CF and OF express overflow situations ONLY for ADDITION and SUBTRACTION !!!!
(The 2 flags OF and CF are set exactly for letting you come up with a correction if you want…)

- If set to 1, both of them are expressing INCORRECT MATHEMATICAL OPERATIONS !!! (That is why in fact they are set to 1 !)

- What about MULTIPLICATION and DIVISION ?

  Op1 – m positions, op2 – n positions , the result og op1 * op2 will be represented on m+n positions. Fortunately, the assembly language IS ALWAYS providing multiplications with enough space for the result.

  Multiplication in assembly language WILL NEVER issue a REAL overflow !!!!

  Anyway, CF and OF will have a role in case of multiplication

  B*b = word; b*b=b (2*3=6) → OF=CF=0 (no "overflow")

  B*b = w (255 * 3 =…) → OF=CF=1 ("overflow")

DIVISION ???

- WE DO HAVE overflow in case of division and IT IS THE WORST CASE OF ALL !!!

  w/b = byte ; 1002/3 = 334, but… 334 is a byte ??? No, it isn't and IT IS FATAL !!!
→ RUN TIME ERROR …. Divide overflow, Zero Divide, Division by zero

Number/0 = Zero divide – this is a forbidden operation ! Why this is a forbidden operation ???

Mathematical analysis tells us that:

Number/(epsilon that approaches zero) = a result that aproaches infinit !!! – this is a correct mathematical operations

Number/0 is assembly language will issue a DIVIDE OVERFLOW (ZERO DIVIDE, DIVISION by ZERO) because infinit doesn't fit a byte, a word, a doubleword

If I intend to make 1002/3 =… the assembly language will issue a DIVIDE OVERFLOW because 334 doesn't fit a byte. But from above the processor already considered that 3 messages equivalent… so, it still will issue the same things …. Why ? because from the mP point of view it the same…

From a technical point of view an INT 0 will be issued in ALL situations of this type !!

- **Do I really need 2 flags for overflow ?**
- What means IMUL and IDIV ? …
- Where are IADD and ISUB ?? … They do not exist because

IADD = ADD, ISUB = SUB … because they work exactly the same way in base 2. Addition and subtraction are VALID - UNDER BOTH INTERPRETATIONS SIMULTANEOUSLY !!!!

When a base 2 addition or subtraction is performed, in fact 2 different simultaneously operations in base 10 are performed !! One for the unsigned interpretation (and that is why we need CF) and the other one for the signed interpretation (and that is why we need OF).

What values will be associated with CF and OF in case of DIVISION ??
In case of division CF and OF are UNDEFINED !!!

1 addition in base 2 = 2 different SIMULTANEOUS additions in base 10 !!

1 subtraction in base 2 = 2 different SIMULTANEOUS subtractions in base 10 !!

That is why CF and OF are NECESSARY to be TOGETHER present and associated with this operations

- Why DO I NEED IMUL AND IDIV ?

Because in contrast from addition and subtraction (which operate identically in base 2 independently of interpretation – signed or unsigned) multiplication and division function DIFFERENTLY ! Here there is mandatory to specify the interpretation of the operands (signed or unsigned) BEFORE the actual operation is performed… and the microprocessor distinguish that by using DIV vs IDIV and MUL vs. IMUL

In case of addition and subtraction there is no need to make such a distinction before performing it, so the decision regarding a possible interpretation (signed or unsigned) is taken only AFTER the actual operation is performed. That is why we do NOT need IADD or ISUB…