Practice Test 4

1. When/Why are branch predictors used?

* Branch prediction is used to overcome the fetch limitation imposed by control hazards in order to expose instruction-level parallelism

2. Explain the difference between static and dynamic branch prediction.

* Static - a prediction that uses information that was gathered before the execution of the program
* Dynamic - uses information about taken or not taken branches gathered at run-time to predict the outcome of a branch

3. Explain the difference between global and local branch predictors. Explain how/why each can be useful.

* Global Branch History - a shift register in which the outcome of any branch is stored. A "one" is stored for a taken branch and a "zero" for a non-taken one. The register is shifted through while storing the newest value. In order to address the table, the *n* last branch outcomes are considered.
* Local Branch History - a table of shift registers of the sort of a global branch history. Each shift register, however, refers to the last outcomes of one single branch. Since this local history table is accessed as a one-level branch prediction table, it is not guaranteed that no overlapping of the branches occurs, and in one shift register may be stored the information of different branches.

4. What is a tournament predictor?

Tournament predictors:

* use 2 predictors
* one based on global information
* the other based on local information
* and combine with a selector
* The selector is driven by a predictor….

5. What issues affect whether a CPU should use branch prediction or branch delay slots?

* The size/scale of the CPU and what it’s trying to accomplish. A CPU that’s is just decided if a toaster goes off should choose branch delay slots over a predictor while a laptop's CPU would benefit greater from branch prediction.
* Branch predictors also require an increase in the amount of hardware needed to make a successful predictor. This can be done easier in a larger scale machine.

6. Dynamic branch predictors need to store the branch history on which the predictions are based.

Explain how branch predictors store this “history” in a time- and space- efficient manner.

7. Consider the instruction moveqz R1, R2, R3 that sets R1 to be equal to R2 if, and only if,

R3 == 0.

Explain how adding this instruction to a CPU can potentially improve performance.

* Moveqz would allow us to replace the following 2 lines of code

bneq r3, r0, 1

r1 = r2

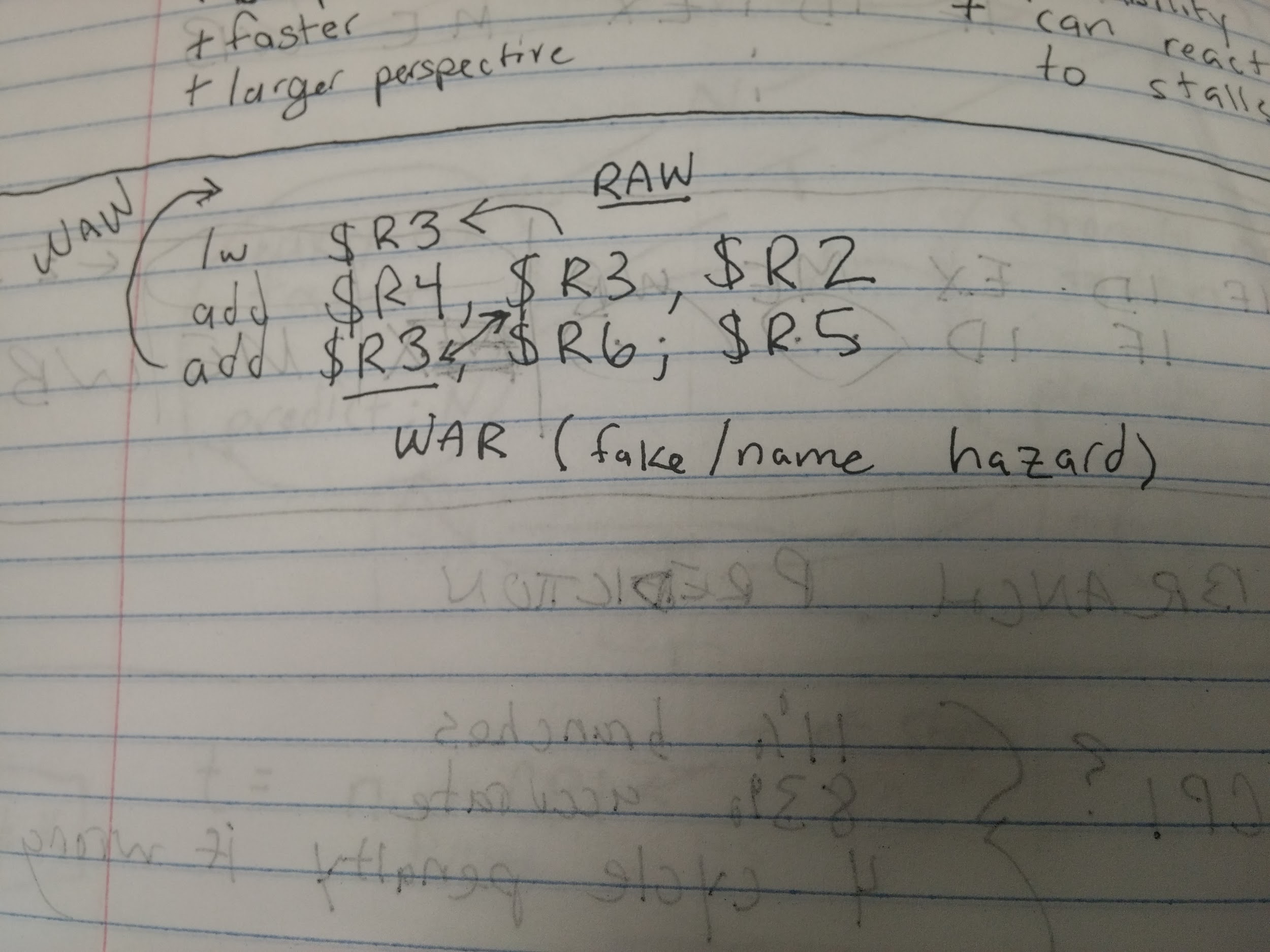
* Moveqz also doesnt have branch penalties so it would improve our cpus performance that way as well.

8. Review the Branch Prediction Homework

9. Review the Superscalar Homework

10. How does a program’s spatial locality affect it’s cache hit rate?

11. Be able to identify WAR and WAW data hazards

.

12. What types of processors are susceptible to WAR and WAW data hazards.

* Only pipelined CPUs are affected by these data hazards because they are processing more than one instruction in the same cycle.

13. How are WAR and WAW data hazards fundamentally different from RAW data hazards?

* WAR and WAW data hazards are ‘fake/name’ hazards and not ‘real’ data hazards, because the data they are accessing or writing to hasn’t been changed so it doesn't effect the program.