3 Solutions

Solution 3.1

3.1.1

a.	3716
b.	6041

3.1.2

a.	3716
b.	1467

3.1.3

a.	1660	1660
b.	2165	–117

3.1.4

a.	6374
b.	753

3.1.5

a.	7504 (–3504)
b.	7777 (–3777)

3.1.6

a.	111000100000
b.	100011110101

The attraction is that each octal digit contains one of 8 different characters (0–7). Since with 3 binary bits you can represent 8 different patterns, in octal each digit requires exactly 3 binary bits. You can write down the conversion directly.

Solution 3.2

3.2.1

a.	7B75
b.	6D95

3.2.2

a.	7B75
b.	6D95

3.2.3

a.	5190	5190
b.	9312	9312

3.2.4

a.	8CA4
b.	5730

3.2.5

a.	FA00
b.	5730

3.2.6

a.	1100001101010010
b.	0101111011010100

The attraction is that each hex digit contains one of 16 different characters (0–9, A–E). Since with 4 binary bits you can represent 16 different patterns, in hex each digit requires exactly 4 binary bits. And bytes are by definition 8 bits long, so two hex digits are all that are required to represent the contents of 1 byte.

Solution 3.3

3.3.1

a.	Underflow (–39)
b.	Neither (63)

3.3.2

a.	Overflow (result = –215, which does not fit into an SM 8-bit format)
b.	Neither (65)

3.3.3

a.	Neither (39)
b.	Overflow (result = –179, which does not fit into an SM 8-bit format)

3.3.4

a.	15 – 117 = –102
b.	–105 – 42 = –128 (–147)

3.3.5

a.	15 + 117 = 127 (132)
b.	–105 + 42 = –63

3.3.6

a.	15 + 139 = 154
b.	151 + 214 = 255 (365)

Solution 3.4

3.4.1

a. 62×12

Step	Action	Multiplier	Multiplicand	Product
0	Initial Vals	001 010	000 000 110 010	000 000 000 000
1	lsb = 0, no op	001 010	000 000 110 010	000 000 000 000
	Lshift Mcand	001 010	000 001 100 100	000 000 000 000
	Rshift Mplier	000 101	000 001 100 100	000 000 000 000
2	Prod = Prod + Mcand	000 101	000 001 100 100	000 001 100 100
	Lshift Mcand	000 101	000 011 001 000	000 001 100 100
	Rshift Mplier	000 010	000 011 001 000	000 001 100 100
3	lsb = 0, no op	000 010	000 011 001 000	000 001 100 100
	Lshift Mcand	000 010	000 110 010 000	000 001 100 100
	Rshift Mplier	000 001	000 110 010 000	000 001 100 100
4	Prod = Prod + Mcand	000 001	000 110 010 000	000 111 110 100
	Lshift Mcand	000 001	001 100 100 000	000 111 110 100
	Rshift Mplier	000 000	001 100 100 000	000 111 110 100
5	lsb = 0, no op	000 000	001 100 100 000	000 111 110 100
	Lshift Mcand	000 000	011 001 000 000	000 111 110 100
	Rshift Mplier	000 000	011 001 000 000	000 111 110 100
6	lsb = 0, no op	000 000	110 010 000 000	000 111 110 100
	Lshift Mcand	000 000	110 010 000 000	000 111 110 100
	Rshift Mplier	000 000	110 010 000 000	000 111 110 100

b. 35×26

Step	Action	Multiplier	Multiplicand	Product
0	Initial Vals	010 110	000 000 011 101	000 000 000 000
1	lsb = 0, no op	010 110	000 000 011 101	000 000 000 000
	Lshift Mcand	010 110	000 000 111 010	000 000 000 000
	Rshift Mplier	001 011	000 000 111 010	000 000 000 000
2	Prod = Prod + Mcand	001 011	000 000 111 010	000 000 111 010
	Lshift Mcand	001 011	000 001 110 100	000 000 111 010
	Rshift Mplier	000 101	000 001 110 100	000 000 111 010

Step	Action	Multiplier	Multiplicand	Product
3	Prod = Prod + Mcand	000 101	000 001 110 100	000 010 101 110
	Lshift Mcand	000 101	000 011 101 000	000 010 101 110
	Rshift Mplier	000 010	000 011 101 000	000 010 101 110
4	lsb = 0, no op	000 010	000 011 101 000	000 010 101 110
	Lshift Mcand	000 010	000 111 010 000	000 010 101 110
	Rshift Mplier	000 001	000 111 010 000	000 010 101 110
5	Prod = Prod + Mcand	000 001	000 111 010 000	001 001 111 110
	Lshift Mcand	000 001	001 110 100 000	001 001 111 110
	Rshift Mplier	000 000	001 110 100 000	001 001 111 110
6	lsb = 0, no op	000 000	001 110 100 000	001 001 111 110
	Lshift Mcand	000 000	011 101 000 000	001 001 111 110
	Rshift Mplier	000 000	011 101 000 000	001 001 111 110

3.4.2

a. 62×12

Step	Action	Multiplicand	Product/Multiplier
0	Initial Vals	110 010	000 000 001 010
1	lsb = 0, no op	110 010	000 000 001 010
	Rshift Product	110 010	000 000 000 101
2	Prod = Prod + Mcand	110 010	110 010 000 101
	Rshift Mplier	110 010	011 001 000 010
3	lsb = 0, no op	110 010	011 001 000 010
	Rshift Mplier	110 010	001 100 100 001
4	Prod = Prod + Mcand	110 010	111 110 100 001
	Rshift Mplier	110 010	011 111 010 000
5	lsb = 0, no op	110 010	011 111 010 000
	Rshift Mplier	110 010	001 111 101 000
6	lsb = 0, no op	110 010	001 111 101 000
	Rshift Mplier	110 010	000 111 110 100

b. 35×26

Step	Action	Multiplicand	Product/Multiplier
0	Initial Vals	011 101	000 000 010 110
1	lsb = 0, no op	011 101	000 000 010 110
	Rshift Mplier	011 101	000 000 001 011
2	Prod = Prod + Mcand	011 101	011 101 001 011
	Rshift Product	011 101	001 110 100 101
3	Prod = Prod + Mcand	011 101	101 011 100 101
	Rshift Mplier	011 101	010 101 110 010
4	lsb = 0, no op	011 101	010 101 110 010
	Rshift Mplier	011 101	001 010 111 001
5	Prod = Prod + Mcand	011 101	100 111 111 001
	Rshift Mplier	011 101	010 011 111 100
6	lsb = 0, no op	011 101	010 011 111 100
	Rshift Mplier	011 101	001 001 111 110

3.4.3 No solution provided

3.4.4

a. $41 \times 33 = 4033$

Step	Action	Mplier	Multiplicand	Product	Sign
0	Initial Values	011 011	000 000 100 001	000 000 000 000	0
	Multiplier.sign XOR Multiplicand.sign (0 XOR 1)				1
	Make positive	011 011	000 000 000 001	000 000 000 000	1
1	Prod = Prod + Mcand	011 011	000 000 000 001	000 000 000 001	1
	Lshift Mcand	011 011	000 000 000 010	000 000 000 001	1
	Rshift Mplier	001 101	000 000 000 010	000 000 000 001	1
2	Prod = Prod + Mcand	001 101	000 000 000 010	000 000 000 011	1
	Lshift Mcand	001 101	000 000 000 100	000 000 000 011	1
	Rshift Mplier	000 110	000 000 000 100	000 000 000 011	1
3	lsb = 0, no op	000 110	000 000 000 100	000 000 000 011	1
	Lshift Mcand	000 110	000 000 001 000	000 000 000 011	1
	Rshift Mplier	000 011	000 000 001 000	000 000 000 011	1

