



UNIVERSITAT DE GIRONA  
Escola Politècnica Superior

**MIPS**  
TECHNOLOGIES



**MIPS 32 - R2000/R3000**

**Estructura i Tecnologia de Computadors  
1er ETIS/ETIG**

Jordi Ferrer Plana  
jferrerp@eia.udg.es

Revisió 1.1 07/09/2004 - Escrit en L<sup>A</sup>T<sub>E</sub>X

# 1 Registres

El MIPS R2000/R3000 disposa de 32 registres de propòsit general i de 3 registres especials.

**Registres de propòsit general.** Són 32 registres de 32 bits cadascun accessibles directament des de l'assemblador utilitzant el seu nom de registre o bé el seu número de registre.

<i>Número</i>	<i>Nom</i>	<i>Significat</i>	<i>Utilització/Comentaris</i>
\$0	\$z0	<i>Zero constant</i>	Registre de només lectura que conté la constant 0.
\$1	\$at	<i>Assembler temporary</i>	Reservat per l'assemblador.
\$2	\$v0	<i>Value 0</i>	Utilitzat pel valor de retorn de les funcions.
\$3	\$v1	<i>Value 1</i>	Valor alt en operacions de 64 Bits. També s'utilitza pel número de funció en una crida al sistema ( <i>syscall</i> ).
\$4	\$a0	<i>Argument 0</i>	Primer paràmetre per les crides a subrutines.
\$5	\$a1	<i>Argument 1</i>	Segon paràmetre per les crides a subrutines.
\$6	\$a2	<i>Argument 2</i>	Tercer paràmetre per les crides a subrutines.
\$7	\$a3	<i>Argument 3</i>	Quart paràmetre per les crides a subrutines.
\$8	\$t0	<i>Temporary 0</i>	Registre temporal 0 (NO salvat en les crides).
\$9	\$t1	<i>Temporary 1</i>	Registre temporal 1 (NO salvat en les crides).
\$10	\$t2	<i>Temporary 2</i>	Registre temporal 2 (NO salvat en les crides).
\$11	\$t3	<i>Temporary 3</i>	Registre temporal 3 (NO salvat en les crides).
\$12	\$t4	<i>Temporary 4</i>	Registre temporal 4 (NO salvat en les crides).
\$13	\$t5	<i>Temporary 5</i>	Registre temporal 5 (NO salvat en les crides).
\$14	\$t6	<i>Temporary 6</i>	Registre temporal 6 (NO salvat en les crides).
\$15	\$t7	<i>Temporary 7</i>	Registre temporal 7 (NO salvat en les crides).
\$16	\$s0	<i>Saved Temporary 0</i>	Registre temporal 0 (salvat en les crides).
\$17	\$s1	<i>Saved Temporary 1</i>	Registre temporal 1 (salvat en les crides).
\$18	\$s2	<i>Saved Temporary 2</i>	Registre temporal 2 (salvat en les crides).
\$19	\$s3	<i>Saved Temporary 3</i>	Registre temporal 3 (salvat en les crides).
\$20	\$s4	<i>Saved Temporary 4</i>	Registre temporal 4 (salvat en les crides).
\$21	\$s5	<i>Saved Temporary 5</i>	Registre temporal 5 (salvat en les crides).
\$22	\$s6	<i>Saved Temporary 6</i>	Registre temporal 6 (salvat en les crides).
\$23	\$s7	<i>Saved Temporary 7</i>	Registre temporal 7 (salvat en les crides).
\$24	\$t8	<i>Temporary 8</i>	Registre temporal 8 (NO salvat en les crides).
\$25	\$t9	<i>Temporary 9</i>	Registre temporal 9 (NO salvat en les crides).
\$26	\$k0	<i>Kernel scratch 0</i>	Reservat pel Kernel del Sistema Operatiu.
\$27	\$k1	<i>Kernel scratch 1</i>	Reservat pel Kernel del Sistema Operatiu.
\$28	\$gp	<i>Global Pointer</i>	Punter Global a l'àrea de dades.
\$29	\$sp	<i>Stack Pointer</i>	Punter de la pila.
\$30	\$fp	<i>Frame Pointer</i>	Punter base de la pila (marca l'àmbit).
\$31	\$ra	<i>Return Address</i>	Utilitzat per guardar l'adreça de retorn a les crida.

**Taula de registres de propòsit general del MIPS32**

**Registres especials.** Són 3 registres de 32 bits: Hi, Lo i PC. Els registres Hi i Lo són accessibles directament des de l'assemblador utilitzant el seu nom. El registre PC té un accés implícit a través de les instruccions de control de fluxe.

Número	Nom	Significat	Utilització/Comentaris
-	Hi	<i>High</i>	Conté els 32 bits de més pes en una multiplicació de 64 bits. També conté el residu en una divisió.
-	Lo	<i>Low</i>	Conté els 32 bits de menys pes en una multiplicació de 64 bits. També conté el quocient en una divisió.
-	PC	<i>Program Counter</i>	Registre que conté el comptador de programa.

**Taula de registres especials del MIPS32**

## 2 Assemblador

L'assemblador del MIPS32 és un assemblador de tres adreces que preserva l'ordre natural dels operands. Per exemple, la següent instrucció en **C**

$$a = b + c$$

es tradueix en la següent instrucció assemblador

add a, b, c

No totes les instruccions assemblador del MIPS R2000/R3000 són de 3 adreces ja que, per exemple, hi ha instruccions que no té sentit que tinguin més de dos operands com les instruccions de càrrega i emmagatzemament (*Load* i *Store*):

- **lw** Registre, Adreça de Memòria
- **sw** Registre, Adreça de Memòria

**Arquitectura *Load/Store*.** El MIPS32 utilitza una arquitectura de *Load* i *Store*, és a dir, totes les operacions aritmètiques i lògiques es realitzen entre registres, per tant, per operar variables que estiguin a memòria, cal carregar primer els valors d'aquestes variables als registres utilitzant les instruccions de càrrega, operar amb els registres i, finalment, si cal, emmagatzemar els resultats a memòria amb les instruccions d'emmagatzemament.

**Accés a memòria.** L'arquitectura MIPS32 té una mida de paraula de 32 bits i pot adreçar a *byte*, és a dir, que les paraules de 32 bits consecutives a memòria difereixen en 4 bytes. Cal anar amb compte sobretot amb les instruccions que accedeixen indexadament a memòria a paraules de 32 bits: cal que l'índex s'incrementi de quatre en quatre si es vol accedir a paraules consecutives a memòria. Passa el mateix amb les mitjes paraules de 16 bits: cal que l'índex s'incrementi de dos en dos si es vol accedir a paraules consecutives a memòria.

Les instruccions de càrrega i emmagatzemament que accedeixen a paraules de 32 bits (*lw*, *sw*, etc... ) només poden accedir a adreces múltiples de 4. Les instruccions de càrrega i emmagatzemament que accedeixen a mitjes paraules de 16 bits (*lh*, *sh*, etc...) només poden adreçar posicions de memòria parelles. Les instruccions que accedeixen a bytes (*lb*, *sb*, etc...), poden adreçar qualsevol posició de memòria. Finalment, com que totes les instruccions de MIPS32 ocupen 32 bits, les instruccions de salt (*bne*, *beq*, *j*, etc...) només poden accedir a adreces múltiples de 4. Les instruccions tenen la possibilitat de codificar adreçaments invàlids, per tant, en temps d'execució es poden produir intents d'adreçar posicions de memòria incorrectes. En cas que passi això es produirà una excepció d'adreçament invàlid.

**Endian del MIPS32.** Quant a l'*Endian*, el MIPS32 pot ser *Little Endian* o *Big Endian*, depenent del fabricant que ha dissenyat la màquina basada en el MIPS32. Per exemple, en una màquina *Silicon Graphics (SGI)*, el MIPS32 opera en *Big Endian*.

Suposant una CPU MIPS32 que es troba en una màquina configurada per operar en *Little Endian*, les dades que hi ha a memòria a la figura següent es poden llegir:

1. L'adreça 0000h, conté la paraula formada pels bytes 10h, 11h, 12h i 13h, o sigui, la paraula de 32 bits 13121110h.
2. Si s'accedeix a l'adreça 0004h es pot llegir la paraula formada pels bytes 14h, 15h, 16h i 17h, o sigui, la paraula 17161514h.
3. Si s'accedeix a l'adreça de memòria 000Eh es pot llegir la mitja paraula formada pels bytes 1Eh i 1Fh, o sigui, la mitja paraula de 16 bits 1F1Eh.
4. Si s'accedeix al byte de l'adreça 0009h es pot llegir el byte 19h.

