COEN 175

Phase 6 - Week 1

TAs

- Chris Desiniotis: cdesiniotis@scu.edu
  - Office Hours: Friday 12 - 2 pm
- Antonio Gigliotti: agigliotti@scu.edu
  - Office Hours: Thursday 11 - 1 pm

Extra Help/Tutoring

- Tau Beta Pi Tutoring
- Link to Tutoring schedule
  - https://sites.google.com/scu.edu/scutaubetapi/tutoring?authuser=1&pli=1

Phase 6 - Code Generation

1. Download new files for phase 6
2. Implement register functions
3. Modify Assignment::generate() to use registers
4. Write generate for add, subtract, multiply, cast
5. More code generation for operators

- **Due Friday March 12, 11:59PM**

1. Download new files for phase 6

- Phase 5 solution will be posted after the due date of Wednesday at 11:59PM
- Until then, use provided files on Camino to work on your own phase 6
    - Create new phase 6 directory
    - Use your current phase 5 solution along with provided files as a starting point
    - Provided files can be found on Camino: Files > labs > 9
    - Run make clean all
    - Work on phase 6
        - Most of your new code will go in generator.cpp. This will make the transition easier when phase 5 solutions are posted
- Once the phase 5 solutions are posted, download and use those like normal
    - Copy the code you wrote in generator.cpp to generator.cpp provided in the solutions
- Note: May have compile time error
    - Make sure you have an empty definition for Expression::operand() in generator.cpp

2. Implement register functions

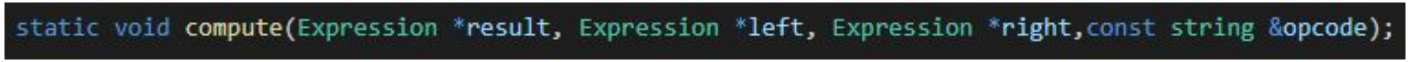
- Check class notes for the code:
  - load()
  - assign()
  - getreg()
- These functions belong in generator.cpp
- Don’t forget to create registers (edx, ecx, eax) at top of generator.cpp
- Free register in Simple::generate()
  - `assign(_expr, nullptr);`

3. Modify Assignment::generate() to use registers

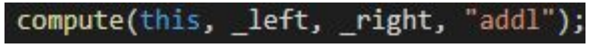
- Phase 5 only had simple assignments
    - **lhs**: scalar variable of type int; **rhs**: integer literal (e.g. x = 5;)
- Let’s now allow for more complex expressions on the right hand side
    - Still assume lhs is a scalar variable of type int
        - Example: x = a * b - (c * (d / e))
    - Later, we will address char’s and the case where lhs is a dereference
- What assignment::generate() should look like now:
    - Generate code for right
    - Load right into a register
    - “movl right, left”
    - Free the register used for right

4. Write generate for add, subtract, multiply, cast

- Start with Add::generate()
  - Don’t forget to declare `virtual void generate()` in Tree.h
  - Check notes for code
- Notice that add, subtract, and multiply all generate the same assembly with the exception of their opcode
- Note: must implement cast::generate() before adding operands with different data types (e.g. int + char)
- You can combine their logic into a generic function which will take the opcode as one of the arguments



```
static void compute(Expression *result, Expression *left, Expression *right,const string &opcode);
```



```
compute(this, _left, _right, "addl");
```

5. More Code Generation for Operators

- Divide/remainder
- Comparative (==, !=, <=, >=, <, >)
- Prefix
    - Negate
    - Not

- **Goal for end of week:** Complete all arithmetic operators, except logical and/or, address/dereference, and assignment