Lab 9

COEN 175 Compilers

Overview For Lab 9

Goal

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

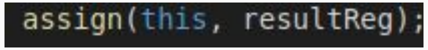
Submission

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

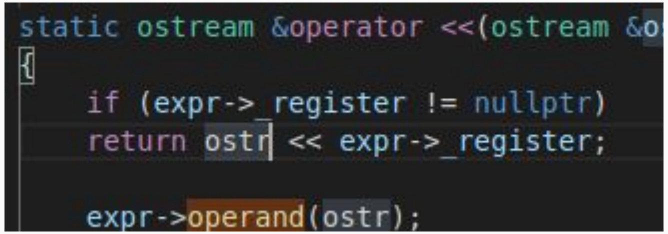
Phase 6 Outline

1. Registers
2. Assignment
3. Add/Sub/Mul
4. Divide/Remainder
5. Relational and Equality Operators
6. Unary Operators
7. Labels and Strings

1. Registers

- Register.h/Register.cpp given!
- Write Helper Functions:
    - Assign - Lecture
        - Most functions end with 

```
assign(this, resultReg);
```

    - Load - Lecture
    - getReg - Lecture
- Modify Ostream Operator
    - 

```
static ostream &operator <<(ostream &o
{
    if (expr->_register != nullptr)
    return ostr << expr->_register;

    expr->operand(ostr);
```

2. Assignment

- Generate right
    - If right is not an immediate then load right
- If left is a dereference
    - Generate and load pointer
    - move (byte or long) right value to dereferenced pointer location
- If not a dereference
    - move (byte or long) right value to left location
- Unassign everything

3. Add/Sub/Mul

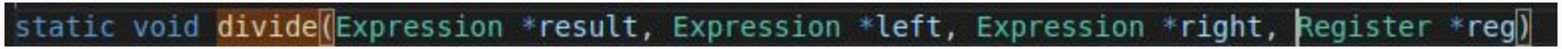
- Create general compute() function

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

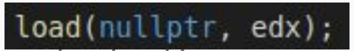
  - Generate left and right
  - Load left
  - Run the operation
  - Unassign right register `assign(right, nullptr);`
  - Assign left register to result expression
- No special case for pointers since amounts already scaled in checker.cpp

4. Divide/Remainder

- Create general divide() function

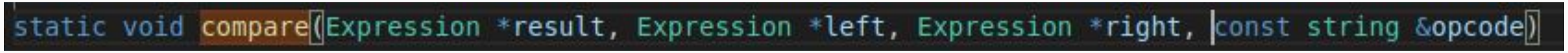


```
static void divide(Expression *result, Expression *left, Expression *right, Register *reg)
```

  - Generate left and right
  - Load left
  - Unload edx  `load(nullptr, edx);`
  - If right is a number load into ecx
  - Sign extend %eax into %edx (use cltd)
  - idivl right
  - Unassign left and right register
  - Assign result to correct register
- Division result in eax, remainder result in edx

5. Relational and Equality Operators

- Create general compare() function



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

  - Generate left and right
  - Load left
  - Compare left and right
  - Unassign left and right
  - Assign result to a register
  - store result of condition code in byte register
  - Zero-extend byte to long (movzbl)
- Condition opcodes in lecture

6. Unary Operators - Cast/Not/Negate

- Cast
  - Strangely easier in assembly
  - Sign extend (movsbl) if going from 1 byte to 4 bytes
- Not
  - cmpl $0, expr
  - sete byteReg
  - zero extend (movzbl)
- Negate
  - Negl
- All
  - Start with generate and load expression
  - End with assign result expression to register

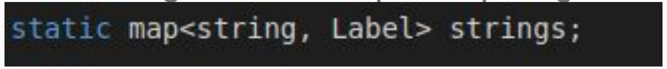
6. Unary Operators - Dereference

- Algorithm
  - Generate and load contained expression
  - Assign resulting expression to register
  - Move value at dereferenced pointer location to result register
- Use size of _type to decide between movb or movl
- Dereference only handles rvalue dereferences (lvalues handled by parent)

6. Unary Operators - Address

- If base is a dereference
    - Generate and load pointer
    - Assign resulting expression to pointer’s register
- If not a dereference
    - Assign resulting expression to a register
    - Get address (leal) of operand and store in result’s register

7. Labels and Strings

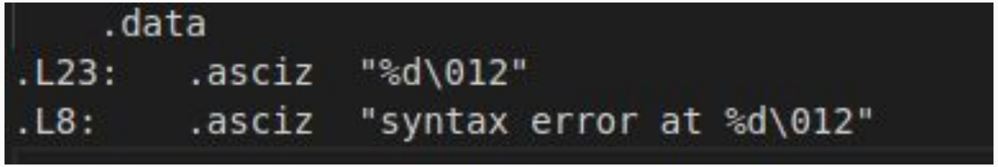
- Label.h/Label.cpp (add to makefile) - Lecture
- Strings
  - Add string to label map to top of generator.cpp
  - 

    ```cpp
    static map<string, Label> strings;
    ```
  - String::operand
    - Check if string already in map (use _value to get string value)
    - If not then create a new label and add an entry for that string
    - Cout the label of that string

7. Labels and Strings (cont.)

- Modify generateGlobals()
- Add .data section for strings
- Loop through string map and output:
  - label: .asciz “string”
- Use escapeString() function in string.cpp for the string literal

Example output:



```
	.data
.L23:	.asciz	"%d\012"
.L8:	.asciz	"syntax error at %d\012"
```

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
- 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

- Make -lib files if you want to use operations you haven't implemented yet
- 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