**Memòria**

@0000h	10h	11h	12h	13h
@0004h	14h	15h	16h	17h
@0008h	18h	19h	1Ah	1Bh
@000Ch	1Ch	1Dh	1Eh	1Fh
@0010h	20h	21h	22h	23h
...				
				...

Figura 1: Disposició de la memòria en un MIPS32

### 3 Modes d'adreçament

Quant als modes d'adreçament, el MIPS32 és extremadament senzill: només disposa de quatre modes d'adreçament diferents.

**Adreçament a registre.** Els operands de la instrucció són registres.

**Adreçament immediat.** En aquest cas l'operand es troba en el propi codi d'instrucció.

**Adreçament base o desplaçament.** En aquest mode l'operand es troba en una adreça de memòria que es calcula mitjançant la suma d'un registre més un valor constant donat. Aquest mode d'adreçament també se sol anomenar *adreçament indexat*.

**Adreçament relatiu al PC.** Aquest mode d'adreçament és una variant de l'anterior. L'adreça de memòria on es troba l'operand es calcula mitjançant la suma del PC i un valor constant donat.

### 3.1 Adreçament a registre

En aquest mode d'adreçament els operands de la instrucció es troben en algun dels registres. Aquest és el mode d'adreçament *més ràpid* ja que no cal accedir a memòria per llegir els operands ni tan sols fer cap càlcul per obtenir-los. En un adreçament a registre, cal que en el codi de la instrucció hi hagi codificats els registres amb els quals es vol operar.

En el cas de MIPS32, cal codificar un, dos o tres registres segons el tipus d'instrucció. Com que hi ha 32 registres per identificar, es necessiten 5 bits per codificar cada un dels registres. En les instruccions assemblador que utilitzin 3 registres en dedicaran 15 bits del codi d'instrucció per codificar aquests 3 registres, en les instruccions que només utilitzin 2 registres, es dedicaran 10 bits del codi d'instrucció per codificar aquests 2 registres.

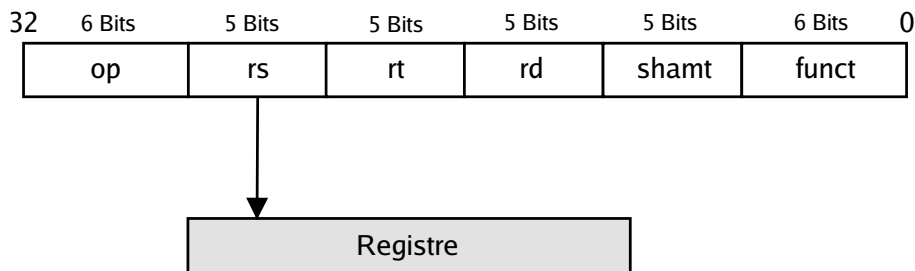


Figura 2: Mode d'adreçament a registre

**Exemple:** `add $1, $0, $0`

Aquesta instrucció suma el contingut del registre \$0 (un zero) amb el contingut del registre \$0 (un zero) i deixa el resultat (zero) al registre \$1. És una forma ràpida de posar un zero en un registre, en aquest cas al registre \$1.

### 3.2 Adreçament immediat

En aquest mode d'adreçament, l'operand de la instrucció es troba codificat en el propi codi d'instrucció. Com que, per a l'execució d'una instrucció, cal que aquesta es carregui al registre d'instrucció (*IR*), aquest mode d'adreçament també és *ràpid* ja que l'operand es troba dins la CPU i no a memòria.

En el cas de MIPS32, el valor immediat ocupa des de 5 bits fins a 26 bits depenent de quina sigui la instrucció que utilitza aquest mode adreçament.

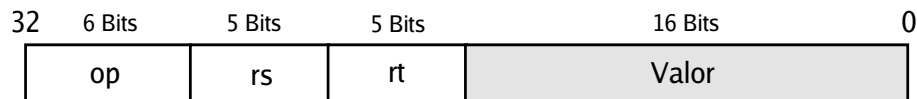


Figura 3: Mode d'adreçament immediat

**Exemple:** `addi $1, $0, 0`

L'execució d'aquesta instrucció produeix el mateix resultat que la de l'exemple anterior: copia un zero al registre \$1 utilitzant una instrucció de suma. En aquest cas, però, se suma el valor del registre \$0 (un zero) amb el valor immediat 0 que es troba codificat al propi codi de la instrucció `addi`.

Cal tenir en compte que la mida de totes les instruccions del MIPS32 és de 32 bits, per tant, els bits dedicats a la codificació dels valors immediats són limitats. En el tipus d'instrucció de l'exemple, es dediquen 16 bits per codificar el valor immediat. En MIPS32 mai es podran carregar valors immediats de la mida de tot un registre de 32 bits amb una sola instrucció, ja que es necessitarien tots els 32 bits que es dediquen a la codificació d'una instrucció i, com a mínim, cal dedicar alguns bits del codi màquina de la instrucció pel codi d'operació.

### 3.3 Adreçament base o desplaçament

En aquest cas, l'operand de la instrucció cal anar-l'ho a buscar a memòria. L'adreça de memòria on cal anar a buscar l'operand s'obté sumant dos valors, un de fixe i un de variable que s'utilitza com a índex. Aquest mode d'adreçament també se sol anomenar adreçament indexat.

Aquest mode, és el que s'utilitza per accedir als valors d'un vector que hi ha memòria: el valor fixe sol ser l'adreça inicial on es troba el vector i, el valor variable, s'utilitza com a índex sumant-l'ho a l'adreça fixe (o base) per obtenir cada un dels elements. Aquest mode d'adreçament és més *lent* que els anteriors, ja que cal anar a cercar a memòria l'operand, per tant, a part del *fetch* de la instrucció cal fer un nou accés a memòria per accedir a l'operand.

En el cas de MIPS32 el valor fixe és un valor immediat codificat en la mateixa instrucció i el valor variable és un registre. Per tant, es pot dir que, en MIPS32, l'adreçament base o desplaçament utilitza dos altres modes d'adreçament implícitament: l'adreçament a registre i l'adreçament immediat.

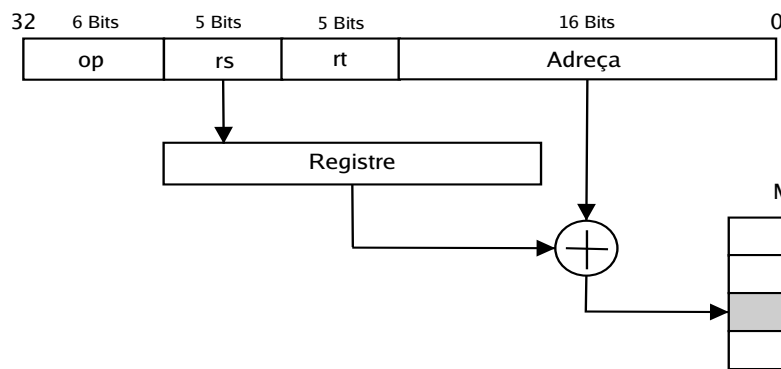


Figura 4: Mode d'adreçament base o desplaçament

**Exemple:** `lw $1, 4($2)`

Aquesta instrucció copia al registre \$1, el valor que hi ha a la posició de memòria indicada per la suma de 4 més el contingut del registre \$2. És a dir, realitza el càlcul  $\$1 = \text{Memòria}[4 + \$2]$ .

### 3.4 Adreçament relatiu al PC

Aquest mode d'adreçament, normalment, s'utilitza en les instruccions de control de fluxe (bifurcacions del programa i salts condicionals). En aquest cas l'operand s'obté de forma indexada (base o desplaçament) utilitzant un valor fixe més el valor variable que es troba en el registre PC (comptador de programa). En el cas que es tracti d'una instrucció de salt el resultat de la suma és una adreça que s'acaba carregant al mateix PC.

