Estructura de Computadores

Tema 5. Aritmética de enteros y coma flotante

Conversión de base 10 a coma flotante

- v = 4,3

Conversión de base 10 a coma flotante

- $v = 4{,}3$

1) Convertir a binario la parte entera

$4 \rightarrow 100$

$v = +100{,}xxxxxxxx$

Conversión de base 10 a coma flotante

- v = 4,3

1) Convertir a binario la parte entera

$4 \rightarrow 100$

v = +100,xxxxxxxx

2) Convertir a binario la parte fraccionaria

$0,3 \times 2 = 0,6$

0,6;**1**,2;**0**,4;**0**,8;**1**,6;**1**,2;**0**,4;**0**,8; **1**,6;**1**,2;**0**,4;**0**,8;**1**,6;
1,2;**0**,4;**0**,8;**1**,6;**1**,2;**0**,4;**0**,8;**1**,6; **1**,2;**0**,4;**0**,8;**1**,6;**1**,2;

Conversión de base 10 a coma flotante

- v = 4,3

1) Convertir a binario la parte entera

$4 \rightarrow 100$

v = +100,xxxxxxxx

2) Convertir a binario la parte fraccionaria

$0,3 \times 2 = 0,6$

0,6;**1**,2;**0**,4;**0**,8;**1**,6;**1**,2;**0**,4;**0**,8; **1**,6;**1**,2;**0**,4;**0**,8;**1**,6;
1,2;**0**,4;**0**,8;**1**,6;**1**,2;**0**,4;**0**,8;**1**,6; **1**,2;**0**,4;**0**,8;**1**,6;**1**,2;

v = +100,01001100110011001100110011

Conversión de base 10 a coma flotante

- v = 4,3

3) Normalizar la mantisa

v = +100,01001100110011001100110011

Conversión de base 10 a coma flotante

- $v = 4{,}3$

3) Normalizar la mantisa

$v = +100{,}01001100110011001100110011$

$v = +1{,}00010011001100110011001100110011 \times 2^2$

Conversión de base 10 a coma flotante

- v = 4,3

3) Normalizar la mantisa

$v = +100{,}01001100110011001100110011$

$v = +1{,}00010011001100110011001100110011 \times 2^2$

4) Redondear la mantisa

$v = +1{,}00010011001100110011001\color{red}{10011} \times 2^2$

Conversión de base 10 a coma flotante

- v = 4,3

3) Normalizar la mantisa

$v = +100{,}01001100110011001100110011$

$v = +1{,}00010011001100110011001100110011 \times 2^2$

4) Redondear la mantisa

$v = +1{,}00010011001100110011001\color{red}{10011} \times 2^2$

$$
\begin{array}{rl}
 & 1{,}00010011001100110011001 \\
+ & 0{,}00000000000000000000001 \\
\hline
 & 1{,}00010011001100110011010
\end{array}
$$

Conversión de base 10 a coma flotante

- v = 4,3

3) Normalizar la mantisa

$v = +100{,}01001100110011001100110011$

$v = +1{,}00010011001100110011001100110011 \times 2^2$

4) Redondear la mantisa

$v = +1{,}00010011001100110011001\color{red}{10011} \times 2^2$

$$\begin{array}{rl} & 1{,}00010011001100110011001 \\ + & 0{,}00000000000000000000001 \\ \hline & 1{,}00010011001100110011010 \end{array}$$

