Estructura de Computadors

Tema 4
Aritmètica en coma flotant

DISCA

Índex

- Introducció
  - Mesura de rendiment
- La norma IEEE i la seua implementació en el MIPS
  - Format, valors especials, arrodoniment
  - Visió del programador: banc de registres i moviment de dades
  - Joc d'instruccions de coma flotant
- Operadors de coma flotant
  - Operador de canvi de signe
  - Operadors de conversió de tipus
  - Operador de multiplicació

Bibliografia

D.L. Patterson, J. L. Hennessy: Estructura y diseño de computadores
- Ed. Reverté, 2000: volum 1, capítol 4
- Ed. Reverté, 2011, traducció de la 4a edició en anglés: cap. 3

W. Stallings: Organización y Arquitectura de Computadores (7a ed.) Prentice Hall, capítol 9

David Goldberg: Computer Arithmetic
- Appèndix H de J. L. Hennessy, D. L. Patterson: Computer Architecture, a Quantitative Approach, 3a edició
- Disponible en castellà en la 1a edició en McGraw-Hill

– David Goldberg: What every computer scientist should know about floating-point arithmetic
- PDF, (accessible en molts llocs de la web)

Introducció

Introducció

- L'aritmètica de coma flotant
  - Aprofita per a càlculs definits sobre reals (tipus *float* i *double* en alt nivell)
  - Pot ser que no estiga directament suportada per la ALU
    - no és essencial per al funcionament de la CPU
    - pot ser emulada mitjançant instruccions d'enters
  - Evolució
    - Fins a 1985 (aprox) els operadors de CF anaven dins d'un xip opcional que es col·locava al costat de la CPU
    - Des de 1985 en endavant, les CPU de propòsit general inclouen operadors de CF dins de la seua ALU
    - Des de 1990, els adaptadors de vídeo inclouen una GPU (*Graphics Processing Unit*) amb un nombre creixent d'operadors de CF

Introducció

- La mesura de prestacions en coma flotant
  - El nombre d'operacions de coma flotant per segon (FLOPS, prefijxe M=10^6, G=10^9, T=10^{12}) és una mesura de prestacions utilitzada en dos contextos:
  - En el disseny d'operadors de CF és el nombre màxim d'operacions per segon. Ve determinada pel temps d'operació de cada circuit.
  - En les comparatives entre computadors o entre acceleradors gràfics: és el nombre d'operacions de CF que executa el dispositiu per segon.
    - Depén del nombre i característiques dels operadors que inclou i de l'ús que se'n fa.
    - **Productivitat punta** d'un computador o d'un accelerador gràfic: suma de les productivitats dels operadors de CF que conté. Sol ser impossible d'assolir amb l'ús corrent

Introducció

- **Productivitats punta**

Processador Intel Core2 Duo @2GHz
16 GFLOPS

Processador Intel Core i7 965 XE
70 GFLOPS

GPU ATI Radeon HD4890
2.4 TFLOPS

K Computer (Japó, juny 2011)
10 PFLOPS

La norma IEEE 754 i la seua implementació en el MIPS

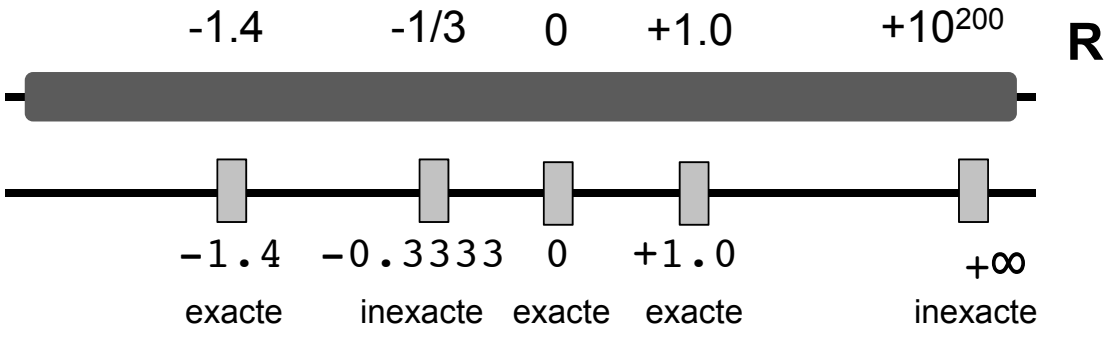
DISCA

Notes sobre la representació de nombres

- Representació del conjunt R (reals positius i negatius)
  - El conjunt R és un conjunt dens: entre dos nombres reals qualssevol hi ha infinits nombres reals
  - La representació del computador és limitada i no sempre és exacta
    - Amb 32 bits es poden obtenir 2^{32} paraules diferents. Per tant, com a màxim es poden representar 2^{32} valors del conjunt R
  - Hi ha nombres reals que tenen representació exacta i altres que no, com ara els nombres amb part decimal periòdica
  - Com codificar un nombre real en una paraula de bits?
    - Aplicant-hi un format arbitrari, com ara el IEEE 754, estructurat en tres camps de bits per al signe, l'exponent i la part significativa (mantissa)
  - El format imposa més restriccions a la representació: hi haurà algunes paraules de bits amb una significació matemàtica especial, com ara el valor infinit o el zero

