Name: ______________________________

Multicycle CPU

Please complete this assignment in pairs.

For reference: The multi-cycle CPU presented in the Harris and Harris textbook has a clock period of 325. The SPECINT benchmark has a CPI of 4.12 on this CPU.

1. One common way to improve the performance of multi-cycle CPUs is to add more complex instructions in an attempt to reduce the number of instructions. What percent of instructions would need to be removed in order for the multi-cycle CPU to have better performance than the single cycle CPU (assuming a clock the CPI remains at 4.12)?

2. Consider the following MIPS assembly code for adding array B to array A:

```
# $a0 contains the address of A
# $a1 contains the address of B
# $a2 contains the lenght of A and B.
add_arrays:
     sll $a2, $a2, 4   # convert array length to size in bytes
     add $a2, $a0, $t7 # $a2 now points to first word past array A
     beq $a0, $a7 END  # skip loop if array has size 0
TOP: lw   $t2, 0($a0)
     lw   $t3, 0($a1)
     add  $t4, $t2, $t3
     sw   $t4, 0($a0)
     addi $a0, $a0, 4
     addi $a1, $a1, 4
     bne  $a0, $a2, TOP
END  jr $ra
```

   (a) How long does the single-cycle CPU take to run `add_arrays` when the arrays contain 10,000 elements?

   (b) How long does the multi-cycle CPU take to run `add_arrays` when the arrays contain 10,000 elements? (Use the CPI for this specific instance. Don't just assume 4.12)

3. Consider a new instruction `addmd` that allows one source and the destination to be in memory. For example, `addmd $t0, $t1` would do this: `M[$t0] += $t1`. (Notice that the destination is the same as one of the sources.)

   Re-write the assembly code above to utilize this new instruction.

Name: ______________________________

4. Design this new instruction
   (a) What data paths / mux inputs must you add to implement this instruction? Why?
   (b) Design the instruction. (Specify how each bit in the instruction is used. For example, which bits specify the destination address?)
   (c) List the micro-ops for this instruction. Clearly separate micro operations by cycle.

5. How long does it take the multi-cycle CPU to run the new code using `addmd`?

6. Even with `addmd`, the single-cycle CPU is faster. How may cycles total can appear in the loop of the multi-cycle code before the single-cycle CPU is faster?

7. We could potentially save more cycles by allowing both parameters to the `add` instruction to be in memory. However, this would require a more complex change to the CPU. Why?

8. Can you come up with another CISC-style instruction or two that allows this loop to run faster on the Multi-cycle CPU?

9. The difficulty making this tight, easily optimized loop faster on the multi-cycle CPU suggests that it will be difficult, if not impossible, to make this multi-cycle CPU faster than the corresponding single-cycle CPU. So, let's try another approach: Specifically, let's try re-defining the steps.

   The main problem with the multi-cycle CPU presented in the textbook is that the CPI of a typical program is 4.12, whereas the clock period is only 2.84 times faster. This difference is primarily caused by unbalanced stages: The Memory stages are much slower than the register writeback stage. Let's see what happens if we combine this fast writeback stage with the ALU and memory stages. This should reduce the CPI by almost 1 while increasing the cycle time only slightly.

   (a) Currently the last two micro-operations for `lw` are
   ```
       MDR <= M[ALU_out]
       RF[..]  <= MDR
   ```
   How much time is needed for a new micro-operation that writes the memory result directly to the register file?
   ```
       RF[..]  <= M[ALU_out]
   ```
   (b) Similarly, how much time is needed to write the results of ALU operation directly to the register file (instead of using $\texttt{ALU}_{\texttt{out}}$)?
   (c) What would the clock period be for this new design?
   (d) What would the new CPI be for the SPECInt benchmark be using this new design?
   (e) What is the speedup of this improved Mulit-cycle CPU relative to the single-cycle CPU?