Systems Programming Concepts

Hic sunt dracones

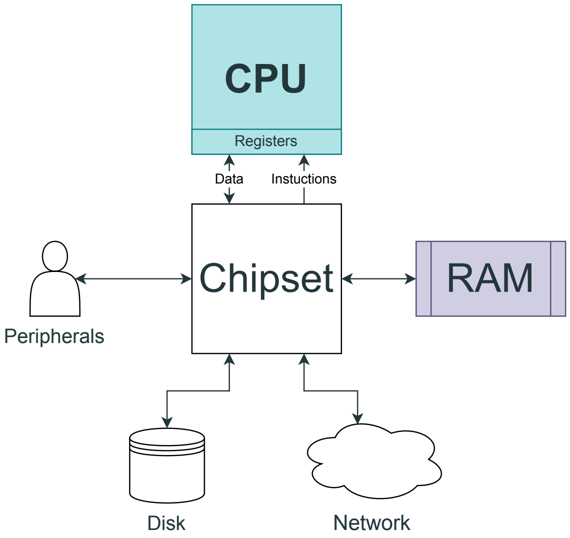
MAJ John Rollinson

2023-03-08

Army Cyber Institute

Virtual Memory

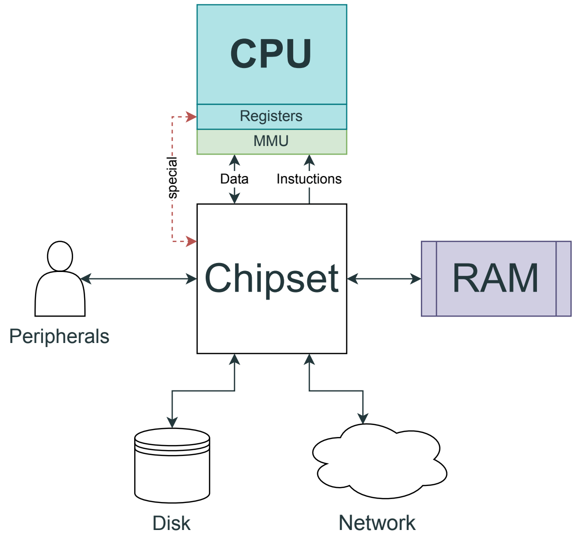
Anatomy of a computer



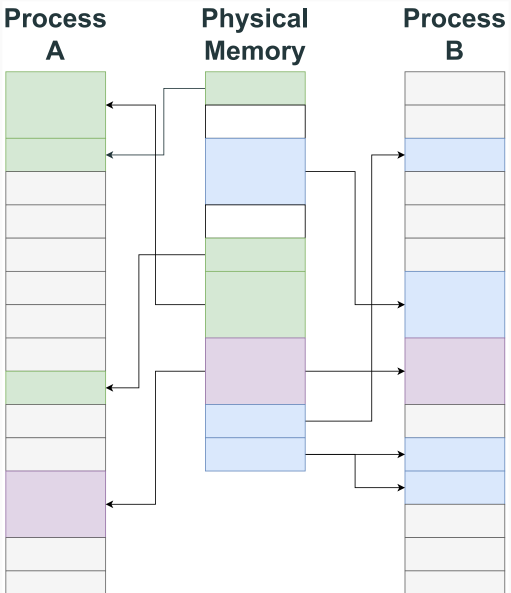
Anatomy of a computer: Managing Memory

What’s the problem with this picture?

The Memory Management Unit



The Memory Management Unit: Logical View



The Memory Management Unit: Implementations

- Segments[1]
- Page tables
  - Hashtable
  - Trees
- Software defined

[1] This isn't necessarily the same, but can achieve similar results.

The Memory Management Unit: Segments

- Effectively multiple address spaces
- Segment is a base address and length in memory
- Segments identified by extra machine registers

