Homework 2 (due September 17, 2019)

Question 1: Assume that the cycle time in the pipelined architecture does not allow the writing and reading of the register file in the same cycle. Hence, if at a given cycle, the ID stage wants to read from the register file and the WB stage wants to write to the register file, then one of the two stages has to stall and use the register file in a later cycle. In this question, assume that when the ID and WB stages compete for the register file in a given cycle, the WB stage gets to use the register file.

a. Using a table similar to the one shown below, trace the execution of the first 7 instructions of the following code segment (ignore instructions I8 and I9 for now) on a 5-stage pipeline architecture that has hardware support for forwarding and stalling:

```
I1:   lw    $2, 100($1)
I2:   addi  $1, $1, -4
I3:   sw    $2, 100($1)
I4:   add   $2, $2, $5
I5:   sw    $2, 100($3)
I6:   addi  $3, $3, -4
I7:   bneq  $1, $6, I1
I8:   sub   $1, $1, $7
I9    sub   $3, $3, $7
```

	IF	ID	EXE	MEM	WB
Cycle 1	lw $2, 100($1)				
Cycle 2	addi $1,$1, -4	lw $2, 100($1)			
Cycle 3	sw $2, 100($1)	addi $1,$1, -4	lw $2, 100($1)		
Cycle 4					
…					
…					

b. In your answer to part a, only stalling due to structural hazards is considered (there is no data or control hazards). In this part, you will assume that the branch condition is resolved in the ID stage, and hence, if the branch condition is true, then the instruction that entered the pipeline after the branch will have to be flushed to deal with control hazards. To demonstrate this effect, augment the table in your answer to part (a) by tracing the pipeline configuration for a few additional cycles after the bneq instruction reaches the WB stage assuming that the branch condition evaluates to “true”.

c. Assuming that the loop (I1-I7) will execute 10000 iterations (in each iteration, except the last time, the branch condition will be true), compute the CPI during the execution of the loop (ignore the time to fill up the pipeline).

Question 2: In this question, we will assume that the following code is executed on a 5-stage pipeline architecture that does not implement forwarding or stalling in hardware.

```
(1)     add     $4, $1, $1
(2)     lw      $2, 40($4)
(3)     sub     $2, $2, $5
(4)     sw      $2, 80($4)
(5)     lw      $3, 10($4)
(6)     sub     $3, $3, $5
(7)     sw      $3, 90($4)
(8)     beq     $3, $6, L
…..
     L: ……
```

a. Identify the data dependences that cause hazards (use sentences such as "dependence of instruction (y) on instruction (z) caused by register $x).
b. If the hazards identified in (a) are to be prevented by the compiler through no-op insertions but without reordering the instructions, indicate where would the compiler add no-ops (use sentences such as "insert x no-ops between instructions (y) and (z)).
c. Now assume a more intelligent compilers that can move instructions in the code. Can such a compiler reduce the number of added no-ops? If so, show the new code and indicate the added no ops.

Question 3: In this question, you will consider the instruction "R-type-rm $r1, $r2, $r3" that is described in Question 3 of Homework 1. This figure shows the modifications to the 5-stage pipelined architecture to accommodate the R-type-rm instruction. Specifically, a second ALU stage, EX2, is added to perform the arithmetic/logic operation for R-type-rm instructions after the second operand is read from memory, thus resulting in a 6-stage pipeline.

(a) Specify the values of the control signals (ALUsrc1, ALUsrc2, MemtoReg, MemWrite and MemRead) to implement the correct data paths for the R-type and the R-type-rm instructions. Also, specify what operation should the ALU in the first ALU stage (ALU1) perform when an R-type-rm instruction is in the ALU1 stage.
(b) In the 5-stage pipeline architecture, we added two forwarding paths from the EX/MEM and the MEM/WB buffers to the ID/EX buffer to avoid data hazards. These paths can be denoted EX/MEM → ID/EX and MEM/WB → ID/EX, respectively. We can demonstrate the use of the MEM/WB → ID/EX path in avoiding hazards using this short code segment:

```
add $1, $2, $3
sw   $4, 100($5)
sub  $6, $7, $1
```

Using a similar notation, identify the forwarding paths that should be added to the 6-stage pipeline to avoid data hazards and for each forwarding path, give a short example of a code segment that uses this path to avoid hazard.

(c) Even with the forwarding paths, the 5-stage pipeline had to stall for one cycle during the execution of the following code segment

```
lw   $4, 100($5)
sub  $6, $7, $4
```

to avoid data hazards. In other words, a one-cycle stall is needed when a “lw” instruction writes to a register and is immediately followed by an instruction that reads from this register. Use short code segments to describe the cases that will necessitate the stalling of the 6-stage pipeline. Note that there are more than one case and in some cases, the pipeline may have to stall for more than one cycle.

Question 4:

Consider two designs for the 5-stage pipelined architecture. In the first design, the branch address and target are resolved in the MEM stage, while in the second design, the branch target and address are resolved in the ID stage but the clock cycle time is 10% larger than in the first design. Assuming that

- the instruction mix executing on the pipeline contains 25% branch instructions, 70% of which are taken
- the CPI of both designs is 1.5 if we do not account for stalling due to control hazards

a. Compute the CPI for each of the two designs when the effect of control hazards is accounted for
b. Specify which of the two designs would be more efficient.