TD n° 1 :

Ex n° 1 :

Soit un processeur à 8 bits à accumulateur de Von Neuman et un espace adressable de 64 KO.

Indiquez : 1/ la taille du M.C bus de données
2/ " " d'adresse
3/ " des registres du processeur.
4/ si un pgme de taille > 64 KO se presente pour sa exécuté ? il être exécuté ? expliquez.

Solution :

1/ taille du bus de données : 8 lignes
2/ " du bus d'adress :

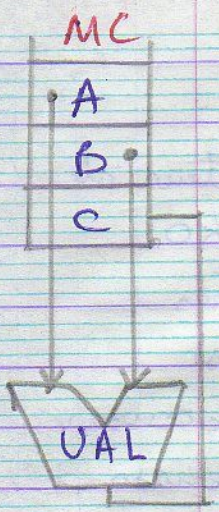
nb de mots mémoire = $\frac{\text{taille de M.C}}{\text{taille du mot}}$

$= \frac{64 \text{ Kilo octet}}{8 \text{ bits} = 1 \text{ octet}} = 64 \text{ K} = 2^6 \cdot 2^{10}$

$= 2^{16}$ mots mémoire

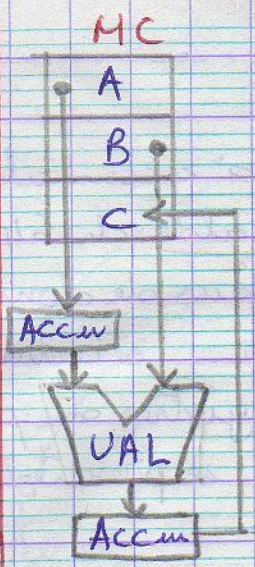
⇒ taille de l'adresse = 16 bits
⇒ taille du bus d'adresse = 16 lignes.
3/ taille des registres = 16 bits
4/ Oui, il peut être exécuté par parties.

Ex n° 2 : Modeles d'exécution.



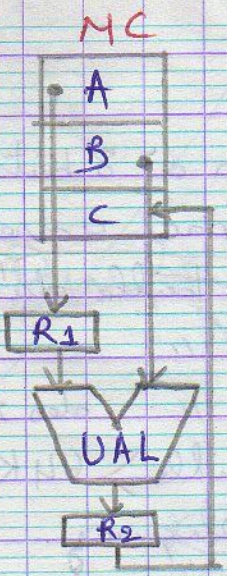
(a)

mémoire - mémoire



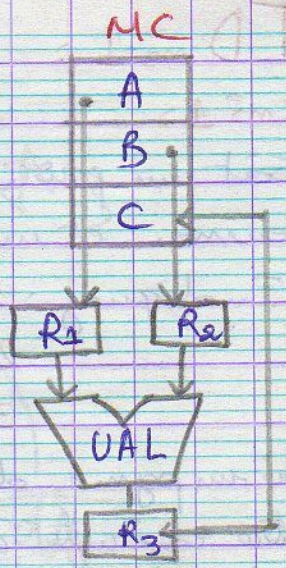
(b)

Accumulateur



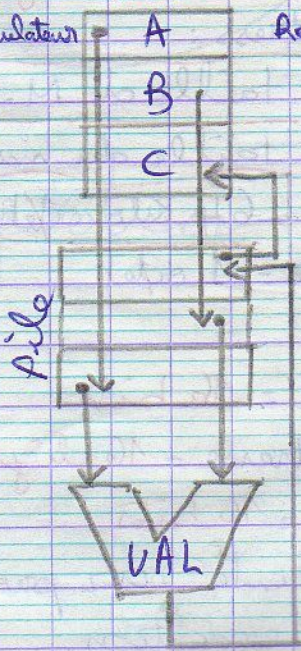
(c)

mémoire - registre



(d)

Registre - Registre



pile - pile

(e)

1/ citez le type d'inst par chaque modèle.

2/ Ecrire le pgme en pseudo-code de l'inst C=A+B.

pseudo-~~code~~ instruction:

ABD Addition

LD chargement d'éperande.

