# CS 240 Programming in C

Bases, Bits, Number Systems and Two's Complement

September 23, 2019

# Schedule

- The scp cheat sheet on class page does not work. The "slash" problem.
- Go through homework 2.
- New business.

# Number System

- As of this point, we have briefly covered what might be called the conventional core of C.
- And we are already able to write useful programs of considerable size.
- Students can explore more practices of greater complexity in K&R chapter 1 practices.
- Later on we will continue get into C thoroughly and deeply.
- Today's topic is about number system and two's compliment for negative numbers in C.

# Number System

- For the knowledge of number system we focused here is about the conversions of them.
- We will not study it in the way of a math class.
- However it is so important that it will be in our first test. That's why we will cover it today in class.
- I will announce the first exam date ahead of 1 week before it takes place.

# Notations for Different Bases

| | math notation | C code |
|---|---|---|
| Decimal | $109_{10}$ | `109` |
| Binary | $01101101_2$ | |
| Octal | $155_8$ | `0155` |
| Hexadecimal | $6D_{16}$ | `0x6D` |

- $10_b$ is the number of digits in the base-$b$ system
- $10_b$ is equal to $b$
- $10_2$ is 2
- $10_8$ is 8
- $10_{10}$ is 10
- $10_{16}$ is 16

# Number Systems

- We are accustomed to the base-10 number system, decimal numbers
- Computers use the base-2 number system, binary numbers
- Binary numbers are not easy for people to read
- Conversion between base-2 and base-10 is not easy
- Solution: convert binary numbers to: octal (base-8) or hexadecimal (base-16)
- These conversions are easier because 8 is $2^3$ and 16 is $2^4$
- Online conversion for testing. `https://codebeautify.org/all-number-converter`

# Convert Decimal to Binary

- Divide the decimal number by 2 and write down the remainder
- Repeat
- The first remainder is the least significant bit
- Example: convert $109_{10}$ to an 8-bit binary

| steps | decimal | quotient | remainder |
|---|---|---|---|
| 0 | 109 | 54 | 1 |
| 1 | 54 | 27 | 0 |
| 2 | 27 | 13 | 1 |
| 3 | 13 | 6 | 1 |
| 4 | 6 | 3 | 0 |
| 5 | 3 | 1 | 1 |
| 6 | 1 | 0 | 1 |
| 7 | 0 | 0 | 0 |

# Convert Binary to Decimal

- Each bit position $i$ corresponds to $2^i$
- Multiply the bit at position $i$ with $2^i$
- Add up all products
- Example: convert $01101101_2$ to decimal

| $i$ | $2^i$ | bit | product |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 2 | 0 | |
| 2 | 4 | 1 | 4 |
| 3 | 8 | 1 | 8 |
| 4 | 16 | 0 | |
| 5 | 32 | 1 | 32 |
| 6 | 64 | 1 | 64 |
| 7 | 128 | 0 | |

# Convert Decimal to Hexadecimal, Method 1

- Divide the decimal number by 16 and write down the remainder in hex Repeat
- The first remainder is the least significant
- Example: convert $109_{10}$ to hex

| steps | decimal | quotient | remainder | hex |
|---|---|---|---|---|
| 0 | 109 | 6 | 13 | D |
| 1 | 6 | 0 | 6 | 6 |

- $6D_{16}$

# Convert Decimal to Hexadecimal, Method 2

- First convert the decimal number to binary
  $109_{10}$ to $01101101_2$
- Then group the binary bits into groups of 4
  0110 1101
- Finally convert 4-bit binary numbers to hex digits
  $6D_{16}$ or `0x6D`

# Convert Hexadecimal to Decimal, Method 1

- Each digit position $i$ corresponds to $16^i$
- Multiply $16^i$ to the hex digit at position $i$
- Add up all products
- Example: convert $100B35_{16}$ to decimal

| $i$ | $16^i$ | digit | product |
|---|---|---|---|
| 0 | 1 | 5 | 5 |
| 1 | 16 | 3 | 48 |
| 2 | 256 | 11 | 2816 |
| 3 | 4096 | 0 | |
| 4 | 65536 | 0 | |
| 5 | 1048576 | 1 | 1048576 |

# Convert Hexadecimal to Decimal, Method 2

- Initialize sum to zero

1. Multiply sum by 16
2. Add the next most significant hex digit to sum

- Repeat these two steps till finishing the least significant digit
- Example: convert $100B35_{16}$ to decimal

| sum | digit |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 16 | 0 |
| 256 | 11 ($B_{16}$) |
| 4107 | 3 |
| 65715 | 5 |
| 1051445 | |

# Representing Negative Integers

- 1's complement
- 2's complement

| binary | unsigned | 1's | 2's |
|---|---|---|---|
| 0000 | 0 | 0 | 0 |
| 0001 | 1 | 1 | 1 |
| 0010 | 2 | 2 | 2 |
| 0011 | 3 | 3 | 3 |
| 0100 | 4 | 4 | 4 |
| 0101 | 5 | 5 | 5 |
| 0110 | 6 | 6 | 6 |
| 0111 | 7 | 7 | 7 |

| binary | unsigned | 1's | 2's |
|---|---|---|---|
| 1000 | 8 | -7 | -8 |
| 1001 | 9 | -6 | -7 |
| 1010 | 10 | -5 | -6 |
| 1011 | 11 | -4 | -5 |
| 1100 | 12 | -3 | -4 |
| 1101 | 13 | -2 | -3 |
| 1110 | 14 | -1 | -2 |
| 1111 | 15 | -0 | -1 |

# Two's Complement

- There are *n* bits
- Zero: *n* 0's
- Positive integers
  - 1 is 0001, 2 is 0010, ..., 7 is 0111
- Negative integers
  - Flip the roles of 0 and 1
  - Start with 1111, which is -1
  - Next 1110, which is -2
  - Last 1000, which is -8
- The highest bit is the sign bit
- See what happens when you add 5 with -5

# Two’s Complement for Negative Numbers in C

All the negative numbers in C are stored in the form of two’s complement such that a subtraction is done as addition.

```
/*                  What is this printing out ?
* Author : Haoyu Wang
* Description:
*       Two’s Complement
*/
#include <stdio.h>
int main(int argc, char *argv[])
{
     int i = -1;
     printf("%x\n", i); /*  */
     printf("%x\n", -i);
    return 0;
}
```

# How to Read Binary Files

- Use the utility `hexdump` to examine a binary file
- Do a hexdump of your hello world executable
- ELF: executable and linkable format
- Use the command `locate` to locate a file
- Use the utility `readelf` to see what is in an ELF file
- Use `readelf` to read your hello world executable by readelf -h [name]