|  |
| --- |
| **Assignment # 3**  **SYSC 5704 – Elements of Computer Systems** |
| Fall 2014  Submitted To  Dr. R. Gregory Franks  By  **Ferhan Jamal (100 953 487)**  Carleton University |

**3.10 [10] <§3.2> Assume 151 and 214 are signed 8-bit decimal integers stored in 2’s complement format. Calculate 151 − 214 using saturation arithmetic. The result should be in decimal.1**

**Answer:**

Let us suppose X=151 and Y=214

It is given that 151 and 214 are signed 8-bit decimal integers stored in 2’s complement format. So,

151 can be written as 10010111 which is then can be written as (-105)10.  The same is in the case of 214 which can be written as 11010110 which is then can be written as (-42)10.

151-214=10010111-11010110

= 1001 0111 + 00101010 [As X-Y=X+(-Y)]

= 1100 0001

= 193 which is the required value in decimal

**3.17 [20] <§3.3> As discussed in the text, one possible performance enhancement is to do a shift and add instead of an actual multiplication. Since 9×6, for example, can be written (2×2×2+1)×6, we can calculate 9×6 by shifting 6 to the left 3 times and then adding 6 to that result. Show the best way to calculate 033 × 055 using shifts and adds/subtracts. Assume both inputs are 8-bit unsigned integers**

**Answer:**

0x33 x 0x55 = 0x10EF

0x33= 51

0x55= 85

The best way to calculate 0x33 x 0x55 are as follows:-

1. Firstly, shift 0x55 left to 5 places (0xAA0).

2. Secondly, add 0x55 which is shifted left 4 places i.e. (0x550).

3. In the 2nd last step, we should add 0x55 shifted left once.

4. Lastly, add 0x55.

So, 0XAA0 + 0x550 + 0xAA + 0x55 = 0x10EF which implies that we have to do 3 shifts and 3 adds

**3.20 [5] <§3.5>What decimal number does the bit pattern 0x0C000000 represent if it is a two’s complement integer? An unsigned integer?**

**Answer:**

The bit pattern 0x0C00 0000 can be represented as 0000 1100 0000 0000 0000 0000 0000 0000.

As the sign bit is 0 in 0x0C00 0000 therefore it is a positive number.

In the 2nd part it is asked whether it is an unsigned integer or not? The answer is: Yes, it is an unsigned integer as 0x0C00 0000 = 201,326,592 and it has the equal value even if it is an 2's complement integer.

**3.22 [10] <§3.5> What decimal number does the bit pattern 0x0C000000 represent if it is a floating point number ?Use the IEEE 754 standard.**

**Answer:**

The decimal value of an IEEE number is given by the formula:

(1 - 2s) \* (1 + f) \*2e-bias where,

**a**. (1 - 2s) can be 1 or -1, depending on whether the sign bit is 0 or 1.

**b**. The implicit 1 is added to the fraction field.

**c**. The bias can be either 127 or 1023, for single or double precision.

The bit pattern 0x0C000000 can be represented in the binary form as:

0 00011000 00000000000000000000000

|  |  |  |
| --- | --- | --- |
| Sign(1) | Exponent (8) | Fraction(23) |
| 0 | 00011000 | 00000000000000000000000 |

Here then value of 1-2s is 1, the value of exponent is 24 and the value of fraction will be 0 as there are all zeros in fraction. So,

(1 - 2s) \* (1 + f) \*2e-bias can now be represented as: 1\*(1)\*224-127 which is equal to 9.8608 x 10-32

**3.23 [10] <§3.5> Write down the binary representation of the decimal number 63.25 assuming the IEEE 754 single precision format**

**Answer:**

IEEE 754 single precision format comprises of s, e and f. The process involved in converting it to IEEE 754 single precision format are as follows:-

1. First convert the decimal value into binary i.e. (63.25)10 = 111111.11001 = 1.1111111001 x 2^5

2. The bits to the right of the binary point comprise the fractional field f.

3. The field e contains: exponent + 127 = 132 or (10000100)

4. The value of Sign bit(s) is: 0 for +ve and 1 for -ve

|  |  |  |
| --- | --- | --- |
| s | e | F |
| 0 | 10000100 | 1111111001 |

The single precision representation is: 0 10000100 1111111001000000000000

**3.27 [20] <§3.5> IEEE 754-2008 contains a half precision that is only 16 bits wide. The leftmost bit is still the sign bit, the exponent is 5 bits wide and has a bias of 15, and the mantissa is 10 bits long. A hidden 1 is assumed. Write down the bit pattern to represent −1.5625 × 10−1 assuming a version of this format, which uses an excess-16 format to store the exponent. Comment on how the range and accuracy of this 16-bit floating point format compares to the single precision IEEE 754 standard.**

**Answer:**

As mentioned in the question, IEEE-754-2008 contains a half precision that is only 16 bits wide. It is stated in the question that the left most bit is the signed bit, the exponent is 5 bits wide and the mantissa is 10 bits long.

