

TASK: Using the HW Counters, measure the number of cycles, instructions, loads and stores in the program from **Error! Reference source not found..** How much time in total (both for reading and writing) does it take to access Main Memory?

```
PROBLEMS  OUTPUT  DEBUG

Connected success !
Cycles = 18393
Instructions = 5306
Loads = 1098
Stores = 1092
```

- The loop contains 5 instructions.
- Ideally, IPC could be up to 1.
- However, we miss several cycles per iteration due to the read/write latency to Main Memory.

TASK: Use the example from *[RVfpgaBasysPath]/Labs/Lab19/LW_Instruction_MainMemory* to estimate the Main Memory read latency using the HW Counters.

Execution in Main Memory:

```
PROBLEMS  OUTPUT  DEBUG

Connected success !
UCycles = 39969
Instructions = 10328
Loads = 4103
Stores = 96
```

TASK: Analyse module **ifu_ic_mem** and the parameters of file *el2_param.vh* to understand how the elements in **Error! Reference source not found.** are implemented.

Solution not provided for this exercise.

TASK: Analyse the Verilog code from **Error! Reference source not found.** and explain how it operates based on the above explanations.

```
else begin : two_ways_plru
    assign replace_way_mb_any[0] = (~way_status_mb_ff & tagv_mb_ff[0] & tagv_mb_ff[1]) / ~tagv_mb_ff[0];
    assign replace_way_mb_any[1] = ( way_status_mb_ff & tagv_mb_ff[0] & tagv_mb_ff[1]) / ~tagv_mb_ff[1] & tagv_mb_ff[0];
```

If both ways are invalid (i.e. **tagv_mb_ff = 00**), way 0 must be replaced first:

- **replace_way_mb_any[0] = 1**, as the second operand of the OR, which is **~tagv_mb_ff[0]**, is 1.

- `replace_way_mb_any[1] = 0`, as the two operands of the OR are 0.

If there is one invalid way, this is the one replaced.

- If way 0 is invalid, the second operand of the OR, which is `~tagv_mb_ff[0]`, is 1.
- If way 1 is invalid and way 0 is valid, the second operand of the OR, which is `~tagv_mb_ff[1] & tagv_mb_ff[0]`, is 1.

If both ways are valid, signal **`way_status_mb_ff`** holds the LRU state (thus, the least recently used way, or the way to replace first) of the selected set, determines the way to replace.

TASK: Analyse the Verilog code that performs the same functionality on a 4-way I\$.

Solution not provided for this exercise.

TASK: Analyse the Verilog code from **Error! Reference source not found.** and explain how it operates based on the above explanations.

```
assign way_status_hit_new[pt.ICACHE_STATUS_BITS-1:0] = ic_rd_hit[0];
assign way_status_rep_new[pt.ICACHE_STATUS_BITS-1:0] = replace_way_mb_any[0];

end
// Make sure to select the way status hit new even when in hit under miss.
assign way_status_new[pt.ICACHE_STATUS_BITS-1:0] = (bus_ifu_wr_en_ff_q & last_beat) ? way_status_rep_new[pt.ICACHE_STATUS_BITS-1:0] :
way_status_hit_new[pt.ICACHE_STATUS_BITS-1:0];
```

The new value of the LRU state is determined by signal **`way_status_new`**.

- If there was a hit, signal **`ic_rd_hit`** determines the new value, as it holds the way where the hit has taken place.
- If there was a replacement, signal **`replace_way_mb_any`** determines the new value, as it holds the way that has been replaced.

TASK: Analyse the Verilog code that performs the same functionality on a 4-way I\$.

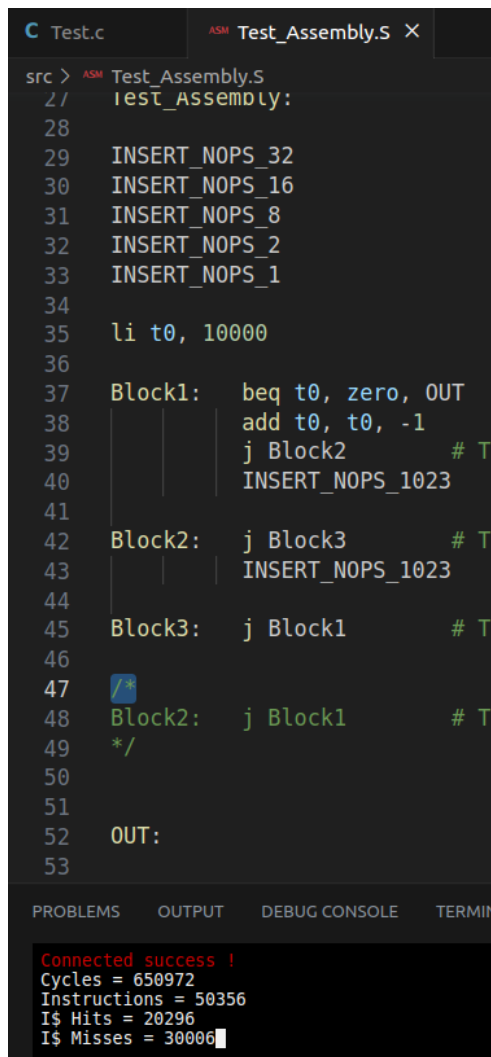
Solution not provided for this exercise.

1. EXERCISES

- 1) Transform the infinite loop from **Error! Reference source not found.** into a loop with 10000 iterations, but keep the `j` instructions at the same addresses. Measure the number of cycles and I\$ hits and misses. Then remove one of the `j` instructions and measure the same metrics. Compare and explain the results.

A Catapult project is provided at:

`[RVfpgaBasysPath]/Labs/RVfpgaLabsSolutions/Lab19/InstructionMemory_LRU_Example_FiniteLoop`



```

src > ASM Test_Assembly.S
27 test_asmblly:
28
29 INSERT_NOPs_32
30 INSERT_NOPs_16
31 INSERT_NOPs_8
32 INSERT_NOPs_2
33 INSERT_NOPs_1
34
35 li t0, 10000
36
37 Block1: beq t0, zero, OUT
38         add t0, t0, -1
39         j Block2      # T
40         INSERT_NOPs_1023
41
42 Block2: j Block3      # T
43         INSERT_NOPs_1023
44
45 Block3: j Block1      # T
46
47 /*
48 Block2: j Block1      # T
49 */
50
51
52 OUT:
53

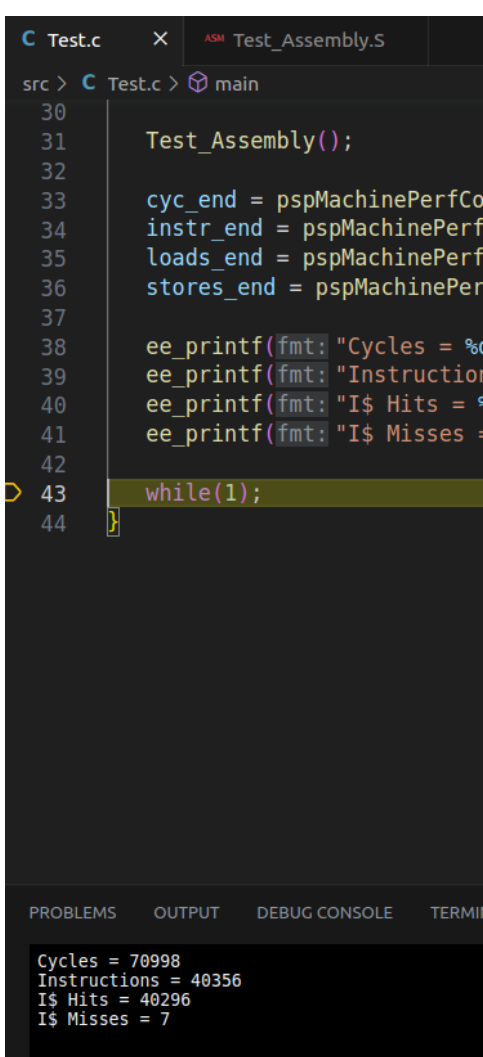
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

Connected success !
Cycles = 650972
Instructions = 50356
I$ Hits = 20296
I$ Misses = 30006

```



```

src > C Test.c > main
30
31 Test_Assembly();
32
33 cyc_end = pspMachinePerfCo
34 instr_end = pspMachinePerf
35 loads_end = pspMachinePerf
36 stores_end = pspMachinePer
37
38 ee_printf(fmt: "Cycles = %d
39 ee_printf(fmt: "Instruction
40 ee_printf(fmt: "I$ Hits = %
41 ee_printf(fmt: "I$ Misses =
42
43 while(1);
44

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

Cycles = 70998
Instructions = 40356
I$ Hits = 40296
I$ Misses = 7

```

- There are no I\$ misses in the code with 2 jump instructions, except for the first iteration. This dramatically decreases the number of cycles.

2) Extend **Error! Reference source not found.** to analyse in detail how each 64-bit chunk is written in the I\$.

Solution not provided for this exercise.

3) Analyse in simulation and on the board other I\$ configurations. For example, it can be very interesting to analyse a 4-way I\$.

Solution not provided for this exercise.

You can find a useful study in RVfpga v2.2 (provided at: <https://university.imgtec.com/rvfpga-download-page-en/>), Lab 19, where a 4-way I\$ is used in SweRV EH1.

4) Analyse the logic that checks the correctness of the parity information.

Solution not provided for this exercise.