****

**Architecture des Ordinateurs**

**Mr Abdelkader CHENINE**

[**Chenine\_a@yahoo.fr**](mailto:Chenine_a@yahoo.fr)

**Vesrion : V1.0** Draft Oct**.**2021 **Université Ibn-Khaldoun Tiaret**

**Département informatique**

**2eme Année License**

**Cours: ……………**1h00

**TD : ………………**1h00

**TP : ……………….**1h00

**Coefficient :…………..** 3

**Crédit : ……………….**5 [**fmi.contenu.web@gmail.com**](mailto:fmi.contenu.web@gmail.com)

**Unité fondamentale :** UEF1

Chapitre 1 – Introduction

Chapitre 2 – Principaux composant d’un ordinateur

Chapitre 3 – Notions sur les instructions d’un ordinateur

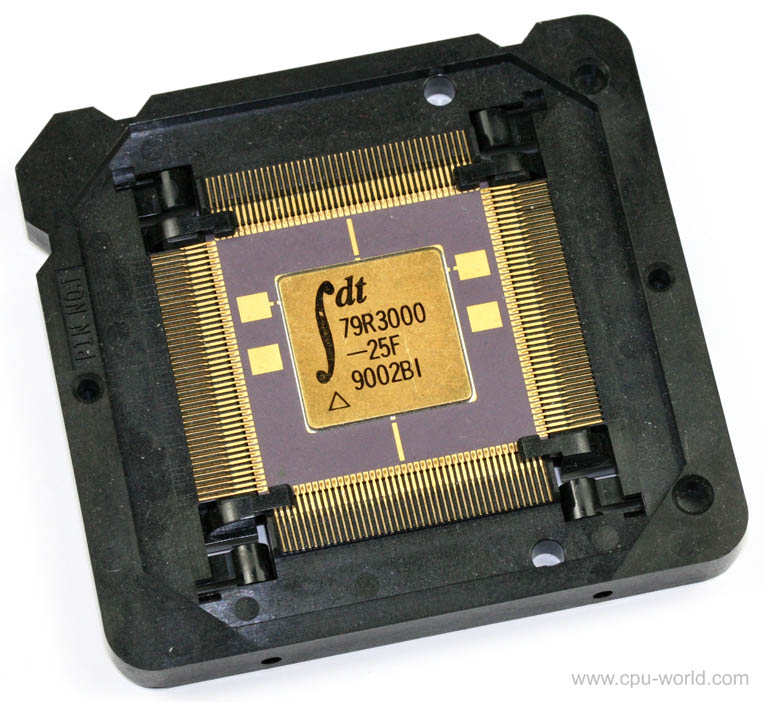
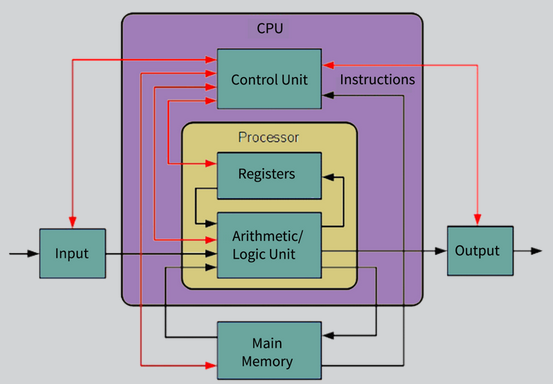
Chapitre 4 – le processeur

Chapitre 5 – instructions spéciales

**Chapitre 1 – introduction**

* 1. **Notion d’architecture des ordinateurs**

L’architecture d’un ordinateur est la description de ses unités fonctionnelles et de leurs interconnexions.



**Fig. 1**. Schéma1 d’un ordinateur

L’architecture a un impact direct sur la logique d’exécution d’un programme

* Jeu d’instructions,
* Nombre de bits pour représenter les types de données (nombres, caractères, …)
* Mécanismes d’E/S,
* Techniques d’adressage de la mémoire

x + y

Unité de contrôle

(C.U)

Unité Arithmétique et Logique (ALU)

Unité d’entrée

Unité de sortie

R1

R2

Rn

Mémoire

**Fig. 2**. Schéma2 d’un ordinateur

Problème à résoudre

Problème à résoudre

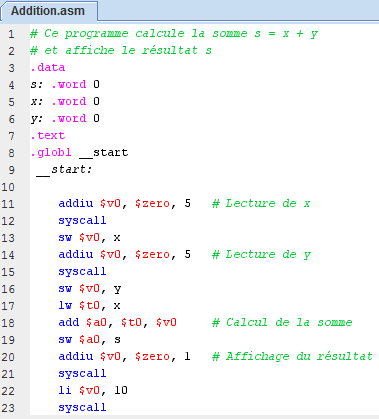
Software

Hardware

Hardware

**Fig. 3**. Les premiers ordinateurs **Fig. 4** Les nouveaux ordinateurs

**Programme = {instructions}**



1.?

