

Antoine Gicquel

A2

**TP5 Assembleur : sous-programmes**

P. Carreno - P. Portejoie

**Exercices 5-1 et 5-2**

Reprenez les programmes des exercices 5-1 et 5-2 du TD et faites-les fonctionner.  
Vérifiez la conformité des résultats.

**1) Test avec « 56 »**

The screenshot shows an x86 assembler interface with the following components:

- Registers and Memory:**
  - AL 00000000 00 +000
  - BL 11000010 C2 -062
  - CL 00111000 38 +056
  - DL 00000000 00 +000
  - IP 00000100 04 +004
  - SP 10111111 BF -065
  - SR 00000010 02 +002
  - ISOZ
- Control Panel:**
  - Buttons: Assemble, Slower, Continue, Step, Faster, Cpu Reset, Run F9, STQP, Show Ram.
  - Checkboxes: ☐ Write Run Log, ☐ Log Assembler Activity.
  - Tabs: Source Code, List File, Configuration, Tokens, Run Log.
- Source Code:**

```

      ORG 20
      MOV CL,0
LIRE: IN 0
      CMP AL,0D
      JZ FIN1
      AND AL,CF
      MUL CL,A
      ADD CL,AL
      JMP LIRE
FIN1: PUSH CL
      POP AL
      RET

      ORG 50
SUITE: MOV BL,0
      PUSH BL
AFFICH: PUSH AL
      POP BL
      MOD AL,A
      OR AL,30
      PUSH AL
      PUSH BL
      POP AL
      DIV AL,A
      CMP AL,0
      JNZ AFFICH
      MOV BL,C0
DEPILE: POP AL
      CMP AL,0
      JZ FIN2
      MOV [BL],AL
      INC BL
      JMP DEPILE
FIN2: RET

      ; Appel des sous programmes
      ORG 0
      CALL 20
      CALL 50

```
- VDU C0..FF:** A window showing the value 56.
- RAM Source Code View:** A window showing memory addresses and their corresponding assembly instructions. The address 04 is highlighted in blue.

## CODE:

```
ORG 20
MOV CL,0
LIRE:  IN 0
CMP AL,0D
JZ FIN1
AND AL,CF
MUL CL,A
ADD CL,AL
JMP LIRE
FIN1:  PUSH CL
POP AL
RET
```

```
ORG 50
SUITE: MOV BL,0
PUSH BL
AFFICH: PUSH AL
POP BL
MOD AL,A
OR AL,30
PUSH AL
PUSH BL
POP AL
DIV AL,A
CMP AL,0
JNZ AFFICH
MOV BL,C0
DEPILE: POP AL
CMP AL,0
JZ FIN2
MOV [BL],AL
INC BL
JMP DEPILE
FIN2:  RET
```

```
; Appel des sous programmes
ORG 0
CALL 20
CALL 50
END
```

## 2) Avec utilisation de la pile

The screenshot shows an assembly software interface. At the top, there's a status bar with registers (AL, BL, CL, DL) and their values, and a control panel with buttons like 'Assemble', 'Step', 'Run F9', etc. Below this is a source code editor with the following code:

```

ORG 20
POP DL
LIRE: IN 0
      CMP AL,0D
      JZ FIN1
      AND AL,CF
      MUL CL,A
      ADD CL,AL
      JMP LIRE
FIN1: PUSH CL
      PUSH DL
      RET

      ORG 50
SUITE: POP DL
      POP AL
AFFICH: PUSH AL
      POP BL
      MOD AL,A
      OR AL,30
      PUSH AL
      PUSH BL
      POP AL
      DIV AL,A
      CMP AL,0
      JNZ AFFICH
      MOV BL,C0
DEPILE: POP AL
      CMP AL,0
      JZ FIN2
      MOV [BL],AL
      INC BL
      JMP DEPILE
FIN2: PUSH DL
      RET

; Appel des sous programmes
ORG 0
MOV AL,0
PUSH AL
CALL 20
CALL 50
END
  
```

Overlaid on the source code editor are two windows:

- VDU C0..FF**: A small window showing the value '56'.
- RAM Source Code View**: A window showing a memory dump. The first column shows addresses from 00 to F0. The second column shows the hex values of the instructions. The third column shows the source code for each instruction. The instruction at address B0 is highlighted in blue and shows '09' in the hex column.

