UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d’Arquitectura de Computadors

Tema 5
Aritmètica d’enters i coma flotant

Estructura de Computadors (EC)
2023 - 2024 Q2
Adrià Armejach (adria.armejach@upc.edu)

Problemes
Aritmètica d’enters

Problema 1

(0,75 p) Suposem que els registres MIPS $t1 i $t2 valen $t1=0x00000010 i $t2=0xFFFFFFFF. Indica el contingut final en hexadecimal de $hi i $lo després d'executar `mult $t1,$t2` i després de `multu $t1,$t2`.

`mult`	`$t1, $t2`	`# $hi=`	`0x`	`$lo=` `0x`
`multu`	`$t1, $t2`	`# $hi=`	`0x`	`$lo=` `0x`

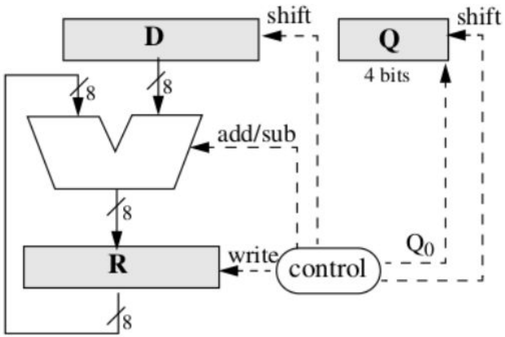
Problema 2

b) Suposem que els registres MIPS `$t1` i `$t2` valen `$t1=0x12345678` i `$t2=0xFFFFFFFF`. Indica el contingut final en hexadecimal de `$hi` i `$lo` després d'executar `divu $t1,$t2` i després de `div $t1,$t2`.

divu:	\$hi=	0x	\$lo=	0x
div:	\$hi=	0x	\$lo=	0x

Problema 3

Sigui el següent circuit seqüencial per a la divisió de números naturals de 4 bits, anàleg al que s'ha estudiat durant el curs, el qual calcula el quocient i el residu amb 4 bits:



Suposem que volem calcular la divisió (en base 2): 1111/0010. Omple la següent taula indicant quin és el valor inicial dels registres R, D i Q, així com el seu valor després de cada iteració de l'algorisme que controla el circuit (omple tantes files com iteracions tingui l'algorisme).

iteració	R (Dividend/Residu)								D (Divisor)								Q (Quocient)			
valor inicial									0	0	1	0	0	0	0	0				
1																				
2																				

Representació de nombres en coma flotant

Com representem números reals o fraccionaris?

- Idea simple: “punt fixe”
    - Alguns bits per la part entera i alguns per la part fraccionària
    - Exemple amb 8 bits:
        - `eeeeefff -> part entera i part fraccionària`

$$10101.110 =$$
$$= 1\times2^{4} + 0\times2^{3} + 1\times2^{2} + 0\times2^{1} + 1\times2^{0} + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} =$$
$$= 16 + 4 + 1 + 0{,}5 + 0{,}25 = 21.75$$

- El rang és molt limitat
- Distància entre números representables és equidistant

Representació de nombres en coma flotant

- Serveix per representar nombres reals de forma aproximada
  - Nombre limitat de bits
- Notació científica (base 10)
  - $234000000 = 234 \times 10^6 = 2{,}34 \times 10^8 = 0{,}0234 \times 10^{10}$
- Notació científica **normalitzada** (base 10)
  - $2{,}34 \times 10^8$
  - $v = m \times 10^n$
  - `m:` mantissa amb part entera `e` i part fraccionària `f`
    - $m = e{,}f$ on e és: $0 < e \leq 9$
  - `n:` exponent
    - Indica la posició de la coma (factor d'escala)

Representació de nombres en coma flotant

- Representació binaria
  - xxx…x: dígits binaris de la mantissa
  - yyy…y: dígits binaris de l'exponent

