Antoine Gicquel

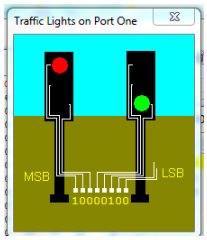
INFO 1 - A2

TP2 Assembleur : boucles et ports

P. Carreno - P. Portejoie

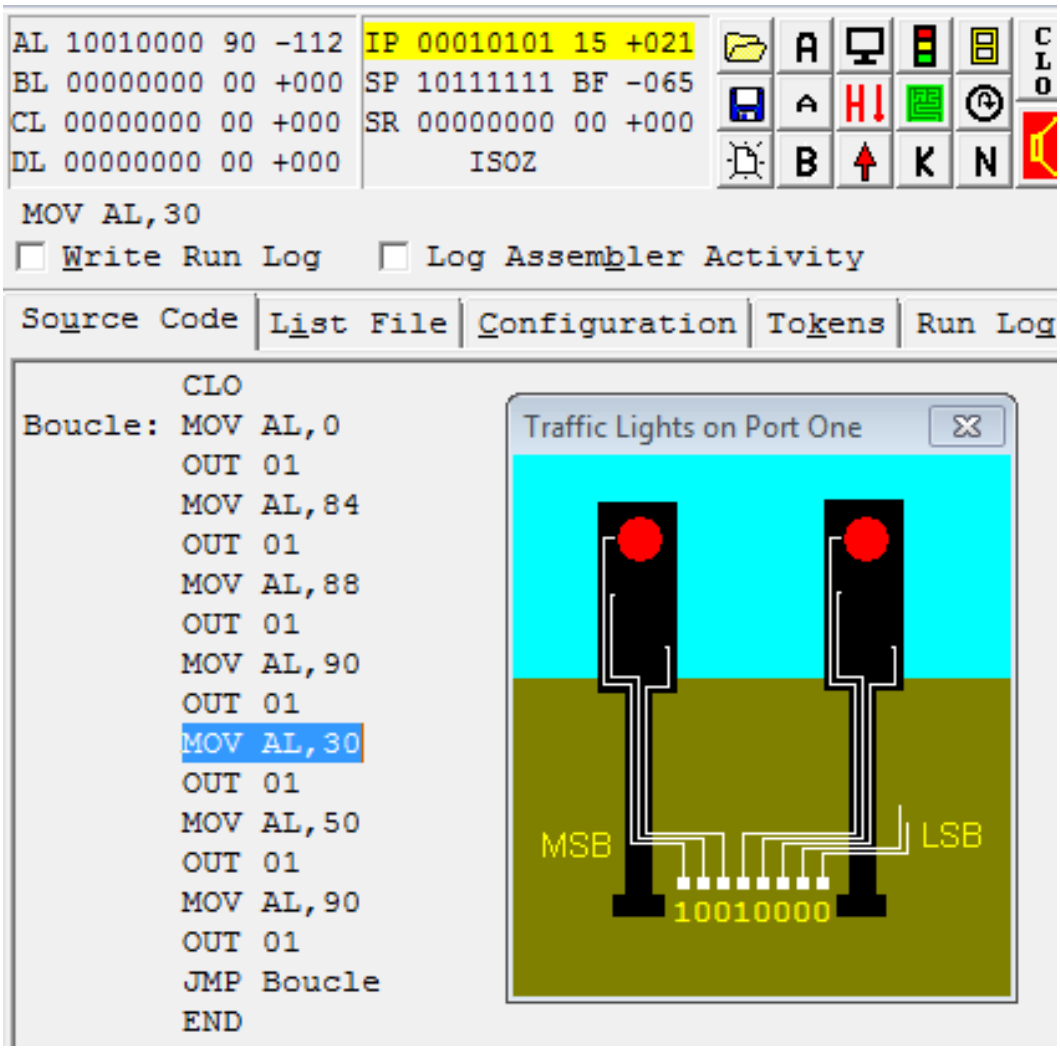
Exercice 2-1

- Reprenez le deuxième programme de l'exercice 2-1 du TD et faites-le fonctionner. Vérifiez la conformité des assemblages faits manuellement en TD.



