Yunlin Su
Song Y. Yan

Principles of Compilers

A New Approach to Compilers Including the Algebraic Method

编译原理

包含代数方法的新编译方法（英文版）

高等教育出版社
HIGHER EDUCATION PRESS

Authors
Prof. Yunlin Su
Head of Research Center of Information Technology Universitas Ma Chung
Villa Puncak Tidar No-01 Malang
Java Timur, Indonesia
E-mail: su. yunlin@ machung. ac. id
Department of Computer Science
Jinan University, Guangzhou 510632, China
E-mail: gxuwzsyl@ yahoo. com. cn

Prof. Song Y. Yan
Department of Mathematics
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge MA 02139, U. S. A.
E-mail: syan@ math. mit. edu

图书在版编目(CIP)数据

编译原理：包含代数方法的新编译方法：英文/苏运霖，颜松远著.
北京：高等教育出版社，2011.6
ISBN 978-7-04-030577-7
Ⅰ. ①编… Ⅱ. ①苏…②颜… Ⅲ. ①编译程序-程序设计-英文
Ⅳ. ①TP314
中国版本图书馆 CIP 数据核字(2011)第 058306 号

策划编辑 陈红英　　责任编辑 陈红英　　封面设计 张 楠
责任校对 胡晓琪　　责任印制 刘思涵

出版发行	高等教育出版社	咨询电话	400-810-0598
社　　址	北京市西城区德外大街 4 号	网　　址	http://www. hep. edu. cn
邮政编码	100120		http://www. hep. com. cn
印　　刷	北京中科印刷有限公司	网上订购	http://www. landraco. com
开　　本	787×1092　1/16		http://www. landraco. com. cn
印　　张	29.25	版　　次	2011 年 6 月第 1 版
字　　数	769 000	印　　次	2011 年 6 月第 1 次印刷
购书热线	010-58581118	定　　价	69.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

物 料 号　30577-00

Yunlin Su
Song Y. Yan

Principles of Compilers

A New Approach to Compilers
Including the Algebraic Method

Preface

The compiler is one of the most important aspects of system software. When any computer user develops a computer program, one must use some programming language, rather than using a computer instruction set. This implies that there must be the compiler of the programming language that has been installed on the computer one uses, and otherwise the developed program cannot be run.

There are some differences between a compiler and programming language. Once language is designed, it must be kept unchanged (except when it contains a mistake that has to be corrected), while the techniques for implementing compilation might be changed over time. Hence people always explore the more efficient and more advanced new techniques to raise the quality of compilers.

The course similar to "The principles of Compilers" has become one of the most important courses in computer science within higher institutes. According to our knowledge, the development of compilation techniques evolves in two directions. One is towards the improvement of the compilation techniques for existing languages. Another is towards the research and development of the compilation techniques of new languages. These new languages include object-oriented languages, distributed languages, parallel languages, etc. This book introduces the newest knowledge in the field, and explores the compilation techniques suitable for the languages and computation. It associates the compilation of programming languages with the translation of natural languages in human brains so that the reader can easier understand the principles of compilers. Meanwhile, it introduces the algebraic method of compilation that belongs to formal technology.

This book consists of 16 chapters. Chapter 1, Introduction, outlines the process of compilation and associates the compilation of programming languages with the comprehension and generation of natural languages in human brains. Chapter 2 introduces the grammar and language. The generation of the language is based on the grammar and languages are the fundamentals of the compilation process. Chapter 3 introduces finite automata and regular languages, together with Chapter 4, it is devoted to lexical analysis, the first task of analysis stage. Chapter 3 may be regarded as the theoretical preparation of lexical analysis; while Chapter 4 is the concrete practice of

lexical analysis. Chapters 5 – 7 commonly work together to discuss syntactical analysis. Chapter 5 introduces push-down automata that correspond to context-free grammars. Chapter 6 devotes to the discussion of context-free grammars and the context-free languages which they generate. Chapter 7 explores the second task of analytical stage — syntactical analysis. Following this is the semantic analysis. After the analytical stage finishes, the synthetic stage starts. The main task of the synthetic stage is to generate object code. Chapter 8 introduces and analyzes attribute grammars. Chapter 9 introduces a new compilation method — the formal method of compilation. Chapter 10 discusses the generation of the intermediate code. Chapter 11 expatiates the debugging and optimization techniques for compilers. Chapter 12 explicates the memory management that is related to compilation of programs. Chapter 13 is the destination of the compilation, the generation of object code. The chapter introduces the virtual machine MMIX that is proposed by D.E. Knuth in his book *The Art of Computer Programming*. This virtual machine is the mixture of features of 14 most popular machines in the current market, it has rich an instruction set, and makes object codes flexible. Chapters 14 and 15 expound the compilation techniques for object-oriented programming languages and parallel programming languages. Chapter 16 discusses issues for grid computing. Though grid computing has attracted one's attention there is no any language especially suitable for grid computing at the present. Hence, we just focus on its features, pointing out the issues which the compilation of the language should be tackled when the language exists.

