COL216 Assignment 2

Stage 2

Rishabh Dhiman
2020CS10837

19 February 2022

1 Objective

Construct an ALU, a register file, a program memory unit and a data memory unit for a rudimentary ARM processor in VHDL.

2 Technical Details

- The VHDL code was analyzed, simulated and synthesized using GHDL 1.0.0.
- The waveform viewer used is GTKWave Analyzer v3.3.104.

3 Documentation

The submission contains five new VHDL files defining the various units,

- `cond_checker.vhdl`,
- `program_counter.vhdl`,
- `flag_circuit.vhdl`,
- `decoder.vhdl`, and
- `processor.vhdl`.

along with few changes to the older files, namely splitting `memory.vhdl` into two files for program and data memory, they now input the byte address of the memory locations, as opposed to the word address in the previous submission, and updating the `types.vhdl` file.

Along with this, there are three new testbenches for testing these units

- `program_counter_tb.vhdl`,
- `flag_circuit_tb.vhdl`, and
- `processor_tb.vhdl`.

Finally, there's a single `processor_synth.vhdl` file in `src/output` which contains the RTL description of the complete processor, synthesized using GHDL.

Along with the code, the waveforms on simulating the testbenches, in the form of `.ghw` and `.pdf` files are stored in the output folder.

4 Testing Procdure

The code was analyzed, simulated and synthesized using GHDL.
You can simulate it yourself using the `makefile` provided in the `src` directory.

1. Ensure that Make and GHDL are installed. Note that older versions of GHDL do not support synthesis, the default version installed on Ubuntu 21.10 is recent enough. However the version in Ubuntu 18.04 LTS is too old.
2. Navigate to the `src` directory.
3. Create a folder called `output` inside the `src` directory (if it doesn't exist already).
4. In the commandline, run `make` or `make all` to analyze, and then simulate the programs.
5. `.ghw` waveform files along with the synthesized VHDL will be created in the output directory. Note that the earlier contents in the output folder will be overwritten.
6. You can finally run `make clean` to delete any temporary files created in the process.

`flag_circuit_tb`

This tests the ALU by iterating over 4 values of the two operands, `0x7FFF_FFFF`, `0x0000_0000`, `0x0000_0001`, and `0xFFFF_FFFF`, two possible opcodes `cmp`, `add` and the two possible states of the `enable` pin.
The program uses the (already tested) ALU unit for computing the result of the operations.

`program_counter_tb`

This first waits for 3 clock cycles, and the `pc` increments by 4. It then enables the `branch` pin and increments `pc` with a positive offset of 2. It then clears the `branch` pinwaits for a few clock periods, and then decrements `pc` with an offset of 5 using the `branch` pin. It then clears the `branch` pin and waits for few cycles before ending the test.

`processor_tb`

This simply executes the program that has been hardcoded into the program memory.
The two sample programs provided, after minor modification, were used for testing.
Namely `#10` in the first program was changed to `#16`.

