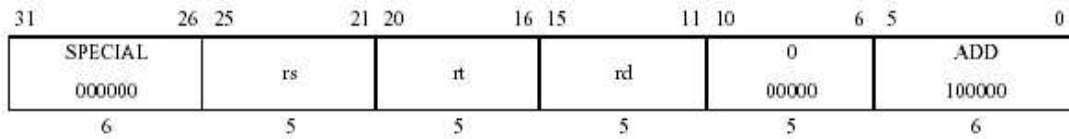


JOC D'INSTRUCCIONS DEL MIPS

1.- INSTRUCCIONS ARITMÈTIQUES:

ADD

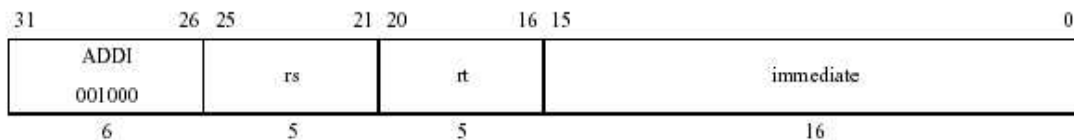


Format: ADD rd , rs , rt

Descripció: rd ← rs + rt

La instrucció Add "suma", és una instrucció que permet sumar registres de 32 bits, si el sumatori dels registres dona lloc a un desbordament aritmètic dels 32 bits, el registre de destinació no es modifica i provoca una excepció de desbordament (trap). En cas contrari si no hi ha desbordament el resultat es col·locarà en a rd.

ADDI

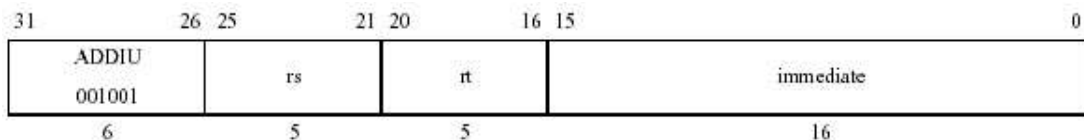


Format: ADDI rd , rs , immediat

Descripció: rd ← rs + immediat

La instrucció Addi "suma immediata ", màxima longitud que pot obtenir l'immediat és de 16 bits, aquesta instrucció permetrà fer una suma entre un registre i un immediat i si no hi ha desbordament la col·locarà en el registre destí en el nostre cas rd.

ADDIU

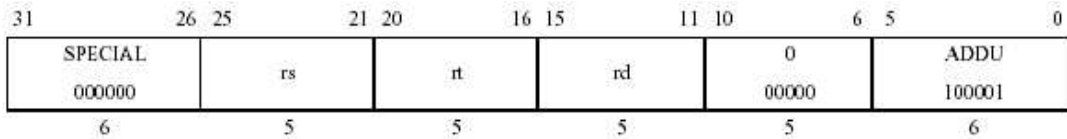


Format: ADDIU rd , rs , immediat

Descripció: rd ← rs + immediat

La instrucció Addi "suma immediata sense signe ", màxima longitud que pot obtenir l'immediat és de 16 bits, aquesta instrucció permetrà fer una suma entre un registre i un immediat i no ens preocuparem per cap desbordament.

ADDU

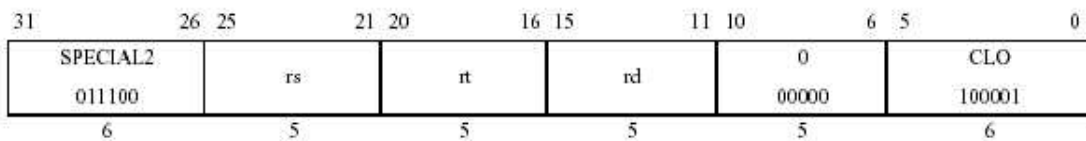


Format: ADDU rd , rs , rt

Descripció: rd ← rs + rt

La instrucció Addu "suma unsigned", és una instrucció que permet sumar registres de 32 bits, en aquests cas no té en compte el desbordament, i el resultat és emmagatzemat al registre destí "rd "

CLO

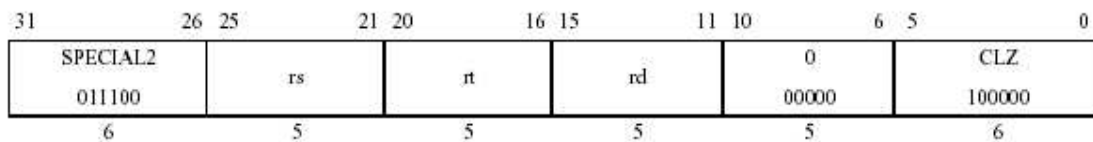


Format: CLO rd , rs

Descripció: rd ← "count_leading_ones",

La instrucció CLO "count_leading_ones", el que fa és comptar el nombre de 1's que hi ha amb una paraula, i ho posa en el registre destí.

CLZ

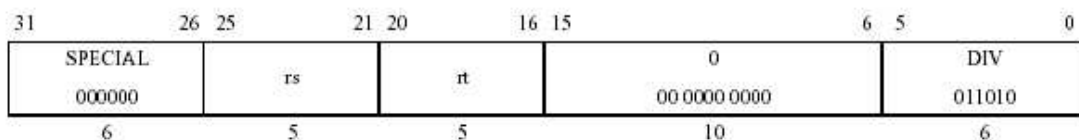


Format: CLZ rd , rs

Descripció: rd ← "count_leading_zeros"

La instrucció CLO "count_leading_zeros", el que fa és comptar el nombre de 0's que hi ha amb una paraula, i ho posa en el destí.

DIV

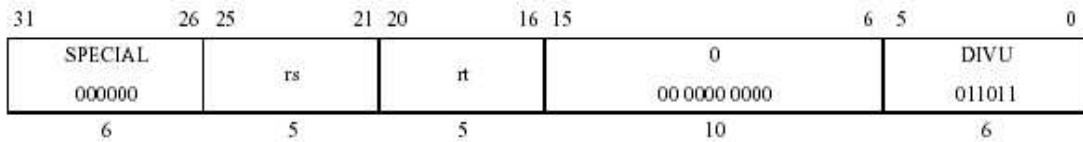


Format: DIV rd , rs

Descripció: (HI, LO) ← rs / rt

El quocient de 32 bits es posa en el registre especial LO i el resta dels 32 bits es desplaçat en el registre HI. No provoca cap excepció aritmètica. Si la divisió dona zero el resultat és imprevisible

DIVU

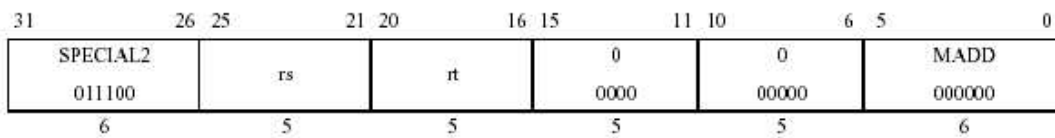


Format: DIVU rd , rs

Descripció: (HI, LO) ← - rs / rt

El quocient de 32 bits es posa en el registre especial LO i el resta dels 32 bits es desplaçat en el registre HI, en tot moment tractarem les paraules com a Unsigned (positius), No provoca cap excepció aritmètica.

MADD

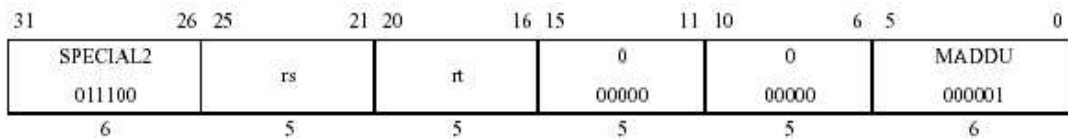


Format: MADD rs , rt

Descripció: (HI,LO) ← (HI,LO) + (rs x rt)

Multiplica els registres rs i rd (de 32 bits amb signe) el resultat és suma amb el registres especials HI i LO. El registre HI és el registre que contindrà els 32bits alts i LO és el registre amb els 32 bits baixos.

MADDU

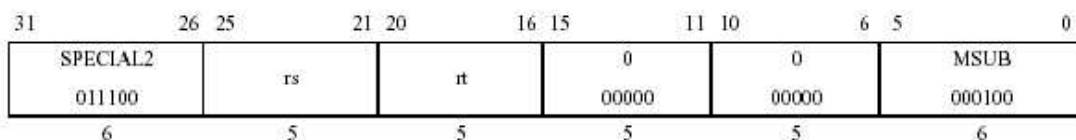


Format: MADDU rs , rt

Descripció: (HI,LO) ← (HI,LO) + (rs x rt)

Multiplica els registres rs i rd (de 32 bits sense signe) el resultat és suma amb el registres especials HI i LO. El registre HI és el registre que contindrà els 32 bits alts i LO és el registre amb els 32 bits baixos.

