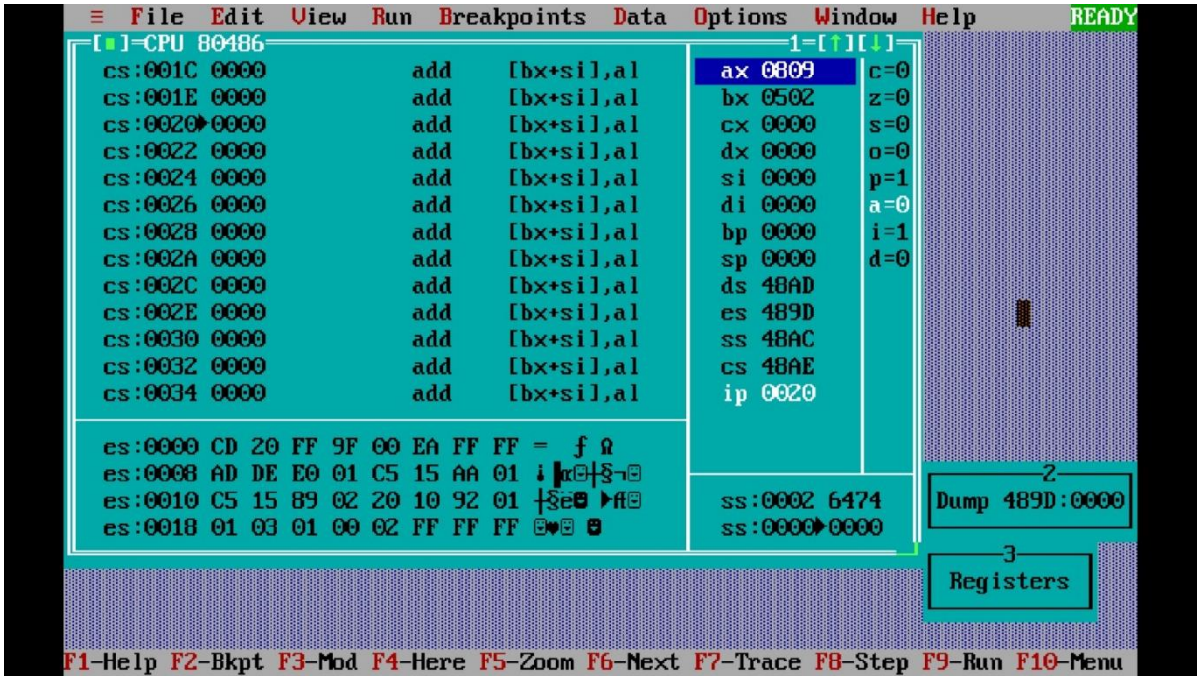
Date of Performance: 29/01/2024

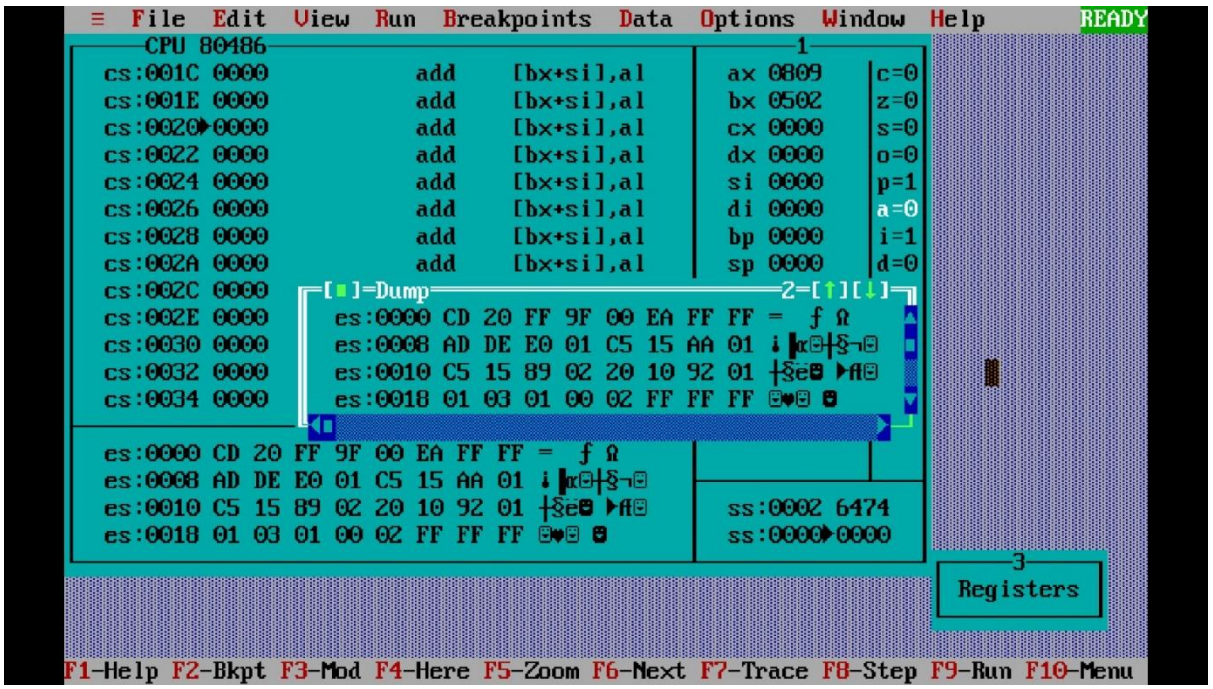
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 29/01/2024

