Validating the TinyRISC Models

Eliseu M. Chaves Filho
e-mail: eliseu@lam.ufrj.br

January 25, 1999

1 Introduction

This document describes the tests performed to validate the structural and gate-level models of the TinyRISC processor. Two sets of tests were performed. The first set contains a small assembly program for each instruction category in TinyRISC instruction set. The second set contains three small C programs: heap sort, matrix multiplication and binary search.

2 The Testbench

The testbench used to validate the TinyRISC models is constituted by the following components:

- ENTITY clock_generator: a clock/reset driver;
- ENTITY ins_memory: the instruction memory;
- ENTITY data_memory: the data memory;
- ENTITY trisc: the TinyRISC processor.

A clock cycle time of 10 ns is used in all the tests. The following parameters are applicable to the instruction and data memories:

- iTacc: the instruction memory access time;
- dTacc: the data memory access time;
- iTack: the delay for the assertion of INSACK. If the value of iTack is greater than the clock cycle time, one or more wait states are inserted in the fetch operation;
- dTack: the delay for the assertion of DACK. If the value of dTack is greater than the clock cycle time, the execute stage stalls the pipeline for one or more cycles on a data or i/o access operation;
3 First Set: Instruction Sequences

For each instruction sequence here presented, it were observed the results written into the data register file. A detailed description of the TinyRISC instructions can be found in the document *The TinyRISC Instruction Set Architecture, Version 2*.

To run one of these tests, copy the file indicated to the file `input.exe` and run Synopsys' Debugger (`vhdlx`).

3.1 Arithmetic Instructions

File: `inputs/testarit.exe`

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Register 1</th>
<th>Register 2</th>
<th>Register 3</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>3b015555</td>
<td>LDUI R1,0xaaa</td>
<td>R1 = 0xaaa0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000001</td>
<td>0311aaa</td>
<td>ORI R1,R1,0xaaa</td>
<td>R1 = 0xaaaaaaa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000002</td>
<td>3b02aaa</td>
<td>LDUI R2,0x5555</td>
<td>R2 = 0x55550000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000003</td>
<td>03225555</td>
<td>ORI R2,R2,0x5555</td>
<td>R2 = 0x55555555</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000004</td>
<td>08132000</td>
<td>ADD R3,R1,R2</td>
<td>R3 = 0xffffffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000005</td>
<td>0913aaaa</td>
<td>ADDI R3,R1,0x5555</td>
<td>R3 = 0xaaaaaffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000006</td>
<td>0a132000</td>
<td>SUB R3,R1,R2</td>
<td>R3 = 0x55555555</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000007</td>
<td>0b132222</td>
<td>SUBI R3,R1,0x5555</td>
<td>R3 = 0xaaaaa555</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000008</td>
<td>0a231000</td>
<td>SUB R3,R2,R1</td>
<td>R3 = 0xaaaaaaab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000009</td>
<td>0b235555</td>
<td>SUBI R3,R2,0xaaa</td>
<td>R3 = 0xaaaaa0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2 Logical Instructions

File: `inputs/testlog.exe`

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Register 1</th>
<th>Register 2</th>
<th>Register 3</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>3b015555</td>
<td>LDUI R1,0x5555</td>
<td>R1 = 0x55550000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000001</td>
<td>03115555</td>
<td>ORI R1,R1,0x5555</td>
<td>R1 = 0x55555555</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000002</td>
<td>3b02aaa</td>
<td>LDUI R2,0xaaa</td>
<td>R2 = 0xaaa0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000003</td>
<td>0322aaa</td>
<td>ORI R2,R2,0xaaa</td>
<td>R2 = 0xaaaaaaa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000004</td>
<td>00132000</td>
<td>AND R3,R1,R2</td>
<td>R3 = 0x00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000005</td>
<td>01135555</td>
<td>ANDI R3,R1,0x5555</td>
<td>R3 = 0x00005555</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000006</td>
<td>02132000</td>
<td>OR R3,R1,R2</td>
<td>R3 = 0xffffffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000007</td>
<td>0313aaa</td>
<td>ORI R3,R1,0xaaa</td>
<td>R3 = 0x5555ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000008</td>
<td>04132000</td>
<td>XOR R3,R1,R2</td>
<td>R3 = 0xffffffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000009</td>
<td>05135555</td>
<td>XORI R3,R1,0x5555</td>
<td>R3 = 0x55550000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000a</td>
<td>06133000</td>
<td>XNOR R3,R1,R3</td>
<td>R3 = 0xffffaaaa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000b</td>
<td>07135555</td>
<td>XNORI R3,R1,0x5555</td>
<td>R3 = 0xaaaaaffff</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.3 Shift Instructions