ST Rangement

PUSH Empiler

pop Dépiler.

2/ Les programme en pseudo-code:

a/ ADD @A, @B, @C couple (n, m) = (3,3)
↑ tout les operand ↑ les operand mémoire

b/ LD @A
ADD @B (1, 1)
ST @C

c/ LD R_1, @A
ADD @B, R_1, R_2 (3, 1)
ST R_2, @C.

d) LD R_1, @A
LD R_2, @B
ADD R_1, R_2, R_3 (3, 0)
St R_3, @C.

(e) pUSH @ A
pUSH @ B
ADD

TD n°2

Exercice n° 1:

Question : Etablir les schemas d'exécution des inst

etiq : ble R0, 12, etiq1.

ADD

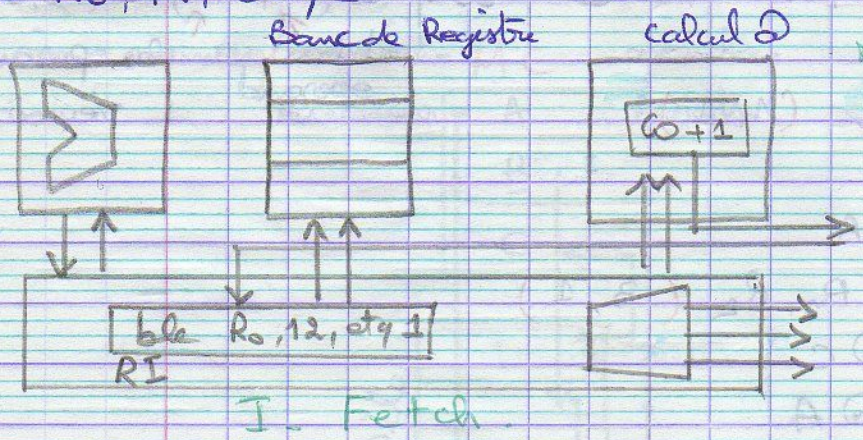
.....

J etiq

etiq1 :

Solution :