- Complétez cette version de façon à ce que le programme simule le fonctionnement d'un carrefour à feux tricolores français. On donne pour cela l'information suivante concernant le paramétrage des feux (vous commencerez par compléter ce tableau) :

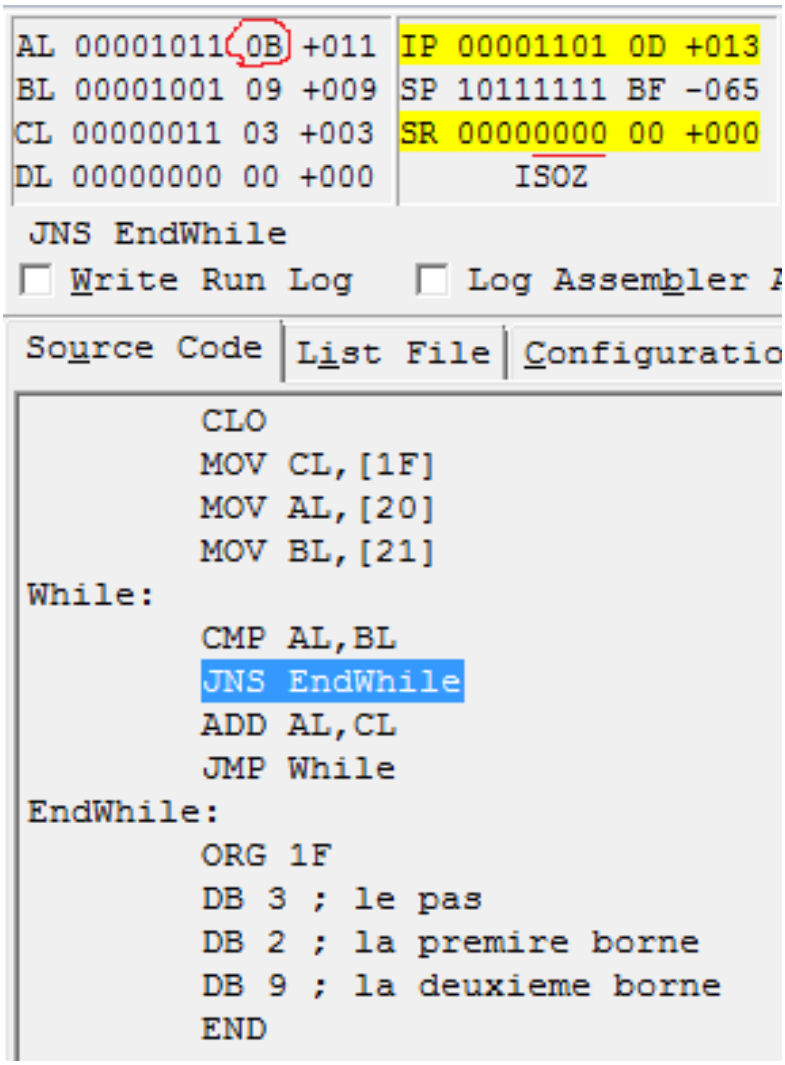
Rouge	Orange	Vert	Rouge	Orange	Vert	Inutilisé	Inutilisé	Hexadecimal	Feux	Temps
1	0	0	0	0	1	0	0	84	R-V	
1	0	0	0	1	0	0	0	88	R-O	
1	0	0	1	0	0	0	0	90	R-R	
0	0	1	1	0	0	0	0	30	V-R	
0	1	0	1	0	0	0	0	50	O-R	
1	0	0	1	0	0	0	0	90	R-R	


AL 10010000 90 -112
BL 00000000 00 +000
CL 00000000 00 +000
DL 00000000 00 +000
IP 00010101 15 +021
SP 10111111 BF -065
SR 00000000 00 +000
ISOZ
MOV AL,30
Write Run Log
Log Assembler Activity
Source Code
List File
Configuration
Tokens
Run Log
CLO
Boucle: MOV AL,0
OUT 01
MOV AL,84
OUT 01
MOV AL,88
OUT 01
MOV AL,90
OUT 01
MOV AL,30
OUT 01
MOV AL,50
OUT 01
MOV AL,90
OUT 01
JMP Boucle
END
Traffic Lights on Port One
MSB
LSB
10010000

Exercice 2-2

- Reprenez le programme de l'exercice 2-2 du TD et faites-le fonctionner. Vérifiez la conformité des assemblages faits manuellement en TD.

