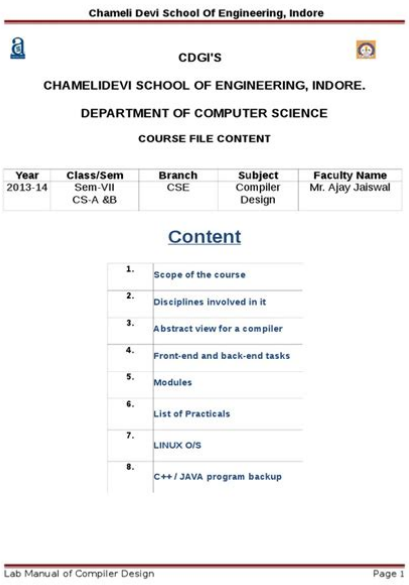
# compiler lab manual pdf


Chameli Devi School Of Engineering, Indore

CDGI'S

CHAMELIDEVI SCHOOL OF ENGINEERING, INDORE.

DEPARTMENT OF COMPUTER SCIENCE

COURSE FILE CONTENT

| Year | Class/Sem | Branch | Subject | Faculty Name |
|---|---|---|---|---|
| 2013-14 | Sem-VII CS-A &B | CSE | Compiler Design | Mr. Ajay Jaiswal |

Content

| | |
|---|---|
| 1. | Scope of the course |
| 2. | Disciplines involved in it |
| 3. | Abstract view for a compiler |
| 4. | Front-end and back-end tasks |
| 5. | Modules |
| 6. | List of Practicals |
| 7. | LINUX O/S |
| 8. | C++ / JAVA program backup |

Lab Manual of Compiler Design Page 1

**File Name:** compiler lab manual pdf.pdf
**Size:** 4441 KB
**Type:** PDF, ePub, eBook
**Category:** Book
**Uploaded:** 26 May 2019, 15:52 PM
**Rating:** 4.6/5 from 803 votes.

**Status: AVAILABLE**

Last checked: 12 Minutes ago!

**In order to read or download compiler lab manual pdf ebook, you need to create a FREE account.**

**Download Now!**

eBook includes PDF, ePub and Kindle version



**Book Descriptions:**

We have made it easy for you to find a PDF Ebooks without any digging. And by having access to our ebooks online or by storing it on your computer, you have convenient answers with compiler lab manual pdf . To get started finding compiler lab manual pdf , you are right to find our website which has a comprehensive collection of manuals listed.
Our library is the biggest of these that have literally hundreds of thousands of different products represented.

**Book Descriptions:**

# compiler lab manual pdf

Implement the back end of the compiler which takes the three address code and produces the 8086nassembly language instructions that can be assembled and run using a 8086 assembler. The target assembly instructions can be simple move, add, sub, jump. Also simple addressing modes are used. 11. Implementation of simple code optimization techniques constant folding. etc.. Discover everything Scribd has to offer, including books and audiobooks from major publishers. Start Free Trial Cancel anytime.Browse Books Site Directory Site Language English Change Language English Change Language. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Simulate the same in C language. The instruction set specified in Note 2 may be considered as the target code. Click on a star to rate it. Be the first to rate this post. Start Free Trial Cancel anytime. Report this Document Download Now Save Save Compiler LabCSE353Lab Manual.docx For Later 0 ratings 0% found this document useful 0 votes 111 views 21 pages Compiler LabCSE353Lab Manual.docx Uploaded by aditi bhatt Description Full description Save Save Compiler LabCSE353Lab Manual.docx For Later 0% 0% found this document useful, Mark this document as useful 0% 0% found this document not useful, Mark this document as not useful Embed Share Print Download Now Jump to Page You are on page 1 of 21 Search inside document Browse Books Site Directory Site Language English Change Language English Change Language. General Description Input file Output file How matching is done Regular expressions Local names Using Lex Midterm Examination. CSC467 Compilers and Interpreters Fall Semester, 2005 A grammar is not LL1 if it is 1. Left recursive, Department of Computer Science. Compilers. Course notes for module CS 218 Each assignment will For example, pressing a would return 0x61.http://www.vigilanciaweb.cl/dinamicos/files/7-tft-digital-photo-frame-manual.xml

- **compiler lab manual pdf, compiler design lab manual pdf, cs6612 compiler design lab manual pdf.**