$v = +1{,}00010011001100110011010 \times 2^2$

Conversión de base 10 a coma flotante

- $v = 4{,}3$

$$v = +1{,}00010011001100110011010 \times 2^2$$

Conversión de base 10 a coma flotante

- $v = 4{,}3$

$v = +1{,}00010011001100110011010 \times 2^2$

5) Codificar el exponente

$2 + 127 = 129 = 2^7 + 2^0$

$E = 10000001$

Conversión de base 10 a coma flotante

- v = 4,3

$v = +1{,}00010011001100110011010 \times 2^2$

5) Codificar el exponente

$2 + 127 = 129 = 2^7 + 2^0$

$E = 10000001$

6) Juntar signo, exponente y fracción

S	E	F
0	10000001	00010011001100110011010

Conversión de base 10 a coma flotante

- v = 4,3

$v = +1{,}00010011001100110011010 \times 2^2$

5) Codificar el exponente

$2 + 127 = 129 = 2^7 + 2^0$

$E = 10000001$

6) Juntar signo, exponente y fracción

S	E	F
0	10000001	00010011001100110011010

$v = 0x4089999A$

Conversión de coma flotante a base 10

- v = 0xC1480000

Conversión de coma flotante a base 10

- v = 0xC1480000

1) Decodificar signo, exponente y mantisa

v = 1100 0001 0100 1000 0000 0000 0000 0000

Conversión de coma flotante a base 10

- v = 0xC1480000

1) Decodificar signo, exponente y mantisa

v = 1100 0001 0100 1000 0000 0000 0000 0000

S	E	F
1	10000010	10010000000000000000000

Conversión de coma flotante a base 10

- v = 0xC1480000

1) Decodificar signo, exponente y mantisa

v = 1100 0001 0100 1000 0000 0000 0000 0000

S	E	F
1	10000010	10010000000000000000000

$E_u = 2^7 + 2^1 = 130,\ E = 130 - 127 = 3$

Conversión de coma flotante a base 10

- v = 0xC1480000

1) Decodificar signo, exponente y mantisa

v = 1100 0001 0100 1000 0000 0000 0000 0000

S	E	F
1	10000010	10010000000000000000000

$E_u = 2^7 + 2^1 = 130,\ E = 130 - 127 = 3$

v = -1,1001$\times 2^3$

Conversión de coma flotante a base 10

- v = 0xC1480000

v = -1,1001$\times 2^3$

Conversión de coma flotante a base 10

- v = 0xC1480000

v = -1,1001$\times 2^3$

2) Eliminar la potencia

v = -1100,1

Conversión de coma flotante a base 10

- v = 0xC1480000

v = -1,1001$\times 2^3$

2) Eliminar la potencia

v = -1100,1

3) Convertir a base 10 la parte entera

1100 $\rightarrow$ 12

Conversión de coma flotante a base 10

- v = 0xC1480000

v = -1,1001$\times 2^3$

2) Eliminar la potencia

v = -1100,1

3) Convertir a base 10 la parte entera

1100 $\rightarrow$ 12

4) Convertir a base 10 la parte fraccionaria

0,1 $\rightarrow$ 0,5

Conversión de coma flotante a base 10

- v = 0xC1480000

v = -1,1001$\times 2^3$

2) Eliminar la potencia

v = -1100,1

3) Convertir a base 10 la parte entera

1100 $\rightarrow$ 12

4) Convertir a base 10 la parte fraccionaria

0,1 $\rightarrow$ 0,5

v = -12,5

Suma en coma flotante - Base 10

- $z = x + y$, con 4 dígitos de precisión en la mantisa
  - $x = 9,999 \times 10^{1}$
  - $y = 1,680 \times 10^{-1}$

Suma en coma flotante - Base 10

- $z = x + y$, con 4 dígitos de precisión en la mantisa
  - $x = 9,999 \times 10^1$
  - $y = 1,680 \times 10^{-1}$

1) Igualar los exponentes al mayor de los dos

$y = 1,680 \times 10^{-1} = 0,01680 \times 10^1$

Suma en coma flotante - Base 10

- $z = x + y$, con 4 dígitos de precisión en la mantisa
  - $x = 9,999 \times 10^1$
  - $y = 1,680 \times 10^{-1}$

1) Igualar los exponentes al mayor de los dos

$y = 1,680 \times 10^{-1} = 0,01680 \times 10^1$

2) Sumar las mantisas

$$\begin{array}{r} 9,999 \\ +\ 0,01680 \\ \hline 10,01580 \end{array}$$

$z = 10,01580 \times 10^1$

Suma en coma flotante - Base 10

- $z = x + y$, con 4 dígitos de precisión en la mantisa
  - $x = 9,999 \times 10^1$
  - $y = 1,680 \times 10^{-1}$

3) Normalizar el resultado (comprobar overflow/underflow)

$z = 10{,}01580 \times 10^1 = 1{,}001580 \times 10^2$

Exponente dentro del rango, no hay excepción

Suma en coma flotante - Base 10

- $z = x + y$, con 4 dígitos de precisión en la mantisa
  - $x = 9,999 \times 10^1$
  - $y = 1,680 \times 10^{-1}$

3) Normalizar el resultado (comprobar overflow/underflow)

$z = 10,01580 \times 10^1 = 1,001580 \times 10^2$

Exponente dentro del rango, no hay excepción

4) Redondear la mantisa

$z = 1,001580 \times 10^2 = 1,002 \times 10^2$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

1) Decodificar signo, exponente y mantisa

x = 0 01111110 10000000000000000001101

E = 126 - 127 = -1

x = $+1{,}10000000000000000001101 \times 2^{-1}$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

1) Decodificar signo, exponente y mantisa

x = 0 01111110 10000000000000000001101
E = 126 - 127 = -1
x = $+1{,}10000000000000000001101 \times 2^{-1}$

y = 1 10000001 00000000000000000000100
E = 129 - 127 = 2
y = $-1{,}00000000000000000000100 \times 2^{2}$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

2) Igualar exponentes (al mayor)

$\texttt{x} = +1,10000000000000000001101 \times 2^{-1}$

$\texttt{y} = -1,00000000000000000000100 \times 2^{2}$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

2) Igualar exponentes (al mayor)

$\texttt{x} = \texttt{+1,10000000000000000001101} \times 2^{-1}$

$\texttt{y} = \texttt{-1,00000000000000000000100} \times 2^{2}$

$\texttt{x} = \texttt{+0,00110000000000000000001101} \times 2^{2}$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

3) Sumar o restar las mantisas

Signo x	Signo y	Operación	Signo del resultado
+	+	Suma	+
-	-	Suma	-
+ -	- +	Al de mayor magnitud se le resta el de menor magnitud	Signo del de mayor magnitud

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

3) Sumar o restar las mantisas

x = +0,00110000000000000000001101$\times 2^2$
y = -1,00000000000000000000100$\times 2^2$
Signo distinto: hay que restar mantisas
y tiene mayor magnitud: y - x, signo del resultado negativo

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

3) Sumar o restar las mantisas

x = +0,00110000000000000000001101$\times 2^2$
y = −1,00000000000000000000100$\times 2^2$
Signo distinto: hay que restar mantisas
y tiene mayor magnitud: y - x, signo del resultado negativo

$$
\begin{array}{rl}
 & |y| = 1,00000000000000000000100 \\
- & |x| = 0,00110000000000000000001101 \\
\hline
= & |z| = 0,11010000000000000000010011
\end{array}
$$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

4) Normalizar la mantisa

$|z| = 0{,}11010000000000000000010011 \times 2^2$

$|z| = 1{,}01000000000000000000010011 \times 2^1$

Si $E < E_{min}$: underflow

Si $E > E_{max}$: overflow

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

5) Redondear la mantisa

$|z| = 1{,}10100000000000000000100\textcolor{red}{11} \times 2^1$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

5) Redondear la mantisa

$|z| = 1{,}10100000000000000000100\,11 \times 2^{1}$

$$
\begin{array}{rl}
 & 1{,}10100000000000000000100 \\
+ & 0{,}00000000000000000000001 \\
\hline
 & 1{,}10100000000000000000101
\end{array}
$$

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

5) Redondear la mantisa

$|z| = 1{,}10100000000000000000100\,11 \times 2^1$

$$
\begin{array}{rl}
 & 1\,,10100000000000000000100 \\
+ & 0\,,00000000000000000000001 \\
\hline
 & 1\,,10100000000000000000101
\end{array}
$$

$|z| = 1{,}10100000000000000000101 \times 2^1$

A veces es necesario volver a normalizar y redondear la mantisa

Suma en coma flotante - IEEE-754 simple precisión

- Sumar $z = x + y$, siendo $x = 0x3F40000D$, $y = 0xC0800004$

6) Codificar signo, exponente y fracción

z = -1,10100000000000000000101$\times 2^1$
E = 1 + 127 = 128 = 10000000

z = 1 10000000 10100000000000000000101
z = 0xC0500005

Bits de guarda - IEEE-754 simple precisión

- Al igualar los exponentes al mayor, ¿cuántas posiciones desplazamos en el peor caso?
  - $E_{min} = -126$, $E_{max} = 127$
  - Más de 200 posiciones a la derecha!
  - Necesitaríamos un sumador de más de 200 bits!
- Se puede conseguir **el mismo resultado** con solo 3 bits de guarda

Bits de guarda - IEEE-754 simple precisión

- Bits de guarda
  - Guard (G): bit 24 de la mantisa
  - Round (R): bit 25 de la mantisa
  - Sticky (S): OR lógica de todos los bits a la derecha del bit de Round

Bits de guarda - IEEE-754 simple precisión

- Bits de guarda
  - Guard (G): bit 24 de la mantisa
  - Round (R): bit 25 de la mantisa
  - Sticky (S): OR lógica de todos los bits a la derecha del bit de Round

```
1,10100000110101010010110
0,00000001101000001101010 10010110
                          GRS
0,00000001101000001101010 101
```

