

TD2 Assembleur : boucles et ports

P. Carreno - P. Portejoie

Samuel Le Berre

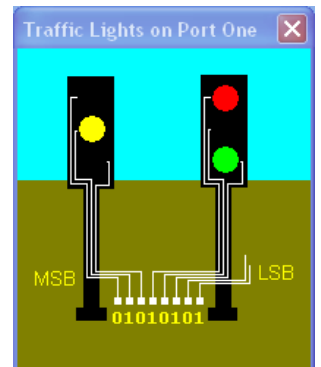
Groupe A2

Exercice 2-1

Observez le programme ci-dessous qui simule succinctement le fonctionnement de feux tricolores.

```
; Rôle : simule succinctement le fonctionnement de feux tricolores
; =====
;
      CLO
Boucle:
      MOV AL,0      ; Eteint toutes les lampes des feux tricolores
      OUT 01        ; en mettant le code 00000000 dans AL
                   ; puis en envoyant AL au port correspondant

      MOV AL,FC     ; De la même façon allume toutes les lampes
      OUT 01        ; des feux tricolores (code FC)
      JMP Boucle    ; Recommence la séquence précédente
      END
```



- Donnez-en la traduction en langage d'assemblage du simulateur sms32 ; vous vous référerez à l'annexe « Calcul des sauts » pour résoudre le branchement inconditionnel *JMP Boucle* (Aidez-vous de la table des références construite par l'assembleur)
- Complétez le programme de façon à ce qu'il s'arrête si l'utilisateur appuie sur la touche *Entrée* en fin de séquence (vous devrez faire en sorte qu'il soit sollicité).
NB : *Entrée* = *Enter* = *RC* (Retour Chariot) = *CR* (Carriage Return)
- Refaites-en l'assemblage en portant votre attention sur le calcul d'offset ; pensez à construire, au fur et à mesure, la table des références et à l'utiliser pour les résoudre. Comme vu précédemment vérifiez vos résultats.



Exercice 2-2

- Ecrivez un programme qui calcule dans AL, par pas de 1, les nombres entre une borne inférieure et une borne supérieure (à définir à l'assemblage par 2 constantes stockées dans le 31^{ème} et le 32^{ème} mot de la mémoire ; attention, réfléchissez bien à leur adresse).
NB : la borne supérieure peut être égale, mais en aucun cas dépassée.

Exemple : 2 et 5 ==> 2 3 4 5 doivent apparaître chronologiquement dans AL

- Vérifiez sa conformité en simulant son exécution (faites-le tourner « à la main »).

- Donnez-en la traduction en langage d'assemblage du simulateur sms32.

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.

Tous est conforme, s'assemble et s'exécute correctement

The screenshot shows an assembler window with the following source code:

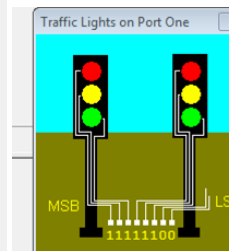
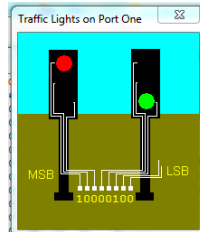
```

Boucle:
MOV AL,0
OUT 01
MOV AL,FC
OUT 01
IN 00
CMP AL,0D
JNZ Boucle
END
  
```

The assembly output shows the generated machine code and addresses:

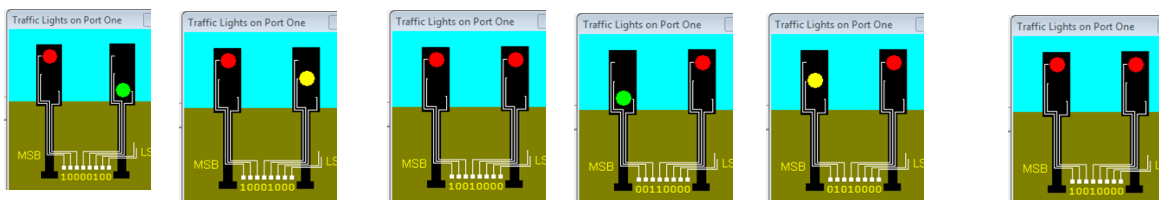
```

; LIST FILE : SUCCESS : No errors found.
; This shows the machine code that was generated and
; the addresses at which the codes were stored.
;
;      CLO      ; [00] FE
BOUCLE:
MOV     AL,0    ; [01] D0 00 00
OUT     01      ; [05] F1 01
MOV     AL,FC   ; [06] D0 00 FC
OUT     01      ; [0A] F1 01
IN      00      ; [0C] F0 00
CMP     AL,0D   ; [0D] DB 00 0D
JNZ     Boucle  ; [10] C2 F1
END      ; [12] 00
; SUCCESS : No errors found
  
