CAS CS 210: COMPUTER SYSTEMS
FALL 2018
Computer Science Department
College of Arts and Sciences
Boston University

Important Dates: Last Day to DROP Classes without a 'W' grade is October 9, 2018. Last Day to DROP Classes with a 'W' grade is November 9, 2018.

INSTRUCTOR Prof. Abraham Matta, 111 Cummington Mall, **MCS 140E** (Math & Computer Science Building). Phone: (617) 353-8919. Email: matta @ bu.edu

To reach me at times other than my office hours, please send me email. I check my email regularly.

TIME AND PLACE Tuesdays and Thursdays 2-3:15pm. STO B50. STO is the Stone Science Building, located at 675 Commonwealth Ave.

COURSE DESCRIPTION This course takes a programmer's perspective to learn about the inner structure of computer systems, the design and implementation of abstractions that enable humans to use computers efficiently, the basics of C and assembly programming, the mapping between C and assembly, and between assembly and machine language, and the role of operating system software.

Our goal is to learn what a “beautiful” computer system is and how it works. Quoting an Italian painter named Elio Carletti: *Beauty is the summation of the parts working together in such a way that nothing needed to be added, taken away or altered.*

We will also learn how to become strong (“brilliant”) programmers who write fast and reliable programs. As the saying goes: *Computers are incredibly fast, accurate, and stupid; humans are incredibly slow, inaccurate and brilliant; together they are powerful beyond imagination.*

CS 210 is a core (group A) course for computer science majors. It provides background for courses in the systems area such as operating systems, compilers, and networks, not to mention more advanced courses in computer architecture.

This course fulfills a single unit in the following BU Hub area: Quantitative Reasoning II.

LEARNING OUTCOMES We will learn of core computer architectural concepts (software hierarchy/abstractions, portability, stored program concept); how data (integers, real numbers, text) is represented and operated upon at the high level (C language) and at the low (machine) level; how can one write high-level code that manipulates bit vectors efficiently; how is computer arithmetic different from real arithmetic, and how does it affect computations we do at the high level; what is visible to programmers at the machine (instruction) level (using Intel's Instruction Set Architecture as a case study); what do compilers try to do: conventions in program translation, memory layout and access for different data structures (arrays, structures), and optimization aspects; what high-level transformations can users do to influence how compilers translate their high-level code so it runs faster; how can one reverse engineer

binary code (e.g. malicious code) to understand its behavior; what are the security implications of writing unsafe code (e.g., buffer overflow vulnerabilities); what do operating systems do: abstractions of file access, virtual memory, processes; what high-level transformations can users do to their high-level code to exploit the memory hierarchy (increasing locality of references); how exceptions (synchronous and asynchronous) are handled by the OS and how this affects program execution time.

PREREQUISITES This course assumes that students have a solid background in high-level programming (in Python, Java, or C++) from CAS CS 111, CS 112, or equivalent. *CAS CS 112 is essential to ensure students have strong programming skills*. A solid working knowledge of operating systems, such as Unix/Linux and Windows, is also assumed. CS 131 or MA 293 is helpful for the material on Boolean logic and data representation, but is not essential.

We will make use of C as an example high-level language because its syntax and semantics are closer to assembly language concepts. C is more suitable for exposing low-level system details and achieving higher performance in real implementations. If you know C++, this should not distract you since C is mostly a subset of C++. C and C++ both share many of the same fundamental programming constructs. C, however, lacks support for object-oriented programming. On the other hand, if you know Java or Python, there are aspects of C, particularly pointers, explicit dynamic memory allocation and formatted I/O, which do not exist in Java and Python. Fortunately, the C language is relatively small and there are many good references (see required text).

SYLLABUS This is a tentative syllabus and subject to change. Dates are approximate. Speed and level of coverage will depend to some extent on the maturity and background of the class. Unless noted otherwise, **readings indicated below are required and from the CMU textbook;** more detailed section-by-section reading assignments are noted.

