Compilers

Masters in Informatics and Computing Engineering (MIEIC), 3rd Year

João M. P. Cardoso

Dep. de Engenharia Informática
Faculdade de Engenharia (FEUP), Universidade do Porto,
Porto, Portugal
Email:jmpc@acm.org

Instructors

- João M. P. Cardoso
  - Room I137
  - E-mail: jmpc@fe.up.pt
- Ricardo Nobre
  - Room J204
- Tiago Carvalho
  - Room J204

Course Webpages

- SiFEUP (rules, timetable, list of students, etc.):
  - https://sigarra.up.pt/feup/pt/ucurr_geral.ficha_uc_view?pv_ocorrencia_id=384946
- Moodle (organization of the course, mailing-lists, etc.):
  - https://moodle.up.pt/course/view.php?id=1686
- Google Drive (files, documents, etc.)

Objectives

- Provide concepts which allow to:
  - understand the languages' compilation phases, in particular for imperative and object-oriented (OO) languages;
  - specify the syntax and semantics of a programming language;
  - understand and use the data structures and the main algorithms used to implement compilers.

Learning Outcomes and Competences

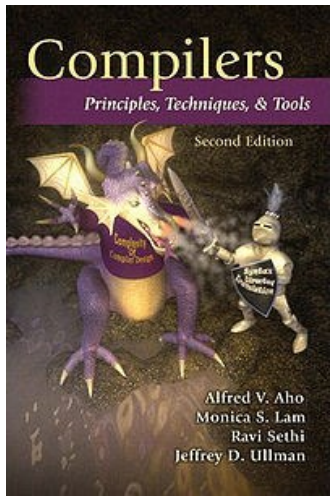
- The skills and learning outcomes will allow students to:
  - develop and implement in software language processing systems of artificial languages and information textually specified under certain lexical and grammar rules;
  - design and implement in software the various compiler stages, namely:
    - regular expressions and finite automata;
    - syntactic and semantic analyzers;
    - semantic analyzers;
    - code optimization;
    - code generators having processors or virtual machines as target;

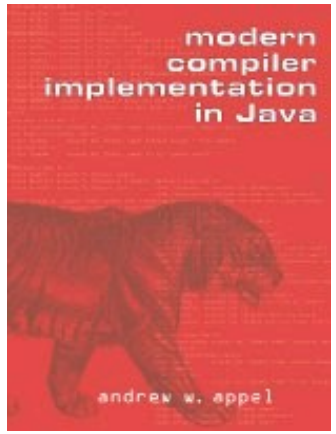
Prior Knowledge

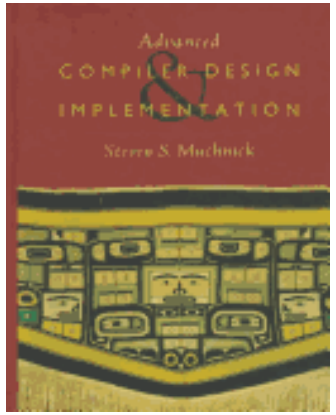
- Pre-requirements (prior knowledge) and co-requirements (common knowledge)
  - Computer Architecture
  - Imperative programming languages, object-oriented programming languages
  - Data structures and algorithms
  - Theory of Computation

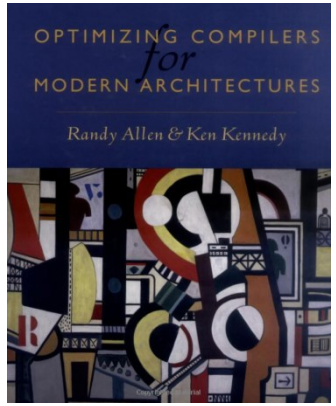
Syllabus

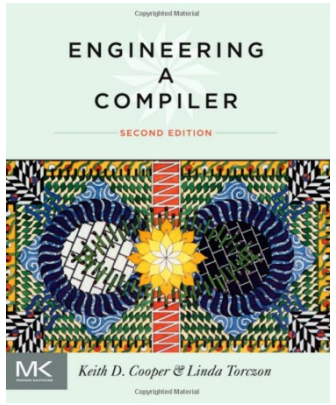
- Introduction. Compilation phases and typical structure of a compiler.
- Lexical analysis. Regular expressions and finite automaton.
- Syntax analysis. Grammars. Syntax analysis' algorithms. Error handling.
- Semantic analysis. Type checking.
- Execution environments. Memory organization and schemes for parameter passing.
- High and Low-level intermediate representations. Intermediate code generation techniques.
- Code generation techniques. Instruction selection, register allocation, and scheduling.
- Compiler optimizations.

