Compilers – 4170 words – college essay

Compiler: A Definition

Compiler, in computer science, computer program that translates source code, instructions in a program written by a software engineer, into object code, those same instructions written in a language the computer’s central processing unit (CPU) can read and interpret. Software engineers write source code using high level programming languages that people can understand. Computers cannot directly execute source code, but need a compiler to translate these instructions into a low level language called machine code.

Compiler: How It Works

Compilers collect and reorganize (compile) all the instructions in a given set of source code to produce object code. Object code is often the same as or similar to a computer’s machine code. If the object code is the same as the machine language, the computer can run the program immediately after the compiler produces its translation. If the object code is not in machine language, other programssuch as assemblers, binders, linkers, and loadersfinish the translation.

Most programming languagessuch as C, C++, and Fortranuse compilers, but somesuch as BASIC and LISPuse interpreters. An interpreter analyzes and executes each line of source code one-by-one. Interpreters produce initial results faster than compilers, but the source code must be re-interpreted with every use and interpreted languages are usually not as sophisticated as compiled languages.

Most computer languages use different versions of compilers for different types of computers or operating systems; so one language may have different compilers for personal computers (PC) and Apple Macintosh computers. Many different manufacturers often produce versions of the same programming language, so compilers for a language may vary between manufacturers.

Consumer software programs are compiled and translated into machine language before they are sold. Some manufacturers provide source code, but usually only programmers find the source code useful. Thus programs bought off the shelf can be executed, but usually their source code cannot be read or modified.

When executing (running), the compiler first parses (or analyzes) all of the language statements syntactically one after the other and then, in one or more successive stages or “ passes”, builds the output code, making sure that statements that refer to other statements are referred to correctly in the final code. Traditionally, the output of the compilation has been called object code or sometimes an object module. (Note that the term “ object” here is not related to object-oriented programming.) The object code is machine code that the processor can process or “ execute” one instruction at a time.

Related Study

More recently, the Java programming language, a language used in object-oriented programming, has introduced the possibility of compiling output (called bytecode) that can run on any computer system platform for which a Java virtual machine or bytecode interpreter is provided to convert the

bytecode into instructions that can be executed by the actual hardware processor. Using this virtual machine, the bytecode can optionally be recompiled at the execution platform by a just-in-time compiler.

Traditionally in some operating systems, an additional step was required after compilation – that of resolving the relative location of instructions and data when more than one object module was to be run at the same time and they cross-referred to each other’s instruction sequences or data. This process was sometimes called linkage editing and the output known as a load module.

A compiler works with what are sometimes called 3GL and higher-level languages. An assembler works on programs written using a processor’s assembler language.

Computer languages are symbolic systems that computers eventually understand. They help your programs serve your needs. Compilers are programs that help to make this “ understanding” happen. While the first generation of high-level programming languages, such as Fortran, are still in wide use and evolving, many new languages with higher-level abstraction capability are emerging. Since the nature of scientific research is to explore the unknown world and test new theories, programming languages are usually the most important interface between scientists and computers. For computer scientists, improving code execution efficiency on a given architecture and developing high-level abstraction capability of the language are challenging.

Machine Code, or machine language, in computer science, low-level programming language that can be understood directly by a computer’s central processing unit (CPU). Machine code consists of sequences of binary numbers, or bits, which are usually represented by 1s and 0s, and which form the basic instructions that guide the operation of a computer. The specific set of instructions that constitutes a machine code depends on the make and model of the computer’s CPU. For instance, the machine code for the Motorola 68000 microprocessor differs from that used in the Intel Pentium microprocessor.

Writing programs in machine code is tedious and time-consuming since the programmer must keep track of each specific bit in an instruction.

Another difficulty with programming directly in machine code is that errors are very hard to detect because rows and columns of 1s and 0s represent the program. To overcome these problems, American mathematician Grace Murray Hopper developed assembly language in 1952. Assembly language uses easy to remember termssuch as SUB for a subtraction operation, and MPY for a multiplication operationto represent specific instructions in machine code.