We would like to express our sincere appreciation to Ms. Chen Hongying of Higher Education Press. Without her encouragement, help and patience, we could not finish the writing of this book. We also want to thank the authors whose contributions were referred to the book. A great part of the contents of the book is taken from them. We would like to acknowledge Tim Lammertink and Myrte de Vos for their kind help. Finally, we would like to express our gratitude to our family and students for their long-term support and understanding.

No doubt, there might be neglects or mistakes remaining in the book. We hope that the reader would be generous with your criticism.

Yunlin Su
Song Y. Yan
March 2011

Contents

Chapter 1 Introduction

Language allows us to know how octopuses make love and how to remove cherry stains and why Tad was heartbroken, and whether the Red Sox will win the World Series without great relief pitcher and how to build an atom bomb in your basement and how Catherine the Great died, among other things.

Steve Pinker

1.1 Language and Mankind

If you read the text above, you must be engaging in one of the mind's most enchanting process—the way one mind influences another through language. However, we put a precondition on it that you have to know English, otherwise the text has no influence at all to you. There are so many languages in the world that even no one can exactly tell how many there are. Therefore, there is the need of a bridge that connects different languages so that people can understand each other. The bridge is the translation. And the subject of the book is the translation between the formal language and the machine language, or compilation.

What is the compiler or the compilation program? Simply speaking, it is a program of which the function is to translate programs written in a programming language into machine codes that are to be run by the same kind of machine the codes belong to. In order to explain things behind this, we need to discuss it further.

Language is main means of human communication and the way in which most information is exchanged. By language, people link up each other, they express their attentions and feelings, and they describe matters or express their understanding [1]. It is one of the kinds of intelligence or the product of intelligence. However, in the long process of human evolution, there was a long period without language. Gradually, they invented oral language to meet the need of living. Therefore, oral language can be considered as the first breakthrough in language, it was also a breakthrough in human civilization. From oral language to written language, it underwent even longer time. The

occurrence of written language represented a more significant breakthrough of human being in terms of languages. Human thinking and problem solving can be conceptualized as processes involving languages. Many, if not most or all, forms of thinking and problem solving are internal, that is, done in the absence of external stimuli. Abstraction of puzzles, for example, into verbal symbols provides a way to think about a solution. It is not difficult to imagine that without language the process of thinking cannot be completed, continued and deepened as if there is no language one simply cannot express his/her ideas to other. When one wants to reminisce, he/she is unable to describe the process that involves many objects and complicated plots. Written language is more powerful than oral language. It not only can link up people at the contemporary era, but also it can link up the present time and the ancient time so that people at the present time can also know things that took place in ancient period. By using written language, people not only can communicate with people in the vicinity, but also contact people at long distance. Especially with the modern communication tools, e.g., computer networks, televisions, and telephones, people may communicate with each other even quicker, more convenient and may make sure the security and secrecy of information. That means that written languages change the extent of time and space of communication of people.

The civilizations of the human being are divided into many branches. Each one is symbolized by different language. Each race or nation formed each own language due to the difference of living locations and evolution conditions. In history, there were several thousands languages. As time passed, many languages, especially the oral languages, that were used only by few people had extinguished. Until now there are still some languages that have only oral versions and have no corresponding written versions. Therefore, the languages that have real impacts and are used by the great throng of peoples are not too many. However, people who use these languages want to share the civilization; they want to cooperate with each other or to do business. Obviously, each language is so different from others that unless one has learnt it otherwise one has no way to understand, and vice versa. Hence, if two different language speakers want to converse with each other, they need a bridge to link them. It is the translation. Its task is to translate a language spoken by A to another language spoken by B and to translate the language spoken by B to a language spoken by A. It is not only necessary to translate the oral language (the translator of colloquial languages is a called interpreter) but also necessary, or even more important to translate the written languages including the works in social science, natural science, novels, etc. Without the translations, people speaking different languages cannot converse, communicate, and exchange their thinking or discoveries. In this sense, we may say that the world is small but the number of languages in the world is far too many.

Today as the rapid development of science and technology and the inevitable tendency of economy globalization happening in almost every country around the world, language translation including colloquial and literal,

has become a heated profession. Take as an example for the colloquial translation or interpretation, it involves three persons, i.e., two, A and B. who want to converse with each other for some purpose, and one, C, who helps them with the thing. Suppose that A speaks the language X and B speaks the language Y. Obviously, in order for A and B understanding each other the task of C is to interpret the words of X spoken by A into language Y, meanwhile, he interprets the words of B spoken in Y spoken by B into language X. Therefore, C must be a bilingual in this circumstance. And this situation is shown in Fig. 1.1.

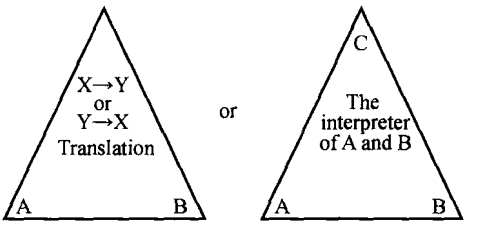

Fig. 1.1 Oral translation.

The role of C sometime needs to be done by two persons, say C or D, is in charge of the interpretation from X to Y or from Y to X.

While in the case of a literal translation, the translator mainly translates a foreign language into his native language for the native readers, or he translates his native language into foreign language to serve the foreign readers. No matter which case it is, the purpose is the same, i.e., to make the listener or reader understanding each other.

1.2 Language and Computer

It is well known that computer is one of the greatest inventions of mankind in the last century. It embodies the newest development of mankind's science and technology. Computer relies on its running to solve problems set by people while the running relies on the program that is composed of a sequence of the instructions in advance from the instruction set of the machine. The instruction set that is used to develop programs can be considered as the language of computer. It acts to follow the sequence of the instructions as if it speaks the language that consists of the sequence. This kind of language consists of the sequence of 0 s and 1 s.

Hence in order to make the computer running and working for peoples, one should develop the program with the purpose of solving intended problem. For doing so one needs to master the instruction set. We do not say that people cannot master the instruction set and develop programs by using it. However, it is really very tedious and cumbersome to "speak" the language to

computer; especially it is too much for the common users of the computers. It is something like that one is required to understand the principles of the television and operate the existing components of the television if one wants to watch the television.

When the computer was just invented, there was not any other language to use for running the computer. The instruction set of computer was the unique language which people may use to develop programs. The historical period is called the period of manually programming. The instruction commonly contains the operation code that indicates the operation it performs, the addresses of the data which the operation performs as the control codes. At that time, only very few people were the designers or developers of the computers. For them to build the programs using the instruction set was not a problem though it also entailed them to work hard and spend lots of time. As computers became more and more popular, the users were no longer those who are very familiar with the principles inside the computers, they are just the real user, no different from the users of televisions. They want to freely use the computer to solve their varieties of problems. In this circumstance, no doubt, the machine language became their stumbling block in using computers [2]. In order to break away from the constraints of the machine language, from soon after the invention of computer, people had started searching the solution to the problem. The first step was to replace the operation codes of the instructions to the notations easily mnemonic, e.g., to use ADD to represent addition, SUB to represent subtract, MUL to represent multiplication, and DIV to represent division, or more simply, just to use $+$, $-$, $\times$, and / (or $\div$) to represent the arithmetic operators. Then, they used symbolic addresses to take the place of real binary addresses, etc. Of course the language transformed in this way is no longer computer language, or original computer instruction set, although it basically corresponds to the computer instruction set, and they are completely equivalent. This was the first step that people broke away from computer language. Though it was a minor step, it was crucial. It indicates that people may not be confined by the computer instruction set, they may use more convenient language to develop programs for computers. This kind of languages is called assembly languages. Here the module given above was still suitable. As shown in Fig. 1.2, the left side of the bottom edge of the triangle represents any program written in assembly language which we call the source program, and the right side of the bottom edge is totally equivalent program written in a computer instruction set which was produced by the assembler on the top of the triangle and has the name of the object program or target program. And the assembler plays the role of the compiler of which the duty is to translate the source program into the executable object program written in machine language. Therefore, the assembler must also be executable on computer and by its operation it produces the object program as its output.