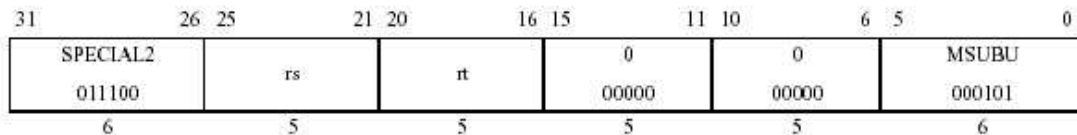
MSUB



Format: MSUB rs , rt
Descripció: (HI,LO) ← (HI,LO) – (rs x rt)

Multiplica els registres rs i rd (de 32 bits amb signe) el resultat és resta amb el registres especials HI i LO. El registre HI és el registre que contindrà els 32bits alts i LO és el registre amb els 32 bits baixos.

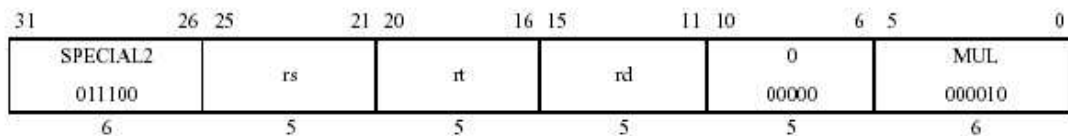
MSUBU



Format: MSUBU rd , rs
Descripció: (HI,LO) ← (HI,LO) – (rs x rt)

Multiplica els registres rs i rd (de 32 bits sense signe) el resultat és resta amb el registres especials HI i LO. El registre HI és el registre que contindrà els 32 bits alts i LO és el registre amb els 32 bits baixos.

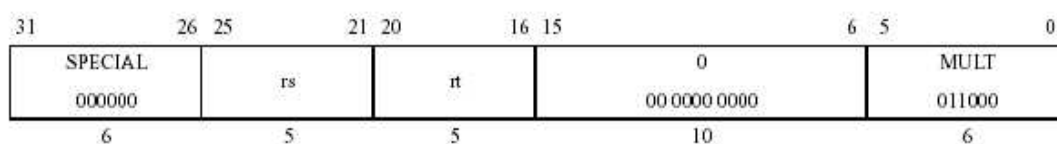
MUL



Format: MUL rd , rs,rt
Descripció: rd ← rs x rt

Multiplica els registres rs i rd (de 32 bits) i es posa en el registre destí, en el nostre cas rd. També és col.locat en el registre especial LO, En el cas que l'operació resultant sigui superior a 32 bits, és a dir entre 32 i 64 bits, el resultat quedarà reflectit en la part alta en el registre HI i la part baixa en el registre LO i rd. En cap cas no provoca cap excepció aritmètica.

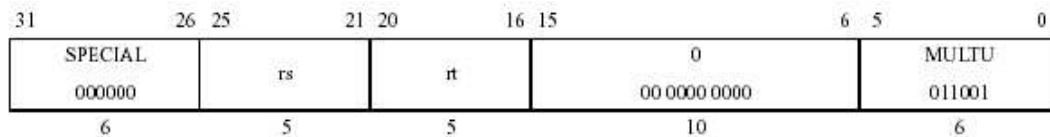
MULT



Format: MULT rs , rt
Descripció: (HI, LO) ← rs x rt

Multiplica els registres rs i rt (de 32 bits) el resultat quedarà reflectit en la part alta en el registre HI i la part baixa en el registre LO. En cap cas no provoca cap excepció aritmètica.

MULTU

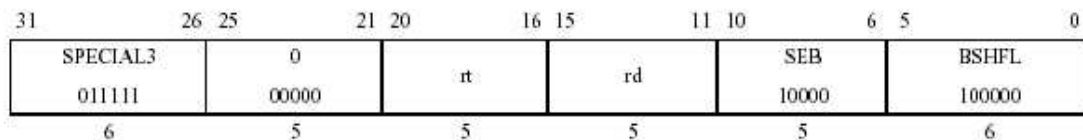


Format: MULTU rs , rt

Descripció: (HI, LO) ← rs x rt

Multiplica els registres rs i rt (de 32 bits sense signe) el resultat quedarà reflectit en la part alta en el registre HI i la part baixa en el registre LO. En cap cas no provoca cap excepció aritmètica.

SEB

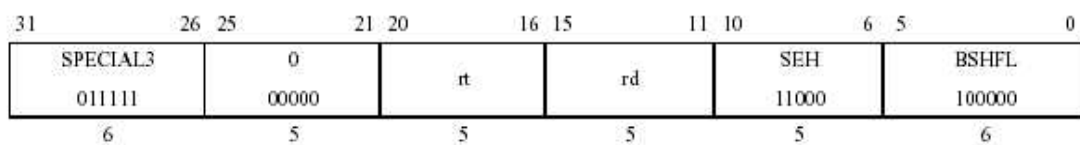


Format: SEB rd , rt

Descripció: rd ← SignExtend (rt_{7..0})

El byte menys significat del registre rt, s'exten amb signe i es col.loca al registre destí.

SEH

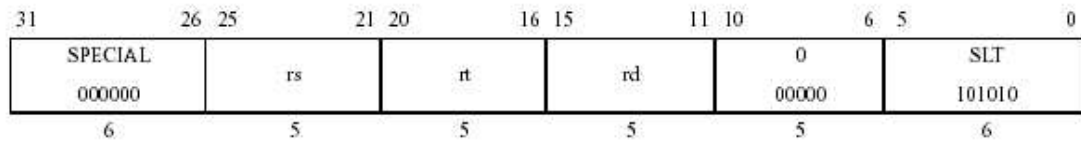


Format: SEH rd , rs

Descripció: rd ← SignExtend (rt_{15..0})

El "halfword" menys significat del registre rt, s'exten amb signe i es col.loca al registre destí.

SLT

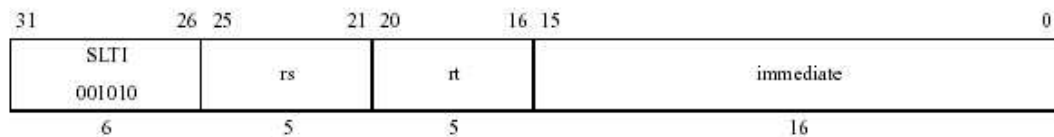


Format: SLT rd , rs, rt

Descripció: rd \leftarrow (rs < rt)

Fa la comparació entre rs i rt, si el registre rs és més petit que rt col·locarà un 1 en el registre destí en cas contrari col·locarà un 0.

SLTI

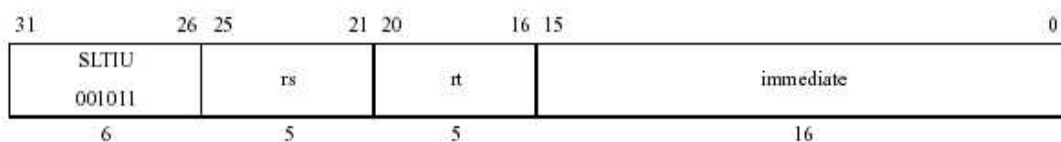


Format: SLTI rd , rs ,immediat

Descripció: rd \leftarrow (rs < immediat)

Fa la comparació entre rs i un immediat de 16 bits, si el registre rs és més petit que l'immediat col·locarà un 1 en el registre destí en cas contrari col·locarà un 0.

SLTIU

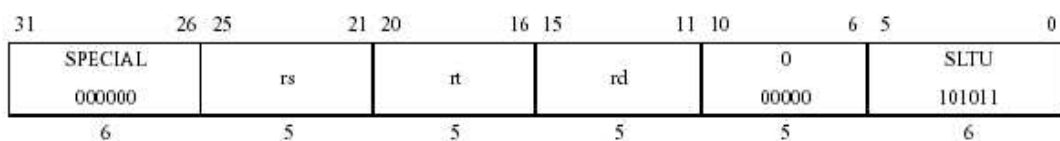


Format: SLTIU rd , rs ,immediat

Descripció: rd \leftarrow (rs < immediat)

Fa la comparació entre rs i un immediat de 16 bits (sense signe), si el registre rs és més petit que l'immediat col·locarà un 1 en el registre destí en cas contrari col·locarà un 0.

SLTU

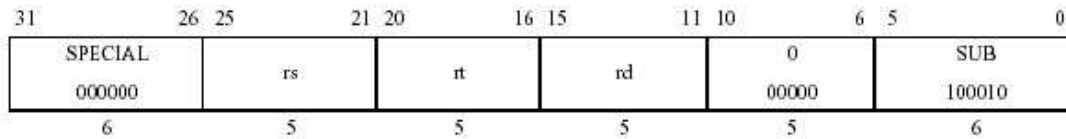


Format: SLTU rd , rs, rt

Descripció: $rd \leftarrow (rs < rt)$

Fa la comparació entre rs i rt (han de ser registres sense signe), si el registre rs és més petit que rt col·locarà un 1 en el registre destí en cas contrari col·locarà un 0.

SUB

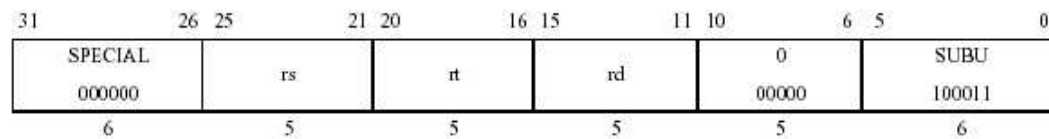


Format: SUB rd , rs, rt

Descripció: $rd \leftarrow rs - rt$

La instrucció SUB "resta", és una instrucció que permet restar registres de 32 bits, si la resta dels registres dona lloc a un desbordament aritmètic dels 32 bits, el registre de destinació no es modifica i provoca una excepció de desbordament (trap). En cas contrari si no hi ha desbordament el resultat es col·locarà en a rd.

