Signed Numbers & Unsigned Numbers

① Why it is important to Understand?

→ Needed in Multiplication & Division.

↳ Different for Signed & Unsigned Numbers

Signed -ve Negative Number (That have sign)

Unsigned Number & positive Number (Not any sign)

② How signed no. Come in Assembly language?

(A) mov bl, 2
Sub bl, 4

-2	Bl

Bl - 4
2 - 4 = -2

(B) Directly assigned Negative Numbers

mov bl, 3
add bl, -2

1	Bl

3 + (-2)

Bl + (-2)
3 + (-2) = 1.

Conversion of Unsigned to Signed Numbers

Decimal 1 to Binary 0001b

How we get binary of Decimal -1 ? highest no. of System

0001 → One's Complement
- 1111
1110 → Two's Complement
+ 1
1111 → Result is called Signed Number

1 ⟶ 0001b
-1 ⟶ 1111b

For Hexadecimal

Decimal 1 ⟶ Hexadecimal 0001h

0-9 A B C D E F

0001h
- FFFF
FFFE
+ F
FFFFh

1 ⟶ 0001h
-1 ⟶ FFFFh

Identification of Signed / Unsigned Numbers

Any number given how identify it is signed or unsigned.

0001 ⟶ LSB (least Significant Bit)

MSB (Most Significant Bit).

If MSB = 1, it is Negative
If MSB = 0, it is positive

Why Div/Mul is Different for Signed & Unsigned :-

$-7 * -7 = 49$ Sign Matters.

$-7 * 7 = -49$

$7 * 7 = 49$

$-7/-7 = 1$

$-7/7 = -1$

$7/7 = 1$.

For example Binary 1111b for Multiplication. given

Division In Assembly Importance :-

1) Convert Decimal to Binary (All CS Depends on this)
2) Print Decimal Of Many Bits.
   - e.g. Mov bl, 12
   - or Add bl, 30
3) Given Number is Odd or Even.
4) Division Program.

Division Program In Assembly

a) 21/5

Dividend 16 (AX) ← 21

Divisor 8 (BL) ← 5

b) Need two storage / two registers to store Dividend & Divisor.
Default Dividend is Ax.

c) Divisor Can be any Register (BL, CL, DL).
Used 8 bit Register.
not write BX, CX, DX.

For example. To Divide.

```
Mov Ax, 21  ⟶ Dividend
Mov bl, 5   ⟶ Divisor.
```

Divide Command / Instruction **Div divisor**

Div bl

Actual Decimal Process look like this,

```
      4  ⟶ Quotient
   5 ) 21 ⟶ Dividend
       20
      ----
       1  ⟶ Remainder
```

Divisor ↙

We have Quotient & Remainder,
How print values of Quotient & Remainder ?

$$\frac{Ax}{BL} = \boxed{Al} \quad \boxed{Ah}$$

Al → Quotient (4)
Ah → Remainder (1)

To print value of Quotient & Remainder, we use the register, move it, used cl register.

```
mov cl, al
mov ch, ah
mov dl, cl
mov ah, 2
int 21h
mov dl, ch
mov ah, 2
int 21h
```

Print

mov al to cl → Remainder
mov ah to ch → Quotient

first cl then ch.

(a) If Divisor is 8 bits, e.g. bl, cl, dl

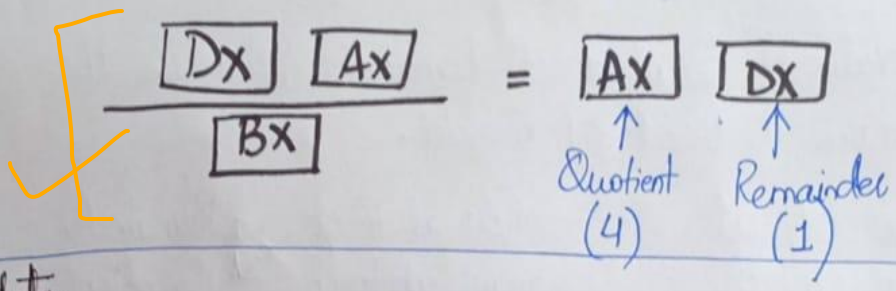
(b) If Divisor is 16 bit e.g. bx, cx, dx

Divisor placed in 16 bit register BX, CX, DX.

what is possibility if working on 16 bit Register ?

DX : AX → By default used it

assigned Zero ←




```
.data
Quo dw ?
Rem dw ?
.code
main proc
mov ax, @data
mov ds, ax
Mov Ax, 21
Mov Dx, 0
Mov Bx, 5
Div Bx
Mov Quo, AX
Mov Rem, DX
```

```
Mov DX, Quo
Mov ah, 2
int 21h
```

```
Mov Dx, Rem
Mov ah, 2
int 21h
```

If Divisor is 16 bits, we use BX, CX, DX it divide by combine two registers, DX : AX

If any value place in DX, empty it first, if not empty answer will be wrong that's why we zero the value of DX

Divide program & Print Quotient & Remainder

```
.data
q db ?
r db ?
.code
main proc
mov ax, @data
mov ds, ax

mov ax, 27
mov bl, 5
div bl

mov q, al
mov r, ah

mov dl, q
add dl, 48
mov ah, 2
int 21h

mov dl, r
add dl, 48
mov ah, 2
int 21h
mov ah, 4ch
int 21h
main endp
end main
```

→ Variable define for quotient & Remainder

mov ax, 27 → Dividend go to ax, 27

mov bl, 5 → Divisor go to bl, 5

div bl → Divide command (divisor given).

mov q, al → move quotient (al) to q

mov r, ah → make remainder (ah) to r

→ Now print Both

moves q to dl

result not in ASCII code b/c we direct move value here. Add 48 in dl the number convert into ASCII code.

Get an integer from user and display whether the number is even or odd

```
model small
.stack 100h
.data
ev db 'Even$'
od db 'Odd$'
.code
main proc
mov ax,@data
mov ds,ax
mov ah,1
int 21h

mov bl,2
div bl
cmp ah,0
je IsEven
mov dx,10
mov ah,2
int 21h

mov dx,13
mov ah,2
int 21h
mov dx,offset od
mov ah,9
int 21h
mov ah,4ch
int 21h

IsEven:
mov dx,10
mov ah,2
int 21h
mov dx,13
mov ah,2
int 21h
mov dx,offset ev
mov ah,9
int 21h
mov ah,4ch
int 21h
main endp
end main
```