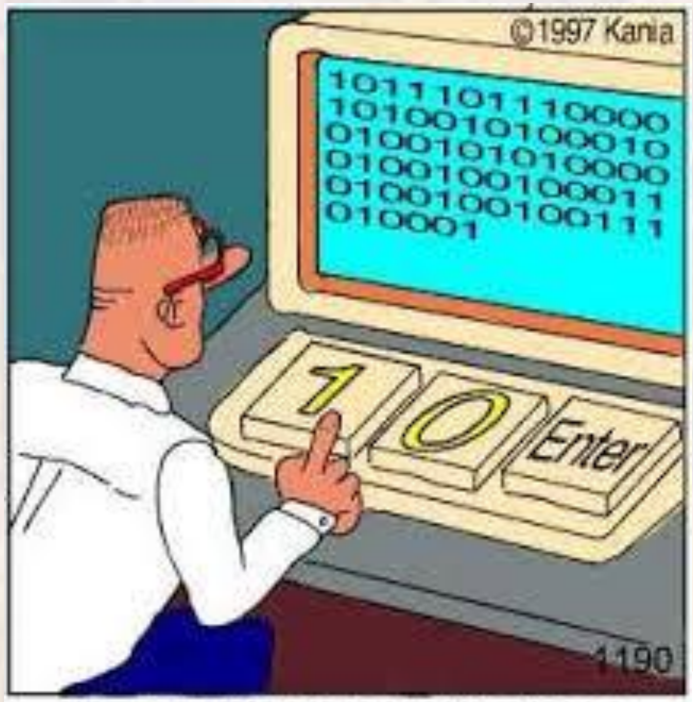
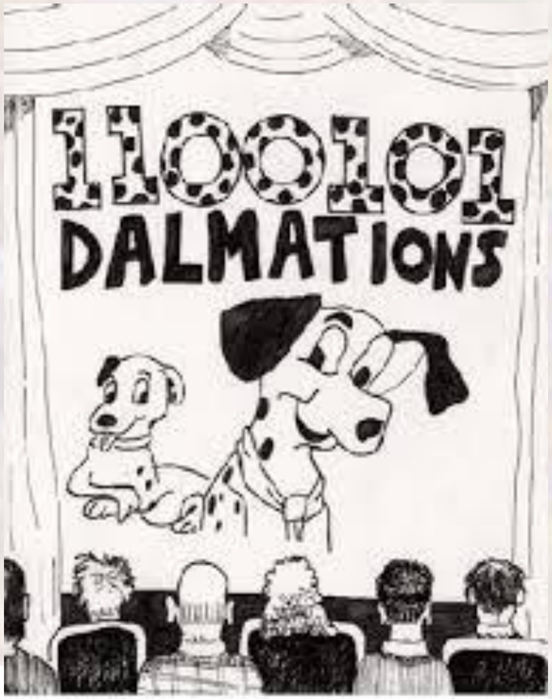


Real programmers code in binary.



Film Night at the Binary Society

# Number Systems

Common – last updated 12/11/19

# Number Systems

- Base 10 (decimal)
  - The most familiar base for most people
    - ones, tens, hundreds, thousands
    - tenths, hundredths, thousandths
  - Base 10 → 10 individual digits
    - Range of individual digit: 0 → 9
  - Each position to the left of the decimal point is 10X the previous position
  - Each position to the right of the decimal point is 1/10th the previous position

| 1 | 2 | 3 | 4 | . | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Thousands | Hundreds | Tens | Ones | decimal point | tenths | hundredths | thousandths |

| 1 | 2 | 3 | 4 | . | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| digit x $10^3$ | digit x $10^2$ | digit x $10^1$ | digit x $10^0$ | decimal point | digit x $10^{-1}$ | digit x $10^{-2}$ | digit x $10^{-3}$ |

# Number Systems

- Base 2 (binary)
  - The most common base for digital electronics
    - ones, twos, fours, eights
    - halves, quarters, eighths
  - Base 2 → 2 individual digits
    - Range of individual digit: 0 → 1
  - Each position to the left of the decimal point is 2X the previous position
  - Each position to the right of the decimal point is 1/2 the previous position

| 1 | 1 | 0 | 1 | . | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| Eights | Fours | Twos | Ones | binary point | Halves | Quarters | Eighths |

| 1 | 1 | 0 | 1 | . | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| digit x $2^3$ | digit x $2^2$ | digit x $2^1$ | digit x $2^0$ | binary point | digit x $2^{-1}$ | digit x $2^{-2}$ | digit x $2^{-3}$ |

# Number Systems

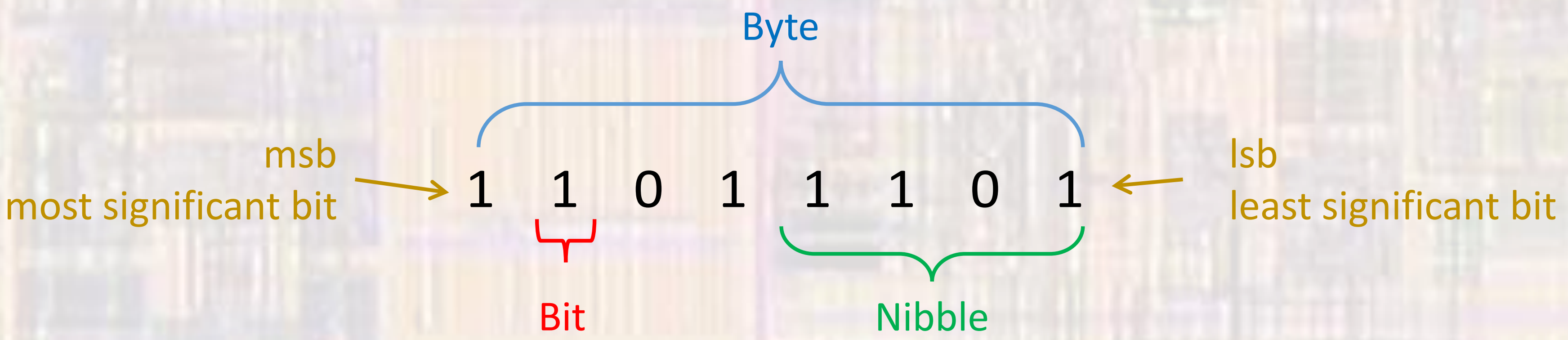
- Base 16 (hexadecimal)
  - Used as a short hand for binary
    - ones, 16s, 256s, 4096s
    - 16ths, 256ths
  - Base 16 → 16 individual digits
    - Range of individual digit: 0 → 9,A→F
    - 10=A, 11=B, 12=C, 13=D, 14=E, 15=F
  - Each position to the left of the decimal point is 16X the previous position
  - Each position to the right of the decimal point is 1/16 the previous position

| 2 | B | 0 | E | . | 3 | A | 2 |
|---|---|---|---|---|---|---|---|
| 4096s | 256s | 16s | Ones | hexadecimal point | 16ths | 256ths | 4096ths |

| 2 | B | 0 | E | . | 3 | A | 2 |
|---|---|---|---|---|---|---|---|
| digit x $16^3$ | digit x $16^2$ | digit x $16^1$ | digit x $16^0$ | hexadecimal point | digit x $16^{-1}$ | digit x $16^{-2}$ | digit x $16^{-3}$ |

# Number Systems

- Binary Terminology



# Number Systems

- Binary Terminology

16 Bit Word

1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1

32 Bit Word

1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1

64, 128, 256, 512, 1024 Bit Words

# Number Systems

- Bit Values

Right to Left

| Bit # | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Value | 2,147,483,648 | 1,073,741,824 | 536,870,912 | 268,435,456 | 134,217,728 | 67,108,864 | 33,554,432 | 16,777,216 | 8,388,608 | 4,194,304 | 2,097,152 | 1,048,576 | 524,288 | 262,144 | 131,072 | 65,536 |

| Bit # | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| Value | 32,768 | 16,384 | 8,192 | 4,096 | 2,048 | 1,024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

