Addition / Subtraction :- (8 bit)

```
.model small
.data
x db 25h
y db 26h
result db 00h
.code
Mov ax,@data
mov ds,ax
mov al,x
mov bl,y
add al,bl
mov result,al
mov ah,4ch
int 21h
ends
end
```

Addition / Subtraction :- (8 bit BCD)

```
.model small
.data
x db 25h
y db 26h
result db 00h
.code
Mov ax,@data
mov ds,ax
mov al,x
mov bl,y
Sub  al,bl
das
mov result,al
mov ah,4ch
int 21h
ends
end
```

Addition / Subtraction :- (32 bit)

```
.model small
.data
x dd 00000025h
y dd 00000026h
result dd 00000000h
.code
mov ax,@data
mov ds,ax
mov ax,word ptr x
mov bx,word ptr y
add ax,bx
mov word ptr result,ax
mov ax,word ptr x+1
mov bx,word ptr y+1
add ax,bx
mov word ptr result+1,ax
mov ah,4ch
int 21h
ends
end
```

Addition / Subtraction :- (16 bit)

```
.model small
.data
No1 dw 0001h
no2  dw 0010h
Res dw ?
.code
mov ax,@data
mov ds,ax
mov ax,no1
add ax,no2
mov res, ax
int 03h
ends
end
```

Addition / Subtraction :- (if res > 16 bit)

```
.model small
.data
X dw 0005H
Y dw 0092H
Res1 ?
Res2 ?
.code
Mov ax, @data
Mov ds, ax
Mov ax, x
Add ax, y
Jnc next
Inc res2
Next : mov res1, ax
Int 03H
Ends
End
```

Addition of series of 8 bit:-

```
.model small
.data
Array db 2H,4H,6H,7H,5H
Rlsb db 00H
Rmsb db 00H
.code
mov ax,@data
Mov ds,ax
mov cx,0005H
LEA si, array
up: mov al,[si]
Add rlsb,al
jnc next
add rmsb,01H
next: inc si
loop up
int 03H
code ends
end
```

Addition of series of 8 bit:- (BCD)

```
.model small
.data
Array db 2H,4H,6H,7H,5H
Rlsb db 00H
Rmsb db 00H
.code
mov ax,@data
Mov ds,ax
mov cx,0005H
LEA si, array
up: mov al,[si]
Add rlsb,al
daa
jnc next
add rmsb,01H
next: inc si
loop up
int 03H
code ends
end
```

Addition of series of 16 bit:-

```
.model small
.data
Array dw 2H,4H,6H,7H,5H
Res dw 00H
.code
mov ax,@data
Mov ds,ax
mov cx,0005H
mov ax,0000H
LEA si, array
Up : add ax, [si]
Inc si
Inc si
Loop up
Mov res, ax
Int 03H
Ends
end
```

Multiplication Unsigned: - (8 bit)

```
.model small
.data
X db 05H
Y db 02H
Res dw ?
.code
Mov ax, @data
Mov ds, ax
Mov ax, 0000H
Mov al,x
Mov bl, y
Mul bl
Mov res, ax
Int 03H
Ends
End
```

Multiplication Signed: - (8 bit)

```
.model small
.data
X db -05H
Y db -02H
Res dw ?
.code
Mov ax, @data
Mov ds, ax
Mov ax, 0000H
Mov al,x
Mov bl, y
imul bl
Mov res, ax
Int 03H
Ends
End
```

Multiplication Unsigned: - (16 bit)

```
.model small
.data
X dw 05H
Y dw 02H
Res1 dw 00H
Res2 dw 00H
.code
Mov ax, @data
Mov ds, ax
Mov ax, 0000H
Mov ax,x
Mov bx, y
Mul bl
Mov res1, ax
mov res2, dx
Int 03H
Ends
End
```

Multiplication Signed: - (16 bit)

```
.model small
.data
X dw -05H
Y dw -02H
Res1 dw 00H
Res2 dw 00H
.code
Mov ax, @data
Mov ds, ax
Mov ax, 0000H
Mov ax,x
Mov bx, y
imul bx
Mov res1, ax
mov res2, dx
Int 03H
Ends
End
```