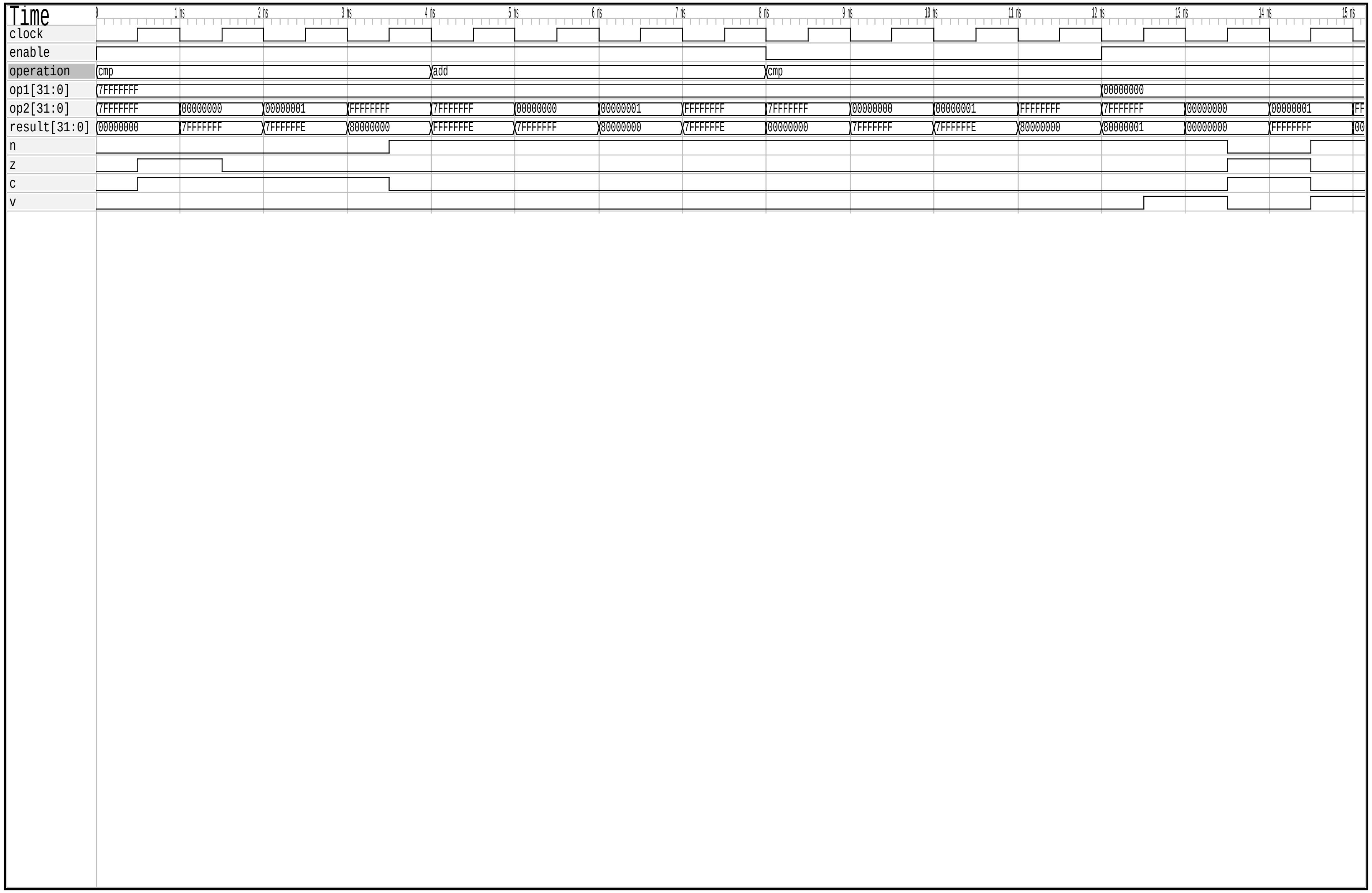
5 Results

All the testbenches gave the expected results.

