Jose Franco Baquera

March 5, 2019

Assignment 3

CS 473: Architectural Concepts I

1. Do the floating point addition for the following: $9.95 \times 10^{1} + 8.0 \times 10^{-1}$.

(NOTE:* Assume that we can store only three digits of precision for the mantissa portion and one digit for the exponent portion.*)

Step #1: Align Decimal Points: Shift Number with Smaller Exponent
The number with the smaller exponent is 8.0×10^{-1}. Therefore, shift the decimal point to the left a total of two positions: 0.08×10^{1}. Now we have $9.95 \times 10^{1} + 0.08 \times 10^{1}$.
Step #2: Add Significands
$9.95 \times 10^{1} + 0.08 \times 10^{1} = 10.03 \times 10^{1}$
Step #3: Normalize Result & Check for Overflow/Underflow
$10.03 \times 10^{1} = 1.003 \times 10^{2}$. We note that neither overflow nor underflow occurred.
Step #4: Round and Renormalize If Necessary
Since we assumed that we have only three-digit precision, we have to round the result to three digits. That is, 1.003×10^{2} rounded to three digits and to the nearest is 1.00×10^{2}.

Answer: $1.00_{ten} \times 10^{2}$ *(NOTE: This is in decimal form/base 10 and not in binary form.)*

2.

a. Given the bit string 0x40500000, what floating point number does it represent? Show if this number is Normalized or De-Normalized?

We note that 0x40500000 corresponds to the following binary bit string: 0b0100-0000-0101-0000-0000-0000-0000-0000. Therefore, using the single-precision template, we have the following:

0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s	exponent								fraction																						

We note that the exponent is equal to 128 – 127 = 1 since the bias for single precision is equal to 127 and the bit pattern 0b10000000 is equal to 128 in decimal. Furthermore, the most significant bit is equal to 0, meaning that the equivalent floating-point number is positive. We can conclude that the equivalent floating-point number in binary form is $1.101_{two} \times 2^{1}$. We can also convert this number into decimal by using the equation found in the slides:

$$x = (-1)^{s} \times (1 + \text{fraction}) \times 2^{\text{exponent} - \text{bias}}$$
$$= (-1)^{0} \times (1 + .10100000000000000000000_{two}) \times 2^{128-127}$$
$$= (-1)^{0} \times \left(1 + \frac{1}{2} + \frac{1}{8}\right) \times 2^{1}$$

$$= \left(\frac{8}{8}+\frac{4}{8}+\frac{1}{8}\right) \times 2^1 = \frac{13}{8} \times 2^1 = \frac{26}{8} = 3.25_{ten}$$

This number is normalized since its exponent component is 0b10000000 and thus nonzero (i.e. denormalized numbers have all 0's in its exponent component).

Answer: The equivalent floating-point number is $1.101_{two} \times 2^1 = 3.25_{ten}$.We note that 0x40500000 is normalized since its exponent component is nonzero.

b. Given the bit string 0x80600000, what floating point number does it represent? Show if this number is Normalized or De-Normalized?

We note that 0x80600000 corresponds to the following binary bit string: 0b1000-0000-0110-0000-0000-0000-0000-0000. Therefore, using the single-precision template, we have the following:

1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	exponent								fraction																						

We note that this is a denormalized number since its exponent component is all zeros. Therefore, to find the corresponding floating-point number, we can use the following equation found on the slides:

$$x = (-1)^s \text{ x } (0 + \text{fraction}) \text{ x } 2^{-\text{bias}}$$
$$= (-1)^1 \times (0 + .11000000000000000000000_{two}) \times 2^{-126} = -1.1_{two} \times 2^{-127}$$
That is: $(-1)^1 \times \left(0 + \frac{1}{2} + \frac{1}{4}\right) \times 2^{-126} = -(2^{-1} + 2^{-2}) \times 2^{-126}$
$$= -2^{-127} - 2^{-128}$$

It is important to note that the bias for single-precision, denormalized floating point numbers is 126 and not 127.

This number is denormalized since its exponent component contains all 0's.

Answer: The equivalent floating-point number is $-1.1_{two} \times 2^{-127} = -2^{-127} - 2^{-128}$. This is approximately equal to $-8.8162076_{ten} \times 10^{-39}$. We note that 0x80600000 is denormalized since its exponent component is all 0's.

3.

a. Using IEEE 754 single precision format show the binary representation of the decimal number 63.25?

