

# 32 bit five stage pipelined RISC V RV32I Core

**About:** This RISC – V Core executes 27 instructions from the RV32I Set. It is in behavioural description.

**Languages used:** System Verilog

## Instructions Implemented

Instruction implemented as of now ( **Highlighted**)

Instruction	Description
lb rd, imm(rs1)	load byte
lh rd, imm(rs1)	load half
lw rd, imm(rs1)	load word
lbu rd, imm(rs1)	load byte unsigned
lhu rd, imm(rs1)	load half unsigned
addi rd, rs1, imm	add immediate
slli rd, rs1, uimm	shift left logical immediate
slti rd, rs1, imm	set less than immediate
sltiu rd, rs1, imm	set less than imm. unsigned
xori rd, rs1, imm	xor immediate
srlr rd, rs1, uimm	shift right logical immediate
srai rd, rs1, uimm	shift right arithmetic imm.
ori rd, rs1, imm	or immediate
andi rd, rs1, imm	and immediate
auipc rd, upimm	add upper immediate to PC
sb rs2, imm(rs1)	store byte
sh rs2, imm(rs1)	store half
sw rs2, imm(rs1)	store word
add rd, rs1, rs2	add
sub rd, rs1, rs2	sub
sll rd, rs1, rs2	shift left logical
slt rd, rs1, rs2	set less than
sltu rd, rs1, rs2	set less than unsigned
xor rd, rs1, rs2	xor
srl rd, rs1, rs2	shift right logical
sra rd, rs1, rs2	shift right arithmetic
or rd, rs1, rs2	or
and rd, rs1, rs2	and
lui rd, upimm	load upper immediate
beq rs1, rs2, label	branch if =
bne rs1, rs2, label	branch if ≠
blt rs1, rs2, label	branch if <
bge rs1, rs2, label	branch if ≥
bltu rs1, rs2, label	branch if < unsigned
bgeu rs1, rs2, label	branch if ≥ unsigned
jalr rd, rs1, imm	jump and link register
jal rd, label	jump and link

# 32 bit five stage pipelined RISC V RV32I Core

## Steps to execute the Instructions:

1. First, we write the instructions in assembly using the above table and convert it to hex file.

Use this site: <https://riscvasm.lucasteske.dev/#>

### RISC-V Online Assembler

Type the assembly code below and click **Build**

```
1 .global _boot
2 .text
3
4 _boot:
5 main:  addi x3, x0, 10      # x3 = 10
6        addi x4, x0, 12      # x4 = 12
7        or x5, x3, x4        # x5 = (10 OR 12) = 14
8        and x6, x3, x4       # x5 = (10 AND 12) = 8
9        bne x5, x6, end      # should be taken
10       addi x4, x4, 3        # shouldn't be executed
11
12 around: slt x7, x5, x6      # x7 = (8 < 14) = 1
13        sub x8, x6, x5       # x8 = (8 - 14) = -6
```

**BUILD**

**Console**  
OK

**Hex Dump**  
00a00193  
00c00213  
0041e2b3  
0041f333  
00628463  
fedff06f

2. Convert the Hex dump into **byte - addressable format** using the python script. (Available in the folder)

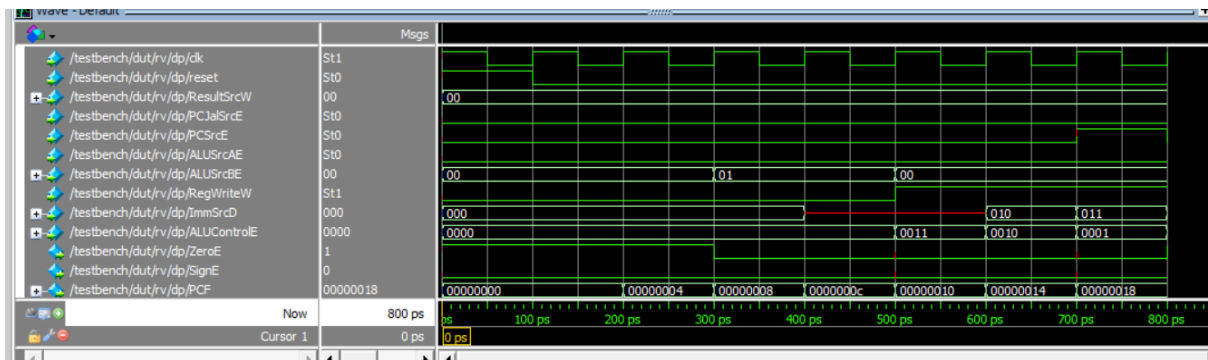
```
1 // Instruction: 00a00193
2 RAM[0] = 8'h93;
3 RAM[1] = 8'h01;
4 RAM[2] = 8'ha0;
5 RAM[3] = 8'h00;
6 // Instruction: 00c00213
7 RAM[4] = 8'h13;
8 RAM[5] = 8'h02;
9 RAM[6] = 8'hc0;
10 RAM[7] = 8'h00;
11 // Instruction: 0041e2b3
12 RAM[8] = 8'hb3;
13 RAM[9] = 8'he2;
14 RAM[10] = 8'h41;
15 RAM[11] = 8'h00;
16 // Instruction: 0041f333
17 RAM[12] = 8'h33;
```

## 32 bit five stage pipelined RISC V RV32I Core

- Copy the output into imem.sv – **instruction memory** file.

```
1  /*
2   Name: Instruction Memory
3  */
4
5
6  module imem(input logic [31:0] a, output logic [31:0] rd);
7
8  logic [7:0] RAM[128:0]; // 128 x 8 = byte addressable memory with 128 locations
9
10 assign rd = {RAM[a+3], RAM[a+2], RAM[a+1], RAM[a+0]};
11
12 // follow little-endian: LSB corresponds to lowest order memory address
13 initial
14 begin
15     // Instruction: 00a00193
16     RAM[0] = 8'h93;
17     RAM[1] = 8'h01;
18     RAM[2] = 8'ha0;
19     RAM[3] = 8'h00;
20     // Instruction: 00c00213
21     RAM[4] = 8'h13;
22     RAM[5] = 8'h02;
23     RAM[6] = 8'h0c;
24     RAM[7] = 8'h00;
25     // Instruction: 0041e2b3
26     RAM[8] = 8'hb3;
27     RAM[9] = 8'he2;
28     RAM[10] = 8'h41;
29     RAM[11] = 8'h00;
30     // Instruction: 0041f333
31     RAM[12] = 8'h33;
32     RAM[13] = 8'hf3;
```

- Set testbench as top module and start analysis and elaboration. Use ModelSim to verify the waveforms.



# 32 bit five stage pipelined RISC V RV32I Core

5. The value of ALU is shown in the transcript every clock cycle. Cross check this with the assembly file.

```

Transcript
add wave -position end $sim/$objectname/dut/rv/dp/
# ** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
#       File in use by: Navin Hostname: NAVIN-LAPTOP ProcessID: 12740
#       Attempting to use alternate WLF file "./wlftdrzfr8".
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
#       Using alternate file: ./wlftdrzfr8
force -freeze sim:/testbench/dut/rv/dp/clk 1 0, 0 {50 ps} -r 100
VSIM9> run
# Simulation starts!
run
# Value of ALU =      0
run
run
run
# Value of ALU =     10
run
# Value of ALU =     12
run
# Value of ALU =     14
VSIM 10> run
# Value of ALU =      8
VSIM 10>

```

Now: 800 ps Delta: 4

dut