Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores

Representaciones numéricas parte 2: Números Racionales

Profesor: Hans Löbel

¿Cómo podemos escribir números racionales posicionalmente?

- Podemos expandir usando exponentes negativos:

$$123{,}45 = 1\cdot10^{2} + 2\cdot10^{1}+3\cdot10^{0}+4\cdot10^{-1}+5\cdot10^{-2}$$

- En binario, ¿es lo mismo?

$$101{,}01 = 1\cdot2^{2} + 0\cdot2^{1}+1\cdot2^{0}+0\cdot2^{-1}+1\cdot2^{-2} = 5{,}25$$

¿Cómo hacemos ahora para pasar de decimal a binario?

Necesitamos obtener el valor de una división en binario

- Por ejemplo

$$10^{-1} = 0,1 = \frac{1}{10} = \frac{(1)_2}{(1010)_2}$$

- Y el resultado de esa división es: $0,0\overline{0011}$

- Decimal finito pasa a ser infinito en binario = Pésimo
- Todo depende de la base elegida (ej. 1/3 en ternario)

Veamos un ejemplo

```
d1 = 0.1
d2 = 0.2
print(d1+d2)
```

```
t = 0
while(t != 100):
    print("Valor actual: " + str(t))
    t += 0.1
```

¿Significa esto que no podemos representar de manera exacta 0,1 con una representación binaria en un computador?

SÍ

El secreto oscuro de la computación científica

- Los resultados de los ejemplos son inesperados

¿Por qué pasa esto?

- Estos números usan memoria finita para manejar rangos muy grandes y densos.
- Luego, existe un trade-off entre rango y precisión.

Caso real: Misil Patriot[1]

- 28 personas murieron en 1991, debido al mal funcionamiento de un misil Patriot.
- El Patriot es un sistema defensivo para interceptar objetivo aéreos, que utiliza misiles.
- Error fue ocasionado por aproximación de un decimal finito mediante un número binario infinito.
- Debido al error, el sistema no siguió correctamente al objetivo y el misil nunca fue disparado.

[1] http://www-users.math.umn.edu/~arnold/disasters/patriot.html

Representación de punto fijo

- Dados *n* dígitos (bits), estos se dividen de manera fija para representar signo, parte entera y parte fraccional.
- Pro: simple y rápido
- Contra: rango pequeño (¿qué pasa con la multiplicación?)
- Idea: mover (flotar) la coma (punto)

Representación de punto flotante

- Basada en notación científica normalizada
- Dos elementos centrales: significante y exponente
- Codifica la posición del punto

- Pro: gran rango
- Contra: pérdida de precisión (¿qué pasa con la suma?)

IEEE754, el formato más usado para números de punto flotante

float (32 bits):

- 1 bit de signo, 8 bit exponente, 23 bit significante
- Significante normalizado
- Exponente desfasado en 127
- 0: exponente = 0, significante = 0
- ±∞: exponente = 11111…, significante = 0
- *NaN*: exponente = 11111…, significante ≠ 0

double: 64 bits, reglas similares

$X = (-1)^{signo} \cdot 1.significante \cdot 2^{exponente-127}$

0.00101b = ?

signo = ?

significante = ?

exponente = ?

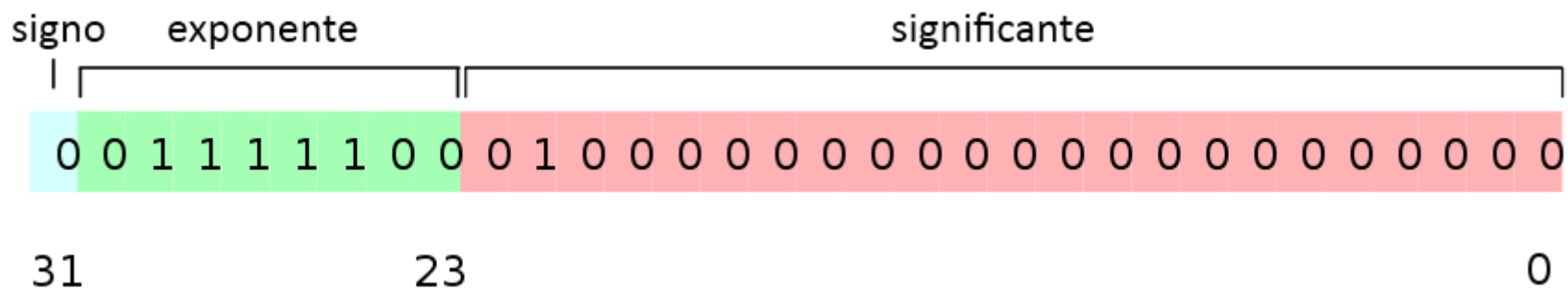
$X = (-1)^{signo} \cdot 1.significante \cdot 2^{exponente-127}$

$0.00101b = (1.01 \cdot 10^{-11})_2 = 1 \cdot 2^{-3} + 1 \cdot 2^{-5} = 0.15625$

signo = ?

significante = ?

exponente = 124 = 01111100



Veamos otro ejemplo

```
from decimal import Decimal
D1 = Decimal("0.1")
D2 = Decimal("0.2")
print(D1+D2)
```

Representaciones Alternativas

- Decimales como números enteros
  - Se utiliza como unidad el menor valor decimal requerido
  - Elimina problemas de aproximación, pero tiene poco rango
- Punto flotante con base decimal
  - Se cambia la base de 2 a 10
  - Elimina problemas de aproximación, pero resulta muy lento
  - Ideal para cálculos “humanos”
- Punto flotante con base decimal y precisión arbitraria
  - Tamaño asignado a significante y exponente se aumenta de acuerdo a las necesidades
  - Lentísimo, pero el más adecuado para cálculos “humanos”

Uso de números de punto flotante debe ser cuidadoso

- Problemas de aproximación con fracciones binarias
  - Racionales infinitos son muy comunes en cálculos humanos
- Representación de punto flotante
  - Precisión vs rango
  - Importante, nunca comparar con “==”
- Alternativas
  - Enteros: poco prácticos
  - Punto flotante con base decimal: lento pero seguro

Algunos ejemplos de pruebas pasadas

- Demuestre que los números de punto flotante del tipo *float* del estándar IEEE754 **no** cumplen el principio de asociatividad de la suma, *i.e.*, $x + (y + z) = (x + y) + z$. **(3 ptos.)**

- Explique por qué el número $2^{50} + 5$ no puede representarse de manera exacta usando el tipo de dato `float` de 32 bits. **(1 pto.)**

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores

Representaciones numéricas parte 2: Números Racionales

Profesor: Hans Löbel