Serie n° 01:

Exercice 1:

Soit un processeur à 8 bits accumulateur de von Neumon est un espace adréssable de 64 Ko ; Indiquez:

1°/ La taille du bus de données.

2°/ " " d'adresse

3°/ " des registres de prosseceur

4°/ Si un programme de taille > 64 Ko se présente puisse t-il être exécuter ? expliquez !!

* 1- La taille de bus de données : 8 bus Lignes.

2- La taille de bus d'adresse est :

nbre de mots mémoire = $\frac{\text{taille de la M.C}}{\text{taille de mot}} = \frac{64\,Ko}{1\,octet}$

$= 64\,K = 2^6 \cdot 2^{10} = 2^{16}$

⇒ 12^{16} mots mémoire donc la taille d'adresse : 16 bits.

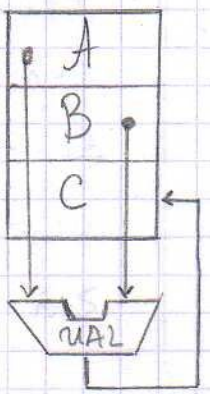
⇒ La taille de bus d'adresse = 16 Lignes.

3- tailles de registres = 16 bits.

4- Oui ; il peut être exécuter en partie.

Exercice 2:

* Les Modèle d'éxécution :



mémoire - mémoire

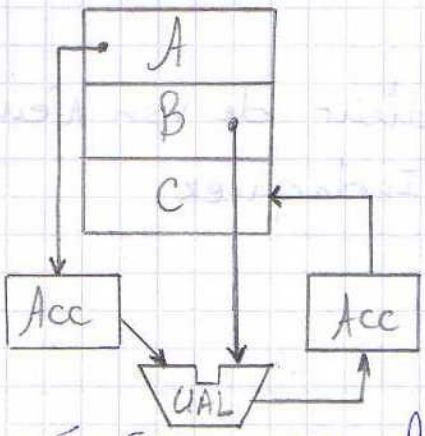
Add @A, @B, @C

Couple $(n, m) = (3, 3)$

n : tous les opérands

m : operand adresse mémoire.