$$addr_p = addr_v + base_{segment}$$

The Memory Management Unit: Hashtables

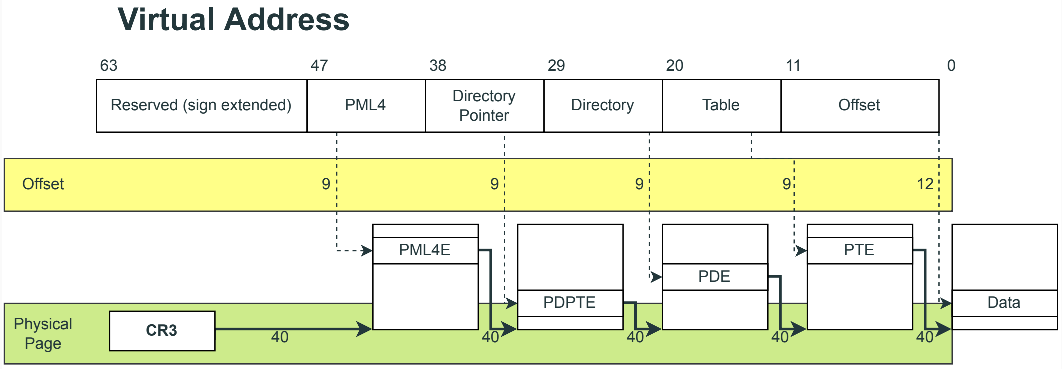
1. Lookup $addr_v$ in a small table of segments.
2. (Optional) Check a hashtable for the segment info.
3. Fault the CPU and have the OS update the segment tables.

Note: Only really used by POWER chips.

The Memory Management Unit: Trees

- Lowest-order bits are the offset into a page.
- Higher-order bits are used to walk a radix tree.

The Memory Management Unit: x86-64 Details



Exercise 2.1

Exercise: Write a library that given a "physical memory dump" and CR3 value, and whether the CPU is in supervisor-mode, can dereference logical values and update memory.

See `mmu.h` in `ex21.tar.gz`.

MMU & Performance

How many physical memory accesses does this instruction require?

```
call [rdi]
```

Caches

Fetching memory from main memory takes hundreds of CPU cycles on modern CPUs. To reduce this effect, they often have **caches** that exploit **locality** of memory accesses to retain data “closer” to the CPU where accesses can be as fast as 2-5 cycles.

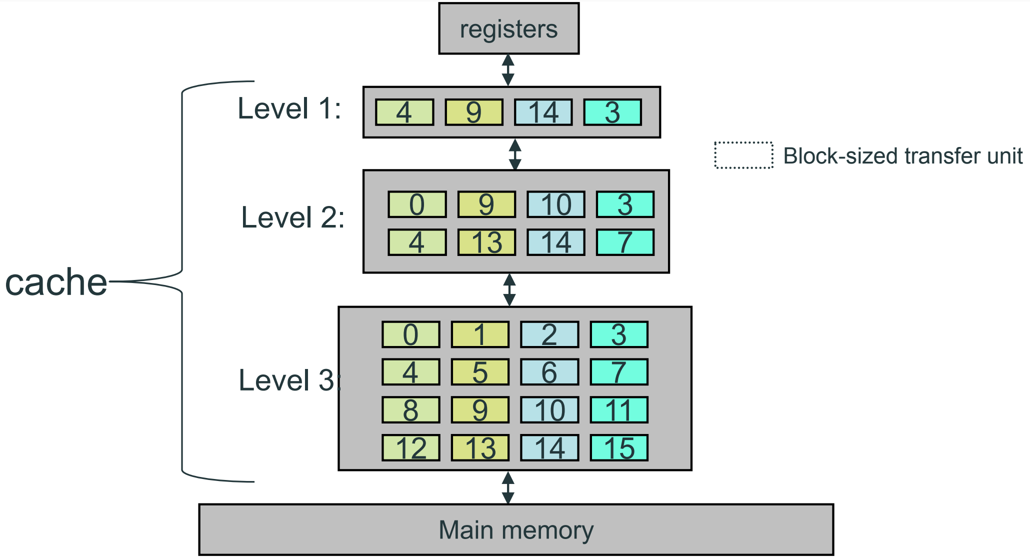
Caches: Locality

Spatial locality: Addresses of memory accesses are close

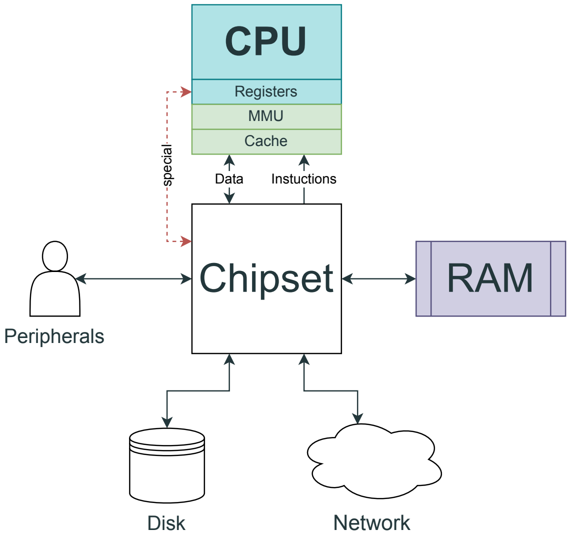
Temporal locality: The same address is accessed repeatedly

```
size_t checksum(const uint8_t *data, size_t size) {
    size_t checksum = 0;
    for (size_t index = 0; index < size, index++) {
        checksum *= SOME_CONST;
        checksum += data[index];
    }
    return checksum;
}
```