We note that the decimal number 63.25 can be represented by the following format:

$$63_{ten} + \frac{1}{4}_{ten} = \frac{252}{4}_{ten} + \frac{1}{4}_{ten} = \frac{253}{4}_{ten} = \frac{11111101_{two}}{2^2{}_{ten}} = 111111.01_{two} \times 2^0$$

We now have to normalize the previously found number:

$111111.01_{two} \times 2^0 = 1.1111101_{two} \times 2^5$

We note the equation x = (-1)^s x (1 + fraction) x 2$^{\text{exponent – bias}}$. Therefore:
$x = (-1)^0\ (1 + .11111010000000000000000_{two}) \times 2^{132-127}$

By using the previously found equation, the final IEEE 754 single precision format that shows the binary representation of the decimal number 63.25 is:

0	1	0	0	0	0	1	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	exponent								fraction																						

Answer: The final representation is the following:
0b01000010011111010000000000000000
(Or, in hexadecimal, 0x427D0000).

b. Using IEEE 754 Double precision format show the binary representation of the decimal number 63.25?

We follow the same procedure from part a):

We note that the decimal number 63.25 can be represented by

$$63_{ten} + \frac{1}{4}_{ten} = \frac{252}{4}_{ten} + \frac{1}{4}_{ten} = \frac{253}{4}_{ten} = \frac{11111101_{two}}{2^2{}_{ten}} = 111111.01_{two} \times 2^0$$

We now have to normalize the previously found number:
$111111.01_{two} \times 2^0 = 1.1111101_{two} \times 2^5$

We note the equation x = (-1)^s x (1 + fraction) x 2$^{\text{exponent – bias}}$.
Therefore:

$$(-1)^0 x\ (1 + .1111101000000000000000000000000000000000000000000000_{two}) x\ 2^{1028-1023}$$

Therefore, the final IEEE 754 double precision format that shows the binary representation is:

0	1	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
s	exponent											fraction (upper 20 bits)																			

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fraction (lower 32 bits)																															

Answer: The final representation is the following:
0b0100000001001111101000000000000000000000000000000000000000000000
(Or, in hexadecimal, 0x404FA00000000000).

4. 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 -1.6875 x 10^0 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.

We note that the decimal number -1.6875 x 10^0 can be represented by the following format:

$$-(1_{ten} + \frac{1}{2_{ten}} + \frac{1}{8_{ten}} + \frac{1}{16_{ten}}) = -(\frac{16}{16_{ten}} + \frac{8}{16_{ten}} + \frac{2}{16_{ten}} + \frac{1}{16_{ten}})$$
$$= -\frac{27}{16_{ten}} = -\frac{11011_{two}}{2^4{}_{ten}} = -1.1011_{two} \times 2^0$$

We note that the previous number is already normalized.

We note the equation x = $(-1)^s$ x (1 + fraction) x $2^{\text{exponent} - \text{bias}}$. Therefore:
$x = (-1)^1\,(1 + .1011000000_{two}) \times 2^{15-15}$

By using the previously found equation, the final IEEE 754-2008 bit pattern that represents -1.6875 x 10^0 is:

1	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0
s	exponent					fraction									

Answer Part 1: The final representation is the following: 0b1011111011000000 (Or, in hexadecimal 0xBEC0).

For part 2 of our answer, we need to comment on how the range and accuracy of this 16-bit floating point format compares to the single precision IEEE 754 standard format. We note that IEEE 754 single precision standard has a total of 32 bits, with 1 sign bit, 8 exponent bits, 23 fraction bits, all while also assuming a hidden 1. Therefore, the range of the IEEE 754 single precision format is the following:

Smallest Values: ≈ ±1.0 × 2^{-126} *(Rough Estimate)*

Largest Values: ≈ ±2.0 × 2^{127} *(Rough Estimate)*

Furthermore, the IEEE 754 single precision format has 23 bits for the fraction portion of the number. In contrast, the IEEE 754-2008 16-bit floating point format has 16 bits, with 1 sign bit, 5 exponent bits, 10 fraction bits, all while also assuming a hidden 1. Therefore, the range of the IEEE 754-2008 16-bit floating point format is the following:

Smallest Values: ≈ ±1.0 × 2^{-14} *(Rough Estimate)*

Largest Values: ≈ ±2.0 × 2^{15} *(Rough Estimate)*

Furthermore, the IEEE 754-2008 16-bit floating point format has 10 bits for the fraction portion of the number.