Left most- Signed bit which is 1 bit

Exponent- 5 bits wide

Mantissa- 10 bits long

Now, we have to write down the bit pattern to represent −1.5625 × 10−1 assuming a version of this format, which uses an excess-16 format to store the exponent. So the format now will be:-

1 sign bit + 5-bits exponent in excess-16 + 10-bits mantissa

The number (-1.5625 x 10-1)10 can be represented as: (-0.15625)10 or (-0.00101)2

(-0.00101)2 can then written as -1.01 x 2-3. By seeing the number, we can state that:

a. The number is negative, therefore sign bit is 1

b. Mantissa is 01 0000 0000 (Dropping the hidden bit)

c. The value of exponent will be: -3+16 = 13 (01001)

IEEE-754-2008 Floating Point representation will be then is: 1 01101 0100000000

|  |  |  |
| --- | --- | --- |
| S | Exponent | Mantissa |
| 1 | 01101 | 01 0000 0000 |

In the last part of the question, we have to calculate the range and accuracy of this 16-bit floating point format compares to the single precision IEEE 754 standard.

Range of this 16-bit floating point format :

In the excess-16 format to store the exponent, the smallest and the largest value can be represented as:-

Smallest value: ±1.0 x 2-15 ≈ ±3.05 x 10-5

Largest value: ±2.0 x 216 ≈ ±131,072

IEEE-754 single precision range is:

± 1.2 × 10–38   to   ± 3.4 × 10+38

The range of this 16-bit floating point format is very low as compared to the range of

single precision IEEE 754 standard.

Precision of this 16-bit floating point format :

The mantissa contains 10 bits, so the precision will be:

10 x log102 = 10 x 0.3 which is equal 3 decimal digits of precision.

In comparison, IEEE-754 single precision format has 6 decimal digits of precision approximately.

**3.32 [20] <§3.9> Calculate (3.984375×10−1+3.4375×10−1)+1.771×103 by hand, assuming each of the values are stored in the 16-bit half precision format described in exercise 3.27 (and also described in the text). assume 1 guard, 1 round bit, and 1 sticky bit, and round to the nearest even. show all the steps, and write your answer in both the 16-bit floating point format and in decimal**

**Answer:**

3.984375 x 10-1 = 0.3984375 = (0.0110011)2 = 1.10011 x 2-2

3.4375 x 10-1 = 0.34375 = (0.01011)2 = 1.011 x 2-2

1.1771 x 103 = 1177.1 = (10010011001.00011001100…)2 = 1.00100110010001100… x 210

(3.984375 x 10-1 + 3.4375 x 10-1 ) = 1.10011 x 2-2 + 1.011 x 2-2

Adding 1.10011 x 2-2 and 1.011 x 2-2 , we have:

 1.10011 x 2-2 +  1.011   x 2-2 = 10.11111 x 2-2     = 1.011111 x 2-1

Adding this result to 1.1771 x 103 using the IEEE-754-2008 with 10-bit mantissa, 1 guard bit, 1 round bit and 1 sticky bit (13 bits in total), we have,

1.0010011001000 x 210 + 0.0000000000101 x 210 = 1.0010011001101 x 210

1.0010011010 x 210 (Rounding to nearest even 10 bits after the binary point)

 IEEE-754-2008 format representation of this number will be:

Sign bit                            = 0

Mantissa                           = 0010011010

Actual exponent              = 10

Exponent with bias 15    = 10 + 15 = 25 = (11001)2

Number is: (0 11001 0010011010)2

|  |  |  |
| --- | --- | --- |
| S | E | M |
| 0 | 11001 | 00 1001 1010 |

The Decimal value is:

**1.0010011010 x 210    = (100 1001 1010)2 = (1178)10**

**3.33 [20] <§3.9> Calculate 3.984375×10−1+(3.4375×10−1+1.771×103) by hand, assuming each of the values are stored in the 16-bit half precision format described in Exercise 3.27 (and also described in the text). Assume 1 guard, 1 round bit, and 1 sticky bit, and round to the nearest even. Show all the steps, and write your answer in both the 16-bit floating point format and in decimal.**

