



# Módulo 05

## Aritmética de Punto Flotante (Pt. 2)



Organización de Computadoras  
Depto. Cs. e Ing. de la Comp.  
Universidad Nacional del Sur



# Copyright

- Copyright © **2011-2024** A. G. Stankevicius
- Se asegura la libertad para copiar, distribuir y modificar este documento de acuerdo a los términos de la **GNU Free Documentation License**, Versión 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera
- Una copia de esta licencia está siempre disponible en la página <http://www.gnu.org/copyleft/fdl.html>
- La versión transparente de este documento puede ser obtenida de la siguiente dirección:

<http://cs.uns.edu.ar/~ags/teaching>



# Contenidos

- Inconvenientes con la representación **FXP**
- Un problema con el escalamiento
- Formatos para la representación **FLP**
- Concepto de orden de magnitud cero (**OMZ**)
- Situaciones especiales a contemplar
- Operaciones aritméticas básicas



# Aritmética de punto flotante

- Recordemos que hay dos variantes de **FLP**:
  - ➔ Normalizada: opera con operandos normalizados a fin de preservar en todo momento la mayor cantidad de bits de significancia
  - ➔ No normalizada: opera con operandos sin normalizar, dejando la decisión de llevar adelante el proceso de normalización en manos del programador



# Formatos de punto flotante

- La elección de un **formato** adecuado para representar números en punto flotante debe contemplar múltiples aspectos:
  - Evidentemente las características elegidas para **m**, **e** y **b** afectarán tanto el rango y como precisión de la representación resultante
  - En particular, **m** y **e** comparten una o más palabras, por lo que asignar más bits de precisión a uno redundará en quitárselos al otro



# Formatos de punto flotante

## ● Continúa:

- Incrementar el rango asignando más bits al exponente va a restringir a la cantidad de bits disponibles para la mantisa, lo cual disminuirá la precisión
- En contraste, asignar una menor cantidad de bits al exponente conduce exactamente a lo opuesto (mejora la precisión a costa del rango)
- Por último, el tercer parámetro, la base **b** elegida para la representación, tiene a su vez impacto tanto en el rango como en la precisión



# Formatos de punto flotante

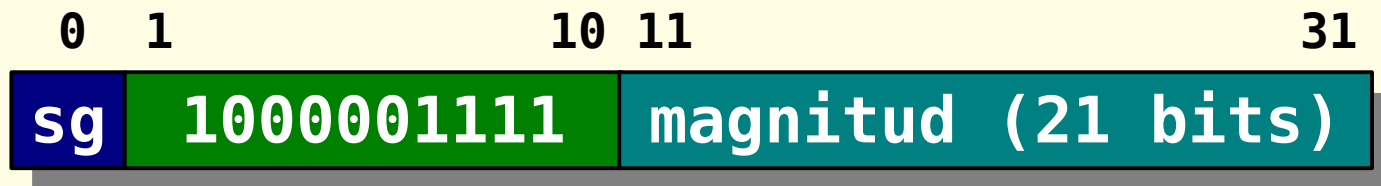
- También hay otras decisiones menos críticas:
  - La mantisa puede ser **entera ó fraccionaria**, según se asuma el punto fraccionario en el extremo derecho o izquierdo, respectivamente
  - A su vez, la mantisa puede estar representado de cualquiera de las formas antes vistas
  - El exponente será necesariamente **un entero signado**, si bien existen distintas formas de representación
  - Finalmente, la base se elegirá de acuerdo a la longitud de palabra y a la precisión y al rango deseado, pero **siempre ha de ser potencia de 2**



# Formatos de punto flotante

## ● Consideraciones acerca del exponente:

- El exponente se suele representar en **notación exceso**
- Para **n** bits, los valores representados se polarizan en  $2^{n-1}$ , es decir, el menor exponente representable es **0** y el mayor  **$2^n - 1$** , simplificando las comparaciones
- Por caso, para un número en punto flotante de **32** bits, con **10** bits para el exponentes y **22** para la mantisa, el exponente **15** se codifica como  **$15 + 2^9 = 527$** :





# Formatos de punto flotante

## ● Consideraciones acerca de la base:

- ➔ La base, al ser implícita, no consume espacio en la representación, por lo que podemos vernos tentados de adoptar una base más grande con el objeto de obtener un rango más amplio
- ➔ Por caso, se han ensayado configuraciones con bases diversas tales como **2**, **4**, **8** e incluso **16**
- ➔ No obstante, el resultado al hacer uso de bases elevadas no fue satisfactorio (se perdía precisión)
- ➔ En la actualidad se usa exclusivamente la base **2**



