Base 2 fractions

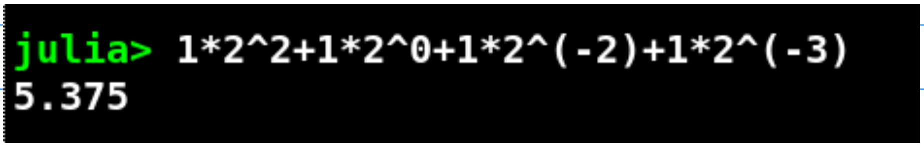
$b_n \cdots b_3 b_2 b_1 b_0 \bullet b_{-1} b_{-2} b_{-3} b_{-4} \cdots$ for $b_i \in \{0,1\}$

Example

$$\underset{2^2}{1}\underset{2^1}{0}\underset{2^0}{1}.\underset{2^{-1}}{0}\;\underset{2^{-2}}{1}\;\underset{2^{-3}}{1} = \boxed{\sum b_k 2^k}$$

$$= 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 1\cdot 2^{-3}$$

$$= 5.375$$

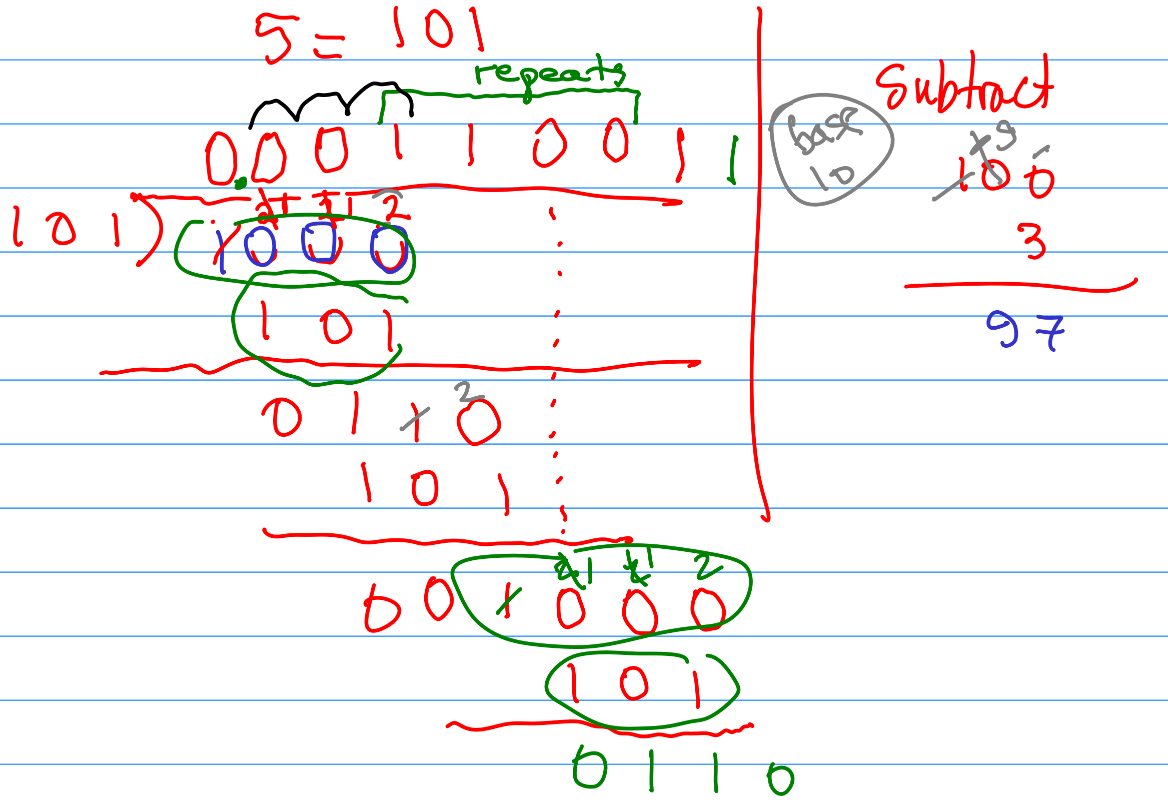


```
julia> 1*2^2+1*2^0+1*2^(-2)+1*2^(-3)
5.375
```

Example 2

What is $\frac{1}{5}$ as a binary number?

binary
0
1
10
11
100
101



$$\frac{1\,0\,1}{1}$$

$$\frac{1}{5} = .\overline{0011}$$

$$= .\underbrace{0011\ 0011\ 0011\ 0011\ldots}_{53 \text{ bits}}$$

don't store those

53 bits mantissa

double precision arithmetic.

for 64-bit float

means 8 bytes... each byte is 8 bits

64

53 = 52 mantissa + 1 sign bit

11 bits left for the exponent

Base 2

$$\overbrace{1.\underbrace{1\,00\,11\,00\,11\,00}_{\text{only need to store 52 bits}}}^{53 \text{ bits}} \times 2^{-3}$$

integer binary...