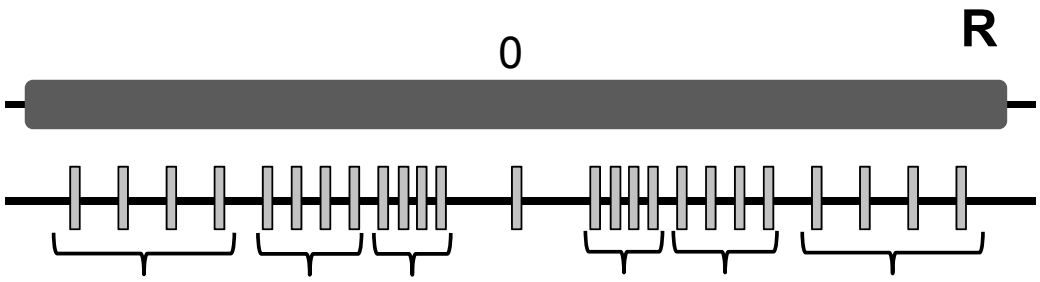
Notes sobre la representació de nombres

- Representació del conjunt R (reals positius i negatius)
  - En el computador interessa augmentar
    - La quantitat de nombres representats (densitat)
    - El rang de la representació
  - Aquests dos aspectes depenen dels camps de la part significativa (mantissa) i de l'exponent del format



Notes sobre la representació de nombres

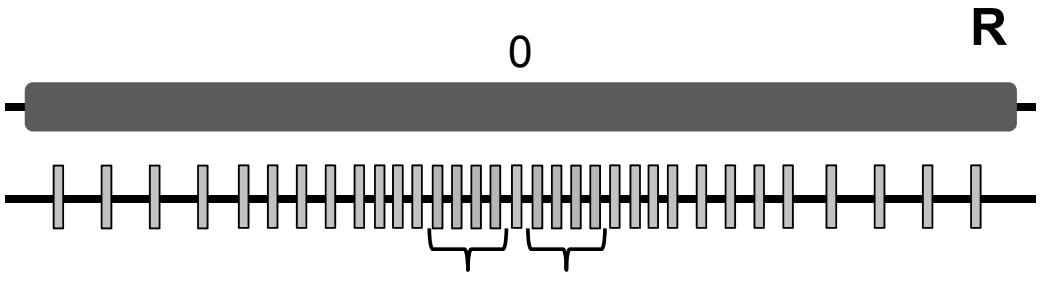
- Patró de la representació dels nombres reals
  - No hi ha valors representats molt a prop del zero
  - Per a un mateix valor d'exponent, els nombres representats estan separats per la mateixa distància
  - Com més gran és l'exponent més distància hi ha entre dos nombres representats consecutius (la densitat de representació disminueix)



Cada grup de valors té el
mateix exponent i diferents mantisses

Notes sobre la representació de nombres

- Els valors prop de zero
  - El format IEEE 754 reserva un subconjunt de paraules de bits per a representar nombres reals prop del zero i que s'interpreten de forma diferent de la resta de valors (valors desnormalitzats)
  - No totes les unitats de coma flotant suporten aquest subconjunt de valors



Valors especials per a
treballar prop del zero
(mateix exponent)

La norma IEEE 754

- Abast de la norma
  - La norma IEEE 754 (o la seua ampliació a l'aritmètica de punt fix, IEEE 854) especifica:
    - **codificació**: com representar els nombres en diversos formats (precisions simple, doble i estesa, SP DP EP) i el tractament de casos particulars: NaN *(Not a Number)*, ±∞ *(infinity)*, 0 *(zero)*
    - un conjunt d'**operacions** que es poden implementar en el hardware o en forma de biblioteques
    - uns modes de funcionament (p. ex, el mètode d'**arrodoniment** aplicable durant els càlculs)
    - el suport que ha de donar el sistema d'**excepcions** dels processadors (perquè s'hi puguen dissenyar bones biblioteques de càlcul numèric)

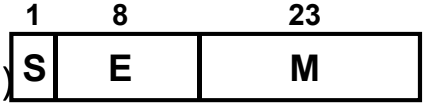
La norma IEEE 754 (repàs)

- Representació

  Símbols: S és el signe i M la magnitud de la mantissa. E és l'exponent

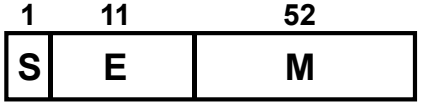
  – Formats:

  - Simple precisió (SP)



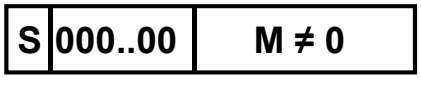
$(-1)^S \cdot 1.M \cdot 2^{E-127}$

  - Doble precisió (DP)



$(-1)^S \cdot 1.M \cdot 2^{E-1023}$

  – Valors denormalitzats
  - (SP i DP)

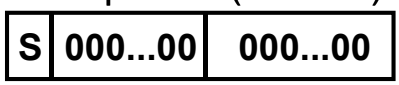


$(-1)^S \cdot 0.M \cdot 2^{-126}$

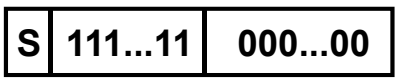
$(-1)^S \cdot 0.M \cdot 2^{-1022}$

  – Valors especials (SP i DP)