<table>
<tr><th colspan="2">Dates</th><th rowspan="2">Topics</th><th rowspan="2">Readings</th></tr>
<tr><th>Tue</th><th>Thu</th></tr>
<tr><td>9/4</td><td>9/6</td><td>Overview: Course Goals, Introduction to Computer Architecture & Organization, Software Hierarchy.</td><td>Ch. 1</td></tr>
<tr><td>9/11</td><td>9/13</td><td>Data Representation: Conversion between Number Systems, Character Codes, Boolean Algebra, Bit-level & Logical Operations in C, Integer & Floating-Point Numbers.</td><td rowspan="2">Ch. 2</td></tr>
<tr><td>9/18</td><td>9/20</td><td>Computer Arithmetic: Addition, Subtraction, and Operations in C. (Skip sections 2.3.4-2.3.7; optional IEEE floating point standard: 2.4.2-2.4.6)</td></tr>
<tr><td>9/25</td><td>9/27</td><td>Instructions: Intel Instruction Set Architecture (ISA): arithmetic, data transfer, addressing modes, …</td><td>Ch. 3</td></tr>
<tr><td>10/2</td><td></td><td>Review for Exam 1</td><td></td></tr>
<tr><td></td><td>10/4</td><td>Exam 1</td><td></td></tr>
<tr><td></td><td>10/11</td><td rowspan="4">More Instructions: control instructions, procedures, Arrays, Structures, Assembly versus Machine Language. (Skip sections 3.9.2 and 3.11)

10/9: No lecture, Substitute Monday Schedule</td><td rowspan="4">Ch. 3</td></tr>
<tr><td>10/16</td><td>10/18</td></tr>
<tr><td>10/23</td><td>10/25</td></tr>
<tr><td>10/30</td><td>11/1</td></tr>
</table>

Dates		Topics	Readings
Tue	Thu		
11/6		Review for Exam 2	
	11/8	**Exam 2**	
11/13		Program Optimization: Program Optimization. ***(Read sections 5.1-5.6)*** If time permits, Translation, Linking & Loading ***(Read sections 7.1-7.9)***.	Ch. 5 & 7
	11/15	Memories: Memory Hierarchy, Locality, Caches: Direct Mapped & Performance, Multi-level Caches. ***(Read sections 6.1.4, 6.2-6.5)*** **11/22: No lecture, Thanksgiving Recess (Wed 11/21-Sun 11/25)**	Ch. 6
11/20			
11/27	11/29	Memory & Operating systems, Virtual Memory: Pages and Page Tables, Exceptions. ***(Read sections 9.1-9.6.1, 8.1, 8.2)*** If time permits, Dynamic memory allocation (***Read section 9.9***).	Ch. 9 & 8
12/4			
	12/6	Input/Output: I/O and OS: Memory-mapped & Instructions, Interrupts, Direct Memory Access (DMA), Disks, … ***(Read section 6.1.2)***	Ch. 6
12/11		Wrap-up & Review for Final Exam (**last lecture**)	
		Final Exam: Tuesday, December 18th, 3-5pm	

OFFICE HOURS Tuesdays 3:30-5:30pm. MCS 140E.

The purpose of the office hours of the Instructor and Teaching Fellows is to answer specific questions or clarify specific issues. Office hours are not to be used to fill you in on a class you skipped or to explain entire topics. Please come to class and to your discussion session.

TEACHING FELLOWS.
Yara Awad.
Email: awadyn @ bu.edu
Office Hours: Wednesdays 6-7:30pm, and Fridays 4-5:30pm. EMA 302.

Tommy Unger.
Email: tommyu @ bu.edu
Office Hours: Wednesdays 1:30-3pm, and Thursdays 9:30-11am. EMA 302.

The Teaching Fellows (TFs) will hold their office hours in EMA 302 (Undergraduate lab in the Engineering Annex at 730 Commonwealth Avenue on the third floor in room 302), since it is usually more convenient to answer lab-related questions.

The TFs will lead the discussion sessions. The objective is to reinforce the concepts covered in the lectures, and answer questions (or provide clarifications) regarding the homework and programming/lab assignments.

The TFs will provide, in coordination with the instructor, online resources (FAQ, helpful hints, etc.) and help sessions on certain topics. Discussion related materials will be posted on the TF's website.

DISCUSSION SECTIONS Students attend one of these sessions each week.

