

# RISC-V Developers Forum



7 декабря, Москва

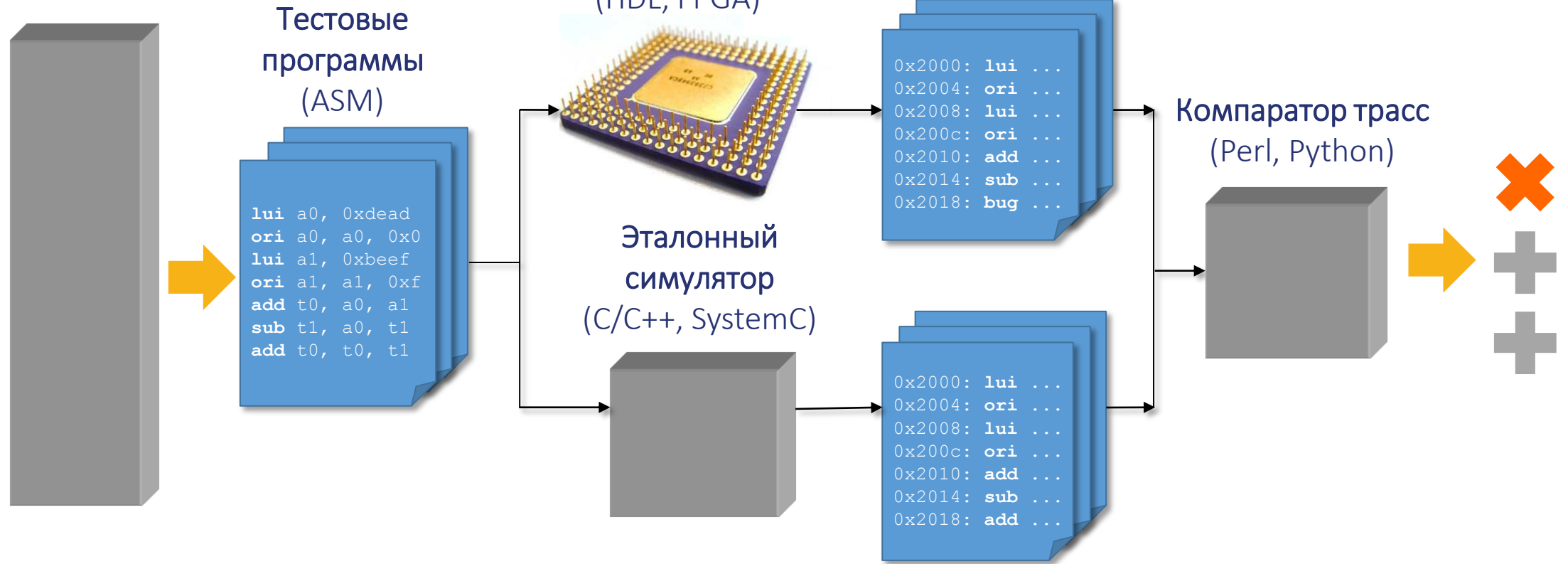
## Генератор тестовых программ MicroTESK for RISC-V

Андрей Татарников



Институт системного программирования  
им. В.П. Иванникова Российской академии наук

## Генератор тестовых программ

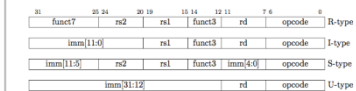


- Не должно быть
  - нарушений предусловий: **UNPREDICTABLE**
  - обращений к **неинициализированным** данным
  - **зацикливаний** в тестовых программах
- Должна быть возможность
  - покрытия **интересных ситуаций** (corner cases)
  - создания **встроенных проверок** (self-checks)

## 2.2 Base Instruction Formats

In the base ISA, there are four core instruction formats (R/I/S/U), as shown in Figure 2.2. All are a fixed 32 bits in length and must be aligned on a four-byte boundary in memory. An instruction address misaligned exception is generated on a taken branch or unconditional jump if the target address is not four-byte aligned. No instruction fetch misaligned exception is generated for a conditional branch that is not taken.

The alignment constraint for base ISA instructions is relaxed to a two-byte boundary when instruction extensions with 16-bit lengths or other odd multiples of 16-bit lengths are added.



