

ARQUITECTURA DE COMPUTADORES

2º Grado en Ingeniería Informática

Curso 2011 – 2012

Página 1 de 3

Relación de ejercicios

Tema 2: Instrucciones: lenguaje máquina. Repertorio de instrucciones MIPS

Ejercicio 1

Realizar la traducción de los siguientes fragmentos de código en lenguaje C a instrucciones MIPS (sin emplear pseudoinstrucciones) utilizando el menor número de instrucciones posible en cada caso. Las variables f, g, h, e i se han asignado a los registros \$s0, \$s1, \$s2, y \$s3 respectivamente. Las direcciones base de los vectores A y B se encuentran en los registros \$s4 y \$s5 respectivamente.

Fragmento 1	Fragmento 2	Fragmento 3
$f = g + h + i$	$f = g - h - i$	$f = g + h + B[4]$
Fragmento 4	Fragmento 5	Fragmento 6
$f = -g + h - B[4]$	$f = g - A[B[4]]$	$f = g + A[B[h] + 1]$

Ejercicio 2

Realizar la traducción de los siguientes fragmentos de código de instrucciones MIPS a lenguaje C. Las variables f, g, h, e i se han asignado a los registros \$s0, \$s1, \$s2, y \$s3 respectivamente. Las direcciones base de los vectores A y B se encuentran en los registros \$s4 y \$s5 respectivamente.

Fragmento 1	Fragmento 2	Fragmento 3	Fragmento 4
add \$s0, \$s1, \$s2 sub \$s0, \$s0, \$s3	lw \$t0, 8(\$s5) add \$t0, \$s0, \$t0 sw \$t0, 4(\$s4)	lw \$t0, 20(\$s5) add \$s0, \$s1, \$t0 lw \$t0, 16(\$s4) sub \$s0, \$t0, \$s0	add \$t0, \$s2, \$s3 sll \$t0, \$t0, 2 add \$s0, \$s1, \$t0 sll \$t0, \$s3, 3 add \$t0, \$s4, \$t0 lw \$t0, 0(\$t0) sub \$s0, \$s0, \$t0

Ejercicio 3

Realizar la codificación en lenguaje máquina de las siguientes instrucciones MIPS.

Instrucción 1	Instrucción 2	Instrucción 3	Instrucción 4
add \$t0, \$t0, \$zero	and \$t1, \$s0, \$s1	lw \$t1, 4(\$s3)	lui \$t0, 255
Instrucción 5	Instrucción 6	Instrucción 7	Instrucción 8
sw \$s4, 8(\$t0)	sll \$t2, \$s0, 4	bne \$t0, \$s5, X X = tres instrucciones por debajo	j 80000

Ejercicio 4

Realizar la traducción desde el lenguaje máquina a instrucciones MIPS.

Instrucción 1	Instrucción 2
0000 0001 1000 1010 1000 0000 0010 0010	1000 1101 0000 1000 1111 1111 1100 0000

ARQUITECTURA DE COMPUTADORES

2º Grado en Ingeniería Informática

Curso 2011 – 2012

Página 2 de 3

Relación de ejercicios

Tema 2: Instrucciones: lenguaje máquina. Repertorio de instrucciones MIPS

Instrucción 3	Instrucción 4
0011 1100 0000 1001 0000 0100 1010 0011	1010 1110 0000 1011 1111 1111 1100
Instrucción 5	Instrucción 6
0000 0010 0001 0001 0100 0000 0010 1010	0000 0000 0001 0001 0100 1001 0000 0010
Instrucción 7	Instrucción 8
0001 0010 0001 0001 0000 0000 0000 0011	0000 1000 0000 0000 0000 0010 0110 1100

Ejercicio 5

Realizar la traducción de los siguientes fragmentos de código en lenguaje C a instrucciones MIPS tratando de hacer la implementación lo más literal posible. Las variables a e i se han asignado a los registros \$s0 y \$s1 respectivamente. La dirección base del vector F se encuentra en el registro \$s2.

Fragmento 1	Fragmento 2
<pre>a = 0; for (i=0; i<5; ++i) { a = a+i; }</pre>	<pre>a = 0; i = 5; do { F[i--] = a++; } while (i >= 0)</pre>

Fragmento 3	Fragmento 4
<pre>for (i=0; i<5; ++i) { F[i] = rst2(4*i,i); } int rst2(int a1, int a2) { return 2*(a1-a2); }</pre>	<pre>for (i=0; i<5; ++i) { swap(F,i,9-i); } void swap(int v[], int a1, int a2) { int temp; temp = v[a1]; v[a1] = v[a2]; v[a2] = temp; }</pre>

Ejercicio 6

Implementar un programa mediante instrucciones MIPS que realice el cálculo de los diez primeros términos de una serie cuyo primer término es 1 y los siguientes se calculan en base a la expresión: $a_n = 2 \cdot a_{n-1} + 1$. Almacenar los términos en un array en memoria.

