Last time...
- Types of cache misses: 3 C's
  - Compulsory, Capacity, and Conflict
- Write policies
  - Write through vs. write back
  - No write allocate vs. write allocate
- Multi-level cache hierarchies reduce miss penalty
  - Inclusive versus exclusive caching policy
  - Can change design tradeoffs of L1 cache if known to have L2
- Prefetching
  - Speculate future I & D accesses and fetch them into caches
  - Usefulness & timeliness

Write-Back Cache Accesses
- Write-back cache
  - Writes only go to cache (make dirty lines)
  - Upon evict, update memory
- 0 mem access
  - Write hit
- 1 mem access
  - Read miss on a clean line
- 2 mem accesses
  - Read miss on a dirty line
- Variable cycles per read/write, might complicate the pipeline control

Hardware Instruction Prefetching
Instruction prefetch in Alpha AXP 21064
- Fetch two blocks on a miss; the requested block (i) and the next consecutive block (i+1)
- Requested block placed in cache, and next block in instruction stream buffer
- If miss in cache but hit in stream buffer, move stream buffer block into cache and prefetch next block (i+2)

Hardware Data Prefetching
- Prefetch-on-miss:
  - Prefetch b+1 upon miss on b
- One Block Lookahead (OBL) scheme
  - Initiate prefetch for block b+1 when block b is accessed
  - Why is this different from doubling block size?
  - Can extend to N-block lookahead
- Strided prefetch
  - If observe sequence of accesses to block b, b+N, b+2N, then prefetch b+3N etc.
Example: IBM Power 5 [2003] supports eight independent streams of strided prefetch per processor, prefetching 12 lines ahead of current access

Software Prefetching
```
for(i=0; i < N; i++) {
    prefetch( &a[i + 1] );
    prefetch( &b[i + 1] );
    SUM = SUM + a[i] * b[i];
}
```

What property do we require of the cache for prefetching to work?
Software Prefetching Issues
- Timing is the biggest issue, not predictability
  - If you prefetch very close to when the data is required, you might be too late
  - Prefetch too early, cause pollution
  - Estimate how long it will take for the data to come into L1, so we can set P appropriately
  - Why is this hard to do?

```c
for(i=0; i<N; i++) {
prefetch(&a[i+P]);
prefetch(&b[i+P]);
SUM = SUM + a[i] * b[i];
}
```

Must consider cost of prefetch instructions

Compiler Optimizations
- Restructuring code affects the data block access sequence
  - Group data accesses together to improve spatial locality
  - Re-order data accesses to improve temporal locality
- Prevent data from entering the cache
  - Useful for variables that will only be accessed once before being replaced
  - Needs mechanism for software to tell hardware not to cache data ("no-allocate" instruction hints or page table bits)
- Kill data that will never be used again
  - Streaming data exploits spatial locality but not temporal locality
  - Replace into dead cache locations

Loop Interchange
```c
for(j=0; j<N; j++) {
  for(i=0; i<M; i++) {
    x[i][j] = 2 * x[i][j];
  }
}
```

What type of locality does this improve?

Matrix Multiply, Naïve Code
```c
for(i=0; i<N; i++) {
  for(j=0; j<N; j++) {
    for(k=0; k<N; k++)
      x[i][j] = x[i][j] + y[i][k] * z[k][j];
  }
}
```

What type of locality does this improve?
CSE 490/590 Administrivia

• Midterm on Friday, 3/4
• Project 1 deadline: Friday, 3/11
• Guest lectures possibly this month
• Course early-evaluation today

Acknowledgements

• These slides heavily contain material developed and copyright by
  – Krste Asanovic (MIT/UCB)
  – David Patterson (UCB)
• And also by:
  – Arvind (MIT)
  – Joel Emer (Intel/MIT)
  – James Hoe (CMU)
  – John Kubiatowicz (UCB)
• MIT material derived from course 6.823
• UCB material derived from course CS252