7. Operadores de bits

Nota: Los siguientes ejercicios se refieren a programas ANSI-C modularizados.

1) Calcular las siguientes operaciones: `11001010&10100101`, `11001010|10100101`, `11001010^10100101`.

2) Escribir las representaciones octales de los números binarios del ejercicio anterior.

3) Determinar los resultados octales de las siguientes operaciones, suponiendo que los números no tienen signo (los números están expresados en sistema octal):

a) `0157` trasladado a izquierda una posición de bit,

b) `0701` trasladado a izquierda dos posiciones de bit,

c) `0673` trasladado a derecha dos posiciones de bit,

d) `067` trasladado a derecha tres posiciones de bit.

4) Una máscara es un patrón binario en una palabra. Las máscaras se utilizan de acuerdo con las siguientes operaciones: and, or y not. Escribir un programa que utilice una máscara para disponer 1's en posiciones que se deseen, para los bits de una palabra.

5) Escribir un fragmento de código/procedimiento que convierta un número entero sin signo (32 bits) a formato Big Endian, sobre un arreglo de 4 bytes. Imprimir por `stdout` las representaciones origen y destino.

Ejemplo: el número `271590900` será codificado como:

```
x[0] = 0x10
x[1] = 0x30
x[2] = 0x25
x[3] = 0xF4
```

6) Escribir un fragmento de código/procedimiento que convierta un número entero en representación Big Endian de cuatro bytes a formato decimal. Imprimir por `stdout` las representaciones origen y destino.

Ejemplo: el número `|0x10|0x30|0x25|0xF4|` será decodificado como: `271590900`.

¿Qué aplicaciones tiene la codificación/decodificación Big Endian en Electrónica & Telecomunicaciones?

7) Escribir un programa que utilice una máscara para invertir los bits de determinadas posiciones.

8) Para un aplicativo a ejecutarse sobre una plataforma de al menos 32 bits que contendrá una interfaz gráfica de usuario (GUI), se requiere disponer de funciones para el tratamiento de colores de los elementos gráficos de la interfaz. El color de cualquier elemento gráfico se almacenará en una variable de tipo `int` sin signo, de acuerdo a la convención RGB (ejemplo: un color con representación RGB = `0xFFCCAA` se corresponde con `0xFF`, `0xCC` y `0xAA` para las componentes Rojo, Verde y Azul, respectivamente). Se pide:

a) Escribir una función denominada `Rojo()` que reciba como argumento el color en representación RGB sobre una variable de tipo `int` sin signo, y que devuelva por el nombre la componente de color rojo como un `char` sin signo. Dar un ejemplo de invocación de la función.

b) Idem para una función denominada `Verde()` que devuelva la componente verde. Dar un ejemplo de invocación de la función.

c) Idem para una función denominada `Azul()` que devuelva la componente Azul. Dar un ejemplo de invocación de la función.

d) Escribir una función llamada `ComponentesRGB()` que reciba como argumentos el color en notación RGB sobre una variable de tipo `int` sin signo, y devuelva a través de la interfaz las tres componentes R, G, y B simultáneamente sobre variables de tipo `unsigned char`. Dar un ejemplo de invocación de esta función.

e) Escribir una función llamada `MezclarColores()` que reciba como argumento dos colores y devuelva el color resultante de la mezcla (síntesis aditiva) de los mismos. Dar un ejemplo de invocación.

f) Escribir una función llamada `ModificarBrillo()` que reciba como argumento un color y un flotante en el rango $[-1, 1]$ y devuelva el color resultante de modificar porcentualmente cada componente por dicho valor. Dar un ejemplo de invocación.

g) Escribir una función llamada `ColorComplementario()` que reciba como argumento un color y devuelva el color resultante de "complementar" cada una de sus componentes (*i.e.* computar la distancia de cada valor a 255). Dar un ejemplo de invocación.

h) Escribir los prototipos de las funciones desarrolladas en los puntos anteriores.

NOTA: Se debe utilizar operadores de bits y máscaras para la resolución de este ejercicio, y documentar todas y cada una de las funciones desarrolladas.

9) Un microcontrolador posee el siguiente registro para el manejo de datos a través de un puerto serie que puede conectarse a una PC.

bit 7	6	5	4	3	2	1	bit 0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8

RXCIE Cuando este bit es 1 indica que se ha terminado de recibir un dato por el puerto serie.

TXCIE Cuando este bit es 1 indica que se ha terminado de enviar un dato por el puerto serie.

UDRIE Cuando este bit es 1 habilita las interrupciones de hardware durante el envío y recepción de datos por el puerto serie.

RXEN Cuando este bit es 1 habilita al puerto serie a recibir datos, si vale 0 RXCIE no tiene funcionalidad.

TXEN Cuando este bit es 1 habilita al puerto serie a enviar datos, si vale 0 TXCIE no tiene funcionalidad.

CHR9 Cuando este bit es 1 el dato tiene 9 bits de longitud más 1 bit de inicio y otro de fin. Cuando vale 0, la longitud del dato es de 8 bits más 1 de inicio y otro de fin.

RXB8 Si CHR9 es 1, este bit contiene el noveno bit del dato a ser recibido.

TXB8 Si CHR9 es 1, este bit contiene el noveno bit del dato a ser transmitido.

Se pide:

a) Defina un tipo enumerativo denominado `bit_t` con los símbolos`HI` y `LO`, para representar un bit. Defina otro tipo enumerativo denominado `bool_t` con los símbolos `TRUE` y `FALSE`.

b) Escriba el código de una función denominada `transmision_completa()` que indique, a partir del contenido del byte de control (bit TXCIE), y mediante la devolución de un valor por el nombre si se ha completado o no la transmisión de un símbolo por el puerto serie. El prototipo de la función pedida es:

```
bit_t transmision_completa(unsigned char);
```

c) Dar un ejemplo de invocación de la función.

d) Modifique el código de la función pedida en el punto $9b$) para que la misma indique, mediante la devolución de un valor por la interfaz, si se ha completado o no la transmisión. Dar un ejemplo de invocación de esta función.

NOTA: Debe utilizarse máscaras y operadores de bits para leer cada bit del registro de datos.

10) Un sintetizador de una radio digital posee un generador de frecuencia controlado por el siguiente registro:

bit 7	6	5	4	3	2	1	bit 0
AFT	BAND	E	D	C	B	A	SYN

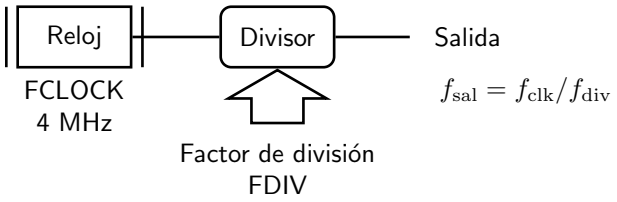
AFT Control de Sintonía Fina Automática (1 = ON , 0 = OFF).

BAND Control de selección de banda (1 = AM , 0 = FM).

E, D, C, B, A Factor de división de la frecuencia de reloj (ver tabla).

SYN Control de activación del sintetizador (1 = ON, 0 = OFF).

E	D	C	B	A	Factor de división
0	0	0	0	0	1
0	0	0	0	1	2
0	0	0	1	0	3
0	0	0	1	1	4
...	...	...	...	...	...
1	1	1	1	0	31
1	1	1	1	1	32



Se pide:

a) Defina un tipo enumerativo denominado `band` con los símbolos `AM` y `FM`, para representar las bandas del sintonizador.

b) Escriba el código de una función denominada `get_synthesizer_divider()` que devuelva por su nombre el factor de división a partir de los bits A, B, C, D y E contenidos en el byte de control. El prototipo de la función pedida es:

```
int get_synthesizer_divider(unsigned char);
```

Dar un ejemplo de invocación de la función.

c) Escriba el código de una función denominada `set_synthesizer_divider()` que reciba como argumento formal el factor de división y a partir del mismo adecúe los bits A, B, C, D y E del byte de control que se recibe como segundo argumento por puntero. El prototipo de la función pedida es:

```
void set_synthesizer_divider(int fdiv, unsigned char *control);
```

Dar un ejemplo de invocación de la función.

d) Escriba el código de una función denominada `get_band()` que reciba como parámetro el registro de control, y retorne por el nombre la banda seleccionada como un tipo enumerativo `band_t`. Dar un ejemplo de invocación.

e) Escriba el código de una función denominada `set_band()` que reciba como parámetro una copia del registro de control y la banda a seleccionar como segundo argumento, y retorne por el nombre el registro actualizado como un tipo `unsigned char`. Dar un ejemplo de invocación de la función.

Prototipo: `unsigned char set_band(unsigned char cotrol, band_t nueva_banda);`

NOTA: Debe utilizarse constantes simbólicas, máscaras y operadores de bits para operar con los bits del registro de datos.

11) Para un subsistema de comunicaciones basado en un microcontrolador, que forma parte de un equipo de medición, se debe programar el registro de control de periféricos (SPCR), el cual tiene la siguiente estructura:

bit 7	6	5	4	3	2	1	bit 0	
SPIE	SPE	-	-	CPOL	SPR1	SPR0	-	SPCR

En donde:

SPIE control de interrupciones (1: interrupciones habilitadas, 0: deshabilitadas)

SPE encendido del periférico (1: encendido, 0: apagado)

CPOL control de polaridad (1: clock activo por nivel bajo, 0: clock activo por nivel alto)

SPR1 & SPR0 factor de división del reloj interno, de acuerdo a la siguiente tabla:

SPR1	SPR0	Factor de división	Frecuencia de clock
0	0	2	1MHz
0	1	4	500kHz
1	0	16	125kHz
1	1	32	62.5kHz

Se pide

a) Escribir dos funciones que devuelvan el estado de los bits SPIE y CPOL respectivamente, y que respondan a los siguientes prototipos:

```
bit_t getSPIE(unsigned char);
bit_t getCPOL(unsigned char);
```

b) Escribir una función que devuelva por el nombre el factor de división del clock interno, que responda al prototipo:

```
int getPrescalingFactor(unsigned char);
```

c) Escribir una función `getCOMControl()` que devuelva simultáneamente por la interfaz el estado de los bits SPIE, y CPOL, y del factor de división del clock interno (esta función no debe devolver nada por el nombre). Escribir su prototipo.

d) Escribir una función cuyo prototipo sea: `void setCPOL(unsigned char *control, bit_t bit);` que permita colocar el bit CPOL en 1 ó 0, dependiendo si el valor del argumento bit es `UNO` o `CERO`, respectivamente.

e) Dar un ejemplo de invocación de cada una de las funciones desarrolladas en los puntos anteriores.

NOTA: Se deben utilizar operadores de bits y máscaras para la resolución de este ejercicio. Asumir que se encuentra definido `typedef enum{UNO, CERO} bit_t;`

12) Para un programa que hará uso del puerto paralelo de una PC se necesita disponer de funciones que permitan colocar a nivel lógico 1 ó 0 un bit determinado dentro de un byte a ser presentado al puerto paralelo. Se pide:

a) Escribir una función llamada `Set()`, que obedezca al siguiente prototipo:

```
unsigned char Set(unsigned char Datos, short Linea);
```

que reciba un byte sobre la variable `datos` y el número de línea que será forzada a nivel lógico 1, devolviendo por el nombre el resultado de la operación.

Ejemplo: Si en `datos` se recibe 1000 0001 y `Linea = 3`, se debe retornar el byte 1000 **1**001.

b) Dar un ejemplo de invocación de la función.

c) Idem 12*a*) pero para una función llamada `Clear()` que coloque a nivel lógico 0 la línea indicada, y devuelva por el nombre el resultado de la operación.

Ejemplo: Si en `Datos` se recibe 0111 1100 y `Linea = 3`, se debe devolver el byte 0111 **0**111.

d) Dar un ejemplo de invocación de la función.

ACLARACION: La línea 0 se corresponde con el bit menos significativo (LSB) del byte, y la linea 7 con el bit más significativo (MSB) del byte.