

T2.INSTRUCCIONESLENGUAJEMQUINA.r...



Anónimo



Arquitectura de Computadores



2º Grado en Ingeniería Informática



**Escuela Politécnica Superior de Córdoba
Universidad de Córdoba**



[Accede al documento original](#)



Escuela de
Organización
Industrial

Contigo que evolucionas.
Contigo que lideras. Contigo que transformas.

**Esto es EOI.
Mismo propósito,
nueva energía.**



Descubre más aquí



EOI Escuela de
Organización
Industrial

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

perdo
espacio



Necesito
concentración

ali ali ooh
esto con 1 coin me
lo quito yo...

WUOLAH

T2. INSTRUCCIONES: LENGUAJE MÁQUINA. REPERTORIO DE INSTRUCCIONES MIPS

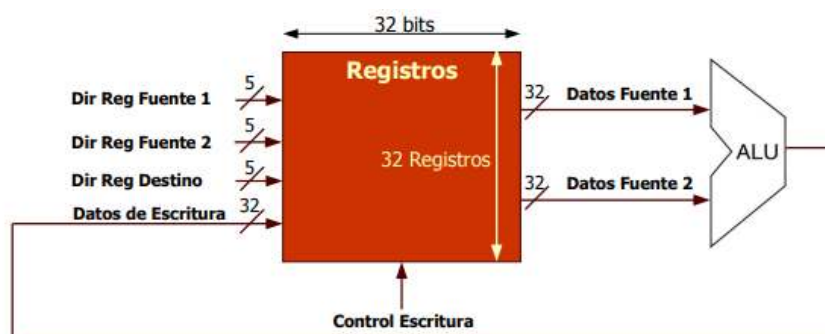
Los primeros computadores tenían un único registro para instrucciones aritméticas, el acumulador. La principal desventaja de este tipo de arquitectura es que todas las variables del programa deben estar ubicadas en memoria. Mientras más accesos a memoria, más lentitud en la ejecución.

El siguiente paso es utilizar un banco de registros de propósito general en donde se encuentren todas las variables.

El procesador MIPS utiliza una arquitectura de registros de propósito general. La CPU contiene un banco de registros compuesto por 32 registros de 32 bits cada uno. Utiliza una arquitectura RISC, es decir, instrucciones de tamaño fijo y presentadas en un reducido número de formatos y de las cuales sólo las de carga y almacenamiento acceden a la memoria de datos.

El banco de registros está formado por 32 registros de 32 bits con dos puertos de lectura y uno de escritura.

Esto proporciona mayor flexibilidad a la hora de realizar las instrucciones.



Operandos en registros

El lenguaje ensamblador MIPS presenta la siguiente notación:

```
add a, b, c #La suma de b y c se pone en a
sub a, b, c #La resta de b y c se pone en a
```

En estos ejemplos las variables se encuentran en registros.

Los registros están numerados de 0 a 31, pero su representación simbólica está formada por dos caracteres precedidos por el símbolo \$:

- \$s0, \$s1, ..., \$s7: son registros utilizados para valores salvados (se corresponden con variables de programas)
- \$t0, \$t1, ..., \$t9: son registros utilizados para valores temporales.
- \$zero : es un registro cuyo valor siempre es 0.

Operandos en memoria

Cuando aparecen variables con estructuras complejas, con un número de elementos superior al número de registros, hay que definir un formato nuevo para poder trabajar con estos datos.

MIPS debe incluir instrucciones que transfieran datos entre memoria y registros. En este caso un registro contiene la dirección base de la tabla, es decir, la dirección de su primer elemento. Este registro es denominado '*registro índice*'.

La memoria principal utilizada por MIPS es de 32 bits por palabra y utiliza 32 bits para ser direccionada. El elemento mínimo direccionable es el byte. Como cada palabra contiene 4 bytes, los dos últimos bits de dirección se encargarán de seleccionar el byte dentro de la palabra.