File: inputs/test.sh.t.exe

```
00000000 3b01aaaa LDUI R1,0xaaaa R1 = 0xaaaa0000
00000001 0311aaaa ORI R1,R1,0xaaaa R1 = 0xaaaaaaaa
00000002 39020000 LDLI R2, 0x8 R2 = 0000000008
00000003 22112000 LSL R1,R1,R2 R1 = 0xaaa0000
00000004 23110008 LSLI R1,R1,0x8 R1 = 0xaaaa000000
00000005 24112000 LSR R1,R1,R2 R1 = 0x000aaaa00
00000006 25110008 LSRI R1,R1,0x8 R1 = 0x0000aaaa
00000007 23110010 LSL R1,R1,0x10 R1 = 0xaaaa0000
00000008 26112000 ASR R1,R1,R2 R1 = 0xffaaaa00
00000009 27110008 ASRI R1,R1,0x8 R1 = 0xffffaaaa
0000000a 23110011 LSLI R1,R1,0x11 R1 = 0x55540000
0000000b 26112000 ASR R1,R1,R2 R1 = 0x00555400
```

3.4 Comparison Instructions

File: inputs/testcmp.exe

```
00000000 3b015555 LDUI R1,0x5555 R1 = 0x55550000
00000001 03115555 ORI R1,R1,0x5555 R1 = 0x55555555
00000002 3b02aaaa LDUI R2,0xaaaa R2 = 0xaaaa0000
00000003 0322aaaa ORI R2,R2,0xaaaa R2 = 0xaaaaaaaa
00000004 10132000 SLT R3,R1,R2 R3 = 0x00000000
00000005 10231000 SLT R3,R2,R1 R3 = 0x00000001
00000006 12231000 SLTU R3,R2,R1 R3 = 0x00000000
00000007 12132000 SLTU R3,R1,R2 R3 = 0x00000001
00000008 14132000 SGE R3,R1,R2 R3 = 0x00000001
00000009 14231000 SGE R3,R2,R1 R3 = 0x00000000
0000000a 16231000 SGEU R3,R2,R1 R3 = 0x00000001
0000000b 16132000 SGEU R3,R1,R2 R3 = 0x00000000
0000000c 18132000 SEQ R3,R1,R2 R3 = 0x00000000
0000000d 18330000 SEQ R3,R3,R0 R3 = 0x00000001
0000000e 11135555 SLTI R3,R1,0x5555 R3 = 0x00000000
0000000f 11235555 SLTI R3,R2,0x5555 R3 = 0x00000001
00000010 1113aaaa SLTI R3,R1,0xaaaa R3 = 0x00000000
00000011 1123aaaa SLTI R3,R2,0xaaaa R3 = 0x00000001
00000012 13135555 SLTUI R3,R1,0x5555 R3 = 0x00000000
00000013 13235555 SLTUI R3,R2,0x5555 R3 = 0x00000000
00000014 1313aaaa SLTUI R3,R1,0xaaaa R3 = 0x00000000
00000015 1323aaaa SLTUI R3,R2,0xaaaa R3 = 0x00000000
00000016 15135555 SGEI R3,R1,0x5555 R3 = 0x00000001
00000017 15235555 SGEI R3,R2,0x5555 R3 = 0x00000000
00000018 1513aaaa SGEI R3,R1,0xaaaa R3 = 0x00000001
00000019 1523aaaa SGEI R3,R2,0xaaaa R3 = 0x00000000
0000001a 17135555 SGEUI R3,R1,0x5555 R3 = 0x00000001
0000001b 17235555 SGEUI R3,R2,0x5555 R3 = 0x00000001
0000001c 1713aaaa SGEUI R3,R1,0xaaaa R3 = 0x00000001
0000001d 1723aaaa SGEUI R3,R2,0xaaaa R3 = 0x00000001
0000001e 19330000 SEQI R3,R3,0x0000 R3 = 0x00000000
```
3.5 Data Access Instructions