- Modifiez le programme afin de pouvoir en fixer le pas **par initialisation d'une constante en mémoire, à l'assemblage (*DB*)**. Testez-le avec *2 à 5, pas de 1 (==> observation de 2 3 4 5 dans AL)*, puis avec *2 à 9, pas de 3 (==> observation de 2 5 8 dans AL)*.

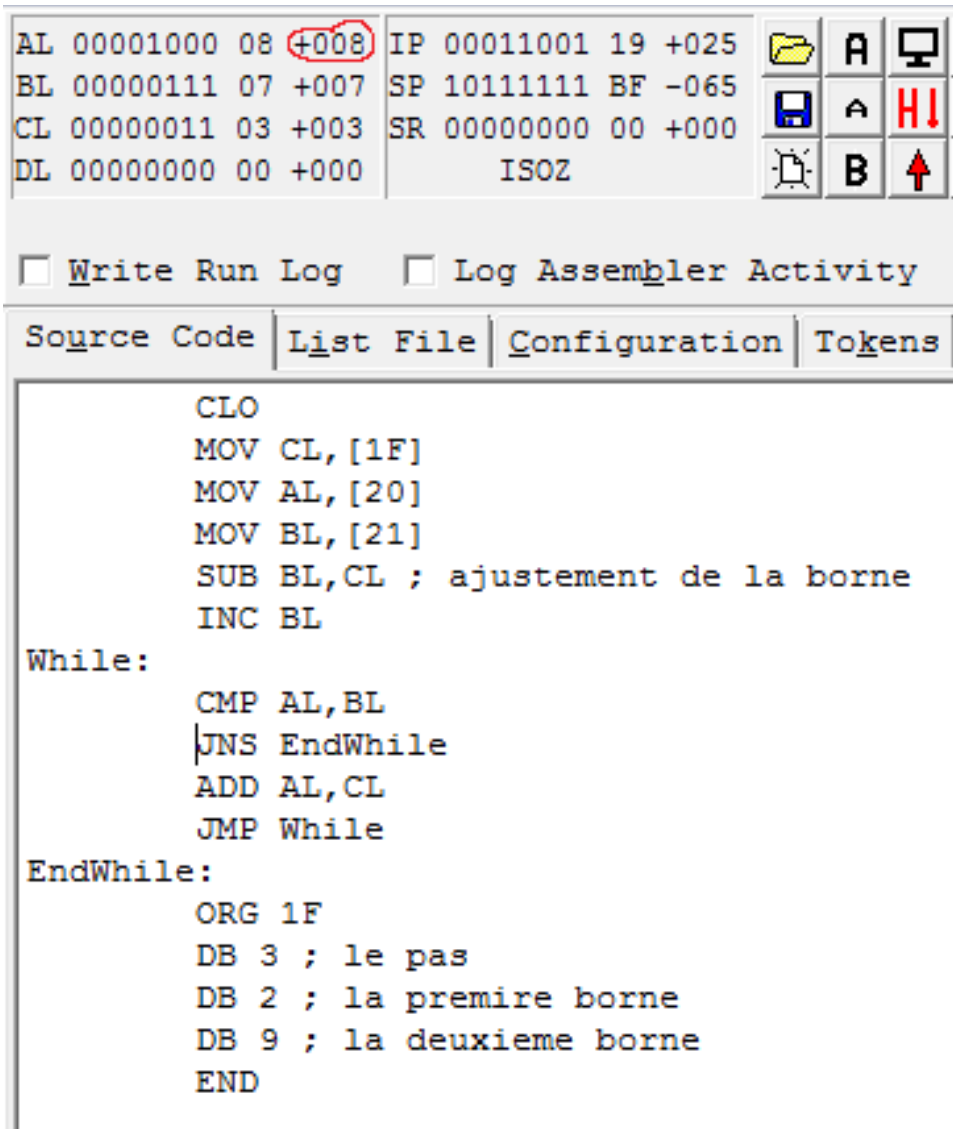


AL passe à 0B c'est à dire 11 en hexa alors qu'il devrait s'arrêter logiquement à 8.

