

University of Montana

ScholarWorks at University of Montana

University of Montana Course Syllabi, 2021-2025

Spring 2-1-2022

CSCI 361.B01: Computer Architecture

Jesse V. Johnson

University of Montana, Missoula, jesse.johnson@umontana.edu

Follow this and additional works at: <https://scholarworks.umt.edu/syllabi2021-2025>

Let us know how access to this document benefits you.

Recommended Citation

Johnson, Jesse V., "CSCI 361.B01: Computer Architecture" (2022). *University of Montana Course Syllabi, 2021-2025*. 185.

<https://scholarworks.umt.edu/syllabi2021-2025/185>

This Syllabus is brought to you for free and open access by ScholarWorks at University of Montana. It has been accepted for inclusion in University of Montana Course Syllabi, 2021-2025 by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.

Computer Architectures

CSCI 361

Spring 2020 Syllabus

We can only see a short distance ahead, but we can see plenty there that needs to be done.

–Alan Turning

Instructor Details

Name: Professor Jesse Johnson
Office: 417 Social Science Building
Telephone: (406) 243-2356
Email: jesse.johnson@umontana.edu
Web: [Faculty Home Page](#)
Office Hours: TWTh 10:00–12:00 , Social Science Building 417
Or, by appointment.

Prerequisites

Students taking this course are expected to have:

- Programming experience demonstrated by passing CSCI 136 (Introduction to CS II) or a similar course.
- Organizational skills and familiarity with computers sufficient to install new software, create a file system for the course, and execute Java programs from the command line.
- Knowledge of the Python language adequate to make alterations to an existing Python program and write new functions according to an API.
- The ability to attend class.

Course Objectives

The course objective is to integrate key notions from algorithms, computer architecture, operating systems, compilers, and software engineering in one unified framework. This will be done constructively, by building a general-purpose computer system from the ground up. In the process, we will explore many ideas and techniques used in the design of modern hardware and software systems, and discuss major trade-offs and future trends. Throughout this journey, you will gain many cross-section views of the computing field, from the bare bone details of switching circuits to the high level abstraction of object-based software design.

Student Outcomes

Upon successful completion of this course, student will be better able to:

- develop computing systems from their most basic components by applying increasing levels of abstraction.
- apply a knowledge of the lower levels of abstraction in a computing system to solve problems that arise at higher levels.
- apply knowledge of computing and mathematics appropriate to the programs student outcomes and to the discipline.
- analyze a problem, and identify and define the computing requirements appropriate to its solution.
- design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
- function effectively on teams to accomplish a common goal.
- communicate effectively with a range of audiences.

Textbook

This semester I'll be using the following text. You will need to purchase a copy, about \$25 for a new paperback.

The Elements of Computing Systems

Nisan and Schocken

MIT Press

2005

Online Resources

Please bookmark the following online resources immediately:

- with the exception of the textbook, all course material will be made available online, through the [University of Montana's Moodle system](#),
- the textbook has a [web site](#), and
- there is a [Coursera Course](#) for this text, and videos of lectures are available if you register (helpful if you have to miss a class).

Software

This course uses simulators to test the design of your hardware. They are written in Java and run on Windows, OSX, or Linux. To run them, you'll need the [Java runtime environment](#) installed on your computer. The software should be downloaded and configured according to the instructions [here](#).

Course Format

This is a hands-on course, structured around building a series of twelve hardware and software projects. Each project is accompanied by a design document, an API, an executable solution, a test script (illustrating what the module is supposed to do), and a detailed implementation plan (proposing how to build it). The projects are spread out evenly, so there will be no special pressure towards the semesters end. That said, they are also quite 'vertical' in that many depend on successful completion of the previous project.

Our time together in the classroom will be spent as follows.

Board work Each contact period begins by having students selected in the previous class go to the board and demonstrate their work. I will randomly select students *with replacement*, meaning that the probability of going to the board is the same for each student, every time. After a student presents four times, they will be removed from further consideration. This will take about 20 minutes and will involve 4–5 students. Each student will be graded based on a rubric appearing on the course Moodle. While presenting, students may ask their group two questions that have one sentence answers.

Bug bounty While students or the teacher are presenting, other students can call out mistakes to claim a “bug bounty”. A student identifying a significant error or demonstrating a correct way of doing a problem to a student that is “stuck” will receive 5% of extra-credit on any in-class or group work assignment they scored less than 100% on. Students must be given at least 30 seconds before “bug bounty” is called out. Despite extra-credit opportunities, no more than 100% will be awarded as a final grade for in-class and group work problems.

Lecture After board work, I will lecture for about 20-30 minutes. Beyond the background theory, I will focus on working examples that are similar to the assigned work.

Group problem solving Either mixed through the lecture, or at the end of lecture, groups of students will have time to work on in-class quizzes based on old test questions. Answers to these in-class quizzes will be submitted at and graded the group level. There will be about 20 minutes of time in each class dedicated to this.

Wrapup After the quizzes, or as part of the quiz, I will go through correct answers.