Code:

```

ORG 20
POP DL
LIRE: IN 0
      CMP AL,0D
      JZ FIN1
      AND AL,CF
      MUL CL,A
      ADD CL,AL
  
```

```
JMP LIRE  
FIN1:  PUSH CL  
        PUSH DL  
        RET
```

```
        ORG 50  
SUITE:  POP DL  
        POP AL  
AFFICH: PUSH AL  
        POP BL  
        MOD AL,A  
        OR AL,30  
        PUSH AL  
        PUSH BL  
        POP AL  
        DIV AL,A  
        CMP AL,0  
        JNZ AFFICH  
        MOV BL,C0  
DEPILE: POP AL  
        CMP AL,0  
        JZ FIN2  
        MOV [BL],AL  
        INC BL  
        JMP DEPILE  
FIN2:   PUSH DL  
        RET
```

```
; Appel des sous programmes  
ORG 0  
MOV AL,0  
PUSH AL  
CALL 20  
CALL 50  
END
```

**Exercice 5-3**

Transformez le programme de calcul du PGCD ré-écrit lors du TP précédent (avec saisie des 2 entiers à traiter et affichage du résultat) de façon à ce qu'il soit construit modulairement (décomposition en sous-programmes). Vous le décomposerez en 3 sous-programmes :

- un sous-programme de saisie d'un nombre  
→ en sortie AL contient le nombre lu
- un sous-programme d'affichage d'un nombre  
→ en entrée AL contient le nombre à afficher
- un sous-programme du calcul du PGCD  
→ en entrée AL contient le premier nombre  
→ en entrée BL contient le deuxième nombre  
→ en sortie CL contient le résultat

Le passage des paramètres s'effectuera donc par l'intermédiaire de registres.

Code :

```
ORG 20
POP DL

;;; 1 - LECTURE
LIRE:
IN 0
CMP AL,0D
JZ SUITE1
AND AL,CF
MUL CL,A
ADD CL,AL
JMP LIRE
SUITE1:
PUSH CL
INC BL
MOV CL,0
CMP BL,2
JNZ LIRE
FIN1:
POP AL
POP BL
PUSH DL
RET

;;; 2 - CALCUL PGCD
ORG 50
POP DL
PGCD:
CMP AL,BL
JZ SUITE2
JNS SI_ANOTINFB
SI_AINFB:
SUB BL,AL
JMP PGCD
SI_ANOTINFB:
SUB AL,BL
```

```
        JMP PGCD
SUITE2:
    PUSH DL
    RET

;;; 3 - AFFICHAGE RESULTAT
    ORG 70
    POP DL
SUITE3:
    MOV BL,0
    PUSH BL

AFFICH:
    PUSH AL
    POP BL
    MOD AL,A
    OR AL,30
    PUSH AL
    PUSH BL
    POP AL
    DIV AL,A
    CMP AL,0
    JNZ AFFICH
    MOV BL,C0

DEPILE:
    POP AL
    CMP AL,0
    JZ FIN2
    MOV [BL],AL
    INC BL
    JMP DEPILE

FIN2:
    PUSH DL
    RET

; Appel des sous programmes
    ORG 0
    CALL 20
    CALL 50
    CALL 70
    END
```

**Exercice 5-4**

Transformez le programme de calcul du PGCD écrit ci-avant de façon à ce que les paramètres soient passés par la pile.

Essai avec 5 et 25 :

The screenshot shows an 8086 assembler interface. The main window displays assembly code with comments. A small window titled 'VDU C0..FF' shows the value '5'. A larger window titled 'RAM Source Code View' displays a memory dump with hexadecimal addresses and source code.

