Concepts of Conversion of Base 10 Number to Base 2 Floating Point Binary Representation

Sean Rodby, Luke Snyder, Autar Kaw

University of South Florida

United States of America

kaw@eng.usf.edu

Website: http://numericalmethods.eng.usf.edu

Version: Maple 12; Last Revised: August 28, 2008

Introduction

The following worksheet illustrates how to convert a base-10 real number to a base-2 floating point binary representation using loops and various conditional statements. The user inputs a base-10 number in any format, the total number of bits, and the number of bits for the mantissa in the *Input* section of the program. The program will then convert the base-10 number into a floating point binary representation by first converting the given number into a number of the format 1.xxx * 2^m where *m* represents the exponent of the number.

Initialization

```
> restart; with(MTM) : with(ArrayTools) :
```

Section 1: Input

This is the only section where the user interacts with the program.

The generalized formula for floating point format in base-10 is given by $y = \sigma \cdot m \cdot 10^e$

Enter a number to be converted to a floating point binary number

```
> dec_num := 5.8946
                                   dec_num := 5.8946                                   (3.1)
```

Enter the total number of bits to be used (1st bit will be used for the sign of the number, 2nd bit will be used for the sign of the exponent.)

```
> tot_bits := 9
                                   tot_bits := 9                                       (3.2)
```

Enter number of bits for mantissa (*m*)

```
> mant_bits := 4
                                   mant_bits := 4                                      (3.3)
```

This is the end of the user section. All information must be entered before proceeding to the next section. **RE-EXECUTE THE PROGRAM.**

Section 2: Procedure

The number of bits to be used for the exponent will be the total number of bits minus the number of bits used for the mantissa, minus the bits used for the sign of the decimal number, and minus the bit used for the sign of the exponent.

```
> exp_bits := tot_bits - mant_bits - 1 - 1;
```

$$exp\_bits := 3 \quad \textbf{(4.1)}$$

To find the sign of the decimal number (sigma), and the sign of the exponent, we test to see if the number is negative and if it is less than 1.

```
> if dec_num > 0 then
  σ := "0";
  else
  σ := "1";
  end if;

  if abs(dec_num) < 1 and abs(dec_num) > 0 then
  exp_sign := "1";
  else
  exp_sign := "0";
  end if;
```

$$\sigma := "0"$$

$$exp\_sign := "0" \quad \textbf{(4.2)}$$

Calculating the maximum possible value given the number of bits as specified by the user.

```
> # Calculating the maximum value for the exponent.
  exp_sum := 0 :

  for i from 0 to exp_bits − 1 do
  exp_sum := exp_sum + 2^i;
  end do:

  # Calculating the maximum value for the mantissa.
    mant_sum := 1 :

  for i to mant_bits do
  mant_sum := mant_sum + 2^(-i);
  end do:

  maxval := mant_sum * 2^(exp_sum) :
  minval := mant_sum * 2^(-exp_sum) :
  printf("\nThe magnitude of the minimum number that can be represented in floating point
       representation given the number of bits is minval = %g.", minval);
       printf("\nThe magnitude of the maximum number that can be represented in floating
       point representation given the number of bits is maxval = %g.", maxval);
```

```
The magnitude of the minimum number that can be represented in floating
point representation given the number of bits is minval = 0.0151367.
The magnitude of the maximum number that can be represented in floating
point representation given the number of bits is maxval = 248.
```

If the maximum value that can be produced given the user specified number of bits is smaller than the number to be represented, then more bits are needed. Similarly, if the minimum value that can be produced given the user specified number of bits is larger than the number to be represented, a change in bits is needed.

```
> if maxval < dec_num  then
  printf("\nSince |%f| > %f, the number of bits specified is not sufficient,dec_num,maxval to
      represent this number in floating point representation.", dec_num, maxval);
  printf("Either specify fewer mantissa bits, or more total bits for the worksheet.");
  elif minval > dec_num then
  printf("Since %g > |%g|, the number of bits specified is not sufficient to represent this
      number in floating point binary format. Either specify fewer mantissa bits, or more total
      bits for the worksheet.", minval, dec_num);
  else
      printf("\nSince %g < |%g| < %g the base−10 number can be represented in floating
      point binary format.", minval, dec_num, maxval);
  end if;
```

```
Since 0.0151367 < |5.8946| < 248 the base-10 number can be represented in
floating point binary format.
```

We will convert the given variable *dec_num* into some number 1.xxx * 2^m , where *m* is the value for the exponent. First, we will find this value *m* for the exponent using the properties of logarithms. Since the exponent value is now known, it is now possible to find the fractional portion of the decimal number (1.xxx). To isolate the fractional portion, we simply subtract one from the *mant_val* variable.

```
> dec_num := abs(dec_num) :
  exp_val := floor(log(dec_num)/log(2)) :
  mant_val := dec_num/2^exp_val:
  mant_frac := mant_val - 1 :
```

Using loops to approximate the mantissa up to the specified number of mantissa bits.

```
> Mantstr := "" :
  for i to mant_bits do
  new_mant_frac := mant_frac * 2;
  Mant[i] := floor(new_mant_frac);
  mant_frac := new_mant_frac - Mant[i];
  Mantissa[i] := convert(Mant[i], string);
  Mantstr := cat(Mantstr, Mantissa[i], "|");
  end do:
```

Approximating the exponent value based on the calculated value of *m*. Then approximating the floating point values for the exponent value *m*.

```
> exp_val := abs(exp_val) :

  for i to exp_bits do
  new_exp_val := floor(exp_val/2);
  Bin_exp[i] := ceil(exp_val/2) - new_exp_val;
  exp_val := new_exp_val;
  end do:
```

Using this method, the floating point values approximated for the exponent are in reverse order. This section of the program reverses this and creates a character array with the existing values as well.

```
> tconv := convert(Bin_exp, list) :
```

```
t := Size(tconv, 2) :
Expstr := "" :
for i to t do
Binary_exp[i] := convert(Bin_exp[(t + 1) - i], string);
Expstr := cat(Expstr, Binary_exp[i], "|");
end do:
```

Concatenating all previously calculated binary components of the floating point binary number.

```
> Mantissa_comb := "" :
  for i to mant_bits do
  Mantissa_comb := cat(Mantissa_comb, Mantissa[i])
  end do:

  Binary_exp_comb := "" :
  for i to t do
  Binary_exp_comb := cat(Binary_exp_comb, Binary_exp[i])
  end do:

  disp(Sign of number entry = σ);
  disp(Sign of exponent entry = exp_sign);
  disp(Mantissa entry = Mantissa_comb);
  disp(Exponent entry =  Binary_exp_comb);
  Final_Form := cat("|", σ, "|", "  |", exp_sign, "|  |", Mantstr, "  |", Expstr) :
  disp(Floating Point Form = Final_Form);
```

```
                    Sign of number entry = "0"

                   Sign of exponent entry = "0"

                     Mantissa entry = "0111"

                     Exponent entry = "010"

        Floating Point Form = "|0|  |0|  |0|1|1|1|  |0|1|0|"            (4.3)
```

Conclusion

This worksheet illustrates the use of Maple to convert a base-10 number to a floating point binary representation. Recall that floating point representation is used more often than fixed point representation due to two primary advantages: floating point representation supports a much larger range of values while maintaining a relative error of similar magnitude for all numbers.

References

Floating Point Representation:
See: http://numericalmethods.eng.usf.edu/mws/gen/01aae/mws_gen_aae_txt_floatingpoint.pdf