Departamento de Informática
Escuela Politécnica
Universidad de Extremadura

REPERTORIO DE INSTRUCCIONES MIPS

Prácticas de la Asignatura
ESTRUCTURA DE COMPUTADORES

2º Ingeniería Informática
2º Ingeniería Técnica en Informática de Sistemas

Curso 2000 / 2001

Juan A. Gómez Pulido
2000

Instrucciones de carga / almacenamiento

Son instrucciones que leen y escriben en memoria utilizando registros.

---

lw $t0, dir

load-word: Carga en el registro $t0 el contenido de la palabra de memoria cuya dirección es dir

Supongamos que en el segmento de datos escribimos:

```
        .data
dir:    .byte  0x0d, 2, 11
```

Entonces la memoria estará así:

00	00	00	00
00	00	00	00
00	00	00	00
00	0b	02	0d

Después de ejecutar la instrucción `lw $t0,dir` , el registro $t0 tomará el valor 0x000b020d

dir es una etiqueta que indica la dirección 0x10010000, que es la dirección del byte 0x0d, es decir, de la primera posición de la palabra de memoria

---

lb $t0, dir

load-byte: Carga en el registro $t0 el contenido del byte de memoria cuya dirección es dir

Siguiendo con el ejemplo de la memoria anterior, si $t0 vale 0x000b020d y ejecutamos la instrucción

lb $t0, dir+2

entonces el contenido de $t0 será ahora 0x0000000b (ojo, no 0x0007020b)

---

la $t0, dir

load-address: Carga en el registro $t0 la dirección de memoria etiquetada con dir

Siguiendo con el ejemplo anterior, después de ejecutar esta instrucción, el contenido de $t0 será 0x10010000

---

sw $t0, dir

store-word: Almacena en la palabra de memoria direccionada por dir, el contenido del registro $t0

Partiendo de esta situación de la memoria dibujada a la derecha, supongamos que el contenido del registro $t0 es 0x010f123a

00	00	00	00
00	00	00	00
00	00	00	00
00	0b	02	0d

Ejecutamos `sw $t0, dir+8`

La situación de la memoria después de la ejecución de esta instrucción es la que está dibujada a la derecha

00	00	00	00
01	0f	12	3a
00	00	00	00
00	0b	02	0d

sb $t0, dir

store-byte: Almacena en el byte de memoria direccionado por dir, el contenido del byte menos significativo del registro $t0

Supongamos que $t0 tiene el valor 0x010f123a, y que la situación actual de la memoria es la de la derecha. Después de ejecutar la instrucción

```
sb $t0, dir+1
```

el contenido de la memoria es ahora el reflejado a la derecha

00	00	00	00
00	00	00	00
00	00	00	00
00	0b	02	0d

00	00	00	00
00	00	00	00
00	00	00	00
00	0b	3a	0d

Las instrucciones hasta aquí descritas son las que necesitaremos para todas las prácticas. A continuación se listqan todas las instrucciones de este tipo:

