Jong Park
CS 472 – Computer Architecture
Professor Lizhong Chen
HW2
April 29, 2019

Problem 1

```
SUB  R3,R1,R2
ADD  R2,R3,R2
XOR  R3,R3,R2
```

Without Forwarding	With Full Forwarding	With ALU-ALU Forwarding Only
250 ps	300 ps	290 ps

1.1) Indicate dependencies and their type (RAW, WAR, WAW, RAR). [Hint: For example, there is a RAW (read-after-write) on r2 from I2 to I3.]

RAW: R3: I1-I2, R2: I2-I3, R3: I1-I3
RAR: R2: I1-I2, R2: I1-I3, R2: I2-I3, R3: I2-I3
WAR: R2: I1-I2, R3: I2-I3
WAW: R3: I1-I3

1.2) Assume that there is no forwarding in this pipelined processor. Please add `nop` instructions to ensure correctness.

```
SUB  R3,R1,R2
NOP
NOP
ADD  R2,R3,R2
NOP
NOP
XOR  R3,R3,R2
```

1.3) Assume that there is full forwarding. Please add `nop` instructions to ensure correctness.

```
SUB  R3,R1,R2
ADD  R2,R3,R2
XOR  R3,R3,R2
```

No Changes.

1.4) What is the total execution time (in wall clock time) of this instruction sequence without forwarding and with full forwarding? What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?

Without Forwarding: (7 instructions + 4) * 250 ps = **2750 ps**
With **Forwarding**: (3 instructions + 4) * 300 ps = **2100 ps**
Speed up: 2750 / 2100 = **1.3095**, saving time of 650 ps

Problem 2

```
LW   R1, 8(R1)
ADD  R1, R1, R2
BEQ  R2, R3, Label      (assume R2 != R3)
SUB  R1, R1, R2
SLT  R1, R1, R2
```

	IF	**ID**	**EX**	**MEM**	**WB**
Given	**250 ps**	**200 ps**	**250 ps**	**280 ps**	**165 ps**
For 2.3	**250 ps**	**300 ps**	**225 ps**	**280 ps**	**165 ps**

2.1) Assuming Stall-on-Branch (that is, inserting bubble(s) after the branch so that only the correct instruction is fetched; in other words, no instruction is fetched until the branch result is known) for the above instruction sequence, what is the total number of clock cycles required to finish the instruction sequence if branch outcomes are determined in the EX stage?

Instructions\ Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
LW	IF	ID	EX	MEM	WB							
ADD		bub	IF	ID	EX	MEM	WB					
BEQ				IF	ID	EX	MEM	WB				
SUB					bub	bub	IF	ID	EX	MEM	WB	
SLT								IF	ID	EX	MEM	WB

12 Clock cycles needed to finish the instruction sequences.

2.2) Assuming Stall-on-Branch for the above instruction sequence, what is the speedup achieved on this code if branch outcomes are determined in the ID stage, relative to the execution where branch outcomes are determined in the EX stage?

Instructions\Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
LW	IF	ID	EX	MEM	WB						
ADD		bub	IF	ID	EX	MEM	WB				
BEQ				IF	ID	EX	MEM	WB			
SUB					bub	IF	ID	EX	MEM	WB	
SLT							IF	ID	EX	MEM	WB

Stall-on-Branch determined on EX stage (Every cycle is MEM time):	280 * 12 = 3360
Stall-on-Branch determined on ID stage (Every cycle is MEM time):	280 * 11 = 3080
Speed up:	3360/3080 = **1.0909**

2.3) Given the pipeline stage latencies, repeat the speedup calculation above, taking into account the (possible) change in clock cycle time. Assume that the latency of the ID stage increases by 50% and the latency of the EX stage decreases by 25ps when branch outcome resolution is moved from EX to ID.

Stall-on-Branch determined on EX stage (Every cycle is same time):	280 * 12 = 3360
Stall-on-Branch determined on ID stage (Every cycle is same time):	300 * 11 = 3300
Speed up:	3360/3300 = **1.018**

2.4) Replace the branch instruction in the above given instruction sequence with the following branch instruction: BEQ R1, R2, Label (assume R1 != R2)
What is the total number of cycles to finish this instruction sequence, assuming branch decision is in the ID stage? (Hint: be careful about what forwarding paths are available).

```
LW    R1, 8(R1)
ADD   R1, R1, R2
BEQ R1, R2, Label                (assume R1 != R2)
SUB   R1, R1, R2
SLT   R1, R1, R2
```

Instructions\ Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
LW	IF	ID	EX	MEM	WB							
ADD		bub	IF	ID	EX	MEM	WB					
BEQ				bub	IF	ID	EX	MEM	WB			
SUB						bub	IF	ID	EX	MEM	WB	
SLT								IF	ID	EX	MEM	WB

12 clock cycles.

Problem 3

```
SEARCH:   LW   R5, 0 (R3)              /I1 Load Item
          BNE  R5, R2, NOMATCH         /I2 Check for match
          ADDI R1, R1, #1              /I3 Count matches
NOMATCH:  ADDI R3, R3, #4              /I4 Next item
          BNE  R4, R3, SEARCH          /I5 Continue until all items
```

3.1) If branch data are needed in the EX stage and results are determined in the MEM stage, how many clock cycles does it take to execute one iteration of the loop on a match?

LW	1
stall	1
BNE	1
ADDI	1
ADDI	1
BNE	1
+3 flush	3
	= **9 cycles**

3.2) If branch data are needed in the EX stage and results are determined in the MEM stage, how many clock cycles does it take to execute one iteration of the loop on NO match?

LW	1
stall	1
BNE	1
+ 3 flush	3
ADDI	1
BNE	1
+ 3 flush	3
	= **11 cycles**

3.3) If branch results are determined in the ID stage, how many clock cycles does it take to execute one iteration of the loop on a match? You can assume that branch data are needed in the ID stage and two more forwarding paths are provided: from EX/MEM to ID and from MEM/WB to ID.

`LW`	1
+2 stall	2
`BNE`	1
`ADDI`	1
`ADDI`	1
stall	1
`BNE`	1
flush	1
	= 9 cycles

3.4) If branch results are determined in the ID stage similar to the above question, how many clock cycles does it take to execute one iteration of the loop on NO match?

`LW`	1
+2 stall	2
`BNE`	1
flush	1
`ADDI`	1
stall	1
`BNE`	1
flush	1
	= 9 cycles

Extra Credit:
In problem 1, if there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage), please add `nop` instruction(s) to this code to ensure correctness. Be careful. Make sure the added `nop` instruction(s) solves all the RAW dependencies. Explain why the `nop` instruction(s) is needed.

```
SUB  R3,R1,R2
ADD  R2,R3,R2          # Since ALU-ALU forwarding, there is no no-op needed here
NOP
NOP                    # There is still 2 no-op needed since only ALU-ALU are forwarded
XOR  R3,R3,R2          # but R2 isn't finished writing by the time XOR operation is called.
```