En el cas de MIPS32, aquest mode d'adreçament només s'utilitza per instruccions de salt. El valor fixe és un valor immediat codificat en el propi codi d'instrucció. Aquest fet és un límit a l'hora de treballar amb l'assemblador de MIPS32 ja que, un salt relatiu al PC, té una mida màxima de llargada, és a dir, que no pot saltar a qualsevol adreça, sinó que ve limitada pel valor immediat que conté el codi d'instrucció.

Cal anar amb compte que l'usuari, normalment, no treballa amb valors numèrics en el programa en assemblador ja que, l'assemblador, permet definir etiquetes que s'acaben traduint als valors numèrics que corresponen a les *adreces relatives*. En el MIPS32 aquestes *adreces relatives* que s'acaben sumant al PC, es codifiquen com a valors immediats en el codi màquina de les mateixes instruccions.

Aquest mode d'adreçament es pot considerar que és un adreçament base o desplaçament utilitzant el registre PC com a índex, per tant, com que el mode d'adreçament base o desplaçament és una combinació dels dos primers modes d'adreçament (a registre i immediat), el mode d'adreçament relatiu al PC es pot considerar una combinació de tots els tres modes anteriors.

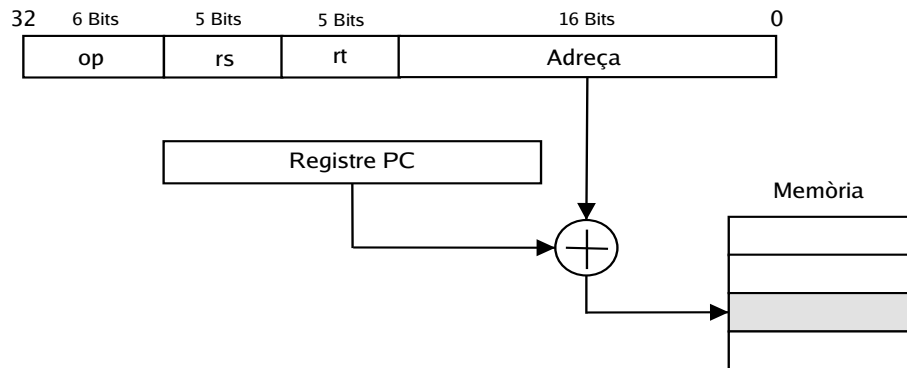


Figura 5: Mode d'adreçament relatiu al PC

**Exemple:** `bne $0, $1, 100`

En aquest exemple, la instrucció executa el salt si el valor del registre \$0 (zero) és diferent del valor del registre \$1. L'adreça destí es calcula a partir del PC més el valor immediat 100. Aquest valor immediat es troba codificat en el propi codi d'instrucció. Com que totes les instruccions de MIPS32 tenen una mida fixe de 32 bits i, en les instruccions de salt relatiu al PC, cal codificar-hi dos registres (que ocupen 5 bits cada un), només queda un espai de 16 bits per codificar l'etiqueta destí.

A més a més, com que cal poder saltar a una adreça posterior o anterior a la instrucció de salt, el valor de 16 bits, és un valor expressat en complement a 2 que permet indicar un *offset* (desplaçament) des del PC cap endavant (valor positiu) o cap endarrera (valor negatiu). Aprofitant que qualsevol instrucció de MIPS32 ocupa 32 bits, l'*offset* es calcula en número d'instruccions i no

en bytes, així es pot saltar a adreces quatre vegades més lluny ja que, el valor immediat (*l'offset*) que hi ha codificat en complement a 2 en el codi d'instrucció, es multiplica per quatre abans de sumar-lo al PC. Cal tenir en compte que, realment, el desplaçament no es calcula a partir del valor del PC a l'adreça de la instrucció de salt, sinó que es calcula a partir del valor del PC a la següent instrucció ( $PC + 4$ ) ja que, el PC, s'autoincrementa a cada *fetch* d'una instrucció.

## 4 Codificació de les instruccions

La senzillesa del repertori d'instruccions del MIPS, juntament amb el fet que totes les instruccions ocupen la mateixa mida d'instrucció (32 bits), fa que hi hagi més d'una forma de codificar i d'interpretar els codis d'instrucció de MIPS32. Cada una d'aquestes formes de codificar i d'interpretar les instruccions és un tipus d'instrucció de MIPS32. Cada tipus d'instrucció té un **format** d'instrucció associat. Mitjançant aquests formats d'instrucció es defineixen una sèrie de **camp**s que ocupen un determinat número de bits dins el codi d'instrucció. Aquests camps poden variar en número, en significat i en el número de bits que ocupen segons cada un dels diferents formats d'instrucció.

En MIPS32 hi ha 3 tipus de formats d'instrucció: instruccions del tipus **R**, instruccions del tipus **I** i instruccions del tipus **J**. Tots els formats d'instrucció conserven un camp comú: el camp *codi d'operació*. Aquest camp serveix a la unitat de control del MIPS32 per decidir de quin tipus d'instrucció es tracta i, per tant, amb quin format d'instrucció cal interpretar la resta d'instrucció. El camp *codi d'operació* que es troba a tots tres formats d'instrucció del MIPS32, sempre es troba representat als 6 bits de més pes de totes les instruccions.

Els diferents tipus d'instruccions o formats d'instrucció del repertori del MIPS32 corresponen, *més o menys*, als diferents modes d'adreçament que s'han definit a l'apartat anterior.

## 4.1 Format R d'instrucció

Aquestes instruccions corresponen al subconjunt d'instruccions del repertori que utilitzen bàsicament adreçaments a registre. Aquest format d'instrucció conté camps que varien entre 5 i 6 bits.

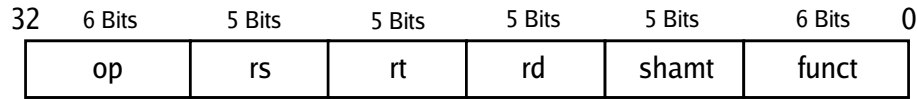


Figura 6: Format de les instruccions del tipus **R**

**Camp *op*** Conté la codificació del codi d'operació genèric.

**Camp *rs*** Conté la codificació del registre que correspon al primer operand font de la instrucció.

**Camp *rt*** Conté la codificació del registre que correspon al segon operand font de la instrucció.

**Camp *rd*** Conté la codificació del registre que correspon a l'operand destí de la instrucció.

**Camp *shamt*** Conté un valor que indica la quantitat de desplaçament. S'utilitza en algunes instruccions com, per exemple, les de desplaçament de bits a l'esquerre o a la dreta. En les instruccions que no s'utilitza aquest camp, es posa a 0.

**Camp *funció*** Aquest camp indica una funció dins la operació genèrica codificada en el primer camp (o codi d'operació). Per exemple, la suma (*add*) i la diferència (*sub*) són la mateixa operació (amb codi 0) però funcions diferents dins aquesta operació (la 32 i la 34 respectivament).

**Exemple:** *add \$1, \$2, \$3*

Aquesta instrucció suma el valor del registre \$2 més el valor del \$3 i deixa el resultat al registre \$1.

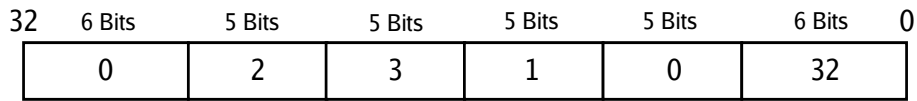


Figura 7: Codificació de la instrucció *add \$1, \$2, \$3*

**Exemple:** *sub \$1, \$2, \$3*

Aquesta instrucció resta el valor del registre \$3 al valor del registre \$2 i deixa el resultat al registre \$1.

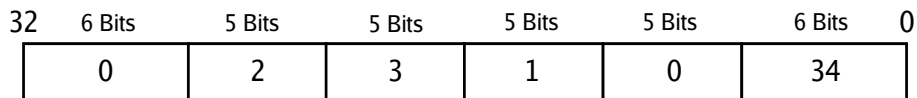


Figura 8: Codificació de la instrucció *sub \$1, \$2, \$3*

**Exemple:** *slt \$1, \$2, \$3*

Aquesta instrucció compara el valor del registre \$2 amb el valor del registre \$3. Si el valor que conté el registre \$2 és més petit que el valor que conté el registre \$3 s'assignarà un 1 al registre \$1, en cas contrari s'hi assignarà un 0.

