



riscv-check

Маленькое приложение для большой задачи

Автор: Зернов Семён Николаевич

Научный руководитель: Кирилл Константинович Смирнов

Научный руководитель: Владимир Александрович Кутуев

Санкт-Петербургский государственный университет

28 июля 2023г.

- Описание проекта
- Цели и задачи
- Аналоги
- Технологический стек
- Особенности реализации

riscv-check — утилита для анализа кодогенерации

```
chansons@chansons:~/Repositories/riscv-check$ python3 riscv_check.py --help
usage: riscv_check.py [-h] [--format {csv,default}] [--params PARAMS] compiler march mabi opt_level

Compiler options

positional arguments:
  compiler      Compiler name
  march        Architecture parameter
  mabi         ABI parameter
  opt_level     Optimization level

options:
  -h, --help            show this help message and exit
  --format {csv,default}
                        output as type of formatted output
  --params PARAMS      forwards parameters directly to compiler
```

Цель работы — создать автоматическую систему анализа кодогенерации

Задачи:

- Разработать архитектуру приложения
 - ▶ shell compatible
 - ▶ Лёгкая для расширения
- Выбрать технические средства для реализации

- **llvm-lit**¹ (внутреннее приложение для тестирования LLVM и Clang)
- **gcc backend**² (используются отдельные тесты внутри модуля кодогенерации)

¹<https://llvm.org/docs/CommandGuide/lit.html>

²<https://patchwork.ozlabs.org/project/gcc/list/?series=263719>

- Python 3.10+ (реализация логики)
- C (транслируемый в ассемблер язык)
- CI
 - ▶ pylint (статический анализатор)
 - ▶ муру (проверка типов)

• Утилита shell-дружелюбна

```
chansons@chansons:~/Repositories/riscv-check$ python3 riscv_check.py --format=csv riscv64-unknown-linux-gnu-gcc rv64gv_zba_zbb_zbc_zbs lp64d 3 --target=riscv64
chansons@chansons:~/Repositories/riscv-check$ python3 riscv_check.py --format=csv riscv64-unknown-linux-gnu-gcc rv64gv_zba_zbb_zbc_zbs lp64d 3 > gcc.txt
```

Рис.: Перенаправим вывод

```
chansons@chansons:~/Repositories/riscv-check$ diff -y --suppress-common-lines gcc.txt clang.txt
{'compiler': 'riscv64-unknown-linux-gnu-gcc', 'arch': 'rv64g' | {'compiler': '/home/chansons/Tools/sc-dt_2022.12-sp1/llvm/bin
7_zbb_rev8_rev8_32_manual,False | 7_zbb_rev8_rev8_32_manual,True
8_zbb_rev8_rev8_64_manual,False | 8_zbb_rev8_rev8_64_manual,True
10_zbb_sext_b_sext_b32,False | 10_zbb_sext_b_sext_b32,True
12_zbb_sext_h_sext_h32,False | 12_zbb_sext_h_sext_h32,True
39_zbs_binv_binv32,False | 39_zbs_binv_binv32,True
42_zbs_bset_bset32,False | 42_zbs_bset_bset32,True
50_zbs_bclr_bclr32,False | 50_zbs_bclr_bclr32,True
51_zbs_bclr_bclr64,False | 51_zbs_bclr_bclr64,True
52_zbs_bext_bext64,False | 52_zbs_bext_bext64,True
53_zbs_bext_bext32,False | 53_zbs_bext_bext32,True
54_zbc_clmulh_clmulh64_builtin,False | 54_zbc_clmulh_clmulh64_builtin,True
55_zbc_clmulh_clmulh32_builtin,False | 55_zbc_clmulh_clmulh32_builtin,True
56_zbc_clmulr_clmulr64_builtin,False | 56_zbc_clmulr_clmulr64_builtin,True
57_zbc_clmulr_clmulr32_builtin,False | 57_zbc_clmulr_clmulr32_builtin,True
58_zbc_clmul_clmul64_builtin,False | 58_zbc_clmul_clmul64_builtin,True
59_zbc_clmul_clmul32_builtin,False | 59_zbc_clmul_clmul32_builtin,True
64_zba_sh2add_sh2add32,False | 64_zba_sh2add_sh2add32,True
67_zba_sh3add_sh3add32,False | 67_zba_sh3add_sh3add32,True
69_zba_sh1add_sh1add32,False | 69_zba_sh1add_sh1add32,True
```

