CS2100 Computer Organisation

Tutorial #10: Pipelining

Answers

Tutorial Questions

1. Given the following three formulas (See Lecture #20, Section 5 Performance):

$$CT_{seq} = \sum_{k=1}^{N} T_k$$

$$CT_{pipeline} = \max(T_k) + T_d$$

$$Speedup_{pipeline} = \frac{CT_{seq} \times InstNum}{CT_{pipeline} \times (N + InstNum - 1)}$$

For each of the following processor parameters, calculate CT_{seq}, $CT_{pipeline}$ and $Speedup_{pipeline}$ (to two decimal places) for 10 instructions and for 10 million instructions.

	Stages Timing (for 5 stages, in ps)	Latency of pipeline register (in ps)
a.	300, 100, 200, 300, 100 (slow memory)	0
b.	200, 200, 200, 200, 200	40
c.	200, 200, 200, 200, 200 (ideal)	0

Answers:

	CT_{seq}	**$CT_{pipeline}$**	**Speedup (10 inst)**	**Speedup (10m inst)**
a.	**1000ps**	**300ps**	(1000×10)/(300×14) **= 2.38**	(1000×10m)/(300 (10m+4)) **= 3.33**
b.	**1000ps**	**240ps**	(1000×10)/(240×14) **= 2.98**	(1000×10m)/(240×(10m+4)) **= 4.17**
c.	**1000ps**	**200ps**	(1000×10)/(200×14) **= 3.57**	(1000×10m)/(200×(10m+4)) **= 5.00**

2. [AY2014/5 Semester 2 Exam]
   Refer to the following MIPS program:

```
     # register $s0 contains a 32-bit value
     # register $s1 contains a non-zero 8-bit value
     #       at the right most (least significant) byte
     add  $t0, $s0, $zero     #inst A
     add  $s2, $zero, $zero   #inst B
lp:  bne  $s2, $zero, done    #inst C
     beq  $t0, $zero, done    #inst D
     andi $t1, $t0, 0xFF      #inst E
     bne  $s1, $t1, nt        #inst F
     addi $s2, $s2, 1         #inst G
nt:  srl  $t0, $t0, 8         #inst H
     j    lp                  #inst J
done:
```

We assume that the register **$s0** contains **0xAFAFFAFA** and **$s1** contains **0xFF**.

Given a 5-stage MIPS pipeline processor, for each of the parts below, give the total number of cycles needed for the first iteration of the execution from instructions **A** to **H** (i.e. excluding the "**j lp**" instruction). Remember to include the cycles needed for instruction **H** to finish the WB stage. Note that the questions are independent from each other.

a. With only data forwarding mechanisms and no control hazard mechanism.

b. With data forwarding and "assume not taken" branch prediction. Note that there is no early branching.
   [Recall that early branching means branch decision is made at stage 2 (Decode stage); no early branch means branch decision is made at stage 4 (Memory stage).]

c. By swapping two instructions (from Instructions **A** to **H**), we can improve the performance of **early branching (with all additional forwarding paths)**. Give the two instructions that can be swapped. You only need to indicate the instruction letters in your answer.

Give the total number of cycles needed for the execution of the whole code in the worst case for each of the following assumptions. You may assume that the jump instruction (**j**) computes the address of the instruction to jump to in the MEM stage.

d. With only data forwarding mechanisms and no control hazard mechanism.

e. With data forwarding and "assume not taken" branch prediction. Note that there is no early branching.

Answers:

(a) 20 cycles

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
add	F	D	E	M	W															
add		F	D	E	M	W														
bne			F	D	E	M	W													
beq							F	D	E	M	W									
andi											F	D	E	M	W					
bne												F	D	E	M	W				
~~addi~~		The addi instruction is not executed.																		
srl																F	D	E	M	W

(b) 14 cycles

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
add	F	D	E	M	W									
add		F	D	E	M	W								
bne			F	D	E	M	W							
beq				F	D	E	M	W						
andi					F	D	E	M	W					
bne						F	D	E	M	W				
addi							F	D	E	*	*			
srl								F	D	*	*	*		
j									F	*	*	*	*	
srl										F	D	E	M	W

Cost of wrong prediction

(c) Swap instructions A and B to reduce the delay between instructions B and C.

```
      add  $s2, $zero, $zero    #inst B
      add  $t0, $s0, $zero      #inst A
lp:   bne  $s2, $zero, done     #inst C
      beq  $t0, $zero, done     #inst D
      andi $t1, $t0, 0xFF       #inst E
      bne  $s1, $t1, nt         #inst F
      addi $s2, $s2, 1          #inst G
nt:   srl  $t0, $t0, 8          #inst H
      j    lp                   #inst J
done:
```

(d)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
add	F	D	E	M	W																
add		F	D	E	M	W															
bne			F	D	E	M	W														
beq							F	D	E	M	W										
andi											F	D	E	M	W						
bne												F	D	E	M	W					
~~addi~~																					
srl																F	D	E	M	W	
j																	F	D	E	M	W
bne																					F

In the worst case, 4 bytes of the data are examined → 4 iterations. The loop from Instructions C to J (one iteration) takes 18 cycles (not counting the WB stage of the j instruction which overlaps with the bne instruction). There are 2 cycles before the first iteration, and 9 cycles for Instructions C and D in the fifth iteration.
Therefore, total = 2 + (4×18) + 9 = **83 cycles**.

