Converting binary-decimal / decimal-binary

Recall you won't have access to online resources during CBTF quizzes. So you should be familiar with conversion methods that don't rely on online calculators or other resources.

If you need to refresh your memory on how to do these conversions "by-hand", take a look at these slides: https://courses.grainger.illinois.edu/cs357/sp2024/assets/lectures/complete-slides/4a-Binary-Numbers.pdf (https://courses.grainger.illinois.edu/cs357/sp2024/assets/lectures/complete-slides/4a-Binary-Numbers.pdf) (they were included in your Lecture 4a: Floating Point)

Integer to Binary

Integer numbers in a computer

From decimal to binary: $(39)_{10}$

Method 1:

#/2	Quotient	Remainder
39/2	19	1
19/2	9	1
9/2	4	1
4/2	2	0
2/2	1	0
1/2	0	1

In [1]:
```
def integer_to_binary(x, n=23):
    f = ''
    dig = 0
    while ((x>0) & (dig<n)):
        remainder = x%2
        x = int(x/2)
        f += str(remainder)
        dig += 1
    return f[::-1]

print(bin(2341))
integer_to_binary(2341)
```

```
0b100100100101
```

Out[1]:
```
'100100100101'
```

In [2]:
```
# another method
x = 120
print(bin(x))

iter = 0
bin_rep = ''
while iter < 1000:
    quotient = int(x/2)
    remainder = x - 2*quotient  # could also do x%2
    bin_rep += str(remainder)
    x = quotient
    if x == 0:
        break
    iter += 1

print(bin_rep[::-1])
```

```
0b1111000
1111000
```

Decimal Fraction to Binary

From decimal to binary: $(39.6875)_{10}$

Method 1:

Same as before for the integer part

$$(39)_{10} = (100111)_2$$

For the decimal part, use the following table:

#×2	Integer part	Fractional part
1.375	1	0.375
0.75	0	0.75
1.5	1	0.5
1.0	1	0

$$(39.6875)_{10} = (100111.1011)_2$$

In [4]:
```
def decimal_to_binary(fdec, n=23):
    f = ''
    dig = 0
    while ((fdec>0) & (dig<n)):
        fdec = fdec*2
        temp_int = int(fdec)
        fdec = fdec - temp_int
        f += str(temp_int)
        dig += 1
    return f

x = 0.6875
decimal_to_binary(x)
```

Out[4]: '1011'

