

Physics 234: Exercise 1 Solutions

1. (a) The number 116 can be expanded in powers of two

$$\begin{aligned} 116 &= 64 + 32 + 16 + 4 \\ &= 2^6 + 2^5 + 2^4 + 2^2, \end{aligned}$$

and hence $116 = 01110100_2$. Positive numbers that can be represented with a zero high bit are the same in both representations.

- (b) We want to take the complement of each bit (i.e., exchange $0 \leftrightarrow 1$) and add 1. In this way 116 goes to -116 as follows: $01110100 \rightarrow 10001011 + 1 = 10001100$. We can check that this is correct by noting that the two interpretations differ by the required shift:

$$10001011_2 = 10001011_2 - 2^8 = 140 - 256 = -116.$$

- (c) The most negative representable number -2^{N-1} is its own two's complement. When $N = 8$, the number -128 is invariant: $10000000 \rightarrow 01111111 + 1 = 10000000$.
- (d) For a conventional binary number, the leftmost bit is the most significant. Addition of two such numbers overflows if the most significant carry digit is 1; that is, if the two most significant carry bits are 10 or 11. On the other hand, the leftmost bit in two's complement encodes the sign information. Overflow in this case occurs if two positive numbers add to give a negative number or if two negative numbers add to give a positive number. This occurs if the two most significant carry bits are 10 or 01.
- (e) An exhaustive example for $N = 2$ is given on the next page:

unsigned binary	3	11	-1	two's complement
	2	10	-2	
	1	01	1	
	0	00	0	

unsigned binary

	0	1	2	3
0	0	1	2	3
1	1	2	3	×
2	2	3	×	×
3	3	×	×	×

two's complement

	0	1	-2	-1
0	0	1	-2	-1
1	1	×	-1	0
-2	-2	-1	×	×
-1	-1	0	×	-2

	00	01	10	11
00	00 00 00 00	00 01 00 01	00 10 00 10	00 11 00 11
01	00 00 01 01	01 01 01 10	00 10 01 11	11 11 01 00
10	00 00 10 10	00 01 10 11	10 10 10 00	10 11 10 01
11	00 00 11 11	11 01 11 00	10 10 11 01	11 11 11 10

2. Octal digits are encoded by 3 binary digits and hexadecimal digits by 4. Regroup the digits of the intermediate binary step and translate:

$$\begin{aligned}
 60177756_8 &= 110 : 000 : 001 : 111 : 111 : 111 : 101 : 110_2 && = \text{c0ffee}_{16} \\
 &= 1100 : 0000 : 1111 : 1111 : 1110 : 1110_2 \\
 3006012_8 &= 11 : 000 : 000 : 110 : 000 : 001 : 010_2 && = \text{c0c0a}_{16} \\
 &= 1100 : 0000 : 1100 : 0000 : 1010_2 \\
 3726755_8 &= 11 : 111 : 010 : 110 : 111 : 101 : 101_2 && = \text{faded}_{16} \\
 &= 1111 : 1010 : 1101 : 1110 : 1101_2 \\
 76545336_8 &= 111 : 110 : 101 : 100 : 101 : 011 : 011 : 110_2 && = \text{facade}_{16} \\
 &= 1111 : 1010 : 1100 : 1010 : 1101 : 1110_2 \\
 5655_8 &= 101 : 110 : 101 : 101_2 && = \text{bad}_{16} \\
 &= 1011 : 1010 : 1101_2 \\
 67545336_8 &= 110 : 111 : 101 : 100 : 101 : 011 : 011 : 110_2 && = \text{decade}_{16} \\
 &= 1101 : 1110 : 1100 : 1010 : 1101 : 1110_2
 \end{aligned}$$

If you're feeling lazy, have the computer do the conversion for you.

```

#include <iostream>
using std::cout;
using std::endl;

int main()
{
    const unsigned int i[6] = { 060177756, 03006012, 03726755,
                               076545336, 05655, 067545336 };

    cout.setf(std::ios::showbase);
    for (int n = 0; n < 6 ++n)
        cout << std::dec << i[n] << " = "
              << std::oct << i[n] << " = "
              << std::hex << i[n] << endl;
    return 0;
}

```

The output from the program above is

```

12648430 = 060177756 = 0xc0ffee
789514 = 03006012 = 0xc0c0a
1027565 = 03726755 = 0xfaded
16435934 = 076545336 = 0xfacade
2989 = 05655 = 0xbad
14600926 = 067545336 = 0xdecade

```

The C++ convention is that the prefix 0 implies octal and 0x implies hexadecimal.

