John Gangemi
u68714612
CDA 4205
9/25/2014

HOMEWORK #4

3.12

* SL = shift left & SR = shift right
* Mcand = multiplicand & Mplier = multiplier

Iteration	Step	Multiplier(62)	Multiplicand(12)	Product
0	initial	110 010	000 000 001 010	000 000 000 000
1	No Op	110 010	000 000 001 010	000 000 000 000
	SL Mcand	110 010	000 000 010 100	000 000 000 000
	SR Mplier	011 001	000 000 010 100	000 000 000 000
2	Add	011 001	000 000 010 100	000 000 010 100
	SL Mcand	011 001	000 000 101 000	000 000 010 100
	SR Mplier	001 100	000 000 101 000	000 000 010 100
3	No Op	001 100	000 000 101 000	000 000 010 100
	SL Mcand	001 100	000 001 010 000	000 000 010 100
	SR Mplier	000 110	000 001 010 000	000 000 010 100
4	No Op	000 110	000 001 010 000	000 000 010 100
	SL Mcand	000 110	000 010 100 000	000 000 010 100
	SR Mplier	000 011	000 010 100 000	000 000 010 100
5	Add	000 011	000 010 100 000	000 010 110 100
	SL Mcand	000 011	000 101 000 000	000 010 110 100
	SR Mplier	000 001	000 101 000 000	000 010 110 100
6	Add	000 001	000 101 000 000	000 111 110 100
	SL Mcand	000 001	001 010 000 000	000 111 110 100
	SR Mplier	000 000	001 010 000 000	000 111 110 100

(000 111 110 100) base 2 == (764) base 8

3.19

* SL = shift left & SR = shift right
* Rem = leftmost 6 bits of Remainder/Quotient register

Iteration	Step	Divisor	Remainder/Quotient
0	initial	010 101	000 001 001 010
	SL R/Q	010 101	000 010 010 100
1	Rem -= Divisor	010 101	101 100 010 100
	Rem < 0:		
	Rem += Divisor	010 101	000 010 010 100
	SL R/Q	010 101	000 100 101 000
	LSB = 0	010 101	000 100 101 000
2	Rem -= Divisor	010 101	101 111 101 000
	Rem < 0:		
	Rem += Divisor	010 101	000 100 101 000
	SL R/Q	010 101	001 001 010 000
	LSB = 0	010 101	001 001 010 000
3	Rem -= Divisor	010 101	110 100 010 000
	Rem < 0:		
	Rem += Divisor	010 101	001 001 010 000
	SL R/Q	010 101	010 010 100 000
	LSB = 0	010 101	010 010 100 000
4	Rem -= Divisor	010 101	111 101 100 000
	Rem < 0:		
	Rem += Divisor	010 101	010 010 100 000
	SL R/Q	010 101	100 101 000 000
	LSB = 0	010 101	100 101 000 000
5	Rem -= Divisor	010 101	010 000 000 000
	Rem >= 0:		
	SL R/Q	010 101	100 000 000 000
	LSB = 1	010 101	100 000 000 001
6	Rem -= Divisor	010 101	001 011 000 001
	Rem >= 0:		
	SL R/Q	010 101	010 110 000 010
	LSB = 1	010 101	010 110 000 011
7	SR Rem	010 101	001 011 000 011

For unsigned 6-bit *integer* division the result is the content of the quotient, thus the result is (000 011) base 2 or (3) base 10

3.23

Write the binary representation of the decimal number 63.25 assuming the IEEE 754 single precision format ...

General form: (-1)^sign * (1 + fraction) * 2^(exponent + bias)
Bias = 127

1) convert to binary:

<u>Integer</u>	<u>Fractional</u>
63/2 = 31, r=1	0.25 * 2 = 0.5, [0] *MSB*
31/2 = 15, r=1	0.50 * 2 = 1.0, [1]
15/2 = 7, r=1	0.0 * 2 = 0.0, [0]
7/2 = 3, r=1	
3/2 = 1, r=1	0.25 = (010) base 2
½ = 0, r=1 *MSB*	

   63 = (111111) base 2

   => (111111.010) base 2

2) normalize:
   1.11111010 x 2^5

   => exponent = 5

3) biased exponent [8-bits]:
   exponent + bias = 5 + 127 = 132 = (1000 0100) base 2

4) mantissa [23-bits]:
   (1111 1010 0000 0000 0000 000) base 2

=> Sign	Exponent	Fraction
0	1000 0100	1111 1010 0000 0000 0000 000

3.24

Write the binary representation of the decimal number 63.25 assuming the IEEE 754 double precision format …

General form: (-1)^sign * (1 + fraction) * 2^(exponent + bias)
Bias = 1023

1) convert to binary:

<u>Integer</u>	<u>Fractional</u>
63/2 = 31, r=1	0.25 * 2 = 0.5, [0] *MSB*
31/2 = 15, r=1	0.50 * 2 = 1.0, [1]
15/2 = 7, r=1	0.0 * 2 = 0.0, [0]
7/2 = 3, r=1	
3/2 = 1, r=1	0.25 = (010) base 2
½ = 0, r=1 *MSB*	

   63 = (111111) base 2

   => (111111.010) base 2

2) normalize:
   1.11111010 x 2^5

   => exponent = 5

3) biased exponent [11-bits]:
   exponent + bias = 5 + 1023 = 1028 = (100 0000 0100) base 2

4) mantissa [52-bits]:
   (1111 1010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000) base 2

=> Sign	Exponent	Fraction
0	100 0000 0100	1111 1010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000