Division Program In Assembly

a) 21/5
- 21 → Dividend
- 5 → Divisor

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

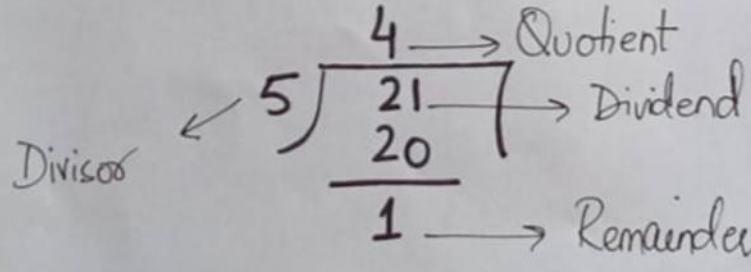
c) Divisor Can be any Register (BL, CL, DL).
Used 8 bit Register.
not write BX, CX, DX.

For example. To Divide.
- Mov Ax, 21 → Dividend
- Mov bl, 5 → Divisor.

Divide Command / Instruction **Div divisor**

Div bl

Actual Decimal Procen look like this,



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

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

↓ Quotient (4) ↓ 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

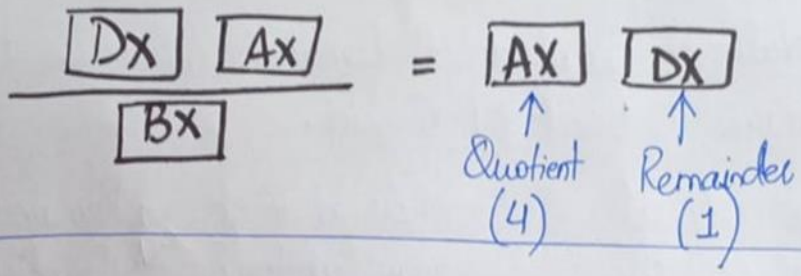
first cl then ch.

(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
assigned Zero ← → By default used it




```
.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 ?          → Variable define for
r db ?            quotient & Remainder
.code
main proc
mov ax, @data
mov ds, ax

mov ax, 27      → Dividend go to ax, 27
mov bl, 5       → Divisor go to bl, 5
div bl          → Divide comand (divisor given).

mov q, al       → move quotient (al) to q
mov r, ah       → make remainder (ah) to r
                → Now print Both
mov dl, q         moves q to dl
add dl, 48      result not in ASCII code b/c we
mov ah, 2       direct move value here. Add 48 in
int 21h         dl the number convert into ASCII code.

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

Multiplication In Assembly.

5 x 9

↙ multiplicand ↘ Multiplier.

1) In assembly if Multiplier is of 8 bits, multiplicand is also of 8 bits,

2) Both should be same.

Multiplicand – Multiplier
AL – BL, CL, DL
AX – BX, CX, DX.

3). Multiplication in Assembly.

Keyword ← mul multiplier → Value of multiplier. (BL, CL, DL, BX, CX, DX).

4). Example of Multiplication for 8 bits.

multiply ↓
Mov al, 5 ⟶ 5 moves to al
Mov bl, 9 ⟶ 9 " bl.
mul bl

5). In case of 8 bits, put 5 to al & 9 to bl..

[al] 5 x [bl] 9 result go to [AX]

put Ax to Dx & print the result.

6) In case of 16 bits, result not handled by single register, we used two registers DX & AX,

$$\frac{AX}{5} \times \frac{BX}{9} = \frac{DX\ AX}{45}$$

for result, two registers are used,
Converted into Binary as, 00000000 00010101

7). Result of 8 bits, result goes to ax then move to dx.

```
mov dx, ax  } ⟶ print a number on screen
add dx, 48  } ⟶ Ascii code convert.
mov ah, 2   } ⟶ Print single digit.
int 21h     }
```

8) If result not single digit, $5 \times 9 = 45$.
function 2 print one digit, (not two digits), technique used

AX = AH : AL ⟶ put 4 in AH & 5 in AL.

9). How this performed in Assembly,
used directive called AAM (Ascii Adjust after Multiplication)

Like,

```
mov al, 5
mov bl, 9
mul bl
```

AAM ⟶ It perform the functionality break the AX in two Parts (AH & AL) and adjust.

10). After writing AAM, 4 goes to AH & 5 to AL. (3)

11). We have to store it in another register to save the value.

```
        mov ch, ah
        mov cl, al
mov dl, ch
add dl, 48
mov ah, 2
int 21h
mov dl, cl
add dl, 48
mov ah, 2
int 21h
```

Code to Multiply two number & print the product

(4)

```
.code
main proc
mov al, 5
mov bl, 9
mul bl
AAM
mov ch, ah
mov cl, al
mov dl, ch
add dl, 48
mov ah, 2
int 21h
mov dl, cl
add dl, 48
mov ah, 2
int 21h
mov ah, 4ch
int 21h
main endp
end main
```