```
::: TP2_DUPONT_Corentin :::
::::::::::::::::::::::::::::::::

TP2_DUPONT_Corentin
|-- [1.1K]  calculette.asm
|-- [ 687]  intervalle.asm
`-- [ 669]  pgcd.asm

0 directories, 3 files
::::::::::::::::::::::::::::::::
=> TP2_DUPONT_Corentin/intervalle.asm :
=========================
        ;DUPONT Corentin
        ;07/04/2020

0 :     In      ;0
1 :     Store Mem [1]   ;1
2 :     In      ;2
3 :     Store Mem [2]   ;3
4 :     Load Mem [2]    ;4
5 :     SUB Mem [1]     ;5
6 :     JC 20   ;6
7 :     In      ;7
8 :     Store Mem [0]   ;8
9 :     CMP Mem [2]     ;9
10 :    JC 17   ;10
11 :    Load Mem [1]    ;11
12 :    CMP Mem [0]     ;12
13 :    JC 17   ;13
14 :    Load 1  ;14
15 :    Out     ;14
16 :    JMP 19  ;15
17 :    Load 0  ;16
18 :    Out     ;17
19 :    JMP 28  ;18
20 :    Load Mem [1]    ;19
21 :    Store Mem [3]   ;20
22 :    Load Mem [2]    ;21
23 :    Store Mem [1]   ;22
24 :    Load Mem [3]    ;23
25 :    Store Mem [2]   ;24
26 :    JMP 7   ;25
27 :    Load 0  ;26
28 :    End     ;27


=> TP2_DUPONT_Corentin/calculette.asm :
=========================
        ;DUPONT Corentin
        ;07/04/2020

0 :     In      ;0
1 :     Store Mem[0]    ;1 -> a
2 :     In      ;2 -> b
3 :     Store Mem[1]    ;3
4 :     In      ;4 -> c
5 :     Store Mem[2]    ;5


        ;Je regarde quel opÃ©ration Ã  faire
6 :     CMP 0   ;6
7 :     JZ 45   ;7
```

peu importe le résultat de la soustraction
=> CMP à la place de SUB (mais du coup les variables sont à prendre dans l'autre sens)

2 fois 14


```
8 :     CMP 1   ;8
9 :     JZ 20   ;9
10 :    CMP 2   ;10
11 :    JZ 23   ;11
12 :    CMP 3   ;12
13 :    JZ 26   ;13
14 :    CMP 4   ;14
15 :    JZ 29   ;15
16 :    CMP 5   ;16
17 :    JZ 35   ;17

        ;Si la valeur de "c" est au dessus de 5
18 :    CMP 0   ;18
        ;Retour Ã  la demande de l'opÃ©ration (c)
19 :    JC 4    ;19

        ;a+b
20 :    Load Mem[0]     ;20
21 :    Add Mem[1]      ;21
22 :    JMP 43  ;22

        ;a-b
23 :    Load Mem[0]     ;23
24 :    SUB Mem[1]      ;24
25 :    JMP 43  ;25

        ;a*b
26 :    Load Mem[0]     ;26
27 :    Mul Mem[1]      ;27
28 :    JMP 43  ;28

        ;a/b
29 :    Load Mem[1]     ;29
30 :    CMP 0   ;30
31 :    JZ 41   ;31
32 :    Load Mem[0]     ;32
33 :    DIV Mem[1]      ;33
34 :    JMP 43  ;34

        ;a%b
35 :    Load Mem[1]     ;35
36 :    CMP 0   ;36
37 :    JZ 41   ;37
38 :    Load Mem[0]     ;38
39 :    Mod Mem[1]      ;39
40 :    JMP 43  ;40

41 :    Load 0  ;41
42 :    JMP 43  ;42

43 :    Out     ;43

        ;Retour Ã  la demande de l'opÃ©ration (c)
44 :    JMP 4   ;44
45 :    End     ;45


=> TP2_DUPONT_Corentin/pgcd.asm :
=========================
        ;DUPONT Corentin
        ;07/04/2020
```

TB.
Et avec les commentaires c'est très clair.

```
0 :     In      ;0
1 :     Store Mem[0]    ;1 -> a
2 :     In      ;2
3 :     Store Mem[1]    ;3 -> b
4 :     CMP Mem[0]      ;4
5 :     JC 7    ;5
6 :     JMP 13  ;6

        ;J'Ã©change a et b pour avoir la plu grande valeur dans a
7 :     Load Mem[0]     ;7
8 :     Store Mem[2]    ;8
9 :     Load Mem[1]     ;9
10 :    Store Mem[0]    ;10
11 :    Load Mem[2]     ;11
12 :    Store Mem[1]    ;12

        ;Euclide
13 :    Load Mem[0]     ;13
14 :    Mod Mem[1]      ;14
15 :    Store Mem[3]    ;15
16 :    CMP 0   ;16
17 :    JZ 23   ;17
18 :    Load Mem[1]     ;18
19 :    Store Mem[0]    ;19
20 :    Load Mem[3]     ;20
21 :    Store Mem[1]    ;21
22 :    JMP 13  ;22

        ;Affichage du PGCD (case mÃ©moire de b)
23 :    Load Mem[1]     ;23
24 :    Out     ;24
25 :    End     ;25
```

il faut des valeurs > 0 => In + CMP 0 + JZ 0 + Store (soackage une fois que c'est OK)

en fait l'échange n'est pas nécessaire :
si les valeurs ne sont pas dans le bon ordre,
le 1er tour les remet "à l'endroit"

; calcul du reste
; sortie si nul

; decalage des deux valeurs "de droite"
; vers les deux cases "de gauche"

; retour au test

avec des commentaires sur la méthode, c'est plus simple