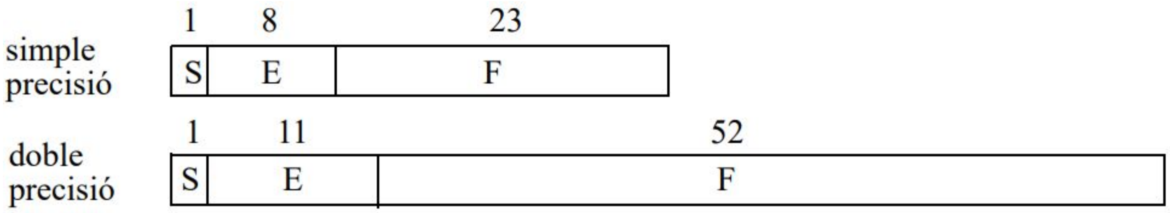
$$v = \underbrace{\pm}_{\text{signe(S)}} \underbrace{1,\overbrace{xxx\ldots x}^{\text{fracció(F)}}}_{\text{mantissa}} \times \underbrace{2}_{\text{base}}{}^{\underbrace{yyy\ldots y}_{\text{exponent(E)}}}$$

- S'emmagatzemen el signe (S), la fracció (F) i l'exponent (E)
- No s'emmagatzemen
  - El dígit enter de la mantissa, sempre és 1 (bit ocult)
  - La base, sempre és 2

$$v = (-1)^s \times (1 + 0,f) \times 2^e$$

L'estàndard IEEE-754

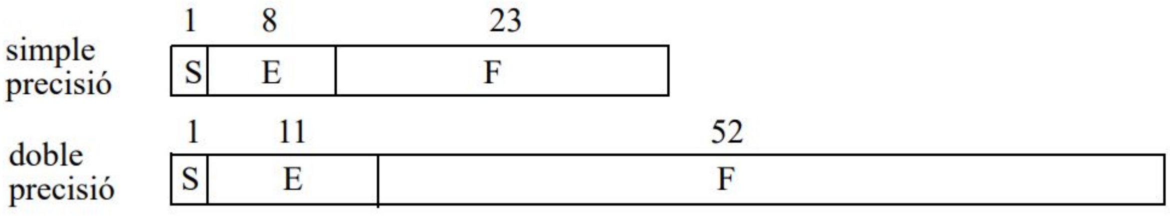
- Defineix el format dels nombres en coma flotant (1985)



- Signe (S)
  - 0: positiu 1: negatiu
- Fracció (F)
  - Part fraccionària de la mantissa. El bit ocult (part entera) no es representa.
- Exponent (E)
  - Codificat en excés 127 (simple precisió) o 1023 (doble precisió)

L’estàndard IEEE-754

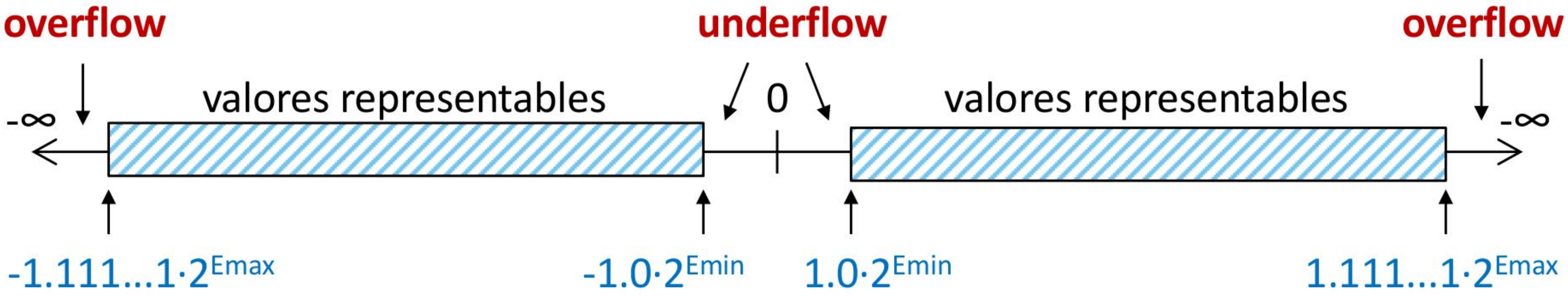
- Defineix el format dels nombres en coma flotant (1985)



- Facilita les operacions de comparació
    - Si tenen signe diferent, el positiu és el major
    - Si tenen el mateix signe, es pot usar un comparador de naturals
        - La codificació de l’exponent en excés a 127 ho facilita
        - Si son negatius s’ha d’invertir el resultat de la comparació

