Representación Computacional de Datos

Arquitectura del Computador LCC-FCEIA

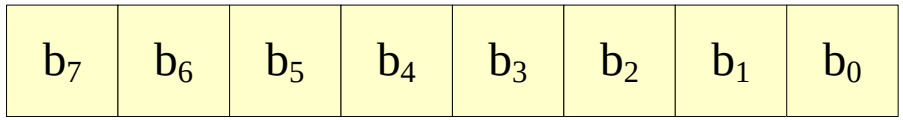
Agosto de 2021

Introducción

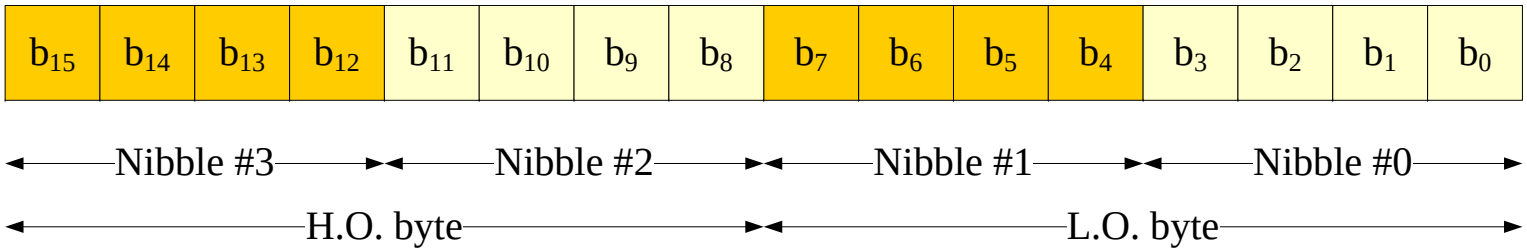
- La aritmética que utilizan las computadoras difiere de la que estamos acostumbrados a usar.
- La diferencia fundamental radica en que las computadoras solo pueden trabajar con números de precisión finita.
- Sistemas de numeración posicionales: decimal, binario, hexadecimal, octal...

Organización de los datos

- Bit: es la mínima unidad de información en un sistema binario.
- Nibble: es un conjunto de cuatro bits.
- Byte: es un conjunto de ocho bits, donde b_0 es el LSB y b_7 es el MSB.



- Palabra: es un conjunto de n bits que son manejados como un conjunto por la máquina.
  Ejemplo: palabra de 16 bits



Con 16 bits se pueden representar 2^{16} valores diferentes.

Sistemas de numeración posicionales

- Es un sistema donde cada dígito tiene un peso diferente según la posición que ocupe.
- Sistema decimal: base 10.
  Ejemplo:

  $$123.59 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 5 \times 10^{-1} + 9 \times 10^{-2}$$

- En general, la representación decimal

  $$(-1)^s(a_n a_{n-1} \ldots a_1 a_0 . a_{-1} a_{-2} \ldots)$$

  corresponde al número

  $$(-1)^s(a_n 10^n + a_{n-1} 10^{n-1} + \ldots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + \ldots),$$

  donde $s = 0$ si el número es positivo y $s = 1$ si es negativo.

Sistemas de numeración posicionales

- Cualquier número natural $\beta \geq 2$ puede ser utilizado como base:

  $$(-1)^s(a_n\beta^n + a_{n-1}\beta^{n-1} + \ldots + a_1\beta^1 + a_0\beta^0 + a_{-1}\beta^{-1} + a_{-2}\beta^{-2} + \ldots),$$

  donde los coeficientes a_i son los dígitos en el sistema con base β: $0 \leq a_i \leq \beta - 1$.

- Los coeficientes a_i con $i \geq 0$ se consideran como los dígitos de la parte entera, en tanto que los a_i con $i < 0$, son los dígitos de la parte fraccionaria.

