



Módulo 02

Sistemas de Representación



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

- Álgebra computacional
- Sistemas de representación
- Conversión entre bases:
 - De números enteros
 - De números fraccionarios



Aritmética en computadoras

- Las computadoras se las usa exclusivamente para **computar**
- Una de las acepciones de computar vimos que es precisamente **calcular**
- Un prerequisite para calcular es contar con un sistema de representación y un conjunto de operaciones definidas sobre ese sistema
- Es decir, necesitamos contar con una **aritmética**



Aritmética en computadoras

- Sea \mathbb{R} el conjunto de los números reales. La aritmética real se define como un mapeo algebraico f :

$$f: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

- Naturalmente, los ejemplos usuales para f son las operaciones de $+$, $-$, \times y $/$ sobre reales
- Nótese que esta álgebra no es apropiada para las computadoras, puesto que éstas sólo pueden operar con **magnitudes finitas**



Algebra para computadoras

- Sea \mathbb{M} el conjunto de números representables en una cierta computadora
 - Obsérvese que \mathbb{M} es un **subconjunto estricto** de \mathbb{R} , puesto que su cardinalidad es finita
- Finalmente, es posible definir una aritmética adecuada para computadoras sobre \mathbb{M} :

$$g: \mathbb{M} \times \mathbb{M} \rightarrow \mathbb{M}$$

- Esta aritmética se denomina **aritmética real aproximada**



Mapeo funcional

- La función máquina **g** se relaciona con la real **f** a través de la siguiente **función compuesta**:

$$g: \rho \circ h$$

- Donde el mapeo **h** se define como:

$$h = f|_{M \times M} \rightarrow \mathbb{R}$$

- Es decir, la función real **f** restringida al dominio máquina $M \times M$



Esquema de redondeo

- Por último, la función ρ debe tener como dominio e imagen los siguientes conjuntos:

$$\rho: \mathbb{R} \rightarrow M$$

- Esta función denota al **esquema de redondeo**
 - El esquema de redondeo tiene por objeto decidir cuál es el número máquina que corresponde asociar a cada número real
 - Necesariamente infinitos números reales se asociarán a uno o más números máquina (teorema de Dirichlet)



Sistemas de representación

- La definición precisa de estas funciones y la implementación de las operaciones aritméticas dependen de la **representación interna** que se adopte para los números
- La elección del sistema de representación afecta tanto al diseño del **hardware** como del **software**
- El diseñador tiene que tener en cuenta los requerimientos de desempeño, de precisión y de capacidad de representación



Sistemas de representación

- En función del objetivo que se priorice, se han ensayado (y se siguen ensayando) distintas alternativas:
 - Sistemas numéricos con base ←
 - Sistemas numéricos de dígito signado
 - Sistemas numéricos residuo
 - Sistemas numéricos racionales
 - Sistemas numéricos logarítmicos



Sistemas con base

- Este es el sistema tradicional al que estamos acostumbrados y es el que analizaremos
- Los números se representan fijando una **base** y haciendo uso de un **conjunto de dígitos**
 - Por caso, para una base **b** se utilizan los **b** dígitos comprendidos en el rango **[0, 1, ..., b-1]**
 - El aporte de cada dígito al valor representado tiene en cuenta la **posición** del mismo
 - Cada número tiene exactamente **una única representación** dentro del sistema



Representación de punto fijo

- En este sistema fijada una base y un conjunto de dígitos, el aporte de cada cada dígito al valor representado depende de su posición
 - La idea es que los primeros **n** dígitos denoten la parte entera y los restante **k** la parte fraccionaria
 - La elección de **n** y de **k**, decisión que toma el diseñador del hardware, fija la posición del punto decimal.
 - Por esta razón, a este tipo de representación se lo denomina de **punto fijo (FXP)**



Representación de punto fijo

- La naturaleza posicional del sistema se observa al considerar el valor representado por una determinada tupla de dígitos:
 - Sea $X = (d_{n-1}, \dots, d_1, d_0, d_{-1}, \dots, d_{-k})_r$ un número representado el sistema con base r .
 - Como se trata de un sistema posicional, el valor representado por el número X se calcula como el producto interno $X \times W$, donde W denota al conjunto de pesos asociados a cada posición, esto es:

$$W = (r^{n-1}, \dots, r^1, r^0, r^{-1}, \dots, r^{-k})$$



Representación de punto fijo

- Por caso, para el número **273** en base **10** se verifica que:

$$\begin{aligned}(273)_{10} &= 2 \times 10^2 + 7 \times 10^1 + 3 \times 10^0 \\ &= 200 + 70 + 3 \\ &= 273\end{aligned}$$