Rang

- Rang de representació
  - Interval format per el major número positiu representable i el seu oposat
    - Major número: màxima mantissa (tots els bits a 1) i el màxim exponent (E_{max})
  - El rang depèn principalment del nombre de bits d'exponent
  - Si el resultat d'una operació dona $E > E_{max}$ tenim **overflow**



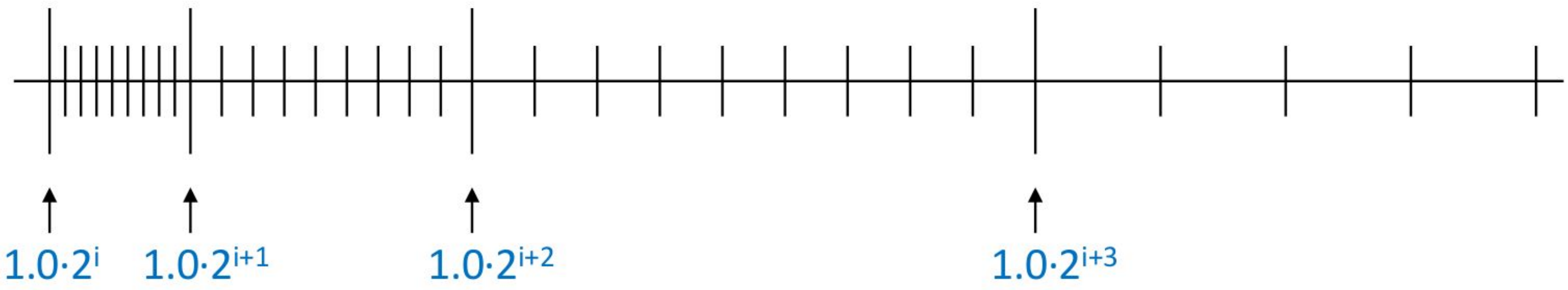
Precisió

- Precisió
  - Expressa el grau de detall amb el qual es pot representar una quantitat
  - Nombre de bits significatius
  - Igual al nombre de bits de la mantissa
    - 24 en precisió simple
    - 53 en precisió doble

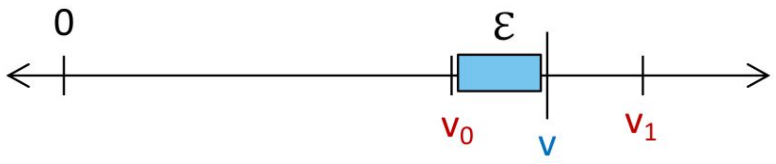
- Compromís entre precisió i rang
  - Augmentar precisió (bits de mantissa) va en detriment del rang (bits d'exponent)

Valors representables

- Els valors representables **no** son equidistants
- Amb 32 bits es poden representar 2^{32} números diferents, igual que amb els enters, però la distància entre dos valors consecutius no és sempre la mateixa

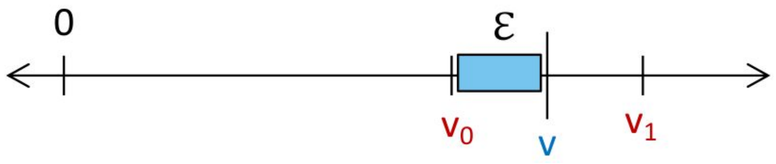


Error de precisió



- L'error de precisió és el que es comet pel fet de tenir un nombre limitat de bits a la mantissa
  - Sigui v un valor no representable
  - Sigui $v_0 < v < v_1$
  - Si aproximem v usant v_0 l'error és: $\varepsilon = |v - v_0|$
  - Màxim **error absolut**: $\varepsilon_{max} < |v_1 - v_0|$
  - En simple precisió:
    - $v_0 = m \times 2^E$
    - $v_1 = (m + 2^{-23}) \times 2^E$
    - $\varepsilon_{max} = 2^{E-23}$

Error de precisió



- Error relatiu: $\eta = \varepsilon \ / \ |v|$
- Màxim **error relatiu** en simple precisió:
  - $\eta_{max} < \varepsilon_{max} \ / \ |v_0|$
  - $\eta_{max} < 2^{E-23} \ / \ (m \times 2^{E}) = 2^{-23} \ / \ m$
  - Com que $m \geq 1{,}0$: $\eta_{max} < 2^{-23} = 1$ ULP (Unit in the Last Place)

