Facultad de Ingeniería - UNLP

E1301 – E0301 Introducción a los Sistemas Lógicos y Digitales

Curso 2025 - Trabajo Entregable 0

Isabella Valenzi Octavio Tomás - 03665/8

Rodriguez Fausto - 03766/2

Cladera Blas - 03694/3

Patané Baltazar - 03773/1

Introducción

En el siguiente informe se analizará la precisión y el rango de valores que se pueden obtener al representar la ecuación de una recta utilizando el formato de punto fijo. Además este estudio será acompañado por sus respectivos códigos desarrollados en lenguaje C para la resolución de los problemas.

Sabemos que la ecuación de una recta es: y = mx + b

En este caso queremos explorar la precisión y rango de valores que pueden obtenerse al trabajar con la ecuación de la recta con representación en punto fijo.

Se adopta para m una representación Q(0,15) y para b una representación Q(7,8) mientras que x e y se van a representar en variables de punto fijo de 32 bits con signo.

Desarrollo

a) ¿Cuál es el rango de representación y la resolución de los valores de m?

- **Dado que se representa en formato Q(0,15) esto significa:**

  1 bit de signo, 0 bits para la parte entera y 15 bits para la parte fraccionaria.

- **Valores representables:**

  El valor más *negativo* se obtiene cuando todos los bits están en 1 excepto el de signo, que es 1. → 1000000000000000_2

  El valor más *positivo* se obtiene cuando el bit de signo es 0 y todos los demás bits son 1. → 0111111111111111_2

  Rango: { -2^0 ; $\sum_{x=1}^{15} 2^{-x}$ }

  Rango: { -1 ; 0,999969482421 }

- **Resolución:**

  La resolución es el *valor del bit menos significativo.*

  $2^{-15} = 0{,}000030517578125$

b) ¿Cuál es el rango de representaciones y la resolución de los valores de b?

- **Dado que se representa en formato Q(7,8) esto significa:**

  1 bit de signo, 7 bits para la parte entera y 8 bits para la parte fraccionaria.

- **Valores representables:**

  El número más negativo se obtiene cuando el bit de signo es 1 y todos los demás bits son 0. → 10000000.00000000_2

  El valor más *positivo* se obtiene cuando el bit de signo es 0 y todos los demás bits son 1. → 01111111.11111111_2

  Rango: $\{ -2^7 ; \sum_{x=-8}^{6} 2^x \}$

  Rango: { -128 ; 127,99609 }

- **Resolución:**

  La resolución es el *valor del bit menos significativo.*

  $2^{-8} = 0,00396$

c) Elija una representación Q(c ,d) para x y para y tal que tanto m como b puedan representarse usando la misma representación sin pérdida de cifras significativas. Indique cuáles serían los desplazamientos y/o máscaras que debe utilizar para ubicar las variables.

Para garantizar que tanto *m* como *b* puedan representarse en 32 bits con la misma configuración que *x* o *y* sin pérdida de información, propusimos dos soluciones: la primera fue usar Q(16,15) para respetar la resolución máxima de ambas representaciones, y la otra fue usar Q(14,17) con la misma idea de la anterior pero para priorizar la resolución y la representación de los reales. Luego de dialogar con el grupo, decidimos usar Q(16,15).

d) ¿Cuál va a ser el rango y la resolución de los valores que pueden representarse en x y en y?

- **Dado que se representa en formato Q(16,15) esto significa:**

  1 bit de signo, 16 bits para la parte entera y 15 bits para la parte fraccionaria.

- **Valores representables:**

  El número más negativo se obtiene cuando el bit de signo es 1 y todos los demás bits son 0. → $1000000000000000.000000000000000_2$

  El valor más *positivo* se obtiene cuando el bit de signo es 0 y todos los demás bits son 1. → $0111111111111111.111111111111111_2$

Rango: { -2^{16}; $2^{16} - 2^{-15}$}

Rango: { -65536 ; 65535,999969482421875 }

- **Resolución:**

  La resolución es el *valor del bit menos significativo.*

  2^{-15} = *0,000030517578125*

e) De acuerdo con la ecuación de la recta y tomando casos límites respecto a los valores de las constantes m y b , por ejemplo el mínimo valor negativo o el máximo valor positivo, a qué valores debería acotarse x para que usando la representación elegida los valores de y obtenidos no produzcan overflow.

