CMSC 313

Binary Arithmetic

Kumaravel Jagasivamani
Jan. 31, 2024

Overview

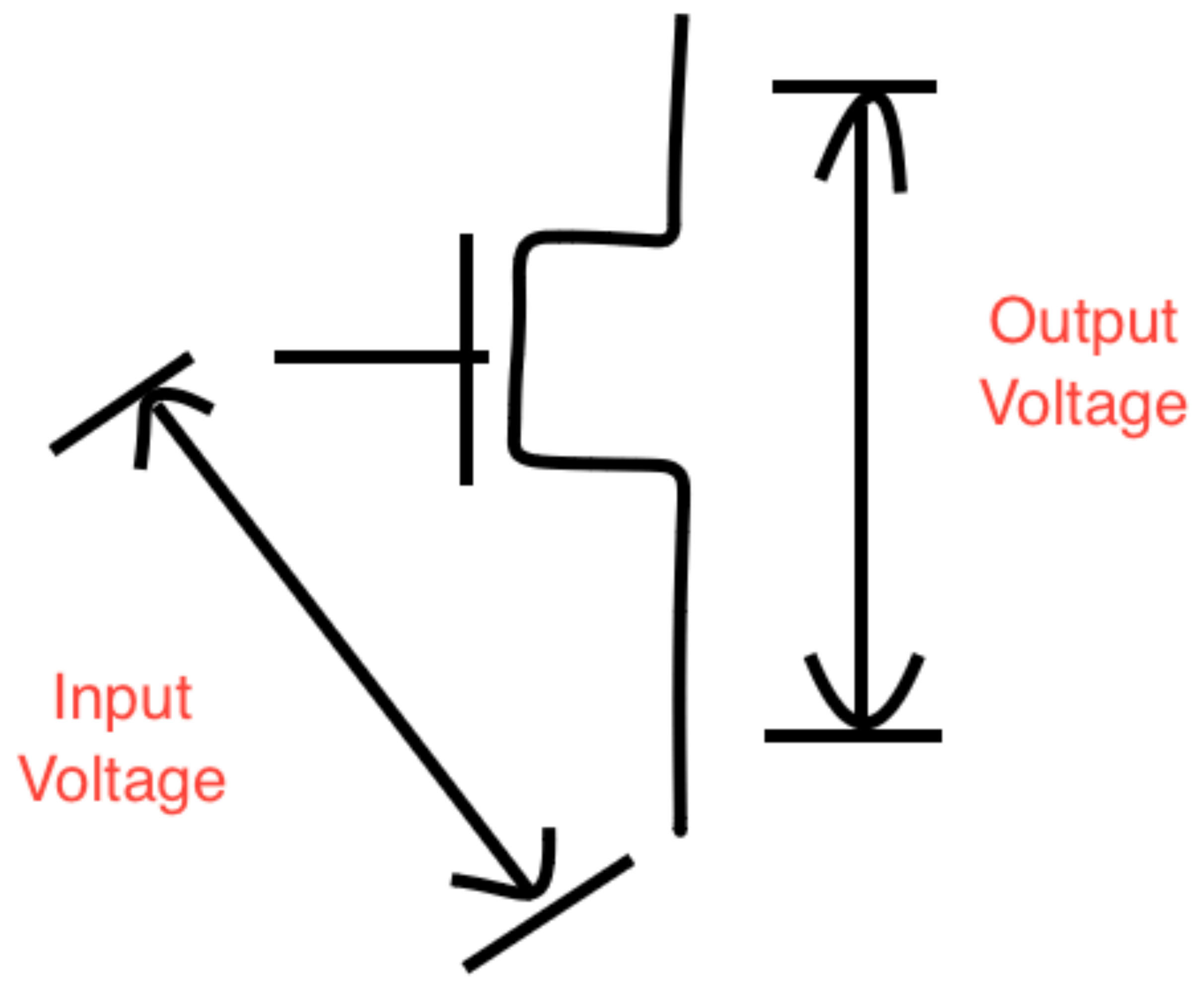
- Number systems
- Why Binary? (Optional)
- Decimal representation
- Binary representation
- Conversion between decimal and binary
- Hexadecimal representation and conversion
- Binary addition
- 2’s complement
- Binary subtraction
- Overflow
- Fractional numbers
- Floating point numbers