3. (a) Start by breaking the number up into its contributions at each order of magnitude:

$$\begin{aligned}
 (\cdots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \cdots)_b &= \cdots + a_2 b^2 + a_1 b + a_0 + a_{-1} b^{-1} + \cdots \\
 &= \cdots (\underbrace{\cdots 00}_{3} a_3 \underbrace{00 \cdots}_2)_b + (\cdots 00 \underbrace{a_2}_{2} 00 \cdots)_b
 \end{aligned}$$

Since $-a_k b^k = b^{k+1} - b b^k - a_k b^k = b^{k+1} - (|b| - a_k) b^k$, negation can be represented by the sum

$$-(\cdots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \cdots)_b = \sum_k (\cdots 00 \underbrace{1}_{k-1} \underbrace{|b| - a_k}_k 00 \cdots)_b$$

Some tedious computation leads to

$$\begin{array}{ll} 1322_{-4} = -22 & 22 = 232_{-4} \\ 133.21_{-5} = 12.64 & -12.64 = 33.44_{-5} \\ 2050.23_{-6} = -462.25 & 462.25 = 14111.53_{-6} \\ 6231.46_{-8} = -2967.40625 & 2967.40625 = 13750.52_{-8} \end{array}$$

- (b) The representation is unique for all numbers expressible with a finite number of digits. Uniqueness fails for numbers with infinite repeating digits after the radix point. For example, consider the two numbers

$$\begin{aligned} n_1 &= (0.a0a0a0a0\cdots)_b = \frac{ab^{-1}}{1-b^{-2}} = \frac{ab}{b^2-1}. \\ n_2 &= (0.0a0a0a0a\cdots)_b = \frac{ab^{-2}}{1-b^{-2}} = \frac{a}{b^2-1}. \end{aligned}$$

Suppose that n_1 and n_2 differ by unity. This is possible if

$$n_2 - n_1 = \frac{a}{b^2-1} - \frac{ab}{b^2-1} = -\frac{a(b-1)}{b^2-1} = -\frac{a}{b+1} = 1.$$

In other words, there is an equivalence

$$(1.0a0a0a0a\cdots)_b = (0.a0a0a0a0\cdots)_b \text{ if } a = |b| - 1.$$

Some examples:

$$\begin{aligned} (1.020202\cdots)_{-3} &= (0.202020\cdots)_{-3} \\ (1.030303\cdots)_{-4} &= (0.303030\cdots)_{-4} \\ (1.040404\cdots)_{-5} &= (0.404040\cdots)_{-5} \end{aligned}$$

- (c) It is clear from our description of the negation procedure that the negative sign can always be absorbed by writing the number with its leading digit at one order higher.
4. (a) The largest number is $654321! = 6 \cdot 6! + 5 \cdot 5! + 4 \cdot 4! + 3 \cdot 3! + 2 \cdot 2! + 1 = 5039$. The smallest number is $000000! = 0$.
- (b) All intermediate numbers are represented. This follows immediately from the relation

$$(0000 \underbrace{k}_k \underbrace{k-1}_{k-1} \cdots 21)_! + 1 = (000 \underbrace{1}_{k+1} \underbrace{0}_k 0 \cdots 00)_!,$$

which can be proved as follows:

$$\begin{aligned} (k+1)! &= (k+1)k! = k \cdot k! + k! \\ &= k \cdot k! + k(k-1)! = k \cdot k! + (k-1+1)(k-1)! \\ &= k \cdot k! + (k-1)(k-1)! + (k-1)! \\ &\vdots \\ &= 1 + \sum_{l=1}^{k-1} l \cdot l! \end{aligned}$$