1K
1M
1G

| Bit # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| . | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| Value | 0.5 | 0.25 | 0.125 | 0.0625 | 0.03125 | 0.015625 | 0.0078125 | 0.00390625 |

Left to Right

# Number Systems

- More Terminology

  - Assume S is an 8 bit binary number

    S = 10010110

    - S[7:0] = 10010110 — 10010110
    - S[3:0] = 0110 — 10010110
    - S[7:6] = 10 — 10010110
    - S[5] = 0 — 10010110
    - S[6,3] = 00 — 10010110
    - S[1] = 1 — 10010110
    - S[0] = 0 — 10010110

# Number Systems

- Unsigned Binary (Binary)

  - All *n* bits used to represent the magnitude of the value
  - No negative values
  - Often used as absolute memory addresses

| | | |
|---|---|---|
| 4 | → | 00000100 |
| 32 | → | 00100000 |
| 16 | → | 00010000 |

| | | |
|---|---|---|
| 50 | → | ? |
| $10010110_b$ | → | ? |
| 0.625 | → | ? |

# Number Systems

- Unsigned Binary (Binary)

convert 50 decimal to 8 bit unsigned binary

8 bits → bit values of 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1

greatest bit value ≤ 50 = 32 — 0 0 1

50 – 32 = 18

greatest bit value ≤ 18 = 16 — 0 0 1 1

18 – 16 = 2

greatest bit value ≤ 2 = 2 — 0 0 1 1 0 0 1

2 – 2 = 0

no more left — 0 0 1 1 0 0 1 0

# Number Systems

- Unsigned Binary (Binary)

convert 10010110 unsigned binary to decimal

8 bits → bit values of 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1

$1*128 + 0*64 + 0*32 + 1*16 + 0*8 + 1*4 + 1*2 + 0*1$

$128 + 16 + 4 + 2 = 150$

$10010110_b \rightarrow 150$

# Number Systems

- Unsigned Binary (Binary)

convert 0.625 decimal to unsigned binary

first few fractional bits → bit values of 1/2 | 1/4 | 1/8 | 1/16

0.5 0.25 0.125 0.0625

greatest bit value ≤ 0.625 = 1/2 . 1

0.625 - 0.5= 0.125

greatest bit value ≤ 0.125 = 1/8 . 1 0 1

0.125 – 0.125 = 0

no more left . 1 0 1 0 or 0.101

# Number Systems

- Unsigned Binary (Binary)

  - Maximum values: (non fractional)
    - 4 bits (1111) = 15
    - 8 bits (1111 1111) = 255
    - 16 bits (1111 1111 1111 1111) = 65,535
    - 32 bits (1111 1111 1111 1111 1111 1111 1111 1111) = 4,294,967,295

  - Wait! 4 bits → $2^4$ = 16, why is the maximum value 15

    8 bits → $2^8$ = 256, why is the maximum value 255

    …

# Number Systems

- Unsigned Binary (Binary)

- Wait! 4 bits → $2^4 = 16$, why is the maximum value 15

  8 bits → $2^8 = 256$, why is the maximum value 255

  …

- Zero is one of our values, that only leaves 15 more …

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| decimal | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| unsigned binary | 1 1 1 1 | 1 1 1 0 | 1 1 0 1 | 1 1 0 0 | 1 0 1 1 | 1 0 1 0 | 1 0 0 1 | 1 0 0 0 | 0 1 1 1 | 0 1 1 0 | 0 1 0 1 | 0 1 0 0 | 0 0 1 1 | 0 0 1 0 | 0 0 0 1 | 0 0 0 0 |

# Number Systems

- Signed Magnitude

  - MSB used to represent the sign of the value
    - MSB = 0 → positive
    - MSB = 1 → negative
  - Remaining bits represent the magnitude of the value
  - Used in most floating point number representations