Рис.: diff

- Утилита независима от bitmanip расширения
- Есть возможность добавить произвольное число тестов
- Попробуем добавить атомарную операцию amoadd.d³

```
chansons@chansons:~/Repositories/riscv-check$ mkdir tests/atomics
chansons@chansons:~/Repositories/riscv-check$ mkdir tests/atomics/amoadd.d
chansons@chansons:~/Repositories/riscv-check$ vim tests/atomics/amoadd.d/my_test.c
```

Рис.: Создание теста для инструкции amoadd.d

```
1 #include <stdint.h>
2 #include <stdatomic.h>
3
4 int64_t test(_Atomic int64_t count, int64_t n) {
5     return atomic_fetch_add_explicit(&count, n, memory_order_relaxed);
6 }
```

Рис.: Имплементация amoadd.d

³<https://msyksphinz-self.github.io/riscv-isadoc/html/rv64a.html#amoadd-d>


```
amoadd.d  
TEST #1 PASSED: my_test is generated  
  
Total: Passed 55/70 tests
```

Рис.: Результат

Расширяемость

- Прямая возможность добавить свой формат

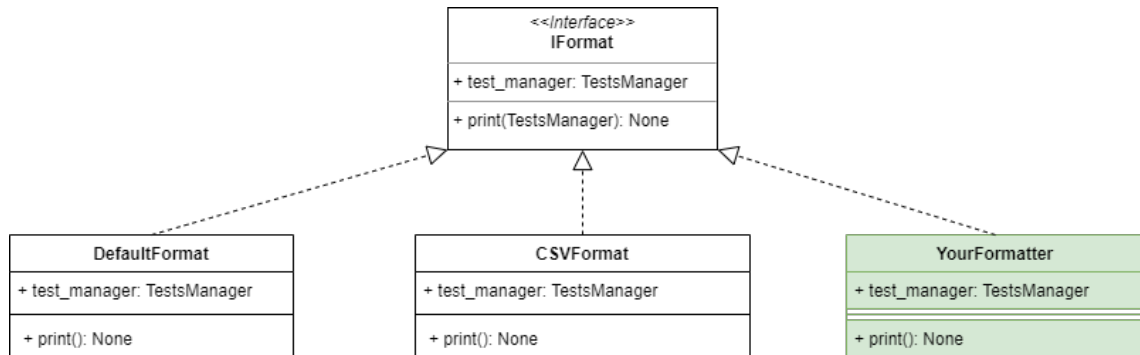


Рис.: Иерархия классов

Результаты

- Поддержка bitmanip в современных компиляторах (gcc и clang⁴) сильно отличается

```
slli.uw
TEST #1 FAILED: slli.uw is not supported

add.uw
TEST #1 PASSED: add.uw is generated

sh2add
TEST #1 FAILED: sh2add32 is not supported
TEST #2 PASSED: sh2add64 is generated

sh3add
TEST #1 PASSED: sh3add64 is generated
TEST #2 FAILED: sh3add32 is not supported

sh1add
TEST #1 PASSED: sh1add64 is generated
TEST #2 FAILED: sh1add32 is not supported

Total: Passed 35/69 tests
```

Рис.: gcc 13.1.0

```
add.uw
TEST #1 PASSED: add.uw is generated

sh2add
TEST #1 PASSED: sh2add32 is generated
TEST #2 PASSED: sh2add64 is generated

sh3add
TEST #1 PASSED: sh3add64 is generated
TEST #2 PASSED: sh3add32 is generated

sh1add
TEST #1 PASSED: sh1add64 is generated
TEST #2 PASSED: sh1add32 is generated

Total: Passed 54/69 tests
```

Рис.: clang 16.0.0 из Syntacore SW Tools 202212

⁴<https://syntacore.com/page/products/sw-tools>

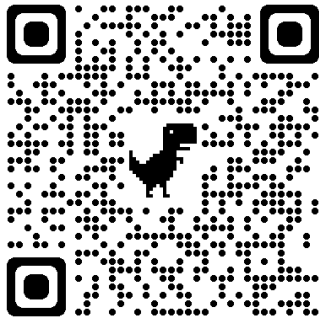


Рис.: Репозиторий (ссылка)