****

**Faculty of Engineering**

**Computer Engineering Department**

Computer Architecture Lab Project” MIPS processor Simulation”

**Submitted to:**

Dr. Khaled Badran

Faculty of Engineering, Computer Engineering Department

Computer Architecture (21COMP05I)

**Submitted by:**

Name: Mahmoud Ashraf Mahmoud Ahmed, ID: 204858, Group: A2

Name: Abdelrahman Mostafa, ID: 206395, Group: A2

Name: Pierre Tamer, ID: 198987, Group: A2

Name: Tomas Ehab, ID: 181954, Group: A2

April 2022, Cairo

# I. Introduction

The MIPS processor, developed by Stanford University researchers in 1984, is a RISC (Reduced Instruction Set Computer) processor.

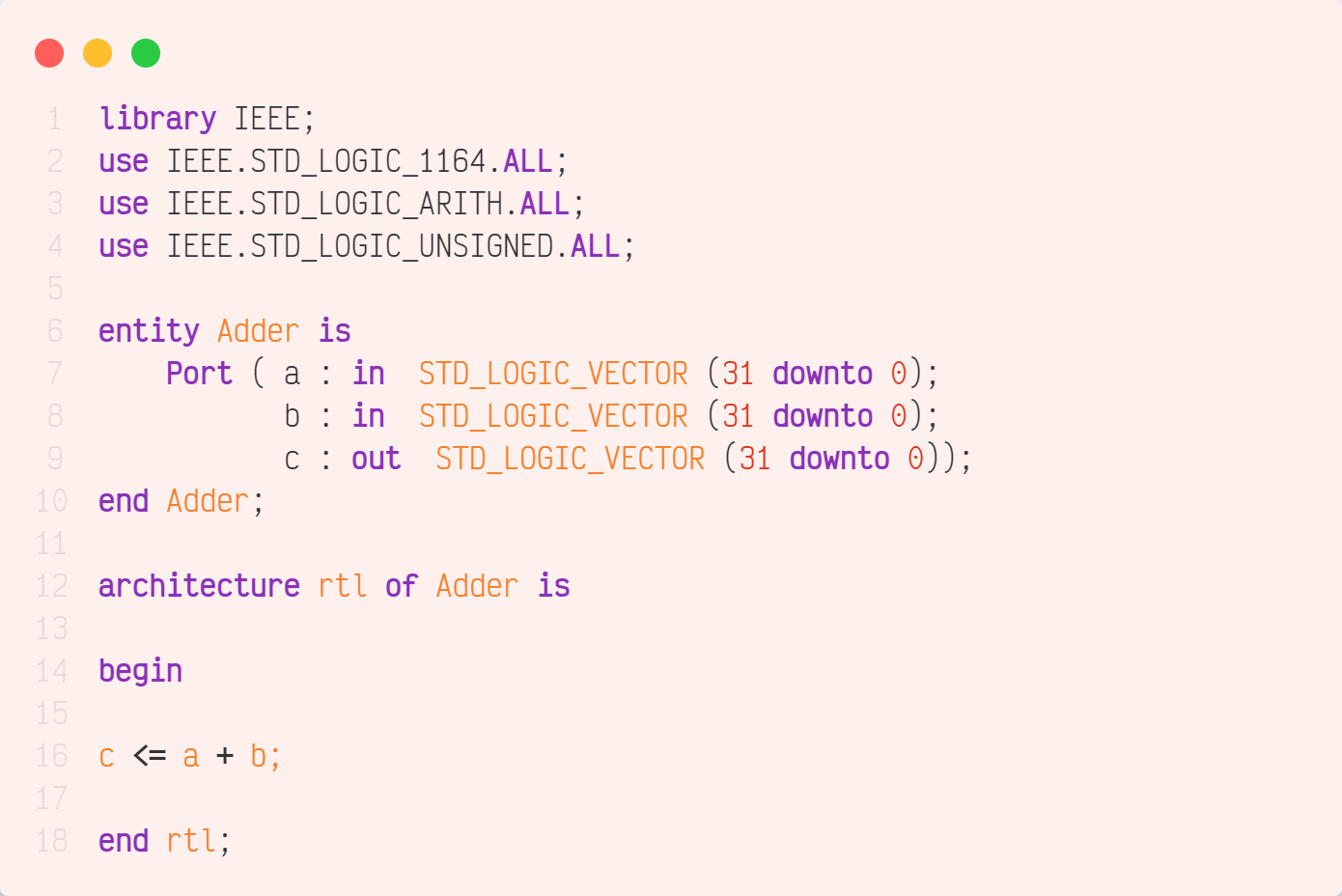
RISC processors usually support fewer and simpler instructions than CISC (Complex Instruction Set Computer) counterparts (such as Intel Pentium processors).

The concept is, however, that because of its simpler design, a RISC processor can be made much faster than a CISC processor.

Nowadays, it is widely accepted that RISC processors are more efficient than CISC processors; in fact, the only popular CISC processor still in use (Intel Pentium) internally translates CISC instructions into RISC instructions before they are executed.

