UNIVERSIDAD DE SANTIAGO DE CHILE
DEPARTAMENTO DE INFORMÁTICA

Conversión Numérica y Precisión Punto Flotante

Anexo Lab 1

Sistemas Numéricos Posicionales

- ✓ Por convención, el sistema numérico comúnmente usado es el **sistema decimal**.

- ✓ El sistema numérico decimal es un sistema “posicional” debido a que el valor de un dígito depende de la posición en la cual se encuentra.

- ✓ En otras palabras, un número en el sistema decimal corresponde a un polinomio en base 10.

Sistemas Numéricos Posicionales

- Por ejemplo, el número 9742 se puede expresar por el polinomio:

$$9742 = 9 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$$

- Para generalizar:

$$D_1 D_2 D_3 \ldots D_M = D_1 \times B^{M-1} + D_2 \times B^{M-2} + D_3 \times B^{M-3} + \ldots + D_M \times B^0$$

- {D1, D2, D3, DM} se denominan **dígitos**. Estos constituyen los únicos símbolos representables. Si la base es B, existen B dígitos representables. En el caso de la base decimal, estos dígitos son: 0,1,..9.

Sistema Binario

- Si la base es B=2, el sistema numérico se denomina **binario**. El conjunto de dígitos representables es {0,1}. Este conjunto se denomina **bits**. Por ejemplo:

$$10011 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 19_{10}$$

- También se puede generalizar para números con punto decimal:

$$1.1101 = 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$

Rangos

- Considere un número decimal de N dígitos:
  - Valores disponibles: 10^n
  - Rango valores: 0, 10^n - 1
  - Ejemplo: Número de 3 dígitos. 10^3 = 1000 posible valores, en el rango 0, 999.

- Considere un número binario de N bits:
  - Valores disponibles: 2^n
  - Rango valores: 0, 2^n - 1
  - Ejemplo: Número de 3 dígitos. 2^3 = 8 posible valores, en el rango 0, 7 = 000 a 111.

Sistema Octal y Hexadecimal

- ✓ Si la base es B=8, el sistema numérico se denomina octal. El conjunto de dígitos representables es {0,1,2,..,7}. Por ejemplo:

$$1753.6_8 = 1\times 8^3 + 7\times 8^2 + 5\times 8^1 + 3\times 8^0 + 6\times 8^{-1} = 1003.75_{10}$$

- ✓ Si la base es B=16, el sistema numérico se denomina hexadecimal. El conjunto de dígitos representables es {0,1,2,..,9,A,B,..F}. Por ejemplo:

$$A67F9_{16} = 10\times 16^4 + 6\times 16^3 + 7\times 16^2 + 15\times 16^1 + 9\times 16^0 = 681977_{10}$$

Sistema Octal y Hexadecimal

- ✓ Ambos sistemas tienen gran importancia en arquitectura de computadores. Esto se debe a que permiten representar información binaria en forma compacta.

- ✓ En el lenguaje C se pueden representar constantes octales y hexadecimales. Por ejemplo:

    - ✓ const int i = **0**56 // prefijo **0** indica **octal**

    - ✓ const int i = **0**xA9 // prefijo **0x** indica **hex**

Conversión entre Bases

- ✓ Una de las primeras conversiones entre bases numéricas que estudiaremos es la conversión de decimal a binario.

- ✓ Para realizar esta conversión, se dividen sucesivamente los cuocientes por 2 y se registran los restos de la división.

- ✓ Por ejemplo, convertir el siguiente número a binario: 19

  - ✓ 19:2=9 :2=4 :2=2 :2=1 :2=0
  - ✓ 1 // 1// 0// 0// 1// <- Restos
  - ✓ Finalmente, el valor de 19 en base 10 es 10011 en base 2.

Conversión entre Bases

✓ Para simplificar la conversión, la división es mejor hacerla en forma tabular.

19	2
9	1
4	1
2	0
1	0
0	1

- El resultado se lee de abajo hacía arriba.

$$19_{10} = 10011_2$$

Conversión entre Bases

- ✓ Para convertir números fraccionarios se multiplica sucesivamente la parte fraccionaria por 2. La parte entera corresponde al número binario.

- ✓ Por ejemplo, convertir 0.753 en base decimal a binario.
  - 0.753*2 = **1**.506 la parte entera es 1 y se remueve.
  - 0.506*2 = **1**.012 la parte entera es 1 y se remueve.
  - 0.012*2 = **0**.024 la parte entera es 0.
  - 0.024*2 = **0**.048 la parte entera es 0.

- ✓ El resultado se aproxima a 0.1100 en base 2

Conversión entre Bases

- ✓ La forma tabular es también en este caso más fácil para convertir números fraccionarios. Por ejemplo, considerando el ejemplo anterior:

0.753	**2**
0.506	**1**
0.012	**1**
0.024	**0**
0.048	**0**

0.753	**2**
1.506	2
1.012	2
0.024	2
0.048	2

$$0.753_{10} = 0.1100_2$$

Conversión entre Bases

- Otro ejemplo: Convertir 23.4375 en base decimal a binario.

23	2
11	**1**
5	**1**
2	**1**
1	**0**
0	**1**

+

0.4375	2
0.875	2
1.750	2
1.500	2
1.000	2

$$23.4375_{10} = 10111.0111_2$$

Conversión entre Bases

- La división de números enteros o la multiplicación de números fraccionarios se puede generalizar para la conversión entre cualquier base. Hay que tener en cuenta que la aritmética debe corresponder a la base original.

- Por ejemplo: Convertir 478 en base decimal a base octal.

478	8
59	6
7	3
0	7

$$478_{10} = 736_{8}$$

Conversión entre Bases

- Considerando el ejemplo anterior, para convertir el número 736 en base octal a base decimal, simplemente habría que evaluar el polinomio correspondiente:

$$736_8 = 7 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 = 478_{10}$$

- Otro Ejemplo: Convertir 478 en base decimal a base hexadecimal:

478	16
29	14
1	13
0	1

$$478_{10} = 1DE_{16}$$

Conversión entre Bases

- ✓ Las bases numéricas que son potencias de dos tienen una interesante propiedad que permite una rápida conversión.

- ✓ Esto debido a que b = 8 = 2^3. También b = 16 = 2^4.

- ✓ En general, si b = 2^n, basta separar en grupos de n bits y convertir sólo el grupo.

- ✓ Ejemplo, convertir a hexadecimal y Octal:

1010	1101	0110	1011
A	D	6	B

1	010	110	101	101	011
1	2	6	5	5	3

Conversión entre Bases

- ✓ Esta propiedad justifica el amplio uso de números octales y hexadecimales como forma de compactar la representación de números binarios. Usaremos esta representación en los **lenguajes de máquina** y para expresar códigos.

Hexadecimal	Decimal	Binario
0	0	0000
1	1	0001
2	2	0010
…..	…..	…..
9	9	1001
A	10	1010
…..	……	……
E	14	1110
F	15	1111

Aritmética Computacional

- ✓ En la actualidad, los computadores son binarios.

- ✓ La información que utiliza un computador es almacenada en dispositivos de hardware llamados **registros**.

- ✓ El ancho del registro representa el número de bits que puede almacenar.

- ✓ Los procesadores actuales tienen registros de ancho de 32 y 64 bits.

1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1

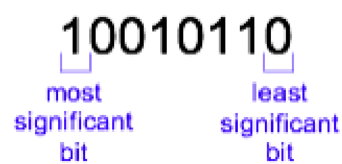
Registro de 16 bits

Aritmética Computacional

✓ Como ya se había mencionado, un grupo de 8 bits se denomina Byte.