50 → 0011 0010

-50 → 1011 0010

-37 →

$10010110_b$ signed magnitude →

# Number Systems

- Signed Magnitude

convert -37 decimal to 8 bit signed magnitude

8 bits → bit values of    s | 64 | 32 | 16 | 8 | 4 | 2 | 1

| | |
|---|---|
| s = negative | 1 |
| \|-37\| = 37 | |
| greatest bit value ≤ 37 = 32 | 1 0 1 |
| 37 – 32 = 5 | |
| greatest bit value ≤ 5 = 4 | 1 0 1 0 0 1 |
| 5 – 4 = 1 | |
| greatest bit value ≤ 1 = 1 | 1 0 1 0 0 1 0 1 |
| 1 – 1 = 0 | |

# Number Systems

- Signed Magnitude

convert 10010110 signed magnitude to decimal

8 bits → bit values of s | 64 | 32 | 16 | 8 | 4 | 2 | 1

0∗64 + 0∗32 + 1∗16 + 0∗8 + 1∗4 + 1∗2 + 0∗1

16 + 4 + 2 = 22

sign = 1 = negative → -22

$10010110_b$ signed magnitude → -22

# Number Systems

- Signed Magnitude

  - Maximum values: (non fractional)
    - 4 bits (s111) = $\pm 7$ = $2^3-1$
    - 8 bits (s111 1111) = $\pm 127$ = $2^7-1$
    - 16 bits (s111 1111 1111 1111) = $\pm 32{,}767$ = $2^{15}-1$

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

# Number Systems

- Signed Magnitude
  - Issues
    - Binary math is difficult with sign magnitude representation
    - 2 zeros really causes a lot of problems

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

# Number Systems

- One's Complement

  - Negative numbers are formed by flipping all bits
  - Most Significant Bit (MSB) represents the sign
    (but it is NOT a sign bit)
    - MSB = 0 → positive
    - MSB = 1 → negative
  - All bits are used to represent the magnitude of the value
  - Not widely used anymore – but a stepping stone to 2's complement

50 → 0011 0010
-50 → 1100 1101
-37 →
$10010110_b$ 1's comp →

# Number Systems

- One's Complement

convert -37 decimal to one's complement

8 bits → positive bit values of x | 64 | 32 | 16 | 8 | 4 | 2 | 1

s = negative → flip all bits at end

|-37| = 37

greatest bit value ≤ 37 = 32 — 0 0 1

37 – 32 = 5

greatest bit value ≤ 5 = 4 — 0 0 1 0 0 1

5 – 4 = 1

greatest bit value ≤ 1 = 1 — 0 0 1 0 0 1 0 1

1 – 1 = 0

Continued

# Number Systems

- One’s Complement

convert -37 decimal to one’s complement – cont’d

s = negative → flip all bits at end

0 0 1 0 0 1 0 1 → 1 1 0 1 1 0 1 0

-37 → 1 1 0 1 1 0 1 0 one’s complement

# Number Systems

- One's Complement

convert 10010110 one's complement to decimal

MSB is 1 (negative) → remember this for the end → flip the bits

10010110 → 01101001

8 bits → positive bit values of x | 64 | 32 | 16 | 8 | 4 | 2 | 1

$1*64 + 1*32 + 0*16 + 1*8 + 0*4 + 0*2 + 1*1$

$64 + 32 + 8 + 1 = 105$

MSB = 1 = negative → -105

$10010110_b$ 1's comp → -105

# Number Systems

- One's Complement
  - Maximum values:
    - 4 bits = $\pm 7$ = $\pm (2^3-1)$
    - 8 bits = $\pm 127$ = $\pm (2^7-1)$
    - 16 bits = $\pm 32{,}767$ = $\pm(2^{15}-1)$

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 | 1111 | 1110 | 1101 | 1100 | 1011 | 1010 | 1001 | 1000 |

# Number Systems

- One’s Complement
  - Issues
    - 2 zeros really causes a lot of problems

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 |
| 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 | 1111 | 1110 | 1101 | 1100 | 1011 | 1010 | 1001 | 1000 |

