SPD 05/04/18

Signo magnitud

Complemento a 1

Complemento a 2

Los 3 tienen el bit de signo a la izquierda (0 = positivo, 1 = negativo).

Signo Magnitud

Hace falta saber cuantos bits se van a usar para el numero

Mayormente se usa 1 Byte, (8 bits).

10 = ,\_,\_,\_,\_,\_,\_,\_,\_,

,\_,0,0,0,1,0,1,0, (se deja el primero sin usar, para guardar el signo del numero)

El bit mas a la izquierda es el “bit de signo”. -----🡪 0 = numero positivo.

|---🡪 1 = numero negativo.

10 = ,0,0,0,0,1,0,1,0, -10 = ,1,0,0,0,1,0,1,0,

Con 8 bits se puede seleccionar desde el -127 al +127, pero el 0 esta dos veces (siendo positivo y negativo), por lo tanto, no es muy especifico

Complemento a 1

8 bits = ,\_,\_,\_,\_,\_,\_,\_,\_, con el primer bit siendo el bit de signo

0 0 0 0 1 0 1 0

10 = ,\_, ,\_, ,\_, ,\_, ,\_, ,\_, ,\_, ,\_,

El complemento a 1 para negar un numero, niega bit a bit

-10 = ,1, ,1, ,1, ,1, ,0, ,1, ,0, ,1,

Sigue con el mismo problema de tener 2 ceros

-128 64 32 16 8 4 2 1 (valores decimales,

-10 = 1 1 1 1 0 1 0 1 se pone el 128

Como negativo y se hace la suma)

Complemento a 2 (complemento a 1 + 1)

Se suma el bit cuando se pasa a negativo, en vez de sumar el decimal cuando se hace la cuenta.

-10= 1 1 1 1 0 1 0 1 (-10 Ca1)

+ 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 1 1 1 0 1 1 0 (-10 Ca2)

Se elimina el problema de los 2 ceros

-128 64 32 16 8 4 2 1

0 0 0 0 0 0 0 0 = 0

-128 64 32 16 8 4 2 1

1 1 1 1 1 1 1 1 = -1

-128 64 32 16 8 4 2 1

1 0 0 0 0 0 0 0 = -128 (al no tener el 0

repetido, se gana un numero negativo)

12/04/18

Representacion en punto flotante

Presicion simple (se usan 32 bits) [usamos este]

Presicion doble (se usan 64 bits)

Presicion ampliada (usa 80 bits)

\_ \_ \_ \_ \_ \_ \_ \_ \_

Bit 8 bits para representar exponente desplazado

De se usa el exponente, + 127 (por que? No hay por que)

Signo

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

23 bits para Mantisa (el numero)

3,700x10^4=37000=1001.0000.1000.1000=1,001.0000.1000.1000x2^15

0 1 0 0 0 1 1 1 0 (15 + 127)

\_ \_ \_ \_ \_ \_ \_ \_ \_

Bit 8 bits para representar exponente desplazado

De se usa el exponente, + 127 (por que? No hay por que)

Signo

0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

23 bits para Mantisa (el numero)

1,001.0000.1000.1000x2^15 se obvia el primero 1 ya que siempre va a ser 1

2^0 2^1 2^2 2^3 2^4 2^5 2^6 2^7 2^8 2^9 2^10 2^11

1 2 4 8 16 32 64 128 256 512 1024 2048

2^12 2^13 2^14 2^15 2^16 2^17 2^18 2^19 2^20 2^21 2^22 2^23

4096 8192 16384 32768 65536 131072

Paridad par o impar

Para saber si los msjs llegan incorrectamente o no (con la cantidad de 1 si es par o impar)

3) Ejercicios Paridad

Agregar bit de paridad a los siguientes números (paridad par)

1101 1 1101

1100 0 1100

1001 0 1001

1011 1 1011

1010 0 1010

1000 1 1000

Indicar cuales de los siguientes números contiene un error considerando que se trabaja con paridad impar

11100 v

11000 f

10001 f

11110 f

10011 v

Código de haming

Numero a transmitir

1 0 1 0

D1 D2 D3 D4 (digitos)

Para transmitir un numero de 4 cifras, voy a necesitar al menos 3 bits de paridad

P1 P2 P3 (paridad)

Los bits de paridad se ubican en las posiciones donde el bit es potencia de 2 (1, 2, 4, 8, 16, etc.)