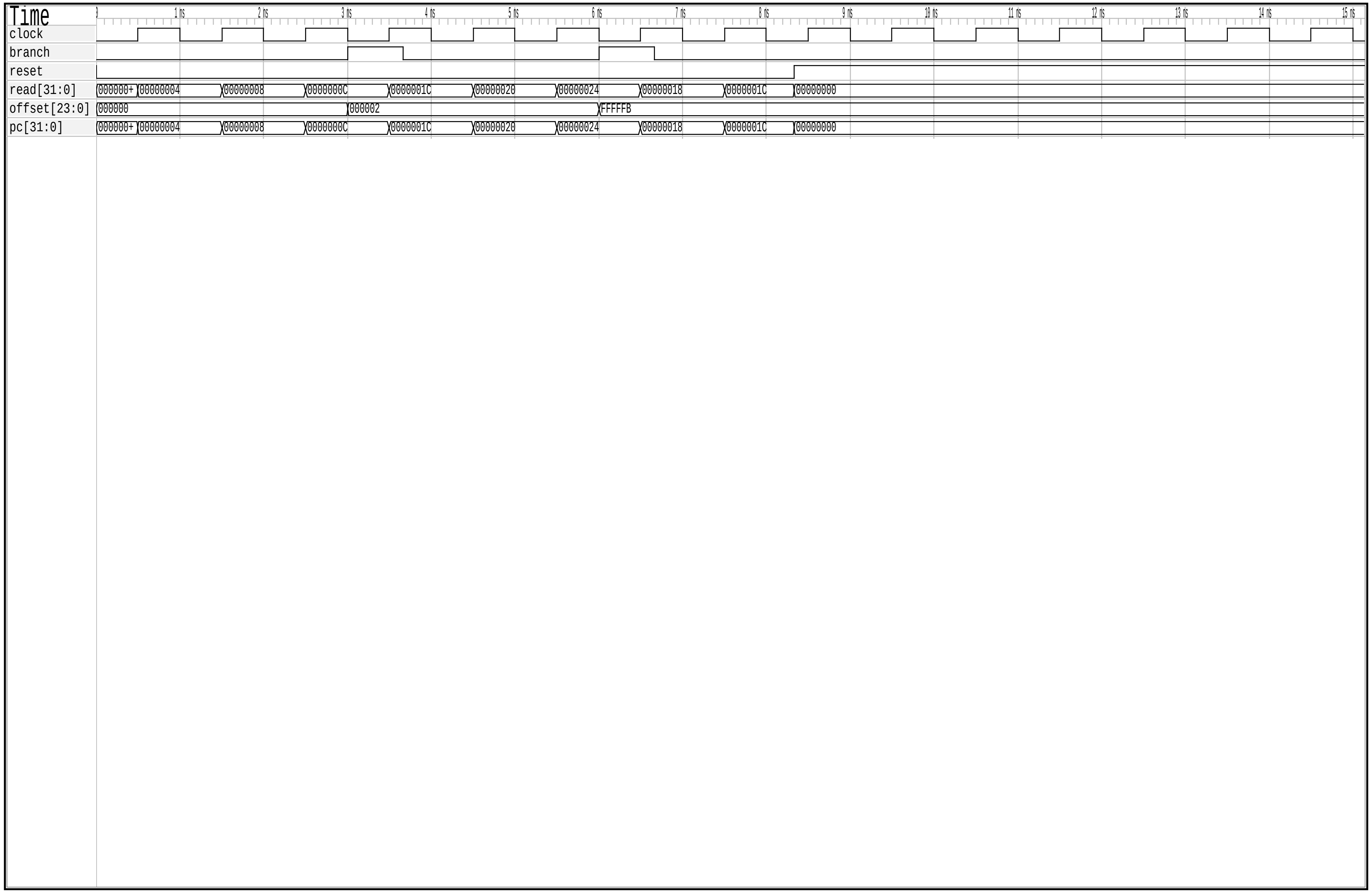
The output waveform of the testbench results can be viewed in .ghw and .pdf files in the output folder.

The .pdf files don't contain the entire waveform in some cases as the total number of pages required would be prohibitively large. However, the .ghw files can be opened in any waveform analyzer like GTKWave to view the output waveform.

