FINAL EXAM GUIDE

The exam is open book and open note, and focuses on material covered in the lectures, labs, assignments, and additional readings. The exam questions will require you to demonstrate a good understanding of the key concepts and the ability to analyze a particular situation and apply your knowledge.

Material Covered: The second half the class concentrates on the following three modules:

1. x86-64 Runtime Stack,
2. Cache Memories,
3. Debugging, Design and Code Optimization,
4. Linking,
5. Heap Allocators

Hence, the final exam will cover all materials contained in Lectures 24-35. Note that, however, some of the questions may require some knowledge about the first half of the class. Specifically, the topics covered in the final exam are listed in detail below:

x86-64 Runtime Stack

- Lecture 24: x86-64 Procedures
  revisiting %rip, the stack, passing control, caller and callee functions, call instruction
- Lecture 25: More on x86-64 Procedures
  push and pop instructions, passing data, local storage, register restrictions, caller-owned vs callee-owned registers
- Lecture 26: Data and Stack Frames
  implementing one-dimensional, multi-dimensional and multi-level arrays, structures and alignment, floating point instructions (covered in Lecture 27)
- Lecture 27: Security Vulnerabilities
  memory layout, buffer overflow, buffer overflow attacks and defences (covered in Lecture 28)

Cache Memories

- Lecture 28: Buffer overflow Attacts, The Memory Hierarchy
  storage technologies and trends
- Lecture 29: Locality and the Memory Hierarchy
  principle of locality (temporal locality, spatial locality), caching in the memory hierarchy
- Lecture 30: Cache Memories
  cache memory organization, the memory mountain, rearranging loops to improve spatial locality, using blocking to improve temporal locality

Debugging, Design, and Code Optimization

- Lecture 31: Debugging and Design
  defects and failures, debugging, debugging tools, managing the complexity, communication, naming, comments,
- Lecture 32: Code Optimization
  what is optimization, constant folding, common sub-expression elimination, dead code, strength reduction, code motion, tail recursion, loop unrolling, limitations of gcc code optimization

Linking

- Lecture 33: Linking
  static linking, why we need linkers, what do linker do, ELF object file format, symbol resolution, relocation, static libraries, shared libraries, library interpositioning

Heap Allocators

- Lecture 34: Managing the Heap
  what is a heap allocator?, heap allocator requirements and goals, fragmentation, implementing heap allocators, method 0: bump allocator
- Lecture 35: More on Managing the Heap
  method 1: implicit free allocator, method 2: explicit free allocator, coalescing, in-place realloc