Université KASDI MERBAH – Ouargla
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département d'Informatique et Technologie de l'information

Cours architectures des ordinateurs

# Cours 3:
# Langage d'assemblage du MIPS R3000
# (Partie 2)

Enseignante: Chafika Benkherourou



## Les instructions du langage MIPS R3000

## Les opérations arithmétiques et logiques

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques

- Les instructions arithmétiques et logiques permettent de faire les 4 opérations de calcul (addition, multiplication, soustraction, division) ainsi que les opérations logiques (Or, And, Or, Xor..) et les opérations de décalage…

1. Add

add $rd, $rs, $rt

- Description

Les contenus des registres $rs et $rt sont ajoutés pour former un résultat qui est placé dans le registre $rd.

Exemple:

L'opération qui permet de faire la somme de 7 et 5 est:

```
li $t1, 7              # load 7 into $t1
li $t2, 5              # load 5 into $t2
add $t0, $t1, $t2      # $t0 = $t1 + $t2
```

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques

Exemple 2:

On peut aussi faire une addition avec les constante au lieu du registre $t2:

```
li $t1, 7              # load 7 into $t1
addi $t0, $t1, 5       # $t0 = $t1 + 5
```

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques

Exemple 3:

Faire la somme de deux nombres demandés à l'utilisateur:

```
## Get first number from user, put into $t0.
li $v0, 5           # load syscall read_int into $v0.
syscall             # make the syscall.
move $t0, $v0       # move the number read into $t0.

## Get second number from user, put into $t1.
li $v0, 5           # load syscall read_int into $v0.
syscall             # make the syscall.
move $t1, $v0       # move the number read into $t1.

add $t2, $t0, $t1    # compute the sum.

## Print out $t2.
move $a0, $t2       # move the number to print into $a0.
li $v0, 1           # load syscall print_int into $v0.
syscall             # make the syscall.

li $v0, 10          # syscall code 10 is for exit.
syscall             # make the syscall.
```

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques

2. Sub

sub $rd, $rs, $rt

- Description

  Le contenu du registre $rt est soustrait du contenu du registre $rs pour former un résultat qui est placé dans le registre $rd.

Exemple:

L'opération qui permet de soustraire 10 de 25 est:

```
li $t1,25           # load 25 into $t1
li $t2, 10          # load 10 into $t2
sub $t0, $t1, $t2   # $t0 = $t1 - $t2
```

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques

3. mul

mul $rd, $rs, $rt

- Description

Les contenus des registres $rs et $rt sont multipliés pour former un résultat qui est placé dans le registre $rd.

Exemple:

L'opération qui permet de multiplier 8 et 4 est:

```
li $t1,8              # load 8 into $t1
li $t2, 4             # load 4 into $t2
mul $t0, $t1, $t2     # $t0 = $t1 * $t2
```

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques

4. mult

mult $rs, $rt

$$\$lo \leftarrow (\$rs \times \$rt)_{31...0}$$
$$\$hi \leftarrow (\$rs \times \$rt)_{63...32}$$

- Description

mult permet de multiplier les contenus des registres $rs et $rt. Le résultat est placé dans les registres spéciaux : $hi et $lo.

Les 32 bits de poids fort du résultat sont placés dans le registre $hi, et les 32 bits de poids faible dans $lo.

Exemple:

L'opération qui permet de multiplier 8 et 4 est:

```
li $t0, 8             # load 8 into $t0
li $t1, 4             # load 4 into $t1
mult $t0, $t1         # ($hi,$lo) = $t0 * $t1
```

## Les instructions du langage MIPS R3000:

### Les instructions arithmétiques

5. Div

   div $rs, $rt

- Description

  Le contenu du registre $rs est divisé par le contenu du registre $rt.

  Le résultat de la division est placé dans le registre $lo, et le reste dans le registre $hi.

- Opération:

$$\$lo \leftarrow \frac{\$rs}{\$rt}$$
$$\$hi \leftarrow \$rs \bmod \$rt$$

## Les instructions du langage MIPS R3000:

### Les instructions arithmétiques

6. mfhi - Move from $hi -

   mfhi $rd

- Description

  Le contenu du registre spécialisé $hi — qui est mis à jour par l'opération de multiplication ou de division — est recopié dans le registre général $rd.

- Opération:

$$\$rd \leftarrow \$hi$$

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques

7. mflo - Move from $lo -

mflo $rd

- Description

Le contenu du registre spécialisé $lo — qui est mis à jour par l'opération de multiplication ou de division — est recopié dans le registre général $rd.

- Opération:

$$\$rd \leftarrow \$lo$$

# Les instructions du langage MIPS R3000:

## Les instructions logiques

1. And - Et logique -

and $rd, $rs, $rt

- Description

Un et bit-à-bit est effectué entre les contenus des registres $rs et $rt. Le résultat est placé dans le registre $rd.

- Opération: $\$rd \leftarrow \$rs \text{ and } \$rd$
- Exemple:

$t8 = 1111 1111 1111 1111 1111 1111 1100 0000

$t9 = 0000 0000 0000 0000 0000 0000 0011 1111

Supposant que les registres $t8 et $t9 sont utilisés comme masques pour effectuer des opérations spécifiques.