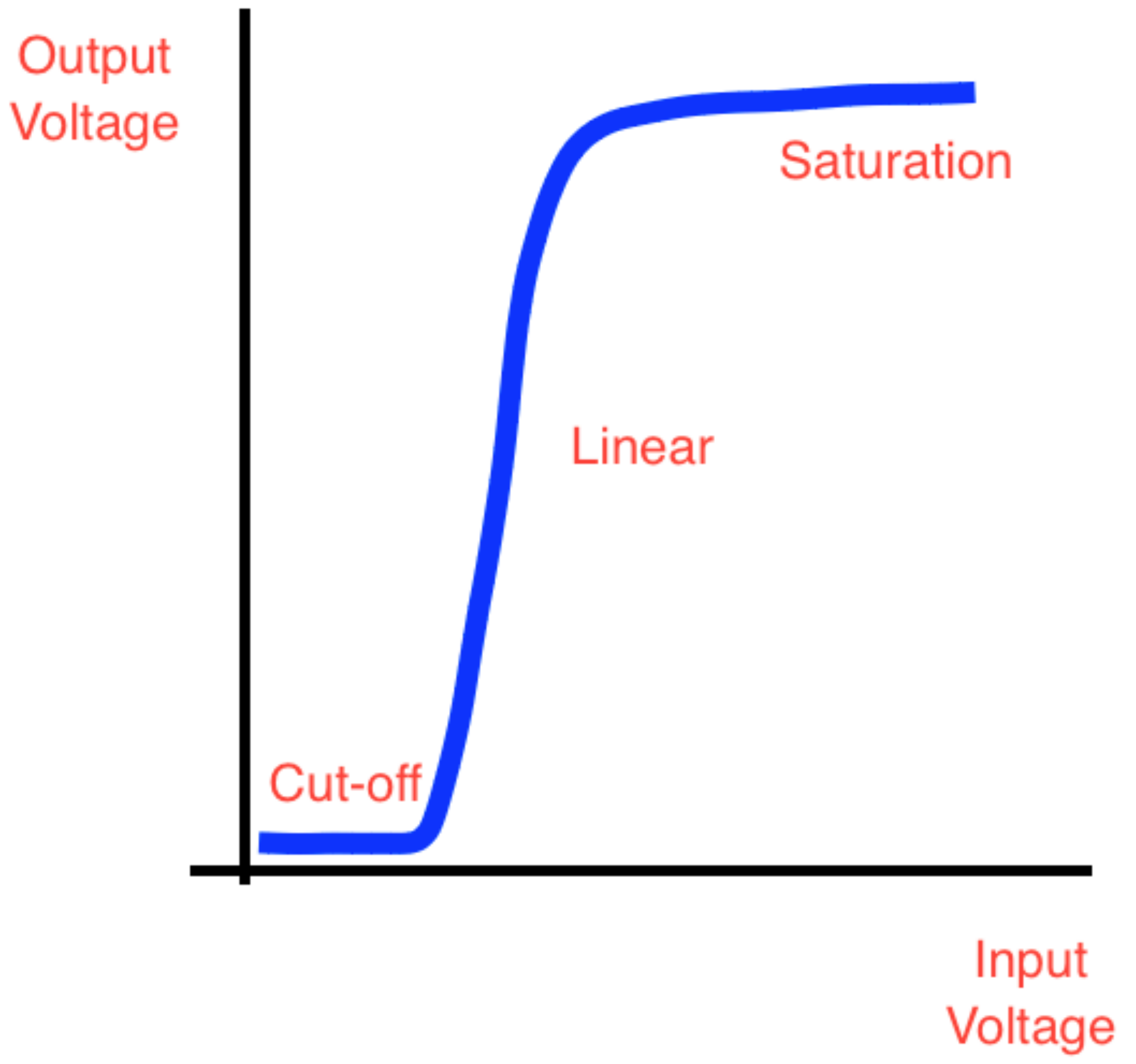
Number systems

- Base of number system: number of digits
- Decimal: 10
  - Why decimal is common in everyday life?
- Binary: base 2
  - Why is binary useful for computer systems?

