7) Arithmetic Instructions

1) ADD BL, CL

BL ← BL + CL

2) ADC BL, CL (Add with carry)

BL ← BL + CL + CF

⇒ This is the carry before instruction took place

$$\begin{array}{rrl} & \overset{CF}{12} & \overset{\boxed{1}\,-AC}{FF}\ h \\ + & 00 & 01\ h \\ \hline & 13 & 00\ h \end{array}$$

So, 12 + 00 + $\underline{1}$
carry of the previ
operation.

So, with ADC

```
MOV BX, 12FFh
MOV CX, 0001h
ADD BL, CL
ADC BH, CH
```

→ It could have been directly, but we are doing it for understanding. Because for adding 32-bits, we use this metho
Add with carry.

CS CamScanner

→ we add in the lower part (ADD)

→ we add with carry in the higher part. (ADC)

3) SUB BL, CL

BL ← BL, CL

4) SBB BL, CL

BL ← BL, CL − CF

So, here we subtract the borrow of the previous operation.

here we are using decimal.

$\boxed{1} \longrightarrow$ CF (Borrow)

$$\begin{array}{r} 4\\ -1\\ -\boxed{1}\,\text{CF}\\ \hline 2 \end{array} \leftarrow \begin{array}{r} 4\ (1)7\\ -1\ \ \ 9\\ \hline 2\ \ \ 8 \end{array}$$

5) INC BL,

BL ← BL+1

6) DEC BL

BL ← BL −1

DEC BL.

00
⋮
FF
00

$\boxed{\begin{array}{r} -01 \\ +00 \end{array}}$ → -1 is written in 2's complement form

-01 ⇒ 0000 0001

$\underbrace{1111}_{F}$ $\underbrace{1111}_{F}$

$\boxed{\text{MUL operand}}$ (here, we have only 1 operand; only 1 is allowed)

So, since you need another operand (which is not allowed to declare) are called accumulator.

→ In 8086, there are 3 accumulators. (fixed operand)

8 bit – AL (only) (Not AH)
16 bit – AX
32 bit – $\underset{\overset{16}{(H)}}{DX}$: $\underset{\overset{16}{(L)}}{AX}$

(8 × 8)

5] MUL BL

→ here, μp understands you want to do 8-bit multiplications.
So, it will do in backened.

MUL AL × BL

$AX \leftarrow AL \times BL$

where you will get your result.

→ here we are ~~addid~~ not adding; we are multiplying; so, MUL 8 bits; answer will be much bigger.

→ the answer cannot be more than 16-bits.

6(a) why? An 8-bit register can hold up to 2^8 values. So, 8 bit register will give you, $= 2^8 \times 2^8 = 2^{8+8} = 2^{16}$ vales (16-bits results.)

→ so, in worth case $FF_{16} \times FF_{16} = FE01_{16}$

→ do not check carry after multiplication.

→ CF will carry garbage value.

So, if we write, MUL BH.

$AX \Leftarrow \underline{AL} \times BH$ (not AH)

fixed

⇒ MUL BX (16x16) (unsigned num)

here, up understands it is a 16-bit number. i.e. the programmer wants to multiply two 16-bit numbers.

$\text{MUL BX} = \underbrace{DX}_{\text{(H)}} \cdot \underbrace{AX}_{\text{(L)}} \leftarrow AX \times BX$

~~⇒ AX × BX~~

II. IMUL = Integer Multiplication

this is used when we want to multiply $\boxed{\text{signed}}$ numbers.

→ signed/unsigned will be mentioned in question. "Assume, the numbers are signed---"

II, DIV divisor

$\hookrightarrow \boxed{16 \div 8}$ **

eg. DIV BL [Accumulator will carry dividend]

Suppose, 75 ÷ 5

```
MOV   BL, 05h
MOV   AL, 75h
DIV   BL.
```

→ here, we are dividing bigger number with a smaller number. **

e.g. DIV BL

$\boxed{32 \div 16}$ **

dividing smaller number with a bigger number is useful when ~~[illegible]~~ fractional answers (floating point numbers) here, 8086 does not deal with floating point numbers.