- Si utilizamos un punto para separar la parte entera de la parte fraccionaria:

  $$N = (-1)^s(a_n a_{n-1} \ldots a_1 a_0 . a_{-1} a_{-2} \ldots)_\beta.$$

Sistema binario

- Las computadoras actuales utilizan internamente el sistema binario.
- Ventaja: reglas simples para realizar operaciones.
- Base 2
- Solo dos dígitos: 0 y 1, llamados *bits*.
- El 1 y el 0 en notación binaria tienen el mismo significado que en notación decimal:

$$0_2 = 0_{10}, \qquad 1_2 = 1_{10}.$$

- Equivalencia entre sistemas de representación: $(1101.01)_2 = (13.25)_{10}$, puesto que

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 13.25$$

- El número 6 se puede representar como $(110)_2$, como $(0110)_2$, o como $(0000\ 0110)_2$.

Conversión entre sistemas: binario a decimal

La conversión de binario a decimal es directa empleando la definición de sistema de numeración posicional ya vista:

Por ejemplo:

$$(0110.1101)_2 = 1\times2^2+1\times2^1+1\times2^{-1}+1\times2^{-2}+1\times2^{-4} = (6.8125)_{10}$$

Conversión entre sistemas: decimal a binario

- El primer paso es separar la parte *entera* de la parte *fraccionaria*.
- Para convertir la parte entera, realizar sucesivas divisiones por dos hasta llegar a cero e ir registrando los restos que resultan ser los bits del número en binario. El primer resto obtenido es b_0, el segundo es b_1 y así sucesivamente.
- Para convertir la parte fraccionaria se realizan multiplicaciones sucesivas por dos de las partes fraccionarias (sin el entero) y se van registrando los dígitos enteros obtenidos. El primer dígito obtenido es b_{-1} y así sucesivamente.
- Finalmente, se pueden juntar ambos resultados.

Conversión entre sistemas: decimal a binario

Ejemplo: convertir el número $(149.56)_{10}$ a binario.

Primero convertimos la parte entera:

$$
\begin{array}{ll}
149/2{=}74 & \text{Resto}{=}1 \rightarrow b_0{=}1 \\
74/2{=}37 & \text{Resto}{=}0 \rightarrow b_1{=}0 \\
37/2{=}18 & \text{Resto}{=}1 \rightarrow b_2{=}1 \\
18/2{=}9 & \text{Resto}{=}0 \rightarrow b_3{=}0 \\
9/2{=}4 & \text{Resto}{=}1 \rightarrow b_4{=}1 \\
4/2{=}2 & \text{Resto}{=}0 \rightarrow b_5{=}0 \\
2/2{=}1 & \text{Resto}{=}0 \rightarrow b_6{=}0 \\
1/2{=}0 & \text{Resto}{=}1 \rightarrow b_7{=}1
\end{array}
$$

Notar que el último resto se considera 1 aunque en realidad es menor que 1. Entonces, $(149)_{10} = (1001\ 0101)_2$.

Conversión entre sistemas: decimal a binario

Luego procedemos con la parte fraccionaria:

$$
\begin{aligned}
0.56 \times 2 &= \mathbf{1}.12 \quad \rightarrow b_{-1}=1 \\
0.12 \times 2 &= \mathbf{0}.24 \quad \rightarrow b_{-2}=0 \\
0.24 \times 2 &= \mathbf{0}.48 \quad \rightarrow b_{-3}=0 \\
0.48 \times 2 &= \mathbf{0}.96 \quad \rightarrow b_{-4}=0 \\
0.96 \times 2 &= \mathbf{1}.92 \quad \rightarrow b_{-5}=1 \\
0.92 \times 2 &= \mathbf{1}.84 \quad \rightarrow b_{-6}=1 \\
0.84 \times 2 &= \mathbf{1}.68 \quad \rightarrow b_{-7}=1 \\
0.68 \times 2 &= \mathbf{1}.36 \quad \rightarrow b_{-8}=1
\end{aligned}
$$

De esta manera, $(0.56)_{10} \approx (1000\ 11111)_2$ empleando 8 bits.
Por lo tanto, uniendo ambos resultados obtenemos

$$(149.56)_{10} \approx (1001\ 0101.1000\ 1111)_2$$

Notar que no tiene una representación exacta y por lo tanto se comete un error de representación que ya analizaremos.

