IC-UNICAMP

Courses from Institute of Computing

MC921 1s21 Compiler Construction

All students of MC921AB are required to fill in this form. (https://docs.google.com/forms/d/e/1FAIpQLSdiZNqaLxIn3FgxCULdRvYjOmQREm9jgffF8kuDynwhv4vv gxids=7628)

Links

- Agenda (https://docs.google.com/spreadsheets/d/1Ergx7kwWr-a4wrlLmPUeOzk4_FOEKonC/edit?dls=true#gid=1770714224)
- Answer keys (https://drive.google.com/drive/folders/1oQrWYOiN9uzjGES57giI_eBRFIrSixLK?usp=sharing)
- Bulletin board (https://classroom.google.com/u/0/c/MjYzMTIyMDczNjUz)
- Class Meet (http://meet.google.com/ogo-cgqr-kch)
- Disclaimer (https://docs.google.com/forms/d/e/1FAIpQLSdiZNqaLxIn3FgxCULdRvYjOmQREm9jgffF8kuDynwhv4vvhg/viewform?gxids=7628)
- FAQ (https://docs.google.com/document/d/10URgpOOJkmkFjB7m65nRrbP0CoDNnDzbRyqxO20PKO4/edit?usp=sharing)
- Grades (https://docs.google.com/spreadsheets/d/13iwhvzxwYtBlVMGwXHQydSI6hU1WxoHB/edit#gid=1937424955)
- Groups (https://docs.google.com/forms/d/e/1FAIpQLSc2nMP7y1qDtukK67OyfwMGt926uy36S9SqhVNGKTjwWgpHIw/viewform)
- Jamboard (https://jamboard.google.com/d/10eU57_x_M2AeulXZHYDg1oNEEt1dc9r50o5qFJDpAco/edit?usp=sharing)
- Notebooks (https://github.com/mc921-1s21/notebooks-1S21)
- Problem set and solutions (https://drive.google.com/drive/folders/1MraDcj0Zx7ARALdS9Oe00LqRkIYOgIEw?usp=sharing)
- Slides (https://drive.google.com/drive/folders/15w5ygLec6KBsy_mlI_WdVtX8DoUxAPDl?usp=sharing)
- Videos (https://drive.google.com/drive/folders/1_QMpZtziwHS-a13x0gVIqbw3c9xBVmuK?usp=sharing)

Adm

- Theory sessions: Tue. (8h – 9h) and Thu. (8h – 9h)
- Lab sessions: Tue. (9h – 10h) and Thu. (9h -10h)
- TA sessions: Mon. and Wed. (12h – 13h)
- Instructors: Guido Araujo and Marcio Pereira
- TAs: Vitória Dias and Luciano Zago

Disclaimer

Before starting the course all students are required to fill in this form, (https://docs.google.com/forms/d/e/1FAIpQLSdiZNqaLxIn3FgxCULdRvYjOmQREm9jgffF8kuDynwhv4vvhg/viewform?gxids=7628) until Tue. 23/03. In this form the student acknowledges that he is aware of certain significant recommendations for an adequate course performance.

Groups

For the course projects students will be divided in groups of at most 2 students each. Please fill in this form (https://docs.google.com/forms/d/e/1FAIpQLSc2nMP7y1qDtukK67OyfwMGt926uy36S9SqhVNGKTjwWgpHIw/viewform) to define group members. Changes in groups are only allowed in the periods below.

- 16/03 – 23/03, during enrolment alteration period.
- 20/05 – 01/06, during enrolment cancelation period.

Bulletin board

The course has a bulletin board (https://classroom.google.com/u/0/c/MjYzMTIyMDczNjUz) for announcements and posts about the course progress. Students are required to closely follow the messages posted on the board, as they include very relevant courseware information.

Syllabus

Classes will use a set of slides (https://drive.google.com/drive/folders/15w5ygLec6KBsy_mlI_WdVtX8DoUxAPDl?usp=sharing) and videos (https://drive.google.com/drive/folders/1_QMpZtziwHS-a13x0gVIqbw3c9xBVmuK?usp=sharing), available through the course agenda (https://drive.google.com/file/d/1Ergx7kwWr-a4wrlLmPUeOzk4_FOEKonC/view?usp=sharing). If necessary, additional lecture notes as well as articles discussed in class will be made available. Classes will work asynchronously, and classes times will be used for Q&A sessions.

Slides and videos are intellectual property of the books' authors, instructors or UNICAMP, and cannot be distributed without their previous authorization.

The course will be strongly based on the slides and videos which use material from the following books:

- Andrew Appel. Modern Compiler Implementation in Java (http://www.amazon.com/Modern-Compiler-Implementation-Andrew-Appel/dp/052182060X/ref=sr_1_16?ie=UTF8&qid=1411044376&sr=8-16&keywords=compilers).
- Aho, Sethi and Ullman. Compilers: Principles, techniques and tools (http://www.amazon.com/Compilers-Principles-Techniques-Alfred-Aho/dp/0201100886/ref=sr_1_4?ie=UTF8&qid=1411044323&sr=8-4&keywords=compilers).
- Keith Cooper and Linda Torczon. Engineering a Compiler (http://www.amazon.com/Engineering-Compiler-Second-Edition-Cooper/dp/012088478X/ref=sr_1_2?ie=UTF8&qid=1411044280&sr=8-2&keywords=compilers).