Normalized Floating Point

Normalized Floating Point

$$x = \pm 1.\, b_1 b_2 b_3 \cdots b_n \times 2^m$$

m → exponent, $m \in [L, U]$

precision ⇒ $p = n+$

n bits : fractional part of significand

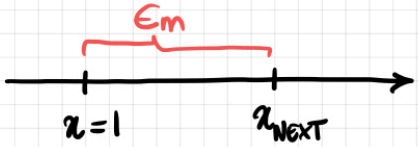
★ Smallest normalized FP

$$|x_{SM}| = 1.\underbrace{00000\ldots00}_{n} \times 2^L = 2^L$$

★ Largest normalized FP

$$|x_{LG}| = 1.\underbrace{11111\ldots11}_{n} \times 2^U = 2^{U+1}(1 - 2^{-p})$$

★ Machine Epsilon



$$x = 1.0000\ldots00 \times 2^0 = (1.0)_{10}$$

$$x_{NEXT} = 1.0000\ldots01 \times 2^0$$

$$\epsilon_m = |x_{NEXT} - x| = 0.\underbrace{0000\ldots01}_{n} \times 2^0$$

$$= 2^{-n}$$

★ Gap between two consecutive numbers

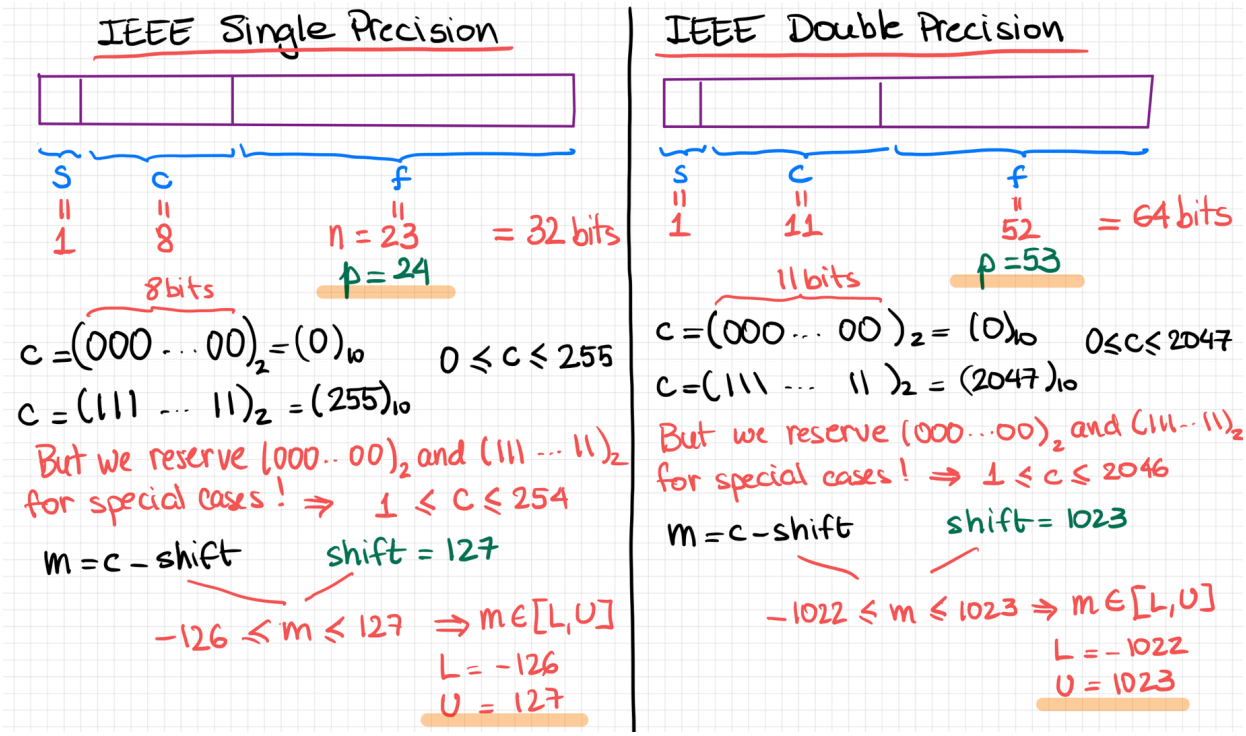
$$x^- = 1.\, b_1 b_2 b_3 \cdots b_n \times 2^m$$

$$x^+ = 1.\, b_1 b_2 b_3 \cdots b_n + 2^m + 0.000\ldots01 \times 2^m$$

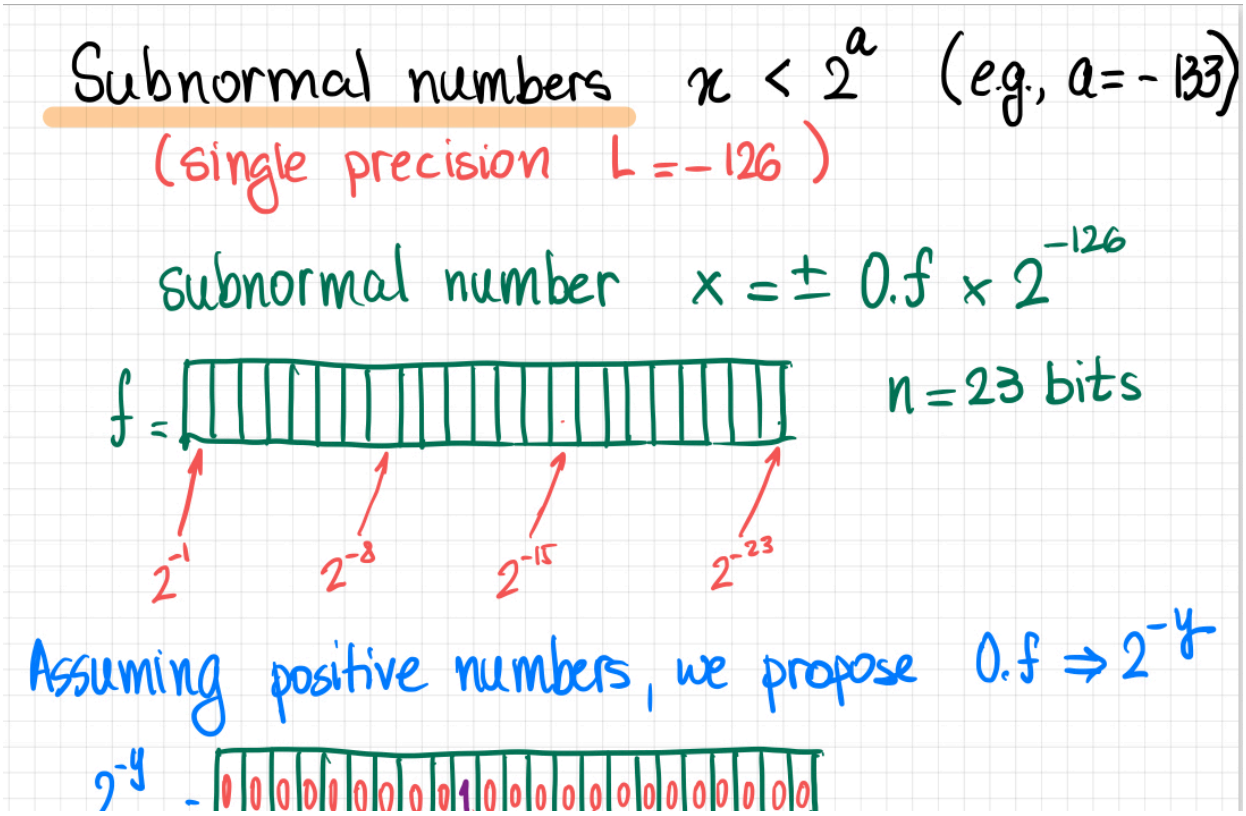
$$|x^+ - x^-| = \underbrace{0.0000\ldots01}_{2^{-n}} \times 2^m$$

