

<b>Course</b>	<p>CIS 560 Computer Systems Programming I Monday &amp; Wednesday (5:30 to 7:15) Prereq: 314 or 321 and 360 or ECE 265, and a writing course. Software engineering as applied to various classical computer systems programs; assemblers, macro processors, linker and loaders; major group project involving the design and implementation of systems software; communication skills emphasized. Software Engineering and Design, Project Management, Compiler Designs, Linker/Loader Designs, Testing (test plan, testing methodology, etc), and Communication Skills (oral and written).</p>
<b>Objectives</b>	<p>Teach software engineering design of professional products based on real world applications. Develop oral and written communication skills. Ability to effectively work as a team member and deal with social and ethical issues of a team oriented project.</p> <ul style="list-style-type: none"><li>• To learn Software Engineering techniques Design, Coding structure and standards, Code management and change management, Testing and validation</li><li>• To experience <b>and succeed</b> on a team project</li><li>• To improve writing, organizational, and presentation skills</li><li>• To master using and implementing components of the assemble-link-load-relocate-execute process.</li><li>• To master using bit manipulation of integers and ASCII characters to be able to emulate a simple computer that handles both integer and character I/O.</li><li>• To be familiar with group project organization techniques including conducting group meetings, recording minutes, and tracking project progress.</li><li>• To write a functioning relocating linking loader.</li><li>• To be familiar with using different addressing modes.</li><li>• To be familiar with subroutine linkage at the assembly level.</li><li>• To be familiar with using compilers, debuggers, word processors, editors, diagram drawing programs, and profilers to design, build, and document a large software project.</li><li>• To be familiar with using macros, including recursive and nested macros.</li><li>• To be familiar with defining the purpose (persuade, inform, etc.) of a written document and select the appropriate rhetorical devices; and to write several pieces of documentation that have different purposes and to use appropriate organization to tie them together.</li><li>• To be familiar with emulating in software, the fetch-decode-execute cycle of a CPU.</li><li>• To be familiar with the concept of a 'machine' and its implementation via either translation or interpretation on lower level machines.</li><li>• To be familiar with making engineering decisions involving tradeoffs (e.g., space-time tradeoffs in choosing a table implementations).</li><li>• To be familiar with the importance of communication skills, including oral, email, and other written documents such as meeting minutes.</li><li>• To be familiar with software testing strategies including black-box versus white-box, unit testing, integration testing, top-down versus bottom-up testing, and construction and implementation of a test plan.</li><li>• To be familiar with the economic and social forces that often drive technology to explain developments in system software.</li><li>• To be familiar with using one structured approach to large software design to carry out a large group project.</li><li>• To be exposed to issues in systems programming as opposed to applications programming.</li><li>• To be exposed to memory management issues including caching, virtual memory, etc.</li><li>• To be exposed to one-pass macro processing techniques.</li></ul>
<b>GEC Writing</b>	<p>Third Writing Course Expected Learning Outcomes</p> <ul style="list-style-type: none"><li>• To master analyzing the intended audience for a written document and to write an audience profile.</li><li>• Through critical analysis, discussion, and writing, students extend their ability to express ideas effectively</li><li>• Develop basic skills in expository writing and oral expression</li><li>• Develop skills in effective communication and in accessing and using information analytically</li><li>• To be familiar with proofreading own and others' writing.</li></ul> <p>Expected Learning Outcomes:</p> <ol style="list-style-type: none"><li>1. Ability to apply writing skills to the major.</li><li>2. Develop and showcase skills in the areas oral articulation of ideas and critical and analytical abilities through reading demanding texts and the synthesizing ideas.</li></ol> <p>In particular in this course you will integrate information from a variety of sources, actively discuss the readings within your team &amp; class and develop a strategy for successful completion of the team project including documentation, software engineering, code development, testing and make presentations regarding your project..</p>