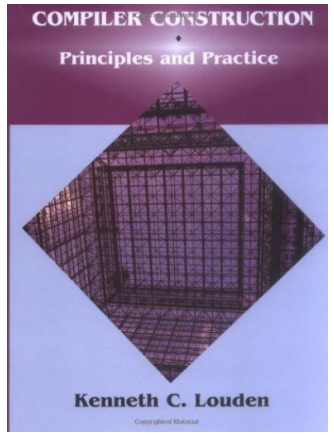












Bibliography

- **Principal**
  - A. Aho, M. Lam, R. Sethi, J. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd Edition, Addison Wesley, 2007. ISBN: 0321486811 (Existe 1ª edição (1986) na biblioteca)
  - Appel, Andrew Wilson, *Modern Compiler Implementation in Java*, 2nd edition. Cambridge University Press, 2002. ISBN 0-521-82060-X
- **Complementary**
  - Muchnick, Steven, *Advanced Compiler Design and Implementation*, Morgan Kaufman Publishers, 1997. ISBN 1-55860-320-4
  - Allen, Randy; and Kennedy, Ken, *Optimizing Compilers for Modern Architectures*, Morgan Kaufman Publishers, 2001. ISBN 1-55860-286-0
  - Cooper, Keith D., and Torczon, Linda, *Engineering a Compiler*, Morgan Kaufmann, 2nd edition, February 21, 2011. ISBN 10: 012088478X
  - Louden, Kenneth C.;Compiler construction. Course Technology, ISBN 0-534-93972-4
  - Pedro Reis Santos, Thinault Langlois; *Compiladores - da Teoria à Prática*, FCA, 2014. ISBN: 978-972-722-768-6 **[in Portuguese]**

Teaching Methods

- 3-hour classes (Ts):
  - Presentation of the topics, exercises related to compiler theory and practice
  - Discussions of ideas, solutions, etc.
- 1-hour classes (TPs):
  - Resolution and discussion of topics related to the project
  - Meeting with instructors

Assessment

- **Assessment Method:** Distributed evaluation without final exam
- **Frequency:** Project (with grade >= 10) and a maximum of 3 absences from the TP classes
- **Final Grade**
  - FIRST ROUND ("Época Normal"):
    - Final Grade = ROUND(0,60*AD + 0,2*T1 + 0,2*T2)
    - T1: grade obtained in the first test [0..20]
    - T2: grade obtained in the second test [0..20]
  - SECOND ROUND ("Época de recurso"):
    - Final Grade = ROUND(0,60*AD + 0,40*EX)
    - EX: grade obtained in the exam [0..20]
  - FIRST ROUND ("Época normal) and SECOND ROUND ("Época de recurso):
    - The maximum final grade is limited to 17 (out of 20). In order to obtain higher grades it is necessary to do an oral exam or additional work.
  - Each student will pass in the course in the "época normal" or in the "época de recurso" if he/she attained the conditions for admission to exams (AD >= 10 and at the most 3 absences from the TP classes), obtained a minimum grade of 8 marks on the Exam (EX), and obtained a Final Grade equal to or greater than 10.
  - AD: grade obtained in the distributed evaluation (lab assignment) [0..20]

Assessment

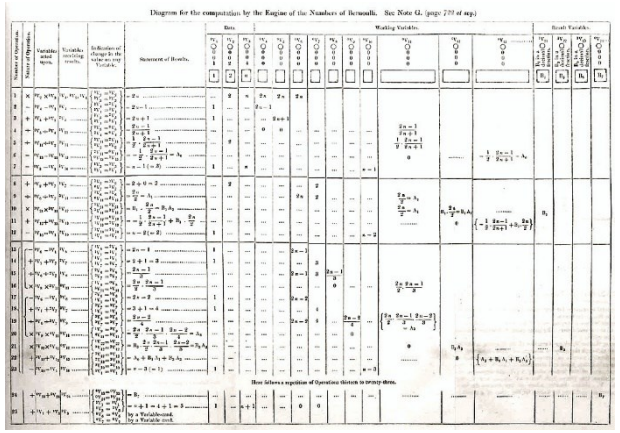
- **AD**: grade of the distributed evaluation (project + participation) [0..20]
- AD grade consists of:
  - Participation: 10%
  - First checkpoint: 10%
  - Second checkpoint: 10%
  - Final work: 50%
  - Presentation/Discussion: 20%