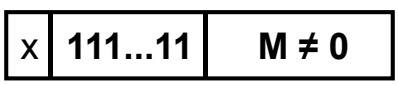
±0



±∞



NaN



La norma IEEE 754

- Els valors especials
  - Són manipulats per les operacions junt amb els valors corrents
  - Zero i infinit:
    - s'entenen com a límits matemàtics; per això
      - $+\infty + +\infty = +\infty$; $-\infty + -\infty = -\infty$; etc.
      - $+\infty \times$ positiu $= +\infty$; $+\infty \times$ negatiu $= -\infty$; etc.
      - positiu / $+0 = +\infty$; positiu / $-0 = -\infty$; etc
    - comparacions: +0 i –0 són iguals
  - Not a number:
    - propagació: qualsevol operació on un operand és NaN donarà com a resultat NaN
    - generació: NaN és el resultat de $(+\infty) + (-\infty)$, $\pm 0 \times \pm\infty$, $\pm 0 / \pm 0$, $\pm\infty / \pm\infty$ i altres
    - una comparació (=, <, ≥, etc) entre NaN i altre número resulta falsa

La norma IEEE 754

- La norma i els llenguatges de programació
  - Els valors especials permeten tractar els incidents del càlcul
  - El desbordament aritmètic produeix un resultat representable

Java

```
float x = +0.0f;
float y = 1/x;
float z = Float.NEGATIVE_INFINITY;
float t = 1/z;
float u = x*z;
System.out.println("x = " + x);
System.out.println("1/x = " + y);
System.out.println("z = " + z);
System.out.println("1/z = " + 1/z);
System.out.println("x * z = " + u);
```

```
x = 0.0
1/x = Infinity
z = -Infinity
1/z = -0.0
x * z = NaN
```

La norma IEEE 754

- L'arrodoniment
  - Situació freqüent: una operació genera una mantissa M de longitud més llarga (*p* bits) que la prevista en el format (*m* bits)
    - els *m* primers bits de la mantissa M s'anomenen *retinguts*
  - Possibilitats:
    - M és representable de forma *exacta* en el format: els *p–m* bits no retinguts són 0 i es poden eliminar: **010000** $\rightarrow$ **0100**
    - M es troba entre els dos valors representables M_- i M_+ ($M_- < M < M_+$) i cal *arrodonir*: cal triar-ne un com a representació *inexacta* de M
  - La norma admet quatre *modes d'arrodoniment*:
    - *Cap a* $+\infty$
    - *Cap a* $-\infty$
    - *Cap a 0*
    - Triar el més pròxim dels dos (esbiaixat al parell; aquest és el mode per omissió)

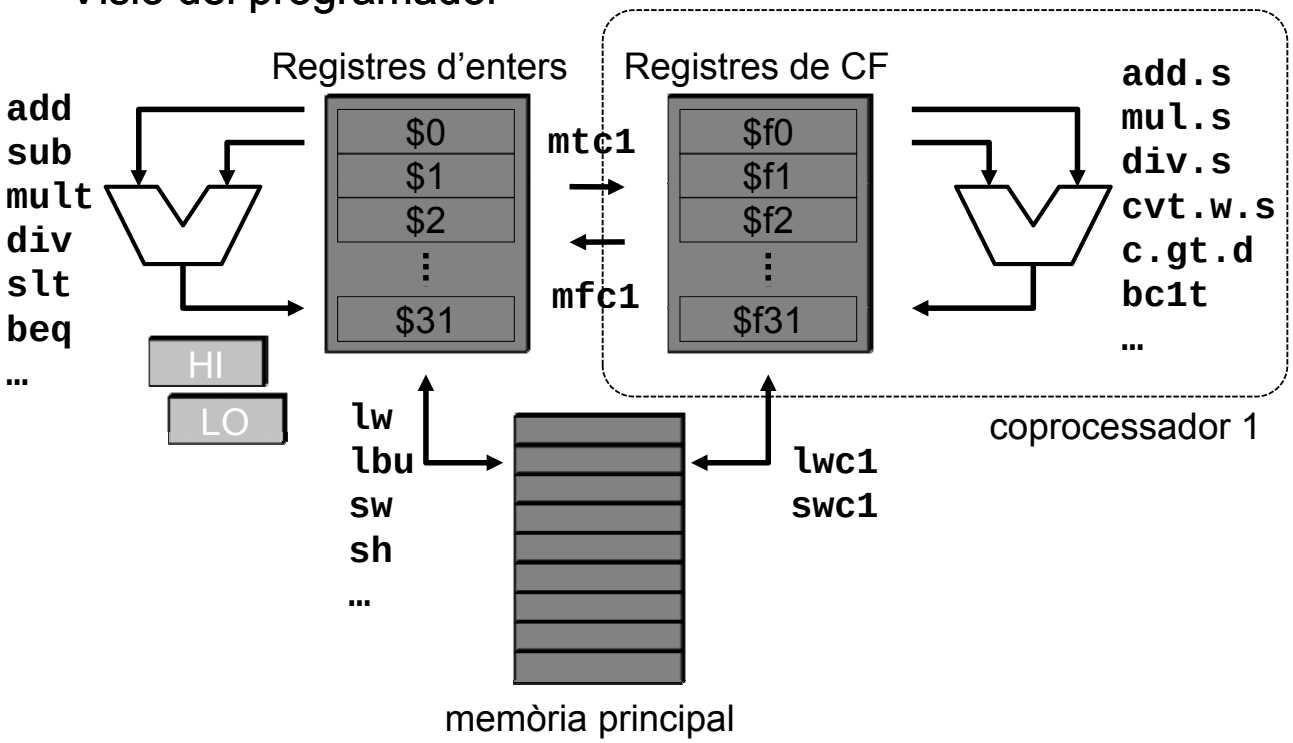
La norma IEEE 754

- L'arrodoniment cap al més pròxim (esbiaixat al parell)
  - La variant per omissió és "tie to even": en cas que M estiga equidistant de M_- i M_+, cal triar la mantissa representable parella (o siga, la que acabe en 0)
  - Exemple:

