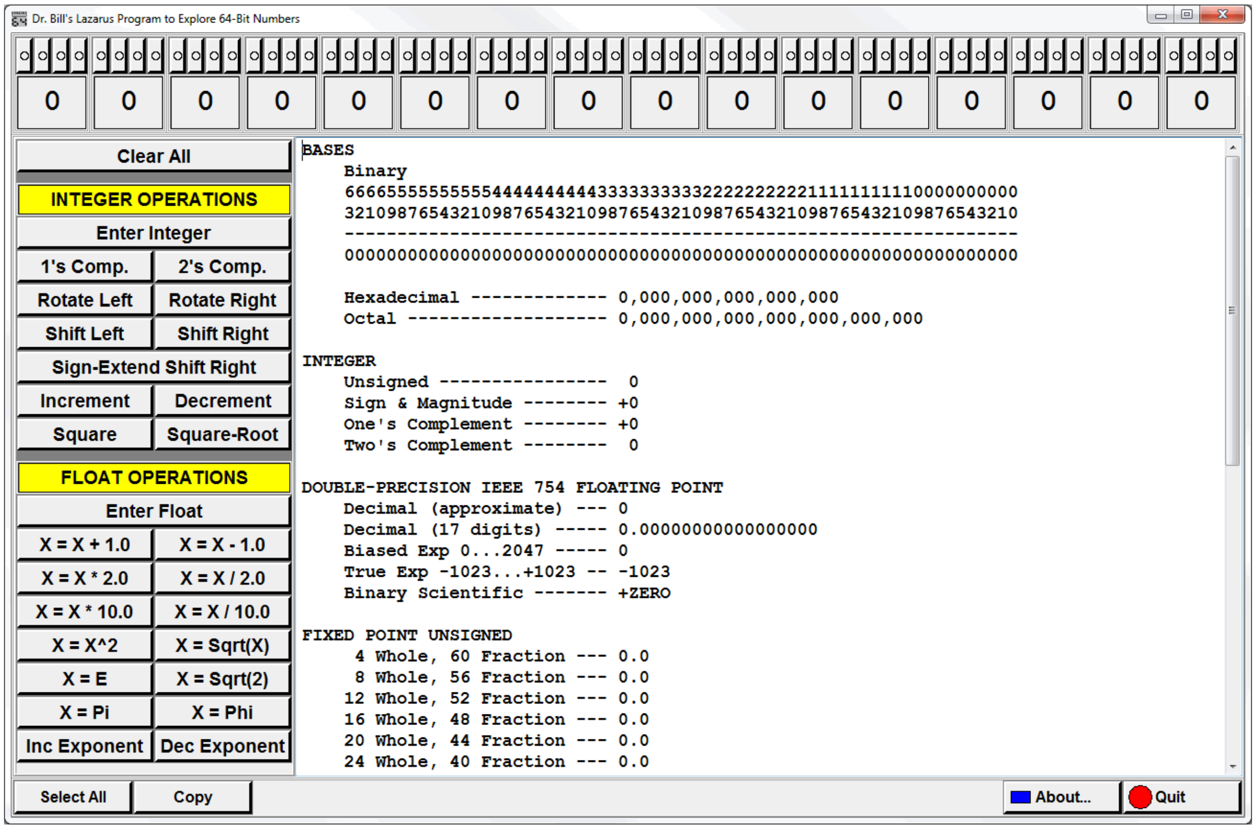
CMPSCI 145
Spring 2017
Professor William T. Verts
Lab Assignment #1

In this assignment, you will be using a program I wrote to explore 64-bit numbers. Go to the class site, and click on the link to "Dr. Bill's Lazarus Software". That link will take you to a DropBox repository with two folders, one for Mac and one for Windows. Click on whichever one corresponds to the type of computer you own. Inside the appropriate folder you will find several files; the one we want for this assignment is called "Numbers64BitProject".

Windows: The file is called **Numbers64BitProject.exe** – download that file, preferably to your desktop. There is no installation process.

Mac: The file is called **Numbers64BitProject.zip** – download that file, then double-click the file when the download is complete. The .ZIP archive will automatically unpack to a folder containing two files. One of those files(the larger) is the actual executable, but the other (the smaller) which may have a file extension of **.app** is the file that you double-click to launch the program.

