



# Ecampus Syllabus

**Course Name:** Computer Architecture and Assembly Language

**Course Number:** CS271

**Term Offered:** All Terms

**Credits:** 4

**Instructor name / email:** Stephen Redfield / Stephen.Redfield(at)oregonstate.edu

**Instructor name / email:** Ramesh Rajagopalan / rajagopr(at)oregonstate.edu

**Teaching Assistants' names and Office Hours Info:**

- TAs and Office Hours Information on Course Syllabus Page (Canvas)

---

## 1. Course Description

- 1.1. Introduction to functional organization and operation of digital computers. Coverage of assembly language; addressing, stacks, argument passing, arithmetic operations, decisions, macros, modularization, linkers and debuggers.

## 2. Prerequisites

- 2.1. Prerequisites: CS 161
- 2.2. Recommended Prerequisites: CS 225 & CS 162

## 3. Communication

- 3.1. Please post all course-related questions in the Piazza page so that the whole class may benefit from your conversation. Not all posts require a reply from the instructor/TA and often it is better for students to hash out an answer to a question.
- 3.2. To contact the TAs, open Canvas Inbox, compose a new message, select this course, then next to the 'To' box click the address book, select "Teaching Assistants", then choose the name(s) of TAs you wish to contact.
- 3.3. Please email your instructor only for matters of a personal or private (grading) nature. The instructor or a TA will reply to most course-related questions within 24-48 hours.
- 3.4. Any email sent to the instructor about this course **must** originate with an OSU supplied email account *and contain the tag [CS271]* at the beginning of the subject. Failure to comply with this will result in delayed (or possibly nonexistent) response to your email.
- 3.5. Office hours will be held online through the class Slack workspace. To get started on slack please go to <https://it.engineering.oregonstate.edu/slack> . You'll want to add the class-specific slack by connecting to the **oregonstate.enterprise.slack.com** workspace. It should then show up as one of your connected Workspaces.

## 4. Technical Assistance

- 4.1. If you experience computer difficulties, need help downloading a browser or plug-in, assistance logging into the course, or if you experience any errors or problems while in your online course, contact the OSU Help Desk for assistance. You can call (541) 737-3474 (USA), email [osuhelpdesk@oregonstate.edu](mailto:osuhelpdesk@oregonstate.edu) or visit the [OSU Computer Helpdesk](#) online.

## 5. Learning Resources

- 5.1. Required Text: Irvine, Kip R., **Assembly Language for x86 Processors** (8th ed.), Prentice-Hall, 2019. (ISBN 978-0135381694)
  - 5.1.1. **Note to prospective students:** Please check with the OSU Bookstore for up-to-date information for the term you enroll (<http://www.osubookstore.com/> or (800) 595-0357 (USA)). If you purchase course materials from other sources, be very careful to obtain the correct ISBN.

## 6. Canvas & Piazza

- 6.1. This course will be delivered via Canvas and you will interact with your classmates and with your instructor through Piazza. Within the course Canvas site you will access the learning materials, such as the syllabus, assignments, projects, and quizzes. Class discussions will be on Piazza.
- 6.2. To preview how an online course works, visit the [Ecampus Course Demo](#). For technical assistance, please visit [Ecampus Technical Help](#).
- 6.3. Canvas is optimized for the most recent versions of most popular browsers. If your browser of choice is an out-of-date version, you should update for use with Canvas, especially for quizzes and exams. If you are having browser troubles, seek out the Technical Assistance described above.
- 6.4. If your device has trouble dealing with Canvas (as some tablets do), make sure you have an alternative available for things like quizzes and exams. If you are having device troubles, seek out the Technical Assistance described above. Telling me the day after an exam has ended that you had browser issues on your smartphone/tablet is unlikely to get you what you want.
- 6.5. When I have general announcements for the class, I will make send them as email from Canvas. It is your responsibility to keep up with messages in Canvas Inbox, and in Piazza. You should check the Piazza at least a couple times per week, just in case.
- 6.6. When there are significant issues on exams (as there sometimes are), I need to receive email from the proctor service about the issue. They also have my cell number for emergencies. We'll find a way to address it.

