Computer Numbers

```
> restart;Digits:=60;
```

$$Digits := 60$$

Some IBM compatible micros once used 64 binary (0 or 1) bits to represent a real number. An example is the following:

```
> r:="0 10000000111
  1011100101010000000000000000000000000000000000000000";
```

$$r := \text{"0 10000000111 1011100101010000000000000000000000000000000000000000"}$$

We use the length command to check the length of our string.

```
> length(r);
```

$$66$$

We are OK since we used two spaces. The first digit indicates the sign of the number, 0 for positive and 1 for negative.

```
> s:=0;
```

$$s := 0$$

The next group of 11 numbers gives an exponent called the characteristic. The characteristic c for our 11 numbers 10000000111 is

$$c = 1*2^{10} + 1*2^2 + 1*2 + 1*2^0.$$

Notice that the powers decrease from 10 to 0 going from left to right.

```
> c:=1*2^10+1*2^2+1*2^1+1*2^0;
```

$$c := 1031$$

You can see the pattern. The largest possible characteristic is

```
> largest_c:=sum(1*2^i,i=0..10);
```

$$largest_c := 2047$$

In forming a computer real, we subtract 1023 from the characteristic, giving exponents ranging from -1023 to 1024.

The final group of 52 numbers 1011100101010000000000000000000000000000000000000000 is a binary fraction called the mantissa. Here the mantissa is

$$f = 1\,\frac{1}{2} + 1\left(\frac{1}{2}\right)^3 + 1\left(\frac{1}{2}\right)^4 + 1\left(\frac{1}{2}\right)^5 + 1\left(\frac{1}{2}\right)^8 + 1\left(\frac{1}{2}\right)^{10} + 1\left(\frac{1}{2}\right)^{12}$$

```
> f:=1*(1/2)^1+1*(1/2)^3+1*(1/2)^4+1*(1/2)^5+1*(1/2)^8+1*(1/2)
  ^10+1*(1/2)^12;
```

$$f := \frac{2965}{4096}$$

We change to a decimal fraction.

```
> df:=evalf(f);
```

$$df := 0.723876953125000000000000000000000000000000000000000000000000$$

The largest possible mantissa is

```
> largest_f:=sum(1*(1/2)^i,i=1..52);
```

$$largest_f := \frac{4503599627370495}{4503599627370496}$$

Again we change this to a decimal fraction.

```
> dlargest_f:=evalf(largest_f);
```

$$dlargest_f := 0.99999999999999977795539507496869191527366638183593750000000$$

Using the sign, characteristic, and mantissa, the real number is then formed as

$$real = (-1)^s \, 2^{(c-1023)} \, (1+f)$$

The 1 added to the f is a normalization that saves storage and provides a unique representation for each floating point number.

```
> computer_real:=evalf((-1)^s*2^(c-1023)*(1+f));
```

$$computer_real := 441.31250000000000000000000000000000000000000000000000000000$$

Let's find the next smallest computer real.

```
> smaller_r:="0 10000000111
  10111001010011111111111111111111111111111111111111111111";
```

$$smaller_r := \text{"0 10000000111 10111001010011111111111111111111111111111111111111111111"}$$

```
> f:=1*(1/2)^1+1*(1/2)^3+1*(1/2)^4+1*(1/2)^5+1*(1/2)^8+1*(1/2)^10+
  sum(1*(1/2)^i,i=13..52);
```

$$f := \frac{3260051976355839}{4503599627370496}$$

```
> df:=evalf(f);
```

$$df := 0.72387695312499977795539507496869191527366638183593750000000$$

```
> smaller_computer_real:=evalf((-1)^s*2^(c-1023)*(1+f));
```

$$smaller_computer_real := 441.31249999999994315658113919198513031005859375000000000000$$

Now let's find the next largest computer real.

```
> larger_r:="0 10000000011
  10111001010100000000000000000000000000000000000000000001";
```

$$larger_r := \text{"0 10000000011 10111001010100000000000000000000000000000000000000000001"}$$

```
> f:=1*(1/2)^1+1*(1/2)^3+1*(1/2)^4+1*(1/2)^5+1*(1/2)^8+1*(1/2)
  ^10+1*(1/2)^12+1*(1/2)^52;
```

$$f := \frac{3260051976355841}{4503599627370496}$$

```
> ff:=evalf(f);
```

$$ff := 0.72387695312500022204460492503130808472633361816406250000000$$

```
> larger_computer_real:=evalf((-1)^s*2^(c-1023)*(1+f));
```

$$larger_computer_real := 441.31250000000005684341886080801486968994140625000000000000$$

In general, one computer real represents half the real numbers between it and its two nearest neighbors.

We jump the value of Digits to 400 for the next part of the worksheet;

```
> Digits:=400;
```

$$Digits := 400$$

The smallest normalized positive real is

```
> "0 00000000000
  0000000000000000000000000000000000000000000000000001";
```

"0 00000000000 0000000000000000000000000000000000000000000000000001"

and evaluates to

```
> smallest_pos_real:=evalf(2^(-1023)*(1+2^(-52)));
```

smallest_pos_real :=

1.112536929253600938577939279289474120394004424341852097806565015605984430199800\
34826489521461063144293195185068351409540085856480363559551775636137065875760 99\
52787002156940228390161668876994081958869366447962303711463505653902690669854 08\
26680648223724764894760609489545391926291682950925894809399942575213757391318 03\
80108402181101556463502274160369824228565519561129819678041612203119484536384 45\
09336 10^{-308}

Numbers with a magnitude less than this result in underflow and are usually set to 0.

The largest positive real is

```
> "0 11111111111
  11111111111111111111111111111111111111111111111111";
      "0 11111111111 11111111111111111111111111111111111111111111111111"
```

and evaluates to

```
> largest_pos_real:=evalf(2^(1024)*(2-2^(-52)));
```

largest_pos_real :=

3.595386269724631416290548474634087135961411350516899931978349536063145215600570\
77521179117265533756343080917907028764928468642653778928365536935093407075033 97\
20998211531025641524909801807786578881517370169102678846091664738064458963316 17\
11866424669654959565240828944633747635436183859976250080805236824971673 6 10^{308}

Numbers larger than this in magnitude result in overflow and typically cause computations to halt.

A natural question would be to ask how many computer reals such a computer has.