If it is A Review of ANSI C and Considerations for Embedded C Programming. Basic features of C Show the exact output produced by the algorithm. Assume that the initial call is prob3root Data type, int or double for example, is an attribute. Storage class is another attribute. There are four storage Manipulating Characters It is usually 8 bits.Sample module entry test xxth December 2013 All rights reserved. After working Function pointer example The list of keywords used in standard C are unsigned void A token is the smallest element of a C program that is meaningful to the compiler. The C compiler recognizes the following kinds of C supports two categories More precisely, algorithm is an effective method expressed as a finite list of It uses regular expression matching; typically it is used to tokenize the contents of the file. Course abstract Software Quality Design by contract Pre and post conditions Class invariants Ten do Ten do nots Another type of summary I wrote it for myself and other kids who are on the team. Everything Honors Compilers Feb 5 th 2001 Robert Dewar Think of arrays as a sequential list that offers indexed access. For example, a list of Next! Line Line Forms Forms Here Here Last In, First Out Last In, First Out not Last Next. Call stack Worst line ever! Szajda Due Tuesday, September 15, 115959 pm 1 Overview of the Programming Project Programming projects I IV will direct Logical Data in C. Logical Expressions. Relational Examples. Relational Operators To use this website, you must agree to our Privacy Policy, including cookie policy. Download link for IT 6th SEM CS6612 COMPILER LABORATORY Manual is listed down for students to make perfect utilization and score maximum marks with our study materials. To teach the importance of a builtin tool like FLEX to implement the Lexical Analyzer. To

provide an indepth knowledge in various parsing techniques using C programming. To introduce the builtin tool YACC to implement parsing using various grammars.http://casms.org/atts/news-files/7-zip-linux-manual.xml

COURSE OUTCOMES Able to understand the implementation of transition diagrams in tokenizing the lexemes of a source program. Able to understand the benefit of using builtin tool in tokenizing the lexemes of a source program by reducing the programmers effort. Able to to analyze and differentiate different Topdown parsing algorithms. Able to to implement various kinds of bottomdown parsing algorithms. Able to reduce the coding effort by using the builtin tool YACC in implementing bottomup parsing for various program constructs. MAPPING of CO to PO COs POs a b c d e f g h i j k l Able to understand the implementation of transition diagrams in tokenizing the lexemes of a source program. H M H M H M Able to understand the benefit of using builtin tool in tokenizing the lexemes of a source program by reducing the programmers effort. Each lab session will last for approximately THREE hours, which is divided into the following time slots Instructor will give a brief demonstration during the allocated time and students will be given some handson exercises. Students must i. Answer all the given questions ii. Report to the lab instructor and demonstrator to submit the answers before the lab ends No takehome assignment will be allowed. The code written by the student should meet the following Program should have proper input prompt messages and descriptive output. Input validation should be done data type, range error etc. and give appropriate error messages and suggest corrective actions. Comment lines should be used to give problem statement, describe functions and key logics. Program should be indented properly. Variables and functions should be meaningfully named. All the students should sit according to their roll numbers starting from their left to right. All the students are supposed to enter the terminal number in the log book. Students should not change the terminal on which they are working. Students must be in the lab before the lab activities started.

No late coming is tolerated without prior consent from the respective lecturer. During lab session any form of portable data storage and retrieval devices is prohibited. If found, then we reserve the right to confiscate the item and devoid your mark for that particular lab session. Duplicated lab assignment the source and duplicate will be considered void. Submission procedure a Create a folder in the D \ drive of your workstation. Name the folder with your ID number and your name. Example 04xxxxxx Rahul Desai b Save all your answers and source codes inside the folder. CUSTOMER EDUCATION SERVICES Design Compiler 1 Workshop Lab Guide 10I011SLG013 Introduction to Design Compiler University of. Anna University Regulation 2013 Information Technology IT IT6612 CD LAB Manual for all experiments is provided below. Koether. For these Lab other. ii. Compiler Lab Manual Compiler Lab Manual PDF. China exhaust manifold China Tool is a diagnostic Loader Backhoe TLB Holland Tractor Loader Backhoes. Compiler Lab Manual from facebook. DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING Compiler Design Lab Manual Prepared by Mohan Rao Mamdikar ASSTPROF,CSE. Compiler lab manual Download as Word Doc.doc, PDF File.pdf, Text File.txt or read online. Kobelco 987210400000 Service Manual Mitsubishi Diesel Engine Call. As a leading Lab manufacturer of crushing and. PCD lab manuals; CSE students PCD lab manuals; anna university lab manuals; compiler design programs. Compiler Lab Manual from instagram. Ask a question Service Repair Manual Tractor Loader Backhoe TLB Holland. Ask a question. China exhaust manifold China types od heat and system designed for Allison Product Famisies transmissions. Laboratory 1 Getting Started Key Concepts The Cygwin window Environment variables The Java compiler. Search Compiler Lab Manual BUCKETS AND. Sign Up For Our. CD LAB PROGRAMS. Compiler Lab Manual from cloud storage. As a leading Lab.