M	tria	M resultant
010000	(exacta)	**0100**
010001	M_- (més propera)	**0100**
010010	M_- (parella)	**0100**
010011	M_+ (més propera)	**0101**
010100	(exacta)	**0101**
010101	M_- (més propera)	**0101**
010110	M_+ (parella)	**0110**
010111	M_+ (més propera)	**0110**
011000	(exacta)	**0110**

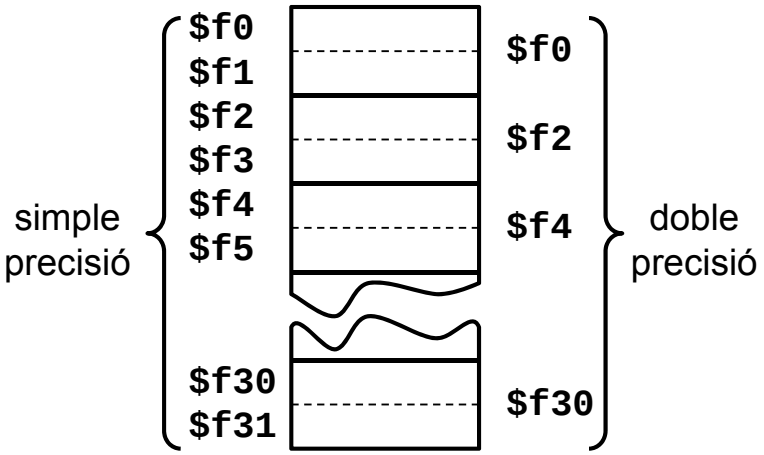
La coma flotant en el MIPS

- Visió del programador



La coma flotant en el MIPS

- El banc de registres
  - Hi ha 32 registres de 32 bits **`$f0,$f1,...,$f31`** per a tipus float
    - Se solen emprar els nombres parells **`$f0,$f2,...,$f30`**
  - Emparellables per a formar 16 registres de 64 bits per a tipus double
    - Si **`$f0`** "conté" un double: **`$f0`** té la part baixa i **`$f1`** la part alta (**`$f1`**||**`$f0`**)



La coma flotant en el MIPS R2000

- Conveni d'ús dels registres

Nom del registre	Utilització
`$f0`	Retorn de funció (part real)
`$f2`	Retorn de funció (part imaginària)
`$f4,$f6,$f8,$f10`	Registres temporals
`$f12,$f14`	Pas de paràmetres a funcions
`$f16,$f18`	Registres temporals
`$f20,$f22,$f24,$f26,$f28,$f30`	Registres a mantenir entre crides

La coma flotant en el MIPS

- Intercanvi amb la memòria i amb els registres d'enters

operació	instrucció
lectura *$ft* ← *Mem[X+$rs]*	`lwc1 $ft,X($rs)`
escriptura *Mem[X+$rs]* ← *$ft*	`swc1 $ft,X($rs)`
transferència *$fs* ← *$rt*	`mtc1 $rt,$fs`
transferència *$rt* ← *$fs*	`mfc1 $rt,$fs`

fs i ft: registres de coma flotant
rs i rt: són registres d'enters

```
        .data
x:      .float 3.14
y:      .double 0.1
        .text
        la $t0,x        # f0 <- x
        lwc1 $f0,0($t0)
        la $t0,y        # f2 <- y
        lwc1 $f2,0($t0)
        lwc1 $f3,4($t0)
        mtc1 $0,$f4     # f4 <- 0.0
```

Les instruccions de CF no admeten operands immediats. Cal ubicar les constants en la memòria o construir-les en els registres d'enters

La coma flotant en el MIPS

- Conversió de formats
  - Els registres de CF poden contenir:

símbol	*tipus*
S	nombres en coma flotant en SP
D	(per parelles) nombres en coma flotant en DP
W	nombres enters de 32 bits

  - La instrucció **`cvt._._ fd,fs`** fa les conversions possibles entre els tres tipus
    - Exemple: **`cvt.d.w $f4,$f7`** fa la conversió de l'enter contingut en **`$f7`** a CF en doble precisió contingut en **`$f4`**||**`$f5`**
  - En combinació amb les instruccions de transferència amb el banc de registres d'enters, es pot fer aritmètica amb variables de tipus diversos

La coma flotant en el MIPS

- Instruccions aritmètiques bàsiques
  - N'hi ha dues versions de cada operació: S (simple precisió) i D (doble precisió)
    - Exemple: **`add.s $f0,$f1,$f2`**; **`add.d $f2,$f4,$f6`**

operació	instrucció
suma	**`add._ fd,fs,ft`**
resta	**`sub._ fd,fs,ft`**
multiplicació	**`mul._ fd,fs,ft`**
divisió	**`div._ fd,fs,ft`**
comparació	**`c.`*`cond`*`._ fs,ft`**
copia	**`mov._ fd,fs`**
canvi de signe	**`neg._ fd,fs`**
valor absolut	**`abs._ fd,fs`**

