Example 1: Take a number such as 7.125 base 10

Convert left side of decimal point to base 2.

0111

Convert right side of decimal point to base 2.

001

and together you get...

0111.001

or...

111.001

In this case, normalize or shift binary point left two places and you get...

1.11001 (1. is implicitly understood by the IEEE format so forget about it, .11001 is the important part)

and so the binary exponent is 2.

Create the 8-bit biased binary exponent by adding 127

and you get 2 + 127 or...

129 decimal or 10000001 base 2.

7.125 is positive so sign bit = 0.

The final 32-bit unsigned int is…

sign bit :: 8-bit biased exponent :: 23-bit fraction

0 10000001 11001000000000000000000

or...

0x40E40000

This should represent decimal 7.125 in 32-bit IEEE single precision formatted number ready to be added subtracted to/from another 32-bit IEEE single precision formatted number.

Example 2: A number such as 0.05 base 10.

Nothing to convert on the left side of the decimal but the fraction side has .05 to convert.

0.05 = 0.000011001100110011001100… in binary

In this case, normalize or shift binary point five places to the right and you get...

1.10011001100110011001100

Again, the 1 to the left of the binary point is implicitly understood by IEEE 754 so it is not used, leaving the 23-bit fraction or mantissa...

10011001100110011001100

Now, the binary point was moved 5 places to the right so 127 - 5 = 122 or this FPN's 8-bit biased exponent.

There is no negative sign so the sign bit is 0.

Put it all together for 32 bit single precision IEEE 754 FPN and you get…

0 01111010 10011001100110011001100

or

00111101010011001100110011001100 = 0.05

and rounding the last bit is optional.

Example 3: A number such as 0.0 base 10.

This special case generates the 32-bit FPN…

00000000000000000000000000000000

If the sign bit is set to negative such as -0.0 then it should be ignored and treated as positive sign.

Example 4: A number such as 1.0 base 10.

1.0 decimal = 1.0 binary

No need to shift the binary point and the 1 is implicitly understood and so the 8-bit biased exponent remains 127 or…

01111111

The 23-bit mantissa is…

00000000000000000000000

The sign bit is…

0

Put it all together and get…

00111111100000000000000000000000 = 1.0 decimal

For Project 2 Part 3, using softfloat.c and softfloat.h

You need to create a C routine to read in and parse a decimal number and construct its 32-bit IEEE single precision formatted equivalent.

Repeat for second number.

Perform the Addition and Subtraction.

And you need to create a C routine to unpack a 32-bit IEEE single precision formatted number (aka answer) in order to construct a finally display it in decimal form.

Eugene Rockey.