Sabbir Ahmed
DATE: October 25, 2017
CMSC 411 HW 03

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

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

(a) **Question** Indicate dependences and their type.
Answer
RAW: I1 -> I2 ($5), I1 -> I3 ($5)
WAR: I2 -> I3 ($5), I1 -> I3 ($5)
WAW: I1 -> I3 ($5)

(b) **Question** Assume there is not forwarding in this pipeline processor, indicate hazards and add NOP instructions to eliminate them.
Answer
With no forwarding, the only hazards are the RAW.

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

(c) **Question** Assume there is full forwarding, indicate hazards and add NOP instructions to eliminate unresolved cases.
Answer
The only hazard with full forwarding is the RAW:

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

The remaining problem in this exercise assumes the following clock cycle times:
Without forwarding: 200 ps
With full forwarding: 250 ps
With ALU-ALU forwarding only: 220 ps

(d) **Question** 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?
Answer
With full forwarding, the instructions need 4 cycles. With no forwarding, the instructions need 5 cycles.

$$\therefore \frac{200 \cdot 5}{250 \cdot 4} = 1$$

(e) **Question** 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?

Answer
Total time: 220 · 5 = 1100 ps
Speed up: $\frac{1000}{1100} = 0.91$

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

Answer
Labeling the instructions:

```
LW      $5,   -16($5)             ; I1
SW      $4,   -16($4)             ; I2
LW      $3,   -20($4)             ; I3
BEQ     $2,   $0,     Label       ; I4
ADD     $1,   $5,     $4          ; I5
Label:  SUB   $2,     $1,     $3
```

Table 1: 12 Cycles Required for 5 Instructions

Instruction	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
I1	IF	ID	EX	-	-	-	MEM	WB				
I2		IF	ID	EX	-	-	-	MEM	WB			
I3			IF	ID	EX	-	-	-	MEM	WB		
I4				IF	ID	EX	-	-	-	MEM	WB	
I5					IF	ID	EX	-	-	-	MEM	WB

(b) **Question** 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)

Answer
Inserting NOP instructions cannot resolve the structural hazard since NOPs requires access to memory.

3. **Question** 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.

Answer
Labelling the instructions as the following:

```
Loop:  LW    $2,   0($10)      ; I1
       SUB   $4,   $2,   $3    ; I2
       SW    $4,   0($10)      ; I3
       LW    $5,   4($10)      ; I4
       SUB   $6,   $5,   $3    ; I5
       SW    $6,   4($10)      ; I6
       ADDI  $10,  $10,  8     ; I7
       BNE   $10,  $30,  Loop  ; I8
```

RAW data dependencies exist in I2 (on I1) and I5 (on I4). Since the branches are assumed not-taken, finding the clock cycles for the current order of instructions:

Table 2: 13 Cycles Required for 8 Instructions

Instruction	**t0**	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**t7**	**t8**	**t9**	**t10**	**t11**	**t12**	**t13**
I1	IF	ID	EX	MEM	WB									
I2		IF	-	ID	EX	MEM	WB							
I3				IF	ID	EX	MEM	WB						
I4					IF	ID	EX	MEM	WB					
I5						IF	-	ID	EX	MEM	WB			
I6								IF	ID	EX	MEM	WB		
I7									IF	ID	EX	MEM	WB	
I8										IF	ID	EX	MEM	WB

Reordering the instructions may decrease the total number of cycles and result in faster execution on the 5-stage MIPS pipeline.

The LW operations can be grouped together at the beginning, and the SUB at the end of the instructions. The reordered instructions would look similar to the following:

Table 3: Rescheduled Instructions

```
Loop:  LW    $2,   0($10)       ; I1
       LW    $5,   4($10)       ; I4
       ADDI  $10,  $10,  8      ; I7
       SUB   $4,   $2,   $3     ; I2
       SW    $4,   0($10)       ; I3
       SUB   $6,   $5,   $3     ; I5
       SW    $6,   4($10)       ; I6
       BNE   $10,  $30,  Loop   ; I8
```

The new schedule of the instructions would now require the following clock cycles:

Table 4: 11 Cycles Required for 8 Instructions

Instruction	**t0**	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**t7**	**t8**	**t9**	**t10**	**t11**
I1	IF	ID	EX	MEM	WB							
I4		IF	ID	EX	MEM	WB						
I7			IF	ID	EX	MEM	WB					
I2				IF	ID	EX	MEM	WB				
I3					IF	ID	EX	MEM	WB			
I5						IF	ID	EX	MEM	WB		
I6							IF	ID	EX	MEM	WB	
I8								IF	ID	EX	MEM	WB