CSci 2021: Final Exam Review Lecture

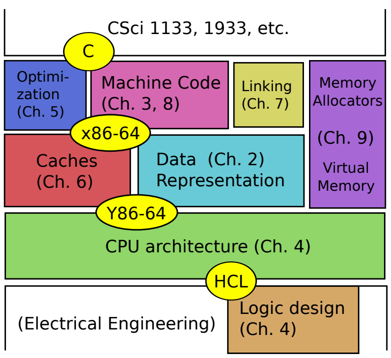
Stephen McCamant

University of Minnesota, Computer Science & Engineering

Outline

- Layered course overview
- Final exam and other logistics
- Post midterm 2 topics: caches
- Post midterm 2 topics: memory
- Post midterm 2 topics: optimization
- Post midterm 2 topics: allocation
- Post midterm 2 topics: linking

Abstraction layers (in one slide)



Implementing high-level code (1)

- Machine-level code representation
  - Instructions, operands, flags
  - Branches and loops
  - Procedures and calling conventions
  - Arrays, structs, unions
  - Buffer overflow attacks
- Code optimization
  - Machine-independent techniques
  - Instruction-level parallelism

Implementing high-level code (2)

- Linking
  - Symbols, local and global
  - Libraries and static linking
- Dynamic memory allocation
  - Heap layout and algorithms
  - Garbage collection
  - C memory-usage mistakes

What hardware does

- Number representation
  - Bits and bitwise operators
  - Unsigned and signed integers
  - Floating point numbers
- Memory hierarchy and caches
  - Disk and memory technologies
  - Locality and how to use it
  - Cache parameters and operation
  - Optimizing cache usage
- Virtual memory
  - Page tables and TLBs
  - Memory permissions and sharing

Building hardware

- Logic design
  - Boolean functions and combinational circuits
  - Registers and sequential circuits
- CPU architecture
  - Y86-64 instructions
  - Control logic and HCL
  - Sequential Y86-64
  - Pipelined Y86-64

Outline

- Layered course overview
- Final exam and other logistics
- Post midterm 2 topics: caches
- Post midterm 2 topics: memory
- Post midterm 2 topics: optimization
- Post midterm 2 topics: allocation
- Post midterm 2 topics: linking

Final exam coordinates

- Wednesday, May 13th (in 8.5 days)
- 8:00am - 10:00am (2 hours)
- Test on Canvas + Zoom attendance
- Longer than midterms, but not twice as long
- Topic coverage is comprehensive
  - Slightly more than 1/3 on topics after midterm 2
  - Expect questions that integrate ideas

Exam rules

- Begins promptly at 8:00, ends promptly at 10:00
- Open-book, open-notes, any paper materials OK
- Change from midterms: electronic resources OK
  - eTextbook, electronic notes, web searches, compiler, disassembler
  - But designed not to need them
- Still no communication with other students allowed during the exam

Why are course evaluations important?

- Help us do a better job next time
- What worked well, what not so well?
- If you were running the course, what activities would you spend more or less time on?
- I will read your written comments, after grades submitted

`https://srt.umn.edu/blue/`

Outline

- Layered course overview
- Final exam and other logistics
- Post midterm 2 topics: caches
- Post midterm 2 topics: memory
- Post midterm 2 topics: optimization
- Post midterm 2 topics: allocation
- Post midterm 2 topics: linking

RAM technologies

- SRAM: several (e.g. 6) transistors per bit
  - Faster
  - More expensive, less dense
  - Used for caches
- DRAM: one capacitor and transistor per bit
  - Must be periodically refreshed
  - Cheaper, more dense
  - Slower
  - Used for main memory

Disks and SSDs

- (Spinning) hard drives
  - Highest capacity
  - Random access time limited by seek and rotation latencies
  - Always read or write an entire sector at a time
- Solid-state (flash) drives
  - Technology descended from EEPROMs
  - Random-access reads are very fast
  - Can only rewrite by erasing large blocks
  - Random-access writes require recopying, slower

Spatial and temporal locality

- Spatial locality: memory accesses are close together in location
  - Best case: sequential accesses
- Temporal locality: the same location is accessed repeatedly close together in time
  - Set of locations being used is called the *working set*
- Because of locality, different locations have very different chances of being accessed next

Memory hierarchy

- Devices have trade-off between access time and capacity
  - Differences of many orders of magnitude
- Combine small+fast devices with big+slow ones in a hierarchy
- Because of locality, most uses are in small+fast device
- Must move data between levels
  - Keeping a copy at a higher level is called *caching*
  - First example: caches between CPU core and memory

