

1 Introduction

The (very inelegant) c code below can be used to determine the unit round (also called precision or epsilon) for the three standard c floating point data types. These unit rounds are frequently defined as macros in a header file, typically `float.h`, with the names `FLT_EPSILON`, `DBL_EPSILON`, `LDBL_EPSILON`, but determining them for yourself can be instructive.

If the three data types follow the IEEE standard for single precision, double precision and extended precision, you will see logical mantissa lengths of 24, 53 and 64 bits (corresponding to physical lengths of 23, 52 and 64 bits, since only the first two are packed). However, a c compiler is not required to follow these standards, so you may see some variations.

The code below also returns the number of bytes required to store each data type. You might expect 4, 8 and 10 bytes, but that need not be the case, since the compiler tries to generate efficient code by byte aligning data to increase access speed.

2 Assignment

Your assignment is to implement the code below, or a variation of it, on various computers. If you do not want to do it in c, feel free to translate the code to some other language. Even if you do it in c, you may have to modify the code below to get it to work correctly. It may be simpler just to write your own. Be very careful that the intermediate results (`temp1`, etc., below) are actually stored in data types of the correct size. The compiler may try to preserve precision by promoting intermediate results to a longer data type. You may also have to declare these variables as `volatile`, or do some other trick, to prevent the compiler from optimizing your code to something meaningless. The code below though does work with the compilers that I tried.

Your report on your experiment(s) should include the output from your code and a brief discussion of the results. Please turn in a very neat and clear report. You should also include an identification of the compiler(s) used (and the version number) and the type of machine(s) used. Sometimes, this information is difficult to determine in the various campus computer labs. Do the best you can in a reasonable length of time.

3 Code

```
/* ur.c
** Mth 351 Fall 2005 Bent Petersen
**
** This simple program computes the unit round for the three
** c floating point data types. Compile without optimization.
```

```

** Use my code or write your own better code. If you can't
** compile a program, try a spreadsheet.
**
** Typical compilation commands:
**
** Microsoft C/C++      cl /W4 ur.c
** GNU GCC C/C++       gcc ur.c -o ur -lm -Wall
*/

#include <stdio.h>
#include <math.h>

int main (void) {
    double ur;
    float temp1 = 2.0;
    double temp2 = 2.0;
    long double temp3 = 2.0;

    char msg[] = "Logical length of mantissa";

    /*** FLOAT ***/

    ur = 1.0;
    while ( temp1 > 1.0 )
        {ur /= 2.0; temp1 = (float)(1.0 + ur);}
    ur *= 2.0;

    printf(
        "\nFloat unit round\t\t= %14.7e\n%s\t= %3d bits",
        ur, msg, (int)(1.05-log(ur)/log(2.0)) );
    printf("\nSize of float\t\t\t= %3d bytes", sizeof(temp1));

    /*** DOUBLE ***/

    ur = 1.0;
    while ( temp2 > 1.0 )
        {ur /= 2.0; temp2 = 1.0 + ur; }
    ur *= 2.0;

    printf(
        "\n\nDouble unit round\t\t= %14.7e\n%s\t= %3d bits",
        ur, msg, (int)(1.05-log(ur)/log(2.0)) );
    printf("\nSize of double\t\t\t= %3d bytes", sizeof(temp2));

    /*** LONG DOUBLE ***/

    ur = 1.0;
    while ( temp3 > 1.0 )
        {ur /= 2.0; temp3 = (long double)(1.0 + ur); }
    ur *= 2.0;

```

```
printf(
    "\n\nLong double unit round\t\t= %14.7e\n%s\t= %3d bits",
    ur, msg, (int)(1.05-log(ur)/log(2.0)) );
printf("\nSize of long double\t\t= %3d bytes\n", sizeof(temp3));

return 0;
}

// end
```