We conclude that the IEEE 754-2008 16-bit floating point format has a smaller range and is "less accurate" (for certain numbers) than the IEEE 754 single

precision format when converting decimal numbers into binary. That is, the IEEE 754-2008 16-bit floating point format becomes less accurate faster for really small and really big decimal numbers. Furthermore, we can also conclude that less numbers can be represented by the half-precision format than the IEEE 754 32-bit single precision format.

5. Calculate the sum of 2.6125 x 10^1 and 4.150390625 x 10^{-1} , assuming A and B are stored in the 16-bit half precision described in Question 4. Assume 1 guard, 1 round bit and 1 sticky bit and round to the nearest even. Show all the steps.

We need to convert the given two numbers into binary numbers. First, convert 2.6125 x 10^1 to binary:

$$2.6125_{\,ten} \times 10^1 = 26.125_{ten} = 26_{ten} + \frac{1}{8_{ten}} = \frac{208}{8}_{\,ten} + \frac{1}{8_{ten}} = \frac{209}{8}_{\,ten} = \frac{11010001_{two}}{2^3{}_{ten}}$$
$$= 11010.001_{two} \times 2^0. \mathit{However, we\ have\ to\ normalize\ this\ number.}$$
$$\mathit{Normalized\ number\ with\ } 10 \mathit{\ bit\ "fracion"} = 1.1010001000_{two} \times 2^4$$

Now we convert 4.150390625 x 10^{-1} to binary:

$$4.150390625_{\,ten} \times 10^{-1} = 0.4150390625_{ten} = \frac{1}{4_{ten}} + \frac{1}{8_{ten}} + \frac{1}{32_{ten}} + \frac{1}{128_{ten}} + \frac{1}{1024_{ten}}$$
$$\mathit{That\ is:}\ \frac{256}{1024_{ten}} + \frac{128}{1024_{ten}} + \frac{32}{1024_{ten}} + \frac{8}{1024_{ten}} + \frac{1}{1024_{ten}} = \frac{425}{1024_{ten}} = \frac{110101001_{two}}{2^{10}{}_{ten}}$$

$$= 0.0110101001_{two} \times 2^0.\ \mathit{However, we\ have\ to\ normalize\ this\ number.}$$
$$\mathit{Normalized\ number\ with\ } 10 \mathit{\ bit\ "fracion"} = 1.1010100100_{two} \times 2^{-2}.$$

We now shift binary point of the second number (i.e. the one with the smaller exponent) to the left 6 times in order to align exponents. After this shift, we get the following: 0.0000011010100100_{two} x 2^4. We note that we get 10 fraction bits, 1 guard bit, 1 round bit, and 1 sticky bit. That is: Guard Bit = 1, Round Bit = 0, Sticky Bit = 1. After we truncate this number, we can do regular addition:

$$\begin{array}{r} 1.1010001000000_{two} \times 2^4 \\ +\ 0.0000011010101_{two} \times 2^4 \\ \hline 1.1010100010101_{two} \times 2^4 \end{array}$$
with Guard Bit = 1, Round Bit = 0, Sticky Bit = 1.

After adding both numbers, we must round to the nearest to get 10 bits after the decimal point. That is, since the Guard Bit = 1, Round Bit = 0, Sticky Bit = 1, we must round up and get a final answer of 1.1010100011_{two} x 2^4. We can now transform this number into IEEE 754-2008 format by using the following equation:

$$x = (-1)^s \times (1 + fraction) \times 2^{exponent-bias}$$
$$x = (-1)^0 \times (1+.1010100011_{two}) \times 2^{19-15}$$

By using the previously found equation, the final IEEE 754-2008 bit pattern that represents the final answer is:

0	1	0	0	1	1	1	0	1	0	1	0	0	0	1	1
s	exponent					fraction									

We can now convert the result into decimal form:

$1.1010100011_{two} \times 2^4 = 11010.100011_{two} = 2^4 + 2^3 + 2^1 + 2^{-1} + 2^{-5} + 2^{-6}$
$= 16 + 8 + 2 + .5 + .03125 + .015625$
$= 26.546875_{ten} = 2.6546875_{ten} \times 10^1$

Answer: The result of the sum in IEEE half precision representation is 0b0100111010100011. The result in decimal form is $2.6546875_{ten} \times 10^1$.

6. Calculate $(3.984375 \times 10^{-1} + 3.4375 \times 10^{-1}) + 1.771 \times 10^3$, assuming each of the values are stored in the 16-bit half precision format described in Question 4. 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.

We need to convert the given three numbers into binary numbers. First, convert 3.984375×10^{-1} to binary:

$$3.984375_{ten} \times 10^{-1} = 0.3984375_{ten} = \frac{1}{4_{ten}} + \frac{1}{8_{ten}} + \frac{1}{64_{ten}} + \frac{1}{128_{ten}}$$
$$= \frac{32}{128_{ten}} + \frac{16}{128_{ten}} + \frac{2}{128_{ten}} + \frac{1}{128_{ten}} = \frac{51}{128_{ten}} = \frac{110011_{two}}{2^7{}_{ten}}$$
$$= 0.0110011_{two} \times 2^0. However, we\ have\ to\ normalize\ this\ number.$$
$$Normalized\ number\ with\ 10\ bit\ "fracion" = 1.1001100000_{two} \times 2^{-2}$$

Now we convert 3.4375×10^{-1} to binary:

$$3.4375_{\ ten} \times 10^{-1} = 0.34375_{ten} = \frac{1}{4_{ten}} + \frac{1}{16_{ten}} + \frac{1}{32_{ten}} = \frac{8}{32_{ten}} + \frac{2}{32_{ten}} + \frac{1}{32_{ten}}$$
$$= \frac{11}{32_{ten}} = \frac{1011_{two}}{2^5{}_{ten}} = 0.01011 \times 2^0. However, we\ have\ to\ normalize.$$
$$Normalized\ number\ with\ 10\ bit\ "fracion" = 1.0110000000_{two} \times 2^{-2}.$$

Lastly, we convert 1.771×10^3 to binary:

$$1.771_{ten} \times 10^3 = 1771_{ten} = 2^{10} + 2^9 + 2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0 = 11011101011_{two} \times 2^0$$
$$Normalized\ number\ with\ 10\ bit\ "fracion" = 1.1011101011_{two} \times 2^{10}.$$

First, we add $3.984375 \times 10^{-1} + 3.4375 \times 10^{-1}$. We note that the normalized, equivalent binary numbers have the same exponent. Therefore, we can add directly:

$$\begin{array}{r} 1.1001100000_{two} \times 2^{-2} \\ +\ 1.0110000000_{two} \times 2^{-2} \\ \hline 10.1111100000_{two} \times 2^{-2} \end{array} = 1.0111110000_{two} \times 2^{-1}$$

We note that we do not have to round or do anything else since IEEE half precision can accurately represent the sum of these two numbers without any error. We now add the previous result to the third number $1.771 \times 10^3 = 1.1011101011_{two} \times 2^{10}$. To do this addition, we shift the binary point of $1.0111110000_{two} \times 2^{-1}$ (i.e. the number with the smaller exponent) to the left 11 times in order to align exponents. That is: $1.0111110000_{two} \times 2^{-1} = 0.0000000000\underline{1}\underline{0}\underline{111110000}_{two} \times 2^{10}$. We note that we get 10 fraction bits, 1 guard bit, 1 round bit, and 1 sticky bit. For this particular step, we get Guard Bit = 1, Round Bit = 0, and Sticky Bit = 1. Now that we have truncated the number, we can do regular addition:

$$\begin{array}{r} 1.1011101011000_{two} \times 2^{10} \\ +\ 0.0000000000101_{two} \times 2^{10} \\ \hline 1.1011101011101_{two} \times 2^{10} \end{array}$$ with Guard Bit = 1, Round Bit = 0, Sticky Bit = 1.

After adding both numbers, we must round to the nearest to get 10 bits after the decimal point. That is, since the Guard Bit = 1, Round Bit = 0, Sticky Bit = 1, we must round up and get a final answer of $1.1011101100_{two} \times 2^{10}$. We can now transform this number into IEEE 754-2008 format by using the following equation:

$$x = (-1)^s \times (1 + fraction) \times 2^{exponent-bias}$$
$$x = (-1)^0 \times (1 + .1011101100_{two}) \times 2^{25-15}$$

By using the previously found equation, the final IEEE 754-2008 bit pattern that represents the final answer is:

0	1	1	0	0	1	1	0	1	1	1	0	1	1	0	0
s	exponent					fraction									

We can now convert the result into decimal form:

$1.1011101100_{two} \times 2^{10} = 11011101100_{two} = 2^{10} + 2^9 + 2^7 + 2^6 + 2^5 + 2^3 + 2^2$
$= 1772$
$= 1.772_{ten} \times 10^3$

Answer: The result of the sum in IEEE half precision representation is 0b0110011011101100. The result in decimal form is $1.772_{ten} \times 10^3$.