Carnegie Mellon University
Electrical and Computer Engineering Department

18-742 Spring 2025
Midterm 1 - Practice Questions

Note: The actual midterm will have 9 questions.
This example has only 4 questions.
(Also, these questions overlap with one another more than the actual midterm will...)

INSTRUCTIONS:

- Closed book. No notes. 80 minutes.
- If you find a question ambiguous, be sure to write down any assumptions you make.
- Be clear and concise.
- **Answer 7 of the 9 questions.** We will grade the **first 7 questions answered.** If you start to answer a question but decide later that you don't want it graded, draw a big **X** across the page.
- For your 7 graded questions, it is better to partially answer a question than to not attempt it at all.
- All questions have the same weight.
- You may find some questions easier than others.
- If you need more space for your answer, there are two blank pages at the end. Be sure to clearly label any answer on these blank pages with BOTH the question number (1-9) and the part number (a,b,c).

QUESTIONS:

Parallel Computer Architectures ("Amdahl's Law, Cache Coherence, Memory Consistency")

1. Multicore processors are prevalent in today's processor market. Within a multicore processor, various cores are capable of executing distinct threads of a program simultaneously. This architectural design mandates the implementation of effective cache coherence models. These models are crucial for ensuring the correct execution of programs.

   (a) Discuss two conditions under which Amdahl's Law may *overestimate* the potential performance improvements in parallel computing systems.

   (b) Discuss a condition under which Amdahl's Law may *underestimate* the potential performance improvements in parallel computing systems.

   (c) Consider the following code segments intended to run concurrently on two different processors. Processor 1 executes Code Segment 1, and Processor 2 executes Code Segment 2. Initially, shared variables A and B are both set to 0. The system does not inherently guarantee the order of memory operations across processors without explicit synchronization instructions. Can an efficient cache coherence protocol ensure correct execution of this program? Please explain your answer.

   Code Segment 1 (Processor 1):

   ```
   A = 1;
   B = 1;
   ```

   Code Segment 2 (Processor 2):

   ```
   while (B == 0);
   int readA = A;
   ```

Cache Coherence **("Parallel Programming, Coherence Protocol, Directory Scalability")**

2. Parallel processing requires the implementation of efficient coherence mechanisms to guarantee the accurate execution of parallel programs.

   (a) Consider a scenario where two processors, P1 and P2, are working on a shared data structure that is cached in both of their local caches. Describe how a lack of cache coherence would affect the execution of a program that requires both processors to update the shared data structure. How would cache coherence mechanisms mitigate these effects?

   (b) In the paper "Why On-Chip Cache Coherence is Here to Stay," which method is recommended to scale the *directory area* for cache coherence? Explain how.
   (A) Data sharing
   (B) Hierarchical design
   (C) Inexact sharer tracking
   (D) Metadata compression

   (c) According to the same paper, how does "inclusion" facilitate scaling on-chip cache coherence?

Data Prefetching **("Memory Access Prediction, Data Access Patterns, Memory Latency")**

3. Data prefetching is the process of predicting and prefetching future memory references before they are explicitly requested by applications, with the goal of hiding the memory access latency.
   (a) Discuss two potential drawbacks of data prefetching.

   (b) As discussed in class, "helper thread prefetching" constructs a *helper* thread which performs data prefetching for the *main* thread. Compare Bingo prefetching with helper thread prefetching, highlighting two advantages and two disadvantages of Bingo relative to helper thread prefetching.

   (c) Why is the consolidation of metadata tables effective in reducing Bingo's storage overhead?

Runahead Execution (**"Memory-level Parallelism, Helper Thread Prefetching"**)

4. Runahead execution allows memory-level parallelism (MLP) to break past reorder buffer (ROB) limits, by speculatively ignoring dependencies and continuing execution of the thread upon a miss in order to issue prefetch requests.

   (a) As discussed in class and the runahead execution paper, another method for prefetching challenging memory references is "helper thread prefetching," which constructs a *helper* thread to perform data prefetching for the *main* thread. Mention one benefit of runahead execution in comparison to helper thread prefetching.

   (b) Runahead execution makes use of a "runahead cache." What is its purpose and how does it function?

   (c) When compared to hardware prefetchers like Bingo, list two advantages and two disadvantages of runahead execution.