ble R0, 12, etiq1



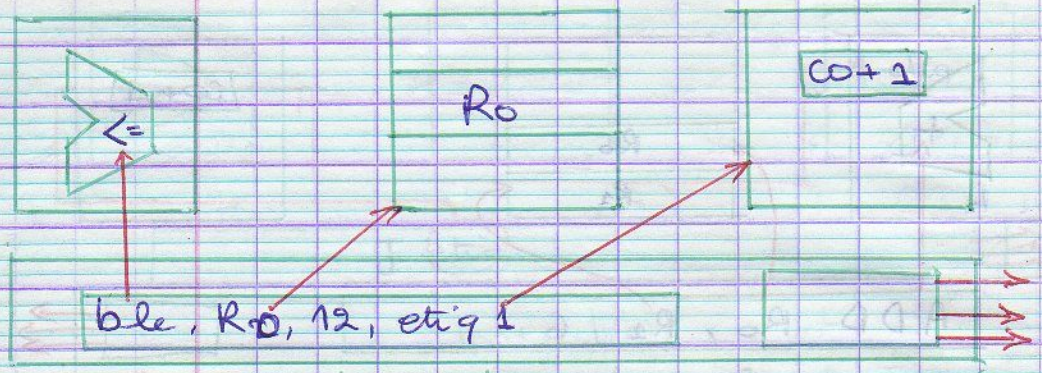
I - Fetch

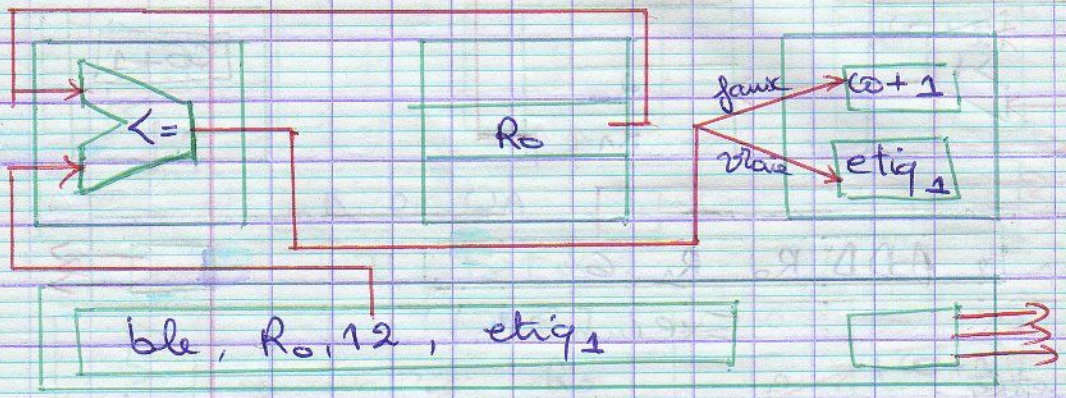
* ble R0, 12, etiq1

I - Fetch

Decode

Executé

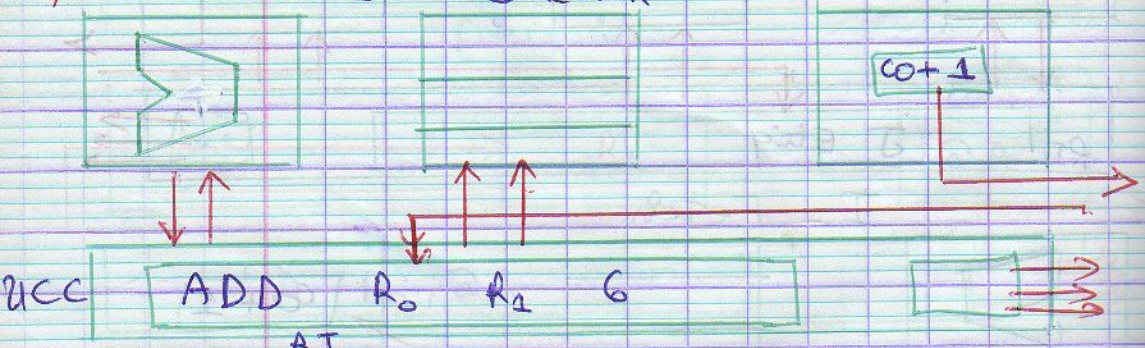


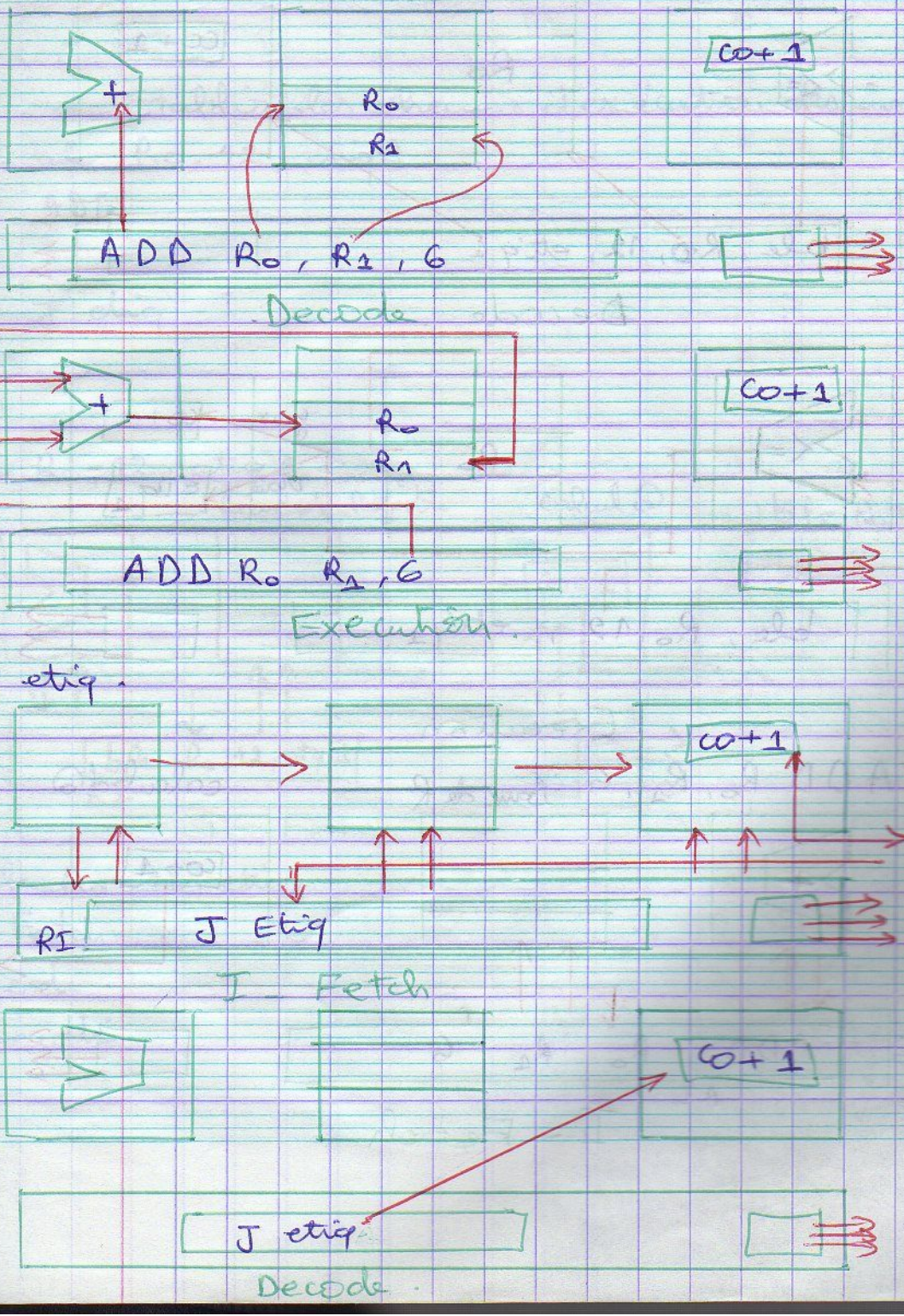


