DCC

DEPARTAMENTO DE CIENCIA
DE LA COMPUTACIÓN

IIC2343 - Arquitectura de Computadores

Clase 10 - Representación de números racionales

Susana Figueroa

Recapitulando…

- Sabemos cómo representar números en binario.
- Más que eso, sabemos cómo representar números enteros en binario utilizando el complemento de 2.
- Pero nos falta algo esencial… como representamos números racionales?

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera.

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1}$$

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1}$$

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1}$$

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1}$$

Qué potencia de 2 (negativa), necesitamos sumarle al 34 para llegar al 34.625?

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1}$$

Qué potencia de 2 (negativa), necesitamos sumarle al 34 para llegar al 34.625?

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1}$$

Qué potencia de 2 (negativa), necesitamos sumarle al 34 para llegar al 34.625?

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1} + 1 \times 2^{-1}$$

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1} + 1 \times 2^{-1}$$

Ahora, qué potencia de 2 (negativa), necesitamos sumarle al 34.5 para llegar al 34.625?

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1} + 1 \times 2^{-1}$$

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Ahora, qué potencia de 2 (negativa), necesitamos sumarle al 34.5 para llegar al 34.625?

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1} + 1 \times 2^{-1} + 1 \times 2^{-3}$$

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1} + 1 \times 2^{-1} + 1 \times 2^{-3}$$

Ahora escribimos todo esto en binario.

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Para representar números racionales se usa la misma lógica que para los números enteros, solo que ahora utilizaremos las potencias negativas de 2.
- Pensemos como implementariamos esto mismo, pero en decimal.

$$34.625_{10} = 3 \times 10^{1} + 4 \times 10^{0} + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

- Ahora en binario… Es la misma idea, solo que con potencias de 2.
- Primero transformamos la parte entera. **Ahora agregamos la parte racional.**

$$34.625_{10} = 1 \times 2^{5} + 1 \times 2^{1} + 1 \times 2^{-1} + 1 \times 2^{-3} = 100010.101_{2}$$

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- **No todos los números racionales finitos en base 10 tienen representación finita en base 2.**

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- **No todos los números racionales finitos en base 10 tienen representación finita en base 2.**
- Por ejemplo, el número 0.1_{10}

$$0.1_{10} = 1 \times 2^{-4} + (?)$$

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- **No todos los números racionales finitos en base 10 tienen representación finita en base 2.**
- Por ejemplo, el número 0.1_{10}

$$0.1_{10} = 1 \times 2^{-4} + 1 \times 2^{-5} + (?)$$

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Potencia	Valor
2^{-5}	0.03125
2^{-6}	0.015625
2^{-7}	0.0078125
2^{-8}	0.00390625

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- **No todos los números racionales finitos en base 10 tienen representación finita en base 2.**
- Por ejemplo, el número 0.1_{10}

$$0.1_{10} = 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-8} + 1 \times 2^{-9} + (?) = 0.0\overline{0011}_2$$

Potencia	Valor
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625

Potencia	Valor
2^{-5}	0.03125
2^{-6}	0.015625
2^{-7}	0.0078125
2^{-8}	0.00390625

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- No todos los números racionales finitos en base 10 tienen representación finita en base 2.
- **No tenemos forma de representar el 0.1_{10} de manera exacta en un computador.**

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- No todos los números racionales finitos en base 10 tienen representación finita en base 2.
- No tenemos forma de representar el 0.1_{10} de manera exacta en un computador.

Pero… cómo los guardamos en un computador?

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- No todos los números racionales finitos en base 10 tienen representación finita en base 2.
- No tenemos forma de representar el 0.1_{10} de manera exacta en un computador.

Pero… cómo los guardamos en un computador?

- De alguna forma tenemos que ser capaces de almacenarlos en nuestro computador, pero tenemos un problema. **Cuál?**

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- No todos los números racionales finitos en base 10 tienen representación finita en base 2.
- No tenemos forma de representar el 0.1_{10} de manera exacta en un computador.

Pero… cómo los guardamos en un computador?

- De alguna forma tenemos que ser capaces de almacenarlos en nuestro computador, pero tenemos un problema. Cuál?
- **Cómo podemos representar el “.”?**

Números racionales

- Perfecto! Tenemos una forma de representar los números racionales en binario. Pero hay que tener ciertas consideraciones…
- No todos los números racionales finitos en base 10 tienen representación finita en base 2.
- No tenemos forma de representar el 0.1_{10} de manera exacta en un computador.

Pero… cómo los guardamos en un computador?

- De alguna forma tenemos que ser capaces de almacenarlos en nuestro computador, pero tenemos un problema. Cuál?
- Cómo podemos representar el “.”?

Realmente no podemos representar el símbolo del punto en binario dado que tenemos solamente el 0 y 1 (en base 2) para elegir y ya los estamos usando para darle valor a nuestros números, qué podemos hacer entonces?

Punto Fijo

- Esta representación corresponde a tener una cantidad fija de bits que representen la parte fraccional del número, o tener un factor de escalamiento conocido. Vamos a usar la primera definición por claridad.
- Para representar un número racional negativo, al igual que los números enteros podemos optar a usar la representación signo/magnitud o complemento de 2.

Punto Fijo: Signo/Magnitud

- Primero, veremos cómo podemos representar números racionales con signo, usando una cantidad fija de bits para la parte fraccional y reservando un bit para el signo.
- Si tenemos un número de *N* bits, podemos usar el primer bit para representar el signo, los siguientes *m* bits la parte entera y los últimos *l* para la fracción.

$$-7.125_{10} = -111.001_{2}$$

Punto Fijo: Signo/Magnitud

- Primero, veremos cómo podemos representar números racionales con signo, usando una cantidad fija de bits para la parte fraccional y reservando un bit para el signo.
- Si tenemos un número de *N* bits, podemos usar el primer bit para representar el signo, los siguientes *m* bits la parte entera y los últimos *l* para la fracción.

$$-7.125_{10} = -111.001_2$$

$$-111.001_2 = 1\ 111\ 0010$$

- El primer bit signo indica el signo.
- Los siguientes *m*=3 bits son la parte entera.
- Los ultimos *l*=4 bits la parte fraccional.

Punto Fijo: Complemento de 2

- Ahora, veremos cómo podemos representar números racionales con signo, usando punto fijo y complemento de 2.
- Si tenemos un número de *N* bits, los primeros *m* bits indican la parte entera y los últimos *l* bits la fracción. Si el número es negativo, es necesario calcular el complemento a 2 de este número.

$$-7.125_{10} = -111.001_{2}$$

Punto Fijo: Complemento de 2

- Ahora, veremos cómo podemos representar números racionales con signo, usando punto fijo y complemento de 2.
- Si tenemos un número de *N* bits, los primeros *m* bits indican la parte entera y los últimos *l* bits la fracción. Si el número es negativo, es necesario calcular el complemento a 2 de este número.

$$-7.125_{10} = -111.001_2$$

$$-111.001_2 = -\ 0111\ 0010$$

- **Los siguientes *m*=4 bits son la parte entera.**
- **Los ultimos *l*=4 bits la parte fraccional.**

Punto Fijo: Complemento de 2

- Ahora, veremos cómo podemos representar números racionales con signo, usando punto fijo y complemento de 2.
- Si tenemos un número de *N* bits, los primeros *m* bits indican la parte entera y los últimos *l* bits la fracción. Si el número es negativo, es necesario calcular el complemento a 2 de este número.

$$-7.125_{10} = -111.001_2$$

$$-111.001_2 = -\ 0111\ 0010$$

- **Los siguientes *m*=4 bits son la parte entera.**
- **Los ultimos *l*=4 bits la parte fraccional.**

Calculamos el complemento a 2

Punto Fijo: Complemento de 2

- Ahora, veremos cómo podemos representar números racionales con signo, usando punto fijo y complemento de 2.
- Si tenemos un número de *N* bits, los primeros *m* bits indican la parte entera y los últimos *l* bits la fracción. Si el número es negativo, es necesario calcular el complemento a 2 de este número.

$$-7.125_{10} = -111.001_2$$

$$-111.001_2 = -\ 0111\ 0010$$

$$-111.001_2 = 1000$$

$$1110$$

- **Los siguientes *m*=4 bits son la parte entera.**
- **Los ultimos *l*=4 bits la parte fraccional.**

Calculamos el complemento a 2

Punto Fijo

- Entonces,
  - Signo/Magnitud: $-7.125_{10} = 11110010$
  - Complemento a 2: $-7.125_{10} = 10001110$

Punto Fijo

- **Ventajas**: Simple de entender y usar.
- **Desventajas**: El rango que se puede representar es muy acotado dado que tenemos fija la cantidad de bits que son utilizados para la parte entera.

Queremos un enfoque más dinámico, a veces queremos representar números muuuyyy grandes y a veces chiquitos.

Punto Flotante

- Esta representación busca tener un punto “movible”, para poder representar números muy grandes o muy chicos sin tener que comprometernos con una cantidad fija de bits para la fracción.
- Como logramos esto? Usaremos la notación científica normalizada pero en base 2.

Punto Flotante

- Esta representación busca tener un punto “movible”, para poder representar números muy grandes o muy chicos sin tener que comprometernos con una cantidad fija de bits para la fracción.
- Como logramos esto? Usaremos la notación científica normalizada pero en base 2.

$$-7.125_{10} = -111.001_{2} = -1.11001_{2} \times 2^{2}$$

Punto Flotante

- Esta representación busca tener un punto “movible”, para poder representar números muy grandes o muy chicos sin tener que comprometernos con una cantidad fija de bits para la fracción.
- Como logramos esto? Usaremos la notación científica normalizada pero en base 2.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

Para tener la notación científica normalizada, siempre debe ser de la forma:

$$1.yyy \times 2^z$$

Punto Flotante

- Esta representación busca tener un punto “movible”, para poder representar números muy grandes o muy chicos sin tener que comprometernos con una cantidad fija de bits para la fracción.
- Como logramos esto? Usaremos la notación científica normalizada pero en base 2.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

Para tener la notación científica normalizada, siempre debe ser de la forma:

$$1.yyy \times 2^z$$

Cómo hacemos para representar el 0 entonces? Lo veremos pronto.

Punto Flotante

- Entonces nos vamos a aprovechar de la representación en notación científica normalizada para simular el movimiento del punto.
- Existen diversas maneras de representarlos y a veces un fabricante puede querer utilizar una representación sobre otra para un producto en específico.
- El estándar que utilizaremos es el **IEEE-754**. De esta forma todos nos podemos entender al usar números de punto flotante y no interpretaremos un mismo número de distintas formas. Viva los estandares!

Punto Flotante: IEEE-754

- Primero veremos los números de punto flotante de precisión simple (single-precision floating-point).
- Estos números son de 32 bits, donde los bits se reparten de la siguiente manera:
  - El primer bit indica el **signo** de la **mantisa**.
  - Los siguientes 8 bits corresponden al **exponente** (veremos como se representa).
  - Y los siguientes 23 bits serán la **mantisa**.

Punto Flotante: IEEE-754

- **Exponente:** El exponente corresponderá al valor del exponente en la representación en notación científica normalizada, desplazado en 127 unidades. Esto nos permite representar el signo del exponente sin usar la representación signo/magnitud ni complemento de 2. Esto nos permite tener fracciones muy chicas, o números muy grandes.

- **Mantisa:** La mantisa en esta representación no solo equivale a la mantisa en la notación científica (mantisa x 2^{exp}), si no que como sabemos que siempre estará normalizada, podemos olvidarnos del primer uno y guardamos lo que sigue, así nos "ahorramos" un bit y podemos usarlo para más información.

Entonces estos valores corresponden a los siguientes

- 1.mantisa x 2^{exp} , siendo exponente = exp + 127

Punto Flotante: IEEE-754

- Ahora, volvamos al problema mencionado anteriormente… cómo representamos el 0?
- Como estamos no hay forma de hacerlo, por eso el estándar reservó la combinación de exponente y mantisa cero, como el 0. Entonces vamos a tener dos formas de representar el cero, uno positivo y otro negativo.

  +0: 0 00000000 00000000000000000000000

  -0: 1 00000000 00000000000000000000000
- También vamos a tener otras tres combinaciones reservadas que son super útiles en la práctica.
  - **+ Infinity**: Si el número equivale a infinito positivo. El exponente corresponde a 11111111 y mantisa 0.

    + Infinity: 0 11111111 00000000000000000000000
  - **- Infinity**: Si el número equivale a infinito negativo. El exponente corresponde a 11111111 y mantisa 0.

    - Infinity: 1 11111111 00000000000000000000000
  - **NaN** (Not a Number): Si el valor no corresponde a un número. El exponente corresponde a 11111111 y la mantisa es distinta de 0.

    NaN: x 11111111 xxxxxxxxxxxxxxxxxxxxxxx

Punto Flotante: IEEE-754

- Veamos un ejemplo.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

Punto Flotante: IEEE-754

- Veamos un ejemplo.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

- Ya tenemos la representación en notación científica normalizada del número. **Recuerden que es muy importante que el número siempre parta con 1.xxx!**

Punto Flotante: IEEE-754

- Veamos un ejemplo.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

- Ya tenemos la representación en notación científica normalizada del número. **Recuerden que es muy importante que el número siempre parta con 1.xxx!**

Construyamos de a poco el número.

-7.125_{10}: x xxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx

Punto Flotante: IEEE-754

- Veamos un ejemplo.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

- Ya tenemos la representación en notación científica normalizada del número. **Recuerden que es muy importante que el número siempre parta con 1.xxx!**

Construyamos de a poco el número.

- **Signo**: El signo de la mantisa es negativo, por lo que el bit de signo equivale a 1.

-7.125_{10}: 1 xxxxxxxx xxxxxxxxxxxxxxxxxxxxxxx

Punto Flotante: IEEE-754

- Veamos un ejemplo.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

- Ya tenemos la representación en notación científica normalizada del número. **Recuerden que es muy importante que el número siempre parta con 1.xxx!**

Construyamos de a poco el número.

- **Signo**: El signo de la mantisa es negativo, por lo que el bit de signo equivale a 1.
- **Exponente**: El valor del exponente es 2, pero debemos desplazarlo en 127 unidades. Entonces nuestro nuevo valor es 2 + 127 = 129, que es $1000\ 0001_2$.

-7.125_{10}: 1 10000001 xxxxxxxxxxxxxxxxxxxxxxx

Punto Flotante: IEEE-754

- Veamos un ejemplo.

$$-7.125_{10} = -111.001_2 = -1.11001_2 \times 2^2$$

- Ya tenemos la representación en notación científica normalizada del número. **Recuerden que es muy importante que el número siempre parta con 1.xxx!**

Construyamos de a poco el número.

- **Signo**: El signo de la mantisa es negativo, por lo que el bit de signo equivale a 1.
- **Exponente**: El valor del exponente es 2, pero debemos desplazarlo en 127 unidades. Entonces nuestro nuevo valor es 2 + 127 = 129, que es $1000\ 0001_2$.
- **Mantisa**: Corresponde al valor que sigue después del 1., en este caso es 11001. Debemos rellenar con ceros a la derecha para completar los 23 bits.

-7.125_{10}: 1 10000001 11001000000000000000000

Punto Flotante: IEEE-754

- Solo queda ver los números de punto flotante de precisión doble (double-precision floating-point).
- Son equivalentes a los de precisión simple, solo que con más bits.

- Estos números son de 64 bits, donde los bits se reparten de la siguiente manera:
  - El primer bit indica el **signo** de la **mantisa**.
  - Los siguientes 11 bits corresponden al **exponente** desplazado en 1023.
  - Y los siguientes 52 bits serán la **mantisa**.

Punto Flotante: IEEE-754

- **Ventajas**: Tenemos un mayor rango de números.
- **Desventajas**: Se pierde precisión.

Este es un *trade-off* que en la práctica vale totalmente la pena.

FPU

- Si quisiéramos trabajar con números flotantes en nuestro computador básico, sería necesario tener una nueva componente capaz de operar sobre ellos.
- A esta la llamaremos *Floating Point Unit* (FPU).
- No la usaremos en nuestro computador básico, pero es necesario saber que las operaciones aritméticas entre números de punto flotante son muy diferentes a las que usamos sobre números enteros con la ALU.

Veremos más de esto en la próxima clase!

DCC

DEPARTAMENTO DE CIENCIA
DE LA COMPUTACIÓN

IIC2343 - Arquitectura de Computadores

Clase 10 - Representación de números racionales

Susana Figueroa