SUBU



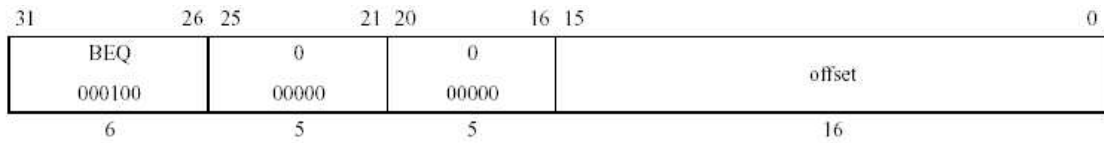
Format: SUBU rd , rs ,rt

Descripció: $rd \leftarrow rs - rt$

La instrucció SUBU "resta sense signe", és una instrucció que permet restar registres de 32 bits, si la resta dels registres dona lloc a un desbordament aritmètic dels 32 bits, el registre de destinació no es modifica i provoca una excepció de desbordament (trap). En cas contrari si no hi ha desbordament el resultat es col·locarà en a rd.

2.- INSTRUCCIONS de BRANCH i JUMP:

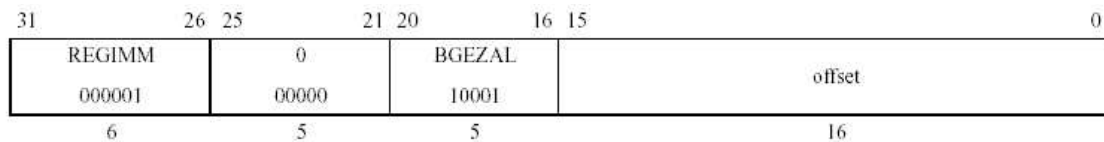
B



Format: B offset
Descripció: branch incondicional

La instrucció B (Branch) fa un salt incondicional a la posició de memòria indicada i s'hi pot indicar un (Offset) o desplaçament de fins a 18 bits amb signe (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de l'instrucció) per tal de sumar-ho o restar-ho de la posició anterior.

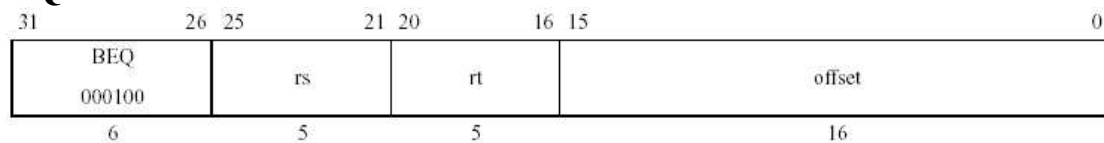
BAL



Format: BAL rs, offset
Descripció: crida a un procediment

La instrucció BAL (Branch and Link) sempre fa una crida a la posició indicada pel registre "R0" amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció). Un cop acaba l'execució del procediment retorna i executa la instrucció de memòria indicada al registre rs.

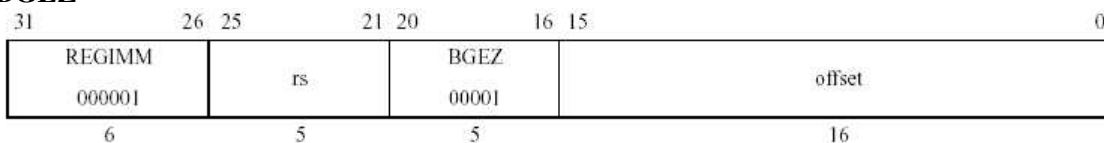
BEQ



Format: BEQ rs, rt, offset
Descripció: compara els dos registres i fa un branch condicional

La instrucció BEQ (Branch on Equal) sempre compara els dos registres indicats i salta si són iguals a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció).

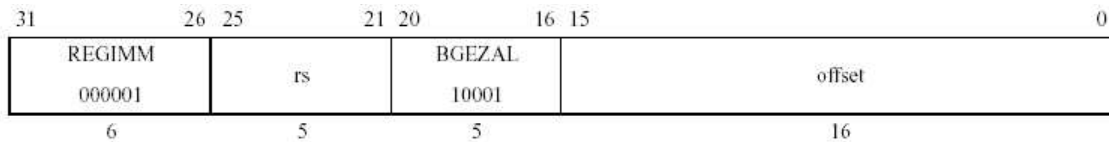
BGEZ



Format: BGEZ rs, offset
Descripció: si el registre rs>=0 fa un branch condicional

La instrucció BGEZ (Branch on Greater than or Equal to Zero) mira si el registre "rs" és igual o més gran que zero en aquest cas salta a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció).

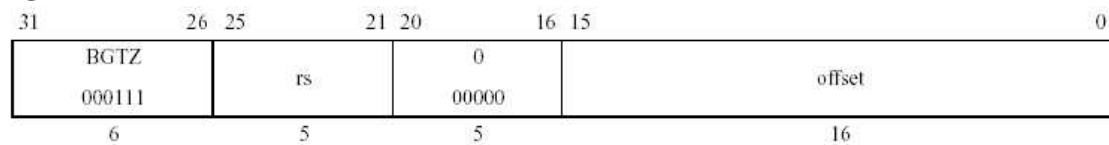
BGEZAL



Format: BGEZAL rs, offset
Descripció: si el registre rs>=0 crida a un procediment

La instrucció BGEZAL (Branch on Greater than or Equal to Zero And Link) mira si el registre "rs" és igual o més gran que zero en aquest cas salta a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció). Al finalitzar el procediment retorna a la posició indicada pel registre "rs".

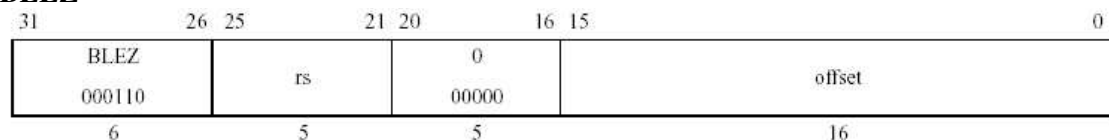
BGTZ



Format: BGTZ rs, offset
Descripció: si el registre rs>0 fa un branch condicional

La instrucció BGEZ (Branch on Greater Than Zero) mira si el registre "rs" és més gran que zero en aquest cas salta a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció).

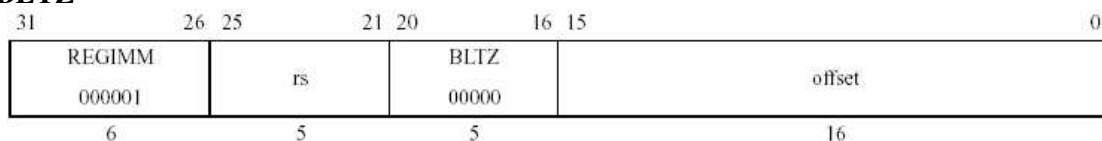
BLEZ



Format: BLEZ rs, offset
Descripció: si el registre rs<=0 fa un branch condicional

La instrucció BGEZ (Branch on Less than or Equal to Zero) mira si el registre "rs" és igual o més petit que zero en aquest cas salta a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció).

BLTZ

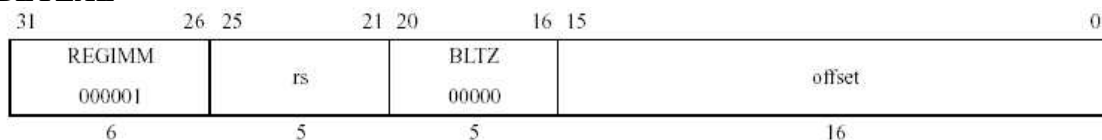


Format: BLTZ rs, offset

Descripció: si el registre rs<0 fa un branch condicional

La instrucció BGEZ (Branch on Less Than Zero) mira si el registre "rs" és més petit que zero en aquest cas salta a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció).

BLTZAL

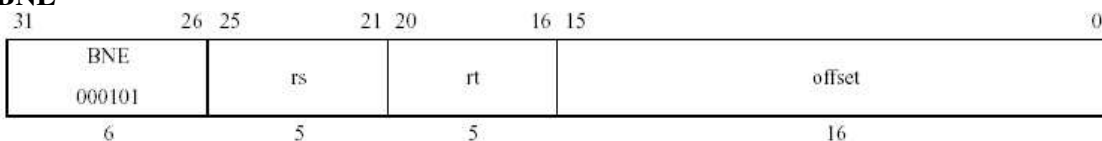


Format: BLTZAL rs, offset

Descripció: si el registre rs<0 crida a un procediment

La instrucció BGEZ (Branch on Less Than Zero and Link) mira si el registre "rs" és més petit que zero en aquest cas salta a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció). Al finalitzar el procediment retorna a la posició indicada pel registre "rs".