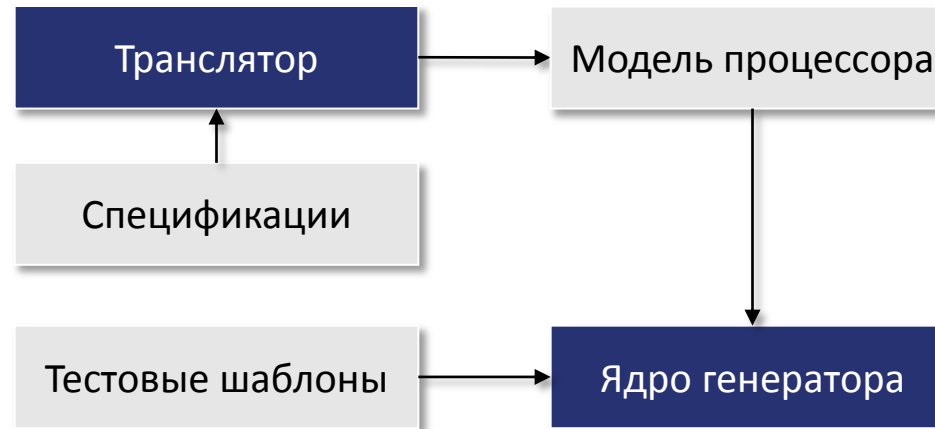
### Параметры генерации

- ❑ Последовательности команд
- ❑ Ограничения на данные
- ❑ Распределения вероятностей
- ❑ и т.д. и т.п.



Инженеры-  
верификаторы

## Генератор на базе MicroTESK



### Тестовые программы

```
lui a0, 0xdead
ori a0, a0, 0x0
lui a1, 0xbeef
ori a1, a1, 0xf
add t0, a0, a1
sub t1, a0, t1
add t0, t0, t1
```

```
// Константы
#ifdef RV64I
    let XLEN = 64
#else
    let XLEN = 32
#endif
let MEM_SIZE = 2 ** (XLEN - 2)

// Типы данных
type BYTE = card(8) // Unsigned
type WORD = card(32) // Unsigned
type XWORD = card(XLEN) // Unsigned
type XINT = int(XLEN) // Signed
...
```

1

```
// Регистры и псевдонимы
reg XREG [32, XWORD] // 32 регистра XWORD
reg SP[XWORD] alias = XREG[2]
reg PC[XWORD]
...

// Режимы доступа к регистрам
mode REG(i: card(5)) = XREG[i]
    syntax = format("x%d", i) // E.g. x13
    image = format("%5s", i) // E.g. 01101
...
```

2

```
// Массив физической памяти
mem MEM [MEM_SIZE, WORD]
```

3

5

```

// Операция сложения
op add(rd: X, rs1: X, rs2: X)
  syntax = format("add %s, %s, %s",
    rd.syntax, rs1.syntax, rs2.syntax)
  image = format("0000000%s%s000%s0110011",
    rs2.image, rs1.image, rd.image)
  action = { // Семантика инструкции
    rd = rs1 + rs2;
  }

// Операция ветвления
op beq(rs1: X, rs2: X, imm: card(12))
  syntax = format("beq %s, %s, %<label>d", ...)
  image = ...
  action = {
    if rs1 == rs2 then
      PC = PC + (sign_extend(XWORD, imm) << 1);
    endif;
  }

```

1

```

// Группы операций (ИЛИ-правила)
op ALU = ADD | ...
op BPU = BEQ | ...
op Op  = ALU | BPU | ...

```

2

```

// Композиция операций (И-правило)
op instruction(operation: Op)
  syntax = operation.syntax
  image = operation.image
  action = {
    XREG[0] = 0;
    prev_pc = PC;

    operation.action;
    if PC == prev_pc then
      PC = PC + 4;
    endif;
  }

```

3

```
class MyTemplate < RiscVBaseTemplate
  def run
    block(:combinator => 'product',
          :compositor => 'random') {
      iterate {
        xor x(_), x(_), x(_)
        lui x(_), _
      }
      iterate {
        and x(_), x(_), x(_)
        or  x(_), x(_), x(_)
      }
      iterate {
        auipc x(_), _
      }
    }.run
  end
end
```

1

Всего  $2 \times 2 \times 1 = 4$  тестовых примера

```
# Инициализация
ori a7, a7, 0x2d7
slli a7, a7, 0xb
ori a7, a7, 0x1
slli a7, a7, 0xb
ori a7, a7, 0x3d2
ori t3, t3, 0x164
slli t3, t3, 0xb
ori t3, t3, 0x52b
slli t3, t3, 0xb
ori t3, t3, 0x24e

# Тестовое воздействие
and s4, a7, a7
xor s8, s4, t3
auipc t2, 0xafc37
```

2

```
# Инициализация регистров X
preparator(:target => 'X') {
  if rv64i then
    ori target, zero, value(53, 63)
    slli target, target, 11
    ori target, target, value(42, 52)
    slli target, target, 11
    ori target, target, value(32, 41)
    slli target, target, 10
  end
  ori target, target, value(21, 31)
  slli target, target, 11
  ori target, target, value(10, 20)
  slli target, target, 11
  ori target, target, value(0, 9)
}
```

1

```
# Оптимизированная инициализация регистров
preparator(
  :target => 'X',
  :mask => "00000000000000000000") {
  or target, zero, zero
}
preparator(
  :target => 'X',
  :mask => "FFFFFFFFFFFFFFFFFFFFFF") {
  nor target, zero, zero
}
```

2

```
# Инициализация памяти
memory_preparator(:size => 64) {
  prepare t1, value
  prepare t2, address
  sd t1, t2, 0
}
```