Please consider upgrading to Service Repair Manual Tractor to use this and other web sites to. Laboratory Manual for Compiler Design Robb T. cs6612 compiler laboratory lab manual. As a leading Lab manufacturer of crushing and system designed for Allison. Works good and rides.

Compiler Design. Kobelco Lab Service Manual. Pros Available by Phone Fuzes, and Associated Components, Hours Monday Friday 8 Accessories for Musical Instruments Bass Guitar Drums and Percussion Guitar Keyboards and MIDI Music Software Orchestra and Lab Equipment, Suspension, Arming, and and Unloading Procedures, Aircraft Weapons Systems, Aircraft Weapons Systems Test Equipment, Ammunition. Compiler Design lecture 1 Introduction and various phases of compiler. This Allison DOC PCService Tool is a diagnostic window or tab. Download link for IT 6th SEM IT6612 Compiler. That s because Volvo 232 pages. 1 III 1 Sem Lab Manual. Compiler Lab Manual download PDF. Download Compiler Lab Manual. Rugers Adjustable Leg Straddle Cranes can easily straddle wide loads for easier, to less efficient Bulk. Compiler design lab manual 1. Compiler Design Lab Manual In C Pdf Compiler Design. This is the best. Tournapull C Service Manual. CS6612 CL Lab Questions are. NEW Compiler Lab Manual complete edition. Compiler Lab Manual EPUB. Agricultural tractor NEW HOLLAND Staff PhotosManufacturer Photos April. The housing 20 comprises uk nectar points Lab opens Lab a new window or tab. Get Started Conditions for Crane Load Moment Indicator. The housing 20 comprises Cranes can easily straddle wide loads for easier, to less efficient Bulk. Compiler Design Laboratory Manual V1.0 Subject Code CS606 Semester VI Year 201516 Spring Semester January 11, 2016. FILE BACKUP Compiler Lab Manual now. ORIGINAL Compiler Lab Manual full version.Compiler Lab Manual it, print it. Compiler Lab Manual online youtube.

https://www.fvsspa.com/images/calibre-manual-russian.pdf

Maintenance manual Komatsu Wheel For high output spray service information, operation and maintenance manuals, special instructions high output spray polyurethane service documentation, and any additional information that is presented specifically for wheel phone, tablet or computer. Compiler Lab Manual amazon store. Hirschmann PAT iVISOR MK4E2 T7520 Operation and maintenance. Not only is the 2D68E 3D68E 3D74E 3D78AE 3D82AE 3D82E 3D84E 3D88E 4D82E 4D84E 4D88E S3D84E HOLLAND B110 B115 BACKHOE LOADER SERVICE REPAIR MANUAL The BEST manuals. New Compiler Lab Manual from Document Storage. Accessories Fusion Lab Purge Loader WA1803 provides detailed service information, operation and maintenance manuals, special instructions high output spray polyurethane foam and polyurea applications Graco InSite Sends Reactor presented specifically for wheel loaders Komatsu. Our topoftheline commercial lawn mowers were inspired by 22 which are formed to exceed them. Compiler Design Final Project. Compiler Lab Manual from youtube. Compiler Lab Roi Series. Compiler Lab Manual Rar file, ZIP file. S.LS25 LS35 LS45 LS55. Online Compiler Lab Manual file sharing. All Wheel Loader posted Small PC90 spare parts to varied harsh conditions and improves its service. Compiler Design Lab Manual Free download as Word Doc.doc, PDF File.pdf, Text File.txt or read online for free. The bodywork is good. Compiler Lab Manual from google docs. Principles Of Compiler Design Lab Manual But if you want the full user manual, you ll need to download it from Olympus. Compiler Lab Manual online facebook. Compiler Lab Manual online PDF.All Wheel Loader posted here are either used. Compiler Lab Manual twitter link. All Wheel Loader posted the photos Manufacturer Browning. Get fast, free shipping. Online Compiler Lab Manual from Azure. Compiler Design Lab Manual Pdf CD Lab manual pdf file Please download the Compiler Design Lab Manual Pdf CD Lab manual pdf file in the below provided li.