Step	Action	Mplier	Multiplicand	Product	Sign
4	Prod = Prod + Mcand	000 011	000 000 001 000	000 000 001 011	1
	Lshift Mcand	000 011	000 000 010 000	000 000 001 011	1
	Rshift Mplier	000 001	000 000 010 000	000 000 001 011	1
5	Prod = Prod + Mcand	000 001	000 000 010 000	000 000 011 011	1
	Lshift Mcand	000 001	000 000 100 000	000 000 011 011	1
	Rshift Mplier	000 000	000 000 100 000	000 000 011 011	1
6	lsb = 0, no op	000 000	000 000 100 000	000 000 011 011	1
	Lshift Mcand	000 000	000 001 000 000	000 000 011 011	1
	Rshift Mplier	000 000	000 001 000 000	000 000 011 011	1
7	Prod msb = sign	000 000	000 001 000 000	100 000 011 011	1

b. $60 \times 26 = 4540$

Step	Action	Mplier	Multiplicand	Product	Sign
0	Initial Values	010 110	000 000 110 000	000 000 000 000	0
	Multiplier.sign XOR Multiplicand.sign (0 XOR 1)				1
	Make positive	010 110	000 000 010 000	000 000 000 000	1
1	lsb = 0, no op	010 110	000 000 010 000	000 000 000 000	1
	Lshift Mcand	010 110	000 000 100 000	000 000 000 000	1
	Rshift Mplier	001 011	000 000 100 000	000 000 000 000	1
2	Prod = Prod + Mcand	001 011	000 000 100 000	000 000 100 000	1
	Lshift Mcand	001 011	000 001 000 000	000 000 100 000	1
	Rshift Mplier	000 101	000 001 000 000	000 000 100 000	1
3	Prod = Prod + Mcand	000 101	000 001 000 000	000 001 100 000	1
	Lshift Mcand	000 101	000 010 000 000	000 001 100 000	1
	Rshift Mplier	000 010	000 010 000 000	000 001 100 000	1
4	lsb = 0, no op	000 010	000 010 000 000	000 001 100 000	1
	Lshift Mcand	000 010	000 100 000 000	000 001 100 000	1
	Rshift Mplier	000 001	000 100 000 000	000 001 100 000	1
5	Prod = Prod + Mcand	000 001	000 100 000 000	000 101 100 000	1
	Lshift Mcand	000 001	001 000 000 000	000 101 100 000	1
	Rshift Mplier	000 000	001 000 000 000	000 101 100 000	1

Step	Action	Mplier	Multiplicand	Product	Sign
6	lsb = 0, no op	000 000	001 000 000 000	000 101 100 000	1
	Lshift Mcand	000 000	010 000 000 000	000 101 100 000	1
	Rshift Mplier	000 000	010 000 000 000	000 101 100 000	1
7	Prod msb = sign	000 000	010 000 000 000	100 101 100 000	1

3.4.5

a. $41 \times 33 = -37 \times 33 = -1505$ (6273)

Step	Action	Multiplicand	Product/Multiplier
0	Initial Vals	100 001	0 000 000 011 011
1	Prod = Prod + Mcand	100 001	1 100 001 011 011
	Rshift Mplier	100 001	1 110 000 101 101
2	Prod = Prod + Mcand	100 001	1 010 001 101 101
	Rshift Product	100 001	1 101 000 110 110
3	lsb = 0, no op	100 001	1 101 000 110 110
	Rshift Mplier	100 001	1 110 100 011 011
4	Prod = Prod + Mcand	100 001	1 010 101 011 011
	Rshift Mplier	100 001	1 101 010 101 101
5	Prod = Prod + Mcand	100 001	1 001 011 101 101
	Rshift Mplier	100 001	1 100 101 110 110
6	lsb = 0, no op	100 001	1 100 101 110 110
	Rshift Mplier	100 001	1 110 010 111 011

b. $60 \times 26 = -20 \times 26 = -540$ (7240)

Step	Action	Multiplicand	Product/Multiplier
0	Initial Vals	110 000	0 000 000 010 110
1	lsb = 0, no op	110 000	0 000 000 010 110
	Rshift Mplier	110 000	0 000 000 001 011
2	Prod = Prod + Mcand	110 000	1 110 000 001 011
	Rshift Product	110 000	1 111 000 000 101
3	Prod = Prod + Mcand	110 000	1 101 000 000 101
	Rshift Mplier	110 000	1 110 100 000 010
4	lsb = 0, no op	110 000	1 110 100 000 010
	Rshift Mplier	110 000	1 111 010 000 001

Step	Action	Multiplicand	Product/Multiplier
5	Prod = Prod + Mcand	110 000	1 101 010 000 001
	Rshift Mplier	110 000	1 110 101 000 000
6	lsb = 0, no op	110 000	1 110 101 000 000
	Rshift Mplier	110 000	1 111 010 100 000

3.4.6 No solution provided

Solution 3.5

3.5.1 For hardware, it takes 1 cycle to do the add, 1 cycle to do the shift, and 1 cycle to decide if we are done. So the loop takes (3 × A) cycles, with each cycle being B time units long.

For a software implementation, it takes 1 cycle to decide what to add, 1 cycle to do the add, 1 cycle to do each shift, and 1 cycle to decide if we are done. So the loop takes (5 × A) cycles, with each cycle being B time units long.

a.	(3 × 8) × 4tu = 96 time units for hardware (5 × 8) × 4tu = 160 time units for software
b.	(3 × 64) × 8tu = 1536 time units for hardware (5 × 64) × 8tu = 2560 time units for software

3.5.2 It takes B time units to get through an adder, and there will be A – 1 adders.

a.	Word is 8 bits wide, requiring 7 adders. 7 × 4tu = 28 time units.
b.	Word is 64 bits wide, requiring 63 adders. 63 × 8tu = 504 time units.

3.5.3 It takes B time units to get through an adder, and the adders are arranged in a tree structure. It will require log2(A) levels.

a.	8-bit wide word requires 7 adders in 3 levels. 3 × 4tu = 12 time units.
b.	64-bit word requires 63 adders in 6 levels. 6 × 8tu = 48 time units.

Solution 3.6

3.6.1

a.	0x33 × 0x55 = 0x10EF. 0x33 = 51, and 51 = 32 + 16 + 2 + 1. We can shift 0x55 left 5 places (0xAA0), then add 0x55 shifted left 4 places (0x550), then add 0x55 shifted left once (0xAA), then add 0x55. 0xAA0 + 0x550 + 0xAA + 0x55 = 0x10EF. 3 shifts, 3 adds. (Could also use 0x55, which is 64 + 16 + 4 + 1, and shift 0x33 left 6 times, add to it 0x33 shifted left 4 times, add to that 0x33 shifted left 2 times, and add to that 0x33. Same number of shifts and adds.)

b.	0x8A × 0xED = 0x7FC2 0x8A = 128 + 8 + 2, 0xED = 128 + 64 + 32 + 8 + 4 + 1. Best way is to shift 0xED left 7 places (0x7680), then add to that 0xED shifted left 3 places (0x768), and then add 0xED shifted left 1 place (0x1DA). 3 shifts, 2 adds.

3.6.2

a.	0x33 × 0x55 = 0x10EF. 0x33 = 51, and 51 = 32 + 16 + 2 + 1. We can shift 0x55 left 5 places (0xAA0), then add 0x55 shifted left 4 places (0x550), then add 0x55 shifted left once (0xAA), then add 0x55. 0xAA0 + 0x550 + 0xAA + 0x55 = 0x10EF. 3 shifts, 3 adds. (Could also use 0x55, which is 64 + 16 + 4 + 1, and shift 0x33 left 6 times, add to it 0x33 shifted left 4 times, add to that 0x33 shifted left 2 times, and add to that 0x33. Same number of shifts and adds.)
b.	0x8A × 0xED = –0x0A × –0x6D = 0x442 0x0A = 8 + 2, 0x6D = 64 + 32 + 8 + 4 + 1. Best way is to shift 0x6D left 3 places (0x368), then add to that 0x6D shifted left 1 place (0xDA). 2 shifts, 1 add.

3.6.3 No solution provided

3.6.4 Quoting the Wikipedia entry directly:

Booth's algorithm involves repeatedly adding one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let x and y be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in x and y.

1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).
   a. A: Fill the most significant (leftmost) bits with the value of x. Fill the remaining (y + 1) bits with zeros.
   b. S: Fill the most significant bits with the value of (–x) in two's complement notation. Fill the remaining (y + 1) bits with zeros.
   c. P: Fill the most significant x bits with zeros. To the right of this, append the value of y. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of P.
   a. If they are 01, find the value of P + A. Ignore any overflow.
   b. If they are 10, find the value of P + S. Ignore any overflow.
   c. If they are 00 or 11, do nothing. Use P directly in the next step.
3. Arithmetically shift the value obtained in the previous step by a single place to the right. Let P now equal this new value.
4. Repeat steps 2 and 3 until they have been done y times.
5. Drop the least significant (rightmost) bit from P. This is the product of x and y.

