



Módulo 04

Aritmética de Punto Fijo

(Pt. 1)



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



Copyright

- Copyright © **2011-2023** 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

- Clasificación de las operaciones
- Codificación decimal en binario (**BCD**)
- Representación **SM**
- Representación **RC**
- Representación **DRC**
- Operaciones de suma y de resta
- Detección de overflow
- Otras codificaciones



Tipos de operaciones

- Las operaciones aritméticas se clasifican en tres grandes categorías:
 - Operaciones aritméticas estándares
 - Funciones aritméticas elementales
 - Operaciones pseudo-aritméticas
- A su vez, se dispone esencialmente de dos modos de operación:
 - Operación en **punto fijo**
 - Operación en **punto flotante**



Tipos de operaciones

● Operaciones aritméticas estándares:

- Esta categoría incluye las cuatro funciones aritméticas primitivas: suma, resta, multiplicación y división, tanto en punto fijo como en punto flotante
- Toda otra función matemática podrá ser expresada como una composición de estas cuatro operaciones



Tipos de operaciones

● Funciones aritméticas elementales:

- ➔ Esta categoría incluye aquellas operaciones usadas frecuentemente en cálculos matemáticos, tales como exponencial, raíz cuadrada, funciones trigonométricas, hiperbólicas, etc.
- ➔ No todas las computadoras implementan estas funciones en hardware, en general se suelen implementar en firmware (microcódigo) o bien directamente como software en una librería.



Tipos de operaciones

- Operaciones pseudo-aritméticas:

- Esta categoría incluye operaciones que requieren un cierto grado de cálculo aritmético, pero están relacionadas con la ejecución de un programa

- Consta de dos subcategorías:

- Aritmética de direccionamiento: operaciones relacionadas al cómputo de la dirección efectiva en memoria de los datos

- Aritmética de edición de datos: operaciones lógicas y de transformación de datos tales como, load/store, empaquetado/desempaquetado, etc.



Modos de operación

- Operación en punto fijo:

- Este modo de operación es usado en problemas comerciales o cálculos estadísticos y se caracteriza por tener el punto decimal en una posición prefijada

- Consta de dos subcategorías:

- Operación entera: los resultado se alinean en el extremo derecho, como si el punto decimal ocupara esa posición

- Operación fraccionaria: los resultados se alinean en el extremo izquierdo, como si el punto decimal ocupara esa posición



Modos de operación

● Operación en punto flotante:

- Este modo de operación es usado en problemas de tipo ingenieril o científico, donde frecuentemente se requiere escalamiento para manejar tanto magnitudes muy grandes como muy pequeñas

● Consta de dos subcategorías:

- Operación normalizada: el resultado de toda operación es normalizado antes de ser retornado
- Operación no normalizada: el resultado de toda operación es retornado tal cual fuera obtenido



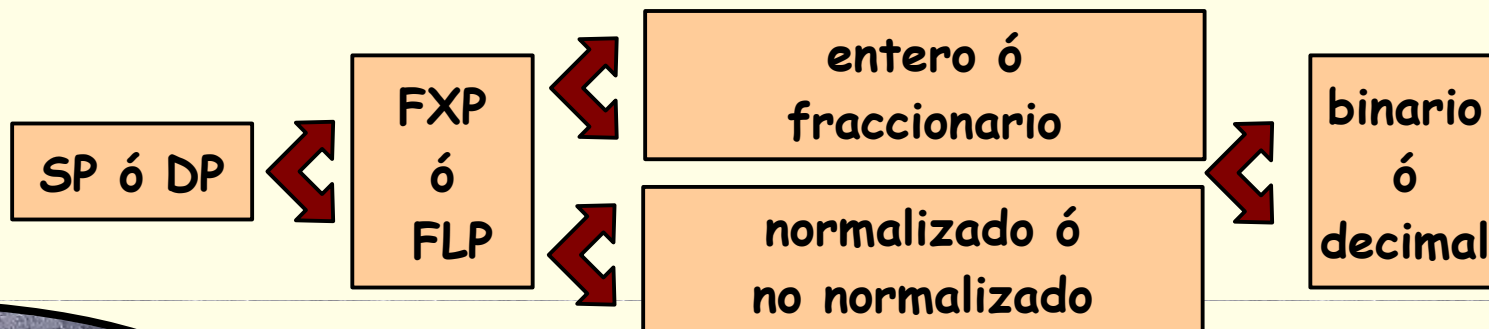
Precisión de las operaciones

- Las instrucciones aritméticas también pueden ser clasificadas de acuerdo a su **precisión**:
 - ➔ Precisión simple (SP): se refiere a las operaciones definidas sobre operandos de tamaño estándar, esto es, con operando de longitud igual a una palabra
 - ➔ Precisión doble (DP): se refiere a las operaciones definidas sobre operandos de tamaño doble, esto es, con operandos de longitud igual a dos palabras
 - ➔ Triple, quádruple y las restantes precisiones pueden definirse de manera análoga



