计算机专业英语文选

下　册

南京大学外文系公共英语教研室编

商 务 印 书 馆

计算机专业英语文选

下　册

南京大学外文系
公共英语教研室编

商　务　印　书　馆
1985年·北京

计算机专业英语文选

下　册

南京大学外文系公共英语教研室编

商　务　印　书　馆　出　版
（北京王府井大街36号）
新华书店北京发行所发行
香　河　安　平　印　刷　厂　印　刷

787×1092毫米 1/32　5 3/8印张　131 千字
1979年6月第1版　　1985年2月第2次印刷
印数：-33,100-册
统一书号：9017·853　　定价：0.78 元

CONTENTS

1. TERMINOLOGY

A translator is a software system that transforms the statements of one computer language into statements in some other computer language.① The first language is usually called the source language; the second language may be called the object language, target language, machine language, or some other descriptive name.

A computer is a translator that transforms a high-level (or problem-oriented) language such as ALGOL or PL/I into a low-level language such as assembly language or machine language. An incremental compiler translates statements of a language, one statement at a time.② This is in contrast to the usual case where the whole program is input to the compiler, which then makes several passes before emitting code.③ Incremental compilers are generally used in interactive or terminal-oriented environments wherein the user interests with the computer directly via a keyboard, console, display unit, or other such device.④

An interpreter is a program that instead of translating source code executes it directly by determining the meaning of each statement or part of a statement as it is encountered and by computing its value or generating its effect (e.g., input/output).⑤

The syntax of a language is the set of formal rules which describes its form:⑥ how statements may be formed, which constructs are legal, the order in which statements must occur,⑦ and so forth.

The semantics of a statement is its meaning, including what values are to be computed, the code necessary to compute them, and any side effects that are to occur, such as input, output, and setting internal switches.⑧

A single formal rule of a syntax is called a production.

A grammer is the set of productions describing some language plus the set of symbols used in the productions.⑨

Parsing is the process of determining which productions are used in the construction of a sentence of a language; a program that performs this process is called a parser. More formal definitions of these concepts may be found in Ginsburg, The Mathematical Theory of Context-Free Languages.

A compiler-compiler is a translator to which input is a source language describing a compiler for some language⑩ and which as output gives a program that is a compiler for that language.⑪ A meta compiler is a translator that translates high-level language to some other language (also possibly high-level). The difference between compiler-compilers and meta compilers is not very distinct, nor is this terminology adhered to strictly.⑫

词　汇

terminology [ˌtəːmiˈnɔlədʒi] *n.* 术语
target [ˈtɑːgit] *n.* 目标；指标
descriptive [disˈkriptiv] *a.* 描述的；说明的
incremental [ˌinkriˈmentl] *a.* 增量的
wherein [weəˈrin] *ad.* 在那里；在那方面
console [ˈkɔnsəul] *n.* 控制台
legal [ˈliːgəl] *a.* 合法的
semantics [siˈmæntiks] *n.* 语义
plus [plʌs] *prep.* 加，加上
parse [pɑːz] *vt.* 句法分析
parser [ˈpɑːzə] *n.* 句法分析程序
Ginsburg（人名）
theory [ˈθiəri] *n.* 理论
meta [ˈmetə] 元（构词成份）
strictly [ˈstriktli] *ad.* 严格地

短　语

in contrast to　与…对比；与…对照　　*and so forth*　等等
instead of　而不…；代替　　*(to) adhere to*　坚持

注　释

① 前置词短语 in some other computer language 作定语，修饰 statements。in 在这里是"用、以"的意思，例如 in English（用或以英语）。

② 名词短语 one statement at a time 作状语，修饰句中的谓语动词 translates。

③ 在本句中，从 where 到结束是定语从句，修饰 the usual case。其中又包含一句非限制性定语从句：从 which 到结束，修饰 the computer。

④ wherein the user ... or other such device 是由关系副词 wherein 引出的定语从句，修饰 environments。

