CSE 140 HW #6

1. Assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

```
          lw   r2, 0(r1)
label1:   beq  r2, r0, label2   # not taken once, then taken
          lw   r3, 0(r2)
          beq  r3, r0, label1   # taken
          add  r1, r3, r1
label2:   sw   r1, 0(r2)
```

   a. Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.
   b. Repeat *Part a*, but assume that delay slots are used. In the given code, the instruction that follows the branch is now the delay slot instruction for that branch.
   c. One way to move the branch resolution one stage earlier is to not need an ALU operation in conditional branches. The branch instructions would be ***"bez rd, label"*** and ***"bnez rd, label"***, and it would branch if the register has and does not have a zero value, respectively. Change the code above to use these branch instructions instead of beq. You can assume that register R8 is available for you to use as a temporary register, and that an seq (set if equal) R-type instruction can be used (you don;'t have to use either).
   d. As discussed in lecture, we can reduce the control hazards by adding a dedicated comparator in the ID stage (perform comparisons at ID stage instead of EXE stage). Describe THREE scenarios when new hazards are introduced in this setting.

2. The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-type	BEQ	Jump	LW	SW
40%	25%	5%	25%	5%

The accuracies of three different branch predictors are as follow:

Always-Taken	Always-Not-Taken	2-bit Predictor
45%	55%	85%

   a. Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the EX stage, that there are no data hazards, and that no delay slots are used.
   b. Repeat *Part a* for the "always-not-taken" predictor.

c. Repeat *Part a* for the 2-bit predictor.

d. With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions in a way that replaces a branch instruction with an ALU instruction? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.

e. With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions in a way that replaced each branch instruction with two ALU instructions? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced.

3. In this exercise we compare the performance of 1-issue and 2-issue processors, taking into account program transformations that can be made to optimize for 2-issue execution. Problems in this exercise refer to the following loop (written in C):

```
for(i = 0; i != j; i += 2)
   b[i] = a[i] - a[i + 1];
```

a. Translate this C code into MIPS instructions. Your translation should be direct, without rearranging instructions to achieve better performance. When writing MIPS code, assume that variables are kept in registers as follows, and that all registers except those indicated as Free are used to keep various variables, so they cannot be used for anything else.

i	j	a	b	c	Free
R5	R6	R1	R2	R3	R10, R11, R12

b. If the loop exits after executing only two iterations, draw a pipeline diagram for your MIPS code from *Part a* executed on a 2-issue processor. Assume the processor has perfect branch prediction and can fetch any two instructions (not just consecutive instructions) in the same cycle.

c. Rearrange your code from *Part a* to achieve better performance on a 2-issue statically scheduled processor.

d. Repeat *Part b*, but this time use your MIPS code from *Part c*.

e. What is the speedup of going from a 1-issue processor to a 2-issue processor? Use your code from *Part a* for both 1-issue and 2-issue, and assume that 1,000,000 iterations of the loop are executed. As in *Part b*, assume that the processor has perfect branch predictions, and that a 2-issue processor can fetch any two instructions in the same cycle.