## 7. Course Learning Outcomes

- 7.1. Identify the major components of computer architecture, and explain their purposes and interactions.
- 7.2. Simulate the internal representation of data, and show how data is stored and accessed in memory.
- 7.3. Explain the relationships between hardware architecture, its instruction set, and simulate microprograms.

- 7.4. Create and simplify circuits that produce specified output for given inputs (e.g., adders, multiplexers, etc.).
- 7.5. Explain the Instruction Execution Cycle.
- 7.6. Explain the differences and relationships among high-level, assembly, and machine languages.
- 7.7. Write well-modularized computer programs in an assembly language, implementing decision, repetition, and procedure structures.
- 7.8. Use a debugger, and explain register contents.
- 7.9. Simulate the system stack as it is used for procedure calls and parameter passing.
- 7.10. Explain how editors, assemblers, linkers, and operating systems enable computer programming.
- 7.11. Explain various mechanisms for implementing parallelism in hardware/software.

**8. Evaluation of Student Performance**

- 8.1. Weekly summaries 10% 6 hour time limit, take twice, open book (1% each)
- 8.2. Quizzes 10% 1 hour time limit, take once, open book (2.5% each)
- 8.3. Midterm 15% PROCTORED, 1 hour 50 min time limit, take once, closed book
- 8.4. Final 25% PROCTORED, 1 hour 50 min time limit, take once, closed book
- 8.5. Homework 40% aka Programming projects
- 8.6. The grading scale is as follows:

93+	A
90 – 92.99	A-
87 – 89.99	B+
83 – 86.99	B
80 – 82.99	B-
77 – 79.99	C+
73 – 76.99	C
70 – 72.99	C-
67 – 69.99	D+
63 – 66.99	D
60 – 62.99	D-
0 – 59.99	F

- 8.7. Other important grading details:
  - 8.7.1. If you wind up with a grade average of 89.99%, you will get a B+ in the class.
  - 8.7.2. You may check the individual assignments' course-grade weighting from within Canvas' Grades tab.

- 8.7.3. The intention is to give you a lot of practice before the exams. By getting the practice, you perform better and more quickly on the exams (and the programming projects).
- 8.7.4. The class grade is divided such that you can be either a really good programmer or a really good test taker and do okay in the class. If you are a really good with the programming, but do poorly on tests, you can still succeed in the class. If you are great with tests, but crummy at coding, again, you can succeed in the class. If you are good at both, you'll excel.
- 8.7.5. I know taking this many assessments is not your favorite thing to do. However, I have found that having frequent assessments helps students keep pace with the work and provides them with quick feedback on how well the material is understood or remembered. Grades from previous terms validate this.

## 9. Weekly Summaries

- 9.1. Weekly summaries are open book, open note, open Internet, and open lecture. You can use just about anything, including your fellow students while taking a weekly summary. The weekly summaries will primarily (but not exclusively) cover assigned reading material and lectures from the week.
- 9.2. Weekly summaries are not proctored.
- 9.3. Weekly summaries have a time limit of 6 hours (so I don't end up with partial tests).
- 9.4. You will be able to take each weekly summary at most twice. The recorded score will be the higher of the two scores you receive.
- 9.5. Weekly summaries cannot be taken after the due date.
- 9.6. The weekly summaries are worth 10% of your final grade. They are to help you pull together the material from the week. This will help you with the quizzes, the exams, and (importantly) the programming projects. The syllabus quiz counts as a weekly summary.

## 10. Quizzes

- 10.1. Quizzes are open book, open note, open Internet, and open lecture. You can use just about anything **except** your fellow students while taking a quiz. Quizzes are not proctored. You will be able to **take each quiz once**. Quizzes cannot be taken after the due date.
- 10.2. Quizzes become available on Thursday of the week they are due.
- 10.3. The 4 quizzes sum to be worth 10% of your final grade. They are to help you pull together the material from the previous couple of weeks. The quizzes are intended to give you some live practice for subjects that will be on the midterm and final.
- 10.4. **Quizzes in this class are timed**. You won't be able to exceed the time limit on the quizzes. It is not the intent of the quizzes to be time pressured, but pace yourself. The rate at which you are able to complete the quizzes will give you a good measure on how quickly you can move through the midterm and final.