Binario vs. decimal

- Algunas arquitecturas ofrecen la posibilidad de operar directamente sobre la base decimal
 - La conversión (operaciones de pack y de unpack), requiere de instrucciones para manejar los datos directamente en formato decimal
 - Por caso, la instrucción **ADD** puede hacer referencia a **16** operaciones aritméticas distintas, a saber:



Codificación de dígitos

- Para codificar cada dígito decimal es necesario hacer uso de **k** bits, donde:

$$k = \lceil \log_2 10 \rceil = 4$$

- Es posible postular múltiples codificaciones para cada dígito decimal haciendo uso de esos bits. Por ejemplo:

0		0000	0000	0011	0000
1		0001	0001	0100	0001
2		0010	0011	0101	0011
3		0011	0111	0110	0101



Código BCD-2421

● El código **BCD-2421** denota en su nombre el peso asignado a cada posición en la codificación

- Se trata de un **sistema posicional**
- Presenta una **línea de simetría por autocomplementación**
- Esta característica facilita el cómputo de la diferencia entre magnitudes de igual signo y la suma de magnitudes de distinto signo

	2421
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111



Código BCD-8421

● El código **BCD-8421** codifica cada dígito decimal directamente en binario

→ También se lo conoce como código **BCD natural**

→ Se trata de un **sistema posicional**

→ El peso de cada posición coincide con las sucesivas **potencias de dos**

→ No es posible encontrar ninguna línea de simetría que resulte de utilidad

	8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001



Código BCD exceso-3

El código **BCD exceso-3** también codifica cada dígito decimal en binario, si bien les suma previamente un exceso

- Producto del exceso, **deja de ser un sistema posicional**
- Como se puede apreciar, resulta **simétrico por autocomplementación**
- En esta codificación **el acarreo binario coincide con el acarreo decimal**

0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100



Código Gray

● El código **Gray** es simétrico por construcción

→ También se lo conoce como código **progresivo cíclico**

→ Se trata de una codificación **no posicional**

→ Como se puede observar **presenta múltiples líneas de simetría**

→ Nótese que entre la codificación de un dígito y el siguiente **se modifica a lo sumo un bit**

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101



Aritmética de punto fijo

- Recordemos que elegida una base r y una precisión p (cantidad de dígitos), el aporte de cada dígito al valor denotado depende exclusivamente de su posición
 - La idea es que los primeros n dígitos denoten la parte entera y los restante k dígitos la parte fraccionaria
 - La elección de n y de k tiene como efecto colateral fijar la posición del punto decimal, razón por la cual hablamos de aritmética de **punto fijo (FXP)**
 - Usualmente, $n = 0$ o bien $k = 0$



Aritmética de punto fijo

- Un número signado representará una magnitud positiva o negativa, pero no ambas
 - Usualmente se reserva el dígito de más a la izquierda (esto es, el más significativo), para **denotar el signo**
 - De los **r** dígitos válidos sólo se han de utilizar dos para codificar el signo
 - Por caso, sea $X = (d_{n-1}, \dots, d_1, d_0, d_{-1}, \dots, d_{-k})_r$ un número representado en una base **r**, entonces:

$$d_{n-1} = \begin{cases} 0 & \text{cuando } X \geq 0 \\ r-1 & \text{cuando } X < 0 \end{cases}$$



Aritmética de punto fijo

- Las posiciones del punto decimal y del signo dependerán de **n** y de **k**
 - Cuando **k = 0**, el signo se ubica en el extremo izquierdo y el punto decimal en el extremo derecho
 - Cuando **n = 0**, el signo se ubica en el extremo izquierdo y el punto decimal inmediatamente a su derecha (justo antes del primer dígito fraccionario)
 - La conversión entre estas dos variantes es sencilla, basta con multiplicar o dividir por una potencia de **r**



Aritmética de punto fijo

- La representación interna del punto decimal **es implícita**, no requiere reservar espacio de almacenamiento
- Para representar un **número positivo**, todos los dígitos salvo el de signo codifican su magnitud
 - ➔ Todas las representaciones que estudiaremos coinciden al codificar número positivos
- En cambio, para representar a un **número negativo** aparecen múltiples alternativas



Signo-magnitud (SM)

- Esta representación es análoga a la utilizada cotidianamente, donde la ausencia de signo representa un número positivo, y la presencia del signo – representa a un número negativo
- En la codificación signo-magnitud, el dígito de signo toma el valor **0** en los positivos y **r-1** en los negativos
- Nótese que **+0** y **-0** denotan al mismo valor, por lo que **la representación del cero no es única**



