This repository Search Pull requests Issues Gist

gangliao / TIGER Private

Unwatch 2 Unstar 1 Fork 0

Code Issues 1 Pull requests 2 Projects 1 Wiki Pulse Graphs Settings

implement a full compiler based on c++ 11

Edit

compiler parser codegenerator scanner Manage topics

308 commits 9 branches 1 release 2 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

gangliao committed on GitHub Merge pull request #106 from gangliao/gang_back_end ... Latest commit ab6effc 4 minutes ago

cmake	add cmake compiler	16 days ago
deprecated	move testcase1 into deprecated	4 hours ago
design_doc	refine doc	6 minutes ago
img	add demo	2 hours ago
src	Add more	an hour ago
testCases2	Fix test5 cfg mode	6 hours ago
.gitignore	Add cmake	16 days ago
.pre-commit-config.yaml	Add clang-format	2 months ago
.travis.yml	load and store vars during blocks	2 days ago
CMakeLists.txt	Add cmake	16 days ago
LICENSE	Initial commit	3 months ago
Phase1_Testing_and_Output.pdf	refine doc	6 minutes ago
Phase2_Testing_and_Output.pdf	fix syntax words	21 minutes ago
README.md	refine doc	6 minutes ago
report.pdf	Update report	a month ago

README.md

TIGER - A Tiny Full Compiler

build error

This compiler includes both front end and back end.

> Front end: Grammar Rules, LL(1) Parse Table, Syntax and Semantic Check and Intermediate Code.

> Back end: IR Optimization (Intra-block CFG optimization), MIPS Register Allocation, Instruction Selection and Code Generation.

How to Build

0. development environment

   Currently, this project repository is maintained on github privately and also been deployed on Travis CI. It supports both Ubuntu and Mac OS X.

1. build:

```
    # cd project dir
    cd Tiger-Compiler
```

```
        # build scanner, parser, generator
        mkdir build && cd build
        # cmake building tool
        cmake ..
        make -j4
```

2. run:

   You can parsing test cases named `*.tiger` under `/testCases2` to generate IR code. Default it will utilize CFG optimized technique to generate MIPS asm code.

   ```
           # verbose mode: "-d to implement a verbose mode"
           ./src/parser <filename> -d
   ```

   If you want to use the naive mode to generate asm code, simply issue:

   ```
           ./src/parser <filename> -d -naive
   ```

3. test:

   In testCases2 directory, it includes a test script `test.sh` to execute all test cases and generate the corresponding asm files `*.naive.s` and `*.cfg.s`.

   After souce code is compiled, you can simply issue the commands:

   ```
   cd testCases2
   sh ./test.sh
   ```

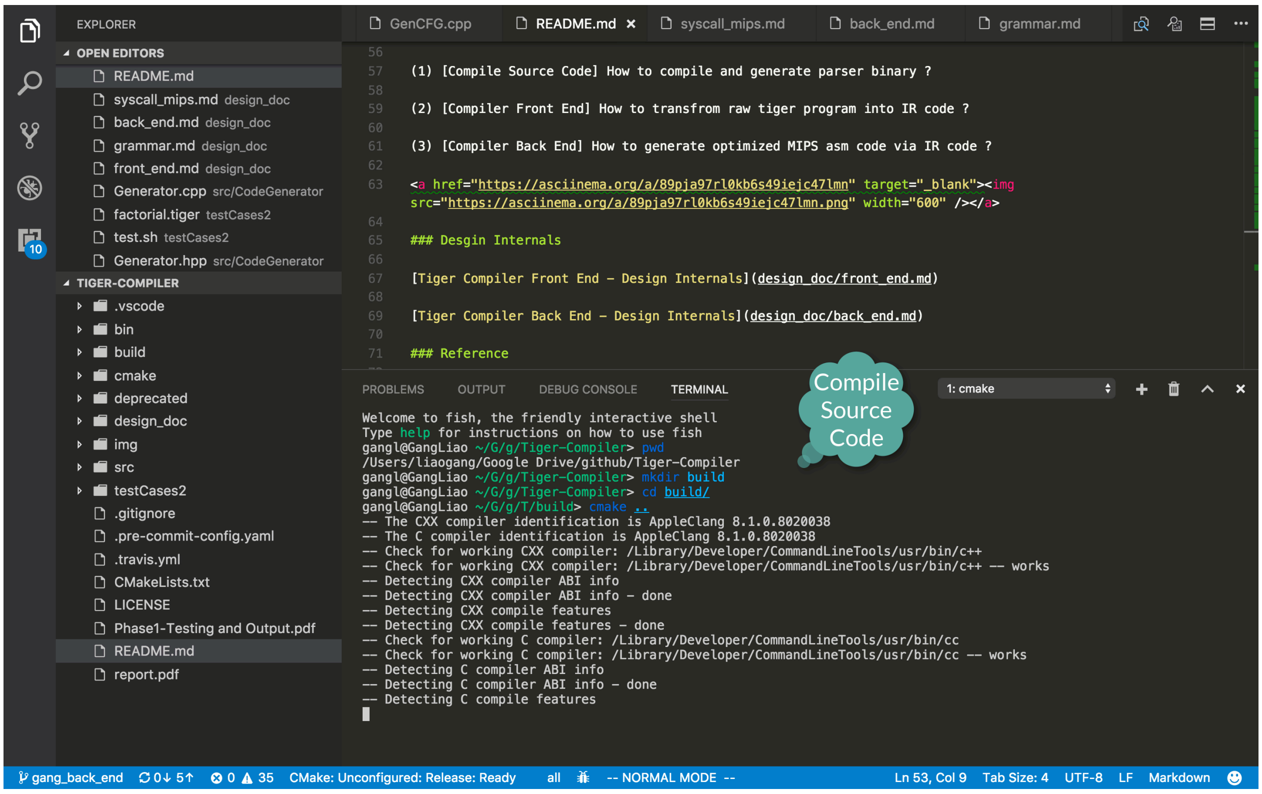
Demo

This Demo shows that

```
(1) [Compile Source Code] How to compile and generate parser binary ?

(2) [Compiler Front End] How to transfrom raw tiger program into IR code ?

(3) [Compiler Back End] How to generate optimized MIPS asm code via IR code ?
```



```

(1) [Compile Source Code] How to compile and generate parser binary ?

(2) [Compiler Front End] How to transfrom raw tiger program into IR code ?

(3) [Compiler Back End] How to generate optimized MIPS asm code via IR code ?

<a href="https://asciinema.org/a/89pja97rl0kb6s49iejc47lmn" target="_blank"><img
src="https://asciinema.org/a/89pja97rl0kb6s49iejc47lmn.png" width="600" /></a>

### Desgin Internals

[Tiger Compiler Front End - Design Internals](design_doc/front_end.md)

[Tiger Compiler Back End - Design Internals](design_doc/back_end.md)

### Reference
```

```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
gangl@GangLiao ~/G/g/Tiger-Compiler> pwd
/Users/liaogang/Google Drive/github/Tiger-Compiler
gangl@GangLiao ~/G/g/Tiger-Compiler> mkdir build
gangl@GangLiao ~/G/g/Tiger-Compiler> cd build/
gangl@GangLiao ~/G/g/T/build> cmake ..
-- The CXX compiler identification is AppleClang 8.1.0.8020038
-- The C compiler identification is AppleClang 8.1.0.8020038
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
```

Desgin Internals

Tiger Compiler Front End - Design Internals

Tiger Compiler Back End - Design Internals

Test Cases

We passed all tests cases which provided by TA.

Please check out the details in report Phase2_Testing_and_Output.pdf from current directory, which includes test cases and their quality comparisons for naive and CFG intra-block register allocation.

Accomplishment

- [x] Register allocation code
  - [x] Naive
  - [x] CFG and intra block allocation
  - [ ] EBB and intra-EBB allocation
  - [x] Whole function register allocation
  - [x] Live Range Analysis and Graph Coloring
- [x] Instruction selection and generation code
- [x] Passes tests using generated code executing on simulator.
- [x] Report (desgin Internals, how to build, run, code quality comparisions, etc.)