File: inputs/testmem.exe

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>ADD R1,R0,R0</td>
<td>R1 = 0x00000000</td>
<td></td>
</tr>
<tr>
<td>00000002</td>
<td>ORI R2,R2,0x5555</td>
<td>R2 = 0x55555555</td>
<td></td>
</tr>
<tr>
<td>00000003</td>
<td>LDUI R3,0xaaaa</td>
<td>R3 = 0xaaaaaaa</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>STW R1,R2</td>
<td></td>
</tr>
<tr>
<td>00000006</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000007</td>
<td>STW R1,R3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000008</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000009</td>
<td>STW R1,R2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000000a</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000000b</td>
<td>STW R1,R3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000000c</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000000d</td>
<td>STW R1,R2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000000e</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000000f</td>
<td>STW R1,R3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000100</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000110</td>
<td>STW R1,R3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000120</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000130</td>
<td>STW R1,R3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000140</td>
<td>ADD R1,R0,R0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000150</td>
<td>LDW R3,R1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000160</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000170</td>
<td>LDW R3,R1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000180</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000190</td>
<td>LDW R3,R1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000200</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000210</td>
<td>LDW R3,R1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000220</td>
<td>ADDI R1,R1,0x0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000230</td>
<td>LDW R3,R1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.6 Control Transfer Instructions