NB : à chaque phase de test observez bien le contenu de AL (c'est là que se construit le résultat) pour en vérifier la conformité avec l'énoncé (dépassement de la borne supérieure interdit). En cas de problème, passez à la question ci-après.

- Dans le cas du pas de 3, le problème de dépassement de la borne supérieure que vous avez dû observer est purement algorithmique : l'algorithme qui ne posait pas de problème par pas de 1 doit être modifié pour permettre un pas supérieur à 1. Pour ce faire il faut ajuster la borne supérieure avant la boucle par : **BL := BL - pas + 1** (hypothèse que BL reçoit RAM[1F] en début de programme).

  Corrigez le programme et vérifiez à nouveau son fonctionnement dans les 2 cas.



Ici AL vaut 8, il s'arrête donc avant 9, le problème est corrigé.

Exercice 2-3

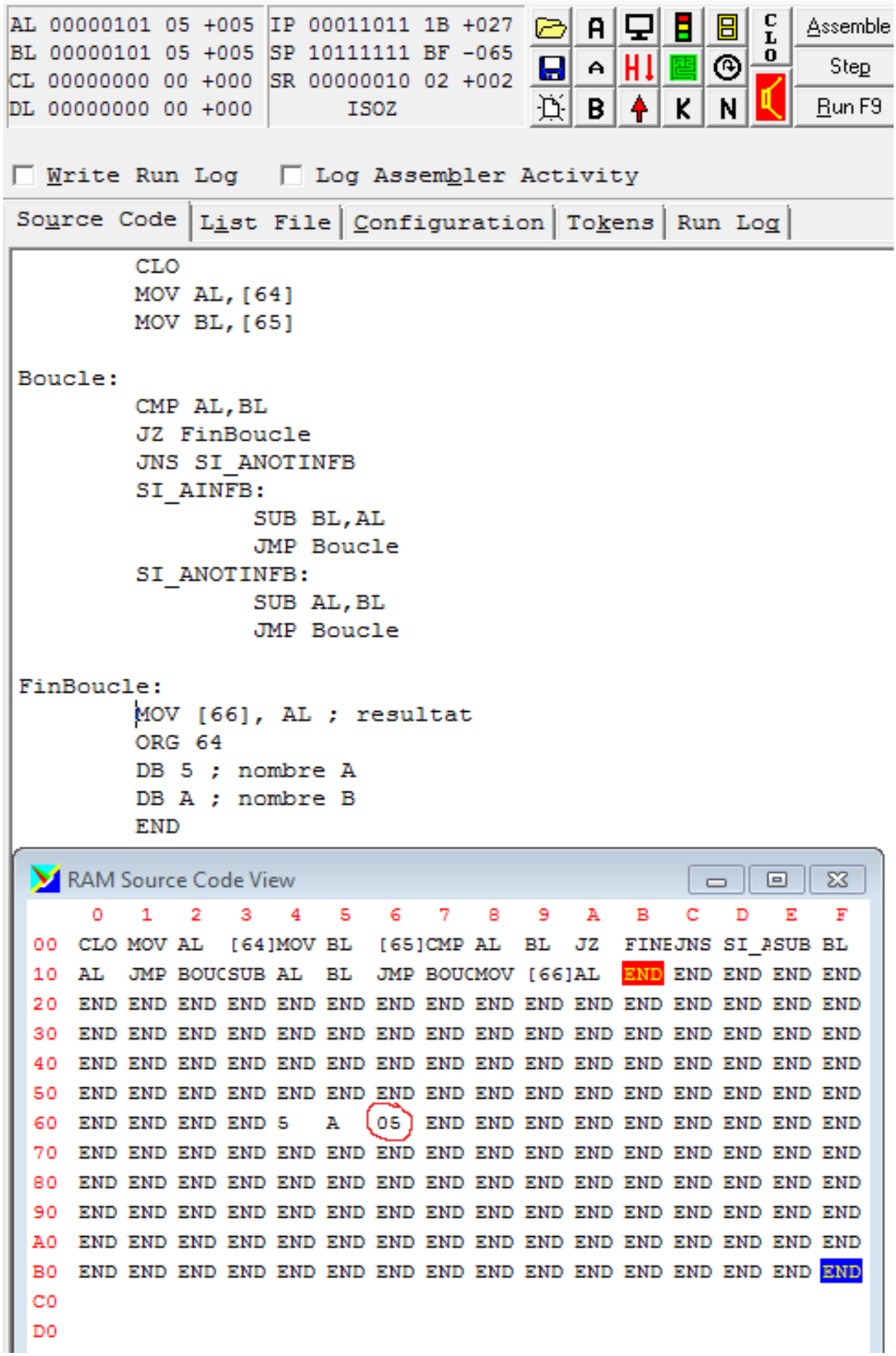
- Écrivez un programme qui calcule le PGCD de 2 nombres initialement stockés en RAM aux adresses 100 et 101 et range le résultat à l'adresse 102. L'algorithme vous est donné ci-dessous ; aidez-vous également des exemples de traduction fournis en annexe.