Code Machine :

00100100000000100000000000000101

00000000000000000000000000001100

00111100000000010001000000000001

10101100001000100000000000000100

00100100000000100000000000000101

00000000000000000000000000001100

00111100000000010001000000000001

10101100001000100000000000001000

00111100000000010001000000000001

10001100001010000000000000000100

00000001000000100010000000100000

00111100000000010001000000000001

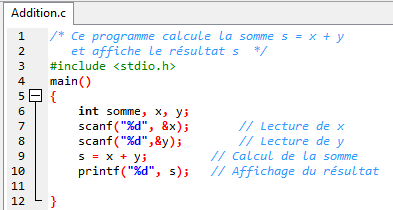
10101100001001000000000000000000

00100100000000100000000000000001

00000000000000000000000000001100

00100100000000100000000000001010

00000000000000000000000000001100



2.?

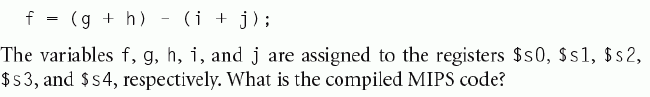
 

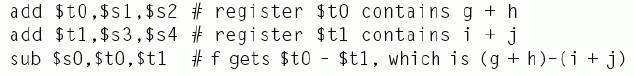


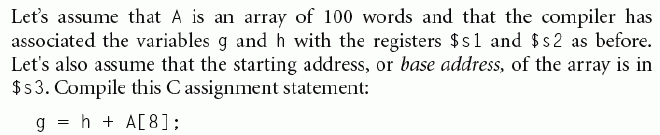