# Formatos de punto flotante

- Consideraciones acerca de la mantisa:
  - ➔ Con respecto a la mantisa, se deben tomar dos decisiones: **qué representación adoptar** y **si se trabajará con mantisas normalizadas o no**
  - ➔ Estas decisiones están relacionadas, ya que la normalización de las mantisas es un procedimiento que variará en función de la representación elegida
  - ➔ La idea básica del proceso de normalización consiste en descartar aquellos dígitos que no estén aportando significancia a la mantisa (por así decir, los “ceros a la izquierda”)



# Proceso de normalización

- El **proceso de normalización** consiste en alterar la mantisa de manera controlada, esto es, sin que se afecte el valor representado
  - Es decir, en caso de **desplazar los bits de la mantisa a derecha**, se debe compensar el corrimiento a izquierda del punto binario **incrementando el exponente de manera acorde**
  - Caso contrario, en caso de **desplazar los bits de la mantisa a izquierda**, se debe compensar el corrimiento a derecha del punto binario **decrementando el exponente de manera acorde**



# Normalización en SM

- La representación **FLP** es **inherentemente redundante**, el mismo número puede ser representado en más de una forma:

$$\dots = \mathbf{1} \times \mathbf{2^{18}} = \mathbf{0.1} \times \mathbf{2^{19}} = \mathbf{0.01} \times \mathbf{2^{20}} = \mathbf{0.001} \times \mathbf{2^{21}} = \dots$$

- En general es deseable en una implementación adoptar una única forma canónica
- Si la mantisa está codificada en **SM** fraccionaria y la base adoptada es **2**, se dice que **la mantisa está normalizada** si el dígito a la derecha del punto fraccionario es distinto de cero, a saber:

$$\mathbf{0.1} \times \mathbf{10^{19}}$$



# Normalización en RC y DRC

- Una fracción binaria en **RC** o en **DRC** se dice normalizada cuando **el bit de signo difiere del bit a su lado** (el bit más significativo de la mantisa)
  - Nótese que no habrá ceros al comienzo de la mantisa de un número normalizado positivo ni habrá unos al comienzo de un número normalizado negativo
  - El **rango de valores** para una mantisa normalizada **m** representada en complemento a la base **b** resulta:

$$1 / b \leq |m| < 1$$



# Hidden bit

- En caso de trabajar con mantisas normalizadas y únicamente si la base implícita es  $b = 2$ , se podrá optimizar la significancia:
  - ➔ Considerando que sólo se opera con magnitudes normalizadas, sabemos de antemano que el primer bit de significancia dependerá del bit de signo
  - ➔ Por ende, como el bit de signo siempre se almacena, se puede prescindir de ese primer bit de significancia
  - ➔ Esta optimización, que permite duplicar la precisión de la mantisa, sólo es aplicable en este contexto en particular y se la denomina **hidden bit**



# Overflow y underflow

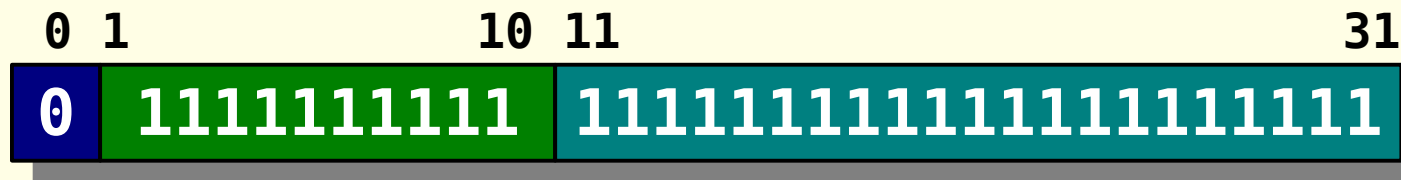
- Las operaciones de punto flotante pueden dar a lugar a **situaciones singulares** aún cuando estas sean aplicadas a datos perfectamente válidos
  - ➔ Usaremos los símbolos  $+\infty$  y  $-\infty$  para referir a los quasi-infinitos positivos y negativos correspondientes a los números en notación **FLP** cuyo exponente excede el máximo valor positivo representable
  - ➔ A su vez, usaremos los infinitesimales  $+\epsilon$  y  $-\epsilon$  para referir a aquellos números en notación **FLP** cuyo exponente excede al máximo valor negativo representable



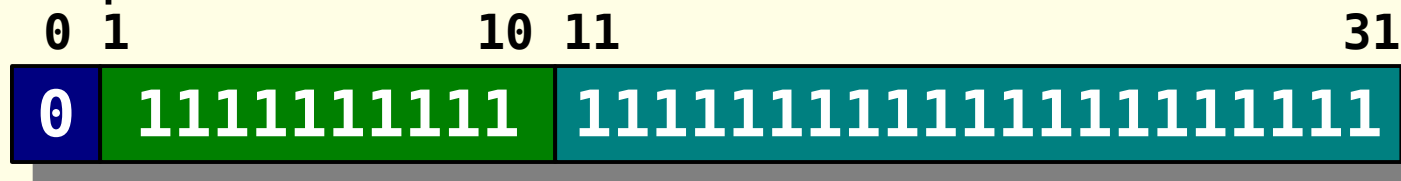
# Overflow y underflow

- Por caso, para una representación **FLP** con **32** bits de precisión, **10** bits para el exponente (representado en exceso-**512**) y **22** para la mantisa fraccionaria en **SM** y base 2:

→ Mayor positivo:  $(-1)^0 \times (1 - 2^{-21}) \times 2^{511}$



→ Mayor positivo normalizado:  $(-1)^0 \times (1 - 2^{-21}) \times 2^{511}$







# Orden de magnitud cero

• Todo número en notación **FLP** cuya mantisa **m** sea cero se denominan **orden de magnitud cero (OMZ)**

- El exponente **e** puede asumir cualquier valor válido
- Estos números surgen de efectuar el siguiente cálculo, para una mantisa **m** y un exponente **e** arbitrarios:

$$(m, e) - (m, e) = (0, e)$$

- Esta familia de números representa un amplio rango de valores indeterminados que para una base **b** y **p** bits de precisión en la mantisa satisfacen que:

$$-b^{e-p} < (0, e) < b^{e-p}$$



# Orden de magnitud cero

- Entre los valores que pertenecen al orden de magnitud cero se distingue aquel que tiene exponente cero como el **verdadero cero**
- La totalidad de los números positivos y negativos representables son entonces:

$$+\varepsilon < +N < +\infty$$

$$-\infty < -N < -\varepsilon$$

- El verdadero cero constituye la línea divisoria entre estos rangos positivos y negativos



# Situaciones especiales

- Consideremos las siguientes operaciones aritméticas la cuales involucran números normalizados, infinitos, infinitesimales y **OMZs**.
- Con operaciones de suma y de resta:

$\infty \pm N = \infty$	$N \pm \varepsilon = N$	$N - \infty = -\infty$
$\varepsilon - N = -N$	$\infty + \infty = \infty$	$\infty \pm \varepsilon = \infty$
$\varepsilon \pm \varepsilon = \varepsilon$	$\varepsilon - \infty = -\infty$	$\infty - \infty = \infty$
$N \pm z = N$	$z - N = -N$	$z_1 \pm z_2 = z_3$



# Situaciones especiales

- Con la operación de multiplicación:

$\infty \times N = \infty$	$N \times \varepsilon = \varepsilon$	$\infty \times \infty = \infty$
$\varepsilon \times \varepsilon = \varepsilon$	$\infty \times \varepsilon = \infty$	$z_1 \times z_2 = z_3$
$N \times z_1 = z_2$		

- Con la operación de división:

$\infty / N = \infty$	$\varepsilon / N = \varepsilon$	$N / \infty = \varepsilon$
$N / \varepsilon = \infty$	$\infty / \varepsilon = \infty$	$\varepsilon / \infty = \varepsilon$
$\infty / \infty = \infty$	$\varepsilon / \varepsilon = \infty$	$z_1 / N = z_2$



# Situaciones especiales

- Cabe acotar que muchas implementaciones eligen adoptar como resultados  $\infty \times \varepsilon = \infty$  y  $\varepsilon / \varepsilon = \infty$  en lugar de los también razonables  $\infty \times \varepsilon = \varepsilon$  y  $\varepsilon / \varepsilon = \varepsilon$ , respectivamente, a fin de destacar esa situación a través de la señalización del overflow
- El intento de ejecución de las operaciones  $N / z$  ó  $N / 0$  será suprimido, activando el flag de división por cero y disparando generalmente la interrupción o el trap asociado



# Operaciones aritméticas

- Las cuatro operaciones estándar de suma, resta, multiplicación y división pueden operar sobre números en notación **FLP** con un mejor rango y precisión que en notación **FXP**
- Asumidos los operandos  $X_1 = (m_1, e_1)$  y  $X_2 = (m_2, e_2)$ , con  $X_i = m_i \times b^{e_i}$ , siendo  $b$  la base implicada, analizaremos de qué manera se implementan cada una de estas cuatro operaciones al operar con operandos normalizados



# Operaciones aritméticas

- Nótese que la mantisa **m** será una fracción con **p** dígitos significativos (excluido el signo), dando el siguiente rango:

$$1 / b \leq |m| \leq 1 - b^{-p} < 1$$

- En el caso del exponente **e**, se trata de un número signado de **q** bits codificado en exceso  $b^{q-1}$ , tal que:

$$0 \leq e \leq b^q - 1$$





# Operaciones aritméticas

## ● Algoritmo para la suma/resta:

- Detectar el operando con mayor exponente
- Desplazar la mantisa del operando con menor exponente  $|e_1 - e_2| \times \log_2 b$  lugares a la derecha, ajustando según corresponda su exponente
- Sumar o restar las mantisas (ya que ahora se trata de dos números con un mismo exponente)
- De ser necesario, normalizar el resultado obtenido







