Laboratory 2

Exploring Number Systems

Objectives

- Experiment with number systems.
- Gain practice adding in binary and converting between bases.

References

Software needed:

1) A web browser (Internet Explorer or Netscape)
2) Applets from the CD-ROM:
   a) Number systems
   b) Binary addition

Textbook reference: Chapter 2, pp. 32–51

Background

Everything you need to learn is explained in Chapter 2, "Binary Values and Number Systems."

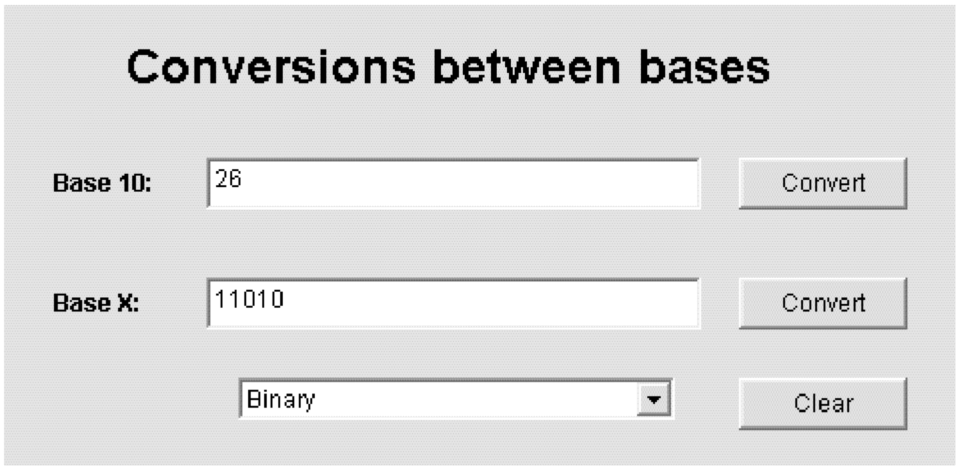
Activity

Part 1

First review positional notation of numbers in several bases, including decimal (base 10), binary (base 2), octal (base 8), and hexadecimal (base 16). See pp. 35–40 of your textbook.

Then bring up the "Number systems" applet and experiment with converting a few numbers, both from decimal to the other base, and from the other base to decimal. For example, try this:

Type the decimal number 26 into the top text area, and press the *Convert* button next to it (or simply press *Return* after you type the decimal number). You will see 11010, which is 26 expressed in binary.



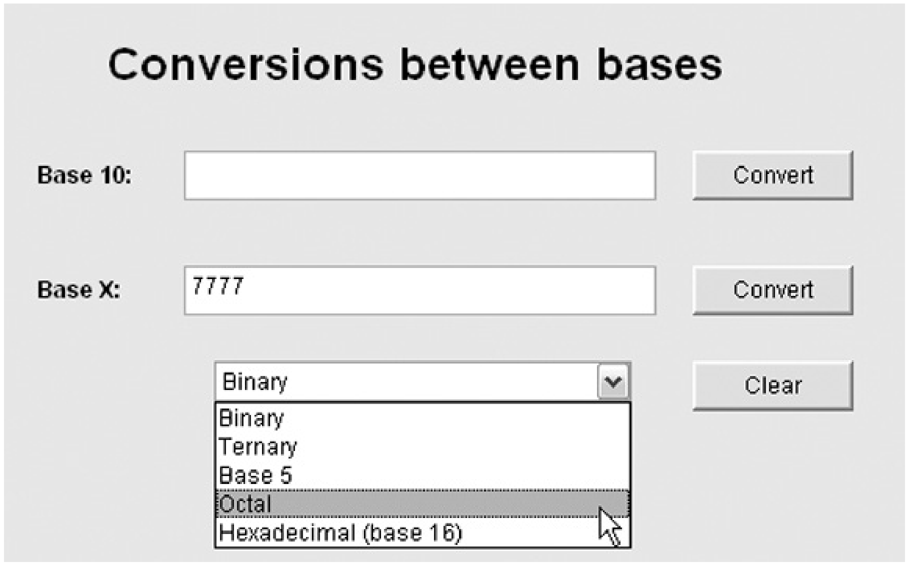
Base 10: 26

Base X: 11010

Binary

Now try converting in the other direction. Type 7777 into the *Base X* text area. Now, before you press the *Convert* button, you need to tell the applet what base 7777 is a numeral of. Could it be binary? Base 5, even base 7? (If you're not sure, read p. 38 of your textbook.)

7777 could be decimal, octal, or hexadecimal. Pull down the choice bar and select *Octal*. Now you can press *Convert*!



Conversions between bases

Base 10:

Base X: 7777

What did you get? 4095? Try verifying that the applet didn't make a mistake (always a smart thing to do, considering that applets are software, and software is written by people, and people make mistakes).

$$\begin{array}{lcr}
7 \times 8^0 = 7 \times 1 & = & 7 \\
7 \times 8^1 = 7 \times 8 & = & 56 \\
7 \times 8^2 = 7 \times 64 & = & 448 \\
7 \times 8^3 = 7 \times 512 & = & \underline{3584} \\
 & & 4095
\end{array}$$

Some calculator programs, such as the Windows Calculator, also allow you to convert between decimal, binary, octal, and hexadecimal.