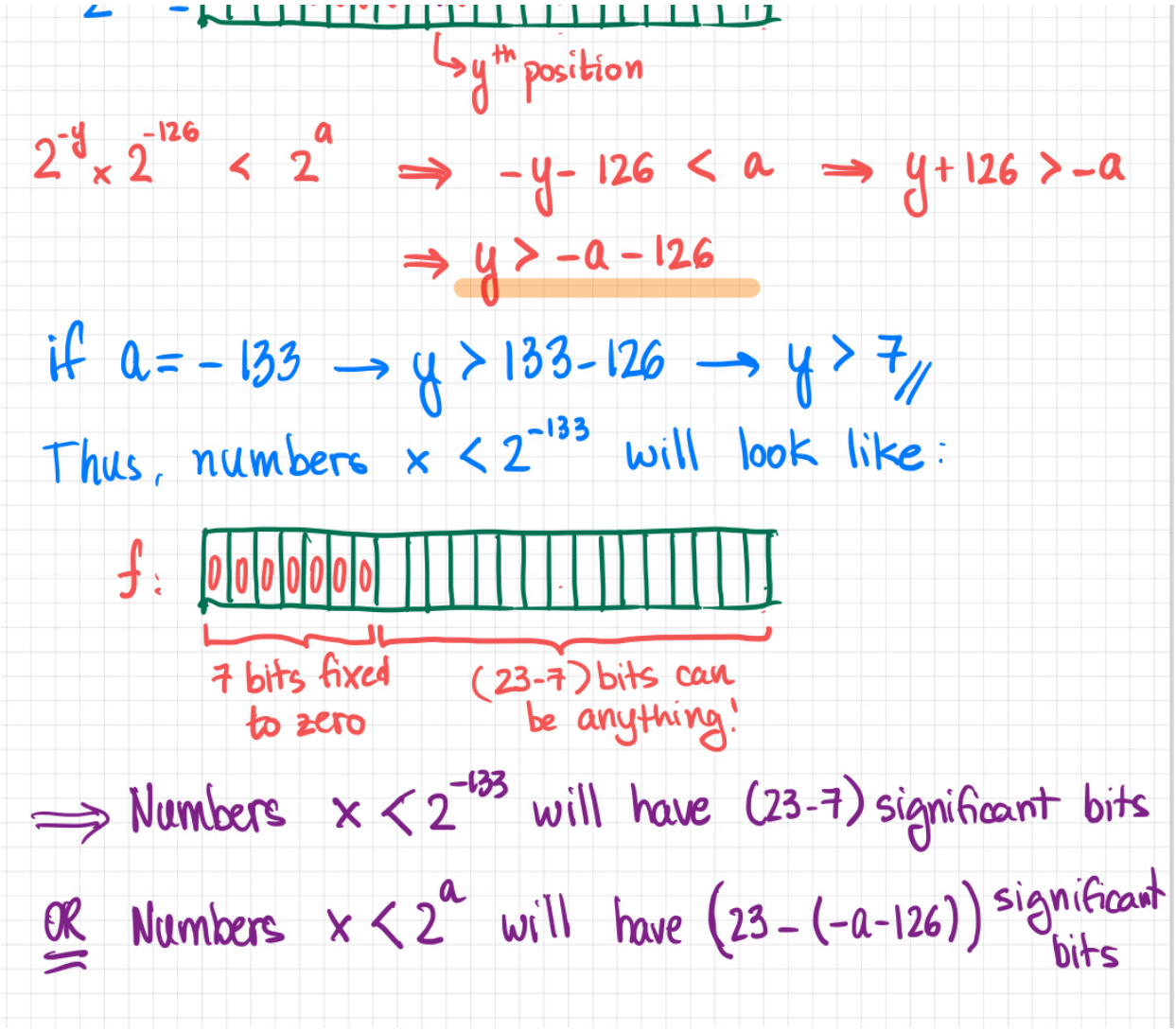
$$gap = 2^{-n} \times 2^m \Rightarrow gap = \epsilon_m \times 2^m$$

IEEE Double and Single Precision

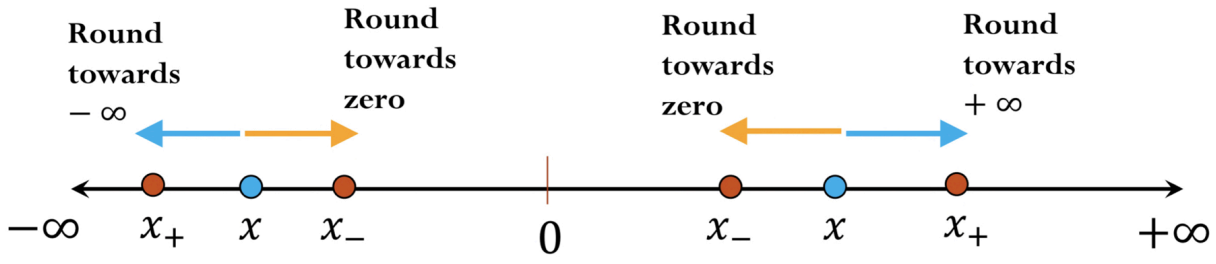


Subnormal Numbers





Rounding



In [ ]:

In [ ]: