Organización de Computadoras

Explicación práctica – 3/6 – Repaso Punto Flotante

Curso 2020

Prof. JorgeRunco

Mantisa BSS 12 bits fraccionaria, normalizada, 6 bits exponente en exceso – Representar 188,99

- 1) 188 = 10111100
- 0,99 x 2 = 1,98 → 1
- 0,98 x 2 = 1,96 → 1
- 0,96 x 2 = 1,92 → 1
- 0,92 x 2 = 1,84 → 1
- 0,84 x 2 = 1,68 → 1
- → 10111100,11111
- 2) 188,99 → 10111100,11111 x 2^0 = 0, 1011110011111 x 2^{+8}
- → 0,101111001111 x 2^{+8}
- Sombreado en amarillo están los 12 bits de la mantisa

La mantisa tiene 12 bits. Después de esta cantidad los bits se descartan

- 3) E= 8 en Ex2
- +8 en Ca2 = 001000
- Sumo el ex2 100000
- +8 en ex2 = 101000 Otra manera → 8+32 = 40 (BSS) (OJO)
- $0{,}101111001111 \times 2^{101000}$ → 101000 101111001111 = Exp Man

¿Qué número representamos finalmente?

- $0{,}101111001111 \times 2^8=(2^{-1} +2^{-3} +2^{-4} + 2^{-5} +2^{-6} + 2^{-9} +2^{-10} + 2^{-11} +2^{-12}) \times 2^8$
- $=2^{+7} +2^{+5} +2^{+4} + 2^{+3} +2^{+2} + 2^{-1} +2^{-2} + 2^{-3} +2^{-4} =$ 128+32+16+8+4+0,5+0,25+0,125+0,0625 = 188,9375 ****
- Error absoluto = 188,99 – 188,9375 = 0,0525

¿Es el más cercano?

- $0{,}101111001111 \times 2^8$ Vamos a buscar el número siguiente
- $(0{,}101111001111 + 0{,}000000000001) \times 2^8$
- $\ 0{,}101111001111 \times 2^8$
- $+\ \underline{0{,}000000000001 \times 2^8}$
- $\ 0{,}101111010000 \times 2^8$
- ¿Qué número representamos finalmente?
- $0{,}101111010000 \times 2^8=(2^{-1} +2^{-3} +2^{-4} + 2^{-5} +2^{-6} + 2^{-8}) \times 2^8 =$
- $2^{+7} +2^{+5} +2^{+4} + 2^{+3} +2^{+2} + 2^0$ = 128+32+16+8+4+1 = 189
- Error absoluto = 189 – 188,99 = 0,01 Esta es la representación más cercana.

Ahora igual que el anterior pero mantisa en BCS

-) + 188 = 0 10111100
- 0,99 x 2 = 1,98 → 1
- 0,98 x 2 = 1,96 → 1
- 0,96 x 2 = 1,92 → 1
- 0,92 x 2 = 1,84 → 1
- 0,84 x 2 = 1,68 → 1
- → 0 10111100,11111
- 0 10111100,11111 x 2^0 → 0 0,10111100111~~11~~ x 2^8 Sombreado en amarillo están los 12 bits de la mantisa.
- 0 101000 10111100111 = S E M así queda almacenado en memoria

- 0 0,10111100111 x 2^8=+(2^{-1} +2^{-3} +2^{-4} + 2^{-5} +2^{-6} + 2^{-9} +2^{-10} + 2^{-11}) x 2^8
- =2^{+7} +2^{+5} +2^{+4} + 2^{+3} +2^{+2} + 2^{-1} +2^{-2} + 2^{-3} = 128+32+16+8+4+0,5+0,25+0,125 = 188,875
- Error absoluto = 188,99 – 188,875 = 0,115 ****

- 0 0,10111100111 x 2^8 Vamos a buscar el número siguiente
- (0,10111100111 + 0,00000000001) x 2^8
- 0,10111100111 x 2^8
- \+ 0,00000000001 x 2^8
- 0,10111101000 x 2^8
- ¿Qué número representamos finalmente?
- 0 0,10111101000 x 2^8= + $(2^{-1} +2^{-3} +2^{-4} + 2^{-5} +2^{-6} + 2^{-8})$ x 2^8 =
- $2^{+7} +2^{+5} +2^{+4} + 2^{+3} +2^{+2} + 2^0$ = 128+32+16+8+4+1 = 189
- Error absoluto = 189 – 188,99 = 0,01 Esta es la representación más cercana.