(e)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
add	F	D	E	M	W																
add		F	D	E	M	W															
bne			F	D	E	M	W														
beq				F	D	E	M	W													
andi					F	D	E	M	W												
bne						F	D	E	M	W											
addi							F	D	E	*	*										
srl								F	D	*	*	*									
j									F	*	*	*	*								
srl										F	D	E	M	W							
j											F	D	E	M	W						
bne															F						

In the worst case, 4 bytes of the data are examined → 4 iterations. The loop from Instructions C to J (one iteration) takes 12 cycles (not counting the WB stage of the j instruction which overlaps with the bne instruction). There are 2 cycles before the first iteration, and 6 cycles for Instructions C and D in the fifth iteration.
Therefore, total = 2 + (4×12) + 6 = **56 cycles**.

3. [AY2017/8 Semester 2 Exam]

Refer to the MIPS code below. *A* and *B* are integer arrays whose base addresses are in **$s0** and **$s1** respectively. The arrays are of the same size *n* (number of elements). **$s2** contains the value *n*. For this question, we will focus on the code from Instruction 1 onwards.

```
 .data
A: .word 11, 9, 31, 2, 9, 1, 6, 10
B: .word 3, 7, 2, 12, 11, 41, 19, 35
n: .word 8
.text
main: la   $s0, A      # $s0 is the base address of array A
      la   $s1, B      # $s1 is the base address of array B
      la   $t0, n      # $t0 is the addr of n (size of array)
                       # $s2 is the content of n
      beq  $s2, $zero, End   # Inst1
      addi $t8, $s2, -1      # Inst2
      sll  $t8, $t8, 2       # Inst3
Loop: add  $t0, $s0, $t8     # Inst4
      add  $t1, $s1, $t8     # Inst5
      lw   $t2, 0($t0)       # Inst6
      lw   $t3, 0($t1)       # Inst7
      andi $t4, $t3, 3       # Inst8
      addi $t4, $t4, -3      # Inst9
      beq  $t4, $zero, A1    # Inst10
      add  $t2, $t2, $t3     # Inst11
      j    A2                # Inst12
A1:   addi $t2, $t2, 1       # Inst13
A2:   sw   $t2, 0($t0)       # Inst14
      addi $t8, $t8, -8      # Inst15
      slt  $t7, $t8, $zero   # Inst16
      beq  $t7, $zero, Loop  # Inst17
End:
```

Assuming a 5-stage MIPS pipeline system with forwarding and early branching, that is, the branch decision is made at the ID stage. No branch prediction is made and no delayed branching is used. For the jump (**j**) instruction, the computation of the target address to jump to is done at the ID stage as well.

Assume also that the first **beq** instruction begins at cycle 1.

a. Suppose arrays *A* and *B* now each contains 200 positive integers. What is the minimum number and maximum number of instructions executed? (Consider only the above code segment from Inst1 to Inst17.)

b. List out the instructions where some stall cycle(s) are inserted in executing that instruction in the pipeline. These include delay caused by data dependency and control hazard. You may write the instruction number InstX instead of writing out the instruction in full.

c. How many cycles does one iteration of the loop (from Inst1 to Inst17) take if the **beq** instruction at Inst10 branches to *A1*? You have to count until the WB stage of Inst17.

d. How many cycles does one iteration of the loop (from Inst1 to Inst17) take if the **beq** instruction at Inst10 does not branch to *A1*? You have to count until the WB stage of Inst17.

Answers:

The code does this:

```
int i;
for (i=n-1; i>=0; i-=2) {
   if (B[i]%4 == 3)
      A[i] = A[i] + 1;
   else
      A[i] = A[i] + B[i];
}
```

(a) Minimum = 3 + 100 × 12 = 1203

Maximum = 3 + 100 × 13 = 1303

In the loop (Inst4 to Inst17), there are two paths after Inst10: one that skips Inst11 and Inst12, and the other skips Inst13.

(b) Due to control: Inst2, Inst4, Inst11, Inst13

Due to data: Inst8, Inst10, Inst17

(c) 24 cycles (see page 10)

(d) 26 cycles (see page 11)

Q3. (c)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
I1 beq	F	D	E	M	W																									
I2 addi			F	D	E	M	W																							
I3 sll				F	D	E	M	W																						
I4 add					F	D	E	M	W																					
I5 add						F	D	E	M	W																				
I6 lw							F	D	E	M	W																			
I7 lw								F	D	E	M	W																		
I8 andi									F	D		E	M	W																
I9 addi										F	D		E	M	W															
I10 beq A1											F			D	E	M	W													
I11 add																														
I12 J A2																														
I13 A1: addi															F	D	E	M	W											
I14 A2: sw																F	D	E	M	W										
I15 addi																	F	D	E	M	W									
I16 slt																		F	D	E	M	W								
I17 beq																			F		D	E	M	W						

Q3. (d)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
I1 beq	F	D	E	M	W																									
I2 addi			F	D	E	M	W																							
I3 sll				F	D	E	M	W																						
I4 add					F	D	E	M	W																					
I5 add						F	D	E	M	W																				
I6 lw							F	D	E	M	W																			
I7 lw								F	D	E	M	W																		
I8 andi									F	D		E	M	W																
I9 addi										F	D		E	M	W															
I10 beq A1											F			D	E	M	W													
I11 add															F	D	E	M	W											
I12 J A2																F	D	E	M	W										
I13 A1: addi																														
I14 A2: sw																		F	D	W	M	W								
I15 addi																			F	D	E	M	W							
I16 slt																				F	D	E	M	W						
I17 beq																					F		D	E	M	W				