La instrucción de transferencia que mueve datos de memoria a algún registro se denomina lw (load word).

Su instrucción complementaria, encargada de transferir datos de un registro a memoria, se denomina sw (store word).

```
lw $t0, 32($s3) #El registro $t0 se carga con la dirección $s3+32 que sería V[8]; 32=8*4bytes.
```

Formatos de instrucción

Las instrucciones MIPS son de 32 bits (8 bytes) y están divididas de la siguiente forma, Tipo-R:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Donde:

- op: Código de operación
- rs: Primer registro operando fuente
- rt: Segundo registro operando fuente
- rd: Registro operando destino
- shamt: Tamaño del desplazamiento (SHift AMounT)
- funct: Función. Selecciona la variante específica de la operación op

En las instrucciones lw y sw sólo se usan dos direcciones de registro. Es necesario definir un nuevo formato de instrucción, Tipo-I:

op	rs	rt	dirección
6 bits	5 bits	5 bits	16 bits

El compromiso de simplicidad que requiere que, de haber más de un formato, sean de la misma longitud.

El lenguaje ensamblador MIPS incluye dos instrucciones de toma de decisiones, similares a una sentencia *if* con un *go to*. (branch if equal) (branch if not equal):

```
beq reg1, reg2, L1 #realiza un salto a la sentencia L1 si reg1==reg2.  
bne reg1, reg2, L1 #realiza un salto a la sentencia L1 si reg1!=reg2.
```

Estas instrucciones suelen ir combinadas con una instrucción de comparación set on less than (slt):

```
slt $t0, $s1, $s2 #$t0=1 si $s1<$s2 sino $t0=0.
```

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

perdo
espacio



Instrucciones inmediatas

Debido a que los programas utilizan muchas veces constantes, sería interesante poder utilizarlas de forma eficiente. Para ello se crean versiones de las instrucciones aritméticas en donde uno de los operandos es una constante, que viene codificada en la misma instrucción.

Utilizando el formato Tipo-I, rs sería el registro donde está el primer operando, rt el registro destino y la constante estaría codificada en los 16 bits destinados a la dirección.

```
addi $s0, $s1, 5 # $s0=$s1+5
slti $t2, $s3, 8 # $t2=1 si $s3<8 sino $t2=0
```

Puede ocurrir que queramos utilizar constantes de más de 16 bits. Para ello se utiliza la instrucción lui (load upper immediate). Esta instrucción carga los 16 bits codificados en la propia instrucción en los 16 bits más significativos del registro destino. Así para poder cargar la constante de 32 bits se podrá hacer:

```
lui $s0, 4 # $s0=0000000000000100 + '16 bits de basura'
addi $s0, $s0, 7856 # $s0=0000000000000100 0001111010110000
```

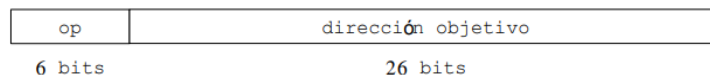
Salto incondicionales

Dentro de un programa, al dividir su curso en varios rumbos, es inevitable tener que saltar a otras partes del programa de forma incondicional.

La instrucción de salto incondicional es j (jump)

```
j Fin # Ir a Fin
```

Como interesa aprovechar el mayor número de bits posible para establecer la dirección de salto, y teniendo en cuenta que ningún registro es utilizado, se plantea el tercer y último formato de instrucción, que es el formato Tipo-J:



Operaciones lógicas

Operaciones de desplazamiento lógico:

```
sll $s0, $s1, 4 # Transfiere el valor de $s1 desplazado 4 bits a la izquierda a $s0  
srl $s0, $s1, 4 # Transfiere el valor de $s1 desplazado 4 bits a la derecha a $s0
```

Operaciones AND lógica y OR lógica:

```
and $t0, $t1, $t2 # $t0 = $t1 & $t2  
or $t0, $t1, $t2 # $t0 = $t1 | $t2
```