Igual que el anterior con bit implícito

- Al tener bit implícito tenemos un bit más. Ahora en memoria no se almacena 0,1.
- Sigo teniendo 12 lugares en memoria: uno para el signo (S) y 11 para la mantisa (M) → 0 0,1M en celeste hay 12 bits + S = 13 en total.

Mantisa BCS 12 bits fraccionaria, normalizada, 6 bits exponente en exceso, bit implícito – Representar 188,99

- 1) Escribir el número en el sistema de la M
- + 188 =0 10111100
- 0,99 x 2 = 1,98 → 1
- 0,98 x 2 = 1,96 → 1
- 0,96 x 2 = 1,92 → 1
- 0,92 x 2 = 1,84 → 1
- 0,84 x 2 = 1,68 → 1
- 0,68 x 2 = 1,36 → 1
- → 0 10111100,111111

- 2) → 0 10111100,111111 x 2^0
- 0   0,101111001111~~11~~ x 2^8
- 0 0,101111001111 x 2^8
- 8 en Ca2 =   001000
- Le sumo el exceso   +100000   (exceso 32)
- 8 en exceso (ex2) =   101000
- En memoria se almacena S E M = 0 101000 01111001111
- Escriba el número = 0   0,1 01111001111

Mantisa BCS 12 bits fraccionaria, normalizada, 6 bits exponente en exceso, bit implícito – Representar - 0,015625

- 0,015625 x 2 = 0,03125 → 0
- 0,03125 x 2 = 0,0625 → 0
- 0,0625 x 2 = 0,125 → 0
- 0,125 x 2 = 0,25 → 0
- 0,25 x 2 = 0,5 → 0
- 0,5 x 2 = 1,0 → 1
- 0,0 x 2 =0
- 1 0,0000010 → es – 0,015625 en BCS

- 1 0,0000010 x 2^0 = 1 0,100000000000 x 2^{-5}
- Exponente
- +5 en Ca2 = 000101
- -5 Ca2 = 111011
- Sumo exceso + 100000
- 011011 - 5 en exceso
- 1 0,100000000000 x 2 ^{011011} → S E M = 1 011011 000000000000
- En amarillo están los 12 bits de la mantisa del enunciado. Cuando agregamos 0,1 hay uno más (13 en total) por el bit implícito.

Exponente

- $0{,}4 \times 10^{0} = 0{,}4 \times 1 = 0{,}4$
- $0{,}04 \times 10^{+1} = 0{,}04 \times 10 = 0{,}4$
- $0{,}0004 \times 10^{+3} = 0{,}0004 \times 1000 = 0{,}4$
- $00040{,}0 \times 10^{+3\ -5} = 40{,}0 \times 10^{-2} = 40{,}0 \times 0{,}01 = 0{,}4$
- $4{,}0 \times 10^{+3\ -4} = 4{,}0 \times 10^{-1} = 4{,}0 \times 0{,}1 =0{,}4$

Ej. $0{,}000010100 \times 2^{-8}$

- $0{,}000010100 \times 2^{-8} = 0{,}0010100 \times 2^{-8-2} = 0{,}0010100 \times 2^{-10}$
- $0{,}000000101 \times 2^{-10+4} = 0{,}000000101 \times 2^{-6}$

Escribir -13,5 en IEEE754 SP (E→otro exceso) M → BCS, Normalizada 1,.....

- 1) 1 1101,10000….0000 = -13,5 en BCS
- 2) 1 1101,10000….0000 x 2^0 = 1 1,10110000….0000 x 2^{+3}
- 3) +3 en ca2 = 00000011
- sumo exceso 01111111
- 10000010 +3 en exceso 127
- S E M = 1 10000010 10110000000…000 → No se almacena 1,

0 10000010 10000000000…000 ¿número? Está en IEEE754

- 0 10000010 10000000000…000
- M= 1,10000000….0 = $2^0 + 2^{-1}$ = 1 + 0,5 =1,5
- E = 10000010 está en exceso 127 → le tengo que restar el exceso
- 10000010
  - 01111111
- 00000011 nuestro exponente es +3
- + 1,5 x 2^{+3} =+ 1,5 x 8 = +12