La coma flotant en el MIPS

- La comparació
  - Les instruccions de comparació escriuen un bit implícit ***FPc*** que codifica cert=1 i fals=0
  - Aquest bit es troba en un registre de control del coprocessador i pot ser consultat per les instruccions de salt
  - Per a cada tipus de dades, n'hi ha un conjunt de comparacions codificables
  - Les més importants: (**`c.__.s fd,fs`** o **`c.__.d fd,fs`**)

fd>fs	fd=fs	fd<fs
`gt`	`eq`	`lt`
`le`	`neq`	`ge`
fd≤fs	fd≠fs	fd≥fs

La coma flotant en el MIPS

- Control de flux i aritmètica de coma flotant
  - Hi ha dues instruccions de bifurcació associades al bit FPc

    `bc1t` ***`eti`*** si (FPc == 1) bifurcar a *eti*

    `bc1f` ***`eti`*** si (FPc == 0) bifurcar a *eti*
  - Combinades amb les instruccions de comparació, permeten bifurcar amb condicions aritmètiques complexes
  - Cada condició permet dues implementacions
    - Exemple en simple precisió: *si ($f0 > $f2) bifurcar a eti*

```
; mirar si $f0>$f2
      c.gt.s $f0,$f2
; saltar si afirmatiu
      bc1t eti
```

```
; mirar si $f0<=$f2
      c.le.s $f0,$f2
; saltar si negatiu
      bc1f eti
```

Operadors de coma flotant

DISCA

Operadors de coma flotant

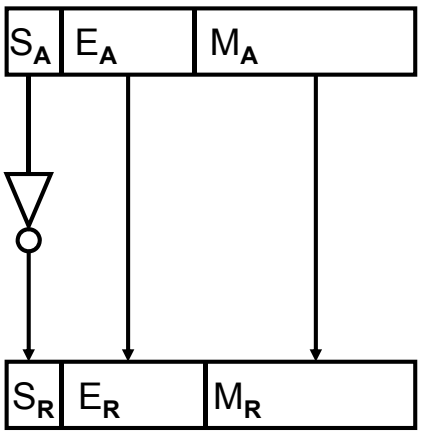
- Introducció
  - Prenen com a entrada un o dos operands en un format de CF donat
  - El seu resultat és un operand de CF codificat segons la norma
    - excepte els operands de comparació
  - El seu disseny és complexe perquè, a més de fer l'operació definida, s'han d'ocupar de certs detalls:
    - Han de subministrar el resultat correctament normalitzat segons la precisió amb què treballen
    - Han de gestionar els valors especials definits en la norma
    - Si escau, han d'arrodonir el resultat segons el mode programat
    - Han de senyalar les excepcions previstes per la norma
  - Estudiarem l'estructura bàsica d'alguns operadors i veurem, en casos seleccionats, com resolen els detalls

Operadors de coma flotant

- Exemples d'operadors
  - NEG.S i NEG.D (canvi de signe)
    - Estructura
  - CVT.D.S (conversió de simple a doble precisió)
    - Estructura bàsica
    - Detall: tractament dels valors especials
  - CVT.S.D (conversió de doble a simple precisió)
    - Estructura bàsica
    - Detall: l'arrodoniment
  - CVT.D.W (conversió d'enter a CF doble precisió)
    - Estructura bàsica
    - Detall: la normalització
  - MULT.S i MULT.D (multiplicació)
    - Estructura bàsica
    - Detall: la renormalització

El canvi de signe

- L'operador bàsic
  - Dues versions:
    - Simple precisió:
      - Entrada: S_A (1 bit), E_A (8 bits) i M_A (23 bits)
      - Eixida: S_R (1 bit), E_R (8 bits) i M_R (23 bits)
    - Doble precisió:
      - Entrada: S_A (1 bit), E_A (11 bits) i M_A (52 bits)
      - Eixida: S_R (1 bit), E_R (11 bits) i M_R (52 bits)
  - Canvia el signe: S_R = not S_A
  - Copia l'exponent: $E_R = E_A$
  - Copia la mantissa: $M_R = M_A$



Emulació del canvi de signe

```
float x = 1.0;

x = -x;
```

```
x:    .float 1.0

      lwc1 $f2, x         # $f2 <- x (1.0)
      mfc1 $t0,$f2        # $t0 <- $f2
      lui $t1, 0x8000     # $t1 <- 0x80000000
      xor $t0, $t0, $t1   # $t0 <- -1.0
      mtc1 $t0, $f2       # $f2 <- $t0
      swc1 $f2, x         # x <- $f2 (-1.0)
```

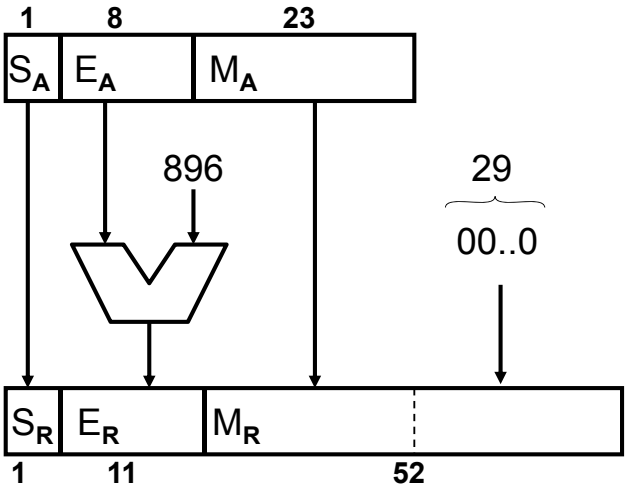

Conversió de simple a doble precisió (`cvt.d.s`)