Ejercicio 7

Implementar una rutina mediante instrucciones MIPS que reciba como parámetro un array de números enteros positivos y devuelva el menor y el mayor valor de los presentes en el array. Implementar un programa para comprobar su funcionamiento.

ARQUITECTURA DE COMPUTADORES

2º Grado en Ingeniería Informática

Curso 2011 – 2012

Página 3 de 3

Relación de ejercicios

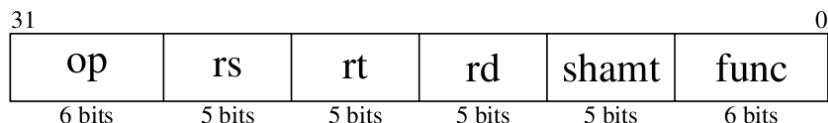
Tema 2: Instrucciones: lenguaje máquina. Repertorio de instrucciones MIPS

Material complementario

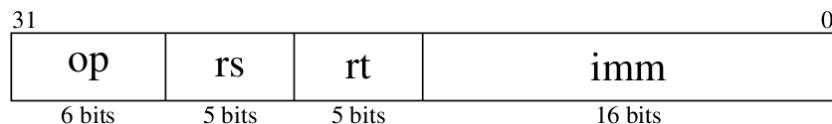
Registros del MIPS:

Números	Nombres	Uso	Preservados entre llamadas
0	\$zero	Valor constante 0	No aplicable
1	\$at	Reservado para uso temporal del ensamblador	No
2, 3	\$v0, \$v1	Resultados de las llamadas a procedimiento y evaluación de expresiones	No
4, ..., 7	\$a0, ..., \$a3	Argumentos de las llamadas a procedimiento	No
8, ..., 15	\$t0, ..., \$t7	Temporales	No
16, ..., 23	\$s0, ..., \$s7	Temporales guardados	Sí
24, 25	\$t8, \$t9	Temporales	No
26, 27	\$k0, \$k1	Reservado para su uso por el núcleo del sistema operativo	No
28	\$gp	Puntero global	Sí
29	\$sp	Puntero de pila	Sí
30	\$fp	Puntero de marco	Sí
31	\$ra	Dirección de retorno	Sí

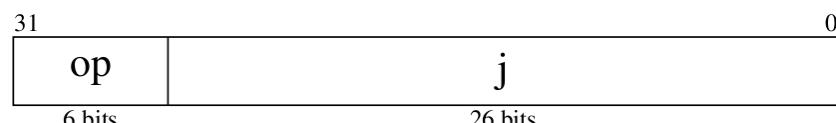
Formato de instrucción tipo R:



Formato de instrucción tipo I:



Formato de instrucción tipo J:



```

add A, B, C # A=B+C
addi A, B, 5 # A=B+5
sub A, B, C # A=B-C
sll A, B, 1 # A=2^B=2B
lw A, 0B # A=B[0]
sw A, 0B # B[0]=A
jal función # llama a función
jr $ra # devuelve $ra
seti A, B, C # { A=1 if B<C
                A=0 if B≥C
bne A, $zero, función # A=0 sale

```

Ejercicio 1

Realizar la traducción de los siguientes fragmentos de código en lenguaje C a instrucciones MIPS (sin emplear pseudoinstrucciones) utilizando el menor número de instrucciones posible en cada caso. Las variables f, g, h, e i se han asignado a los registros \$s0, \$s1, \$s2, y \$s3 respectivamente. Las direcciones base de los vectores A y B se encuentran en los registros \$s4 y \$s5 respectivamente.

Fragmento 1	Fragmento 2	Fragmento 3
$f = g + h + i$	$f = g - h - i$	$f = g + h + B[4]$

Fragmento 4	Fragmento 5	Fragmento 6
$f = -g + h - B[4]$	$f = g - A[B[4]]$	$f = g + A[B[h] + 1]$

$\$s0 \equiv f$

$\$s1 \equiv g$

$\$s2 \equiv h$

$\$s3 \equiv i$

$\$s4 \equiv \text{dir}(A[0])$

$\$s5 \equiv \text{dir}(B[0])$

1. add \$s0, \$s1, \$s2

add \$s0, \$s0, \$s3

2. sub \$s0, \$s1, \$s2

sub \$s0, \$s0, \$s3

3. add \$s0, \$s1, \$s2

lw \$t0, 16(\$s5) # B[4]

add \$s0, \$s0, \$t0

4. sub \$s0, \$zero, \$s1

add \$s0, \$s0, \$s2

lw \$t0, 16(\$s5)

sub \$t1, \$zero, \$t0

add \$s0, \$s0, \$t1

5. lw \$t0, 16(\$s5) # b[4]

sll \$t0, \$t0, 2 # 4(B[4])

add \$t0, \$s4, \$t0 # dir A[B[4]]

lw \$t0, 0(\$t0) # A[B[4]]

sub \$s0, \$s1, \$t0

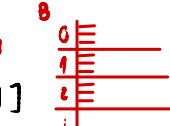
6. lw \$t0, 16(\$s2+1)(\$s5) # B[h]+1

sll \$t0, \$t0, 2 # 4(B[h]+1)

add \$t0, \$s4, \$t0 # dir A[B[h]+1]

lw \$t0, 0(\$t0) # A[B[h]+1]

add \$s0, \$s1, \$t0



Ejercicio 2

Realizar la traducción de los siguientes fragmentos de código de instrucciones MIPS a lenguaje C. Las variables f, g, h, e i se han asignado a los registros \$s0, \$s1, \$s2, y \$s3 respectivamente. Las direcciones base de los vectores A y B se encuentran en los registros \$s4 y \$s5 respectivamente.

Fragmento 1	Fragmento 2	Fragmento 3	Fragmento 4
<pre>add \$s0, \$s1, \$s2 sub \$s0, \$s0, \$s3</pre>	<pre>lw \$t0, 8(\$s5) add \$t0, \$s0, \$t0 sw \$t0, 4(\$s4)</pre>	<pre>lw \$t0, 20(\$s5) add \$s0, \$s1, \$t0 lw \$t0, 16(\$s4) sub \$s0, \$t0, \$s0</pre>	<pre>add \$t0, \$s2, \$s3 sll \$t0, \$t0, 2 add \$s0, \$s1, \$t0 sll \$t0, \$s3, 3 add \$t0, \$s4, \$t0 lw \$t0, 0(\$t0) sub \$s0, \$s0, \$t0</pre>

$\$s0 \equiv f$

$\$s1 \equiv g$

$\$s2 \equiv h$

$\$s3 \equiv i$

$\$s4 \equiv \text{dir}(A[0])$

$\$s5 \equiv \text{dir}(B[0])$

1. add \$s0, \$s1, \$s2 $\# f = g + h$
 sub \$s0, \$s0, \$s3 $\# f = g + h - i$

2. lw \$t0, 8(\$s5) $\# t0 = B[2]$
 add \$t0, \$s0, \$t0 $\# t0 = f + B[2]$
 sw \$t0, 4(\$s4) $\# A[1] = f + B[2]$

3. lw \$t0, 20(\$s5) $\# t0 = B[5]$
 add \$s0, \$s1, \$t0 $\# f = g + B[5]$
 lw \$t0, 16(\$s4) $\# t0 = A[4]$
 sub \$s0, \$t0, \$s0 $\# f = A[4] - g - B[5]$

4. add \$t0, \$s2, \$s3 $\# t0 = h + i$
 sll \$t0, \$t0, 2 $\# t0 = 4(h + i)$ $001 \rightarrow 100$
 add \$s0, \$s1, \$t0 $\# f = g + 4(h + i)$
 sll \$t0, \$s3, 3 $\# t0 = 8i$
 add \$t0, \$s4, \$t0 $\# t0 = d(A[0]) + 8i = d(A[2i])$
 lw \$t0, 0(\$t0) $\# t0 = A[2i]$
 sub \$s0, \$s0, \$t0 $\# f = g + 4(h + i) - A[2i]$

