**EXERCISE 1:**

You should initially observe a miss. During the miss, *dcc\_dmiss\_m*=1 (LED1) and *dcc\_pm\_dhit\_m*=0 (LED0). Moreover, as we will explain in Lab 22-A, the pipeline stops for several cycles, during which the core requests the missed line to main memory. This can be observed through signals *mpc\_run\_ie* (LED4), *mpc\_run\_m* (LED3) and *mpc\_run\_w* (LED2), which go low for several cycles.

After each miss, you should observe 4 consecutive hits. The first one is due to the instruction that missed, when it receives the requested data, and the three next hits are due to the subsequent three lw instructions, which are accessing to the same D$ line and thus hit.

**EXERCISE 2:**

You should observe that, when each lw instruction executes the M-Stage, a miss is signaled (*dcc\_dmiss\_m*=1 (LD1)), and the processor stalls for some cycles, until the requested word arrives from memory.

When you remove the third lw, the two other lw instructions will always hit in the D$. Thus, you should observe that the pipeline never stops when executing the loop (*mpc\_run\_ie*=*mpc\_run\_m*=*mpc\_run\_w*=1) and that all lw instructions generate a hit at the M-Stage (*dcc\_pm\_dhit\_m*=1 (LD0)).

**EXERCISE 3:**

1st code: When the program reaches the two nested loops, you will observe, in LD0 (D$ hit: *dcc\_pm\_dhit\_m*) and LD1 (D$ miss: *dcc\_dmiss\_m*), a pattern with 1 miss followed by 3 hits.

2nd code: When the program reaches the two nested loops, you will observe, in LD0 (D$ hit: *dcc\_pm\_dhit\_m*) and LD1 (D$ miss: *dcc\_dmiss\_m*), a series of eight cache misses for the first iteration of the outer loop, followed by several outer loop iterations with only hits, and then the same pattern repeated again.