Week 2 - Wednesday

CS222

Last time

- What did we talk about last time?
- Makefiles
- Binary
- C literals

Questions?

Project 1

Quotes

Unity can only be manifested by the Binary. Unity itself and the idea of Unity are already two.

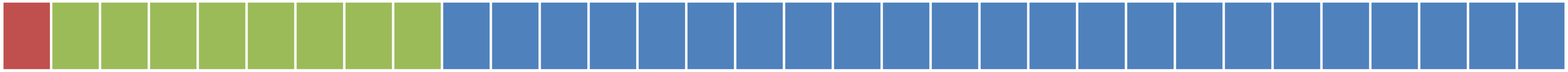
Buddha

Two's complement practice

- Convert the 8-bit two's complement binary representation **`10111010`** to the equivalent decimal integer
- Convert the decimal integer -117 to the equivalent 8-bit two's complement binary representation

Floating point representation

- Okay, how do we represent floating point numbers?
- A completely different system!
  - IEEE-754 standard
  - One bit is the sign bit
  - Then some bits are for the exponent (8 bits for float, 11 bits for double)
  - Then some bits are for the mantissa (23 bits for float, 52 bits for double)



More complexity

- They want floating point values to be unique
- So, the mantissa leaves off the first 1
- To allow for positive and negative exponents, you subtract 127 (for **`float`**, or 1023 for **`double`**) from the written exponent
- The final number is:
  - $(-1)^{\textit{sign bit}} \times 2^{(\textit{exponent} - 127)} \times 1.\textit{\textbf{mantissa}}$

Except even that isn't enough!

- How would you represent zero?
  - If all the bits are zero, the number is 0.0
- There are other special cases
  - If every bit of the exponent is set (but all of the mantissa is zeroes), the value is positive or negative infinity
  - If every bit of the exponent is set (and some of the mantissa bits are set), the value is positive or negative NaN (not a number)

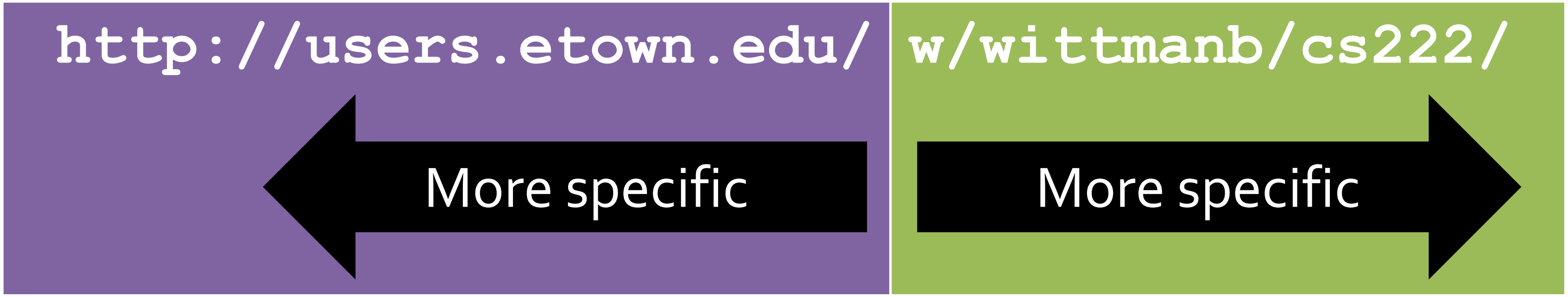
Number	Representation
`0.0`	`0x00000000`
`1.0`	`0x3F800000`
`0.5`	`0x3F000000`
`3.0`	`0x40400000`
+Infinity	`0x7F800000`
-Infinity	`0xFF800000`
+NaN	`0x7FC00000` and others

One little endian

- For both integers and floating-point values, the **most significant bit** determines the sign
  - But is that bit on the rightmost side or the leftmost side?
  - What does left or right even mean inside a computer?
- The property is the **endianness** of a computer
- Some computers store the most significant bit first in the representation of a number
  - These are called **big-endian** machines
- Others store the least significant bit first
  - These are called **little-endian** machines

Why does it matter?

- Usually, it doesn't!
- It's all internally consistent
  - C uses the appropriate endianness of the machine
- With pointers, you can look at each byte inside of an **`int`** (or other type) in order
  - When doing that, endianness affects the byte ordering
- The term is also applied to things outside of memory addresses
- Mixed-endian is rare for memory, but possible in other cases:



`http://users.etown.edu/` **`w/wittmanb/cs222/`**

More specific

More specific

Math library

Function	Result	Function	Result
`cos(double theta)`	Cosine of `theta`	`exp(double x)`	e^x
`sin(double theta)`	Sine of `theta`	`log(double x)`	Natural logarithm of `x`
`tan(double theta)`	Tangent of `theta`	`log10(double x)`	Common logarithm of `x`
`acos(double x)`	Arc cosine of `x`	`pow(double base, double exponent)`	Raise `base` to power `exponent`
`asin(double x)`	Arc sine of `x`	`sqrt(double x)`	Square root of `x`
`atan(double x)`	Arc tangent of `x`	`ceil(double x)`	Round up value of x
`atan2(double y, double x)`	Arc tangent of `y/x`	`floor(double x)`	Round down value of `x`
`fabs(double x)`	Absolute value of `x`	`fmod(double value, double divisor)`	Remainder of dividing `value` by `divisor`

Math library in action

- You must **`#include <math.h>`** to use math functions

```
#include <math.h>
#include <stdio.h>

int main()
{
	double a = 3.0;
	double b = 4.0;
	double c = sqrt(a*a + b*c);
	printf("Hypotenuse: %f\n", c);
	return 0;
}
```

It doesn't work!

- Just using **`#include`** gives the headers for math functions, not the actual code
- You must link the math library with flag **`–lm`**

```
> gcc hypotenuse.c -o hypotenuse -lm
```

- Now, how are you supposed to know that?

```
> man 3 sqrt
```

My main man

- Man (manual) pages give you more information about commands and functions, in 8 areas:
  1. General commands
  2. System calls
  3. Library functions (C library, especially)
  4. Special files and devices
  5. File formats
  6. Miscellaneous stuff
  7. System administration
- Try by typing **`man topic`** for something you're interested in
- If it lists topics in different sections, specify the section

```
> man 3 sqrt
```

- For more information:

```
> man man
```

Example

- You are sitting at the origin
- There's a hyperspace ghost demon at location (***x***,***y***)
- Write a program to determine the angle to fire your C-controlled proton accelerator in order to remove the deadly menace

Quiz

Upcoming

Next time…

- Single character input
- Lab 2

Reminders

- Read LPI chapter 11