Skill Assessment #2 Written Questions (.pdf)

Graded

Student

HARRY KIM

Total Points

34 / 40 pts

Question 1

S1 - Student ID number conversions

20 / 24 pts

– **0 pts** Well Done!

– **2 pts** steps missing

– **2 pts** Minor mistakes

✔ – **4 pts** Minor mistakes and wrong answers

– **6 pts** Significant mistake and incomplete answers

– **8 pts** Incomplete steps and wrong answers

– **12 pts** Inadequate

– **24 pts** No Submission

1 correct answer = 0xFFFFFB36

Question 2

S2 - Floating point conversion

6 / 8 pts

– **0 pts** Excellent

✔ – **2 pts** Good

– **4 pts** Fair

– **6 pts** Poor

– **8 pts** Inadequate or No Submission

The correct answer is -29.25

Question 3

S3 - Reasoning about floating point

8 / 8 pts

✔ **– 0 pts** Numbers are ordered correctly. For the second part, answer talks about how exponent is biased (i.e. "We can quickly see which numbers are bigger based on the exponent" or "The leftmost bit of the exponent can tell you whether the exponent is positive or negative")

– 1 pt Numbers are ordered correctly, but the answer to the "exponent properties" follow-up question is missing or doesn't talk about exponent bias.

– 4 pts Numbers are out of order but explanation is sufficient and alludes to exponent bias.

– 6 pts Numbers are out of order and exponent bias is not mentioned, but some effort was made.

– 8 pts Inadequate or No Submission

– 2 pts Answer is correct but work is handwritten.

– 1 pt Small math mistake leads to misordering

Question 4

Late Penalty

0 / 0 pts

✔ **– 0 pts** On time

– 4 pts Late submission

Question assigned to the following page: 1

Harry Kim
u1226472

Skill Assessment #2

S1: Following the instructions on S1, I result in my UID number being converted to 1226.3. To Convert the integer portion of my number (1226) to hexadecimal using unsigned binary notation, I took a step by step approach by first converting 1226 into binary notation by dividing by 2 and looking at the remainder until reaching 0.

1226 / 2 = 613 R 0
613 / 2 = 306 R 1
306 / 2 = 153 R 0
153 / 2 = 76 R 1
76 / 2 = 38 R 0
38 / 2 = 19 R 0
19 / 2 = 9 R 1
9 / 2 = 4 R 1
4 / 2 = 2 R 0
2 / 2 = 1 R 0
1 / 2 = 0 R 1
0 / 2 = 0 R 0

1226_10 = 010011001010_2

With the least significant digit being the first remainder we calculated, 1226 in base two turns out to be 010011001010. Next we can divide the binary number into four equal parts (counting up from the least significant digit) and quickly convert each group of four digits into their respective hexadecimal values.

0100 = 4
1100 = c
1010 = a

1226 converted into hexadecimal is 4ca.

Next, S1 asks us to negate the integer portion of the number, then convert that negative integer to hexadecimal using a two's complement representation. To get two’s complement of 1226, we take its binary notation, flip each bit, then add one.

1226_10 = 010011001010_2
010011001010 => 101100110101
101100110101 + 1 = 101100110110

Next we can divide the binary number into four equal parts (counting up from the least significant digit) and quickly convert each group of four digits into their respective hexadecimal values.

Question assigned to the following page: 1

Harry Kim
u1226472

1011 = b
0011 = 3
0110 = 6

The negation of 1226 converted to hexadecimal using a two's complement representation turns out to be 1 6.

Next, S1 asks us to encode the decimal number (including the .3) as a 32-bit floating point number and write the answer in hexadecimal. We first need to convert 1226.3 into binary. We already know the binary notation of 1226 which leaves us to find the binary notation of 0.3. What I did was double 0.3 each time and if the doubling of the previous number was more than one, then the binary encoding would be one and zero otherwise. Then by slapping that puppy onto the end of 1226 in binary (which we've done above), we can get 1226.3 in binary notation.

0.3 * 2 = 0.6
0.6 * 2 = 1.2, 1.2 - 1 = 0.2
0.2 * 2 = 0.4
0.4 * 2 = 0.8
0.8 * 2 = 1.6, 1.6 - 1 = 0.6
0.6 * 2 = 1.2, 1.2 - 1 = 0.2
0.2 * 2 = 0.4
0.4 * 2 = 0.8
0.8 * 2 = 1.6, 1.6 - 1 = 0.6
0.6 * 2 = 1.2, 1.2 - 1 = 0.2
And so on…

$0.3_{10} = 0.0100110011..._2$
$1226.3_{10} = 010011001010.0100110011..._2$

Next, we must normalize the mantissa and calculate the exponent which means finding the scientific notation of 1226.3 in binary notation. The mantissa can be found by dropping the most significant digit from the scientific notation and only looking at the decimal value. Then we can calculate the exponent by counting how many times the decimal moves to the right and adding it to 127.

010011001010.0100110011... = 01.0011001010100110011... * 2^10
127 + 10 = 137

Putting all together by putting the exponent with the mantissa right after it without the decimal point, we can now divide the number into parts of four and finally calculate the floating point hexadecimal for 1226.3.

$137_{10} = 010001001$
137 with mantissa: 010001001/00110010100100110011…

Questions assigned to the following page: 2 and 1

Harry Kim
u1226472

0100_2 = 4_16
0100_2 = 4_16
1001_2 = 9_16
1001_2 = 9_16
0100_2 = 4_16
1001_2 = 9_16
1001_2 = 9_16
1001_2 = 9_16
…

1226.3 = 0x44994999

Thus, the floating point decimal for 1226.3 would be written as 0x44994999. Note that 0x44994999 converted back to decimal is actually 1226.2999 and the 0.0001 was lost when converting due to 1001 being repeated forever and only having a finite amount of bytes to convert the binary into hexadecimal.

S2: The first step taken to convert the floating point 0xC1EA0000 to a decimal was to write the floating point into binary notation which can be quickly done by converting each hex value to its respective 4 digit binary number.

C1EA0000 = 11000001111010100000000000000000

We can then section the parts of the binary number that tell us about what number the floating point represents. The first bit is the sign bit where 0 means that the number is positive and 1 means that it is negative. The next seven bits are the exponent bits. The rest of the bits are the fractional bits.

11000001111010100000000000000000
Number is negative.

11000001111010100000000000000000
10000011_2 => 1 * 2^0 + 1 * 2 ^1 + 1 * 2 ^6 + 1 * 2 ^7
10000011_2 = 131_10
131-127 = 4
Exponent is 4.

11000001111010100000000000000000;
11010100000000000000000 => 1*2^-1 + 1*2^-2 + 0*2^-3 + 1*2^-4 + 0*2^-5 + 1*2^-6 + …
= 1*2^-1 + 1*2^-2 + 1*2^-4 + 1*2^-6 = 0.828125
Fractional mantissa is 0.828125

Knowing the sign, exponent, and mantissa, we can put it all together using a handy equation in order to get the decimal number.

Questions assigned to the following page: 2 and 3

Harry Kim
u1226472

(-1)^(*sign bit*) * (1+*mantissa*) * 2^(*exponent*)

(-1)^(1) * (1+0.828125) * 2^(4) = -29.25

The decimal value of the floating point 0xC1EA0000 is -29.5.

S3: In order to put these five 32-bit floating point numbers in ascending order (in order of increasing value), we need to be aware of a few things in the following order: the sign, the exponent, and the mantissa. Any floating point whose most significant digit is larger than 7 is a negative number. This is because the binary value of 7 is 0111 and anything above that would mean the first digit would be 1 which would indicate a negative number (i.e. 8 in binary is 1000 and 1 as the most significant digit means a negative). Therefore, we can conclude that the lowest floating point number out of the five is 0xC72E77C5.

Next we need to know the exponents of the remaining numbers as a higher exponent is more significant that the mantissa of each number. A higher exponent means a larger value so we know that the largest number of the four is 0x672E5816 and the next largest number would be 0x572EF00D.

Lastly we need to look at the remaining two numbers. Since they have the same sign bit and exponent, we need to look at their mantissas which would be everything after the third value of the floating point (actually the last binary digit of the third floating point value and everything after, but it is fine for the last two floating point numbers since they both have the same values until the 5th). We can see that 9 is greater than 2 so 0x472E9188 is the greater value and 0x472E24EE is the smallest.

Putting them all in order, we get the following list:

0xC72E77C5
0x472E24EE
0x472E9188
0x572EF00D
0x672E5816

S4: The code for this section will be turned in separately.