**Registres du processeur MIPS**





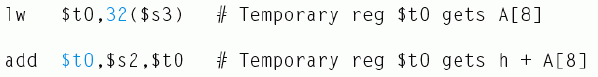










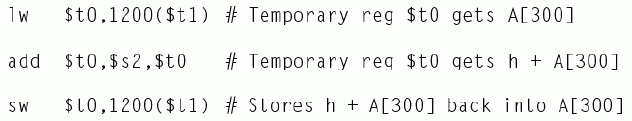


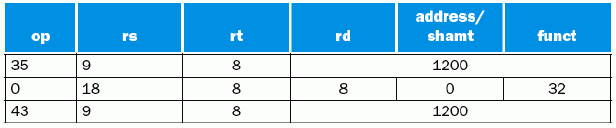


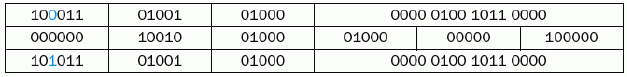


Formats

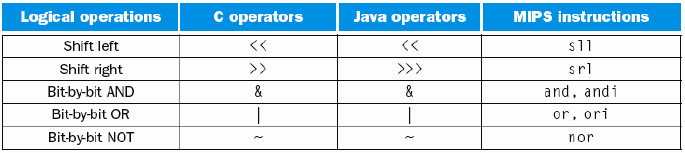








**Instructions logiques**



























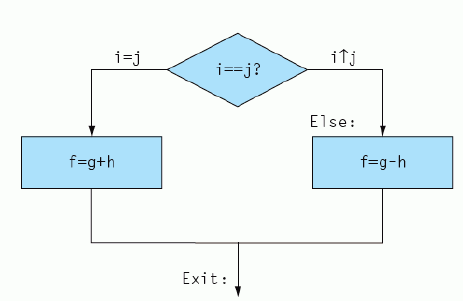
**Branchement**



















**Boucle**











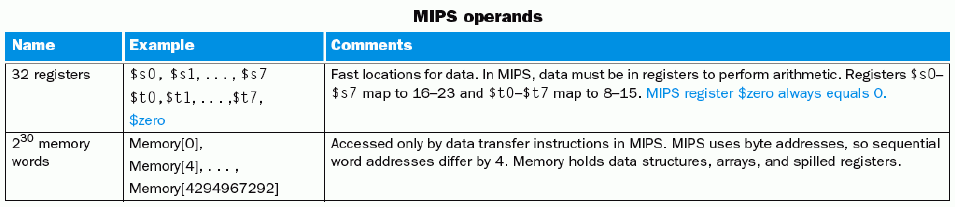


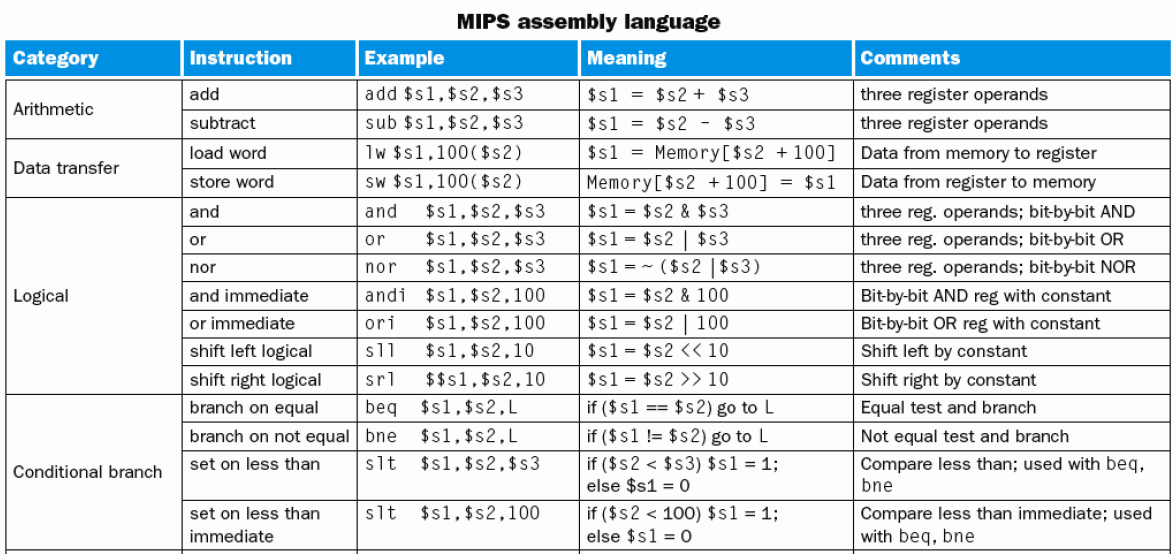


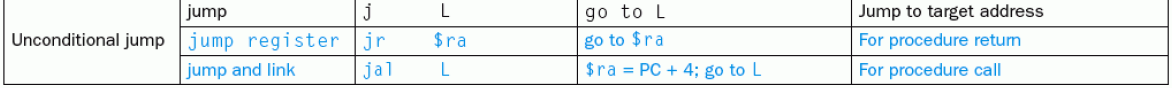
**>,>=,<,<=**

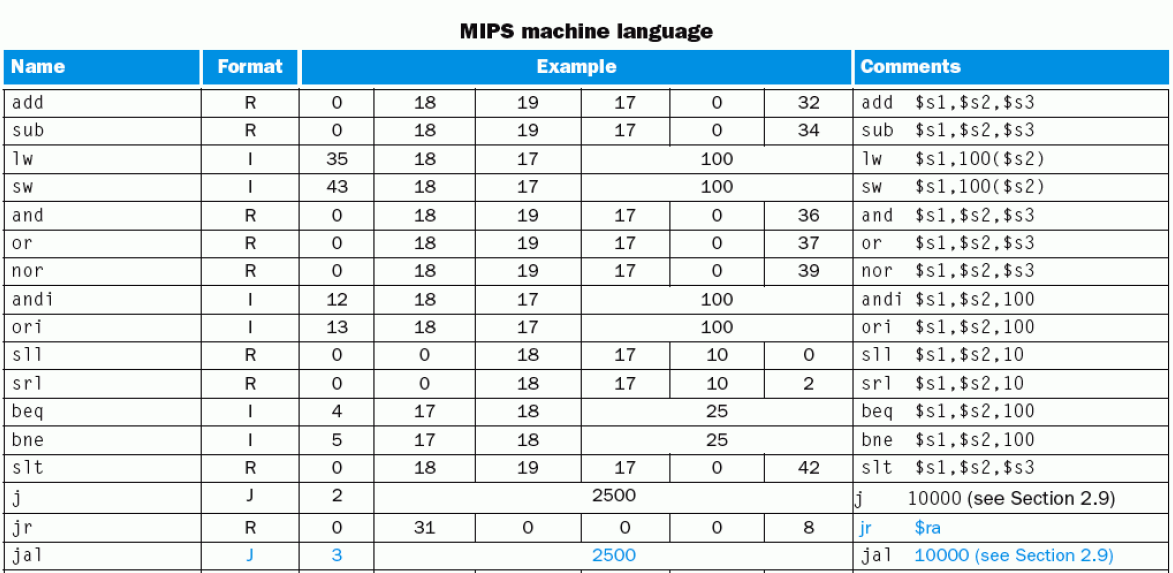




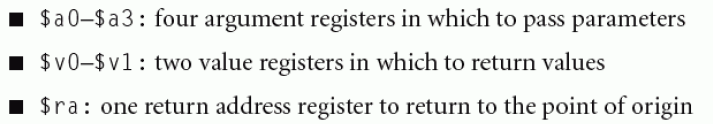


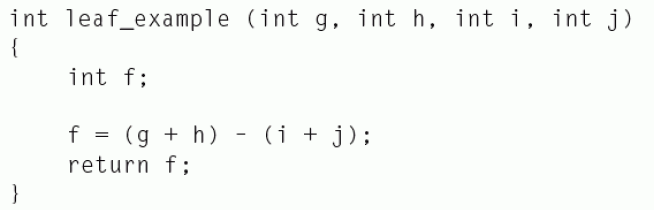




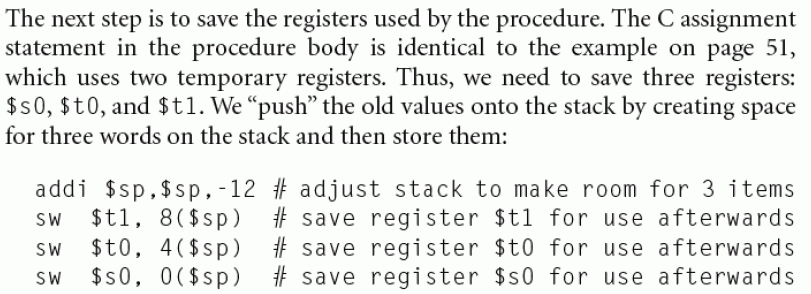


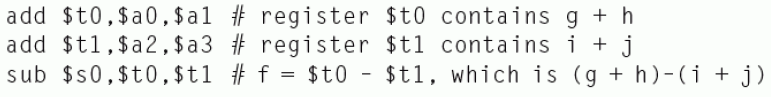
**Fonctions**

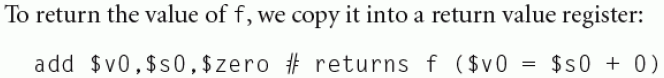