Software

- JavaCC, https://javacc.dev.java.net/
- ANTLR - Another Tool for Language Recognition, http://www.antlr.org/
- JASMIN, http://jasmin.sourceforge.net/
- clang: a C language family frontend for LLVM, http://clang.llvm.org/
- JavaScript parser and CST generator built in JavaScript: http://esprima.org/demo/parse.html
- Graphviz - Graph Visualization Software, http://www.graphviz.org/
- Graph libraries: GraphT, Gephi, etc.
- IDEs: Eclipse, NetBeans, Visual Studio
- COINS, http://www.coins-project.org/international/

Beginning of High-Level Languages

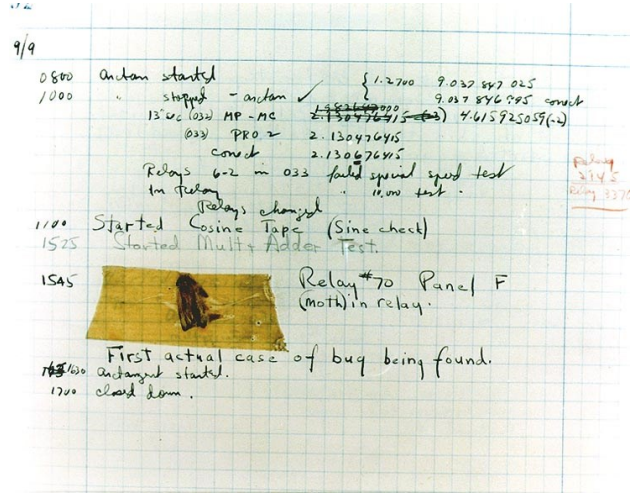
- **Ada Lovelace** (1815-1852)
  - "The **Ada Lovelace Award** is named in honor of the first computer programmer, Augusta Ada Byron Lovelace, whose writings developed the idea of programming and explained the operation and theory of Charles Babbage's Analytical Engine."



In note G, she describes an algorithm for the Analytical Engine to compute Bernoulli numbers.

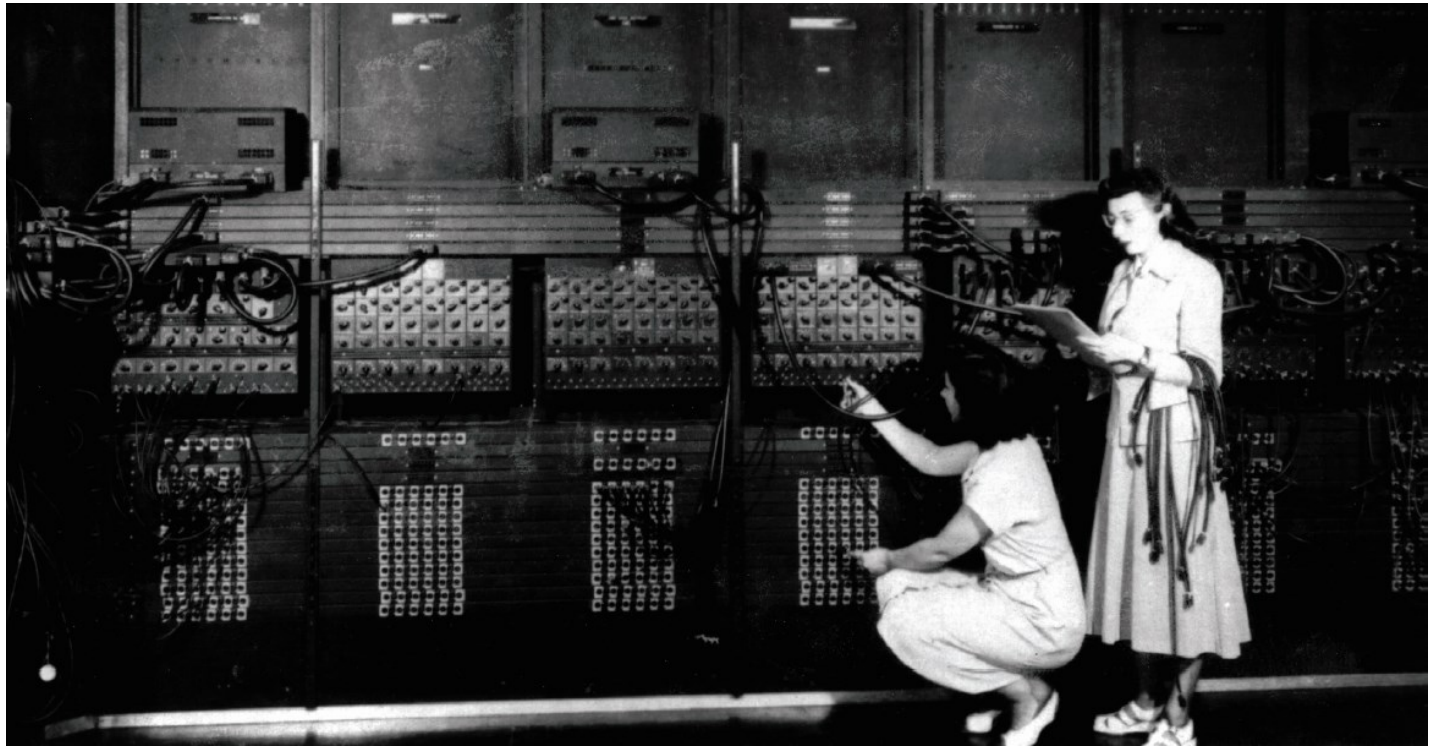
Beginning of the Compiler

- **Grace Brewster Murray Hopper** (1906 –1992)
  - "One of the first programmers of the Harvard Mark I computer in 1944, invented the first compiler for a computer programming language, and was one of those who popularized the idea of machine-independent programming languages." [source: wikipedia]
  - Also associated to the term "bug"

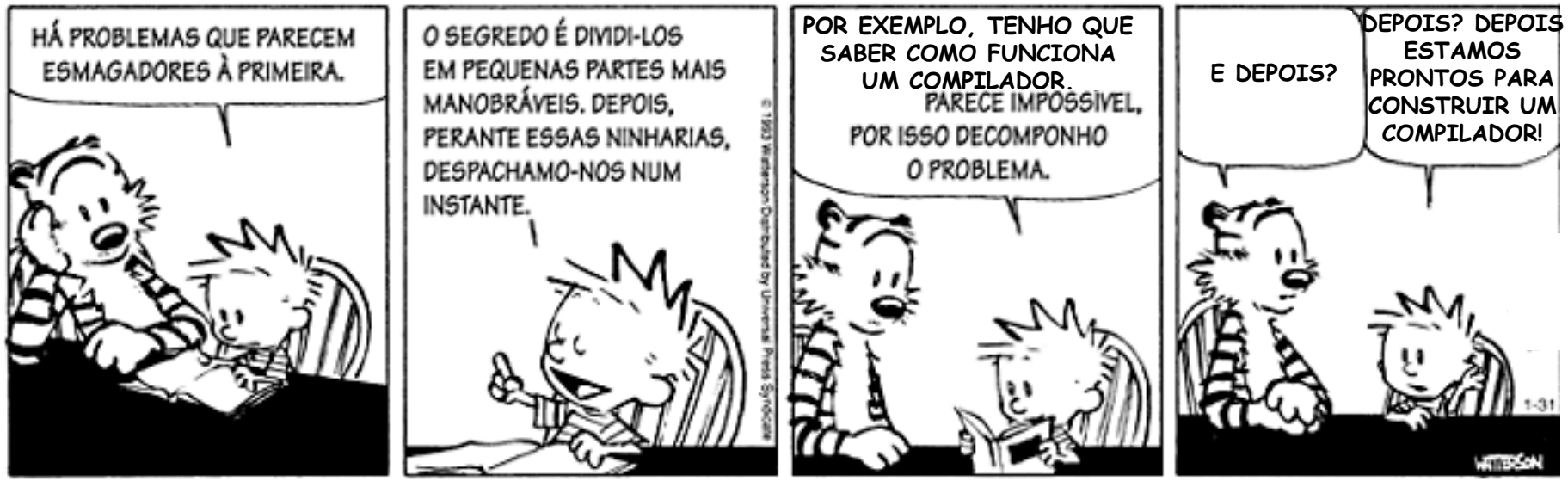


Programming

- Two programmers wiring the right side of the ENIAC with a new program



Source: Actually, Turing Did Not Invent the Computer
By Thomas Haigh
Communications of the ACM, Vol. 57 No. 1, 2014, pp. 36-41



YOU WE WISH A SUCCESSFULL WORK!