**Homework Assignment #3**

***Due Date: 11/3, 11:59 p.m. Please submit via Blackboard. Late submissions are accepted till 11/8, 11:59 p.m, with 10% penalty each day. For all questions, please note that you need to show the steps how you obtain your result and please do NOT just provide the final answer.***

***Please name your submission file starting with “LastName\_FirstName\_HW3”.***

**Q1. (10 points)** Please briefly discuss what structural hazard and data hazard are. Please show a sample solution to overcoming each of these hazards.

**Q2. (15 points)** In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages have the following latencies:

![Graphical user interface

Description automatically generated]()

Also, assume that instructions executed by the processor are broken down as follows:

![A picture containing text

Description automatically generated]()

a. What is the clock cycle time in a pipelined and non-pipelined processor?

b. What is the total latency of an lw instruction in a pipelined and non-pipelined processor?

c. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

**Q3. (5 points)** What is the minimum number of cycles needed to completely execute *n* instructions on a CPU with a *k* stage pipeline? Justify your formula.

**Q4. (5 points)** Assume that x11 is initialized to 11 and x12 is initialized to 22. Suppose you executed the code below on a version of the pipeline that does NOT handle data hazards (i.e., the programmer is responsible for addressing data hazards by inserting NOP (no operations) instructions where necessary). What would the final values of registers x13 and x14 be?

addi x11, x12, 5

add x13, x11, x12

addi x14, x11, 15

**Q5. (5 points)** Add NOP instructions to the code below so that it will run correctly on a pipeline that does not handle data hazards.

addi x11, x12, 5

add x13, x11, x12

addi x14, x11, 15

add x15, x13, x12

**Q6. (15 points)** Assume we have the following sequence of instructions, and assume that it is executed on a 5-stage pipelined datapath:

add x15, x12, x11

lw x13, 8(x15)

lw x12, 0(x2)

or x13, x15, x13

sw x13, 0(x15)

a. If there is no forwarding or hazard detection, insert NOPs to ensure correct execution and draw the pipeline execution diagram.

b. Now, change and/or rearrange the code to minimize the number of NOPs needed. You can assume register x17 can be used to hold temporary values in your modified code.

c. Now assume we schedule the above reordered instructions to be executed on a 2-issue statically scheduled processor, as shown below, draw the pipeline execution diagram again.



**Q7. (5 points)** Please briefly discuss what the control hazard is. Please show a sample solution to overcoming the control hazard.

**Q8. (10 points)** An (*m*,*n*) correlating branch predictor uses the behavior of the most recent *m* executed branches to choose from 2*m* predictors, each of which is an *n*-bit predictor. A two-level local predictor works in a similar fashion, but only keeps track of the past behavior of each individual branch to predict future behavior. There is a design trade-off involved with such predictors: correlating predictors require little memory for history, which allows them to maintain 2-bit predictors for a large number of individual branches (reducing the probability of branch instructions reusing the same predictor), while local predictors require substantially more memory to keep history and are thus limited to tracking a relatively small number of branch instructions. For this exercise, consider a (1,2) correlating predictor that can track four branches (requiring 16 bits) versus a (1,2) local predictor that can track two branches using the same amount of memory. For the following branch outcomes, provide each prediction, the table entry used to make the prediction, any updates to the table as a result of the prediction, and the final misprediction rate of each predictor. Assume that all branches up to this point have been taken. Initialize each predictor to the following:

Table

Description automatically generatedTable

Description automatically generated

Table

Description automatically generated

**Q9. (10 points)** Please discuss what a superscalar processor is. Please discuss what are the advantages and disadvantages of a superscalar processor compared to a static multiple issue processor.

**Q10. (20 points)**

a. Please use the sequence of 4 instructions we discussed in class, i.e., below instructions, to illustrate how Tomasulo’s dynamic scheduling algorithm issues these 4 instructions, i.e., how Tomasulo’s algorithm works in the first issue step. Assume we have the exactly same hardware as discussed in the class and shown in Figure 3.10 in the textbook.

I1: fadd.d f2, f4, f1

I2: fmul.d f1, f2, f3

I3: fsub.d f4, f1, f2

I4: fadd.d f1, f2, f3

b. Based on Tomasulo’s dynamic scheduling algorithm, what are the possible orders of the execution of the above 4 instructions? Please list all possible orders. Please give the final results of f1, f2, f3, and f4 registers for each possible order and explain how Tomasulo’s algorithm results in dispatching this order and leads to the results of each register.

THE END.