**Memory Operations**

**Increment I**

1- in R2

2- in R3

3- in R4

4- LDM R1,F5

5- PUSH R1  ***-- Hazard1***

6- PUSH R2

7- POP R1

8- POP R2

9- STD R2,200 ***-- Hazard2***

10- STD R1,202

11- LDD R3,202

12- LDD R4,200

Hazard1

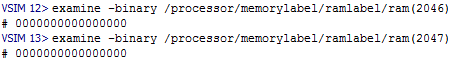
Data Hazard on R1, needs 2 NOPs before it to be executed properly.

Hazard2

Data Hazard on R2, needs 2 NOPs only, although it resembles load use case, STD instruction already stalls one cycle before reading the EA, that's why 2 NOPs are sufficient instead of 3.

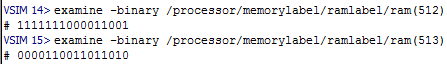
**Screenshots:**

When instruction 5 is at Memory stage:



False old value (0) of R1 was used and pushed in stack

When instruction 9 is at Memory stage:



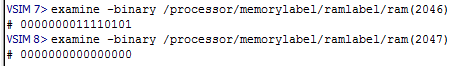
False old value (0CDAFE19) of R2 was used and stored in memory.

**Increment II**

All hazards are supposedly solved with Forwarding Unit

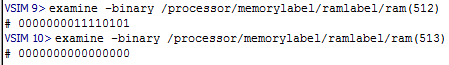
**Screenshots:**

When instruction 5 is at Memory stage:

****

A proper value of R1 (F5) was forwarded and stored in stack.

When instruction 9 is at Memory stage:



A proper value of R2 (F5) was forwarded and stored in memory, with no need to load use case hazard detection unit, for the bubble already inserted by STD as previously clarified.

**Increment III and Increment IV**

Exactly the same as II, It was already hazard-free, no more hazards need to be handled by stalling or flushing. The only load use case is already handled through the Nop introduced by STD normally according to our design.