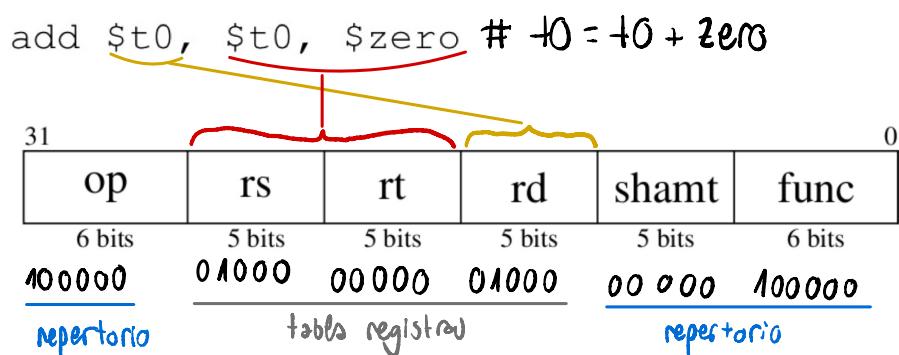
Ejercicio 3 NO ENTRA

Realizar la codificación en lenguaje máquina de las siguientes instrucciones MIPS.

Instrucción 1	Instrucción 2	Instrucción 3	Instrucción 4
add \$t0, \$t0, \$zero	and \$t1, \$s0, \$s1	lw \$t1, 4(\$s3)	lui \$t0, 255

Instrucción 5	Instrucción 6	Instrucción 7	Instrucción 8
sw \$s4, 8(\$t0)	sll \$t2, \$s0, 4	bne \$t0, \$s5, X X = tres instrucciones por debajo	j 80000

1.



Ejercicio 4 NO ENTRA

Realizar la traducción desde el lenguaje máquina a instrucciones MIPS.

Instrucción 1	Instrucción 2
0000 0001 1000 1010 1000 0000 0010 0010	1000 1101 0000 1000 1111 1111 1100 0000

sub \$t4, \$t2, \$t0 rep rep

Ejercicio 5

Realizar la traducción de los siguientes fragmentos de código en lenguaje C a instrucciones MIPS tratando de hacer la implementación lo más literal posible. Las variables a e i se han asignado a los registros \$s0 y \$s1 respectivamente. La dirección base del vector F se encuentra en el registro \$s2.

Fragmento 1	Fragmento 2
<pre>a = 0; for (i=0; i<5; ++i) { a = a+i; }</pre>	<pre>a = 0; i = 5; do { F[i--] = a++; } while (i >= 0)</pre>

Fragmento 3	Fragmento 4
<pre>for (i=0; i<5; ++i) { F[i] = rst2(4*i,i); } int rst2(int a1, int a2) { return 2*(a1-a2); }</pre>	<pre>for (i=0; i<5; ++i) { swap(F,i,9-i); } void swap(int v[], int a1, int a2) { int temp; temp = v[a1]; v[a1] = v[a2]; v[a2] = temp; }</pre>

$\$s0 \equiv a$

$\$s1 \equiv i$

$\$s2 \equiv \text{dir}(F[0])$

1. $a = 0;$
 $\text{for } (i=0; i<5; ++i) \{$
 $\quad a = a+i;$
 $\}$

$\text{add } \$s0, \$zero, \$zero \# j=0$

$\text{add } \$s1, \$zero, \$zero \# i=0$

BUCLE $\left\{ \begin{array}{l} \text{add } \$s0, \$s0, \$s1 \# j=j+1 \\ \text{addi } \$s1, \$s1, 1 \# i=i+1 \\ \text{seti } \$t0, \$s1, 5 \# t0 \begin{cases} 1 & \text{si } i < 5 \\ 0 & \text{si } i \geq 5 \end{cases} \\ \text{bne } \$t0, \$zero, \text{bucle} \# \text{si } t0=0 \text{ sale} \end{array} \right.$

FIN

2. a = 0;

i = 5;

do {

 F[i--] = a++;

} while (i >= 0)

add \$S0, \$zero, \$zero # i=0

addi \$S1, \$zero, 5 # i=5

BUCLE { addi \$S1, \$S1, 1 # i++

 sll \$T0, \$S1, 2 # T0 = 4i

 add \$T0, \$T0, \$S2 # T0 = F[i]

 sw \$S0, 0(\$T0) # F[i] = 2

 addi \$S1, \$S1, -1 # i = i-1

 seti \$T0, \$S1, 0 # T0 = {1 if i<0
 0 if i≥0}

 bne \$T0, \$zero, bucle # si T0=0 salte

3. for (i=0; i<5; ++i) {

 F[i] = rst2(4*i, i);

}

int rst2(int a1, int a2) {

 return 2*(a1-a2);

