Sistemas Digitales

Nad Garraz y comunidad (ojalá)
Facultad de Ciencias Exactas y Naturales
UBA

Choose your destiny:

(dobleclick en los ejercicio para saltar)

Notas teóricas:

- *Complejidad:*

  Es lo que nos da la necesida de abstraer.

- *Abstracción*:

  Para lidiar con la *complejidad* que tienen los sistemas usamos la abstracción.

  De menor a mayor abstracción:

  electrones → transistores → circuitos analógicos → circuitos digitales → circuitos de lógica → micro-arquitectura → arquitectura → sistema operativo → aplicaciones de software

  Lo más abstracto es más fácil de controlar e implementar que lo menos abstracto.

- *Sistemas numéricos:*

  - *Decimal:*

    $$\texttt{9745}_{10} = 9 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

    El rango de un número *decimal* con $\texttt{n}$ dígitos mayor o igual a cero es $10^{\texttt{n}} : 0, 1, \ldots, 10^{\texttt{n}} - 1$

  - *Binario:*

    $$\texttt{1101}_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 10^1 + 1 \times 10^0 = \texttt{13}_{10}.$$

    El rango de un número *binario* con $\texttt{n}$ dígitos mayor o igual a cero es $2^{\texttt{n}} : 0, 1, \ldots, 2^{\texttt{n}} - 1$. Si tengo solo un bit, *binary digit*, puedo obtener o bien 0 o bien 1. Si tengo dos bits, entonces puedo tener 2^2 dígitos, 00, 01, 10, 11. (el rango es después de todo 4).

  - *Hexadecimal:*

    $$\texttt{2AD}_{16} = 2 \times 16^2 + \texttt{A} \times 16^1 + \texttt{D} \times 16^0 = \texttt{685}_{10}$$

    El rango de un número *hexadecimal* con $\texttt{n}$ dígitos mayor o igual a cero es $16^{\texttt{n}} : 0, 1, \ldots, 16^{\texttt{n}} - 1$. Cada dígito de un número en base hexadecimal corresponde a un número binario de 4-bits. Después de todo un número i_{16} de *un solo dígito* tiene un rango de 16: $0, \ldots 15$ y un número binario de 4-bits tiene un rango de $2^4 = 16$ ✓

    $$\texttt{2AD}_{16} = \underbrace{\texttt{0010}}_{\texttt{2}}\underbrace{\texttt{1010}}_{\texttt{A}}\underbrace{\texttt{1101}}_{\texttt{D}}\,_{\texttt{2}}$$

- *Operaciones entre números binarios:* Se poné picante según la representación usada.

  - Sumar es fácil si los números son positivos. Tengo *overflow* si el resultado tiene más cifras que bits disponibles para almacenar dicho resultado.

- *Números binarios con signo:*

  Hay distintas *representaciones* para hacer esto, acá están las 2 más usadas:

  - *Signo / magnitud:*

    El bit más significativo, el de más a la izquierda, marca el signo. Un número con $\texttt{n}$ bits en esta representación tiene un rango: $[-2^{\texttt{n}-1} + 1, 2^{\texttt{n}-1} - 1]$. Por ejemplo con 3-bits:

El rango es [-3, 3]

Base 2	→	Base 10
000	→	0
001	→	1
010	→	2
011	→	3
100	→	-0
101	→	-1
110	→	-2
111	→	-3

Sumar normalmente en esta representación no tiene sentido. Representa en total $2^{n} - 1$ elementos porque el -0 está usando un lugar al pedo.

- *Complemento de 2:*
  - Menos intuitivo, pero más útil. Si tengo un número de n-bits, voy a tener siempre:

0_{10}	→	$00\ldots00_2$
-1_{10}	→	$11\ldots11_2$
$-2^{n-1}{}_{10}$	→	$100\ldots00_2$ → *weird number*
$2^{n-1} - 1_{10}$	→	$011\ldots11_2$

    Teniendo al 0, -1, -2^{n-1} y $(2^{n-1} - 1)$, puedo encontrar el resto de la *representación*:
  - Para encontrar el opuesto a un número, se cambian los 0 por 1 y bicerveza [1] , luego se le suma 1 a eso, ej:

$$5_{10} = 0101_2 \xrightarrow[-5]{\text{busco}} 1010_2 + 0001_2 = 1011_2 = -5_{10}.$$

    Cosa que no funciona con el *weird number*, porque su complemento te da a él mismo.
  - El rango es de $[-2^{n-1}, 2^{n-1} - 1]$, 2^{n} elementos. Hay un elemento negativo más que positivos.
  - Ejemplito: En 3-bits me encuentro todo el conjunto, $[-4, 3]$:

000_2	→	0_{10}
001_2	→	1_{10}
010_2	→	2_{10}
011_2	→	3_{10}
100_2	→	-4_{10}
101_2	→	-3_{10}
110_2	→	-2_{10}
111_2	→	-1_{10}

  - *Suma en complemento de 2:* Si sumo dos números de distinto signo no voy a tener *overflow*!

$$\text{4 bits: } -4_{10} + 5_{10} = 1100_2 + 0101_2 = \cancel{1}0001_2 = 1_{10}$$

  - *Sign extension:* Para encontrar la representación de un número conocido con más bits. Copio el signo al resto de los númerosengo que mandar el bit del signo hacia el dígito más significativo, :

$$\text{4 bits: } 6_{10} = 0110_2 \text{ y } -5_{10} = 1011_2$$
$$\text{Extendido a 8 bits: } 6_{10} = (0000\ 0110)_2 \text{ y } -5_{10} = (1111\ 1011)_2$$

[1] chiste: bicerveza = . Se escribe viceversa .

Ejercicios de la guía:

Ejercicio 1

a) Utilizando el método del cociente, expresar en bases 2, 3 y 5 los números 33_{10} y 511_{10}.

b) Expresar en decimal los números 1111_2, 1111_7 y $\mathtt{CAFE}_{10}$.

c) Expresar 17_8 en base 5 y $\mathtt{BABA}_{13}$ en base 6.

d) Pasar $(1010\ 1110\ 1010\ 1101)_2$, $(1111\ 1011\ 0010\ 1100\ 0111)_2$, $(0\ 0110\ 0010\ 1001)_2$, a base 4, 8 y 16 agrupando bits.

e) Expresar en decimal los números `0x142536`, `0x142536` y `0xFCD9`, y pasar a base 16 los números 7848_{10} y 46183_{10}.

---

a) *Fijarse si el número es cercado a una potencia de la base buscada.*

$33_{10} = (10\ 0001)_2 = 1020_3 = 113_5$

$511_{10} = (1111\ 1111)_2 = (20\ 0221)_3 = 4021_5$

b) ¿Tiene truco?

$1111_2 = 15_{10}$

$1111_7 = 400_{10}$

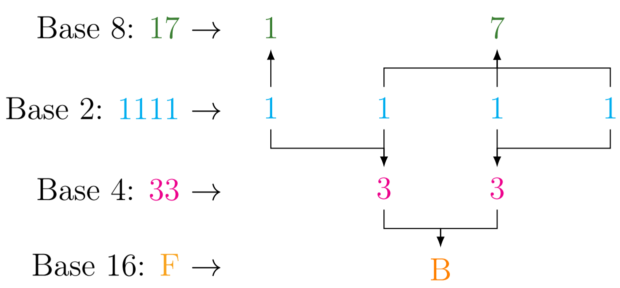
$\mathtt{CAFE}_{16} = \mathtt{C}\times16^3 + \mathtt{A}\times16^2 + \mathtt{F}\times16^1 + \mathtt{E}\times16^0 = 12\times16^3 + 10\times16^2 + 15\times16^1 + 14\times16^0 = 51966_{10}$

c) ¿Hay truco para no pasar por base 10?

$17_8 = 15_{10} = 30_5$

$\mathtt{BABA}_{13} = 11\times13^3 + 10\times13^2 + 11\times13^1 + 10\times13^0 = 26180_{10} = 321112_6$

d) Como las bases se tiene que cambiar a otras bases que son potecia de 2, se puede hacer agrupando los bits.



$(1010\ 1110\ 1010\ 1101)_2 = (22\ 32\ 22\ 31)_4 = 5555_8 = \mathtt{AEAD}_{16}$

$(1111\ 1011\ 0010\ 1100\ 0111)_2 = (33\ 23\ 02\ 30\ 13)_4 = (373\ 1307)_8 = \mathtt{FB2CD}_{16}$

$(0\ 0110\ 0010\ 1001)_2 = (12\ 02\ 21)_4 = 3051_8 = \mathtt{C29}_{16}$

e) $\mathtt{0x142536} = 142536_{16} = 1\times16^5 + 4\times16^4 + 2\times16^3 + 5\times16^2 + 3\times16^1 + 6\times16^0 = 1320246_{10}$

$\mathtt{0xFCD9} = \mathtt{FCD9}_{16} = 15\times16^3 + 12\times16^2 + 13\times16^1 + 9\times16^0 = 64729_{10}$

$7848_{10} = \mathtt{0x1EA8}$ $46183_{10} = \mathtt{0xB467}$

Ejercicio 2 Realizar las siguientes sumas de precisión fija, sin convertir a decimal. Indicar en cada caso si hubo acarreo.

a) $\begin{array}{r} \texttt{100001}_2 \\ + \;\; \texttt{011110}_2 \\ \hline \texttt{------}_2 \end{array}$ b) $\begin{array}{r} \texttt{100001}_2 \\ + \;\; \texttt{011111}_2 \\ \hline \texttt{------}_2 \end{array}$ c) $\begin{array}{r} \texttt{9999}_{16} \\ + \;\; \texttt{1111}_{16} \\ \hline \texttt{----}_2 \end{array}$ d) $\begin{array}{r} \texttt{F0F0}_2 \\ + \;\; \texttt{F0CA}_2 \\ \hline \texttt{----}_2 \end{array}$

---

a) $\begin{array}{r} \texttt{100001}_2 \\ + \;\; \texttt{011110}_2 \\ \hline \texttt{111111}_2 \end{array}$ c) $\begin{array}{r} \texttt{9999}_{16} \\ + \;\; \texttt{1111}_{16} \\ \hline \texttt{AAAA}_2 \end{array}$

b) $\begin{array}{lr} \texttt{Acarreo} & \texttt{111111} \\ & \texttt{100001}_2 \\ + & \texttt{011111}_2 \\ \hline & \texttt{1000000}_2 \end{array}$ d) $\begin{array}{lr} \texttt{Acarreo} & \texttt{1 1} \\ & \texttt{F0F0}_2 \\ + & \texttt{F0CA}_2 \\ \hline & \texttt{1E1BA}_2 \end{array}$

Precisión fija? Tengo overflow en el b y en el d? Se tira el decimal de más? Cómo se expresa el resultado?

Ejercicio 3 ¿Puede suceder en alguna base que la suma de dos números de precisión fija tenga un acarreo mayor que 1? Exhibir un ejemplo o demostrar lo contrario.

Ejercicio 4 Sean los siguientes numerales binarios de ocho dígitos:

$$\texttt{r} = (\texttt{1011 1111})_2,\ \texttt{s} = (\texttt{1000 0000})_2 \text{ y } \texttt{t} = (\texttt{1111 1111})_2.$$

¿Qué números representan si asumimos que son codificaciones de enteros en complemento a 2? ¿ Y si fueran codificaciones en signo+magnitud?

En la teoría están cuales son los valores *amigos* de la representación complemento a 2

- Notar que si $\texttt{r}$ es de complemento:

  $\texttt{r} + 64 = (\texttt{1011 1111})_2 + (\texttt{0100 0000})_2 = (\texttt{1111 1111})_2 = -1_{10} \to \texttt{r} = -65_{10}$

  Puedo llegar a lo mismo usando la técnica para encontrar el complemento de $\texttt{r}$:

  $\overline{\texttt{r}} = \overline{\texttt{10111111}_2} \xrightarrow[1\leftrightarrow 0]{\text{cambio}} (\texttt{0100 0000})_2 \xrightarrow[1]{\text{sumo}} (\texttt{0100 0001})_2 = {}_2 = 65_{10} = \overline{\texttt{r}}$, entonces $\texttt{r} = -65_{10}$

  Si fuese de signo+magnitud: $\texttt{r} = (\texttt{1011 1111})_2 = -63_{10}$

  Sumar normalmente en la representación de signo+magnitud es para problemas.

- Para complemento a 2: $\texttt{s} = (\texttt{1000 0000})_2 = -128_{10}$ es el *weird number*

  Y en signo+magnitud: $\texttt{s} = (\texttt{1000 0000})_2 = -0_{10}$

- Para complemento a 2: $\texttt{t} = (\texttt{1111 1111})_2 = -1_{10}$

  Y en signo+magnitud: $\texttt{t} = (\texttt{1111 1111})_2 = -127_{10}$

Ejercicio 5 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 6 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 7 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 8 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 9 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 10 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 11 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 12 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.

Ejercicio 13 ... hay que hacerlo!
Si querés mandarlo: Telegram →, o mejor aún si querés subirlo en LaTeX→.