<b>Instructor</b>	Al Stutz, Senior Lecturer Department of Computer Science and Engineering, and Biomedical Informatics. Currently CIO, avetec.org. Retired Chief Technology & Operating Officer and Director of Ohio Supercomputer Center and OARnet. Office hours will be immediately following class until all questions are answered. I am also available for scheduled appointments. If you have a question you can send me e-mail stutz.1@osu.edu or call my CSE office number 247-7338 or my cell phone 614-561-1823 (I can answer 95% of the questions over email/phone) or contact the grader.		
<b>Office Hours</b>	M & W after class or by appointment Drees 297 office phone 247-7338; cell 561-1823		
<b>Graders</b>	Josh Meek meek.65@buckeyemail.osu.edu Jenna McAuley mcauley.16@osu.edu Brian Arand arand.3@buckeyemail.osu.edu		
<b>Text</b>	none, WEB materials and lectures		
<b>Reference</b>	<b>System Software: An Introduction to Systems Programming, Leland L. Beck, ADDISON WESLEY Linkers and Loaders John R. Levine, Morgan-Kauffman Oct. 1999, ISBN 1-55860-496-0. Systems programming, John J. Donovan, McGraw-Hill,1972 Assemblers, Linkers, and the SPIM, Simulator, James R. Larus, Microsoft Research <a href="http://www.cse.ohio-state.edu/~al">www.cse.ohio-state.edu/~al</a> select Larus Book</b>		
<b>Exams</b>	All Exams and Quizzes are closed book and closed notes.		
<b>Grading</b>	Midterm & final exams (2: 150+150)	300	35%
	Software Projects (5: 50+80+100+60+60)	350	41%
	Peer Evaluations (4: (50+50+50+50)/2)	100	12%
	Discussions and Meetings	50	6%
	Instructor Evaluation (1: 50)	50	6%
	<b>Total</b>	<b>850</b>	<b>100%</b>

*In order to pass the class, students must receive an average grade of at least 65% on each of the following:*

*Software Projects  
Both examinations*

*Be careful students fail every quarter due to this rule*

*Discussions and Meetings. Each group is to meet with one of the graders at least once a week beginning in week 2 through the last week of the quarter. The meetings are focused on the labs and group participation. The entire group must be present.*

*Instructor Evaluation, The instructor will consider your class participation and contributions during real-time grading of the labs and your level of participation in the team project.*

<b>Software Projects</b>	SP-0) Preliminary Pass 1 show casing: LC computation, parsing, and symbol table construction
	SP-1) Working Pass 1 of an Assembler with full documentation (80 pts)
	SP-2) Complete Assembler project with full updated documentation (100 pts)
	SP-3) Hardware Simulator with full documentation (60 pts)
	SP-4) Linker with full documentation (60 pts)

<b>Peer Evaluations</b>	Each student will have the responsibility to fill out an evaluation form for their team members for software projects SP-1, 2 and 3+4. PE1-4) Team-mate Peer Review (100 pts) (one for each team software project 1-4)
-------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Bonuses &amp; Penalties</b>	I recommend that you plan your schedule carefully. It is definitely worth your while to avoid turning in any of your assignments late. The penalty is extremely severe.	
	1 to n full calendar days early	10% (maximum) bonus based on project score
	1 calendar day late	10% off the score received on the project
	2 calendar days late	too late
	In order to receive an early bonus the project assignment must be 99% complete and working.	

**Your program must be generating results in order to be graded. Projects that fail to generate output will be assigned a grade of zero, regardless of the documentation.**

**Lectures** The lecture style of the course will be more like project status meetings rather than traditional lectures. I will pace you, lecture on selected topics, and lead discussion. I would expect that a student would spend at least two hours on course material for every hour in class - maybe more!!!! Come to class prepared to ask questions. The graders will attend the start of many classes to answer project questions. After the first several weeks we may opt to make a lab day in lieu of a formal lecture.

**Team Work** All project assignments (except evaluations) will be done in teams. Each person is to make their own arrangements to form a team. The team should consist of four people. Each team member must do equal work across the entire set of assignments. One person should not be responsible for the documentation, design or coding alone. If a team member is not doing equal work it is the responsibility of the other members to let me know. There are actions that I can take that will not penalize the team. Primarily the graders will grade the projects. The graders will expect you to arrange a time when the whole team can meet. The graders will have a series of test programs to evaluate your code. This will give the team an opportunity to explain the program so a fair grade can be assigned. **If there is evidence that a team member is not providing the same level of effort as the rest of the team, then different grades may be assigned.** If one person opts to do MOST of the work there is no guarantee that he/she will receive a better grade. If I feel that he/she blocked the other people from participating in the project he/she may end up with a grade lower than the team. If I feel that a person is simply not participating then that person will receive a lower grade. None of the tasks can be completed by one or even two people--share the load.

**Labs** You may use Visual C++, C#, RESOLVE, JAVA, or any programming language of your group's choice. Only one submission needs to be turned in by each team for grading. If your software project has known errors, you are responsible for reporting them when the project is turned in via the ERRATA Report

**Submitting** **REQUIRED:** You **MUST** submit your software to us by **CD (n=2)** and via an email attachment. For JAVA you will need to submit a **JAR** file ready to run the archive. To submit your lab electronically be sure to email to all the graders and the instructor.

When you submit the lab electronically and on the CD you need the follow the naming standard below (when you submit electronically the same string should be in the subject line):

<group-id>.sp<sp#>.{zip, jar}.{early, on time, late}]

For example:  
 "c560ab04.sp3.zip.early"  
 "c560ab03.sp2.jar.ontime"  
 "c560ab05.sp1.zip.late"

**Misconduct** Cheating, copying programs from other teams or destroying files or sessions of others will not be tolerated. Those caught will be reported to the University Committee on Academic Misconduct. We know that there are examples and templates for making compilers. Use of these templates is considered academic misconduct. Any evidence found of such programs will be dealt with accordingly. Your entire team is responsible to verify that the team members wrote the software provided.

**Test Plan** **REQUIRED:** Describes the testing philosophy and the purpose of each test run. It is also to assist the team in the development of logical and well-structured tests. This task is very large and requires significant attention to detail.

Test #	Purpose	Expected Results Achieved	Concerns
		YES/NO	
		YES/NO	

**Documentation** **REQUIRED:** Each project will require "professional looking documentation." The documents are to be web based, well written, and in a "pleasing, understandable, and logical format." There must be a table of contents with relative links to the specific section. There are three major sections (User Guide, Developers Guide, and Test Plan). **Groups must provide the complete web content on two identical CD's.** See *Web Documentation grading tool*.

Writing Quality and design of the publications are very important. Don't simply take the course handout materials and "paste it" into your documentation. I need to see your ideas on how to make a usable set of documentation. A sample of minimum components is located in the Grading Steps Section.

Common sense: The document should be designed for a reasonably skilled technical person to read and understand. The material should be clearly written with the facts and diagrams needed to understand how to code, submit, debug, etc. This is not a sales brochure. The product has been purchased. Animation is rarely a valuable tool for this type of user.

**Log Book**

**REQUIRED:** A record of team meetings including minutes, task assignments, record progress, documentation assignments, documentation progress and to record task due dates. This should be done for formal or informal meetings. At the end of each section for each meeting there must be a summary of commitments.

Task	Who	Initials	Due Date
------	-----	----------	----------

Each team should use a spiral bound 8.5" X 11" notebook. Other methods will be allowed as long as a record can be kept as to clear assignments and responsibility. The logbook must accessible at the grading session.

**DED**

**REQUIRED:** For each variable declared in your code a Data Element Dictionary item must be included. You must follow the format below:

Variable name	Module where defined	Data type	Local or Global	Purpose

**Coding Style**

**REQUIRED:** For each procedure you are **required** to follow the template below:

Procedure Name: Description: Specification reference codes: Calling Sequence Input parameters Output parameters  Errors Conditions Tested: Error Messages Generated: Original Author: Procedure Creation Date: Modification Log: Who when why Coding standards met: "signed code leader" Testing standards met: signed test leader"	This is exactly the same format required for the Developers Guide. So be creative so that you can strip out the template and move a copy into your web documentation.
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

The rule for comments is fairly simple. You should be able to use the template rather than embed comments in the code. There may be occasions where specific lines of comment are necessary in complex sections of code.

**Specification Reference Code:** As the machine and the language were developed the design team assigned Reference Codes for each requirement. It is important that within the source code we clearly indicate which of the specifications this module on instruction set is designed to address.

**Note: Be sure to write your code in modular form. Many routines can be shared. Also test your modules individually. This will reduce the code development time.**

**Errata Report**

**REQUIRED:** Explains any known errors (coding and documentation) and the proposed CODE to repair the error. Reporting the errors and their corrections will give us the opportunity to credit your project for some of the lost points. If there are no known errors then you must turn in an Errata Sheet indicating that there are no known errors. **This must be a separate link in the web documentation. There is a penalty if an errata report is not provided.**

- Expectations** I expect that a student in this class on a regular basis to:
- Read email
  - Meet with one of the graders on a weekly basis
  - Be Web knowledgeable & check the class web-site
  - Read the class materials: C.R.T., PPT, lecture, & reference notes
  - Bring questions to class
  - Develop a program design before writing any code
  - Test the code fully BEFORE coming to a grading session
  - Meet or exceed due dates
  - Meet or exceed project and class requirements
  - Attend at least the 1<sup>st</sup> half hour of every class – project discussion and issues

- Be Prepared** **REQUIRED:** Individual Student Responsibility
- Visit the graders BEFORE the assignments are due to discuss:  
Design, requirements, expectations, questions about the assignment, interactive grading, documentation
  - Work with your team and share the workload and functions (design, coding, testing, documentation)
  - Come to the interactive grading prepared with:
    - Working program
    - Grading Sheet
    - Peer Evaluation form
    - Written Peer Evaluation

### Team Leadership Assignments

Below is a list of key leadership positions for a successful software project. I am recommending that people be assigned these roles, however all group members MUST participate in each one of the areas.

1. Project Leader  
Overall project manager; Schedules meetings; Verifies work of other leaders.
2. Design Leader  
Organizes the materials for the design document and class presentation; verifies that the software, documentation and testing meets the criteria presented in the design document; that people are meeting their commitments regarding the schedule.
3. Documentation Leader  
Verifies that documentation is a key priority for the team, verifies that the documentation being submitted by team members meet the group standards; develops layout, TOC format, etc.; that people are meeting their commitments regarding the schedule.
4. Test and Validation Leader  
Makes sure that each procedure is tested prior to installation in the source tree; that testing standards are met; develops test documents (including a table of contents and a description of each test; that people are meeting their commitments regarding the schedule.
5. Code Manager  
Verifies that the developer is following the group coding standards, that people are meeting their commitments regarding the schedule.

### Standard Libraries

You are allowed to use the standard C++ libraries from Microsoft Visual Studio. This includes libraries such as the STL. In theory, the STL is supposed to be optimized, but is typically slow in use. You are still allowed to use the STL, even though it is slow. Basically we're not going to penalize you just because implementations of the STL are slow. It is not acceptable to use any libraries that are not standard C++, Java, or C# libraries. Any use of non-standard libraries will be treated as academic misconduct.

### Coding

Write clean concise code. Comment your code using the template, include the specified function headers and class headers where appropriate. Do not use "magic numbers". You are also required to implement error-handling mechanisms in your code. Do not write error messages inline. You should try to generalize your code as much as possible. The more you abstract away the architecture from the parser and code generator, the better your software will run. Hints: Use n-functions for data input/output. Check bounds on all strings and variables.

## Tentative Schedule

WK	Date	Text	Topics	What's due
1	Sep. 21	Handouts	Introduction, Course review Meet graders Review assignments High Level Design Working on Team Reading/Writing Binary files Software Engineering Test Plans and Software Guide Design for testing	Form Groups
2	Sep. 26	CRT	Machine Architecture Assemblers	
2	Sep. 28	CRT	Assemblers Power of a robust intermediate file	
3	Oct. 3		Assemblers	
3	Oct. 5	CRT	SP0 Presentation	SP0
4	Oct. 10		Assemblers	
4	Oct. 12		Operating System primitives	
5	Oct. 17		Lab work Catch-up SP1 presentations	SP1 & PE1
5	Oct. 19		Review and Catch-up	
6	Oct. 24		Exam	EXAM
6	Oct. 26	CRT	Exam Returned Writing a linker/loader	
7	Oct. 31	CRT	Writing a hardware simulator	
7	Nov. 2	CRT	Writing a linker/loader Writing a hardware simulator	
8	Nov. 7	CRT	Writing a linker/loader Writing a hardware simulator	SP2 & PE2
8	Nov. 9		SP2 presentations	
9	Nov. 14	Levine 1-3 Levine 4-5 Levine 6-8	Loaders and Linkers Linker Design, relocations, jumps Dynamic linking and loading	
9	Nov. 16		Exam review	
10	Nov. 21		Lab Presentations	SP3, SP4 PE-3, PE-4
10	Nov. 23		Holiday	
11	Nov. 28		Preparing for the final	
11	Nov. 30		Catch up	
12	Dec. 5		Final 5:30 to 7:30 in classroom	EXAM