Computer Architecture and Systems Programming Lab

Course Requirements

1. Prerequisites: Digital Systems (361-13591), Introduction to Computers (in parallel), Systems Programming.
2. Credits: 4 (for 2.5 lecture hours, 1 exercise session, 3 lab hours).
3. Grading (approximate):
    - **Architecture and SP lab**: 55% final exam (**must pass** final exam to pass the course!), 15% homework assignments, 30% lab grades (lab attendance mandatory! Must pass at least 7 labs to pass the course.).
    - Students in **SPlab** only: 30% final exam (at same time as architecture exam, **must pass** final exam to pass the course! Must pass at least 7 labs to pass the course.), 70% lab grades (lab attendance mandatory! Must pass at least 7 labs to pass the course.).
    - Students in 2.5 credit computer **architecture** only: 70% final exam (**must pass** final exam to pass the course!), 30% homework assignments.

No cheating! You are required to get a non-zero grade on all assignments and labs in order to pass the course. An unsubmitted assignment or lab gets 1/100. An assignment too similar to someone else's assignment (i.e. cheating) gets you 0/100 and no credit in the course.

Syllabus and Detailed Description

Topics Covered in the Course (Lectures and Practical Sessions)

1. Review of basic architecture
2. Machine and assembly languages (generic)
3. The process of assembly and linking
4. Special assembly language programming issues
    1. Parameter passing: assembly language and "external"
    2. Return addresses and co-routines
    3. Traps and interrupts: mechanism and handling
    4. Worms and viruses: malicious code and defending against it.

5. Architecture design considerations - CISC vs. RISC
6. Special miscellaneous topics
    1. Operating system interface: traps and interrupts *
    2. Access to IO devices: display, keyboard, real-time clocks, DMA *
    3. Communications (serial and parallel), simple error detection and correction codes (parity, Hamming), handshakes.

Topics Covered in the Labs (including special lectures)

The Lab will be based on a LINUX platform, and use the C programming language. The emphasis is on low-level programming. Goals of this lab are to introduce issues in low level programming, as well as techniques on how to learn needed information on demand. The following topics will be covered via hands-on experience during this course:

1. Low level programming in C. This includes all sorts of "tricks" that emphasise the power of low-level computing, as an aid to understanding computing systems in depth:
    1. Pointers to functions and their applications.
    2. Self modifying code and applications.
2. Binary files of various types: structure and processing.
    1. Maintaining data structures in files (e.g. pointers to structtures, Linux directories).
    2. Object and executable files (demonstrated through ELF files).
    3. Linking and Loading.
3. Using operating systems services (system calls):
    1. Process control: creating and terminating processes, process control, signals. Will be introduced by programming a simple shell.
    2. System-level Input/Output: read, write files, file metadata, sharing files.
4. Issues in program developement:
    1. Debugging programs, and the effect of compound bugs (e.g. various types of memory leaks, compiler bugs).
    2. Patching and hacking.

Sourcebooks and Written Material

1. NASM online manual.
2. IBM PC Assembly Language Programming, Peter Abel, Prentice Hall (second edition 1991)
3. D. A. Patterson and J. L. Hennessey, Computer Organization & Design: The Hardware Software Interface, Morgan Kauffman, second edition 1998.

Back to BGU CS HomePage