# II. VHDL Coding & Simulation of Single Blocks

## Adder Block



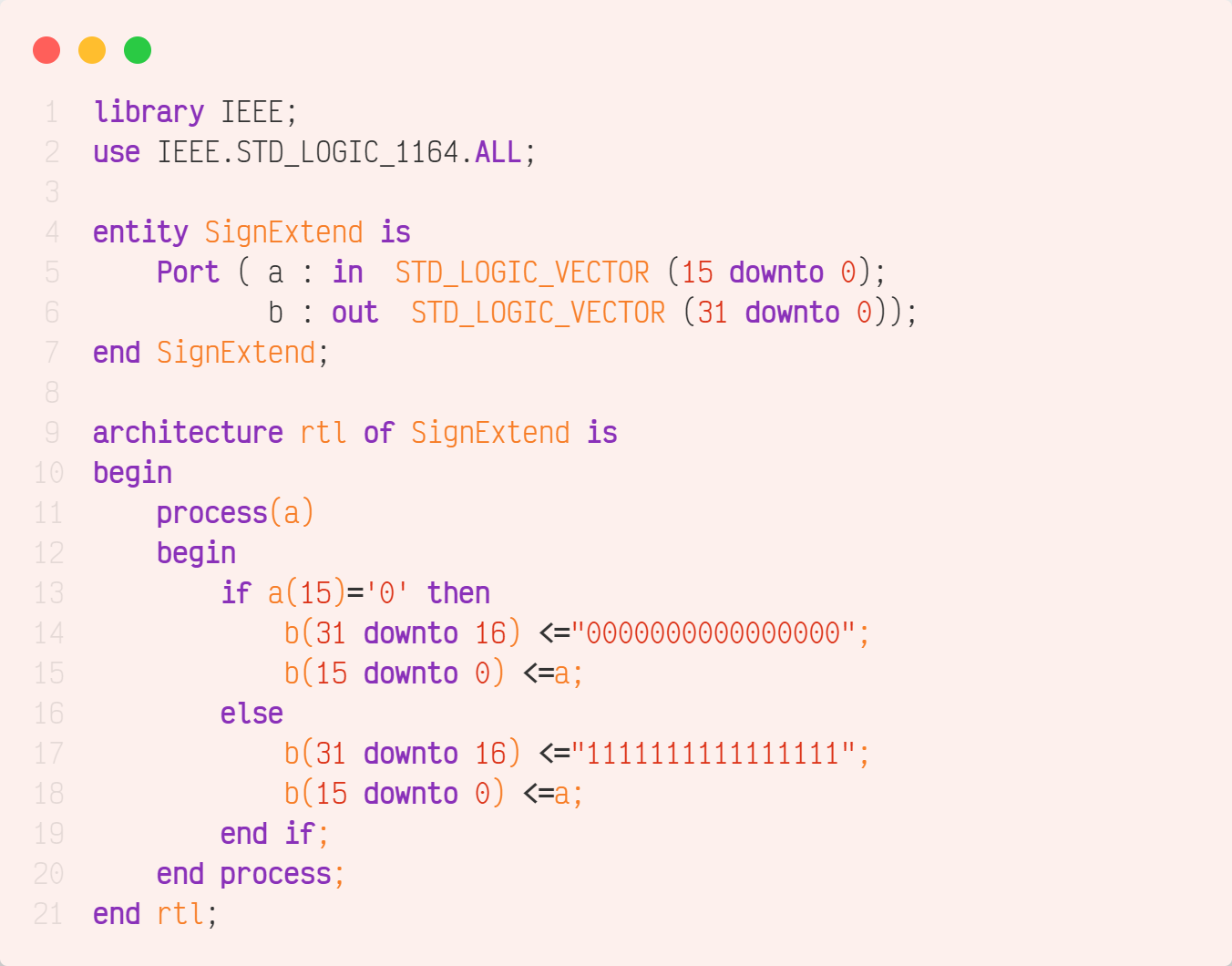


force -freeze sim:/adder/a 16#00000001 0

force -freeze sim:/adder/b 16#00000002 0

run

## Sign Extend







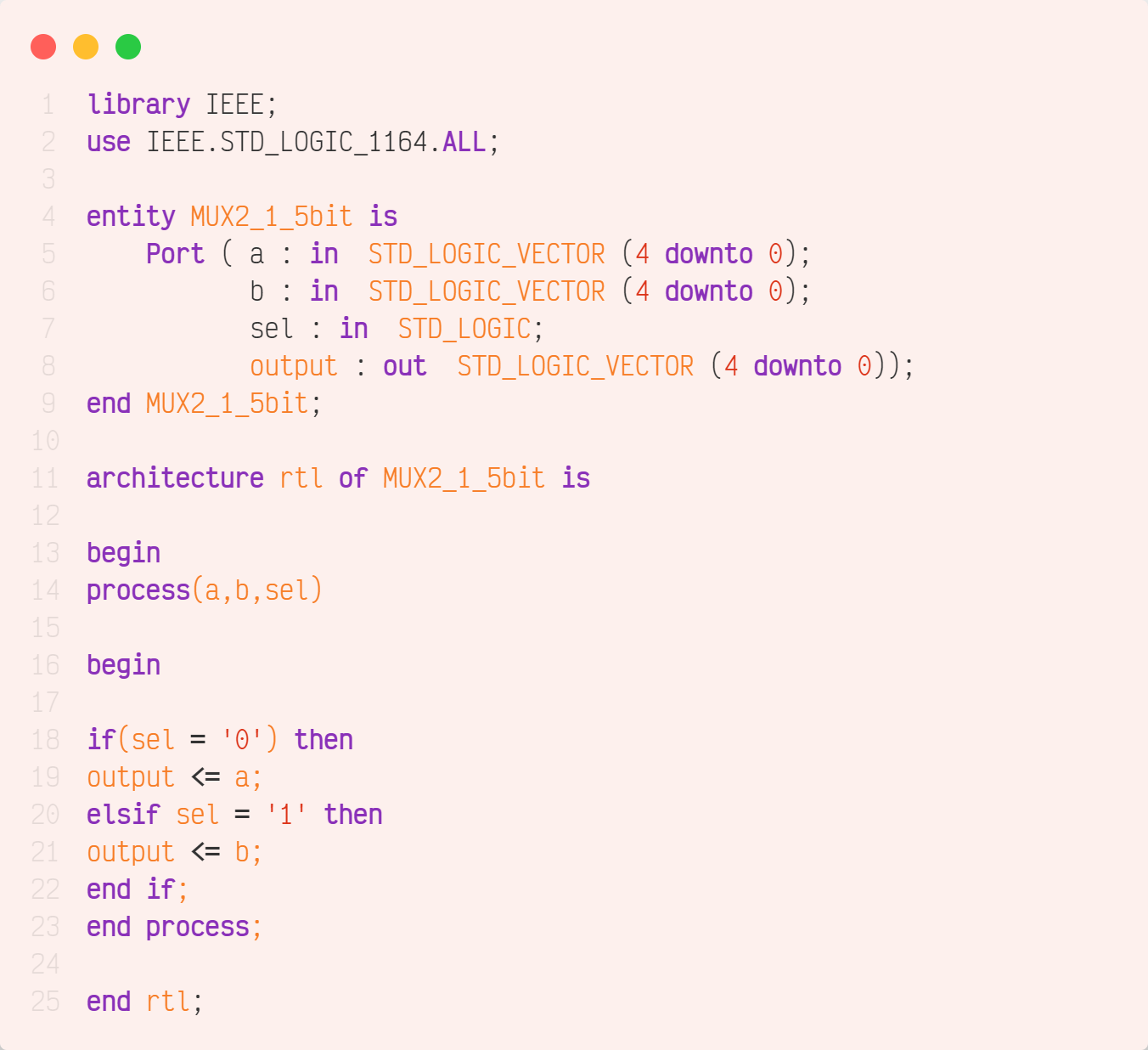
force -freeze sim:/signextend/a 16#0001 0