# Les instructions du langage MIPS R3000:

## Les instructions logiques

- Exemple:

Pour faire une copies du registre $t0 dans le registre $t1 en mettant les 6 premiers bits à zéro, on utilise l'instruction suivante:

```
and     $t1, $t0, $t8          # $t1 = $t0 & $t8
```

$t0 = 1000 1000 1111 0111 0001 0000 0100 0100

and

$t8 = 1111 1111 1111 1111 1111 1111 1100 0000

$t1 = 1000 1000 1111 0111 0001 0000 0100 0000

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques et logiques

2. Or - Ou logique -

Or $rd, $rs, $rt

- Description

  Un ou bit-à-bit est effectué entre les contenus des registres $rs et $rt. Le résultat est placé dans le registre $rd.

- Opération:

$$\$rd \leftarrow \$rs \text{ or } \$rt$$

# Les instructions du langage MIPS R3000:

## Les instructions logiques

- Exemple:

Pour faire une copies du registre $t0 dans le registre $t1 en mettant les 6 premiers bits à un, on utilise l'instruction suivante:

```
or      $t1, $t0, $t9          # $t1 = $t0 | $t9
```

$t0 = 1000 1000 1111 0111 0001 0000 0100 0100

or

$t9 = 0000 0000 0000 0000 0000 0000 0011 1111

$t1 = 1000 1000 1111 0111 0001 0000 0111 1111

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques et logiques

3. Xor - Ou exclusif -

Xor $rd, $rs, $rt

- Description

  Un ou bit-à-bit est effectué entre les contenus des registres $rs et $rt. Le résultat est placé dans le registre $rd.

- Opération:

$$\$rd \leftarrow \$rs \text{ or } \$rt$$

# Les instructions du langage MIPS R3000:

## Les instructions logiques

- Exemple:

Pour faire une copies du registre $t0 dans le registre $t1 en inversant les 6 premiers bits, on utilise l'instruction suivante:

```
xor     $t1, $t0, $t9          # $t1 = $t0 ^ $t9
```

$t0 = 1000 1000 1111 0111 0001 0000 0100 0100

Xor

$t9 = 0000 0000 0000 0000 0000 0000 0011 1111

$t1 = 1000 1000 1111 0111 0001 0000 0111 1011

# Les instructions du langage MIPS R3000:

## Les instructions logiques

4. neg - negatif -

neg $rd,$rs

- Description

Donne le contraire de $rs.

$Rd= -($Rs)

- Exercice:

```
li $t0,688
neg $t1,$t0        # $t1 = -688
move $a0,$t1       # pour imprimer $t1, il faut le placer dans $a0
li $v0, 1          # code pour print_int
syscall
```

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques et logiques

4. Sll - Shift Left Logical -

- Action

  Décalage à gauche immédiat

- Syntaxe

  sll $rd, $rt, shamt

- Description

✓ Le registre $rt est décalé à gauche de la valeur immédiate shamt.
✓ Des zéros étant introduits dans les bits de poids faibles.
✓ Le résultat est placé dans le registre $rd.

- Opération:

  Rd <- Rt<<shamt

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques et logiques

- Exemple:

  Sll $v1, $t3, 5 # $v1 = $t3 << 5

- Supposons que $t3 contient la valeur : 0000000000011010 = 26,

- Après le décalage à gauche (Sll) le résultat 0000001101000000 est placé dans le registre $v1.

- L'opération donne $832_{10} = (26 * 32) = (26 * 2^5)$

- Le Sll est utilisé pour réalisé la multiplication par des puissances de 2.

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques et logiques

### 11. Sra - Shift Right Logical arithmetic -

- Action:

  Décalage à droite arithmétique immédiat

- Syntaxe:

  sra $rd, $rt, shamt

- Description:

  Le registre $rt est décalé à droite de la valeur immédiate. Le résultat est placé dans le registre $rd.

Exemple:

- $a0 contient la valeur –32 (11111111111111111111111111100000).
- On veut diviser $a0 par (4 =$2^2$). Le décalage arithmétique à gauche (Sra) par 2 positions peut effectuer cette opération. Le résultat = –8 (11111111111111111111111111111000).

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques et logiques

### 12. Slt - Set if Less Than -

- Action:

  Comparaison signée registre registre

- Syntaxe:

  slt $rd, $rs, $rt

- Description:

✓ Le contenu du registre $rs est comparé au contenu du registre $rt.

✓ Si la valeur contenue dans $rs est inférieure à celle contenue dans $rt, alors $rd prend la valeur 1, sinon il prend la valeur 0.

- Opération:

**if ([rs] < [rt] ) then [rd] =1 else [rd] = 0**

# Les instructions du langage MIPS R3000:

## Les instructions arithmétiques et logiques

Autres instructions arithmétiques:

- Addu Subu Addiu Multu Divu
- Nor Ori Andi Xori
- Srav (Shift right arithmetical variable)
- Sltu (Set if Less Than Unsigned)
- Mthi (Move to hi)
- Mtlo (Move to Lo)