

Tema 3. Traducció de Programes

Estructura de Computadors (EC)

Rubèn Tous

rtous@ac.upc.edu
Computer Architecture Department
Universitat Politècnica de Catalunya



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índex

- 1 3.1 Desplaçaments de bits
- 2 3.2 Operacions lògiques bit a bit
- 3 3.3 Comparacions i operacions booleanes
- 4 3.4 Salts
- 5 3.5 Sentències alternatives if-then-else

3.1.1 Desplaçaments lògics a esquerra i dreta

Instruccions sll (Shift Left Logical) i srl (Shift Right Logical):

```
1 sll rd, rt, shamt # rt << shamt  
2 srl rd, rt, shamt # rt >>> shamt
```

shamt = shift amount (natural de 5 bits).

3.1.1 Desplaçaments lògics a esquerra i dreta

Exemple sll:

```
1 li $t0, 0x88888888
2 sll $t0, $t0, 1 # resultat: $t0 = 0x11111110
```

Exemple srl:

```
1 li $t0, 0x99999999
2 srl $t0, $t0, 2 # resultat: $t0 = 0x26666666
```

3.1.2 Desplaçament aritmètic a la dreta

Instrucció sra (Shift Right Arithmetic):

```
1 sra rd , rt , shamt
```

Exemple:

```
1 li $t0 , 0x99999999  
2 sra $t0 , $t0 , 2    # resultat: $t0 = 0xE6666666
```

3.1.3 Repertori d'instruccions de desplaçament

Resum instruccions de desplaçament de bits:

sll/srl/sra/sllv/srlv/srav			
sll	rd, rt, shamt	$rd = rt \ll shamt$	
srl	rd, rt, shamt	$rd = rt \gg shamt$	Inserta zeros a l'esquerra
sra	rd, rt, shamt	$rd = rt \gg shamt$	Extén signe a l'esquerra
sllv	rd, rt, rs	$rd = rt \ll rs_{4:0}$	
srlv	rd, rt, rs	$rd = rt \gg rs_{4:0}$	Inserta zeros a l'esquerra
srav	rd, rt, rs	$rd = rt \gg rs_{4:0}$	Extén signe a l'esquerra

3.1.3 Repertori d'instruccions de desplaçament

- Operador \ll : Instrucció sll o sllv.
- Operador \gg : Instrucció srl/srlv si unsigned (natural) o sra/srva altrament.

Exemple:

```
1 a = (a << b) >> 2;
```

```
1 sllv $t4, $t0, $t1  
2 sra  $t0, $t4, 2
```

3.1.5 Aplicacions: multiplicació i divisió per potències de 2

- Desplaçar a l'esquerra un enter/natural (sll) equival a multiplicar-lo per 2^{shamt} .
- Desplaçar a la dreta un natural (srl) o un enter (sra) equival a dividir-lo per 2^{shamt} .
- Però si el dividend és negatiu i senar, la instrucció sra donarà diferent que la instrucció div. Exemple $-7/2$:

En 4 bits, $-7 = 1001$.

$1001 \gg 1 = 1100$ (-4).

- Tot i que també satisfà la condició $dividend = divisor * quocient + residu$ si el residu és positiu ($-7 = 2 * -4 + 1$).

3.1.5 Aplicacions: multiplicació i divisió per potències de 2

- Però en C la divisió entera sempre retorna un residu del mateix signe que el dividend ($-7 = 2 * -3 + (-1)$).
- Per això no usarem sra per dividir si el dividend és negatiu i imparell.

3.2.1 Operacions and, or, xor, i not bit a bit

Exemples en C:

```
1 a = a & b; //bitwise and
2 a = a | b; //bitwise or
3 a = a ^ b; //bitwise xor
4 a = ~a;    //bitwise not
```

En MIPS:

```
1 and $t0 , $t0 , $t1
2 or  $t0 , $t0 , $t1
3 xor $t0 , $t0 , $t1
4 nor $t0 , $t0 , $zero
```

