EXP1: (For 8 bit)

```
.model small
.data
msg1 db 10,13,"Eter 8 bit nos:$"
msg2 db 10,13,"8 bit nos is:$"
.code
.startup
mov ah,09h
lea dx,msg1
int 21h
mov ah,01h
int 21h
sub al,30h
mov cl,04h
shl al,cl
mov bl,al
mov ah,01h
int 21h
sub al,30h
add al,bl
mov bh,al
mov ah,09h
lea dx,msg2
int 21h
mov bl,bh
and bl,0f0h
shr bl,cl
add bl,30h
mov dl,bl
mov ah,02h
int 21h
mov bl,bh
and bl,0fh
add bl,30h
mov dl,bl
mov ah,02h
int 21h
.exit
end
```

EXP1:(For 16 bit)

```
.model small
.data
msg1 dw 10,13,"Enter 16bit nos:$"
msg2 dw 10,13," 16 bit nos is:$"
.code
.startup
mov ah,09h
lea dx,msg1
int 21h
mov ah,01h
int 21h
sub al,30h
mov cl,04h
shl al,cl
mov bh,al
mov ah,01h
int 21h
sub al,30h
add bh,al
mov ah,01h
int 21h
sub al,30h
mov cl,04h
shl al,cl
mov bl,al
mov ah,01h
int 21h
sub al,30h
add bl,al
mov ah,09h
lea dx,msg2
int 21h
mov ch,bh
and ch,0F0h
mov cl,04h
shr ch,cl
add ch,30h
mov dl,ch
mov ah,02h
int 21h
mov ch,bh
and ch,0fh
add ch,30h
```

```
mov dl,ch
mov ah,02h
int 21h
mov dh,bl
and dh ,0f0h
mov cl,04h
shr dh,cl
add dh,30h
mov dl,dh
mov ah,02h
int 21h
mov dh,bl
and dh,0fh
add dh,30h
mov dl,dh
mov ah,02h
int 21h
.exit
end
```

EXP2:

```
.model small
.data
M1 db 10,13,"Addition is:$"
M2 db 10,13,"Subtraction is:$"
NUM1 dw 5347H
NUM2 dw 1342H
RES dw ?
.CODE
DISP MACRO XX
   MOV ah,09
   LEA dx,XX
   INT 21H
ENDM
.STARTUP
DISP M1
MOV ax,NUM1
ADD ax,NUM2
MOV RES,ax
CALL DISP1
DISP M2
MOV ax,NUM1
SUB ax,NUM2
MOV RES,ax
```

```
CALL DISP1
JMP LAST
 DISP1 PROC
    MOV bx,RES
    AND bh,0F0H
    MOV cl,4
    SHR bh,cl
    ADD bh,30H
    MOV dl,bh
    MOV ah,02
    INT 21H
     MOV bx,RES
    AND bh,0FH
    ADD bh,30H
    MOV dl,bh
    MOV ah,02
    INT 21H
     MOV bx,RES
    AND bl,0F0H
    MOV cl,4
    SHR bl,cl
    ADD bl,30H
    MOV dl,bl
    MOV ah,02
    INT 21H
     MOV bx,RES
    AND bl,0FH
    ADD bl,30H
    MOV dl,bl
    MOV ah,02
    INT 21H
     RET
    DISP1 ENDP
 LAST:
.EXIT
END
```

EXP3:

```
.model small
.data
hex dw 0ACH
counter db 0
M1 db 10,13,"BCD:$"
.code
```

```
DISP MACRO XX
   MOV ah,09h
   LEA dx,XX
   INT 21h
ENDM
mov ax,@DATA
mov DS,ax
DISP M1
MOV dx,00h
mov ax,hex
mov bx,000Ah
L:
inc counter
div bx
push dx
cmp ax,0
mov dx,00h
je exit
jmp L
exit:
mov cl,counter
mov ch,00h
L1:
POP dx
add dl,30h
mov ah,02h
int 21h
LOOP L1
mov ah,4ch
int 21h
ret
ends
end
```

EXP4:

```
.model small
.stack
.data
M1 db 10,13,"Enter string1:$"
M2 db 10,13,"Length of string 1:$"
M3 db 10,13,"Display String 1:$"
M4 db 10,13,"Enter string2:$"
```

```
M5 db 10,13,"Length of string 2: $"
M6 db 10,13,"Display String 2:$"
M7 db 10,13,"Compare String: $"
M8 db 10,13,"String not equal$"
M9 db 10,13,"String equal$"
STR1 db 50,?,50 DUP(?)
STR2 db 50,?,50 DUP(?)
L1 db ?
L2 db ?
.code
DISP MACRO XX
mov ah,09
lea dx,xx
int 21h
endm
.startup
DISP M1
mov ah,0Ah
lea dx,STR1
int 21h
DISP M2
lea si,STR1+1
mov cl,[si]
mov l1,cl
add cl,30h
mov dl,cl
mov ah,02
int 21h
DISP M3
lea si,STR1+2
mov cl,l1
back1:
mov dl,[si]
mov ah,02
int 21h
inc si
dec cl
jnz back1
DISP M4
mov ah,0Ah
```

```
lea dx,STR2
int 21h
DISP M5
lea di,STR2+1
mov cl,[di]
mov l2,cl
add cl,30h
mov dl,cl
mov ah,02
int 21h
DISP M6
lea di,STR2+2
mov cl,l2
back2:
mov dl,[di]
mov ah,02
int 21h
inc di
dec cl
jnz back2
disp m7
mov cl,l1
mov ch,l2
cmp cl,ch
jnz n_equal
lea si,str1+2
lea di,str2+2
back3:
mov dl,[si]
cmp dl,[di]
jnz n_equal
inc si
inc di
dec cl
jnz back3
disp m9
jmp exit
n_equal:
disp m8
exit:
```

```
.EXIT
END
```