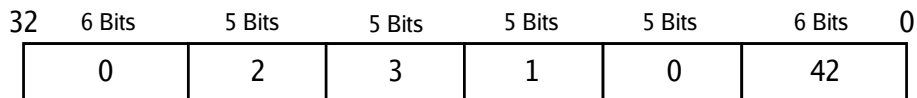


Figura 9: Codificació de la instrucció *slt \$1, \$2, \$3*



**Exemple:** *addi \$1, \$2, 100*

Aquesta instrucció suma el valor contingut en el registre \$2 més el valor 100 i deixa el resultat al registre \$1.

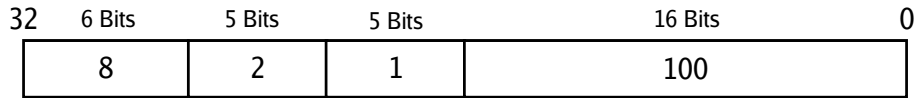


Figura 11: Codificació de la instrucció *addi \$1, \$2, 100*

**Exemple:** *lw \$1, 100(\$2)*

Aquesta instrucció copia al registre \$1 els 32 bits continguts a l'adreça de memòria obtinguda de la suma de 100 més el valor contingut en el registre \$2. És a dir, calcula  $\$1 = \text{Memòria}[100 + \$2]$ .

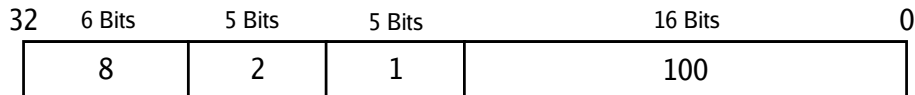


Figura 12: Codificació de la instrucció *lw \$1, 100(\$2)*

**Exemple:** *slti \$1, \$2, 100*

Aquesta instrucció compara el valor contingut en el registre \$2 amb 100. Si el valor contingut en el registre \$2 és més petit que 100, es copia un 1 al registre \$1, en cas contrari es copia un 0 al registre \$1.

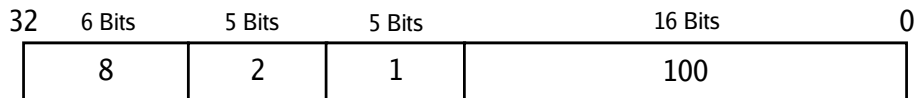


Figura 13: Codificació de la instrucció *slti \$1, \$2, 100*

**Exemple:** *bne \$1, \$2, 100*

Aquesta instrucció salta a l'adreça obtinguda de sumar  $100 * 4$  al PC, en cas que el valor contingut en el registre \$1 sigui diferent al valor contingut en el registre \$2. En cas contrari s'executarà l'instrucció que hi ha immediatament després de la instrucció de salt.

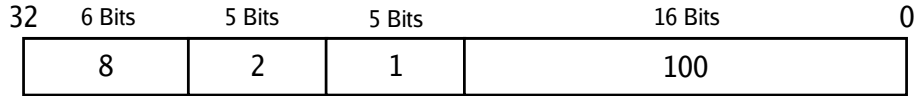


Figura 14: Codificació de la instrucció *bne \$1, \$2, 100*

Existeixen altres adreçaments immediats que s'ajusten al format **R** d'instrucció. Per exemple, les instruccions de desplaçament de bits (*sll, srl, etc...*), utilitzen el camp **shamp** com a operand font en valor immediat.

**Exemple:** *srl \$1, \$2, 10*

Aquesta instrucció desplaça 10 bits cap a la dreta el contingut del registre \$2 i deixa el resultat al registre \$1. En aquesta instrucció es pot comprovar que no s'utilitza el camp *rs* i en el seu lloc s'utilitza el camp *shamp*.

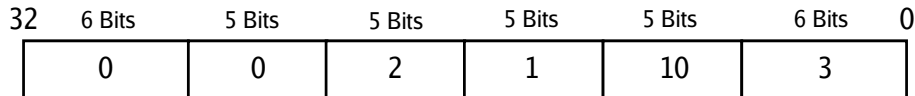


Figura 15: Codificació de la instrucció *srl \$1, \$2, 10*

### 4.3 Format J d'instrucció

Aquest format d'instrucció s'utilitza en algunes instruccions de salt. En aquest cas només existeixen dos camps en el format d'instrucció: el codi d'operació i una adreça destí. Les instruccions de salt que utilitzen aquest format d'instrucció, l'adreça especificada en el codi d'instrucció és una adreça absoluta, és a dir, que indica directament el valor que cal carregar al PC i, per tant, serà l'adreça de memòria on s'anirà a executar codi.

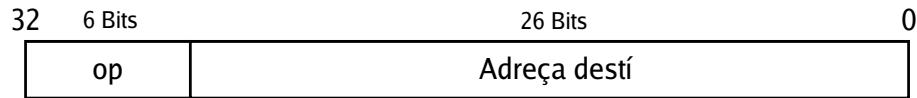


Figura 16: Format de les instruccions del tipus J

**Camp *op*** Conté la codificació del codi d'operació genèric.

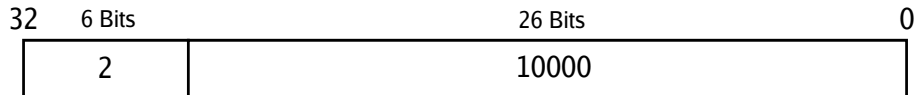
**Camp *Adreça destí*** Conté la codificació d'un valor immediat de 26 bits que representa una adreça absoluta de memòria.

Tal i com passa en el cas dels salts relatius al PC (que utilitzen el format d'instrucció de tipus *I*) l'adreça indicada no és un valor en bytes, sinó que és una quantitat en paraules de 32 bits, és a dir, que cal multiplicar per quatre el valor que hi ha al camp d'*adreça destí* que indica una posició de memòria.

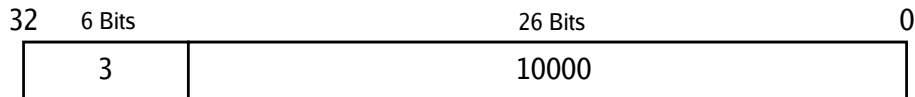
En aquest cas, l'adreça és *absoluta*: no cal sumar-la al PC ni a cap altre registre. Per tant, el valor que hi ha codificat dins el camp d'*adreça destí* s'interpreta com un valor sense signe que indica una posició absoluta de memòria. Com que l'adreça destí es multiplica per quatre, és com si es codifiqués un valor de 28 bits per l'adreça. El registre PC, que és on s'acaba copiant el valor immediat, té 32 bits, per tant, hi ha 4 bits del PC (els de més pes), que no es modifiquen. Les instruccions del tipus **J**, només actualitzen els bits 2, 3, 4, ..., 26, 27 del PC, és a dir, els salts absoluts del MIPS32, no poden abarcar qualsevol posició de memòria direccionable per la CPU, sinó que només poden abarcar les  $2^{28}$  posicions més properes al PC amb els seus 4 bits de més pes ja fixats (no modificant del valor que ja tenen) i els 2 de menys pes a 0.

**Exemple:** *j 10000*

Aquesta instrucció salta incondicionalment a l'adreça fixe que s'obté assignant el valor immediat 10000 als bits 2, 3, 4, ..., 26, 27 del PC, deixant tal i com es troben els bits 28, 29, 30 i 31 del PC i a 0 els bits de menys pes 0 i 1. És a dir, es realitza l'operació  $PC = (PC \text{ and } 0xF0000000) \text{ or } (10000 * 4)$ .

Figura 17: Codificació de la instrucció *j 10000***Exemple:** *jal 10000*

Aquesta instrucció de salt és idèntica a l'anterior però, a més a més, abans de realitzar el salt incondicional, es guarda el valor de  $PC + 4$  al registre \$31, és a dir, es guarda l'adreça de retorn. Aquesta instrucció és l'equivalent al mnemònic *CALL* que se sol utilitzar per indicar les crides a subrutines.

Figura 18: Codificació de la instrucció *jal 10000*

## 5 Repertori d'instruccions

El repertori d'instruccions d'una CPU, és el conjunt d'instruccions que la CPU interpreta en llenguatge màquina i que, per tant, estan definides en l'assemblador. Les instruccions de MIPS32 estan formades per un mnemònic que identifica el tipus d'operació més els seus 0, 1, 2 o 3 operands, segons el tipus d'instrucció. Els mnemònics de les instruccions tenen significat, normalment són lexemes del nom l'operació en llenguatge natural (normalment anglès). Per exemple, la instrucció MIPS32 **beq**, està formades per les inicials de **B**ranch **E**qual.

