CSE490/590, Spring 2025 Homework 4 (not graded)

1.

(a) Schedule the following instruction sequence for dual-issue MIPS. Consider one ALU/branch instruction and one load/store instruction can be executed in parallel when there are no data dependencies:

```
Loop: lw $t0, 0($s1)       // $t0=array element
      add $t0, $t0, $s2    // add scalar in $s2
      sw $t0, 0($s1)       // store result
      addi $s1, $s1,-4     // decrement pointer
      add $s4, $s5, $s4    // update $s4
      bne $s1, $zero, Loop // branch when $s1!=0
```

	ALU/Branch	Load/Store	Cycle
			1
			2
			3
			4
			5
			6

(b) Compute the IPC in part (a)

2.

Assume a MIPS 5-stage pipelined processor **without** forwarding, show loop unrolling so that there are four copies of the loop body for the following instruction sequence. Schedule the instructions to avoid stalls.

```
Loop: lw $s0, 0($t0)
      addi $t2, $t2, $s0 // $t2 contains the sum of the array
      addi $t0, $t0, 4
      bne $t0, $t1, Loop
```

The above code iterates over an array of integers and computes its sum.

Assume that `$t0` contains the address of the first word, (`$t1 - 4`) is the address of last word, and that the difference between the addresses in `$t0` and `$t1` is a multiple of 16.

Eliminate any obviously redundant computations. Assume registers `$s0` to `$s7` and `$t3` to `$t9` are free to use for any purpose.

CSE490/590, Spring 2025 Homework 4 (not graded)

3. For the code sequence shown below

```
loop:
    l.d $f12, 0($f5)
    add.d $f6, $f6, $f12
    daddui $f5, $f5, -8
    bne $f5, $f9, loop // $f9 holds the address of the last value to be operated on.
```

a) Show loop unrolling so that there are four copies of the loop body Assume $f5, $f9 (that is, the size of the array) are initially a multiple of 32, which means that the number of loop iterations is a multiple of 4. Eliminate any obviously redundant computations and do not reuse any of the registers.

b) Compute the number of cycles needed for 4 iterations

Instruction producing result	Instruction using result	Latency in cycles
FP ALU op	Another FP ALU op	3
FP ALU op	Store double	2
Load double	FP ALU op	1
Load double	Store double	0

4. Consider the following code sequence.

I1: lw $s4, 0($s1)
I2: or $s2, $s4, $s1
I3: and $s6, $s5, $s3

Highlight the Hazard and discuss how out of order processor will help when lw $s4, 0($s1) encounters a cache miss?

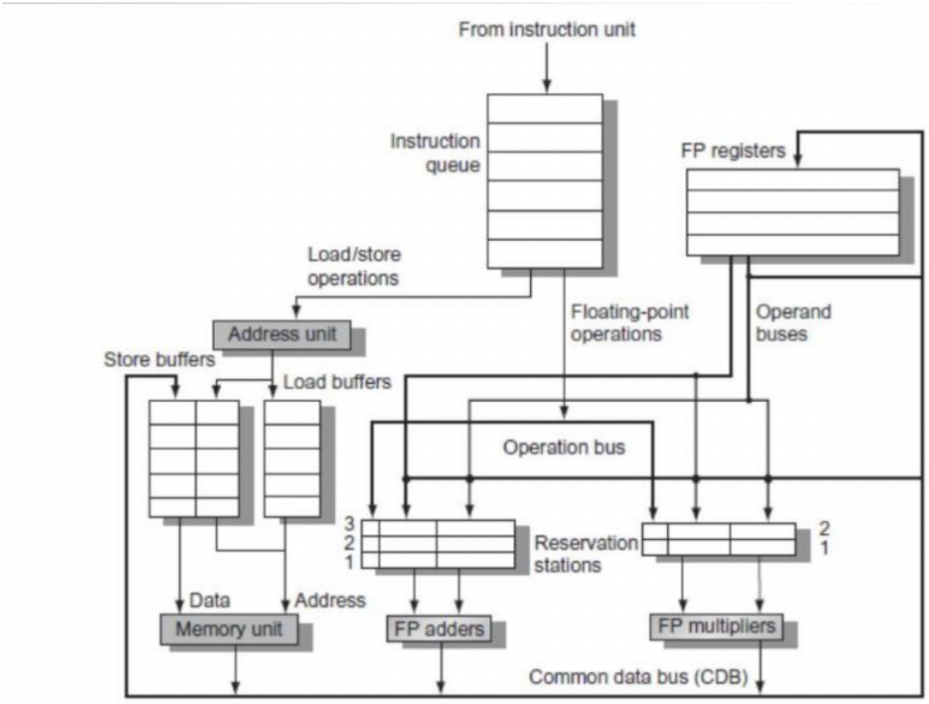
5. In the following instruction sequence, find the hazards. Rename the registers to eliminate the anti and output dependences

div.s r1,r2,r3
mult.s r4,r1, r5
add.s r1 ,r3, r6
sub.s r3,r1, r4

CSE490/590, Spring 2025 Homework 4 (not graded)

6. Consider the following instruction sequence (floating point) on a processor (shown below) which uses Tomasulo's algorithm to dynamically schedule instructions (dual issue per cycle - no speculation)

```
w:  ADD R4, R0, R8
x:  MUL R2, R0, R4
y:  ADD R4, R4, R8
z:  MUL R8, R4, R2
```



The processor has the following non-pipelined execution units:
A 2-cycle, FP add unit
A 3-cycle, FP multiply unit
Assume instructions can begin to execute in the same cycle as soon as its dispatched and resides in Reservation Stations
Trace the execution by showing Reservation Stations and FP Registers at the end of cycles# 1,2,3,5 and 6