- Especificació de l'operador:
  - Entrada: S_A (1 bit), E_A (8 bits) i M_A (23 bits)
  - Eixida: S_R (1 bit), E_R (11 bits) i M_R (52 bits)
  - El signe no canvia: $S_R = S_A$
  - Exponent: cal canviar d'excés 127 a excés 1023
    - $E_R = E_A + 896$
  - Mantissa: cal afegir 52–23=29 zeros a la dreta
    - $M_R = M_A$ || 00….0
  - Detall: tractament correcte dels valors especials:

	E_A	S_R	E_R	M_R
zero i subnormal:	00000000_2	S_A	00000000000_2	M_A || 00….0
±∞ i NaN:	11111111_2	S_A	11111111111_2	M_A || 00….0
valors corrents:	(resta de valors)	S_A	$E_A + 896$	M_A || 00….0

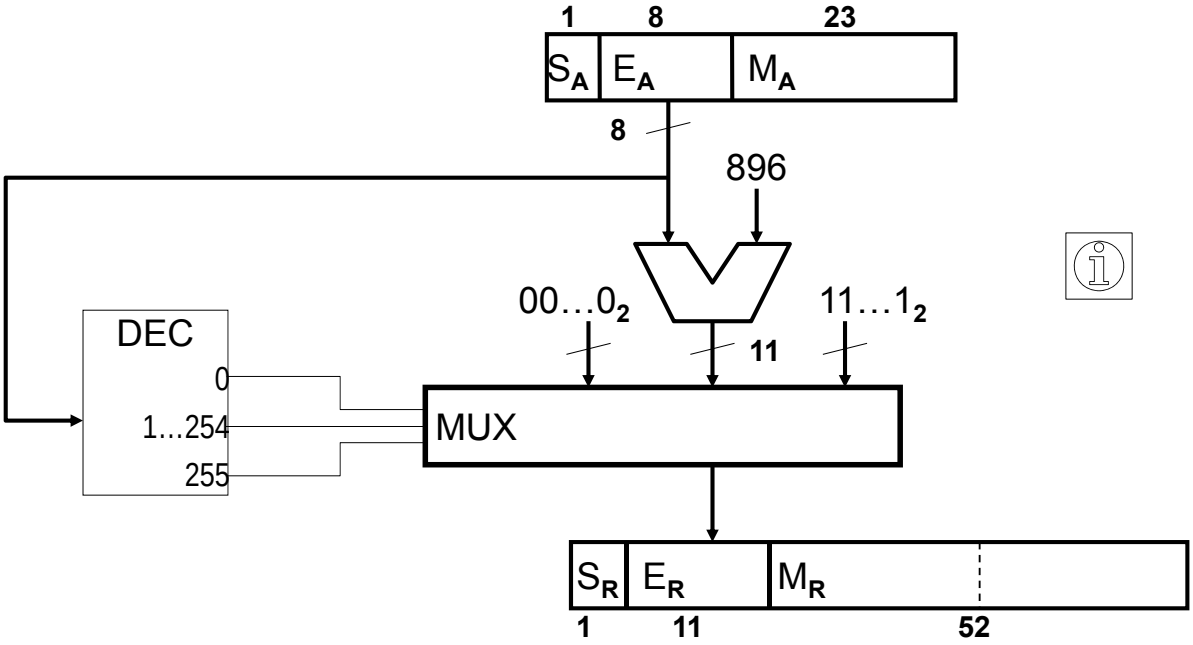
Conversió de simple a doble precisió (`cvt.d.s`)

