CS103 – Monsoon 2018 — Homework 6

Neeraj Pandey

Collaborators: None

Question 1: Consider an in-order 5-stage pipeline similar to the one discussed in class. First assume that the pipeline does not support bypassing (forwarding). What are the stall cycles introduced between the following pairs of back-to-back instructions? Then, solve the same problem while assuming support for bypassing. Clearly show your work, i.e., show how each instruction goes through the 5 stages, indicate the point of production and point of consumption, show how the consuming instruction is held back in the D/R stage when there are stalls. Recall that a register read is performed in the second half of the D/R stage and a register write is performed in the first half of the RW stage.

(a) lw $\$1, 8(\$2)$
add $\$4, \$1, \$3$
Solution (a):
I1 = lw $\$1, 8(\$2)$
I2 = add $\$4, \$1, \$3$

Without Bypassing

C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	I2							
D/R	D/R I1	D/R I2	D/R I2	D/R I2	D/R	D/R	D/R	D/R
ALU	ALU	ALU I1	ALU	ALU	ALU I2	ALU	ALU	ALU
DM	DM	DM	DM I1	DM	DM	DM I2	DM	DM
RW	RW	RW	RW	RW I1	RW	RW	RW I2	RW

Stall cycles = C3 and C4

With Bypassing

C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	I2							
D/R	D/R I1	D/R I2	D/R I2	D/R	D/R	D/R	D/R	D/R
				POC				
ALU	ALU	ALU I1	ALU	ALU I2	ALU	ALU	ALU	ALU
DM	DM	DM	DM I1	DM	DM	DM I2	DM	DM
			POP					
RW	RW	RW	RW	RW I1	RW	RWI I2	RW	RW

Stall cycle = 1 for I2

(b) lw $\$1, 8(\$2)$
sw $\$3, 8(\$1)$

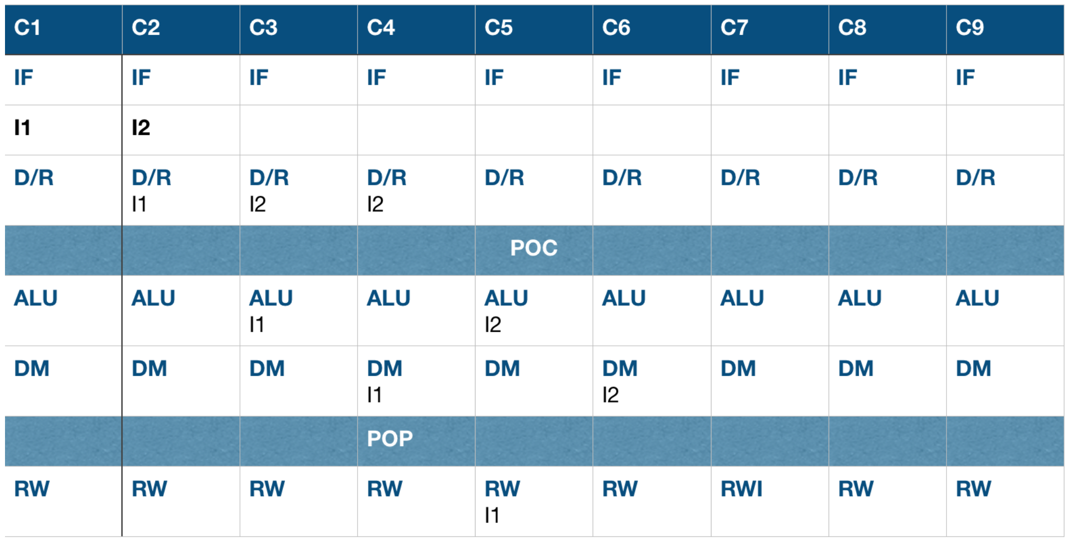
Solution (b) : I1 = lw $\$1, 8(\$2)$
I2 = sw $\$3, 8(\$1)$

Without Bypassing

C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	I2							
D/R	D/R I1	D/R I2	D/R I2	D/R I2	D/R	D/R	D/R	D/R
ALU	ALU	ALU I1	ALU	ALU	ALU I2	ALU	ALU	ALU
DM	DM	DM	DM I1	DM	DM	DM I2	DM	DM
RW	RW	RW	RW	RW I1	RW	RWI	RW	RW