Bits de guarda - IEEE-754 simple precisión

- Bits de guarda
  - Guard (G): bit 24 de la mantisa
  - Round (R): bit 25 de la mantisa
  - Sticky (S): OR lógica de todos los bits a la derecha del bit de Round

```
1,10100000110101010010110
0,00000001101000001101010 10010110
                          GRS
0,00000001101000001101010 101
```

- El resultado al operar con los 3 bits de guarda (GRS) es el mismo que si se utilizaran infinitos bits

Multiplicación/División en coma flotante

Multiplicación	**División**
1) Sumar los exponentes	1) Restar los exponentes
2) Multiplicar las mantisas	2) Dividir las mantisas
3) Normalizar la mantisa del resultado	
4) Redondear la mantisa del resultado	
5) Asignar signo negativo al resultado si los operandos tienen distinto signo	

Multiplicación en coma flotante - simple precisión

- $z = x \times y$, siendo $x = 0x3F600000$, $y = 0xBED00002$

x = 0 01111110 11000000000000000000000
E = 126 - 127 = -1
x = $+1{,}11000000000000000000000 \times 2^{-1}$

y = 1 01111101 10100000000000000000010
E = 125 - 127 = -2
y = $-1{,}10100000000000000000010 \times 2^{-2}$

Multiplicación en coma flotante - simple precisión

- $z = x \times y$, siendo $x = 0x3F600000$, $y = 0xBED00002$

1) Sumar los exponentes

E = -1 + (-2) = -3

Multiplicación en coma flotante - simple precisión

- $z = x \times y$, siendo $x = 0x3F600000$, $y = 0xBED00002$

2) Multiplicar las mantisas

```
        1110
x       11010000000000000000010
-------------------------------------------
                                    1110
        1110
     1110
   1110
-------------------------------------------
  10110110000000000000000011110
```

Con la coma: $10{,}11011000000000000000011110 \times 2^{-3}$

Multiplicación en coma flotante - simple precisión

- $z = x \times y$, siendo $x = 0x3F600000$, $y = 0xBED00002$

3) Normalizar la mantisa

$|z| = 10,11011000000000000000011110 \times 2^{-3}$

$|z| = 1,01101100000000000000011110 \times 2^{-2}$

Exponente dentro del rango, no hay overflow/underflow

Multiplicación en coma flotante - simple precisión

- $z = x \times y$, siendo $x = 0x3F600000$, $y = 0xBED00002$

4) Redondear la mantisa

$$
\begin{array}{l}
\quad 1,01101100000000000000011110 \times 2^{-2} \\
+\ 0,00000000000000000000001 \\
\hline
\quad 1,01101100000000000000010 \times 2^{-2}
\end{array}
$$

Multiplicación en coma flotante - simple precisión

- $z = x \times y$, siendo $x = 0x3F600000$, $y = 0xBED00002$

5) Codificar signo, exponente y mantisa

z = -1,01101100000000000000010$\times 2^{-2}$

E = -2 + 127 = 125 = 01111101

z = 1 01111101 01101100000000000000010

z = 0xBEB60002

Coma flotante - No asociatividad

- El formato de coma flotante IEEE-754 NO es asociativo

$$\boxed{x + (y + z) \neq (x + y) + z}$$

- Ejemplo: $x = -1,1 \times 2^{127}$, $y = 1,1 \times 2^{127}$, $z = 1,0$
  - $x + (y + z)$
    - $-1,1 \times 2^{127} + (1,1 \times 2^{127} + 1,0) = -1,1 \times 2^{127} + 1,1 \times 2^{127} = 0,0$
  - $(x + y) + z$
    - $(-1,1 \times 2^{127} + 1,1 \times 2^{127}) + 1,0 = 0,0 + 1,0 = 1,0$
- Se introduce un error al aproximar el resultado (redondeo)

Coma flotante en MIPS

- Coprocesador CP1
  - La unidad de coma flotante es un coprocesador con su propio banco de registros
  - Juego de instrucciones específico para coma flotante

Coma flotante en MIPS

- Coprocesador CP1
  - La unidad de coma flotante es un coprocesador con su propio banco de registros
  - Juego de instrucciones específico para coma flotante
- 32 registros de 32 bits: $f0-$f31
  - $f12 y $f14: Paso de parámetros a subrutina
  - $f0: Retorno del resultado de la subrutina
  - $f20-$f31: Registros seguros
  - Para doble precisión solo se utilizan registros pares ($f0, $f2...)
- Registro de Control
  - Excepciones
  - Overflow/underflow
  - Bit de Condición (CC): resultado de las comparaciones