Assembly language makes programming much easier, but an assembly language program must be translated into machine code before it can be understood and run by the computer. Special utility programs called assemblers perform the function of translating assembly language code into machine code. Like machine code, the specific set of instructions that make up an assembly language depend on the make and model of the computer’s

CPU. Other programming languages such as Fortran, BASIC, and C++, make programming even easier than with assembly language and are used to write the majority of programs. These languages, called high-level languages, are closer in form to natural languages and allow very complicated operations to be written in compact notation.

Like assembly languages, all high-level languages must first be translated into machine code before a computer can run them. To accomplish this, programmers use various types of utility programs, such as compilers, assemblers, linkers, and debuggers, which help them translate high-level language into machine code. The computer code used to write a program is called source code before being translated into machine code, and object code after it has been translated.

Kinds of Compilers

B/D — interactive programming language for analyses of Bayes linear statistical problems.

ACL2 — programming language to model computer system and prove properties.

ADAMO — scientific programming system based on the Entity- Relationship (ER) model.

ADL — a specification language for programming interfaces.

Algae — a high-level interpreted language for numerical analysis.

AMC — a compiler of C with model and object-oriented functionality.

APL — a unique, general-purpose high level programming language

APL c compiler — a compiler which translates APL to C code.

CAPLIB2 — an APL interpreter and a C library for APL calls.

Applix SHELF — an embedable fully featured programming language.

APRIL — a symbolic programming language for multi-agent systems.

ASpecT — a strict functional language.

BASIC —

Bywater BASIC — ANSI BASIC interpreter.

CINQ — to create a BASIC compiler system.

Chipmunk Basic — an old fashioned Basic interpreter.

Harbour — a project to develop a free, cross-platform compiler for Xbase language.

HotTEA — an implementation of the BASIC language written in Java.

Instant Basic for Java — pure Java certified 4GL tool similar to Visual Basic.

OmniBasic — a structured dialect of Basic with many extensions.

OpenBasic — Basic interpreter and run-time system similar to Basic Four Business Basic.

PRO/5 — business application development tools based on BBx BASIC language.

QB2C — a QuickBASIC to C translator with X11 graphics support.

QueriX 4GL — 4GL compilers which are Informix 4GL compatible.

ScriptBasic — cross-platform BASIC scripting language.

STBasic — a structured BASIC interpreter.

VBVM — a portable version of MS Visual Basic 5 virtual machine.

YABASIC — implements most comman BASIC elements with some graphics.

Zen — a full featured BASIC interpreter.

BCPL/MCPL — simple typeless languages.

b2c — a compiler which reads BETA and translates it to C.

BETA System — modern object-oriented language.

gbeta — a new implementation of a generalization of BETA language.

Blue — an object-oriented programming language that was developed especially for teaching.

BOIL — C-like language with distributed programming, accounting, etc.

Brain — a fully object-oriented high level scripting language.

C/C++ —

Comeau C++ — a multi-platform C++ front end compiler.

Compaq C for Linux Alpha — Alpha chip optimized C compiler.

CH — The CH language environment is a superset of C.

CINT — a C/C++ interpreter which is aimed at processing C/C++ scripts.

C Scripting Language — an embeddable scripting language with C syntax.

Dynace — an object-oriented extension to C.

egcs — an experimental step in the development of GCC.

EiC — an extensible interactive, pointer-safe, bytecode C interpreter/compiler.

Fujitsu High Performance C/C++ and Fortran 95 Compiler

GCC — GNU C/C++/Objective-C compiler system.

Intel C++ and Fortran Compilers

KAI C++ — an optimized cross-platform C++ compiler.

lcc — a retargetable compiler for ANSI C.

OpenC++ — a version of C++ with Metaobject Protocol.

optimizer — optimizes GCC/G++ assembler code to run with Pentium instructions.

Pentium-GCC — Pentium optimized GCC.

Portable Object Compiler — an independent implementation of Objective C.

Portland Group parallelizing HPF, F77, C, and C++ Compilers

SGI Pro64 — a suite of optimizing compiler development tools for Linux/Intel Itanium(TM) systems.

Stepstone Objective C compiler — objective C compiler.