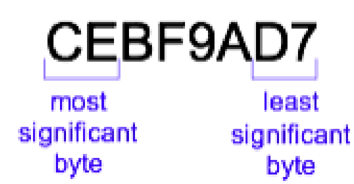
✓ Seguiremos la notación tradicional, es decir, b para bits y B para Byte. El registro anterior tiene 16b o 2B.

- Bits



- Bytes & Nibbles

- Bytes



Aritmética Computacional

- La información que puede manejar un computador está limitada por el tamaño de los registros. Con un registro de 16b, el mayor entero representable es:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

$$VMAX = 65535_{10}$$

Suma

- Decimal

$$\begin{array}{r l} \color{blue}{11}\;\; & \color{blue}{\leftarrow \text{carries}} \\ 3734 & \\ +\;\;5168 & \\ \hline 8902 & \end{array}$$

- Binary

$$\begin{array}{r l} \color{blue}{11}\;\; & \color{blue}{\leftarrow \text{carries}} \\ 1011 & \\ +\;\;0011 & \\ \hline 1110 & \end{array}$$

Suma

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
1 \\
1001 \\
+\ 0101 \\
\hline
1110
\end{array}
$$

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
111 \\
1011 \\
+\ 0110 \\
\hline
10001
\end{array}
$$

Overflow!

✓ Overflow: El resultado de una operación es más grande que la cantidad de bit´s disponibles para guardar el resultado

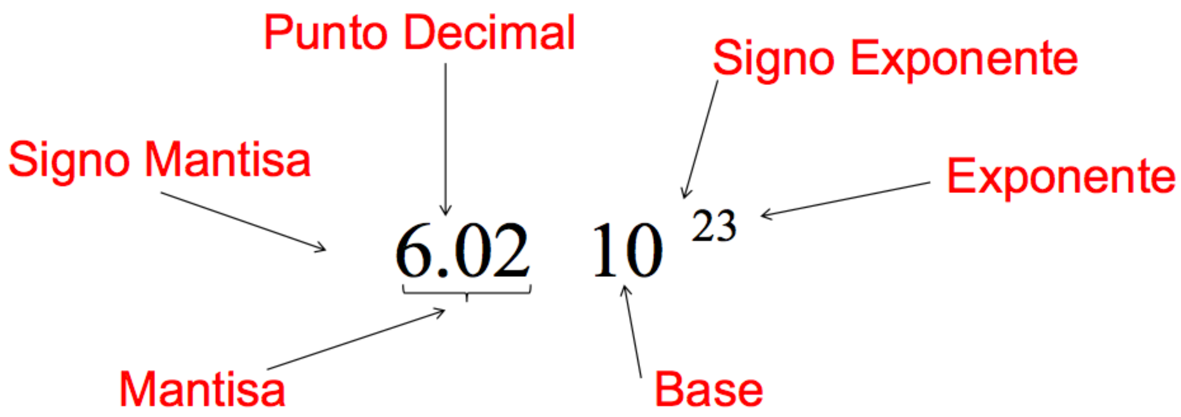
Precisión Punto Flotante

- ✓ Por los años 80, con la ayuda de varios ingenieros, entre ellos W. Kahan, se desarrolló un estándar para el sistema de punto flotante, el cual adoptó la IEEE, conocido como **IEEE 754**.
- ✓ Con la introducción de este sistema, se resolvieron las limitaciones que tiene el sistema de punto fijo (ubica siempre el punto en alguna posición a la derecha del dígito menos significativo.)
- ✓ El estándar cumple con 3 requisitos:
  - ✓ La representación del punto flotante debe ser consistente en todas las máquinas que lo adopten.
  - ✓ La aritmética de redondeo debe ser correcta.
  - ✓ El tratamiento de casos especiales debe ser consistente (overflow, división por cero, etc.)

Precisión Punto Flotante

- ✓ La notación punto flotante se asemeja a la notación científica o exponencial en la que representamos a los números.

- ✓ La idea principal de esta notación es facilitar la comparación entre los números y su ordenamiento.



