

Antoine Gicquel**A2****TP4 Assembleur : E/S en ASCII**

P. Carreno - P. Portejoie

Exercices 4-1 et 4-2

Reprenez les programmes des exercices 4-1 et 4-2 du TD et faites-les fonctionner.
Vérifiez la conformité des résultats.

1)

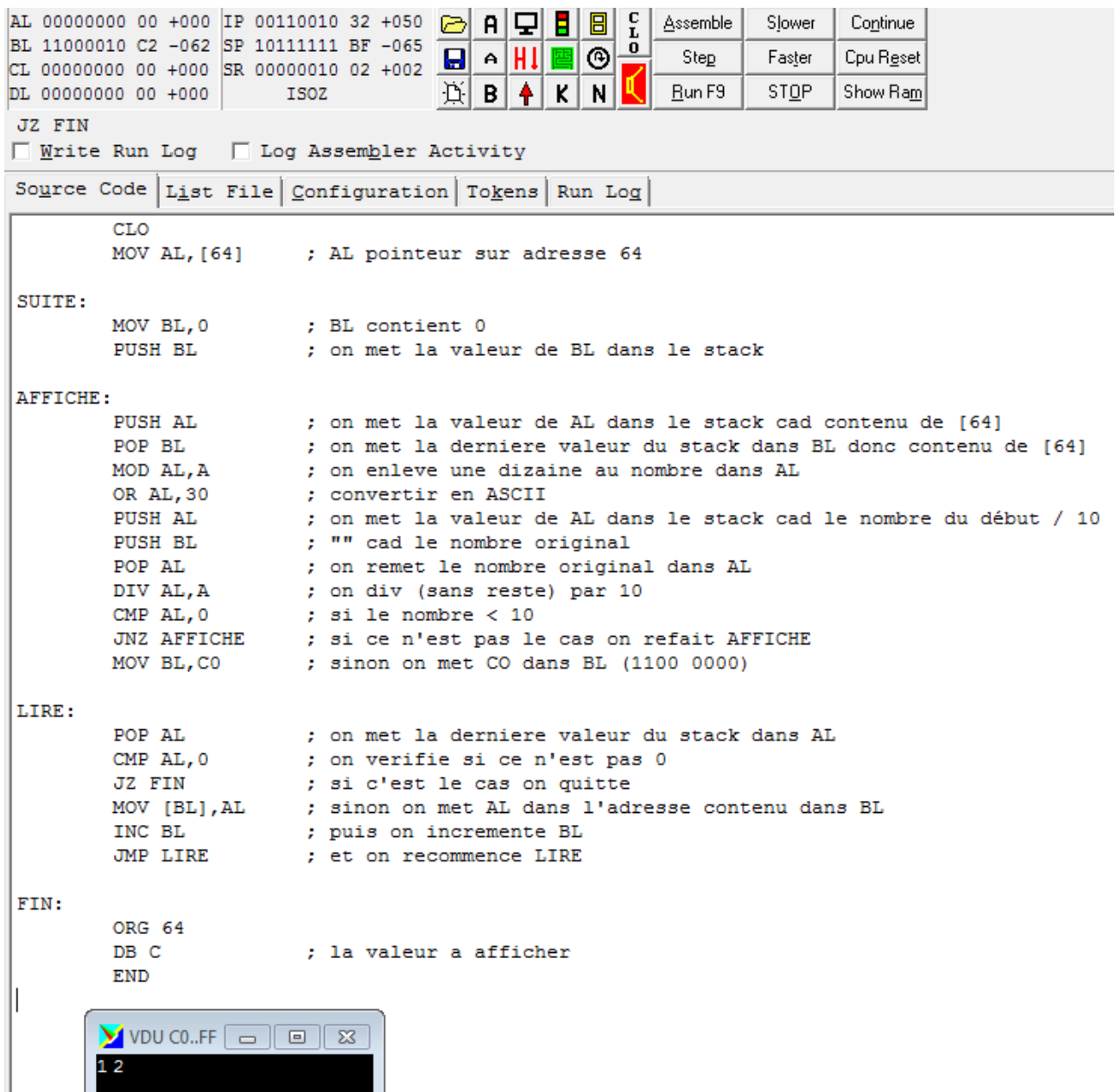
AL 00001101 0D +013	IP 00011001 19 +025
BL 00000000 00 +000	SP 10111111 BF -065
CL 00100010 22 +034	SR 00000010 02 +002
DL 00000000 00 +000	ISOZ

☐ Write Run Log ☐ Log Assembler

Source Code	List File	Configuratio
CLO MOV CL,0 BOUCLE: IN 0 CMP AL,0D JZ FIN AND AL,CF MUL CL,A ADD CL,AL JMP BOUCLE FIN: MOV [64],CL END		

Essai avec les nombres 3 et 4 qui donne 34 en CL.