Operaciones elementales en sistema binario

Suma:

En binario, la tabla de adición toma la siguiente forma:

$$
\begin{aligned}
0 + 0 &= 0 \\
0 + 1 &= 1 \\
1 + 0 &= 1 \\
1 + 1 &= 10
\end{aligned}
$$

Ejemplo, sumar 0101_2 y 0011_2:

$$
\begin{array}{cccccc}
 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \\
 & 0 & 1 & 0 & 1 \\
+ & 0 & 0 & 1 & 1 \\
\hline
 & 1 & 0 & 0 & 0
\end{array}
$$

En sistema decimal sería $5 + 3 = 8$.

Operaciones elementales en sistema binario

Resta:
La tabla de resta toma la siguiente forma:

$$\begin{aligned} 0 - 0 &= 0 \\ 1 - 0 &= 1 \\ 1 - 1 &= 0 \\ 0 - 1 &= \mathbf{1} \end{aligned}$$

Ejemplo, restar 0010_2 a 1001_2:

$$\begin{array}{cccccc} & \mathbf{1} & \mathbf{1} & & \\ & 1 & 0 & 0 & 1 \\ - & 0 & 0 & 1 & 0 \\ \hline & 0 & 1 & 1 & 1 \end{array}$$

En sistema decimal sería $9 - 2 = 7$.

Números con signo

- Hasta ahora hemos visto números sin signos, ¡pero necesitamos también números negativos!
- Convenciones para representar números con signo:
  - **Magnitud y signo**: Se emplea el bit más significativo para representar el signo mientras que el resto de los bits indican la magnitud (valor absoluto) del número a representar. Por convención, el bit más significativo en uno indica un número negativo:

$$N = (-1)^{b_{n-1}} \times (b_{n-2} \times 2^{n-2} + \ldots + b_1 \times 2^1 + b_0 \times 2^0)$$

Por ejemplo:

$$(-28)_{10} = (\mathbf{1}001\ 1100)_2.$$

  - **Complemento a la base**
  - **Complemento a la base menos uno**

Complemento a la base

Definición

Sea un número N representado con n dígitos, el complemento a la base β de N se define como

$$C_\beta^N = \begin{cases} N & \text{si } N \geq 0, \\ \beta^n - |N| & \text{si } N < 0. \end{cases}$$

Esto se cumple para todo N incluso con fracción decimal. El único caso especial a considerar es cuando la parte entera es cero. Esto se interpreta como que $n = 0$.

Complemento a la base

Ejemplos en base 10:

- Complemento a 10 de $(-987)_{10} \Longrightarrow |N| = 987$ y $n = 3$, entonces $C_{10}^{N} = 10^3 - 987 = 1000 - 987 = (13)_{10}$.
- Complemento a 10 de $(-0.125)_{10} \Longrightarrow |N| = 0.125$ y $n = 0$, entonces $C_{10}^{N} = 10^0 - 0.125 = 1 - 0.125 = (0.875)_{10}$.
- Complemento a 10 de $(-987.125)_{10} \Longrightarrow |N| = 987.125$ y $n = 3$, por lo tanto $C_{10}^{N} = 10^3 - 987.125 = 1000 - 987.125 = (12.875)_{10}$

Es importante notar que NO es lo mismo calcular el complemento de la parte entera y de la fracción decimal por separado y juntar los resultados.

Complemento a dos

El complemento a dos es un sistema posicional de representación para números enteros en el que los positivos se representan en binario natural y los negativos se representan restando su magnitud de la cantidad 2^n, siendo n el número de dígitos:

$$C_2^N = \begin{cases} N & \text{si } N \geq 0, \\ 2^n - |N| & \text{si } N < 0. \end{cases}$$

Ejemplos:

- Para el complemento a 2 de $N = (1010\ 1100)_2$ tenemos que $n = 8$, entonces:

$$C_2^N = (1\ 0000\ 0000)_2 - (1010\ 1100)_2 = (0101\ 0100)_2$$

- También podemos hacer el cálculo pasando al sistema decimal como paso intermedio:

$$C_2^N = (2^8)_{10} - (172)_{10} = (256)_{10} - (172)_{10} = (84)_{10} = (0101\,0100)_2$$

Complemento a dos

Método alternativo

Para calcular el complemento a 2 de un número binario sólo basta con revisar todos los dígitos desde el menos significativo hacia el más significativo y mientras se consiga un cero dejarlo igual. Al conseguir el primer número 1, dejarlo igual para luego cambiar el resto de ellos hasta llegar al más significativo.