File: inputs/testctl.exe

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>39010001</td>
<td>LDLI R1,0x0001 R1 = 0x00000001</td>
</tr>
<tr>
<td>00000001</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>00000002</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>00000003</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>00000004</td>
<td>3710ffff</td>
<td>BRTI R1,loop1</td>
</tr>
<tr>
<td>00000005</td>
<td>39010000</td>
<td>LDLI R1,0x0000</td>
</tr>
<tr>
<td>00000006</td>
<td>08000000</td>
<td>loop2 NOP</td>
</tr>
<tr>
<td>00000007</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>00000008</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>00000009</td>
<td>3510ffff</td>
<td>BRFI R1,loop2</td>
</tr>
<tr>
<td>0000000a</td>
<td>39010001</td>
<td>LDLI R1,0x0001</td>
</tr>
<tr>
<td>0000000b</td>
<td>3b035555</td>
<td>LDUI R3,0x5555</td>
</tr>
<tr>
<td>0000000c</td>
<td>03335555</td>
<td>ORI R3,R3,0x5555 R3 = 0x55555555</td>
</tr>
<tr>
<td>0000000d</td>
<td>3b04aaa</td>
<td>LDUI R4,0xaaaa</td>
</tr>
<tr>
<td>0000000e</td>
<td>0344aaa</td>
<td>ORI R4,R4,0xaaaa R4 = 0xaaaaaaaa</td>
</tr>
<tr>
<td>0000000f</td>
<td>41430000</td>
<td>BRTL R3,R4,jump1</td>
</tr>
<tr>
<td>00000010</td>
<td>3905011d</td>
<td>LDLI R5,0x0011 jal jumps back to 0x0000001d</td>
</tr>
<tr>
<td>00000011</td>
<td>43340000</td>
<td>BRLE R3,R3,jump2</td>
</tr>
<tr>
<td>00000012</td>
<td>3905001f</td>
<td>LDLI R5,0x0013 jal jumps back to 0x0000001f</td>
</tr>
<tr>
<td>00000013</td>
<td>45340000</td>
<td>BRE R4,R4,jump3</td>
</tr>
<tr>
<td>00000014</td>
<td>39050021</td>
<td>LDLI R5,0x0015 jal jumps back to 0x00000021</td>
</tr>
<tr>
<td>00000015</td>
<td>47430000</td>
<td>BRNE R3,R4,jump4</td>
</tr>
<tr>
<td>00000016</td>
<td>39050023</td>
<td>LDLI R5,0x0017 jal jumps back to 0x00000023</td>
</tr>
<tr>
<td>00000017</td>
<td>3905002e</td>
<td>LDLI R5,0x0027 jal jumps to 0x0000002e</td>
</tr>
<tr>
<td>00000018</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>00000019</td>
<td>30570000</td>
<td>JAL R7,R5</td>
</tr>
<tr>
<td>0000001a</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>0000001b</td>
<td>08000000</td>
<td>jump1 NOP</td>
</tr>
<tr>
<td>0000001c</td>
<td>30570000</td>
<td>JAL R7,R5</td>
</tr>
<tr>
<td>0000001d</td>
<td>08000000</td>
<td>NOP bubble</td>
</tr>
<tr>
<td>0000001e</td>
<td>08000000</td>
<td>jump2 NOP</td>
</tr>
<tr>
<td>0000001f</td>
<td>30570000</td>
<td>JAL R7,R5</td>
</tr>
<tr>
<td>00000020</td>
<td>08000000</td>
<td>NOP bubble</td>
</tr>
<tr>
<td>00000021</td>
<td>08000000</td>
<td>jump3 NOP</td>
</tr>
<tr>
<td>00000022</td>
<td>30570000</td>
<td>JAL R7,R5</td>
</tr>
<tr>
<td>00000023</td>
<td>08000000</td>
<td>jump4 NOP</td>
</tr>
<tr>
<td>00000024</td>
<td>08000000</td>
<td>NOP bubble</td>
</tr>
<tr>
<td>00000025</td>
<td>30570000</td>
<td>JAL R7,R5</td>
</tr>
<tr>
<td>00000026</td>
<td>08000000</td>
<td>NOP</td>
</tr>
<tr>
<td>00000027</td>
<td>02334000</td>
<td>OR R3,R3,R4 R3 = 0xffffffff</td>
</tr>
</tbody>
</table>

5
4  Second Set: C Programs

4.1  Sort Program

Files: inputs/sort.c (source), inputs/sort.exe (executable)

main()
{
    int ra[9], rra;
    int l, j, ir, i, n;

    ra[1] = 40;
    ra[2] = 8;
    ra[3] = 48;
    ra[4] = 16;
    ra[5] = 56;
    ra[6] = 24;
    ra[7] = 46;
    ra[8] = 32;
    n = 8;
    l = (n >> 1) + 1;
    ir = n;
    for (;;) {
        if (l > 1)
            rra = ra[--l];
        else {
            rra = ra[ir];
            ra[ir] = ra[l];
            if (!--ir)
                break;
        }
    }
    ir = l;
    j = l << 1;
    while (j <<= ir) {
        if (j < ir & ra[i] < ra[j+1]) ++j;
        if (rra < ra[j]) {
            ra[i] = ra[j];
            j += (i == j);
        } else
            j = ir + 1;
        ra[i] = rra;
    }
}
With a 10 ns clock cycle, this program runs for 8610 ns. The following two tables show where to look for the results after execution have finished. Table 1 is applicable if the internal data cache is enabled, while Table 2 is for the case when the data cache is disabled.

Table 1: Result locations, data cache enabled.

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>Word 0</th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x029</td>
<td></td>
<td></td>
<td></td>
<td>0x0008</td>
</tr>
<tr>
<td>0x030</td>
<td>0x0010</td>
<td>0x0018</td>
<td>0x0020</td>
<td>0x0028</td>
</tr>
<tr>
<td>0x031</td>
<td>0x002E</td>
<td>0x0030</td>
<td>0x0038</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Result locations, data cache disabled.

