DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2025)

Ayudantía 7

Ayudantes: Daniela Ríos (danielaarp@uc.cl), Alberto Maturana (alberto.maturana@uc.cl), Nicolás Romo (nroma@uc.cl)

Pregunta 1: Conversión a `IEEE-754`

Transforme los siguientes números al formato IEEE-754 (32 bits):

1. 293
2. 0,3125
3. -14,5
4. 0,2

Solución:

1. La representación binaria de 293 es `100100101b` $=$ `1,00100101b` $\times 2^8$, por lo tanto, su exponente desfasado es: $127 + 8 = 135 =$ `10000111b` y su bit de signo es `0`, dado que el número es positivo.

Considerando lo anterior se obtiene que la representación es:

`0 10000111 00100101000000000000000`

2. Es posible observar que la representación binaria del número es: $0{,}3125 = 5^5 \times 10^{-4} = 5 \times 2^{-4} =$ `101b` $\times 2^{-4} =$ `0.0101b`. Según el estándar, se debe recordar que el número debe estar normalizado, por lo tanto, su valor es: `1,01b` $\times 2^2$. Con lo anterior es posible observar en la representación que:

- El número es positivo, por lo tanto, el valor del bit de signo es `0b`.
- El valor del exponente desfasado es exp - 127 = -2, por lo tanto: exp = 125 = `01111101b`.
- El valor de la mantisa es: `01000000000000000000000b`

Considerando lo anterior se obtiene que la representación es:

`0 01111101 01000000000000000000000`

3. Es posible observar que la representación binaria del número -14,5 es `1110,1b`. Según el estándar se debe recordar que debe estar normalizado, por lo tanto, su valor es: -14,5 = `1.1101b` $\times 2^3$. Con lo anterior es posible observar en la representación que:

- El número es negativo, por lo tanto, el valor del bit de signo es `1b`.
- El valor del exponente desfasado es exp - 127 = 3, por lo tanto: exp = 130 = `10000010b`.
- el valor de la mantisa es: `11010000000000000000000b`.

Considerando lo anterior se obtiene que la representación es:

`1 10000010 11010000000000000000000`

4. Es posible observar que 0.2 se puede representar como la división de 2 entre 10 en binario, por lo tanto, la representación binaria del número es: 0.2 = `10b` : `1010b` = $\mathtt{0.\overline{0011}b}$ (ocupando el algoritmo de división binaria visto en clases). Normalizando el número, se obtiene que su representación es: $\mathtt{0.\overline{0011b}} = \mathtt{1.\overline{10011}b} \times 2^{-3}$, por lo tanto, es posible observar que en la representación:

- El número es positivo, por lo tanto, el valor del bit de signo es 0.
- El valor del exponente desfasado es exp - 127 = -3, por lo tanto: exp = 124 = `01111100b`.
- El valor de la mantisa es: `10011001100110011001101b`.

Considerando lo anterior se obtiene que la representación es:

`0 01111100 10011001100110011001101`

Pregunta 2: Conversión a decimal

Transforme los siguientes números de su representación hexadecimal a binario. Luego, interpretándolos con el estándar `IEEE-754`, indique su valor en decimal.

1. `0x43e88000`
2. `0xc35e5000`

Solución:

1. Recuerde que, para la transformación de hexadecimal a binario, se debe reemplazar el valor de cada dígito hexadecimal a su representación binaria usando 4 bits, con esto, es posible observar que el valor en binario del número es: `01000011111010001000000000000000b`. Luego, recordando el estándar `float32`, se obtiene que el valor del signo, exponente y mantisa del número son:

`0 10000111 11010001000000000000000`

Con esto, es posible se puede deducir que:

- El valor del signo es `0`, por lo tanto el número es positivo.
- El valor del exponente es `10000111b` = 135, esto implica que el exponente no desfasado es: 135 - 127 = 8.

Finalmente, se tiene que la representación queda de la siguiente manera: `1,11010001` $\times 2^8 =$ `111010001b`, el cual corresponde a 465.

2. En este caso la representación del número es: `11000011010111100101000000000000b`. Luego, en el estándar `float32`, se obtiene que el valor del signo, exponente y mantisa del número son:

`1 10000110 10111100101000000000000`

Con esto, es posible se puede deducir que:

- El valor del signo es `1`, por lo tanto el número es negativo.
- El valor del exponente es `10000110b` = 134, esto implica que el exponente no desfasado es: 134 - 127 = 7.

Finalmente, se tiene que la representación queda de la siguiente manera: `1,10111100101` $\times 2^7 =$ `11011110,0101b`, el cual corresponde a -222.3125.