⑤ 在本句中，从 that 到结束是定语从句，修饰 a program。其中，关系代词 that 是主语，谓语动词是 executes，前置词短语 instead of translating source code 作状语，修饰 executes，it 指 source code，两个前置词短语 by determining ... as it is encountered 和 by computing ... (e.g., input/output) 均作状语，也修饰 executes，在第一个前置词短语中，as it is encountered 是时间状语从句，修饰动名词 determining，it 指 source code。

⑥ its form = the form of the language。

⑦ 名词从句 how statements may be formed 和 which constructs are legal 以及名词短语 the order ... occur 均是 the set of formal rules 的同位语。名词短语 the order ... occur 又含有一个定语从句 in which statements must occur，修饰 the order。

⑧ 现在分词短语 including ... internal switches 作定语，修饰 its meaning。including 有三个并列的宾语：1) what values are to be computed；2) 名词短语 the code...them，其中 necessary to compute them 是形容词短语，修饰 code，them 指 values；3) 名词短语 any side effects ... switches，其中 that are to occur 是定语从句，修饰 any side effects，such as input，output 和 setting internal switches 是进一步说明 any side effects 的，与它同位。

⑨ plus ... productions 是前置词短语，修饰前面的 the set of productions。其中 used in the productions 是过去分词短语，修饰 the set of sym-

bols。

⑩ 前置词短语 for some language 作定语，修饰 a compiler。

⑪ 定语从句 which as output ... language 修饰前面主句中的表语 a translator。

⑫ nor is this terminology adhered to strictly 是倒装句，原因是句首用了 nor。这句等于：the terminology is not adhered to strictly either。另外注意：adhere to 是一个固定的短语，当句子从主动语态变为被动语态时，前置词 to 必须保留。

2. TRANSLATORS

Most programming today is in a language different from machine language.① We say that the program is in machine language when it can be fed directly into the computer, where the loader places it in memory and it is ready to run.② A source language program requires one or more stages of translation to produce the machine language program.

Source Languages

The programmer usually writes in a source language. We distinguish several kinds of source languages according to their use.

Assembly Language

Assembly language provides commands which are very close to machine-language commands, and it falls into the three categories which will be discussed. The mnemonic language is an assembly language. Other extant assembly languages for IBM machines include FAP and SOS; Univac assembly languages include UTMOST and ALMOST.

POLs

The programmer is provided with languages which convey information in a manner similar to that in which he normally expresses himself in writing out his algebraic or business problem.③ These are called procedure oriented languages, abbreviated occasionally as POLs.④ They include FORTRAN, ALGOL, and COBOL. These are also called compiler languages after the programming system which translates them,

the compiler.⑤

Problem Oriented

Problem oriented languages are aimed at stating a problem for which a general solution has already been programmed.⑥ The Report Generator Language enables us to state the design of a given report;⑦ the Sort Generator allows us to communicate the nature and forma of records to be sorted.⑧

Special

Special languages are structured for the statement of special kinds of problems.⑨ A System Simulator Language is designed to state the properties of a system which is being simulated on the computer.⑩ IPL-V (Information Processing Language Number Five) is designed to express problems which simulate human thinking.

词　汇

feed [fi:d] *vt.* 馈给;输入 (fed [fed], fed [fed])
category [kætigəri] *n.* 种类
mnemonic [ni(:)'mɔnik] *n.* 记忆的;帮助记忆的
extant [eks'tænt] *a.* 现存的
Univac = **Universal Automatic Computer** 通用电子数字计算机(计算机型号名)
convey [kən'vei] *vt.* 传达,传递
algebraic [ˌældʒi'breiik] *a.* 代数的
abbreviate [ə'bri:vieit] *vt.* 缩写
sort [sɔ:t] *vt.* 把…分类
simulate ['simjuleit] *vt.* 模拟