El repertori d'instruccions, així com tota l'arquitectura de MIPS32 segueix unes pautes fixes de disseny. Per exemple, en el conjunt d'operacions aritmètiques i lògiques, existeixen les típiques instruccions que operen amb registres (adreçament a registre); existeixen aquestes mateixes instruccions, però acabades amb '**i**' que indica que utilitzen un adreçament immediat, i a més a més si incorporen una '**u**' indica que són operacions sense signe. Per exemple, una suma pot ser:

**add** Suma registres, tenint en compte el signe i l'*overflow*.

**addi** Suma registres amb un valor immediat tenint en compte el signe.

**addu** Suma registres sense signe.

**addiu** Suma sense signe amb un valor immediat.

Per definir les instruccions dels següents apartats s'utilitza la següent simbologia:

**rd, rs, rt** Indiquen algun dels 32 (\$0-\$31) registres de propòsit general. Si la instrucció és de 3 operands, *rd* és el registre destí i, *rs* i *rt* són operands font o part d'aquests.

**Hi, Lo** Indica un dels registres especials *Hi* i *Lo*.

**Hi:Lo** Paraula de 64 bits formada pels 32 bits del registre **Hi** i els 32 bits del registre **Lo**. El registre **Hi** serà la part alta i el **Lo** la part baixa.

**imm** Indica un valor immediat.

**s** El subíndex *s* (*Reg<sub>s</sub>* o *imm<sub>s</sub>*) indica un valor amb signe (*signed*).

**u** El subíndex *u* (*Reg<sub>u</sub>* o *imm<sub>u</sub>*) indica un valor sense signe (*unsigned*).

(*Reg*) Els parèntesi indiquen el contingut del registre *Reg*.

$0^n$  Indica una seqüència de *n* zeros.

$1^n$  Indica una seqüència de *n* uns.

*bitseq<sub>1</sub>||bitseq<sub>2</sub>* Indica la concatenació de les dues seqüències de bits *bitseq<sub>1</sub>* i *bitseq<sub>2</sub>*.

*cnt<sub>1</sub>(Reg)* Recompte dels uns del registre *Reg*.

*cnt<sub>0</sub>(Reg)* Recompte dels zeros del registre *Reg*.

*ImmReg<sub>MSB..LSB</sub>* Indica el rang de bits [*MSB..LSB*] del valor immediat o registre *ImmReg*.

**&** And lògica bit a bit.

**|** Or lògica bit a bit.

**xor** Xor lògica bit a bit.

*div* Divisió entera.

*mod* Residu d'una divisió entera.

$\sim$  Not lògica bit a bit.

*ImmReg*  $\ll n$  Desplaçar  $n$  bits a l'esquerre el valor immediat o registre *ImmReg*.

*ImmReg*  $\gg n$  Desplaçar  $n$  bits a la dreta el valor immediat o registre *ImmReg*.

**Nota.** En el cas d'operar aritmèticament dos valors de diferent mida en bits, per exemple, sumar un registre de 32 bits amb un valor immediat de 16 bits,

*addi \$1, \$2, 10*

cal *extendre el signe* del valor que té menys bits.

Per exemple, per sumar 1 (expressat en 32 bits en complement a dos) i -2 (expressat en 16 bits en complement a dos):

0000 0000 0000 0000 0000 0000 0000 0001<sub>b</sub>

i

1111 1111 1111 1110<sub>b</sub>

cal *extendre el signe* del -2 (el valor de 16 bits) fins a 32 bits.

*Extendre el signe* equival a repetir per l'esquerre, el bit de signe del valor que té menys bits, tantes vegades com sigui necessari, fins a arribar a la mida del valor que té més bits. Per exemple:

*ExtendreSigne*<sub>32</sub>(1111 1111 1111 1110<sub>b</sub>) = 1111 1111 1111 1111 1111 1111 1111 1110<sub>b</sub>

Si el valor de 16 bits fos 2 enlloc de -2, el resultat seria

*ExtendreSigne*<sub>32</sub>(0000 0000 0000 0010<sub>b</sub>) = 0000 0000 0000 0000 0000 0000 0000 0010<sub>b</sub>

Aquesta operació cal que la realitzi el computador abans del càlcul aritmètic per tal que s'interpreti correctament el valor de menys bits representat en complement a 2. Si no es realitzés aquesta operació i, per exemple, es consideressin zeros els bits que falten per l'esquerre al valor de menys bits, un valor negatiu de 16 bits passaria a ser positiu en 32 bits. Si, per altra banda, es considerés que fossin uns, un valor positiu de 16 bits, passaria a ser negatiu en 32 bits.

En el cas de les instruccions que operen sense signe, i amb valors de diferent mida en bits com, per exemple,

*addiu \$1, \$2, 10*

es consideren zeros els bits que falten per l'esquerre al valor que té menys bits.

## 5.1 Instruccions aritmètiques

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
add	<b>Add</b> Word	Sumar paraules.
addi	<b>Add Immediate</b> Word	Sumar amb un valor immediat.
addiu	<b>Add Immediate Unsigned</b> Word	Sumar amb un valor immediat sense signe.
addu	<b>Add Unsigned</b> Word	Sumar paraules sense signe.
clo	<b>Count Leading Ones</b> in Word	Comptar els uns d'una paraula.
clz	<b>Count Leading Zeros</b> in Word	Comptar els zeros d'una paraula.
div	<b>Divide</b> Word	Dividir paraules.
divu	<b>Divide Unsigned</b> Word	Dividir paraules sense signe.
madd	Multiply and <b>Add</b> Word to Hi:Lo	Multiplicar i sumar una paraula a Hi:Lo.
maddu	Multiply and <b>Add Unsigned</b> Word to Hi:Lo	Multiplicar i sumar una paraula sense signe a Hi:Lo.
msub	Multiply and <b>Subtract</b> Word to Hi:Lo	Multiplicar i restar una paraula a Hi:Lo.
msubu	Multiply and <b>Subtract Unsigned</b> Word to Hi:Lo	Multiplicar i restar una paraula sense signe a Hi:Lo.
mul	Multiply Word to GPR	Multiplicar paraules amb signe i obtenir els 32 bits de menys pes.
mult	Multiply Word	Multiplicar paraules amb signe.
multu	Multiply <b>Unsigned</b> Word	Multiplicar paraules sense signe.
slt	<b>Set on Less Than</b>	Comparació de més petit.
slti	<b>Set on Less Than Immediate</b>	Comparació de més petit amb un valor immediat.
sltiu	<b>Set on Less Than Immediate Unsigned</b>	Comparació de més petit amb valor immediat sense signe.
sltu	<b>Set on Less Than Unsigned</b>	Comparació de més petit sense signe.
sub	<b>Subtract</b> Word	Restar paraules.
subu	<b>Subtract Unsigned</b> Word	Restar paraules sense signe.

