ECE/CS 472/572 – Computer Architecture
Instructor: Prof. Lizhong Chen
Homework #2 – Due: Monday, April 29 at 8:30m, Hard copy

Problem 0

- Read book chapter 4.5 to 4.7.
- We may not finish HW 2 grading and hand back your graded solutions by Midterm 1, so please keep a copy of your own solution if you need it.

Problem 1 (40 pts)

In this exercise, we examine how data dependences affect execution in the basic 5-stage pipeline described in Section 4.5. Problems in this exercise refer to the following sequence of instructions:

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

Also, assume the following cycle times for each of the options related to forwarding.

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

Without forwarding, there is no additional forwarding paths. Note that a new value can be written into a register in the first half of a clock cycle and this value can be read from the same register in the second half of a clock cycle. Full forwarding means the pipeline has two forwarding paths: from EX/MEM to EX (also called ALU-ALU forwarding), and from MEM/WB to EX.

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

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

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

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?

Problem 2 (40 pts)

Assume Full-forwarding for all the questions in this problem; full forwarding means the pipeline has two forwarding paths: from EX/MEM to EX and from MEM/WB to EX. There is no other forwarding path. Consider the instruction sequence given below:

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

Assume the individual pipeline stages have the following latencies:

IF	ID	EX	MEM	WB
250ps	200ps	250ps	280ps	165ps

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?

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?

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.

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).

Problem 3 (20 pts)

The classic pipeline design has 5 stages: IF, ID, EX, MEM, and WB. Consider the following program, which searches an area of memory and counts the number of times that the value of a word fetched from memory is equal to a value stored in R2.

```
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
```

Full forwarding is provided by the hardware. Both branches are always predicted untaken. Please answer the following questions for each loop iteration, except the last one.

[Hint 1: Bubbles may be needed between the end of a loop and the beginning of the next loop because of mis-prediction.]

[Hint 2: Consider "steady-state". Different from Problem 2, the instruction sequence here is not executed once but many times. The number of cycles of one iteration is the time interval from one `LW` to the next `LW`.]

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?

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?

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.

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?

Extra Credit (10 pts)

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.