Part 2

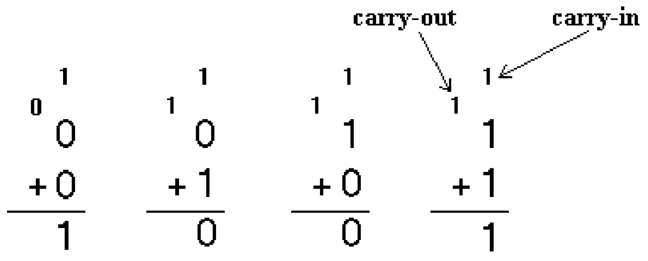
Now you will hone a skill crucial to a budding computer scientist, which is binary addition. While it might seem unfamiliar at first, binary addition works much the same as addition in the decimal system you know and love, where adding numbers in a column will sometimes generate a carry value that must be added to the column to the left.

First review binary addition on pp. 39–40 of your textbook. Make sure you understand the process of generating a carry from one column to the next left one. Here's a handy review table:

$$\begin{array}{r} {}^{0}\,0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} {}^{0}\,0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} {}^{0}\,1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} {}^{1}\,1 \\ +\,1 \\ \hline 0 \end{array}$$

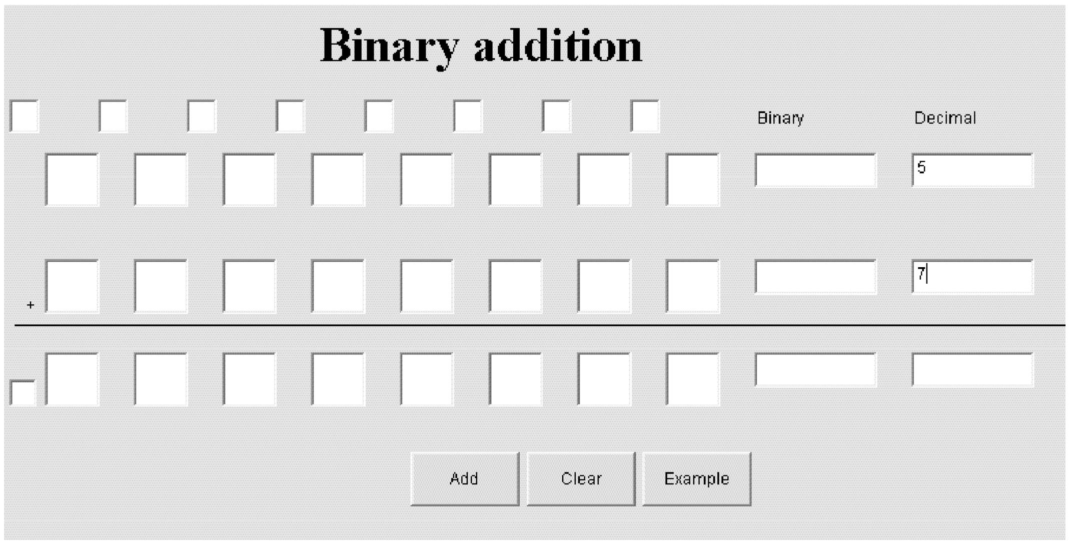
The smaller number to the left and above the bigger numbers is, of course, the carry. It must be added to the next column to the left, unless it is in the last column, in which case it just drops down.

One thing that textbooks seldom show is a table where the carry-in is added to the two digits, producing the sum bit and the carry-out. The reason is probably due to space constraints, since two digits and a carry-in would give eight combinations. However, we can quickly cut that down to four combinations, because four of the possible combinations are the same as previously shown, when the carry-in is 0. So here's the useful part of the three-bit addition table:



Using this table, see if you can do the binary addition shown on p. 39 of the textbook yourself. Remember that if there is no bit, just pretend there is a 0.