Taula de descripció de les instruccions aritmètiques del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
add	R	add rd, rs, rt	$rd \leftarrow (rs)_s + (rt)_s$	add \$1, \$2, \$3	Sumar paraules amb signe.
addi	I	addi rt, rs, imm	$rt \leftarrow (rs)_s + imm_s$	addi \$1, \$2, 100	Sumar paraula amb constant amb signe.
addiu	I	addiu rt, rs, imm	$rt \leftarrow (rs)_u + imm_u$	addiu \$1, \$2, 100	Sumar paraula amb constant sense signe.
addu	R	addu rd, rs, rt	$rd \leftarrow (rs)_u + (rt)_u$	addu \$1, \$2, \$3	Sumar paraules sense signe.
clo	R	clo rs, rt	$rs \leftarrow cnt_1(rt)$	clo \$1, \$2	Comptar els uns.
clz	R	clz rs, rt	$rs \leftarrow cnt_0(rt)$	clz \$1, \$2	Comptar els zeros.
div	R	div rs, rt	$Lo \leftarrow (rs)_s \text{ div } (rt)_s$ $Hi \leftarrow (rs)_s \text{ mod } (rt)_s$	div \$2, \$3	Divisió amb signe. Lo = quocient, Hi = reste.
divu	R	divu rs, rt	$Lo \leftarrow (rs)_u \text{ div } (rt)_u$ $Hi \leftarrow (rs)_u \text{ mod } (rt)_u$	divu \$2, \$3	Divisió sense signe. Lo = quocient, Hi = reste.
madd	R	madd rs, rt	$(Hi : Lo) \leftarrow (Hi : Lo)_s + ((rs)_s * (rt)_s)_s$	madd \$1, \$2	Multiplicació i suma de 64 bits amb signe. <sup>1</sup>
maddu	R	maddu rs, rt	$(Hi : Lo) \leftarrow (Hi : Lo)_u + ((rs)_u * (rt)_u)_u$	maddu \$1, \$2	Multiplicació i suma de 64 bits sense signe. <sup>1</sup>
msub	R	msub rs, rt	$(Hi : Lo) \leftarrow (Hi : Lo)_s - ((rs)_s * (rt)_s)_s$	msub \$1, \$2	Multiplicació i resta de 64 bits amb signe. <sup>1</sup>
msubu	R	msubu rs, rt	$(Hi : Lo) \leftarrow (Hi : Lo)_u - ((rs)_u * (rt)_u)_u$	msubu \$1, \$2	Multiplicació i resta sense signe de 64 bits. <sup>1</sup>

Taula A d'instruccions aritmètiques del MIPS32

---

<sup>1</sup>A **Hi** hi haurà la part alta i a **Lo** la part baixa.

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
mul	R	mul rd, rs, rt	$rd \leftarrow (rs_s * rt_s)_{31..0}$	mul \$1, \$2, \$3	Producte amb signe. 32 bits de menys pes a rd.
mult	R	mult rs, rt	$(Hi : Lo) \leftarrow rs_s * rt_s$	mult \$2, \$3	Producte amb signe. HI = 32 bits de més pes.
multu	R	multu rs, rt	$(Hi : Lo) \leftarrow rs_u * rt_u$	multu \$2, \$3	Producte sense signe. HI = 32 bits de més pes.
slt	R	slt rd, rs, rt	$rd \leftarrow rs_s < rt_s$	slt \$1, \$2, \$3	Resultat és 1 si rs < rt, altrament 0.
slti	I	slti rt, rs, imm	$rt \leftarrow rs_s < imm_s$	slti \$1, \$2, 100	Resultat és 1 si rs < immediat, altrament 0.
sltiu	I	sltiu rt, rs, imm	$rt \leftarrow rs_u < imm_u$	sltiu \$1, \$2, 100	Resultat és 1 si rs < immediat, altrament 0.
sltu	R	sltu rd, rs, rt	$rd \leftarrow rs_u < rt_u$	sltu \$1, \$2, \$3	Resultat és 1 si rs < rt, altrament 0.
sub	R	sub rd, rs, rt	$rd \leftarrow rs_s - rt_s$	sub \$1, \$2, \$3	Diferència entre paraules amb signe.
subu	R	subu rd, rs, rt	$rd \leftarrow rs_u - rt_u$	subu \$1, \$2, \$3	Diferència entre paraules sense signe.

Taula B d'instruccions aritmètiques del MIPS32

## 5.2 Instruccions de salt

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
b	Unconditional <b>B</b> ranch	Salt relatiu incondicional.
bal	<b>B</b> ranch <b>A</b> nd <b>L</b> ink	Salt relatiu incondicional guardant l'adreça de retorn.
beq	<b>B</b> ranch on <b>E</b> qual	Saltar si igual.
bgez	<b>B</b> ranch on <b>G</b> reater Than or <b>E</b> qual to <b>Z</b> ero.	Saltar si més gran o igual que zero.
bgezal	<b>B</b> ranch on <b>G</b> reater Than or <b>E</b> qual to <b>Z</b> ero and <b>L</b> ink.	Saltar si és més gran o igual que zero guardant l'adreça de retorn.
bgtz	<b>B</b> ranch on <b>G</b> reater Than <b>Z</b> ero	Saltar si més gran que zero.
blez	<b>B</b> ranch on <b>L</b> ess Than or <b>E</b> qual to <b>Z</b> ero	Saltar si més petit o igual que zero.
bltz	<b>B</b> ranch on <b>L</b> ess Than <b>Z</b> ero	Saltar si més petit que zero.
bltzal	<b>B</b> ranch on <b>L</b> ess Than <b>Z</b> ero and <b>L</b> ink	Saltar si més petit que zero guardant l'adreça de retorn.
bne	<b>B</b> ranch on <b>N</b> ot <b>E</b> qualv	Saltar si no igual.
j	<b>J</b> ump	Saltar incondicionalment.
jal	<b>J</b> ump <b>A</b> nd <b>L</b> ink	Saltar guardant l'adreça de retorn.
jalr	<b>J</b> ump <b>A</b> nd <b>L</b> ink <b>R</b> egister	Saltar mitjançant registre guardant l'adreça de retorn.
jr	<b>J</b> ump <b>R</b> egister	Saltar mitjançant registre.

Taula de descripció de les instruccions de salt del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
b	I	b imm	$PC \leftarrow PC + imm_s * 4$	b 100	Salt relatiu incondicional. Equival a <i>beq \$0, \$0, imm</i> .
bal	I	bal rs, imm	$rs \leftarrow PC + 4$ $PC \leftarrow PC + imm_s * 4$	bal \$31, 100	Salt relatiu guardant adreça de retorn. <sup>1</sup>
beq	I	beq rs, rt, imm	<b>Si</b> $(rs == rt) \rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$	beq \$1, \$2, 100	Salt relatiu si igual. <sup>1</sup>
bgez	I	bgez rs, imm	<b>Si</b> $(rs_s \geq 0) \rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$	bgez \$31, 100	Salt relatiu si més gran o igual que zero. <sup>1</sup>
bgezal	I	bgezal rs, imm	$\$31 \leftarrow PC + 4$ <b>Si</b> $(rs_s \geq 0) \rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$	bgezal \$1, 100	Salt relatiu si més gran o igual que zero guardant adreça de retorn. <sup>1</sup>
bgtz	I	bgtz rs, imm	<b>Si</b> $(rs_s > 0) \rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$	bgtz \$1, 100	Salt relatiu si més gran que zero. <sup>1</sup>
blez	I	blez rs, imm	<b>Si</b> $(rs_s \leq 0) \rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$	blez \$1, 100	Salt relatiu si més petit o igual que zero. <sup>1</sup>
bltz	I	bltz rs, imm	<b>Si</b> $(rs_s < 0) \rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$	bltz \$1, 100	Salt relatiu si més petit que zero. <sup>1</sup>

Taula A d'instruccions de salt del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
bltzal	I	bltzal rs, imm	$\$31 \leftarrow PC + 4$ <b>Si</b> ( $rs_s < 0$ ) $\rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$	bltzal \$1, 100	Salt relatiu si més petit que zero guardant adreça de retorn. <sup>1</sup>
bne	I	bne rs, rt, imm	<b>Si</b> ( $rs <> rt$ ) $\rightarrow$ $PC \leftarrow PC + imm_s * 4$ <b>Altrament</b> $PC \leftarrow PC + 4$		Salta relatiu si diferent. <sup>1</sup>
j	J	j imm	$PC \leftarrow (PC \& F0000000_h)   (imm_s * 4)$	j 100	Saltar a adreça absoluta. <sup>2</sup> Saltar a adreça absoluta guardant adreça de retorn a \$31. <sup>2</sup>
jal	J	jal imm	$\$31 \leftarrow PC + 4$ $PC \leftarrow (PC \& F0000000_h)   (imm_s * 4)$	jal 100	
jalr	J	jalr rd, rs	$rd \leftarrow PC + 4$ $PC \leftarrow rs$	jalr \$31, \$1	Salt incondicional a l'adreça continguda en el registre salvant adreça de retorn.
jr	J	jr rs	$PC \leftarrow rs$	jr \$1	Salt incondicional a l'adreça continguda en el registre.

Taula B d'instruccions de salt del MIPS32

<sup>1</sup>El valor constant del salt relatiu és un valor immediat de 16 bits codificat en el propi codi d'instrucció i representat en complement a 2.

<sup>2</sup>El valor del salt és un valor immediat de 26 bits codificat en el propi codi d'instrucció i representat sense signe.