Why Binary number system?
(Optional)

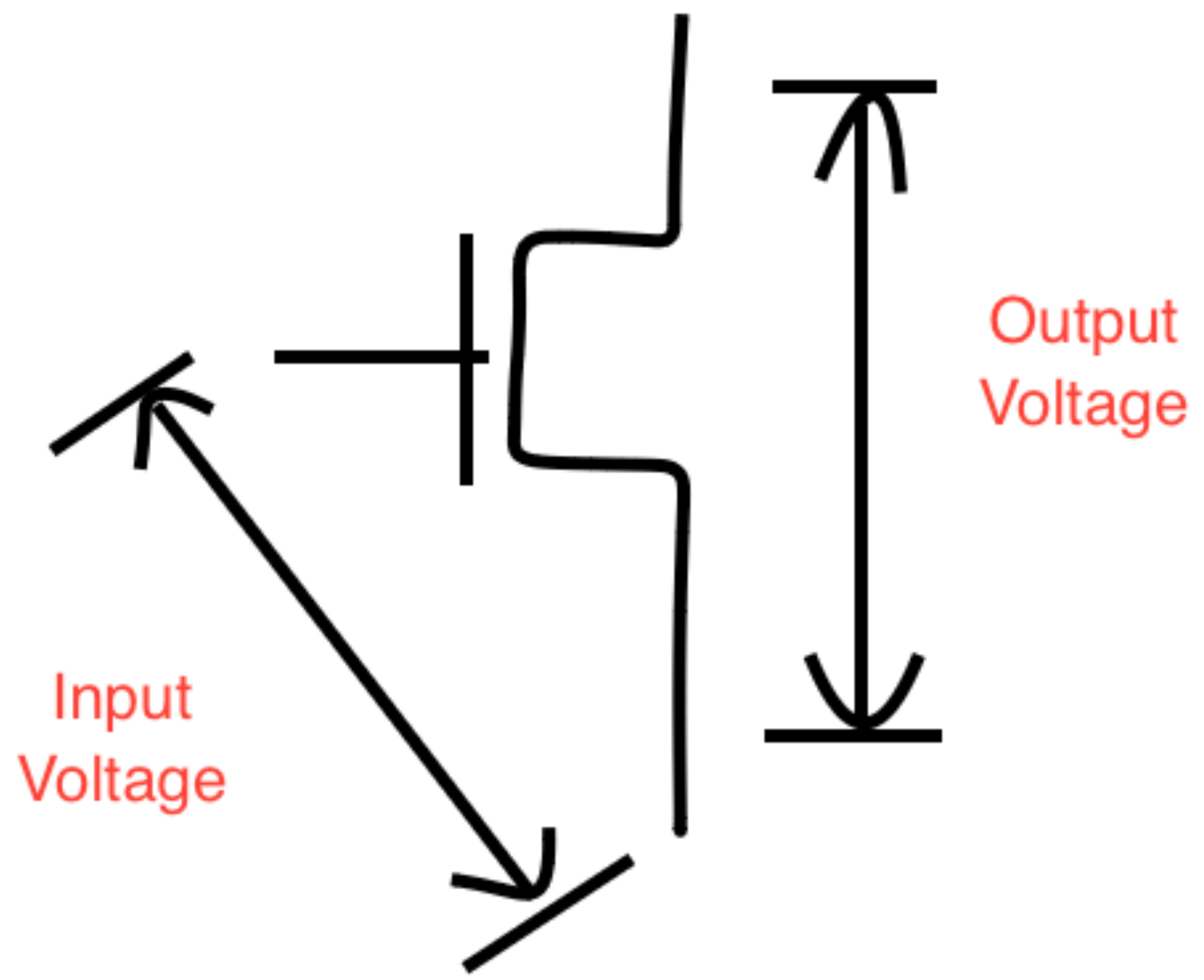
- Building block of processors is **transistor**
- Output voltage of transistor determines signal ("variable") value
- Transistor is an analog device
    - Output voltage can range from 0 V to supply voltage
        - Dependent on input voltage
    - One option for storing information:
        - Scale transistor voltage to desired range
        - Problems with this approach?