短　语

(to be) different from 与…不同
(to be) ready to (不定式符号)即将
(to be) close to 接近于
in a manner similar to 以类似于…的方式

注　释

① in a language ... language 是前置词短语,作表语,说明 Most programming。其中,形容词短语 different from machine language 作定语,

修饰 a language。

② when it can be fed ... to run 是时间状语从句，修饰主句的谓语动词 say, that the program is in machine language 是 say 的宾语从句，where the loader ... to run 是非限制性定语从句，修饰 the computer。三个 it 均指 the program。

③ 从 which convey 到结束是定语从句，修饰 languages。其中，in a manner 到结束是前置词短语，作状语，修饰谓语动词 convey。在这前置词短语中，又有一个形容词短语 similar to that ... in which ... business problem，作定语，修饰 a manner。在这形容词短语中，that 指 the manner，in which 到结束是定语从句，修饰 that。

④ abbreviated occasionally as POLs 是过去分词短语，作非限制性定语，修饰 procedure oriented languages。其中，as POLs 是主语补足语，补足 abbreviated 的逻辑主语，即 procedure oriented languages。

⑤ after the programming system ... the compiler 是前置词短语，作方式状语，修饰谓语动词 are called。前置词 after 在这里是"仿照、依照"的意思。在这前置词短语中，有一个定语从句 which translates them，修饰 the programming system; the compiler 是 the programming system 的同位语。

⑥ for which ... programmed 是定语从句，修饰 a problem。

⑦ 不定式短语 to state the design of a given report 作状语。

⑧ 被动语态不定词 to be sorted 作定语，修饰 records。

⑨ 在 the statement ... problems 中，the statement 与 special kinds of problems 在逻辑上是动宾关系，由 of 连接。

⑩ which is being simulated on the computer 是定语从句，修饰 a system。这句定语从句是现在进行时态、被动语态。

3. ASSEMBLY LANGUAGES

We distinguish three kinds of assembly language:

absolute assembly language	AAL
symbolic assembly language	SAL
macro assembly language	MAL

Absolute assembly language

Early FLAP, introduced in Chapter 2,① is an example of an absolute assembly language. It differs in only two ways from actual machine language:

- A mnemonic — a set of letters — is substituted for the binary command code.
- A letter or number code is used for a memory cell instead of the binary representation of this code.②

AAl is a shorthand notation for machine language commands. These are translated to machine language by a very simple assembler. No storage allocation is done.

Symbolic assembly language

A symbolic assembly language is discussed in Chapter 5 and is exemplified by Middle FLAP. In this language, the location of data is denoted symbolically and need not be allocated by the programmer. The assembler does the full allocation of storage and keeps track of all memory cells. The SAL can also deal with arrays and provide address modification by addition and subtraction.

Macro assembly language

Most notably, the macro assembly language permits the

programmer to define and name a subroutine and then to call it forth any place in the program that he desires.③

AL — POL Contrast

The absolute assembly language has the characteristic that each source language command is represented by exactly one machine language command.④ Symbolic assembly language introduces pseudos present in source language but absent from machine language.⑤ The macro assembly language introduces strings of commands in the macro definition which are entirely absent from the machine language translation.⑥ It also introduces single macro commands which are replaced in translations by a string of machine language commands. Despite these variations a pattern is evident: most assembly language translations present a one-to-one structure — one machine language command for one source language command. When the structure is violated, the sequence of machine language commands is evident in the source language problem statement.

For the procedure oriented language one source language statement is usually translated into many machine language statements. More important,⑦ the sequence of statements produced⑧ is a function of the POL compiler and the object machine, and is not at all evident in the source language statements.

Summary

The most important differences between the AL and POL are:

- A sequence of machine language steps is nowhere implied in the POL.
- The one-to-many characteristic is customary in the compiler whereas in the assembler it is an exception.