3.6.5

a. $0xF6 \times 0x7F = -0xA \times 0x7F = -10 \times 127 = -1270 = 0xFB0A$

Action	Multiplicand	Product/Multiplier
Initial Vals	1111 0110	0000 0000 0111 1111 0
10, subtract	1111 0110	0000 1010 0111 1111 0
shift	1111 0110	0000 0101 0011 1111 1
11, nop	1111 0110	0000 0101 0011 1111 1
shift	1111 0110	0000 0010 1001 1111 1
11, nop	1111 0110	0000 0010 1001 1111 1
shift	1111 0110	0000 0001 0100 1111 1
11, nop	1111 0110	0000 0001 0100 1111 1
shift	1111 0110	0000 0000 1010 0111 1
11, nop	1111 0110	0000 0000 1010 0111 1
shift	1111 0110	0000 0000 0101 0011 1
11, nop	1111 0110	0000 0000 0101 0011 1
shift	1111 0110	0000 0000 0010 1001 1
11, nop	1111 0110	0000 0000 0010 1001 1
shift	1111 0110	0000 0000 0001 0100 1
01, add	1111 0110	1111 0110 0001 0100 1
shift	1111 0110	1111 1011 0000 1010 0

b. $0x08 \times 0x55 = 0x2A8$

Action	Multiplicand	Product/Multiplier
Initial Vals	0000 1000	0000 0000 0101 0101 0
10, subtract	0000 1000	1111 1000 0101 0101 0
shift	0000 1000	1111 1100 0010 1010 1
01, add	0000 1000	0000 0100 0010 1010 1
shift	0000 1000	0000 0010 0001 0101 0
10, subtract	0000 1000	1111 1010 0001 0101 0
shift	0000 1000	1111 1101 0000 1010 1
01, add	0000 1000	0000 0101 0000 1010 1
shift	0000 1000	0000 0010 1000 0101 0
10, subtract	0000 1000	1111 1010 1000 0101 0
shift	0000 1000	1111 1101 0100 0010 1
01, add	0000 1000	0000 0101 0100 0010 1
shift	0000 1000	0000 0010 1010 0001 1

Action	Multiplicand	Product/Multiplier
10, subtract	0000 1000	1111 1010 1010 0001 0
shift	0000 1000	1111 1101 0101 0000 1
01, add	0000 1000	0000 0101 0101 0000 1
shift	0000 1000	0000 0010 1010 1000 1

3.6.6 No solution provided

Solution 3.7

3.7.1

a. 74/21 = 3 remainder 9

Step	Action	Quotient	Divisor	Remainder
0	Initial Vals	000 000	010 001 000 000	000 000 111 100
1	Rem = Rem – Div	000 000	010 001 000 000	101 111 111 100
	Rem < 0, R + D, Q<<	000 000	010 001 000 000	000 000 111 100
	Rshift Div	000 000	001 000 100 000	000 000 111 100
2	Rem = Rem – Div	000 000	001 000 100 000	111 000 011 100
	Rem < 0, R + D, Q<<	000 000	001 000 100 000	000 000 111 100
	Rshift Div	000 000	000 100 010 000	000 000 111 100
3	Rem = Rem – Div	000 000	000 100 010 000	111 100 101 100
	Rem < 0, R + D, Q<<	000 000	000 100 010 000	000 000 111 100
	Rshift Div	000 000	000 010 001 000	000 000 111 100
4	Rem = Rem – Div	000 000	000 010 001 000	111 110 110 100
	Rem < 0, R + D, Q<<	000 000	000 010 001 000	000 000 111 100
	Rshift Div	000 000	000 001 000 100	000 000 111 100
5	Rem = Rem – Div	000 000	000 001 000 100	111 111 111 000
	Rem < 0, R + D, Q<<	000 000	000 001 000 100	000 000 111 100
	Rshift Div	000 000	000 000 100 010	000 000 111 100
6	Rem = Rem – Div	000 000	000 000 100 010	000 000 011 010
	Rem > 0, Q << 1	000 001	000 000 100 010	000 000 011 010
	Rshift Div	000 001	000 000 010 001	000 000 011 010
7	Rem = Rem – Div	000 001	000 000 010 001	000 000 001 001
	Rem > 0, Q << 1	000 011	000 000 010 001	000 000 001 001
	Rshift Div	000 011	000 000 001 000	000 000 001 001

b. 76/52 = 1 remainder 24

Step	Action	Quotient	Divisor	Remainder
0	Initial Vals	000 000	101 010 000 000	000 000 111 110
1	Rem = Rem – Div	000 000	101 010 000 000	101 001 000 010
	Rem < 0, R + D, Q<<	000 000	101 010 000 000	000 000 111 110
	Rshift Div	000 000	010 101 000 000	000 000 111 110
2	Rem = Rem – Div	000 000	010 101 000 000	101 011 111 110
	Rem < 0, R + D, Q<<	000 000	010 101 000 000	000 000 111 110
	Rshift Div	000 000	001 010 100 000	000 000 111 110
3	Rem = Rem – Div	000 000	001 010 100 000	110 110 011 110
	Rem < 0, R + D, Q<<	000 000	001 010 100 000	000 000 111 110
	Rshift Div	000 000	000 101 010 000	000 000 111 110
4	Rem = Rem – Div	000 000	000 101 010 000	111 011 101 110
	Rem < 0, R + D, Q<<	000 000	000 101 010 000	000 000 111 110
	Rshift Div	000 000	000 010 101 000	000 000 111 110
5	Rem = Rem – Div	000 000	000 010 101 000	111 110 010 110
	Rem < 0, R + D, Q<<	000 000	000 010 101 000	000 000 111 110
	Rshift Div	000 000	000 001 010 100	000 000 111 110
6	Rem = Rem – Div	000 000	000 001 010 100	111 111 101 101
	Rem < 0, R = D, Q<<	000 000	000 001 010 100	000 000 111 110
	Rshift Div	000 000	000 000 101 010	000 000 111 110
7	Rem = Rem – Div	000 000	000 000 101 010	000 000 010 100
	Rem > 0, Q << 1	000 001	000 000 101 010	000 000 010 100
	Rshift Div	000 001	000 000 010 101	000 000 010 100

3.7.2 In these solutions a 1 or a 0 was added to the quotient if the remainder was greater than or equal to 0. However, an equally valid solution is to shift in a 1 or 0, but if you do this you must do a compensating right shift of the remainder (only the remainder, not the entire remainder/quotient combination) after the last step.

a. 74/21 = 3 remainder 11

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	010 001	000 000 111 100
1	R<<	010 001	000 001 111 000
	Rem = Rem – Div	010 001	111 000 111 000
	Rem < 0, R + D	010 001	000 001 111 000

Step	Action	Divisor	Remainder/Quotient
2	R<<	010 001	000 011 110 000
	Rem = Rem – Div	010 001	110 010 110 000
	Rem < 0, R + D	010 001	000 011 110 000
3	R<<	010 001	000 111 100 000
	Rem = Rem – Div	010 001	110 110 110 000
	Rem < 0, R + D	010 001	000 111 100 000
4	R<<	010 001	001 111 000 000
	Rem = Rem – Div	010 001	111 110 000 000
	Rem < 0, R + D	010 001	001 111 000 000
5	R<<	010 001	011 110 000 000
	Rem = Rem – Div	010 001	111 110 000 000
	Rem > 0, R0 = 1	010 001	001 101 000 001
6	R<<	010 001	011 010 000 010
	Rem = Rem – Div	010 001	001 001 000 010
	Rem > 0, R0 = 1	010 001	001 001 000 011

b. 76/52 = 1 remainder 24

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	101 010	000 000 111 110
1	R<<	101 010	000 001 111 100
	Rem = Rem – Div	101 010	101 001 111 100
	Rem < 0, R + D	101 010	000 001 111 100
2	R<<	101 010	000 011 111 000
	Rem = Rem – Div	101 010	100 111 111 000
	Rem < 0, R + D	101 010	000 011 111 000
3	R<<	101 010	000 111 110 000
	Rem = Rem – Div	101 010	100 011 110 000
	Rem < 0, R + D	101 010	000 111 110 000
4	R<<	101 010	001 111 100 000
	Rem = Rem – Div	101 010	100 101 100 000
	Rem < 0, R + D	101 010	001 111 100 000

Step	Action	Divisor	Remainder/Quotient
5	R<<	101 010	011 111 000 000
	Rem = Rem – Div	101 010	110 101 000 000
	Rem < 0, R + D	101 010	011 111 000 000
6	R<<	101 010	111 110 000 000
	Rem = Rem – Div	101 010	010 100 000 000
	Rem > 0, R0 = 1	101 010	010 100 000 001