### 5.3 Instruccions de control

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
nop	No <b>O</b> peration	Instrucció que no fa res.

**Taula de descripció de les instruccions de control del MIPS32**

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
nop	R	nop		nop	No fa res.

**Taula d'instruccions de control del MIPS32**

### 5.4 Instruccions de càrrega

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
lb	Load <b>B</b> yte	Carregar un byte amb signe.
lbu	Load <b>B</b> yte <b>U</b> nsigned	Carregar un byte sense signe.
lh	Load <b>H</b> alfword	Carregar mitja paraula (16 bits).
lhu	Load <b>H</b> alfword <b>U</b> nsigned	Carregar mitja paraula sense signe (16 bits).
lui	Load <b>U</b> pper <b>I</b> mmEDIATE	Carregar un valor immediat a la part alta.
lw	Load <b>W</b> ord	Carregar una paraula.

**Taula de descripció de les instruccions de càrrega del MIPS32**

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
lb	I	lb rt, imm(\$rs)	$rt \leftarrow MemByte_s[rs_u + imm_s]$	lb \$1, 100(\$2)	Càrrega d'un byte amb signe amb indexat.
lbu	I	lbu rt, imm(\$rs)	$rt \leftarrow MemByte_u[rs_u + imm_s]$	lbu \$1, 100(\$2)	Càrrega d'un byte sense signe amb adreçament indexat.
lh	I	lh rt, imm(\$rs)	<b>Si</b> $(rs_u + imm_s) \bmod 2 == 0 \longrightarrow$ $rt \leftarrow MemHalfword_s[rs_u + imm_s]$ <b>Altrament</b> <i>Excepció</i>	lh \$1, 100(\$2)	Càrrega de 16 bits amb signe amb adreçament indexat. L'adreça final ha de ser parell. .
lhu	I	lhu rt, imm(\$rs)	<b>Si</b> $(rs_u + imm_s) \bmod 2 == 0 \longrightarrow$ $rt \leftarrow MemHalfword_u[rs_u + imm_s]$ <b>Altrament</b> <i>Excepció</i>	lhu \$1, 100(\$2)	Càrrega de 16 bits sense signe amb adreçament indexat. L'adreça final ha de ser parell. .
lui	I	lui rt, imm	$rt \leftarrow imm_u    0^{16}$	lui \$1, 100	Desplaçar immediat 16 bits a l'esquerre i escriure al registre.
lw	I	lw rt, imm(\$rs)	<b>Si</b> $(rs_u + imm_s) \bmod 4 == 0 \longrightarrow$ $rt \leftarrow MemWord[rs_u + imm_s]$ <b>Altrament</b> <i>Excepció</i>	lw \$1, 100(\$2)	Càrrega d'una paraula amb adreçament indexat. L'adreça final ha de ser múltiple de 4.

Taula d'instruccions de càrrega del MIPS32

## 5.5 Instruccions d'emmagatzemament

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
sb	<b>S</b> to <b>r</b> e <b>B</b> yte	Emmagatzemar un byte.
sh	<b>S</b> to <b>r</b> e <b>H</b> alfword	Emmagatzemar mija paraula (16 bits).
sw	<b>S</b> to <b>r</b> e <b>W</b> ord	Emmagatzemar una paraula.

**Taula de descripció de les instruccions d'emmagatzemament del MIPS32**

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
sb	I	sb rt, imm(\$rs)	$MemByte[rs_u + imm_s] \leftarrow rt_{7..0}$	sb \$1, 100(\$2)	Emmagatzemament dels 8 bits menys significatius amb adreçament indexat.
sh	I	sh rt, imm(\$rs)	<b>Si</b> $(rs_u + imm_s) \bmod 2 == 0 \longrightarrow$ $MemHalfword[rs_u + imm_s] \leftarrow rt_{15..0}$ <b>Altrament</b> <i>Excepció</i>	sh \$1, 100(\$2)	Emmagatzemament dels 16 bits menys significatius amb adreçament indexat. L'adreça final ha de ser parell.
sw	I	sw rt, imm(\$rs)	<b>Si</b> $(rs_u + imm_s) \bmod 4 == 0 \longrightarrow$ $MemWord[rs_u + imm_s] \leftarrow rt$ <b>Altrament</b> <i>Excepció</i>	sw \$1, 100(\$2)	Emmagatzemament d'una paraula amb adreçament indexat. L'adreça final ha de ser múltiple de 4.

Taula d'instruccions d'emmagatzemament del MIPS32

## 5.6 Instruccions lògiques

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
and	<b>And</b>	And lògica entre paraules.
andi	<b>And Immediate</b>	And lògica amb constant immediata.
nor	<b>Not Or</b>	O lògica negada.
or	<b>Or</b>	O lògica.
ori	<b>Or Immediate</b>	O lògica amb constant immediata.
xor	<b>Exclusive Or</b>	O exclusiva.
xori	<b>Exclusive Or Immediate</b>	O exclusiva amb constant immediata.

Taula de descripció de les instruccions lògiques del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
and	R	and rd, rs, rt	$rd \leftarrow rs \ \& \ rt$	and \$1, \$2, \$3	Instrucció <b>and</b> lògica entre registres.
andi	I	andi rs, rt, imm	$rt \leftarrow rs \ \& \ (0^{16} \parallel imm_u)$	andi \$1, \$2, 100	Instrucció <b>and</b> lògica amb valor immediat.
nor	R	nor rd, rs, rt	$rt \leftarrow \sim (rs \   \ rt)$	nor \$1, \$2, \$3	Instrucció lògica <b>not or</b> entre registres.
or	R	or rd, rs, rt	$rt \leftarrow rs \   \ rt$	or \$1, \$2, \$3	Instrucció lògica <b>or</b> entre registres.
ori	I	ori rt, rs, imm	$rt \leftarrow rs \   \ (0^{16} \parallel imm_u)$	ori \$1, \$2, 100	Instrucció or lògica amb valor immediat.
xor	R	xor rd, rs, rt	$rd \leftarrow rs \ xor \ rt$	xor \$1, \$2, \$3	Instrucció lògica <b>or exclusiva</b> entre registres.
xori	I	xori rt, rs, imm	$rt \leftarrow rs \ xor \ (0^{16} \parallel imm_u)$	xori \$1, \$2, 100	Instrucció lògica <b>or exclusiva</b> amb valor immediat.

Taula d'instruccions lògiques del MIPS32

## 5.7 Instruccions de còpia

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
mfhi	<b>M</b> ove <b>F</b> rom <b>hi</b> register	Moure del registre especial Hi.
mflo	<b>M</b> ove <b>F</b> rom <b>lo</b> register	Moure del registre especial Lo.
movn	<b>M</b> ove Conditional on <b>N</b> ot <b>Z</b> ero	Moure si no zero.
movz	<b>M</b> ove Conditional on <b>Z</b> ero	Moure si zero.
mthi	<b>M</b> ove <b>T</b> o <b>Hi</b>	Moure cap a Hi.
mtlo	<b>M</b> ove <b>T</b> o <b>Lo</b>	Moure cap a Lo.

Taula de descripció de les instruccions de còpia del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
mfhi	R	mfhi rd	$rd \leftarrow Hi$	mfhi \$1	Copiar el registre especial Hi a un registre de propòsit general.
mflo	R	mflo rd	$rd \leftarrow Lo$	mflo \$1	Copiar el registre especial Lo a un registre de propòsit general.
movn	R	movn rd, rs, rt	<b>Si</b> $(rt <> 0) \rightarrow rd \leftarrow rs$	movn \$1, \$2, \$3	Copiar si no zero.
movz	R	movn rd, rs, rt	<b>Si</b> $(rt == 0) \rightarrow rd \leftarrow rs$	movz \$1, \$2, \$3	Copiar si zero.
mthi	R	mthi rs	$Hi \leftarrow rs$	mthi \$1	Copiar el registre de propòsit general al registre especial Hi.
mtlo	R	mtlo rs	$Lo \leftarrow rs$	mtlo \$1	Copiar el registre de propòsit general al registre especial Lo.