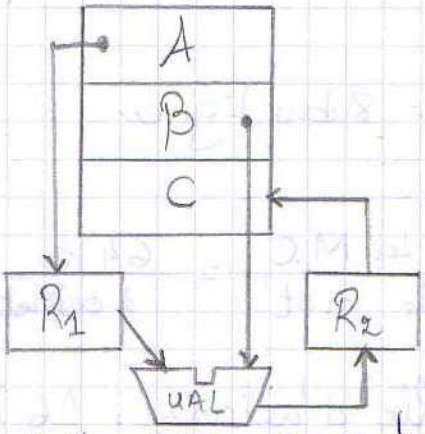
b°/



mémoire - accumulateur

```
LD @A
Add @B
ST @C
   couple (1,1)
```

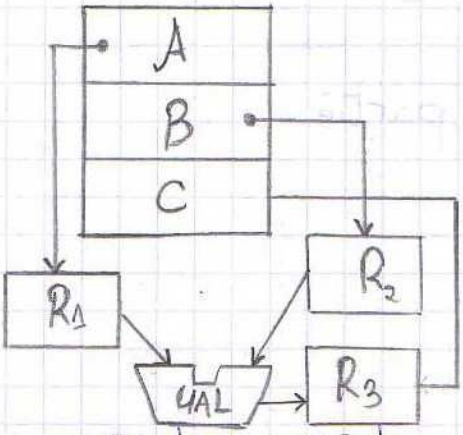
c°/



mémoire - registre

```
LD R1, @A
Add @B, R1, R2
ST R2, @C
   couple (3,1)
```

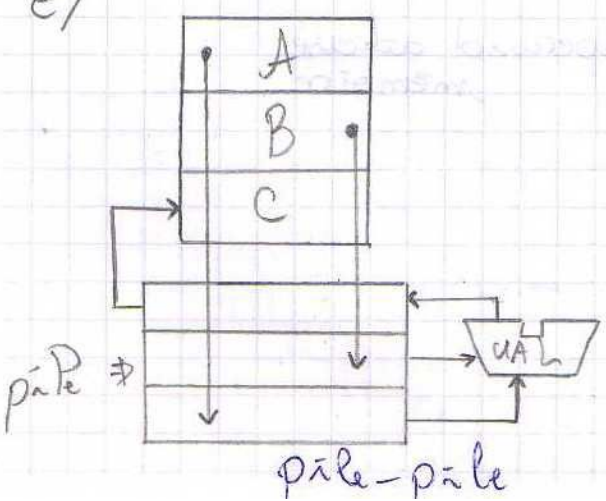
d°/



registre - registre

```
LD R1, @A
LD R2, @B
Add R1, R2, R3
ST R3, @C
   couple (3,0)
```

e°/



pile - pile

```
Push @A
Push @B
Add
Pop @C
   couple (0,0)
```

TD 03 : Pipeline :

1/ Le principe de pipeline est celui de la machine de montage ; Fonctionner la tache en sous taches du duré égale appellée étage.

- Exécuter simultanement les differents taches de plusieurs taches. Plus généralement un système pipeline est caractérisé par 2 paramétres T : la duré individuelle de l'étage et N : le nombre d'étages de pipeline pour mieux a préciser le principe de pipeline.

On considére les différents étapes d'exécution d'une instruction Risc qui sont LI, DI, EX, MEM, ER

La fois le pipeline a morcée = 5 cycles de porte = 1 cycle par instruction
5 étages

1/ Aléas structurelles :

Se probléme lorsqu'on a un materiel qui ne peut pas gérer tout les combinaisons de cheranchement possible.

Exemple : Micro-processeur qui ne comporte qu'un seul porte mémoire.

	1	2	3	4	5	6	7
inst i	Li	Di	Ex	MEM	ER	-	-
i+1		Li	Di	Ex	MEM	ER	-
i+2				susp	susp	susp	Li
i+3							
i+4							
i+5							

susp : suspendre
MEM : stocket les résultats dans la mémoire
ER : " dans les registres.

2 : Aléas de données :

Se probléme se pose lorsque l'acces au opérand est modifié par l'ordre séquentiel des instructions.

Exemple : Add R_1, R_2, R_3
Sub B_3, R_4, R_1
⋮

Num Inst	1	2	3	4	5	
Add	Li	Di	Ex	MEM	ER	← donnée écrite dans R_1
Sub		Li	Di	susp susp	Ex	MEM

susp ← susp : donnée liée : R1 inconnu !

3. Aléas de contrôles :

Le problème se pose avec les instructions de branchement c-à-d les instructions qui modifient les valeurs du contenu ordinal ;

Exemple : beq R_0, R_1, ebq
load R_2, 12
Add R_0, R_0, R_1
ebq

	1	2	3	4	5	6
beq	Li	Di	Ex	MEM		
		susp	susp	susp	Li	
				valeur de Co écrite		

TD 4 : Assembleur :

Exercice 1 :

Ecrire un programme assembleur qui détermine le max entre A et B.
- A = 12 et B = 36
- A et B sont 2 variables à lire sur le clavier.

```
#.Data
A: .word 12
B: .word 36
mess: .asciiz "Le max est"
.Text
main. Lw $8, A
Lw $8, B
La $a0, mess  # $a0 ← mess
```

```
li $v0, 4
syscall          } # Afficher la chaîne
li $v0, 1
ble $8, $9, Etiq     # Si $8 <= $9
la $a0, ($8)         # a0 ← $8
j fin                # Saut vers étiquette fin
Etiq: la $a0, ($9)
fin : syscall        # Afficher l'entier
li $v0, 10
syscall              fin du programme
```

/ ~~Data~~

```
li $v0, 1
lw $a, N
syscall
li $v0, 10
syscall
```

b°/ .Data

```
ch: .Asciiz " "
N: .word
mess1: .Asciiz "Entrer une chaîne"
mess2: .Asciiz "Entrer un nombre"
.Text
li $v0, 4
la $v0, mess1
syscall
li $v0, 8
la $a0, ch       } # Saisie d'une chaîne.
```

```
syscall
li $V0, 4
la $V0, mess2
syscall
li $V0,5
syscall } # l'affichage
sw $V0, N
la $a0, ch
li $V0, 4
syscall
lw $a0, N
li $V0, 1
syscall
li $V0, 10
syscall
```