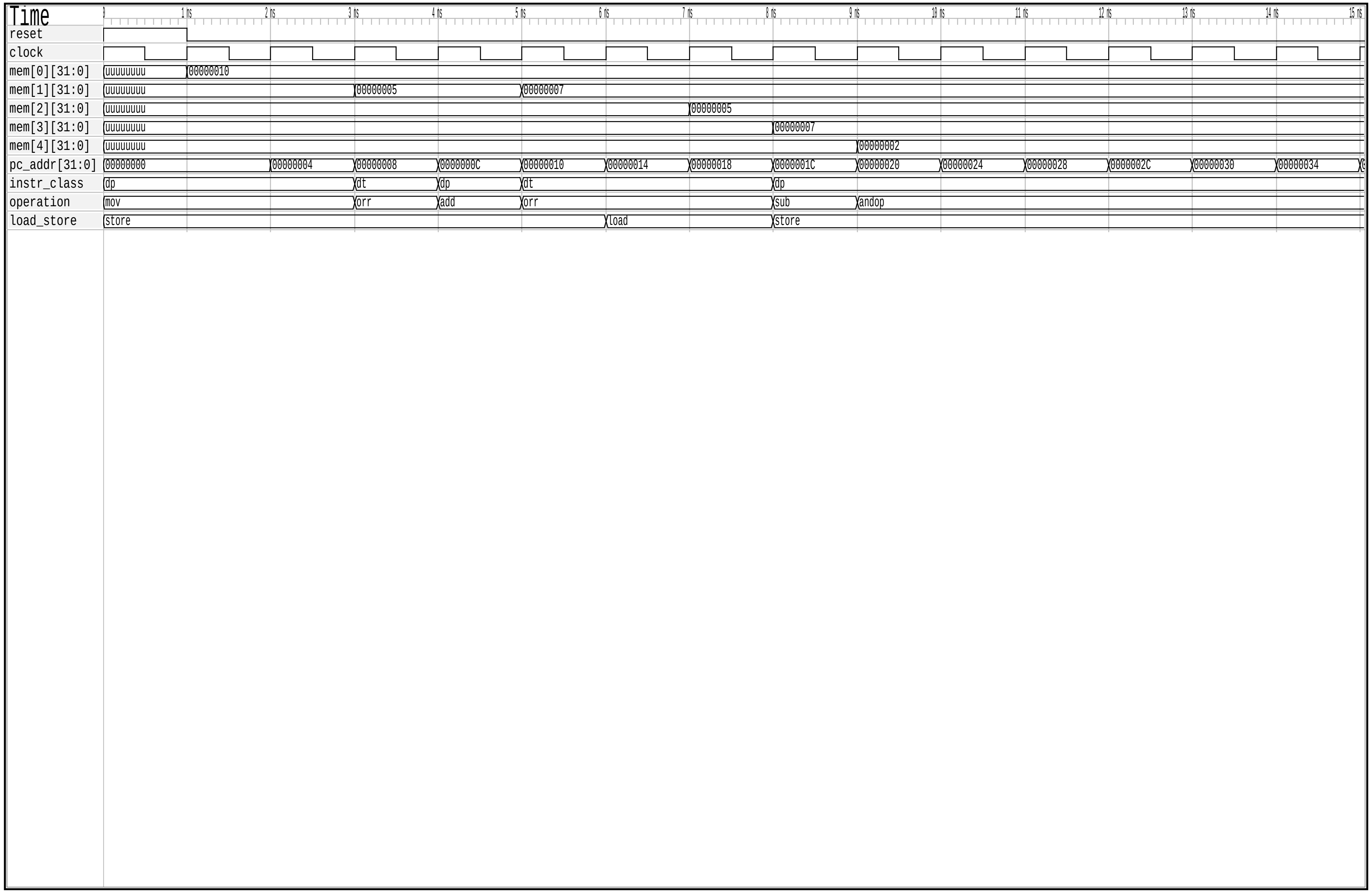
The pdf files are also attached at the end of this report.


Time
0
1 ns
2 ns
3 ns
4 ns
5 ns
6 ns
7 ns
8 ns
9 ns
10 ns
11 ns
12 ns
13 ns
14 ns
15 ns
clock
enable
operation
cmp
add
cmp
op1[31:0]
7FFFFFFF
00000000
op2[31:0]
7FFFFFFF
00000000
00000001
FFFFFFFF
7FFFFFFF
00000000
00000001
FFFFFFFF
7FFFFFFF
00000000
00000001
FFFFFFFF
7FFFFFFF
00000000
00000001
FF
result[31:0]
00000000
7FFFFFFF
7FFFFFFE
80000000
FFFFFFFE
7FFFFFFF
80000000
7FFFFFFE
00000000
7FFFFFFF
7FFFFFFE
80000000
80000001
00000000
FFFFFFFF
00
n
z
c
v