Meeting Times/Place

Times: Tuesday, Thursday 15:30–16:50

Place: Liberal Arts Building 103B

Final Exam Time and Place

Time: 13:10-15:10, Tuesday, May 5

Place: Liberal Arts Building 201

Grading Policy

Grading scale

| | |
|----|--------|
| A | 94-100 |
| A- | 90-93 |
| B+ | 87-89 |
| B | 83-86 |
| B- | 80-82 |
| C+ | 77-79 |
| C | 73-76 |
| C- | 70-72 |
| D+ | 67-69 |
| D | 63-66 |
| D- | 60-62 |
| F | 0-59 |

Students achieving the numerical scores above are guaranteed the associated letter grade. However, if average performance is low, I may decide to assign a higher letter grade for a lower score; e.g. a B+ for a numerical score of 84.

Students taking the course pass/no pass are required to earn a grade of D (63%) or better in order to pass.

Assessments and weights

The following assessments will be used and weighted according to the values in the table to determine final grades.

| Component | Description | Weight |
|-------------------|--|---------------|
| In-class problems | Problems worked on the board, by individual students. A rubric of assessment appears on the course web site. | 20% |
| Group work | In each class there will be an online quiz, to be completed by the group. All students attending class and in the group will receive the same grade. Students with an excused absence receive the same grade. Unexcused absences receive a zero. | 20% |
| Midterm I | Test of your knowledge of material presented in class and projects. Inclusive of material presented since first day of class. | 15% |
| Midterm II | Test of your knowledge of material presented in class and projects. Inclusive of material presented after midterm I. | 15% |
| Final Exam | Test of your knowledge of all material presented in class and projects. Inclusive of all material. | 30% |

Tentative schedule:

| TUESDAY | | THURSDAY | |
|--|----|---|----|
| Jan 14th | 1 | 16th | 2 |
| Course introduction and demonstration of tools, Introduction to Hardware Description Language (HDL), logic gates | | Combinational logic and the ALU (Arithmetic-Logic Unit) | |
| 21st | 3 | 23rd | 4 |
| Combinational logic and the ALU (Arithmetic-Logic Unit) | | Sequential logic: memory hierarchy | |
| 28th | 5 | 30th | 6 |
| Sequential logic: flip-flop gates, registers, and RAM | | Machine language: instruction set, assembly and binary versions | |
| Feb 4th | 7 | 6th | 8 |
| Machine language: assembly language programs | | Computer architecture I | |
| 11th | 9 | 13th | 10 |
| Computer architecture II | | Assembler: language translation - parsing and symbol table | |
| 18th | 11 | 20th | 12 |
| Assembler: language translation - macro-assembly and construction of assembler | | Virtual machine I: modern virtual machines, stack based arithmetic, logical and memory access operations | |
| 25th | 13 | 27th | 14 |
| Virtual machine I: implementation of a VM from assembler language previously developed | | Virtual machine II: stack-based flow-of-control and subroutine call-and-return techniques, complete VM implementation | |
| Mar 3rd | 15 | 5th | 16 |
| High level language: introduce <i>Jack</i> , a simple high level language with Java like syntax | | High level language: trade-offs in language design and a simple, interactive game in <i>Jack</i> | |
| 10th | 17 | 12th | 18 |
| Day to account for delays | | Midterm Exam I | |
| 17th | | 19th | |
| <i>Spring Break</i> | | <i>Spring Break</i> | |

| TUESDAY | | THURSDAY | |
|---|----|--|----|
| 24th | 19 | 26th | 20 |
| Compiler I: context-free grammars and recursive parsing algorithms, building a tokenizer and parser for <i>Jack</i> . | | Compiler I: syntax analyzer and XML output | |
| 31st | 21 | Apr 2nd | 22 |
| Compiler II: code generations, low-level handling of arrays and objects | | Compiler II: a full-scale compiler, generating VM code from XML produced previous week | |
| 7th | 23 | 9th | 24 |
| Operating system: design of OS/hardware and OS/software with regard to time/space efficiency of design | | Operating system: classic algorithms in OS design | |
| 14th | 25 | 16th | 26 |
| Day to account for delays | | Midterm Exam II | |
| 21st | 27 | 23rd | 28 |
| ARM Architecture I | | ARM Architecture II | |
| 28th | 29 | 30th | 30 |
| ARM Architecture III | | ARM Architecture IV Wrap up/Course evaluation | |
| May 5th | 31 | 7th | 32 |
| Final Exam, 13:10-15:10 | | Finals Week | |

Attendance Policy

Attendance will not be taken. Students absent for end of class quizzes will be given a grade of 0%. Students informing the instructor of a valid reason for missing class *in advance*, via email, will be excused. Valid reasons include family emergencies and illness. I may ask for documentation of absence (doctors note, death certificate, etc.) if the number of absences exceeds two. Cell phone photos are very useful for this - a selfie in the doctors office, or next to a car that won't start...

Academic Integrity

All students must practice academic honesty. Academic misconduct is subject to an academic penalty by the course instructor and/or a disciplinary sanction by the University. All students need to be familiar with the [Student Conduct Code](#). I will follow the guidelines given there. In cases of academic dishonesty, I will seek out the maximum allowable penalty. If you have questions about which behaviors are acceptable, especially regarding use of code found on the internet or shared by your peers, please ask me.

Disabilities

Students with disabilities may request reasonable modifications by contacting me. The University of Montana assures equal access to instruction through collaboration between students with disabilities, instructors, and Disability Services for Students. Reasonable means the University permits no fundamental alterations of academic standards or retroactive modifications.