Signo-magnitud (SM)

- Sea \bar{X} el complemento de un dado número X
- En esta representación, \bar{X} y X difieren sólo en el dígito que codifica al signo
- Comprobemos esta situación con un ejemplo para una base $r = 2$ y con una precisión $n = 8$:

$$\begin{array}{ll} (00101011)_2 = +(43)_{10} & (10101011)_2 = -(43)_{10} \\ (01111111)_2 = +(127)_{10} & (11111111)_2 = -(127)_{10} \\ (00000000)_2 = +(0)_{10} & (10000000)_2 = -(0)_{10} \end{array}$$



Signo-magnitud (SM)

- ¿Qué rango de representación brinda usar SM con n dígitos en una base r ?
 - El menor número posible es $(r-1 \ r-1 \ r-1 \dots r-1)_r$
 - El mayor número posible es $(0 \ r-1 \ r-1 \dots r-1)_r$
 - Es decir, el rango de representación para signo-magnitud es $[-(r^{n-1} - 1), r^{n-1} - 1]$
- Por caso, para una base $r = 2$, con $n = 8$ dígitos binarios, el rango se calcula como:
 - $[-(2^7 - 1), 2^7 - 1] = [-127, 127]$



Complemento a la base (RC)

- Las computadoras suelen hacer uso de algún **esquema de complementación** para representar a los números negativos
 - ➔ Esto simplifica las operaciones suma de números de distintos signo, o lo que es lo mismo, de resta de números de igual signo
- Imaginemos el odómetro de una moto que queremos vender. Le pedimos a un mecánico amigo que le “rebobine” el odómetro, pero él se descuida y se pasa de largo... ¿qué sucede?



Complemento a la base (RC)

- Supongamos que el odómetro tiene sólo tres dígitos (a manera de simplificación)
- ¿Qué sucede al tratar de retroceder **5** Km estando inicialmente en el Km **003**?

003 → **002** → **001** → **000** → **999** → **998**

- En algún sentido, **998** tiene que representar al número **-2**
- Nótese que al sumar un valor positivo con su complemento negativo, se obtiene siempre el mismo resultado... ¿cuál es ese valor? ¿y al tener **n** dígitos?



Complemento a la base (RC)

- El resultado anterior permite determinar en general el valor de \bar{X} para cualquier X
 - Como comprobamos, $X + \bar{X} = r^n$, por lo que despejando \bar{X} nos queda $\bar{X} = r^n - |X|$
 - No es conveniente tener que computar una resta toda vez que se desee saber qué valor representa un cierto número negativo en complemento a la base
 - Por suerte, como $r^n = (r^n - 1) + 1$, se puede expresar a \bar{X} como $\bar{X} = ((r^n - 1) - |X|) + 1$
 - Obsérvese que $r^n - 1$ es un número especial, está compuesto de n dígitos iguales, de valor $r-1$



Complemento a la base (RC)

- En síntesis, el mecanismo simplificado para expresar un cierto número negativo en complemento a la base consiste en:
 - Primero expresar el valor absoluto del número en cuestión en el sistema complemento a la base (que por tratarse de un número positivo coincide con su representación en **SM**)
 - Complementar cada dígito del número, esto es, reemplazar cada dígito d_i por el valor $(r-1) - d_i$ (nótese que el signo también es complementado)
 - Finalmente, incrementar en **1** el valor obtenido



Complemento a la base (RC)

- Para $r = 2$ y $n = 8$ se desea saber qué valores codifican las siguientes cadenas de bits:

→ Cuando el bit de signo es **0**, la cosa es fácil:

$$\begin{array}{ll} (00001111)_2 = +(15)_{10} & (01001101)_2 = +(77)_{10} \\ (01111111)_2 = +(127)_{10} & (00000000)_2 = +(0)_{10} \end{array}$$

→ En cambio, si el bit de signo es **1**, se trata de un número negativo el cual se debe complementar:

$$\begin{array}{ll} (10001111)_2 = & (10000000)_2 = \\ -[(01110000)_2 + 1] = & -[(01111111)_2 + 1] = \\ - (113)_{10} & - (128)_{10} \end{array}$$



Complemento a la base (RC)

- ¿Qué rango de representación brinda usar complemento a la base r con n dígitos?
 - El menor número posible es $(r-1 \ 0 \ 0 \dots 0)_r$
 - El mayor número posible es $(0 \ r-1 \ r-1 \dots r-1)_r$
 - Es decir, el rango de representación en complemento a la base es $[-r^{n-1}, r^{n-1} - 1]$
- Por caso, para una base $r = 2$, con $n = 16$ dígitos binarios, el rango se calcula como:
 - $[-2^{15}, 2^{15} - 1] = [-32768, 32767]$