Multiplication (Successive addi.) :-(8 bit)

```
.model small
.data
x db 5h
y db 4h
res db 00h
Res_m db 00h        ;result more than 8-bit
.code
Mov ax,@data
mov ds, ax
Mov ax, 0000h
Mov cx ,0000 h
mov cl, y
up : add al, x
daa
Jnc next
Inc res_m
next : Loop up
mov res, al
ends
end
```

Division Unsigned: - (8 bit)

```
.model small
.data
x db 0008h
y db 0002h
q db 0000h
r db 0000h
.code
mov ax, @data
mov ds, ax
mov ax, 0000H
mov al, x
mov bl, y
div bl
mov q, al
mov r, ah
int 03H
ends
end
```

Division Unsigned: - (16 bit)

```
.model small
.data
x dw 0008h
y dw 0002h
q dw 0000h
r dw 0000h
.code
mov ax, @data
mov ds, ax
mov dx, 0000H
mov ax, 0000H
mov ax, x
mov bx, y
div bx
mov q, ax
mov r, dx
int 03H
ends
end
```

Division Unsigned: - (32/16 bit)

```
.model small
.data
x dd 22222222h
y dw 0002h
q dw 0000h
r dw 0000h
.code
mov ax, @data
mov ds, ax
mov ax, word ptr x
mov dx, word ptr x+1
mov bx, y
div bx
Mov q, ax
mov r, dx
ends
end
```

Division (Successive subt.) :-(8 bit)

```
.model small
.data
x db 5h
y db 4h
rem db 00h
q db 00h
Res_m db ooh        ;result more than 8-bit
.code
Mov ax, @data
mov ds, ax
Mov al, x
Next : sub al, y
Das
Inc q
Cmp al, y
Jnc next
Inc res_m
mov rem, al
ends
end
```

Length of String

```
.model small
.data
S1 db ‘COMPUTER$’
Len db 00H
.code
Mov ax, @data
Mov ds, ax
LEA si, s1
Next : mov al, [si]
Cmp al, ‘$’
Je exit
Inc si
Inc len
Jmp next
Exit : int 03H
Ends
End
```

Concatenate 2 String

```
.model small
.data
s1 db 'COMPUTER DEPARTMENT$'
s2 db ‘DEPARTMENT$’
msg db 'Concatenated String : $'
.code
Mov ax, @data
Mov ds, ax
LEA si, s1
Next: mov al, [si]
Cmp al, ‘$’
Je exit
Inc si
Jmp next

Exit:
LEA di, s2
Next1: mov al, [di]
Cmp al, ‘$’
Je exit1
Mov [si],al
Inc si
Inc di
Jmp next1

Exit1:
Mov al, ‘$’
Mov [si], al

Mov ah, 09H
LEA dx, msg
Int 21H

Mov ah, 09H
LEA dx, s1
Int 21H

Ends
End
```

Compare 2 Strings

```
.model small
.data
S1 db ‘COMPUTER$’
S2 db ‘COMPUTER$’
S1len db 00H
S2len db 00H
msg1 db 'Strings are equal$'
msg2 db 'Strings are not equal$'
.code
Mov ax, @data
Mov ds, ax
Mov es, ax

LEA si, s1
Next: mov al, [si]
Cmp al, ‘$’
Je exit
Inc si
Inc s1len
Jmp next

Exit:
LEA si, s2
Next1: mov all,[si]
Cmp al, ‘$’
Je exit1
Inc si
Inc s2len
Jmp next1

Exit1:
Mov al, s1len
Cmp al, s2len
Jne exit2

Cld
Mov cx, 0000H
Mov cl, s1len
LEA si, s1
LEA di, s2
Up : cmpsb
Jnz exit2
Loop up
Inc si

Mov ah,09H
LEA dx, msg1
Int 21H
Jmp exit3

Exit2:
Mov ah,09H
LEA dx, msg2
Int 21H

Exit3:
Int 03H
Ends
End
```