3.7.3 No solution provided

3.7.4

a. 72/07 = 3 remainder 5: Dividend negative

Sign of Quotient = (Sign bit of Divisor) XOR (Sign bit of Dividend) = negative
Sign of Remainder = Sign of Dividend = negative

Step	Action	Quotient	Divisor	Remainder
0	Initial Vals	000 000	000 111 000 000	000 000 011 010
1	Rem = Rem – Div	000 000	000 111 000 000	111 001 011 010
	Rem < 0, R + D, Q<<	000 000	000 111 000 000	000 000 011 010
	Rshift Div	000 000	000 011 100 000	000 000 011 010
2	Rem = Rem – Div	000 000	000 011 100 000	111 100 111 010
	Rem < 0, R + D, Q<<	000 000	000 011 100 000	000 000 011 010
	Rshift Div	000 000	000 001 110 000	000 000 011 010
3	Rem = Rem – Div	000 000	000 001 110 000	111 110 101 010
	Rem < 0, R + D, Q<<	000 000	000 001 110 000	000 000 011 010
	Rshift Div	000 000	000 000 111 000	000 000 011 010
4	Rem = Rem – Div	000 000	000 000 111 000	111 111 100 010
	Rem < 0, R + D, Q<<	000 000	000 000 111 000	000 000 011 010
	Rshift Div	000 000	000 000 011 100	000 000 011 010
5	Rem = Rem – Div	000 000	000 000 011 100	111 111 111 110
	Rem < 0, R + D, Q<<	000 000	000 000 011 100	000 000 011 010
	Rshift Div	000 000	000 000 001 110	000 000 011 010
6	Rem = Rem – Div	000 000	000 000 001 110	000 000 001 100
	Rem > 0, Q << 1	000 001	000 000 001 110	000 000 001 100
	Rshift Div	000 001	000 000 000 111	000 000 001 100

Step	Action	Quotient	Divisor	Remainder
7	Rem = Rem – Div	000 001	000 000 000 111	000 000 000 101
	Rem < 0, Q << 1	000 011	000 000 000 111	000 000 000 101
	Rshift Div	000 011	000 000 000 011	000 000 000 101
8	Set sign bits	100 011	000 000 000 011	100 000 000 101

b. 75/44 = 7 remainder 1: Dividend negative

Sign of Quotient = (Sign bit of Divisor) XOR (Sign bit of Dividend) = positive
Sign of Remainder = Sign of Dividend = negative

Step	Action	Quotient	Divisor	Remainder
0	Initial Vals	000 000	000 100 000 000	000 000 011 101
1	Rem = Rem – Div	000 000	000 100 000 000	111 100 011 101
	Rem < 0, R + D, Q<<	000 000	000 100 000 000	000 000 011 101
	Rshift Div	000 000	000 010 000 000	000 000 011 101
2	Rem = Rem – Div	000 000	000 010 000 000	111 110 011 101
	Rem < 0, R + D, Q<<	000 000	000 010 000 000	000 000 011 101
	Rshift Div	000 000	000 001 000 000	000 000 011 101
3	Rem = Rem – Div	000 000	000 001 000 000	111 111 011 101
	Rem < 0, R + D, Q<<	000 000	000 001 000 000	000 000 011 101
	Rshift Div	000 000	000 000 100 000	000 000 011 101
4	Rem = Rem – Div	000 000	000 000 100 000	111 111 111 101
	Rem < 0, R + D, Q<<	000 000	000 000 100 000	000 000 011 101
	Rshift Div	000 000	000 000 010 000	000 000 011 101
5	Rem = Rem – Div	000 000	000 000 010 000	000 000 001 101
	Rem > 0, Q << 1	000 001	000 000 010 000	000 000 001 101
	Rshift Div	000 001	000 000 001 000	000 000 001 101
6	Rem = Rem – Div	000 001	000 000 001 000	000 000 000 101
	Rem > 0, Q << 1	000 011	000 000 001 000	000 000 000 101
	Rshift Div	000 011	000 000 000 100	000 000 000 101
7	Rem = Rem – Div	000 011	000 000 000 100	000 000 000 001
	Rem > 0, Q << 1	000 111	000 000 000 100	000 000 000 001
	Rshift Div	000 111	000 000 000 010	000 000 000 001
8	Set sign bits	000 111	000 000 000 010	100 000 000 001

3.7.5

a. 72/07 = 3 remainder 5: Dividend negative

Sign of Quotient = (Sign bit of Divisor) XOR (Sign bit of Dividend) = negative
Sign of Remainder = Sign of Dividend = negative

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	000 111	000 000 011 010
1	R<<	000 111	000 000 110 100
	Rem = Rem – Div	000 111	111 001 110 100
	Rem < 0, R + D	000 111	000 000 110 100
2	R<<	000 111	000 001 101 000
	Rem = Rem – Div	000 111	111 010 101 000
	Rem < 0, R + D	000 111	000 001 101 000
3	R<<	000 111	000 011 010 000
	Rem = Rem – Div	000 111	111 100 010 000
	Rem < 0, R + D	000 111	000 011 010 000
4	R<<	000 111	000 110 100 000
	Rem = Rem – Div	000 111	111 111 100 000
	Rem < 0, R + D	000 111	000 110 100 000
5	R<<	000 111	001 101 000 000
	Rem = Rem – Div	000 111	000 110 000 000
	Rem > 0, R0 = 1	000 111	000 110 000 001
6	R<<	000 111	001 100 000 010
	Rem = Rem – Div	000 111	000 101 000 010
	Rem > 0, R0 = 1	000 111	000 101 000 011
7	Adjust signs	000 111	100 101 100 011 (Q = –3, Rem = –5)

b. 75/44 = 7 remainder 1: Dividend negative

Sign of Quotient = (Sign bit of Divisor) XOR (Sign bit of Dividend) = positive
Sign of Remainder = Sign of Dividend = negative

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	000 100	000 000 011 101
1	R<<	000 100	000 000 111 010
	Rem = Rem – Div	000 100	111 100 111 010
	Rem < 0, R + D	000 100	000 000 111 010

Step	Action	Divisor	Remainder/Quotient
2	R<<	000 100	000 001 110 100
	Rem = Rem – Div	000 100	111 101 110 100
	Rem < 0, R + D	000 100	000 001 110 100
3	R<<	000 100	000 011 101 000
	Rem = Rem – Div	000 100	111 111 101 000
	Rem < 0, R + D	000 100	000 011 101 000
4	R<<	000 100	000 111 010 000
	Rem = Rem – Div	000 100	000 011 010 000
	Rem > 0, R0 = 1	000 100	000 011 010 001
5	R<<	000 100	000 110 100 010
	Rem = Rem – Div	000 100	000 010 100 010
	Rem > 0,R0 = 1	000 100	000 010 100 011
6	R<<	000 100	000 101 000 110
	Rem = Rem – Div	000 100	000 001 000 110
	Rem > 0, R0 = 1	000 100	000 001 000 111
7	Adjust signs	000 100	100 001 000 111 (Q = 7, Rem = –1)

3.7.6 No solution provided

Solution 3.8

3.8.1 In these solutions a 1 will be shifted into the quotient and a compensating right shift of the remainder will be performed. This is the alternate approach mentioned in Solution Solution 3.7.2: In these solutions a 1 or a 0 was added to the quotient if the remainder was greater than or equal to 0..

a. 26/05 = 4 remainder 2

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	000 101	000 000 010 110
	R<<	000 101	000 000 101 100
	Rem = Rem – Div	000 101	111 011 101 100
1	Rem < 0, Q << 0, Addnext	000 101	110 111 011 000
	Rem = Rem + Div	000 101	111 100 011 000
2	Rem < 0, Q << 0, Addnext	000 101	111 000 110 000
	Rem = Rem + Div	000 101	111 101 110 000