Stall cycles = C3 and C4

With Bypassing



C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	**I2**							
D/R	D/R I1	D/R I2	D/R I2	D/R	D/R	D/R	D/R	D/R
				POC				
ALU	ALU	ALU I1	ALU	ALU I2	ALU	ALU	ALU	ALU
DM	DM	DM	DM I1	DM	DM I2	DM	DM	DM
			POP					
RW	RW	RW	RW	RW I1	RW	RWI	RW	RW

Stall cycle = 1 for I2

Question 2: Consider an in-order pipeline that has the same stages as a 5 stage pipeline discussed in class. However, unlike the 5-stage pipeline discussed in class, a register read in this design takes an entire cycle and a register write takes an entire cycle (and not a half cycle). Use the same sequence of instructions from Question 1, and show how these sequences proceed through the pipeline. Indicate the number of stall cycles that are experienced by each sequence. Show your work for both versions of the pipeline: one without bypassing, and another one with bypassing.

Solution 2(a):

I1 = lw $\$1, 8(\$2)$

I2 = add $\$4, \$1, \$3$

Without Bypassing

C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	**I2**							
D/R	D/R I1	D/R I2	D/R I2	D/R I2	D/R I2	D/R	D/R	D/R
ALU	ALU	ALU I1	ALU	ALU	ALU	ALU I2	ALU	ALU
DM	DM	DM	DM I1	DM	DM	DM	DM I2	DM
RW	RW	RW	RW	RW I1	RW	RW	RW	RW I2

Stall cycles = 3 for I2

With Bypassing

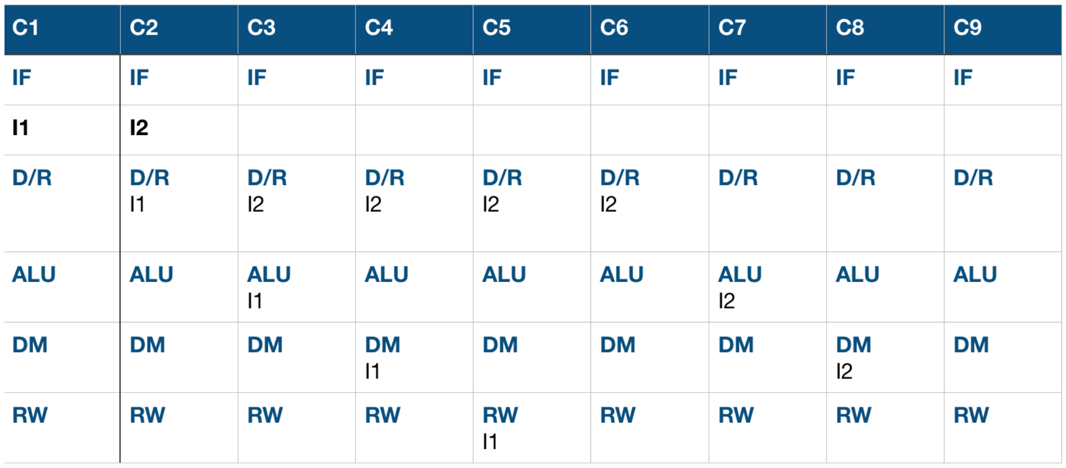
C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	**I2**							
D/R	D/R I1	D/R I2	D/R I2	D/R	D/R	D/R	D/R	D/R
				POC				
ALU	ALU	ALU I1	ALU	ALU I2	ALU	ALU	ALU	ALU
DM	DM	DM	DM I1	DM	DM I2	DM	DM	DM
			POP					
RW	RW	RW	RW	RW I1	RW	RW I2	RW	RW

Stall cycles = 1 for I2

Solution 2(b):