Precisión Punto Flotante

- ✓ IEEE 754 especifica cuatro formatos para la representación de valor en punto flotante:
  - ✓ Precisión simple (32 bits).
  - ✓ Precisión doble (64 bits).
  - ✓ Precisión simple extendida (valores mayores o igual a 43 bits).
  - ✓ Precisión doble extendida (valores mayores o igual a 79 bits).

Precisión Punto Flotante

- ✓ Nos centraremos en la precisión simple 32 bits.

1 bit	8 bits	23 bits
S	E	M

- ✓ S: Signo (0: positivo, 1: negativo)
- ✓ E: Exponente sesgado
- ✓ M: Mantisa (magnitud del número normalizado)

Precisión Punto Flotante

- ✓ El exponente sesgado corresponde al valor:

$$E = e + (B^{n-1} - 1)$$

- ✓ Donde e es el exponente real y n el número de bits para representar el exponente (8 para p.s 32).

- ✓ Para precisión simple tenemos: $E = e + 127$

Precisión Punto Flotante

- La normalización de un número corresponde a escribirlo de la forma:

$$\pm 1.b_1 b_2 b_3 \ldots b_{23} \times 2^{\pm e}$$

- En los 23 bits de la mantisa, se almacenan los bits desde el b1 al b23 sin considerar el 1 que está a la izquierda de la coma, llamado bit oculto.

- La representación IEEE 754 en precisión simple corresponde a:

$$(-1)^S \times (1 + Mantisa) \times 2^{(E-127)}$$

Precisión Punto Flotante

- Por ejemplo: escribir -118,625 en precisión simple 32 bits.
- El primer paso, será convertir dicho número a base binaria.

118	2
59	0
29	1
14	1
7	0
3	1
2	1
1	1

$$118_{10} = 1110110_2$$

0.625	2
1.25	2
0.5	2
1.00	2

$$0.625_{10} = 0.101_2$$

$$118.625_{10} = 1110110.101_2$$

Precisión Punto Flotante

✓ Dicho resultado, lo podemos escribir como:

$$1.110110101 \times 2^{6}$$

✓ Es decir, hemos desplazado la coma 6 ubicaciones hacía la izquierda. Con esto podemos calcular el valor de E = 6 + 127 = 133

✓ Con esto ya podemos calcular todos los valores correspondientes:

- ✓ S: 1 (número negativo).
- ✓ E: 10000101 (133 en binario)
- ✓ M: 11011010100000000000000

S (1 bit)	E (8 bits)	M (23 bits)
1	10000101	110110101000…000

Precisión Punto Flotante

- Algunas casos particulares:
- Si E=0, M=0 y S=1, entonces el valor corresponde a -0.
- Si E=0, M=0 y S=0, entonces el valor corresponde a 0.

```
0 00000000 00000000000000000000000 = 0
1 00000000 00000000000000000000000 = -0
```

- Si E=255, M=0 y S=1, entonces el valor corresponde a -00.
- Si E=255, M=0 y S=0, entonces el valor corresponde a 00.

```
0 11111111 00000000000000000000000 = Infinito
1 11111111 00000000000000000000000 = -Infinito
```

Precisión Punto Flotante

- ✓ Algunas casos particulares:
- ✓ Si E=255, M no nulo y S=1 o S = 0, entonces el valor representable corresponde a NaN (Not a number).

```
0 11111111 00000100000000000000000 = NaN
1 11111111 00100010001001010101010 = NaN
```

- ✓ Otros casos:

```
0 10000000 00000000000000000000000 = +1 * 2**(128-127) * 1.0 = 2
0 10000001 10100000000000000000000 = +1 * 2**(129-127) * 1.101 = 6.5
1 10000001 10100000000000000000000 = -1 * 2**(129-127) * 1.101 = -6.5

0 00000001 00000000000000000000000 = +1 * 2**(1-127) * 1.0 = 2**(-126)
0 00000000 10000000000000000000000 = +1 * 2**(-126) * 0.1 = 2**(-127)
0 00000000 00000000000000000000001 = +1 * 2**(-126) *
                                     0.00000000000000000000001 =
                                     2**(-149)  (valor positivo más pequeño)
```