BNE

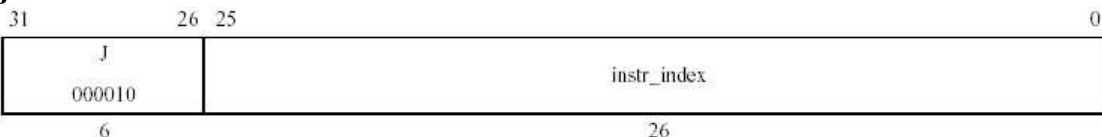


Format: BNE rs, rt, offset

Descripció: si rs!=rt fa un branch condicional

La instrucció BNE (Branch or Not Equal) mira si els registre "rs" i "rt" són diferents en aquest cas salta a la posició indicada per la següent instrucció a executar amb un offset de 18 bits (en realitat l'offset n'agafa 16 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció).

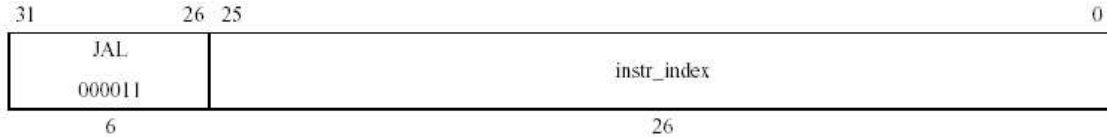
J



Format: J target
Descripció: fa un salt incondicional a la posició de memòria indicada

La instrucció J (Jump) salta a la posició indicada de 28 bits (en realitat l'offset n'agafa 26 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció).

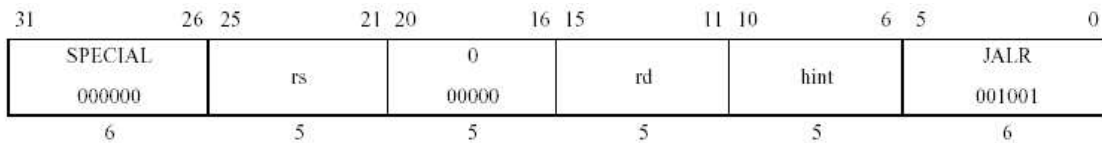
JAL



Format: JAL target
Descripció: fa un salt incondicional a la posició de memòria indicada on executa un procediment

La instrucció JAL (Jump And Link) salta a la posició indicada de 28 bits (en realitat l'offset n'agafa 26 però els dos restants no els perd només els desplaça cap a l'esquerra de la instrucció). Al retornar del procediment executa la següent instrucció a continuació d'ell.

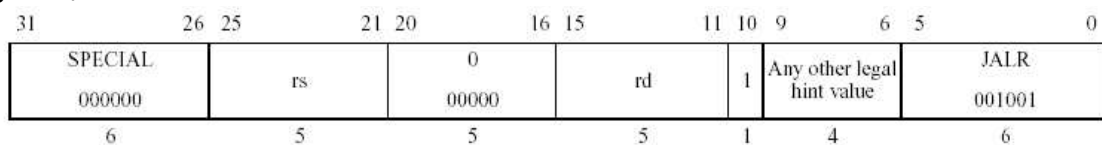
JALR



Format: JAL rs (implica utilitzar com a "rd" el registre 31)
 JAL rd, rs
Descripció: fa un salt incondicional a la posició indicada per "rs"

La instrucció JALR (Jump And Link Register) salta a la posició indicada pel registre "rs" per tal d'executar un procediment. Al finalitzar l'execució retorna a la posició indicada pel registre "rd".

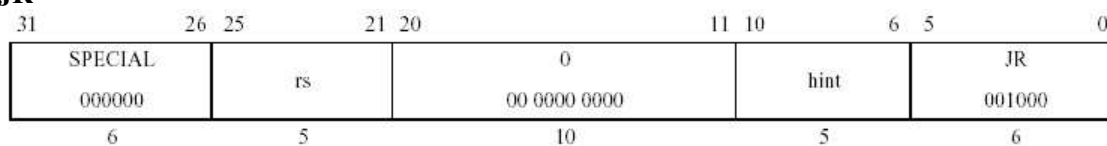
JALR.HB



Format: JALR.HB rs (implica utilitzar com a "rd" el registre 31)
 JALR.HB rd, rs
Descripció: fa un salt incondicional a la posició indicada per "rs"

La instrucció JALR.HB (Jump And Link Register with Hazard Barrier) salta a la posició indicada pel registre "rs" per tal d'executar un procediment. Al finalitzar l'execució retorna a la posició indicada pel registre "rd". La particularitat que té és que finalitza totes les possibles operacions perilloses que hagin pogut ser generades.

JR

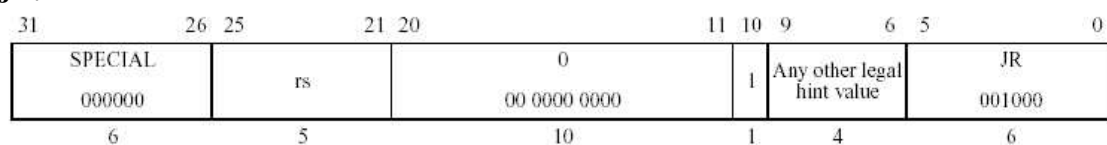


Format: JR rs

Descripció: fa un salt incondicional a la posició indicada pel registre "rs"

La instrucció JR (Jump Register) salta a la posició indicada pel registre "rs".

JR.HB



Format: JR.HB rs

Descripció: fa un salt incondicional a la posició indicada pel registre "rs"

La instrucció JR (Jump Register) salta a la posició indicada pel registre "rs". La particularitat que té és que finalitza totes les possibles operacions perilloses que hagin pogut ser generades.

3.- INSTRUCCIONS de CONTROL:

EHB

31	26	25	21	20	16	15	11	10	6	5	0
SPECIAL	0	0	0	0	3	SLL					
000000	00000	00000	00000	00000	00011	000000					
6	5	5	5	5	5	6					

Format: EHB

Descripció: finalitza operacions perilloses

La instrucció EHB (Execution Hazard Barrier) para l'execució del programa fins que les intruccions perilloses han estat finalitzades.

NOP

31	26	25	21	20	16	15	11	10	6	5	0
SPECIAL	0	0	0	0	0	SLL					
000000	00000	00000	00000	00000	00000	000000					
6	5	5	5	5	5	6					

Format: NOP

Descripció: no fa res

La instrucció NOP (No OPeration) no fa res, és a dir, internament executa SLL r0, r0, 0.

SSNOP

31	26	25	21	20	16	15	11	10	6	5	0
SPECIAL	0	0	0	0	1	SLL					
000000	00000	00000	00000	00000	00001	000000					
6	5	5	5	5	5	6					

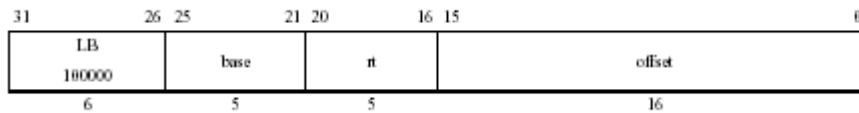
Format: SSNOP

Descripció: para les rutines superescalars.

La instrucció SSNOP (SupereScalar No OPeration) para les rutines superescalars, és a dir, internament executa SLL r0, r0, 1.

4.- INSTRUCCIONS de CÀRREGA, EMMAGATZAMAMENT i CONTROL de MEMÒRIA

LB

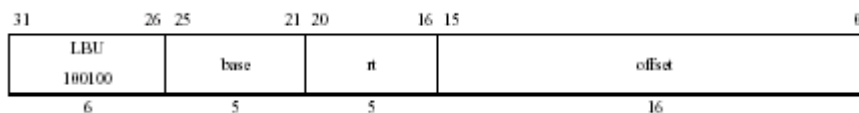


Format: LB rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Carrega al registre rt el primer byte de memòria especificat per l'adreça obtinguda per la suma de base + offset. Manté el signe.

LBU

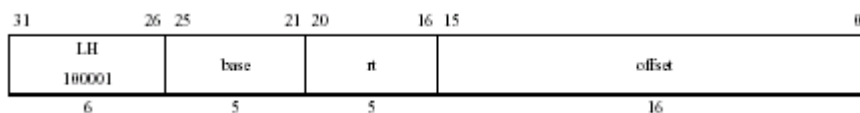


Format: LBU rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Carrega al registre rt el primer byte de memòria especificat per l'adreça obtinguda per la suma de base + offset. No manté el signe.

LH

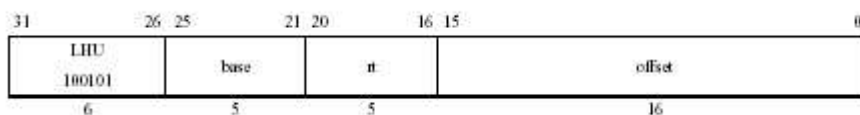


Format: LH rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Carrega al registre rt els 2 primers bytes de memòria especificat per l'adreça obtinguda per la suma de base + offset. Manté el signe.

LHU

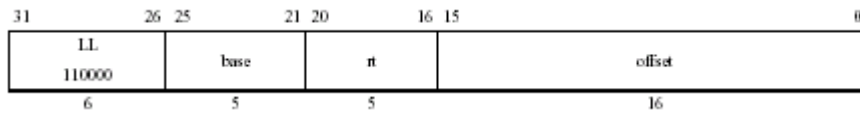


Format: LHU rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Carrega al registre rt els 2 primers bytes de memòria especificat per l'adreça obtinguda per la suma de base + offset. No manté el signe.

