CSE 141: Introduction to Computer Architecture Spring 2024

Homework 2

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

Name: ______________________
PID: ______________________
Email: ______________________

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.

Solution:
Each stall in your instruction increases the CPI of that instruction by one. Since CPI is an average, repeated occurrences of stalls do not contribute any further to CPI.
CPI = Base CPI + Stalls
In this system, we assume bypassing is implemented. Therefore, the only stalls are due to branch mispredictions and load-use latency.
CPI = Base CPI + Load Stalls + Branch Stalls
Loads comprise 20% of instructions and 50% of the time they cause a 2-cycle stall. Branches comprise 30% of instructions and 55% of the time they mispredict, which causes a 4-cycle penalty.
CPI = 1 + (0.20 * 0.50 * 2) + (0.30 * 0.55 * 4)
CPI = 1 + 0.2 + 0.66 = 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 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
```

Solution
Therefore, the total number of cycles to execute the program = 19

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
lw $r6 0($r10)	F	D	X	M	W															
lw $r7 0($r11)		F	D	X	M	W ↓														
add $r2 $r6 $r7			F	D	D	X	M	W ↓												
beq $r2 $r3 label				F	F	D	D	D	X	M	W									
~~sub $r6 $r8 $r4~~						F	F	F	--	--	--	--								
~~sw $r6 0($r10)~~									--	--	--	--	--							
label: lw $r1 0($r2)									F	D	X	M	W ↓							
or $r4 $r2 $r1										F	D	D	X	M	W ↓					
mul $r5 $r4 $r9											F	F	D	D	P0	P1	P2	P3	W	

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

Solution:

li R5, 1
li R6, 1
add R7, R5, R6
addi R8, R7, 1
sub R7, R8, R5
mov R5, R7
mov R6, R8

Also accepted solution:
addi R5, R0, 1
addi R6, R0, 1
add R7, R5, R6
addi R8, R7, 1
sub R7, R8, R5
add R5, R7, R0
add R6, R8, R0

(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 a pipeline diagram?(6 points)
Solution:
From the pipeline diagram, we can observe that without stall there are about 5 data hazards due multiple data dependencies.

- R5 value is available only after 5th cycle, but the add instruction is accessing R5 value in 5th cycle

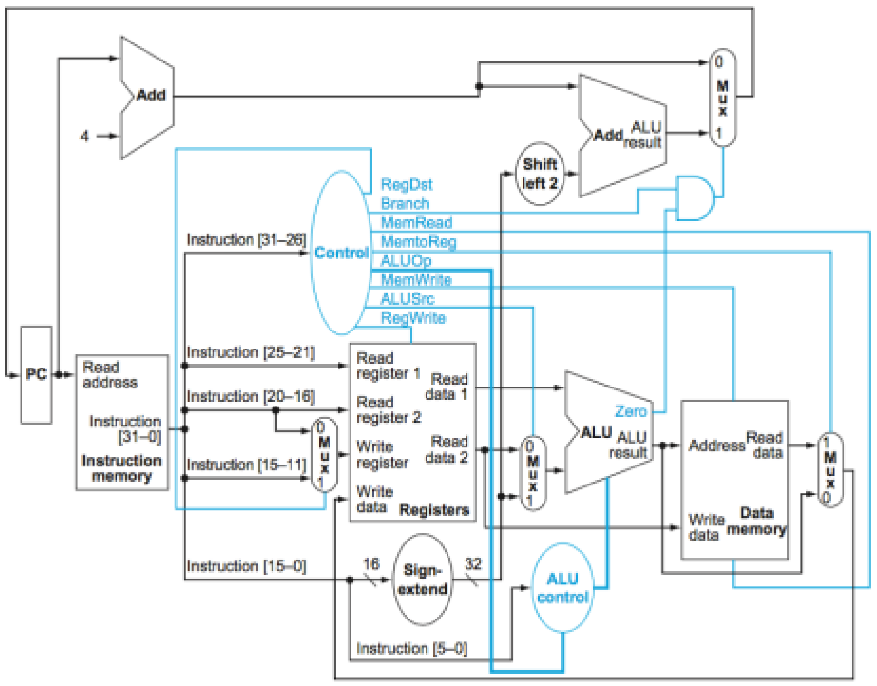
- R6 value is available only after 6th cycle, but the add instruction is accessing R6 value in 5th cycle
- R7 value is available only after 7th cycle, but the addi instruction is accessing R7 value in 6th cycle
- R8 value is available only after 8th cycle, but the sub instruction is accessing R8 value in 7th cycle
- R7 value is available only after 9th cycle, but the mov instruction is accessing R7 value in 9th cycle

There are no control hazards as there are no branch instructions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
li R5,1	F	D	X	M	W															
li R6,1		F	D	X	M	W														
add R7, R5, R6			F	D	X	M	W													
addi R8, R7, 1				F	D	X	M	W												
sub R7, R8, R5					F	D	X	M	W											
mov R5, R7						F	D	X	M	W										
mov R6, R8							F	D	X	M	W									

Data Dependencies	→
Data hazards	—

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.



- Load Word (e.g., lw $t1, 8($t0))

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

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

Solution:

- Load Word (e.g., lw \$t1, 8(\$t0))
  Time taken = Instruction Memory + Control Unit + MUX + ALU + Data Memory + MUX + Register Write
  = 290 + 140 + 50 + 180 + 290 + 50 + 140 = 1140 ps
- Store (e.g., store \$t0, 12(\$t1))
  Time taken = Instruction Memory + Control Unit + MUX + ALU + Data Memory
  = 290 + 140 + 50 + 180 + 290 = 950 ps
- And (e.g., and \$t2, \$t0, \$t1)
  Time taken = Instruction Memory + Register Read + MUX + ALU + MUX + Register Write
  = 290 + 140 + 50 + 180 + 50 + 140 = 850 ps