**Answer:**

3.984375 x 10-1 + ( 3.4375 x 10-1  + 1.1771 x 103 )

3.984375 x 10-1         = 0.3984375  = (0.0110011)

                                                 = 1.10011 x 2-2

3.4375 x 10-1           = 0.34375             = (0.01011)

= 1.011 x 2-2

1.1771 x 103                        = 1177.1               = (10010011001.0001100…)2           = 1.0010011001000… x 210

(3.4375 x 10-1 + 1.1771 x 103 )       = 1.011 x 2-2 + 1.00100110010001100… x 210

IEEE-754-2008 with 10-bit mantissa, 1 guard bit, 1 round bit and 1 sticky bit:

0.0000000000011 x 210 +        (Since there are 1’s after, sticky bit set to 1)

1.0010011001000 x 210

-------------------------------

1.0010011001011 x 210

--------------------------------------------------

Rounding to nearest even 10 bits after the binary point, we have :

1.0010011010 x 210

Now adding this result to 3.984375 x 10-1 using the IEEE-754-2008 with 10-bit mantissa, 1 guard bit, 1 round bit and 1 sticky bit (13 bits in total), we have,

1.0010011010000 x 210 +

0.0000000000011 x 210

-------------------------------------

1.0010011010011 x 210

-----------------------------------------------------------

Rounding it to nearest even 10 bits after the binary point, we have:

1.0010011010 x 210

Representing this number in the IEEE-754-2008 format,

Sign bit =                             0

Mantissa =                          0010011010

Actual exponent              = 10

Exponent (bias 15)    = 10 + 15 = (25)10 = (11001)2

Number will be:

0 11001 0010011010

|  |  |  |
| --- | --- | --- |
| S | E | M |
| 0 | 11001 | 00 1001 1010 |

The decimal value is

1.0010011010 x 210           = (100 1001 1010)2 = (1178)10

**3.34 [10] <§3.9> Based on your answers to 3.32 and 3.33, does (3.984375 × 10−1 + 3.4375 × 10−1) + 1.771 × 103 = 3.984375 × 10−1 + (3.4375 × 10−1 + 1.771 × 103)?**

Yes, both on the answers to 3.32 and 3.33:

( 3.984375 x 10-1 + 3.4375 x 10-1 ) + 1.1771 x 103 is equal to 3.984375 x 10-1 + ( 3.4375 x 10-1 + 1.1771 x 103 ).

The reason for this is that the value of 3rd operand is very large as compare to the other two operands in 16-bit half precision IEEE -754-2008 format. The smaller operands get shifted to 10 bits which is eventually the entire length of the mantisaa in IEEE -754-2008 format in which siginificands is contributing 3 bits (sticky, round and guard) in the arithmetic process. When we rounded off these bits, the final result is the same in both calculations.

**3.41 [10] <§3.5> Using the IEEE 754 floating point format, write down the bit pattern that would represent −1/4. Can you represent −1/4 exactly ?**

It is given in the question that we have to find the bit pattern that would represent -1/4 using IEEE 754 floating point format. -1/4 can be represented as: -0.01

-0.01 is -1x2^-2

As -1/4 is a negative number so the value of signed bit is 1, the value of e will be 125 which can be represented in binary form as 1111101 and the value of fraction will be 0.

|  |  |  |
| --- | --- | --- |
| S | e | f |
| 1 | 01111101 | 00000000000000000000000 |

**The single precision representation will be:**

1 01111101 00000000000000000000000

Yes (-1/4)10 can be represented exactly.

**3.42 [10] <§3.5> What do you get if you add −1/4 to itself 4 times? What is −1/4×4? Are they the same? What should they be?**

(-1/4)10 can be written as: (-0.25)10 and (-0.01)2

Adding (-0.01)2 or -1x2^-2to itself 4 times:

-1x2^-2 + -1x2^-2 +-1x2^-2 +-1x2^-2 = -100.0 x 2-2

**-100.0 x 2-2 is equal to -1.0 x 20 which can also be written as (-1)10 or(-1)2** -----(1)

(-1)10 is the same value which is obtained after adding -1/4 itself to 4 times.

Now multiplying (-1/4) and 4: (-1/4)10 x (4)10 **or**(-1x2^-2 )2 x ( 1x2^2)2 **which is equal to (-1.0)2 or (-1)10** -----------(2)

From equations (1) and (2):-

Yes, the values are same from the above calculations and both are same.