Instrucción	Acción	Descripción
la Rd, dir	Carga dirección	Carga en Rd la dirección dir (no su contenido)
lb Rd, dir	Carga byte	Carga en Rd el byte de memoria direccionado por dir
lbu Rd, dir	Carga byte sin signo	Carga en Rd el byte de memoria direccionado por dir (extiende el signo)
ld Rd, dir	Carga palabra doble	Carga en Rd y en Rd+1 las dos palabras de memoria (64 bits) direccionadas a partir de dir
lh Rd, dir	Carga mitad de palabra	Carga en Rd la mitad inferior de la palabra de memoria (16 bits) direccionada por dir
lhu Rd, dir	Carga mitad de palabra sin signo	Carga en Rd la mitad inferior de la palabra de memoria (16 bits) direccionada por dir, y extiende el signo
lw Rd, dir	Carga palabra	Carga en Rd la palabra de memoria direccionada por dir
lwcz Rd, dir	Carga palabra en coprocesador	Carga en el registro Rd del coprocesador z (0-3), la palabra de memoria direccionada por dir
lwl Rd, dir **lwr** Rd, dir	Carga palabra izquierda (derecha)	Carga en Rd los bytes izquierdos (derechos) de la palabra de memoria posiblemente no alineada direccionada por dir
sb Rs, dir	Almacena byte	Almacena el byte menos significativo del registro Rs en la posición de memoria direccionada por dir
sd Rs, dir	Almacena palabra doble	Almacena los 64 bits de los registros Rs y Rs+1 en la posición de memoria direccionada por dir
sh Rs, dir	Almacena mitad de palabra	Almacena los dos bytes menos significativa del registro Rs en la posición de memoria direccionada por dir
sw Rs, dir	Almacena palabra	Almacena la palabra del registro Rs en la posición de memoria direccionada por dir
swcz Rs, dir	Almacena palabra de coprocesador	Almacena la palabra del registro Rs del coprocesador z en la posición de memoria direccionada por dir
swl Rs, dir **swr** Rs, dir	Almacena palabra izquierda (derecha)	Almacena los bytes izquierdos (derechos) del registro Rs en la posición de memoria posiblemente no alineada direccionada por dir
ulh Rd, dir	Carga mitad de palabra no alineada	Carga en Rd la mitad de palabra de memoria posiblemente no alineada direccionada por dir
ulhu Rd, dir	Carga mitad palabra no alineada sin signo	Carga en Rd (y extiende el signo) la mitad de palabra de memoria posiblemente no alineada direccionada por dir
ulw Rd, dir	Carga palabra no alineada	Carga en Rd la palabra de memoria posiblemente no alineada direccionada por dir
ush Rs, dir	Almacena mitad de palabra no alineada	Almacena la mitad de palabra menos significativa del registro Rs en la posición de memoria posiblemente no alineada direccionada por dir
usw Rs, dir	Almacena palabra no alineada	Almacena la palabra del registro Rs en la posición de memoria posiblemente no alineada direccionada por dirr

Instrucciones aritméticas.

Todas las instrucciones aritméticas "funcionan" igual que en el siguiente ejemplo:

add $t0, $t1, $t2 *Suma el contenido de los registros $t1 y $t2, y el resultado se almacena en el registro $t0*	Supongamos que los registros $t1 y $t2 tienen los valores 0x00000007 y 0x00000003 respectivamente. Después de ejecutarse la instrucción de la izquierda, el registro $t0 toma el valor de 0x0000000a. Esta instrucción admite como segundo sumando un número entero. Por ejemplo, *add $t0, $t1, 3* produce el mismo resultado que la anterior instrucción. Esto no vale para el primer sumando, que tiene que ser siempre un registro. Todas las instrucciones aritméticas que terminan con "*i*" (de *integer*, entero) indican que el segundo operando sea un número entero (por ejemplo, *addi $t0, $t1, $t2*).

Las instrucciones que se van a necesitar para todas las prácticas son:

add	suma
sub	resta
mul	multiplicación

La siguiente tabla ofrece el repertorio completo de instrucciones aritmético-lógicas. En todas las siguientes instrucciones:

- Rd es el registro destino.
- Rs1 es un registro que hace de primer operando fuente.
- Rs2 es el segundo operando fuente, que puede ser registro o valor entero (las formas inmediatas de las instrucciones se incluyen únicamente como referencia; el ensamblador traducirá la forma más general de una instrucción -e.g., add- en su forma inmediata -e.g., addi- si el segundo argumento es constante). RRs2 indica que el segundo operando fuente solo puede ser un registro
- Imm es un valor inmediato.