Pregunta 3: Suma y multiplicación de `floats` (I1 2025-1)

En los siguientes incisos, se le entregará un par de números reales A, B en formato `float` del estándar `IEEE754` con notación hexadecimal. En cada caso, debe interpretar sus valores y realizar las operaciones de suma o multiplicación solicitadas. Luego, debe indicar tanto el valor teórico (resultado de la operación) como el valor real (almacenado como `float` del estándar `IEEE754`) de la operación. Si son iguales, explicite que lo son. Los valores deben presentarse en base binaria con notación científica normalizada.

1. A + B. A = `0x40400000`, B = `0x3F800000`.
2. A × B. A = `0x41820000`, B = `0x3F200000`.
3. A × B. A = `0x42b20000`, B = `0x7F800000`.

Solución:

1.

- A = `0100000001000000 0000000000000000b`.
  - **Signo**: `0b` → positivo.
  - **Exponente**: `10000000b` → 128 - 127 = 1.
  - **Significante**: `10000000000000000000000b` → `1,1b`.
  - **Valor**: $(\texttt{1,1} \times \texttt{10}^{1})\texttt{b} = \texttt{11b} = 3$.
- B = `0011111110000000 0000000000000000b`.
  - **Signo**: `0b` → positivo.
  - **Exponente**: `10000000b` → 128 - 127 = 1.
  - **Significante**: `00000000000000000000000b` → `1,0b`.
  - **Valor**: $(\texttt{1,0} \times \texttt{10}^{0})\texttt{b} = \texttt{1b} = 1$.

De lo anterior, se deduce que la operación a realizar es es 3 + 1. En este caso, no se evaluará la forma en la que se realice la suma; solo que su resultado sea correcto. Aquí se opta por seguir el algoritmo visto en clases: se igualan los exponentes y B se expresa como $\texttt{0,1} \times \texttt{101}$. Luego, sumando los significantes, se tiene A + B = $(\texttt{10,0} \times \texttt{101})\texttt{b} = (\texttt{1,0} \times \texttt{1010})$. Finalmente, si se almacena como float, se tiene el valor `01000000100000000000000000000000b` = `0x40800000` = 4. No existe pérdida de precisión.

2.

- A = `0100000110000010 0000000000000000b`.
  - **Signo**: `0b` → positivo.
  - **Exponente**: `10000011b` → 131 - 127 = 4.
  - **Significante**: `00000100000000000000000b` → `1,000001b`.
  - **Valor**: $(\texttt{1,000001} \times \texttt{10}^{100})\texttt{b} = \texttt{10000,01b} = 16{,}25$.
- B = `0011111100100000 0000000000000000b`.
  - **Exponente**: `01111110b` → 126 - 127 = -1.
  - **Significante**: `01000000000000000000000b` → `1,01b`.
  - **Valor**: $\texttt{1,01} \times \texttt{10}^{\texttt{-1}} = \texttt{0,101b} = 0{,}625$.

- **Signo**: `0b` $\rightarrow$ positivo.

De lo anterior, se deduce que la operación a realizar es igual a $16,25 \times 0,625$. En este caso, no se evaluará la forma en la que se realice la suma; solo que su resultado sea correcto. Aquí se opta por seguir el algoritmo visto en clases: se suman los exponentes sin desfase y se llega al valor $-1 + 4 = 3 =$ `11b`. Luego, se multiplican los significantes: $\texttt{1,01} \times \texttt{1,000001} = (\texttt{101} \times \texttt{10}^{\texttt{-2}}) \times (\texttt{1000001} \times \texttt{10}^{\texttt{-6}}) = (\texttt{101} \times \texttt{1000001}) \times \texttt{10}^{-8} = \texttt{101000101} \times \texttt{10}^{\texttt{-8}} = \texttt{1,01000101}$. Finalmente, se tiene como resultado $\texttt{1,01000101} \times \texttt{10}^{11} = \texttt{1010,00101} = 10,15625$, que, en formato float, se almacena como `01000001001000101000000000000000b` = `0x41228000`. No existe pérdida de precisión.

3.

- A = `0100001010110010 0000000000000000b`.
  - **Signo**: `0b` $\rightarrow$ positivo.
  - **Exponente**: `10000101b` $\rightarrow$ 133 - 127 = 6.
  - **Significante**: `01100100000000000000000b` $\rightarrow$ `1,011001b`.
  - **Valor**: $(\texttt{1,011001} \times \texttt{10}^{110})\texttt{b} = \texttt{1011001b} = 89$.
- B = `0111111110000000 0000000000000000b`.
  - **Signo**: `0b` $\rightarrow$ positivo.
  - **Exponente**: `11111111b` $\rightarrow$ 255 - 127 = 128.
  - **Significante**: `00000000000000000000000b` $\rightarrow$ `1,0b`.
  - **Valor**: `inf`.

De lo anterior, se deduce que la operación a realizar es igual a $89 \times$ `inf`. Lo cual es infinito. Es importante notar que, es bueno identificar los números con los que se está operando, pues, en este caso, solamente con verificar que B es infinito se puede saber el resultado de la operación.

Pregunta 4: Preguntas conceptuales

1. Decida si las siguientes afirmaciones son verdaderas o falsas:

   1. `0xff818210` interpretado en `IEEE-754` corresponde a $-\infty$.
   2. $2^{25} + 1$ se puede representar de manera exacta en el formato `IEEE-754`.

2. **(I1 2023-2)-`arquiFloat`**: Suponga que se crea el tipo de dato `arquiFloat` de 32 bits a partir del nuevo estándar `IIC2343` para representar números de punto flotante. A diferencia del tipo de dato `float` del estándar `IEEE754`, este posee el siguiente formato:

- Un bit de signo de significante.
- Un bit de signo de exponente.
- 10 bits de exponente no desfasado.
- 20 bits de significante normalizado con bit implícito.

Compare los dos estándares respondiendo las siguientes preguntas:

1. Señale una ventaja y una desventaja del tipo de dato `arquiFloat` respecto del tipo de dato `float`.
2. Señale cómo se representa el resultado de la operación $2^{22}+7$ en formato `float` y `arquiFloat`. ¿Existe pérdida de precisión en alguno de los casos?

Solución: 1.1 La afirmación es FALSA, convirtiendo el número a su represntación binaria se obtiene que esta es: `11111111100000011000001000010000`. Al interpretarla con el estándar IEEE-754 se puede observar que:

- Su bit de signo es `1`.
- Su exponente es: `11111111`.
- Su mantisa es: `00101100000000000000000`.

Con lo anterior se concluye que la representación equivale a un `NaN`.

1.2 La afirmación es FALSA, esto se debe a que $2^{25} = \texttt{1,0000000000000000000000000} \times 2^{25}$. Luego, se debe sumar `1 = 0,0000000000000000000000001` $\times$ 225. Lo cual resulta en que $225 + 1 = \texttt{1,0000000000000000000000001} \times 2^{25}$. Sin embargo, como en el estándar `IEEE-754` toma los 23 primeros bits para representar la mantisa, entonces, esta tiene como valor: `00000000000000000000000`, por lo tanto, se pierde precisión para representar este número.

2.1. Una ventaja es que `arquiFloat` posee un rango mayor respecto a los números que puede representar dada su cantidad de bits de exponente. En este caso, el mayor exponente representable es `1111111110b` = 1022 (si se asume la reserva para los valores infinitos y con bit de signo positivo), que es significativamente mayor al exponente máximo de `float` igual a 127. Por otra parte, una desventaja es la pérdida de precisión. En este caso, si incluimos el bit implícito, `arquiloat` puede representar números de hasta 21 bits en comparación a float que puede representar números de hasta 24 bits.

2.2. El número que se busca representar, haciendo la suma entre 2^{22} y 7 con notación científica, es igual a $\texttt{1,0000000000000000000111} \times \texttt{10}^{\texttt{00010110}}$. A continuación veremos cómo se almacena en cada representación:

- `float`: El número es positivo (bit de signo igual a cero) y el exponente desfasado es igual a 22 + 127 = 149. Por lo tanto, se almacena la secuencia:

$$\texttt{0}_{\texttt{signo}}\texttt{10010101}_{\texttt{exp}}\texttt{00000000000000000001110}_{\texttt{signif}}$$

. Por lo tanto, no existe pérdida de precisión con esta representación.

- `arquiFloat`: Tanto el número como su exponente son positivos (bits de signo iguales a cero). Por lo tanto, se almacena la secuencia:

$$\mathtt{0}_{\mathtt{signo.signif}}\mathtt{0}_{\mathtt{signo.exp}}\mathtt{0000010110}_{\mathtt{exp}}\mathtt{00000000000000000001}_{\mathtt{signif}}$$

  Se pierden dos bits de precisión, por lo que se pierde el valor exacto del número: se redondea a 222 + 4.

Feedback ayudantía

Escanee el QR para entregar feedback sobre la ayudantía.