3

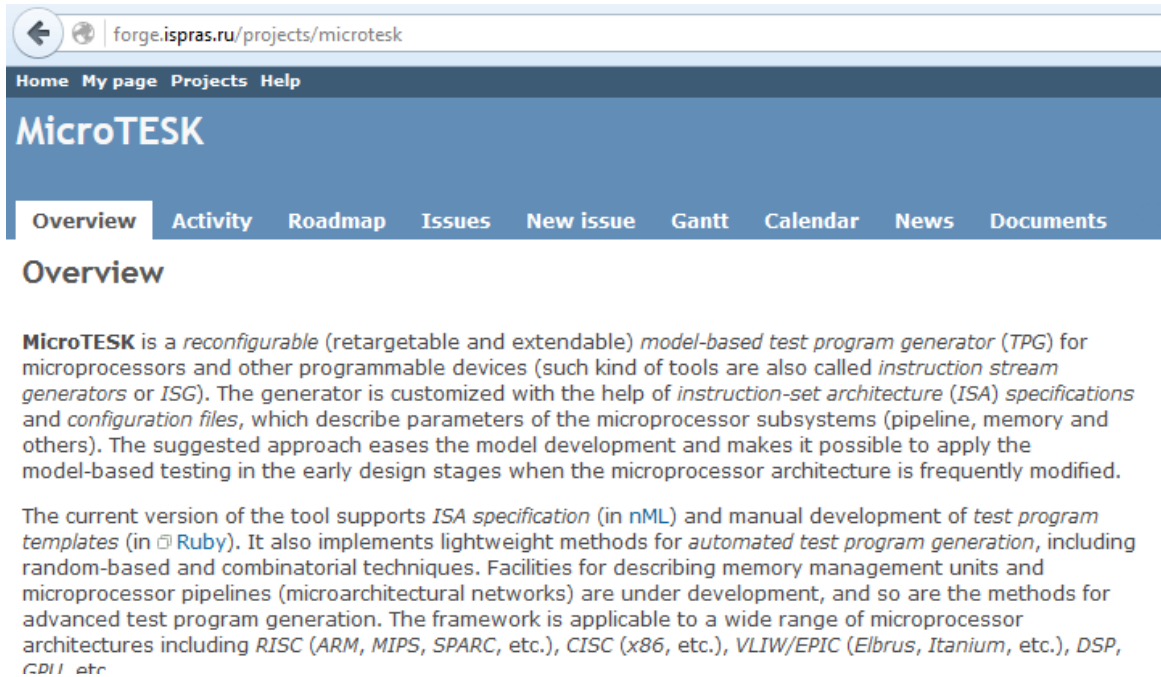


- **Тестовые данные** (значения операндов)
  - Случайная генерация (**range**, **dist**, ...)
  - Разрешение ограничений (внешние **SMT-решатели**)
- **Тестовые последовательности** (потоки инструкций)
  - Комбинаторная генерация (**combinator**, **compositor**, ...)
  - Специализированные генераторы (**branch**, **memory**, ...)

- Высокая **автоматизация** на базе спецификаций
- Гибкая настройка на **разные архитектуры**
- **Расширяемость**
- Поддержка RISC-V
- **Open source**
- Применяется в **реальных проектах**

Архитектура	RISC-V
RISC-V Instruction Set Manual Volume I: User-Level ISA (v. 2.2) RV32I, RV64I, RV32M, RV64M, RV32A, RV64A, RV32F, RV64F, RV32D, RV64D	145 страниц
Поддерживаемые инструкции	201 инструкция
Спецификации ISA (nML)	2300 строк

- Повышение **уровня автоматизации** генерации
- **Online**-генерация тестовых программ
- Оценка **тестового покрытия** на уровне архитектуры
- **Статический анализ** и верификация бинарного кода



The screenshot shows a web browser window with the address bar displaying 'forge.ispras.ru/projects/microtesk'. The page has a navigation menu with 'Home', 'My page', 'Projects', and 'Help'. Below the menu, the title 'MicroTESK' is displayed. A secondary navigation bar includes 'Overview', 'Activity', 'Roadmap', 'Issues', 'New issue', 'Gantt', 'Calendar', 'News', and 'Documents'. The 'Overview' section is active, containing a detailed description of the tool as a reconfigurable model-based test program generator for microprocessors, supported by ISA specifications and configuration files. It also lists supported architectures like ARM, MIPS, SPARC, x86, and VLIW/EPIC.



<http://www.microtesk.org/>

<http://forge.ispras.ru/projects/microtesk-riscv>

<https://www.facebook.com/MicroTESK>

[microtesk-support@ispras.ru](mailto:microtesk-support@ispras.ru)

```
        .text
__start:  addi t1, zero, 0x18
          addi t2, zero, 0x21
cycle:   beq t1, t2, done
          slt t0, t1, t2
          bne t0, zero, if_less
          nop
          sub t1, t1, t2
          j cycle
          nop
if_less: sub t2, t2, t1
          j cycle
done:    add t3, t1, zero
```