**Assembly Code:**

```

AL 00000000 00 +000 IP 00000110 06 +006
BL 11000001 C1 -063 SP 10111111 BF -065
CL 00000000 00 +000 SR 00000010 02 +002
DL 00000110 06 +006      ISOZ

ORG 20
POP DL

;;; 1 - LECTURE
LIRE:
    IN 0
    CMP AL,0D
    JZ SUITE1
    AND AL,CF
    MUL CL,A
    ADD CL,AL
    JMP LIRE

SUITE1:
    PUSH CL
    INC BL
    MOV CL,0
    CMP BL,2
    JNZ LIRE

FIN1:
    PUSH DL
    RET

;;; 2 - CALCUL PGCD
ORG 50
POP DL
POP AL
POP BL

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

**RAM Source Code View:**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	CALI20	CALI50	CALI70	END	END	END	END	END	END	END	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	POP DL	IN 0	CMP AL	0D	JZ	SUITI	AND AL	CF	MUL CL	A	ADD					
30	CL	AL	JMP	LIRE	PUSFCL	INC BL	MOV CL	0	CMP BL	2	JNZ	LIRE				
40	PUSFDL	RET	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	POP DL	POP AL	POP BL	CMP AL	BL	JZ	SUITI	JNS	SI_A	SUB	BL	AL				
60	JMP	PGCISUB	AL	BL	JMP	PGCIPUSHAL	PUSFDL	RET	END	END	END	END				
70	POP DL	POP AL	MOV BL	0	PUSHBL	PUSHAL	POP BL	MOD	AL	A						
80	OR	AL	30	PUSHAL	PUSHBL	POP AL	DIV	AL	A	CMP	AL	0	JNZ			
90	AFFI	MOV	BL	C0	POP	AL	CMP	AL	0	JZ	FIN2	MOV	[BL]	AL	INC	BL
A0	JMP	DEPI	PUSFDL	RET	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	05	35	06
C0	35															
D0																
E0																
F0																

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

Code:

```

ORG 20
POP DL
  
```

```

;;; 1 - LECTURE
LIRE:
    IN 0
    CMP AL,0D
    JZ SUITE1
  
```

```
    AND AL,CF
    MUL CL,A
    ADD CL,AL
    JMP LIRE
SUITE1:
    PUSH CL
    INC BL
    MOV CL,0
    CMP BL,2
    JNZ LIRE
FIN1:
    PUSH DL
    RET

;;; 2 - CALCUL PGCD
    ORG 50
    POP DL
    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:
    PUSH AL
    PUSH DL
    RET

;;; 3 - AFFICHAGE RESULTAT
    ORG 70
    POP DL
SUITE3:
    POP AL
    MOV BL,0
    PUSH BL

AFFICH:
    PUSH AL
    POP BL
    MOD AL,A
    OR AL,30
    PUSH AL
    PUSH BL
    POP AL
    DIV AL,A
    CMP AL,0
    JNZ AFFICH
    MOV BL,C0

DEPILE:
    POP AL
    CMP AL,0
    JZ FIN2
```



```
MOV [BL],AL
INC BL
JMP DEPILE
```

```
FIN2:
PUSH DL
RET
```

```
; Appel des sous programmes
ORG 0
CALL 20
CALL 50
CALL 70
END
```

### **Exercice 5-5 (si le temps le permet...)**

Mettez en oeuvre le programme de tri à bulles avec sous-programme de saisie des caractères à trier (cf Exercice 5-3 du TD).

Code :

```
;;; 1 - LECTURE
```

```
ORG 20
POP DL
MOV BL,C0
LIRE:
IN 0
CMP AL,0D
JZ FIN1
MOV [BL],AL
INC BL
JMP LIRE
FIN1:
MOV AL,0
MOV [BL],AL
PUSH DL
RET
```

```
;;; 2 - Calcul du dernier indice
```

```
ORG 40
POP DL
MOV AL,C0
DERNIER_INDICE:
MOV BL,[AL]
CMP BL,0
JZ FIN2
INC AL
JMP DERNIER_INDICE
FIN2:
DEC AL ; de 0 a TAILLE-1
PUSH AL ; parametre avec la pile
PUSH DL
RET
```

---

;;; 3 - tri a bulle

```
ORG 60
POP DL
POP AL
TRI_BULLES:
CMP AL,C0
JZ FIN3
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

FIN3:
    PUSH DL
    ;RET
```

```
; Appel des sous programmes
ORG 0
CALL 20
CALL 40
CALL 60
END
```