EXP5:(8 bit large number)

```
.model small
.data
array db 03h,05h,02h,08h,07h
largeno db 0
.code
mov ax,@data
mov ds,ax
mov cl,05
mov si, offset array
mov al,[si]
dec cl
up:
inc si
cmp al,[si]
jnc next
mov al,[si]
next:
loop up
mov largeno,al
add al,30h
mov dl,al
mov ah,02h
int 21h
mov ah,4ch
int 21h
ends
end
```

EXP5:(8 bit small number)

```
.model small
.data
array db 03h,05h,02h,08h,07h
smallno db 0
.code
mov ax,@data
```

```
mov ds,ax
mov cl,05
mov si, offset array
mov al,[si]
dec cl
up:
inc si
cmp al,[si]
jc next
mov al,[si]
next:
loop up
mov smallno,al
add al,30h
mov dl,al
mov ah,02h
int 21h
mov ah,4ch
int 21h
ends
end
```

EXP5:(16 bit large number)

```
.model small
.data
array dw 1003h,1005h,1002h,1008h,1007h
largeno dw 0
.code
mov ax,@data
mov ds,ax
mov cx,0005
mov si, offset array
mov ax,[si]
dec cx
up:
inc si
inc si
cmp ax,[si]
jnc next
```

```
mov ax,[si]
next:
loop up
mov largeno,ax
mov bx,largeno
and bh,0f0h
mov cl,04h
shr bh,cl
add bh,30h
mov dl,bh
mov ah,02h
int 21h
mov bx,largeno
and bh,0fh
add bh,30h
mov dl,bh
mov ah,02h
int 21h
mov bx,largeno
and bl,0f0h
mov cl,04h
shr bl,cl
add bl,30h
mov dl,bl
mov ah,02h
int 21h
mov bx,largeno
and bl,0fh
add bl,30h
mov dl,bl
mov ah,02h
int 21h
mov ah,4ch
int 21h
ends
end
```

EXP5:(16 bit small number)

```
.model small
```

```
.data
array dw 1003h,1005h,1002h,1008h,1007h
smallno dw 0
.code
mov ax,@data
mov ds,ax
mov cx,0005
mov si, offset array
mov ax,[si]
dec cx
up:
inc si
inc si
cmp ax,[si]
jc next
mov ax,[si]
next:
loop up
mov smallno,ax
mov bx,smallno
and bh,0f0h
mov cl,04h
shr bh,cl
add bh,30h
mov dl,bh
mov ah,02h
int 21h
mov bx,smallno
and bh,0fh
add bh,30h
mov dl,bh
mov ah,02h
int 21h
mov bx,smallno
and bl,0f0h
mov cl,04h
shr bl,cl
add bl,30h
mov dl,bl
mov ah,02h
```

```
int 21h
mov bx,smallno
and bl,0fh
add bl,30h
mov dl,bl
mov ah,02h
int 21h
mov ah,4ch
int 21h
ends
end
```

EXP7:

```
#include<stdio.h>
void main()
{
int a=15,b=3,c,d,e,f;
asm{
mov ax,a
mov bx,b
add ax,bx
mov c,ax
}
asm{
mov ax,a
mov bx,b
sub ax,bx
mov d,ax
mov ax,a
mov bx,b
mul bx
mov e,ax
mov ax,a
mov bx,b
div bx
mov f,ax
}
clrscr();
printf("Addition is: %d",c);
```

```
printf("\nSubtraction is: %d",d);
printf("\nMultiplication is: %d",e);
printf("\nDivision is: %d",f);
getch();
}
```

EXP8:

```
.model small
.stack
.data
msg1 db 10,13,"Mouse driver present:$";
.code
disp macro xx
mov ah,09
lea dx,xx
int 21h
endm
.startup
mov ax,0000
int 33h
cmp ax,00h
je last
disp msg1
mov ax,0004
mov cx,0
mov dx,0
int 33h
mov ax, 0007
mov cx,0010
mov dx,055h
int 33h
mov ax, 0008
mov cx,0010
mov dx,055h
int 33h
pixel:
mov ax,0001
int 33h
mov ax,0003
int 33h
cmp bx,01
je left
jmp right
left:
```

```
mov bx,0011h
int 10h
mov ah,0ch
int 10h
right:
mov ax,0001
int 33h
cmp bx,02
je last
jmp pixel
last:
mov ax,00
int 10h
.exit
end
```