Introduction to Software Development – CS 6010
Lecture 11 – Characters, Binary and Hexadecimal Numbers

Master of Software Development (MSD) Program

Varun Shankar

Fall 2022

Miscellaneous

- Midterm Exam is on Friday
  - Short answer
  - Write short functions
  - Execute code by hand
- Check your grades in Canvas to make sure all of your grades have been recorded.

Lecture 11 – Characters, Binary and Hexadecimal Numbers

- Topics
  - Characters
  - Binary Numbers
  - Hex Numbers
  - Conversion between Number Formats
  - Code Review – Poker

Characters Recap

- Characters are stored in1 Byte (8 bits)
- The computer will "look them up" in the ASCII Table (When displaying)
- Addition works (sort of)
  - 'A' + 3 == 'D'
- Check to see if a capital letter…
  - if a character is <= 90 and >= 65 // Note: it is bad to use the ASCII value…
  - Remember, never write 90, or 65. Use 'Z' or 'A'
    - Or 'a' or 'z' or 'm' or '0' (the letter zero) or whatever letter you need.
    - char >= 'A' && char <= 'Z'
- How to convert the letter '3' to the number 3?
  - '3' – '0'
- Convert the letter 'C' to the number 12
  - 'C' – 'A' + 10

Base 10 Numbers (Decimal)

- When we write the number 365 – what does this really mean?
- Starting with the right most digit (the ones position), we multiply each digit by the *number base* raised to increasing powers, starting at power 0
- What is 10^0?   What is #^0? (Where # means any number)
  - 1
- 365 ==
  - 5 * 10^0 == 5 *   1 == 5
  - 6 * 10^1 == 6 * 10 == 60
  - 3 * 10^2 == 3 * 100 == 300
  - 300 + 60 + 5 == 365

$$Value = \sum_{i=0}^{\#digits-1} d_i * 10^i$$

i^{th} digit * (10 to the i^{th} power)

Range of Digits in a Numeric Base

- Base 10 (Decimal). What digits do we have?
  - 0 – 9
- Base 2 (Binary). What digits do we have?
  - 0 – 1
- Notice the largest digit is always one smaller than the base.
  - Decimal – 9, Binary – 1
- Base 16 (Hex). What digits do we have?
  - 0 – f where *a* is 10, *b* is 11, *c* is 12, *d* is 13, *e* is 14, and *f* is 15. (Largest *digit* is 15)
- Representation – What value does 10 have when converted to decimal?
  - 10_{10} == 10
  - 10_{2} == 2
  - 10_{16} == 16 (Why is this?)

Binary

- Base 2
  - So 1 is the largest digit
- Follows the same rule (starting at least significant digit (ones place)):
- 1010
  - 0 * 2^0 == 0 * 1 == 0 // Remember, starting from the right to the left
  - 1 * 2^1 == 1 * 2 == 2
  - 0 * 2^2 == 0 * 4 == 0
  - 1 * 2^3 == 1 * 8 == 8
- So 1010 is equal to 0 + 2 + 0 + 8 == 10 (ten) in decimal.
- What is the building block of computer memory?
  - The bit – a one or a zero… Hence the reason understanding binary is important.

Binary Joke

There are only 10 types of people in the world…

…those who understand binary, and those who do not.

Binary Numbers To Know

- 1 One
- 11 Three
- 111 Seven
- 1111 Fifteen
- What do these numbers have in common?
    - One less than the next power of two (2^n – 1).
- What is a byte?
    - 8 bits
- What size of number can we store in an 8 bit integer?
    - 2^8 == 256
    - 0 – 255; or, if we want signs…-128 to 127!

Hexadecimal

- Base 16. Usually prefixed by 0x to distinguish from decimal.
- With base 16, the largest number is 15… so again, remember we have:
- A == 10, B == 11, C == 12, D == 13, E == 14, F == 15
- So what is 0xA3F?
  - F * 16^0 == 15 * 1 == 15
  - 3 * 16^1 == 3 * 16 == 48
  - A * 16^2 == 10 * 256 == 2560
  - So 0xA3F == 15 + 48 + 2560 == 2623
- Why Hex?
  - *Easier* than binary to read. Usually have 16 (or 32) bits, which looks like
  - 0100111101101101 Binary
  - 4F6D Hex – Easier to visualize