11 bits

2^{11} different bit patterns

2048 different bit patterns

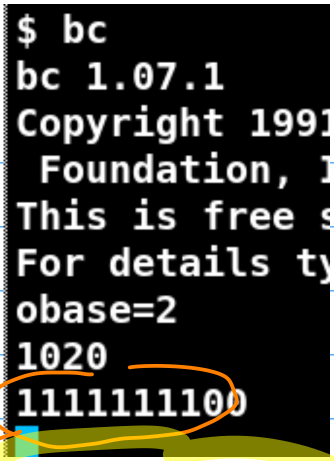
1023 → 0 exponent

1022 → -1 exponent

1020 → -3 exponent

0 → -1023 exponent

1020 =



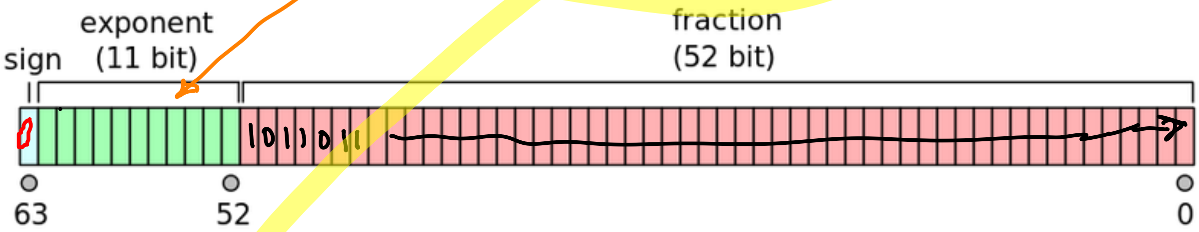
```
$ bc
bc 1.07.1
Copyright 1991
 Foundation, I
This is free s
For details ty
obase=2
1020
1111111100
```

I had an error in the lecture with the sign bit. For a positive number the bit should be 0.

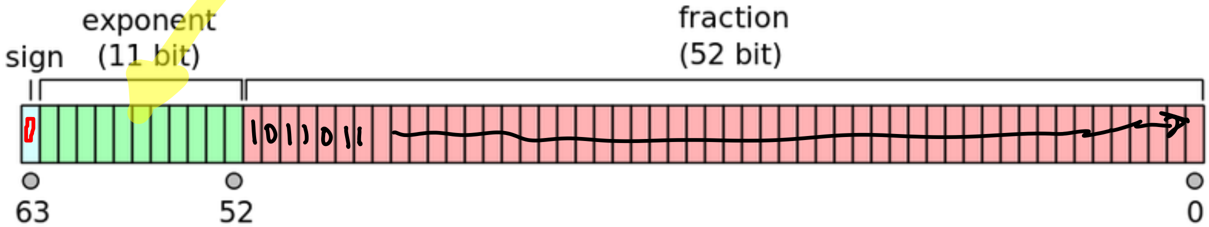
11011

11111111011 ≈ -4 exponent

$\frac{1}{5}$ ≈



$\frac{1}{10}$ ≈



Note that since there is always an implicit 1 assumed at the start of the mantissa, it is impossible to store the number 0 without further provisions.

It is convenient if all bits zero correspond to the floating point number zero.

That's the reason we use a bias for the exponent rather than, for example, a 2's compement representation as is more common when storing signed integers.

The bias means that when memory is zeroed the 11 bits for the exponent are interpreted as -1023. Thus, without any other provisions making all bits zero corresponds to the floating point number closest to zero. Because true zero is important in, the computer rounds this number to exact zero when it is used in calculations.

Although 2^-1023 is not representable as a stored value, there has to be a limit anyway on the smallest non-zero number that is possible to store. Treating this value as exact zero then is a sensible tradeoff and why the exponent is biased like it is.

Again, the bias of 1023 in the representation of the exponent in the floating point numbers stored on a computer allow the bit pattern of all zeros to be naturally treated as exact zero in actual calculations.