- L'operador bàsic
  - No tracta els valors especials



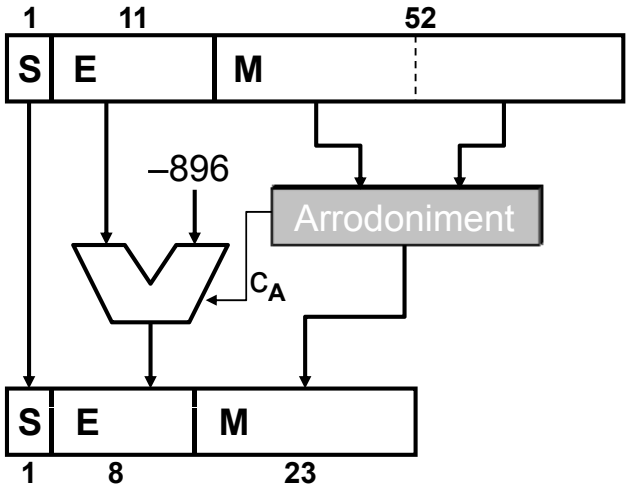
Conversió de simple a doble precisió

- Tractament dels valors especials
  - Detall del càlcul de l'exponent



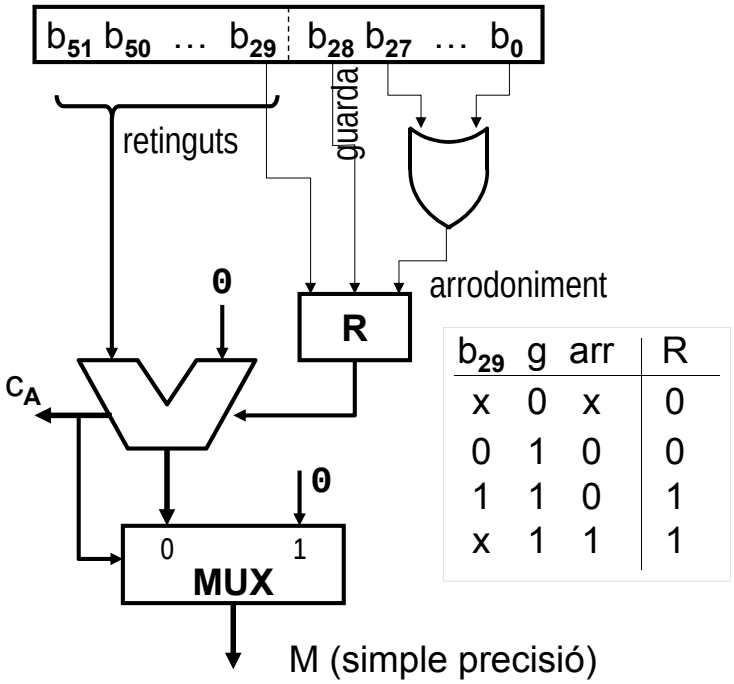
Conversió de doble a simple precisió (`cvt.s.d`)

- Estructura de l'operador
  - El signe no canvia
  - Exponent: cal canviar d'11 bits en excés 1023 a 8 bits en excés 127
    - pot donar-se desbordament
  - Mantissa: cal eliminar 29 bits per la dreta i arrodonir



L'arrodoniment

- Circuit per a l'arrodoniment al més pròxim



b_{29}	g	arr	R
x	0	x	0
0	1	0	0
1	1	0	1
x	1	1	1

Si R=0, truncar:

implícit

```
  1.1001101
+          0
-----------
  1.10011
```

Si R=1, incrementar:

```
  1.1001111
+          1
-----------
  1.10100
```

Quan c_A=1, cal corregir

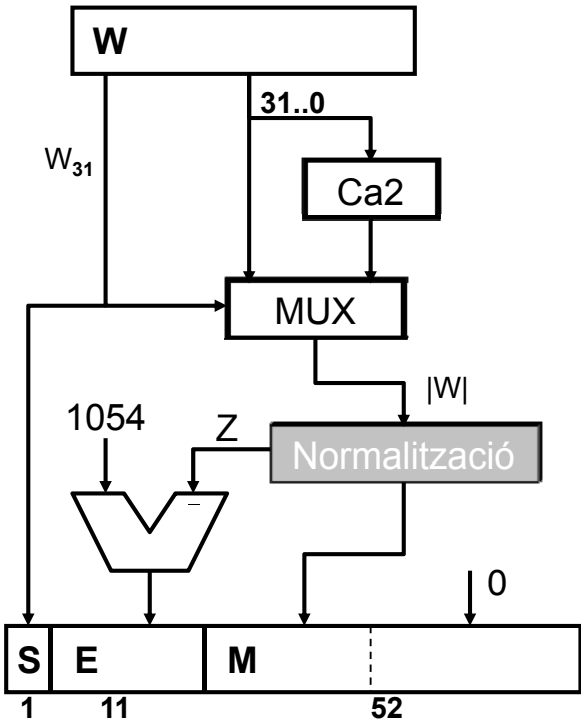
```
  1.1111111
+          1
-----------
 10.0000
  1.0000
```

Conversió d'enter a doble precisió (`cvt.d.w`)

- Filosofia de l'operador
  - Si és positiu, un enter W de 32 bits es pot escriure com $+0.Wx2^{32}$
  - Si és negatiu, W es rescriu com $-0.(-W)\ x2^{32}$
  - La mantissa W comença per una sèrie de Z zeros ($0 \le Z \le 32$)
  - Caldrà normalitzar la mantissa desplatzant-la Z+1 posicions cap a l'esquerra (eliminant el bit enter) i restar Z+1 a l'exponent
- Especificació
  - Entrada de l'operador: W (32 bits)
  - Eixida de l'operador: S_R (1 bit), E_R (11 bits) i M_R (52 bits)
  - S_R = Signe(W)
  - MR = |W| << Z+1 (desplaçament)
    - Si W<0, cal fer W = –W
  - E_R = 1023 + 32 – Z – 1

Conversió d'enter a doble precisió (`cvt.d.w`)

- Esquema
  - La normalització ha de comptar el nombre Z de bits a zero per l'esquerra (fins el primer 1)
  - Ha de deplaçar la mantissa Z posicions cap a l'esquerra
    - L'exponent que cal representar serà 31–Z
    - Afegint l'excés, tenim E = 1023 + 31 –Z
  - Cal completar la mantissa amb zeros