# Ejemplo

## ● Continúa:

- Nótese que el resultado intermedio obtenido ya está normalizado, por lo que ese es el resultado final
- Finalmente, podemos volver a convertir a notación **FXP** el resultado antes obtenido, a manera de verificación de que todo haya salido bien:

$$\begin{aligned}X_1 + X_2 &= (-1)^{sg} \times m \times b^e \\ &= (-1)^0 \times 0.9296875 \times 2^4 \\ &= 14.875 \quad \checkmark\end{aligned}$$



# Operaciones aritméticas

## ● Algoritmo para la multiplicación:

- Sumar los dos exponentes; éste será en principio el exponente del resultado
- Multiplicar las dos mantisas, lo que a su vez determina el signo del resultado
- Normalizar el resultado de ser necesario, corriendo la mantisa a la izquierda y ajustando el exponente de manera acorde





# Ejemplo

## Continúa:

- El exponente del resultado resulta (luego de ajustar de manera adecuada el exceso):

$$X_1 \times X_2 = \begin{array}{cccc} & 0 & 1 & & 10 & 11 & & & & & 31 \\ & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ & - & 1000000110 & & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \end{array}$$

- Luego, la mantisa y el signo resultan (en este paso en principio descartamos los  $p$  bits menos significativos del producto):

$$X_1 \times X_2 = \begin{array}{cccc} & 0 & 1 & & 10 & 11 & & & & & 31 \\ & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ & 0 & 1000000110 & & 1000010101100000000000 & & & & & & & \end{array}$$



# Ejemplo

## Continúa:

- Nótese que el resultado intermedio obtenido nuevamente ya está normalizado, por lo que ese es el resultado final
- Finalmente, podemos volver a convertir a notación **FXP** el resultado antes obtenido, a manera de verificación de que todo haya salido bien:

$$\begin{aligned} X_1 \times X_2 &= (-1)^{sg} \times m \times b^e \\ &= (-1)^0 \times 0.52099609375 \times 2^6 \\ &= 33.34375 \quad \checkmark \end{aligned}$$





# Operaciones aritméticas

## ● Algoritmo para la división:

- Restar los dos exponentes (este será en principio el exponente del resultado)
- Dividir las dos mantisas (lo que a su vez determina el signo del resultado)
- Normalizar el resultado de ser necesario, corriendo la mantisa a la derecha y ajustando el exponente de manera acorde



# Observaciones

- Al calcular una suma o una resta, la mantisa del resultado se ubica necesariamente en el siguiente rango:

$$0 \leq |m| < 2$$

- Esto constituye un inconveniente cuando  $1 \leq |m| < 2$ , situación que se denomina **overflow de mantisa** u **overflow virtual**
  - Para solucionar esta situación simplemente se corre la mantisa un lugar a derecha, ajustando el exponente de manera acorde



# Observaciones

- En caso de obtenerse un **OMZ** como resultado, se debe tener en cuenta que **no será posible normalizarlo**
- Al calcular una multiplicación o una división es posible calcular en paralelo la mantisa y el exponente del resultado:

$$1 / b^2 \leq |m_1 \times m_2| < 1 \text{ (si } m_1 \neq 0 \text{ y } m_2 \neq 0 \text{)}$$

$$1 / b < |m_1 / m_2| < b \text{ (si } m_2 \neq 0 \text{)}$$



# Observaciones

● Luego de la multiplicación, pueden suscitarse dos escenarios:

→ Cuando  $1 / b \leq |m_1 \times m_2| < 1$ , no hace falta corregir la mantisa (pues ya está normalizada)

→ Cuando  $1 / b^2 \leq |m_1 \times m_2| < 1 / b$ , hay que corregir la mantisa (pues no está normalizada)

● La corrección de la mantisa, de requerirse, equivale a efectuar el siguiente cálculo:

$$(m_1, e_1) \times (m_2, e_2) = (m_1 \times m_2 \times b, e_1 + e_2 - 1)$$



# Observaciones

• Para el caso de la división, nunca hace falta normalizar, pues se presentan los siguientes escenarios:

→ Si  $m_1 < m_2$ , entonces  $m_1 / m_2$  ya está normalizado pues se verifica que  $1 / b \leq |m_1 / m_2| < 1$

→ Caso contrario, cuando  $m_1 \geq m_2$ , con  $m_2 \neq 0$ , se presenta un overflow virtual si  $1 \leq |m_1 / m_2| < b$ , cuyo ajuste correspondiente es el siguiente:

$$(m_1, e_1) / (m_2, e_2) = (m_1 / m_2 \times b^{-1}, e_1 - e_2 + 1)$$



# ¿Preguntas?