- **Restricción de X:**

  Para la búsqueda del valor más chico al que X debe acotarse, usamos la ecuación de la recta Y = mX + b, y la reemplazamos con el menor valor de m y el máximo valor de b.

  Límite inferior → $(-1)*X* + (2^7 - 2^{-8}) <= 2^{16} - 2^{-15}$ → X >= -65408,00388

  Para la búsqueda del valor más grande al que X debe acotarse, usamos la ecuación de la recta Y = mX + b, y la reemplazamos con el menor valor de m y el menor valor de b.

  Límite superior → $(-1)*X* + (-2^7) >= -2^{16}$ → X <= 65408

  Rango: { -65408,003875732421875 *; 65408}*

f) Escriba un programa en lenguaje de programación C que permita el ingreso de un valor expresado en notación decimal ±eee.ffff y lo convierta a representación en punto fijo de 16 bits Q(7,8). El programa debe validar la entrada y determinar si el número ingresado está dentro del rango representable. La salida debe expresarse en Hexadecimal (0xHHHH)

El programa comienza leyendo el número que ingresa el usuario y lo guarda como String. Luego inicia la transformación para pasar el número a *Q(7, 8)*: se fija si el formato está bien, separa signo, parte entera y parte fraccionaria y se fija si están bien los rangos

Al cumplir los requisitos, la parte entera no hay que transformarla si no que hay que correrla hasta la posición que le corresponde. Por otro lado, la fraccionaria se analiza para ver qué bits se prenden o cuáles se apagan.

Al finalizar, se juntan todas las partes en una variable (si es negativa se pasa a *CA2*) y se informa el número pero en su formato hexadecimal.

g) Escriba otro programa que dado un número expresado en punto fijo de 16 bits Q(7,8) exprese el valor decimal equivalente eee.ffff. La entrada debe ser un número hexadecimal expresado en la forma 0xHHHH que debe ser validada.

El programa solicita al usuario que ingrese un número en formato hexadecimal. Si el número ingresado no es válido, muestra un mensaje de error y finaliza la ejecución.

Una vez que se tiene un valor correcto, convierte el número hexadecimal a su equivalente en binario y a partir de esa nueva cifra, lo transforma en una notación decimal con un formato específico *(±eee.ffff)*.

Al finalizar, muestra el número final en el formato decimal establecido.

h) Escriba un programa que permita el ingreso de los valores de m , b y x en forma decimal y muestre su representación en punto fijo en formato hexadecimal validando la entrada como en los puntos anteriores. Luego realizando todas las operaciones en punto fijo con las representaciones adoptadas calcule el valor de la ordenada y y lo muestre en punto fijo en forma hexadecimal y en decimal.

El programa inicia solicitando al usuario ingresar tres valores: *m, b y x*, donde cada uno se introduce como una cadena de texto. A partir de estas se determina la parte entera y fraccionaria del número, las cuales se convierten en enteros diferentes para crear una única representación en punto fijo de *16 o 32 bits*, según corresponda. Si alguno de los valores no es válido, el programa muestra un mensaje de error y finaliza.

Luego, se convierten los valores *m, b y x* en su equivalente hexadecimal y se calcula la ordenada y a través de la ecuación de la recta $y = mx + b$. Es importante aclarar, que a pesar de que tanto *X* como *Y* están expresados en *Q(16,15)*, se utilizan las restricciones para *X* del punto *e)* para que no genere overflow. Al finalizar, el resultado de estas operaciones se muestra al usuario en su forma hexadecimal y decimal.

Conclusión

A través del estudio de las representaciones *Q(0,15), Q(7,8) y Q(16,15)*, se determinó que es posible operar sin pérdida de información al adoptar una representación común para todas las variables.

Se calcularon los rangos y resoluciones de cada formato, verificando cómo afectan la precisión y los valores máximos y mínimos representables. También se establecieron restricciones sobre *X* para evitar desbordamientos en *Y*, garantizando cálculos dentro de los límites del sistema.

Además, se desarrollaron programas en *C* (https://github.com/Faus142005/IDL) que permiten la conversión entre valores decimales y punto fijo, validando entradas y asegurando que los cálculos se realicen correctamente sin el uso de variables de punto flotante.