Advance Caching
Speeding up Memory

• \( ET = IC \times CPI \times CT \)
• \( CPI = \text{noMemCPI} \times \text{noMem\%} + \text{memCPI} \times \text{mem\%} \)
• \( \text{memCPI} = \text{hit\%} \times \text{hitTime} + \text{miss\%} \times \text{missTime} \)

• Miss times:
  • L1 -- 20-100s of cycles
  • L2 -- 100s of cycles

• How do we lower the miss rate?
Know Thy Enemy

• Misses happen for different reasons
• The three C’s (types of cache misses)
  • Compulsory: The program has never requested this data before. A miss is mostly unavoidable.
  • Conflict: The program has seen this data, but it was evicted by another piece of data that mapped to the same “set” (or cache line in a direct mapped cache)
  • Capacity: The program is actively using more data than the cache can hold.

• Different techniques target different C’s
Reducing Compulsory Misses

- Increase cache line size so the processor requests bigger chunks of memory.
- For a constant cache capacity, this reduces the number of lines.
- This only works if there is good spatial locality, otherwise you are bringing in data you don’t need.
- If you are reading a few bytes here and a few bytes there (i.e., no spatial locality) this will hurt performance.
- But it will help in cases like this:

```c
for(i = 0; i < 1000000; i++) {
    sum += data[i];
}
```

One miss per cache line worth of data
Reducing Compulsory Misses

• HW Prefetching

```c
for(i = 0; i < 1000000; i++) {
    sum += data[i];
}
```

• In this case, the processor could identify the pattern and proactively “prefetch” data program will ask for.

• Keep track of `delta = thisAddress - lastAddress`, it’s consistent, start fetching `thisAddress + delta`.

• Current machines do this alot... Prefetchers are as closely guarded as branch predictors.

• Learn lots more in 240A, if you’re interested.
Reducing Compulsory Misses

- Software prefetching
- Use register $zero!

```c
for(i = 0; i < 1000000; i++) {
    sum += data[i];
    if (i % 16 == 0) {
        "load data[i+16] into $zero"
    }
}
```

For exactly this reason, loads to $zero never fail (i.e., you can load from any address into $zero without fear)