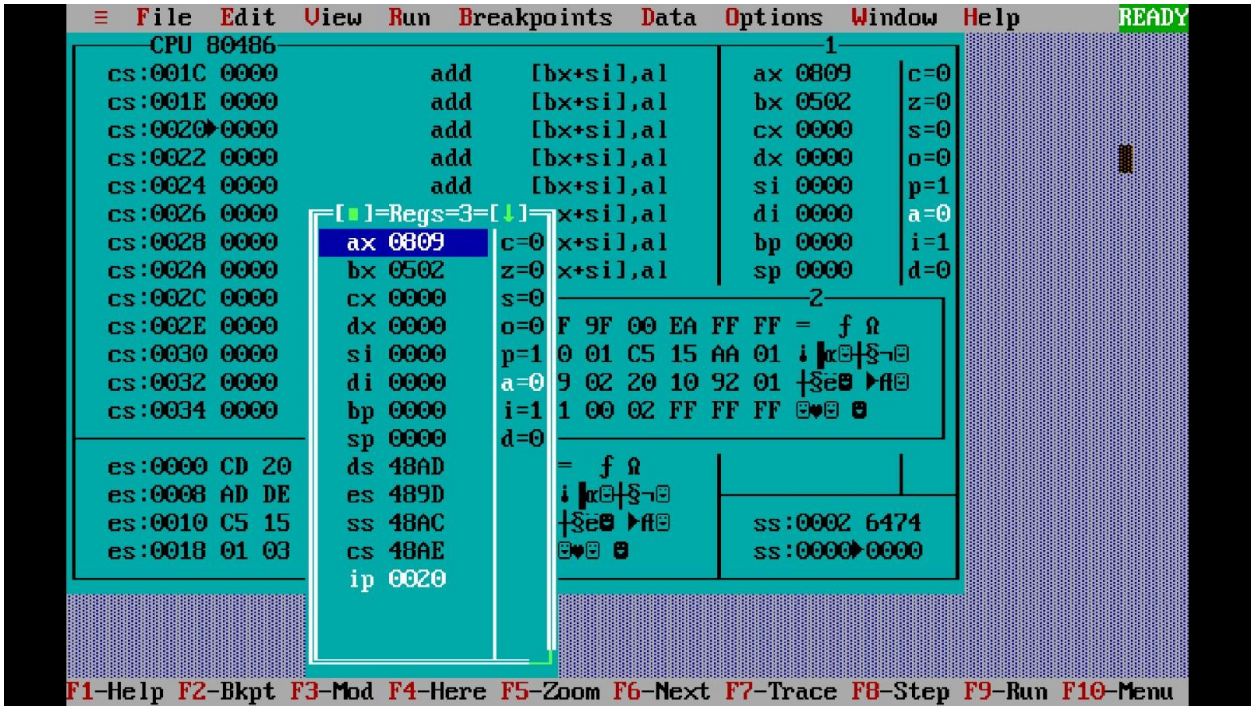
Assembly language code

```
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:







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

Name of student: Meet Raut Class Roll Number: 2201084

Date of Performance: 29/01/2024

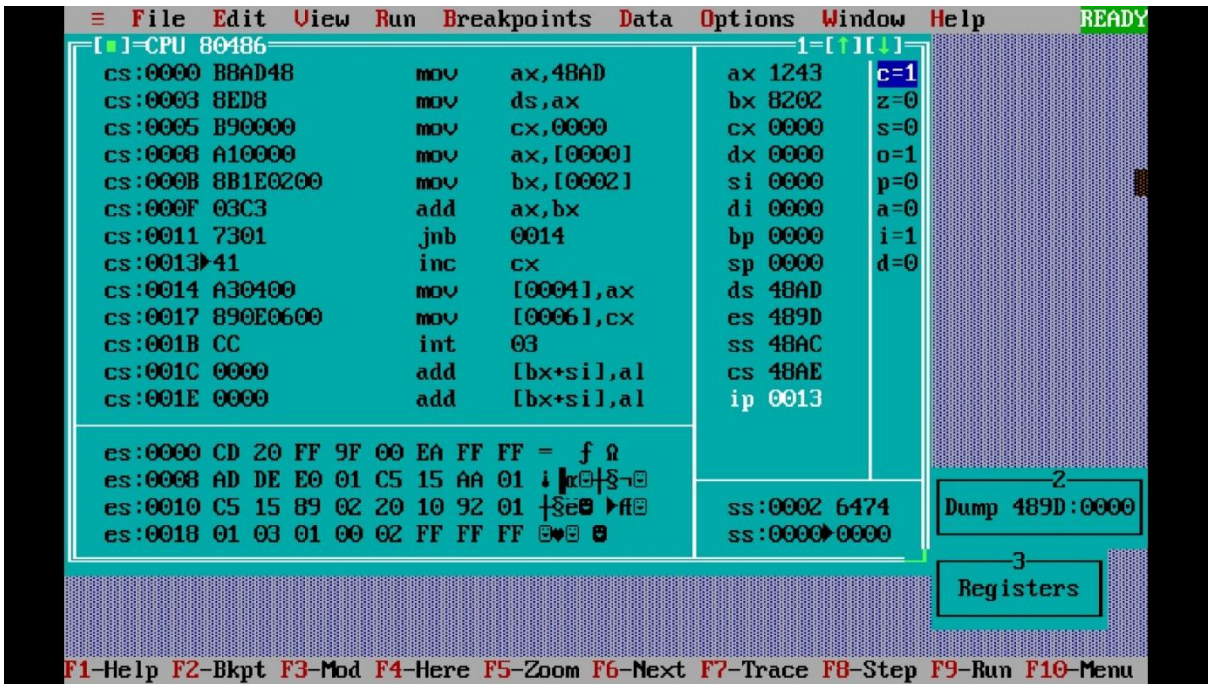
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 29/01/2024

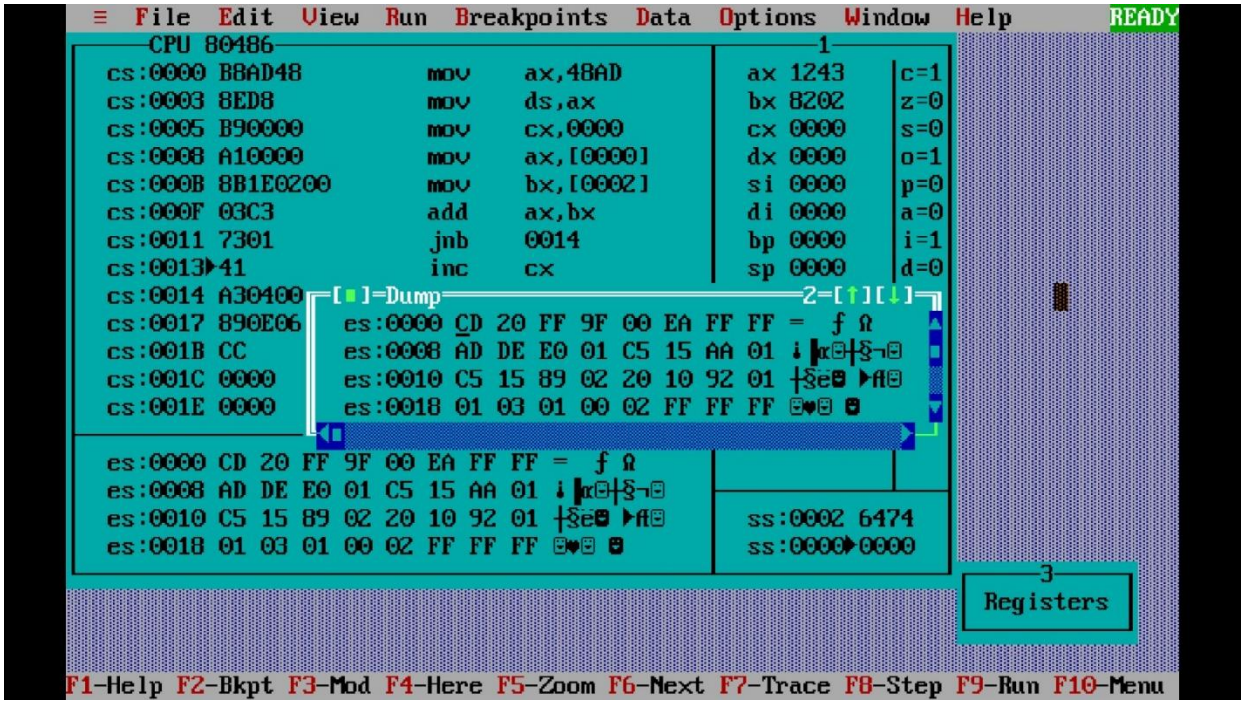
<u>Assembly language code</u>

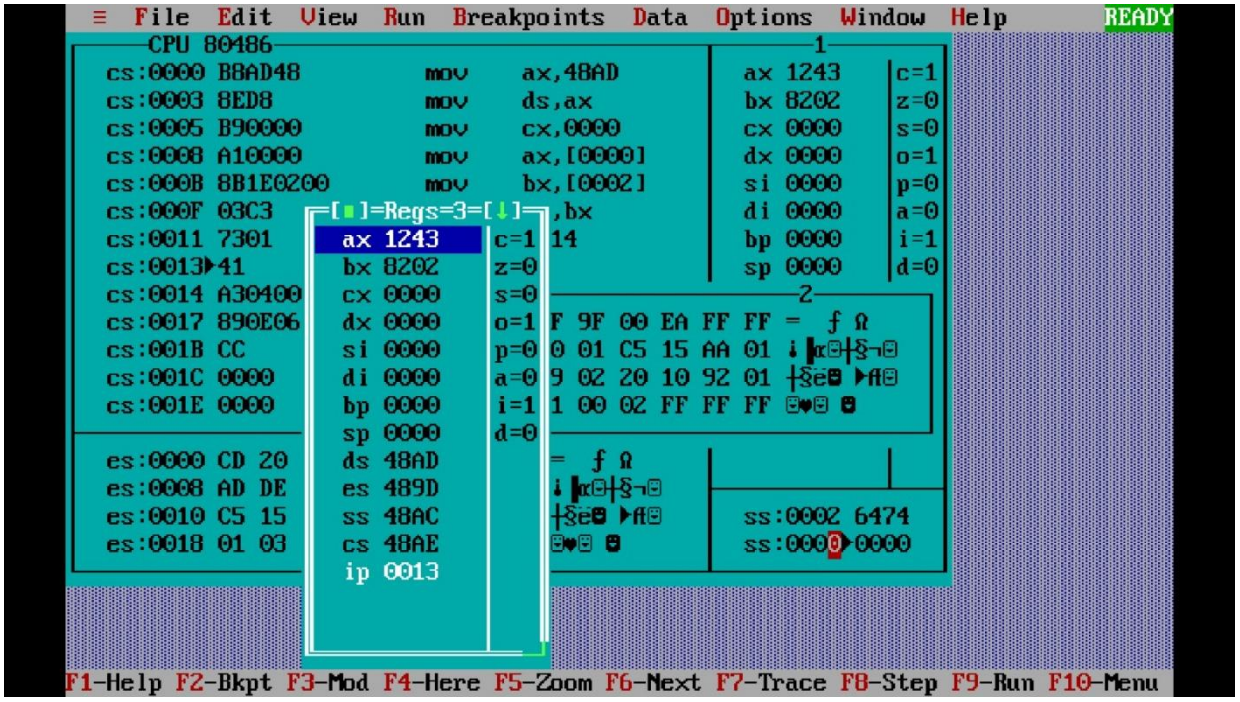
```
data segment            # Data segment

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                # 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
add ax,bx                  # Addition 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 sum,ax              # result stored in ax moved to variable sum
   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 addition:




File Edit View Run Breakpoints Data Options Window Help READY
CPU 80486
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 B90000 mov cx,0000
cs:0008 A10000 mov ax,[0000]
cs:000B 8B1E0200 mov bx,[0002]
cs:000F 03C3 add ax,bx
cs:0011 7301 jnb 0014
cs:0013 41 inc cx
cs:0014 A30400
cs:0017 890E06
cs:001B CC
cs:001C 0000
cs:001E 0000
[■]=Dump
2=[↑][↓]
es:0000 CD 20 FF 9F 00 EA FF FF
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
ax 1243
bx 8202
cx 0000
dx 0000
si 0000
di 0000
bp 0000
sp 0000
c=1
z=0
s=0
o=1
p=0
a=0
i=1
d=0
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


File Edit View Run Breakpoints Data Options Window Help READY
CPU 80486
[■]=Regs=3=[↓]
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
c=1
z=0
s=0
o=1
p=0
a=0
i=1
d=0
ss:0002 6474
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