Así podemos decir rápidamente que el complemento a 2 de $(1010\ 0000)_2$ es $(0110\ 0000)_2$, que el complemento a 2 de $(111)_2$ es $(001)_2$, etc.

Método alternativo 2

Tener en cuenta que $C_2^N = C_1^N + 1$, donde $C_1^N + 1$ es el complemento a la base menos uno.

Complemento a dos

- El rango de valores decimales para n bits será:

$$-2^{n-1} \leq \ Rango \leq \ 2^{n-1} - 1$$

- El total de números positivos (incluyendo el cero) será 2^{n-1} y el de negativos 2^{n-1}.

- Por ejemplo, con cuatro bits se pueden representar 16 números en el rango $-8 \leq Rango \leq 7$.

- Con ocho bits se pueden representar 256 números en el rango $-128 \leq Rango \leq 127$.

Complemento a dos

Decimal	Complemento a dos	Complemento a uno
7	0111	0111
6	0110	0110
5	0101	0101
4	0100	0100
3	0011	0011
2	0010	0010
1	0001	0001
0	0000	0000
0	0000	1111
-1	1111	1110
-2	1110	1101
-3	1101	1100
-4	1100	1011
-5	1011	1010
-6	1010	1001
-7	1001	1000
-8	1000	

Complemento a dos - Conversión a decimal

Dado un número expresado en binario con n bits utilizando complemento a dos, $N = (b_{n-1}b_{n-2}\cdots b_1 b_0)_2$, para convertir a decimal lo primero que tenemos que ver es el bit más a la izquierda:

1. Si es cero, entonces el número es positivo y está expresado en binario natural. Ejemplo: $N = (0110)_2 = 2^2 + 2^0 = (6)_{10}$
2. Si es uno, entonces el número es negativo. Podemos hacer la conversión de dos maneras:
   - 2.1 Hacer el C_2^N y luego convertirlo a binario: $N = (1010)_2$, entonces $C_2^N = (0110)_2 = (6)_{10}$. Por lo tanto, $N = (-6)_{10}$.
   - 2.2 Utilizar la fórmula:
     $$N = -b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_1 2^1 + b_0 2^0$$
     $$N = (1010)_2 = -2^3 + 2^1 = (-6)_{10}$$

Operaciones en complemento a dos

Sean dos números A y B en base 2, la suma $S = A + B$:

1. $A > 0$ y $B > 0$: La suma $S = A + B$ es correcta y el resultado es positivo ya que ambos números lo son.
2. $A < 0$ y $B < 0$: Podemos escribir la suma como $S = (2^n - |A|) + (2^n - |B|) = 2^n + 2^n - (|A| + |B|)$, siendo la suma correcta si se ignora 2^n. El resultado es negativo ya que ambos números lo son y queda expresado en el complemento correspondiente.
3. $A < 0$, $B > 0$ y $|A| < |B|$: En este caso podemos escribir la suma como $S = (2^n - |A|) + B = 2^n + (|B| - |A|)$, siendo la suma correcta si se ignora 2^n. El resultado es positivo por la relación propuesta para los valores absolutos.
4. $A > 0$, $B < 0$ y $|A| < |B|$: En este caso la suma se puede escribir como $S = A + (2^n - |B|) = 2^n - (|B| - |A|)$, siendo la suma correcta pues el resultado es negativo y está expresado en el complemento correspondiente.

Operaciones en complemento a dos (Ejemplos)

1. $A = 5$, $B = 12$, $S = A + B = 17$.

$$\begin{array}{rcccccccc} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hline & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}$$

y acarreo $c = 0$.

2. $A = -5$, $B = -12$, $S = A + B = -17$.

$$\begin{array}{rcccccccc} & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ + & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ \hline & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

y acarreo $c = 1$.

Complemento a dos - Banderas

- Es importante tener en cuenta el estado de las banderas dado que la ALU no tiene en cuenta si se está trabajando con matemática *Signed* o *Unsigned*.
- *Unsigned*, verificar si se ha seteado la bandera de *carry*:

```
 1111
+0001
-----
 0000      CF=1
```

  El resultado es incorrecto dado que $15 + 1 \neq 0$.