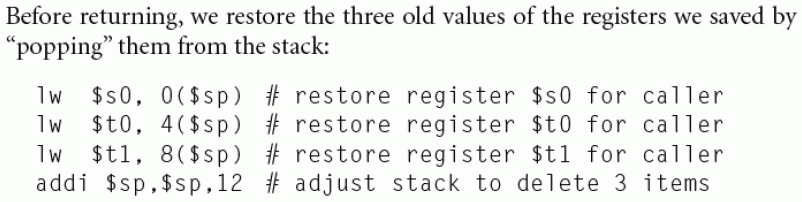


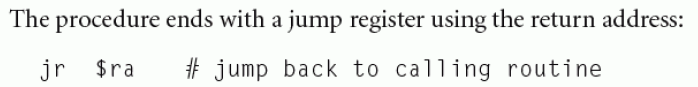


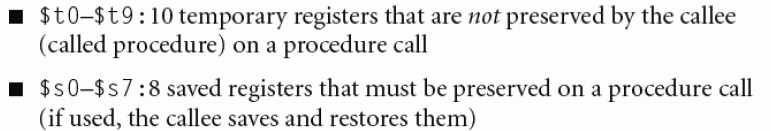


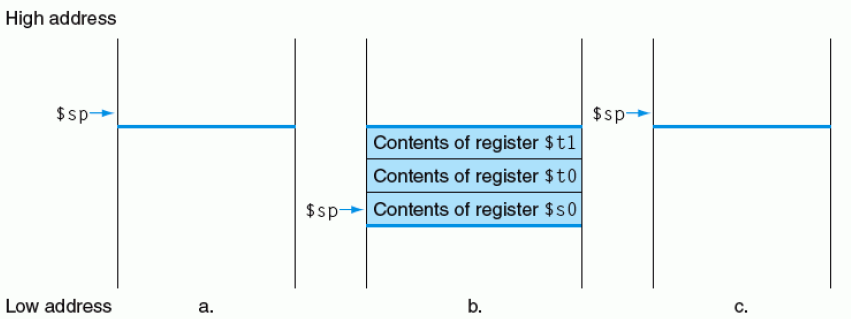


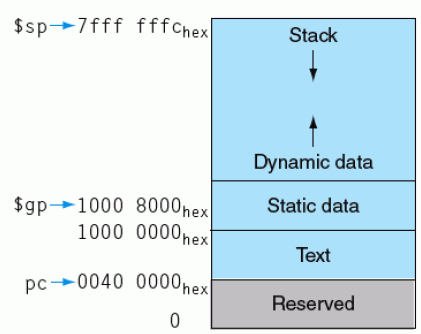


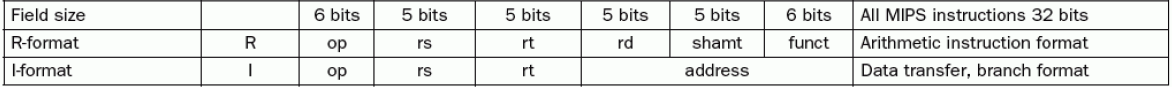




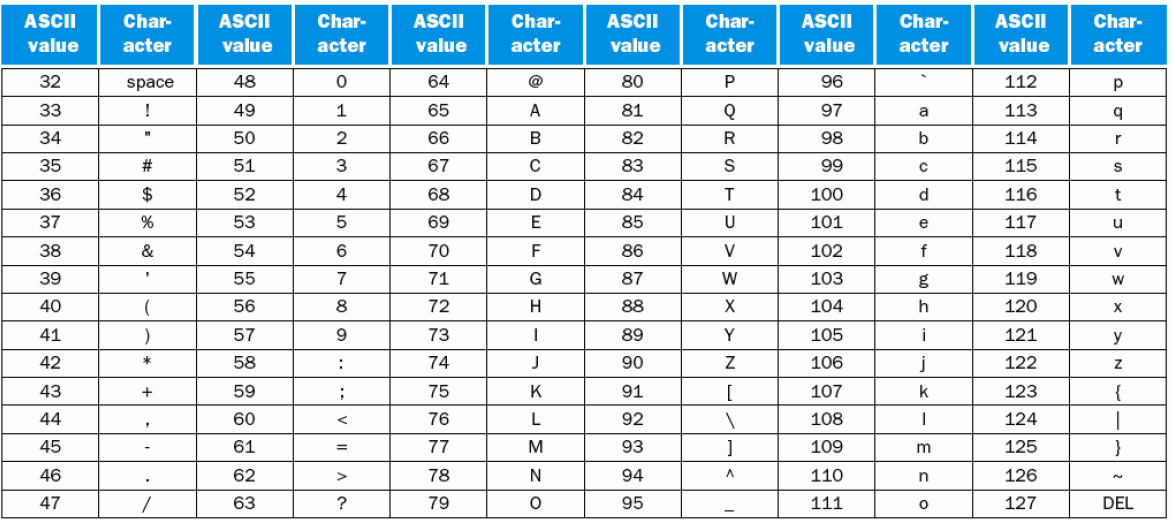




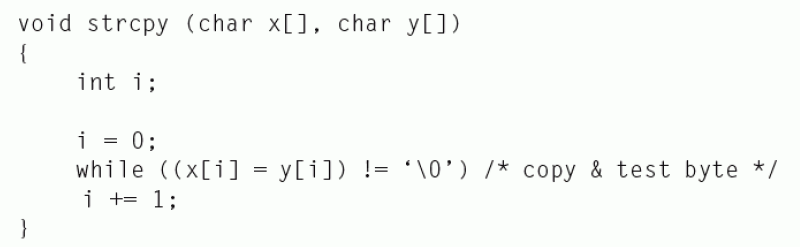


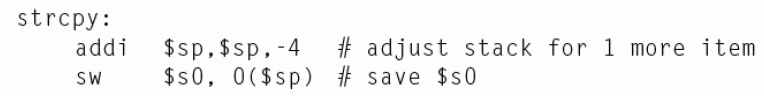














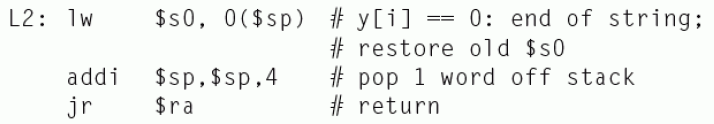




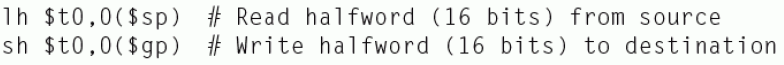








Half word





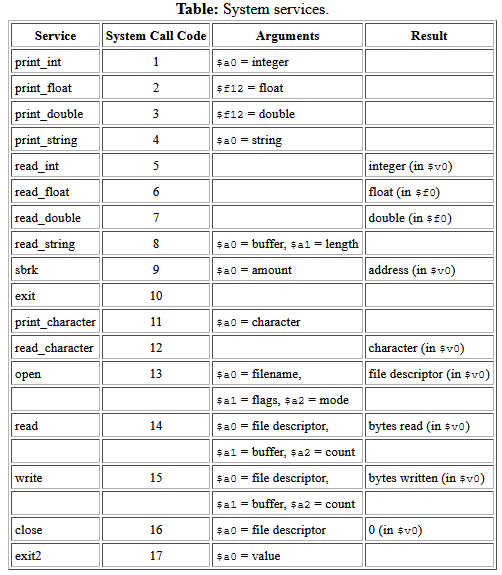


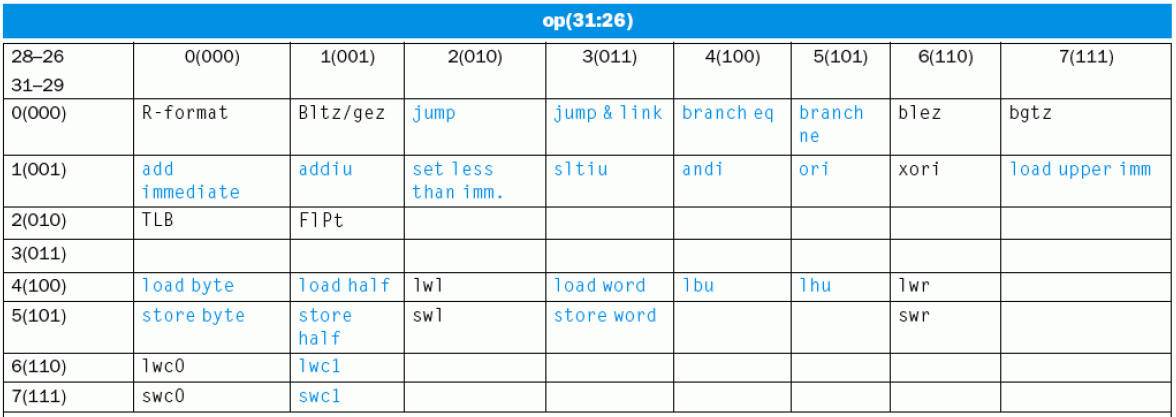


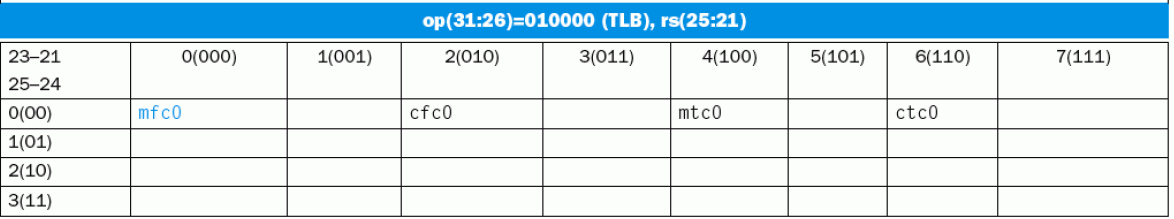


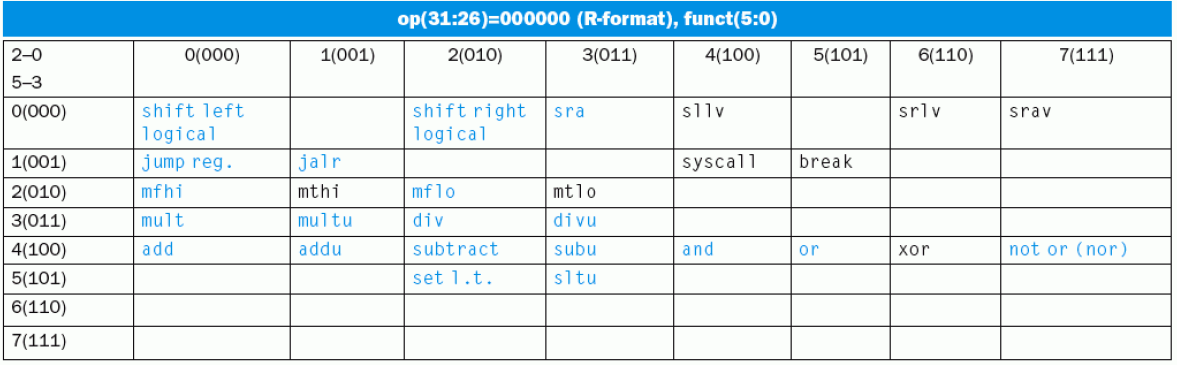


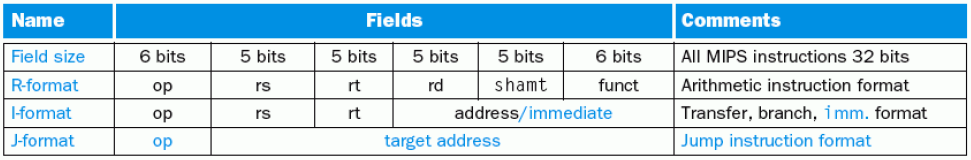












**Exemple**

.data

out\_string: .asciiz "\nHello, World!\n"

.text

