Grupo 4

Caycho José[1] , Sahua Jaafar[2], Junco Frank[3], Limache Ebert[4], Oyola Renzo[5], Rojas Carlos[6]

Universidad Nacional de Ingeniería

2021

Índice de la presentación

Solución 4

Representación Decimal	Representación Signo-magnitud	Representación Complemento a dos
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
+0	0000	0000

Solución 4

Representación Decimal	Representación Signo-magnitud	Representación Complemento a dos
-0	0000	0000
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001
-8	1000	1000

Solución 4

Ejemplos de Signo-magnitud:
Representamos el número 6 (decimal). Procedemos a:
1. Tomar nota del signo del número reducido o simplificado 6, que siendo positivo, llevará como bit de signo un 0;
2. Realizar la conversión: el valor absoluto de 6 es $|6| = 6$. Que en binario es: 110_2;
3. Colocar todo junto, el número 6 en binario con formato de Signo y Magnitud es: 0110_2. Donde el 0 en el bit más significativo indica un número positivo.

Solución 4

Representamos el número -6 (decimal). Procedemos a:
1. Tomar nota del signo del número reducido o simplificado -6, que siendo negativo, llevará como bit de signo un 1;
2. Realizar la conversión: el valor absoluto de -6 es $|-6| = 6$. Que en binario es: 110_2;
3. Colocar todo junto, el número 6 en binario con formato de Signo y Magnitud es: 1110_2. Donde el 1 en el bit más significativo indica un número negativo.

Solución 4

Ejemplos de Complemento a dos:
Representamos el número 5 (decimal). Procedemos a:
1. Convertirlo a base 2: 5 $= 101_2$
2. Agregamos la cantidad de ceros necesaria para que complete la longitud de 4 bits: 5 $= 0101_2$

Representamos el número -5 (decimal). Procedemos a:
1. Convertir a base 2 su valor absoluto: $|-5| =$ 5 $= 101_2$
2. Agregamos la cantidad de ceros necesaria para que complete la longitud de 4 bits: 5 $= 0101_2$
3. Invertimos todos los bits del número $(0 \leftarrow 1)$ y $(1 \leftarrow 0)$: 1010_2
4. Sumo 1 al resultado obtenido anteriormente, entonces: -5 $=$ $(1010 + 1)_2 = (1011)_2$

Solución 8

Si tenemos $\beta = 2, t = 3, L = -2$ y $U = 2$
a)Determine el intervalo donde se representa los números reales.
Solución:
los números reales x se encuentran en un determinado intervalo correspondiente a sus respectivos elementos y a estos son β, t, L, U

$$\beta^{L-1} \leq |x| \leq \beta^{U}(1 - \beta^{-t}) \tag{1}$$

reemplazando en (1) para $\beta = 2, t = 3, L = -2$ y $U = 2$

$$2^{-3} \leq |x| \leq 2^{2}(1 - 2^{-3})$$
$$\frac{1}{8} \leq |x| \leq \frac{7}{2}$$

Solución 8

c) Determine los números de máquina que contiene dicho intervalo.

Solución:

Como $L = -2$ y $U = 2$ los números se expresan de la siguiente manera

-2	-1	0	1
$(0.100_2) \times 2^{-2}$	$(0.100_2) \times 2^{-1}$	$(0.100_2) \times 2^{0}$	$(0.100_2) \times 2^{1}$
$(0.101_2) \times 2^{-2}$	$(0.101_2) \times 2^{-1}$	$(0.101_2) \times 2^{0}$	$(0.101_2) \times 2^{1}$
$(0.110_2) \times 2^{-2}$	$(0.110_2) \times 2^{-1}$	$(0.110_2) \times 2^{0}$	$(0.110_2) \times 2^{1}$
$(0.111_2) \times 2^{-2}$	$(0.111_2) \times 2^{-1}$	$(0.111_2) \times 2^{0}$	$(0.111_2) \times 2^{1}$

Solución 8

2
$(0.100_2) \times 2^2$
$(0.101_2) \times 2^2$
$(0.110_2) \times 2^2$
$(0.111_2) \times 2^2$

Solución 8

-2	-1	0	1	2
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{1}$	$\frac{2}{1}$
$\frac{5}{32}$	$\frac{5}{16}$	$\frac{5}{8}$	$\frac{5}{4}$	$\frac{5}{2}$
$\frac{3}{16}$	$\frac{3}{8}$	$\frac{3}{4}$	$\frac{3}{2}$	$\frac{3}{1}$
$\frac{7}{32}$	$\frac{7}{16}$	$\frac{7}{8}$	$\frac{7}{4}$	$\frac{7}{2}$ height

Finalmente los numeros de maquinas son los mostrados en la tabla con signo mas y menos ademas del cero.

