Lecture C5: Number representation

Please take a look at the handput "**Number Systems**" which has answers to lots of the muddy card question's from this lecture.

Response to 'Muddiest Part of the Lecture Cards'

(53 respondents, out of 67students)

1) ***Why is it important to know binary?*** (and similar questions) (7 students)

Binary is crucial because 0's and 1's are the only language that a computer understands. All your high level languages are transformed into machine language (fixed patterns of 0's and 1's). We will look at a simple machine language in upcoming lectures.

2) ***How to represent numbers w/ bit patterns?*** (1 student)

The representation schemes such as ASCII, Hex, Binary coding, Float-point representations, all provide means of representing numbers as bit patterns.

3) ***How do we know what format of binary string is being used, how do we know which is best to employ?*** (and similar questions) (4 students)

Good question. There are number of ways (I am sure you can come up with your own scheme) of representing a number in binary representation. There are standardization committees that decide which formats are used and these are agreed upon universally.

4) ***On slide number 2 in today's lecture, you show that 101101 = 37, but doesnt that equal 45?*** (1 student)

It sure does! There is a typo in the figure I coiped from Brookshear (Figure 1.15). 101101 equals 45.

5) ***How do you store values w/ base 10 using ASCII?*** (1 student)

In ASCII, we represent each symbol by an integer value. For example 'a' is using decimal number system = 97 or using hexadecimal number system 61.
So the bit pattern to represent 'a' = 01100001

6) ***110 +100=010 = 2?*** (1 student)

	1	1	0
+	1	0	0
1	0	1	0

If you have only three bits to represent the number, the '**1**' is lost (this phenomenon is called overflow).

7) ***So, 1's complement is not used?*** (1 student)

1's complement is used. You need to compute the 1's complement before you can compute the 2's complement. The handout on number systems has a detailed discussion on the topic.

8) ***2's complement?*** (1 student)

The algorithm for computing 2's complement is as follows:
- Compute the 1's complement
- Add 1

Note that in 2's complement notation, the bit patterns are unique. There is no possibility of the numbers being represented twice.

9) ***In 2's complement, is the first bit only the sign bit or is it used?*** (1 student)

The process of converting the number into 2's complement representation causes the most significant bit to become 1. It is not explicitly set. You do use all the bits to represent a number.

10) ***Is the MSB still a sign indicator in 2's compliment?*** (1 student)

Yes, but it is not explicitly set that way.

11) ***Using 8 bits and 2's complement method to represent negative numbers, you can only represent values between 127 and -128?*** (1 student)
Yes

12) ***Excess notation, magic number?*** (18 students)

The numbers written in two's complement form are not inorder i.e. the bit pattern 1111 does not represent the largest number. Excess notation provides a means of ordering the numbers. To compute the excess representation of a number, it is added to 2N-1, where N is the number of bits used to represent the number. This 2N-1 is refereed to as the magic number. For example for excess 4, the magic number is 4 (the number of bits used is 3).

13) ***Normalization, Mantissa, Exponent?*** (9 students)

Normalization is the process of converting a number in regular format into a format in the from 0.xy x 10n+1 (where x is the first digit and y is the number following the first digit). The notation that we are most familiar with is the scientific notation which is of the form x.y x 10n.

Consider 124.3, it is represented in scientific notation as 1.243 x 102. Here, the 0.243 is called the mantissa, and 2 is the exponent (the power to which 10 is raised).

For a real number, the fractional part is called the mantissa. The power to which the base is raised is called the exponent. In decimal the base is 10, for binary the base is 2.

14) ***How can there be a decimal point in binary?*** (1 student)

The decimal point in the binary number is implicit. For example in a 8 bit representation of a floating point number is

x xxx xxxx
sign exp mantissa

15) ***Floating point?*** (1 student)

There are a number of ways of representing a real number in a computer.

1. One notation is the fixed point, wherein the decimal point (radix) is fixed at some position between the digits. The digits to the left of the radix are integer values and those to the right of the radix are fractions of some fixed unit. For example: 10.82 = 1 x 101 + 0 x 100 + 8 x 10-1 + 2 x 10-2, the radix is between 0 and 8.
2. Another notation is to use rational notation (represent the real number as a ratio of two integers). The limitation however is that all real numbers are not necessarily rational.
3. The floating-point notation is the most common of the notations. It is based on scientific notation. It addresses the loss of precision seen in fixed-point notations and allows a larger range of numbers to be represented. For example, 124.3 in scientific notation is represented as 1.243 x 102

16) ***Why do we lose a bit when coming up with the mantissa?*** (1 student)

When using the process defined in Brookshear, there is no loss in precision. Whereas when we do scientific normalization, the 1 can be omitted safely and we gain an extra bit in precision.

17) ***For negative floating points, we only do 2's complement for the number on the left of the decimal, right?*** (1 student)

When you are representing floating point numbers, you don't use two's complement notation at all. You will directly convert the number into the floating point number format using the brookshear algorithm and use the sign bit to represent whether the number was negative or positive.

18) ***When I perform calculations on my computer I never get errors because of overflow of a number. Do computers use more than 8 bits for numbers? If computers are so powerful now, do we have to worry about overflow for such small numbers?*** (and similar questions) (5 students)

Good question. When you program yourself, it is critical to take care of both overflow and underflow errors. One of the causes of the Therac-25 accidents was an overflow. The counter would increment the number by 1 every time radiation therapy was performed. When the count reached the maximum number, adding 1 caused it to be reset to zero, dosing the patient with a toxic amount of radiation. You can represent numbers as sequences of bytes. Typically, an integer is 16bits, a single-precision floating-point number is 32 bit and a double-precision floating-point number is 64 bits. The representation of numbers as sequences of bytes, brings up the issue of "endian" representation. (For the roots of why word endian is used, see the handout on number systems).

The computer has no cognitive capabilities – they cannot think on their own. What a computer can do (and very well at that) is do what it is instructed to do – really fast. It cannot understand the nuances of communication that way human beings can. So the quality of your program determines whether the computer performs the operation correctly or not. The ability to think is what makes even the youngest child smarter than a computer.

19) ***Can we practice number transformation?*** (2 students)

Please attend recitation, talk to TAs, or let me know if you want more examples than those shown so far.

20) ***"No mud"*** (7 students)
Good :o)