Симуляция RISC-V ISA

Симуляция

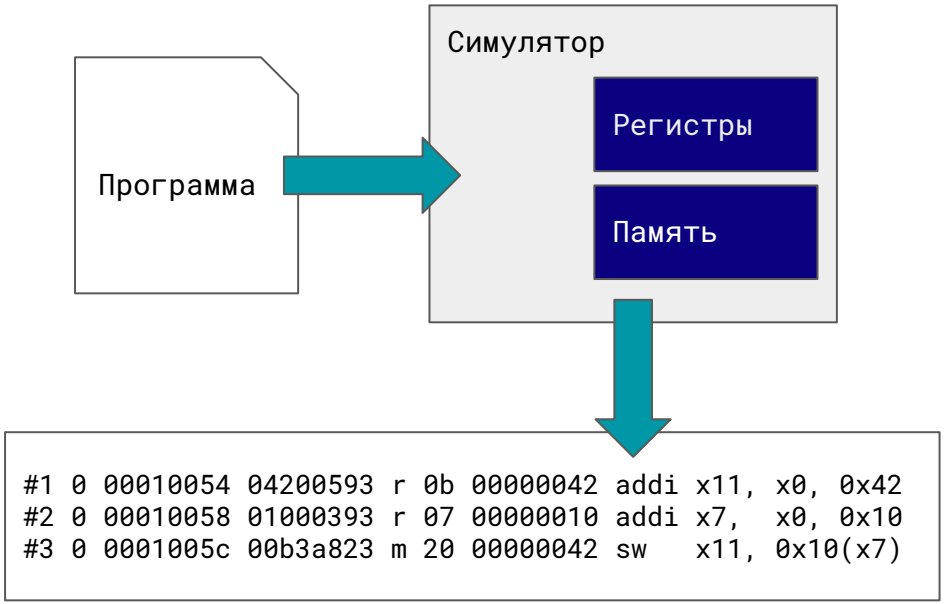
Симулятор — программа, которая моделирует работу некоторой компьютерной архитектуры

- Потактовый симулятор (cycle-accurate simulator)
  - Симулирует работу микроархитектуры с точностью до одного такта
  - Может использоваться для анализа производительности.
- Функциональный симулятор (functional simulator)
  - Не обращает внимание на время исполнения инструкций

Симуляция ISA

Симулятор архитектуры набора команд (instruction set simulator, ISS)

- Инструкции
- Регистры
- Память



Симуляция RISC-V ISA

- Spike RISC-V ISA Simulator
  - https://github.com/riscv/riscv-isa-sim
- Western Digital’s RISC-V SweRV ISS (Whisper)
  - https://github.com/westerndigitalcorporation/swerv-ISS

Name	Links	Priv. spec	User spec	License	Maintainers
rocket	GitHub	1.11-draft	2.3-draft	BSD	SiFive, UCB Bar
freedom	GitHub	1.11-draft	2.3-draft	BSD	SiFive
Berkeley Out-of-Order Machine (BOOM)	GitHub	1.11-draft	2.3-draft	BSD	Esperanto, UCB Bar
ORCA	GitHub		RV32IM	BSD	VectorBlox
RI5CY	GitHub		RV32IMC	Solderpad Hardware License v. 0.51	ETH Zurich, Università di Bologna
Zero-riscy	GitHub		RV32IMC	Solderpad Hardware License v. 0.51	ETH Zurich, Università di Bologna
Ariane	Website,GitHub		RV64IMC	Solderpad Hardware License v. 0.51	ETH Zurich, Università di Bologna
Roa Logic RV12	GitHub	1.9.1	2.1	Non-Commercial License	Roa Logic
SCR1	GitHub	1.10	2.2, RV32I/E[MC]	Solderpad Hardware License v. 0.51	Syntacore
Hummingbird E200	GitHub	1.10	2.2, RV32IMAC	Apache 2.0	Bob Hu
Shakti	Website,GitLab	1.11	2.2, RV64IMAFDC	BSD	IIT Madras
ReonV	GitHub			GPL v3	
PicoRV32	GitHub		RV32I/E[MC]	ISC	Clifford Wolf
MR1	GitHub		RV32I	Unlicense	Tom Verbeure
SERV	GitHub		RV32I	ISC	Olof Kindgren
SweRV EH1	GitHub		RV32IMC	Apache 2.0	Western Digital Corporation
Reve-R	GitHub	1.10	RV32IMAC	Apache 2.0	Gavin Stark

riscv.org/risc-v-cores/

Симуляция RV32I

Файл sw.s

```
.text
.globl _start
.globl _finish

_start:
    li   x11, 0x42
    li   x7,  0x10
    sw   x11, 0x10(x7)

_finish:
     nop
```

```
$ riscv64-linux-gnu-as -march=rv32i -mabi=ilp32 -c sw.s -o sw.o

$ riscv64-linux-gnu-ld -melf32lriscv sw.o -o sw.elf

$ riscv64-linux-gnu-objdump -d sw.elf

00010054 <_start>:
   10054:  04200593              li    a1,66
   10058:  01000393              li    t2,16
   1005c:  00b3a823              sw    a1,16(t2)
00010060 <_finish>:
   10060:  00000013              nop

$ whisper --isa=i -l sw.elf

#1 0 00010054 04200593 r 0b       00000042 addi x11, x0, 0x42
#2 0 00010058 01000393 r 07       00000010 addi x7,  x0, 0x10
#3 0 0001005c 00b3a823 m 00000020 00000042 sw   x11, 0x10(x7)
Stopped -- Reached end address
Retired 3 instructions in 0.00s  15306 inst/s
```

Программа на C для RV32IM

Файл loader.s

```
.text
.globl _start
.globl _finish
.globl main

_start:
    call main

_finish:
    nop
```

Файл factorial.c

```
typedef unsigned int uint32_t;
typedef unsigned char uint8_t;
#define OUT_ADDR ((uint8_t *)0x100)

uint32_t f(uint32_t n) {
    if (n > 1)
        return n * f(n - 1);
    else
        return 1;
}

void main() {
    *(volatile uint32_t *)OUT_ADDR = f(5);
}
```

```
$ riscv64-linux-gnu-gcc -nostdlib -march=rv32im -mabi=ilp32 -c factorial.c -o factorial.o
$ riscv64-linux-gnu-ld -melf32lriscv loader.o factorial.o -o factorial.elf
```

Симуляция RV32IM

```
$ whisper --isa=im -l factorial.elf

#1   0 00010054 060000ef r 01         00010058  jal      x1, 0x60
#2   0 000100b4 ff010113 r 02         fffffff0  addi     x2, x2, -0x10
.................
#101 0 00010094 02f707b3 r 0f         00000078  mul      x15, x14, x15
.................
#109 0 000100d8 00f4a023 m 00000100   00000078  sw       x15, 0x0(x9)
#110 0 000100dc 00000013 r 00         00000000  addi     x0, x0, 0x0
#111 0 000100e0 00c12083 r 01         00010058  lw       x1, 0xc(x2)
#112 0 000100e4 00812403 r 08         00000000  lw       x8, 0x8(x2)
#113 0 000100e8 00412483 r 09         00000000  lw       x9, 0x4(x2)
#114 0 000100ec 01010113 r 02         00000000  addi     x2, x2, 0x10
#115 0 000100f0 00008067 r 00         00000000  jalr     x0, 0x0(x1)
Stopped -- Reached end address
Retired 115 instructions in 0.00s  210622 inst/s
```

GitHub

github.com/viktor-prutyanov/drec-fpga-intro