# Lab 10

COEN 175 Compilers

# Overview For Lab 10

**Goal**

- Finish the compiler (generate code for expressions and statements)

**Submission**

- Submit a tarball of your cpps and make files in folder called phase6
- Due Date: **Friday,** March 11

# Phase 6 Outline

1. Structures
2. Account for structs in Assignment
3. Remaining Expressions
4. Remaining Statements

# 1. Structures

- Implementation covered in class
- findBaseAndOffset()
  - Base is from the expression
  - Offset should iterate through field offsets
- Field::generate()

# 1. Structures - Field::generate()

- Call findBaseAndOffset()
- If base is a dereference
  - Generate and load pointer
  - Assign resulting expression to a register
  - move (byte or long) value at dereferenced pointer location to result register (adjusted by offset)
- If not a dereference
  - Assign resulting expression to a register
  - move (byte or long) base to result register (adjusted by offset)

# 1. Structures - findBaseAndOffset()

- Used for
  - Generating fields
  - Assignment
  - Address
- Call findBaseAndOffset() at the beginning
- Replace all _left/_expr with base
- Add offset during final operation Ex:

```
cout << "movl" << _right << ", " << _left << endl;
```

```
cout << "movl" << _right << ", " << offset << "+" << base << endl;
```

# 2. Remaining Expressions

- LogicalOr & LogicalAnd
  - Create labels for skip and short-circuit
  - Test left and right expressions
    - Think about short circuit evaluation for ifTrue value
  - Assign and move 0/1 to current
  - Jump to skip
  - Write short-circuit label
  - Move 1/0 to current
  - Write skip label

# 2. Remaining Expressions - Test

- Need for general expression - Given in class
  - Pass whether you are jumping on true/false and the label to jump to
  - Generate
  - If current is a number, load it
  - Compare to 0
  - Jump accordingly with jne/je
  - Unassign itself
- Explanation: Will jump to given label if expression value matches the passed ifTrue

# 3. Remaining Statements

- Return::generate()
  - Generate and load expression into eax
  - Jump to [function name].exit
    - Use global funcname variable
  - Unassign expression
- If::generate()
  - Test the expression
  - Generate the statement for then
  - Check if there is an else
    - Jump to end label
    - Write else label
    - Generate else statement
  - Write end label

# 3. Remaining Statements

- While::generate() - Given in class
  - Similar to if, but use loop and exit labels
- For::generate()
  - Generate init
  - Enter loop
  - Test expression
  - Generate statement
  - Generate increment
  - Jump to loop label
  - Write exit label

# Tips

- Recompile your code frequently to make sure it still works
- Since you are overriding functions, you don't need to have the others completed to *compile*
- _type will give the resulting type of the expression
- Don't forget to assign resulting expression "this" at end
- Type checks should be with _left, not base
- NO changes in parser
- Check the lectures!
- Check your output with the gcc using the -S flag
    - This will generate more optimal code than yours most likely, worry about correctness

# Checking your code

- $ scc < file.c > file.s 2> /dev/null
- $ gcc -m32 file.s [additional-source-files]
- $ ./a.out

- You don’t need to change any report() since those go to stderr
- Make sure you are sending your generated code to stdout (>>)
- Run with CHECKSUB before submission