Instrucción	Acción	Descripción
abs Rd, Rs	Valor absoluto	Pone el valor absoluto del entero del registro Rs en el registro Rd
add Rd, Rs1, Rs2	Suma (con desbordamiento)	Suma Rs1 y Rs2, y lo pone en Rd
addi Rd, Rs1, Imm	Suma inmediata (con desbordamiento)	Suma Rs1 y el valor Imm, y lo pone en Rd
addu Rd, Rs1, Rs2	Suma (sin desbordamiento)	Suma Rs1 y Rs2, y lo pone en Rd.
addiu Rd, Rs1, Imm	Suma inmediata (sin desbordamiento)	Suma Rs1 y el valor Imm, y lo pone en Rd.
and Rd, Rs1, Rs2 **andi** Rd, Rs1, Imm	AND AND inmediato	Pone en Rd el resultado de aplicar AND lógico a los contenidos de Rs1 y Rs2 (o inmediato)

div Rs1, RRs2 **divu** Rs1, RRs2	División (con desbordamiento) División (sin desbordamiento)	Divide el contenido de los dos registros. Deja el cociente en el registro LO y el resto en el registro HI. Si un operando es negativo, el resto no es especificado por la arquitectura MIPS y depende de las convenciones del computador en el cual SPIM se ejecuta.
div Rd, Rs1, Rs2 **divu** Rd, Rs1, Rs2	División (con desbordamiento) División (sin desbordamiento)	Pone el cociente de los enteros que hay en los registros Rs1 y Rs2, en el registro Rd
mul Rd, Rs1, Rs2 **mulo** Rd, Rs1, Rs2 **mulou** Rd, Rs1, Rs2	Multiplicación(con desbordamiento) Multiplicación (sin desbordamiento) Multiplicación sin signo (con desbordamiento)	Pone el producto de los enteros que hay en los registros Rs1 y Rs2, en el registro Rd
mult Rs1, RRs2 **multu** Rs1, RRs2	Multiplicación Multiplicación sin signo	Multiplica el contenido de los dos registros. Deja la palabra menos significativa del producto en el registro LO, y la más significativa en HI.
neg Rd, Rs **negu** Rd, Rs	Negate Value (with overflow) Negate Value (without overflow)	Pone een Rd el negativo del entero contenido en Rs
nor Rd, Rs1, Rs2	NOR	Pone en Rd el NOR lógico de los enteros contenidos en Rs1 y Rs2
not Rd, Rs	NOT	Pone en Rd el NOT lógico del entero contenido en Rs
or Rd, Rs1, Rs2 **ori** Rd, Rs1, Imm	OR OR inmediato	Pone en Rd el OR lógico de los enteros contenidos en Rs1 y Rs2 (o inmediato)
rem Rd, Rs1, Rs2 **remu** Rd, Rs1, Rs2	Resto Resto sin signo	Pone en Rd el resto de dividir el entero de Rs1 por el entero de Rs2. Si un operando es negativo, el resto no es especificado por la arquitectura MIPS, y depende de las convenciones del computador en el cual SPIM se ejecuta.
rol Rd, Rs1, Rs2 **ror** Rd, Rs1, Rs2	Rotación a la izquierda Rotación a la derecha	Rota el contenido de Rs1 a la izquierda (derecha) según la distancia indicada por Rs2, y pone el resultado en Rd
sll Rd, Rs1, Rs2 **sllv** Rd, Rs1, RRs2 **sra** Rd, Rs1, Rs2 **srav** Rd, Rs1, RRs2 **srl** Rd, Rs1, Rs2 **srlv** Rd, Rs1, RRs2	Desplazamiento lógico a la izquierda Desplazamiento lógico a la izquierda variable Desplazamiento lógico a la derecha aritmético Desplazamiento lógico a la derecha variable Desplazamiento lógico a la derecha Desplazamiento lógico a la derecha variable	Desplaza el contenido de Rs1 a la izquierda (derecha) según la distancia indicada por Rs2, y pone el resultado en Rd
sub Rd, Rs1, Rs2 **subu** Rd, Rs1, Rs2	Substracción (con desbordamiento) Substracción (sin desbordamiento)	Pone en Rd la diferencia de los enteros contenidos en Rs1 y Rs2
xor Rd, Rs1, Rs2 **xori** Rd, Rs1, Imm	XOR XOR inmediato	Pone en Rd el XOR lógico de los enteros contenidos en Rs1 y Rs2 (o inmediato)

