CS271: Computer Architecture and Assembly Language
Syllabus – Fall 2016

Instructor name: Stephen Redfield
Instructor email: Stephen.Redfield(at)oregonstate.edu
Instructor phone: (541) 737-6022

Instructor Office Hours:	**Tuesday**	**11:00am – 12:00pm**	**KEC 2105**
	Thursday	**1:30 – 2:30pm**	**KEC 2105**

Teaching Assistants' names and office hours:

- Fariba Khan — Office hours Monday/Friday 12:00 – 1:00pm (KEC 1005)
- Xuanyi Dong — Office hours Wednesday 5:00 – 6:00pm (KEC 4126) – Knock on Door

_____

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.	Enforced Prerequisites:	CS 161
2.2.	Other Prerequisites:	CS 225 or MTH 231
2.3.	Courses that require this class as a prerequisite:	CS 311

3. Course Design

3.1. This is a partially "flipped" course. This means that some of your normal in-class activity (lecture) will be accessed outside of class (via pre-recorded video lectures). Some of your normal outside-of-class activities (worksheets, study) will be completed in-class.

3.2. You will NEED a computer with a battery that will last you at least 1.5 hours.

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 and email within 24-48 hours.

3.4. Any email sent to the instructor/TA about this course **must** originate with an OSU supplied email account and contain the tag **[CS271]** at the beginning of the subject, or be from Canvas Inbox. Failure to comply with this will result in delayed response to your email.

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 or visit the OSU Computer Helpdesk online.

5. Learning Resources

5.1. Required Text: Irvine, Kip R., **Assembly Language for x86 Processors** (7th ed.), Prentice-Hall, 2014. (ISBN 0133769402)

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

6.1. This course's lectures and quizzes will be delivered via Canvas. Within the course Canvas site you will access the learning materials, such as the syllabus, class discussions, assignments, projects (and submissions), and quizzes.

6.2. If your device has trouble dealing with Canvas (as some tablets do), make sure you have an alternative available for things like quizzes. If you are having device troubles, seek out the Technical Assistance described above. Telling me the day after a quiz has ended that you had browser issues on your smartphone/tablet is unlikely to get you what you want.

6.3. When I have general announcements for the class, I will make them in class, or via mass Canvas Inbox message. It is your responsibility to keep up with announcements made through either of these media.

7. Measurable Student 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 Quizzes	16%	Online, weekly through Canvas, 50m time limit, open book
8.2.	Class Exercises	9%	On paper, Groups of 2, Tues/Thurs in class
8.3.	Midterm	15%	In class Oct 27th, 1h 20m time limit, closed book
8.4.	Final	20%	Friday Dec 9th 9:30am, 1h 50m time limit, closed book
8.5.	Projects	40%	Programming assignments

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 criteria:

8.7.1. **All programming projects must be submitted in order to pass the course.** Students missing programming projects at the end of the term will receive an F grade.

8.7.2. Students who do not take the final exam will automatically receive an F grade.

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. Quizzes

9.1. 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.

9.2. Quizzes become available after the material for that week has been covered in class.

9.3. **Quizzes in this class are timed**. You won't be able to exceed the time limit on the quizzes. As stated above, 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.

10. Exams (Midterm and Final)

10.1. The Midterm and Final are **closed book and limited notes**.

10.2. You are allowed to bring a calculator, blank scratch paper, and a single sheet (8.5x11) of handwritten or typed notes (double sided) into the midterm and the final.

10.3. You may use only handheld non-phone calculators.

10.4. You are **not** allowed to take anything else into an exam.

10.5. The Midterm is 80 minutes and the Final is 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.

10.6. The midterm exam is worth 15% of your class grade and the final exam is worth 25% of your class grade.

10.7. 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).

10.8. Makeup Exams

10.8.1. 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 3 days on 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.

10.9. Exam Time Limits

10.9.1. Exams in this class are timed. If you have an allowance for extra time, please remind me (via e-mail) to add in your time accommodation. It won't let me do this until after the exam is published, "better safe than sorry" as they say.