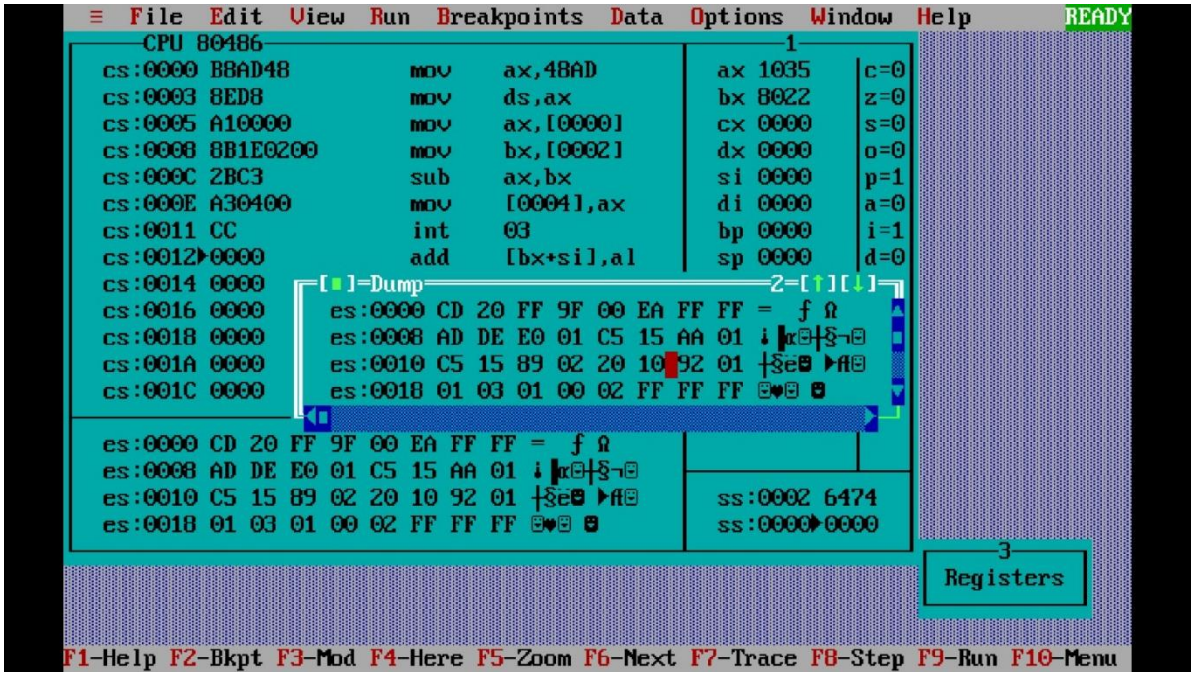
Date of Performance: 05/02/2024

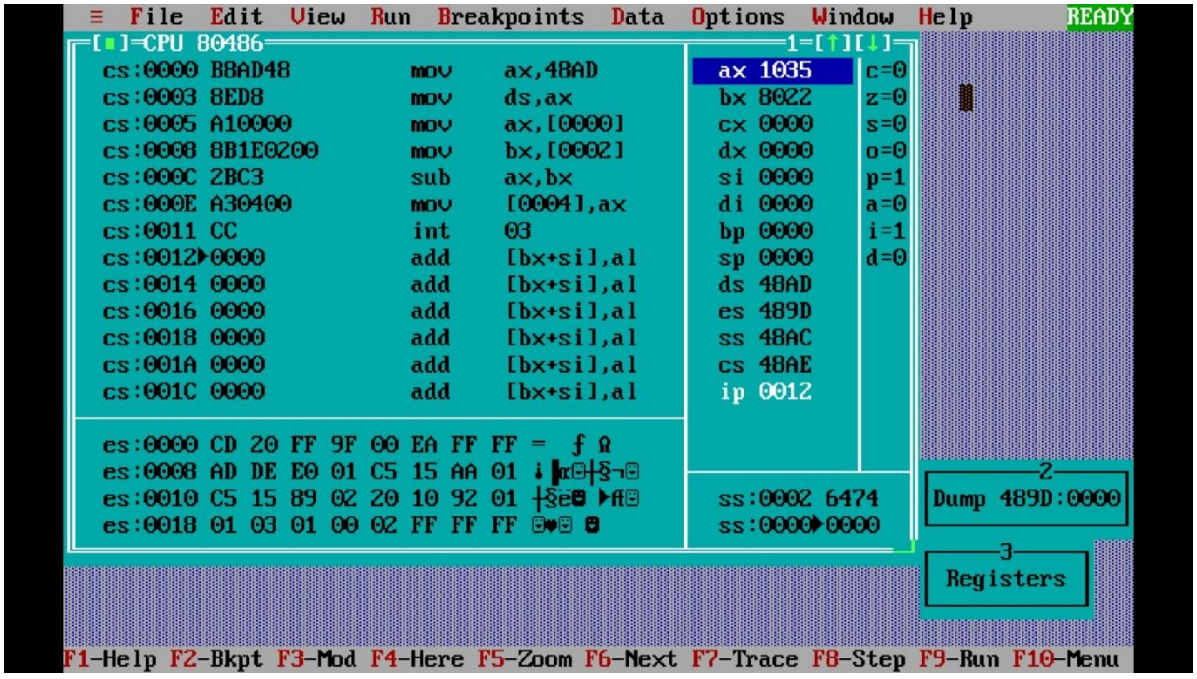
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