}

add \$S1, \$zero, \$zero # i=0

BUCLE { sll \$A1, \$S1, 2 # A1 = 4i

 add \$A2, \$S1, \$zero # A2 = i

 jal rst2 # v[0] = rst2(a1, a2) # CSR

 add \$T0, \$S2, \$A1 # T0 = d(F[0]) + 4i = d(F[i])

 sw \$V0, 0(\$T0) # F[i] = v0

 addi \$S1, \$S1, 1 # i++

 seti \$T0, \$S1, 5 # T0 = (i<5)

 bne \$T0, \$zero, bucle

FIN

RST2 { sub \$V0, \$A1, \$A2 # V0 = A1 - A2

 sll \$V0, \$V0, 1 # V0 = 2V0

 jr \$ra # JMPI

2.

```

4. for (i=0; i<5; ++i) {
    swap(F,i,9-i);
}

void swap(int v[], int a1, int a2) {
    int temp;

```

temp = v[a1];
v[a1] = v[a2];
v[a2] = temp;

} add \$s1, \$zero, \$zero # i = 0

add \$t0, \$zero, 9 # t0 = 9

add \$a0, \$zero, \$s2 # a0 = dir(F[0]) par valor

BUCLE { add \$a1, \$zero, \$s1 # a1 = i
sub \$a2, \$t0, \$s1 # a2 = 9 - i
jal swap # llamo a swap
addi \$s1, \$s1, 1 # i++
seti \$t0, \$s1, 5 # t0 = (i < 5)
bne \$t0, \$zero, bucle # si t0 != 0 loop

FIN

SWAP { sll \$t2, \$a1, 2 # t2 = 4 * i
sll \$t3, \$a2, 2 # t3 = 4 * 2
add \$t2, \$t2, \$a0 # t2 = d(F[0]) + 4 * i = d(F[a1])
add \$t3, \$t3, \$a0 # t3 = d(F[a2])
lw \$t4, 0(\$t2) # t4 = F[a1]
lw \$t5, 0(\$t3) # t5 = F[a2]
sw \$t4, 0(\$t3) # F[a1] = F[a2]
sw \$t5, 0(\$t2) # F[a2] = F[a1]
jr \$ra

Ejercicio 6

Implementar un programa mediante instrucciones MIPS que realice el cálculo de los diez primeros términos de una serie cuyo primer término es 1 y los siguientes se calculan en base a la expresión: $a_n = 2 \cdot a_{n-1} + 1$. Almacenar los términos en un array en memoria.

```
$s0 = i  
$s1 = ldc A[0]  
add $s0, $zero, $zero      # i=0  
BUCLE { sll $t0, $s0, 2      # t0=4i  
    add $t0, $t0, $s1      # A[i]  
    sll $t1, $s0, 1      # t1=2i  
    addi $t1, $t1, 1      # t1=2i+1  
    sw $t1, 0($t0)      # A(i)=2i+1  
    addi $s0, $s0, 1      # i++  
    seti $t0, $s0, 10      # { t0=1 i<10  
    seti $t0, $s0, 10      # t0=0 i≥10  
    bne $t0, $zero, bucle  # si t0≠0 sale
```

Ejercicio 7

Implementar una rutina mediante instrucciones MIPS que reciba como parámetro un array de números enteros positivos y devuelva el menor y el mayor valor de los presentes en el array. Implementar un programa para comprobar su funcionamiento.

```
$s0 = i  
$s1 = ldc(F[0])  
$s2 = m2  
$s3 = m3  
add $s0, $zero, $zero      # i=0  
sw $s2, 0($s1)      # m2 = F[0]  
sw $s3, 0($s1)      # m3 = F[0]  
BUCLE { sll $t0, $s0, 2      # t0=4i  
    lw $t0, 4($s0)($s1)  # F[i]  
    IF $s2 < $t0 : lw $s2, 0($t0) # m2=F[i]  
    IF $s3 > $t0 : lw $s3, 0($t0) # m3=F[i]  
    addi $s0, $s0, 1      # i++  
    seti $t1, $s0, 5      # { t1=1 i<5  
    seti $t1, $s0, 5      # t1=0 i≥5  
    bne $t0, $zero, bucle  # si t0≠0 sale
```

```
add A, B, C  # A=B+C  
addi A, B, 5  # A=B+5  
sub A, B, C  # A=B-C  
sll A, B, 1  # A=2^B=2B  
lw A, 0B      # A=B[0]  
sw A, 0B      # B[0]=A  
jal función  # llama a función  
jr $ra        # devuelve $ra  
seti A, B, C  # { A=1 if B<C  
                # A=0 if B≥C  
bne A, $zero, función # A=0 sale
```