Underflow

- Es produeix quant $|v|$ és menor que el mínim positiu representable
  - $0 < |v| < 2^{Emin}$
  - Si aproximem v per 0:
    - Error absolut: $\mathcal{E} = |v - 0| = v$
    - Error relatiu: $\eta = \mathcal{E} / |v| = 1$
  - L'error relatiu en el interval $(-2^{Emin}, 2^{Emin})$ és 6 ordres de magnitud major que a la resta del rang
    - Codificació especial

Arrodoniment

- El resultat exacte `v` d'una operació pot no ser representable
  - La mantissa te un nombre limitat de bits (precisió)
- S'ha d'aproximar `v` per un dels dos valors representables més pròxims
  - v_0 o v_1, essent $v_0 < v < v_1$
- Aquesta aproximació rep el nom d'arrodoniment
- Quatre modes d'arrodoniment a l'estandard per escollir entre v_0 i v_1
  - Al més pròxim a `v` (per defecte)
  - Al més pròxim a zero
  - Al més pròxim a +Infinit
  - Al més pròxim a -Infinit

Arrodoniment al més pròxim

- Proporciona el menor error de precisió
  - $\varepsilon_{max} = |v_1 - v_0| / 2$
  - $\eta_{max} < 0{,}5 \times 2^{-23}$
  - $\eta_{max} < 2^{-24} = 0{,}5$ ULP

- Suposem els següents resultats amb 6 bits que volem arrodonir a 3 bits:
  - 101,**010** -> 101 (en decimal: 5,250 -> 5)
  - 101,**011** -> 101 (en decimal: 5,375 -> 5)
  - 101,**101** -> 110 (en decimal: 5,625 -> 6)
  - 101,**100** -> ? (en decimal: 5,500 -> ?)
    - En cas d'equidistància l'estàndard arrodoneix cap al parell. En aquest cas -> 6

Resum: Arrodoniment

- Arrodoniment al més pròxim a `v`
  - Dona el menor error
  - S'utilitza per defecte
  - Arrodoneix al número parell en cas d'equidistància
- Arrodoniment al més pròxim a zero
  - Truncar els bits que sobren
  - Fàcil d'implementar
- Arrodoniment al més pròxim a `+Infinit`
- Arrodoniment al més pròxim a `-Infinit`

Codificacions especials

- L'estàndard reserva dos codificacions de l'exponent per valors especials
  - E = 000…0 (tot zeros)
  - E = 111…1 (tot uns)

- Rang dels exponents: E_{min} = 00…001 i E_{max} = 111…110
  - Precisió simple: [-126, +127]
  - Precisió doble: [-1022, +1023]

Codificacions especials - Denormals

- Error relatiu enorme en l'interval $(-2^{Emin}, 2^{Emin})$
- Per reduir l'error es permeten els valors no-normalitzats (denormals)
  - "omplen" el gran forat entre 0 i 2^{Emin}
- Els denormals tenen la forma: $v = \pm\ \mathbf{0},xxx...x \times 2^{Emin}$
  - El bit ocult és zero
- Millor aproximació per nombres molt propers a zero
- Representació de denormals
  - Exponent amb tots els bits a zero, i mantissa amb almenys un bit no nul
    - E = 000…0 F=xxx…x (algun $x \neq 0$)

Codificacions especials - Zero

- Representació del zero
  - Tots els bits de l'exponent i la fracció a zero
    - E = 000…0 F=000…0
  - Degut al signe, el zero té dos representacions

Codificacions especials - Infinit

- Resulta convenient operar amb l'infinit

$$y = \frac{1}{1 + \frac{100}{x}}$$

- 1 / 0 = Inf
- 1 / Inf = 0
- x + Inf = Inf
- Representació de l'infinit
    - Tots els bit de l'exponent a 1 i els de la fracció a zero
        - E = 111…1 F=000…0
    - Amb el bit de signe tenim: `+Infinit i -Infinit`

Codificacions especials - Not a Number (NaN)