Reverse of a String

```
.model small
.data
s1 db 'COMPUTER DEPARTMENT$'
s2 db 50 DUP('$')
msg1 db 'The source string is: $'
msg2 db 'The reverse string is: $'
s1len db 00H
.code
Mov ax, @data
Mov ds, ax
LEA si, s1
Next: mov al, [si]
Cmp al,'$'
Je exit
Inc si
Inc s1len
Jmp next
Exit:
LEA di, s2
Up: dec si
Mov al, [si]
Mov [di], al
Inc di
Dec s1len
Jnz up
Mov al, '$'
Mov [di], al

Mov ah, 09H
LEA dx, msg1
Int 21H

Mov ah, 09H
LEA dx, s1
int 21H

Mov ah, 09H
LEA dx, msg2
Int 21H

Mov ah, 09H
LEA dx, s2
int 21H
ends
end
```

Smaller No. in Array (8 bit)

```
.model small
.data
array db 2H,4H,6H,7H,5H
smaller db 00H
.code
mov ax, @data
Mov ds, ax
mov cx,0005H
LEA si, array
mov al,[si]
Dec cx
Up : Inc si
Cmp   AL, [si]
jc next
Mov al, [si]
next: loop up
Mov smaller, al
int 03H
ends
end
```

Larger No. in Array (8 bit)

```
.model small
.data
array db 2H,4H,6H,7H,5H
larger db 00H
.code
mov ax, @data
Mov ds, ax
mov cx,0005H
LEA si, array
mov al,[si]
Dec cx
Up : Inc si
Cmp   AL, [si]
jnc  next
Mov al, [si]
next: loop up
Mov larger, al
int 03H
ends
end
```

Odd Nos in Array

```
.model small
.data
array dw 134H, 65H, 876H, 976H, 23H
odd_no db 00H
.code
Mov ax, @data
Mov ds, ax
Mov cx, 0005H
LEA si, array
Next: mov ax, [si]
Ror ax,1
Jnc dn
Inc odd_no
Dn: add si,2H
Loop next
Int 03H
Ends
End
```

Even Nos in Array

```
.model small
.data
array dw 134H, 65H, 876H, 976H, 23H
even_no db 00H
.code
Mov ax, @data
Mov ds, ax
Mov cx, 0005H
```

```
LEA si, array
Next: mov ax, [si]
Ror ax,1
Jc dn
Inc even_no
Dn: add si,2H
Loop next
Int 03H
Ends
End
```

Find NO is EVEN / ODD

```
.model small
.data
x db 87H
od db 00H
ev db 00H
.code
Mov ax, @data
Mov ds, ax
Mov al, x
ROR al, 1
Jnc dn
Rol al,1
Mov od, al
Jmp exit
Dn : rol al,1
Mov ev, al
Exit:
Int 03H
Ends
End
```

Add all ODD no. in Array

```
.model small
.data
array db 6, 5, 21, 3, 8, 9, 11, 13, 1, 2
arr db 10 DUP(0)
count dw 00H
sum db 00H
.code
Mov ax, @data
Mov ds, ax
LEA si, array
LEA di, arr

Up: mov al, [si]
Ror al, 1
Jnc dn
Rol al, 1
Mov [di], al
Inc count
Inc di
Dn: inc si
Loop up

Mov cx, count
LEA si, arr
Up1: mov al, [si]
Add sum, al
Inc si
Loop up1
Int 03H
Ends
End
```

Count Odd & Even Nos in Array

```
.model small
.data
array dw 134H, 65H, 876H, 976H, 23H
od db 00H
ev db 00H
.code
Mov ax, @data
Mov ds, ax
Mov cx, 0005H
LEA si, array
Up: mov al, [si]
Ror al,1
Jnc dn
Inc od
Jmp down
Dn: inc ev
Down : in si
Loop up
Int 03H
Ends
End
```