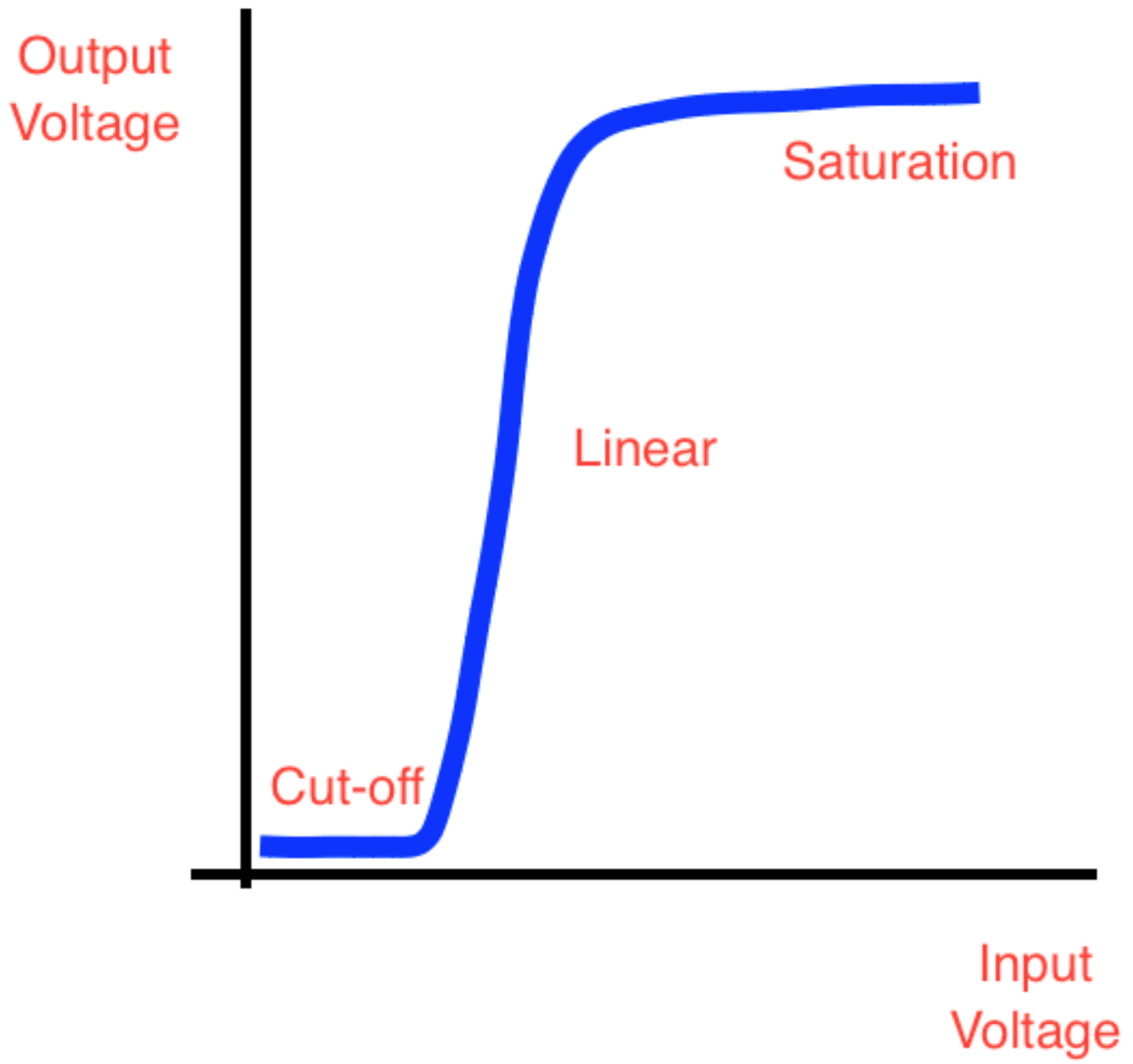




Why Binary number system?
(Optional)

- Building block of processors is transistor
- Output voltage of transistor determines signal (“variable”) value
- Transistor is an analog device
    - Output voltage can range from 0 V to supply voltage
        - Dependent on input voltage
    - One option for storing information:
        - Scale transistor voltage to desired range
        - Problems with this approach?
- More accurate to identify values with on/off metric
    - Digital vs. Analog
    - Transistor turned off or saturated
- Group multiple transistors together to get desired range of output





Decimal value example

- Place value of nth digit position: 10^{n-1}
    - n^{th} digit referred as 1s (10^0), 10s (10^1), 100s (10^2), … digit
- Show derivation of 1s, 10s, 100s units’ digit
    - 846_{10}

Decimal to Binary value

- Place value of bit n (n starting from 0): 2^n
- Show derivation of binary value
  - 27_{10}
  - 38_{10}

Bit	Place Value
0	1
1	2
2	4
3	8

Binary to Decimal value

- Show derivation
    - 11011_2
    - 100110_2

Hexadecimal value

- Digits: 0-9, A, B, C, D, E, F
- Group 4 binary bits together
  - Easier to convey information:
    - B_{16} instead of 1011_2
    - 63_{16} instead of 01100011_2
  - Number of units reduced going from binary to hexadecimal
    - Requires same number of bits to store information

Converting to/from hexadecimal

- Hexadecimal to binary:
  - Ungrouping
- Binary to Hexadecimal:
  - Grouping
- Hexadecimal to decimal:
  - Place value of position n (n starting from 0): 16^n
- Decimal to hexadecimal:
  - Can also convert to binary first

Hexadecimal conversion examples

- B_{16}
- 1011_2
- 63_{16}
- 01100011_2
- 27_{10}
- 38_{10}
- 300_{10}
- $12C_{16}$

Binary Addition

- Adding 2 numbers:
  - Sum is same bit position
  - Carry goes to next bit position
    - Multiplied by base of number system
- Examples:
  - 0101 + 0110
  - 1011 + 0111

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Binary Addition

3 inputs

- Sum = 1:
  - Odd number of inputs are 1
- Carry Out = 1:
  - 2 or more inputs are 1
- Examples:
  - 0101 + 0110 + 1
  - 1011 + 0111 + 1

A	B	Carry In	Sum	Carry Out
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1

2's Complement of a number

- 1's complement of a number:
  - Bit inversion
  - Example of 5 (0101_2)
- 2's complement of a number:
  - Bit inversion + 1
  - Example of 5 (0101_2)
  - Example of 7 (0111_2)
  - Most significant bit is 1
- Example of 9 (1001_2)
  - 2's complement value sign bit is not 1
    - Why?

2's Complement of a number

- 1's complement of a number:
    - Bit inversion
    - Example of 5 (0101_2)
- 2's complement of a number:
    - Bit inversion + 1
    - Example of 5 (0101_2)
    - Example of 7 (0111_2)
    - Most significant bit is 1
- Example of 9 (1001_2)
    - 2's complement value sign bit is not 1
        - Why?
    - Most significant bit needs to be 0 before 2's complement
        - Need to zero-extend binary value by 1 bit first
- Sign bit: most significant bit
    - 0: Positive
    - 1: Negative

Binary subtraction

- 7 - 4
- 36 - 18
- 18 - 36
-

Why 2’s complement?

Why not 1’s complement?

- 7 - 4 (1’s complement): result is 2 (1 less)
- 4 - 7 (1’s complement): result is -2 (1 more)
- 2’s complement
  - Output matches expected value
  - Same operation can be used to convert numbers
    - Positive to Negative
    - Negative to Positive

Overflow in 2’s complement

- 2’s complement 4-bit value
  - Bit 3: sign bit
- 7 + 7:
  - Both inputs are positive
  - Output is negative
- -7 -7:
  - Both inputs are negative
  - Output is positive
- How to avoid overflow?
- What is the range of allowed values?

Overflow in 2’s complement

- 4-bit 2’s complement value
  - Bit 3: sign bit
- 7 + 7:
  - Both inputs are positive
  - Output is negative
- -7 -7:
  - Both inputs are negative
  - Output is positive
- Avoiding overflow:
  - Sign-extend before addition
  - Be aware of range
    - 4-bit 2’s complement value: -8 to +7

Fractional numbers

- Place value of digits after “decimal point”:
  - 1st fractional bit: 2^{-1}
  - 1st fractional bit: 2^{-2}
  - nth fractional bit: 2^{-n}
- Converting from decimal to binary:
  - Multiply by 2 and check whole number
- Converting from binary to decimal:
  - Multiply by 2^{-n} and add

Floating Point Numbers

- Signed magnitude format (Not 1s/2s complement)
- 32-bit floating point number:
    - 1-bit Sign bit
    - 8-bit Exponent
        - Bias of 127 (Add 127 to exponent value)
    - 23-bit Mantissa
- Example
    - -3.5:
        - Sign bit 1
        - Binary value: 11.1 = 1.**11** * 2^{1}
        - Exponent: **1 + 127 = 128 = 10000000**
        - Mantissa: 110000…000
            - Mantissa doesn’t include the “1.” part of 1.11 * 2^{1} binary value
        - Final value for 32-bit floating point number = {1-bit sign, 8-bit exponent, 23-bit mantissa} = 1 10000000 110000…000 = 1100 0000 0110 0000 0000 0000 0000 0000_2
            - Hexadecimal value (optional): $C0600000_{16}$
    - 0 (special case): 00000000_{16} and 80000000_{16}