CSE 320 **Fall 2017**

Computer Project #6

Assignment Overview

This assignment develops familiarity with the C programming language, the "gcc" compiler, integer bitwise operations, and floating point representation and arithmetic.

It is worth 40 points (4% of course grade) and must be completed no later than 11:59 PM on Thursday, 10/19.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj06.makefile` – the makefile which produces "proj06"
`proj06.support.c` – the source code for your library module
`proj06.driver.c` – the source code for your driver module

Be sure to use the specified file names and to submit them for grading via the CSE handin system before the project deadline.

Assignment Specifications

Some computer systems 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.

1. You will develop the C function listed below:

```
float mult( float, float );
```

The two arguments are the single precision floating point values to be multiplied, and the return value is the product of those two values. The product will be as accurate as possible.

Arguments which are denormal will be processed as the value zero.

If either argument is the floating point value "not-a-number", function "mult" will return that value. Otherwise, if either argument is the floating point value "infinity", function "mult" will return that value.

Function "mult" (and any associated "helper" functions which you develop) will constitute the library module. The functions in that library module will not call any C library functions, and they will not use any floating point operations. There is one exception: the functions may use the assignment operation to copy a floating point value from one variable to another.

2. You will develop a driver module to test your implementation of the support module. The driver module will consist of function "main" and any additional helper functions which you choose to implement. All output will be appropriately labeled.

Assignment Notes

1. Your driver module and your library module must be in separate source code files.

2. Your source code must be translated by "gcc", which is a C compiler and accepts C source statements.

3. You must supply a "makefile" (named "proj06.makefile"), and that makefile must produce an executable program named "proj06".

4. Your driver module may not be written as an interactive program, where the user supplies input in response to prompts. Instead, your test cases will be included in the source code as literal constants.

5. The Harris and Harris textbook contains information about IEEE floating point representation and operations (section 5.3.2).

Please note that infinity, not-a-number, zero and denormal numbers are special cases in IEEE floating point notation.

6. An outline of one method for computing C = A * B is given below:

Extract sign, exponent, and fraction fields of both A and B.
Place leading bit in both significands.
Determine sign of C based on signs of A and B.
Determine exponent of C based on exponents of A and B.
Compute significand of C = significand of A * significand of B.
Normalize significand of C.
Insert sign, exponent, and fraction fields of C.

Note that this outline does not describe special cases, such as operands which are infinity, not-a-number, zero or denormal numbers.

7. In order to work with a 32-bit item as both a floating point value and an integer value, you might consider using a union. For example:

```
#include <stdio.h>

union sp_union
{
  float frep;
  unsigned irep;
};

int main()
{
  union sp_union num;

  num.frep = 1.5;

  printf( "Real: %f  Hex integer: %08x \n", num.frep, num.irep );
}
```

When compiled and executed, that program produces the following:

```
Real: 1.500000  Hex integer: 3fc00000
```

Note that the four-byte memory location "num" can be referenced as both an object of type "float" and an object of type "unsigned int".