- Obsérvese que a mayor **r**, harán falta más bits para codificar cada uno de los dígitos
 - Hacen falta al menos **s** bits para codificar cada dígito de la base **r**, donde **s = $\log_2(r)$**



Elección de la base

- Para un dado sistema con base r , el disponer de n dígitos implica lo siguiente:
 - La **precisión del sistema**, es decir, la cantidad de números representables se calcula como $N = r^n$
 - La **ineficacia en el uso del espacio**, es decir, la cantidad de símbolos distintos que se deben almacenar se cuantifica como $D = n \times r$
- Para determinar cuál es la mejor base se debe primero fijar la precisión para luego minimizar la ineficacia en el uso del espacio



Elección de la base

• En otras palabras, queremos determinar qué r minimiza a D para un N constante

→ Despejando a n en la ecuación $N = r^n$ nos queda que $n = \ln(N) / \ln(r)$

→ Usando el resultado anterior, queremos minimizar la expresión $D = n \times r = \ln(N) / \ln(r) \times r$

→ Derivando con respecto a r nos queda:

$$dD / dr = \ln(N) \times (\ln(r) - 1) / \ln(r)^2$$

→ Igualando a cero, la solución es $r = e$. Como la base tiene que ser entera, $r = 3$ sería la mejor opción



Como diría Tu-Sam...

A Cuál sería la derivada respecto a r de $\ln(n) / \ln(r) \times r$?

View other drafts

La derivada respecto a r de $\ln(n) / \ln(r) \times r$ es:

$$d / dr (\ln(n) / \ln(r) \times r) = (1 / \ln(r)) \times (1 / r) \times n$$

Otra forma de escribir la respuesta es:

$$\ln(n) / \ln(r)^2$$

Para derivar la expresión, podemos usar la regla de la cadena:

$$d / dx (u(v(x))) = u'(v(x)) \times v'(x)$$

En este caso, $u(x) = \ln(x)$ y $v(x) = \ln(r)$. Entonces:

$$\begin{aligned} d / dr (\ln(n) / \ln(r) \times r) &= d / dr (u(v(r))) \\ &= u'(v(r)) \times v'(r) \\ &= (1 / \ln(r)) \times (1 / r) \times n \end{aligned}$$

Google it



¡Zapatero a tus zapatos!



D[ln(n) * r/ln(r),r]

NATURAL LANGUAGE MATH INPUT

EXTENDED KEYBOARD EXAMPLES UPLOAD RANDOM

Derivative

Step-by-step solution

$$\frac{\partial}{\partial r} \left(\frac{\log(n) r}{\log(r)} \right) = \frac{\log(n) (\log(r) - 1)}{\log^2(r)}$$

log(x) is the natural logarithm



Elección de la base

- Hay cuatro sistemas en uso, los que corresponden a las bases **2**, **8**, **10** y **16**
 - La base **10** es obviamente la preferida por los humanos en cualquier actividad
 - La base **2** es la preferida por las computadoras, debido a que la electrónica digital biestable es mucho más sencilla, económica y confiable que la triestable
 - Las bases **8** y **16** se utilizan frecuentemente para comunicar número binarios compuestos de muchos dígitos de manera compacta



Conversión entre bases

- Es posible convertir un número de una base a otra mediante dos métodos:
 - Método de la **multiplicación**
 - Método de la **división**
- También se debe distinguir entre la conversión de **números enteros** o de **números fraccionarios**
- En otras palabras, debemos considerar cuatro métodos de conversión entre bases



Conversión de enteros

● Método de la multiplicación:

- Se opera usando la **aritmética destino** (base r_d)
- La idea es hacer uso de la naturaleza posicional del sistema, calculando el siguiente producto interior:

$$|X| = \sum_{i=0}^{n-1} (x_i \times r_o^i)$$

- Por ejemplo, para convertir **1672** de octal a decimal:

$$\begin{aligned} (1672)_8 &= 1 \times 8^3 + 6 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 \\ &= 512 + 384 + 56 + 2 \\ &= (954)_{10} \end{aligned}$$