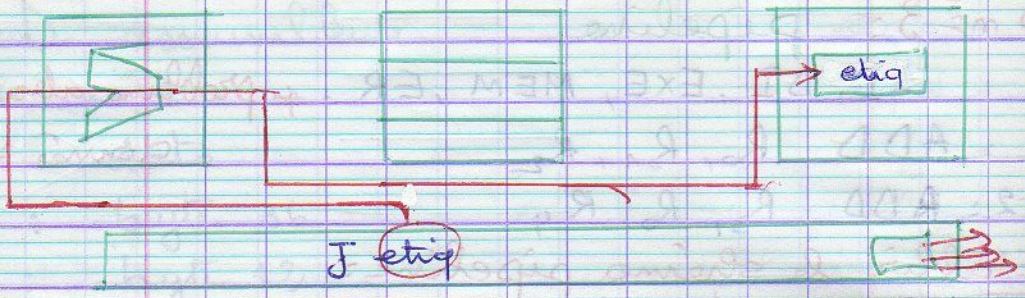
Exécution

2/ ADD R_0, R_1, 6 Banc de R

calcul d②



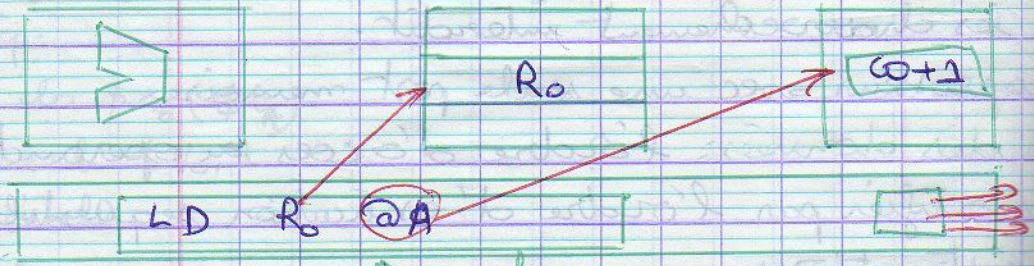

+
R0
R1
CO+1
ADD R0, R1, 6
Decode
+
R0
R1
CO+1
ADD R0 R1, 6
Execution
3/ J etiq.
CO+1
RI
J Etiq
I Fetch
CO+1
J etiq
Decode



