Full Name:_____

15-740/18-740, Spring 2018

Exam 1

March 8, 2018, 3:00pm-4:20pm

Instructions:

- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer. A few pages of scratch paper are provided at the end of the text booklet, but your final answer should be written in the space provided.
- Show your work and discuss your answer. You will be graded more on your explanation than on your final answer.
- The exam has a maximum score of 80 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is CLOSED BOOK, CLOSED NOTES. You may use a calculator, but no networked devices (e.g., phones, laptops, etc.).
- The exam introduces techniques and key ideas from research papers and asks you analyze them. *Confine your answers to the ideas that are presented in the exam.* We are not looking for answers that involve other techniques in these papers which we do not present.

Do not write below this line

Problem	Your Score	Possible Points
1		40
2		40
Total		80

Trade-offs in Architecting DRAM Caches

Problem 1. (40 points total):

This problem analyzes the design trade-offs in architecting large-scale DRAM caches. Accessing main memory is becoming increasingly expensive, so many architects have proposed adding an additional layer to the cache hierarchy. These caches need to be very large (hundreds of MBs–several GBs) in order to provide significant benefit over existing last-level caches, which are already tens of MBs. As a result, these caches are often built using DRAM instead of SRAM.

A. Based on the above discussion, why would DRAM be more attractive than SRAM for these caches?

2 points

B. Figure 1 shows a baseline system with an SRAM on-chip L3 cache and off-chip main memory (there is no DRAM cache yet). The size and latency of each are shown: L3 is 32 MB with a latency of 20 cycles, and main memory is 512 GB with a latency of 100 cycles. Additionally, the L3's miss ratio is 50%.

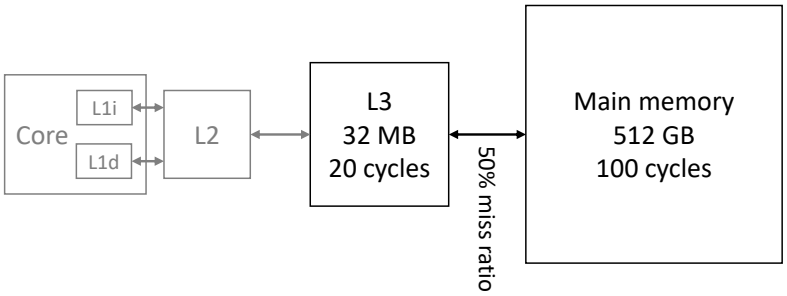

Figure 1: Baseline system without a DRAM cache. CPU and L1/L2 caches not shown in detail.

What is the average memory access time (AMAT) of this system's **L3** cache? *(Show your work!)*

3 points

$\Rightarrow$ L3 AMAT = _____ cycles

C. We will now add a DRAM cache to our design. Figure 2 shows the system with a 2 GB L4 DRAM cache.

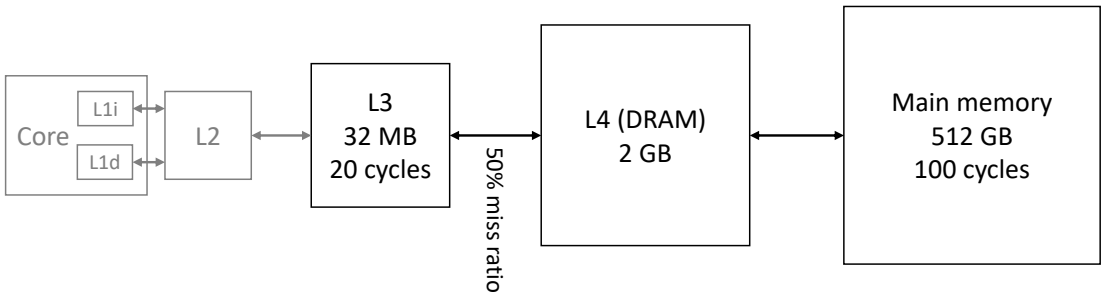

Figure 2: Baseline system with an L4 DRAM cache.

One of the main differences between SRAM and DRAM caches is that DRAM caches do not contain any specialized internal logic. (They are built from commodity DRAM chips to reduce cost.) As a result, *all information stored in the cache, including tags, must be read out* ***sequentially*** and processed on the processor die. In particular, it is not possible to match tags in parallel, as we did in SRAM cache designs; instead, tags must be matched sequentially.

We will first consider a set-associative design that places an entire set in each DRAM row.[1] This cache uses 64 B cache lines (like the L3). Each 2 KB DRAM row can store **29** lines, saving **3** lines for tags.

It takes **35** cycles to activate a row (i.e., to read 2 KB into the row buffer), and **20** cycles to activate a column (i.e., to read 64 B from the row buffer). Ignore precharge and other timing restrictions.

What is this set-associative design's **average hit time**? Assume the desired set is always *not* loaded into the row buffer.

5 points

[1] Loh and Hill, ...

If the set-associative L4's miss ratio is **25%**, what is the **L3's** AMAT in Figure 2?
3 points

$\Rightarrow$ L3 AMAT = _____ cycles

D. One alternative design for DRAM caches is *page caches*,[2] which significantly increase the cache's line size so that it matches the row buffer size (e.g., each line is 2 KB).

One of the main advantages of page caches is that they allow the L4's tags to be stored in on-chip SRAM. Why is this possible for page caches but not for the two other designs? Feel free to assume in your answer that each tag is 8 B in all designs.

2 points

What are the advantages of a page cache design for DRAM caches? List as many as you can and explain.

5 points

What are the disadvantages of a page cache design for DRAM caches? List as many as you can and explain.

5 points

[2] Falsafi, ...

E. Suppose we were computing a dot product on two **1 GB** vectors using the following code, similar to the example discussed in class:

```
float dotproduct(float x[], float y[], int n) {
  float result = 0;
  for (int i = 0; i < n; i++) {
    result += x[i] * y[i];
  }
  return result;
}
```

Where is the locality in this program?

2 points

Would you recommend the original set-associative or page-cache design? Under what circumstances? Explain your reasoning and assumptions.

8 points

Heterogeneous Multicore Systems

Problem 2. (40 points total):

To satisfy the needs of different kinds of programs, systems are increasingly heterogeneous, incorporating several kinds of cores within a single processor. Heterogeneity is potentially more efficient, but introduces its own complications. This problem discusses some opportunities and challenges presented by heterogeneous systems in the simplest context: a so-called "big.LITTLE" design with many small, simple cores and one (or a few) large, powerful cores.

Specifically, we consider 2 kinds of cores

(a) "LITTLE" cores are designed for area and energy efficiency; and

(b) "big" cores are designed for maximum performance.

The "big" core is **twice** as fast as the "LITTLE" core.

A. You are trying to run a benchmark that takes **200s** to run sequentially on a single "LITTLE" core, and thus 100s on a "big" core. The benchmark consists of both sequential and parallel parts.

Let f be the fraction of the full execution that is parallel. If you are given a machine with N "LITTLE" cores and M "big" cores, give an expression for the **minimum execution time** of the benchmark.

6 points

B. Compute the execution time of the benchmark on a machine that $N = 8$ LITTLE cores and $M = 0$ big cores for $f = 0.5$, 0.75, and 0.99. (Feel free to leave your answer in fractions.)

3 points

C. Compute the execution time of the benchmark on a machine that $N = 4$ LITTLE cores and $M = 1$ big core for $f = 0.5$, 0.75, and 0.99. (Feel free to leave your answer in fractions.)

3 points

D. Which system performs better? Explain using key concepts discussed in class.

3 points

E. Accelerated Critical Sections (ACS)[3] is a technique that leverages the "big" cores to accelerate the execution of critical sections. In ACS, selected critical sections are executed by a high-performance (e.g., "big") core, which can execute the critical section faster than the other, "LITTLE" cores. In ACS, when any LITTLE core encounters a critical section, it requests for the big core to execute that critical section. The big core acquires the lock, executes the critical section, and notifies the requesting LITTLE core when the critical section is complete.

Explain the motivation for this design. E.g., what simplification in the previous analysis is ACS attempting to address?

5 points

[3]Suleman et al., ...

F. ACS will change coherence traffic. Contrast the coherence state transistions *and* network traffic with and without ACS for the following kinds of data.

(E.g., "at the start of every critical section, x transitions from y_L to z_L in the LITTLE's cache, and from y_b to z_b in the big's cache, producing w messages on the network. This is because ...")

For answers without ACS, assume threads are initially scheduled at random on big or LITTLE cores and do not move thereafter.

(a) Shared data (e.g., the synchronization variable).
5 points
Without ACS:

With ACS:

(b) Private data (e.g., the top of requesting thread's stack).
5 points
Without ACS:

With ACS:

G. Describe a program whose performance would improve with ACS. Explain your reasoning.
5 points

H. Describe a program whose performance ACS would degrade with ACS. Explain your reasoning.
5 points

(Bonus:) Suggest a way to fix the performance degradation in ACS.
5 points

Scratch paper

Scratch paper

Scratch paper