li $v0, 4

la $a0, out\_string

syscall

li $v0, 10

syscall

* 1. **Architecture de Von Neumann**

L’[architecture](https://fr.wikipedia.org/wiki/Architecture_%28informatique%29) dite architecture de von Neumann est un modèle pour un [ordinateur](https://fr.wikipedia.org/wiki/Ordinateur) qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données demandées ou produites par le calcul. Ce modèle est le résultat des travaux publiés en 1946 par le mathématicien John von Neumann, il est à la base des ordinateurs actuels.

1. Processor (ALU + CU + Memory (Registers)),
2. Mémoire pour le programme & les données (MM),
3. Dispositif d’entrée/Sortie (I/O)

Programme

CPU

Processeur

Microprocesseur

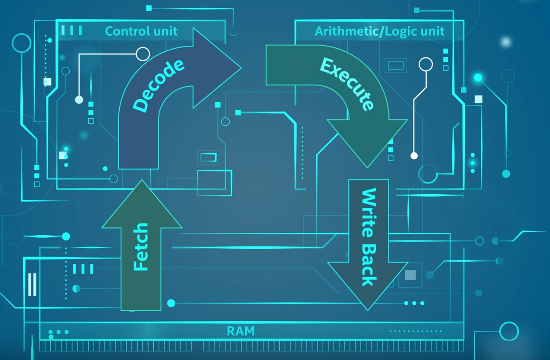
Données

Entrées/Sorties

(E/S) (I/O)

**Fig. 5**. Architecture de Von Neumann

* Un seul bus pour le transfert des instructions et des données.
* Cycle d’exécution est séquentielle



**Fig. 6**. Cycle d’exécution

* 1. **Architecture de Harvard**

Le nom de cette structure vient du nom de l’[université Harvard](https://fr.wikipedia.org/wiki/Universit%C3%A9_Harvard) où une telle architecture a été mise en pratique pour la première fois avec Le [Mark I](https://fr.wikipedia.org/wiki/Harvard_Mark_I) en [1944](https://fr.wikipedia.org/wiki/1944).

CPU

Processeur

Microprocesseur

Programme



Entrées/Sorties

(E/S) (I/O)

**Fig. 7**. Architecture de Harvard

* Les deux mémoires peuvent avoir des caractéristiques différentes,
  + Taille du mot,
  + Timing,
  + Technologie,
  + Structure de l’adresse
* Mémoires séparées l’une pour le programme et l’autre pour les données accès simultanés aux données et aux instructions.
  1. **Comparaison**

|  |  |  |
| --- | --- | --- |
|  | **Architecture de Von Neumann** | **Architecture de Harvard** |
| 1 | Mémoire partagée par les données et les instructions | Mémoires séparées pour les données et les instructions |
| 2 | Deux cycles d’horloges (un cycle pour la recherche de l’instruction et un cycle pour la recherche des données | Un seul cycle d’horloge |
| 3 | Le pipeline n’est pas possible | Le pipeline possible |
| 4 | Conception simple | Conception complexe (C.U) |
| 5 | Hardware minime | Plus de hardware |
| 6 | Moins d’espace | Plus d’espace |
| 7 | Exécution moins rapide | Exécution rapide |
| 8 | Meilleure exploitation de la mémoire | L’espace libre dans une mémoire ne peut être utilisé par l’autre mémoire |