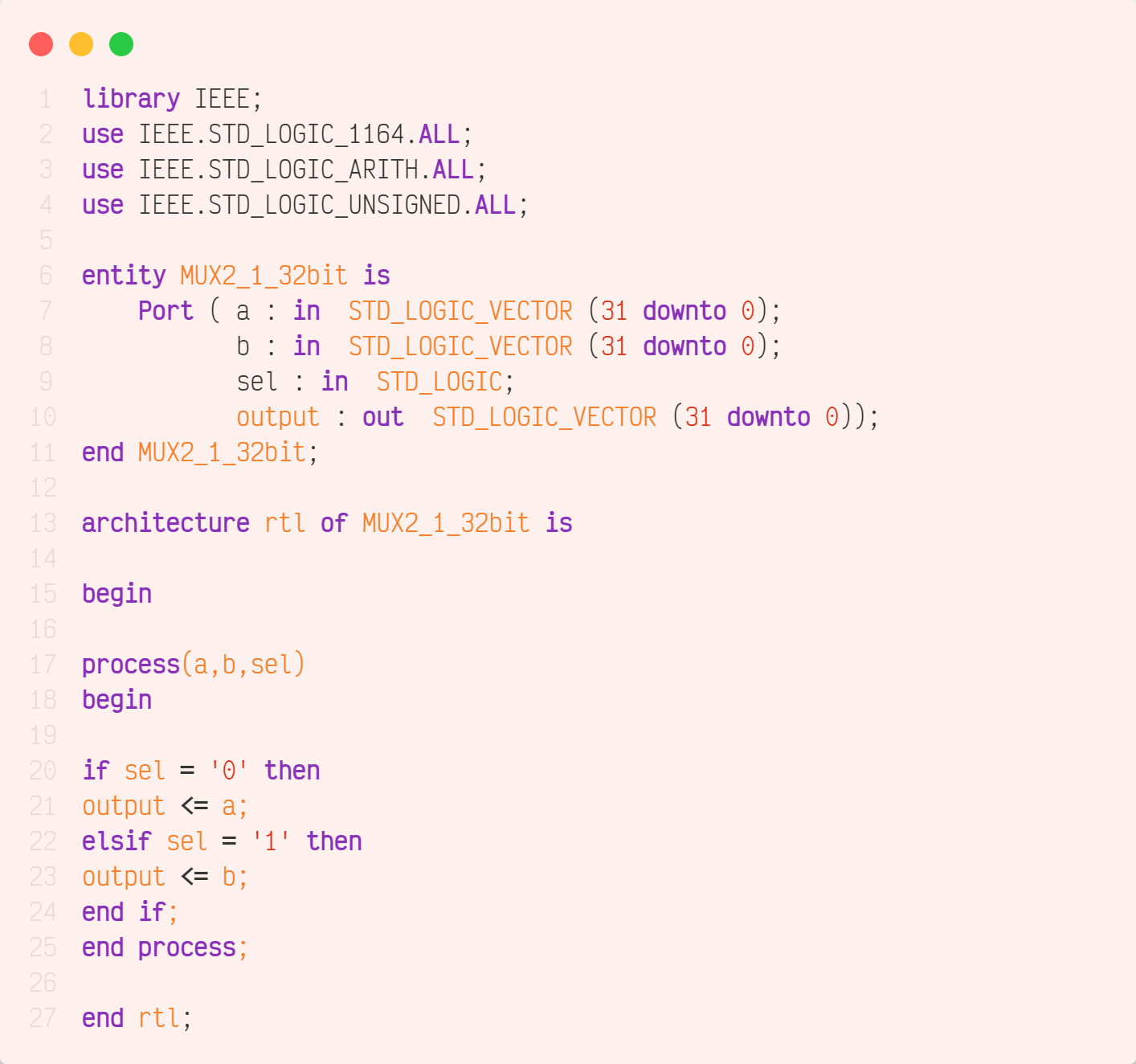
run

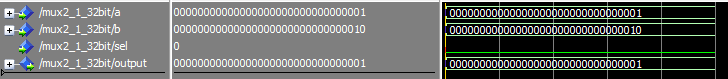
force -freeze sim:/signextend/a 1000000000000001 0

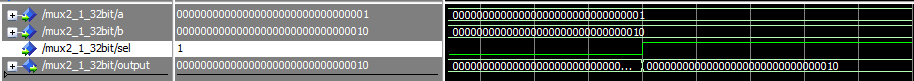
run

## 2x1 MUX 32 bits & 5 bits









force -freeze sim:/mux2\_1\_32bit/a 16#00000001 0

force -freeze sim:/mux2\_1\_32bit/b 16#00000002 0

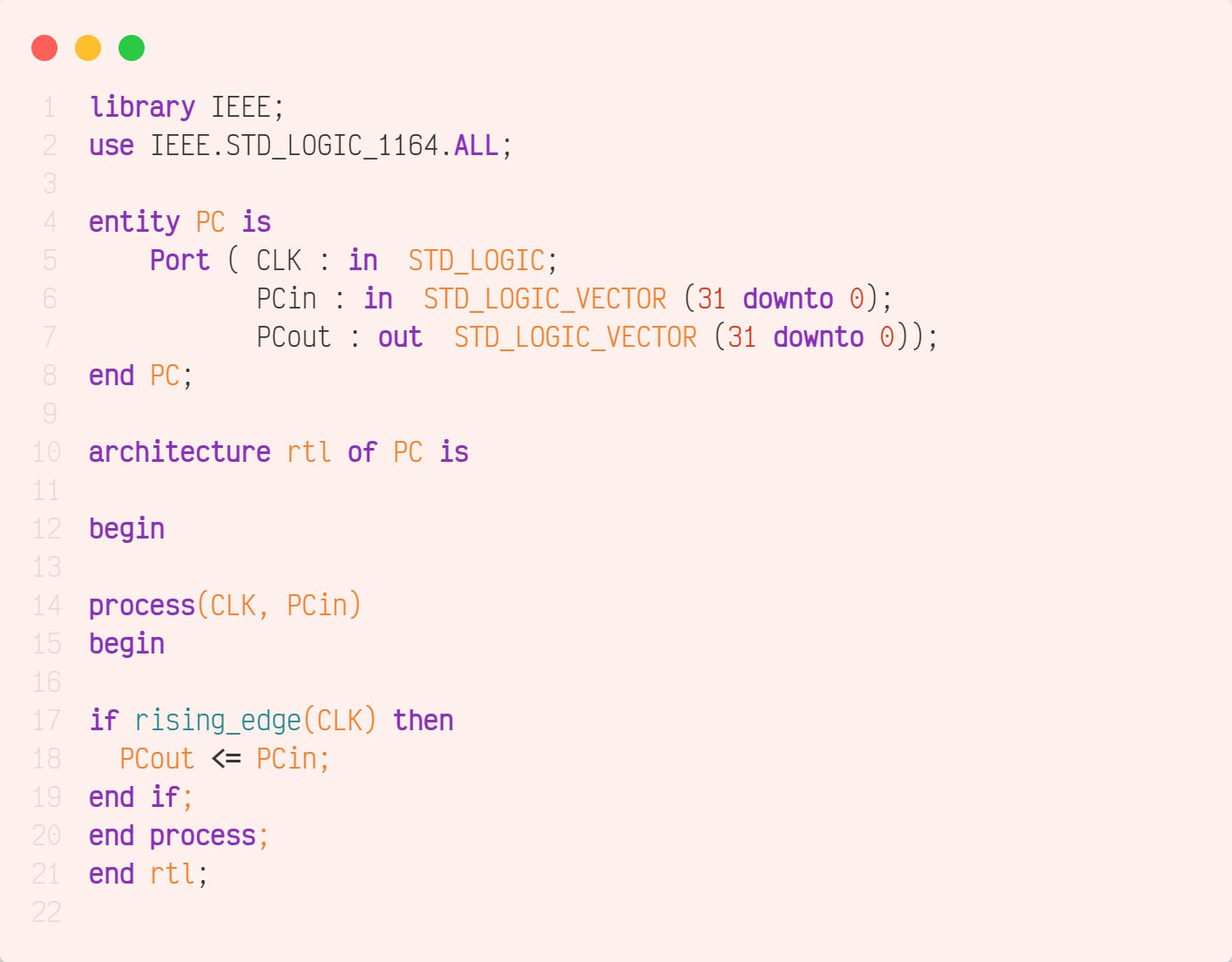
force -freeze sim:/mux2\_1\_32bit/sel 0 0

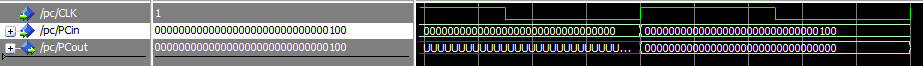
run

force -freeze sim:/mux2\_1\_32bit/sel 1 0

run

## Program Counter





force -freeze sim:/pc/CLK 1 0, 0 {50 ps} -r 100

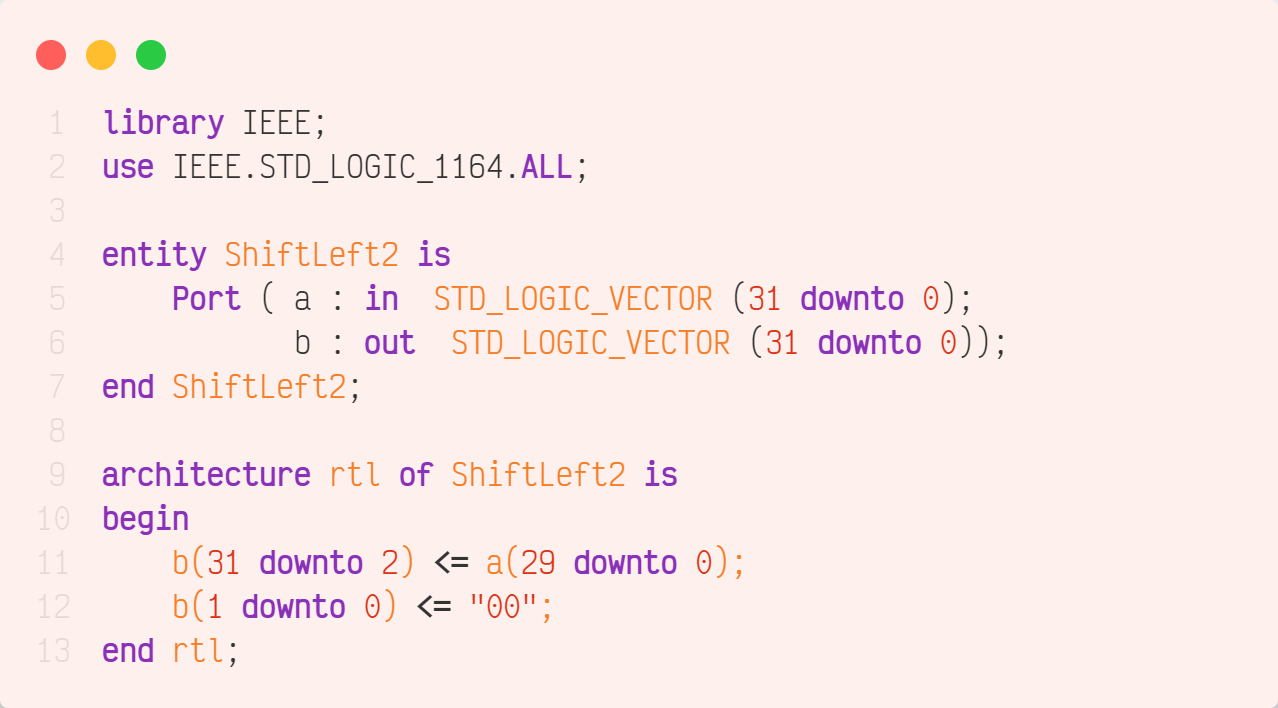
force -freeze sim:/pc/PCin 16#00000000 0