Exercice 03:

Ecrire un programme qui fait la somme des 10 premiers nombres entiers.

```
.Data
mess: .Asciiz "la somme est:"
.Text
li $t1, 0      # $t1 = la somme
li $t0, 1      # $t0 = registre compteur
Etiq1: beq $t0, 11, Etiq
Add $t1, $t0, $t1
Add $t0, $t0, 1
j Etiq1
Etiq:
```

```
Li $V0, 4
La $a0, mess
syscall
move $a0, $t1
Li $V0, 1
syscall
Li $V0, 10
syscall
```

Exercice 04:

Écrire un programme qui compte le nombre de caractère d'une chaîne

```
.Data
ch: Asciiz "bonjour"
mess: Asciiz "le nbre de caractère est:"
.Text
main: La $a0, ch
move $V0, $a0
Addiu $t0, $V0, 1
compt: Lb $t1, ($V0)   # Load bites
Addiu $V0, $V0, 1   # Incrémenter
bnez $t0, compt
Subu $V0, $V0, $t0
move $t0, $V0
Li $V0, 4
La $a0, mess   } # Affichage du mess
syscall
Li $V0, 1
La $a0, to   # Affichage du nbre de caractère
```

```
sysall
Li $V0, 10
Lsysall          # Exit
```

Exercice 05:

```
.Data
origine : .Asciz "Voici la chaîne à copier"
copie   : .Asciz "                        "
text 1  : .Asciz "chaîne à copier: "
text 2  : .Asciz "chaîne copiée "
Rchariot: .Asciz "\n"
b       : .Asci " "

.Text
Jal main
Li $V0, 10  # Exit
sysaall
.Global main
main: La $a0, copie
La $a1, origine
La $a2, b
move $t0, $a0
move $t1, $a1
move $t3, $a2
Lecture: Lb $t2, ($t1)
beq $t2, ($t3), suite
beq3 $t2, fin
sb $t2, ($t0)
Addiu $t0, $t0, 1
```

Suite : Addiu $t1, $t1, 1
j Lecture

fin : li $v0, 4
la $a0, text1
syscall
la $a0, origine
syscall
la $a0, Rchariot
syscall
la $a0, text2
syscall
la $a0, copie
syscall.

⇒ : Le programme qui lit une chaîne et supprime tous les caractéres blanc.

Exercice 06 :

Le programme qui va copier tous les éléments d'une chaîne rangée en mémoire dans une autre zone mémoire

.Data
origne : .Asciiz "Voici la chaîne à copier"
copie : .Asciiz " "
text1 : .Asciiz "chaîne à copier"
text2 : .Asciiz "chaîne copiée"
Rchariot: .Asciiz "\n"

.Text
Jal main
li $v0, 10 # Exit
syscall

```
.Global main
main:   la $a0, copie
        la $a1, origine
        move $t0, $a0
        move $t1, $a1
Debu_copy: lb $t2, ($t1)
        sb $t2, ($t0)
        Addiu $t0, $t0, 1
        Addiu $t1, $t1, 1
        bnez $t2, Debu_copy
        Li $V0, 4
        La $a0, text1
        syscall
        la $a0, origine
        syscall
        la $a0, Rchariot
        syscall
        la $a0, text2
        syscall
        la $a0, copie
        syscall
```

Résoudre l'exercice 03 Avec les piles :

```
.Data
mess: .Asciiz "le resultat est"
.Text
main = sw $a0, 33($sp)   ← stock pointeur
                           # Sommet
```

```
        sw  $0,44($sp) # 2' indice
Etiq($32):  lw  $14, 33($sp)
            addiu $14, $14, $15
            sw  $14, $33($sp)
            addi $15, $15, 1
            sw  $15, 44($sp)
            blt $15, 10, $32
            li  $v0, 4
            la  $a0, mess
            syscall
            lw  $a0, 33($sp)
            li  $v0, 1
            syscall
            li  $v0, 10
            syscall.
```