Step	Action	Divisor	Remainder/Quotient
3	Rem < 0, Q << 0, Addnext	000 101	111 011 100 000
	Rem = Rem + Div	000 101	000 000 100 000
4	Rem > = 0, Q << 1, Sub	000 101	000 001 000 001
	Rem = Rem – Div	000 101	111 100 000 001
5	Rem < 0, Q << 0, Add	000 101	111 000 000 010
	Rem = Rem + Div	000 101	111 101 000 010
6	Rem < 0, Q << 0, Add	000 101	111 010 000 100
	Rem = Rem + Div	000 101	111 111 000 100
7	Rem < 0, Rem = Rem + Div	000 101	000 100 000 100
	Shift Rem >> 1	000 101	000 010 000 100 (Q = 4, Rem = 2)

b. 37/15 = 2 remainder 5

Step	Action	Divisor	Remainder/Quotient
0	Initial Vals	001 101	000 000 011 111
	R<<	001 101	000 000 111 110
	Rem = Rem – Div	001 101	110 011 111 110
1	Rem < 0, Q << 0, Addnext	001 101	100 111 111 100
	Rem = Rem + Div	001 101	110 100 111 100
2	Rem < 0, Q << 0, Addnext	001 101	101 001 111 000
	Rem = Rem + Div	001 101	110 110 111 000
3	Rem < 0, Q << 0, Addnext	001 101	101 101 110 000
	Rem = Rem + Div	001 101	111 010 110 000
4	Rem < 0, Q << 0, Addnext	001 101	110 101 100 000
	Rem = Rem + Div	001 101	000 010 100 000
5	Rem > 0, Q << 1, Subnext	001 101	000 101 000 001
	Rem = Rem – Div	001 101	111 000 000 001
6	Rem < 0, Q << 0, Addnext	001 101	110 000 000 010
	Rem = Rem + Div	001 101	111 101 000 010
7	Rem < 0, Rem = Rem + Div	001 101	001 010 000 010
	Shift Rem >> 1	001 101	000 101 000 010 (Q = 2, Rem = 5)

3.8.2 No solution provided

3.8.3 No solution provided

3.8.4

a. 27/6 = 3 remainder 5

Step	Action	Quotient	Temp	Divisor	Remainder
0	Initial Vals	000000	000000 000000	000110 000000	000000 010111
1	Temp = Rem – Div	000000	111010 010111	000110 000000	000000 010111
	Temp < 0, Q << 0	000000	111010 010111	000110 000000	000000 010111
	Rshift Div	000000	111010 010111	000011 000000	000000 010111
2	Temp = Rem – Div	000000	111101 010111	000011 000000	000000 010111
	Temp < 0, Q << 0	000000	111101 010111	000011 000000	000000 010111
	Rshift Div	000000	111101 010111	000001 100000	000000 010111
3	Temp = Rem – Div	000000	111111 110111	000001 100000	000000 010111
	Temp < 0, Q << 0	000000	111111 110111	000001 100000	000000 010111
	Rshift Div	000000	111111 110111	000000 110000	000000 010111
4	Temp = Rem – Div	000000	111111 100111	000000 110000	000000 010111
	Temp < 0, Q << 0	000000	111111 100111	000000 110000	000000 010111
	Rshift Div	000000	111111 100111	000000 011000	000000 010111
5	Temp = Rem – Div	000000	111111 111111	000000 011000	000000 010111
	Temp < 0, Q << 0	000000	111111 111111	000000 011000	000000 010111
	Rshift Div	000000	111111 111111	000000 001100	000000 010111
6	Temp = Rem – Div	000000	000000 001011	000000 001100	000000 010111
	T > 0, Q << 1, R = T	000001	000000 001011	000000 001100	000000 001011
	Rshift Div	000001	000000 001011	000000 000110	000000 001011
7	Temp = Rem – Div	000001	000000 000101	000000 000110	000000 001011
	T > 0, Q << 1, R = T	000011	000000 000101	000000 000110	000000 000101
	Rshift Div	000011	000000 000101	000000 000011	000000 000101

b. 54/12 = 4 remainder 4

Step	Action	Quotient	Temp	Divisor	Remainder
0	Initial Vals	000000	000000 000000	001010 000000	000000 101100
1	Temp = Rem – Div	000000	110110 101100	001010 000000	000000 101100
	Temp < 0, Q << 0	000000	110110 101100	001010 000000	000000 101100
	Rshift Div	000000	110110 101100	000101 000000	000000 101100
2	Temp = Rem – Div	000000	111011 101100	000101 000000	000000 101100
	Temp < 0, Q << 0	000000	111011 101100	000101 000000	000000 101100
	Rshift Div	000000	111011 101100	000010 100000	000000 101100

Step	Action	Quotient	Temp	Divisor	Remainder
3	Temp = Rem – Div	000000	111110 001100	000010 100000	000000 101100
	Temp < 0, Q << 0	000000	111110 001100	000010 100000	000000 101100
	Rshift Div	000000	111110 001100	000001 010000	000000 101100
4	Temp = Rem – Div	000000	111111 011100	000001 010000	000000 101100
	Temp < 0, Q << 0	000000	111111 011100	000001 010000	000000 101100
	Rshift Div	000000	111111 011100	000000 101000	000000 101100
5	Temp = Rem – Div	000000	000000 000100	000000 101000	000000 101100
	T > 0, Q << 1, R = T	000001	000000 000100	000000 101000	000000 000100
	Rshift Div	000001	000000 000100	000000 010100	000000 000100
6	Temp = Rem – Div	000001	111111 110000	000000 010100	000000 000100
	Temp < 0, Q << 0	000010	111111 110000	000000 010100	000000 000100
	Rshift Div	000010	111111 110000	000000 001010	000000 000100
7	Temp = Rem – Div	000010	111111 111010	000000 001010	000000 000100
	Temp < 0, Q << 0	000100	111111 111010	000000 001010	000000 000100
	Rshift Div	000100	111111 111010	000000 000101	000000 000100

3.8.5 No solution provided

3.8.6 No solution provided

Solution 3.9

3.9.1 No solution provided

3.9.2 No solution provided

3.9.3 No solution provided

Solution 3.10

3.10.1

a.	201326592	201326592
b.	–1000144896	3294822400

3.10.2

a.	`jal 0x00000000`
b.	`lwc1 $3,0($3)`

3.10.3

a.	0x0C000000 = 0000 1100 0000 0000 0000 0000 0000 0000 = 0 0001 1000 0000 0000 0000 0000 0000 000 sign is positive exp = 0x18 = 24 – 127 = –103 there is a hidden 1 mantissa = 0 answer = 1.0 x 2^{-103}
b.	0xC4630000 = 1100 0100 0110 0011 0000 0000 0000 0000 = 1 1000 1000 1100 0110 0000 0000 0000 000 sign is negative exp = 0x88 = 136 – 127 = 9 there is a hidden 1 mantissa = 0xC60000 = 12 x 16^{-1} + 6 x 16^{-2} = .75 +.0234375 answer = –1.7734375 x 2^{9}

3.10.4

a.	63.25 x 10^{0} = 111111.01 x 2^{0} normalize, move binary point 5 to the left 1.1111101 x 2^{5} sign = positive, exp = 127 + 5 = 132 Final bit pattern: 0 1000 0100 1111 1010 0000 0000 0000 000 = 0100 0010 0111 1101 0000 0000 0000 0000 = 0x427D0000
b.	146987.40625 x 10^{0} = 100011111000101011.011010 x 2^{0} normalize, move binary point 17 to the left 1.00011111000101011011010 x 2^{17} sign = positive, exp = 127 + 17 = 144 Final bit pattern: 0 1001 0000 0001 1111 0001 0101 1011 010 = 0100 1000 0000 1111 1000 1010 1101 1010 = 0x480F8ADA

3.10.5

a.	63.25 x 10^{0} = 111111.01 x 2^{0} normalize, move binary point 5 to the left 1.1111101 x 2^{5} sign = positive, exp = 1023 + 5 = 1028 Final bit pattern: 0 100 0000 0100 1111 1010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = 0x404FA00000000000
b.	146987.40625 x 10^{0} = 100011111000101011.011010 x 2^{0} normalize, move binary point 17 to the left 1.00011111000101011011010 x 2^{17} sign = positive, exp = 1023 + 17 = 1040 Final bit pattern: 0 100 0001 0000 0001 1111 0001 0101 1011 0100 0000 0000 0000 0000 0000 0000 0000 = 0x4101F15B40000000

3.10.6

a.	$63.25 \times 10^0 = 111111.01 \times 2^0 = 3F.40 \times 16^0$ move hex point 2 to the left $.3F40 \times 16^2$ sign = positive, exp = 64 + 2 Final bit pattern: 01000010001111110100000000000000
b.	$146987.40625 \times 10^0 = 10\ 0011\ 1110\ 0010\ 1011.011010 \times 2^0$ $= 23E2B.68 \times 16^0$ move hex point 5 to the left $.0010001111100010101101101 0 \times 16^5$ sign = positive, exp = 64 + 5 = 69 Final bit pattern: 01000101001000111110001010110110