# Number Systems

- Two's Complement

  - Negative numbers are formed by flipping all bits and adding 1
  - Positive numbers are formed in normal binary format
  - Most Significant Bit (MSB) represents the sign

    (but it is NOT a sign bit)
    - MSB = 0 → positive
    - MSB = 1 → negative
  - All bits are used to represent the magnitude of the value
  - The dominant representation for binary arithmetic

| | | |
|---|---|---|
| 50 | → | 0011 0010 |
| -50 | → | 1100 1110 |

-37 →

$10010110_b$ 2's comp →

# Number Systems

- Two's Complement

convert -37 decimal to two's complement

8 bits → positive bit values of x | 64 | 32 | 16 | 8 | 4 | 2 | 1

s = negative → flip all bits and add 1 at end

|-37| = 37

greatest bit value ≤ 37 = 32 0 0 1

37 – 32 = 5

greatest bit value ≤ 5 = 4 0 0 1 0 0 1

5 – 4 = 1

greatest bit value ≤ 1 = 1 0 0 1 0 0 1 0 1

1 – 1 = 0

# Number Systems

- Two's Complement

convert -37 decimal to two's complement – cont'd

s = negative → flip all bits and add 1 at end

0 0 1 0 0 1 0 1 →(flip) 1 1 0 1 1 0 1 0 →(+ 1) 1 1 0 1 1 0 1 1

-37 → 1 1 0 1 1 0 1 1 two's complement

# Number Systems

- Two's Complement

convert 10010110 two's complement to decimal

MSB is 1 (negative) → remember this for the end
→ flip the bits and add 1 (works both directions)

10010110 $\xrightarrow{\text{flip}}$ 01101001 $\xrightarrow{+1}$ 01101010

8 bits → positive bit values of x | 64 | 32 | 16 | 8 | 4 | 2 | 1

$1*64 + 1*32 + 0*16 + 1*8 + 0*4 + 1*2 + 0*1$
$64 + 32 + 8 + 2 = 106$

sign = 1 = negative → -106

$10010110_b$ 2's comp → -106

# Number Systems

- Two's Complement

  - Maximum values:

    - 4 bits = +7, -8 = $2^3$-1, -$2^3$
    - 8 bits = + 127, -128 = $2^7$-1, -$2^7$
    - 16 bits = + 32,767, -32,768 = $2^{15}$-1, -$2^{15}$

    - Not Symmetric

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 | 1111 | 1110 | 1101 | 1100 | 1011 | 1010 | 1001 | 1000 |

# Number Systems

- Two's Complement
  - Advantages
    - Addition is done the same way as unsigned numbers – same adder circuit
    - ONLY 1 ZERO !
    - Simple word length extension
  - Disadvantages
    - Asymmetric range
    - Harder to do comparisons
    - Not intuitive

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 | 1111 | 1110 | 1101 | 1100 | 1011 | 1010 | 1001 | 1000 |

# Number Systems

- Two's Complement

  - Sign Extension

    - When extending to larger word sizes, extend the MSB to the left

| 4 bit | | 8 bit | | 16 bit |
|---|---|---|---|---|
| 0110 | → | 00000110 | → | 0000000000000110 |
| 1001 | → | 11111001 | → | 1111111111111001 |

this works for 1's complement also
not the same for signed magnitude: -1 = 1001 → 10000001 = -1

# Number Systems

- Two's Complement

  - Fast way to do 2's complement conversions

    - working from the right

      find the first 1 and leave it and all preceding 0's the same

      flip all remaining bits to the left

      remember the MSB value and set the sign

10010110  2's complement

10  - first 1 from the right

01101010  - all remaining bits flipped

106

- 106  - since we started with a MSB = 1 (negative)

# Number Systems

- Binary Coded Decimal (BCD)

