Week 2 Reading Guide

> ***Assigned:*** *End of Week 1*
> ***Due:*** *Before/After Week 2 Lecture*
> ***Time estimate:*** *45-60 minutes (Required) + 30 min (Optional)*

Reading 1: Memory Hierarchy

Title: "What Every Programmer Should Know About Memory" by Ulrich Drepper

Source: https://people.freebsd.org/~lstewart/articles/cpumemory.pdf

Sections to Read: 1, 2, and 3 ONLY (pages 1-23 approximately)

- Section 1: Introduction
- Section 2: Commodity Hardware Today
- Section 3: CPU Caches

Time: ~60 minutes (this is dense material)

What to Focus On

1. **Memory hierarchy diagram** (Section 2)
    - Understand the levels: registers → L1 → L2 → L3 → DRAM
    - Know approximate latencies (don't memorize exact numbers, understand the ratios)
2. **Why caches work** (Section 3.1-3.2)
    - Temporal locality
    - Spatial locality
    - Cache lines (typically 64 bytes)
3. **Cache organization** (Section 3.3)
    - Don't need to understand all details
    - Key point: understanding cache associativity helps explain why some access patterns cause thrashing
4. **Write behavior** (Section 3.3.4)
    - Write-through vs write-back
    - Why writes can be expensive

What You Can Skip (For Now)

- Detailed hardware implementation (transistor-level)
- Virtual memory details (we'll cover this later)
- NUMA details (Section 5+)
- Multi-threaded cache considerations (Section 6+) — we'll revisit in Week 3

Key Concepts to Extract

- What is a cache line and why is it 64 bytes?
- Why is accessing arr[i] followed by arr[i+1] fast?
- Why is accessing arr[i] followed by arr[i+1000] slow?

- What is the approximate ratio of L1 cache hit latency to DRAM latency?

Verification Questions

Think about these questions

Question 1: Cache Lines

What is a cache line? A typical cache line is 64 bytes. If you have an array of 8-byte integers:

- How many integers fit in one cache line?
- If you access `arr[0]`, which other elements are likely to be brought into cache?

Question 2: Latency Calculation

A program has 5% L3 cache miss rate. Given:

- L3 cache hit: 40 cycles
- DRAM access: 200 cycles

What is the effective memory access latency? Show your calculation.

Optional Deep Dive

If you want to go deeper:

1. **Continue Drepper paper** — Sections 4-6
    - Virtual memory, NUMA, multi-threaded considerations
    - Dense but valuable for understanding modern systems

How This Connects to the Lab

In Lab 1, you will:

1. Measure quicksort performance on different inputs
2. Use `perf stat` to see cache misses and branch misses
3. **Explain** why sorted input causes worst-case behavior

If you understand the reading:

- You'll know that cache misses cost ~200 cycles vs ~4 cycles for hits
- You'll understand why memory access patterns matter
- You'll be able to calculate the impact of cache miss rates

If you skip the reading:

- You'll collect numbers but not understand what they mean
- Your lab report explanations will be shallow
- You'll struggle to connect measurements to mechanisms

Reading Check (Self-Test)

Before Week 2 class, you should be able to answer:

- [ ] I know the approximate latency ratio between L1 cache and DRAM (within 10x)
- [ ] I can explain what a cache line is and why spatial locality matters
- [ ] I can calculate effective memory latency given hit rate and miss penalty