A problem set (https://drive.google.com/drive/folders/1mdoOSPPn5hFXJODf_z29Ade5BL3sWPow?usp=sharing) from the above books and some of their corresponding solutions (https://drive.google.com/drive/folders/1lpU2_vXjBIe71bE2tmr-BvuNZDJOI728?usp=sharing) are provided as references to the level of questions in the exams. We strongly recommend that the students work on these problems. The problem set list is below:

- Appel (2nd Edition): 2.2, 2.4, 2.5, 2.8, 2.9 3.1, 3.3, 3.4, 3.6, 3.9, 3.11, 3.12 e 3.13
- Appel (2nd Edition) : 9.1, 9.3, 10.1,10.5, 11.2(a), 11.3(a),17.1, 17.2, 17.5
- Aho, Sethi and Ullman (1st Edition): 3.16, 4.1, 4.2, 4.11, 4.14, 4.15, 4.33
- Aho, Sethi and Ullman (1st Edition): 9.12(a-c), 9.14, 9.15, 10.1, 10.2, 10.3(a-c), 10.3(g), 10.5, 10.6, 10.7, 10.8
- Cooper and Torczon (2nd Edition):   2.1, 2.7, 2.8, 3.4, 3.5, 3.7, 3.9, 3.10, 3.11 e 3.12

Assignments

The final course grade will be based on 7 programming lab projects, and 2 theory exams.

Projects will use the GitHub Classroom environment, where each project has an associated template repository. Students have to pull the assignment templates locally to work, and push it for testing, and before the deadline for grading. The GitHub system will run the tests and automatically compute the assignment grade. To better understand how this process works please have a look at this video (https://drive.google.com/file/d/1169itCXPkkJE6LmIpShOZtRmpIpz47mZ/view?usp=sharing).

All test inputs for the projects are open, and there are no closed tests. The correct output for each test is open, and their evaluation will take into consideration not only execution correctness but also performance for some projects.

GitHub will automatically close the submission system after each project deadline, and there will be **no extensions**. Hence, we **strongly** recommend that the student submit its work even if the testing is incomplete.

A link to each project notebook, containing a detailed description of the project, programming guidelines, code snippets, etc. can be found in the appropriate entry of the course agenda (https://drive.google.com/file/d/1Ergx7kwWr-a4wrlLmPUeOzk4_FOEKonC/view?usp=sharing). (https://docs.google.com/spreadsheets/d/1yculItZMIjpmQdZoO6D_SlgMkjquGvIdL62GBzOBN88/edit#gid=0)

Grades

Grades (https://drive.google.com/file/d/13PLRArAEwu0C8CNEX_aU9fwxi2aRZSiq/view?usp=sharing)will be available at most 15 days after the project/exam due date. Regarding the calculation of the course final grade, the following rules apply:

- **Exams Average (E)**
  - The average of the exams is E = average(Ei), i = 1-2. Exams will start at the beginning of a Theory session day (i.e. 8h00), and must be submitted at most 24h after.
- **Projects Average (P)**
  - Rule for P1-P5 and P7: the grade of each project is computed as Pi = Ci/Ni * 10,0, where Ci is the number of correct tests, and Ni the total number of tests.
  - Rule for P6:The grade of this project is computed as Pi = Di/Ni * 10,0 + Bi where: (a) Di is the number of tests for which the output is correct and at least one instruction related to the project specification is removed (optimized) from it ; (b) Bi is a bonus (computed only if Di/Ni = 1,0), where Bi = sum(Rj/Sj)/Ni and, for each test j, Rj is the number of instructions in the output code of the reference compiler, and Sj the number of instructions in the output code of the student compiler.
  - The set of projects evaluated for grading Pi (i = 1-7), will consider the six projects resulting after removing from the seven projects' list the one with lowest grade, not considering P6 and P7 which are required.
  - The average of the projects is P = average(Pi), for the Pi corresponding to the highest six grades.
- **Course Average (A)**
  - The course average before the final exam is: A = E * 0.3 + P * 0.7
- **Final (F)**
  - Students with E < 5.0 are required to take the Theory Final Exam (TF).
  - Students with any Pi < 5.0 (after removing the lowest grade) are required to take the Lab Final Exam (LF).
  - Project Final Exam PF = average(Qi), i = 1-7, if Qi >= 5.0 where Qi is the corrected version of project Pi. Otherwise PF = min(Qi), i = 1-7.
  - Final Exam F = min(TF, PF), if TF or PF is smaller than 5.0. Otherwise, F = TF * 0.3 + PF * 0.7.
- **Course final grade (G)**
  - For those who did not take TF nor PF: G = A
  - For those who F < 5.0, G = F, otherwise G = (A + F)/2

Grade review requests must follow the rules below:

- Review requests must be made exclusively through this form (https://docs.google.com/forms/d/e/1FAIpQLSdcwnynAH_6bbZMdyshchFGcZjsXdx-dz1nA2Gz_cI29UFyHw/viewform).
- Review requests will be received only within 48 hours after the grade is released. After that, it will not be considered.
- The review will be done within 15 days after the request is received, and the result will be informed to the student via his/her DAC/Unicamp e-mail.

