**Sam Lazrak**

**CS 330 Computer Organization/Assembly Language**

**Homework Assignment 6**

**4/6/18**

We need to practice with converting floating point numbers using the IEEE standard. A quick trip to a physics textbook, Wikipedia, and Google reveals these physical constants. Convert each to a 64-bit floating point number following the IEEE standard. Show your work, and give the final result expressed in hexadecimal notation. (Don’t worry about the physical units.)

**Avogadro’s Number NA = 6.022140857 × 1023 mol−1**

1. Convert to binary: 1111111100001100001011101001011110101010010110110011100000000000000000000000000
2. Normalize:

1.1111111000011000010111010010111101010100101101100111 \* 2^78

1. Sign bit: 0
2. Exponent: (78 + 1023) = &1101 = %10001001101
3. Mantissa: 1111111000011000010111010010111101010100101101100111
4. The floating point is in the form: sign bit | exponent | mantissa

0 10001001101 1111111000011000010111010010111101010100101101100111

1. Hexadecimal notation:

**44dfe185d2f54b67**

**Mass of an electron m = 9.10938356 ×10−31 kg**

1. Convert to binary:

0.00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010010011110011101110011000011111001100001010001100001

1. Normalize:

1.0010011110011101110011000011111001100001010001100001 \* 2^-100

1. Sign bit: 0
2. Exponent: (-100 + 1023) = &923 = %01110011011
3. Mantissa: 0010011110011101110011000011111001100001010001100001
4. The floating point is in the form: sign bit | exponent | mantissa
5. 01110011011 0010011110011101110011000011111001100001010001100001
6. Hexadecimal notation:

**39b279dcc3e61461**

**Coulomb's constant ke = 8.9875517873681764 × 109 N·m2 /C2**

1. Convert to binary:

1000010111101100110010100000101011.010111100100000011

1. Normalize:

1.000010111101100110010100000101011010111100100000011 \* 2^33

1. Sign bit: 0
2. Exponent: (33 + 1023) = &1056 = %10000100000
3. Mantissa: 0000101111011001100101000001010110101111001000000110
4. The floating point is in the form: sign bit | exponent | mantissa

0 10000100000 0000101111011001100101000001010110101111001000000110

1. Hexadecimal notation: **4200bd99415af206**

Going the other way: here are two IEEE 64-bit floating point numbers. Convert them to decimal representation, and see if you can figure out what physics constant is represented.

**$3D A3 78 76 F1 48 11 2E** (Hint: if I did this correctly, the units are farads-per-meter (F·m−1))

1. Convert to floating point:

0011 1101 1010 0011 0111 1000 0111 0110 1111 0001 0100 1000 0001 0001 0010 1110

1. Sign bit: 0
2. Exponent: %01111011010 = &986 = 986 - 1023 = -37
3. Mantissa: 0011 0111 1000 0111 0110 1111 0001 0100 1000 0001 0001 0010 1110
4. Normalized: 1.0011011110000111011011110001010010000001000100101110 \* 2^-37
5. Binary: 0.00000000000000000000000000000000000010011011110000111011011110001010010000001000100101110
6. Decimal:

**0.0000000000088541878170000004715705485452132534295033305937749901204369962215423583984375**

1. Physics constant: permittivity of free space

**$3F 55 15 37 0F 97 60 38** (Hint: if I did this correctly, the units are newtons-per-amps-square (N·A−2))

1. Convert to floating point:

0011 1111 0101 0101 0001 0101 0011 0111 0000 1111 1001 0111 0110 0000 0011 1000

1. Sign bit: 0
2. Exponent: %01111110101 = &1013 = 1013 - 1023 = -10
3. Mantissa: 0101 0001 0101 0011 0111 0000 1111 1001 0111 0110 0000 0011 1000
4. Normalized: 1.0101000101010011011100001111100101110110000000111000 \* 2^-10
5. Binary: 0.00000000010101000101010011011100001111100101110110000000111000
6. Decimal: **0.00000125679635087359904577208880027683335356414318084716796875**
7. Physics constant: magnetic constant