PUNTO FLOTANTE

número representado = $M \times base^{exp}$

exp → exponente

M → mantisa

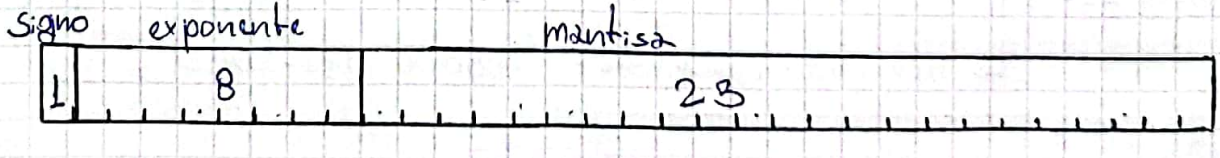
- Casi todos los lenguajes de programación ofrecen datos en punto flotante.
- Todo sistema operativo debe responder a excepciones punto flotante (overflow)

Estandarización del formato punto flotante: IEEE-754 (año 1982)

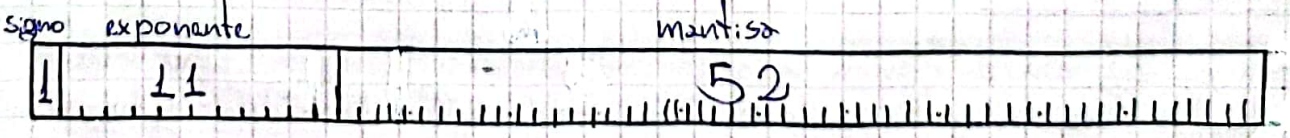
- Se implementó por primera vez en los Intel 8087.
- En 1985 se aceptó como estandar universal.
- En 2008 fue modificado.
- La precisión puede ser → simple
  → doble.

Norma IEEE-754

Simple Precisión → 32 bits



Doble Precisión → 64 bits



→ Cuando lo paso a binario tengo $1.\text{mantisa} \cdot 2^{exp}$

↳ siempre voy a tener un 1 → bit implícito

- Lo escribo tal que siempre la "," se encuentre a la derecha del dígito no nulo más significativo

Ventajas:

- La representación binaria es única para un número dado.
- Todos los bits de la mantisa son significativos.
- Es más fácil comparar dos números:
  i. Comparo exponentes.
  ii. Comparo mantisas.

Representación

→ Se usa exceso n (ni Cb, ni Cb-1, ni M y S)

- Escribo valor en binario.
- Le sumo n.

Para simple precisión → exp. de 8 bits

Uso exceso 127 → máx. valor representable: $1111\,1110|_2 = 127|_{10}$.
→ mín. valor representable: $0000\,0001|_2 = -126$

1 ≤ Mantisa < 2

Para doble precisión → exp. de 11 bits

Uso exceso 1023 → máx. valor representable: $1111\,1110|_2 = 1023|_{10}$
→ menor valor representable: $0000\,0001|_2 = -1022|_{10}$

1 ≤ Mantisa < 2

Rango representable

→ lo que determina de a cuánto se puede representar es la mantisa. Entre 1,00..0 y 1,00...1 "de a cuánto avanzo"
hay varios números que no puedo representar → $\Delta = 2^{-cant.\ bits\ mantisa}$ y luego $\Delta \cdot 2^{exponente}$
↳ salto mantisa

⇒ el salto es $2^{exp.} - cant.\ bits\ mantisa$
↳ pueden ser 23 (sp.) o 52 (dp)

NOTA

Exponent Underflow: No puedo representar un valor por ser muy pequeño.
⇒ El resultado es 0. (forzado)

Exponent Overflow: No puedo representar un valor por ser muy grande (fuera de rango).
⇒ El resultado es ∞ (forzado)

Valores especiales

Cero: todo en 0.
↳ el signo es indistinto.

Infinito: todo el exponente en 1, la mantisa en 0.
↳ el signo determina si es $+\infty$ ó $-\infty$.

