Today

- Announcements
- Buflab and Stack Review
- Structs
- Optimization Basics
- Memory Hierarchy and Caching
Announcements

• Datalab can be picked up at the ECE Course Hub, which is on the D level of Hammerschlag.
• Buflab due this Thursday, 2/25
• Exam 1 in class next Tuesday, 3/02
Buflab
Consider when this function from Buflab is about to call Gets().

```c
int getbuf()
{
    char buf[32];
    Gets(buf);
    return 1;
}
```
What happens if we overflow buf?

```c
int getbuf()
{
    char buf[32];
    Gets(buf);
    return 1;
}
```
x86 Stack Frame

- What if we set the return address to &buf?
- What if we don’t know what &buf is?

```c
int getbuf()
{
    char buf[32];
    Gets(buf);
    return 1;
}
```
Questions?
Structs
Structs

- Allow you to declare a contiguous block of memory which can include various data types.
- Types are subject to alignment rules (Why?).

```c
struct node {
    char c;
    int x;
}
```
Old Exam Question

- Show the memory layout of the following struct on a 64-bit (x86_64) machine.
- Reorder the fields to have a more optimal packing.

```c
struct foo {
    char a[9];
    short b[3];
    float c;
    char d;
    int e;
    char *f;
    short g;
}
```
Old Exam Question

• Show the memory layout of the following struct on a 64-bit (x86_64) machine.
• Reorder the fields to have a more optimal packing.

```c
struct foo
{
    char a[9];         // AAAAAAAXB1B2B3
    short b[3];        // CCCCCDxxxxEEEExxxx
    float c;           // FFFFFFFFGGxxxxxx
    char d;
    int e;             // FFFFFFFFFCCCCCEE
    char *f;           // B1B2B3GGAAAAA
    short g;           // ADxxxxxx
}
```

Answers:
- `float c`: FFFFFFFFGGxxxxxx
- `char d`: 
- `int e`: FFFFFFFFFCCCCCEE
- `char *f`: B1B2B3GGAAAAA
- `short g`: ADxxxxxx
Questions?
Optimization
void func(int a, int b, char data[]) 
{
    for(int i=0; i<10; i++)
        if(data[i] < 'z' && data[i] != 'n')
            data[i]++;
}

• You could declare char c = data[i] before the if statement to avoid recalculating and re-accessing data[i].
Code Hoisting

```c
void func(int a, int b, char data[]) {
    for(int i=0; i<10; i++)
        data[a*b+i] = 'A';
}
```

- You can calculate \(a \times b\) outside the loop instead of every iteration.
- This also applies to loop limits, i.e. if we had \(i < (a+b)\).
Loop Unrolling

- A technique to reduce loop overhead.
- When accessing array elements, why not go two or more at a time?
- This results in fewer iterations, which means fewer jumps and condition checking.
- However, it adds code bloat.
- All that extra code may not fit in the instruction cache.
Optimization Blockers

• As good as compilers can be at optimizing, sometimes we can help it do better.

• Function calls can add a lot of overhead.
  • You move function code into the main procedure at the cost of lower modularity and added code bloat.
  • The `inline` keyword or preprocessor macros can have the compiler do this for you.

• Memory aliasing
  • The compiler doesn’t know if more than one pointer is accessing the same memory location.
  • Use a temporary variable to do a calculation and store the result in memory when you’re done.
Questions?
A Gentle Introduction to Caching
Memory Hierarchy

- Registers
- L1 Cache (SRAM)
- L2 Cache (SRAM)
- Main Memory (DRAM)
- Local Storage (Disk)
- Remote Storage (Tape, Web)

Small, Fast, Expensive

Large, Slow, Cheap
Caching Introduction

• When memory is accessed, it tends to be accessed again within a short amount of time.

• Instead of accessing slow memory twice, stash a copy in a faster memory.

• Cache “Hit” when memory being accessed is cached, Cache “Miss” otherwise.

• Hit/Miss rate is the ratio of cache hits/misses to total memory accesses, respectively.
Cache Types

• Direct Mapped
  • Data at each memory address is loaded into a specific cache block.
  • Hardware is simple, but you can end up lots of collisions if multiple variables vie for the same block.

• \( n \)-Way Associative
  • Data at each memory address can be loaded into one of \( n \) cache blocks.
  • Fewer collisions, but how do you figure out which cache block to fill, or which block has your data?
Review

- Buflab Thursday, Exam 1 next week.
- Stack Review
- Structs
- Optimization Basics
- Memory Hierarchy and Caching
  - Plenty more on these in lecture this week.
- Questions?