run

force -freeze sim:/pc/PCin 00000000000000000000000000000100 0

run

## Shift Left 2

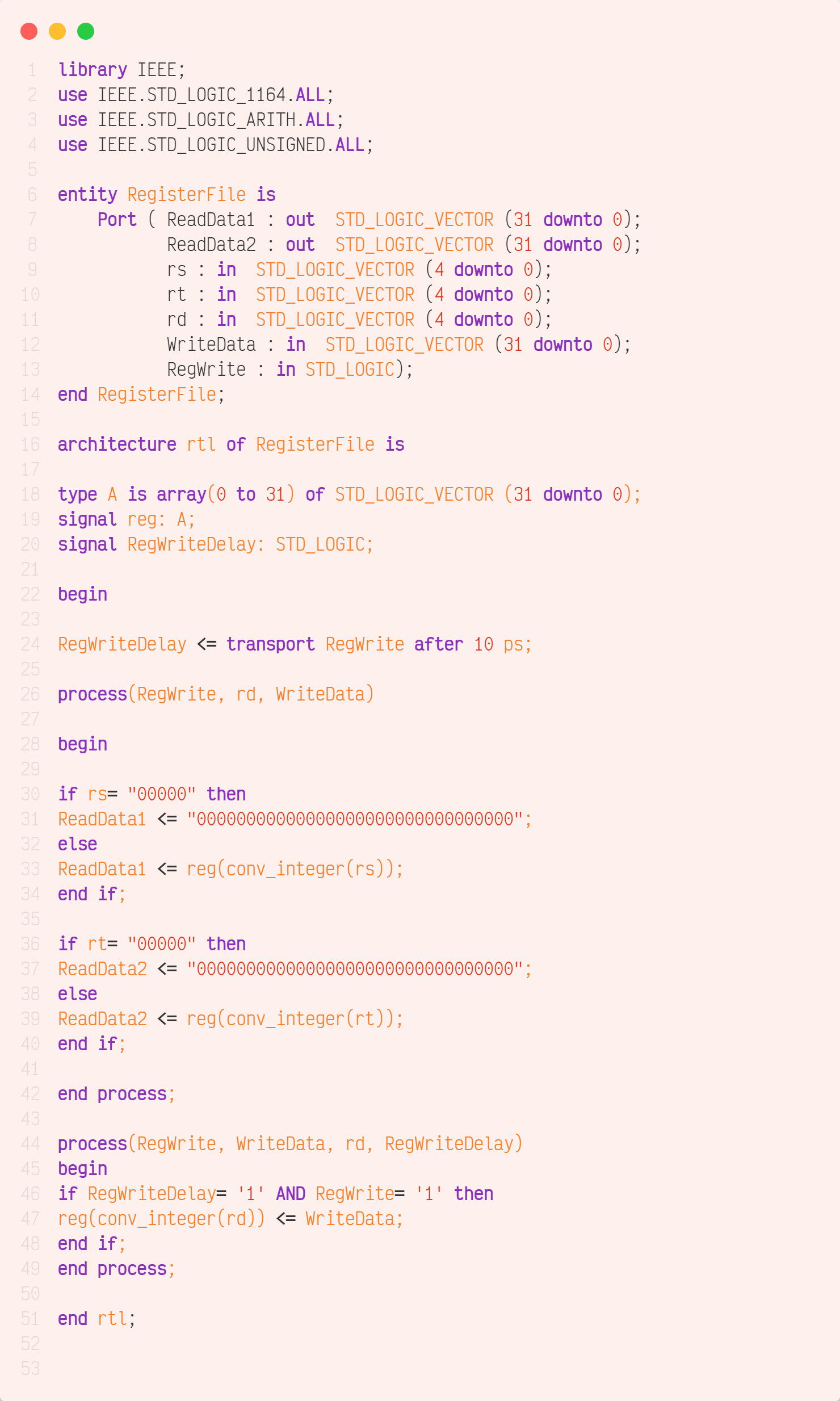




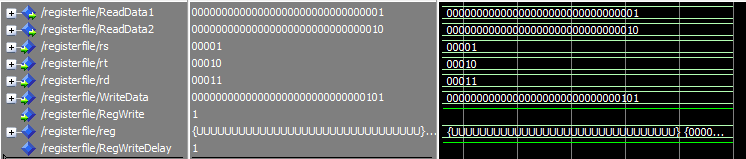
force -freeze sim:/shiftleft2/a 16#00000001 0

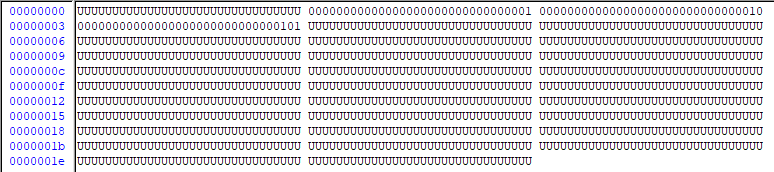
run

## Register File









mem load -filltype value -filldata 16#00000001 -fillradix symbolic /registerfile/reg(1)

mem load -filltype value -filldata 16#00000002 -fillradix symbolic /registerfile/reg(2)

force -freeze sim:/registerfile/rs 00001 0

force -freeze sim:/registerfile/rt 00010 0

force -freeze sim:/registerfile/rd 00011 0

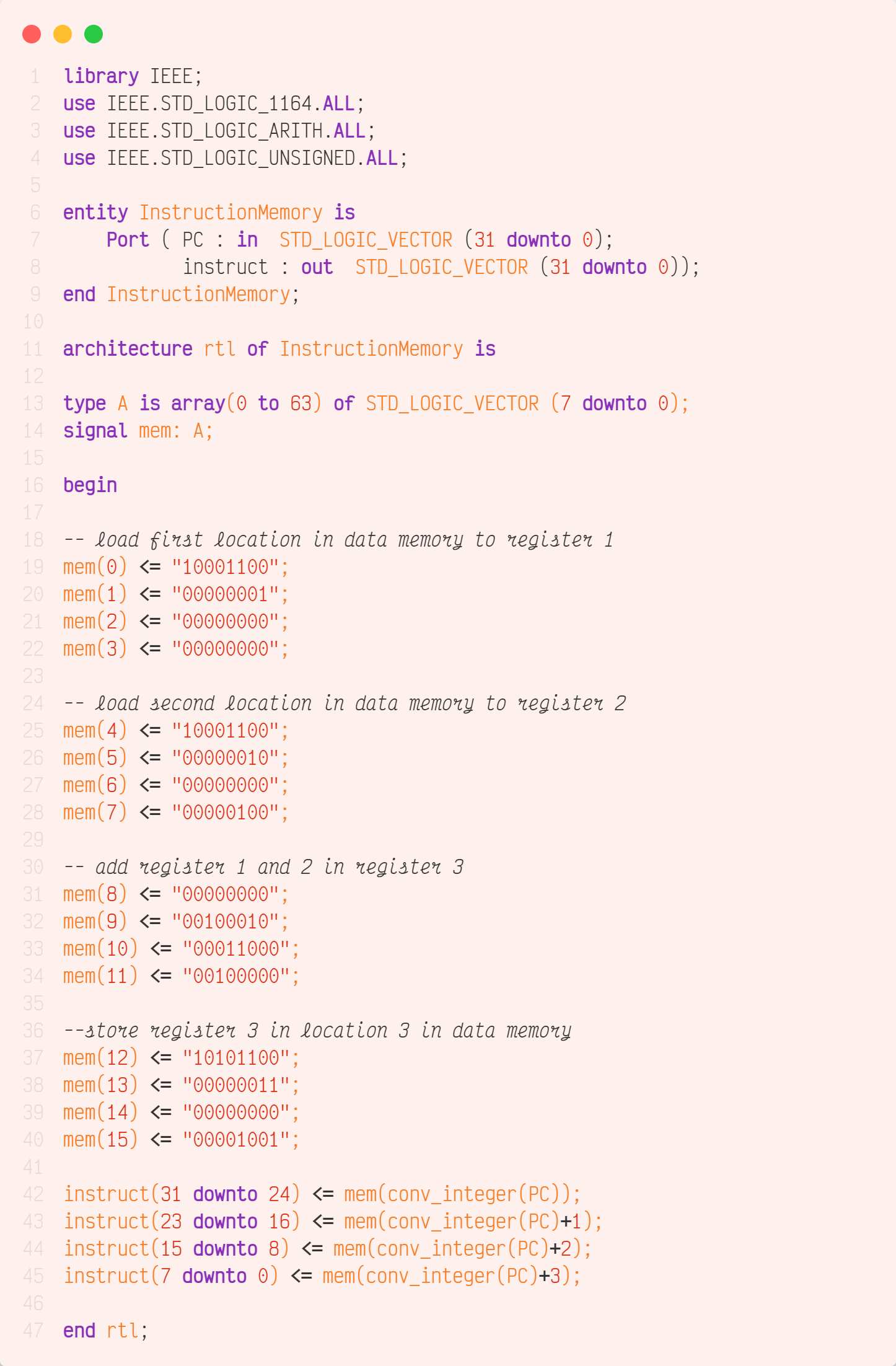
force -freeze sim:/registerfile/WriteData 16#00000005 0

force -freeze sim:/registerfile/RegWrite 1 0

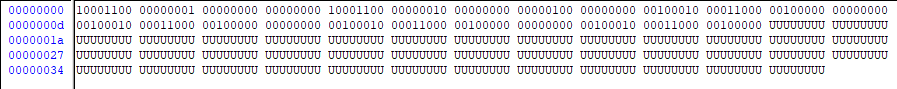
force -freeze sim:/registerfile/RegWriteDelay 1 0

run

## Instruction Memory



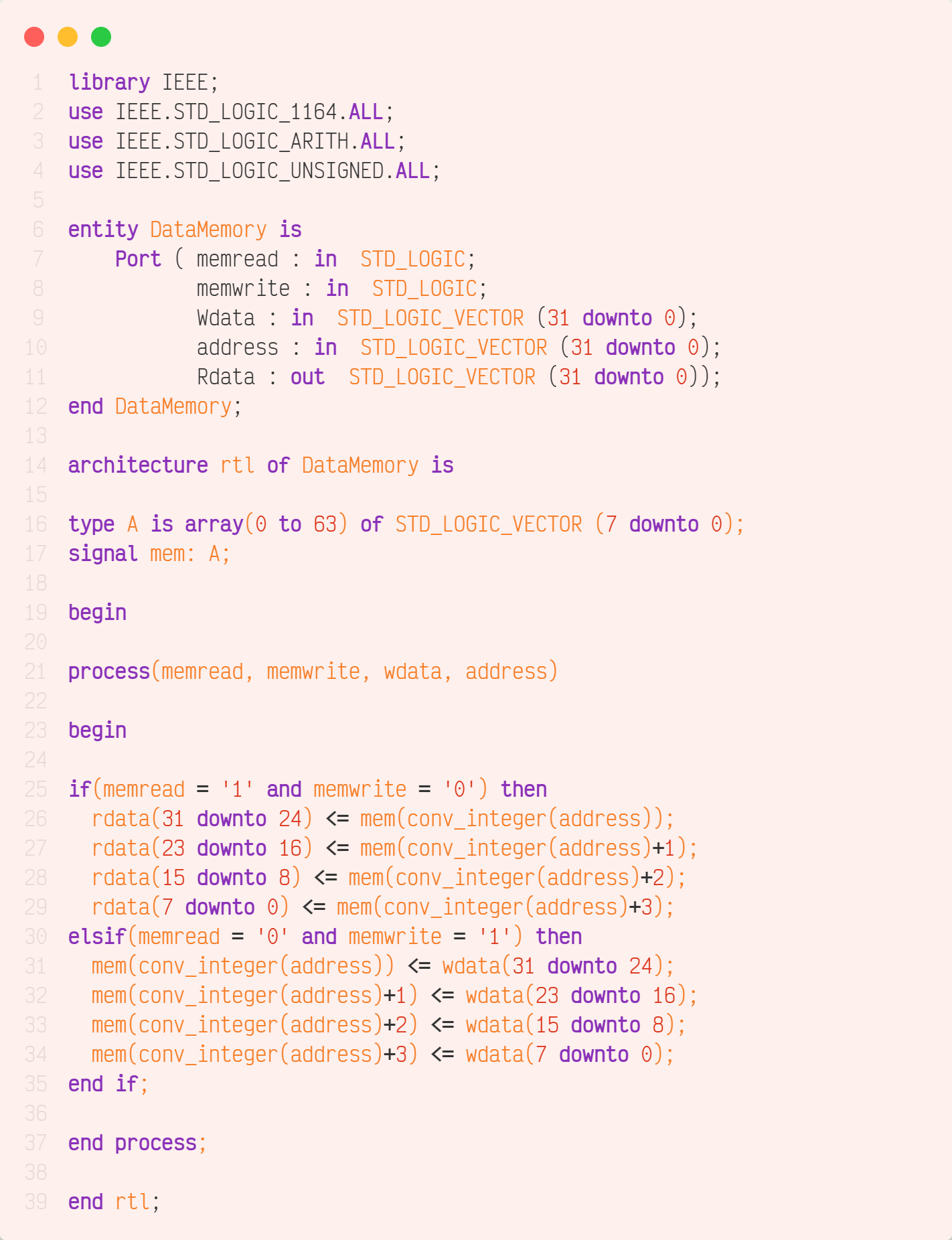


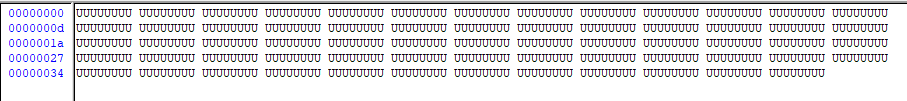


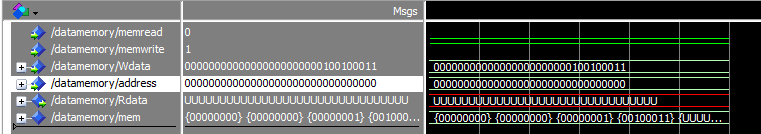
force -freeze sim:/instructionmemory/PC 16#00000000 0

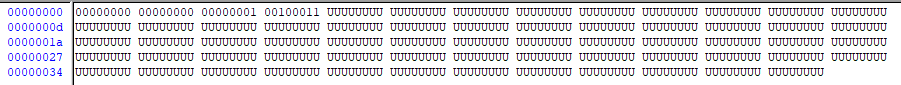
run

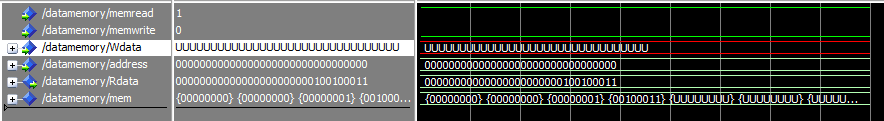
## Data Memory











force -freeze sim:/datamemory/memwrite 1 0

force -freeze sim:/datamemory/memread 0 0

force -freeze sim:/datamemory/Wdata 16#000000123 0

force -freeze sim:/datamemory/address 16#00000000 0

run

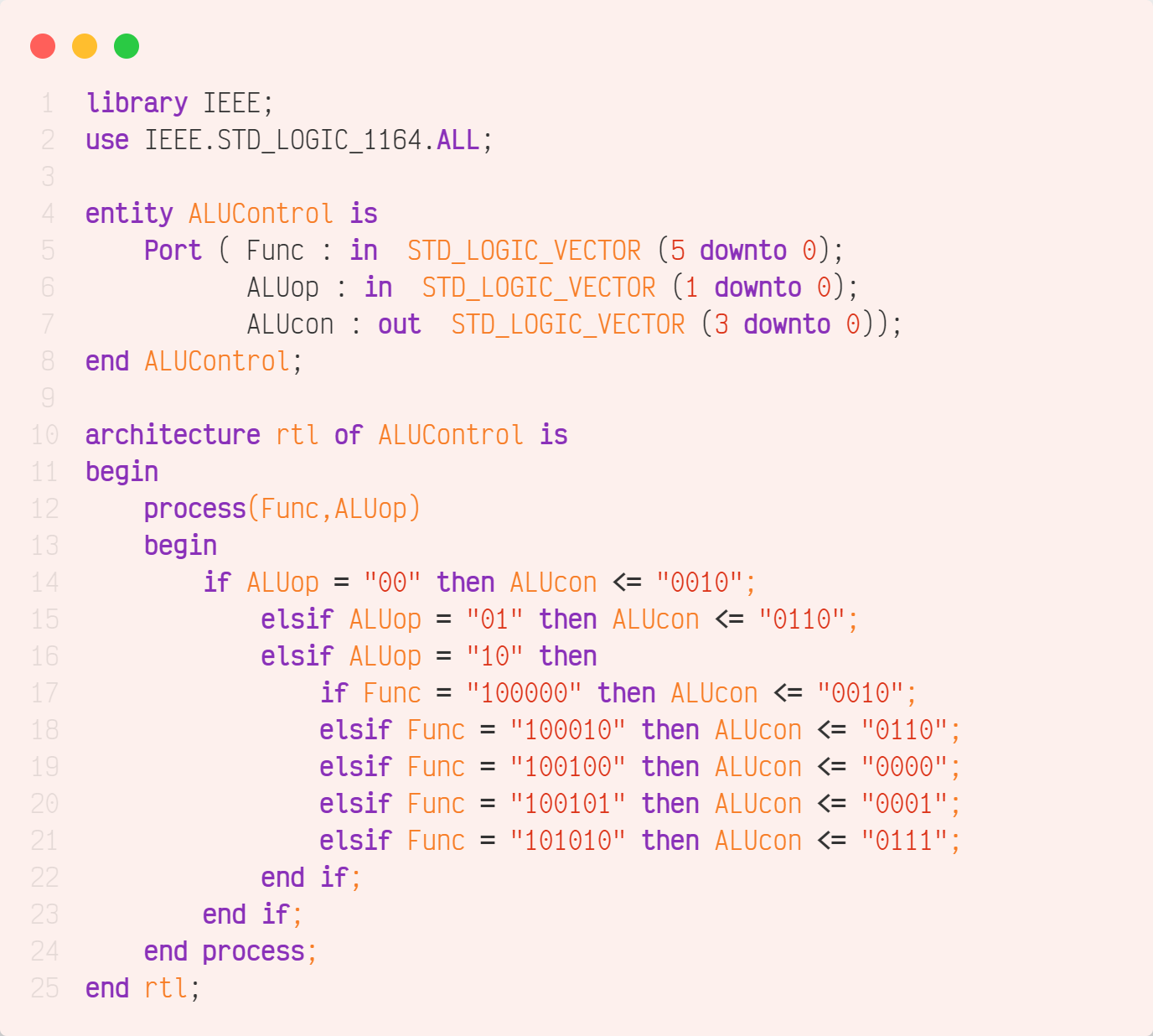
force -freeze sim:/datamemory/memread 1 0

force -freeze sim:/datamemory/memwrite 0 0

noforce sim:/datamemory/Wdata

run

## ALU Control



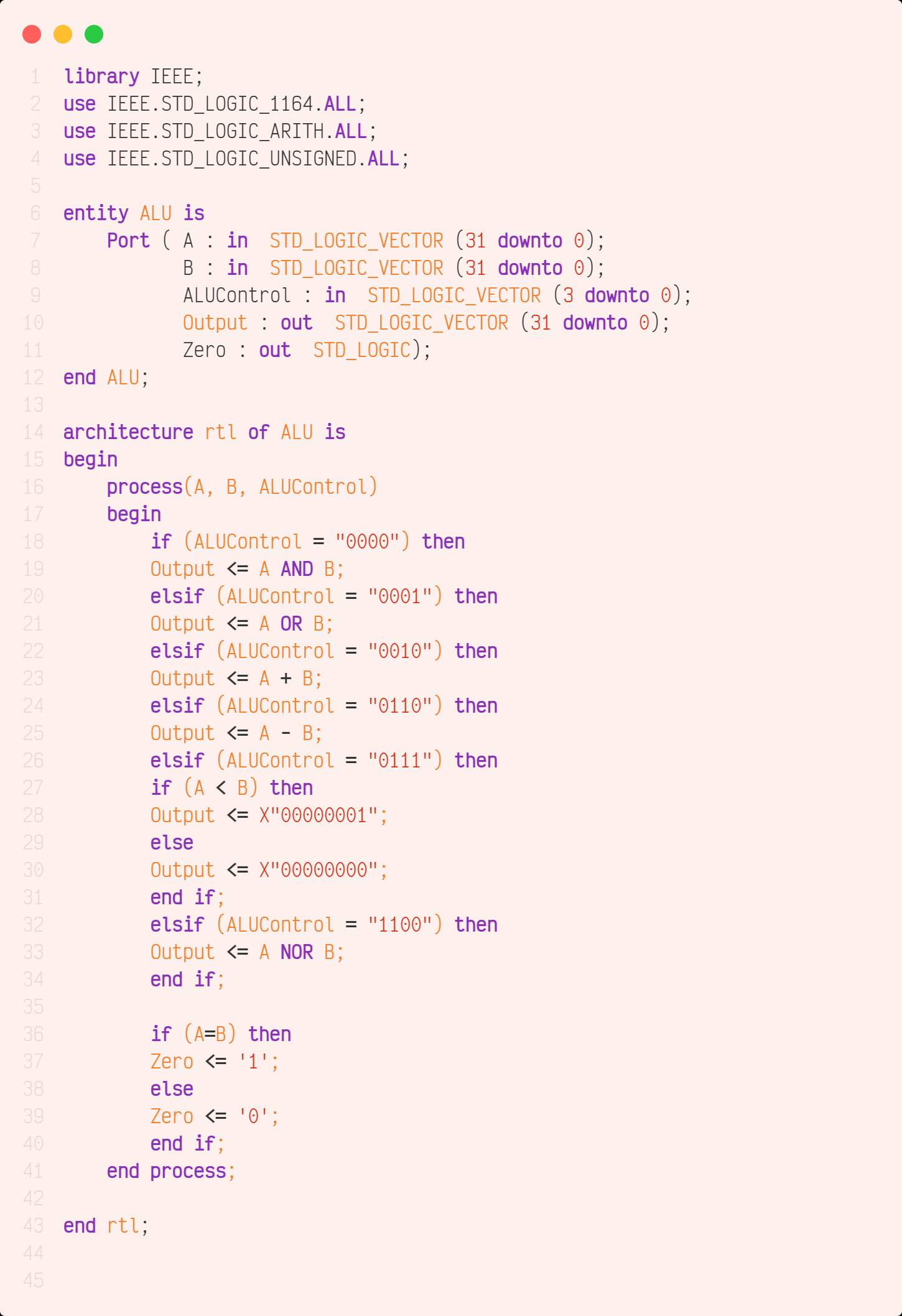


force -freeze sim:/alucontrol/Func 100000 0

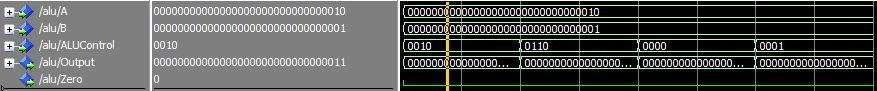
force -freeze sim:/alucontrol/ALUop 10 0

run

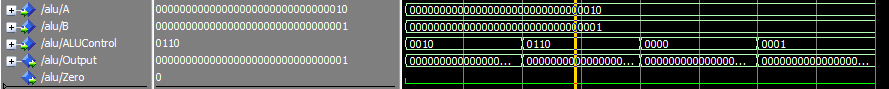
## ALU



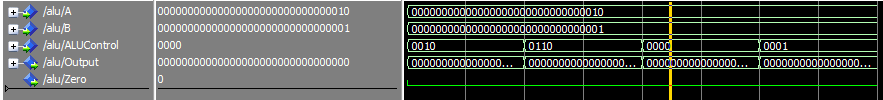
Add



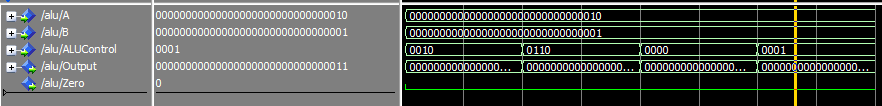
Subtract



AND



OR



force -freeze sim:/alu/A 16#00000002 0

force -freeze sim:/alu/B 16#00000001 0

force -freeze sim:/alu/ALUControl 0010 0

run

force -freeze sim:/alu/ALUControl 0110 0

run

force -freeze sim:/alu/ALUControl 0000 0

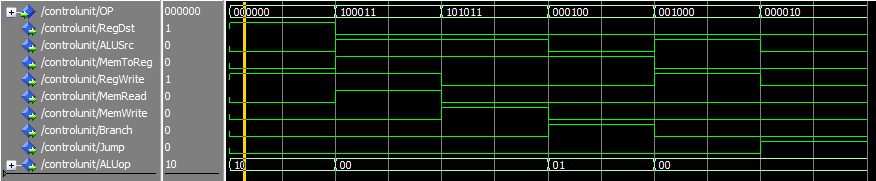
run

force -freeze sim:/alu/ALUControl 0001 0

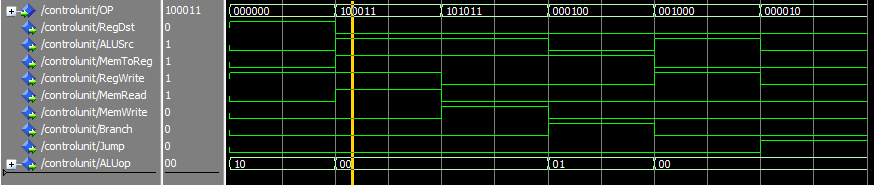
run

## Control Unit

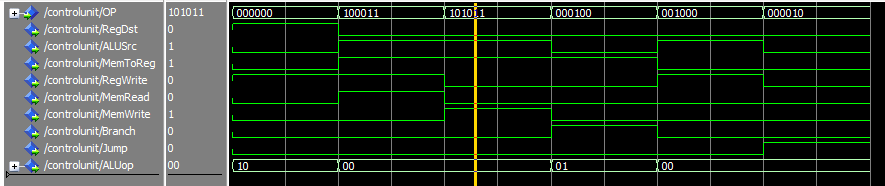
R-Type



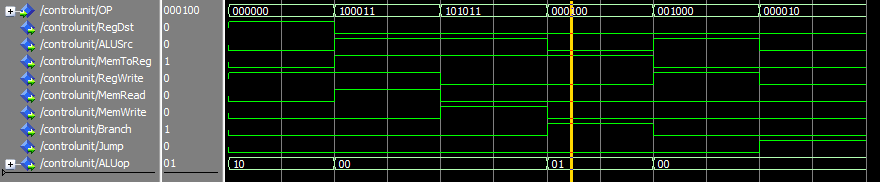
Load Word



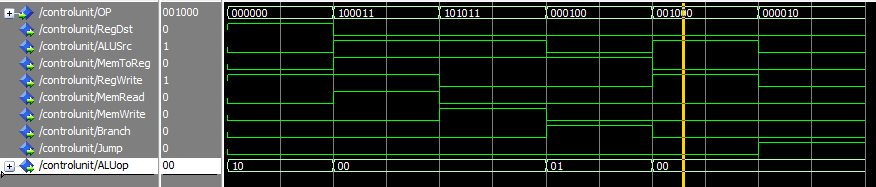
Store Word



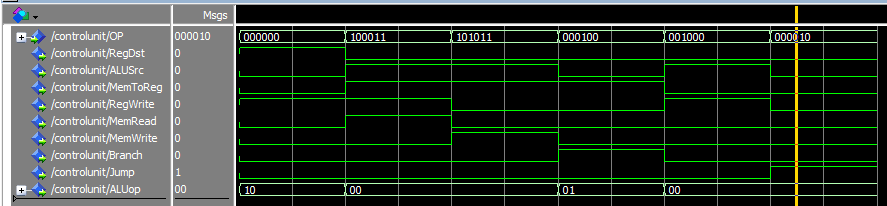
Branch on Equal



Add Immediate



Jump



force -freeze sim:/controlunit/OP 000000 0

run

force -freeze sim:/controlunit/OP 100011 0

run

force -freeze sim:/controlunit/OP 101011 0

run

force -freeze sim:/controlunit/OP 000100 0

run

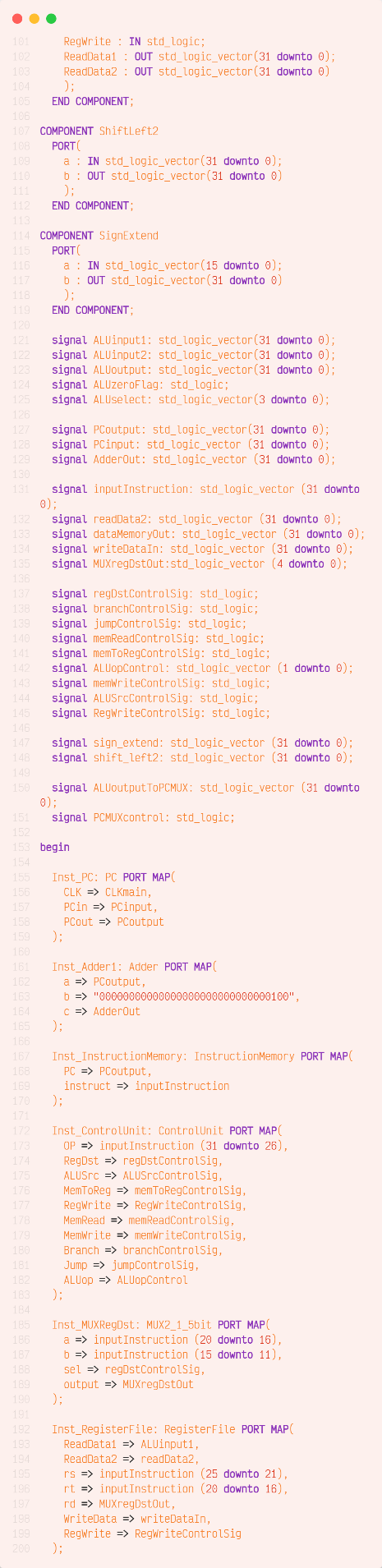
force -freeze sim:/controlunit/OP 001000 0

run

force -freeze sim:/controlunit/OP 000010 0

run

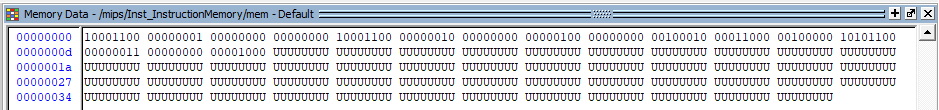
# III. VHDL Coding & Simulation of MIPS CPU

