CS 740, Fall 1998
Homework Assignment 3, Part A
Computer Arithmetic
Assigned: Oct. 8, Due: Thurs., Oct. 22, 1:20 PM

The purpose of this assignment is to give you hands-on experience with the IEEE floating point representation.

Logistics

You may work in a group of up to 3 people in solving the problems for this assignment. There will be both electronic and hard copy hand-ins, as described below. Any clarifications and revisions to the assignment will be posted on Web page `assigns.html` in the class WWW directory. All files for this assignment are in the tar file:

`/afs/cs.cmu.edu/academic/class/15740-f98/asst3/fp.tar`

The files are:

`README` A description of the files

`Makefile` A make file, including required compiler settings

`fp.h` A `.h` file declaring function prototypes and some useful macros.

`fp.c` The file that you will modify.

`ftest.c` A program that lets you examine floating point formats and the effects of conversion routines. This program only runs correctly on the Alpha (i.e., it assumes that `long`'s are 64 bits).

The Alpha provides two instructions for converting between the two IEEE floating point formats: `cvtst` converts from single to double precision, while `cvtts` converts from double to single. In fact, the floating point hardware only implements a subset of the conversion possibilities. Some of the trickier cases, typically involving denormalized numbers, infinities, and/or NaN's cause the conversion instruction to trap to an exception handler that implements the conversion in software. To enable this exception handling, you must compile your code using the `-ieee` directive on the command line. [This seems to work properly for the `cc` compiler, but not for `gcc`.]

Your first task is to implement pure software versions of these conversion routines. That is, you should modify the code for the following functions in `fp.c`:

```
double float_to_double(float x)
{
    /* write code functionally equivalent to: */
    return (double) x;
}

float double_to_float(double x)
{
    /* write code functionally equivalent to: */
    return (float) x;
}
```

Your code should produce the same effect on the Alpha as would the compiled versions of the above routines (compiled with `cc -ieee`), but they must not make use of any floating point operations.

Some of the interesting cases for `float_to_double` include:

- NaN's
- Infinities
- Denormalized floats that become normalized doubles

Some of the interesting cases for `double_to_float` include:

- NaN's
- Infinities
- Out of range doubles that yield infinity
- Denormalized doubles that go to zero (underflow)
- Normalized doubles that go to zero (underflow)
- Normalized doubles that go to denormalized floats
- Normalized doubles that require rounding

You will need to determine the behavior of the Alpha for these interesting cases by looking at the Alpha documentation or by running experiments on the machines.

Hard Copy Hand In

Describe how your conversion routines work, by describing the different argument ranges and what is required to perform conversion for these ranges.

Electronic Hand In

In the file `fp.c` fill in the fields of the structure `team_members` to document the members of your group. Then give the command:

```
make handin NAME=yourname
```

where `yourname` is the login id of the first team member.

If you need to submit a second copy, you can run:

```
make handin NAME=yourname VERSION=versionnumber
```

where `versionnumber` is `2` for your second submission, `3` for your third, and so on. We will grade the tarfile with the largest version number.