3.2.2 Repertori d'instruccions lògiques bit a bit

Resum instruccions lògiques bit a bit:

and/or/xor/nor/andi/ori/xori		
and rd, rs, rt	rd = rs AND rt	
or rd, rs, rt	rd = rs OR rt	
xor rd, rs, rt	rd = rs XOR rt	
nor rd, rs, rt	rd = rs NOR rt = NOT (rs OR rt)	
andi rt, rs, imm16	rt = rs AND ZeroExt(imm16)	imm16 ha de ser un natural
ori rt, rs, imm16	rt = rs OR ZeroExt(imm16)	imm16 ha de ser un natural
xori rt, rs, imm16	rt = rs XOR ZeroExt(imm16)	imm16 ha de ser un natural

3.2.3 Operació and

Aplicació de la and bit a bit: Seleccionar bits (posant la resta a zero).

```
1 andi $t0 , $t0 , 0xFFFF
```

3.2.3 Operació and

Exemple: comprovar si la variable *b* té actius els bits 0 i 4, i inactius els bits 2 i 6:

```
1  a = b & 0x55;  
2  if (a == 0x11)  
3      {...}
```

```
1  andi $t0, $t1, 0x0055  
2  li    $t4, 0x0011  
3  bne   $t0, $t4, endif  
4  ...   # codi si cert  
5  endif:
```

3.2.4 Operacions or

Aplicació de la or bit a bit: posar bits a u.

```
1 ori $t0 , $t0 , 0xFFFF
```

3.2.5 Operacions xor

Aplicació de la xor bit a bit: complementar bits.

```
1 li    $t1, 0x55555555 # bits parells  
2 xor   $t0, $t0, $t1
```

3.3 Comparacions i operacions booleanes

- En C no existeix el tipus booleà, es fa servir un enter.
- 0 = fals
- Diferent de 0 = cert.
- No obstant, els operadors C retornen un booleà **normalitzat**, retornant 1 si cert.

3.3 Comparacions i operacions booleanes

Operadors C que retornen un valor booleà:

- Comparació d'enters/naturals: ==, !=, <, <=, >, >=.
- Operadors booleanes: &&, ||, !.

3.3.1 Repertori d'instruccions de comparació

MIPS només implementa l'operació $<$ (retorna 0 o 1).

slt/sltu/slti/sltiu			
slt	rd, rs, rt	rd = rs < rt	comparació d'enters
sltu	rd, rs, rt	rd = rs < rt	comparació de naturals
slti	rd, rs, imm16	rd = rs < Sext(imm16)	comparació d'enters
sltiu	rd, rs, imm16	rd = rs < Sext(imm16)	comparació de naturals

3.3.2 Comparació <

Exemple

1 *c = a < b // a i b a \$t0 i \$t1 respectivament. c a \$t2.*

```
1 sltu $t2, $t0, $t1 # si a, b naturals  
2 slt  $t2, $t0, $t1 # si a, b enters
```

Mitjançant l'operació < i les operacions bit a bit traduirem totes les comparacions i les operacions booleanes.

3.3.3 Traducció de la negació booleana: !v

Negació booleana:

1 !a

1 `sltiu $t1, $t0, 1`

3.3.3 Traducció de la negació booleana: !v

Si el valor a negar està normalitzat també podem fer servir xori:

```
1 xori $t2 , $t4 , 1
```

Exemple $c = \neg(a < b)$:

```
1 slt  $t4 , $t0 , $t1  
2 xori $t2 , $t4 , 1
```

3.3.4 Traducció de les operacions booleanes AND i OR

Normalització d'un booleà (a \$t0)

```
1 sltu $t0, $zero, $t0
```

3.3.4 Traducció de les operacions booleanes AND i OR

AND (&&) booleana:

1 `c = a && b` // *a i b a \$t0 i \$t1 respectivament. c a \$t2.*

Normalitzem i fem and bit a bit:

```
1 sltu $t0, $zero, $t0
2 sltu $t1, $zero, $t1
3 and  $t2, $t0, $t1
```

3.3.4 Traducció de les operacions booleanes AND i OR

OR booleana:

1 `c = a || b` // *a i b a \$t0 i \$t1 respectivament. c a \$t2.*

No cal normalitzar a i b, però sí cal normalitzar el resultat:

```
1 or    $t2, $t0, $t1
2 sltu  $t2, $zero, $t2
```


