CS 3432 Computer Organization and Design (Arch 1)
Fall 2021 Syllabus
Lecture: CRN 17591; TR 9:00-10:20am; Geology 123
Lab: CRN 17602; TR 10:30-11:50am; CCSB 1.0704
Instructor: Shirley Moore
Course webpage: http://svmoore.pbworks.com/

CS 3432 is a first systems course about how computer systems work. Whereas computer architecture describes the functional behavior of a computer, computer organization describes the structural relationships. CS 3432 is a course in computer organization that answers questions such as the following:

1. How are programs written in a high-level language, such as C or Java, translated into the language of the hardware, and how does the hardware execute the resulting program?
2. What is the interface between the software and the hardware, and how does software instruct the hardware to perform the necessary functions?
3. What determines the performance of a program, and how can hardware designers or programmers improve the performance?
4. What are the reasons for and the consequences of the recent switch from sequential to parallel processing?

- **Teaching Team**
  - Instructor: Shirley Moore, svmoore@utep.edu
    - Office: CCSB 3.0608
    - Office phone: 915-747-5054
    - Office hours: T 1:30-2:30pm, W 12:00-1:00pm, others by appointment (in person or virtual)
    - Webpage: http://www.cs.utep.edu/svmoore
  - Teaching Assistant: Steven Ibarra, sibarra7@miners.utep.edu
  - Instructional Assistants: Esteban Munoz, emunoz22@miners.utep.edu
    Alan Perez, aeperez8@miners.utep.edu

- **Prerequisites**
  - To succeed in this course, you need familiarity and maturity with concepts and techniques taught in digital design (EE 2369/2169), elementary data structures and algorithms (CS 2401), and discrete math (Math 2300).
  - As stipulated in the course catalog, the usual way to demonstrate this familiarity is by earning a C or above in (1) all of these courses and (2) Data Structures (CS 2302).
  - Students who earn B's or above in EE 2369/2169, CS 2401, and Math 2300 and are taking CS 2302 concurrently, are also considered ready.

- **Lab Section**
  - Students **must** enroll in the associated lab section.
  - Lab sessions and assignments will be managed by the TA.
  - **Participation is mandatory.** It is extremely easy to fall behind and imperative that you make arrangements with the instructor or TA to make-up missed lessons and work. Students at risk of failing due to not engaging with the lab section may be dropped.
  - There will be scheduled and unscheduled quizzes in the lab section.

- **Texts and Readings**
  - Required: David A. Patterson and John L. Hennessy. Computer Organization and Design RISC-V Edition: The Hardware Software Interface, Second edition, Morgan Kaufmann, ISBN: 978-0128203316.
  - Required: Kernighan, Brian W & Ritchie, Dennis M. The C Programming Language, Second edition, Prentice Hall, ISBN: 0-13-115817-1.
  - Other readings and resources will be posted on the course webpage.
- **Learning Outcomes**
  - Learning outcomes are in the appendix at the end of this document.
- **Computers**
  - You will need access to a computer capable of running Linux (either natively or using virtualization software) to participate in class activities. You will need to install an ANSI C compiler prior to the start of lab sessions the second week of class. Should you not have access to an appropriate machine, the department has a limited number of portable computers that can be lent out to students.
- **Attendance**
  - Attendance is mandatory at all lecture and lab sessions unless special circumstances are arranged ahead of time with the instructor, or as soon afterwards as possible in the event of an emergency. Lecture sessions will be recorded, but viewing the recorded session is not a substitute for attending class. Breakout groups and hands-on exercises will be part of lecture sessions and it will not be possible to records those portions.
- **Grading Breakdown**

  1000-900 = A 899-800 = B 799-700 = C 699-600 = D 599 and Below = F
  - Lab assignments and quizzes: 40% (400 points)
  - Class participation and homework: 20% (200 points)
  - Midterm exam: 20% (200 points)
  - Final exam: 20% (200 points)
- **Course Schedule**
  - The course schedule of topics and assignments will be posted on the course website.

Homework, Tests, and Labs

- **Homework**
  - Students are expected to review topics taught in class, work on solutions to assigned problems, and be able to demonstrate skills and solutions during class. Homework assignments will be posted on Teams with instructions on how to submit solutions.
- **Exams and quizzes**
  - Quizzes
    - Quizzes assess students' abilities to demonstrate knowledge, to design solutions to realistic problems, and to present these solutions in a clear and professional fashion.
    - Quizzes can cover any concept or skill previously taught in the course, are generally offered at the rate of one per week and are generally unannounced (so students must be continuously prepared).

- Midterm and Final Exams
    - The midterm exam will occur approximately midway through the course.
    - The final exam date is scheduled by the university based on lecture time.
    - Like quizzes, exams may cover any concept or skill previously taught in the course.
- The following are prohibited during quizzes and exams unless unambiguous and explicit permission is provided by the instructor:
    - Collaboration or communication with others.
    - Looking up answers in books, on the Internet, etc.

- **Labs**
    - Labs are intended to provide an opportunity for students to explore and practice the use of tools and to apply concepts presented in class within the context of programming projects.
    - Lab assignments will be posted on Teams.
    - Lab assignments will be graded during in-person demonstrations with individual students.
        - Demonstrations must occur within one week following the lab due date.
        - Demonstrations may require students to modify their programs and demonstrate competency with development tools.
    - Students are expected to act professionally
        - By reading and studying relevant resources
        - By attributing credit to any person or reference materials that substantively contributed to their solutions
        - By only submitting solutions they fully understand
        - Professionalism includes honesty, clarity, and accuracy.
        - Students are encouraged to help each other develop and tune their lab projects. There is no penalty for collaboration as long as it is documented.
    - Requirements
        - **Functional:** Assignments will require students to create complete programs or designs or modify programs or designs provided by the instructor.
        - **Documentation:** Submissions should include documentation that facilitates the grader's determination of
            - How to compile, use, and test
            - Principles of operation (e.g., comments and other descriptive prose)
            - Elements of the submission that were developed by others. This includes both algorithms and code. Vague attributions of credit (e.g., "Assistance was received from Jim Smith.") are unacceptable.
        - **Completeness:** Students whose labs do not substantially satisfy functional and documentation requirements will receive no credit for that lab.

Disabilities and Accommodations

- If you have a disability and accommodations, please contact The Center for Accommodations and Support Services (CASS) at 747-5148, or by email to **cass@utep.edu**. For additional information, please visit the CASS website at **www.sa.utep.edu/cass**.

Academic Honesty

- Students are expected to conduct themselves in a professional and courteous manner, as prescribed by the Standards of Conduct: https://www.utep.edu/hoop/section-2/student-conduct-and-discipline.html
- Submitted work should be unmistakably your own. You may not transcribe or copy a solution taken from another person, book, or other source (e.g., a web page). Copying other's work will not be tolerated. Professors are required to report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.
- Permitted collaboration: Students may discuss requirements, background information, test sets, solution strategies, and the output of their programs. However, implementations and documentation must be their own creative work. Students are required to document advice received from others and all resources utilized in the preparation of their assignments.
- If academic dishonesty is suspected: The Dean of Students office will be contacted for adjudication. A temporary "incomplete" grade will be issued if their investigation extends beyond the grading period.

COVID-19 Precaution Statement

- Please stay home if you have been diagnosed with COVID-19 or are experiencing COVID-19 symptoms. If you are feeling unwell, please let me know as soon as possible, so that we can work on appropriate accommodations. If you have tested positive for COVID-19, you are encouraged to report your results to covidaction@utep.edu, so that the Dean of Students Office can provide you with support and help with communication with your professors. The Student Health Center is equipped to provide COVID-19 testing.
- The Center for Disease Control and Prevention recommends that people in areas of substantial or high COVID-19 transmission wear face masks when indoors in groups of people. The best way that Miners can take care of Miners is to get the vaccine. If you still need the vaccine, it is widely available in the El Paso area, and will be available at no charge on campus during the first week of classes. For more information about the current rates, testing, and vaccinations, please visit epstrong.org.

Appendix. CS 3432 Learning Outcomes

Level 1

A1. Define and explain the purpose of an instruction set architecture (ISA).
HSI2. Explain the relationship and differences between a high-level programming language, assembly language, and machine language.
HSI3. Describe the fetch-execute cycle in terms of the hardware-software interface between machine instructions and processor components.
NR1. Explain the relationship between a high-level language basic data type (e.g., signed or unsigned integer, floating point number) and its representation as a bit pattern inside the computer.
A2. Describe the basic components of a processor (e.g., register file, special-purpose registers, control unit, memory) and how they interact with one another.
HSI4. Explain how procedures are supported by processor hardware.
HSI5. Explain exception/interrupt handling in terms of the hardware-software interface.
HSI6. Explain various ways an operand can be addressed in an assembly language instruction.
HSI7. Describe the process of compiling/assembling, linking, loading, and executing a program.

Level 2

NR2. Convert between different integer data representations (e.g., decimal, binary, hexadecimal, octal).
NR3. Interpret the bit representation of a floating-point number.
NR4. Perform addition and subtraction on two's complement representation of integers.
NR5. Use bitwise operators to access and manipulate values stored in a subset of bits within a byte or word.
NR6. Determine range and precision (if applicable) of numbers that can be stored for a given data type and determine whether an integer operation will result in overflow.
HSI8. Convert between machine and assembly language representations of instructions – i.e., encode and decode instructions.
HSI9. Trace the datapath through the processor for a given class of instructions (e.g., arithmetic-logical, memory access, conditional branch)
HSI10. Trace the execution of an assembly language program with procedure calls in terms of allocation and deallocation of stack frames.
L1. Translate expressions and assignment statements from C to assembly language.
L2. Translate Boolean logic and control flow constructs (decisions, loops) from C to assembly language.
L3. Translate operations on arrays, structs, and pointers from C to assembly language.

Level 3

HSI11. Implement/debug simple imperative programs in assembly/machine language.
HSI12. Write or call a procedure with local variables, parameters, and return value in assembly language.
HSI13. Implement a simple interrupt handler.
T1. Compose, compile/assemble, execute, and debug simple programs in a command-line environment, using appropriate modularization and multiple files.