Name: _____

Practice Final

Revised April 21, 2021

None of these problems will be due for credit. This is more of a review check-list than an actual practice final; however there are a few practice problems.

1. When we first looked at instruction sets, we listed four design principles:

   (a) regularity promotes simplicity,
   (b) smaller is faster.
   (c) good designs require good compromises,
   (d) make the common case fast

   How do these design principles apply to each of the following:

   - The MIPS instruction set
   - Pipelining
   - The memory hierarchy

2. What aspects of the above seem to violate the four principles?

3. Show how 32-bit addresses are divided into tag, index, and offset given the following cache descriptions:

   (a) 32KB, byte addressable, 8-way set associative cache with 4 byte blocks.
   (b) 16KB, byte addressable, direct mapped cache with 8 byte blocks.
   (c) 64KB fully associative cache with 64 byte blocks

4. Calculate the overhead for each of the above caches.

Name: _____

5. You have an 128KB, byte addressable, 2-way set associative cache with 16 byte blocks. This cache uses an LRU replacement policy. For the following sequence of addresses, show whether each is a hit or a miss. For each cache miss, tell which bytes were replaced.

   Extra Credit: *Write your own cache replacement problem (a problem similar to this one with a configuration and different sequence of memory accesses) and post it to Piazza.*

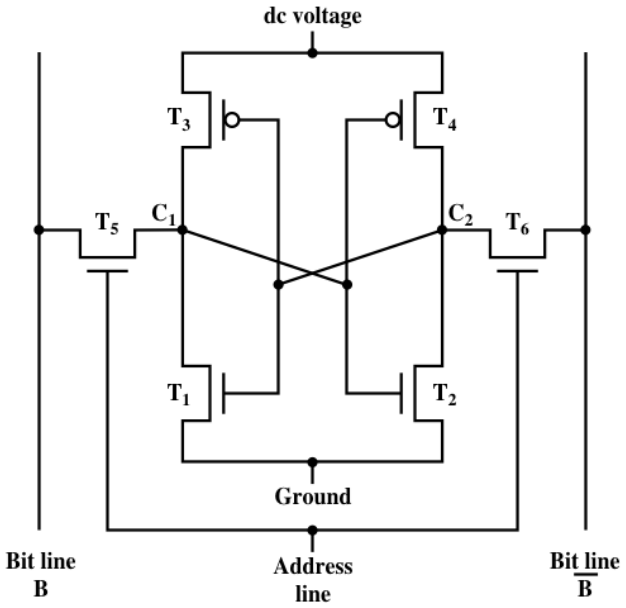
Address	Tag	Index	H / M	Tags Replaced	Notes
0x12341110					Cold
0x12341113					Same block
0x12351113					Cold
0x12341113					
0x12361114					
0x12351113					
0x12361114					
0x12341110					
0x12361110					Same block
0x12351113					

6. Set-associative caches generally have better hit rates than direct-mapped caches for a given size. However, it is possible to find counter examples. Construct a repeating sequence of references such that the 1KB direct-mapped cache described in problem 5 achieves a better hit rate than the 2-way set-associative version of the cache.

7. LRU is generally the best replacement policy when using an associative cache. Construct a repeating sequence of memory references for which FIFO or Random is better than LRU.

8. Consider the SRAM diagram below:

Name: _____



(b) Static RAM (SRAM) cell

(a) Be sure you understand that a a "closed" transistor allows current to pass; and "open" transistor does not allow current to pass. Thus, placing a 1 on the address line closes transistors T_5 and T_6.

(b) Explain how transistors T_3 and T_4 are different from the other transistors in this diagram.

9. Explain the difference between spatial locality and temporal locality.

10. How does a program's temporal locality affect its cache hit rate?

11. How does a program's spatial locality affect its cache hit rate?

12. What is "cache coherence" and why is it important for multi-core processors?

13. Consider a non-pipelined, single-cycle CPU (i.e., all instructions complete in one CPU cycle). Suppose you are asked to design a similar CPU with a two-stage pipeline. List the factors that may prevent you from making the cycle time of the new pipelined machine half of the cycle time of the non-pipelined machine.

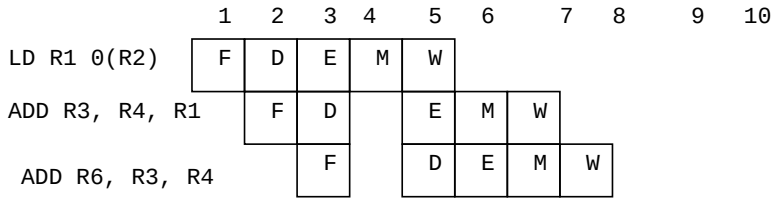
14. Consider two CPUs that differ only in that one is pipelined and one is not. Is the total time needed to completely execute a single instruction greater on the pipelined or non-pipelined machine? Why? (In other-words, does pipelining tend to increase or decrease the latency of individual instructions?)