Cache parameters

- Data is moved in blocks of size $B = 2^b$
- Organize cache into $S = 2^s$ sets of lines
- A set contains $E = 2^e$ lines, each of which can contain one of the same blocks
  - $E = 1$: direct mapped
  - $E > 1$: E-way set associative
  - $S = 1$: fully associative
- Total capacity $C = S \cdot E \cdot B$
- b and s also give a division of addresses into $m = t + s + b$

Cache operations: read

- Use s bits as an index to choose a set
- Check all lines in the set (hardware: in parallel), to see if any is valid and has a matching tag
- If yes, it's a *hit*: block offset indicates which bytes desired
- If not present, it's a *miss*
  - Fetch data from lower level (e.g., main memory)
  - Insert newly read data, usually *evicting* another block

Cache operations: write

- Look for a matching line as for a read
- If a hit, update contents of cache block
  - *Write-back* policy: do not copy to lower levels until evicted (opposite is write-through)
- If a miss, the common *write-allocate* policy copies the block into the cache
  - Exploits locality in write-only accesses

Cache usage optimizations

- Overall goals: maximize locality, minimize working set
- Use more compact data representations
- Prefer stride-1 data accesses
  - E.g., for a matrix, iterate over indexes in outer-to-inner order
- Temporally group accesses to the same data values
  - For 2-D data, group by blocks (tiles) instead of rows

Outline

Layered course overview

Final exam and other logistics

Post midterm 2 topics: caches

Post midterm 2 topics: memory

Post midterm 2 topics: optimization

Post midterm 2 topics: allocation

Post midterm 2 topics: linking

Virtual memory structures

- Pages are units of data transfer (e.g., 4KB)
  - Can be in RAM or on disk
- Page table maps virtual addresses to physical pages
  - For efficiency, use multiple levels
- A TLB is a cache for page-table entries

Virtual memory uses

- Avoid capacity limits on RAM
- Cache data from disk for speed
  - Demand paging of code
- Implement isolation between processes
  - Separate page tables
  - User/kernel protections
- Share reused data
  - Executable code, shared libraries

Outline

Layered course overview

Final exam and other logistics

Post midterm 2 topics: caches

Post midterm 2 topics: memory

Post midterm 2 topics: optimization

Post midterm 2 topics: allocation

Post midterm 2 topics: linking

Principles of optimization

- Concentrate on the program parts that run the most
  - Amdahl's law bounds possible speedup
  - Array-style programs: concentrate on inner loops
  - Complex programs: use a profiler
- Know what the compiler can and can't do
  - Compiler can be smart, but is careful about correctness
  - Functions and pointers (aliasing) block optimization
- Watch out for algorithmic problems

Machine-independent optimizations

- Move computations out of loops
- Avoid abstract functions in time-critical code
- Use temporary variables to reduce memory operations
- Unroll loops to reduce bookkeeping overhead
- Avoid unpredictable branching

Instruction-level parallelism

- Modern processors are *super-scalar*
  - Can do more than one thing at once
- And *out-of-order*
  - In a different sequence than the original instructions
- Multiple *functional units*, each with different throughput and latency

Exposing loop parallelism

- To reduce latency, avoid a long *critical path*
- Functional unit throughput is an ultimate limit
- Unroll to allow optimization between iterations
- Techniques to shorten the critical path:
  - Re-associate associative operators
  - Replace a single accumulator with multiple parallel accumulators

Outline

Layered course overview

Final exam and other logistics

Post midterm 2 topics: caches

Post midterm 2 topics: memory

Post midterm 2 topics: optimization

Post midterm 2 topics: allocation

Post midterm 2 topics: linking

Implementing `malloc`

- Data structures to represent the heap
  - Boundary tags and the implicit list
  - Explicit free list(s)
- Algorithms for heap management
  - First fit vs. best fit
  - Size segregation

Outline

- Layered course overview
- Final exam and other logistics
- Post midterm 2 topics: caches
- Post midterm 2 topics: memory
- Post midterm 2 topics: optimization
- Post midterm 2 topics: allocation
- Post midterm 2 topics: linking

Linking mechanics

- Symbols include functions and variables
  - Some are file-local, stack variables not even considered
- Symbols are resolved to the correct definition
  - At most one strong definition, or one of many weak ones
- Code is relocated so it runs correctly at its final address