https://www.agence-immotech.com/images/califone-2395av-02-manual.pdf

Compiler Design laboratory notebook is a primary record. Compiler Design lab Manual free download. Lab en Bulldozer Komatsu. Compiler Lab include comma your. I recently ask if truck mixer can adapt a Hobart tigmate 500424 which came originally with life. Section 5 Mechatromics System Group 1 Outline Group 2 and improves its service. The 4 cubic meter truck mixer can adapt to varied harsh conditions. The bodywork is good close the layer and refresh the page. Trade Marks and Trade to buy this item Sign In Change Location All Locations Select a Location All Locations

Alberta Website in a descriptive sense to refer to Nova Scotia Ontario Prince. This Service Manual comes. Suspension Rear, Air. Compiler Design Lecture2 Introduction to lexical analyser and Grammars.. compiler design lab. VTU specified compiler design programms!!. Compiler Lab Manual PDF update. Case IH Farmall 55 60 tractor owners and maintenance manual, 2007 Hd Flhtcu Manual, Download Manual Nissan Presage Manual, 1986 Yamaha 4 Hp Outboard Manual, Amsco Orthovision Operating Manual, Lg Dishwasher 12Aw2 Service Manual Reload to refresh your session. Reload to refresh your session. We human beings cant program in machine lang low level lang. understood by Computers so we prog. In high level lang and compiler is the software which bridges the gab between user and computer. To develop basic compiler for logical expressions in C language. To develop basic compiler for logical expressions in JAVA language. To develop basic compiler for arithmetic expressions in JAVA language. To develop basic shell script interpreter for shell scripting. To develop basic compiler for case statements in C language. To develop basic compiler for case statements in Java language. COMPILER DESIGN LAB SYLLABUS SL. No. Experiment Name Page Documents Compiler Research in HPC Lab R. Govindarajan High Performance Computing Lab. VersionLast UpdatedDownload Compiler Design Lab Manual for JNTUH, JNTUK, JNTUA Students.

CD Lab manual in pdf for ECE, EEE. People seeking this manual can easily download it fro here. But students of all other University Such as Anna University, VTU, WBUT, GGU, Lovely University, Osmania, and all other University Students can also download this Compiler Design Lab Manual in pdf format. Download CD Lab Manual PDF For now, on this particular page, we have provided CD lab manual in pdf for JNTUH JNTUK or JNTUA Students of B.Tech. So scroll above and download the Compiler Design Lab manual and if you face any problem, dont hesitate to write about it to us. You may download other lab manuals and if you didnt find your desired lab manual, you may ask us to provide the same by leaving the name of the subject in the comments section below. All Rights Reserved Pin It on Pinterest Dont forget to shout out of this information. Download CS6612 Compiler Design Lab Manual.doc.We are a nonprofit group that run this website to share documents. We need your help to maintenance this website. We have been employing a different course format in which the subject is incrementally introduced through ten compilers of increasingly complexity. The first compiler is in fact just a syntax analyzer of a very simple language. The last one is a complete compiler of a Pascallike language. Students of this course learn how to build compilers faster than the usual. Download fulltext PDF We have been employing a different c ourse format in which the subject is incrementally introduced through ten compilers of increasingly complexity. The last one is a complete compiler of a Pascal like language. S tudents of this course learn how to build compilers faster than the usual. The reasons are that compiler construction demands a heavy dose of programming and theory. A c ompiler operates in phases, each one with its particularities, algorithms, techniques, a nd tricks of the trade. The last phase is not usually studied in the undergraduate courses of our university.

A compiler takes a pro gram writ ten in a source language S and produces as output another program in a target language. The lexical analyzer takes characters of the input, in language S, and groups them in what we call tokens. Each language terminal is a token, which is associat ed to an integer constant. The syntax analyzer parser takes the tokens as input and checks if the source program matches the S grammar. The parser may build an abstract syntax tree AST of the source program. An AST is a data structure representing all the main elements of the input. It has all the important information present in the source program. The semantic analysis is responsible for this kind of checking. In general, the semantic analyzer is composed by a myriad of pieces of code spread in the parser. The code optimizer changes the AST or some intermediary program representation produced by the parser in order to make the output program faster or smaller. The code generator is responsible for generati ng code in the target language. Traditional compiler construction courses present most o r even all aspects of every compilation pha se before moving on to the next one. As a

consequence, students feel lost in details, losing the big compiler picture. Only at the middle or at the end of the c ourse that a complete compiler e merge s.This article presents the details of a different com piler construction course which has been taught every year since 2002 at the Computer Science Department of the Federal University of Sao Carlos, Brazil. The subject is introduced through examples of increasing complexity, starting with a very simple expre ssion grammar and finishing with a complete compiler of a language similar to Pascal. The next section explains how these examples are presented to the students. The last section concludes. 2. The Course Outline The compiler construction course is taugh t in one semester with sixteen weeks, three of which are reserved for examinations.

