Escuela Técnica Superior de Ingeniería Industrial y de Telecomunicación.
Grado en Ingeniería de Tecnologías de Telecomunicación. Curso 2021/22.
Electrónica Digital I.

Trabajo nº 1: Códigos Binarios.

Se valorará la corrección del trabajo así como la presentación del mismo. La presentación debe hacerse preferentemente por ordenador y entregarse mediante correo electrónico.

Incluir una referencia al tiempo utilizado para hacer todo el trabajo (resolución y presentación), y el trabajo realizado por cada miembro del grupo indicando el tanto por ciento del trabajo total y las tareas realizadas. Última fecha de entrega: 1-10-2021.

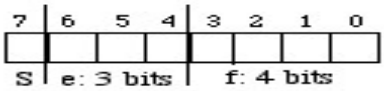
1º. Los números se pueden describir en un formato del tipo punto flotante. En sistemas binarios se suele utilizar un formato basado en la ecuación $(-1)^s * 2^{ex-bias} * (1.f)_2$, donde:

- S (1 bit): bit de signo del número.
- ex (M bits): corresponde al exponente un decimal entero calculado a partir de su descripción en binario.
- bias: constante que en decimal se calcula como $2^{M-1} - 1$. Permite tener exponentes positivos para definir números grandes, y exponentes negativos para definir números pequeños.
- f (N bits): parte fraccionaria del número binario, teniendo en cuenta que la parte entera es 1 y no se incluye en la definición, aunque se da por supuesta. En $(1.f)_2$ el bit más significativo de f es el bit de peso 2^{-1}, el siguiente el de peso 2^{-2}, etc, obteniendo en $(1.f)_{10}$ un número decimal entre 1 y 2.

Los bits se incluyen en una única palabra binaria de N+M+1 bits en el orden (s|ex|f). En el formato estándar IEEE de 32 bits (https://en.wikipedia.org/wiki/IEEE_754-1985, incluye codificaciones especiales para 0, ±∞, NaN, etc, que no se usan en este trabajo), M = 8 y N = 23.

En este ejercicio se quieren hacer operaciones de números descritos en binario en punto flotante. Para cada operando, los números de bits de ex (M) y f (N) son respectivamente (VER HOJA POR GRUPO) y la operación a realizar es (VER HOJA POR GRUPO). Los operandos de entrada se muestran como datos hexadecimales que codifican los bits de un número descrito en punto flotante. A partir de este formato, obtener los números decimales equivalentes y obtener el resultado decimal de las operaciones aritméticas que se indican. Finalmente, hay que expresar el resultado decimal en binario en coma flotante, y mediante un código hexadecimal que codifique sus bits (s|ex|f). Las operaciones se pueden realizar siguiendo estos pasos:

- Obtener los bits de cada operando a partir del dato hexadecimal (desagrupando cada dígito en 4 bits) tomando N+M+1 bits desde la derecha hacia la izquierda.
- Obtener los bits de s, ex y f, siendo s el bit más a la izquierda, f los N bits más a la derecha, y ex los siguientes M bits a la izquierda de F (siendo los bits de la derecha los bits de peso menos significativo y los de la izquierda los bits de peso más significativo). Por ejemplo para M = 3, N = 4 los bits quedarían así:



- Aplicar la fórmula $(-1)^s * 2^{ex-bias} * (1.f)_2$ para obtener el valor de los operandos en decimal sin redondear.
- Obtener el resultado $(R)_{10}$ de la operación en aritmética decimal (a mano o usando una calculadora) sin redondear.
- Calcular los valores de s, ex y f correspondientes a $(R)_{10}$ y codificarlos en binario situando los N+M+1 bits en el formato (s|ex|f). En el caso de que el resultado de una operación requiera más bits de los N bits que se puedan almacenar en f, realizar un redondeo por truncamiento, tomando sólo los N bits más significativos. Indicar en este caso el valor real decimal correspondiente al número en punto flotante resultante.
- Agrupar los bits de 4 en 4 de derecha a izquierda para codificar el resultado mediante un código hexadecimal.

