CSE 351: The Hardware/Software Interface

Section 6
Midterm review
Non-inclusive topic list

- Addressing data in memory
  - Pointers, byte ordering
- Bit-level operators
  - &, |, ^, ~, +, !, <<, >>
- Integer representations
  - Two's complement
- Floating point numbers
  - Representation, conversion
Non-inclusive topic list

- Program state representation
  - How registers, stack, heap, and text segment are used
- Assembly instructions
  - mov, lea, add, and so forth. Moving data between registers and memory
- Control flow
  - cmp, test, conditional jumps, and how they are used to represent if/then, for, and do-while
- Calling conventions
  - Passing arguments in x86 versus x86-64, recursive function calls
- Arrays
  - Representation in memory, accesses using assembly instructions
- Buffer overflows
  - What they are, how they can be used maliciously, how to prevent against them
The x86 assembly instructions can be broken down into several basic categories:

- Data movement instructions
- Arithmetic instructions
- Control flow instructions
Data Movement Instructions

- MOV
  - Moves data between registers and memory

- PUSH
  - Decrements stack pointer
  - Places value on top of stack

- POP
  - Increases stack pointer
  - Removes value from top of stack

- LEA
  - Loads address into register
  - Useful for pointer operations

2/12/2014
Arithmetic Instructions

* Most are pretty self-explanatory
  * ADD, SUB, IMUL, IDIV, INC, DEC
* These operations can set flags:
  * CF: carry flag
  * ZF: zero flag
  * SF: sign flag
  * OF: overflow flag
Control Flow Instructions

- **CMP**: compare two operands
  - It is equivalent to a SUB command, except the result is not stored, only the flags are set

- **CALL**: call a subroutine
  - Pushes the next instruction onto the stack
  - Jumps to the code location specified by the operand

- **RET**: return from subroutine
  - Pops an instruction address off the stack
  - Jumps to that instruction

- **LEAVE**: eliminates the current stack frame
  - Moves %esp to %ebp
  - Pops old %ebp off stack into %ebp
Control Flow Instructions

- JMP: jump to a particular label
- Can create conditional jumps using CMP
  - JNE: jump if not equal
  - JE: jump if equal
  - JZ: jump if zero
  - JG: jump if greater than
  - JGE: jump if greater than or equal to
  - JL: jump if less than
  - JLE: jump if less than or equal to
**Calling Conventions**

**Things to remember:**

- Arguments passed in registers for x64
  - %rdi, %rsi, %rdx, %rcx, etc...
- Caller-save vs. Callee-save
- Stack frame structure
  - Subtract from %rsp to create space for locals
  - Return address, old %rbp pushed onto stack
  - (%rbp) is highest address
  - (%rsp) is lowest address
C Unions

* Allows you to store data types in the same memory location

* Example:

```c
union Data {
    int i;
    float f;
    char str[20];
} data;
```

* A variable of type Data will occupy 20 bytes
  * Always occupies the size of the largest member
C Unions

- Members of a union are accessed using the same "." operator used for structs
  - If we declare a variable of type Data named `data_union`:
    - `data_union.i`
    - `data_union.f`
    - `data_union.str`

- Only one of the members is valid at one time
  - Before using a member, your code must ensure that it is the "active" member
Questions

- Question time!

- If you don’t have any questions, we can look at implementing `strlen()` in x64 assembly