Hence the assembler is the early version of the compilers. As the language which source programs used was assembly language, it was only the

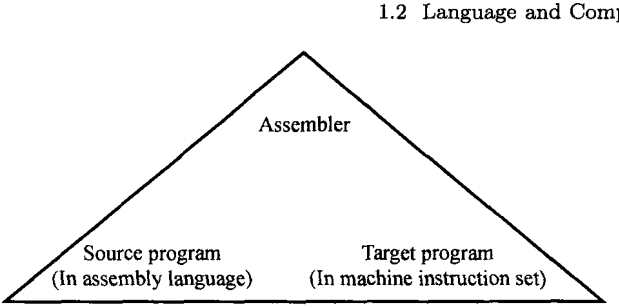

Fig. 1.2 Translation of computer assembly language.

simple adaptation of the machine instruction set (e.g., the operation code was the mnemonic code of the original one). Hence it is also called low-level language. Here the word low means that it is machine-oriented (low-level) and isn't mankind-oriented (high-level). Assembler is also a low-level form of the compilers as it hasn't used much the compilation principles which we used in the compilers for high-level programming languages.

After the success of assembly languages and their assemblers, people started the design of the mankind-oriented high-level programming languages. The common feature of these languages is that they broke away from the restriction of the computer instruction set. They adopted a subset of the commonly used language (in general it is English) and established the grammar to describe the statements or elements which people used to develop the programs. These languages are called procedure-oriented languages or simply procedural languages, or imperative languages. The earliest programming languages include FORTRAN (stands for FORmula TRANslation, it was first designed as early as 1954 [3]), ALGOL 60 [4], COBOL (stands for Common Business Oriented Language, it was first designed in 1959, and its success was strongly influenced by the United States Department of Defense). In terms of the occurrence of the programming languages, the 1960s was stormy. It was said that at that period over two thousand languages were developed, but only thirteen of them ever became significant either in terms of concept or usage. Among them, APL (stands for A Programming Language, developed by Dr. Kenneth Iverson at IBM [5]) is an interactive language. It devises a powerful yet compact notation for computation which incorporated many concepts from mathematics. PL/1 (stands for Programming Language/1) is suitable for scientific computation. With 1 in the name it probably intends to be number one in terms of its great deal of functionality. LISP (stands for List Processing, developed by McCarthy and his co-workers to design a conceptually simple language for handling symbolic expressions with its domain being artificial intelligence) [6]. PROLOG (stands for Programming for Logic) is another effort for use in artificial intelligence. SNOBOL (developed in the mid-1960s at Bell Telephone Laboratory [7]) is a language whose main strength is in processing string data. As the name SIMULA67 indicated that SIMULA was designed in 1967 and had simulation as its major appli-

cation domain. And it was later refined in CLU, Euclid, and MODULA [8]. GPSS or SIMSCRIPT [9] provided the example that conventional programming languages can and have been augmented so that simulations can be easily described. The later development of the programming languages was the coming of the general-purpose language called ADA [10] in honor of Ada Augusta, Countess of Lovelace, the daughter of the famous poet Byron. She collaborated with Charles Babbage (1792–1871) who between 1820 and 1850 designed two machines for computation. One relied on the theory of finite difference and so he called it Difference Engine. The other embodied many of the principles of a modern digital computer and he called this Analytical Engine. Ada, as the collaborator of Charles Babbage, helped him with developing programs for the analytical engine. Therefore she has recently been recognized as the first programmer. The other language that later became very popular is C [11]. It initially was used for writing the kernel of the operating system UNIX.

Apart from few (if any) languages the languages aforementioned basically all are procedure-oriented languages. After the software crisis that took place in the late 1960s, the structured programming method was proposed, and it hastened parturition of Pascal (in honor of French mathematician Pascal, designed by Swiss computer scientist Niklaus Wirth [12]). Another methodology that was proposed to solve the software crisis is the object-oriented software design method, and it caused the production of the object-oriented languages. For example, based upon the C language, C++ was developed. Soon after it Java was also designed based upon C. In addition, SMALLTALK [13] is also of this kind.

As hardware unceasingly develops it also puts forward the new requirements to software. New computer architectures like distributed systems, parallel computer systems, computer networks, etc. all propose new requirements and challenges to computer programming. New languages that meet these needs sequentially come out.

No matter how the languages change, there is one thing unchanged that the source programs written in these languages must be compiled first before they become executable object programs on computers. That is to say that they obey the module as shown in Fig. 1.3.

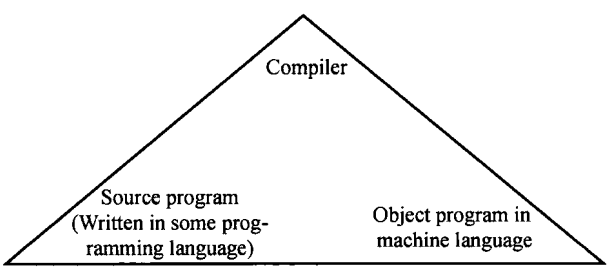

Fig. 1.3 A process of compilation.