Book: chapters 5.6 – 5.8, 5.10 – 5.14,5.16-5.17, 6.1 – 6.7,6.10-6.11,6.13-6.14
Read the book! For this part of the class, lectures didn't necessarily cover all details covered in the book.

The final is cumulative, with a focus on coalescing the knowledge gained throughout the quarter.
The list below focuses on the topics after the second midterm.
Not every topic on this list is covered on the practice final.
Check the written homeworks for more questions.
If you want to review previous topics, check the written homeworks or practice midterms.

- Virtual memory
    - Translation mechanisms
    - Paging
    - Page tables
    - TLB
    - AMAT with TLBs
    - Reducing translation overhead
    - Virtualizing the whole system
- Branch prediction
    - Perceptron paper
    - Timing constraints
- Parallel programming
    - Flynn's taxonomy
    - Parallel processing and memory
    - Warehouse-scale computing
    - Accelerators

Question 1. Current processors and operating systems support “large pages” or “huge pages”. These are virtual memory page sizes larger than the base page size (e.g., x86 supports 2 MB and 1 GB pages in addition to the base page size of 4 KB).

a. What kind of locality do large pages sizes take advantage of?

b. In the x86 multi-level page table we discussed in class, will large pages increase, decrease, or have no effect on the TLB miss penalty compared to the standard 4 KB page size? Why?

c. Consider the TLB miss ratio with nominal page size (4 KB) and huge pages (2 MB). Will a TLB with 2 MB or 4 KB pages have a lower miss ratio (higher hit ratio)? Why?

Question 2. Shared memory versus message passing.

a. Describe a shared memory machine.

b. Describe a message passing machine.

c. Name an example system architecture that is a shared memory machine.

d. Which is easier to write a program for: a shared memory machine or a message passing machine? Why?

Question 3. Vector machines are an example of a SIMD style of parallel processing. They feature instructions that look like VR0 = VR1 + VR2. Explain briefly what a vector register is, and why these machines can fetch and decode many fewer instructions than a traditional processor does. Use pictures if that will help get your point across.

Question 4. Parallel architectures and programming models.

a. Describe a SISD architecture.

b. Describe a SIMD architecture.

c. Which is easier to write a program for: SISD or SIMD? Why?

Question 5. 75% of an application is data parallel work (e.g., vector operations).

a. What is the maximum speedup on a SIMD machine with 8 lanes?

b. What is the maximum speedup on a SIMD machine with 16 lanes?

c. What is the speedup of the application on 16 lanes compared to 4 lanes? Why is it not 4x?

d. What is the maximum possible speedup? How many lanes are necessary to achieve this?

Question 6. SIMD machines can be more efficient than SISD machines. Why?

Question 7. True/False: Many applications exhibit data-level parallelism, which can be exploited by vector architectures.

Question 8. Compare a scalar machine to a multi-***threaded*** machine:

a. For a SISD application, how will the performance compare between the two machines?

b. For a MIMD application, how will the performance compare between the two machines?

c. If there are many ***different*** applications running on the system, how will the throughput compare between the two systems?

Question 9. Assume a 40-bit virtual address and 4 KB page size.

a. Draw the address breakdown used to access the MMU.

b. For a fully associative TLB, which bits are used for the tag?

c. What is the data in a TLB? Why do we cache this data?

d. If the TLB had 16 entries and was 8-way set associative, which bits would be used for the index?

Question 10. For the following addresses, mark whether it hits or misses in the TLB. For the addresses that hit in the TLB, write the physical address. The page size is 4 KB.

TLB	
Tag	*Data*
`0x56076`	`0xc0268`
`0x04d22`	`0x0f20a`
`0xb1186`	`0x46861`
`0x5363e`	`0xb5b6b`
`0x2509c`	`0x1491c`

`0x5363ea56` `0x2509c928`

`0x504d22d1` `0x4604d20c`

`0x10256076` `0xb5b6b2f2`

`0xb1186cd2` `0x04d22caf`

Question 11. Accelerators.

a. Why would we want to add an accelerator to our system?

b. Should we add an accelerator to every system we build? Is there a reason why we shouldn't add accelerators for everything?

Question 12. We are building a new dynamic branch predictor for use in a CPU. This branch predictor is more accurate than the static branch predictor that we've used previously but consumes a decent amount of additional power. We can use either the dynamic one that we've built or go back to the static one.

a. If our design team was solely concerned with the power usage of the CPU, which branch predictor should we pick? Why?

b. If our design team was solely concerned with the application performance of the CPU, which branch predictor should we pick? Why?

c. At a higher level (without numbers), if our design team needed to decide based on all factors, not just power or performance, what would influence our decision? Listing those other factors may help.