Lab 1: Binary

By: CS 382 CAs

Binary Recap

Structure of a Binary Number

In our “normal” decimal system:

Consider the number 65,420

Place	10,000	1,000	100	10	1
Digit	6	5	4	2	0

In binary:

Consider the number

11010100b = 420d

Place	256	128	64	32	16	8	4	2	1
Digit	1	1	0	1	0	0	1	0	0

Bit Shifting and Bitwise Operations in C

Binary Numbers in C:

- Generally, ints are 4 bytes (32 bits)
- C works with your integer as if it is already this binary number.
    - Ex. `int x = 53;` the value of x is: 00000000 00000000 00000000 00110101 (tho without this spacing)

Bit Shifting:

Left Shift	5 << 1	0101	1010
Right Shift	5 >> 1	0101	0010

Bitwise AND (&) 1011 & 1000 = 1000

Bitwise OR (|) 1011 | 1000 = 1011

Data Types

Basic Primitive Data Types:

type	void	char	int	float	double
bytes	-	1	4	4	8

Data Type Qualifiers:

- unsigned/signed – char,int
- short – int
- long – int,double

Arrays:

```
int array[10]; //creates an array of 10 ints
```

Basic I/O

```
printf("Hello, %d!\n", 382);  → Hello, 382!
```

What is a string?

- char* : null terminated char pointer

#include <stdio.h>

- putchar(char)   : write raw character
- puts(string)   : write raw string
- printf(format, args…) : write formatted string

man <function/command> in general (in browser or command line) for documentation for C/Linux things

Format Specifiers:

specifier	%d, %u	%f, %lf	%c	%s	%p
meaning	(un)signed integers	float, double	character	string	pointer

Consider this:

When we use a number that is 2^n, that binary number will always be 1 followed by n number of 0 digits

- 2^0 is 1
- 2^1 is 10 (1 << 1)
- 2^2 is 100 (1 << 2)
- 2^3 is 1000 (1 << 3)

… and so on.

- So what happens when we & or | a number by a power of 2?

- What happens when we shift a number?

VM Setup Check-In: Are you set up?

- At this point, you should have an Ubuntu VM set up on your machine

- If not, please consult the “Ubuntu VM Setup” for your OS
  - On MacOS, **multipass** is used
  - On Windows, **VMWare VirtualBox** is used

- It’s INCREDIBLY important your VM is working now, since you’ll need it the rest of the semester
  - And definitely in your next class, CS 392!

Lab 1 Assignment

Peek at the Starter Code

```c
#include <stdio.h>
#include <stdlib.h>

void display(int8_t bit) {
    putchar(bit + 48);
}

void display_32(int32_t num) {
    /* Your code here */
}

int main(int argc, char const *argv[]) {
    display_32(382);
    return 0;
}
```

- #include …
  - stdio.h - Input/Output
  - stdlib.h - General purpose functions
- display() - why add 48?
  - What is putchar doing?
  - ASCII Code
- display_32()
  - YOUR JOB FOR THIS ASSIGNMENT
- main()
  - Add as many test cases as you want
  - Just keep the return 0 at the very end!

How to Compile & Run your lab1.c

In your terminal:

```
gcc <some_file>.c -o <name_of_output>
```

- Enter “ls”, see you have an executable file called <name_of_output>
- If you don't specify output, your output file will be named a.out

Provided you don’t have errors, run it with:

```
./<name_of_output>
```

- We could be using makefiles like in 385, but it is not necessary for this assignment.

Hints, Restrictions, Requirements

- No multiplication, division, or modulus allowed!
  - Addition and subtraction are fine
- Permitted binary operations
  - Left Shifting (<<) and Right Shifting (>>)
  - Bitwise AND (&), bitwise OR (|) (you probably don’t need XOR)
- How can we use bitwise operations to “extract” individual bits?

MUST-DOs

- Make sure your code compiles!!!
- Include your name and the honor pledge (typed correctly, please)
- Display all 32 bits