# Lab #1

## *Number Systems*

### PURPOSE

This lab explores the binary, decimal, and hexadecimal number systems.

Upon completion of this lab, students will be able to do the following:

- Convert between the different base number systems.

### PREPARATION

Read the following sections from your textbook:
- Chapter 0.1, 0.2, 0.3

### MATERIALS

- Graph paper
- Calculator

*Software:*
- C compiler (or other high-level programming language)

### INTRODUCTION

Computers store information as "bits", short for *bi*nary digi*t*. A bit is a 0 or a 1, but can be represented in many ways. For example, as "true" or "false", or a light being on or off. In a computer, a 1 bit is stored as an electrical voltage of 5V, and a 0 bit is 0 volts.

More complex information is stored in groups of bits, and needs to be encoded. For example, letters and other text characters are stored in ASCII, a fairly arbitrary assignment of one 7-bit code to each character.

Humans encode numbers in a positional number system. Each number is represented by a string of digits, where each digit position has an associated weight. For example:

$$1734 = 1*1000 + 7*100 + 3*10 + 4*1$$

Here, the "base" of the number system is 10, since each weight is a power of 10 corresponding to the digit's position. Base 10 is called the decimal number system.

The natural system for computers is to use base 2, the binary number system. Binary numbers use only the digits 0 and 1, and each position corresponds to a power of 2. To make it clear that a number is represented in binary (as opposed to decimal), one often puts a "%" symbol in front. So:

%101011 = 1*32 + 0*16 + 1*8 + 0*4 + 1*2 + 1*1 = 43

Because the leftmost digit contributes the most to the value of the number, it is known as the "most significant bit", or MSB for short. Similarly, the rightmost digit is the "least significant bit", or LSB.

Base 16 is called the hexadecimal number system. It needs 16 digits, so we use 0-9 and then A, B, C, D, E, F to represent the (single digit numbers!) 10, 11, 12, 13, 14, 15. Hexadecimal numbers are usually preceded by "\$". So:

$$\begin{aligned} \$\text{1E2A} &= 1*16^{3} \quad + 14*16^{2} \quad + 2*16^{1} \quad + 10*16^{0} \\ &= 1*4096 \quad + 14*256 \quad + 2*16 \quad + 10*1 \\ &= 7722 \end{aligned}$$

Hexadecimal is useful because binary numbers are inconveniently long, but converting them to hexadecimal is easy. Simply group the binary numbers into 4-bit blocks (called nibbles), then replace each block with the corresponding hexadecimal digit:

%1110 0111 0101 1011 = \$E75B

There is another way to convert a base 2 or base 16 number to decimal. Factor the above examples, and write:

$$\begin{aligned} \%101011 &= ((((((\mathbf{1})*2 + \mathbf{0})*2 + \mathbf{1})*2 + \mathbf{0})*2 + \mathbf{1})*2 + \mathbf{1}) \\ \$\text{1E2A} &= ((((\mathbf{1})*16 + \mathbf{14})*16 + \mathbf{2})*16 + \mathbf{10}) \end{aligned}$$

These expressions are difficult to read, but suggest a simple algorithm:

Algorithm 1 (convert from a different base)

```
/* assume the digits are stored in an array digits[] */
num = 0;
i = 0;
while (digits[i] is still part of the number) {
        num = num * base;
        num = num + digits[i];
        i = i + 1;
}
```

To convert a decimal number to base 2 or base 16, we do the same process in reverse:

Algorithm 2 (convert to a different base)

```
/* assume num contains the number to be converted */
do {
        digit = num % base;
        /* now do whatever you want with the digit */
        num = num / base;  /* int divis. rounds down */
} while (num != 0)
```

This algorithm gives the digits from least significant (right) to most significant (left).

## PROCEDURE

*Part I: Coding Information*

1. For this part of the lab, we will encode a bit as a square on a piece of graph paper, with the square either left empty or shaded in. A message will consist of one entire row of squares on the paper.

2. Find a partner, and together devise a system for communicating simple text messages using the graph paper. You may use ASCII codes if you like, but it might be easier to use a code with fewer bits per letter.

3. Show how you would encode the message "hi mom":

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

4. Practice coding messages to each other. When you are proficient, demonstrate your skills to your lab instructor.

1.1 ☑

*Part II: Base Conversions*

1. Practice converting between bases by filling in the missing boxes in the following table. Each row should have the same number written three times, once in decimal, once in binary, and once in hexadecimal.

| Decimal | Binary | Hexadecimal |
|---|---|---|
| 212 | %11010100 | |
| 167 | | $A7 |
| 199 | | |
| | | $C3D |
| | %1000101011110101 | |

2. Write a computer program to convert between binary, decimal, and hex. Use whatever programming language you are most comfortable with.

   As input, the program should read a string. If the string begins with a digit, it contains a decimal number. If the string begins with % or $, it contains binary or hex. The program should print out the number with all three representations. Example:

```
Input an integer decimal, binary, or hex: $B30
2864 = $B30 = %101100110000
```

Hints: You can convert the string to an integer variable using Algorithm 1 from the introduction (it works for base 10 as well). You can then print out the variable by calculating its digits using Algorithm 2. But watch out - the digits are generated from right to left.

3. Demonstrate your program to your lab instructor

1.2 ☑

## CONCLUSION

This was a brief introduction to codes and number systems. You should be comfortable using binary or hexadecimal numbers, and converting between the various bases. You have written a converter program that you may find useful later in the semester.