Positive or Negative

```
.model small
.data
X db -9H
Pos db 00H
Neg db 00H
.code
Mov ax, @data
Mov ds, ax
Mov al, x
Rol al, 1
Jnc dn
Ror al,1
Mov neg, al
Jmp exit
Dn: Ror al,1
Mov pos, al
Exit:
Int 03H
Ends
End
```

Count Positive or Negative in Array

```
.model small
.data
array dw -0009H, 0010H, -0008H, 0001H
pos dw 00H
neg dw 00H
.code
Mov ax, @data
Mov ds, ax
```

```
Mov cx, 0004H
LEA si, array
Next: mov ax, [si]
Rol ax,1
Jnc dn
Inc neg
Jmp down
Dn: inc pos
Down: add si, 2H
Loop next
Mov ax, pos
Mov bx, neg
Int 03H
Ends
end
```

Add All Positive Nos

```
.model small
.data
array dw -0009H, 0010H, -0008H, 0001H
sum dw 00H
.data
Mov ax, @data
Mov ds, ax
Mov cx, 0004H
LEA si, array
Next: mov ax, [si]
Rol ax,1
Jc dn
Ror ax,1
Add sum, al
Dn: add si, 2H
Loop up
Int 03H
Ends
end
```

Add All Negative Nos

```
.model small
.data
array dw -0009H, 0010H, -0008H, 0001H
sum dw 00H
.data
Mov ax, @data
Mov ds, ax
Mov cx, 0004H
LEA si, array
Next: mov ax, [si]
Rol ax,1
Jnc dn
Ror ax,1
Add sum, al
Dn: add si, 2H
Loop up
Int 03H
Ends
end
```

Count 1's in given number

```
.model small
.data
x db 0FFH
ones db 00H
.code
Mov ax, @data
Mov ds, ax
Mov al, x
Up: Ror al,1
Jnc dn
Inc ones
Dn: Loop up
Mov al, ones
Int 03H
Ends
end
```

Count 0's in given number

```
.model small
.data
x db 0FFH
zeros db 00H
.code
Mov ax, @data
Mov ds, ax
Mov al, x
Up: Ror al,1
Jc dn
Inc zeros
Dn: Loop up
Mov al, ones
Int 03H
Ends
end
```

Ascending Order

```
.model small
.data
Array dw 99H,12H,56H,45H,36H
Count dw 0004H
.code
Mov ax, @data
Mov ds, ax
Mov dx, count

NEXT: LEA si, array
Mov cx, dx
UP: mov ax, [si]
Cmp ax, [si+2]
Jc down
Xchg ax, [si+2]
Mov [si], ax
Down: add si, 2H
Loop UP
Dec dx
Jnz NEXT
Int 03H
Ends
End
```

Descending Order

```
.model small
.data
Array dw 99H,12H,56H,45H,36H
Count dw 0004H
.code
Mov ax, @data
Mov ds, ax
Mov dx, count

NEXT: LEA si, array
Mov cx, dx
UP: mov ax, [si]
Cmp ax, [si+2]
Jnc down
Xchg ax, [si+2]
Mov [si], ax
Down: add si, 2H
Loop UP
Dec dx
Jnz NEXT
Int 03H
Ends
End
```

Lowercase to Uppercase

```
.model small
.data
s1 db 'computer$'
s2 db 20 dup(‘$’)
.code
Mov ax, @data
Mov ds, ax
LEA si, s1
LEA di, s2
Up: mov al, [si]
Cmp al, ‘$’
Je EXIT
Sub al, 20H
Mov [di], al
Inc si
Inc di
Jmp up
Int 03H
Ends
end
```

Uppercase to Lowercase

```
.model small
.data
s1 db 'COMPUTER$'
s2 db 20 dup(‘$’)
.code
Mov ax, @data
Mov ds, ax
LEA si, s1
LEA di, s2
Up: mov al, [si]
Cmp al, ‘$’
Je EXIT
Add al, 20H
Mov [di], al
Inc si
Inc di
Jmp up
Int 03H
Ends
end
```