I1 = lw $\$1, 8(\$2)$
I2 = sw $\$3, 8(\$1)$

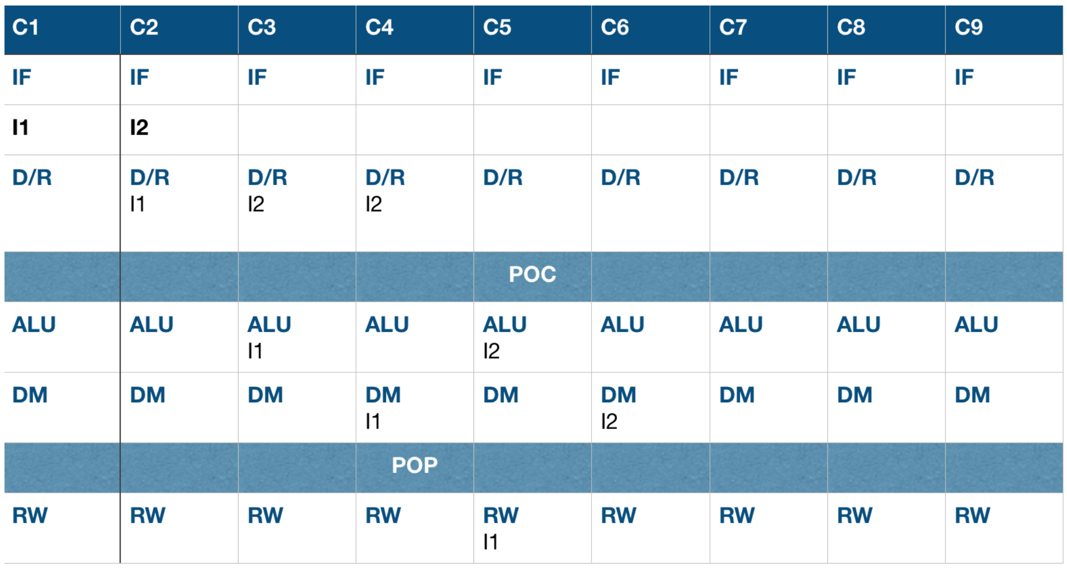
Without Bypassing



C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	I2							
D/R	D/R I1	D/R I2	D/R I2	D/R I2	D/R I2	D/R	D/R	D/R
ALU	ALU	ALU I1	ALU	ALU	ALU	ALU I2	ALU	ALU
DM	DM	DM	DM I1	DM	DM	DM	DM I2	DM
RW	RW	RW	RW	RW I1	RW	RW	RW	RW

Stal cycles = 3 for I2

With Bypassing



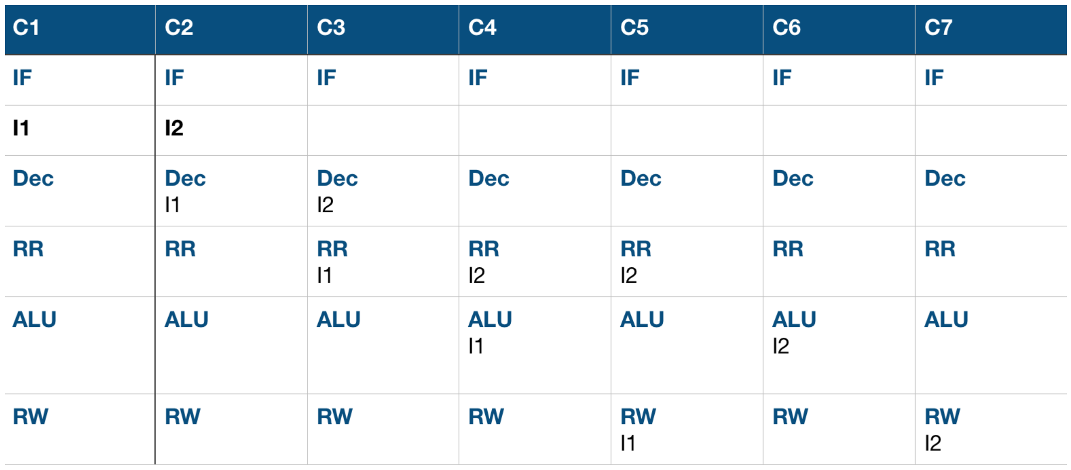
C1	C2	C3	C4	C5	C6	C7	C8	C9
IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	I2							
D/R	D/R I1	D/R I2	D/R I2	D/R	D/R	D/R	D/R	D/R
				POC				
ALU	ALU	ALU I1	ALU	ALU I2	ALU	ALU	ALU	ALU
DM	DM	DM	DM I1	DM	DM I2	DM	DM	DM
			POP					
RW	RW	RW	RW	RW I1	RW	RW	RW	RW

Stall cyles = 1 for I2

Question 3: Assume a new pipeline design as shown in the pipeline diagram below. This is an example of a "bifurcated" pipeline that we have discussed in class. After instruction fetch, the instruction goes through a separate Decode stage where dependences are analyzed, then a separate RegRead stage where input operands are read from the register file. After this, an instruction takes one of two possible paths. Int-adds go through the stages labeled "IntALU" and "RegWrite". Loads/stores go through the stages labeled "IntALU", "DataMem", "DataMem", and "RegWrite", i.e., it takes two cycles to retrieve data from the data memory unit. How many stall cycles are introduced between the following pairs of successive instructions (i) for a processor with no register bypassing and (ii) for a processor with full bypassing?