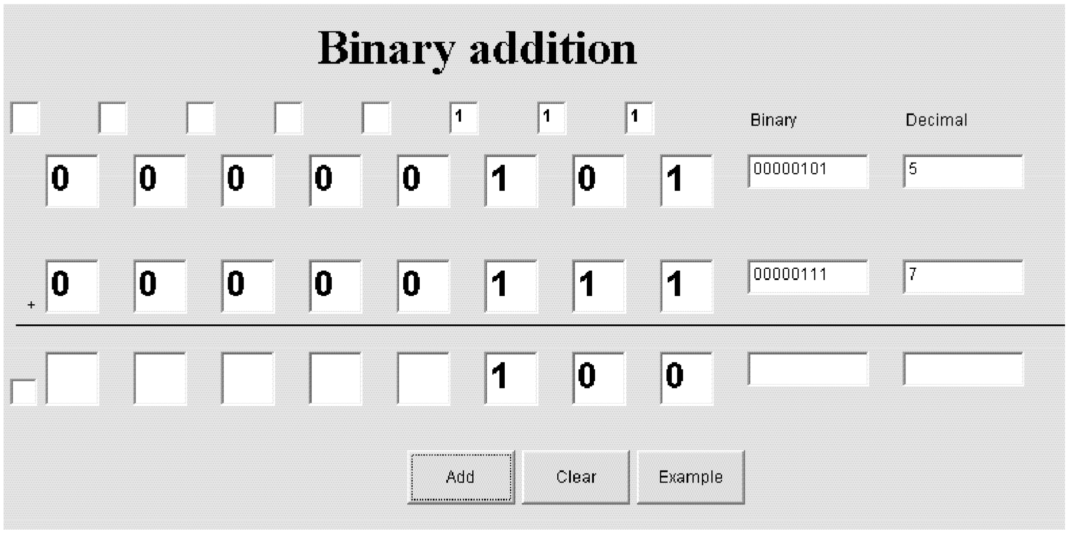
Now start the "Binary addition" applet. Type 5 into the top box on the far right, labeled *Decimal*. Type 7 into the box below it. We want to start off with easy numbers so we can verify the applet ourselves.



5

7

Now press the *Add* button and watch it go. First, the applet converts 5 to binary, which is 101, and stashes that in the top binary box. Then it converts 7 to binary (111) and tucks that into the bottom box. Next, it splits those binary numbers into bits and pads them on the left side with 0s. (Why do you suppose computers insist on putting "useless" 0s on the left side of numbers?) Finally, the applet animates its addition, column by column.



```
                    1  1  1
  0  0  0  0  0  1  0  1    00000101   5
+ 0  0  0  0  0  1  1  1    00000111   7
                 1  0  0
```

When it is done, you will see 00001100 both in the *Binary* box and split up into bits. Can you convert that into decimal? You can confirm on the applet screen that it's 12.

Many of the applets in this lab manual have an *Example* button (or buttons) that you should explore. The *Example* button usually displays a harder task than you might think of inputting yourself. In later labs, the examples show you complex patterns that would take too long to type in. In any case, press it now to see what the applet does.

Exercise 1

Name ______________________________ Date ____________________

Section ______________________________

1) Convert the binary number 11011101 to decimal by hand, showing all steps.

2) Using the "Number systems" applet, type 11011101 into the *Base X* text area. Verify your answer above. Take a screenshot.

3) One way to convert a decimal number to binary is to repeatedly divide the number by 2, writing down the remainder at each step. (See p. 43 of your textbook for an example that does this using hexadecimal, or base 16, instead of binary.) To see how to do this in binary, type a number, such as 37, into the Base 10 text area of the "Number Systems" applet and press Convert. The steps are shown in the little pink window that pops up.

4) To learn how to do this yourself, convert the decimal number 53 to binary using this method of repeated division, just like the applet did.

5) Use the "Number systems" applet to check your result. Type 37 into the *Base 10* text area and press *Convert*. Take a screenshot.

6) Use the "Number systems" applet to convert 111 from base 5 to decimal. Type 111 into the *Base X* text area, select *Base 5* from the choices in the pull-down menu, and click *Convert*. Does this make sense to you? Why? Take a screenshot and write on the paper why 111_5 equals whatever the applet shows you. (Review p. 36 of the textbook for an explanation.)

7) Add the two binary numbers together by hand, showing all carries:

$$\begin{array}{r} 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1 \\ \underline{+\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1} \end{array}$$

8) Now type the numbers 10101101 and 00111001 into the *Binary* column of the "Binary addition" applet. Click on *Add* and verify your answer with the applet's result. Take a screenshot.

9) Finally, what are the limits of the "Binary addition" applet? What are the largest numbers it can add? What is the largest result it can produce? Record your answers below, in both binary and decimal values.

Largest numbers it can add ____________________

Largest result it can produce ____________________

Deeper Investigation

Many anthropologists believe that most human societies use base 10 (decimal) for their counting system because we have ten fingers. (Why people don't use their toes as well, hence favoring a base 20 system, is a good question. Perhaps during the Ice Ages people had to keep coverings on their feet when they tramped from continent to continent!)

There are a few interesting discrepancies. For example, the ancient Babylonians had a base 60 system, which shows itself still today in our minutes and seconds, and in our 360-degree compasses.

Imagine what kinds of problems you would have to solve if you insisted on using a base 60 system. What would your calculator keypad look like? Would it be practical?

An even deeper question is what base 1 would look like. Base 2, 3, 4, 5,...8,...10,...16, and even 60 are all very similar. In base X, there are X symbols, from 0 up to (X-1). These symbols are multiplied by powers of X. But what about base 1? If you multiply 1 by any power, you still get 1. However, base 1, also called *unary*, actually has a place in theoretical computer science. Describe what a base 1 number would look like and how you could convert between decimal and unary.