CS 3853: Computer Architecture
Midterm Examination
Total Marks: 100
Time: 1 hour 15 min

Name:

Banner Id:

Alias:

1. [20 pt] Indicate **True** or **False** for the following statements:

   - A program takes 100ms to run in system B. If you run it on system A which is 10% faster than system B, the execution time of the program in A would be 110ms.
   - Data dependence can be ignored without causing correctness problem.
   - Throughput is a measure of performance.
   - Geometric mean of x, y and z is $\sqrt{x \times y \times z}$.
   - Execution time is inversely proportional to IPC.
   - Memory stage of MIPS pipeline can raise arithmetic exception.
   - Structural hazard can be eliminated by duplicating various resources.
   - Delay slot instructions are not control dependent on the branch.
   - A multi cycle MIPS pipeline can have WAR hazard.
   - The 5 stage MIPS pipeline discussed in the class uses static scheduling.
   - Pipeline interlock is in charge of hazard detection.

2. [10 pt] Amdahl's law:

   Suppose you are considering adding a vector unit in your processor. When a computation is run on the vector unit, it is 10 times faster than the normal mode of execution.

   (a) [2 pt] Write down Amdahl's law with an explanation of every term used.

   (b) [3 pt] If the percentage of time spent using vector unit is 70, what is the speed up?

   (c) [3 pt] What should be the percentage of time spent using vector unit if we need a speed up of 2?

3. [15 pt] Suppose a program has following information for a machine.

Instruction	Frequency	CPI
Load	15%	1.5
Store	5%	2.5
Branch	25%	0.5
Arithmetic	55%	1

Table 1: System information.

We are considering an optimization that can combine a load and the following arithmetic instruction into a single *load-compute* instruction. The CPI of a load-compute instruction is 1.75. 40% of the original load instructions can be converted into load-compute instructions. The optimization increases the original clock cycle time by 10%.

(a) [2 pt] What is the CPI of the original machine?

(b) [2.5 pt] Fill out the following table considering the optimization.

Instruction	New Frequency
Load	
Store	
Branch	
Arithmetic	
Load-compute	

Table 2: Instruction frequency considering optimization.

(c) [2.5 pt] What is the CPI of the optimized machine?

(d) [2 pt] If the original program has 10 billion instructions, how many instructions does the optimized program has?

(e) [2 pt] If the original machine has a clock cycle time of 1 ns, what is the execution time in the original machine?

(f) [2 pt] What is the execution time in the optimized machine?

(g) [2 pt] Which machine is faster?

4. [35 pt] Pipeline & CPI:

   (a) Let us consider a classic 5 stage MIPS pipeline where the **branch is resolved in ID stage**. Consider the following code fragment:

```
L1: LD R2, 4(R1)
ADD R2, R2, R3
ADD R3, R4, R2
ADDI R5, R5, #-1
SD R3, 0(R10)
BEQZ R5, L1
```

   Assume that the initial value of R5 is 3.

   i. [3 pt] How many times the branch will be taken? How many times it will not be taken?

   ii. [4 pt] List all immediate RAW, WAR and WAW dependences in the code.

iii. [14 pt] Assume that the pipeline has no forwarding support and branches are handled using static *freeze* strategy. Show the pipeline timing diagram of the same code.

Ins	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
LD R2, 4(R1)																					
ADD R2, R2, R3																					
ADD R3, R4, R2																					
ADDI R5, R5, #-1																					
SD R3, 0(R10)																					
BEQZ R5, L1																					
LD R2, 4(R1)																					

(b) [14 pt] Assume that the pipeline has full forwarding support and branches are handled using static *predict not taken* strategy. Show the pipeline timing diagram of the same code.

Ins	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
LD R2, 4(R1)																					
ADD R2, R2, R3																					
ADD R3, R4, R2																					
ADDI R5, R5, #-1																					
SD R3, 0(R10)																					
BEQZ R5, L1																					
LD R2, 4(R1)																					

Table 3: The last LD is the instruction from the target of the branch.

5. [20 pt] Assume our 5 stage MIPS pipeline. Let us assume that the branch is resolved in the EX stage. Assume that 25% of the branches are unconditional. 80% of the conditional branches are taken, and unconditional branches are always taken.

   (a) Assume *predict taken* policy for branches.

      i. [2 pt] What is the number of stall cycles due to an unconditional branch?

      ii. [2 pt] What is the number of stall cycles due to a conditional branch that is taken?

      iii. [2 pt] What is the number of stall cycles due to a conditional branch that is *not* taken?

      iv. [4 pt] What is the branch penalty (i.e., expected number of stall cycles for a branch)?

   (b) [10 pt] Calculate similar numbers for *predict not taken* policy.