- Resultats invalids (NO confondre amb overflow/underflow)
  - Arrel quadrada de un nombre negatiu
  - Logaritme de un nombre negatiu
  - 0 / 0
  - $0 \times \text{Inf}$
  - Inf / Inf
- Representació de NaN
  - Tots els bits del exponent a 1 i mantissa different de zero
    - E = 111…1 F=xxx…x (algun x ≠ 0)
- Propagació de Nan
  - Qualsevol operació on un dels operands es NaN dona com a resultat NaN

Codificacions especials - Taula Resum

<table>
<tr><th></th><th></th><th colspan="3">Exponent (E)</th></tr>
<tr><th></th><th></th><th>Tot 0</th><th>Altres</th><th>Tot 1</th></tr>
<tr><th rowspan="2">Mantissa (F)</th><th>Tot 0</th><td>Zero</td><td rowspan="2">Normalitzat</td><td>Infinit</td></tr>
<tr><th>Altres</th><td>Denormal</td><td>NaN</td></tr>
</table>

Conversió de base 10 a coma flotant

- `v = -1029,68`

1. Convertir a binari la part entera
   - Per divisions successives obtenim: $1029 = 10000000101_2$ (11 bits)
2. Convertir a binari la part decimal
   - Algorisme de multiplicacions successives per 2
   - `0,68 x 2 = `**`1`**`,36 ->` el primer bit fraccionari és un 1
     - **1**,36 **0**,72 **1**,44 **0**,88 **1**,76 **1**,52 **1**,04 **0**,08 **0**,16 **0**,32 **0**,64 **1**,28 **0**,56 **1**,12 **0**,24 **0**,48 **0**,96 **1**,92 . . .
   - `v = -10000000101,10101110000101000 1…`

Conversió de base 10 a coma flotant

- `v = -1029,68`

3. Normalitzar la mantissa
   - `v = -1,0000000101101011100001010001… ×` 2^{10}
4. Arrodonir la mantissa
   - En simple precisió la mantissa només pot tenir 24 bits
     - El resultat actual en té 29, cal arrodonir-lo
     - Per defecte, l’estàndard estipula arrodoniment al més pròxim
       - v = -1,00000001011010111000010**10001…** × 2^{10}

```
  1,00000001011010111000010
+                         1
---------------------------
  1,00000001011010111000011
```

sumem 1 ULP al valor truncat

Conversió de base 10 a coma flotant

- `v = -1029,68`

5. Codificar el exponent
    - Representem `E = 10`, en excés a 127, i el codifiquem en binari
    - $E = 10 + 127 = 137 = 10001001_2$
6. Ajuntem Signe, Exponent i Fracció (descartant el bit ocult!)
    - $v = 1\ 10001001\ 00000001011010111000011_2$
    - Expressat en hexadecimal queda:
        - $v = 1100\ 0100\ 1000\ 0000\ 1011\ 0101\ 1100\ 0011_2 = \text{0xC480B5C3}$

Conversió de coma flotant a base 10

- v = 0x45814140

1. Decodificar signe, exponent i mantissa

$$V = \underbrace{0}_{S}\ \underbrace{10001011}_{E}\ \underbrace{00000010100000101000000}_{F}$$

   - Signe positiu
   - Exponent: $E_u = 10001011_2 = 139$
     - L'exponent enter és: $E = 139-127 = 12$
   - Mantissa: 1,00000010100000101000000

$$V = +1{,}00000010100000101000000_2 \times 2^{12}$$

Conversió de coma flotant a base 10

- `v = 0x45814140`

2. Eliminar la potència
    - Efectuem el producte movent la coma 12 posicions a la dreta
        - $v = 1000000101000,00101_2$
3. Convertir la part entera a decimal
    - $1000000101000_2 = 1 \cdot 2^{12} + 1 \cdot 2^5 + 1 \cdot 2^3 = 4096 + 32 + 8 = 4136$
4. Convertir la part fraccionària a decimal
    - $0,00101_2 = 101_2 \times 2^{-5} = 5/32 = 0,15625$

Ajuntant les dues parts obtenim el resultat: $v = 4136,15625$

Link - eina d'interès

https://www.h-schmidt.net/FloatConverter/IEEE754.html