CDA 4205 Computer Architecture

Assignment 4: MIPS Programming

1. (10 pts) What is the decimal value of the following single-precision floating-point numbers?
   1. **1010 1101 0001 0100 0000 0000 0000 0000** (binary)
   2. **0100 0110 1100 1000 0000 0000 0000 0000** (binary)

**Solution:**

* 1. **Sign bit = 1 (negative)**

**Biased Exponent = 010 1101 0 = 90**

**Exponent Value = 90 – 127 = -37**

**Decimal Value = -**

**= - 1.15625 = - 8.412826**

* 1. **Sign bit = 0 (positive)**

**Biased Exponent = 100 0110 1 = 141**

**Exponent Value = 141 – 127 = 14**

**Decimal Value =**

**= 1.5625 = 25600**

1. (10 pts) Show the IEEE 754 binary representation for: -75.4 in …
   1. Single Precision
   2. Double Precision

**Solution:**

1. **=**

**(rounded to nearest)**

**Biased exponent = 6 + 127 = 133**

**1 10000101 001011 0110 0110 0110 0110 1 (rounded to nearest)**

**b) Biased exponent = 6 + 1023 = 1029**

**1 100 0000 0101**

**001011 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 10 (rounded to even)**

1. (10 pts) Single-precision float-point numbers, and are as follows:

*x* = **1100 0110 1101 1000 0000 0000 0000 0000** (binary) and

*y* = **0011 1110 1110 0000 0000 0000 0000 0000** (binary)

Perform the following operations showing all work:

* 1. x + y
  2. x \* y

**Solution:**

**Value of Exponent(x) = 100011012 – 127 = 141 – 127 = 14**

**x = - 1.101 1000 0000 0000 0000 00002 × 214**

**Value of Exponent(y) = 011111012 – 127 = 125 – 127 = -2**

**y = 1.110 0000 0000 0000 0000 00002 × 2-2**

**a) x + y**

**- 1.101 1000 0000 0000 0000 00002 × 214**

**+ 1.110 0000 0000 0000 0000 00002 × 2-2**

**- 1.101 1000 0000 0000 0000 00002 × 214**

**+ 0.000 0000 0000 0000 1110 00002 × 214 (shift right 16)**

**1 0.010 1000 0000 0000 0000 00002 × 214 (2's complement)**

**0 0.000 0000 0000 0000 1110 00002 × 214**

**1 0.010 1000 0000 0000 1110 00002 × 214 (2’s add)**

**- 1.101 0111 1111 1111 0010 00002 × 214 (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 00002**

**b) x \* y**

**Biased exponent(x\*y) = 100011012 + 011111012 – 127**

**= 139 = 100010112**

**Sign(x\*y) = 1 (negative)**

**1.101 1000 0000 0000 0000 00002**

**× 1.110 0000 0000 0000 0000 00002**

**0.01101 1000 0000 0000 0000 00002**

**0.11011 0000 0000 0000 0000 0002**

**1.10110 0000 0000 0000 0000 002**

**10.11110 1000 0000 0000 0000 00002**

**Normalize by shifting right 1 bit and increment exponent**

**Significand = 1.011 1101 0000 0000 0000 00002**

**Biased exponent = 139 + 1 = 140 = 100011002**

**Significand is already rounded**

**x \* y = 1 10001100 011 1101 0000 0000 0000 00002**

1. (15 pts) Single precision IEEE 754 floating-point numbers,, and are as follows:

*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)

Perform the following operations.

* 1. x + y
  2. Result of (**a**)+ z
  3. Why is the result of (**b**) counterintuitive?

**Solution:**

1. **x = 1.011 1110 0100 0000 0000 00002 × 264**

**y = 1.111 1000 0000 0000 0000 00002 × 20**

**x + y = x = 1.011 1110 0100 0000 0000 00002 × 264 because y is too small with respect to x and the significand bits of y are truncated after rounding.**

1. **Result of (a) is x**

**X = 0 10111111 011 1110 0100 0000 0000 00002**

**z = 1 10111111 011 1110 0100 0000 0000 00002 = -x**

**Therefore, Result of (a) + z = x – x = 0**

**0 00000000 000 0000 0000 0000 0000 00002**

1. **Computing (x + y) + z where z = -x**

**Intuitively (x + y) + (-x) = y which is not 0**

**However in part (b), (x + y) + (-x) = 0**

**This is because of the limited number of fraction bits.**

1. 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.
   1. (2 pts) What is the bias in the exponent?
   2. (3 pts) 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 = 215 – 1 = 32767**