```



- 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): **On remplace FC par l'Hexadécimal correspondant à l'état souhaité.**

Rouge	Orange	Vert	Rouge	Orange	Vert	Inutilisé	Inutilisé	Hexadécimal	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	

**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. **Tout est normal et l'assemblage est bien fait**

- 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 (\Rightarrow observation de 2 3 4 5 dans AL), puis avec 2 à 9, pas de 3 (\Rightarrow observation de 2 5 8 dans AL).

Write Run Log Log Assembler Activity

Source Code List File

RAM Source Code View

```

CLO
MOV AL, [1F]
MOV BL, [20]
MOV CL, 1
While:
  CMP AL, BL
  JNS EndWhile
  ADD AL, CL
  JMP While
EndWhile:
  ORG 1F
  DB 2
  DB 5
  END

```

0 1 2 3 4 5 6 7 8 9 A B C D E F

00 CLO MOV AL [1F] MOV BL [20] MOV CL 1 CMP AL BL JNS END*ADD :

10 AL CL JMP WHILE END END END END END END END END END 2

20 5 END END END END END END END END END END END END END END

30 END END END END END END END END END END END END END END

40 END END END END END END END END END END END END END END

50 END END END END END END END END END END END END END END

60 END END END END END END END END END END END END END END

70 END END END END END END END END END END END END END END

80 END END END END END END END END END END END END END END

90 END END END END END END END END END END END END END END

A0 END END END END END END END END END END END END END END

B0 END END END END END END END END END END END END END END

END

H:\DUT\Semestre2\M2101\TP2\TP2.ASM

File Edit View Examples Help

AL 00001011 0B +011 IP 00010100 14 +020

BL 00001001 09 +009 SP 10111111 BF -065

CL 00000011 03 +003 SR 00000000 00 +000

DL 00000000 00 +000 ISOZ

Write Run Log Log Assembler Activity

Source Code List File

RAM Source Code View

```

CLO
MOV AL, [1F]
MOV BL, [20]
MOV CL, 3
While:
  CMP AL, BL
  JNS EndWhile
  ADD AL, CL
  JMP While
EndWhile:
  ORG 1F
  DB 2
  DB 9
  END

```

0 1 2 3 4 5 6 7 8 9 A B C D E F

00 CLO MOV AL [1F] MOV BL [20] MOV CL 3 CMP AL BL JNS END*ADD :

10 AL CL JMP WHILE END END END END END END END END END 2

20 9 END END END END END END END END END END END END END END

30 END END END END END END END END END END END END END END

40 END END END END END END END END END END END END END END

50 END END END END END END END END END END END END END END

60 END END END END END END END END END END END END END END

70 END END END END END END END END END END END END END END

80 END END END END END END END END END END END END END END

90 END END END END END END END END END END END END END END

A0 END END END END END END END END END END END END END END

B0 END END END END END END END END END END END END END END

END

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.

The screenshot displays the MASM68000 assembler interface. The top window shows the source code for a program with a loop step of 1. The bottom window shows the same program but with a loop step of 3. Both versions include a 'While' loop and a 'RAM Source Code View' window showing memory addresses 0 to B0.