2)



Exemple avec C qui vaut 12.

CODE :

```

CLO
MOV AL,[64]  ; AL pointeur sur adresse 64
  
```

SUITE:

```

MOV BL,0      ; BL contient 0
PUSH BL      ; on met la valeur de BL dans le stack
  
```

AFFICHE:

```

PUSH AL      ; on met la valeur de AL dans le stack cad contenu de [64]
POP BL      ; on met la dernière valeur du stack dans BL donc contenu de [64]
  
```

```
MOD AL,A      ; on enleve une dizaine au nombre dans AL
OR AL,30      ; convertir en ASCII
PUSH AL       ; on met la valeur de AL dans le stack cad le nombre du début / 10
PUSH BL       ; "" cad le nombre original
POP AL        ; on remet le nombre original dans AL
DIV AL,A      ; on div (sans reste) par 10
CMP AL,0      ; si le nombre < 10
JNZ AFFICHE   ; si ce n'est pas le cas on refait AFFICHE
MOV BL,C0     ; sinon on met C0 dans BL (1100 0000)
```

LIRE:

```
POP AL        ; on met la dernière valeur du stack dans AL
CMP AL,0      ; on vérifie si ce n'est pas 0
JZ FIN        ; si c'est le cas on quitte
MOV [BL],AL   ; sinon on met AL dans l'adresse contenu dans BL
INC BL        ; puis on incrémente BL
JMP LIRE      ; et on recommence LIRE
```

FIN:

```
ORG 64
DB C          ; la valeur à afficher
END
```

Exercice 4-3

Ré-écrivez le programme de calcul du PGCD vu lors du TD-TP n°2 en lui ajoutant la saisie des 2 entiers à traiter ainsi que l'affichage du résultat.

Test avec 25 et 5 :

The screenshot shows an 8086 assembler interface. At the top, there's a status bar with registers: AL 00000000, BL 11000001, CL 00000000, DL 00000000, IP 01101000, SP 10111111, SR 00000010, and ISOZ. Below this are icons for file operations and a control panel with buttons: Assemble, Step, Run F9, Slower, Faster, STOP, Continue, and Show Ram. There are checkboxes for 'Write Run Log' and 'Log Assembler Activity'. A tabbed interface shows 'Source Code', 'List File', 'Configuration', 'Tokens', and 'Run Log'. The 'Source Code' tab is active, displaying assembly code with comments in French. A small window titled 'V...' shows the value '5'. A larger window titled 'RAM Source Code View' is open, showing a memory dump with columns for address (0-15), hex, and source code. The source code in the RAM view matches the code in the main window, with 'END' highlighted in red at address 60.

```

SI_ANOTINFB:
    SUB AL,BL
    JMP PGCD

SUITE2:
    MOV BL,0      ; BL contient 0
    PUSH BL       ; on met la valeur de BL dans le stack

AFFICHE:
    PUSH AL       ; on met la
    POP BL        ; on met la
    MOD AL,A      ; on enleve
    OR AL,30      ; convertir
    PUSH AL       ; on met la
    PUSH BL       ; "" cad le
    POP AL        ; on remet
    DIV AL,A      ; on div (se
    CMP AL,0      ; si le nom
    JNZ AFFICHE   ; si ce n'es
    MOV BL,C0     ; sinon on

LIRE:
    POP AL        ; on met la
    CMP AL,0      ; on verifie
    JZ FIN        ; si c'est
    MOV [BL],AL   ; sinon on
    INC BL        ; puis on i
    JMP LIRE      ; et on rec

FIN:
    END
  