LL

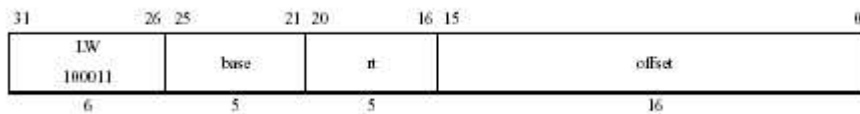


Format: LL rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Es suma el valor d'rt amb l'adreça especificada al segon operand. El resultat apuntarà també a l'etiqueta.

LW

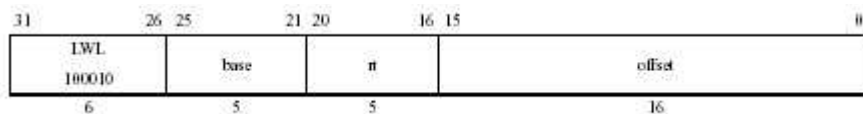


Format: LWU rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Carrega al registre rt els 4 primers bytes de memòria especificat per l'adreça obtinguda per la suma de base + offset. Manté el signe.

LWL

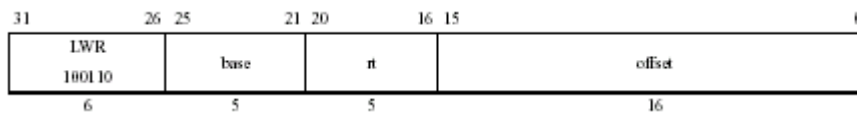


Format: LWL rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Fa un desplaçament de mida word cap a l'esquerra del registre(cap als bits de més pes).

LWR

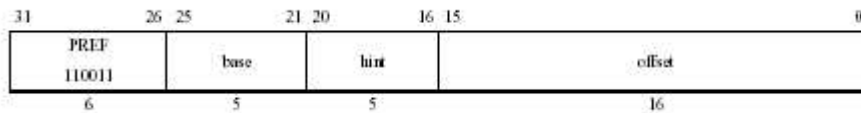


Format: LBU rt,offset(base)

Descripció: $rt \leftarrow \text{memory}[\text{base} + \text{offset}]$

Fa un desplaçament de mida word cap a la dreta del registre(cap als bits de menys pes).

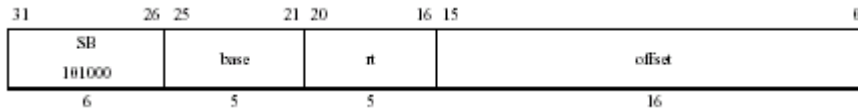
PREF



Format: PREF hint, offset(base)

Descripció: Transferència de dades entre memòria i caché.

SB

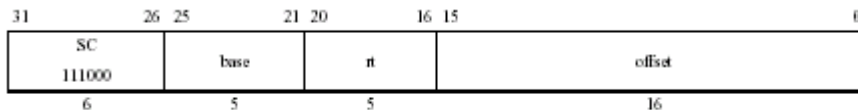


Format: SB rt,offset(base)

Descripció: memory[base+offset]<- rt

Carrega el contingut del registre rt a l'adreça especificada. Carrega un byte.

SC

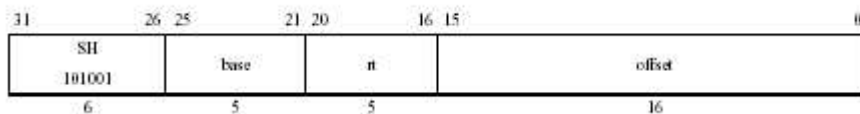


Format: SC rt,offset(base)

Descripció: memory[base+offset]<- rt

Carrega el contingut del registre rt a l'adreça especificada en funció del valor d'un flag.

SH



Format: SH rt,offset(base)

Descripció: memory[base+offset]<- rt

Carrega el contingut del registre rt a l'adreça especificada. Carrega dos bytes

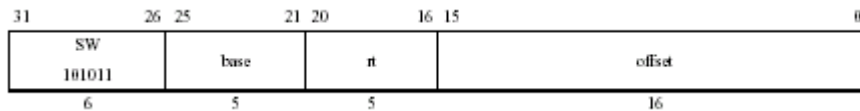
SD

Format: SD rt,offset(base)

Descripció: memory[base+offset]<- rt

Carrega el contingut del registre rt a l'adreça especificada. Carrega 8 bytes.

SW

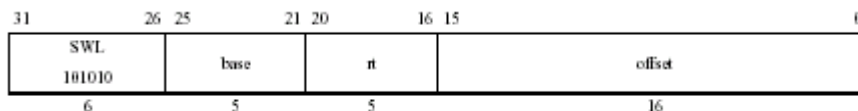


Format: SW rt,offset(base)

Descripció: memory[base+offset]<- rt

Carrega el contingut del registre rt a l'adreça especificada. Carrega 4 bytes.

SWL

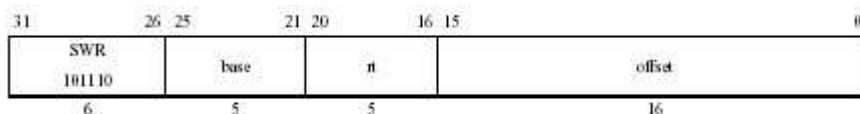


Format: SWL rt,offset(base)

Descripció: memory[base+offset]<- rt

Carrega el contingut del registre rt a l'adreça especificada. Carrega 4 bytes desplaçats cap a l'esquerra.

SWR

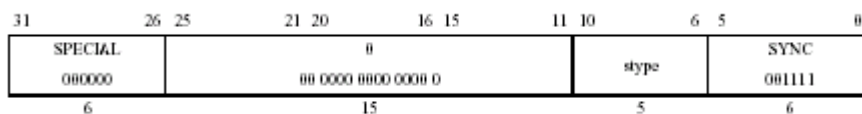


Format: SWR rt,offset(base)

Descripció: memory[base+offset]<- rt

Carrega el contingut del registre rt a l'adreça especificada. Carrega 4 bytes desplaçats cap a la dreta.

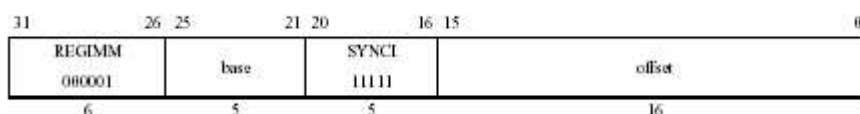
SYNC



Format: SYNC (stype =0 implied)

Descripció: Sincronització de la memòria(loads & stores).

SYNCHI

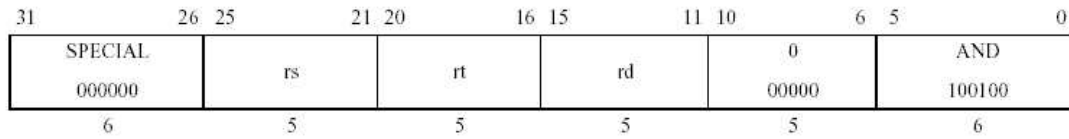


Format: SYNCHI (stype =0 implied)

Descripció: Sincronització de la memòria(loads & stores) amb valor immediat.

5.- INSTRUCCIONS LÒGIQUES de CPU:

AND

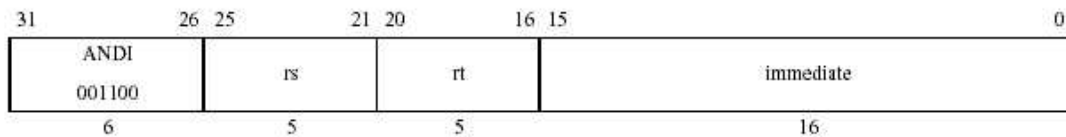


Format: AND rd, rs, rt

Descripció: realitza una AND bit a bit entre els continguts de 2 registres.

El contingut del registre de propòsit general rs es combina amb el de rt mitjançant una AND i el resultat va a parar a rd.

ANDI

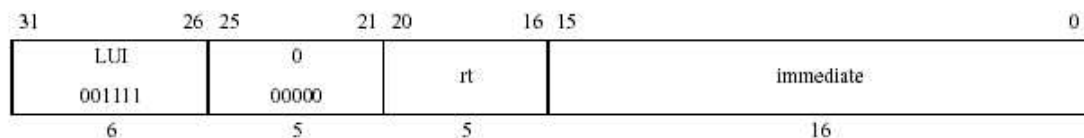


Format: AND rt, rs, immediat

Descripció: realitza una AND bit a bit entre el contingut d'un registre i una constant.

L'immediat s'extén amb zeros cap a l'esquerra i es combina amb el contingut de rs en una AND bit a bit. El resultat va a parar a rt.

LUI

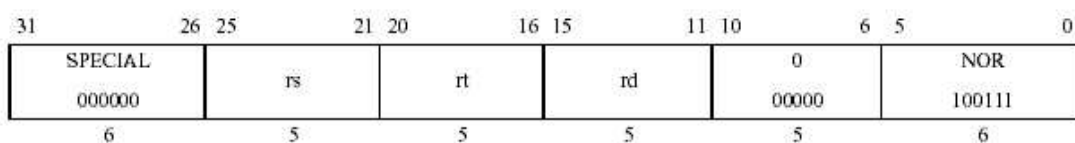


Format: LUI rt, immediat

Descripció: carrega una constant a la mitja part alta d'una paraula.

$rt \leftarrow \text{immediat} \ll 0^{16}$. L'immediat és desplaçat 16 bits a l'esquerra i aquest espai és ocupat per zeros.

NOR

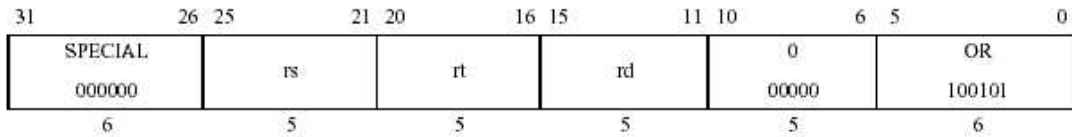


Format: NOR rd, rs, rt

Descripció: realitza una NOR bit a bit entre el contingut de 2 registres..

El contingut del registre de propòsit general rs es combina amb el de rt mitjançant una NOR i el resultat va a parar a rd.

OR

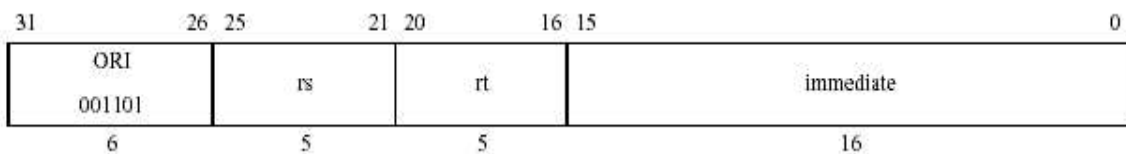


Format: OR rd, rs, rt

Descripció: realitza una OR bit a bit entre el contingut de 2 registres.

El contingut del registre de propòsit general rs es combina amb el de rt mitjançant una OR i el resultat va a parar a rd.

ORI

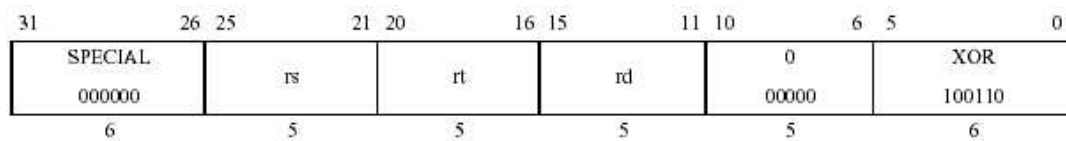


Format: ORI rt, rs, immediat

Descripció: realitza una AND bit a bit entre el contingut d'un registre i una constant.

L'immediat s'extén amb zeros cap a l'esquerra i es combina amb el contingut de rs en una OR bit a bit. El resultat va a parar a rt.

XOR

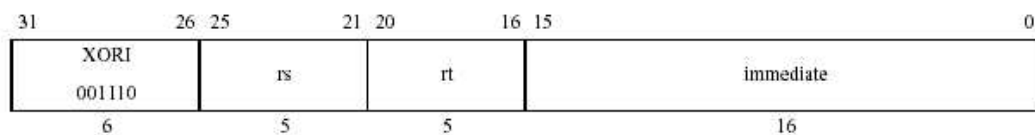


Format: XOR rd, rs, rt

Descripció: realitza una XOR bit a bit entre el contingut de 2 registres.

El contingut del registre de propòsit general rs es combina amb el de rt mitjançant una OR exclusiva i el resultat va a parar a rd.

XORI



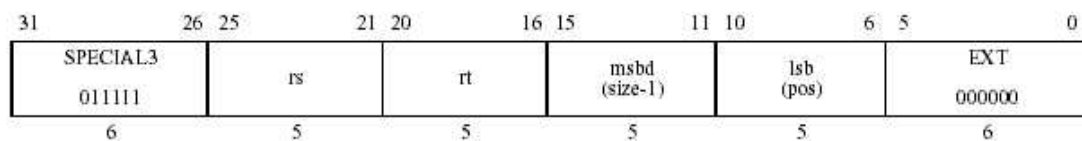
Format: XORI rt, rs, immediat

Descripció: realitza una XOR bit a bit entre el contingut d'un registre i una constant.

L'immediat s'extén amb zeros cap a l'esquerra i es combina amb el contingut de rs en una XOR bit a bit. El resultat va a parar a rt.

6.- INSTRUCCIONS INSERCIÓ EXTRACCIÓ de CPU:

EXT



Format: ext rt, rs, pos, size

Descripció: extreure un camp de bits del registre rs i emmagatzemar-los justificats a la dreta dins rt.

El camp de bits comença a la posició pos i s'extén tant com marca size. Aquest camp s'extreu de rs, es justifica a la dreta i s'emplena amb zeros per l'esquerra si cal. Llavors es guarda a rt. pos i size són convertits per l'assemblador a msbd i lsb de la següent manera:

$$\text{msbd} = \text{size} - 1$$

$$\text{lsb} = \text{pos}$$

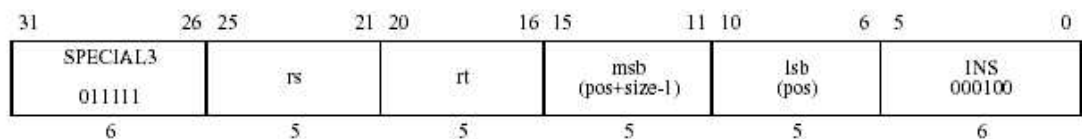
A més, han de complir:

$$0 \leq \text{pos} < 32$$

$$0 < \text{size} \leq 32$$

$$0 < \text{pos} + \text{size} \leq 32$$

INS



Format: ins rt, rs, pos, size

Descripció: col·locar un camp de bits justificat a la dreta de rs dins un camp determinat de rt.

Els size bits de més a la dreta de rs es col·loquen en el valor de rt, començant per la posició pos. El resultat es col·loca a rt. pos i size són convertits per l'assemblador a msbd i lsb de la següent manera:

$$\text{msbd} = \text{pos} + \text{size} - 1$$

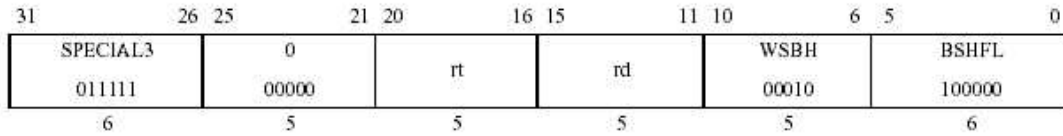
$$\text{lsb} = \text{pos}$$

A més, han de complir:

$$0 \leq \text{pos} < 32$$

$0 < \text{size} \leq 32$
 $0 < \text{pos} + \text{size} \leq 32$

WSBH

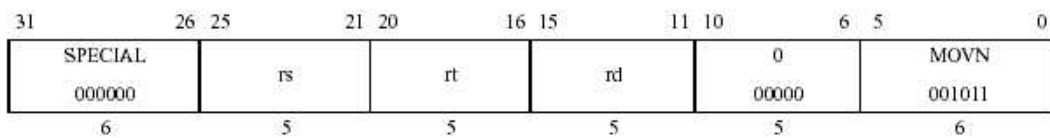


Format: wsbh rd, rt

Descripció: intercanviar els bytes entre cada mitja paraula de rt i guardar el valor a rd.

7.- INSTRUCCIONS de MOVIMENT:

MOVN

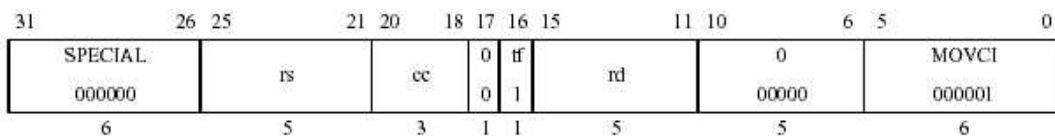


Format: MOVN rd,rd,rt

Descripció: Moviment condicional testejat.

Si el contingut d'rt és diferent de 0, llavors rd \odot rs.

MOVT

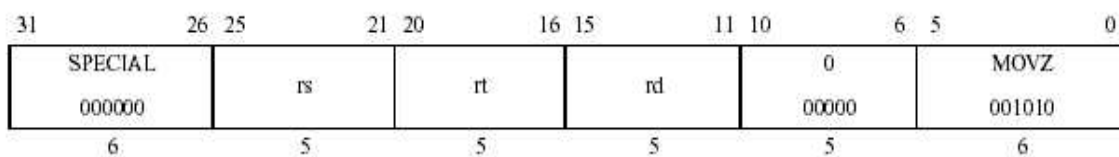


Format: MOVT rd,rs,cc

Descripció: Moviment condicional testejat.

Si el contingut de cc=1(cc determina el carry), llavors rd \odot rs.

