CS 282 – Computer Architecture and Organization – Assignment 2 (50 points)

Q1. (15 pts) Use the following code fragment:

```
Loop:  LD     R1, 0(R2)       ; load R1 = MEM[0+R2]
       DADDI  R1, R1, 1       ; R1 = R1+1
       SD     0(R2), R1       ; store MEM[0+R2] = R1
       DADDI  R2, R2, 4       ; R2 = R2+4
       DSUB   R4, R3, R2      ; R4 = R3-R2
       BNE    R4, R0, Loop    ; branch to Loop if (R4 != 0)
```

Assume that the initial value of R3 is R2 + 396.

Throughout this exercise use the classic 5-stage MIPS pipeline and assume that all memory accesses take 1 clock cycle.

a. (5 pts) Show the timing of this instruction sequence for the RISC pipeline *without* any forwarding hardware, but assuming that a register read and a write in the same clock cycle forwards through the register file. Use a pipeline timing chart, such as Figure A.5. Assume that branch is handled by flushing the pipeline and that the PC is updated at the end of the execute stage. How many clock cycles does this loop take to execute (for all iterations)?

b. (5 pts) Show the timing of this instruction sequence for the RISC pipeline with normal forwarding and bypassing hardware. Assume that the branch instruction is handled by flushing the next instruction only if the branch is taken and the PC is updated at the end of the decode stage, how many cycles does this loop take to execute?

c. (5 pts) Assume that the RISC pipeline with a single-cycle delayed branch and normal forwarding. Schedule the instructions in the loop including the branch delay slot. You may reorder instructions and modify the register operands, but do not change the instruction opcodes or number of instructions. Show a pipelining timing diagram and compute the number of cycles needed to execute the entire loop.

Q2. (5 pts) Suppose the branch frequencies (as percentages of all instructions) are as follows:

Conditional branches	15%
Jumps and Calls	1%

60% of Conditional branches are taken

We are examining a 4-stage pipeline (IF, ID, EX, WB) where the branch is resolved (PC updated) at the end of the second (ID) stage for unconditional jumps and at the end of the third (EX) stage for conditional branches. Assuming that conditional branches are predicted to be not taken (you don't flush the pipeline if the branch is not taken), how much faster would the processor be without any branch hazards?

Q3. (20 pts) In this problem, we will explore a pipeline for register-memory architecture. We will now add support for register-memory ALU operations to the classic 5-stage pipeline. All memory addressing will be restricted to register indirect. For example, the register-memory instruction ADD R4, R5, (R1) means ADD R4 = R5 + MEM(R1). Register-register ALU operations are unchanged.

a. (4 pts) List a rearranged order of the five traditional stages of the RISC pipeline that will support register-memory operations implemented exclusively by register indirect addressing.

b. (4 pts) Describe what new forwarding paths are needed for the rearranged pipeline by stating the source, destination, and information transferred on each needed new path.

c. (4 pts) For the reordered stages of the RISC pipeline, what new hazards are created by this addressing mode? Give an instruction sequence illustrating each new hazard.

d. (4 pts) List all ways that the RISC pipeline with register-memory ALU operations can have a different instruction count for a given program than the original RISC pipeline. Give a pair of specific instruction sequences, one for the original pipeline and one for the rearranged pipeline, to illustrate each way.

e. (4 pts) Assume that all instructions take 1 cycle per stage. List all ways that the register-memory RISC can have different CPI for a given program as compared to the original RISC pipeline.

Q4. (10 pts) Scheduling branch delay slots (see Figure A.14) can improve performance. Assume a single branch delay slot and an instruction execution pipeline that determines branch outcome in the second stage.

a. (5 pts) For a delayed branch instruction, what is the penalty for each branch delay slot scheduling scheme if the branch is taken and if it is not taken, and what condition, if any, must be satisfied to ensure correct execution?

b. (5 pts) A *cancel-if-not-taken* branch instruction (also called *branch likely* and implemented in MIPS) does not executed the instruction in the delay slot if the branch is not taken. Thus, a compiler need not be as conservative when filling the delay slot. For each branch delay slot scheduling scheme, what is the penalty if the branch is taken and if it is not taken, and what condition, if any, must be satisfied to ensure correct execution?