11.8

Content

- Memory Hierarchy
  - Storage
  - Locality
- Optimization
  - General optimization
  - Programmer's effot

Memory hierarchy – storage

- Disk
  - Disk architecture – track section cylinder...
  - Disk volume
  - Disk speed and latency
- Main memory
  - The architecture
  - What happens when reading and writing
- Cache

Memory hierarchy – storage: cache

- Cache size calculation
- Cache hit and cache miss
- Eviction algorithm
- Cache simulation – in midterm exam

Memory hierarchy – locality

- Temporal locality and Spatial locality
  - Difference between them – in midterm test
- Locality and memory hierarchy

Optimization – general

- Code motion
- Redundancy elimination
  - Example in ppt

Optimization – programmer perspective

- Cache friendly code
- Avoid optimization blockers
- Exploit modern processors

Cache friendly code

- Better locality
- Example: use block in GEMM
- Example: shared memory in GPU

Avoid optimization blockers

- Function calls
  - Move some function call to eliminate redundancy ON YOUR OWN
- Memory aliasing
  - Use restrict keyword in C
  - **void*** memcpy(**void** **restrict* dest, **const void** **restrict* src, **size_t** count); [since C99]
  - When calling memcpy, programmers should make sure there's no memory aliasing!
  - Otherwise, **void*** memmove(**void*** dest, **const void*** src, **size_t** count); should be used

Take advantage of modern processors

- Unroll loops
  - Reduce the loop-overhead
  - 1.27 --> 1.01
- Exploit registers and ALUs
  - More than 1 accumlators when doing reduction
  - ***Critical path***
  - Performance is limited not only by # of ALUs, but also # of regs
- Vectorization
  - Use new intrinics
- Sometimes, compiler can help us do such optimizations

Operation	Add
Combine4	1.27
Unroll 2x1	1.01