Operaciones AND lógica y OR lógica inmediatas:

```
andi $t0, $t1, 0xFF00  
ori $t0, $t1, 0xFF00
```

Carga y almacenamiento de bytes

A la hora de manejar caracteres, MIPS utiliza código ASCII. Esa es la razón por la que el elemento mínimo direccionable sea el byte. Para cargar/almacenar bytes entre registros y memoria, existen instrucciones equivalentes a lw y sw, pero que solamente transfieren un único byte.

```
lb $t0, 2($s1) # Carga un byte de memoria
```

La instrucción lb (load byte) carga el byte seleccionado de memoria y lo transfiere a los 8 bits menos significativos del registro destino.

```
sb $t1, 4($s2) # Guarda un byte en memoria
```

La instrucción sb (store byte) carga los 8 bits menos significativos del registro fuente y los transfiere al byte seleccionado en memoria.

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	 PLAN TURBO	 PLAN PRO	 PLAN PRO+
 Descargas sin publi al mes	10 	40 	80 
 Elimina el video entre descargas			
 Descarga carpetas			
 Descarga archivos grandes			
 Visualiza apuntes online sin publi			
 Elimina toda la publi web			
 Precios Anual <input type="checkbox"/>	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo,
¿Qué nota vas a sacar?



WUOLAH

Instrucciones de llamada a procedimientos

Para llamar a un procedimiento o subrutina, el lenguaje ensamblador MIPS incluye la instrucción `jal` (jump and link).

Esta instrucción salta a la dirección donde continúa el programa, guardando la dirección de retorno.

```
jal DirecciónDelProcedimiento
```

El salto lo efectúa igual que la instrucción `j`. La dirección de retorno (encontrada en el PC) es almacenada en el registro `$ra`, que es utilizado específicamente para esta tarea.

Para poder realizar el salto de retorno, necesitamos una instrucción que realice un salto a la dirección contenida en un registro. Esa instrucción es `jr` (jump registry).

```
jr $t1 # Salta a la dirección contenida en $t1
```

Pseudoinstrucciones

El ensamblador MIPS acepta variaciones de las instrucciones de su repertorio, aunque estas variaciones no estén implementadas en el hardware.

El único coste para poder realizar estas instrucciones es la reserva de un registro `$at`, para ser usado por el ensamblador.

Las más usadas son `li` (load-immediate) que carga una constante de 32 bits en un registro destino:

```
li $s1, 270000 # $s1 = 270000
```

`blt` (branch if less than)

```
blt $s1, $s2, L1 # Salta a L1 si $s1 < $s2
```


Tema-2-Lenguaje-del-computador-M...



CodeWolf



Arquitectura de Computadores



2º Grado en Ingeniería Informática



**Escuela Politécnica Superior de Córdoba
Universidad de Córdoba**



[Accede al documento original](#)



Escuela de
Organización
Industrial

Contigo que evolucionas.
Contigo que lideras. Contigo que transformas.

**Esto es EOI.
Mismo propósito,
nueva energía.**



Descubre más aquí



EOI Escuela de
Organización
Industrial

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo espacio



Necesito concentración

ali ali ooh
esto con 1 coin me
lo quito yo...

WUOLAH

Tema 2: Lenguaje del computador MIPS

➤ Arquitectura con Acumulador:

Las primeras computadoras tenían esta arquitectura implementa ; teniendo un único registro para instrucciones aritméticas. La principal desventaja es que todas las variables del programa deben de estar ubicadas en memoria. Por lo cual ↑ Acceso a memoria ↓ Velocidad en la ejecución.