MOVZ

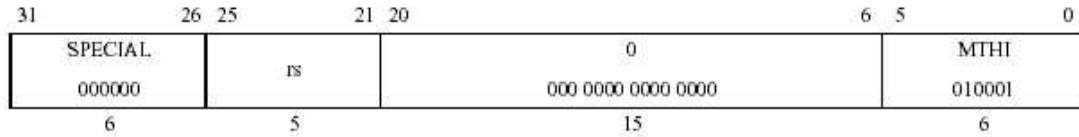


Format: MOVZ rd,rs,rt

Descripció: Efectua el move depenent del resultat del testeig del registre rt.

Si el contingut del registre rt=0 llavors rd <-rs.

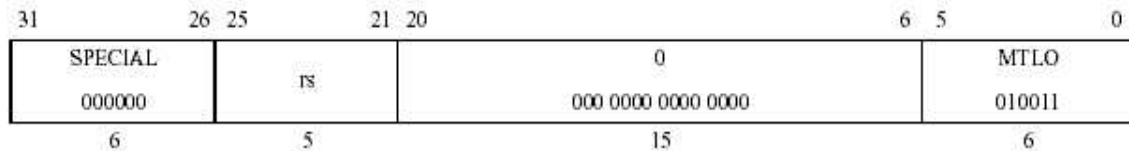
MTHI



Format: MTHI rs

Descripció: El contingut de rs és carregat al registre especial HI.
HI ← rs

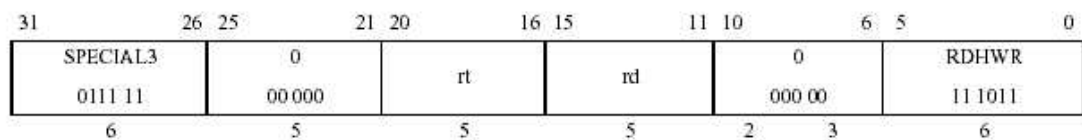
MTLO



Format: MTLO rs

Descripció: El contingut de rs és carregat al registre especial LO.
LO ← rs

RDHWR



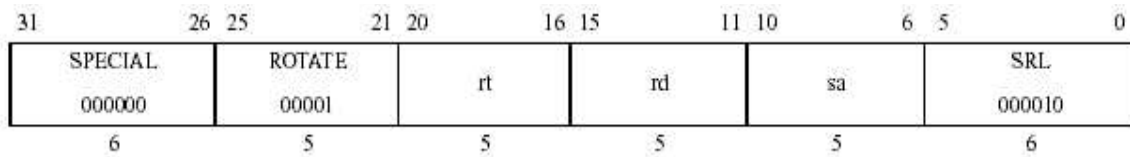
Format: RDHWR rt,rd

Descripció: rt ← HWR[rd]

Mou el contingut del registre de hardware rd si l'operació ha estat habilitada mitjançant un software privilegiat.

8.- INSTRUCCIONS de DESPLAÇAMENT:

ROTR

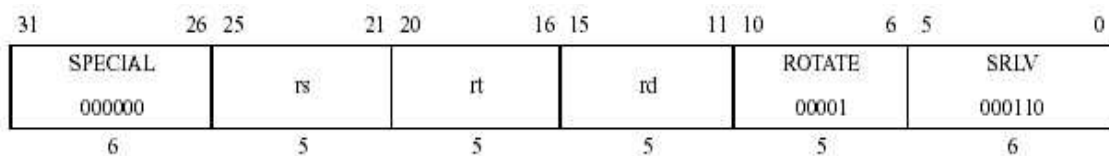


Format: ROTR rd,rt,sa

Descripció: rd ←- rt ←-> (right)sa

El contingut del registre rt es desplaça cap a la dreta. El resultat s'emmagatzema al registre rd. sa especifica el desplaçament.

ROTRV

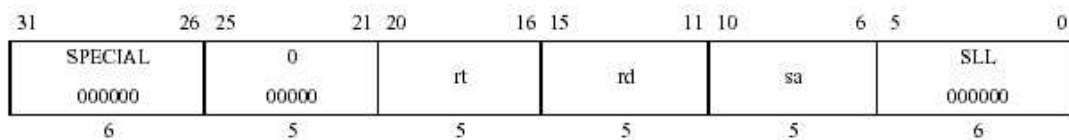


Format: ROTRV rd,rt,rs

Descripció: rd←- rt ←-> (right)rs

El contingut del registre rt es desplaça a la dreta. El resultat s'emmagatzema al registre rd. La quantitat de desplaçament s'especifica per els 5 bits de menys pes del registre rs.

SLL

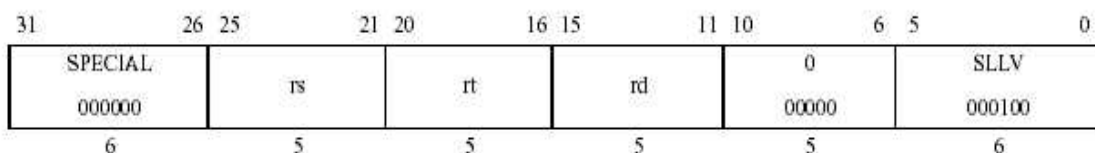


Format: SLL rd,rt,sa

Descripció:

La paraula de 32 bits continguda al registre rt es desplaça cap a l'esquerra, insertant 0 per la dreta. El nombre de 0 a insertar bé determinat per sa. El resultat el guardem a rd.

SLLV

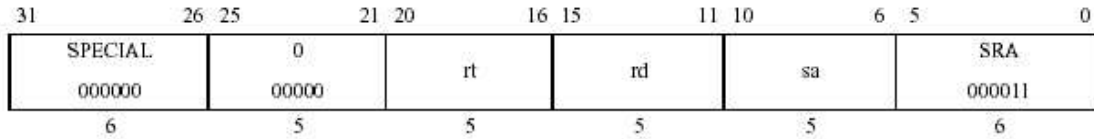


Format: SLLV rd,rt,rs

Descripció:

El contingut del registre rt es desplaça cap a l'esquerra, insertant X(indicat pels 5 bits de menys pes d'rs) zeros per la dreta. El resultat el guardem a rd.

SRA

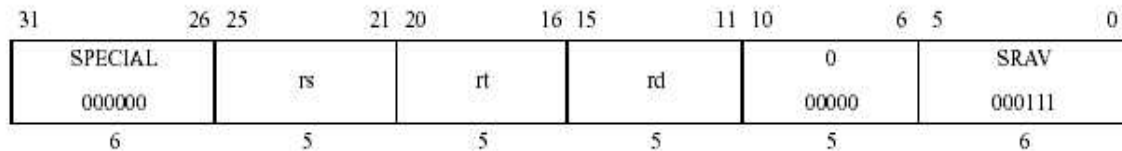


Format: SRA rd,rt,sa

Descripció: Executar un desplaçament aritmètic cap a la dreta d'un word amb un nombre fixat de bits.

El contingut del registre rt és desplaçat cap a la dreta, duplicant el bit 31(bit de signe) cap als bits buits. La paraula resultant s'emmagatzema a rd. sa indica el nombre de bits a desplaçar.

SRAV

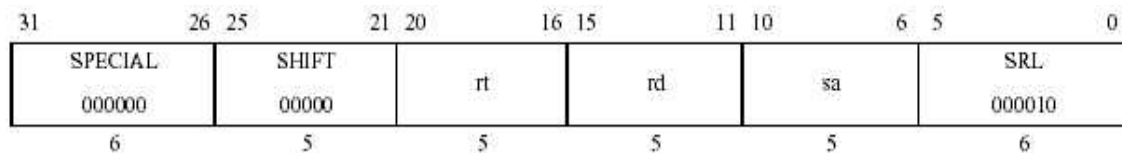


Format: SRAV rd,rt,rs

Descripció: Executar un desplaçament aritmètic a la dreta amb un nombre variable de bits.

El contingut del registre rt es desplaça cap a la dreta, duplicant el bit 31(bit de signe) als bits buits. El resultat es guarda a rd. El desplaçament de bits s'especifica amb els 5 bits de menys pes del registre rs.

SRL

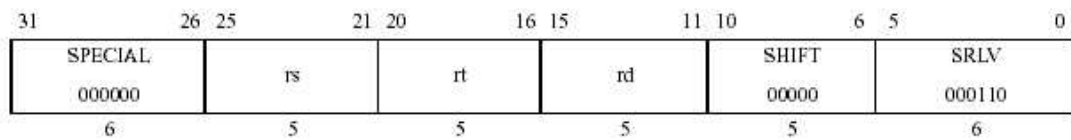


Format: SRL rd,rt,sa

Descripció: rd rt>>sa

Executar un desplaçament lògic cap a la dreta d'una paraula amb un nombre fixat de bits. El contingut del registre rt es desplaça a la dreta insertant 0 als bits buits. La paraula resultant es guarda a rd. El desplaçament d'especifica mitjançant el registre sa.

SRLV

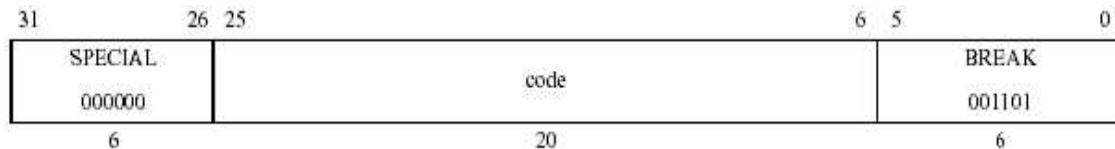


Format: SRLV rd,rt,rs

Descripció: El contingut del registre rt es desplaça cap a la dreta insertant 0 als bits buits. La paraula resultant es guarda a rd. El desplaçament s'especifica pels 5 bits de menys pes del registre rs.