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>1015</th>
<th>1016</th>
<th>1017</th>
<th>1018</th>
<th>1019</th>
<th>1020</th>
<th>1021</th>
<th>1022</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0x0008</td>
<td>0x0010</td>
<td>0x0018</td>
<td>0x0020</td>
<td>0x0028</td>
<td>0x002E</td>
<td>0x0030</td>
<td>0x0038</td>
</tr>
</tbody>
</table>
### 4.2 Matrix Multiplication Program

Files: inputs/matrix.c (source), inputs/matrix.exe (executable)

```c
#include "trisc.h"

main()
{
    int ma[3][3];
    int mb[3][3];
    int mc[3][3];
    int n, i, j, k;

    ma[0][0] = 58; ma[0][1] = 50; ma[0][2] = 42;
    ma[1][0] = 26; ma[1][1] = 18; ma[1][2] = 10;
    ma[2][0] = 60; ma[2][1] = 52; ma[2][2] = 44;

    mb[0][0] = 40; mb[0][1] = 8; mb[0][2] = 48;
    mb[1][0] = 56; mb[1][1] = 24; mb[1][2] = 64;
    mb[2][0] = 39; mb[2][1] = 7; mb[2][2] = 47;

    n = 3;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            mc[i][j] = 0;
    for (k = 0; k < n; k++)
        for (i = 0; i < n; i++)
            mc[i][j] = mc[i][j] + TRMultiply32Unsigned(ma[i][k], mb[k][j]);
}
```

With a 10 ns clock cycle, this program runs for 307600 ns. The following two tables show where to look for the results after execution have finished. Table 3 is applicable if the internal data cache is enabled, while Table 4 is for the case when the data cache is disabled.

Table 3: Result locations, data cache enabled.

<table>
<thead>
<tr>
<th>Cache Line</th>
<th>Word 0</th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0x1A66</td>
<td>0x07A6</td>
<td>0x1F16</td>
<td>0x0986</td>
</tr>
<tr>
<td>26</td>
<td>0x02C6</td>
<td>0x0B36</td>
<td>0x1B74</td>
<td>0x07F4</td>
</tr>
<tr>
<td>27</td>
<td>0x2054</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: Result locations, data cache disabled.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>996</td>
<td>0x1A66</td>
</tr>
<tr>
<td>997</td>
<td>0x07A6</td>
</tr>
<tr>
<td>998</td>
<td>0x1F16</td>
</tr>
<tr>
<td>999</td>
<td>0x0986</td>
</tr>
<tr>
<td>1000</td>
<td>0x2C6</td>
</tr>
<tr>
<td>1001</td>
<td>0xB36</td>
</tr>
<tr>
<td>1002</td>
<td>0x1B74</td>
</tr>
<tr>
<td>1003</td>
<td>0x07F4</td>
</tr>
<tr>
<td>1004</td>
<td>0x2054</td>
</tr>
</tbody>
</table>
4.3 Binary Search Program

Files: inputs/search.c (source), inputs/search.exe (executable)

```c
#include "trisc.h"

#define SIZE 40
#define KEY 16

main()
{
    int size, tab[SIZE];
    int i;
    unsigned int low, high, mid;
    int key, cond;
    int pos, value

    size = SIZE;
    key = KEY;
    for (i = 0; i < size; i++)
        tab[i] = i;
    low = 0;
    high = size - 1;
    while (low <= high) {
        mid = TRDivide32Unsigned((low + high), 2);
        cond = key - tab[mid];
        if (cond > 0)
            low = mid + 1;
        else
            if (cond < 0)
                high = mid - 1;
            else
                break;
    }
    pos = mid;
    value = tab[mid];
}
```

With a 10 ns clock cycle, this program runs for 25260 ns. After that, register R12 should have the value 0.
4.4 Cache Test Program

Files: inputs/cache.c (source), inputs/cache.exe (executable)

main()
{
}