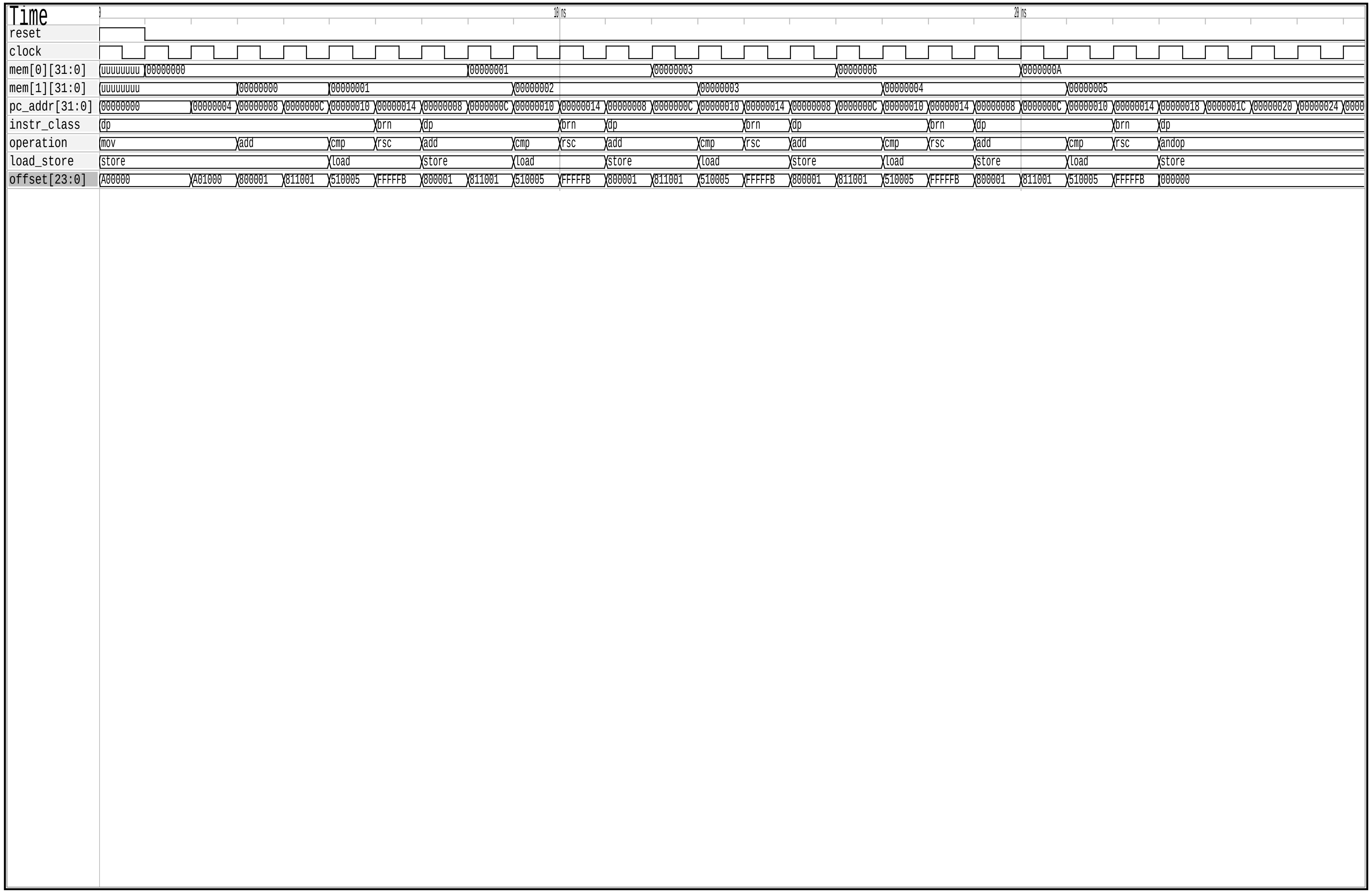
program_counter_tb.pdf



processor_tb_program_1.pdf



processor_tb_program_2.pdf


Time
0
10 ns
20 ns
reset
clock
mem[0][31:0]
uuuuuuuu 00000000 00000001 00000003 00000006 0000000A
mem[1][31:0]
uuuuuuuu 00000000 00000001 00000002 00000003 00000004 00000005
pc_addr[31:0]
00000000 00000004 00000008 0000000C 00000010 00000014 00000008 0000000C 00000010 00000014 00000008 0000000C 00000010 00000014 00000008 0000000C 00000010 00000014 00000008 0000000C 00000010 00000014 00000018 0000001C 00000020 00000024 0000
instr_class
dp brn dp brn dp brn dp brn dp brn dp
operation
mov add cmp rsc add cmp rsc add cmp rsc add cmp rsc add cmp rsc andop
load_store
store load store load store load store load store load store
offset[23:0]
A00000 A01000 800001 811001 510005 FFFFFB 800001 811001 510005 FFFFFB 800001 811001 510005 FFFFFB 800001 811001 510005 FFFFFB 800001 811001 510005 FFFFFB 000000