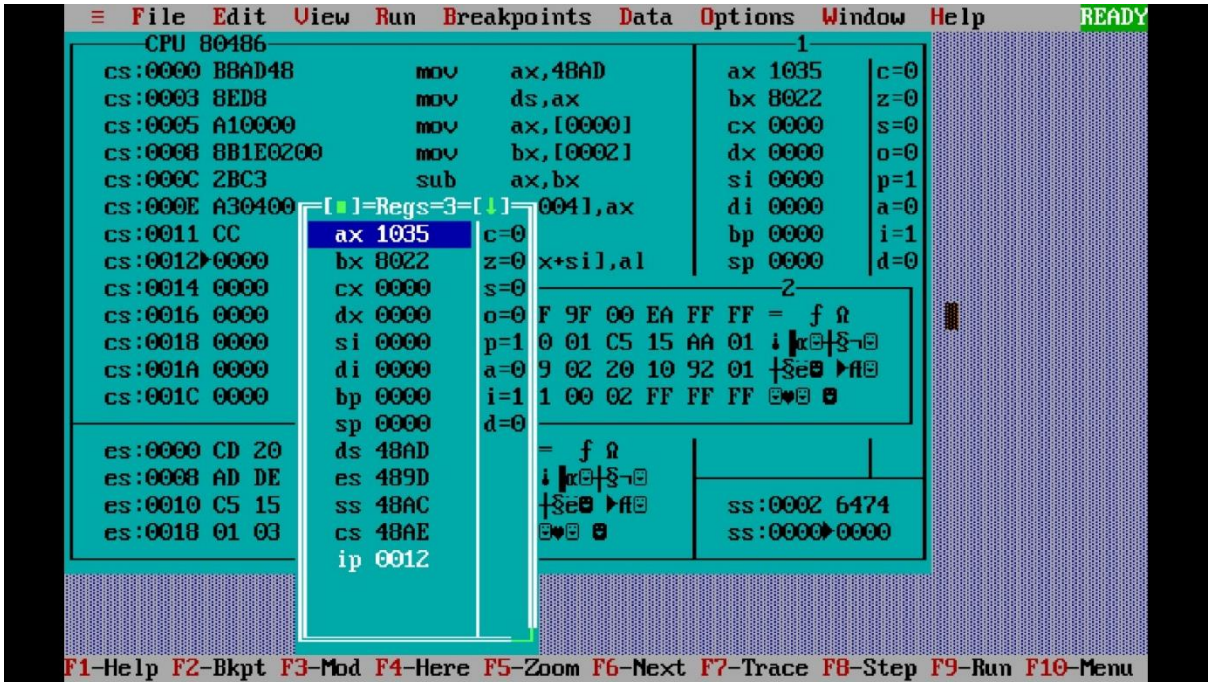
Assembly language code

```
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:


File Edit View Run Breakpoints Data Options Window Help READY
CPU 80486
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A10000 mov ax,[0000]
cs:0008 8B1E0200 mov bx,[0002]
cs:000C 2BC3 sub ax,bx
cs:000E A30400 mov [0004],ax
cs:0011 CC int 03
cs:0012 0000 add [bx+si],al
cs:0014 0000
cs:0016 0000
cs:0018 0000
cs:001A 0000
cs:001C 0000
Dump
es:0000 CD 20 FF 9F 00 EA FF FF
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
ax 1035
bx 8022
cx 0000
dx 0000
si 0000
di 0000
bp 0000
sp 0000
c=0
z=0
s=0
o=0
p=1
a=0
i=1
d=0
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:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A10000 mov ax,[0000]
cs:0008 8B1E0200 mov bx,[0002]
cs:000C 2BC3 sub ax,bx
cs:000E A30400 mov [0004],ax
cs:0011 CC int 03
cs:0012 0000 add [bx+si],al
cs:0014 0000 add [bx+si],al
cs:0016 0000 add [bx+si],al
cs:0018 0000 add [bx+si],al
cs:001A 0000 add [bx+si],al
cs:001C 0000 add [bx+si],al
es:0000 CD 20 FF 9F 00 EA FF FF
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
ax 1035
bx 8022
cx 0000
dx 0000
si 0000
di 0000
bp 0000
sp 0000
ds 48AD
es 489D
ss 48AC
cs 48AE
ip 0012
c=0
z=0
s=0
o=0
p=1
a=0
i=1
d=0
ss:0002 6474
ss:0000 0000
Dump 489D:0000
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

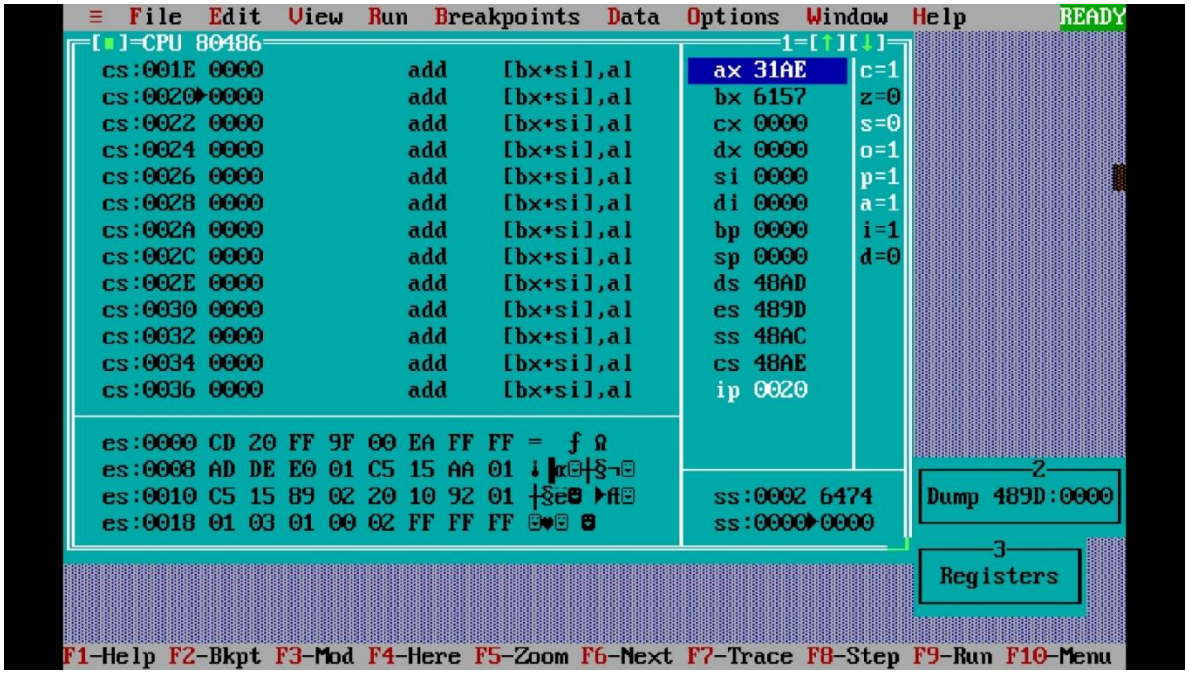
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

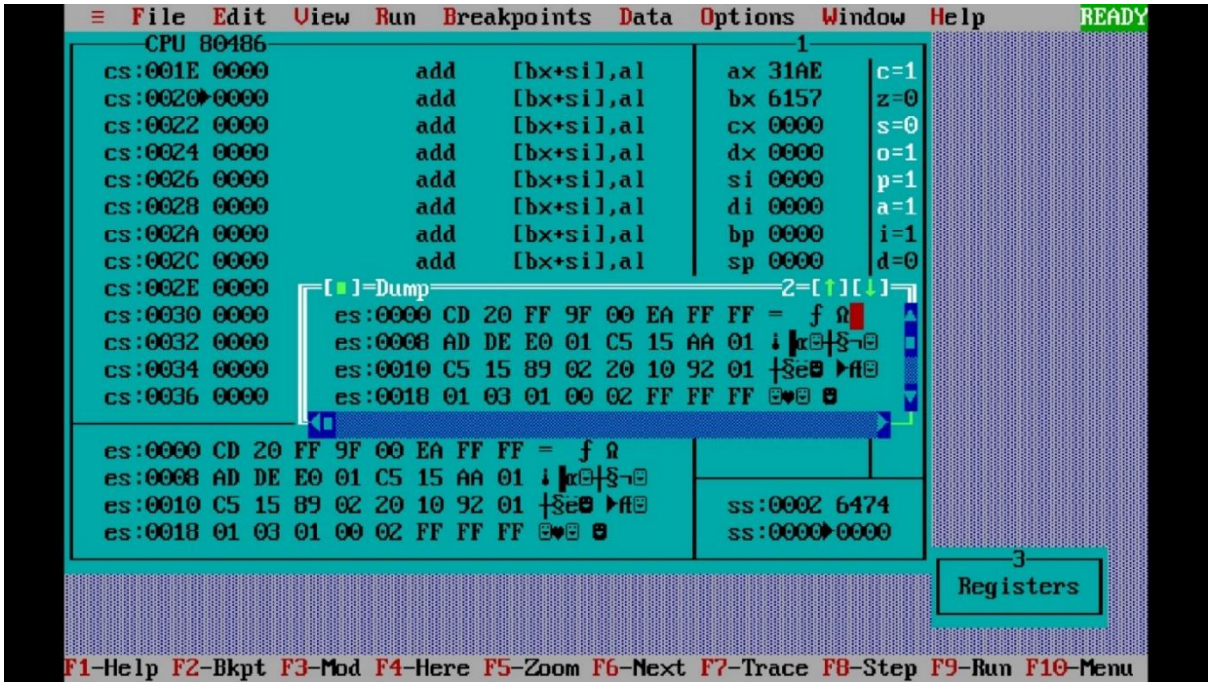
Assembly language code

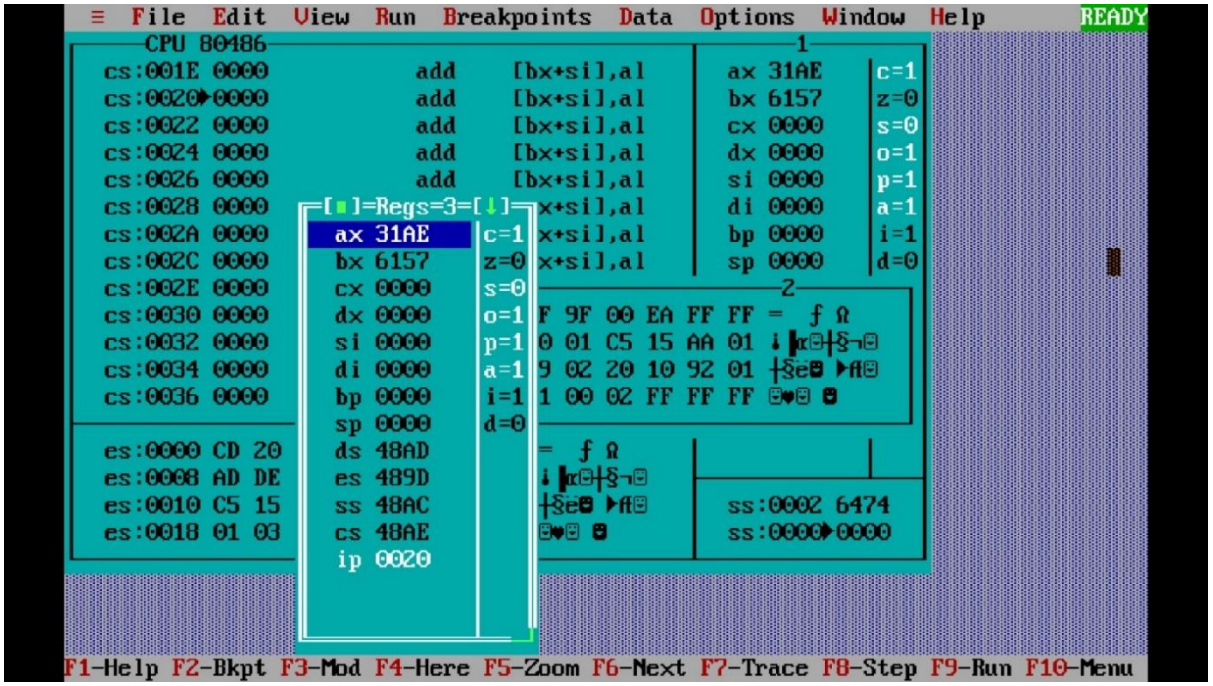
```
data segment            # Data segment

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
cs:0020 0000 add [bx+si],al
cs:0022 0000 add [bx+si],al
cs:0024 0000 add [bx+si],al
cs:0026 0000 add [bx+si],al
cs:0028 0000 add [bx+si],al
cs:002A 0000 add [bx+si],al
cs:002C 0000 add [bx+si],al
cs:002E 0000
cs:0030 0000
cs:0032 0000
cs:0034 0000
cs:0036 0000
[ ]=Dump
2=[↑][↓]
es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
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
ax 31AE c=1
bx 6157 z=0
cx 0000 s=0
dx 0000 o=1
si 0000 p=1
di 0000 a=1
bp 0000 i=1
sp 0000 d=0
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


File Edit View Run Breakpoints Data Options Window Help READY
CPU 80486
[ ]=Regs=3=[↓]
ax 31AE c=1
bx 6157 z=0
cx 0000 s=0
dx 0000 o=1
si 0000 p=1
di 0000 a=1
bp 0000 i=1
sp 0000 d=0
ds 48AD
es 489D
ss 48AC
cs 48AE
ip 0020
2
ss:0002 6474
ss:0000 0000
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

