

ICS 233 – Computer Architecture & Assembly Language

Assignment 3 Solution

Integer Multiplication and Division, Floating-Point

1. (3 pts) Using the refined **signed** multiplication hardware, show the multiplication of:

Multiplicand = **00101101** by Multiplier = **11010110**

The result of the multiplication should be a 16 bit signed number in HI and LO registers. Eight iterations are required because there are 8 bits in the multiplier. Show the steps.

Solution:

Iteration	Multiplicand	Sign	HI	LO
0: Initialize	00101101		00000000	11010110
1: Shift right			00000000	01101011
2: LO[0] = 1	ADD	0	00101101	01101011
2: Shift right			00010110	10110101
3: LO[0] = 1	ADD	0	01000011	10110101
3: Shift right			00100001	11011010
4: Shift right			00010000	11101101
5: LO[0] = 1	ADD	0	00111101	11101101
5: Shift right			00011110	11110110
6: Shift right			00001111	01111011
7: LO[0] = 1	ADD	0	00111100	01111011
7: Shift right			00011110	00111101
8: LO[0] = 1	SUB	1	11110001	00111101
8: Shift right			11111000	10011110

Therefore, Multiplicand = $00101101_2 = 45$

multiplied by Multiplier = $11010110_2 = -42$

Product = -1890 (decimal) = $11111000\ 10011110$ (binary)

2. (3 pts) Using the refined division hardware (slides 24-25), show the **unsigned** division of:
 Dividend = 11101101 by Divisor = 00110110

The result of the division should be stored in the Remainder and Quotient registers. Eight iterations are required. Show the steps.

Solution:

Iteration	Remainder	Quotient	Divisor	Difference
0: Initialize	00000000	11101101	00110110	
1: SLL, Diff	00000001	11011010	00110110	< 0
2: SLL, Diff	00000011	10110100	00110110	< 0
3: SLL, Diff	00000111	01101000	00110110	< 0
4: SLL, Diff	00001110	11010000	00110110	< 0
5: SLL, Diff	00011101	10100000	00110110	< 0
6: SLL, Diff	00111011	01000000	00110110	00000101
6: Rem = Diff	00000101	01000001		
7: SLL, Diff	00001010	10000010	00110110	< 0
8: SLL, Diff	00010101	00000100	00110110	< 0

Therefore, Dividend = $11101101_2 = 237$ (unsigned)

divided by Divisor = $00110110_2 = 54$

Quotient = $00000100_2 = 4$ and Remainder = $00010101_2 = 21$

3. (2 pts) What is the decimal value of the following single-precision floating-point numbers?
 a) 1010 1101 0001 0100 0000 0000 0000 0000 (binary)
 b) 0100 0110 1100 1000 0000 0000 0000 0000 (binary)

Solution:

a) Sign is negative

Exponent value = $01011010_2 - 127 = -37$

Significand = $1.001\ 0100\ 0000\ 0000\ 0000\ 0000_2$

Decimal value = $-1.00101_2 \times 2^{-37} = -1.15625 \times 2^{-37} = -8.412826 \times 10^{-12}$

b) Sign is positive

Exponent value = $10001101_2 - 127 = 14$

Significand = $1.100\ 1000\ 0000\ 0000\ 0000\ 0000_2$

Decimal value = $1.1001_2 \times 2^{14} = 1.5625 \times 2^{14} = 25600$

4. (2 pts) Show the IEEE 754 binary representation for: -75.4 in ...

- a) Single Precision
- b) Double precision

Solution:

$$75 = 1001011_2$$

$$0.4 = 0.\overline{0110}_2 = 0.01100110_2 \dots$$

$$75.4 = 1001011.\overline{0110}_2 = 1.001011\overline{0110}_2 \times 2^6$$

a) Single-Precision: Biased exponent = $6 + 127 = 133$

$$1 \ 10000101 \ 0010110110011001100110\mathbf{1}_2 \text{ (rounded to nearest)}$$

b) Double-Precision: Biased exponent = $6 + 1023 = 1029$

$$1 \ 10000000101$$

$$0010110110011001100110011001100110011001100110\mathbf{10}_2 \text{ (rounded)}$$

5. (4 pts) $x = 1100 \ 0110 \ 1101 \ 1000 \ 0000 \ 0000 \ 0000 \ 0000$ (binary)

and $y = 0011 \ 1110 \ 1110 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000$ (binary)

are single-precision floating-point numbers. Perform the following operations showing all work:

- a) $x + y$
- b) $x * y$

Solution:

$$\text{Value of Exponent}(x) = 10001101_2 - 127 = 14$$

$$x = -1.101 \ 1000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_2 \times 2^{14}$$

$$\text{Value of Exponent}(y) = 01111101_2 - 127 = -2$$

$$y = 1.110 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_2 \times 2^{-2}$$

a) $x + y$

$$\begin{array}{r} - \ 1.101 \ 1000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_2 \times 2^{14} \\ + \ 1.110 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_2 \times 2^{-2} \end{array}$$

$$\begin{array}{r} - \ 1.101 \ 1000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_2 \times 2^{14} \\ + \ 0.000 \ 0000 \ 0000 \ 0000 \ 1110 \ 0000_2 \times 2^{14} \text{ (shift right 16)} \end{array}$$

$$\begin{array}{r} 1 \ 0.010 \ 1000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_2 \times 2^{14} \text{ (2's complement)} \\ 0 \ 0.000 \ 0000 \ 0000 \ 0000 \ 1110 \ 0000_2 \times 2^{14} \end{array}$$

$$1 \ 0.010 \ 1000 \ 0000 \ 0000 \ 1110 \ 0000_2 \times 2^{14} \text{ (add)}$$

$$- \ 1.101 \ 0111 \ 1111 \ 1111 \ 0010 \ 0000_2 \times 2^{14} \text{ (2's complement)}$$

Result is negative and is normalized

All shifted out bits were zeros, so result is also exact

$$x + y = 1 \ 10001101 \ 101 \ 0111 \ 1111 \ 1111 \ 0010 \ 0000_2$$

b) $x * y$

$$\text{Biased exponent}(x*y) = 10001101_2 + 01111101_2 - 127$$

$$\text{Biased exponent}(x*y) = 139 = 10001011_2$$

$$\text{Sign}(x*y) = 1 \text{ (negative)}$$

$$\begin{array}{r} 1.101\ 1000\ 0000\ 0000\ 0000\ 0000_2 \\ \times 1.110\ 0000\ 0000\ 0000\ 0000\ 0000_2 \\ \hline 1\ 1111 \\ 1101\ 1000\ 0000\ 0000\ 0000\ 0000_2 \\ 11011\ 0000\ 0000\ 0000\ 0000\ 000_2 \\ 1.10110\ 0000\ 0000\ 0000\ 0000\ 00_2 \end{array}$$

$$10.11110\ 1000\ 0000\ 0000\ 0000\ 0000_2$$

Normalize by shifting right 1 bit and increment exponent

$$\text{Significand} = 1.011\ 1101\ 0000\ 0000\ 0000\ 0000_2$$

$$\text{Biased exponent} = 139+1 = 140 = 10001100_2$$

Significand is already rounded

$$x*y = 1\ 10001100\ 011\ 1101\ 0000\ 0000\ 0000\ 0000_2$$

6. (4 pts) $x = 0101\ 1111\ 1011\ 1110\ 0100\ 0000\ 0000\ 0000$ (in binary)
and $y = 0011\ 1111\ 1111\ 1000\ 0000\ 0000\ 0000\ 0000$ (in binary)
and $z = 1101\ 1111\ 1011\ 1110\ 0100\ 0000\ 0000\ 0000$ (in binary)
represent single precision IEEE 754 floating-point numbers. Perform the following operations showing all work:

a) $x + y$

b) Result of (a) + z

c) Why is the result of (b) counterintuitive?

Solution:

$$\text{a) } x = 1.011\ 1110\ 0100\ 0000\ 0000\ 0000_2 \times 2^{64}$$

$$y = 1.111\ 1000\ 0000\ 0000\ 0000\ 0000_2 \times 2^0$$

$$\text{Difference in exponent} = 64$$

Shift significand of y right by 64 bits and add to x

The significand bits of y are truncated after rounding

$x + y = x$ because y is too small with respect to x

$$\text{Therefore, } x + y = 1.011\ 1110\ 0100\ 0000\ 0000\ 0000_2 \times 2^{64}$$

b) Result of (a) is $x = 0\ 10111111\ 01111100100000000000000_2$
 $z = 1\ 10111111\ 01111100100000000000000_2 = -x$
 Therefore, Result of (a) + z = $x - x = 0$
 $0\ 00000000\ 00000000000000000000000_2$

c) We are computing $(x+y) + z$ where $z = -x$
 Intuitively $(x+y) + -x = y$ which is not 0
 However, in this example $(x+y) + -x = 0$
 This is because we have limited number of fraction bits

7. (2 pts) IA-32 offers an 80-bit extended precision option with a 1 bit sign, 16-bit exponent, and 63-bit fraction (64-bit significand including the implied 1 before the binary point). Assume that extended precision is similar to single and double precision.

- a) What is the bias in the exponent?
- b) What is the range (in absolute value) of normalized numbers that can be represented by the extended precision option?

Solution:

- a) With a 16-bit exponent, bias = $2^{15} - 1 = 32767$
- b) largest normalized $\approx 2 \times 2^{32767} = 2^{32768} = 1.415.. \times 10^{9864}$
 smallest normalized: $1.0 \times 2^{-32766} = 2.8259.. \times 10^{-9864}$