Not a number (NaN): todo el exponente en 1, la mantisa con lo que sea (distinto de 0).
↳ el signo es indistinto
↳ por ej. ∞/∞ = NaN

→ Al tener estos valores, puedo manejar muchos casos que el punto fijo no.
Por ej: $6/3/\infty$ = → error en punto fijo.
↳ 0 en punto flotante

	Signo	Exponente	Mantisa
Cero	indistinto	0's	0's
Infinito	+ → $+\infty$ - → $-\infty$	1's	0's
Not a Number	indistinto	1's	cualquier cosa (menos todo 0's)

Conversión

Conversión de cualquier base a cualquier otra base en punto flotante

- Realizo la conversión del número a binario (punto fijo) → ese número está multiplicado por $2^0 = 1$.
- Movemos la coma d espacios de manera tal que quede a la derecha del primer valor no nulo → normalizamos.

  Dependiendo de si movimos la coma a derecha o izquierda, cambia el signo del exponente:
  - Si se movió a izquierda ⇒ el valor multiplica a 2^d
  - Si se movió a derecha ⇒ el valor multiplica a 2^{-d}

  El valor del exponente a utilizar será el obtenido + el exceso correspondiente (127 ó 1023)
- Para la mantisa ignoro el valor a la izquierda de la coma, y escribo los que están a la derecha.

  Si no son suficientes bits, agrego 0s a derecha
- Convierto el resultado a la base deseada

Ejemplo: $2149{,}35|_{10}$ a hexadecimal punto flotante en simple precisión

- Lo paso a binario: $100001100101{,}0101100110$

  $= 100001100101{,}0101100110 \cdot 2^0$ ←

- Corro la coma → $1{,}00001100101\,0101100110 \cdot 2^{11}$

  11 lugares a izquierda

- Al exponente le sumo 127 ⇒ exponente = 138.
- Para la mantisa ignoro el 1 (dígito a la izquierda de la coma): 00001100101010101100110
  → Son 21 bits y necesito 23 ⇒ agrego dos 0s a derecha: 00001100101010101100110000.

signo	exponente	mantisa
0	10001010	00001100101010101100110000

→ binario.

- En hexadecimal: 45065598.

Conversión de cualquier base en punto flotante a cualquier otra base

- Realizo la conversión del número a base binaria pura.
  - Obtengo el signo: primer dígito.
  - Obtengo el exponente: siguientes 8 (simple precisión) o 11 (doble precisión) dígitos.
  - Obtengo la mantisa: siguientes 23 (simple precisión) o 52 (doble precisión) dígitos.

→ El verdadero exponente es el obtenido menos el exceso correspondiente (127 ó 1023).

- Escribo el número obtenido: (sin signo) $1, \text{mantisa} \cdot 2^{\text{exponente real}}$
- Escribo el número tal que el exponente sea 0.
- Convierto el número obtenido (que está en binario) a la base que quiera. Le agrego el signo

Ejemplo: $C28FFF00|_{16}$ (simple precisión) a base decimal.

- $C28FFF00|_{16} = 1100\ 0010\ 1000\ 1111\ 1111\ 1111\ 0000\ 0000|_2$.

⇒

signo	exponente	mantisa
1	10000101	000 1111 1111 1111 0000 0000

: signo: negativo

- exponente obtenido = 133 → exponente real = 133 − 127 = 6.
- Escribo el número obtenido: (sin signo) $1,000111111111111111 \cdot 2^6$ (ignoro 0s)
- Lo escribo con exponente nulo = $1000111,111111111 \cdot 2^0$.
  = 1000111,111111111
- Lo paso a decimal = 71,99804688.
- Le agrego signo = −71,99804688

Suma

Para sumar dos números en punto flotante:

1. Calculo la diferencia entre los exponentes: $|exp_1 - exp_2| = d$
2. Determino cuál es el número mayor y cuál el menor.
3. Corro d posiciones a la derecha la coma del número menor.
4. Encolumno y sumo las mantisas.
5. El exponente del resultado es el exponente del número mayor.
6. Normalizo la mantisa del resultado, ajustando el exponente de ser necesario.