CS75 Project 4: Compiler Optimizations

Spring 2007, Meeden

Due by midnight, Monday, May 7

Required Functionality

As you enhance your compiler with optimizations remember that the two main concerns are safety and profitability. We only want to modify the code in ways that improve it while still maintaining its meaning. Therefore you should only include optimizations that you can guarantee will always be correct.

Implement two choices from the following list as optimizations that will be done prior to code generation. This phase of the optimization will take the abstract syntax tree as input and return an updated abstract syntax tree as output.

1. Eliminating redundant expressions using value numbering
2. Simplifying expressions using identities
3. Changing more expensive operations to equivalent cheaper operations
4. Constant propagation
5. Constant folding
6. Eliminating dead code
7. Eliminating unreachable code
8. Moving invariant code from within a loop
9. Loop unrolling

You must also implement peephole optimization. This phase of the optimization will take in the code table containing MIPS instructions as input and return an updated code table as output.

Your compiler should be implemented so that it can be invoked in the following ways.

- Without optimizations:

```
python compiler.py foo.c-- foo.mips
```

- With only the optimizations that are done before code generation:

```
python compiler.py -o1 foo.c-- foo.mips
```

- With only the optimizations that are done after code generation:

```
python compiler.py -o2 foo.c-- foo.mips
```

- With all possible optimizations:

```
python compiler.py -o3 foo.c-- foo.mips
```

Optional Features

Once you have successfully completed all aspects of the basic C- - compiler (including optimizations), you can then attempt additional features for extra credit. An incomplete implementation of the required functionality plus an added feature will result in a lower grade than a complete implementation of the required functionality without any additional features.

Possible additional features:

- no limit on the number of function parameters
- attaching line numbers to the AST for better error reporting during code generation
- for loops
- supporting assignment statements as expressions, such as `a = b = c = 0;`
- floats
- strings
- pass by reference for int and char variables

Project Report

In your project directory you must include a project report. Using at most two pages, give an overview of the design of your compiler. Explain the overall approach as well as each major component. You should describe the optimizations that you've included, as well as any additional features you've provided. If there are any aspects of your compiler that have not been fully implemented or are not working properly, your should discuss them here.