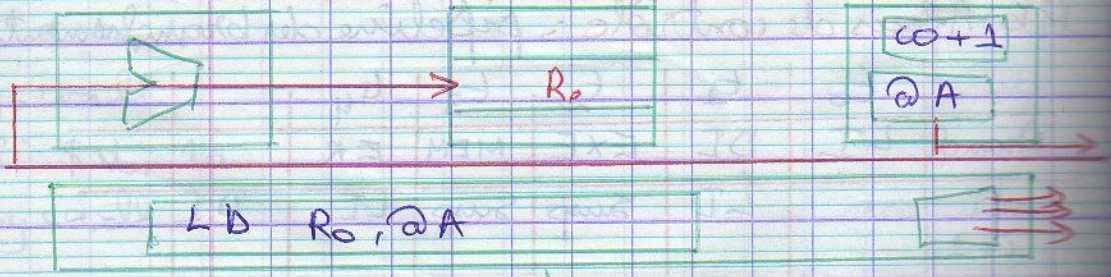
Execution.

* LD R_0, @A.

I. Fetch



Decode



Execution.

TD n° 3 : pipeline

LI, DI, EXE, MEM, ER.

* problème aléas de données :

inst1 : ADD R_0, R_1, R_2

inst2 : ADD R_3, R_0, R_4

* Etablir le schéma pipeline :

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	
inst 1	LI	DI	EXE	MEM	ER				
inst 2	-	LI	DI	suspendu		EXE	MEM	ER	

Aléa de données : matériel ne peut gérer tte les combinaison de chevauchement possible Mem, mem au même temps.

ex : les chevauchement interdit.

microprocesseur avec une seule port mémoire

Aléa des données : l'ordre d'accès au operande est modifier par l'ordre d'éxécution sequentille des instructions.

Pb Aleas de contrôle : pipeline des branchements :

	t_0	t_1	t_2	t_3	t_4		
Branch	LI	DI	EXE	MEM	ER		
instr $i+1$		LI	Susp	Susp	LI	DI	...

(Inst qui modifient le co)

T.D assemblleur

Exercice n° 1 :

1/ .data

A : .byte 36

B : .byte 12

mess : .asciiz " : est le max "

Les .text

lb $t0, A

lb $t1, B

BLe $t0, $t1 etiq

lbi $a0, ($t0)

J : fin

etiq :

la $a0 ($t1)

fin : li $V0, 1

syscall

lbi $a0, mess

li $V0, 4

syscall

li $V0, 10

syscall

byte = 0 ... 256 (بيت)

word = 2^{16-1} = 65536

2/ .data
A: .byte
B: .byte
mess: .Asciiz "est le max"
ch: .Asciiz "Donner deux nbr"
.text
la $a0, ch
li $V0, 4
Syscall
li $V0, 5
Syscall
Sb $V0, A
li $V0, 5
syscall
Sb $V0, B
Lb $t0, A
lb $t1, B
BLe $t0, $t1 etiq
la $a0, ($t0)
J: fin
etiq = la $a0 ($t1)
fin: li $V0, 1
syscall
la

Exercice n°2 :

1/
```
.data
ch : .Asciiz "Bonne année"
N : .word 2n10
.text
la $a0, ch
li $V0, 4
Syscall
lw $a0, N
li $V0, 1
syscall
li $V0, 10
syscall
```

