Ejercicios de operadores a nivel de bits

Gonzalo F. Perez Paina

Ejercicicios

1. Realice las operaciones que se indican a continuación considerando que los operandos son números de 8 bits en notación decimal. Expresar el resultado en notación decimal, binaria y hexadecimal.

   a) 13 & 10

   e) 1 ^ 6

   i) 90 >> 1

   b) 26 & 21

   f) 1 ^ 27

   j) 1 << 2

   c) 14 | 13

   g) 32 >> 4

   k) 3 << 4

   d) 33 | 5

   h) 128 >> 3

   l) 11 << 4

2. Resolver las siguientes operaciones expresando el resultado en notación decimal, binaria y hexadecimal.

   a) 240 | (1 << 1)

   e) (1 << 0) | (1 << 2)

   b) 168 | (1 << 2)

   f) (1 << 2) | (1 << 4)

   c) 255 & ~(1 << 2)

   g) (1 << 2) | ~(1 << 5)

   d) 254 & ~(1 << 3)

   h) ~(1 << 1) | (1 << 3)

3. Si `num` es una variable tipo `unsigned char` cuyo bit menos significativo se designa como b0 y el más significativo como b7. Indicar cuáles son los bits que se modifican y qué valores toman al aplicar las siguientes operaciones:

a) num |= 1 << 2;

d) num |= (1 << 1) | (1 << 3);

b) num &= ~(1 << 1)

e) num &= ~((1 << 2) | (1 << 6));

c) num ^= (1 << 3);

f) num ^= (1 << 0) | (1 << 1);

4. Escribir un programa que imprima en notación binaria un número de 8 bits recibido desde la línea de comandos. El siguiente código fuente muestra el esqueleto del programa con una posible implementación de la función para imprimir en notación binaria una variable tipo `unsigned char`.

```c
#include <stdio.h>
#include <stdlib.h>

void binary_print(unsigned char );

int main(int argc , char *argv[])
{
    /* Completar */
}

void binary_print(unsigned char val)
{
    unsigned char b, mask = 1 << 7;

    for(b = 1; b <= 8*sizeof(unsigned char); b++)
    {
        putchar(val & mask ? '1' : '0');
        mask >>= 1;
    }
}
```

Completar el cuerpo de la función `main()` para que el programa tenga la interacción del usuario como se muestra a continuación.

```
./binary
Uso: ./binary <num>
./binary 16
 16 = 00010000
./binary 128
 128 = 10000000
./binary 160
 160 = 10100000
```

Al ejecutar el programa sin argumentos, el mismo muestra la forma correcta de uso indicando que espera recibir un valor numérico como argumento.

5. Modificar el programa del ejercicio anterior para que pueda imprimir en binario números de 16 bits. Responder además las siguientes preguntas:
   1. ¿Qué tipo de dato del lenguaje C resulta adecuado para almacenar un valor entero positivo de 16 bits?
   2. ¿Cuál es el valor máximo que puede almacenar este tipo de dato?
   3. ¿Qué valores hay que pasarle al programa para que el número binario tenga un único bit en uno (desde el bit b0 al bit b15)?

6. Escribir un programa que realice las operaciones AND, OR y OR exclusiva a nivel de bits entre dos valores numéricos de 8 bits cargados desde la línea de comandos. La interacción con el usuario debe ser como se muestra a continuación, donde los valores de entrada se expresan en notación decimal y el resultado en notación decimal y hexadecimal. `./and_or_xor Uso: ./and_or_xor <num1> <num2> ./and_or_xor 128 160 128 & 160 = 128 0x80 & 0xA0 = 0x80 128 | 160 = 160 0x80 | 0xA0 = 0xA0 128 ^ 160 = 032 0x80 ^ 0xA0 = 0x20`

7. Escribir un programa que muestre el valor de los bits menos significativo (LSB, Least Significant Bit) y más significativo (MSB, Most Significant Bit) de un valor numérico de 8 bits cargado desde la línea de comandos. La interacción con el usuario debe ser como se muestra a continuación. `./lsb_msb Uso: ./lsb_msb <num> ./lsb_msb 128 El LSB de 128 es 0 El MSB de 128 es 1`

8. Escribir un programa que muestre el valor de un bit a elección de una variable numérica de 8 bits. Tanto el número como el bit a leer se pasan por línea de comandos. La interacción con el usuario debe ser como se muestra a continuación. `./get_bit Uso: ./get_bit <num> <bit> ./get_bit 160 0 El bit 0 de 160 es 0 ./get_bit 160 5 El bit 5 de 160 es 1`

9. Escribir un programa para evaluar el operador de desplazamiento a la izquierda que reciba los dos operandos desde la línea de comandos. La interacción con el usuario debe ser como se muestra a continuación. `./left_shift 2 4 2 << 4 = 32 00000010 << 4 = 00100000 ./left_shift 3 3 3 << 3 = 24 00000011 << 3 = 00011000`

10. Escribir un programa para evaluar el operador de desplazamiento a la derecha que reciba los dos operandos desde la línea de comandos.

11. Escribir un programa que ponga a uno lógico un bit específico en un número de 8 bits. La interacción con el usuario debe ser como se muestra

a continuación. `./set_bit 6 0  6 = 00000110  7 = 00000111 ./set_bit 11 6  11 = 00001011  75 = 01001011`

12. Escribir un programa que ponga a cero lógico un bit específico en un número de 8 bits.

13. Escribir un programa que conmute (toggle) el estado (de 0 a 1 y de 1 a 0) un bit en un número de 8 bits.

14. Escribir un programa que reciba por línea de comandos un entero de 32 bits e imprima en pantalla el valor de los bytes que lo componen en notación hexadecimal. La interacción con el usuario debe ser como se muestra a continuación. `./int_bytes 255  255 = 00 00 00 FF  ./int_bytes 65535  65535 = 00 00 FF FF  ./int_bytes 123456789  123456789 = 07 5B CD 15`