

SchoolRISCV. Пример добавления инструкции

<https://github.com/zhelnio/schoolRISCV>

Stanislav Zhelnio, 2020

Общий порядок

- анализ спецификации
- тестовая программа
- модификация тракта данных
- модификация АЛУ
- модификация устройства управления
- проверка в симуляторе
- оценка максимальной частоты
- проверка на отладочной плате

SLLI: анализ спецификации

31	25 24	20 19	15 14	12 11	7 6	0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode	
7	5	5	3	5	7	
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM	
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM	
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM	

Shifts by a constant are encoded as a specialization of the I-type format. The operand to be shifted is in *rs1*, and the shift amount is encoded in the lower 5 bits of the I-immediate field. The right shift type is encoded in bit 30. SLLI is a logical left shift (zeros are shifted into the lower bits);

SLLI: тестовая программа

- скопировать каталог **program/00_counter** с НОВЫМ ИМЕНЕМ

```
cp -r program/00_counter program/03_slli
```

- откорректировать программу **main.S**

```
# program/03_slli/main.S
#
# RISC-V new instruction (slli) software test
#
        .text
start:   addi a0, zero, 1          # a0 = 1
shift:   slli a0, a0, 1           # a0 = a0 << 1
        beq a0, zero, start      # if a0 == 0 then start
        beq zero, zero, shift    #           else shift
```

SLLI: Проверка тестовой программы

в каталоге `program/03_slli`

```
make rars
```

The screenshot shows a debugger window with a menu bar (File, Edit, Run, Settings, Tools, Help) and a toolbar with various icons. The main window is divided into two panes. The left pane, titled 'Text Segment', displays assembly code with columns for 'Bkpt', 'Address', 'Code', and 'Basic'. The right pane, titled 'Control and Status', shows a 'Registers' table.

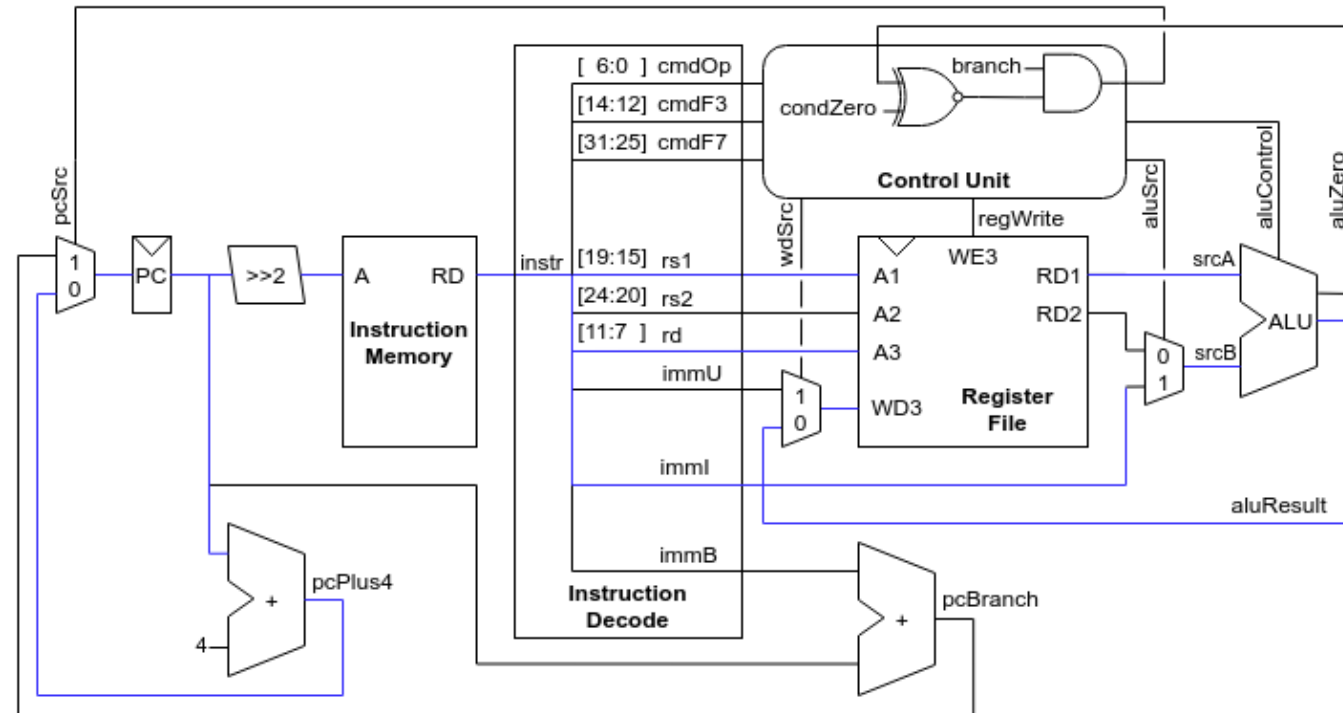
Bkpt	Address	Code	Basic
<input type="checkbox"/>	0x00400000	0x00100513	addi x10,x0,0x00000001
<input type="checkbox"/>	0x00400004	0x00151513	slli x10,x10,0x00000001
<input type="checkbox"/>	0x00400008	0xfe050ce3	beq x10,x0,0xffffffffc
<input type="checkbox"/>	0x0040000c	0xfe000ce3	beq x0,x0,0xffffffffc

Name	Num...	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffefffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00001000
a1	11	0x00000000

SLLI: Модификация тракта данных

Не требуется: коммутация тракта данных аналогична инструкции ADDI.