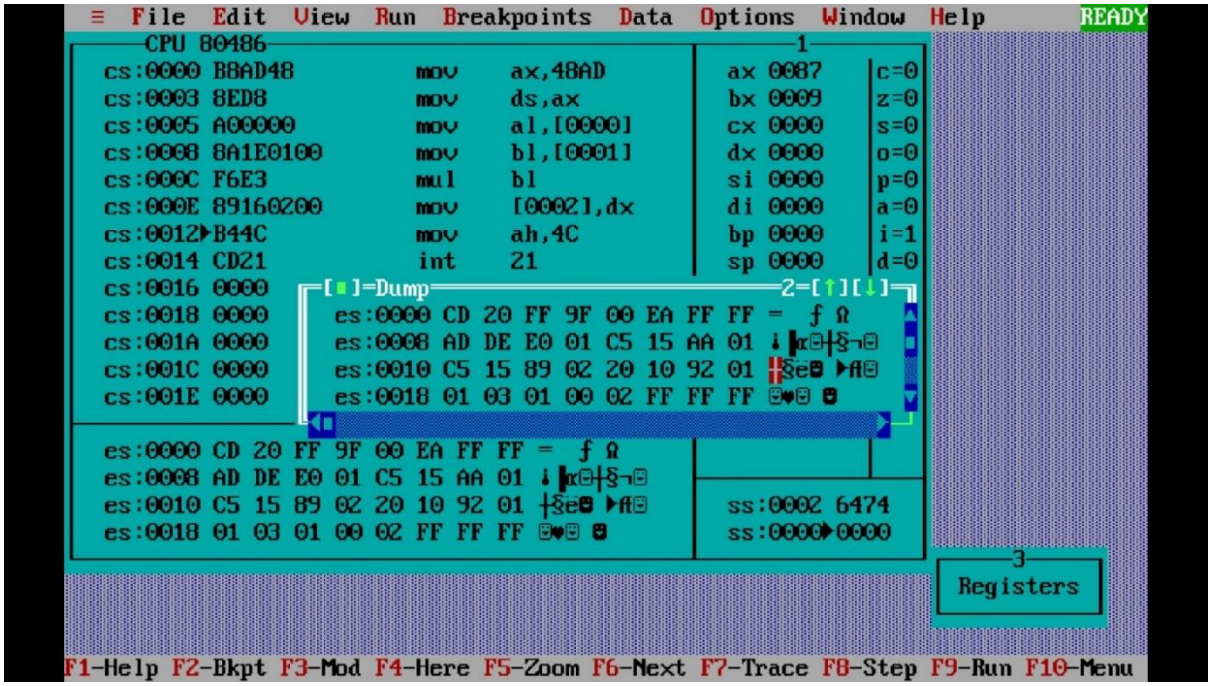
Date of Performance: 05/02/2024

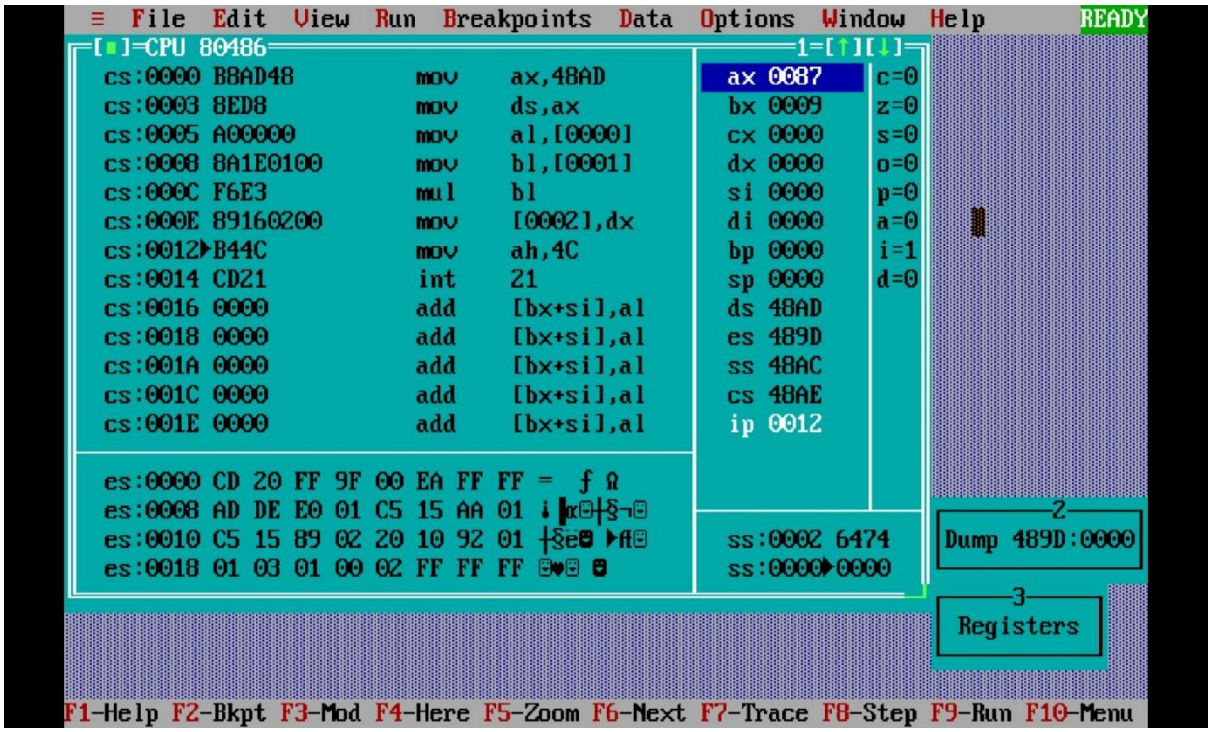
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

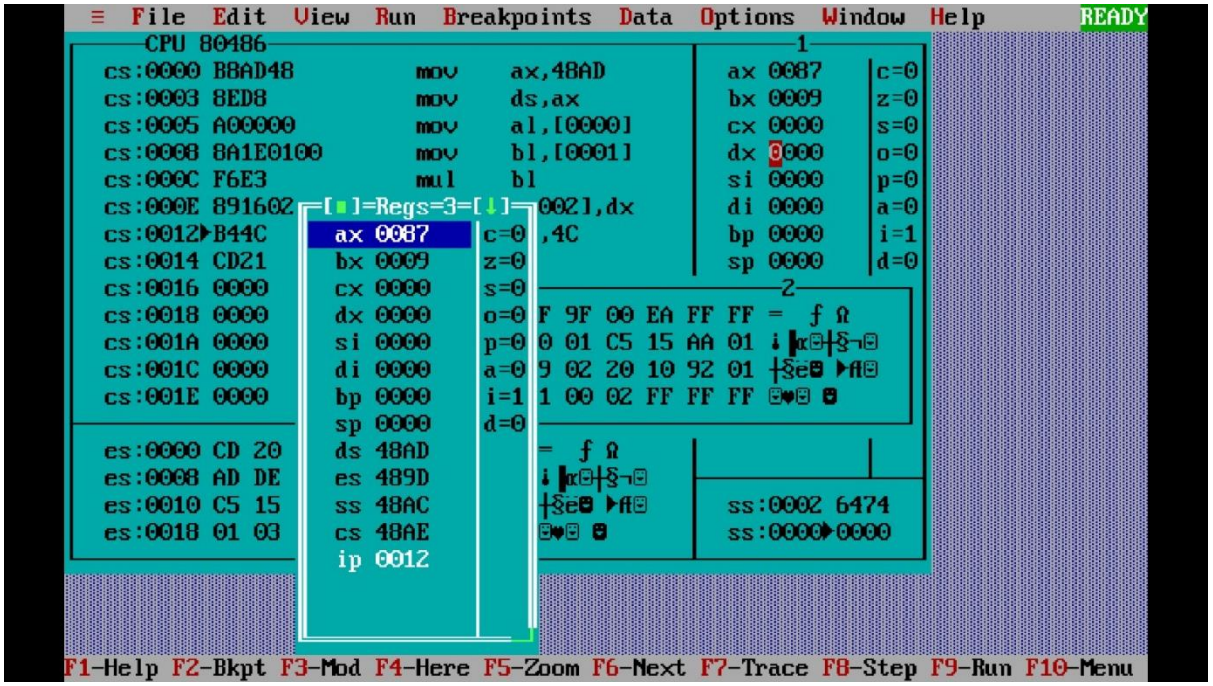
Assembly language code

```
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
int 21h                   # breakpoint interrupt

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
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A00000 mov al,[0000]
cs:0008 8A1E0100 mov bl,[0001]
cs:000C F6E3 mul bl
cs:000E 89160200 mov [0002],dx
cs:0012 B44C mov ah,4C
cs:0014 CD21 int 21
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 = ƒ Ω
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
1
ax 0087 c=0
bx 0009 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
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


File Edit View Run Breakpoints Data Options Window Help READY
[■]=CPU 80486
1=[↑][↓]
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A00000 mov al,[0000]
cs:0008 8A1E0100 mov bl,[0001]
cs:000C F6E3 mul bl
cs:000E 89160200 mov [0002],dx
cs:0012 B44C mov ah,4C
cs:0014 CD21 int 21
cs:0016 0000 add [bx+si],al
cs:0018 0000 add [bx+si],al
cs:001A 0000 add [bx+si],al
cs:001C 0000 add [bx+si],al
cs:001E 0000 add [bx+si],al
es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
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
ax 0087 c=0
bx 0009 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 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(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

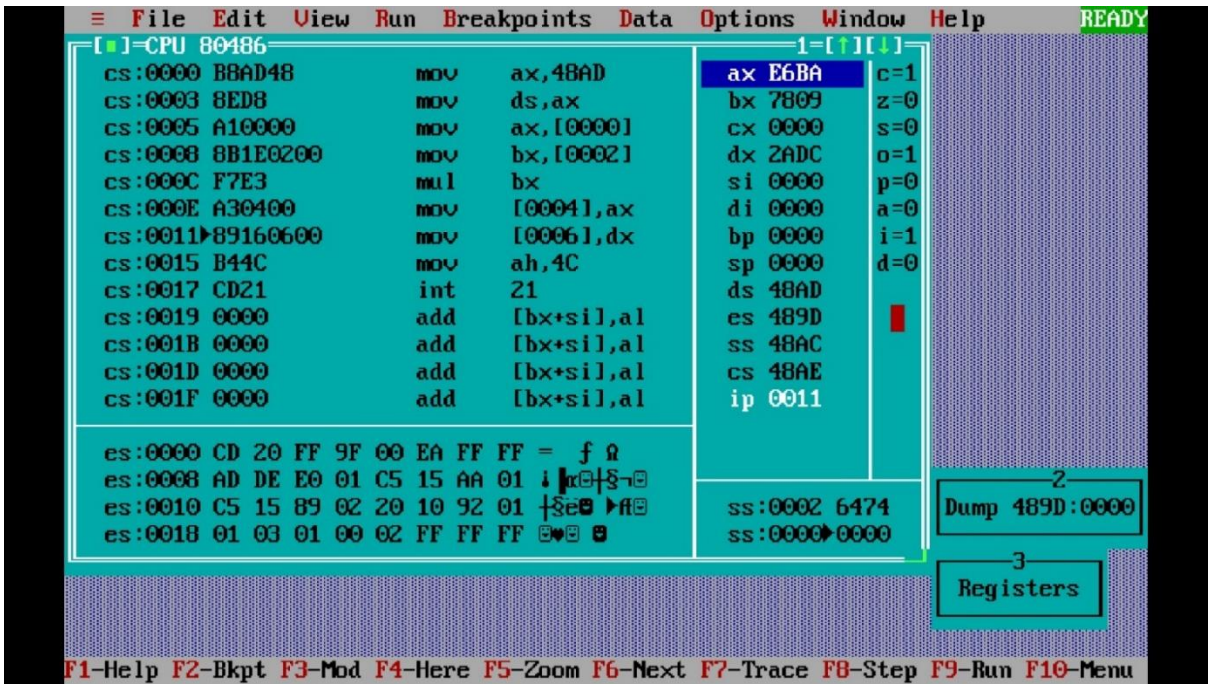
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

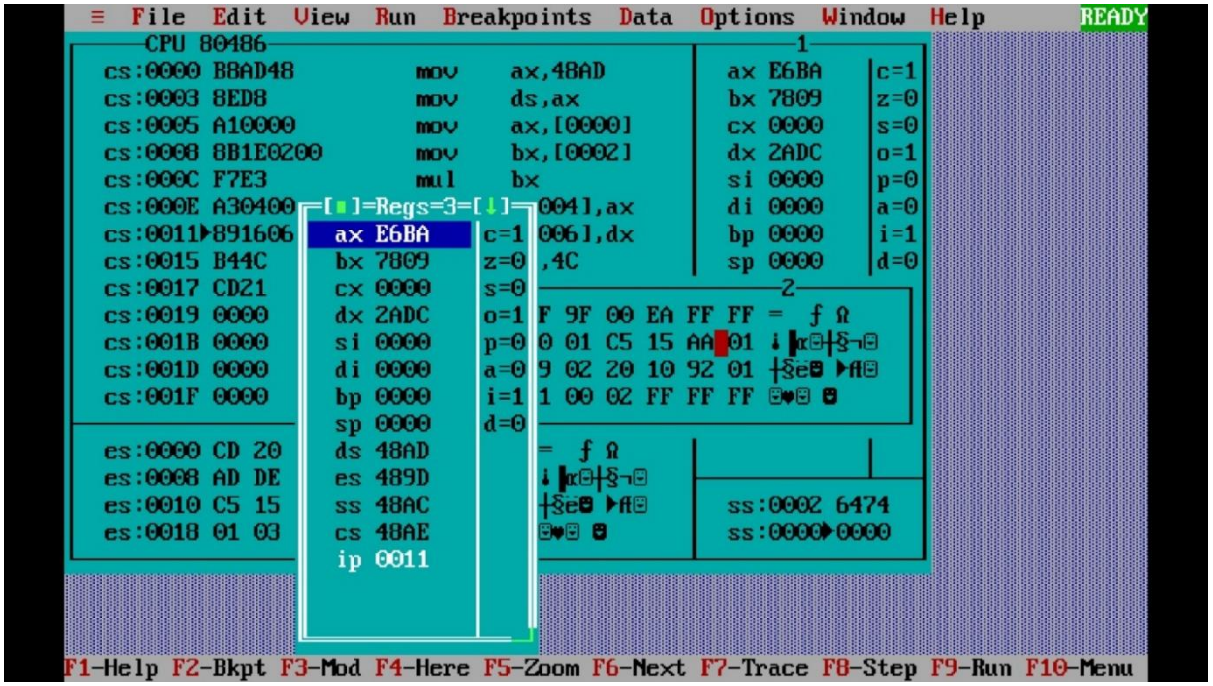
Assembly language code

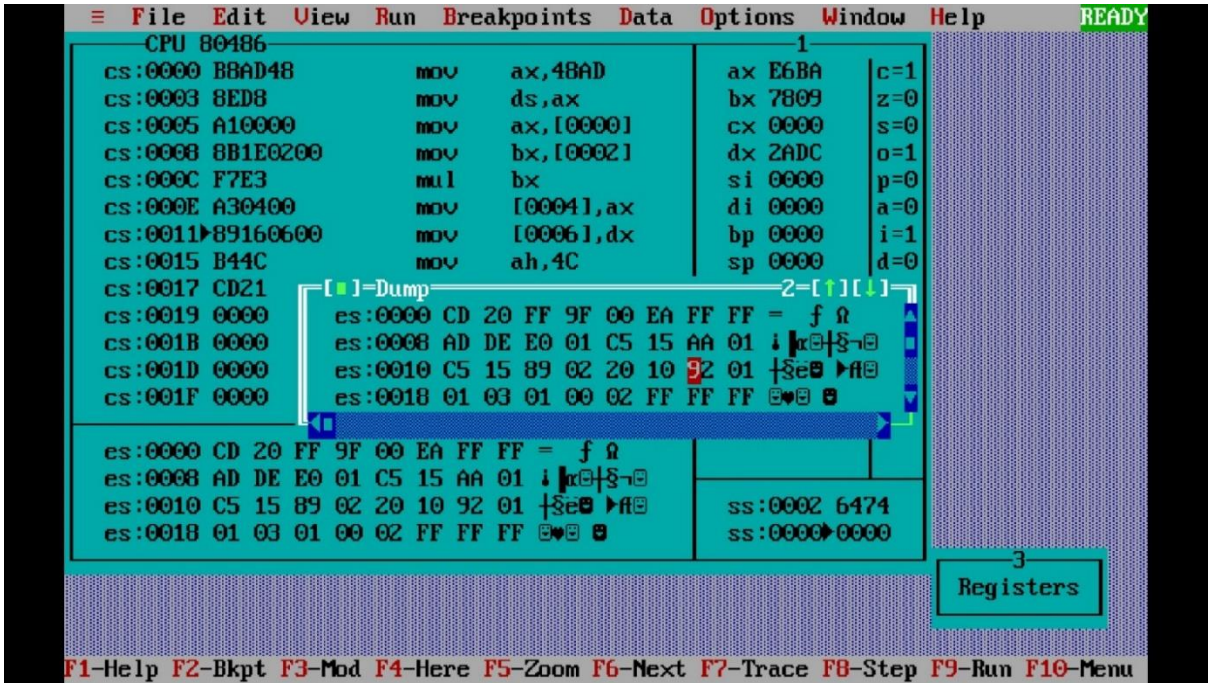
```
data segment            # Data segment

