# COE 501: Computer Architecture

## Problem Set 4: Pipelining Basic and Intermediate Concepts

**1)** (15 pts) Use the following code fragment:

```
I1:     LD      R1, 0(R2)          ; Load R1 = Memory(R2)
I2:     DADDI   R1, R1, 1          ; R1 = R1 + 1
I3:     SD      R1, 0(R2)          ; Store Memory(R2) = R1
I4:     DADDI   R2, R2, 8          ; R2 = R2 + 8
I5:     DADDI   R4, R4, -1         ; R4 = R4 – 1
I6:     BNE     R4, R0, I1         ; Branch if R4 != 0
```

Assume that the initial value of R4 is 100.

a) (2 pts) List all the true data dependences in the code above within one loop iteration. Record the register, source instruction, and destination instruction.

b) (4 pts) Show the timing of the above instruction sequence for the 5-stage MIPS pipeline without any forwarding hardware. Use a pipeline timing chart to show all stall cycles. Assume that the branch is handled by predicting it as NOT taken. If the branch outcome is TAKEN, it kills the next two instructions in the pipeline. How many cycles does this loop take to execute? What is the average CPI?

c) (5 pts) Assuming delayed branching, rewrite the above code to take advantage of the branch delay slot. Show the timing of the above instruction sequence for the 5-stage MIPS pipeline with full forwarding hardware. How many cycles does this loop take to execute? What is the average CPI?

d) (4 pts) Cache memory stages sometimes take longer to access than other pipeline stages. Consider a 7-stage pipeline: IF1, IF2, ID, EX, MEM1, MEM2, WB, where instruction fetch is split into two stages: IF1 and IF2, and the data memory is also split into two stages: MEM1 and MEM2. Show the timing of the above instruction sequence for the 7-stage pipeline will full forwarding hardware. Assume that the branch is handled by predicting it as always TAKEN with zero delay in the IF1 stage. How many cycles does this loop take to execute? What is the average CPI?

**2)** (5 pts) Consider the following branch and jump frequencies. Assume there is NO branch target buffer (BTB) in the first stage and that branches and jumps are not resolved until later stages in the pipeline.

Conditional branches = 20%
Unconditional Jumps and Calls = 3%
70% of conditional branches are taken

a) (2 pts) We are examining a 5-stage processor pipeline where the unconditional jump and call instructions are resolved at the end of the second stage, and the conditional branches are resolved at the end of the third stage. Ignoring other pipeline stalls, how much faster would the processor pipeline be without any control hazards?

b) (3 pts) Now assume a 10-stage deep pipeline, where unconditional jumps and calls are resolved at the end of the fourth stage and conditional branches are resolved at the end of the seventh stage. Ignoring other pipeline stalls, how much faster would the processor pipeline be without any control hazards?

**3)** (7 pts) In this problem, we will explore how a deep processor pipeline affects performance in two ways: faster clock cycle and increased stalls due to data and control hazards. Assume that the original processor is a 5-stage pipeline with a 1 ns clock cycle. The second processor is a 12-stage pipeline with a 0.5 ns clock cycle. The 5-stage pipeline experiences one stall cycle due to a data hazard every 5 instructions, whereas the 12-stage pipeline experiences 3 stall cycles every 8 instructions. In addition, branches constitute 20% of the instruction count, and the misprediction rate for both pipelines is 5%.

a) (3 pts) What is the speedup of the 12-stage pipeline over the 5-stage pipeline, taking into account only data hazards?

b) (4 pts) If the branch misprediction penalty is 2 cycles for the 5-stage pipeline, but 6 cycles for the 12-stage pipeline, what are the CPIs of each, taking into account the stalls of the data hazards and branch hazards?

**4)** (13 pts) We will now add support for register-memory ALU operations to the classic five-stage MIPS pipeline. To simplify the problem, all memory addressing will be restricted to register indirect. All addresses are simply a value held in a register. No displacement may be added to the register value. For example, **ADD R4, R5, (R8)** means **R4 = R5 + Memory(R8)**. Only one memory operand can be read, but not written. To write memory, the store instruction should be used instead. Register-register ALU operations are unchanged. For example, the instruction **ADD R4, R5, R8** means **R4 = R5 + R8**.

a) (2 pts) List a rearranged order of the five traditional stages of the MIPS pipeline that will support register-memory operations implemented exclusively by register indirect addressing.

b) (5 pts) Describe what forwarding paths are needed for the rearranged pipeline by stating the source stage, destination stage, and information transferred on each needed new path. Give an instruction sequence showing each data hazard that can be resolved by forwarding data between stages. Draw a timing diagram showing the forwarding between stages.

c) (3 pts) For the reordered stages of the pipeline, what data hazards cannot be forwarded and cause stall cycles? Give an instruction sequence showing each data hazard that causes stall cycles. Draw a timing diagram showing the stall cycles caused by each data hazard.

d) (1 pts) What is the penalty of the branch instruction in the new pipeline?

e) (2 pts) List all of the ways that the new pipeline with register-memory ALU operations can have a different instruction count for a given program than the original pipeline (that supports register-register ALU operations only). Give specific instruction sequences, one for the original pipeline and one for the rearranged pipeline, to illustrate each way.