11. Incompletes

11.1. 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.

11.2. Completion of an incomplete (I) grade will require **additional** work from you. You won't simply have an opportunity to do the work late; you'll do more, possibly a lot more.

12. Homework (aka Programming Projects)

12.1. 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.

12.1.1. You may not have used Visual Studio before.

12.1.2. Intel x86 Assembler code will be new.

12.1.2.1. Programming at the assembler level is very different from using higher level languages.

12.1.3. Stepping through the assembler code in the debugger will be new.

12.2. All programming projects must be submitted by 11:59pm on the due date to Canvas.

12.3. 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.

12.4. You have the right to make use of two grace days for submission of programming projects, 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.

12.4.1. 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.

12.4.2. 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.

12.4.3. 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.

12.4.3.1. Before the due date, in Canvas, go to "Grades".
12.4.3.2. Select the assignment (e.g. Program 1).
12.4.3.3. 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.

12.4.4. 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.

12.4.5. 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?

12.5. 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.

12.5.1. Your name.

12.5.2. Your OSU email address.

12.5.3. The class number, and section (CS271-400).

12.5.4. The assignment number.

12.5.5. Assignment due date.

12.6. The programming projects are designed to **not** build on each other. This is so that you can limit losses on one assignment and move on to the next one. 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.

12.7. 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. As stated above, you must submit **all** programming projects for the class in order to pass the class.

12.8. 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.

12.9. You must submit all your assignments through Canvas.

12.9.1. 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.

12.9.2. You can submit your assignments more than once through Canvas. Each will be time stamped. We will grade only the last one submitted.

12.10. All programs must be done individually unless specifically allowed to work in groups. The homework programming projects are **not** group projects. You must do your own programming projects. You may share snippets of code for assistance but do not share entire source files. You may share pseudo code and ideas about how to solve or approach problems, but write your own code. 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.

12.10.1. There are some very good tools for detecting when students are copying code from each other. I have used those tools. I have found students who copied code. No one enjoyed it, and I'm sick of failing students for it. Programs **must** be done individually.

12.10.2. Copying code is a violation of the student conduct policy. Even if the code comes from a previous term, it is still copying.

12.10.3. **We reserve the right to ask you to explain a complicated piece of code**. If you cannot explain your own code to us, you will receive a grade of 0 on the assignment and I will submit you for a student conduct violation (see sections 16, 17, and 18).

12.11. 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).

12.11.1. The book has many excellent descriptions and examples of the topics covered in the programming assignments.

12.11.2. The lectures are very often additional examples of those same topics.

12.11.3. 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.

12.11.4. Run the program in the Visual Studio debugger yourself. The way to get better at debugging code is to use the debugger.

12.11.5. Make sure you read the entire assignment. There can be some really useful information in all that text.

12.12. We will be using Visual Studio as the development environment for this class, using MASM (Microsoft Macro Assembler). I recommend Visual Studio 2015. 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) through the Dreamspark link in TEACH (See the Getting Started Videos).

12.13. Feedback for your programming assignments will be given through Canvas. There should be a link to an Excel spreadsheet with details on how your assignment was graded. You'll want/need to review the feedback. It is one of the important ways to learn in the class. You might see rows in the spreadsheet that look something as shown below:

Documentation (9 pts)		**Possible**	**Earned**	
	Identification block	4	1	Each procedure should have its own header block as discussed in Lecture 15.

Or this

Requirements (70 pts)		**Possible**	**Earned**	
	Modular design (uses procedures main, displayList)	8	0	Program does not use multiple procedures

12.14. This is important feedback. You don't want to repeat this sort of error on following assignments. The spreadsheet will also identify by whom you assignment was graded, making it much easier to contact her/him if you have questions about your grade.

12.15. If you are unable to locate the feedback on your assignments, ask a TA to guide you to it.

12.16. It is your responsibility to keep up with your assignment/exam/quiz/summary grades and initiate contact if you have a question.