2º. Además del complemento-2, la representación de números con signo también puede hacerse mediante un código de dígitos con signo, en el que cada dígito binario puede tomar tres valores: +1, 0 y -1. El valor +1 añade el peso positivo del dígito en binario natural (1, 2, 4, 8, 16, …) al valor final del número, el valor -1 añade el peso negativo, y el valor 0 no añade peso. Ejemplos:

(+1 0 -1 -1 0 +1) = +32 – 8 – 4 + 1 = +21 (-1 +1 0 0 +1 -1) = –32 + 16 + 2 – 1 = –15

Los códigos de dígitos con signo son redundantes ya que existen varias formas de representar un número. Por ejemplo 7 = 4 + 2 + 1 = 8 – 1 = 8 – 2 + 1 = ….

En uno de estos códigos cada dígito D_i se representa en binario por dos bits (S R), y el valor del dígito $D_i = S – R$: (1 0) es +1, (0 1) es –1; (0 0) y (1 1) son 0. Un número X de n bits en complemento-2 (X_{n-1} X_{n-2}… X_1 X_0) puede pasarse directamente a un código D de n dígitos con signo (D_{n-1} D_{n-2}… D_1 D_0) mediante la transformación:

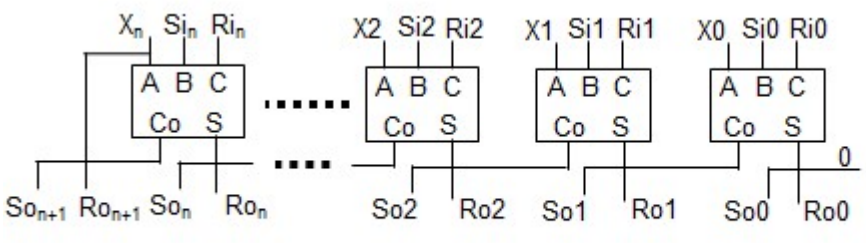
(X_{n-1} X_{n-2}… X_1 X_0) = (S_{n-1} R_{n-1}) (S_{n-2} R_{n-2}) … (S_1 R_1) (S_0 R_0), donde
(S_{n-1} R_{n-1}) (S_{n-2} R_{n-2}) … (S_1 R_1) (S_0 R_0) = (0 X_{n-1}) (X_{n-2} 0) … (X_1 0) (X_0 0)

También es posible realizar la transformación de un número D (S R) en código de dígitos con signo a un número X en complemento-2. En este caso, un número D de n dígitos se debe transformar a un número X de n+1 bits ya que, por ejemplo, si n = 4, -15 (-1 -1 -1 -1) se transforma en (10001) en complemento-2. La transformación puede hacerse extendiendo primero D a N+1 bits añadiendo un dígito más significativo D_n a valor 0 => (S_n R_n) = (0 0) o (1 1). A partir de D extendido se generan dos valores intermedios Y, Z de n+1 bits, y de ellos se obtiene X, en estos tres pasos:

- Para cada dígito D_i (i entre 0 y n), Y_i es 1 si el dígito es +1 o –1, y 0 si el dígito es 0.
- Para cada dígito D_i (i entre 0 y n), Z_i se obtiene del dígito menos significativo (D_0) al más significativo (D_n). $Z_0 = 0$; para el resto de los dígitos Z_i:
  - $Z_i = 1$ si el dígito D_{i-1} toma el valor -1.
  - $Z_i = 0$ si el dígito D_{i-1}toma el valor +1.
  - $Z_i = Z_{i-1}$ si el dígito D_{i-1} toma el valor 0.
- Para cada bit Xi (i entre 0 y n) es 1 si Y_i y Z_i son distintos, y Xi es 0 si Y_i y Z_i son iguales.