Top Window (Step 1):

```

AL 00000101 05 +005 IP 00011010 1A +026
BL 00000101 05 +005 SP 10111111 BF -065
CL 00000001 01 +001 SR 00000010 02 +002
DL 00000000 00 +000      ISOZ

JNS EndWhile
Write Run Log Log Assembler Activity

Source Code List File
CLO
MOV AL,[1F]
MOV BL,[20]
MOV CL,1
SUB BL,CL
ADD BL,1
While:
CMP AL,BL
JNS EndWhile
ADD AL,CL
JMP While
EndWhile:
ORG 1F
DB 2
DB 5
END
  
```

Bottom Window (Step 3):

```

AL 00001000 08 +008 IP 00011010 1A +026
BL 00000111 07 +007 SP 10111111 BF -065
CL 00000011 03 +003 SR 00000000 00 +000
DL 00000000 00 +000      ISOZ

JNS EndWhile
Write Run Log Log Assembler Activity

Source Code List File
CLO
MOV AL,[1F]
MOV BL,[20]
MOV CL,3
SUB BL,CL
ADD BL,1
While:
CMP AL,BL
JNS EndWhile
ADD AL,CL
JMP While
EndWhile:
ORG 1F
DB 2
DB 9
END
  
```

RAM Source Code View:

The RAM Source Code View window shows memory addresses 0 to B0. The top version (step 1) shows the program code starting at address 00. The bottom version (step 3) shows the program code starting at address 00, but with a different loop step (3) and a different initial value for BL (11).

Exercice 2-3

- Ecrivez 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).

The screenshot shows an assembly editor window titled "H:\DUT\Semestre2\M2101\TP2\TP2.ASM". The main window displays assembly code with a list of instructions and their hex values. Below the code, there are checkboxes for "Write Run Log" and "Log Assembler Activity". A "RAM Source Code View" window is open, showing a memory dump with addresses 00 to E0 and corresponding hex values. The source code in the main window includes a loop structure using "WHILE" and "ENDWHILE" keywords, with instructions like "MOV AL, [63]", "MOV BL, [64]", "CMP AL, BL", "JZ ENDWHILE", "JNS SI_ANOTINB", "SI_AINB:", "SUB BL, AL", "JMP FINI", "SI_ANOTINB:", "SUB AL, BL", "FINI:", and "ENDWHILE:". The RAM source code view shows a memory dump with addresses 00 to E0 and corresponding hex values, including "CLO MOV AL [63] MOV BL [64] CMP AL BL JZ ENDWJNS SI_ASUB BL", "10 AL JMP FINESUB AL BL MOV [65] AL", "20 END END END END END END END END END END END END END END END END", "30 END END END END END END END END END END END END END END END END", "40 END END END END END END END END END END END END END END END END", "50 END END END END END END END END END END END END END END END END", "60 END END END 4 8 04 END END END END END END END END END END END", "70 END END END END END END END END END END END END END END END END", "80 END END END END END END END END END END END END END END END END", "90 END END END END END END END END END END END END END END END END", "A0 END END END END END END END END END END END END END END END END", "B0 END END END END END END END END END END END END END END END END", "C0", "D0", "E0".

Calcul par table de référence :

Etat	Emplacement	Référence
ENDWHILE	16	0A
SI_ANOTINB	13	0C
FINSI	16	11

- $16-0A \Leftrightarrow 01 + (-0A) \Leftrightarrow 16 + F6 = 0001\ 0110 + 1111\ 0110 = 0000\ 1100 = 0C$

$0A = 0000\ 1010$

$-0A = 1111\ 0101 + 1 \Rightarrow 1111\ 0110 \Rightarrow F6$

- $13-0C \Leftrightarrow 13 + (-0C) \Leftrightarrow 13 + F4 = 0001\ 0011 + 1111\ 0100 = 0000\ 0111 = 07$

$0C = 0000\ 1100$

$-0C = 1111\ 0011 + 1 \Rightarrow 1111\ 0100 \Rightarrow F4$

- $16-11 \Leftrightarrow 16 + (-11) \Leftrightarrow 16 + EF = 0001\ 0110 + 1110\ 1111 = 0000\ 0101 = 05$

$11 = 0001\ 0001$

$-11 = 1110\ 1110 + 1 \Rightarrow 1110\ 1111 \Rightarrow EF$

Calcul par les mots :

$ENDWHILE \Rightarrow 11\ \text{mots} \Leftrightarrow 0B + 1 = 0000\ 1011 + 1 = 0000\ 1100 \Rightarrow 0C$

$SI_ANOTINB \Rightarrow 6\ \text{mots} \Leftrightarrow 06 + 1 = 0000\ 0110 + 1 = 0000\ 0111 \Rightarrow 07$

$FINSI \Rightarrow 4\ \text{mots} \Leftrightarrow 04 + 1 = 0000\ 0100 + 1 = 0000\ 0101 = 05$