CMPE 200 HW#2

Due: Monday, Sep. 30, 11:59pm
Total Score: /100

Instructor: Hyeran Jeon

Computer Engineering Department, San Jose State University

Pipelined Datapath and Data Hazards (10pts + 45pts + 45pts)

1. Assume that individual pipeline stages of a five-stage piplined architecture take the following latencies:

IF	ID	EX	MEM	WB
100ps	120ps	220ps	300ps	120ps

   a. What is the clock latency in a pipelined processor?

   b. Suppose that the clock latency is set as your answer above. What is the total latency of a Beq instruction in a pipelined?

2. Consider the following code.

```
addi  $sp, $sp, -4        // I1
lw    $a0, 0($sp)         // I2
slti  $t0, $a0, 60        // I3
addi  $t0, $t0, 1         // I4
sw    $t0, 0($sp)         // I5
addi  $sp, $sp, 4         // I6
```

   a. Indicate dependencies among instructions to the following table.

Type	Dependencies in the code Format: “register id” of “instruction id” → “dependent instructions id” (i.e. $t0 of I1 → I2)
Read-After-Write (RAW)	
Write-After-Write (WAW)	
Write-After-Read (WAR) • Do not consider the dependency within an instruction (ex. Register value is read and the same register is written with a calculation result)	

b. Among the three types of dependencies, which dependency type would cause Hazards? and Why?

c. Assume that we have a hazard detection unit in a 5-stage pipelined architecture. We do not have a forwarding unit. The hazard detection unit stalls instructions to resolve hazards. We use an advanced register file design that can read a register value that is written in the same cycle. How many cycles would take to execute the code?

d. Repeat c. but assume that we also have a forwarding unit that forwards data from EX/MEM to EXE and from MEM/WB to EXE.

3. Consider the following code.

```
I1:   LOOP:     lw   $a0, 0($sp)
I2:             slti $t0, $a0, 1
I3:             beq  $t0, $zero, L1
I4:             addi $sp, $sp, 8
I5:   L1:       lw   $ra, 4($sp)
I6:             addi $t4, $ra, 8
I7:             sw   $t4, 0($t1)
I8:             addi $v0, $zero, 1
I9:             b    LOOP
I10:            add  $t4, $t4, $t3
```

Assume that we have data forwarding paths from EX/MEM to EXE stage and from MEM/WB to EXE stage. We use an advanced register file. We do early branch determination (branch outcome is generated in ID stage and the PC value is updated in ID stage). Branches are predicted untaken and the actual outcome is shown in the table. Show the first 18-cycles of execution of the code in a timing diagram shown below by assuming that each branch's outcome is as follow. When an instruction is stalled in a pipeline stage, fill the pipeline stage's name for the instruction until the end of stall. Draw additional rows if needed. Show instruction id to the first column.

ID	Branch instructions	Branch outcome (NT: not taken, T: taken)
I3	`beq $t0, $zero, L1`	T→NT
I9	`b LOOP`	T

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18
I1																		