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:




File Edit View Run Breakpoints Data Options Window Help READY
CPU 80486
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A10000 mov ax,[0000]
cs:0008 8B1E0200 mov bx,[0002]
cs:000C F7E3 mul bx
cs:000E A30400 [■]=Regs=3=[↓]004],ax
cs:0011▶891606 ax E6BA c=1 006],dx
cs:0015 B44C bx 7809 z=0 ,4C
cs:0017 CD21 cx 0000 s=0
cs:0019 0000 dx 2ADC o=1
cs:001B 0000 si 0000 p=0
cs:001D 0000 di 0000 a=0
cs:001F 0000 bp 0000 i=1
sp 0000 d=0
es:0000 CD 20 ds 48AD
es:0008 AD DE es 489D
es:0010 C5 15 ss 48AC
es:0018 01 03 cs 48AE
ip 0011
ax E6BA c=1
bx 7809 z=0
cx 0000 s=0
dx 2ADC o=1
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
ss:0002 6474
ss:0000▶0000
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:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A10000 mov ax,[0000]
cs:0008 8B1E0200 mov bx,[0002]
cs:000C F7E3 mul bx
cs:000E A30400 mov [0004],ax
cs:0011▶89160600 mov [0006],dx
cs:0015 B44C mov ah,4C
cs:0017 CD21
cs:0019 0000
cs:001B 0000
cs:001D 0000
cs:001F 0000
[■]=Dump 2=[↑][↓]
es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
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
es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
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
ax E6BA c=1
bx 7809 z=0
cx 0000 s=0
dx 2ADC o=1
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 0000 d=0
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