Caches: Model



Caches: Anatomy of a computer



Caches: Computational Complexity

Should we count loads/stores, comparisons, or something else? Why?

When the cache is full, does our replacement strategy matter?

Translation Lookaside Buffers (TLBs)

- Does the cache fully solve our memory translation problems?

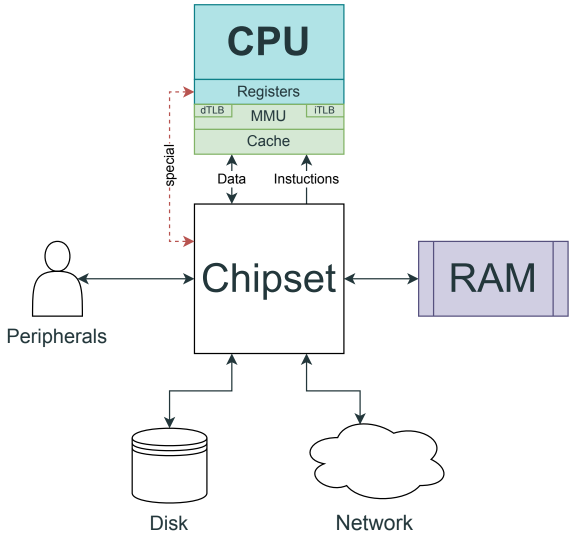
Translation Lookaside Buffers (TLBs)

- Does the cache fully solve our memory translation problems?
- TLBs are special caches for page mapping

Translation Lookaside Buffers (TLBs)

- Does the cache fully solve our memory translation problems?
- TLBs are special caches for page mapping
- Separate paths for instructions and data

Translation Lookaside Buffers (TLBs): Anatomy of a computer



Context

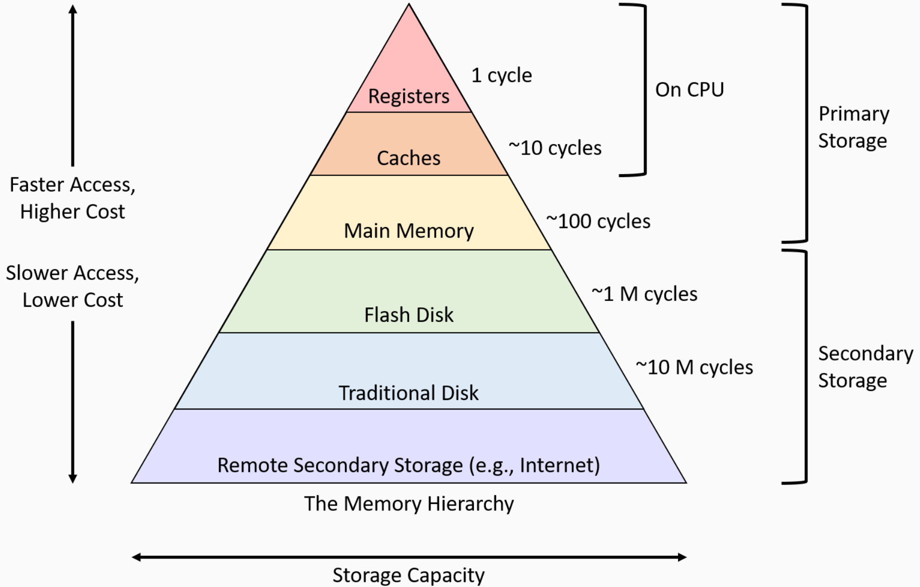
Why does all of this matter?

Exercise 2.2

Write programs to measure and determine (or approximate) the following information:

1. A cache line
2. The L1 cache (Bonus points for L1 iCache)
3. The L2 cache
4. The L3 cache
5. TLB size (bonus points for iTLB size)

Storage hierarchy



The Memory Hierarchy