- All sessions in **EMA 304** (third floor of 730 Commonwealth Avenue): Mondays 9:05-9:55am (A2), 10:10-11am (A3), 11:15am-12:05pm (A4), 12:20-1:10pm (A5), 1:25-2:15pm (A6), and 2:30-3:20pm (A7).

Tommy will lead the discussions for A2-A4, and Yara for A5-A7.

The following is a tentative schedule of the discussion sections. The TFs will maintain a **web page** with the actual schedule and related material. Unless noted otherwise, **readings indicated below are required and from the C textbook** - more detailed section-by-section reading assignments will be noted. You may be asked to submit some lab reports as part of your homework assignments or at the end of your discussion sessions.

Dates	Topics	Readings
9/10	**Getting started:** connecting to CS servers (puTTY, ssh), basic Linux commands (mkdir, ls, cd), editing files (emacs), C Primer (pointers, scanf), compilation (gcc), using gsubmit	Ch. 1, 2 + online references
9/17	Types, operators, PS#1 discussion	Ch. 3, 11
9/24	Debugging (gdb), basics of Makefile, discussion of Data Lab assignment	Ch. 13, 17 + online references
10/1	Exam 1 Review	
10/9	10/8 Holiday - Columbus Day Tuesday 10/9 is Monday's schedule ~~Arrays, pointers, strings~~ [cancelled: TFs on conference travel]	~~Ch. 6, 9, 10~~
10/15	Review of assembly instructions, reverse engineering binary code using gdb	Ch. 17
10/22	Arrays, pointers, strings, structures	Ch. 6, 9, 10, 8
10/29	Control flow, PS#2 Discussion	Ch. 4, 5
11/5	Procedures, recursion, discussion of Assembly Lab assignment. Exam 2 Review	Ch. 7
11/12	Structures, discussion of Assembly Lab assignment Veterans Day but classes are held	Ch. 8
11/19	Discussion of Performance Lab assignment: matrix blocking, loop unrolling	Online references + section 5.8 in CMU text
11/26	Discussion of Performance Lab assignment	
12/3	PS#3 discussion	
12/10	Final Exam Review	

GRADING Grading (except for the final exam) is done by a number of class graders, under the direct supervision of the Teaching Fellows and the Instructor. If you have an issue with a grade (homework or exam), **please submit a regrade request on**

Gradescope. **Grades must be appealed within one week of receipt**, but December 11th is the last day to request a regrade.

TUTORING HELP You can get extra help during the TF tutoring hours scheduled in EMA room 302. Note that terminal assistants in the EMA 302 lab are not supposed to help with course material, but to maintain the lab environment.

TEXTBOOKS

[Required CMU Text] Computer Systems: A Programmer's Perspective, Randal E. Bryant and David R. O'Hallaron, Pearson, Third Edition, ISBN-10: 0134154304, ISBN-13: 9780134092669.

[Required C Text] Programming in C, Stephen G. Kochan, Addison-Wesley, Fourth Edition, 2015, ISBN-10: 0-321-77641-0, ISBN-13: 978-0-321-77641-9.

Both books are available from the BU bookstore. You can also check the publisher's site or sites like Amazon for cheaper options (rental, electronic or kindle versions).

ONLINE RESOURCES We will be using Piazza for class discussion. The system is highly catered to getting you help fast and efficiently from classmates, the TFs, and the Instructor. Rather than emailing questions to the teaching staff, you are encouraged to post your questions on Piazza. If you have any problems or feedback for the developers, email team@piazza.com. Find our class page at:
https://piazza.com/bu/fall2018/cs210/home

Besides the Q&A's, all course material will be accessible through Piazza. You should regularly visit Piazza for online references (including C and Unix tools) and up-to-date information regarding readings, assignments, exam-related material, announcements, etc.

For grading and grade reporting, we will make use of Gradescope.

GRADING POLICY There will be two midterm exams and one final exam, which will include all material that is covered from the beginning of the semester until the day of the exam. All exams will be closed books and closed notes, except for cheat sheet(s) that the instructor will provide, or your own handwritten 8.5"x11" cheat sheet, as will be announced by the instructor before each exam. There will be absolutely no make-up exams, except for medical emergencies. In that case, blue slips from Health Services will not be accepted; you must justify your medical problem with a letter from a doctor, specifying the period of time during which you were unable to attend one of the exams.

Short quizzes may be given throughout the semester to make sure you are doing the readings on time, and also as a measure of attendance. There will also be assignments and labs, which will be a combination of "pencil and paper" problems and programming assignments. There will be about 3-4 problem sets, and 3-4 programming/lab assignments in C or Intel assembly language. Problem sets may also involve smaller scale programming/lab exercises, *e.g.,* write or debug small programs, etc.

Your final grade will be determined approximately as follows:

- 30% by the average of the midterm exams
- ~~20%~~ 25% by the final exam
- 45% by the assignments (roughly 15% on problem sets and 30% on programming)
- ~~5% by the quizzes/attendance~~

The midterm exam average will be tentatively weighted 60% of the best grade and 40% of the lower grade.

Unless automated and otherwise specified, the grading of programming assignments will be based on the following policy:

- **Program**: works correctly (60%); in-line documentation (10%); design quality (10%)
- **Design document**: how it works (5%); tradeoffs and extensions (5%)
- **Testing document**: compilation instructions (5%); thoroughness of test cases (5%)

We will use csa1.bu.edu, csa2.bu.edu, and csa3.bu.edu, 64-bit Intel-compatible Linux CS machines to grade your programming/lab assignments. Although you may use your own machine, it is your responsibility to ultimately port your assignment to one of our CS machines to make sure it is graded correctly.

Each assignment will have a due date. If late submissions are allowed for an assignment, there will be 10% penalty per day for late submissions. But, no late assignments will be accepted after two days from the due date, and the last day to submit any late assignments is December 11th. Extensions may be granted only for religious holidays and certified medical reasons.

No incompletes will be given, except for reasons of dire illness shortly before the end of the course, and only if a significant amount of work has been completed (*e.g.,* attending lectures, handing in most assignments, and attending the midterms).

Attendance is important: I will take attendance at any time. I will depart from the textbook and its flow on occasions, and I will not provide backup lecture notes on certain additional details that I will cover in class, so it is imperative that you attend all lectures and take careful notes.

COLLABORATION POLICY You are strongly encouraged to collaborate with one another in studying the lecture materials and preparing for quizzes and exams. You may discuss ideas and approaches to the assignments with others (provided that you acknowledge doing so in your solution), but such discussions should be kept at a high level, and should not involve actual details of the code or of other types of answers. ***You must complete the actual solutions on your own***.

ACADEMIC HONESTY We will assume that you understand BU CS's Academic Conduct Code: https://www.bu.edu/cs/undergraduate/undergraduate-life/academic-integrity/ Prohibited behaviors include:

- copying all or part of someone else's work, even if you subsequently modify it; this includes cases in which someone tells you what you should write for your solution
- viewing all or part of someone else's work
- showing all or part of your work to another student
- consulting solutions from past semesters, or those found online or in books
- posting your work where others can view it (e.g., online).

Incidents of academic misconduct will be reported to the Academic Conduct Committee (ACC). The ACC may suspend/expel students found guilty of misconduct. ***At a minimum, students who engage in misconduct will have their final grade reduced by one letter grade (e.g., from a B to a C).***

We may use an automated plagiarism checker. Cheating will not be tolerated under any circumstances. Handing in your own work a day or two late will affect your grade far less than turning in a copy of someone else's work on time!

CAS CS 210: COMPUTER SYSTEMS
FALL 2018

ASSIGNMENTS

Dates are tentative, until the assignment is actually posted or handed out.

All assignments are to be completed **individually**. See Academic Honesty on the syllabus. When explicitly specified, you may have the option to work in teams of two on large programming assignments. Large assignments may have multiple deadlines for submission in phases.

PS=Problem Set, PA=Programming Assignment

Assignment	Out Date	Due Date(s)
PA#0 (First C program)	Fri 9/7	Fri 9/14
PS#1 (Representation & Arithmetic)	Fri 9/14	~~Fri 9/21~~ Sun 9/23
PA#1 (Data Lab)	~~Fri 9/21~~ Sun 9/23	~~Fri 9/28~~ Sun 9/30
EXAM 1 on Thursday 10/4		
PS#2 (Assembly)	Fri 10/12	~~Fri 10/26~~ Sat 10/27
PA#2 (Assembly Lab)	~~Fri 10/26~~ Sat 10/27	Sat 11/3 & Sat 11/17
EXAM 2 on Tuesday 11/8		
PA#3 (Performance Lab)	Fri 11/16	Fri 11/30
PS#3 (Memory)	Fri 11/30	Fri 12/7
FINAL EXAM on Tuesday 12/18		

GUIDELINES ON SUBMISSIONS OF ASSIGNMENTS

Assignments will be available on-line.

