TP MIPS

Les registres sont dans les CPU, c'est des petites mémoires très rapides, ils stockent des informations, on a beaucoup de mémoires :Mémoire secondaire, RAM, et mémoire cache.

Les registres sont les mémoires les plus rapides de l'ordinateur, quelques de ces registres sont réservés mais la plupart d'entre eux sont importants pour le programmeur.
L'assemblage vous donne plus de contrôle que C++ et java on peut faire avec beaucoup de choses parce qu'il est en bas niveau.

- $zero : c'est le registre zéro qui stocke l'instante zéro
- $ât : c'est le registre assembly temporary
- $v0 et $v1 retournent des résultats des variables
- $a sont des registres d'argument des fonctions
- $t stockent des informations
- $s vous permettent de sauvegarder des informations des appels
- $k sont réservés pour le kernel (système d'exploitation)
- $gp pour l'environnement global
- $sp pour la pile

Name	Register number	Usage
$zero	0	the constant value 0
$at	1	reserved for the assembler
$v0–$v1	2–3	values for results and expression evaluation
$a0–$a3	4–7	arguments
$t0–$t7	8–15	temporaries
$s0–$s7	16–23	saved
$t8–$t9	24–25	more temporaries
$k0–$k1	26–27	reserved for the operating system
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

Dans un programme assembleur il 'y a deux sections :
Data section et text section

.data contient toutes les données de notre programme assembleur et .text contient toutes les instructions que le programme a besoin.
\n veut dire sauter une ligne.
Syscall veut dire au système c'est bon exécute l'instruction.

Fonctions SYSCALL disponible dans MARS
Introduction :

Un nombre de services system pour les entrées et les sorties sont disponibles à l'utilisation par votre programme MIPS, ils sont décrits dans le tableau suivant.

Table des services disponibles :

Service	Code dans $v0	Arguments	Résultats
Afficher un entier	1	$a0 = entier à afficher	
Afficher un réel	2	$f12 = réel à afficher	
Afficher un double	3	$f12 = double à afficher	
Afficher une chaine de caractères	4	$a0=adresse de la chaine terminée à afficher	
Lire un entier	5		$v0 contient la valeur de l'entier lu
Lire un réel	6		$f0 contient la valeur du réel lu
Lire un double	7		$f0 contient la valeur double lu
Lire une chaine	8	$a0 = adresse du buffer d'entrée $a1 = nombre maximal de caractères à lire	
Sbrk (allouer de la mémoire)	9	$a0 = nombre d'octets à allouer	$v0 contient l'adresse de la mémoire allouée
Sortir (exécution terminée)	10		
Afficher un caractère	11	$a0 = caractère à afficher	
Lire un caractère	12		$v0 contient le caractère lu
Ouvrir un fichier	13	$a0 = adresse d'une chaine terminée contenant nom de fichier $a1 = drapeaux $a2 = mode	$v0 contient description du fichier (négative s'il y a erreur)
Lire à partir d'un fichier	14	$a0 = description de fichier $a1 = adresse du buffer d'entrée $a2= nombre	$v0 contient un nombre de caractères lus

		maximal des caractères à lire	
Ecrire dans fichier	15	$a0 = description de fichier $a1 = adresse du buffer de sortie $a2= nombre maximal des caractères à écrire	$v0 contient un nombre de caractères écrits.
Fermer fichier	16	$a0 = description de fichier	
Sortir2 (termier avec valeur	17	$a0= résultat de terminaison	

Comment utiliser les services système SYSCALL

Etape 1 : Charger le numéro du service dans le registre $v0.

Etape 2 : Charger les valeurs d'arguments, $a0, $a1, $a2, ou $f12 sont spécifié.

Etape 3 : Lancer l'instruction SYSCALL.

Etape 4 : Retrouver les valeurs retournéesà partir des registres des résultats comme spécifiées.

L'opération d'addition (add) prend trois opérandes :

1. Un registre qui va être utilisée pour stocker le résultat de l'addition.
2. Un registre qui contient le premier nombre à être additionner.
3. Un registre qui contient le second nombre, ou une constante 32-bits.

Exemple : le code suivant montre la syntaxe de l'opérateur add

```
li  $v0, 1           # service 1 est l'affichage d'un entier

add $a0, $t0, $zero  # additionne la valeur du registre t0 avec zéro et
#charge le résultat dans le registre d'argument $a0,
syscall
```

L'opération de soustraction est faite avec l'opérateur : sub.
L'opération de multiplication est faite avec l'opérateur : mul.
L'opération de puissance est faite avec l'opérateur : ssl.