3.3.4 Traducció de les operacions booleanes AND i OR

Traducció d'operacions booleanes mitjançant salts

Sovint no ens caldrà emmagatzemar el resultat d'una comparació o operació booleana, simplement avaluar-la dins la condició d'un if o un while. En aquests casos, l'hauem de traduir mitjançant salts.

```
1  if (a && b)
2      CODI
```

```
1  beq $t1, $zero, fisi
2  beq $t2, $zero, fisi
3  CODI
4  fisi:
```

3.3.4 Traducció de les operacions booleanes AND i OR

- Avaluació lazy (gandula)
- Ho veurem a la secció 5.2
- Si el valor del primer operand ja determina el valor de tota la condició, el segon operand no s'ha d'avaluar.
- **No és opcional!**

3.3.5 Traducció de les comparacions $>$, \leq , \geq

1 `c = (a > b) // a i b a $t0 i $t1 respectivament. c a $t2.`

```
1 slt $t2, $t1, $t0 # c = b < a
```

3.3.5 Traducció de les comparacions $>$, \leq , \geq

1 *c = (a <= b) // a i b a \$t0 i \$t1 respectivament. c a \$t2.*

```
1 slt    $t4, $t1, $t0 # calcula (b<a)
2 sltiu  $t2, $t4, 1   # NOT bool. !(b<a)
```

3.3.5 Traducció de les comparacions $>$, \leq , \geq

1 *c = (a >= b) // a i b a \$t0 i \$t1 respectivament. c a \$t2.*

```
1 slt    $t4, $t0, $t1 # calcula (a<b)
2 sltiu  $t2, $t4, 1   # NOT bool. !(a<b)
```

3.3.6 Traducció de les comparacions == i !=

1 *c = (a == b) // a i b a \$t0 i \$t1 respectivament. c a \$t2.*

```
1 sub    $t4, $t0, $t1 # si zero iguals
2 sltiu  $t2, $t4, 1    # NOT bool.
```

3.3.6 Traducció de les comparacions == i !=

1 *c = (a != b) // a i b a \$t0 i \$t1 respectivament. c a \$t2.*

```
1 subu $t4, $t0, $t1 # si zero iguals  
2 sltu $t2, $zero, $t4 # normalitzem el resultat
```

3.4.1 Salts condicionals relatius al PC (branch): beq i bne

- MIPS només implementa els salts condicionals **beq** i **bne**.
- **beq \$t1, \$t2, etiq** salta a etiq si $\$t1 == \$t2$.
- **bne \$t1, \$t2, etiq** salta a etiq si $\$t1 \neq \$t2$.
- **beq \$zero, \$zero, etiq** = salt incondicional = macro **b**.

3.4.1 Salts condicionals relatius al PC (branch): beq i bne

- Codifiquem la distància a saltar respecte al PC en un immediat de 16 bits.
- Número d'instruccions.
- Respecte a l'adreça de la instrucció següent al salt ($PC_{cup} = PC + 4$).
- Permet saltar dins el rang de $[-2^{15}, 2^{15} - 1]$ instruccions de distància respecte al PC_{cup} .

3.4.1 Salts condicionals relatius al PC (branch): beq i bne

beq/bne i la macro b		
beq rs, rt, label	si (rs==rt) $PC = PC_{up} + \text{Sext}(\text{offset}16*4)$	
bne rs, rt, label	si (rs!=rt) $PC = PC_{up} + \text{Sext}(\text{offset}16*4)$	
b label	$PC = PC_{up} + \text{Sext}(\text{offset}16*4)$	beq \$0,\$0 label

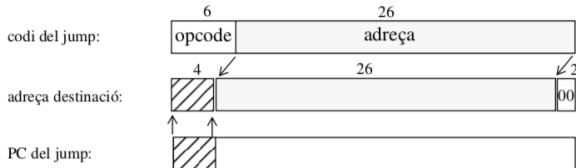
3.4.2 Altres macros per a salts condicionals relatius al PC: blt, bgt, bge, ble

macros blt/bgt/bge/ble/bltu/bgtu/bgeu/bleu			
blt rs, rt, label	si (rs<rt) saltar a label	slt \$at, rs, rt	bne \$at, \$zero, label
bgt rs, rt, label	si (rs>rt) saltar a label ¹	slt \$at, rt, rs	bne \$at, \$zero, label
bge rs, rt, label	si (rs>=rt) saltar a label ²	slt \$at, rs, rt	beq \$at, \$zero, label
ble rs, rt, label	si (rs<=rt) saltar a label ³	slt \$at, rt, rs	beq \$at, \$zero, label

3.4.3 Salts incondicionals en mode registre o pseudodirecte: j, jr, jal, jalr

- **j etiq**
- Format J (també **jal**, que veurem més endavant).
- Adreça destinació en 26 bits usant mode pseudodirecte.
- Quan s'executa, la CPU completa els 6 bits que falten.
- Zeros els 2 bits de menor pes.
- Els 4 bits de major pes es copien dels del registre PC.
- Saltem dins un bloc de 2^{28} bytes (256MB).

