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 3)

Enseignante: Chafika Benkherourou

Les instructions du langage MIPS R3000

Les opérations de Saut (Jump) et de Branchement (Branch)

Introduction:

- Les structures de contrôles comme le IF..THEN..ELSE, WHILE et FOR n'existent pas en langage d'assemblage.
- Pour les programmer, il faut utiliser les instructions de sauts et de branchements.
- Ces instructions sont de type conditionnel ou inconditionnel.

Les instructions du langage MIPS R3000:

Les instructions de branchement

beq $s0, $s1, label	if $s0==$s1 goto label	branch if equal
bne $s0, $s1, label	if $s0!=$s1 goto label	branch if not equal
bge $s0, $s1, label	if $s0>=$s1 goto label	Branch if greater or equal
bgt $s0, $s1, label	if $s0> $s1 goto label	Branch if greater than
ble $s0, $s1, label	if $s0<=$s1 goto label	Branch if less or equal
blt $s0, $s1, label	if $s0<$s1 goto label	Branch if less than
bgez $s0, label	if $s0>=0 goto label	Branch if greater or equal zero
bgtz $s0, label	if $s0> 0 goto label	Branch if greater than zero
blez $s0, label	if $s0<=0 goto label	Branch if less or equal zero
bltz $s0, label	if $s0< 0 goto label	Branch if less than zero
bnez $s0, label	if $s0!=0 goto label	branch if not equal zero
beqz $s0, label	if $s0==0 goto label	branch if equal zero

Les instructions du langage MIPS R3000:

Les instructions de branchement

- Beq (Branch if equal):
  - `beq $R1, $R2, Label`
  - Signifie :

    `Si ($R1==$R2) goto Label`

- Bne (Branch if not equal):
  - `bne $R1, $R2, Label`
  - Signifie :

    `Si ($R1≠$R2) goto Label`

- Bgt (Branch if greater than):
  - `bgt $R1, $R2, Label`
  - Signifie :

    `Si ($R1>$R2) goto Label`

Les instructions du langage MIPS R3000:

Les instructions de branchement

- Bgez (Branch if greater or equal zero):
  - `bgez $R1, Label`
  - Signifie :

    `Si ($R1>=0) goto Label`

- Bltz (Branch if less than zero):
  - `bltz $R1, Label`
  - Signifie :

    `Si ($R1<0) goto Label`

Les instructions du langage MIPS R3000:

Les instructions de branchement

- Beqz (Branch if equal zero):
  - `beqz $R1, Label`
  - Signifie :

    `Si ($R1=0) goto Label`

- Bnez (Branch if not equal zero):
  - `bnez $R1, Label`
  - Signifie :

    `Si ($R1≠0) goto Label`

Les instructions du langage MIPS R3000:

Les instructions de saut

1. J - Jump -

- Action

  Le programme saute inconditionnellement à Label.

- Syntaxe

  j Label

- Exemple:

```
blockA: ... instructions
        j exit
blockB: ... instructions
exit:   ...
```

Les instructions du langage MIPS R3000:

Les instructions de saut

2. Jr - Jump register -

- Action

  Le programme saute à l'adresse contenue dans le registre $rs.

- Syntaxe

  Jr $rs

Les instructions du langage MIPS R3000:

Les instructions de saut

3. Jal - Jump and link -

- Action:

  Utilisé dans les appels des procédures et fonctions.

  L'adresse de l'instruction suivant le jal est sauvée dans le registre ($ra).

- Syntaxe

  Jal Label

IF…..THEN…..

Pour réaliser l'instruction If..Then.. suivante:

```
IF num < 0 THEN

 num := num+5


ENDIF
```

- Le code équivalent en Mips est:

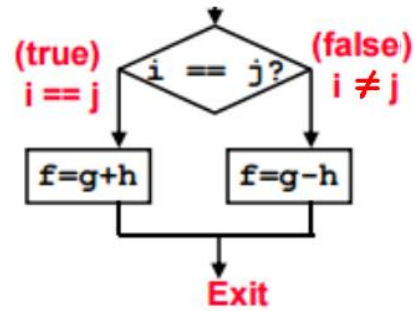
```
        lw   $t0, num

       bgez $t0, pos
          addi  $t0,$t0,5
       pos:
      # ……..
```

Condition de sortie

IF…..THEN…..ELSE……

Pour réaliser l'instruction If..Then..Else …suivante:

```
if (i == j)   then
     f = g+h
else
     f = g-h;
```



- Le code Mips équivalent est :

Supposons que :

f= $t0, g=$t1, h=$t2, i=$t3, j=$t4

```
beq $t3, $t4, True
 sub $t0, $t1, $t2 # f=g-h(false)
 j Fin # go to Fin
True: add $t0,$t1,$t2 # f=g+h (true)
Fin:
```

Jump obligatoire avant else

IF…..THEN…..ELSE……

Exemple 2:

Le code suivant:

```
If $t8<0 then
    $t1=$t1+1
else
    $t2=$t2+1
```

- Le code Mips correspondant est:

```
        bgez    $t8, else           # if ($t8 is > or = zero) branch to else
        addi    $t1, $t1, 1         # increment $t1 by 1
        j       next                # branch around the else code
else:
        addi    $t2, $t2, 1         # increment $t2 by 1
next:
```

WHILE…..DO…

Pour réaliser l'instruction While…Do…suivante:

```
While ($a1 < $a2) do
        {
    $a1 = $a1 + 1
    $a2 = $a2 - 1
        }
    return
```

Condition pour sortir de la boucle

```
loop:
        bge     $a1, $a2, done      # If( $a1 >= $a2) Branch to done
        addi    $a1, $a1, 1         # $a1 = $a1 + 1
        addi    $a2, $a2, -1        # $a2 = $a2 - 1
        j       loop                # Branch to loop
done:
```

jump pour refaire la boucle

WHILE…..DO…

Exemple:

- Calculer la somme de N nombres saisis au clavier jusqu'à entrer le nombre 0.

```
.text
main:

loop:
li $v0,5
 syscall
 beqz $v0,finish
 add $s1, $s1, $v0 # $s1 is the sum
 j loop

finish:

li $v0, 10
syscall

.end main
```

Condition pour sortir de la boucle

Refaire la boucle

FOR…DO…

Pour réaliser l'instruction For… To …Do {… } suivante:

```
$a0:=0;
For $t0:=1 to 10 Do
        $a0=$a0+$t0
```

- Le code Mips correspondant est:

```
        li      $a0, 0          #  $a0 = 0
        li      $t0, 10         # Initialize loop counter to 10
loop:
        add     $a0, $a0, $t0
        addi    $t0, $t0, -1    # Decrement loop counter
        bgtz    $t0, loop       # If ($t0 > 0) Branch to loop
```

Compteur au début de la boucle

Tant que la condition est vraie on branche vers loop