***Consider the following MIPS code fragments, each containing two instructions. For each code fragment identify the type of hazard that exists between the two instructions and the registers involved.***

***a.***

|  |  |  |
| --- | --- | --- |
| ***LD DADD*** |  | ***R1, 0(R2) R3, R1, R2*** |

:: Data hazard- RAW (Read After Write)**;** DADD reads R1, which is being written by LD.

***b.***

|  |  |
| --- | --- |
| ***MULT DADD*** | ***R1, R2, R3 R1, R2, R3*** |

:: Data hazard- WAW (Write After Write); Both MULT and DADD write to R1. If DADD executes before MULT completes, the final value of R1 may be incorrect.

***c.***

|  |  |
| --- | --- |
| ***MULT MULT*** | ***R1, R2, R3 R4, R5, R6*** |

:: Structural Hazard; If the processor has only one multiplier unit, this creates a structural hazard because the second MULT must wait.

***d.***

|  |  |
| --- | --- |
| ***DADD SD*** | ***R1, R2, R3 R1, 2000(R0)*** |

:: Data hazard- RAW (Read After Write); SD needs the value in R1 that is being written by DADD.

***e.***

|  |  |
| --- | --- |
| ***DADD SD*** | ***R1, R2, R3 2000(R1), R4*** |

:: Data hazard- RAW (Read After Write); SD uses R1 for its address calculation, but R1 is written by DADD.

***a. Explain the behaviour of a 2-bit saturating counter branch predictor. Show the state of the predictor and the transition for each outcome of the branch.***

***b. Consider the following code:***

***for (i=0; i<N; i++)***

***if (x[i] == 0)***

***y[i] = 0.0;***

***else***

***y[i] = y[i]/x[i];***

***Assume that the assembly code generated is then:***

|  |  |  |
| --- | --- | --- |
| ***Loop:***    ***else:***  ***fall:*** | ***L.D***  ***L.D***  ***BNEZ***  ***ADD.D***  ***BEZ***  ***DIV.D DADDI DADDI DSUBI***  ***S.D***  ***BNEZ*** | ***F1, 0(R2)***  ***F2, 0(R3)***  ***F1, else***  ***F2, F0, F0***  ***R0, fall***  ***F2, F2, F1***  ***R2, R2, 8***  ***R3, R3, 8***  ***R1, R1, 1***  ***-8(R3), F2***  ***R1, loop*** |

***where:***

***• the value of N is already stored in R1***

***• the base addresses for x and y are stored in R2 and R3, respectively***

***• register F0 contains the value 0***

***• register R0 (always) contains the value 0***

***Assuming that every other element of x has the value 0, starting with the first one, show the outcomes of predictions when a 2-bit saturating counter is used to predict the inner branch BNEZ F1, else. Assume that the initial value of the counter is 00***

**(a): 2-Bit Saturating Counter Branch Predictor**

A 2-bit saturating counter is a finite state machine used to predict branch outcomes. It has four states:

* 00 (Strongly Not Taken)
* 01 (Weakly Not Taken)
* 10 (Weakly Taken)
* 11 (Strongly Taken)

State Transitions:

* + If branch is taken increment the counter (saturates at 11).
  + If branch is not taken decrement the counter (saturates at 00).

**(b): Predictor Behavior for Given Code**

Branch Pattern: The inner branch BNEZ F1, else alternates between Not Taken (when x[i] = 0) and Taken (when x[i] ≠ 0).

Initial State: 00 (Strongly Not Taken).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Branch Outcome | Prediction | Transition | New State |
| 1 | Not Taken | Not Taken | Stay at 00 | 00 |
| 2 | Taken | Not Taken | Increment to 01 | 01 |
| 3 | Not Taken | Not Taken | Decrement to 00 | 00 |
| 4 | Taken | Not Taken | Increment to 01 | 01 |