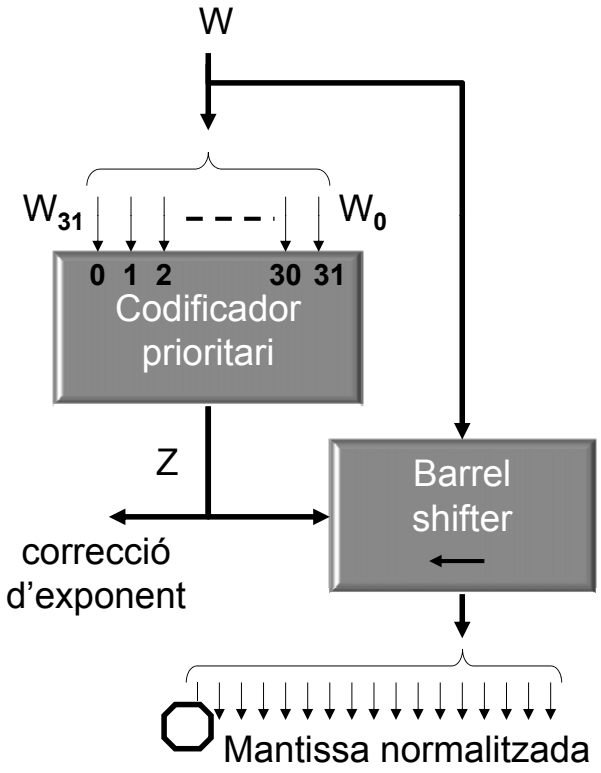


La normalització

- Circuit de normalització
  - Un codificador prioritari (que codifica l'entrada de menor índex amb un 1) calcula Z
  - Un barrel shifter desplaça la mantissa cap a l'esquerra i elimina els ceros sobrants
  - Cal descartar el bit implícit

Codificador prioritari

$W_{31}W_{30}W_{29}W_{28}$ …	W_1 W_0	Z
1 X X X …	X X	00000
0 1 X X …	X X	00001
0 0 1 X …	X X	00010
… … … … …	… …	…
0 0 0 0 …	0 1	11111

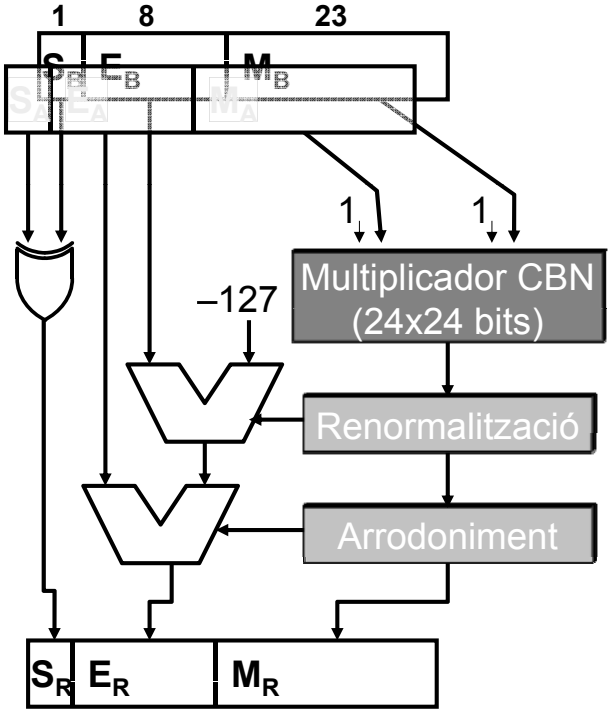


La multiplicació (`mul.s` y `mul.d`)

- Especificació
  - Dos versions: simple precisió i doble precisió
  - Dues entrades (A i B):
    - S_A (1 bit), E_A (8/11 bits) i M_A (23/52 bits)
    - S_B (1 bit), E_B (8/11 bits) i M_B (23/52 bits)
  - Eixida: S_R (1 bit), E_R (8/11 bits) i M_R (23/52 bits)
  - Càlcul del signe: $S_R = S_A$ xor S_B
  - Càlcul de l'exponent: cal sumar compensant l'excés
    - Simple precisió $E_R = E_A + E_B - 127$
    - Doble precisió $E_R = E_A + E_B - 1023$
  - Càlcul de la mantissa:
    - Cal multiplicar $1.M_A$ x $1.M_B$ (tot considerant el bit implícit)
    - El multiplicador depén de la precisió: 24x24 bits o 53x53 bits
    - Caldrà renormalitzar (llevant el bit implícit) i arrodonir la mantissa resultant deixant-la en 23/52 bits

La multiplicació (`mul.s`)

- Operador (SP)
  - Dos sumadors per a sumar els exponents representats en excés 127
  - Multiplicador per a les mantisses
    - Opera amb el bit implícit afegit: 24x24 bits
  - Caldrà renormalitzar el resultat (ara ho veurem) i llevar el bit implícit
    - potser caldrà incrementar l'exponent
  - El resultat ocuparà 47 bits i caldrà arrodonir-lo a 23
    - potser caldrà incrementar l'exponent



La renormalització

- Renormalització després d'un producte
  - Si la mantissa del producte comença per 0:
    - Cal desplaçar a l'esquerra en una posició
  - Si comença per 1:
    - Cal corregir l'exponent incrementant en 1
  - En qualsevol cas:
    - Cal eliminar el bit enter implícit

```
      1000
x     1000
 01000000
 ⇩
  0000000

      1011
x     1101
 01110101
 ⇩
  1101010

      1111
x     1111
 11100001
 ⇩
  1100001
```