CSE 141: Introduction to Computer Architecture Spring 2024

Homework 2

Instructor: Jishen Zhao **Due on:** May 10, 2024 *(42 points)*

Name: Andrew Onozuka
PID: A16760043
Email: ronozuka@ucsd.edu

Instructions

- The homework must be submitted to Gradescope by 11:59pm. Anything later is a late submission
- Handwritten or typed responses are accepted.
- All responses must be neat and legible. Illegible answers will result in zero points.
- Provide details on how to reach a solution. An answer without explanation gets no credit. Clearly state all assumptions.

1. **Branch Prediction (8 points)**
   Assume the following instruction mix for a 5-stage MIPS pipeline:

Instruction	Proportion
Load	20%
Store	15%
Arithmetic	35%
Branch	30%
Total	100%

   The processor's base CPI = 1. The branch is always predicted as not taken. 55% of the branches are taken. 50% of the load instructions are immediately followed by an instruction that uses the loaded value. There are no other stalls in the pipeline. Branch misprediction has a 4 cycle penalty. Stalling due to a load instruction has a 2 cycle overhead. Calculate the CPI of this pipeline.

   Mis-prediction rate = 55% of branches = $0.55 \times 30\% = 16.5\%$.
   Load penalty rate = 50% of loads = $0.5 \times 20\% = 10\%$.

   CPI = Base CPI + Branch misprediction penalty + Load stall penalty
   CPI = $1 + (0.165 \times 4) + (0.10 \times 2)$
   CPI = $1 + 0.66 + 0.20$
   CPI = 1.86

   The CPI of the pipeline is **1.86**.

2. **Pipelining(15 points)**
   Assume the following program is running on the 5-stage in-order pipeline processor shown in class. All registers are initialized to 0. Assuming only WX(WP0) and WD (register file internal forwarding) forwarding, branches are resolved in **Decode** stage, and branches are always predicted **not-taken**. How many cycles will it take to execute the program, if the branch outcome is **actually taken**? Draw a pipeline diagram (table) to show the details of your work. Use arrows to indicate forwarding.

```
lw $r6 0($r10)
lw $r7 0($r11)
add $r2 $r6 $r7
beq $r2 $r3 label
sub $r6 $r8 $r4
```

```
      sw $r6 0($r10)
label:lw $r1 0($r2)
      or $r4 $r2 $r1
      mul $r5 $r4 $r9
```

Base Cycles: Instructions count = 9 (1 cycle for each initiation)
Stalls: Load-use = 2 cycles after each lw (total 6 for three lw's)
Branch Misprediction: The branch is taken, so we add misprediction penalty = 4 cycles

For a total of 9 (instructions) + 6 (stalls) + 4 (misprediction).

The pipelining takes **19 cycles**.

3. **MIPS ISA(10 points)**

   (a) Write the MIPS instructions for the code given below. The values of u,v,x and y need to be in R5, R6, R7 and R8 registers respectively at the end of the code.(Note: Consider R0 as the zero register)(4 points).
   u=1
   v=1
   x = u+v
   y = 1+x
   x = y-u
   u = x
   v = y

addi $r5, $r0, 1	**0 + 1 into R5**
addi $r6, $r0, 1	**0 + 1 into R6**
add $r7, $r5, $r6	**R5 + R6 into R7**
addi $r8, $r7, 1	**R7 + 1 into R8**
sub $r7, $r8, $r5	**R8 - R5 into R7**
add $r5, $r7, $r0	**R7 + 0 into R5**
add $r6, $r8, $r0	**R8 + 0 into R6**

   (b) What are the potential hazards that would occur in the above question if no stalls were present. Explain why it would be a hazard with pipeline diagram?(6 points)

   Some potential hazards are:
   - Between addi $r5, $r0, 1 and sub $r7, $r8, $r5, there's a use of $r5 right after it's defined. In a pipelined architecture without forwarding or with only partial forwarding, this can lead to a hazard.
   - Between add $r7, $r5, $r6 and addi $r8, $r7, 1, where $r7 is used right after being written.
   - Between addi $r8, $r7, 1 and sub $r7, $r8, $r5, where $r8 is used right after being written.

   Looking at the second example:
   Cycle 1: IF
   Cycle 2: ID
   Cycle 3: EX
   Cycle 4: MEM
   Cycle 5: WB (R7 is written)
   For addi $r8, $r7, 1:
   Cycle 1: IF
   Cycle 2: ID (needs R7)
   Without forwarding, the second instruction needs to wait until R7 is written back, causing a stall.

4. **Critical Path(9 points)**
   For these problems, consider the figure of the datapath , and the given table which shows the delay of micro-architectural components in the datapath. Calculate the time it takes to execute the following instructions. You may assume that the delay of anything not mentioned in the table (including wires and ALU control) is zero.

`datapath.png`

Unit	Instruction Memory	Register Read/Write	ALU	Data Memory	Mux	Add	Control Unit
Delay(ps)	290	140	180	290	50	70	140

- Load Word (e.g., lw \$t1, 8(\$t0))
- Store (e.g., store \$t0, 12(\$t1))
- And (e.g., and \$t2, \$t0, \$t1)

Load Word
Critical path for lw:
IF: Instruction Memory = 290 ps
ID: Register Read + Control Unit = 140 ps + 140 ps = 280 ps
EX: ALU (address computation) = 180 ps
MEM: Data Memory access = 290 ps
WB: Register Write = 140 ps
Total time = 290 + 280 + 180 + 290 + 140 = **1180 ps**

Store
Critical path for store:
IF: Instruction Memory = 290 ps
ID: Register Read + Control Unit = 280 ps
EX: ALU (address computation) = 180 ps
MEM: Data Memory access = 290 ps
WB: No write back for store
Total time = 290 + 280 + 180 + 290 = **1040 ps**

And
Critical path for and:
IF: Instruction Memory = 290 ps
ID: Register Read + Control Unit = 280 ps
EX: ALU (operation itself) = 180 ps
MEM: No memory access for and
WB: Register Write = 140 ps
Total time = 290 + 280 + 180 + 140 = **890** ps