Solution 3.11

3.11.1

a.	$-1.5625 \times 10^{-1} = -.15625 \times 10^0$ $= -.00101 \times 2^0$ move the binary point 2 to the right $= -.101 \times 2^{-2}$ exponent = –2, mantissa = –.1010000000000000000000 answer: 111111111110101100000000000000000
b.	$9.356875 \times 10^2 = 935.6875 \times 10^0$ $= 0x3A7.B \times 16^0 = 1110100111.1011 \times 2^0$ move the binary point 10 to the left $= .11101001111011 \times 2^{10}$ exponent = +10, mantissa = +.11101001111011 answer: 000000001010011101001111011000000000

3.11.2

a.	$-1.5625 \times 10^{-1} = -.15625 \times 10^0$ $= -.00101 \times 2^0$ move the binary point 3 to the right, $= -1.01 \times 2^{-3}$ exponent = –3 = –3 + 16 = 13, mantissa = –.0100000000 answer: 1011010100000000
b.	$9.356875 \times 10^2 = 935.6875 \times 10^0$ $= 0x3A7.B \times 16^0 = 1110100111.1011 \times 2^0$ move the binary point 9 to the left $= 1.1101001111011 \times 2^9$ exponent = +9 = 9 + 16 = 25, mantissa = +.1101001111011 answer: 0110011101001111

3.11.3

a.	$-1.5625 \times 10^{-1} = -.15625 \times 10^{0}$ $= -.00101 \times 2^{0}$ move the binary point 2 to the right $= -.101 \times 2^{-2}$ exponent = –2, mantissa = –.10100000000000000000000000 answer: 10110000000000000000000000000101
b.	$9.356875 \times 10^{2} = 935.6875 \times 10^{0}$ $= 0x3A7.B \times 16^{0} = 1110100111.1011 \times 2^{0}$ move the binary point 10 to the left $= .11101001111011 \times 2^{10}$ exponent = +10, mantissa = +.11101001111011 answer: 01110100111101100000000000010100

3.11.4

a.	$2.6125 \times 10^{1} + 4.150390625 \times 10^{-1}$
	$2.6125 \times 10^{1} = 26.125 = 11010.001 = 1.1010001000 \times 2^{4}$
	$4.150390625 \times 10^{-1} = .4150390625 = .011010100111 = 1.1010100111 \times 2^{-2}$
	Shift binary point 6 to the left to align exponents,
	`GR`
	`1.1010001000 00`
	`+.0000011010 10 0111  (Guard = 1, Round = 0, Sticky = 1)`
	`-------------------`
	`1.1010100010 10`
	In this case the extra bits (G,R,S) are more than half of the least significant bit (0).
	Thus, the value is rounded up.
	$1.1010100011 \times 2^{4} = 11010.100011 \times 2^{0} = 26.546875 = 2.6546875 \times 10^{1}$

b.	$-4.484375 \times 10^{1} + 1.3953125 \times 10^{1}$
	$-4.484375 \times 10^{1} = -44.84375 = -1.0110011011 \times 2^{5}$
	$1.1953125 \times 10^{1} = 11.953125 = 1.0111111010 \times 2^{3}$
	Shift binary point 2 to the left and align exponents,
	`GR`
	`-1.0110011011 00`
	`0.0101111110 10   (Guard = 1, Round = 0, Sticky = 0)`
	`------------------`
	`-1.0000011100 10`
	In this case, the Guard is 1 and the Round and Sticky bits are zero. This is the "exactly half" case—if the LSB was odd (1) we would add, but since it is even (0) we do nothing.
	$-1.0000011100 \times 2^{5} = -100000.11100 \times 2^{0} = -32.875 = -3.2875 \times 10^{1}$

3.11.5 No solution provided

3.11.6 No solution provided

Solution 3.12

3.12.1

a.	$-8.0546875 \times -1.79931640625 \times 10^{-1}$
	$-8.0546875 = -1.0000000111 \times 2^{3}$
	$-1.79931640625 \times 10^{-1} = -1.0111000010 \times 2^{-3}$
	Exp: –3 + 3 = 0, 0 + 16 = 16 (10000)
	Signs: both negative, result positive
	Mantissa:
	`  1.0000000111`
	`× 1.0111000010`
	`------------`
	`00000000000`
	`10000000111`
	`00000000000`
	`00000000000`
	`00000000000`
	`00000000000`
	`10000000111`
	`10000000111`
	`10000000111`
	`00000000000`
	`10000000111`
	`1.01110011000001001110`
	1.0111001100 00 01001110 Guard = 0, Round = 0, Sticky = 1: NoRnd
	$1.0111001100 \times 2^{0}$ = 0100000111001100 (1.0111001100 = 1.44921875)
	–8.0546875 × –.179931640625 = 1.4492931365966796875
	Some information was lost because the result did not fit into the available 10-bit field. Answer (only) off by .0000743865966796875.

b.

$8.59375 \times 10^{-2} \times 8.125 \times 10^{-1}$
$8.59375 \times 10^{-2} = .0859375 = 1.0110000000 \times 2^{-4}$
$8.125 \times 10^{-1} = .8125 = 1.1010000000 \times 2^{-1}$

Exp: –4 + –1 = –5, –5 + 16 = 11 (01011)
Signs: both positive, result positive

Mantissa:

```
                     1.0110000000
                   × 1.1010000000
                     ------------
                      00000000000
                     00000000000
                    00000000000
                   00000000000
                  00000000000
                 00000000000
                00000000000
               10110000000
              00000000000
             10110000000
            10110000000
           1000111100000000000000 Normalize, add one to exponent, negate
```

–1.0001111000 00 000000000 Guard = 0, Round = 0, Sticky = 0: Nornd

$-1.0001111000 \times 2^{-4}$ = 1011000001111000 (.00010001111000) = .06982421875

.0859375 × .8125 = .06982421875

In this case the two match exactly, since no information was lost during the shifting.

3.12.2 No solution provided

3.12.3 No solution provided

3.12.4

a.

$8.625 \times 10^1 / -4.875 \times 10^0$
$8.625 \times 10^1 = 1.0101100100 \times 2^6$
$-4.875 = -1.0011100000 \times 2^2$

Exponent = 6 – 2 = 4, 4 + 16 = 20 (10100)
Signs: one positive, one negative, result negative

Mantissa:

```
                         1.00011011000100111
10011100000. | 10101100100.0000000000000000
              -10011100000.
              -------------
                  10000100.0000
                  -1001110.0000
                  -------------
                   1100110.00000
                   -100111.00000
                   -------------
                      1111.0000000
                     -1001.1100000
                     -------------
                       101.01000000
                      -100.11100000
                      -------------
                       000.011000000000
                         -.010011100000
                           -------------
                          .000100100000000
                         -.000010011100000
                           -----------------
                          .0000100001000000
                         -.0000010011100000
                           -----------------
                          .00000011011000000
                         -.00000010011100000
                           ------------------
                          .000000000110000000
```

1.000110110001001111 Guard = 0, Round = 1, Sticky = 1: No Round, fix sign

-1.0001101100×2^4 = 1101000001101100 = 10001.101100 = –17.6875

86.25 / –4.875 = –17.692307692307

Some information was lost because the result did not fit into the available 10-bit field. Answer off by .00480769230.

b.

1.84375 × 10^0 / 1.3203125 × 10^0
1.84375 × 10^0 = 1.84375 = 1.1101100000 × 2^0
1.3203125 × 10^0 = 1.3203125 = 1.0101001000 × 2^0

Exponent = 0–0 = 0, 0 + 16 = 16 (10000)
Signs: both positive, result positive

Mantissa:

```
                                  1.011001010111110
         10101001000. | 11101100000.0000000000000000
                       -10101001000.
                       -----------
                         1000011000.00
                        - 101010010.00
                          -----------
                           11000110.000
                         - 10101001.000
                           -----------
                              11101.000000
                            - 10101.001000
                              -----------
                                111.11100000
                              - 101.01001000
                                -----------
                                 10.1001100000
                               -  1.0101001000
                                  -----------
                                  1.01000110000
                                -  .10101001000
                                   -----------
                                   .100111010000
                                -  .010101001000
                                    -----------
                                   .0100100010000
                                -  .0010101001000
                                    -----------
                                   .00011110010000
                                -  .00010101001000
                                    -----------
                                   .00001001001000
                                -  .000010101001000
```

1.0110010101 11 110   Guard = 1, Round = 1, Sticky = 1: Round up

1.0110010110 × 2^0 = 0100000110010110 = 1.0110010110 = 1.396484375

1.84375 / 1.3203125 = 1.396449704142011834319526 6

Some information was lost because the result did not fit into the available 10-bit field. Answer off by .000034671.

3.12.5 No solution provided

3.12.6 No solution provided

Solution 3.13

3.13.1

a.

$(3.984375 \times 10^{-1} + 3.4375 \times 10^{-1}) + 1.771 \times 10^{3})$
$3.984375 \times 10^{-1} = 1.1001100000 \times 2^{-2}$
$3.4375 \times 10^{-1} = 1.0110000000 \times 2^{-2}$
$1.771 \times 10^{3} = 1771 = 1.1011101011 \times 2^{10}$

shift binary point of smaller left 12 so exponents match

```
(A)       1.1001100000
(B)      +1.0110000000
         --------------
         10.1111100000 Normalize,
(A+B)     1.0111110000 × 2^-1
(C)      +1.1011101011
(A+B)      .0000000000 10 111110000 Guard=1, Round=0, Sticky=1
         --------------
(A+B)+C +1.1011101011 10 1 Round up
(A+B)+C =1.1011101100 × 2^10 = 0110101011101100 = 1772
```

b.

$(3.96875 \times 10^{0} + 8.46875 \times 10^{0}) + 2.1921875 \times 10^{1}$
$3.96875 \times 10^{0} = 1.1111110000 \times 2^{1}$
$8.46875 \times 10^{0} = 1.0000111100 \times 2^{3}$
$2.1921875 \times 10^{1} = 1.0101111011 \times 2^{4}$

shift binary point of smaller left 6 so exponents match

```
(A)         .0111111100 00 Guard=0, Round=0, Sticky=0
(B)        1.0000111100
          --------------
(A+B)      1.1000111000 No round
(A+B)       .1100011100 0  Guard=0, Round=0, Sticky=0
(C)       +1.0101111011
          --------------
(A+B)+C   10.0010010111 Normalize, add 1 to exponent, round to even
(A+B)+C = 1.0001001100 × 2^5 = 0101010001001100 = 34.375
```

3.13.2

a.

$3.984375 \times 10^{-1} + (3.4375 \times 10^{-1} + 1.771 \times 10^{3})$
$3.984375 \times 10^{-1} = 1.1001100000 \times 2^{-2}$
$3.4375 \times 10^{-1} = 1.0110000000 \times 2^{-2}$
$1.771 \times 10^{3} = 1771 = 1.1011101011 \times 2^{10}$
shift binary point of smaller left 12 so exponents match

```
(B)        .0000000000  01 0110000000 Guard=0, Round=1, Sticky=1
(C)      +1.1011101011
         --------------
(B+C)    +1.1011101011
(A)        .0000000000 011001100000
         --------------
A+(B+C) +1.1011101011 No round
A+(B+C) +1.1011101011 × 2^10 = 0110101011101011 = 1771
```

b.	$3.96875 \times 10^0 + (8.46875 \times 10^0 + 2.1921875 \times 10^1)$
	$3.96875 \times 10^0 = 1.1111110000 \times 2^1$
	$8.46875 \times 10^0 = 1.0000111100 \times 2^3$
	$2.1921875 \times 10^1 = 1.0101111011 \times 2^4$
	shift binary point of smaller left 6 so exponents match
	`(B)        .1000011110 0 Guard=0, Round=0, Sticky=0`
	`(C)       1.0101111011`
	`        --------------`
	`(B+C)     1.1110011001 No round`
	`(A)        .0011111110 000 Guard=0, Round=0, Sticky=0`
	`(B+C)     1.1110011001`
	`        --------------`
	`(A+B)+C  10.0010010111 Normalize, add 1 to exponent, round to even`
	(A+B)+C = 1.0001001100 × 2^5 = 0101010001001100 = 34.375

3.13.3

a.	No, they are not equal: (A + B) + C = 1772, A + (B + C) = 1771 (steps shown above). Exact: .398437 + .34375 + 1771 = 1771.742187.
b.	Yes, they are equal: (A + B) + C = 34.375, A + (B + C) = 34.375 (steps shown above). Exact answer is 34.359375.

3.13.4

a.	$(3.41796875 \times 10^{-3} \times 6.34765625 \times 10^{-3}) \times 1.05625 \times 10^2$
	(A) $3.41796875 \times 10^{-3} = 1.1100000000 \times 2^{-9}$
	(B) $4.150390625 \times 10^{-3} = 1.0001000000 \times 2^{-8}$
	(C) $1.05625 \times 10^2 = 1.1010011010 \times 2^6$
	Exp: –9 –8 = –17
	Signs: both positive, result positive
	Mantissa:
	`(A)                   1.1100000000`
	`(B)                 × 1.0001000000`
	`                      ------------`
	`                 11100000000`
	`             11100000000`
	`            ----------------------`
	`            1.11011100000000000000`
	`A×B         1.1101110000 00 00000000 Guard = 0, Round = 0, Sticky = 0:`
	`            No Round`
	A×B 1.1101110000 × 2^{-17} UNDERFLOW: Cannot represent number

b. $(1.140625 \times 10^2 \times -9.135 \times 10^2) \times 9.84375 \times 10^{-1}$

(A) $1.140625 \times 10^2 = 1.1100100001 \times 2^6$
(B) $-9.135 \times 10^2 = -1.1100100011 \times 2^9$
(C) $9.84375 \times 10^{-1} = 1.1111100000 \times 2^{-1}$

Exp: 6 + 9 = 15
Signs: one positive, one negative - result negative

Mantissa:

```
(A)                     1.1100100001
(B)                   × 1.1100100011
                        ------------
                         11100100001
                        11100100001
                    11100100001
                 11100100001
                11100100001
               11100100001
              ----------------------
            11.00101110000010000011 Normalize, add 1 to exponent
              1.1001011100 00 010000011 Guard=0, Round=0, Sticky=1: No Round
```

A × B $-1.1001011100 \times 2^{16}$ OVERFLOW: Cannot represent number

3.13.5

a.

$3.41796875 \times 10^{-3} \times (6.34765625 \times 10^{-3} \times 1.05625 \times 10^{2})$

(A) $3.41796875 \times 10^{-3} = 1.1100000000 \times 2^{-9}$
(B) $4.150390625 \times 10^{-3} = 1.0001000000 \times 2^{-8}$
(C) $1.05625 \times 10^{2} = 1.1010011010 \times 2^{6}$

Exp: $-8 + 6 = -2$
Signs: both positive, result positive

Mantissa:

```
(B)                       1.0001000000
(C)                     × 1.1010011010
                          ------------
                          10001000000
                        10001000000
                       10001000000
                    10001000000
                  10001000000
                 10001000000
             ----------------------
             1.1100000001110100000000
             1.1100000011 10 100000000  Guard=1, Round=0, Sticky=1:
             Round
```

B × C $1.1100000100 \times 2^{-2}$

Exp: $-9 - 2 = -11$
Signs: both positive, result positive

Mantissa:

```
(A)                       1.1100000000
(B × C)                 × 1.1100000100
                          ------------
                         11100000000
                   11100000000
                  11100000000
                 11100000000
             ----------------------
            11.00010001110000000000 Normalize, add 1 to exponent
             1.1000100011 10 0000000000 Guard = 1, Round = 0, Sticky = 0:
             Round to even
```

A × (B × C) $1.1000100100 \times 2^{-10}$

<table>
<tr><td>b.</td><td>

$1.140625 \times 10^2 \times (-9.135 \times 10^2 \times 9.84375 \times 10^{-1})$

(A) $1.140625 \times 10^2 = 1.1100100001 \times 2^6$
(B) $-9.135 \times 10^2 = -1.1100100011 \times 2^9$
(C) $9.84375 \times 10^{-1} = 1.1111100000 \times 2^{-1}$

Exp: 9 – 1 = 8
Signs: one negative, one positive - result negative

Mantissa:

<pre>
(B)   1.1100100011
(C)   × 1.1111100000

                  11100100011
                 11100100011
                11100100011
               11100100011
              11100100011
             11100100011

           11.100000110011101 Normalize, add 1 to exponent
            1.1100000110 01 1101000000 Guard=0, Round=1, Sticky=1:
            No Round
</pre>

B × C -1.1100000110×2^9
Exp: 5 + 9 = 14
Signs: one negative, one positive - result negative

Mantissa:

<pre>
(A)   1.1100100001
(B×C)   × 1.1100000110

                      11100100001
                     11100100001
               11100100001
              11100100001
             11100100001

           11.00100001000111000110 Normalize, add 1 to exponent
            1.1001000010 00 111000110 Guard=0, Round=0, Sticky=1:
            No Round
</pre>

A × (B × C) $1.1001000010 \times 2^{15}$

</td></tr>
</table>

3.13.6

a.	b) No: $A \times B = 1.1101110000 \times 2^{-17}$ UNDERFLOW: Cannot represent $A \times (B \times C) = 1.1000100100 \times 2^{-10}$ A and B are both small, so their product does not fit into the 16-bit floating point format being used.
b.	e) No: $A \times (B \times C) = -1.1001000010 \times 2^{15}$ $A \times B = -1.1001011100 \times 2^{16}$ OVERFLOW: Cannot be represented A and B are both large, so their product does not fit into the 16-bit floating point format being used.

Solution 3.14

3.14.1

a.	$1.666015625 \times 10^0 \times (1.9760 \times 10^4 - 1.9744 \times 10^4)$
	(A) $1.666015625 \times 10^0 = 1.1010101010 \times 2^0$
	(B) $1.9760 \times 10^4 = 1.0011010011 \times 2^{14}$
	(C) $-1.9744 \times 10^4 = -1.0011010010 \times 2^{14}$
	Exponents match, no shifting necessary
	`(B)    1.0011010011`
	`(C)   -1.0011010010`
	`      ---------------`
	(B+C) 0.0000000001 $\times 2^{14}$
	(B+C) 1.0000000000 $\times 2^4$
	Exp: 0 + 4 = 4
	Signs: both positive, result positive
	Mantissa:
	`(A)                     1.1010101010`
	`(B+C)                 × 1.0000000000`
	`                        ------------`
	`              11010101010`
	`             ----------------------`
	`             1.10101010100000000000`
	`A×(B+C)      1.1010101010 0000000000 Guard=0, Round=0, Sticky=0: No Round`
	$A \times (B + C)$ 1.1010101010×2^4

b.	$3.48 \times 10^2 \times (6.34765625 \times 10^{-2} - 4.052734375 \times 10^{-2})$
	(A) $3.48 \times 10^2 = 1.0101110000 \times 2^8$
	(B) $6.34765625 \times 10^{-2} = 1.0000010000 \times 2^{-4}$
	(C) $-4.052734375 \times 10^{-2} = 1.0100110000 \times 2^{-5}$
	Shift binary point of smaller left 1 so exponents match
	(B) 1.0000010000 $\times 2^{-4}$
	(C) -.1010011000 0 $\times 2^{-4}$
	`      ---------------`
	`(B+C)  .0101111000 Normalize, subtract 2 from exponent`
	(B + C) $1.0111100000 \times 2^{-6}$
	Exp: 8 – 6 = 2
	Signs: both positive, result positive
	Mantissa:
	`(A)                     1.0101110000`
	`(B+C)                 × 1.0111100000`
	`                        ------------`
	`                   10101110000`
	`                  10101110000`
	`                 10101110000`
	`                10101110000`
	`              10101110000`
	`A×(B+C)       1.1111111100 10000000000 Guard=1, Round=0, Sticky=0:`
	`              Round to even`
	$A \times (B + C)$ 1.1111111100×2^2

3.14.2

a.	$1.666015625 \times 10^0 \times (1.9760 \times 10^4 - 1.9744 \times 10^4)$
	(A) $1.666015625 \times 10^0 = 1.1010101010 \times 2^0$
	(B) $1.9760 \times 10^4 = 1.0011010011 \times 2^{14}$
	(C) $-1.9744 \times 10^4 = -1.0011010010 \times 2^{14}$
	Exp: 0 + 14 = 14
	Signs: both positive, result positive
	Mantissa:
	`(A)                    1.1010101010`
	`(B)                  × 1.0011010011`
	`                       ------------`
	`                        11010101010`
	`                       11010101010`
	`                    11010101010`
	`                  11010101010`
	`                 11010101010`
	`              11010101010`
	`             ----------------------`
	`             10.0000001001100001111 Normalize, add 1 to exponent`
	`A×B           1.0000000100 11 00001111 Guard=1, Round=1, Sticky=1: Round`
	`A×B 1.0000000101 × 2`^{15}
	Exp: 0 + 14 = 14
	Signs: one negative, one positive, result negative
	Mantissa:
	`(A)                    1.1010101010`
	`(C)                  × 1.0011010010`
	`                       ------------`
	`                       11010101010`
	`                    11010101010`
	`                  11010101010`
	`                 11010101010`
	`              11010101010`
	`            ------------------------`
	`             10.0000000111110111010 Normalize, add 1 to exponent`
	`A×C          1.0000000011 11 101110100 Guard=1, Round=1, Sticky=1: Round`
	`A×C -1.0000000100 × 2`^{15}
	`A×B      1.0000000101 × 2`^{15}
	`A×C     -1.0000000100 × 2`^{15}
	`         -------------`
	`A×B+A×C    .0000000001 × 2`^{15}
	$A \times B + A \times C$ 1.0000000000×2^5

b. $3.48 \times 10^2 \times (6.34765625 \times 10^{-2} - 4.052734375 \times 10^{-2})$

(A) $3.48 \times 10^2 = 1.0101110000 \times 2^8$
(B) $6.34765625 \times 10^{-2} = 1.0000010000 \times 2^{-4}$
(C) $-4.052734375 \times 10^{-2} = 1.0100110000 \times 2^{-5}$

Exp: 8 – 4 = 4
Signs: both positive, result positive

Mantissa:

```
(A)                     1.0101110000
(B)                   × 1.0000010000
                        ------------
                     10101110000
               10101110000
              ----------------------
              1.01100001011100000000
A×B           1.0110000101 11 00000000 Guard=1, Round=1, Sticky=0: Round
```

A × B 1.0110000110×2^4
Exp: 8 – 5 = 3
Signs: one negative, one positive, result negative

Mantissa:

```
(A)                     1.0101110000
(C)                   × 1.0100110000
                        ------------
                     10101110000
                    10101110000
                 10101110000
               10101110000
              ------------------------
               1.11000011010100000000
A×C            1.1100001101 0100000000 Guard=0, Round=1, Sticky=0: No Round
```

A × C -1.1100001101×2^3

```
A×B       1.0110000110 × 2^4
A×C       -.1110000110 1 × 2^4 (Guard=1, Round=0, Sticky=0: Round to even)
          -------------
A×B+A×C    .1000000000 × 2^4
```

A × B + A × C 1.000000000×2^3

3.14.3

a.	b) No: $A \times (B + C) = 1.1010101010 \times 2^4 = 26.65625$, and $(A \times B) + (A \times C) = 1.0000000000 \times 2^5 = 32$ Exact: $1.666015625 \times (19760 - 19744) = 26.65625$
b.	e) No: $A \times B + A \times C = 1.0000000000 \times 2^3 = 8$, and $A \times (B + C) = 1.1111111100 \times 2^2 = 7.984375$ Exact: $348 \times (.0634765625 - .04052734375) = 7.986328125$

3.14.4

	Answer	Sign	Exp	Exact?
a.	1 01111101 00000000000000000000000	–	–2	Yes
b.	0 01111011 10011001100110011001101	+	–4	No

3.14.5

a.	b + b + b + b = –1 b × 4 = –1 They are the same
b.	e + e + e + e + e + e + e + e + e + e = 1.00000000000000000000100 e × 10 = 1.00000000000000000000100

3.14.6 No solution provided

Solution 3.15

3.15.1

a.	0101 0101 0101 0101 0101 0101	0x.555555	No
b.	0001 1001 1001 1001 1001 1001	.199999	No

3.15.2

a.	0011 0011 0011 0011 0011 0011	.33333	No
b.	0001 0000 0000 0000 0000 0000	.100000	Yes

3.15.3

a.	0101 0000 0000 0000 0000 0000	.500000	Yes
b.	0001 0111 0111 0111 0111 0111	.177777	No

3.15.4

a.	01010 00000 00000 00000	.A000	Yes
b.	00011 00000 00000 00000	.3000	Yes