CSSE 132 – Introduction to Systems Programming
Rose-Hulman Institute of Technology

Exam 1 Review Guide

This exam measures your mastery of these learning objectives:

1. Describe the functions of common computer system hardware elements including CPU, memory hierarchy and input/output devices.
2. Implement and analyze software in the C programming langauge using:
   - Standard C data types
   - Binary arithmetic, boolean and logical operations
   - Functions, Arrays, C Strings
   - Pointers and Pointer Arithmetic (including Function Pointers)
3. Demonstrate ability to perform tasks like these in a variety of operating environments including the Linux system environment:
   - compile and debug software
   - leverage a version control system
   - manipulate data
   - command-line (shell) navigation and manipulation

1 Format

Two-part exam. 2h50m total time (lab period)

- Part 1 (paper). By-hand problems. **Closed resources**. You are allowed to use one single-sided 8 1/2 by 11 inch sheet of hand-written notes and a calculator. A simplified ARM guide is provided on the last page of the paper part for your reference.
- Part 2 (coding). You are allowed to use only these acceptable resources:
  - Your computer
  - Your assignments and labs submitted in your individual repository for this term
  - The CSSE 132 course website and things directly linked from it

You are not allowed to use other Internet resources, instant messaging, your smartphone, or other communication means during any part of the exam. Use of other resources is considered academic dishonesty and will result in a penalty grade.

2 Topics to study

- Numbers
  - Number representation in binary, hexadecimal, and decimal.
  - Conversions from one number system (binary, hex, decimal) to another.
  - Two's complement
- Memory
  - Memory addressing, endianness
  - Memory hierarchy: types of memory (DRAM, SRAM, SSD/Flash, Hard Disk)
  - Effective access time (given a cache strategy and hit rate)
- Command Line Interface
  - Access your Linux
  - Create and edit files. Use Git to obtain, track, and store files and changes.
  - Use basic command line tools (like cd, cat, ls, grep, vim or emacs) to navigate, search and manipulate files.
  - Compiling, running, and debugging Assembly and C programs
- ARM Assembly
  - Reading assembly: Given code, what does it do? Write corrsponding C code
  - Writing assembly: basic instructions, pointers, branches, loops
  - Conventions for procedure call, return address, arguments, return value
- C
  - Pointers, arrays, strings
  - Basic function calls
  - C data types such as `int`, `char`, pointers
  - printf and format specifiers
  - structs
  - Pass by value vs. "pass by reference" using pointers

3 Problem-by-Problem Review

Paper Part (45 pts)

1. Problem 1: Multi-choice (4 problems × 3 = 12 pts)
   - 1.1 Assembly *[Quiz 04, ldr/str]*
   - 1.2 Compilation Steps *[Quiz 04]*
   - 1.3 C struct *[Quiz 08]*
   - 1.4 C string *[Quiz 07]*
2. Problem 2: Number conversions (10 pts) *[Quiz 02: Bin/Dec/Hex, signed/unsigned]*
3. Problem 3: Endianness (6 pts) *[Quiz 04]*
4. Problem 4: Memory Addressing (5 pts) *[Quiz 03]*
5. Problem 5/6: Memory Hierarchy (12 pts) *[Quiz 09: SRAM/DRAM/SSD]*

Coding Part (55 pts)

1. Problem 1/2 (10 pts) Linux commands *[Lab1 Bandit, Lab 2 mystery, sample exam]*
2. Problem 3 (45 pts) *[HW2/3 coding part]*
   - 2.1 `problems.s`: 3 functions (20 pts)
     - (5 pts) Accessing memory *[ldr/str]*
     - (7 pts) Writing a loop *[cmp, b__]*
     - (8 pts) Writting a loop to read an int array *[Quiz 05]*
   - 2.2 `problems.c`: 4 functions (25 pts)
     - (5 pts) Array
     - (5 pts) String/pointers
     - (7 pts) `printf` with format specifiers
     - (8 pts) `struct` *[Quiz 08]*