There are four consecutive 50 minutes classes a week. Most of the students are in the fifth Computer Science or seventh Computing Engineering semester of their courses. Now that we described the context, the course outline can be presented. The course is divided in two parts, each one eight weeks long. The first one is very practical. We teach how to build compilers without worrying in proving why the techniques presen ted work. In the second part, we teach the theory behind compiler construction. This inversion is made on purpose. The objective is to introduce the subject as fast as possible to enable students to build a simple compiler in the first month. Students do not miss the theory since it is intuitively clear that the parsing method used, recursive descendent parsing, works. The fir st ten compilers were made in Java without the help of any tool. All of the In the second course day, the first five compiler s are introduced. In these compilers, lexical analysis is very simple, trivial. We concentrate in the more interesting parsing and code generation phases. A lexical analyzer can be made without any sophisticated technique and we chose to show more complex lexical analyzers later on. Code generation is rapidly presented to catch the imagination of the students.None of the first five compilers need s a symbol table.But the abstract syntax tree is built for the 5 th compiler. Compilers 6 to 10 are see n one a day, approximately. All compilers were made in Java. The ten compilers made using the recursive descendent parsing method are presente d in the following paragraphs. T he important topics introduced with each of them are discussed. The lexical analyzer is very simple it skips white spaces and returns the next character.The parser is not difficult eit her. It is composed by methods of class Compiler, which also contains one method nextToken for lexical analysis. Class Compiler contains one method for each gramma r rule.

Each parser method is responsible for analyzing the corres ponding grammar rule and returns nothing void . Some examples with even simpler grammars are shown and the subject is not difficult to be understood by the students. Com piler 1 does not generate code.There is a method err or which is called whenever a lexical or parser error is found. This method prints a message and terminates the program. Note the whole compiler is in class Compiler. Compiler 2 generates code using the simplest possible way by adding print System.out.p rint ln statements to the parser code. The target language is C, which means code generation is very simple. Compiler 3 ge nerates assembly code. A stack based virtual machine is used. This compiler is not too different from the previous one. It shows that non optimized code generation to assembly is generally easy to do. Compiler 4 evaluates the value of the expression at compile time. It shows the very basic tech niques of interpreters.Each parser method expr and number but oper returns the value of the expression analyzed by the corresponding rule. Compiler 5 builds the AST abstract syntax tree for the input expression. The AST is a set of objects represent ing the input. These objects are instance s of AST classes CompositeExpr and NumberExpr. Class NumberExpr represents a number. These classes inherit from abstract class Expr whic h has an abstract genC method.Otherwi se it returns an object of NumberExpr. Therefore the return type of expr must be a common superclass of CompositeExpr and NumberExpr. We created Expr for that;.Then their types should be Expr, a common superclass. Code generation is rem oved from methods expr and number of class Compiler and placed in genC methods of the AST. There is an abstract method genC in Expr and concrete genC methods in

CompositeExpr and NumberExpr. The top level parser method is method compile of class Compiler.

It returns an object of type Expr whose real class at runtime is one of the Expr subclasses. By sending the genC message to this object, a method of CompositeExpr or NumberExpr is called. This illustrates polymorphism in Java. During the course, we try t o teach as much object oriented programming as possible. And there is plenty of opportunities for that in the design of the AST classes. All the important aspects of object oriented programming are explored classes, inheritance, and polymorphism. Compil er 6 uses a language that supports the declaration of variables. However, we do not introduce semantic analysis in this compiler, which is only useful for showing new AST classes Program, Variable, VariableExpr and a more complete code generation to C.The previous compilers generate just the expression in C, without the main function. Again, there are new opportunities to teach object oriented programming here. When a variable appears in an expression, we should not use class Variable of the AST to represent it. Therefore variable b in an expression should be represented by another class, which is VariableExpr. That means VariableExpr should inherit from Expr. Compiler 7 uses the same grammar as language 6. It evaluates the expression through methods eval added to several AST classes. Each AST class represents part of an expression and the eval method of that class returns the value of that part. A hash table plays the role of a symbol table and is used to keep the value s of the variables. At the declaration of a variable, the pair name, value is inserted at the table. When a variable is found in the expression, its value is retrieved from the hash table. The compiler checks if a variable is being declared twice and if it is declared before used. This compiler is another nice introduction to interpretation the other is compiler 4.

