CS 442/542

Course overview and introduction to the compiling process
Compiling Process

• Scanner
  – Input: Stream of characters
  – Output: Stream of tokens

• Parser
  – Input: Stream of tokens
  – Output: Syntax tree or parse tree

• Semantic Analysis
  – Input: Syntax or parse tree
  – Output: machine independent intermediate
Compiling Process

- **Machine independent Optimizations**
  - Input: intermediate code
  - Output: optimized intermediate code

- **Machine code generation**
  - Input: intermediate code
  - Output: Machine code or assembly code for a particular architecture
Compiling Process

• Machine dependent optimizations
  – Input: machine or assemble code
  – Output: optimized machine or assembly code
Simplified Compiling Process for CS 442/542 Final Project

• Scanner built with lex and C
• Parser built with yacc and C
  – The parser will get tokens from the scanner
  – Syntax checking, typing checking and MIPS code generation will occur as the source program is being parsed
Course Overview

• Scanning/Lexical Analysis
  – Finite State Automata
    • Deterministic FA
    • Non-deterministic FA
  – Regular Expressions
  – Scanner generator
    • lex (flex)
Course Overview

• Parsing/Syntax Analysis
  – Pushdown automata
  – Context free grammars
    • LL
    • LR
  – Parser Generator
    • yacc (bison)
Course Overview

• Semantic Analysis
  – Static Semantics
    • Type checking
  – Run-time semantics
    • Intermediate code generation

• Optimization
  – Machine independent
  – Machine dependent
Course Overview

• Code Generation
  – MIPS
Course Overview/Project

- Symbol table
- IO Manager
- lex/yacc practice
- Final project
- All projects will be done in C
- The final project will generate MIPS code that will be executed on a MIPS (SPIM or MARS) interpreter
- The final project will be demonstrated to me during the last week of class
- Each student will do her/his own project
Example Source Code

```plaintext
int x;
int y;
int z;
read x;
read y;
z = x+y;
print z;
```
Example MIPS Code

.text
.globl main
main:
    li   $v0, 5
    syscall
    sw   $v0, x
    li   $v0, 5
    syscall
    sw   $v0, y
    lw   $t0, x
    lw   $t1, y
    add  $t2, $t0, $t1
    sw   $t2, z
    lw   $t0, z
    li   $v0, 1
    move  $a0, $t0
    syscall
    li   $v0, 4
    la    $a0, _nl
    syscall
    li   $v0, 10
.data
_nl:   .asciiz  "\n"
x:    .word    0
y:    .word    0
z:    .word    0
Homework 0 (nothing to turn in)

• Read chapter 1
• Begin reading chapter 2
• Mac users have all the tools (c compiler, flex (lex) and bison (yacc) assuming you have installed the developer tools
• The tools are available on the machines in the lab.
Homework 0 (nothing to turn in)

• Linux

• Compute

• On a Windows machine you will need to install gcc, flex and bison. There are not (as far as I know) versions that run on Windows so you will need to use the Linux sub-system that comes with Windows 10