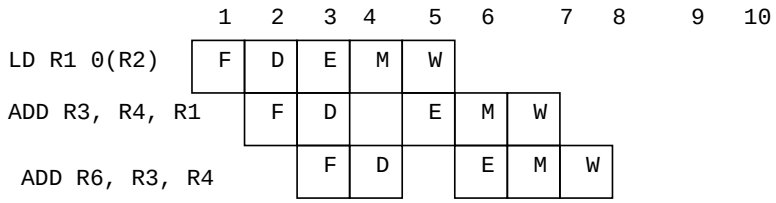
Name: _____

15. Look at the two tables below.

    (a) Assume that your processor is a 5-state pipeline with both data and register forwarding. Which table is correct? Why?

    (b) Draw a diagram showing how the instructions would flow through the pipeline if it didn't have any data or register forwarding.





16. What are structural hazards? What causes them? How are they resolved?

17. Are structural hazards unique to pipelined CPUs? (As opposed to non-pipelined CPUS?) Why or why not?

18. What are data hazards? What causes them? How are they resolved?

19. Are data hazards unique to pipelined CPUs? (As opposed to non-pipelined CPUs?) Why or why not?

20. What are control hazards? What causes them? How are they resolved?

21. Are control hazards unique to pipelined CPUs? (As opposed to non-pipelined CPUs?) Why or why not?

Name: _____

22. (Problem 6.4 from Patterson and Hennessy): Identify all of the data dependencies in the following code. Which dependencies are data hazards that will be resolved by forwarding? Which dependencies are data hazards that will cause a stall?

```
1: add $3, $4, $2
2: sub $5, $3, $1
3: lw  $6, 200($3)
4: add $7, $3, $6
```

23. The performance of a pipelined computer, in terms of how long it takes to run a program, can be expressed as:

$$\frac{seconds}{program} = \frac{instructions}{program} \times \frac{cycles}{instruction} \times \frac{seconds}{cycle}$$

(a) What term in the equation above corresponds to n?

(b) What term in the equation above corresponds to p?

(c) What does $\frac{cycles}{instruction}$ represent with respect to a pipelined CPU?

(d) How does k affect this formula?

Name: _____

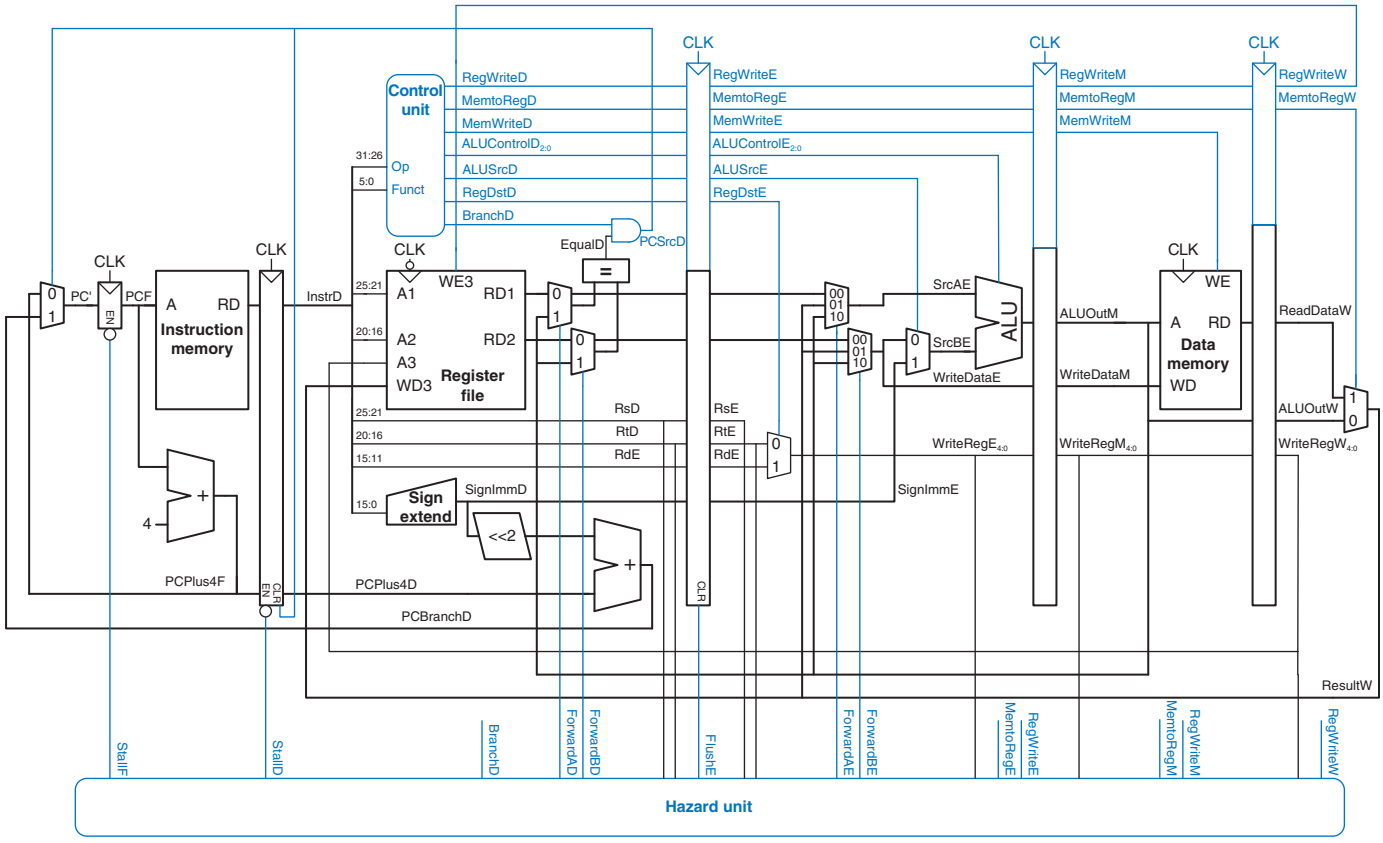
24. Explain the purpose of each component and control wire in the figure below:

25. Highlight the forwarding paths used by the following sequence of instructions:

```
add $t0, $t1, $t2
sub $t3, $t4, $t5
add $t5, $t0, $t3
```

26. Highlight the forwarding paths used by the following sequence of instructions:

```
add $t0, $t1, $t2
sub $t3, $t4, $t5
beq $t0, $t3, END
```



Name: _____

27. Consider the assembly code fragment below:

```
sw $16, 12($6)
lw $16, 8($6)
beq $5, $4 LABEL
add $5, $1, $4
sub $5, $15, $4
```

Suppose we modify the pipeline so that it has only one memory (that handles both instructions and data). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

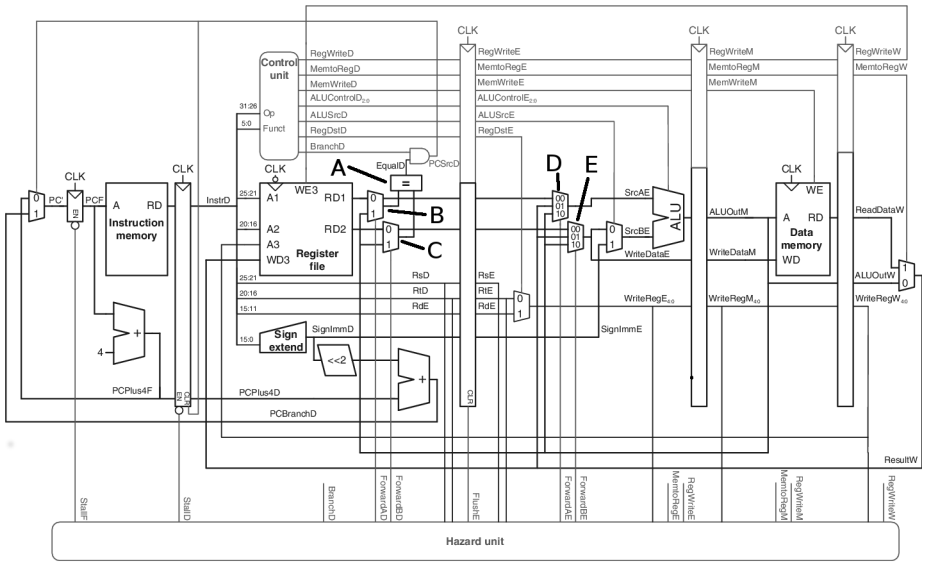
(a) Draw a pipeline diagram to show were the code above will stall.

(b) In general, is it possible to reduce the number of stalls/`nop`s resulting from this structural hazard by reordering code?

(c) Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding `nop`s to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.

(d) Approximately how many stalls would you expect this structural hazard to generate in a typical program? (Assume 25% of instructions are loads and 10% are stores.)

28. What are the purposes of items D and E in the diagram below?



Name: _____

29. Convert the following Java code to assembly. Use standard calling and register usage conventions. You may assume `nCk` has already been written. (In other words, call it, but don't implement it.)

```
int row_sum(int n) {
  int sum = 0;
  for (int i = 0; i <= n; i++) {
     sum += nCk(n, i);
  }
  return sum;
}
```