Name Zachary Grella — Assignment 4 CS 3339 – Spring 2019

netID ztg5 — Due: per TRACS Friday @ 11:55pm

(email not long Axxxxx number) — 40 points (late until Sat @ noon -10 pts)

All submissions must be written in very neat handwriting and scanned (or typed) and submitted in PDF format to TRACS with the filename of Ax_netID.pdf. You may submit as many times as you like prior to the deadline; only the most recent submittal will be graded. All assignments must be submitted individually and reflect your own work; however, you are encouraged to work in groups and discuss the problems with your classmates.

1) [4 points] Given the following sequence of two MIPS instructions without forwarding paths how many stall cycles are required.

```
sw $t1, 0($t2)
beq $t1, $zero, label
```

sw available between mem + wb
beq resolved in decode

two stalls

2) [4 points] To prepare your p3 or p4 project for submission you need to execute the tar command to build a **T**ape **AR**chive (no tape involved it's a holdover). Complete the command line you need to execute in your project directory to build the file you will submit on TRACS:

```
$ tar czvf ztg5_project4.tgz *.cpp *.h makefile
```

Four command line options must be passed in addition to the filename of the created "tarball" and the input filename(s). List and Briefly describe the purpose of each command line arg (letter).

c-creates new archive v-tell it verbose
z-puts a file through gzip f-means file you created is the archive you created

3) [6 points] The project 4 assignment includes the following statement: "The table should be indexed as follows: index = (PC_{branch} >> 2) % BPRED_SIZE."

What is the value of BPRED_SIZE and where is it assigned?

64

Why is the address of the branch shifted by 2?

to move the branch to the right location

What does the '%' do?

prevents from indexing outside of the array

4) [6 points] Early version of the MIPS processor (without "**M**icroprocessor without **I**nterlocked **P**ipeline **S**tages") did not check for data hazards.

How was correct operation achieved?

the compiler was flooding it with no ops

Why was this technique abandoned?

It was flooding the cache and slowing operation

What was added to the processor in order to ensure correct operation?

hardware stalls

5) [4 points]Write the following 4 methods of branch prediction in order from lowest to highest expected performance.

1-bit dynamic prediction	Static prediction, never taken
Runtime prediction with BHT/BTB	2-bit dynamic prediction

Static prediction
1 bit dynamic
2 bit dynamic
Runtime prediction

6) [6 points]Complete the table to the right showing the steps for unsigned integer multiplication.

All entries except the result at the top and bottom are in binary.

Refer to lecture 3b.

$9_{decimal}$ X	$13_{decimal}$	= 117 decimal
Multiplier 4 bits ->	Multiplicand 8 bits <-	Product
1001	0000 1101	0000 0000
		+ 0000 1101
0100	0001 1010	0000 1101
		+ 0000 0000
0010	0011 0100	0000 1101
		+ 0000 0000
0001	0110 1000	0000 1101
		+ 0110 1000
	75 hex	⇦ 0111 0101

7) [6 points] To implement static multiple issue the compiler will group multiple instructions into "issue packets" based on the hardware microarchitecture. If there are many instructions in a single issue packet this is known as a VLIW (Very Long Instruction Word) implementation. For this question assume there are only two instructions in an issue packet.

Why does the instruction pairing of *add* and `lw` benefit the hardware implementation?
Bc we have the extra adder on the ALU when can load memory and do arithmetic at the same time

Based on the implementation discussed in class pairing an *add* instruction with an *and* instruction would introduce what type of hazard? A structural hazard will occur b/c both add + and cannot go through ALU at same time

How will the compiler resolve this issue?
The compiler will schedule the "issue packets"

8) [4 points] Briefly explain the difference between static and dynamic multiple issue pipeline scheduling. Include specifically who (or more correctly what) is responsible for the scheduling. For credit answer needs to be more than static = fixed and dynamic = changes.

Static
- Compiler remove all hazards
- Compiler groups instructions

Dynamic
- CPU avoids structural + data hazards by deciding what to issue
- CPU responsible for scheduling