(a) add $\$1, \$2, \$3$
add $\$4, \$1, \$5$
Solution (a):
I1 = add $\$1, \$2, \$3$
I2 = add $\$4, \$1, \$5$

Without Bypassing



C1	C2	C3	C4	C5	C6	C7
IF	IF	IF	IF	IF	IF	IF
I1	I2					
Dec	Dec I1	Dec I2	Dec	Dec	Dec	Dec
RR	RR	RR I1	RR I2	RR I2	RR	RR
ALU	ALU	ALU	ALU I1	ALU	ALU I2	ALU
RW	RW	RW	RW	RW I1	RW	RW I2

Stall cyles = 1 for I2

With Bypassing

C1	C2	C3	C4	C5	C6
IF	IF	IF	IF	IF	IF
I1	**I2**				
Dec	Dec I1	Dec I2	Dec	Dec	Dec
RR	RR	RR I1	RR I2	RR	RR
				POC	
ALU	ALU	ALU	ALU I1	ALU I2	ALU
			POP		
RW	RW	RW	RW	RW I1	RW I1

Stall cycles $= 1$ for I2

(b) lw $\$1, 8(\$2)$
lw $\$3, 8(\$1)$

Solution (b):

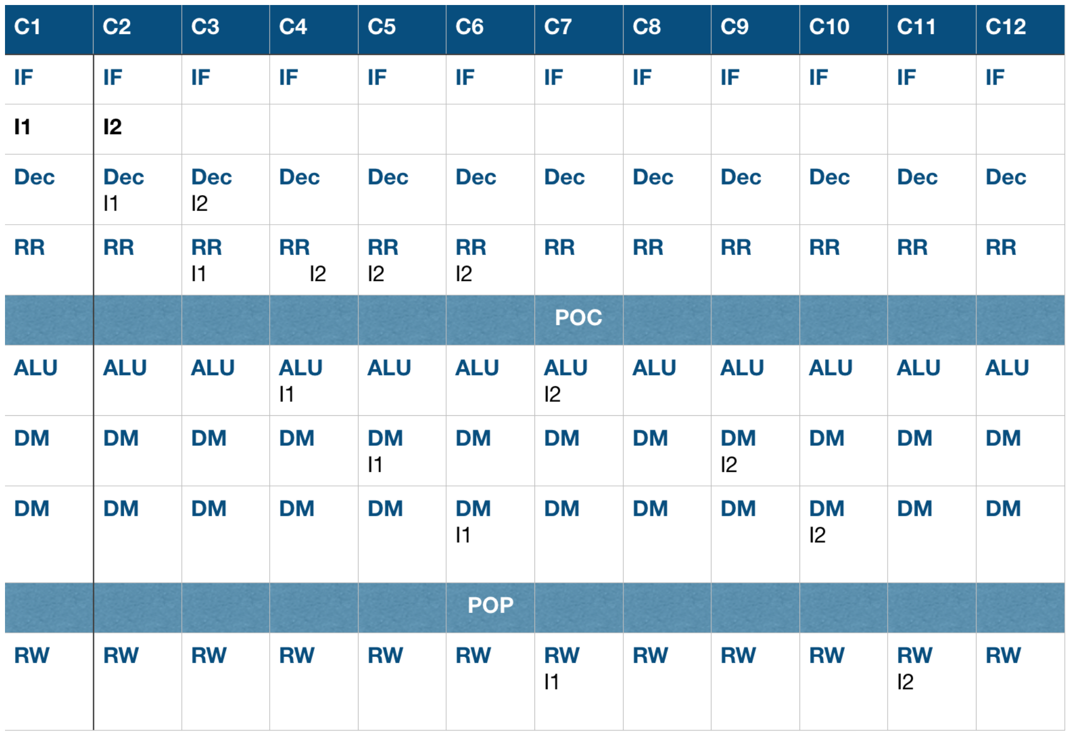
I1 $=$ lw $\$1, 8(\$2)$
I2 $=$ lw $\$3, 8(\$1)$

Without Bypassing

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	**I2**										
Dec	Dec I1	Dec I2	Dec	Dec	Dec	Dec	Dec	Dec	Dec	Dec	Dec
RR	RR	RR I1	RR I2	RR I2	RR I2	RR I2	RR	RR	RR	RR	RR
ALU	ALU	ALU	ALU I1	ALU	ALU	ALU	ALU I2	ALU	ALU	ALU	ALU
DM	DM	DM	DM	DM I1	DM	DM	DM	DM I2	DM	DM	DM
DM	DM	DM	DM	DM	DM I1	DM	DM	DM	DM I2	DM	DM
RW	RW	RW	RW	RW	RW	RW I1	RW	RW	RW	RW I2	RW