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

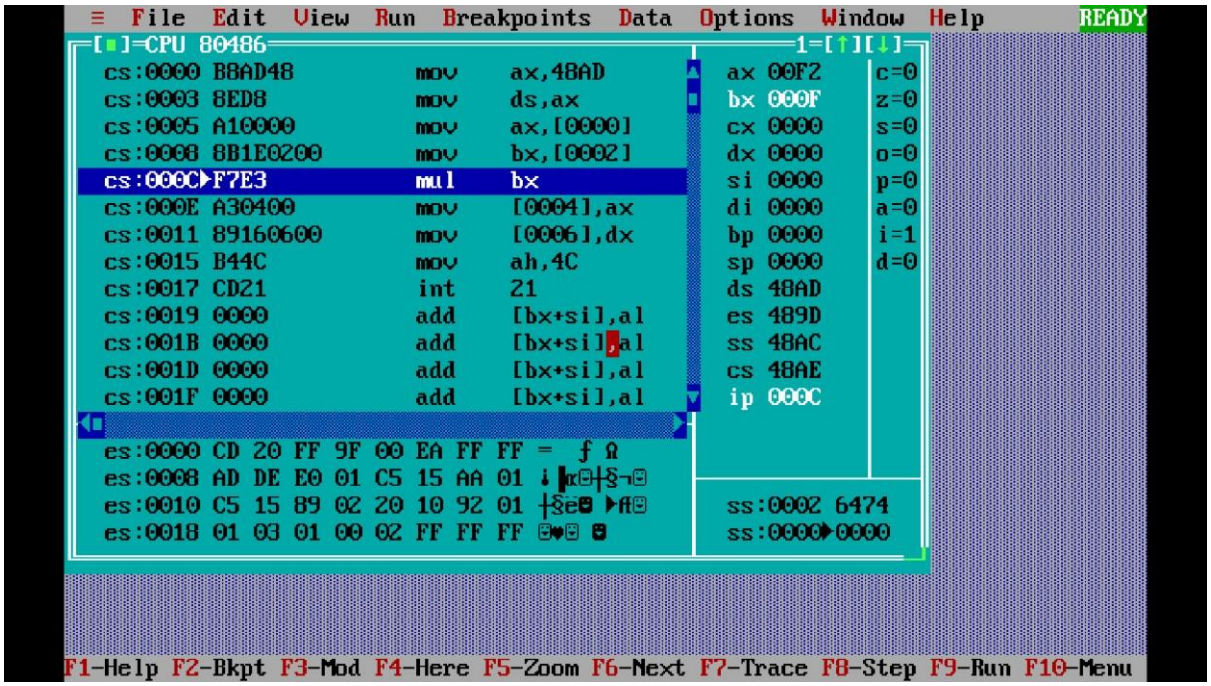
Assembly language code

```
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:



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

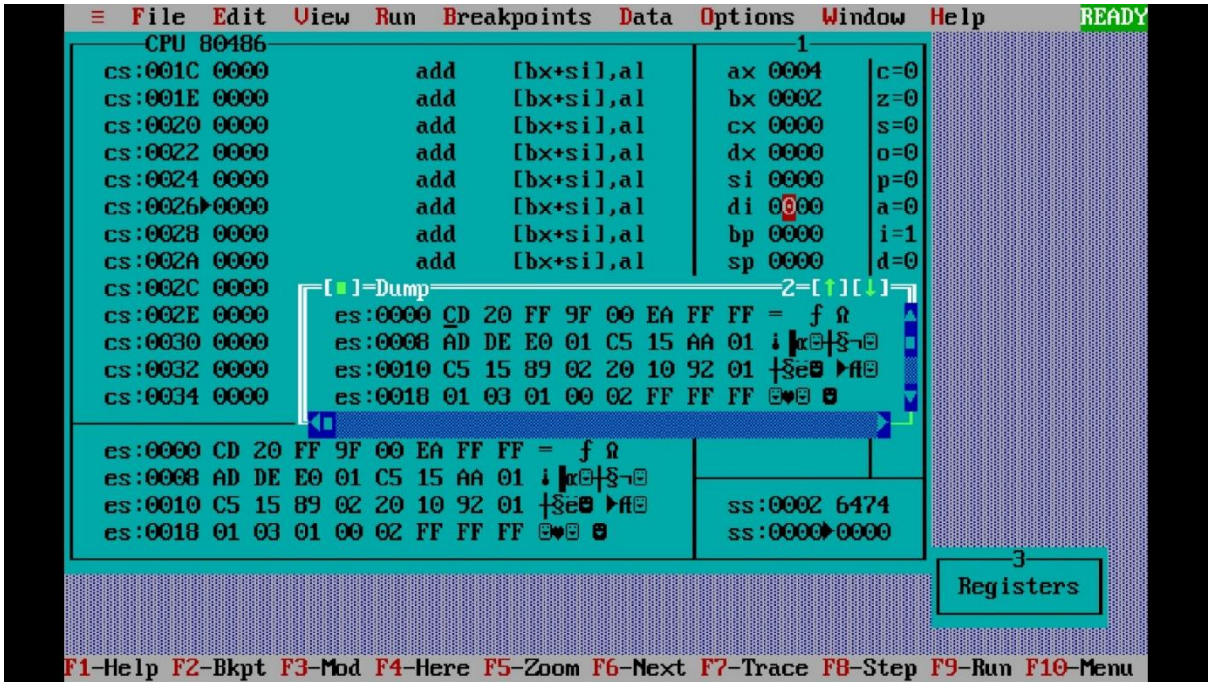
Date of Performance: 05/02/2024

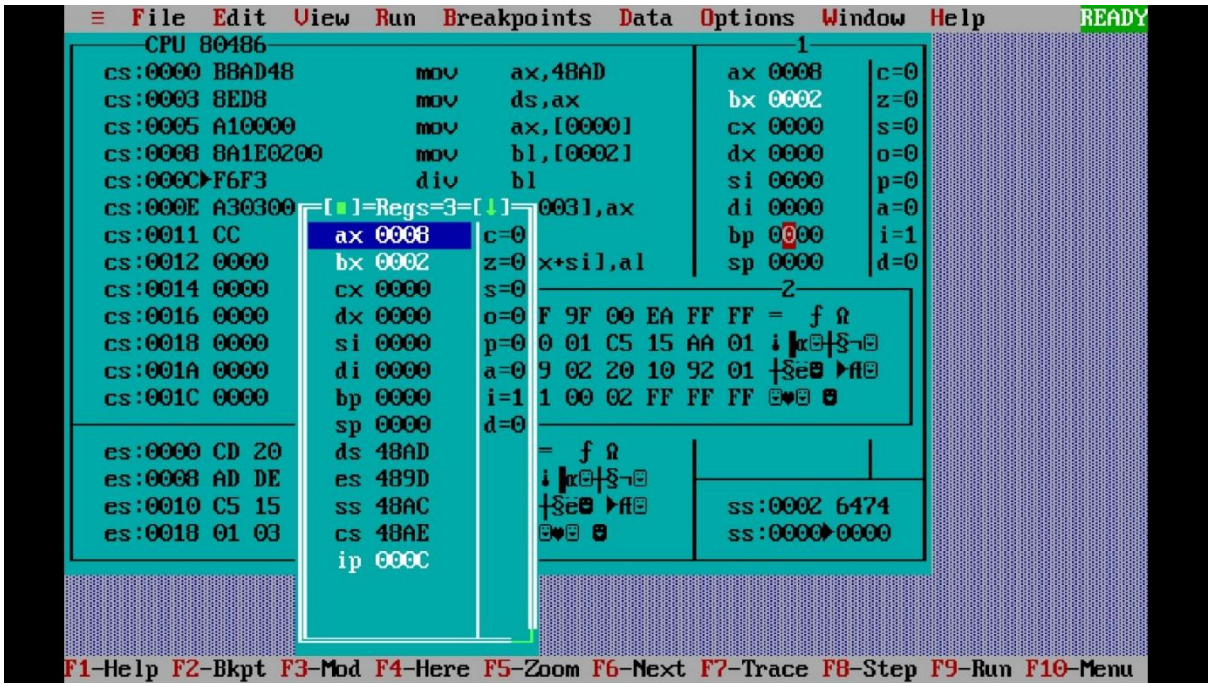
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

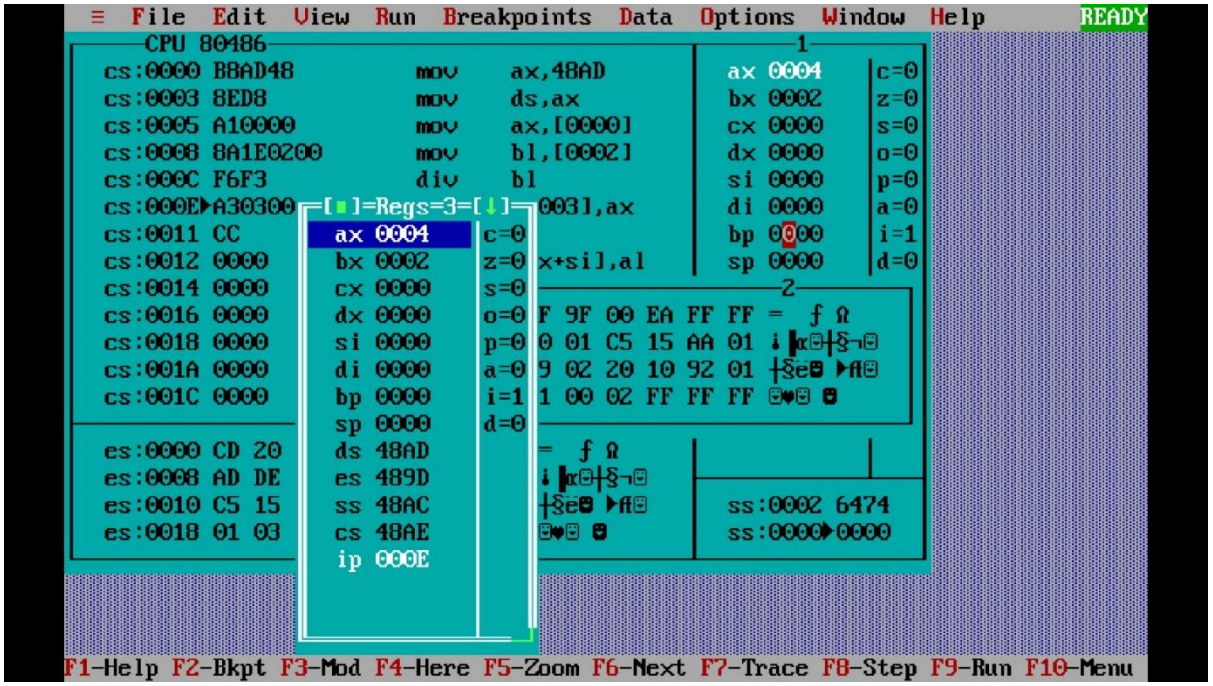
Assembly language code

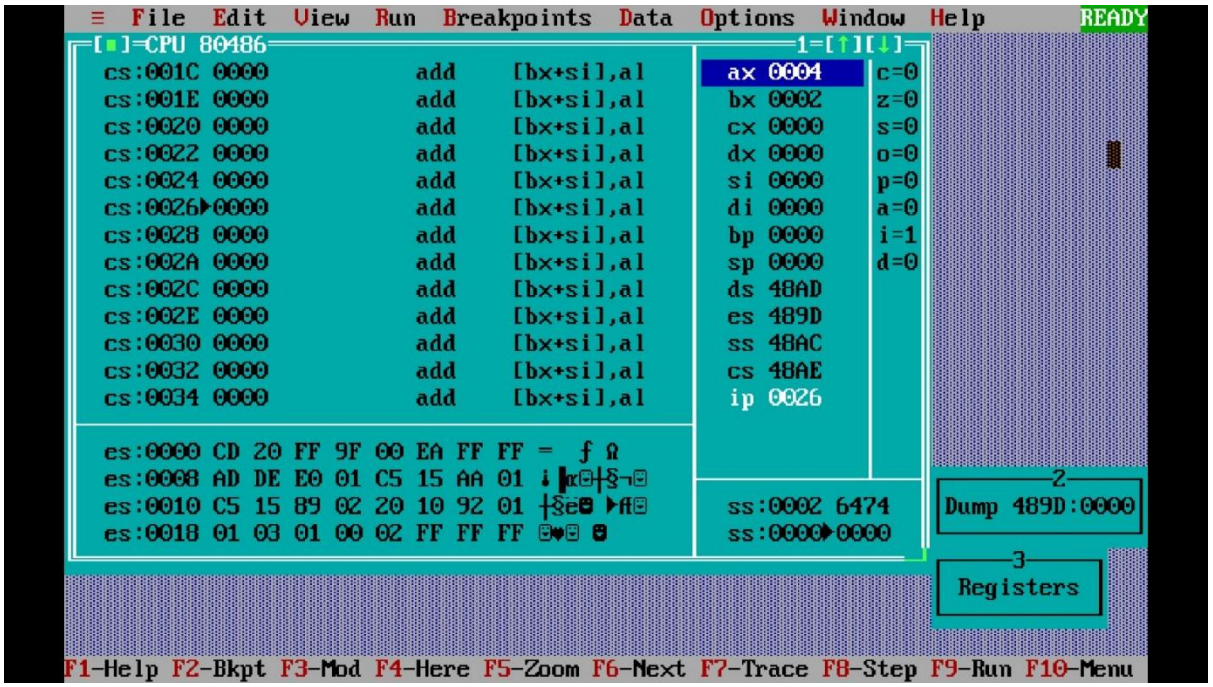
```
data segment            # Data segment