2/
```
.data
ch : .asciiz " "
N : .word
mess1 : .asciiz "Donner un nb"
mess2 : .asciiz "Donner un chaine"
.Text
main : li $V0, 4
la $a0, mess1
syscall
```

```
li $V0, 5
syscall
sw $V0, N
li $V0, 4
la $a0, mess2
syscall
li $V0, 8
la $a0, ch
syscall
↓
li $V0, 4
la $a0, ch
syscall
li $V0, 1
lw $a0, N
syscall
li $V0, 10
syscall.
```

Exercice n°3 :

```
.Data
msg : .asciiz
.text
debut: li $t0, 10 # somme
       addi $t1, $t0, -1 # compte
boop : addu $t0, $t0, $t1
       subui $t1, $t1, 1
       BNEZ $t1, boop
       la $a0, msg
       li $V0, 4
       syscall
       move $a0, $t0
       li $V0, 1
       syscall
       li $V0, 10
       syscall.
```

2ème Méthode :

```
.data
mess: asciiz "somme"
.text
```

Sp : indique l'adresse d'un 1 ou dernier elt de la pile

```
main : sw $0, 32($sp) # somme
       sw $0, 44($sp) # indice i
$32 :  Lw $14, 32($sp)
etiq   Lw $15, 44($sp)
       Addu $24, $14, $15 # S := S + i
       Sw $24, 32($sp)
       Lw $25, 44($sp)
       Addui $8, $25, 1
       Sw $8, 44($sp)
①
contenu $8 < 10  BLT $8, 10, $32
       li $v0, 4
       la $a0, mess
       Syscall
       li $v0, 1
       la $a0, ($24)
       syscall
       li $v0, 10
       syscall.
```

Exercice n° 4 :

```
    .data
ch: .asciiz "Bonjour"
mess: .asciiz "Résultat = "
```

.text sauvegardé @ 2 Registre $31

Jal main procedure

Addui $0, $0, 0

Addui $V0, $0, 10

syscall

globl

main :

la $a0, ch

move $t0, $a0

lecture : lb $t1, ($a0) # un carc

BEZ $t1, fin

Addui $a0, $a0, 1

J lecture

fin : subu $t0, $a0, $t0

SUBui $t0, $t0, 1 # Nb de caract

li $V0, 4

la $a0, mess

syscall

li $V0, 1

move $a0, $t0

syscall

Exercice 5 :

.data

ch_1 : .asciiz "Bonjour année "

ch_2 : .asciiz " "

C : .asciiz " "

.text

Jal main

Addui $0, $0, 0

Addui $V0, $t0, 10

syscall

globl

main :

la $a0, ch_1

move $t0, $a0 # t0 indice de ch_1

la $a0, ch_2

move $t1, $a0 # t1 indice de ch_2

la $a0, C

Lb $t3, ($a0)

lecture : Lb $t2, ($t0)

Bnez $t2, fin

Beq $t2, $t3 etiq

sb $t2, ($t1)

Addui $t1, $t1, 1

```
etiq : Addui $t0, $t0, 1
       J lecture
fin
       li $v0, 4
       la $a0, ch1
       syscall
       la $a0, ch2
       syscall
```

Exercice n°6 :

```
    .Data
ch1 : .asciiz "Bonjour"
ch2 : .asciiz "     "
    .text
    Jal main
Addui $0, $0, 0
Addui $v0, $0, 10
syscall
main:
    la $a0, ch1
    la $a1, ch2
lecture : lb $t0, ($a0)
          beq $t0, fin
```

Sb $t0, ($a1)
Addui $a0, $a0, 1
Addui $a1, $a1, 1
J lecture
fin : li $V0, 4
la $a0, ch1
syscall
la $a0, ch2
syscall.

Exercice n° 7 :
Ecrire un pgme qui evalue l'espression : $5x^2 - 12xy + 6y^2$ utilisant les fcts pour n'importe quelles valeurs de x et y

Les fonctions :
lecteur nbr : li $V0, 5
syscall
la $a0, ($V0)
Jr $31.

sortie $a0.
Add MON = Addu $V0, $a0, $a1
Jr $31
Mulitiplication : Mul $V0, $a1, $a0
Jr $31
entrée $a0, $a1
sortie $V0.

soustraction SUBU V_0, a_0, a_1

Jr $31

Afficher_nb: li V_0, 1 entrée a_0

syscall

Jr $31.