13. Keys to Success

13.1. 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:

13.1.1. Watch the lectures and take notes (just like you would in an on-campus lecture)
13.1.2. Complete the self-check exercises (do this after watching the lecture, not during!)
13.1.3. Read the assigned material (and take notes)
13.1.4. Review quizzes after they're graded to see where you're having issues.
13.1.5. Start the programming assignments early and complete them on time.
13.1.6. Read Appendix B (in the textbook), frequently.
13.1.7. 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.
13.1.8. 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.
13.1.9. 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.

13.2. When struggling with a homework assignment:

13.2.1. Re-Watch the lectures

13.2.2. Create tests to isolate the problem.
13.2.3. Use the debugger (while running the test cases)
13.2.4. Look in the book
13.2.5. Look online
13.2.6. Look for/create a reply in Canvas Q&A Threads.
13.2.7. Email the instructor and/or TA

13.3. The best key to success in this class is: keep up. Don't let yourself fall behind.

13.4. Another valuable asset to have in this class (and the entire program) is a study group. I recommend using the in-class exercise groups outside of class as an additional study resource.

13.5. 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.

13.6. 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. I'm not going to try and 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.

14. Statement Regarding Students with Disabilities

14.1. 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).

15. Expectations for Student Conduct

15.1. Student conduct is governed by the university's policies, as explained in the Office of Student Conduct: Information and Regulations.

16. Academic Honesty

16.1. Students are expected to do their own work. Individuals are expected to be the sole source of their code. We use software designed to find similarities between programs. Each individual's program is compared to every other individual's program to find similarities. Please, do you own work.

16.2. Programming assignments present unique challenges for graders. It is often difficult for a grader to distinguish between legitimate help and plagiarism. Therefore, it is sometimes possible to get a good score without really understanding what you have handed in. Understanding is the real point of the class.

16.3. Honesty is absolutely essential in order for learning to take place. It will form the foundation of your professional integrity in your career.

16.4. If you are having trouble with an assignment, you are encouraged to discuss it with other students, TAs, the instructor, or anyone else who will listen, but don't just have someone else tell you how to solve the problem! If other students ask you for help, don't just let them copy your work! It is possible to discuss problems without plagiarizing. One of the best methods of debugging is to explain your solution to someone else.

16.5. If you get help from, give help to, or work together with someone, you must (in the program header block) list that person as a collaborator and describe the help. Programs that are very similar will be subjected to review unless both programs indicate that they were produced collaboratively. We use plagiarism-detection software to check your code against the code from other students. It is quite sophisticated and can easily see through variable name changes and formatting differences.

16.6. If you get help from printed or online sources, ***you must cite your references***. Failure to do this results in me filing an Academic Dishonesty form, and you getting a zero on the assignment.

16.7. If you are found in violation of any of the above policies, whether you are the giver or receiver of help, you will receive a zero on the assignment or fail the course (instructor's discretion). The academic dishonesty charge will be documented and sent to your school's dean and the Office of Student Conduct. The first offense results in a warning; the second offense results in an academic dishonesty charge on your transcript, a disciplinary hearing, and possible expulsion.

16.8. The bottom line is: Each student is expected to understand all aspects of the programs s/he submits for credit.

17. Academic Integrity

17.1. Students are expected to comply with all regulations pertaining to academic honesty. For further information, visit Avoiding Academic Dishonesty, or contact the office of Student Conduct and Mediation at 541-737-3656.

17.1.1. OAR 576-015-0020 (2) Academic or Scholarly Dishonesty:
Academic or Scholarly Dishonesty is defined as an act of deception in which a Student seeks to claim credit for the work or effort of another person, or uses unauthorized materials or fabricated information in any academic work or research, either through the Student's own efforts or the efforts of another.

17.2. Academic Dishonesty cases are handled initially by the academic units, following the process outlined in the University's Academic Dishonesty Report Form, and will also be referred to SCCS for action under these rules.

18. Tutoring

18.1. 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.

19. OSU Student Evaluation of Teaching

19.1. 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.