```

The 'RAM Source Code View' window shows the following memory dump:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00	CLO	MOV	CL	0	MOV	BL	0	IN	0	CMP	AL	0D	JZ	SUIT	AND	AL
10	CF	MUL	CL	A	ADD	CL	AL	JMP	BOUC	PUSH	CL	INC	BL	MOV	CL	0
20	CMP	BL	2	JNZ	BOUC	POP	AL	POP	BL	CMP	AL	BL	JZ	SUIT	JNS	SI_AN
30	SUB	BL	AL	JMP	PGCIS	SUB	AL	BL	JMP	PGCIM	MOV	BL	0	PUSH	FBL	PUSH
40	AL	POP	BL	MOD	AL	A	OR	AL	30	PUSH	AL	PUSH	FBL	POP	AL	DIV
50	AL	A	CMP	AL	0	JNZ	AFFI	MOV	BL	C0	POP	AL	CMP	AL	0	JZ
60	FIN	MOV	[BL]	AL	INC	BL	JMP	LIRE	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	05	35
C0	35															00
D0																
E0																
F0																

At the bottom of the RAM Source Code View window, there are radio buttons for 'Hexadecimal', 'ASCII', and 'Source', with 'Source' selected.

CODE:

```

CLO
MOV CL,0
MOV BL,0
BOUCLE:
IN 0
CMP AL,0D      ; si on appuye sur entrée
JZ SUITE
AND AL,CF
MUL CL,A
ADD CL,AL
JMP BOUCLE

SUITE:
PUSH CL
INC BL
MOV CL,0
CMP BL,2
JNZ BOUCLE
  
```

```
POP AL
POP BL
```

```
PGCD:
```

```
CMP AL,BL
JZ SUITE2
JNS SI_ANNOTINFB
SI_AINFB:
    SUB BL,AL
    JMP PGCD
SI_ANNOTINFB:
    SUB AL,BL
    JMP PGCD
```

```
SUITE2:
```

```
MOV BL,0      ; BL contient 0
PUSH BL       ; on met la valeur de BL dans le stack
```

```
AFFICHE:
```

```
PUSH AL       ; on met la valeur de AL dans le stack cad contenu de [64]
POP BL        ; on met la dernière valeur du stack dans BL donc contenu de [64]
MOD AL,A      ; on enlève une dizaine au nombre dans AL
OR AL,30      ; convertir en ASCII
PUSH AL       ; on met la valeur de AL dans le stack cad le nombre du début / 10
PUSH BL       ; "" cad le nombre original
POP AL        ; on remet le nombre original dans AL
DIV AL,A      ; on div (sans reste) par 10
CMP AL,0      ; si le nombre < 10
JNZ AFFICHE   ; si ce n'est pas le cas on refait AFFICHE
MOV BL,C0     ; sinon on met C0 dans BL (1100 0000)
```

```
LIRE:
```

```
POP AL        ; on met la dernière valeur du stack dans AL
CMP AL,0      ; on vérifie si ce n'est pas 0
JZ FIN        ; si c'est le cas on quitte
MOV [BL],AL   ; sinon on met AL dans l'adresse contenu dans BL
INC BL        ; puis on incrémente BL
JMP LIRE      ; et on recommence LIRE
```

```
FIN:
```

```
END
```

Exercice 4-4 (si le temps le permet...)

Ecrivez un programme qui trie sur place (ie sans zone de stockage intermédiaire), par ordre croissant, des caractères situés à partir du début de la mémoire vidéo (vous pourrez ainsi visualiser leur progression lors du tri). L'initialisation en sera faite à l'assemblage.

Le nombre de caractères à traiter sera disponible à l'adresse de votre choix et devra faire également l'objet d'une initialisation à l'assemblage.

Algorithme de tri proposé – Tri à bulles :

Il existe de nombreux algorithmes de tri. Vous mettrez en oeuvre le tri à bulle dont l'algorithme vous est donné ci-après. Ce n'est pas le plus performant mais c'est un des plus simples à programmer.

Son principe est de parcourir la liste (a_1, a_2, \dots, a_n) en intervertissant toute paire d'éléments consécutifs (a_{i-1}, a_i) non ordonnés. Ainsi après le premier parcours, l'élément maximum se retrouve en a_n . On suppose que l'ordre croissant s'écrit de gauche à droite (à gauche le plus petit élément, à droite le plus grand élément).

On recommence l'opération avec la nouvelle sous-suite (a_1, a_2, \dots, a_{n-1}), et ainsi de suite jusqu'à épuisement de toutes les sous-suites (la dernière est un couple). A tout instant, la partie gauche du tableau est non triée alors que la partie droite l'est. L'algorithme s'arrête lorsque la partie non triée devient vide.

Le nom de tri à bulle vient de ce qu'à la fin de chaque itération interne, le plus grand élément de chaque sous-suite se déplace vers la droite à l'image d'une bulle.

début

j := dernier_indice

ttque j <> 0 ; tant que la partie non triée n'est pas vide

i := 0

ttque i <> j ; remontée de la plus grande valeur du sous-tableau non trié

si T[i+1] < T[i] alors permutation de T[i+1] avec T[i] fsi

i := i+1

finttque

j := j-1 ; la dernière valeur traitée est considérée comme triée

finttque ; il y a donc une valeur de moins dans le sous-tableau non trié

fin

Essai avec "153476298"

The screenshot shows an 8086 assembler interface. The main window displays assembly code for a program that processes a string. The code includes labels like TRI_BULLES, BOUCLE, FINBOUCLE, FIN, and SI. The RAM Source Code View window shows the memory layout, with addresses 00 to FF. The code is assembled into machine code, with the first few bytes being CLO MOV AL, C0 and MOV BL, [AL].

```

AL 11000000 C0 -064 IP 01000000 40 +064
BL 11000001 C1 -063 SP 10111111 BF -065
CL 00110001 31 +049 SR 00000010 02 +002
DL 00110010 32 +050      ISOZ

TRI_BULLES:
    CMP AL,C0
    JZ FIN
    MOV BL,C0
    BOUCLE:
        CMP AL,BL
        JZ FINBOUCLE
        PUSH BL
        MOV CL,[BL] ; T[i]
        INC BL
        MOV DL,[BL] ; T[i+1]
        SI:
            CMP DL,CL
            JNS FINSI ; T[
            ; permutation
            MOV [BL],CL
            DEC BL
            MOV [BL],DL
        FINSI:
            POP BL
            INC BL
            JMP BOUCLE
    FINBOUCLE:
        DEC AL
        JMP TRI_BULLES
FIN:
    ORG C0
    DB "153476298"
    DB 0
    END
  
```

RAM Source Code View:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00	CLO	MOV	AL	C0	MOV	BL	[AL]	CMP	BL	0	JZ	INDI	INC	AL	JMP	DERNI
10	DEC	AL	CMP	AL	C0	JZ	FIN	MOV	BL	C0	CMP	AL	BL	JZ	FINE	PUSHE
20	BL	MOV	CL	[BL]	INC	BL	MOV	DL	[BL]	CMP	DL	CL	JNS	FIN	MOV	[BL]
30	CL	DEC	BL	MOV	[BL]	DL	POP	BL	INC	BL	JMP	BOUC	DEC	AL	JMP	TRI_B
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	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	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	1	32	33	34	35	36	37	38	39	0	END					
D0																
E0																
F0																

CODE:

```

CLO
MOV AL,C0
  
```

; AL contiendra le dernier indice

DERNIER_INDICE:

```

MOV BL,[AL]
CMP BL,0
JZ INDICE
INC AL
JMP DERNIER_INDICE
  
```

INDICE:

DEC AL ; de 0 a TAILLE-1

TRI_BULLES:

```

CMP AL,C0
JZ FIN
MOV BL,C0
BOUCLE:
    CMP AL,BL
    JZ FINBOUCLE
    PUSH BL
    MOV CL,[BL] ; T[i]
    INC BL
    MOV DL,[BL] ; T[i+1]
    SI:
  
```

```
        CMP DL,CL
        JNS FINSI ; T[i+1] > T[i]
        ; permuttation
        MOV [BL],CL
        DEC BL
        MOV [BL],DL

    FINSI:
        POP BL
        INC BL
        JMP BOUCLE

    FINBOUCLE:
        DEC AL
        JMP TRI_BULLES

FIN:
    ORG C0
    DB "153476298"
    DB 0
    END
```