Formula para saber cuandtos bits de paridad se necesitan:

2^P >= d + P + 1

P = Puntos de paridad

d = cantidad de digitos

\*Se ordena así:

b1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 P2 D1 P3 D2 D3 D4

1 0 1 0

Grupo 1 (P1)

Aquellos que tengan 1 en el bit de unidad (b1 “001”, b3 “011”, b5 “101”, b7 “111”) tiene que ser par (si estamos en paridad par)

b1 (P1) b3 b5 b7

1 1 0 0

Grupo 2 (P2)

Aquellos que tienen 1 en el bit de decena (b2 “010”, b3 “011”, b6 “110”, b7 “111”) tiene que ser par (si estamos en paridad par)

b2 (P2) b3 b6 b7

0 1 1 0

Grupo 3 (P3)

Aquellos que tienen 1 en el bit de centena (b4 “100”, b5 “101”, b6 “110”, b7 “111”) tiene que ser par (si estamos en paridad par)

b4 (P3) b5 b6 b7

1 0 1 0

\* Se ordena así:

b1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 P2 D1 P3 D2 D3 D4

1 0 1 1 0 1 0

Ejemplo 2 (decodificar, es erroneo)

b1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 P2 D1 P3 D2 D3 D4

1 0 1 1 1 1 0

Grupo 1 (P1) MAL (la cantidad de 1 es impar) = 1

Aquellos que tengan 1 en el bit de unidad (b1 “001”, b3 “011”, b5 “101”, b7 “111”) tiene que ser par (si estamos en paridad par)

b1 (P1) b3 b5 b7

1 1 1 0

Grupo 2 (P2) BIEN (la cantidad de 1 es par) = 0

Aquellos que tienen 1 en el bit de decena (b2 “010”, b3 “011”, b6 “110”, b7 “111”) tiene que ser par (si estamos en paridad par)

b2 (P2) b3 b6 b7

0 1 1 0

Grupo 3 (P3) MAL (la cantidad de 1 es impar) = 1

Aquellos que tienen 1 en el bit de centena (b4 “100”, b5 “101”, b6 “110”, b7 “111”) tiene que ser par (si estamos en paridad par)

b4 (P3) b5 b6 b7

1 1 1 0

1(G3) 0(G2) 1(G1) Se pone 1 si el grupo es erróneo y 0 si es correcto, y el numero en binario que te da es la posición del bit (en decimal)

4 2 1 = 4 + 1 = 5

1 0 1 = 5

Ejercicios Hamming

Codificar los siguientes números en Hamming (par)

1111

1001

1110

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

1 1 1 1 1 1 1

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

0 0 1 1 0 0 1

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

0 0 1 0 1 1 0

Codificar los siguientes números en Hamming (impar)

1111

1001

1110

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

0 0 1 0 1 1 1

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

1 1 1 0 0 0 1

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

1 1 1 1 1 1 0

Ejercicio par

1 1 0 1 1 0 0

G1 = 0

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

1 1 0 1 1 0 0

G2 = 1

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

1 1 0 1 1 0 0

G3 = 0

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

1 1 0 1 1 0 0

010 bit Erroneo = 2

1 0 0 1 1 0 0 CORRECTO

Mensaje 4 bits (datos)

Hamming (par)

Arduino

GND = ground (tierra)

TX = Transmitir

RX = Recibir

Pines: 2 3 4 5 6 7 8

P1 P2 D1 P3 D2 D3 D4

1101

B1 b2 b3 b4 b5 b6 b7

001 010 011 100 101 110 111

P1 p2 d1 p3 d2 d3 d4

1 0 1 0 1 0 1