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:


File Edit View Run Breakpoints Data Options Window Help READY
CPU 80486
cs:001C 0000 add [bx+si],al
cs:001E 0000 add [bx+si],al
cs:0020 0000 add [bx+si],al
cs:0022 0000 add [bx+si],al
cs:0024 0000 add [bx+si],al
cs:0026 0000 add [bx+si],al
cs:0028 0000 add [bx+si],al
cs:002A 0000 add [bx+si],al
cs:002C 0000
cs:002E 0000
cs:0030 0000
cs:0032 0000
cs:0034 0000
[■]=Dump
2=[↑][↓]
es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
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
1
ax 0004 c=0
bx 0002 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
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


File Edit View Run Breakpoints Data Options Window Help READY
CPU 80486
cs:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A10000 mov ax,[0000]
cs:0008 8A1E0200 mov bl,[0002]
cs:000C F6F3 div bl
cs:000E A30300 003],ax
cs:0011 CC
cs:0012 0000 x+si],al
cs:0014 0000
cs:0016 0000
cs:0018 0000
cs:001A 0000
cs:001C 0000
[■]=Regs=3=[↓]
ax 0008 c=0
bx 0002 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
1
ax 0008 c=0
bx 0002 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
2
F 9F 00 EA FF FF = ƒ Ω
0 01 C5 15 AA 01
9 02 20 10 92 01
1 00 02 FF FF FF
es:0000 CD 20
es:0008 AD DE
es:0010 C5 15
es:0018 01 03
= ƒ Ω
ss:0002 6474
ss:0000 0000
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:0000 B8AD48 mov ax,48AD
cs:0003 8ED8 mov ds,ax
cs:0005 A10000 mov ax,[0000]
cs:0008 8A1E0200 mov bl,[0002]
cs:000C F6F3 div bl
cs:000E A30300 003],ax
cs:0011 CC
cs:0012 0000 x+si],al
cs:0014 0000
cs:0016 0000
cs:0018 0000
cs:001A 0000
cs:001C 0000
es:0000 CD 20
es:0008 AD DE
es:0010 C5 15
es:0018 01 03
[■]=Regs=3=[↓]
ax 0004 c=0
bx 0002 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 000E
1
ax 0004 c=0
bx 0002 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
2
ss:0002 6474
ss:0000 0000
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:001C 0000 add [bx+si],al
cs:001E 0000 add [bx+si],al
cs:0020 0000 add [bx+si],al
cs:0022 0000 add [bx+si],al
cs:0024 0000 add [bx+si],al
cs:0026 0000 add [bx+si],al
cs:0028 0000 add [bx+si],al
cs:002A 0000 add [bx+si],al
cs:002C 0000 add [bx+si],al
cs:002E 0000 add [bx+si],al
cs:0030 0000 add [bx+si],al
cs:0032 0000 add [bx+si],al
cs:0034 0000 add [bx+si],al
es:0000 CD 20 FF 9F 00 EA FF FF
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
1=[↑][↓]
ax 0004 c=0
bx 0002 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 0026
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

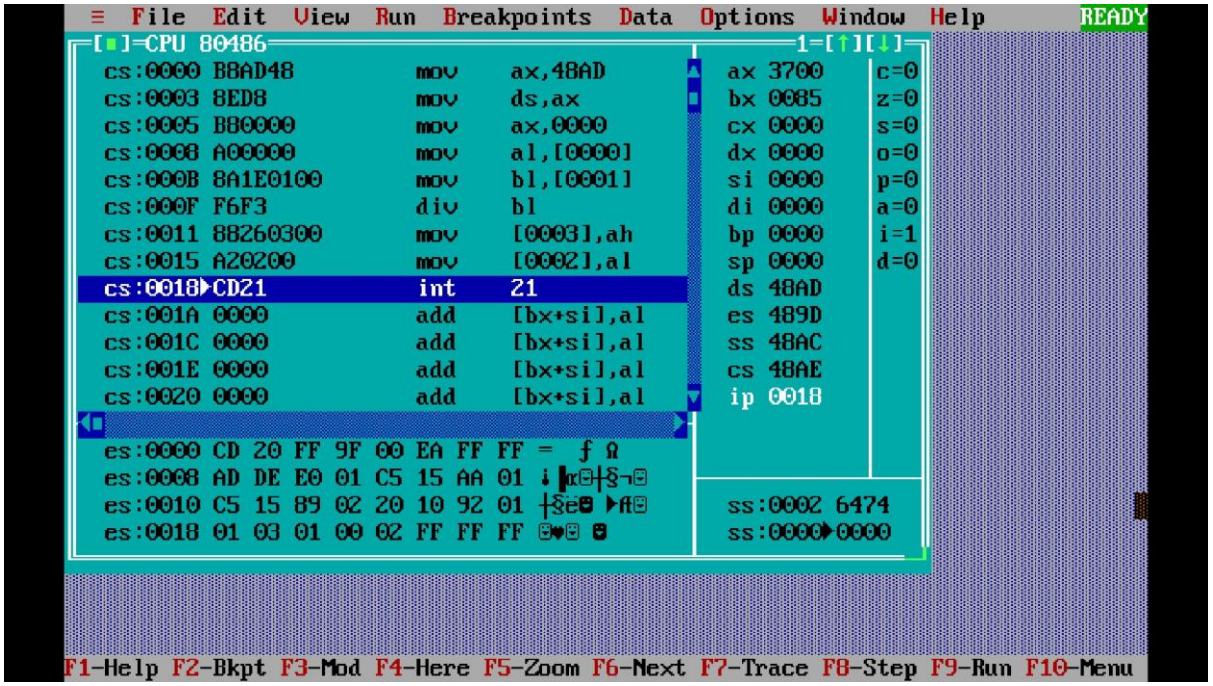
Date of Performance: 05/02/2024

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         # 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
        MOV AL, 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:



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

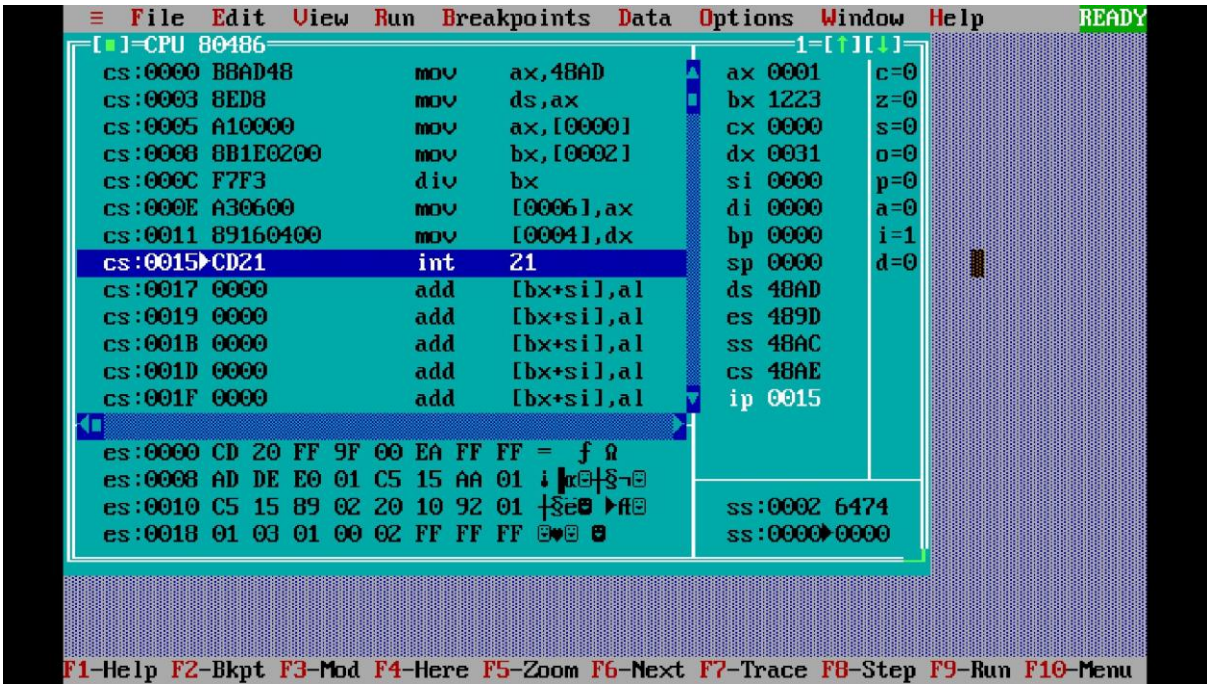
Batch: S2-1 Timing: 3:00-5:00 Date of Submission: 05/02/2024

Assembly language code

```
DATA SEGMENT                    # Data segment
    NUM1 DB 1254H               # define 16 bit numbers
     NUM2 DB 1223H
     QUOTIENT DB ?
     REMAINDER DB ?
DATA ENDS                       # Assemble directives
```

```
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
        MOV AL, 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 :



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

----------********----------**