Afficher_ch: li V_0, 4

syscall

Jr $31.

-entrée a_0.

.data

x: .word

y: .word

ch: .asciiz "Resultat="

mess: .asciiz "Entrer un nb:"

.text

Jal main

fin: Addui $0, $0, 0.

Addui V_0, $0, 10

syscall

.globle main

main:

la a_0, mess

```
Jal affich_ch
Jal lecture_nb
sw $a0, x
la $a0, mess
Jal Affich_ch
Jal lecture_nb
 SN $a0, y
lw $a0, x
 move $a1, $a0
Jal multiplication # x²
  li $a0, 5
  move $a1, $Vo
Jal multiplication # 5x²
 move $to, $Vo, # to = 5x²
 lw $a0, x
 lw $a1, y
Jal multiplication # x×y
li $a0, 12
move $a1, $Vo
Jal multiplication # 12·x·y
move $a0, $to # 5x²
move $a1, $Vo # 12x·y
```

```
Jal soustraction # 5x²-12x.y
move $t0, $V0 # t0 = 5x²-12xy
lw $a0, y
move $a1, $a0
Jal multiplication # y²
move $a0, $V0.
li $a1, 6.
Jal multiplication
 move $a01, $t0
 move $a1, $V0
Jal Addcmb # 5x²-12xy+6y²
 la $a0, ch
 Jal Affiche-ch
move $a0, $V0.
 Jal afficher-nb
  J: fin
Afiche-ch
```

Ex n°8 :

fct copier de chaine

La fct copie :

```
lecture : lb $t0, ($a0)
          beq $t0, fin
          sb $t0, ($a1)
          Addui $a0, $a0, 1
          Addui $a1, $a1, 1
            J lecture
```

entrée $a0, $a1 fin : Jr $31

fct compteur :

```
compteur : li $V0, 0
  lecture : lb $t0, ($a0)
            beq $t0, fin
            Addui $V0, $V0, 1
            Addui $a0, $a0, 1
              J lecture
  fin : Jr $31
```

entrée : $a0

sortie : $V0

Exercice n=2 : 2009

2/ .data

ch : .Asciiz "chaine de caractère"

a : .Asci "a"

e : .Asci "e"

o : .Asci "o"

Tab : .word

.text

la $a0, a

lb $t0, ($a0)

la $a0, e

lb $t1, ($a0)

la $a0, o

lb $t2, ($a0)

li $t3, 0

la $a0, ch

boucle : lb $V0, ($a0)

beq $V0, fin

beq $V0, $t0 etiq

beq $V0, $t1, etiq

beq $V0, $t2, etiq

sw $V0, tab ($t3)

addi $t3, $t3, 4

etiq : add $a0, $a0, 1

J. boucle

Exercice 3 : 2/ position :

```
        .Data
ch :    .Asciiz "chaine de caractère"
str :   .Asciiz "                    "
        .text
        li $t0, 0 # position ch.
        la $a0, ch
        la $a1, str
boucle : lb $v0, ($a0)
        beq $v0, fin
        li $t1, 0 # position str.
boucle2 : bgt $t1, $t0, etiq
        sb $v0, ($a1)
        Adolui $t1, $t1, 1
        Adolui $a1, $a2, 1
        J. boucle2
etiq :  Adolui $a0, $a0, 1
        Adolui $t0, $t0, 1
        J. boucle
```

décalage à gauche x 2
" à droite ÷ 2

Ex n°4 : le masc (pile) :

```
.text
sw $0, 30($sp) # inchè
sw $0, 50($sp) # masc.
lecture: lw $t1, 30($sp)
         bgt $t1, 9, fin
         li $v0, 5
         syscall
         lw $t0, 50($sp)
         blt $v0, $t0, etiq
         sw $v0, 50($sp)
etiq: Addui $t1, $t1, 1
      sw $t1, 30($sp)
      J. lecture
```