ECE XX/XX (TBD) Syllabus

Computer System and Software

1. **Description**: Programming is an abstract endeavor, but that abstraction hides noticeably more detailed and meticulous low-level implementations. Computer systems programming involves developing software to connect the low-level computer hardware to high-level and user-facing application software, and usually requires careful consideration of performance and resource constraints. Computer systems software includes compilers, operating systems, databases, numerical libraries, and embedded controllers. This course introduces computer systems from a programmer's perspective, rather than a system implementer's perspective, which prepares students for more advanced topics that discuss the internals of a computer system. It covers computer systems hardware organization and the programmer interface with the goal of improving students' abilities to effectively reason about the execution of their programs, write system software, and enhance the performance of the programs. This course aims to provide a strong foundation in the art, principles, and practices of computer systems programming using C/C++ and the languages of choice for system-level programmers, so it has hands-on discussion sections as well as theoretical and programming assignments for students to apply their learned principles and gained skills in real applications. In overall, the course shows how software and hardware interact and demonstrates that understanding lower-level implementation can often lead to much higher quality software and user experiences.

2. **Number of Credits**: Three.

3. **Instructor**: Dr. Shayan (Sean) Taheri.

4. **Instructor University Role**: TBD in the Department of Electrical and Computer Engineering at the Gannon University.

5. **Office**: TBD.

6. **Phone**: TBD.

7. **Email**: TBD.

8. **Lecture and Laboratory Date and Time**: TBD.

9. **Lecture Location**: TBD.

10. **Laboratory Location**: TBD.

11. **Office Hours**: TBD (or by appointment).

12. **Teacher Assistant(s)**: TBD.

13. **Prerequisites**: Solid knowledge of C/C++ and Assembly programming languages, algorithms, and data structures.

14. **Textbook**:

A. Bryant, R.E., David Richard, O.H. and David Richard, O.H., 2003. ***Computer systems: a programmer's perspective*** (Vol. 2). Upper Saddle River: Prentice Hall.

B. Kernighan, B.W. and Ritchie, D.M., 1988. ***The C programming language***. Pearson Education.

15. **Class Requirements**: Please bring a paper notebook or a laptop to your lectures for taking notes as well as working on short in-class exercises.

16. **References (Optional)**:

A. Stevens, W.R., Rago, S.A. and Ritchie, D.M., 1992. ***Advanced programming in the UNIX environment*** (Vol. 4). New York.: Addison-Wesley.

B. Selected material from recent publications.

17. **Learning Objectives**: By the end of the course, you should be able to do the following functions:

A. Convert data to different representations.

B. Reverse-engineer machine code and assembly code to a behavioral (i.e., high-level) description.

C. Translate high-level software code to corresponding instruction sequences.

D. Assess the cache performance of a system given its memory/cache specifications and a specific address trace.

E. Identify various software vulnerabilities and how they may be exploited.

F. Manually simulate address translation as a means of understanding hardware and software components that do likewise.

G. Experiment to determine efficient storage (specifically heap memory) allocation strategies.

H. Organize code and use compiler settings to achieve enhanced performance on specific processor architectures.

18. **Course Webpage**: "TBD" (in "**https://my.gannon.edu**").

19. **Assignments**: The assignments are provided (approximately) on a biweekly basis to improve the learning process. A number of assignments involve computer-aided tools and/or programming.

20. **Computer Usage**: This course involves sufficient utilization of computer-aided tools and programming in C/C++ and Assembly languages.

21. **Exams and Quizzes**: There are a midterm and a final exam for overall evaluations. Unannounced quizzes may be considered for certain assessments. Access to books and/or notes along with simple calculators "might" be allowed.

22. **Grading Policy**: The following weights are used:

A. Assignments: 45%.

B. Midterm Exam: 25%.

C. Final Exam: 25%.

D. Lecture Class Participation: 5%. Laboratory Class Participation is "Mandatory".

Grading is based on a conventional fixed scale, and grades are "**A**": 90-100%; "**A-**": 85-89%; "**B+**": 80-84%; "**B**": 70-79%; "**B-**": 65-69%; "**C+**": 60-64%; "**C**": 55-59%; "**C-**": 50-54%; "**D+**": 47-49%; "**D**": 44-46%; "**D-**": 40-43%; and "**F**": 0-39%.

The instructor reserves the right to curve up the final scores.

23. **Late Policy**: Late submissions are not accepted without prior approval by the instructor.

24. **Disabilities**: In cooperation with the designated centers, reasonable accommodation is provided for qualified students with disabilities. Please meet with the instructor during the first week of semester to discuss possible arrangements.

25. **Cheating and Plagiarism**: Cheating and plagiarism are not permitted in any form and cause certain penalties. The instructor reserves the right to fail culprits.

26. **Course Outline**: The following topics are covered with certain variations:

A. ***A Tour of Computer Systems*** (Overview): Physics, Transistors, Moore's Law, Processor, Operating System, and Program.

B. ***Representing and Manipulating Information***: Information Storage, Integer Representations and Arithmetic, and Floating-Point Representation and Arithmetic.

C. ***Machine-Level Representation of Programs***: Overview, Data Formats and Operations, Control, Procedures, Memory Allocations and Access, and Advanced Codes.

D. ***Processor Architecture***: The Y86 Instruction Set Architecture, Logic Design and Hardware Control Language, Sequential Y86 Implementations, Pipelining, and Pipeline Y86 Implementations.

E. ***Optimizing Program Performance***: Compiler Optimization, Program Performance, Program Inefficiencies, Modern Processors, Loop Unrolling, Enhancing Parallelism, Memory Performance, and Performance Bottlenecks.

F. ***The Memory Hierarchy***: Storage Technologies, Locality, The Memory Hierarchy, Cache Memories, and Cache-Friendly Code.

G. ***Linking***: Compiler Drivers, Static Linking, Object Files, Relocatable Object Files, Symbols and Symbol Tables, Symbol Resolution, Relocation, Executable Object Files, Linking with Shared Libraries, Position-Independent Code, and Tools for Object File Manipulation.

H. ***Exceptional Control Flow***: Exceptions, Processes, System Call Error Handling, Process Control, Signals, Non-Local Jumps, and Tools for Process Manipulation.

I. ***Virtual Memory***: Physical and Virtual Addressing, Address Spaces, Virtual Memory (VM) for Caching, VM for Memory Management, VM for Memory Protection, Address Translation, Memory Mapping, Dynamic Memory Allocation, Garbage Collection, and Memory-Related Program Bugs.

J. ***System-Level Input/Output***: Unix Input/Output (I/O), Operations on Files, I/O Redirection, and Standard I/O.

Meanwhile, all the mentioned topics along with more complementary and advanced subjects will be covered if time permits.