**b) largest normalized ≈ 2 × 232767 = 232768 = 1.415...× 109864**

**smallest normalized: 1.0 × 2-32766 = 2.8259...× 10-9864**

1. (10 pts) Using the refined division hardware, show the unsigned division of:

Dividend = 11011001 (binary) by Divisor = 00001010 (binary)

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

**Solution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iter.** | **Operations** | **Remainder (HI)** | **Quotient (LO)** | **Divisor** | **Diff.** |
| **0** | **Initialize** | **0000 0000** | **1101 1001** | **0000 1010** |  |
| **1** | **SLL, Diff** | **0000 0001** | **1011 0010** | **0000 1010** | **< 0** |
| **2** | **SLL, Diff** | **0000 0011** | **0110 0100** | **0000 1010** | **< 0** |
| **3** | **SLL, Diff** | **0000 0110** | **1100 1000** | **0000 1010** | **< 0** |
| **4** | **SLL, Diff** | **0000 1101** | **1001 0000** | **0000 1010** | **0000 0011** |
| **Rem = Diff** | **0000 0011** | **1001 0001** |  |  |
| **5** | **SLL, Diff** | **0000 0111** | **0010 0010** | **0000 1010** | **< 0** |
| **6** | **SLL, Diff** | **0000 1110** | **0100 0100** | **0000 1010** | **0000 0100** |
| **Rem = Diff** | **0000 0100** | **0100 0101** |  |  |
| **7** | **SLL, Diff** | **0000 1000** | **1000 1010** | **0000 1010** | **< 0** |
| **8** | **SLL, Diff** | **0001 0001** | **0001 0100** | **0000 1010** | **0000 0111** |
| **Rem = Diff** | **0000 0111** | **0001 0101** |  |  |

**Checking:**

**Dividend = 110110012 = 217 (unsigned)**

**Divisor = 000010102 = 10**

**Quotient = 000101012 = 21 and Remainder = 000001112 = 7**

1. (10 pts) Using the refined signed multiplication algorithm, show the multiplication of:

Multiplicand = 00101101 by Multiplier = 11010110 (signed)

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**  **Operations** | | **Multiplicand** | **Sign** | **HI** | **LO** |
| **0** | **Initialize** | **0010 1101** |  | **0000 0000** | **1101 0110** |
| **1** | **Shift right** |  |  | **0000 0000** | **0110 1011** |
| **2** | **LO[0] = 1** | **ADD** | **0** | **0010 1101** | **0110 1011** |
| **Shift right** |  |  | **0001 0110** | **1011 0101** |
| **3** | **LO[0] = 1** | **ADD** | **0** | **0100 0011** | **1011 0101** |
| **Shift right** |  |  | **0010 0001** | **1101 1010** |
| **4** | **Shift right** |  |  | **0001 0000** | **1110 1101** |
| **5** | **LO[0] = 1** | **ADD** | **0** | **0011 1101** | **1110 1101** |
| **Shift right** |  |  | **0001 1110** | **1111 0110** |
| **6** | **Shift right** |  |  | **0000 1111** | **0111 1011** |
| **7** | **LO[0] = 1** | **ADD** | **0** | **0011 1100** | **0111 1011** |
| **Shift right** |  |  | **0001 1110** | **0011 1101** |
| **8** | **LO[0] = 1** | **SUB** | **1** | **1111 0001** | **0011 1101** |
| **Shift right** |  |  | **1111 1000** | **1001 1110** |

**Checking Result:**

**Multiplicand = 0010 11012 = 45**

**Multiplier = 1101 01102 = -42**

**Product = -1890 (decimal) = 1111 1000 1001 1110 (binary in 2’s complement)**

* **Submission Requirements**
* Your solutions must be in a single file with a file name yourname-hw1.
* If scanned from hand-written copies, then the writing must be legible, or loss of credits may occur.
* Only submissions via the link on Canvas where this description is downloaded are graded. Submissions to any other locations on Canvas will be ignored.
* Late submissions are accepted for a maximum of 3 late days with 20% assignment credit off as late penalization. Assignments submitted after 3 late days will not be accepted.