Start the program running. You should see something like the following image (depending on the size of your computer's screen):



```
BASES
    Binary
    6666555555555544444444443333333333222222222211111111110000000000
    3210987654321098765432109876543210987654321098765432109876543210
    ----------------------------------------------------------------
    0000000000000000000000000000000000000000000000000000000000000000

    Hexadecimal ------------- 0,000,000,000,000,000
    Octal ------------------- 0,000,000,000,000,000,000,000

INTEGER
    Unsigned ----------------  0
    Sign & Magnitude -------- +0
    One's Complement -------- +0
    Two's Complement --------  0

DOUBLE-PRECISION IEEE 754 FLOATING POINT
    Decimal (approximate) --- 0
    Decimal (17 digits) ----- 0.00000000000000000
    Biased Exp 0...2047 ----- 0
    True Exp -1023...+1023 -- -1023
    Binary Scientific ------- +ZERO

FIXED POINT UNSIGNED
     4 Whole, 60 Fraction --- 0.0
     8 Whole, 56 Fraction --- 0.0
    12 Whole, 52 Fraction --- 0.0
    16 Whole, 48 Fraction --- 0.0
    20 Whole, 44 Fraction --- 0.0
    24 Whole, 40 Fraction --- 0.0
```

Each of the 64 buttons at the top of the window represents a single bit of the 64-bit number. Clicking a button toggles its value between 0 and 1. Bit buttons are divided into groups of four, and below each group of four buttons the hexadecimal value of that nybble is shown. Buttons in the tool panel to the left perform operations on the 64-bit number as if it was an integer or as if it was a double-precision floating-point number. In some cases an integer operation makes sense to the floating-point value.

The large white screen shows a number of interpretations for the 64 bits, including various flavors of integers (unsigned, sign & magnitude, one's complement, and two's complement), double-precision floating-point, unsigned fixed-point, and unsigned rational. As you change bits or click buttons, the white display will automatically update to show the new values for each of the interpretations.

Initial Testing (Not for Credit)

The following steps are to familiarize yourself with the program and some of the actions that it can take. It doesn't matter if you mess these steps up; if you do then just backtrack to the last **Clear All** or **Enter** and start over again.

1. If there are any 1-bits, click **Clear All**.
2. Click **Enter Integer**, and put in the number 1969.
3. Click **Shift Left**. What is happening?
4. Do this several more times.
5. Click **Shift Right** until the value is 1969 once more. What is happening?

6. Click **Clear All**.
7. Click **Increment** twice (the value should be 2).
8. Click **Square** until the value goes to zero (surprise!). How many times did you click?
9. Click **Decrement** (the value should now be -1).
10. Click the left-most bit to make it 0. Note that Sign & Magnitude changed only the sign.

11. Click **Enter Float**, and put in the number 6.94. Note the round-off error in the 17-digit answer.
12. Click **X = X ^ 2**. Note that the round-off error is larger.
13. Click **X = X ^ 2** again, and then once more. Where did the round-off error go?

14. Click **Clear All**.
15. Click **X = X + 1.0** (the value should be 1.0).
16. Click **Inc Exponent** a few times. What is happening?
17. Click **Dec Exponent** until the number is 1.0 again, then several more times. What is happening?

The Assignment (For Credit)

On the next page are a number of problems that will require you to use the program to explore bit patterns and values. As you go through each one, some lines will ask for a particular value from the display; write those answers on a sheet of paper for submission, labeled with the <u>appropriate underlined code</u> (you are allowed to use Word, Notepad (Windows), or TextEdit (Mac) to type in your answers if you prefer). For example, in problem set A, you are required to give me answers for <u>A2</u>, <u>A4</u>, and <u>A6</u>.

Problem #1 (Integer)

A0. Click **Clear All**.
A1. Set the bits so the **Unsigned** value shows: **5**.
A2. What is the hexadecimal value for this number?
A3. Click **1’s Comp**.
A4. What is the **One’s Complement** decimal value for this new number?
A5. Click **Increment**.
A6. What is the **Two’s Complement** decimal value for this new number?

Problem #2 (Floating-Point)

B0. Click **Clear All**.
B1. Enter the **largest non-infinite positive** floating-point number possible.
B2. What is the hexadecimal value for this number?
B3. What is the decimal value for this number?

C0. Click **Clear All**.
C1. Enter **positive infinity** (you’ll have to figure out an efficient way to do this!).
C2. Click **Dec Exponent**.
C3. What is the hexadecimal value for this number?
C4. What is the decimal value for this number?

D0. Click **Clear All**.
D1. Enter the **smallest non-zero normalized positive** floating-point number possible.
D2. What is the hexadecimal value for this number?
D3. What is the decimal value for this number?

E0. Click **Clear All**.
E1. Enter the **smallest non-zero denormalized positive** floating-point number possible.
E2. What is the hexadecimal value for this number?
E3. What is the decimal value for this number?

Problem #3 (Fixed-Point)

F0. Click **Clear All**.
F1. Set the bits so that the **32 Whole 32 Fraction** value shows: **255.9375**.
F2. What is the hexadecimal value for this number?
F3. What is the decimal value shown for **28 Whole 36 Fraction**?

Problem #4 (Rational)

G0. Click **Clear All**.
G1. Set the bits so that the **32 Bits / 32 Bits** value shows: **3 / 5**.
G2. What is the hexadecimal value for this number?
G3. What is the fraction shown for **36 Bits / 28 Bits**?