Operaciones con Bits y Bytes

Pablo L. De Nápoli

Departamento de Matemática
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Álgebra I - Primer cuatrimestre de 2020

Bits y bytes

En esta clase discutiremos una aplicación del álgebra booleana a la computación. Aparece comentada en el ejercicio 37 de la práctica 1 y también se relaciona con el ejercicio 15.

En las computadoras digitales, la unidad básica de información es el bit (palabra que procede de la abreviatura de binary digit, dígito binario en inglés). Un bit es un componente en la memoria de la computadora que puede tener dos estados

- encendido: convencionalmente representado por 1, que podemos pensarlo V (verdadero)
- apagado: convencionalmente representado por 0, que podemos pensarlo F (falso)

Usualmente los bits se agrupan en bloques. Un bloque de 8 bits se denumina un byte. Por ejemplo:

0	1	1	0	0	0	0	1

es un byte.

¿Cómo representa una computadora la información?

Internamente, en una computadora digital todo se representa como bits. Empecemos por los números enteros: se representan usualmente utilizando el sistema binario de numeración. Un byte como

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

codifica al número

$$(b_7\,b_6\,b_5\,b_4\,b_3\,b_2\,b_1\,b_0)_2 = 2^7 \cdot b_7 + 2_6 \cdot b_6 + 2^5 \cdot b_5 + 2^4 \cdot b_4 + 2^3 \cdot b_3 + 2^2 \cdot b_2 + 2^1 \cdot b_1 + 2^0 \cdot b_0$$

Por ejemplo el byte

0	1	1	0	0	0	0	1

representa al número

$$2^6 + 2^5 + 2^0 = 97$$

De esta forma, un byte puede representar un número entero entre 0 y $255 = 2^8 - 1$.

¿Qué pasa con las letras y demás caracteres?

Internamente las computadoras representan todos los caracteres (letras, números o simbolos especiales) como números binarios de acuerdo al código ASCII:

Caracter	Número Binario	Eq. Decimal
espacio	00100000	32
'!'	00100001	33
'A'	01000001	65
'a'	01100001	97

De modo que si lo interpretamos como un caracter, nuestro bit puede representar a la letra a Pueden probarlo en Python 3,

```
>>> chr(97)
'a'
>>> ord('A')
65
>>> bin(97)
'0b1100001'
```

Otros usos de los bits

Todo en una computadora se representa internamente por medio de bits.

- Por ejemplo, en mi primera computadora (la ZX-spectrum) cada punto de la pantalla (pixel) podía tener dos estados: encendido/apagado.
- Similarmente, había 8 colores se se formaban con los colores básicos R (red, rojo), G (green, verde), B (blue, azul).

G	R	B
1	1	0

representaba al color amarillo (mezcla de verde y rojo). Hoy HTML usa un sistema similar con 3 bytes de información.

Operaciones con bits

- ´Las operaciones lógicas booleanas que introdujimos en la primer clase puden pensarse entonces como operaciones con bits

p ((C, Python 3)) (lógica matemática) (ej. 37, Pca,1)	q	$NOT(p)$ $!p$ $\sim p$	p AND q p & q $p \wedge q$ A_b	p OR q $p \vert q$ $p \vee q$ O_b	p XOR q $p \hat{} q$ $p \veebar q$ X_b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

- Las operaciones con bits se pueden hacer con bytes bit a bit.
- El conjunto de posibles bytes B con las operaciones lógicas bit a bit es otro ejemplo de un álgebra de Boole.

Operaciones con bytes: la operación OR

Consideramos el byte que representaba a la letra A

0	1	0	0	0	0	0	1

Si hacemos la operación OR con el byte

0	0	1	0	0	0	0	0

obtenemos el byte que representa la letra a en el código ASCII.

0	1	1	0	0	0	0	1

Notemos que hacer esta operación permite prender uno o más bits elegidos.

Esto podría usarse por ejemplo para pasar un texto de minúsculas a mayúsculas.

Notación del ejercicio 37: O_b es la operación OR con un byte fijo b (máscara).

Operaciones con bytes: la operación AND

Similarmente, consideramos el byte que representaba a la letra a

0	1	1	0	0	0	0	1

Si hacemos la operación AND con el byte

1	1	0	1	1	1	1	1

obtenemos el byte

0	1	0	0	0	0	0	1

que representa la letra A en el código ASCII.

Notemos que hacer esta operación permite apagar uno o más bits elegidos.

Esto podría usarse por ejemplo para pasar un texto de mayúsculas a minúsculas.

Notación del ejercicio 37: A_b es la operación AND con un byte fijo b (máscara).

Operaciones con bytes: la operación XOR

De nuevo, consideramos el byte que presentaba la letra a

0	1	1	0	0	0	0	1

y hagamos la operación XOR con el byte

1	0	0	0	0	0	0	1

obtenemos el byte

1	1	1	0	0	0	0	0

Vemos que la operación XOR permite cambiar el estado de ciertos bits elegidos.

Notación del ejercicio 37: X_b es la operación XOR con un byte fijo b (máscara).

¡Lo hacemos en la computadora! (en Python 3)

```
>>> x=ord('a');x; bin(x)
97
'0b1100001'
>>> b= 0b00100000; b ; bin(b)
32
>>> bin(x or b); x or b; chr(x or b)
'0b1100001'
97
'a'
>>> bin(x and b); x and b; chr(x and b)
'0b100000'
32
' '
>>> bin(x ^ b); x ^ b; chr(x ^ b)
'0b1000001'
65
'A'
```

¿Cómo suma una computadora?

Supongamos que tenemos dos bits y queremos sumarlos (en binario)

p	q	$p+q$
0	0	00
0	1	01
1	0	01
1	1	10

Escribiendo

$$p + q = (b_1 b_0)_2$$

vemos que

$$b_0 = p \ XOR \ q$$

$$b_1 = p \ AND \ q$$

o sea que ¡la suma puede hacerse mediante estas operaciones lógicas!. (que pueden implementarse mediante circuitos).
Referencia:
https://www.electronics-tutorials.ws/combination/comb_7.html

Operaciones de corrimiento (shift)

Además de las operaciones lógicas, en el ejercicio 37 pca.1 se definen las operaciones de corrimiento

- Corrimiento hacia la derecha (R): desplaza cada bit un lugar hacia la derecha, pone un 0 en el bit 7 y descarta el bit 0.
- Corrimiento hacia la izquierda (L): desplaza cada bit un lugar hacia la izquierda, pone un 0 en el bit 0 y descarta el bit 7.

Ejemplo: si aplicamos el corrimiento a derecha al byte

0	1	1	0	0	0	0	1

obtenemos el byte

0	0	1	1	0	0	0	0

mientras que si le aplicamos el corrimiento a la izquierda obtenemos el byte

1	1	0	0	0	0	1	0

Resolución de uno de los ítems del ejercicio 37

- En el ejercicio 37 se considera las operaciones R,L,A_b,O_b y X_b como funciones de B en B y se pide calcular algunas de sus composiciones. Consideremos por ejemplo la composición $A_b \circ O_b$. Por definición, si $x \in B$,

$$O_b \circ A_b(x) = (\ x\ AND\ b)\ OR\ b$$

Como estas operaciones actúan bit a bit sobre los bytes de x, nos bastará ver qué efecto producen sobre un bit

x	b	$x\ AND\ b$	$(\ x\ ANDb)\ OR\ b$
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	1

Vemos que $O_b \circ A_b(x) = b$ [una función constante].
En el lenguaje de los conjuntos esto se escribiría

$$(X \cap B) \cup B = B$$

[Ley de absorción del álgebra booleana]