Taula d'instruccions de còpia del MIPS32

## 5.8 Instruccions de desplaçament de bits

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
sll	<b>Shift Word Left Logical</b>	Desplaçament lògic a l'esquerra.
sllv	<b>Shift Word Left Logical Variable</b>	Desplaçament lògic variable a l'esquerra.
sra	<b>Shift Word Right Arithmetic</b>	Desplaçament aritmètic a la dreta.
srav	<b>Shift Word Right Arithmetic Variable</b>	Desplaçament aritmètic variable a la dreta.
srl	<b>Shift Word Right Logical</b>	Desplaçament lògic a la dreta.
srlv	<b>Shift Word Right Logical Variable</b>	Desplaçament lògic variable a la dreta.

Taula de descripció de les instruccions de desplaçament de bits del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
sll	R	sll rd, rt, shamp	$rd \leftarrow rt_u \ll shamp_u$	sll \$1, \$2, 3	Desplaçament a l'esquerre a partir d'un valor immediat de 5 bits.
sllv	R	sllv rd, rt, rs	$rd \leftarrow rt_u \ll (rs_u \& 0000001F_h)$	sllv \$1, \$2, \$3	Desplaçament a l'esquerre a partir de registre.
sra	R	sra rd, rt, shamp	$rd \leftarrow rt_s \text{ div } 2^{shamp_u}$	sra \$1, \$2, 3	Desplaçament aritmètic a la dreta a partir d'un valor immediat de 5 bits.
srav	R	srav rd, rt, rs	$rd \leftarrow rt_s \text{ div } 2^{(rs_u \& 0000001F_h)}$	srav \$1, \$2, \$3	Desplaçament aritmètic a la dreta a partir de registre.
srl	R	srl rd, rt, shamp	$rd \leftarrow rt_u \gg shamp_u$	srl \$1, \$2, 3	Desplaçament lògic a la dreta a partir de valor immediat.
srlv	R	srlv rd, rt, rs	$rd \leftarrow rt_u \gg (rs_u \& 0000001F_h)$	srlv \$1, \$2, \$3	Desplaçament lògic a la dreta a partir de registre.

**Taula d'instruccions de desplaçament de bits del MIPS32**

## 5.9 Instruccions de *Trap*

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
break	<b>B</b> reakpoint	Definició un punt de parada.
syscall	<b>S</b> ystem <b>C</b> all	Crida al sistema.
teq	<b>T</b> rap if <b>E</b> qual	Crida al sistema si igual.
teqi	<b>T</b> rap if <b>E</b> qual <b>I</b> mmEDIATE	Crida al sistema si igual amb adreçament immediat amb signe.
tge	<b>T</b> rap if <b>G</b> reater or <b>E</b> qual	Crida al sistema si més gran o igual amb signe.
tgei	<b>T</b> rap if <b>G</b> reater or <b>E</b> qual <b>I</b> mmEDIATE	Crida al sistema si més gran o igual amb immediat amb signe.
tgeiu	<b>T</b> rap if <b>G</b> reater or <b>E</b> qual <b>I</b> mmEDIATE <b>U</b> nsigned	Crida al sistema si més gran o igual amb immediat sense signe.
tgeu	<b>T</b> rap if <b>G</b> reater or <b>E</b> qual <b>U</b> nsigned	Crida al sistema si més gran o igual sense signe.
slt	<b>T</b> rap if <b>L</b> ess <b>T</b> han	Crida al sistema si més petit amb signe.
slti	<b>T</b> rap if <b>L</b> ess <b>T</b> han <b>I</b> mmEDIATE	Crida al sistema si més petit amb immediat amb signe.
sltiu	<b>T</b> rap if <b>L</b> ess <b>T</b> han <b>I</b> mmEDIATE <b>U</b> nsigned	Crida al sistema si més petit amb immediat sense signe.
sltu	<b>T</b> rap if <b>L</b> ess <b>T</b> han <b>U</b> nsigned	Crida al sistema si més petit sense signe.
tne	<b>T</b> rap if <b>N</b> ot <b>E</b> qual	Crida al sistema si no igual.
tnei	<b>T</b> rap if <b>N</b> ot <b>E</b> qual <b>I</b> mmEDIATE	Crida al sistema si no igual amb immediat.

Taula de descripció de les instruccions de *trap* del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
break	R	break	<i>Excepció</i>	break	Excepció de punt de parada. <sup>1</sup>
syscall	R	syscall	<i>Excepció</i>	syscall	Excepció de crida al sistema. <sup>1</sup>
teq	R	teq rs, rt	<i>Si (rs == rt) → Excepció</i>	teq \$1, \$2	Excepció de crida al sistema si igual. <sup>2</sup>
teqi	I	teqi rs, imm	<i>Si (rs<sub>s</sub> == imm<sub>s</sub>) → Excepció</i>	teq \$1, 100	Excepció de crida al sistema si igual amb immediat. <sup>2</sup>
tge	R	tge rs, rt	<i>Si (rs<sub>s</sub> &gt;= rt<sub>s</sub>) → Excepció</i>	tge \$1, \$2	Excepció de crida al sistema si més gran o igual amb signe. <sup>2</sup>
tgei	I	tgei rs, imm	<i>Si (rs<sub>s</sub> &gt;= imm<sub>s</sub>) → Excepció</i>	tgei \$1, 100	Excepció de crida al sistema si més gran o igual amb immediat amb signe. <sup>2</sup>

Taula A d'instruccions de *trap* del MIPS32

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
tgeiu	I	tgeiu rs, imm	<b>Si</b> ( $rs_u \geq imm_u$ ) $\rightarrow$ <i>Excepcio</i>	tgeiu \$1, 100	Excepció de crida al sistema si més gran o igual amb immediat sense signe. <sup>2</sup>
tgeu	R	tgeu rs, rt	<b>Si</b> ( $rs_u \geq rt_u$ ) $\rightarrow$ <i>Excepcio</i>	tgeu \$1, \$2	Excepció de crida al sistema si més gran o igual sense signe. <sup>2</sup>
tlt	R	tlt rs, rt	<b>Si</b> ( $rs_s < rt_s$ ) $\rightarrow$ <i>Excepcio</i>	tlt \$1, \$2	Excepció si més petit amb signe. <sup>2</sup>
tlti	I	tlti rs, imm	<b>Si</b> ( $rs_s < imm_s$ ) $\rightarrow$ <i>Excepcio</i>	tlti \$1, 100	Excepció si més petit amb signe amb immediat. <sup>2</sup>
tltiu	I	tltiu rs, imm	<b>Si</b> ( $rs_u < imm_u$ ) $\rightarrow$ <i>Excepcio</i>	tltiu \$1, 100	Excepció si més petit sense signe amb immediat. <sup>2</sup>
tltu	R	tltu rs, rt	<b>Si</b> ( $rs_u < rt_u$ ) $\rightarrow$ <i>Excepcio</i>	tltu \$1, \$2	Excepció si més petit sense signe. <sup>2</sup>
tne	R	tne rs, rt	<b>Si</b> ( $rs \langle \rangle rt$ ) $\rightarrow$ <i>Excepcio</i>	tne \$1, \$2	Excepció si no igual. <sup>2</sup>
tnei	I	tnei rs, imm	<b>Si</b> ( $rs_s \langle \rangle imm_s$ ) $\rightarrow$ <i>Excepcio</i>	tnei \$1, 100	Excepció si no igual amb immediat. <sup>2</sup>

Taula B d'instruccions de *trap* del MIPS32

<sup>1</sup>Els 20 bits [25..6] del codi d'operació contenen el codi de l'excepció.

<sup>2</sup>Els 10 bits [15..6] del codi d'operació contenen el codi de l'excepció.

## 5.10 Pseudoinstruccions

<i>Mnemònic</i>	<i>Instrucció</i>	<i>Significat</i>
li	<b>L</b> oad <b>I</b> mmEDIATE	Carregar valor immediat.
la	<b>L</b> oad <b>A</b> dress	Carregar adreça immediata.

**Taula de descripció de les pseudoinstruccions utilitzades en l'assemblador del MIPS32**

<i>Mnemònic</i>	<i>Format</i>	<i>Operació</i>	<i>Significat</i>	<i>Exemple</i>	<i>Comentaris</i>
li	-	li rd, $imm_{31..0}$	$rd \leftarrow imm$	li \$1, 100000	Carregar valor immediat de 32 bits a un registre.
la	-	la rd, $imm_{31..0}$	$rd \leftarrow imm$	la \$1, 100000	Carregar valor immediat de 32 bits a un registre.

**Taula de pseudoinstruccions utilitzades en l'assemblador del MIPS32**