9.- INSTRUCCIONS de TRAP:

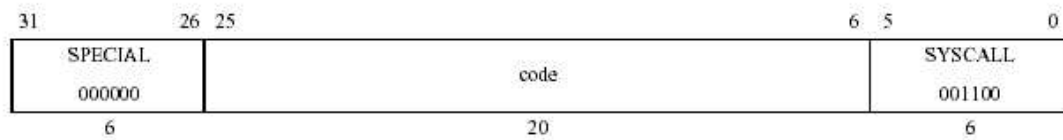
BREAK



Format: BREAK

Descripció: Causa una excepció de break. Inmediatament i incondicionalment es transfereix el control al gestor d'excepcions.

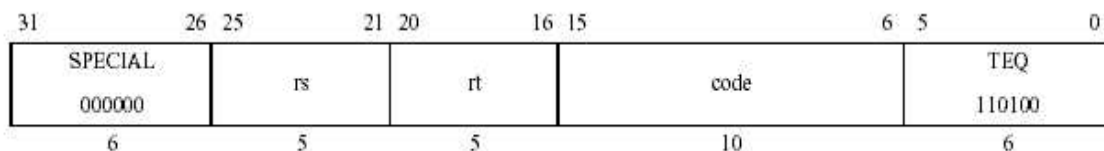
SYSCALL



Format: SYSCALL

Descripció: Causa una excepció de crida al sistema (trap) i incondicionalment transferim el control al gestor d'excepcions.

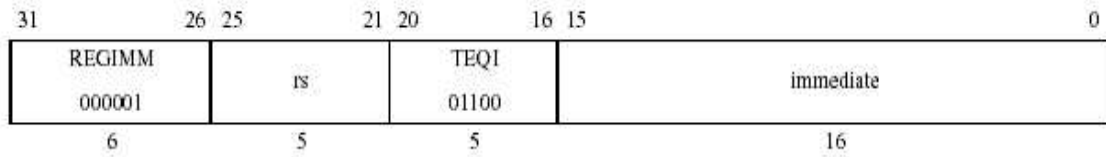
TEQ



Format: TEQ rs,rt

Descripció: Compara els continguts dels registres rs i rt, tractant-los com a enters amb signe. Si rs=rt, s'executa la trap

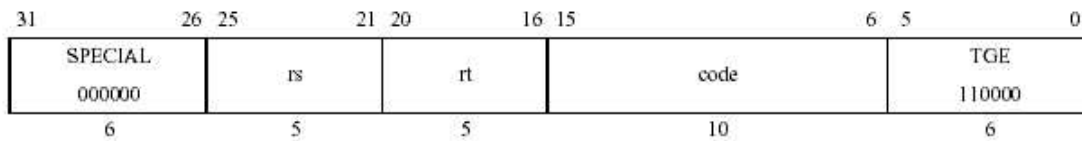
TEQI



Format: TEQI rs,rt

Descripció: Compara el contingut del register rs amb el valor immediate "immediate". Si rs=immediate, tractats com a enter amb signe, s'executa la trap.

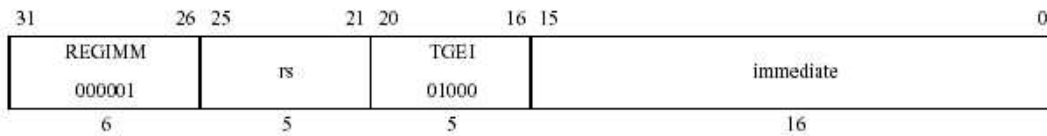
TGE



Format: TGE rs,rt

Descripció: Comparem rs amb rt com a enters amb signe. Si $rs \geq rt$, s'executa la trap.

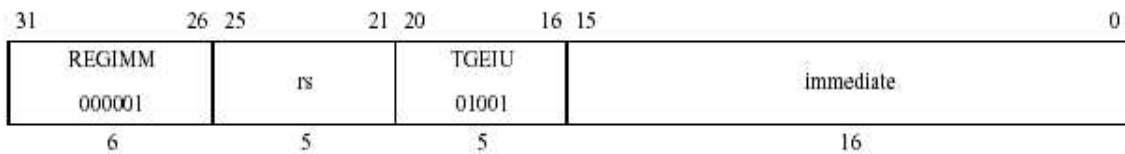
TGEI



Format: TGEI rs,immediate

Descripció: Comparem el contingut del register rs amb el valor immediat "immediate" com a enters amb signe. Si $rs \geq$ "immediate", s'executa la trap.

TGEIU

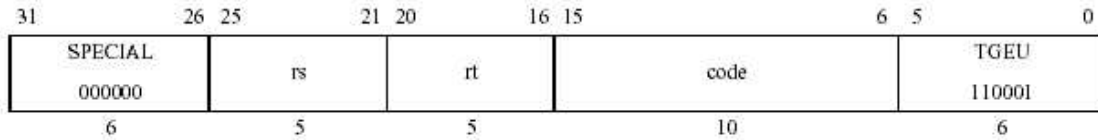


Format: TGEIU rs, immediate

Descripció: if $rs \geq$ immediate then trap.

Comparem el register rs amb els 16 bits de signe estès d'"immediate" com a enters sense signe. Si $rs \geq$ "immediate", s'executa l'excepció de trap.

TGEU

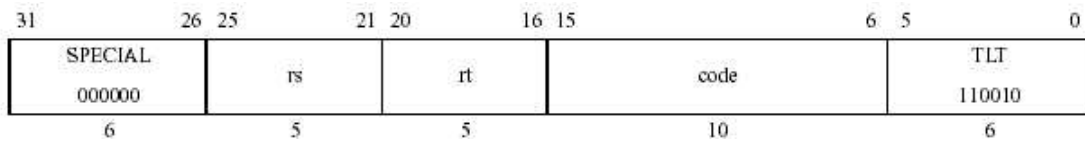


Format: TGEU rs,ru

Descripció: if rs>rt then trap

Comparem rs i rt com a enters sense signe. Si $rs \geq rt$, s'executa la trap.

TLT

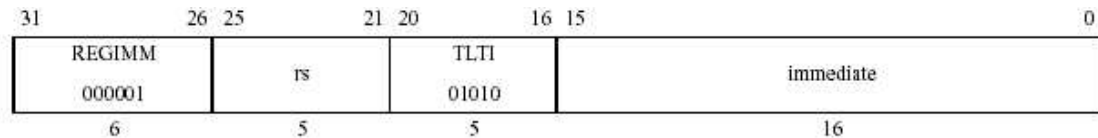


Format: TLT rs,rt

Descripció: if rs<rt then trap

Comparem rs amb rt com a enters amb signe. Si $rs < rt$, s'executa la trap.

TLTI

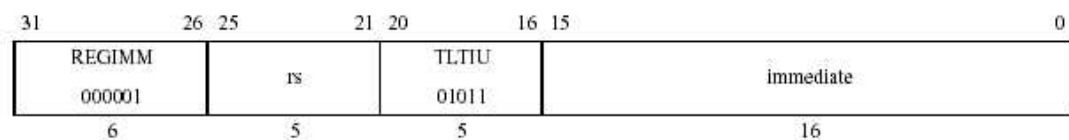


Format: TLTI rs, immediate

Descripció: if rs<"immediate" then trap

Comparem rs amb els 16 bits d'"immediate" com a enter amb signe. Si $rs < \text{"immediate"}$ s'executa la trap.

TLTIU

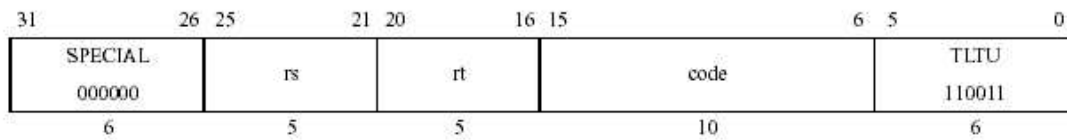


Format: TLTIU rs,immediate

Descripció: if rs<immediate then trap

Comparem el contingut del register rs amb els 16 bits amb signe extès d'"immediate" com a enters sense signe. Si $rs < \text{immediate}$, s'executa la trap.

TLTU

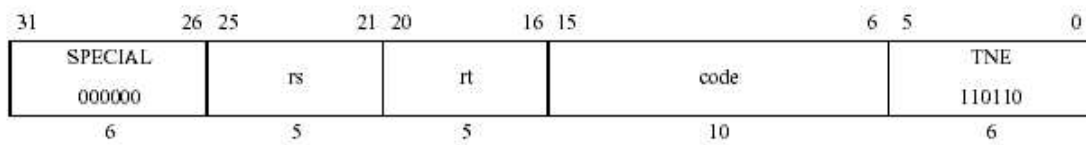


Format: TLTU rs,rt

Descripció: if rs<rt then trap

Comparem rs i rt com a enters sense signe. Si $rs < rt$, executarem la trap.

TNE



Format: TNE rs,rt

Descripció: if $rs <> rt$ then trap

Comparem rs amb rt com a enters amb signe. Si $rs <> rt$, s'executa la trap.