Conversión de enteros

● Método de la división:

- Se opera usando la **aritmética origen** (base r_o)
- Una vez más, la idea es hacer uso de la naturaleza posicional del sistema teniendo en cuenta que los exponentes en esta ocasión son negativos
- Los dígitos se calculan sucesivamente dividiendo por la base destino hasta llegar a un cociente nulo

$$X = Q_0 \times r_d + X_0, \text{ con } X_0 = |X| \bmod r_d$$

$$\text{Luego, } Q_0 = Q_1 \times r_d + X_1, \text{ con } X_1 = |Q_0| \bmod r_d$$

y así sucesivamente hasta que $Q_i = 0$



Conversión entre enteros

- Consideremos el ejemplo recíproco al anterior, esto es, convertir **954** de decimal a octal

$$X_0 = 954 \bmod 8 = 2 \text{ y } Q_0 = (954 - 2) / 8 = 119$$

$$X_1 = 119 \bmod 8 = 7 \text{ y } Q_1 = (119 - 7) / 8 = 14$$

$$X_2 = 14 \bmod 8 = 6 \text{ y } Q_2 = (14 - 6) / 8 = 1$$

$$X_3 = 1 \bmod 8 = 1 \text{ y } Q_3 = (1 - 1) / 8 = \underline{0}$$

- El resultado es finalmente **(1672)₈**

→ Los dígitos van apareciendo en orden inverso



Conversión de fracciones

- Todo **número entero** representable con una cantidad finita de dígitos en una cierta base tendrá una representación con una cantidad también finita de dígitos en cualquier otra base
- En contraste, en los **números fraccionarios** puede que un número representado con una cantidad finita de dígitos en una base requiera una cantidad infinita de dígitos en otra
 - Siempre es posible calcular los primeros **k** dígitos como su **representación aproximada**



Conversión de fracciones

● Método de la multiplicación:

→ Se opera usando la **aritmética origen** (base r_o)

→ La idea es ir obteniendo los distintos dígitos X_i multiplicando la fracción por la base r_d

→ El primer dígito X_{-1} , se obtiene como: "parte entera de"

$$X_{-1} = \lfloor |X| \times r_d \rfloor$$

→ Luego, sea $\sim X_{-1}$ la parte fraccionaria de $|X| \times r_d$, los restantes dígitos se deben calcular como

$X_{-2} = \lfloor \sim X_{-1} \times r_d \rfloor$, y así sucesivamente



Conversión de fracciones

- Por ejemplo, para convertir 0.7632 de octal a decimal, con cuatro dígitos de precisión:
 - $(0.7632)_8 \times (12)_8 = (11.6004)_8$, por lo que $X_{-1} = 9$
 - $(0.6004)_8 \times (12)_8 = (7.405)_8$, por lo que $X_{-2} = 7$
 - $(0.405)_8 \times (12)_8 = (5.062)_8$, por lo que $X_{-3} = 5$
 - $(0.062)_8 \times (12)_8 = \underline{(0.764)_8}$, por lo que $X_{-4} = 0$
- Finalmente, $(0.7632)_8$ se corresponde con $(0.9750)_{10}$ al usar cuatro dígitos de precisión



Conversión de fracciones

● Método de la división:

- Se opera usando la **aritmética destino** (base r_d)
- Es análogo al método del producto para los enteros, sacamos provecho de la naturaleza posicional del sistema calculando el siguiente producto interior:

$$|X| = \sum_{i=1}^k (x_{-i} \times r_o^{-i})$$

- Por caso, para convertir **0.7632** de octal a decimal, se calcula lo siguiente operando en base **10**:

$$((((((2 / 8) + 3) / 8) + 6) / 8) + 7) / 8 = (0.9750)_{10}$$



Métodos preferidos

- Los humanos hemos de preferir naturalmente operar en base **10**, por lo que usaremos cualquier método que nos permita hacer uso de nuestra familiar base
- Las computadoras han de hacer algo análogo, prefiriendo toda vez que se pueda la base **2**, si bien también se intenta evitar hacer un uso excesivo de la operación de división
 - La razón es que se trata de una operación costosa a nivel de tiempo de ejecución



¿Preguntas?