Algorithme

```
TANT_QUE (A<>B)
       SI (A<B) ALORS
            B := B - A
       SINON
            A := A - B
       FINSI
FIN_TANT_QUE
PGCD := A              ; ou PGCD := B
```

- Faites-le fonctionner (jeu de tests au choix).

- De façon identique à ce que vous avez fait en TD, justifiez le calcul d'offset pour 2 instructions de branchement de votre choix (un cas de déplacement positif et un cas de déplacement négatif).



Le PGCD de 5 et 10 est bien 5 donc le résultat est correct.

Le code machine obtenu du simulateur :

```
        CLO                         ; [00]  FE
        MOV  AL,[64]                ; [01]  D1 00 64
        MOV  BL,[65]                ; [04]  D1 01 65
                                    ;
BOUCLE:                             ;
        CMP  AL,BL                  ; [07]  DA 00 01
        JZ   FINBOUCLE              ; [0A]  C1 0E
        JNS  SI_ANOTINFB            ; [0C]  C4 07
SI_AINFB:                           ;
        SUB  BL,AL                  ; [0E]  A1 01 00
        JMP  BOUCLE                 ; [11]  C0 F6
SI_ANOTINFB:                        ;
        SUB  AL,BL                  ; [13]  A1 00 01
        JMP  BOUCLE                 ; [16]  C0 F1
                    ;
FINBOUCLE:                  ;
        MOV  [66],AL                ; [18]  D2 66 00
        ORG  64                     ;
        DB   5                      ; [64]  05
        DB   A                      ; [65]  0A
        END                         ; [66]  00
```

Calcul du offset :

Offset(Etq) = Emplacement(Etq) + Complément à deux de Référence(Etq)

Étiquette	Emplacement	Référence
Boucle	07	16

07 = 0000 0111
16 = 0001 0110

¬16 = 1110 1001
\+ 1
1110 1010

¬16 + 07 = 0000 0111
\+
1110 1010
= 1111 0001
= F1

C'est donc OK.

Étiquette	Emplacement	Référence
Boucle	07	11

07 = 0000 0111
11 = 0001 0001

¬11 = 1110 1110
+ 1
= 1110 1111

¬11 + 07 = 1110 1111
+ 0000 0111
= 1111 0110
= F6

C'est donc OK.

Étiquette	Emplacement	Référence
FinBoucle	18	0A

18 = 0001 1000
0A = 0000 1010

¬0A = 1111 0101
+ 1
= 1111 0110

¬0A + 18 = 1111 0110
+ 0001 1000
= 1 0000 1110
= 0E (car 8 bits uniquement)

C'est donc OK.

Étiquette	Emplacement	Référence
SI_ANOTINFB	13	0C

13 = 0001 0011
0C = 0000 1100

¬0C = 1111 0011
+ 1
= 1111 0100

¬0C + 13 = 1111 0100
+ 0001 0011
=1 0000 0111
= 07 (car 8 bits uniquement)

C'est donc OK.