Stall cycle = 3 for I2

With Bypassing



C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF
I1	I2										
Dec	Dec I1	Dec I2	Dec	Dec	Dec	Dec	Dec	Dec	Dec	Dec	Dec
RR	RR	RR I1	RR I2	RR I2	RR I2	RR	RR	RR	RR	RR	RR
						POC					
ALU	ALU	ALU	ALU I1	ALU	ALU	ALU I2	ALU	ALU	ALU	ALU	ALU
DM	DM	DM	DM	DM I1	DM	DM	DM	DM I2	DM	DM	DM
DM	DM	DM	DM	DM	DM I1	DM	DM	DM	DM I2	DM	DM
					POP						
RW	RW	RW	RW	RW	RW	RW I1	RW	RW	RW	RW I2	RW

Stall cycles = 2 for I2

Question 4: Consider a program that executes a large number of instructions. Assume that the program does not suffer from stalls from data hazards or structural hazards. Assume that 20% of all instructions are branch instructions, and 75% of these branch instructions are Taken. What is the average CPI for this program when it executes on each of the processors listed below? All of these processors implement an 10-stage pipeline and resolve a branch outcome at the end of the 4th stage. The 1st stage fetches an instruction, the 2nd stage does decode, the 3rd stage does register read, and the 4th stage does the computations for the branch.

(a) The processor pauses instruction fetch as soon as it fetches a branch. Instruction fetch is resumed after the branch outcome has been resolved.
Solution (a): The next instruction is called for after the fourth stage because the processor will not call any instruction unless the outcome is resolved. Cycles lost: 0.2 x 3 = 0.6 times the initial instructions.
Average CPI $= \frac{\text{Number of cycles to execute the program}}{\text{Initial instructions}}$
$\frac{1.6k}{k}$; for k + 0.6k initial instructions $\implies$ 1.6

(b) The processor always fetches instructions sequentially. If a branch is resolved as Taken, the incorrectly fetched instructions after the branch are squashed.
Solution (b): Here we need to squash the three instructions because the branch is resolve after the fourth stage.
Cycles lost: 0.15 x 3 = 0.45 times the initial instructions
Average CPI $= \frac{\text{Number of cycles to execute the program}}{\text{Initial instructions}}$
$\frac{1.45k}{k}$; for k + 0.45k initial instructions $\implies$ 1.45

(c) The processor implements three branch delay slots. The compiler fills the branch delay slots with three instructions that come before the branch in the original code.
Solution (c): Here, Average CPI = 1 as we don't loose any additional cycles for the instructions because the compiler fills up the delay shots with the instructions.

(d) The processor does not implement branch delay slots. Instead, it implements a hardware branch predictor that makes correct predictions for 90% of all branches. When an incorrect prediction is discovered, the incorrectly fetched instructions after the branch are squashed.

Solution (d): Here, 10% predictions are incorrect, which means we are loosing 0.1 x 0.2 = 0.02 times the initial instructions.
Instructions to squash: 0.02 x 3 = 0.06 times the initial instructions
Average CPI $= \frac{\text{Number of cycles to execute the program}}{\text{Initial instructions}}$
$\frac{1.06k}{k}$; for k + 0.06k initial instructions $\implies$ 1.06

Question 5: Assume that you have a 2 bit saturating counter (with the initial value of 11). Every time the branch is taken, the counter is incremented, and is decremented every time the branch is not taken.

(a) Show the change in the counter for the following sequence of 9 events: T,N,N,T,N,N,T,N,N (T stands for taken and N stands for not taken, and these are the actual outcomes of the branch). (5 points)
Solution (a): Total events = 9
Intitial value = 11
T = 11
N = 10
N = 01
T = 10
N = 01
N = 00
T = 01
N = 00
N = 00

(b) What will the prediction be for the 10th sequence? Assume the predictions for the 2 bit predictor as discussed in class
Solution (b): 10th sequence will be predicted as not taken.

If you have a 2-bit bimodal predictor with 4096 entries, how many bits from the Branch PC will be required to index entries in this table? What is the total storage capacity of the table? If we change the predictor to be a 1 bit bimodal predictor, while keeping constant the number of table entries, how many bits from the Branch PC will be required to index into the table?

Solution: Given: 2-bit bi-modal predictor with 4096 entries.
So, we require 12 bits from the Branch PC ($2^!2 = 4096$)
Storage capacity = 4096 * 2 = 1024 bytes