CSE3666

Homework 5 Solutions

1

a.

0b1011.1101 = 1 * 8 + 1 * 2 + 1 + 1 * 0.5 + 1 * 0.25 + 1 * 0.0625 = 11.8125

b.

Use the similar method in a).
or we can move the points first.

0b0.00111 = 0b111 * 2^{-5} = 7 / 32 = 0.21875

2

a.

0x45652000 = 0b 0100 0101 **0110 0101 0010 0000 0000 0000**
sign is positive
exp = 0x8A - 127 = 138 – 127 = 11
The hidden bit is 1.
significand = 1.**110 0101 0010 0000 0000 0000**
The answer is $1.110\ 0101\ 0010_2 \times 2^{11}$ = 0b 1110 0101 0010 = 3666

b.

0x00070000 = 0b 0000 0000 **0000 0111 0000 0000 0000 0000**
sign is positive
All bits in the exponent are 0. This is a denormal number.
exp = -126
The hidden bit is 0.
significand = 0.**000 0111 0000 0000 0000 0000**
The answer is $0.000\ 0111_2 \times 2^{-126} = 1.11_2 \times 2^{-5} \times 2^{-126} = 1.75 \times 2^{-131}$

3

	S	Exponent	Fraction	SP Repr.
2.32	0	1000 0000	001 0100 0111 1010 1110 0001	0x40147AE1
$-15.82*2^{-10}$	1	0111 1000	111 1101 0001 1110 1011 1000	0xBC7D1EB8
-831.9	1	1000 1000	100 1111 1111 1001 1001 1010	0xC44FF99A
-831	1	1000 1000	100 1111 1100 0000 0000 0000	0xC44FC000
0	0	0000 0000	000 0...	0x00000000
0.123	0	0111 1011	111 1011 1110 0111 0110 1101	0x3DFBE76D

$-15.82*2^{-10}$ = $-1111.1$1010001111010111000 010100…$_2$ * 2^{-10}

$= -1.11111010001111010111000_2 * 2^{-7}$
The encoded exponent = -7 + 127 = 120 = 127 – 7 = 0b0111_1000

$-831.9 = -1100111111.1110011001101101 100\ldots_2$
$= -1.10011111111100100110101_2 * 2^9$
rounding up, the last 2 bits 01 become 10.
The encoded exponent = 9 + 127 = 136 = 128 + 8 = 0b1000_1000

For encoded exponents, the expression before the excess-127 representation shows how we can write bits without using a calculator.

4

The largest odd integer.
All bits in fraction are 1, to make the significand as large as possible. The significand (in binary) is:

`1.111_1111_1111_1111_1111_1111`

To make it an odd integer, we move the binary points to right 23 places. The (actual) exponent in the normalized representation is 23. If we move 24 places, we will get an even number.

The encoded exponent is 23 + 127 = 150, in decimal.

S	Exponent	Fraction	SP Representation
0	1001 0110	111 1111 1111 1111 1111 1111	0x4B7F FFFF

The decimal value is $2^{24} – 1 = 16777215$.

5

An array of floating-point number is an array, and it is passed to a function as an array.

6

a.
0.1 * 1 + 0.2 * 2 + 0.5 * 3 + 0.2 * 3 = 2.6

b.
0.1 * 1 + 0.2 * 1 + 0.5 * 4 + 0.2 * 5 = 3.3

c.

$$\frac{\text{CPUTime}_1}{\text{CPUTime}_2} = \frac{\text{IC} \times \text{CPI}_1 \times \text{CycleTime}_1}{\text{IC} \times \text{CPI}_2 \times \text{CycleTime}_2} = \frac{2.6 \times \frac{1}{2}}{3.3 \times \frac{1}{3}} = 1.18$$

Note that P2 is faster although its CPI is higher.

d.

If the instruction count (IC) is 1 before the optimization, it becomes 1 + 0.2 = 1.2 after the optimization. IC for each class after the optimization is listed below.

	Class A	**Class B**	**Class C**
Cycle	1	1	4
IC for each class	0.1 + 0.2 * 2 = 0.5	0.2	0.5

The new average CPI is = (0.5 * 1 + 0.2 * 1 + 0.5 * 4) / 1.2 = 2.25

We could calculate the new percentages of each class:
Class A: 0.5 / 1.2, Class B: 0.2/1.2, and Class C: 0.5/1.2.

e.

The speedup achieved by the compiler is (3.3 * 1) / (2.25 * 1.2) = 1.22.

Write down every factor in CPU time. The clock cycle time is canceled out.

7

a. Speedup of method 1:

$$\frac{1}{0.8 + \frac{0.2}{100}} = 1.247 = 1.25$$

b. Speedup of method 2:

$$\frac{1}{1 - 0.2 - 0.15 + \frac{0.2}{10} + \frac{0.15}{6}} = 1.439 = 1.44$$

c. Speedup of both methods 1 and 2:

$$\frac{1}{1 - 0.2 - 0.2 - 0.15 + \frac{0.2}{100} + \frac{0.2}{10} + \frac{0.15}{6}} = 2.012$$

d. After both methods 1 and 2 are applied, the execution time of unoptimized code is

$$1 - 0.2 - 0.2 - 0.15 = 0.45$$

The execution time of optimized code is:

$$\frac{0.2}{100} + \frac{0.2}{10} + \frac{0.15}{6} = 0.047$$

If we keep optimizing the code that is already optimized, the best speed up is

$$\frac{0.45 + 0.047}{0.45} = 1.1044$$

8 Appendix

The detailed conversion of fractions in problem 3.

	0.123	0.82	0.9
1	0.246	1.64	1.8
2	0.492	1.28	1.6
3	0.984	0.56	1.2
4	1.968	1.12	0.4
5	1.936	0.24	0.8
6	1.872	0.48	
7	1.744	0.96	
8	1.488	1.92	
9	0.976	1.84	
10	1.952	1.68	
11	1.904	1.36	
12	1.808	0.72	
13	1.616	1.44	
14	1.232	0.88	
15	0.464	1.76	
16	0.928	1.52	
17	1.856	1.04	
18	1.712	0.08	
19	1.424	0.16	
20	0.848	0.32	
21	1.696	0.64	
22	1.392		
23	0.784		

0.82 starts to have repeating bits. The repeating pattern is long.

0.9 starts to have repeating bits. The pattern is 1100. So binary is 0.1110011001100….