## 11. Exams (Midterm and Final)

- 11.1. The Midterm and Final are **not open book and not open note**.
- 11.2. You are allowed to take a calculator (including Windows Calculator, but not phone calculator), blank scratch paper (dry erase whiteboard for ProctorU), and a single sheet (8.5x11) of handwritten or typed notes (double sided) into the midterm and the final.
- 11.3. You may use any calculator other than phone or online calculators.
- 11.4. You are **not** allowed to take anything else into an exam.
- 11.5. Both Exams are proctored.
- 11.6. Exams have a due date.
- 11.7. The Midterm has 110 minutes and the Final has 110 minutes (the maximum allowed), and it is my experience that most students use the entire time. The better you prepare for the midterm and final, the easier it will be for you to complete the midterm and final within the allotted time.
- 11.8. Material presented during the first half of the class and covered on midterm will also be required for the final exam (the Final is comprehensive).
- 11.9. Exam Proctoring
  - 11.9.1. This course requires that you take exams (the midterm and the final) under the supervision of an approved proctor. Proctoring guidelines and registration for proctored exams are available online through the Ecampus [testing and proctoring website](#). It is important to submit your proctoring request as early as possible to avoid delays.
  - 11.9.2. If you happen to be an on-campus student taking the on-line class, you still need to arrange for a proctor. You can take advantage of on-campus proctoring services, but the best times tend to fill up quickly (especially during finals week). If you are going to use on-campus proctoring resources, schedule your time at least 2 weeks prior to taking the exam (especially the final). If you cannot find an on-campus time, you'll need to arrange to use one of the on-line proctoring services.
    - 11.9.2.1. Inability of finding an on-campus proctoring time slot is not a reason to change/extend the end date on an exam.
- 11.10. Makeup Exams
  - 11.10.1. Makeup exams will be given only for missed exams excused in **well in advance** by the instructor. Excused absences will not be given for airline reservations, routine illness (colds, flu, stomach aches), or other common ailments. You'll have 4 days in which to take the Midterm and the Final. Excused absences will generally not be given after the absence has occurred, except under very unusual circumstances.

## 12. Incompletes

- 12.1. Incomplete (I) grades will be granted only in emergency cases (usually only for a death in the family, major illness or injury, or birth of your child), and if the student has turned in 80% of the points possible (in other words, usually everything but the final exam). If you are having any difficulty that might prevent you completing the coursework, please don't wait until the end of the term; let me know right away.

## 13. Homework (aka Programming Projects)

- 13.1. The programming projects (homework) are a significant portion of this class (40% of the final grade). The programming projects are the most common place for students to struggle in this class. Several things about this class' programming projects may be new to you.

- 13.1.1. You may not have used Visual Studio before.
- 13.1.2. Intel x86 Assembler code will be new.
  - 13.1.2.1. Programming at the assembler level is very different from using higher level languages.
- 13.1.3. Stepping through the assembler code in the debugger will be new.
- 13.2. All programming projects must be submitted by 11:59pm Pacific on the due date to Canvas.
- 13.3. Late Projects have exactly two (2) days from the due date, no more, to be submitted. Since programming assignments are normally due on a Sunday, 2 days late makes that Tuesday. **Late work is penalized 15% per day.** Any programming project submitted more than 2 days after the due date will automatically receive a grade of zero (0). Don't make the mistake of submitting your assignment late just trying and get the last few points by making it perfect. Perfection is the enemy of done. You want to be done.
- 13.4. You have the right to make use of two grace days for submission of programming projects **other than the last program**, used in increments of one day. The grace days allow you to have an un-penalized late assignment by up to two days or 2 assignments up to 1 day each. Grace days may not be used on the final program  
**SPRING 2020 NOTE: This term you have 5 grace days instead of 2, but a maximum of 2 may be used per assignment, and they may be used on the final program.**
  - 13.4.1. The use of grace days does not extend the last day on which you can submit an assignment to be graded, it is still a maximum of 2 days past the assignment due date.
  - 13.4.2. Grace days don't change the assignment due date, they only change deductions for late. If you use 2 grace days and submit the assignment 3 days late, you've used your grace days and received a 0 on the assignment.
  - 13.4.3. Grace days *must* be invoked prior to the submission deadline, and must be invoked via the following process. Failure to follow this procedure renders the grace day invalid, though used.
    - 13.4.3.1. Before the due date, in Canvas, go to "Grades".
    - 13.4.3.2. Select the assignment (e.g. Program 1).
    - 13.4.3.3. On the right side of the screen, in the "Add a Comment" box enter "X Grace Days" where X is the number of grace days you will use for that assignment.
  - 13.4.4. I encourage you to not use up your grace days early in the term. Programming assignments get harder as the as the term progresses. You'd hate to waste grace days on early and easy assignments when the assignments get harder. Start your programming assignments as soon as possible. Do not wait until the last weekend to begin them.
  - 13.4.5. Don't be lulled into over-confidence from easy early assignments only to be surprised by later assignments. Watching the clock tick past midnight for an assignment that feels far from working is not enjoyable. It causes stress, and stress is bad, mmkay?
- 13.5. All source files (.asm files), must include a comment block at the top that contains the following information. Neglecting this information is an automatic 20% deduction from your grade. It is easy to do, so please just do it.
  - 13.5.1. Your name.
  - 13.5.2. Your OSU email address.

- 13.5.3. The class number, and section (CS271-400).
- 13.5.4. The assignment number.
- 13.5.5. Assignment due date.
- 13.6. You can overcome a poor grade on one assignment and still do well in the class. Do not allow struggling on one programming assignment to cause you to be late on all programming assignments.
- 13.7. Don't miss submitting a programming assignment. You are much better off to submit a partially functional assignment than to not submit anything for an assignment.
- 13.8. Your programming assignments must run in Visual Studio to be graded. If your assignment does not run in Visual Studio, then you will get a zero for a grade. Running under some other assembler or emulator in addition to VS is fine, but it must still run under Visual Studio.
- 13.9. You must submit all your assignments through Canvas.
  - 13.9.1. Submit your work for each assignment as a **single asm** file through Canvas. You should not need to submit any additional files for a programming assignment. If you use external library other than the Irvine32 library your code will fail to assemble and link in the standard Visual Studio environment that we use to grade the assignment. That means you will be disappointed with your grade. We expect you to make use of the Irvine32 library and no other libraries in your code. If you need to comment on your code, place your comments into the asm file.
  - 13.9.2. You can submit your assignments more than once through Canvas. Each will be time stamped. We will grade only the last one submitted.
- 13.10. **We reserve the right to ask you to explain a complicated piece of code.** If you cannot explain your own code to us, you may be investigated for violation of student conduct (see sections 16, 17, and 18).
- 13.11. If you are struggling on a programming assignment, the first thing you should do is make sure you have read the assigned readings for the week and prior weeks and watched the lectures (review the notes you've taken).
  - 13.11.1. The book has many excellent descriptions and examples of the topics covered in the programming assignments.
  - 13.11.2. The lectures are very often additional examples of those same topics.
  - 13.11.3. Sending your entire source code to the instructor or TA with a note saying "Something wrong can you fix it?" is unlikely going to get you the response you want. The instructor and TA are not debuggers.
  - 13.11.4. Run the program in the Visual Studio debugger yourself. The way to get better at debugging code is to use the debugger.
  - 13.11.5. Make sure you read the entire assignment. There is useful information in all that text.
- 13.12. We will be using Visual Studio as the development environment for this class, using MASM (Microsoft Macro Assembler). You can use most versions of Visual Studio after 2015, but I recommend 2019. If you use some other assembler (e.g. NASM), your code likely will not assemble and you'll lose most/all of your points. If you don't already have Visual Studio, you can get it free for student use. Check the Tools tab of the Canvas Syllabus Page.
- 13.13. Feedback for your programming assignments will be given through Canvas Rubrics.



- 13.14. This is important feedback. You don't want to repeat this sort of error on following assignments. The spreadsheet will also identify by whom your assignment was graded, making it much easier to contact her/him if you have questions about your grade.
- 13.15. If you are unable to locate the feedback on your assignments, ask a TA to guide you to it.
- 13.16. If you wish to petition a grade, you must do so within one week of receipt of the grade.
- 13.17. It is your responsibility to keep up with your assignment/exam/quiz/summary grades and initiate contact if you have a question.

## 14. Keys to Success

- 14.1. This class requires a keen attention to detail. Particularly when you are working with an unfamiliar x86 instruction (and they all start as unfamiliar). Some of the keys to success are:
  - 14.1.1. Watch the lectures and take notes (just like you would in an on-campus class)
  - 14.1.2. Complete the self-check exercises (do this after watching the lecture, not during!)
  - 14.1.3. Read the assigned material (and take notes)
  - 14.1.4. Take the weekly summaries (more than once if it helps).
  - 14.1.5. Start the programming assignments early and complete them on time.
  - 14.1.6. Check Appendix B (in the textbook), frequently. It's a great resource for understanding Assembly Language instructions.
  - 14.1.7. Don't get discouraged if your code initially does not execute correctly; many problems are simple to fix and it just takes time to isolate and identify the problem.
  - 14.1.8. Although mentioned in the course lectures, this cannot be emphasized enough, **learn to use the Visual Studio debugger**. It's much faster to troubleshoot a problem while using the debugger, so take the time to understand how it works.
  - 14.1.9. Assemble your code often. Writing a few lines of code and then double-check it to make sure that it assembles correctly. This is especially important when you are first learning to program in assembly. By assembling your code frequently you can locate mistakes more quickly and have a better idea of where a problem originates.
- 14.2. When struggling with a homework assignment:
  - 14.2.1. Re-Watch the lectures
  - 14.2.2. Create tests to isolate the problem.
  - 14.2.3. Use the debugger (while running the test cases)
  - 14.2.4. Look in the book
  - 14.2.5. Look online
  - 14.2.6. Look for/create a reply in Piazza.
  - 14.2.7. Hop onto Slack!
- 14.3. Another valuable asset to have in this class (and the entire program) is a study group. I recommend using the Introduction thread to get connected with some people near you, or at least who have similar schedules.
- 14.4. One of the skills you'll develop in this class is how to look for things in the resources listed above. You'll spend some quality time with your favorite Internet search engine. You'll learn how to wade through the chaff of Stack Overflow to find the relevant example from the hundreds of search hits. You'll probably be able to remember the page number of certain examples from the textbook. Some of this won't be fun, but you'll learn that you can learn it.
- 14.5. Make sure the code you submit actually assembles and links. If your code does not assemble (using MASM in Visual Studio), you will receive a zero (0) for that portion of the programming project. Our graders won't guess what portions of your code may be correct if it does not assemble. If you are unable to get your program to assemble correctly, comment out the



portion of the code that causes the assembly process to fail. You are better off getting partial credit on a programming project than getting a zero for code that does not assemble.

## 15. Statement Regarding Students with Disabilities

- 15.1. Accommodations are collaborative efforts between students, faculty and [Disability Access Services \(DAS\)](#) with accommodations approved through DAS are responsible for contacting the faculty member in charge of the course prior to or during the first week of the term to discuss accommodations. Students who believe they are eligible for accommodations but who have not yet obtained approval through DAS should contact DAS immediately at (541) 7374098 (USA).

## 16. Expectations for Student Conduct

- 16.1. Student conduct is governed by the university's policies, as explained in the [Office of Student Conduct: Information and Regulations](#).