- *Signed*, verificar la bandera *overflow*:

```
 0100
+0100
-----
 1000      OF=1
```

  El resultado es incorrecto dado que $4 + 4 \neq -8$.

Overflow flag

Las reglas para "encender" la bandera overflow son dos:

- Si la suma de dos números con el bit más significativo en cero resulta en un número con el bit de signo en uno, la bandera *overflow* se enciende.

- Si la suma de dos números con el bit más significativo en uno resulta en un número con el bit de signo en cero, la bandera *overflow* se enciende.

Otros sistemas de representación: Sistema hexadecimal

- Un problema importante del sistema binario es su "verbosidad".
- Por ejemplo, para representar $(158)_{10} = (1001\ 1110)_2$ son necesarios ocho cifras en lugar de las tres necesarias en el sistema decimal.
- El sistema hexadecimal tiene las dos grandes ventajas:
  - La notación es muy compacta
  - Es muy simple convertir entre hexadecimal y binario, y viceversa.
- La base del sistema hexadecimal es 16.
- Los dígitos del sistema son los siguientes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- Notación: 0xFFFF ó 0xffff son notaciones equivalentes en C o Assembler para representar el número $(\text{ffff})_{16}$.

Sistema hexadecimal - Tabla de equivalencia

Hexadecimal	Decimal	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Conversión hexadecimal-decimal

- **Hexadecimal a decimal**: De acuerdo a la tabla anterior (y teniendo en cuenta el concepto de sistema de), el número $(15\mathrm{E})_{16}$ representa a

$$(15\mathrm{E})_{16} = 1 \times 16^2 + 5 \times 16^1 + 14 \times 16^0 = (350)_{10},$$

  dado que $(\mathrm{E})_{16} = (14)_{10}$.
- **Decimal a hexadecimal**: Deberemos dividir el número sucesivamente por 16. Ejemplo: $(465)_{10} = (1D1)_{16}$

```
465/16=29      resto=1
29/16=1        resto=13=D
```

Conclusión

Dado que $(158)_{10} = (9E)_{16} = (10011110)_2$, vemos que efectivamente aumentando la base conseguimos una representación más compacta.

Conversión hexadecimal-binario

- Ejemplo, convertir el número $(0AF3)_{16}$ a binario:

HEXADECIMAL	0	A	F	3
BINARIO	0000	1010	1111	0011

- Para convertir un número de binario a hexadecimal el primer paso es rellenar el número con ceros para asegurarse que el número de bits es múltiplo de cuatro. Luego, separar el número en grupos de cuatro bits y convertir cada grupo utilizando las equivalencias.
- Por ejemplo, para convertir el número binario $(0001\ 0011\ 0010\ 1110)_2$ en hexadecimal:

BINARIO	0001	0011	0010	1110
HEXADECIMAL	1	3	2	E

Operaciones en sistema hexadecimal

Suma

La suma se realiza de manera análoga a lo visto teniendo en cuenta el acarreo correspondiente. Ejemplo:

$$\begin{array}{rccc} & & \mathbf{1} & \\ & 1 & A & E \\ + & & 1 & 8 \\ \hline & 1 & C & 6 \end{array}$$

Resta

Para realizar la resta, podemos proceder restando dígito a dígito y teniendo en cuenta los acarreos. Ejemplo:

$$\begin{array}{rccc} & & \mathbf{1} & \\ & 1 & A & 6 \\ - & & 1 & C \\ \hline & 1 & 8 & A \end{array}$$

Representación octal

- El sistema octal tiene base 8, por lo cual utiliza 8 ocho dígitos diferentes para la representación.
- Los dígitos utilizados son los números enteros del 0 al 7.
- La conversión entre los mismos se puede realizar de manera análoga a lo ya visto.
- Tener en cuenta que para representar los 8 dígitos en sistema octal son necesarios 3 dígitos binarios.
- Ejemplo: $(1514)_8 = (001101001100)_2$

OCTAL	1	5	1	4
BINARIO	001	101	001	100