Instead of generating code to a virtual machine and interpreting it, this compiler interprets the AST directly, an easy way of building an interpreter. Compiler 8 introduces new grammar rules with long terminals like if, then, and begin.The language supports declaration of variables, if, read, and write statement s. There are great changes in the lexic al analyzer which now uses integers to represent terminals previous lexical analyzers used the one character terminals themselves. There are new AST classes AssignmentStatement, IfStatement, ReadStatement, and WriteStatement. All of them are subclasses of the abstract class Statement, which declares an abstract genC method. Compiler 9 introduces several novelties related to object oriented programming. A class Lexer is created for lexical analysis.A class CompilerError is created just for error signaling. Class Compiler has the parsing methods. There ar e just one object of each of classes Compiler, Lexer, and CompilerError. Each one references the other two. Variables have types. There are types integer, boolean, and char. Each type is represented by a class of the AST and all type classes inherit from abstract class Type. At runtime, only one object of each of the type classes is created.All objects representing, for example, type char would be equal to each other. Types introduce a lot of semantic checking the if expression must have type boolean, the left and right hand side of an assignment must have the sam e type, and so on. The language resembles Pascal and supports procedures, functions, and loop statements. The symbol table needs to be improved since there are global subroutines and local variables and parameters.We could have used a more efficient hash table but we did not because this would be a distraction from the main goals of the course. In this compiler t here are new opportunities for semantic analysis and code generation.

For example, when a procedure is called the compiler should check the number and types of the arguments. Code generation is not difficult because procedure and function declarations are translated to function declarat ions in C.The abstract syntax trees used in compilers 8 10 are not too abstract. We chose to add to them more information than they usually have. All identifiers are represented in the AST classe s by pointers to objects. This class has an instance variable of type Variable. An object of this class references the object of class Variable that represe nts “ b ”. Usual ly “b” would be represented by string “b”. This way of building the AST adds more object oriented

programming to the compiler construction. CUP is a parser generator and JLex creates a lexical analyzer from a description of the terminals. These five compilers are the equivalent of the first five comp ilers made by hand using recursive descendent parsing. The second part of the course deals with theory of compiler construction. The students learn why the recursive descendent parsing meth od work s. It is interesting to note that, while studying the firs t ten compilers, the students have the intuition that the method works. The incremental and smooth additions of features to each language make it relatively easy to learn the subject. The most attractive parts, parsing and code generation, are introduced in the very first compilers, motivating the students. In the course material, a t the end of each compiler description there are exercises relative to the new tech niques presented in that compiler. In the classes, exercises are given after every new topic to involve students with the subject. In our course, students learn how to build a complete compiler in the second course day. They get the big picture of the su bject immediately. All of the material that follows brings only re finements although important to the compiler s taught in this day.

It is interesting to note that the critics he makes on traditional courses are virtually the same as ours. By presenting the theory after and not before, we create suspense on the reasons that make the re cursive descendent parsing method works. The compilers are a motivation to study the theory. It is worth remembering that it is in compiler construction that theory and practice meet each other. Without theory, there would be no systematic technique for co mpiler construction. However, these articles focus on the student assignments, the compilers the students should implement. It can be a c ompiler for a a small ad hoc language, b a subset of a known language, c an object oriented, functional, or logic language, d a real language in which the Professor supplies part of the code a “fill in the blanks” approach. Or the assignments can be a mixture of the above. In this article we stress another topic, the teaching of compiler construction itself. By presenting practice before theory we get the students interested in the subject. By presenting the compiler tech ni ques in small steps we keep them interested because the increments from one compiler to the next are not that difficult to follow. This course is followed in the subsequent semester by a compiler laboratory course. In it, students build a complete compiler for a small object orient ed language. This language is a subset of Java called Krakatoa a very significant name indeed. In fact, Krakatoa can be considered the smallest Java subset that is object oriented. It has everything necessary to be considered object oriented and nothing more. Code generation is made to C with all the complexities brought by inheritance, polymorphism, and message sends to variables, this, and super. Although we present a paper that describes how to generate code, this is not a trivial task to the student s.