词　汇

binary ['bainəri] *a.* 二进制的
shorthand ['ʃɔ:thænd] *n.* 速记
exemplify [ig'zemplifai] *vt.* 举例说明；作为…的例子
modification [ˌmɔdifi'keiʃən] *n.* 修改；改变
macro ['mækrəu] 宏，大（构词成份）
notably ['nəutəbli] *ad.* 显著地；著名地
pseudo ['(p)s(j)u:dəu] 伪，拟，假（构词成份） *n.* 伪指令
violate ['vaiəleit] *vt.* 扰乱；侵犯；违背
nowhere ['nəuhwɛə] *ad.* 任何地方都不
customary ['kʌstəməri] *a.* 惯例的，通常的

短　语

(to) differ from 不同于；与…有区别
(to be) absent from 不在…（地方）；缺…
not at all 毫不

注　释

① 过去分词短语 introduced in Chapter 2 作定语，修饰 Early FLAP。
② instead of ... code 是前置词短语，作状语，修饰谓语动词 is used。
③ 在本句中，不定式短语 to define and name a subroutine 和 then to call it forth ... desires 与宾语 the programmer 一起构成了复合宾语。在 then to call ... desires 这个不定式短语中，any place in the program that he desires 是名词短语，作地点状语，修饰 call it forth。其中，that he desires 是定语从句，修饰 any place。
④ that ... one machine language command 是与 characteristic 同位的同位语从句。
⑤ present in source language 和 absent from machine language 是两个形容词短语，修饰 pseudos。
⑥ 定语从句 which are ... translation 是修饰前面 strings of commands 的。
⑦ more important 作状语，修饰在其后面的整个句子。
⑧ produced 是过去分词，修饰 the sequence of statements。

4. COMPILERS

Compiler is the part of the programming system that translates programs from their original language into machine language. Compilers are often classified as one pass, two pass, three pass compilers and so on.① These are terms which are most② relevant in cases where the program is large compared with the internal storage system,③ so that the entire program cannot be held within the machine while it is being translated from source to machine language. What one usually does then is to keep it on magnetic tape and simply scan it repeatedly, performing a further translation on it at each stage,④ and the number of times that it has to be scanned in order to get the complete translation done⑤ is called the number of passes.

This term might also be applicable in cases where it is not necessary to keep it on the magnetic tape,⑥ because there is room for the whole program inside the machine. Even then a compiler may be organized in such a way that it scans through the program being interpreted from beginning to end (or from end to beginning) a certain number of times,⑦ and it may still be described as a two, or a three or a four pass compiler. But many compilers do not work in this kind of way.

If a compiler does make a clear number of passes, then obviously in between⑧ two passes the program is still there in some form or other; in fact, it is in an intermediate language, between the source language and the final language.

Another is the langauge in which the compiler itself is writ-

ten.⑨ Compilers are such enormous things that one rarely writes them in machine language any more,⑩ and one can choose, in the same way as in the writing of any program,⑪ between various possible languages in which it might be written. It is important to distinguish between the language in which a compiler is written, and the languages to which, through which, and from which it compiles.⑫ It may happen in some cases that the language from which it compiles is the same as the language in which the compiler itself is written,⑬but this is not necessarily so.

In considering the best language in which to write a compiler,⑭ we are effectively considering a system designed to accept compilers, i.e. a system to which the compiler itself acts as data.⑮ The purpose of this data is to select the required compiler from among conceivable compilers (i.e. compilers which accept various source languages or produce various machine languages), and so really what we want to be able to write as our compiler⑯ is simply a definition of the source language or of the machine language or both. We want to have a system into which we could feed, for example, a definition of a source language,⑰ so that having supplied that definition we would now have an effective compiling system into which we could feed the source language itself.

In fact what this system has to accept is what mathematicians call a meta-language, that is a language which is used for describing languages. This kind of approach has been explored by several people. Various kinds of meta-languages have been devised, and various means for building systems that will accept these languages.

Another aspect of compilers is the question of diagnostics: