**Abstraction layers (in one slide)**

- CSci 1133, 1933, etc.
- Machine Code (Ch. 3, 8)
- Linking (Ch. 7)
- Memory Allocators (Ch. 9)
- Caches (Ch. 6)
- Data (Ch. 2)
- Representation
- Virtual Memory
- CPU architecture (Ch. 4)
- Logic design (Ch. 4)

**Implementing high-level code (1)**

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

**Implementing high-level code (2)**

- Linking
  - Symbols, local and global
  - Relocation and position-independent code
  - Libraries, static and dynamic
- 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
Final exam coordinates
- Thursday, May 12th (week from tomorrow)
- 10:30am - 12:30pm (2 hours)
- 3-210 Keller Hall (same room as lecture)
- Longer than midterms, but not twice as long
  - Last year’s was six questions
  - Topic coverage is comprehensive
    - About 1/3-1/4 on topics after midterm 2
    - Expect questions that integrate ideas

Exam rules
- Begins promptly at 10:30, ends promptly at 12:30
- Open-book, open-notes, any paper materials OK
- No electronics: no laptops, smartphones, calculators, etc.
  - Arithmetic will use easy numbers, but know your powers of two
- Sit in alternating seats as long as possible

Exam strategy suggestions
- Writing implement: mechanical pencil plus good eraser
- Make a summary sheet to save flipping through notes or textbook
- Show your work when possible
- Do the easiest questions first
- Allow time to answer every question
- Sample final posted (from last year), solutions later

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

Memory allocation
- 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
- Memory errors in C code
  - Alternative: garbage collection

Outline
Layered course overview
Final exam logistics
Post midterm 2 topics: memory
Post midterm 2 topics: optimization
Post midterm 2 topics: linking
Review questions

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 logistics
- Post midterm 2 topics: memory
- Post midterm 2 topics: optimization
- Post midterm 2 topics: linking
- Review questions

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

Libraries

- Collections of reusable code
- Static libraries
  - Several .o files grouped together
  - Only needed files are selected
  - Copied into executable just like other object files
- Dynamic shared libraries
  - Not loaded until program startup or later
  - Single copy can be used by different programs
  - Uses position-independent code

Smallest and largest integers (1)

Match the symbolic name to its hex representation

1. TMin
2. TMax
3. UMin
4. UMax

A. 0x0000...000
B. 0x1000...000
C. 0x8000...000
D. 0xf000...000
E. 0x0fff...fff
F. 0x1fff...fff
G. 0x7fff...fff
H. 0xefff...fff
I. 0xffff...fff

https://chimein.cla.umn.edu/course/view/2021
Smallest and largest integers (2)

Put the values in order according to unsigned comparison $< u$:
A. $\text{UMin} < u \text{TMin} < u \text{UMax} < u \text{TMax}$
B. $\text{TMin} < u \text{UMin} < u \text{UMax} < u \text{TMax}$
C. $\text{TMin} < u \text{UMin} < u \text{TMax} < u \text{UMax}$
D. $\text{UMin} < u \text{TMax} < u \text{TMin} < u \text{UMax}$
E. $\text{UMin} < u \text{TMin} < u \text{TMax} < u \text{UMax}$

Smallest and largest integers (3)

Put the values in order according to signed comparison $< s$:
A. $\text{UMin} < s \text{TMin} < s \text{UMax} < s \text{TMax}$
B. $\text{TMin} < s \text{UMin} < s \text{UMax} < s \text{TMax}$
C. $\text{TMin} < s \text{UMin} < s \text{TMax} < s \text{UMax}$
D. $\text{UMin} < s \text{TMax} < s \text{TMin} < s \text{UMax}$
E. $\text{UMin} < s \text{TMin} < s \text{TMax} < s \text{UMax}$

Floating point properties

Suppose that $a$, $b$, and $c$ are double-precision floating point values, and not NaN. Which of the following properties are always true?
1. $a + (b + c) = (a + b) + c$
2. $a + (b + a) = (a + b) + a$
3. $a + (a + a) = (a + a) + a$
4. $a \times a \geq a$
5. $a \times a \geq 0$
6. $a / a = 1$

Calling conventions

According to the standard x86-64 calling convention, which of these registers would your function need to save before modifying it?
A. %rdi
B. %rsi
C. %r10
D. %rbx
E. %rax

x86-64 instructions

Which two instructions can be used to compare %rax to zero?
A. cmp $0$, %rax and test $0$, %rax
B. cmp $0$, %rax and test %rax, %rax
C. cmp %rax, %rax and test $0$, %rax
D. cmp %rax, %rax and test %rax, %rax

for loops

Which of these while loop patterns is equivalent to the loop for (A; B; C) { D; }?
A. A; while (B && C) { D; }
B. B; while (A) {D; C}
C. A; while (B) {C; D}
D. A; while (B) {C; D; C}
E. A; while (B) {D; C}
Structure padding

Because of padding, which of these structs would not be the same size as the others?
A. struct { short s; long l; }
B. struct { float f; double d; }
C. struct { char c; long l; }
D. struct { long l1; long l2; }
E. struct { int i1; int i2; }

Y86-64 instructions

Which of these Y86-64 instructions is an indirect jump?
A. call
B. ret
C. jmp
D. jle
E. jne

Kinds of cache misses

Match each kind of cache miss with how you fix it:
1. Compulsory miss
   A. Increase the size of the cache
2. Capacity miss
   B. Unavoidable
3. Conflict miss
   C. Increase the cache associativity