Coma flotante en MIPS

- Copia entre registros

mfc1/mtc1/mov.s		
`mfc1 rt,fs`	rt = fs	còpia CPU ← CP1
`mtc1 rt,fs`	fs = rt	còpia CP1 ← CPU
`mov.s fd, fs`	fd = fs	còpia CP1 ← CP1

Coma flotante en MIPS

- Acceso a memoria

`lwc1/ldc1/swc1/sdc1`		
`lwc1 ft, off16(rs)`	$ft = M_w[rs + SignExt(off16)]$	load float
`ldc1 ft, off16(rs)`	$ft = M_d[rs + SignExt(off16)]$	load double
`swc1 ft, off16(rs)`	$M_w[rs + SignExt(off16)] = ft$	store float
`sdc1 ft, off16(rs)`	$M_d[rs + SignExt(off16)] = ft$	store double

Coma flotante en MIPS

- Aritméticas

add.s/add.d/sub.s/sub.d/mul.s/mul.d/div.s/div.d		
add.s fd, fs, ft	fd = fs + ft	suma floats
add.d fd, fs, ft	fd = fs + ft	suma doubles
sub.s fd, fs, ft	fd = fs - ft	resta floats
sub.d fd, fs, ft	fd = fs - ft	resta doubles
mul.s fd, fs, ft	fd = fs × ft	multiplica floats
mul.d fd, fs, ft	fd = fs × ft	multiplica doubles
div.s fd, fs, ft	fd = fs / ft	divideix floats
div.d fd, fs, ft	fd = fs / ft	divideix doubles

Coma flotante en MIPS

- Comparaciones y saltos condicionales

c.eq.s/c.eq.d/c.lt.s/c.lt.d/c.le.s/c.le.d		
`c.eq.s fs,ft`	bit de condició cc=1 si fs==ft, sino cc=0	igual que float
`c.eq.d fs,ft`	bit de condició cc=1 si fs==ft, sino cc=0	igual que double
`c.lt.s fs,ft`	bit de condició cc=1 si fs < ft, sino cc=0	menor que float
`c.lt.d fs,ft`	bit de condició cc=1 si fs < ft, sino cc=0	menor que double
`c.le.s fs,ft`	bit de condició cc=1 si fs ≤ ft, sino cc=0	menor o igual que float
`c.le.d fs,ft`	bit de condició cc=1 si fs ≤ ft, sino cc=0	menor o igual que double

bc1t/bc1f		
`bc1t etiqueta`	Salta si cc=1 (true)	
`bc1f etiqueta`	Salta si cc=0 (false)	

Coma flotante en MIPS

- Variables globales en C:

```
float var1, var2[2] = {1.0, 3.14};
double var3 = -37.55;
```

- Variables globales en MIPS:

```
        .data
var1:   .float  0.0
var2:   .float  1.0, 3.14
var3:   .double -37.55
```

- `.float` y `.double` alinean automáticamente

Conversión a coma flotante - C vs MIPS

- En C:

```
int a = 2;
float b = (float)a; /* b = 2.0 */
```

Conversión a coma flotante - C vs MIPS

- En C:

```
int a = 2;
float b = (float)a; /* b = 2.0 */
```

- En MIPS:

```
li    $t0, 2
mtc1 $t0, $f0  # f0 NO vale 2.0
# f0 = 0 00000000 00000000000000000000010
```

Conversión a coma flotante - C vs MIPS

- En C:

```
int a = 2;
float b = (float)a; /* b = 2.0 */
```

- En MIPS:

```
li   $t0, 2
mtc1 $t0, $f0  # f0 NO vale 2.0
# f0 = 0 00000000 00000000000000000000010

li   $t0, 0x40000000
mtc1 $t0, $f0  # f0 = 2.0
```

```
float func(float x)
{
   if (x < 1.0)
      return x*x;
   else
      return 2.0-x;
}
```

```
          .data
const1:   .float 1.0

          .text
func:
   la     $t0, const1
   lwc1   $f16, 0($t0)
   c.lt.s $f12, $f16
   bc1f   sino
   mul.s  $f0, $f12, $f12
   b      fisi
sino:
   add.s  $f16, $f16, $f16
   sub.s  $f0, $f16, $f12
fisi:
   jr $ra
```