CS 240
Computer
Systems

Week 2: Computer Architecture

CS 240, Spring 2021 - Week 2
Wade Fagen-Ulmschneider

Programming in C

One example of a plain-text file in a C source code file. Today, you'll see your very first Machine Problem in CS 240!

- You already know how to program in C++! 🎉
- Programming in C is a simplification of the C++ programming.

Program Starting Point:

Printing Using **`printf()`**:

- Use of printf requires an include: **`#include <cstdlib.h>`**
- printf is designed to have a variable number of arguments:
  - First argument:
  - Additional arguments:

```
#include <stdlib.h>

int main() {
  int i = 42;
  char *s = "Hello, world!”;
  float f = 3.14;

  printf("%d %s %f\n”, i, s, );
  printf("%d\n”, s[0]);
  printf("%f\n”, s[0]);
  printf("%d\n”, s);
  return 0;
}
```

Pointers:

Heap Memory Allocation:

```
int main() {
  char *s = malloc(10);
  int *num = malloc( sizeof(int) );

  printf("%p %p\n”, s, num);
  return 0;
}
```

Strings -- **`#include <string.h>`**

Four Key Functions:

- **`strcmp(char *s1, char *s2)`** -- Compares two strings
- **`strcat(char *dest, char *src)`** -- Concatenate two strings
- **`strcpy(char *dest, char *src)`** -- Copies a string
- **`strlen(char *s)`** -- Returns the length of the string

Bit Manipulation: Binary Addition

For the past three lectures, we've focused on the first foundation: data. Today, we are going to begin the transition away from data and into our CPU. Just like decimal addition, but with only **`0`**s and **`1`**s:

```
  0b 010011              0b 0011
+ 0b 001001            + 0b 0111
```

Negative Numbers:

Two's Complement
The Two's Complement is a way to represent signed (ex: positive vs. negative) numbers!

For simplicity, let's imagine running on an ***8-bit machine****:*

`-17` =

`-4` =

`-1` =

`42 - 18` =

`18 - 42` =

Towards Multiplication
With Two's Complement, we can add and subtract numbers! What about more complex operations?

`10 x 2 =`

`10 x 4 =`

`10 x 9 =`

Bit Shift Operations:

1. [Left Shift]:

2. [Right Shift]:

Logic Gates and Truth Tables

Let's simplify the bit manipulation all the way to single bits. We can begin to define the building blocks of the CPU by basic instructions with input bits and output bits.

- By convention, you will see that the input bits are labeled **`A`** and **`B`** by default.

Logic Gate #1:

Logic Gate #2:

Logic Gate #3:

Logic Gate Challenge: ***`A XOR B`***

Truth Table: Binary Addition

In the last lecture, we explored "Binary Addition". Let's see if we can begin to design the circuit to complete the binary addition!

Truth Table:

A	B	A + B	SUM	CARRY

Circuit Diagram for a "Half Adder":

Truth Table:

A	B	CARRY$_{in}$		SUM	CARRY$_{out}$

Circuit Diagram for a “Full Adder”:

Complete Circuit for a “Ripple Carry Adder”:

Disadvantages:

Analysis and Comparisons of Operations: