CSE 320 Spring 2013

Computer Project #6 -- Floating Point Representation

Assignment Overview:

For this assignment, you will develop a set of C functions which operate on floating point values without the use of floating point hardware.

It is worth 40 points (4% of course grade), and must be completed no later than 11:59 PM on Thursday, March 14.

Assignment Specifications:

1) Some microprocessors do not have floating point hardware (they have an Integer Unit, but not a Floating Point Unit). Thus, floating point operations on those systems must be performed using software.

You will develop the C functions listed below:

```
int is_negative( double );    /* Test if argument is negative */
int is_zero( double );        /* Test if argument is zero */
int is_denormal( double );    /* Test if argument is denormal */
int is_nan( double );         /* Test if argument is NaN */
int is_infinite( double );    /* Test if argument is infinite */
int is_finite( double );      /* Test if argument is finite */

double negate( double );      /* Return negation of argument */
double absolute( double );    /* Return absolute value of argument */

void display( double );       /* Display fields of argument */
```

Each function will perform the indicated operation on double precision floating point values using only integer arithmetic and bitwise operations.

The first six functions return 0 if the specified condition is false, and 1 if the condition is true. Functions "negate" and "absolute" will perform the appropriate operation on the argument and return the result.

Function "display" will decompose the argument into its constituent parts and will display the following information, appropriately labeled:

a. the biased exponent in hexadecimal and decimal
b. the true exponent in hexadecimal and decimal
c. the significand in hexadecimal
d. the classification of the floating point value

Note that infinity, not-a-number, zero and denormal numbers are special cases in IEEE floating point notation. For item (d), function "display" will print the most appropriate term from the set {"normal", "infinity", "not-a-number", "zero", "denormal"}.

Those nine functions (and any associated "helper" functions which you develop) will constitute a module named "proj06.support.c". The functions in that module will not use any floating point operations, and they will not call any C library functions. There are two exceptions: any of the functions may use the assignment operation to copy a floating point value from one variable to another, and function "display" may call "printf".

2) You will develop a driver to test your implementation of the functions in your support module. The source code for the driver must be in a separate file. All output will be appropriately labeled.

Your driver will not be interactive. If your driver accepts any input, you will supply that input in a text file named "proj06.tests".

Assignment Deliverables:

The deliverables for this assignment are:

```
proj06.makefile  -- the makefile which produces "proj06"
proj06.support.c -- the source code for your support module
proj06.driver.c  -- the source code for your driver module
proj06.tests     -- the input to your driver, if needed
```

Be sure to submit your files for grading via the "handin" program.

Assignment Notes:

Please note that your source code must be translated by "gcc", which is a C compiler and accepts C source statements (a subset of C++).

Please note that you must supply a "makefile" (named "proj06.makefile"), and that your makefile must produce an executable program named "proj06".

Please note that your driver may not be written as an interactive program, where the user supplies input in response to prompts. Instead, your test cases will either be included in the source code as literal constants, or will be supplied as inputs to the driver in a file named "proj06.tests".

If your driver requires no inputs, your solution will be executed using:

```
proj06
```

If your driver does require inputs, your solution will be executed using:

```
proj06 < proj06.tests
```

The Murdocca and Heuring text contains information about floating point representation and operations (particularly pages 45-48 and pages 74-76).