$7 \div 2 = \underset{\text{Quot.}}{\boxed{3}} \quad \underset{\text{Rem.}}{\boxed{1}}$

division is the exact reverse of multiplication.

~~A~~

$AX \leftarrow AL \times BL$

So, $16 \div 8 = 8$ bit ans. $\boxed{AL}$ Ans.

$\underset{\text{Rem.}}{\underset{8bit}{AH}} \cdot \underset{\text{Quo}}{\underset{8bit}{AL}} \leftarrow AX \div BL$

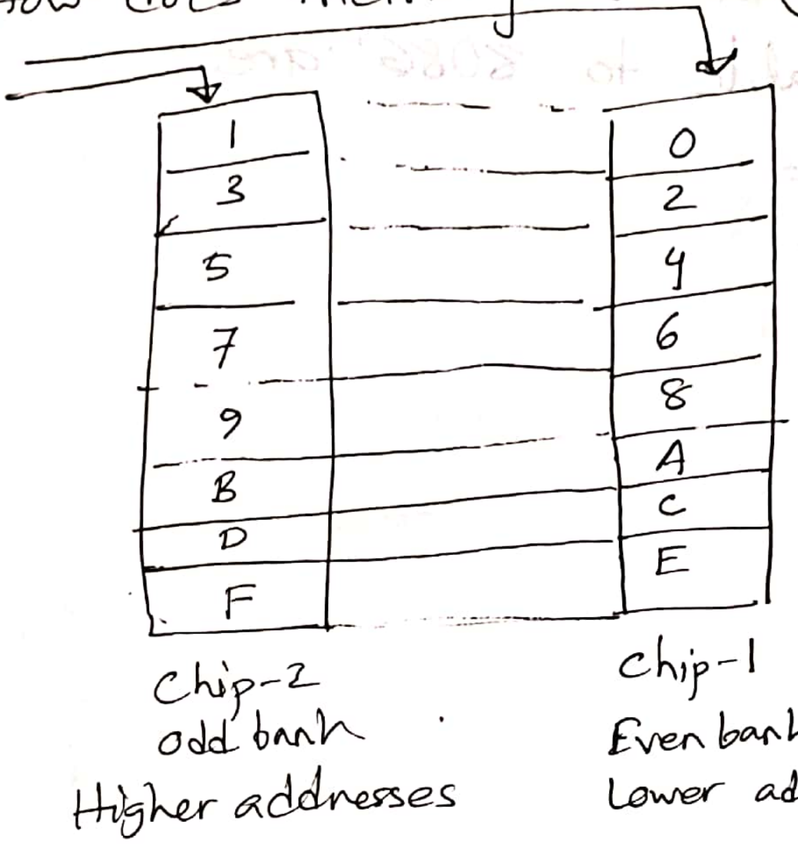
$32 \div 16$

DIV BX

$\underset{R}{\underset{16\text{-bit}}{DX}} \quad \underset{Q}{\underset{16\ bit}{AX}} \leftarrow DX \cdot AX \div BX$

IDIV → signed division.

4. (a) How does memory banking work?



MP can address one address at a time, so, µp will go to both the ~~the~~ chips. starting from even bank & then odd bank (always) ; this process is called alligned data access.

→ Whether µp need 16/8 bit ; it will go to both the banks, but depending upon the situation, it ~~will~~ can transfer 8 or 16 bit at a time.

CamScanner

☐ How does it facilitate ?

→ in terms of performance, it makes µp easier to generate one address at a time, but can access two address so. as a result ~~[illegible]~~ via this process µp can transfer 16 bit or 8 bit at a time. So, by imposing this process, µp has the luxury of getting full bit or half bit or both at the same time depending on the condition.

4(b) Pointer & index registers usually holds the offset addresses, which is added with the base registers for generating 20-bit physical addresses.

→ After ~~[illegible]~~ travelling to 1MB memory, the pointer & index registers ~~has~~ ~~of~~ moves in various directions. IP goes downward, SP & BP goes upwards, SI & DI can move in both directions.

CS CamScanner