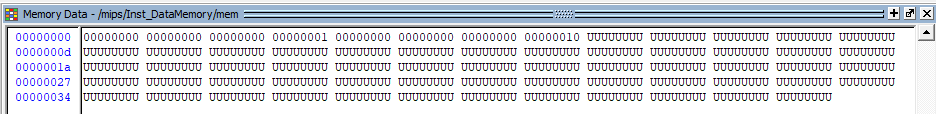


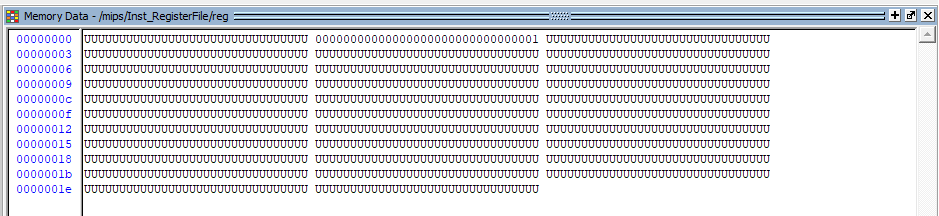
Test is running four instructions

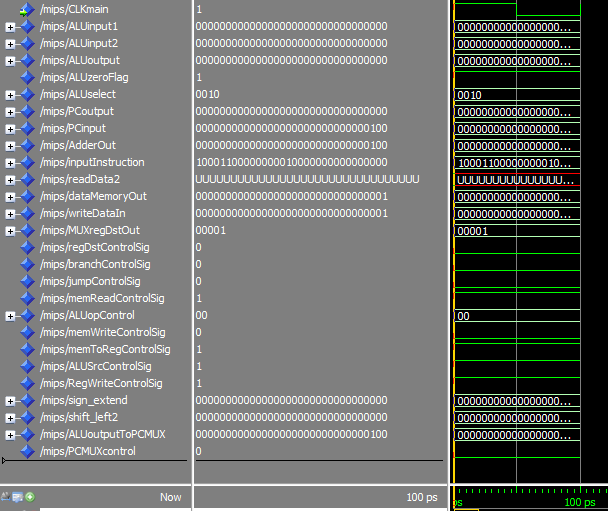
1. Load the first 32bit word from Data memory to Register 1 (value = 1)
2. Load the second 32bit word from Data memory to Register 2 (value = 2)
3. Adding Register 1 and 2 to Register 3 (value = 1 + 2 = 3)
4. Store Register 3 value in the third 32bit location in data memory

After first run

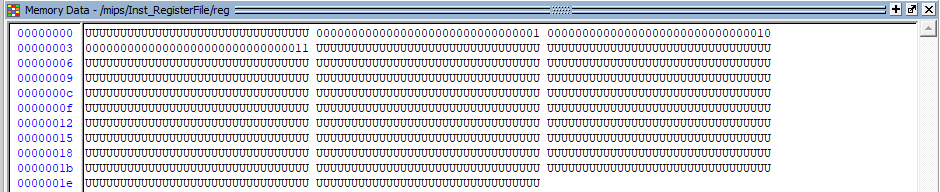


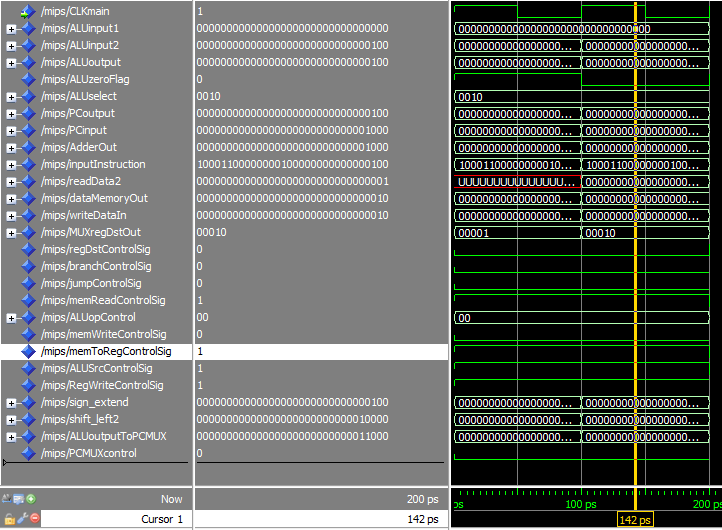




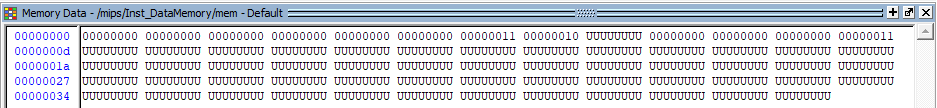


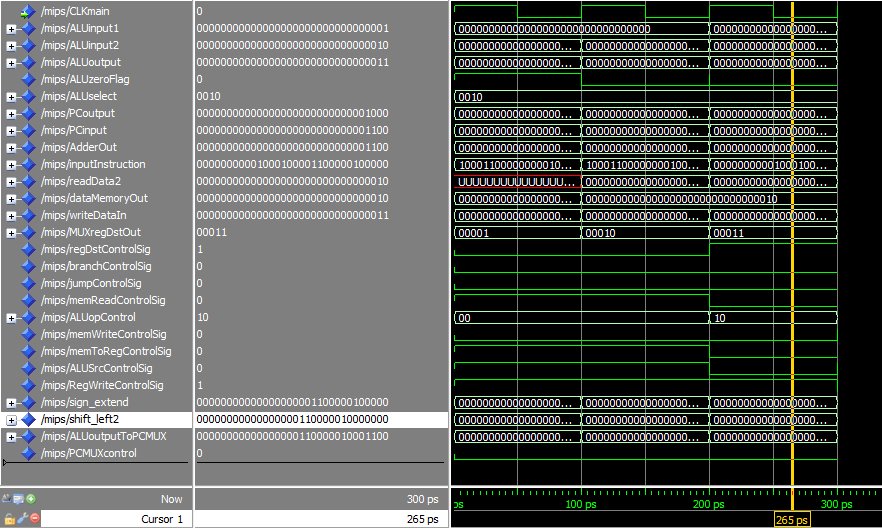
After Second Run





After Third Run





force -freeze sim:/mips/CLKmain 1 0, 0 {50 ps} -r 100

force -freeze sim:/mips/PCoutput 16#00000000 0 -cancel 100ps

mem load -filltype value -filldata 16#00 -fillradix symbolic /mips/Inst\_DataMemory/mem(0)

mem load -filltype value -filldata 16#00 -fillradix symbolic /mips/Inst\_DataMemory/mem(1)

mem load -filltype value -filldata 16#00 -fillradix symbolic /mips/Inst\_DataMemory/mem(2)

mem load -filltype value -filldata 16#01 -fillradix symbolic /mips/Inst\_DataMemory/mem(3)

mem load -filltype value -filldata 16#00 -fillradix symbolic /mips/Inst\_DataMemory/mem(4)

mem load -filltype value -filldata 16#00 -fillradix symbolic /mips/Inst\_DataMemory/mem(5)

mem load -filltype value -filldata 16#00 -fillradix symbolic /mips/Inst\_DataMemory/mem(6)

mem load -filltype value -filldata 16#02 -fillradix symbolic /mips/Inst\_DataMemory/mem(7)

run

run

run