If the student misses any exam for personal reasons, it should use this form to upload a handwritten signed letter explaining the situation. Any missed exam will be automatically substituted by the grade in the Theory Final Exam (TF). A second missed exam will have no replacement.

Collaboration policy

Exams are individual assignments, and collaboration for their execution is not permitted. Any violation will be considered fraud.

Projects are group assignments (maximum 2 students per group). Groups can collaborate with the goal of understanding and discussing the assignment solution. Nevertheless, code sharing and copying are not allowed, and will be considered fraud.

Each submitted project will be checked for fraud using automatic tools. Only code from the currently submitted project, and not from previously submitted projects, will be considered for fraud evaluation. If the student takes the Project Final Exam (PF), the code from all his/her projects will be checked for fraud.

Frauds will not be accepted, G = 0.0 will be assigned to everyone involved, and the case will be brought to the Undergraduate Dean.

Module	#	Date	Syllabus	Slides	Videos	Exams	Project Available	Due	Project Videos	Notes
1 Lexical Analysis	1	16/3	Tokens and regular expressions	1.1 - 1.3	1.1 - 1.3		P1 Lexer		P1-Lexer	Group definition starts
	2	18/3	DFA and NFA	1.4 - 1.5	1.4 - 1.5				Intro Labs	
	3	23/3	DFA-NFA simulation and conversion	1.6 - 1.7	1.6 - 1.7					Group definition ends
2 Synthatic Analysis	4	25/3	Parser tree introduction and ambiguity	2.1 - 2.2	2.1 - 2.2		P2 Parser	P1 Lexer	P2-Parser	
	5	30/3	LR(0) introduction and construction	2.3 - 2.4	2.3 - 2.4				PyCharm	
	6	1/4	LR(1) construction, ambiguity, and error recovery	2.5 - 2.7	2.5 - 2.7					
	7	6/4	Designing LR parser (TODO)	2.8	2.8					
	8	8/4	LL(1) construction, ambiguity, and error recovery	2.9 - 2.11	2.9 - 2.11					
3 Semantic Analysis	9	13/4	Abstract Syntax Tree (AST)	3.1	3.1		P3 AST	P2 Parser	P3 AST	
	10	15/4	Designing AST (TODO)	3.2	3.2					
	11	20/4	Symbol table and semantic analysis	3.3 - 3.4	3.3 - 3.4				_	_
	12	22/4	Designing semantic analyzer (TODO)	3.5	3.5					
	13	27/4	Exame 1			E1				Classes 1-8
4 Code Generation	14	29/4	Stack-frame	4.1	4.1					
	15	4/5	IR, Trees and DAGs	4.2	4.2		P4 Semantic	P3 AST	P4 Semantic	
	16	6/5	Instruction selection maximal munch	4.3	4.3					
	17	11/5	Instruction selection dynamic programming	4.4	4.4					
	18	13/5	Local register allocation	4.5	4.5					
	19	18/5	Address register allocation	4.6	4.6					
	20	20/5	Linear IR and basic blocks	4.7	4.7		P5 Codegen	P4 Semantic	P5 CodeGen	
	21	25/5	Designing code generator (TODO)	4.8	4.8					Group re-definition starts
5 Data-flow Analysis and Basic Optimizations	22	27/5	Data-flow analysis introduction	5.1	5.1					
	23	1/6	Reaching definitions and UD-chain	5.2 - 5.3	5.2 - 5.3					Group re-definition ends
	24	3/6	Basic optimizations	5.4 - 5.6	5.4 - 5.7					
	25	8/6	Holiday	-	-					
	26	10/6	Designing Data-flow analyzer and optimizer (TODO)	5.7	5.7		P6 DFA	P5 Codegen	P6 DFA	
6 Advanced Optimizations	27	15/6	Liveness Analysis and interference graph	6.1 - 6.4	6.1 - 6.4					
	28	17/6	Global register allocation	6.5 - 6.6	6.5 - 6.6					
	29	22/6	Global register allocation with coalescing	6.7 - 6.8	6.7 - 6.8					
	30	24/6	Loop optimizations	6.9 - 6.12	6.9 - 6.12					
	31	29/6	Designing LLVM optimizer (TODO)	6.13	6.13		P7 LLVM	P6 DFA	P7 LLVM	
	32	6/7	Exame 2			E2				Classes 14-29
7 The End	33	8/7	Lab Q&A session	-	-					
	34	13/7	Lab Q&A session	-	-			P7 LLVM		
	35	29/7	Final Exam			E		F		Classes 1-8 e 14-29