You are required to **submit an electronic version** of your homework solutions and your programming/lab documentation/reports. We will use **Gradescope** to grade your submissions online. You can type and submit your write-up in PDF, or if handwritten, you can scan your write-up and submit a PDF. The timestamp on your electronic submission should indicate submission by the due date/time.

Graded assignments will be available online from Gradescope.

Check for Assignments regularly. Start early!

PROGRAMMING / LAB ASSIGNMENTS

You will be required to **submit an electronic copy of your code using the gsubmit program** (as described below). For supporting documentation of your code and any requested written report, you should submit electronic copies using Gradescope (as described above).

When explicitly specified, you may have the option to work in teams of two on large programming assignments.

General Requirements on What to Submit

Unless otherwise specified, the program you submit should work correctly and be documented. You should submit an electronic copy of the following:

1. **Program:** a program listing containing in-line documentation (i.e., comments).
2. **Design document:** a separate file (a page or so) describing the overall program design, a verbal description of "how it works", and design tradeoffs considered and made. Also, describe possible improvements and extensions to your program (and sketch how they might be made).
3. **Testing document:** a separate file describing how to run your program. Specify the steps that must be followed to successfully run your program. Also, describe the tests you ran on your program to convince yourself that it is indeed correct. Also, describe any cases for which your program is known not to work correctly.

If a programming assignment asks you to follow a specific design, then you don't have to describe that design in your documentation, but you must still describe possible improvements and extensions.

HOW TO SUBMIT AN ELECTRONIC COPY USING GSUBMIT FROM YOUR CSA ACCOUNT (*Only plain ASCII files!*)
[Adapted from D. Metcalf's note]

gsubmit is an electronic file submission engine which will submit files or directories of files to the grader so they can be marked.

Every file submitted by a given student for a given assignment should have a unique file name. If a file is submitted with a duplicate name it will either overwrite the file or generate an error message.

To make it easy for the grader to find the files relating to a specific assignment, all files for each assignment should be stored in a subdirectory called ps1, ps2, pa1, pa2, pa3, etc. and the entire directory should be submitted.

To submit an assignment:

- Create a subdirectory "pa#", where # is the assignment number. This is done using the mkdir command: e.g., **mkdir pa5**
- Copy all files necessary for that assignment into the new subdirectory, using the cp command: e.g., **cp prog1.c pa5**
  Be sure to copy only the files you need to submit into this subdirectory.
- Use **gsubmit** to submit the entire subdirectory: **gsubmit cs210 -cp pa5**
  If submission is successful a status message will be printed.

To submit a file to an already-submitted subdirectory:
If you only submitted part of the assignment and would like to add another file:

- To submit a file README.txt to subdirectory pa5, type (at the prompt):
  gsubmit cs210 -cp README.txt pa5

To resubmit a file:

- To resubmit a file prog1.c in subdirectory pa5, first un-submit the file:
  gsubmit cs210 -rm pa5/prog1.c
- Then resubmit it:
  gsubmit cs210 -cp prog1.c pa5

List all files which you have submitted:

- To list all files that you have submitted, type: **gsubmit cs210 -ls**

Looking at a file which has already been submitted:

- To look at a file that has already been submitted, type: **gsubmit cs210 -cat pa5/prog1.c**
- You can store this in a file foo.c by typing **gsubmit cs210 -cat pa5/prog1.c > foo.c**

Where do submitted files go?

Each student who submits an assignment has a subdirectory created to hold his/her files, in a directory for the specified course. This is called the student's “submission spool directory”.

How can the grader tell when a file has been submitted?

Every gsubmit command is automatically logged in a log file, along with a time stamp.

For further information:

Note - The information in this document is taken from the gsubmit man page.
For further information, type **man gsubmit**.

Last updated Tuesday, January 8, 2019