

Physics 350 Problem Set 8 (Spring Semester 2009)
Due Thu., April 9 at 4:30PM

Throughout this homework assignment, numerical subscripts are used to indicate the base the preceeding number is in.

1. Convert 227_{10} (recall the subscript 10 means “base 10”) into:

- (a) Binary (base 2)
- (b) Hexadecimal (base 16)

(**HINT:** If it is helpful, you can compute a logarithm in any base in Maple using the command `log[base](number)` [*e.g.* - the base 2 logarithm of 16 can be computed as `evalf(log[2](16))`], which returns 4]. This hint may be more useful in base 2 than base 16 if you recall the discussion in lecture. In any case, you can get away without using logarithms if you prefer.)

2. Convert 258_{10} into:

- (a) Binary
- (b) Hexadecimal

3. Convert $(2^{33} + 9)_{10}$ into:

- (a) 32-bit integer
- (b) 64-bit integer
- (c) Why are the values for 3a and 3b different?

4. Convert $(9 + \frac{1}{4})_{10}$ into a (single-precision) 32-bit floating point binary number as stored in memory on a computer.

HINT: $(9 + \frac{1}{4})_{10} = \frac{37_{10}}{4_{10}}$. Represent that fraction in base 2 and then convert into a normalized scientific number with a leading “1.” (*e.g.* 101010.1 would become 1.010101×2^5).

HINT #2: As a reminder, a 32-bit floating point binary number is stored with a leading sign bit (0 if positive, 1 if negative), followed by 8 bits dedicated to the exponent where the exponent is the stored number minus 127, to allow for the storage of negative exponents (*e.g.* - a 4 in the exponent would be stored as 131 [properly converted to

binary]), followed by 23 bits for the mantissa where the leading “1” in the mantissa is assumed and not recorded (*e.g.* - the binary floating point number 1.010101 would be stored as 010101, with the leading “1” assumed). Details of this standard, called IEEE-754-1985, can be seen online at

http://en.wikipedia.org/wiki/IEEE_754-1985#Single-precision_32-bit.

5. Assuming your result from 4 is 32-bit big-endian. Find the decimal form of the 32-bit little-endian floating point number represented by the same bit sequence. **HINT:** Recall that 32-bit big-endian numbers have their four 8-bit bytes stored in the opposite order as the 32-bit little-endian numbers. Therefore if the 4 bytes making up a 32-bit big-endian number were 00000000 00000001 00000010 00000011, they would be read on a little-endian system as 00000011 00000010 00000001 00000000.
6. Answer the following questions. Clearly explain your answers if you expect full credit:
 - (a) What is the smallest number that can be stored as a 32-bit (single precision) floating point number?
 - (b) What is the largest number that can be stored as a 32-bit (single precision) floating point number?

HINT: As a warning, the highest number storable in the exponent bits is 254_{10} or 1111 1110, representing $2^{254-127} = 2^{127}$. The number 255_{10} or 1111 1111 in the exponent is reserved for representing “infinity,” something that wasn’t mentioned in lecture.