University of Maryland Baltimore County
Department of Computer Science and Electrical Engineering

CMSC 411, Computer Architecture

Assignment #4 Solutions — Due: Thursday 10/26/17 in class

Question 1: *(40 Points)*

Problems in this exercise refer to the following sequence of instructions:

```
LW   $5, -16($5)
SW   $5, -16($5)
ADD $5, $5, $5
```

A) Indicate dependences and their type.

1. There is data dependence between the line 1 and line 2; SW instruction depended on $5 generated by LW
2. There is data dependence between the line 1 and line 3; ADD instruction depended on $5 generated by LW

B) Assume there is not forwarding in this pipeline processor, indicate hazards and add *NOP* instructions to eliminate them.

The code that eliminates these hazards with *NOP* without forwarding is;

LW $5, -16($5)	F	D	X	M	W				
nop									
nop									
SW $5, -16($5)				F	D	X	M	W	
ADD $5, $5, $5					F	D	X	M	W

C) Assume there is full forwarding, indicate hazards and add NOP instructions to eliminate unresolved cases.

LW $5, -16($5)	F	D	X	M	W			
nop								
SW $5, -16($5)			F	D	X	M	W	
ADD $5, $5, $5				F	D	X	M	W

The remaining problem in this exercise assumes the following clock cycle times:

Without forwarding	With full forwarding	With ALU-ALU forwarding only
200 ps	250 ps	220 ps

D) What is the total execution time of this instruction sequence without forwarding and with full forwarding? What is the speed-up achieved by adding full forwarding to a pipeline that had no forwarding?

The total execution time is the clock cycle time multiplies with number of cycles. From the above figures, we get 9 and 8 cycles for without-forwarding and with-forwarding respectively.

Without forwarding; 9 * 200 ps = 1800 ps
With forwarding; 8 * 250 ps = 2000 ps

So that the speedup is (1800/2000) = 0.9

It is slow down due to clock cycle time.

E) What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speed-up over a no-forwarding pipeline?

With ALU-ALU-only forwarding, LW can't forward because it determines the data value in MEM stage (load the value of $5 from address calculation in EX stage), thus it is too late for ALU-ALU forwarding. Thus the number of cycles is the same as first case; 9 cycles

Without forwarding; 9 * 200 ps = 1800 ps
With ALU-ALU forwarding; 9 * 220 ps = 1980 ps

So that the speedup is (1800/1980) = 0.91

Again, it is slow down due to clock cycle time.

Question 2: *(30 Points)*

For this problem, assume that all branches are resolved in ID stage and are perfectly predicted (this eliminates all control hazards). For the following fragment of MIPS code:

```
        LW  $5, -16($5)
        SW   $4, -16($4)
        LW   $3, -20($4)
        BEQ $2, $0, Label    ;Assume $2 ≠ $0
        ADD $1, $5, $4
Label:  SUB $2, $1, $3
```

If we only have one memory (for both instruction and data), there is a structural hazard every time we need to fetch an instruction in the same cycle in which another instruction accesses data. To guarantee forward progress, this hazard must always be resolved in favor of the instruction that accesses data.

A) What is the total execution time of this instruction sequence in the 5-stage pipeline that only has one memory?

Question 2.A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LW $5, -16($5)	F	D	X	M	W										
SW $4, -16($4)		F	D	X	M	W									
LW $3, -20($4)			F	D	X	M	W								
BEQ $2, $0, Label ;Assume $2 != $0				*S*	*S*	*S*	F	D							
ADD $1, $5, $4								F	D	X	M	W			
Label: SUB $2, $1, $3									F	D	X	M	W		

The execution time is 13 cycles

The stages that need to access memory are Fetch (instruction) and Mem (data). These 2 stages might not be existed in the same column (cycle). If conflicted, the subsequence instruction(s) need to be stalled until the memory is free from earlier instructions.

Note that $4 of SW is not a hazard. SW uses this $4 to store the value while LW uses this old $4 value added with the offset and load from the memory address.

Finally, if the branch is taken, so ADD $1,$5,$4 is flushed (in this case, it is not taken).

B) How can the structural hazard be eliminated by adding *NOP* to the code? (Please show a modified version of the program with the added *NOP* instructions)

The figure in Q.2.A is answer. To show as *NOP*, it is a below figure.

Question 2.B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LW $5, -16($5)	F	D	X	M	W										
SW $4, -16($4)		F	D	X	M	W									
LW $3, -20($4)			F	D	X	M	W								
nop															
nop															
nop															
BEQ $2, $0, Label ;Assume $2 != $0							F	D							
ADD $1, $5, $4								F	D	X	M	W			
Label: SUB $2, $1, $3									F	D	X	M	W		

Question 3: *(30 Points)*

For following code, assume that the loop index ($10) is a multiple of 8:

```
Loop:  LW    $2,   0($10)
       SUB   $4,   $2,   $3
       SW    $4,   0($10)
       LW    $5,   4($10)
       SUB   $6,   $5,   $3
       SW    $6,   4($10)
       ADDI  $10,  $10,  8
       BNE   $10,  $30,  Loop
```

Schedule this code (reorder the instructions and make any necessary changes) for fast execution on the 5-stage MIPS pipeline. Assume data forwarding and not-taken prediction of conditional branching.

There is no information for branch resolution stage, so for conditional branch BNE, it is resolved at EX stage
The original version: 13 cycles

Question 3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Loop: LW $2, 0($10)	F	D	X	M	W										
SUB $4, $2, $3		F	D	*S*	X	M	W								
SW $4, 0($10)			F	*S*	D	X	M	W							
LW $5, 4($10)					F	D	X	M	W						
SUB $6, $5, $3						F	D	*S*	X	M	W				
SW $6, 4($10)							F	*S*	D	X	M	W			
ADDI $10, $10, 8									F	D	X	M	W		
BNE $10, $30, Loop										F	D	X			

The reordered version: 10 cycles

Question 3	1	2	3	4	5	6	7	8	9	10	11	12
Loop: LW $2, 0($10)	F	D	X	M	W							
LW $5, 4($10)		F	D	X	M	W						
SUB $4, $2, $3			F	D	X	M	W					
SW $4, 0($10)				F	D	X	M	W				
SUB $6, $5, $3				F	D	X	M	W				
ADDI $10, $10, 8					F	D	X	M	W			
SW $6, -4($10)						F	D	X	M	W		
BNE $10, $30, Loop							F	D	X			

Note: Since the value of $10 is already added by 8, in order to get the correct address for SW, we need to adjust the displacement to the value of offset by 8.