Precisión Punto Flotante

- ✓ Doble precisión (64 bits):

1 bit	11 bits	52 bits
S	E	M

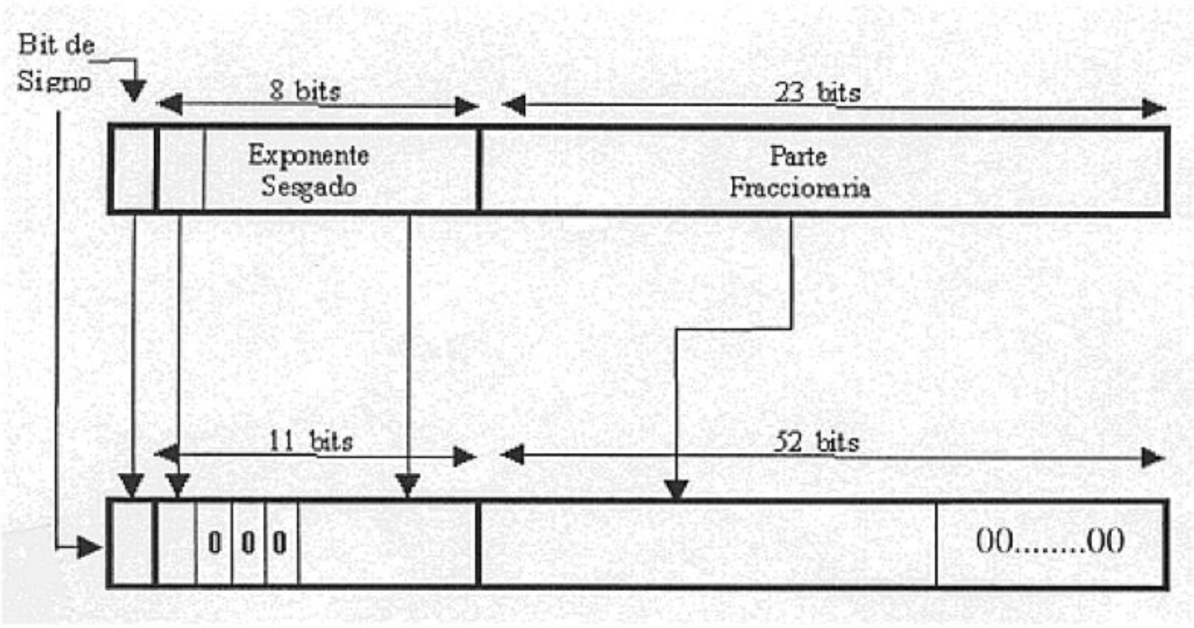
- ✓ S: Signo (0: positivo, 1: negativo)
- ✓ E: Exponente sesgado
- ✓ M: Mantisa (magnitud del número normalizado)

Precisión Punto Flotante

	Simple	Doble
L. De Palabra (bits)	32	64
L. De Exponente (bits)	8	11
Sesgo del Exponte	127	1023
Exponente Máximo	127	1023
Exponente Mínimo	-126	-1022
Rango de Números	10^{-38} – 10^{38}	10^{-308} – 10^{308}
L. Mantisa	23	52
Número de Exponentes	254	2046
Número de Mantisas	2^23	2^52

Precisión Punto Flotante

✓ Es posible realizar una conversión trivial para pasar de simple precisión a doble precisión y viceversa.



UNIVERSIDAD DE SANTIAGO DE CHILE
DEPARTAMENTO DE INFORMÁTICA

Conversión Numérica y Precisión Punto Flotante

Anexo Lab 1