Manipulación de constantes.

li $t0, 26 *load-integer: Carga en el registro $t0 un valor entero*	Esta instrucción sirve para dar un valor entero a un registro de una forma sencilla. La ejecución de la instrucción de la izquierda produce que $t0 tome el valor 0x0000001a (realmente, no es una instrucción de carga o almacenamiento en memoria, sino de manipulación de constante)

Si bien solo vamos a necesitar la instrucción **li**, a continuación se listan todas las existentes.
Rd es el registro destino, imm un valor inmediato, float un número en punto flotante, e integer un número entero.

Instrucción	Acción	Descripción
li Rd, imm	Cargar valor inmediato (*Load Immediate*)	Carga el valor inmediato *imm* en el registro Rd
li.d FRd, float	Cargar inmediato de doble precisión (*Load Immediate Double*)	Carga el número en punto flotante de doble precisión *float* en los registros de punto flotante FRd y FRd + 1
li.s FRd, float	Cargar inmediato de simple precisión (*Load Immediate Single*)	Carga el número en punto flotante de simple precisión *float* en el registro de punto flotante FRd
lui Rd, integer	Cargar inmediato (parte superior) (*Load Upper Immediate*)	Carga la mitad inferior de la palabra del valor entero en la mitad superior de la palabra del registro Rd. Los restantes bits de menor peso del registro Rd se ponen a 0.

Comparación.

Son instrucciones que sirven para operaciones en las que se comparan los valores de dos registros fuente y, dependiendo del resultado de la comparación, se inicializa a 1 el registro destino.

En todas las siguientes instrucciones:

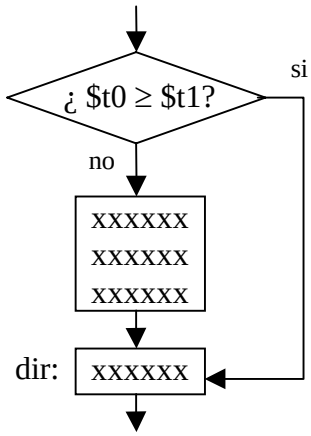
- Rd es el registro destino.
- Rs1 es un registro que hace de primer operando fuente.
- Rs2 es el segundo operando fuente, que puede ser registro o valor entero.
- Imm es un valor inmediato.

Instrucción	Acción	Descripción
seq Rd, Rs1, Rs2	Inicializar si igual (*Set Equal*)	Rd = 1 si Rs1 = Rs2 Rd = 0 en cualquier otro caso
sge Rd, Rs1, Rs2	Inicializar si mayor o igual que (*Set Greater Than Equal*)	Rd = 1 si Rs1 ≥ Rs2 Rd = 0 en cualquier otro caso
sgeu Rd, Rs1, Rs2	Inicializar si mayor o igual que (*Set Greater Than Equal Unsigned*)	Igual que la anterior instrucción, Rs2 es ahora un entero sin signo
sgt Rd, Rs1, Rs2	Inicializar si mayor que (*Set Greater Than*)	Rd = 1 si Rs1 > Rs2 Rd = 0 en cualquier otro caso
sgtu Rd, Rs1, Rs2	Inicializar si mayor que (*Set Greater Than Unsigned*)	Igual que la anterior instrucción, Rs2 es ahora un entero sin signo
sle Rd, Rs1, Rs2	Inicializar si menor o igual que (*Set Less Than Equal*)	Rd = 1 si Rs1 ≤ Rs2 Rd = 0 en cualquier otro caso
sleu Rd, Rs1, Rs2	Inicializar si menor o igual que (*Set Less Than Equal Unsigned*)	Igual que la anterior instrucción, Rs2 es ahora un entero sin signo
slt Rd, Rs1, Rs2	Inicializar si menor que (*Set Less Than*)	Rd = 1 si Rs1 < Rs2 Rd = 0 en cualquier otro caso
slti Rd, Rs1, Imm	Inicializar si menor que (*Set Less Than Immediate*)	Igual que la anterior instrucción, pero con un valor inmediato
sltu Rd, Rs1, Rs2	Inicializar si menor que (*Set Less Than Unsigned*)	Igual que la anterior instrucción, Rs2 es ahora un entero sin signo
sltiu Rd, Rs1, Imm	Inicializar si menor que (*Set Less Than Unsigned Immediate*)	Rd = 1 si Rs1 < Rs2 (o inmediato) Rd = 0 en cualquier otro caso
sne Rd, Rs1, Rs2	Inicializar si distinto que (*Set Not Equal*)	Rd = 1 si Rs1 ≠ Rs2 Rd = 0 en cualquier otro caso

