

Test Generator **MicroTESK** for **RISC-V**

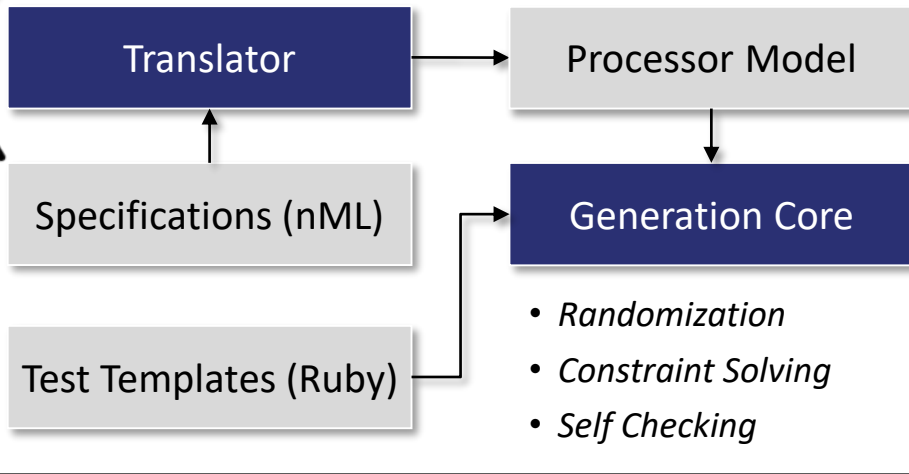
2.2 Base Instruction Formats

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 addresses with 16-bit lengths or other odd multiples of 16-bit lengths are used.

R	OP	RS1	RS2	RS3	RS4	OPCODE	U
funct7	rd	rs1	rs2	rs3	rs4	opcode	R-type
imm[11:0]	rd	rs1	rs2	rs3	rs4	opcode	I-type
imm[11:0]	rd	rs1	rs2	imm[4:0]	opcode	opcode	S-type
imm[31:12]	rd	opcode	opcode	opcode	opcode	opcode	U-type

MicroTESK (Open-Source Framework)



Test Programs
(Assembly Code)

```
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
```



Alexander Kamkin



Andrei Tatarnikov

RISC-V Specifications and Test Templates

Instruction Set Architecture	RISC-V
RISC-V Instruction Set Manual Volume I: User-Level ISA (v. 2.2)	145 pages
Specified Instructions	226 instructions
ISA Specifications	3300 LOC

```
// ISA Specification in nML
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 = {
    rd = rs1 + rs2;
  }
}
```

```
# Test Program Template in Ruby
class MyTemplate < RiscVBaseTemplate
  def run
    block(:combinator => 'product') {
      iterate {
        xor x(_), x(_), x(_)
        lui x(_), _
      }
      iterate {
        and x(_), ...
        or x(_), ...
      }
      iterate {
        auipc x(_), _
      }
    }.run      2 × 2 × 1 = 4
  end          test cases
end
```

```
# Test Program
# Initialization
ori a7, a7, 0x2d7
slli a7, a7, 11
ori a7, a7, 0x1
slli a7, a7, 11
ori a7, a7, 0x3d2
ori t3, t3, 0x164
slli t3, t3, 11
ori t3, t3, 0x52b
slli t3, t3, 11
ori t3, t3, 0x24e
# Stimulus
and s4, a7, a7
xor s8, s4, t3
auipc t2, 0xafc37
```