Solución 12

En principio tenemos el algoritmo en el lenguaje de programación Python, al ejecutarlo obtenemos el siguiente resultado:

Solución 12

En la imagen del resultado anterior observamos que:

$p = 53$
$B = 2.0$

Podemos interpretar esto como que la iteración se ha realizado 53 veces o que el último valor que observamos de A es igual a 2^{53}. Esto quiere decir que se puede ejecutar el algoritmo 53 veces con un error igual a 0 en ((A+1)-A)-1. A su vez observamos que en la segunda condición se cumple que ((A+B)-A)-B es diferente de 0, por lo tanto, se le suma 1.0 a B resultando de esto el 2.0.

Solución 12

Como dice el enunciado ejecuté el código en otra computadora, sin embargo, obtuve el mismo resultado.

Solución 12

Así que decidí realizar el mismo algoritmo en el lenguaje C++, aquí es donde obtuve un resultado diferente:

Solución 16

Se procede a modificar la fórmula de la siguiente manera,

multiplicando por el conjugado.

$$2^{n+1}[\sqrt{1+2^{-n}x_n}-1]\times\frac{\sqrt{1+2^{-n}x_n}+1}{\sqrt{1+2^{-n}x_n}+1}$$

$$\frac{2^{n+1}[1+2^{-n}x_n-1]}{\sqrt{1+2^{-n}x_n}+1}$$

Se logra eliminar la resta del 1, con lo que no quedan sustracciones

$$\frac{2^{n+1}[2^{-n}x_n]}{\sqrt{1+2^{-n}x_n}+1}$$

La fórmula modificada para que no haya pérdida de dígitos significativos quedaría de la siguiente manera

$$x_{n+1} = \frac{2x_n}{\sqrt{1+2^{-n}x_n}+1}$$

Entonces se llevara a cabo una prueba en Python tomando $x_0 = 4$
Se tendría

$$ln(4+1) = 1.6094379124341003$$

```
import math
print("Log(4+1)")
print(math.log(5))
Log(4+1)
1.6094379124341003
```

Entonces ese es el valor al que deberia converger

Fórmula inicial

En primer lugar se probará para la formula sin modificar.

```
print("Formula inicial")
v1=4
for n in range(0,50):
v1=pow(2,n+1)*(pow(1+pow(2,-n)*v1,0.5)-1)
print(v1)
```

Para 40 iteraciones vemos que:

$$v1 = 1.609375$$

y en 50 iteraciones

$$v1 = 1.5$$

Fórmula modificada

Finalmente se usara la fórmula modificada para no perder cifras significativas print("Formula Modificada")

```
v2=4
for m in range(0,50):
v2=2*v2/(pow(1+pow(2,-m)*v2,0.5)+1)
print(v2)
```

Se tiene finalmente en 50 iteraciones el valor de

$$v2 = 1.609437912434102$$

Finalmente se hace un cálculo de los errores relativos para ambas fórmulas

```
print("Error Relativo 1")
er1=abs(v-v1)/v
print(er1)
print("Error Relativo 2")
er2=abs(v-v2)/v
er2=er2
print(er2)
```

Resultando:
Error Relativo 1
0.06799759816058223
Error Relativo 2
1.1037125605632736e-15
Vemos como el error luego de modificar es notablemente inferior.
Y concluimos del problema que la modificación sí redujo la pérdida de cifras significativas.

Solución 20

Del desarrollo en serie de:

$$e^x - 1 = \sum_{i=1}^{n} \frac{x^i}{i!}$$

Determine el error mínimo, y el número de términos necesarios hasta llegar a dicho error mínimo.

Solución 20

Desarrollo

Teorema

Si x **e** y son números binarios de puntos flotante positivos tal que:

$$2^{-q} \leq 1 - \frac{y}{x} \leq 2^{-p}$$

entonces la sustracción $x - y$ perderá de p a q bits significativos

Entonces del teorema,como e^x y 1 son positivos,entonces por lo menos se perderá un bit en la diferencia $e^x - 1$

$$2^{-1} \leq 1 - \frac{1}{e^x}$$

$$\rightarrow 0.7 \leq x$$

Ahora para $0 \leq x < 0.7$:

Consideramos números de punto flotante de 32 bits, entonces la precisión asociada sera 2^{-23}

Ahora hallaremos el número de iteraciones para obtener dicho error mínimo

Solución 20

Recordando E_n:

$$\frac{f^{(n+1)}(a).x^{n+1}}{(n+1)!}$$

donde: $0 < a < x \quad \wedge \quad 0 \leq x < 0.7$

Entonces:

$$\frac{f^{(n+1)}(a).x^{n+1}}{(n+1)!} \leq \frac{1}{(n+1)!}e^a 0.7^{n+1} \leq \frac{1}{(n+1)!}3^{0.7}0.7^{n+1} < 2^{-23}$$

Obtenemos el numero de iteraciones mínimo probando valores, obteniendo como respuesta $n = 9$