Los códigos de dígitos con signo se utilizan en multiplicaciones o sumas de varios operandos, reduciendo la propagación del acarreo. La suma de varios operandos dados inicialmente en complemento-2 se realiza siguiendo un esquema en el que el primer operando se transforma a

código dígitos con signo, y se suma sucesivamente con los siguientes operandos dados en complemento-2, generando en cada paso un nuevo resultado en código dígitos con signo. Después de sumar todos los operandos, el resultado final está dado en código dígitos con signo que se puede transformar en complemento-2 según el método explicado anteriormente. La suma de un número en código dígitos con signo D codificado por (S R) con un número en complemento-2 (X) se realiza mediante el siguiente circuito:



A	B	C	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

donde el módulo que se utiliza está basado en un sumador completo, con la diferencia que, para la generación de la salida Co, la entrada C opera al revés (1s como 0s, 0s como 1s) frente al funcionamiento del sumador normal.

Generar mediante la utilización de códigos dígitos con signo la suma del siguiente conjunto de M números: (VER HOJA POR GRUPO), descritos en complemento-2 en N bits (VER HOJA POR GRUPO). Seguir estos pasos (se recomienda comprobar los resultados intermedios en cada paso):

a) Describir los números en complemento-2 en N bits.
b) Transformar el primer número para describirlo en código de dígitos con signo de N dígitos.
c) Sumar el primer número en código dígitos con signo con el segundo en complemento-2 mediante el circuito de la figura para operandos de N dígitos. El resultado se obtiene en código de dígitos con signo de N+1 dígitos.
d) Sucesivamente sumar el resultado de la suma previa con el siguiente número en complemento-2 (extendiendo antes su signo para que tenga el mismo número de bits que dígitos con signo en el otro operando). En cada suma el circuito debe tener un bloque de cómputo más que en la suma anterior, y el resultado en código de dígitos con signo debe tener un dígito más. El resultado final deberá tener N+M-1 dígitos con signo.
e) Transformar el resultado final a su descripción en complemento-2 de N+M bits.
f) Comprobar que el resultado obtenido es correcto en las dos codificaciones (dígitos con signo y complemento-2).

3º. El código exp-Golomb es un código en el que los datos se codifican con diferente longitud de bits, que se utiliza en compresión de datos. Se puede utilizar sobre números enteros no negativos (positivos con 0), siendo muy adecuado para situaciones en las que los valores bajos son mucho más probables que los valores altos, como en distribuciones geométricas ($P(x) = \rho^x(1-\rho)$, para enteros $x \geq 0$ y $0 < \rho < 1$). El código exp-Golomb puede formarse para un número N mediante el siguiente algoritmo simple:

- Dado el número N obtener N+1 en binario con el menor número M de bits.
- Formar el código añadiendo de izquierda a derecha M-1 bits a 0, seguido de N+1 en binario.

a) Suponiendo el ordenamiento de las 27 letras en español por su porcentaje de aparición de mayor a menor, que se cita en wikipedia (aparecen ordenadas por dicho porcentaje), y codificando dicho orden (la letra más probable) de 0 en adelante, escribir el siguiente mensaje (VER HOJA POR GRUPO) en código exp-Golumb. Comparar el número de bits resultante con el producido por un código binario normal.
b) Suponiendo el mismo sistema de codificación, indicar el mensaje correspondiente a la siguiente secuencia de bits, leída de izquierda a derecha (VER HOJA POR GRUPO).
c) Suponiendo que cada carácter del alfabeto tuviese un tanto por uno Pr de probabilidad de aparición en distribución geométrica con ρ = (VER HOJA POR GRUPO), calcular aproximadamente el promedio P del número de bits por carácter del alfabeto. Suponiendo que la suma de probabilidad de todos los caracteres es aproximadamente 1.0 (valido para $\rho < 0.80$, aproximadamente) dicho promedio se puede calcular como:

$$P = \sum_{i=0}^{26} \Pr(i) \bullet (nº\ bits)_i$$