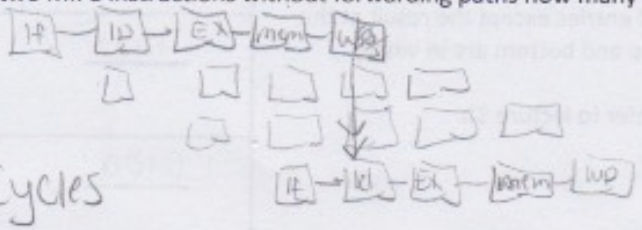
Name Derek Hernandez
netID djh119
(email not long Axxxxx number)

Assignment 4 CS 3339 – Spring 2019
Due: per TRACS Friday @ 11:55pm
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
```



Stall 2 cycles

2) [4 points] To prepare your p3 or p4 project for submission you need to execute the tar command to build a Tape ARchive (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 djh119_Project.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: Create Archive V: Verbose; Print all filenames as they are added
Z: GZipped; GZip file f: use following archive for operation.

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, #define BPRED_SIZE 64 (Preprocessor directive)
Why is the address of the branch shifted by 2?
Its storing 2-bit counters, so PC is incremented by 2-bits.
What does the '%' do?
Its a remainder operation, it will divide (PC_{branch} >>2) by 64 and take the remainder resulting in a 2-bit value.

4) [6 points] Early version of the MIPS processor (without "Microprocessor without Interlocked Pipeline Stages") did not check for data hazards.

How was correct operation achieved?
By seperating dependent instructions with a nop
Why was this technique abandoned?
Very inefficient and time dependent.
What was added to the processor in order to ensure correct operation?
Multiplexers (ALU Control)

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 Prediction | 2 bit prediction | runtime BHT BTB prediction

low high

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
	0x75 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.

LW add

Why does the instruction pairing of *add* and *lw* benefit the hardware implementation?

because they are able to run simultaneously, if they have different registers.

Based on the implementation discussed in class pairing an *add* instruction with an *and* instruction would introduce what type of hazard?

Structural Hazard

How will the compiler resolve this issue?

Stall / bubble.

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 Scheduling: relies on the compilers operation, in which it must sort instructions into VLIW, which are executed together.

Dynamic Scheduling: relies on the CPU to schedule, based on value each instruction is given. CPU will execute instructions out of order but will store to registers in order