➤ Procesador MIPS

- Diseñado en la Universidad de Stanford.
- Procesador que utiliza una arquitectura de registros de propósito general .(CPU de 32 registros de 32 bits).
- Utiliza arquitectura RISC:
 - ◆ Instrucciones de tamaño fijo y presentadas en un reducido número de formatos.
 - ◆ Solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.
- El banco de registros está formado por 32 registros de 32 bits:
 - ◆ Dos puertos de lectura.
 - ◆ Un puerto de Escritura
- Proporcionando una mayor flexibilidad a la hora de realizar las instrucciones.
 - ◆ Implicará plantearse nuevos formatos de instrucción.

★ Operandos en registros.

La notación que se utiliza es: (instrucción) registro donde se guarda, registro 1 , registro _2

➤ Instrucciones MIPS:

- **add** a , b , c # $b + c \Rightarrow a$
- **sub** a, b, c # $b - c \Rightarrow a$
- **lw** \$t0 , 4 (\$s1) # Carga lo de la derecha en la izquierda
- **sw** \$t0 , 4 (\$s1) # Guarda lo de la derecha en la izquierda
- **beq** registro 1, registro 2 , L1 # Salta a la sentencia L1 si el valor del reg1 es igual a reg2
- **bne** registro 1, registro 2 , L1 # Salta a la sentencia L1 si el valor del reg1 no es igual a reg2
- **slt** reg resul, reg1, reg2 # $reg_resul = 1$ si $reg1 < reg2$ en caso contrario $reg_resul = 0$;
- **addi** reg guar , reg1 , constantes # $reg\ guar = reg1 + constantes$
- **slti** reg_resul , reg1 ,constantes # $reg_resul = 1$ si $reg1 < constantes$
- **lui**
- **j** dirección # Salta a la dirección . Usa el Tipo-J (Formato)
- ★ **Desplazamientos lógicos**
- **sll** reg1 ,reg2, 4 # Transfiere el valor de reg2 desplazado 4 bits a la izquierda a reg1
- **srl** reg1 ,reg2, 4 # Transfiere el valor de reg2 desplazado 4 bits a la derecha a reg1

WUOLAH

★ **Operaciones AND lógica y OR lógica**

→ **and** reg_result ,reg1 ,reg2 # reg_result = reg1 & reg2

→ **or** reg_result ,reg1 ,reg2 # reg_result = reg1 | reg2

→ **andi**

→ **ori**

→ **lb** \$t0, 4(\$s2) # Carga un byte de memoria (derecha) en la izquierda

→ **sb** \$t0 , 4(\$s2) # Guarda los 8 bits menos significativos del registro fuente

→ **jal** Dirección del Procedimiento # Salta la dirección donde continúa el programa guardando la dirección de retorno en el registro \$ra

→ **jr** \$reg1 # Salta a la dirección contenida en \$reg1

MIPS debe incluir instrucciones que transfieren datos entre memoria y registros . En este caso se guarda la dirección “ base ” .

★ La memoria principal utilizada por MIPS es de 32 bits por palabra y utiliza 32 bits para ser direccionada.

◆ El elemento mínimo direccionable es el byte. Como cada palabra contiene 4 bytes, los dos últimos bits de dirección se encargará de seleccionar el byte dentro de la palabra.

◆ Por tanto, las direcciones consecutivas difieren de 4 .

Las instrucciones MIPS son de 32 bits y están divididas en los siguientes campos: **(TIPO R)**

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Donde:

❖ **op**: Código de operación.

❖ **rs**: Primer registro de operando fuerte

❖ **rt**: Segundo registro de operando fuerte.

❖ **rd**: Registro operando destino.

❖ **shamt** : Tamaño del desplazamiento

❖ **funct** : Función, Selecciona la variable específica de la operación op.

op	rs	rt	dirección
6 bits	5 bits	5 bits	16 bits

Para las instrucciones de lw y sw se define otro formato : **(TIPO I)**

★ **Formato para los saltos incondicionales :**

op	dirección objetivo
6 bits	26 bits

Utilizado para la instrucciones j , jal y jr. **(TIPO J)**