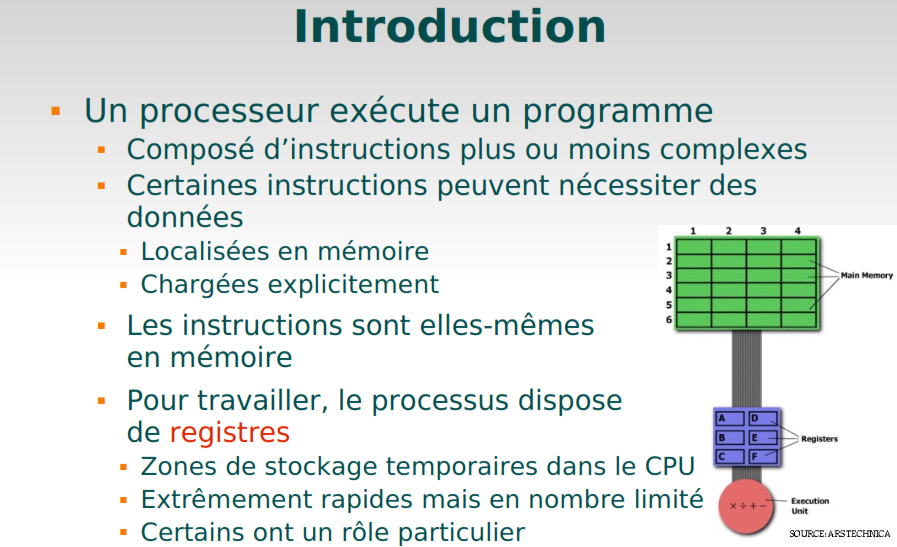
**TD0** – Représentation de l’information

**TP1** – Exécution de programme MIPS sur un simulateur (MARS/PCSPIM)

* 1. Modified harvard architecture
  2. Von Neumann’s 5 bottlenecks and CCIX

**Chapitre 2 – Principaux composant d’un ordinateur**

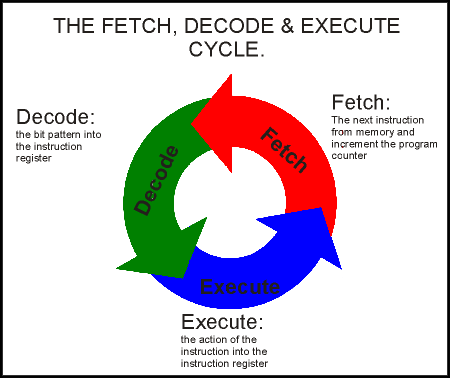
**2.1 Introduction**



(calculs en flottant, calcul d'adresses ...)

**2.2 Exécution d'une instruction**

**2.2.1 Cycle d’instruction**



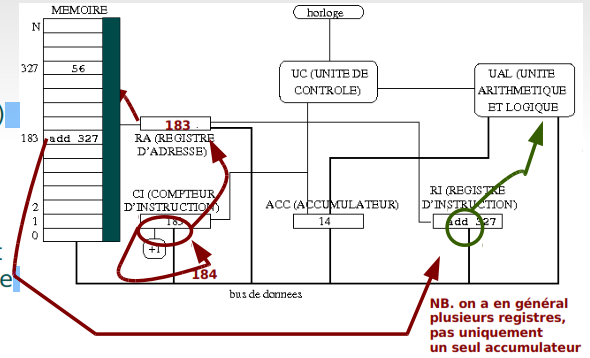
**Traiter une instruction implique les opérations suivantes :**

* Lire l’instruction en mémoire : fetch,
* Comprendre ce que fait l’instruction (addition ? multiplication ? …) : decode,
* Aller chercher les opérandes si nécessaire (en mémoire ou en registre) : read
* Exécuter l’instruction : execute,
* Éventuellement écrire les résultats dans un registre ou en mémoire : write back

**2.2.2 Cycle d'exécution d'une instruction**

À chaque signal de l'horloge :

1. **fetch** :
   1. l'adresse i de l'instruction (par exemple 183) est transférée de CI vers RA,



* 1. on augmente de 1 la valeur de CI
  2. le contenu du mot d'adresse mémoire i va (via le bus de données) dans RI

1. **decode :**

l'instruction est décodée (par exemple ici add 327, c'est à dire ajouter le mot à l'adresse 327 à l'accumulateur)

1. **Execute :** 
   1. l'adresse de la donnée (ici 327) est transférée dans RA
   2. le contenu du mot d'adresse 327 (ici 56) est transféré dans l'UAL, ainsi que le contenu de l'accumulateur (ici 14) : read
   3. les 2 valeurs sont additionnée : execute proprement dit et le résultat est mis dans l'accumulateur : write back