Instrucciones para tomar decisiones.

bge $t0, $t1, dir

Branch on Greather or Equal than:
Salta a ejecutar la instrucción etiquetada por "dir" si el contenido de $t0 es mayor o igual que el contenido de $t1; si no, se ejecuta la siguiente instrucción



- organigrama -

Ejemplo: Supongamos el siguiente código:

```
        li  $t0,3
        li  $t1,4
        bge $t0,$t1,dir
        instrucción1
        instrucción2
dir:    instrucción3
        instrucción4
```

Entonces, después de ejecutar la instrucción bge, se ejecuta la instrucción1 y siguientes. Supongamos ahora este código:

```
        li  $t0,4
        li  $t1,3
        bge $t0,$t1,dir
        instrucción1
        instrucción2
dir:    instrucción3
        instrucción4
```

Entonces, después de ejecutar la instrucción bge, se ejecuta la instrucción3 y siguientes.

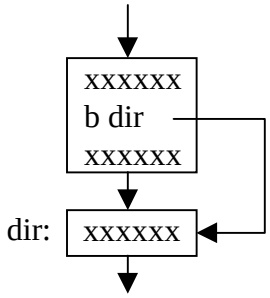
Con un funcionamiento análogo a esta instrucción, están:

beq bifurcar si igual
ble bifurcar si menor o igual
bne bifurcar si distindo
bgt bifurcar si mayor
blt bifurcar si menor

---

b dir (o también **j dir**)

Branch: Salta a ejecutar la instrucción etiquetada por "dir" incondicionalmente



Ejemplo: Supongamos el siguiente código:

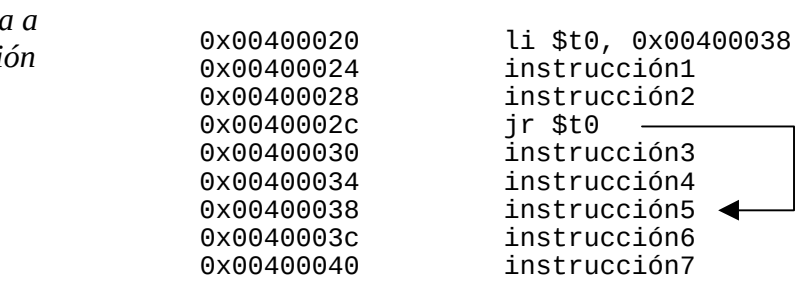
```
        instrucción1
        instrucción2
        b  dir          #también, j  dir
        instrucción3
        instrucción4
dir:    instrucción5
        instrucción6
```

Entonces, no se ejecutarán las instrucciones 3 y 4, pues después de ejecutar la instrucción b, se ejecuta la instrucción5 y siguientes.

jr $t0

Jump Register: Salta a ejecutar la instrucción cuya dirección es el contenido de $t0

Supongamos el siguiente código, donde aparecen las direcciones de cada instrucción:

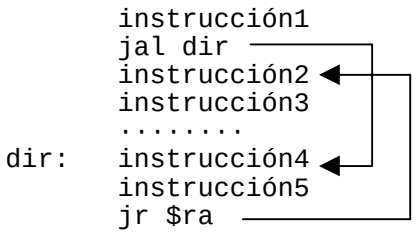


Al ejecutar este código, se ejecutan todas las instrucciones hasta la jr, y después se ejecutan a partir de la instrucción5

jal dir

Jump and Link: Salta a ejecutar la instrucción cuya dirección está etiquetada por "dir", y enlaza (guarda la dirección de la siguiente instrucción en el registro $ra).

Supongamos el siguiente código:



Al ejecutar la instrucción "jal dir", en el registro $ra se guarda la dirección de la siguiente instrucción (instrucción2), y luego salta a ejecutar la instrucción4 y siguientes. Como hemos puesto, en este ejemplo, al final la instrucción "jr $ra", saltaremos a ejecutar la instrucción cuya dirección está guardada en $ra, es decir, la instrucción2, y luego se ejecutan las siguientes.

Las instrucciones de salto o bifurcación, tanto condicional como incondicionales más importantes, y que serán las que se utilizen en las prácticas, son: **b, bge, beq, ble, bne, bgt, blt, j, jal y jr**.

A continuación se listan todas las instrucciones. Antes, tener en cuenta estas consideraciones:

- Los compiladores MIPS utilizan slt, beq, bne y el valor fijo de 0 en $0 para crear todas las condiciones relativas.
- Rs2 puede ser un registro o un valor inmediato (entero). Las instrucciones de salto usan un campo de desplazamiento con signo de 16 bits, por lo que pueden saltar hasta 2^{15}-1 instrucciones (no bytes) hacia delante, o 2^{15} instrucciones hacia detrás. Las instrucciones de salto incondicional contienen un campo de direccionamiento de 26 bits.

Instrucción	Acción	Descripción
b dir	Salto	Salto incondicional a la instrucción etiquetada por dir
bczt dir	Salto al coprocesador z si verdadero	Salto condicional a la instrucción etiquetada por dir si el bit de condición del coprocesador z es verdadero
bczf dir	Salto al coprocesador z si falso	Salto condicional a la instrucción etiquetada por dir si el bit de condición del coprocesador z es vfalso

beq Rs1, Rs2, dir	Salta si =	Salta a la instrucción etiquetada por dir si Rs1 = Rs2
beqz Rs, dir	Salta si = 0	Salta a la instrucción etiquetada por dir si Rs = 0
bge Rs1, Rs2, dir	Salta si ≥	Salta a la instrucción etiquetada por dir si Rs1 ≥ Rs2
bgeu Rs1, Rs2, dir	Salta si ≥ sin signo	Salta a la instrucción etiquetada por dir si Rs1 ≥ Rs2
bgez Rs, dir	Salta si ≥ 0	Salta a la instrucción etiquetada por dir si Rs ≥ 0
bgezal Rs, dir	Salta si ≥ 0 y enlaza	Salta a la instrucción etiquetada por dir si Rs ≥ 0, y guarda la dirección de la siguiente instrucción en $ra
bgt Rs1, Rs2, dir	Salta si >	Salta a la instrucción etiquetada por dir si Rs1 > Rs2
bgtu Rs1, Rs2, dir	Salta si > sin signo	Salta a la instrucción etiquetada por dir si Rs1 > Rs2.
bgtz Rs, dir	Salta si > 0	Salta a la instrucción etiquetada por dir si Rs >0
ble Rs1, Rs2, dir	Salta si ≤	Salta a la instrucción etiquetada por dir si Rs1 ≤ Rs2
bleu Rs1, Rs2, dir	Salta si ≤ sin signo	Salta a la instrucción etiquetada por dir si Rs1 ≤ Rs2
blez Rs, dir	Salta si ≤ 0	Salta a la instrucción etiquetada por dir si Rs ≤ 0
bgezal Rs, dir	Salta si ≥ 0 y enlaza	Salta a la instrucción etiquetada por dir si Rs ≥ 0, y guarda la dirección de la siguiente instrucción en $ra
bltzal Rs, dir	Salta si < y enlaza	Salta a la instrucción etiquetada por dir si Rs < 0, y guarda la dirección de la siguiente instrucción en $ra
blt Rs1, Rs2, dir	Salta si <	Salta a la instrucción etiquetada por dir si Rs1 < Rs2
bltu Rs1, Rs2, dir	Salta si < sin signo	Salta a la instrucción etiquetada por dir si Rs1 < Rs2
bltz Rs, dir	Salta si < 0	Salta a la instrucción etiquetada por dir si Rs < 0
bne Rs1, Rs2, dir	Salta si ≠	Salta a la instrucción etiquetada por dir si Rs1 ≠ Rs2
bnez Rs, dir	Salta si ≠ 0	Salta a la instrucción etiquetada por dir si Rs ≠ 0
j dir	Salta	Salto incondicional a la instrucción cuya etiqueta es dir.
jal dir	Salta y enlaza	Salta a la instrucción etiquetada por dir, y guarda la dirección de la siguiente instrucción en $ra
jalr Rs	Salta y enlaza según registro	Salta a la instrucción cuya dirección está contenida en el registro Rs, y guarda la dirección de la siguiente instrucción en $ra
jr Rs	Salta según registro	Salta a la instrucción cuya dirección está contenida en el registro Rs

Transferencia de Datos.

De todas estas instrucciones, la que nos interesa para las prácticas es **move**.

Instrucción	Acción	Descripción
move Rd, Rs	Mueve	Copia (mueve) el contenido del registro Rs al registro Rd
Cuando se realizan tareas de multiplicar y dividir, el resultado utiliza dos registros adicionales: HI y LO. Las siguientes instrucciones mueven valores hacia y desde estos dos registros. Las instrucciones de multiplicación y división son pseudo-instrucciones que hacen parecer como si se operase con los registros generales y detectasen condiciones de error tales como división por cero o desbordamiento.		
mfhi Rd	Mueve desde HI	
mflo Rd	Mueve desde LO	Mueve el contenido del registro HI (LO) al registro Rd
mthi Rd	Mueve hacia HI	
mtlo Rd	Mueve hacia LO	Mueve el contenido del registro Rd al registro HI (LO)
Los coprocesadores tienen sus propios conjuntos de registros. Las siguientes instrucciones mueven valores entre los registros de la CPU y los registros de los coprocesarores.		
mfcz Rd, Cops	Mueve desde el Coprocesador z	Mueve el contenido del registro Cops del coprocesador z, hacia el registro Rd de la CPU
mfc1.d Rd, FRs1	Mueve desde el Coprocesador 1 un valor de doble precisión	Mueve el contenido de los registros de punto-flotante FRs1 y FRs1+1 hacia los registros Rd y Rd+1 de la CPU
mtcz Rs, Copd	Mueve hacia el Coprocesador z	Mueve el contenido del registro Rs de la CPU hacia el registro Copd del coprocesador z

Excepciones e Interrupciones.

Estas instrucciones están asociadas a las interrupciones que se pueden producir en el ciclo de ejecución de las instrucciones. Tan solo la instrucción **syscall** nos será útil para las prácticas.

Instrucción	Acción	Descripción
rfe	Vuelta desde una excepción (*return from exception*)	Restablece el registro Status
syscall	Llamada al Sistema (*system call*)	El registro **$v0** contiene el número de la llamada al sistema, y dependiendo de este número, el sistema operativo realizará la tarea asociada (imprimir en consola, leer de teclado, abortar la ejecución de un programa, etc).
break	nBreak	Produce la excepción número *n*. La excepción 1 está reservada para el depurador.
nop	Ninguna operación (*no operation*)	No hace nada