  - Encode base 10 digits into 4 bit nibbles
  - No negative representation
  - Used in some financial applications

| | | |
|---|---|---|
| 50 | → | 0101 0000 |
| 79 | → | 0111 1001 |
| | | |
| 37 | → | BCD |
| $10010110_{BCD}$ | → | decimal |

# Number Systems

- Binary Coded Decimal

convert 37 decimal to BCD

4 bits → bit values of 8 | 4 | 2 | 1

3 → 0011 0 0 1 1

7 → 0111 0 0 1 1 0 1 1 1

37 → 0011 0111 BCD

# Number Systems

- Binary Coded Decimal

convert 10010110 BCD to decimal

Break into 4 bit nibbles

10010110 → 1001 0110

1001 → 9
0110 → 6

10010110 BCD → 96

# Number Systems

- Binary Coded Decimal

  - Maximum values:

    - 4 bits = 9
    - 8 bits = 99
    - 16 bits = 9999

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1001 | 1000 | 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 |

# Number Systems

- Binary Coded Decimal

  - Issues
    - No negative values
    - Not efficient – limited range

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1001 | 1000 | 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 | 0000 |

# Number Systems

- Representation Summary

| | Unsigned Binary | Signed Magnitude | 1's Complement | 2's Complement | BCD |
|---|---|---|---|---|---|
| 50 | 0011 0010 | 0011 0010 | 0011 0010 | 0011 0010 | 0101 0000 |
| -50 | N/A | 1011 0010 | 1100 1101 | 1100 1110 | N/A |

| 0110 1001 Unsigned Binary | 0110 1001 Signed Magnitude | 0110 1001 1's Complement | 0110 1001 2's Complement | 0110 1001 BCD |
|---|---|---|---|---|
| 105 | 105 | 105 | 105 | 69 |

| 1001 0110 Unsigned Binary | 1001 0110 Signed Magnitude | 1001 0110 1's Complement | 1001 0110 2's Complement | 1001 0110 BCD |
|---|---|---|---|---|
| 150 | -22 | -105 | -106 | 96 |

# Number Systems

- Special note on binary numbers in C programming

  - Some but not all compilers allow binary numbers to be represented in C code directly

    95 → 0b01011101

  - To be safe and ensure our code is portables we will NOT use this notation.
  - Binary numbers can be represented with:
    - Their decimal equivalents 95
    - Their hexadecimal equivalents 0x5D

# Number Systems

- Hexadecimal
  - Group sets of 4 binary bits
  - 0-9
    - Represent them with their decimal values
  - 10-15
    - Represent them with letters of the alphabet
    - 10 <-> A (or a)
    - 11 <-> B (or b)
    - 12 <-> C (or c)
    - 13 <-> D (or d)
    - 14 <-> E (or e)
    - 15 <-> F (or f)

# Number Systems

- Use hexadecimal (hex) as a shorthand for binary

  - Group sets of 4 binary bits and represent them with the hexadecimal equivalent
    - 1011 → B    0110 → 6    1110 → E
    - 10110110 → B6    01101110 → 6E
    - 1011011001101110 → B66E

  - Often it is easier if a space is inserted when writing these
    - 1011 0110 0110 1110 → B66E

  - When it is not obvious from the context you need to indicate the binary representation that the hex represents
    - Address = B66E → binary equivalent is unsigned binary → 46,702
    - Data value = B66E → binary equivalent is 2's complement → -18,834

# Number Systems

- Use hexadecimal (hex) as a shorthand for binary
  - Multiple ways to indicate a hex value
    - 12CDh — h at end
    - h12CD — h at beginning
    - $12CD — $ at beginning
    - 0x12CD — 0x at beginning
  - Different processors/people use different shorthand

# Number Systems

- Use hexadecimal (hex) as a shorthand for binary

| | Unsigned Binary | Signed Magnitude | 1's Complement | 2's Complement | BCD |
|---|---|---|---|---|---|
| 50 | 0011 0010 | 0011 0010 | 0011 0010 | 0011 0010 | 0101 0000 |
| | h32 | 32h | $32 | 0x32 | 32h |
| -50 | N/A | 1011 0010 | 1100 1101 | 1100 1110 | N/A |
| | | B2h | $CD | 0xCE | |

| h96<br>Unsigned Binary | 96h<br>Signed Magnitude | $96<br>1's Complement | 0x96<br>2's Complement | 96<br>BCD |
|---|---|---|---|---|
| 150 | -22 | -105 | -106 | 96 |

# Number Systems

- Scientific Number Representation

  - $1.60217657 \times 10^{-19}$ coulombs
  - $6.0221413 \times 10^{+23}$ units/mole

  - Normalized to have only 1 digit (non-zero) to the left of the decimal point
  - multiplied by a power of 10

  - 5692.3456 → $5.6923456 \times 10^{+3}$
  - .00023456 → $2.3456 \times 10^{-4}$

  - format is: mantissa × $10^{\text{exponent}}$

# Number Systems

- Binary Floating Point Number Representation

  - Normalized to have only 1 digit to the left of the decimal point
    - this must be a 1 since our choices are only 0 and 1 and we don't use 0
  - multiplied by a power of 2

  - 1011.1101 → $1.0111101 \times 2^{+3}$
  - .00011001 → $1.1001 \times 2^{-4}$

  - format is: mantissa $\times\ 2^{exponent}$

BUT

  - since the mantissa always starts with "1." we can use
    1.fraction $\times\ 2^{exponent}$

# Number Systems

- Binary Floating Point Number Representation

  - It is simpler to work with only positive exponents

  - Bias the exponent

    - With an 8 bit exponent the range is:

      +127 to -127 using signed magnitude notation

    - Add 127 to the desired exponent value (for use in the representation)

      actual range is still +127 to -127

      representation range is 254 to 0

    - called an exponent with +127 bias

  - format is now: value = 1.fraction × $2^{(\text{exponent} - 127)}$

    desired value representation

# Number Systems

- Binary Floating Point Number Representation

  - IEEE Standard
    - value = (-1 x sign) x 1.fraction × $2^{(exponent - 127)}$
    - 32 bit format

| Bit # | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| value | s | e | e | e | e | e | e | e | e | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f |

Sign — Exponent — Fraction

- Special cases
  - If E = 255, and F is non-zero, then the value is NaN (Not a Number)
  - If E = 255, F = 0 and S = 1, then the value is -infinity
  - If E = 255, F = 0, and S = 0, then the value is +infinity
  - If E = 0, and F = 0, then the value is 0

- Range
  - $1.11111111111111111111111_2 \times 2^{+127} = 3.4028 \times 10^{38}$
  - $1.00000000000000000000001_2 \times 2^{-127} = 1.1754 \times 10^{-38}$
  - 24 bit fractional precision ↔ 6 to 7 decimal digits

# Number Systems

- Example

use IEEE standard floating point to represent: 2,345,678.7109375

2,345,678 = 0010 0011 1100 1010 1100 1110 = 0x23CACE
0.7109375 = 0.10110110 = 0x0.B6

2,345,678.7109375 = 0010 0011 1100 1010 1100 1110 . 1011 0110
= 1. 0 0011 1100 1010 1100 1110 1011 0110 × $2^{21}$

fraction = 0001 1110 0101 0110  0111 0101 1011 0 will not fit in fraction part of the notation
exponent = 21 + 127 = 148 = 1001 0100
sign = 0

0   10010100    0001 1110 0101 0110 0111 010

# Number Systems

- Example

convert the IEEE floating point number
0 10010100 0001 1110 0101 0110 0111 010 to decimal

sign = 0
exponent = 1001 0100 = 148 → $2^{148-127} = 2^{21}$
fraction = 0001 1110 0101 0110 0111 010

+ 1.0001 1110 0101 0110 0111 010 × $2^{21}$
= 1 0001 1110 0101 0110 01110 . 10
= 2345678.5

error = (0.5 – 0.7109375)/2345678.5 = - $9x10^{-8}$
~7 decimal digits of precision