Converting Between Bases

- Binary <-> Hex is easy
  - Each hex digit is 4 bits
  - Hex -> Binary
    - Replace each hex digit with 4 bit binary representation, e.g.:
    - 1 -> 0001, 2 -> 0010,   7 -> 0111
    - A -> 1010,   F -> 1111
    - 0xA7F
      - A == 1010
      - 7 == 0111
      - F == 1111
      - 0xA7F == 1010   0111 1111 or 1010 01111111

- Binary -> Hex
  - Group the binary into groups of 4, then convert to hex
  - 11110 grouped: 1 1 1 1 0
  - 1E

Converting Between Bases

- Hex or Binary to Decimal
  - Use the formula from before. Each digit times base to the appropriate power.
- Decimal to Binary
  - See next slide
- Decimal to Hex
  - Usually easiest to convert to Binary, and then convert to Hex if necessary.

Decimal To Binary (i)

- Let's turn 92 into binary…
- What does the *one's digit* in a binary number tell us about the binary number?
  - Binary   Decimal   Here are some examples… See a pattern?
  - 10   2
  - 11   3
  - 01   1
  - 100   4
  - 101   5
  - The *one's digit* in a binary number tells us whether the number is odd or even.
- So what is the one's digit value (in binary) for the number 92?
  - 0

Decimal To Binary (ii)

- Let's turn 92 into binary…
- How do we do this?
- Let's do it the long way, then do the shortcut.
- $(92)_{10} = x_n\ 2^n + x_{n-1}\ 2^{n-1} + \ldots + x_1\ 2^1 + x_0$.
  - But x_0 is just the least significant digit, 1 if odd, 0 if even. Here 0.
- $(92)_{10} = x_n\ 2^n + x_{n-1}\ 2^{n-1} + \ldots + x_1\ 2^1 + 0$.
- $(92)_{10} = 2(x_n\ 2^{n-1} + x_{n-1}\ 2^{n-2} + \ldots + x_1) + 0 = 2*46 + 0$.
- $(46)_{10} = x_n\ 2^{n-1} + x_{n-1}\ 2^{n-2} + \ldots + x_1$
  - But this number is even, so x_1 is 0!
- $(46)_{10} = x_n\ 2^{n-1} + x_{n-1}\ 2^{n-2} + \ldots + x_2 2 + 0$
- $(46)_{10} = 2(x_n\ 2^{n-2} + x_{n-1}\ 2^{n-3} + \ldots + x_2) + 0 = 2*23 + 0$.

Decimal To Binary (iii)

- $(46)_{10} = 2(x_n\, 2^{n-2} + x_{n-1}\, 2^{n-3} + \ldots + x_2) + 0 = 2*23 + 0.$
- $(23)_{10} = x_n\, 2^{n-2} + x_{n-1}\, 2^{n-3} + \ldots + x_2$
  - But $x_2 = 1$ since 23 is odd.
- $(23)_{10} = 2(x_n\, 2^{n-3} + x_{n-1}\, 2^{n-4} + \ldots + x_3) + 1 = 2*11 + 1.$
- Keep going like this:
- $11 = 2*5 + 1$, i.e., $(x_3 = 1)$
- $5 = 2*2 + 1$, i.e., $(x_4 = 1)$
- $2 = 2*1 + 0$, i.e., $(x_5 = 0)$
- $1 = 0 + 1$, i.e., $(x_6 = 1)$
- The binary number is then 1011100.

Decimal To Binary (iv)

- Do you see the shortcut?
- Just repeatedly divide by 2 and keep track of remainders!
- Remainders in **reverse order** (bottom to top) form the number from left to right.
- The process of dividing by two is called right shifting.
  - For example, in binary, 110 shifted to the right (divided by 2) is just 11.
- Most languages have left and right shift operators. In C++, it's << for left shift and >> right shift.

Binary to Decimal?

- Here's an easier way than fully expanding into powers of 2.
- Given a binary number:
  - 1 0 1 1 0 0 0 1 0

256	128	64	32	16	8	4	2	1
256		64	32				2	

- Write the value of each position:
  - Start with the least significant digit and put a 1 under it, then a 2 under the next digit, followed by a 4, then an 8, etc…
- Then add them up (the numbers with a 1 above them):
  - 256 + 64 + 32 + 2 => 354

Today’s Assignment(s)

- Code Review – Poker
- Lab – Binary and Hex
- Group Homework – Number Converter