Instruction	cmdOp	cmdF3	cmdF7	aluSrc	aluControl	wdSrc	regWrite	pcSrc	branch	condZero
ADDI	0010011	000	???????	1	000	0	1	0	0	0



SLLI: Модификация АЛУ

```
// sr_cpu.vh
...
`define ALU_ADD      3'b000  // A + B
...
```

```
// sr_cpu.v
module sr_alu
    always @ (*) begin
        case (oper)
            default      : result = srcA + srcB;
            ...
            `ALU_SLLI : result = srcA << srcB [4:0];
        endcase
    end
    ...
endmodule
```

SLLI: Модификация устройства управления 1

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
funct7			rs2			rs1		funct3		rd		opcode		R-type
imm[11:0]						rs1		funct3		rd		opcode		I-type
imm[11:5]			rs2			rs1		funct3		imm[4:0]		opcode		S-type
imm[12 10:5]			rs2			rs1		funct3		imm[4:1 11]		opcode		B-type
imm[31:12]										rd		opcode		U-type
imm[20 10:1 11 19:12]										rd		opcode		J-type

RV32I Base Instruction Set

0000000	shamt	rs1	001	rd	0010011	SLLI
---------	-------	-----	-----	----	---------	------

```
// sr_cpu.vh
`define RVOP_SLLI    7'b0010011
`define RVF3_SLLI   3'b001
`define RVF7_SLLI   7'b0000000
```


SLLI: Модификация устройства управления 2

```
// sr_cpu.v
always @ (*) begin
    branch      = 1'b0;
    condZero    = 1'b0;
    wdSrc       = 1'b0;
    ...
    casez( {cmdF7, cmdF3, cmdOp} )
        ...
        { `RVF7_SLLI, `RVF3_SLLI, `RVOP_SLLI } :
            begin
                regWrite = 1'b1; aluSrc = 1'b1; aluControl = `ALU_SLLI;
            end
        ...
    endcase
end
```

Проверка в симуляторе

в каталоге `program/03_slli`

```
make build
make modelsim
```

```
3 pc = 00000000 instr = 00100513 v0 = 1 addi $10, $0, 0x00000001
4 pc = 00000000 instr = 00100513 v0 = 1 addi $10, $0, 0x00000001
5 pc = 00000004 instr = 00151513 v0 = 1 new/unknown
6 pc = 00000008 instr = fe050ce3 v0 = 2 beq $10, $0, 0xffffffff8 (-8)
7 pc = 0000000c instr = fe000ce3 v0 = 2 beq $0, $0, 0xffffffff8 (-8)
8 pc = 00000004 instr = 00151513 v0 = 2 new/unknown
9 pc = 00000008 instr = fe050ce3 v0 = 4 beq $10, $0, 0xffffffff8 (-8)
10 pc = 0000000c instr = fe000ce3 v0 = 4 beq $0, $0, 0xffffffff8 (-8)
11 pc = 00000004 instr = 00151513 v0 = 4 new/unknown
12 pc = 00000008 instr = fe050ce3 v0 = 8 beq $10, $0, 0xffffffff8 (-8)
13 pc = 0000000c instr = fe000ce3 v0 = 8 beq $0, $0, 0xffffffff8 (-8)
14 pc = 00000004 instr = 00151513 v0 = 8 new/unknown
```

Синтез проекта

- копирование дампа памяти программ, в каталоге **program/03_slli**

```
make board
```

- сборка проекта в Quartus, в каталоге **board/de10_lite**

```
make create  
make build
```

- просмотр отчета о сборке, в каталоге **board/de10_lite**

```
make open
```

Анализ максимальной частоты

Quartus -> Processing -> Compilation Report

The screenshot shows the 'Compilation Report - de10_lite' window. The 'Table of Contents' on the left lists various reports, with 'Fmax Summary' selected under the 'Slow 1200mV 85C Model' folder. The main content area displays the 'Slow 1200mV 85C Model Fmax Summary' table.

	Fmax	Restricted Fmax	Clock Name	
1	74.45 MHz	74.45 MHz	clk	
2	392.31 MHz	250.0 MHz	MAX1...1_50	limit due to minimum period

Проверка на отладочной плате

- выполнить прошивку платы, в каталоге **board/de10_lite**

```
make load
```

- включить тактирование
- настроить делитель частоты
- выбрать регистр результата a0 для вывода на индикаторы
- убедиться в работоспособности программы и процессора

ГОТОВО!