## 17. Academic Honesty

- 17.1. Students are expected to do their own work. The only sources you're allowed to use code from are the textbook and lecture slides, and you must make a comment with the chapter/page or lecture/slide if you do this. Direct use of any other resources is prohibited
- 17.2. Programming assignments present unique challenges for graders. It is often difficult for a grader to distinguish between legitimate help and plagiarism. We use plagiarism-detection software to check your code against the code from other students. It is quite sophisticated and can see through variable name changes and formatting differences
- 17.3. Honesty is essential for learning to take place. It will form the foundation of your professional integrity in your career.
- 17.4. You **may** ask conceptual questions related to optimizing your code on Slack or Piazza.
- 17.5. You **may** post Weekly Summary questions and solutions at any time to Slack or Piazza.
- 17.6. You **may** post Quiz questions and solutions **after the due date** to Slack or Piazza,
- 17.7. You are **encouraged to** discuss course content with other students, TAs, the instructor, or anyone else who will listen, including general discussion of homework assignments and how to fix specific issues.
- 17.8. You **may** share pseudo code and ideas about how to solve or approach problems. If you are getting odd assembler messages, you can share the snippet of code that is producing the message; you don't need to share the entire file.
- 17.9. You **may not** copy anyone's (or allow someone to copy your) solutions. It is possible to discuss problems without plagiarizing. One of the best methods of debugging is to explain your solution to someone else.
- 17.10. You **may not** post any complete functions/procedures/logic blocks to Slack or Piazza.
- 17.11. You **may not** post any exam questions or solutions in any form.
- 17.12. You **must** make any git repo you post this code on private, with the exception of the Portfolio assignment.
- 17.13. If you are found in violation of any of the above policies, whether you are the giver or receiver of help, you will be subject to the University Academic Misconduct process. The first offense usually results in a warning and an assignment penalty (0-grade); the second offense can result in a disciplinary hearing, and possibly removal from the Program.
- 17.14. The bottom line is: Each student is expected to understand all aspects of the programs s/he submits for credit.

- 17.15. The following are examples of plagiarism, drawn from actual submitted and penalized cases.
- 17.15.1. Student Googled the problem and ended up finding a code written online. Student used the code there to improve your own work.
  - 17.15.2. Student worked with a classmate, each tackling one half of the program. Both students modified the identifiers, labels, comments, etc...
  - 17.15.3. Student found a solution to the problem online and changed a few variable names and methods and handed it as their own.

## 18. Academic Integrity

- 18.1. The Code of Student Conduct prohibits Academic Misconduct and defines it as:
- 18.1.1. Any action that misrepresents a student or group's work, knowledge, or achievement, provides a potential or actual inequitable advantage, or compromises the integrity of the educational process.
- 18.2. To support understanding of what can be included in this definition, the Code further classifies and describes examples of Academic Misconduct, including cheating, plagiarism, assisting and others. See the [Code of Student Conduct](#) for details.
- 18.3. You are expected to do your own work and demonstrate academic integrity in every aspect of this course. Familiarize yourself with the standards set forth in the **OSU Code of Student Conduct Section 4.2**. You must only access sources and resources authorized by the instructor. You may not show your work to any other current or future students without the instructor's authorization. Violations of these expectations or the Code of Student Conduct will be reported to the Office of Student Conduct and Community Standards. If there is any question about whether an act constitutes academic misconduct, it is your responsibility to seek clarification and approval from the instructor prior to acting.

## 19. Tutoring

- 19.1. [NetTutor](#) is a leading provider of online tutoring and learner support services fully staffed by experienced, trained and monitored tutors. Students connect to live tutors from any computer that has Internet access. NetTutor provides a virtual whiteboard that allows tutors and students to work on problems in a real time environment. They also have an online writing lab where tutors critique and return essays within 24 to 48 hours. Access NetTutor from within your Canvas class by clicking on the button in your course menu.

## 20. OSU Student Evaluation of Teaching

- 20.1. Course evaluation results are extremely important and are used to help me improve this course and the learning experience of future students. Results from the multiple choice questions are tabulated anonymously and go directly to instructors and department heads. Student comments on the open-ended questions are compiled and confidentially forwarded to each instructor, per OSU procedures. The online Student Evaluation of Teaching form will be available toward the end of each term, and the Office of Academic Programs, Assessment, and Accreditation will send you instructions via your ONID email address. You will log in to "Student Online Services" to respond to the online questionnaire. The results on the form are anonymous and are not tabulated until after grades are posted.