3.4.3 Salts incondicionals en mode registre o pseudodirecte: j, jr, jal, jalr



3.4.2 Altres macros per a salts condicionals relatius al PC: blt, bgt, bge, ble

j/jr/jal/jalr		
j target	PC = target	Jump, mode pseudodirecte
jr rs	PC = rs	Jump, mode registre
jal target	PC = target; \$ra = PC _{up}	Jump and Link, mode pseudodirecte
jalr rs, rd	PC = rs; rd = PC _{up}	Jump and Link, mode registre

3.4.3 Salts incondicionals en mode registre o pseudodirecte: j, jr, jal, jalr

Si el rang de salt de **j** és insuficient:

```
1 la $t0, etiqueta_llunyana  
2 jr $t0
```

3.5.1 Sentència if-then-else

```
1  if (condicio)
2      sentencia_then
3  else
4      sentencia_else
```

El patró en MIPS seria:

```
    avaluar condicio
    salta si és falsa a sino
    traducció de sentencia_then
    salta a fisi
sino:
    traducció sentencia_then
fisi:
```


3.5.1 Sentència if-then-else

Exemple (suposant que a, b, c, d són enters i ocupen \$t0, \$t1, \$t2, \$t3): :

```
1  if (a >= b)
2      d = a;
3  else
4      d = b;
```

```
1  blt $t0, $t1, sino
2  move $t3, $t0
3  b fisi
4  sino:
5  move $t3, $t1
6  fisi:
```

3.5.2 Avaluació 'lazy' dels operadors booleanes AND i OR

- En C, els operadors `&&` i `||` s'avaluen d'esquerra a dreta de forma 'lazy'.
- Si la part esquerra ja determina el resultat, la part dreta NO s'ha d'avaluar.

3.5.2 Avaluació 'lazy' dels operadors booleanes AND i OR

Avaluació lazy d'una AND:

```
1  if ( a >= b && a < c )  
2    d = a;  
3  else  
4    d = b;
```

```
1  blt $t0, $t1, sino # macro. salta si a<b  
2  bge $t0, $t2, sino # macro. salta si a>=c  
3  move $t3, $t0  
4  b fisi  
5  sino:  
6  move $t3, $t1  
7  fisi:
```

3.5.2 Avaluació 'lazy' dels operadors booleanes AND i OR

Avaluació lazy d'una OR:

```
1  if ( a >= b || a < c )  
2    d = a;  
3  else  
4    d = b;
```

```
1  bge $t0, $t1, llavors # macro. salta si a>=b  
2  bge $t0, $t2, sino    # macro. salta si a>=c  
3  llavors:  
4  move $t3, $t0  
5  b fisi  
6  sino:  
7  move $t3, $t1  
8  fisi:
```

3.5.2 Avaluació 'lazy' dels operadors booleanes AND i OR

Traducció de condicions complexes:

- Abans d'un && saltem si FALS.
- Abans d'un || saltem si CERT.
- Al final de l'expressió sempre saltem si FALS.

3.5.2 Avaluació 'lazy' dels operadors booleanes AND i OR

```
1  if ((A&&B)|| (C&&D)) {  
2      S;  
3  }
```

Si A FALS salta a OR

Si B CERT salta a LLAVORS

OR:

Si C FALS salta a FISI

Si D FALS salta a FISI

LLAVORS:

S

FISI:

3.5.2 Avaluació 'lazy' dels operadors booleanes AND i OR

```
1  if ((A||B)&&(C||D)) {  
2      S;  
3  }
```

Si A CERT salta a AND

Si B FALS salta a FISI

AND:

Si C CERT salta a LLAVORS

Si D FALS salta a FISI

LLAVORS:

S

FISI: