

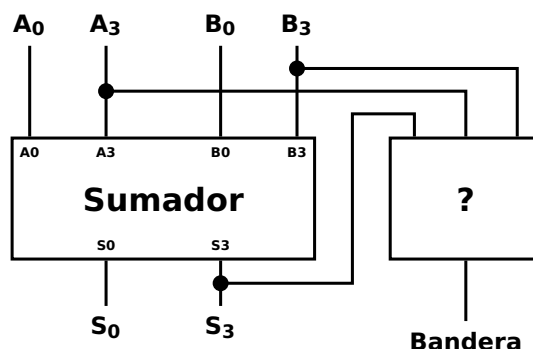
Guía de Solución para el Trabajo Práctico Número 1

Sistemas con Microprocesadores y Microcontroladores

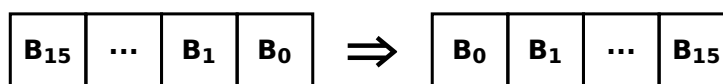
24/08/2026

Problemas Propuestos

- 5) Construir un detector de desbordamiento para un circuito sumador que recibe dos números de cuatro bits y genera un resultado también de cuatro bits más una bandera. Resolver el ejercicio para el caso de los números sin signo (acarreo) y con signo (desbordamiento).



- 14) Escriba una función en lenguaje C que realice un intercambio de bits en un entero sin signo de 16 bits. La función deberá intercambiar el bit 15 con el 0, el 14 con el 1 y así sucesivamente:



Soluciones Completas

- 5) Antes de plantear la solución del ejercicio es conveniente recordar la diferencia entre el acarreo y el desbordamiento:

- El acarreo se da en las operaciones de suma cuando el resultado de la suma requiere un bit más para poder ser representado, y también en las operaciones de resta cuando el sustraendo es mayor que el minuendo y es necesario "pedir prestada una unidad a la izquierda". Este bit de acarreo permite realizar operaciones de múltiple precisión, y también permite detectar que el resultado es mayor que los que se puede representar cuando los operandos son números sin signo.
- El desbordamiento se da cuando el resultado de una operación no se puede representar con la cantidad de bits utilizadas. En el caso de los números sin signo coincide con el acarreo, pero en el caso de los números con signo se puede detectar en base a los bits más significativos de los operandos y del resultado: Si los operandos son de signo contrario entonces no puede haber desbordamiento, si los operandos son del mismo signo y el resultado es de signo contrario entonces hay desbordamiento.

Teniendo en cuenta estas definiciones realizamos las tablas de verdad de ambas banderas, y a partir de ellas podemos escribir las ecuaciones lógicas, recordando que $\cdot$ es equivalente a la operación AND y $+$ a la operación OR.

A3	B3	S3	Acarreo
0	0	X	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	X	1

A3	B3	S3	Desbordamiento
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$\text{Desbordamiento} = \overline{A3} \cdot \overline{B3} \cdot S3 + A3 \cdot B3 \cdot \overline{S3}$$

14) Para empezar, tenemos un numero de 16 bits:

$|b_{15}|b_{14}|b_{13}|b_{12}|b_{11}|b_{10}|b_9|b_8|b_7|b_6|b_5|b_4|b_3|b_2|b_1|b_0|$

El cual nuestro código debe devolver como:

$|b_0|b_1|b_2|b_3|b_4|b_5|b_6|b_7|b_8|b_9|b_{10}|b_{11}|b_{12}|b_{13}|b_{14}|b_{15}|$

Si ponemos un ejemplo numérico, cuando enviamos a la función el valor decimal 45091 (0xB023), así serían la entrada y la salida:

Entrada	Salida
1011 0000 0010 0011	1100 0100 0000 1101

Resolveremos esto utilizando operaciones bit a bit en C, para lo cual vamos a utilizar una "máscara", la cual inicialmente definiremos como un 1 en la posición del bit menos significativo. En cada iteración del lazo lo que haremos es un AND bit a bit (operador **&**) de nuestro número inicial con la máscara y cuando el resultado de esta operación sea distinto de cero agregamos el bit menos significativo en el resultado con una operación OR bit a bit (operador **|**). Después vamos desplazando la máscara a la izquierda (operador **<<**) y el resultado también a la izquierda, para de esta forma obtener el número invertido. El código en C del programa sería el siguiente:

```
unsigned int rotar( unsigned int numero) {
    unsigned int resultado;
    unsigned int mascara;

    mascara = 1;
    resultado = 0;

    while (mascara != 0) {
        resultado <<= 1;
        if ((numero & mascara) != 0) resultado |= 1;
        mascara <<= 1;
    }
    return resultado;
}
```

Para entender mejor su funcionamiento, lo vemos en un ejemplo: Si quisiera mandar a ejecutar la función pasando como parámetro el número 61475, dentro del while se modifica primero el resultado, luego se realiza el AND del número con la máscara, se cambia nuevamente el resultado, y por último la máscara. Mostramos a continuación los valores que irían tomando en cada iteración (por razones de espacio se muestran sólo los primeros cuatro y los últimos cuatro bits de cada variable):

Numero	Resultado
1011 0000 0010 0011	1100 0100 0000 1101

Iteración	Resultado «= 1	Mascara	Numero & Mascara	Resultado = 1	Mascara «= 1
1	0000...0000	0000...0001	0000...0001	0000...0001	0000...0010
2	0000...0010	0000...0010	0000...0010	0000...0011	0000...0100
3	0000...0110	0000...0100	0000...0000	0000...0110	0000...1000
4	0000...1100	0000...1000	0000...0000	0000...1100	0000...0000
	...	...	...	...	...
13	0001...0000	0001...0000	0001...0000	0001...0001	0010...0000
14	0011...0000	0010...0010	0010...0000	0011...0011	0100...0000
15	0110...0000	0100...0110	0000...0000	0110...0110	1000...0000
16	1100...0000	1000...1100	1000...0000	1100...1101	0000...0000