Complemento a la base disminuida (DRC)

- Recordemos que al representar un número negativo en complemento a la base debemos incrementar en **1** luego de complementar
 - Esta operación puede ser costosa en tiempo de ejecución, sobre todo **si existen múltiples acarreo**
 - Una posibilidad para evitar esta operación consiste en representar los números negativos haciendo uso del **complemento a la base disminuida**
 - Por caso, para un cierto número positivo **X**, su complemento a la base disminuida \bar{X} se calcula directamente como $\bar{X} = (r^n - 1) - |X|$



Complemento a la base disminuida (DRC)

- Volviendo al ejemplo del odómetro, el valor representado se debe retrasar una unidad (ya que falta tener en cuenta el incremento final)

0003 → **0002** → **0001** → **0000** → **9999** → **9998** → **9997**
+3 → **+2** → **+1** → **+0** → **-0** → **-1** → **-2**

- Como se puede apreciar existen nuevamente **dos representaciones para el cero**:

$$\rightarrow +0 = (0 \ 0 \ \dots \ 0 \ 0)_r$$

$$\rightarrow -0 = (r-1 \ r-1 \ \dots \ r-1 \ r-1)_r$$



Complemento a la base disminuida (DRC)

- El mecanismo simplificado para expresar un cierto número negativo en complemento a la base disminuida ahora consiste en:
 - Primero expresar el valor absoluto del número en cuestión en el sistema complemento a la base disminuida (que por tratarse de un número positivo coincidirá con su representación en signo-magnitud y en complemento a la base)
 - Finalmente, complementar cada dígito del número, esto es, reemplazar cada dígito d_i por el valor $(r-1) - d_i$ (el signo también es complementado)



Complemento a la base disminuida (DRC)

• Para la base $r = 2$ se desea saber qué números codifican las siguientes cadenas de bits:

→ Cuando el bit de signo es **0**, la cosa sigue siendo fácil:

$$(00001111)_2 = +(15)_{10} \quad (01001101)_2 = +(77)_{10}$$

$$(01111111)_2 = +(127)_{10} \quad (00000000)_2 = +(0)_{10}$$

→ En cambio, si el bit de signo es **1**, se trata de un número negativo el cual se debe complementar:

$$(10001111)_2 = \quad (10000000)_2 =$$

$$-(01110000)_2 = \quad -(01111111)_2 =$$

$$-(112)_{10} \quad -(127)_{10}$$



Complemento a la base disminuida (DRC)

- ¿Qué rango brinda usar complemento a la base disminuida con **n** dígitos para una base **r**?
 - El menor número es $(r-1 \ 0 \ 0 \dots 0)_r$
 - El mayor número es $(0 \ r-1 \ r-1 \dots r-1)_r$
 - Es decir, el rango de representación en complemento a la base disminuida es $[-(r^{n-1} - 1), r^{n-1} - 1]$
- Por caso, para una base **r = 2**, con **n = 16** dígitos binarios, el rango se calcula como:
 - $[-(2^{15} - 1), 2^{15} - 1] = [-32767, 32767]$



Ejemplo comparativo

- A manera de comparación, contrastemos la representación del número $(547)_{10}$ usando $n = 16$ dígitos y una base $r = 2$

→ Tener en cuenta que $(547)_{10} = (1000100011)_2$

signo-magnitud	0000001000100011	
complemento a 1	0000001000100011	← $+(547)_{10}$
complemento a 2	0000001000100011	
	1000001000100011	signo-magnitud
$-(547)_{10}$ →	1111110111011100	complemento a 1
	1111110111011101	complemento a 2



Análisis

- En la elección de un sistema de representación de números signados se deben tener en cuenta diversos aspectos:
 - ¿Qué tan sencillo resulta **detectar el signo**?
 - ¿Resultan **equivalentes** los rangos de representación para positivos y negativos?
 - ¿El cero tiene una **representación unívoca**?
 - ¿Qué tan **eficiente** resulta la implementación de las operaciones básicas y de la operación de complementación?



Análisis

- Analicemos los tres sistemas a la luz de los criterios recién introducidos:
 - La detección del signo es equivalente en los tres casos, consiste en inspeccionar el primer dígito
 - En relación a la simetría del rango, signo-magnitud y complemento a la base disminuida resultan simétricos, mientras que complemento a la base no
 - El cero tiene representación doble en signo-magnitud y en complemento a la base disminuida, pero en complemento a la base no
 - Resta analizar la eficacia de las operaciones



¿Preguntas?