TenDRA — a public domain C/C++ compiler and checker technology.

C-INTERCAL — an implementation of language INTERCAL which look like no others.

Cecil/Vortex — a language-independent optimizing compiler back-end for OO languages.

cim — a compiler for the programming language Simula.

CLAIRE — a functional and OO language with advanced rule processing capabilities.

Clif — a C-like interpreter framework.

CM3 — a cross-platform compiler and runtime libraries.

COBOL —

COBOL Interpreter — a compact and easy to use COBOL interpreter.

ACUCOBOL-GT — GUI-enhanced COBOL applications developing system.

Deskware COBOL Interpreter — a compact and easy to use COBOL interpreter.

PERCobol — COBOL compiler bridging COBOL and Java technology.

tiny cobol — an effort to bring a free cobol compiler to Linux.

CompWork — a framework for the TCL compiler suite.

DINO — an interpreter with high level scripting dynamic-typed language.

EGR Cobra — an interpreted programming language with C++–like syntax.

ECLiPSe — a development environment for constraint programming applications.

Eiffel — an advanced object-oriented programming language.

Eiffel/S_1. 3S — Eiffel Compiler for Linux.

Eon/Eiffel — an implementation of the Eiffel Language.

ISE Eiffel — includes wide range of products based on Eiffel.

SmallEiffel — a free Eiffel compiler.

TowerEiffel — a complete software development environment.

elastiC — a portable high-level OO interpreted language with a C like syntax.

emu — a clean, fast, flexible, and embeddable programming language.

eNITL — a scripting and template language engine for C++ applications.

Esterel — a family of synchronous languages for programming reactive systems.

Euphoria — a simple, flexible, fast, and easy to learn interpreted language.

Expect — a scripting language to interace with programs such as FTP, telnet, fsck, etc.

Flick IDL Compiler — a compiler for interface definition language.

Forth —

bigFORTH — a 32bit, native ANS Forth compiler.

Gforth — a fast and portable implementation of the ANS Forth language.

ficl — a Forth interpreter writtenin C.

kForth — a simple Forth programming language and environment.

PFE — a programming environment for the programming language Forth.

ThisForth — a macro-oriented Forth based on Standard (ANS) Forth

Fortran —

Absoft Pro Fortran — globally optimizing Fortran 77/90 compilers, and debugger.

Compaq Fortran for Linux Alpha — Alpha optimized Fortran compiler.

Intel C++ and Fortran Compilers

F — a carefully crafted subset of the most recent version of Fortran.

f2c — a Fortran 77 to C translator.

f77reorder — a f77 filter which solves some compatibility problem of f2c.

Fujitsu High Performance C/C++ and Fortran 95 Compiler

Lahey/Fujitsu Fortran 95 Express — full Fortran 77, 90 and 95 language system.

NASoftwae FortranPlus — a full implementation of Fortran90/95 standard.

NDP Fortran — a full implementation of Fortran 77 with lots of extentions.

GNU Fortran — GNU Fortran compiler (g77).

g95 — a project to create a free, open source Fortran 95 compiler.

NAGWare f77 to f90 Converter — a GUI for upgrading code to Fortran 90.

NAGWare f77 Tools — processing/analysing/transforming Fortran 77 code.

NAGWare f90 Compiler — Fortran 90 compiler from NAG.

NAGWare f90 Tools — source-to-source translations of Fortran 77 and 90 code.

Portland Group parallelizing HPF, F77, C, and C++ Compilers

Ratfor — converts the Rational Fortran dialect into ordinary Fortran 77.

SGI Pro64 — a suite of optimizing compiler development tools for Linux/Intel Itanium(TM) systems.

VAST/77to90 — a Fortran 77 to Fortran 90 translator.

VAST/f90 — a Fortran 90 to g77 translator.

FPL — a C-like interpreting script/macro language.

GNAT — a high-quality and complete compiler for Ada95 integrated into GCC.

GNU Sather — object oriented language which designed to be simple, efficient, safe.

Groovy Java Genetic Programming — a strongly-typed genetic programming experimentation platform.

Gdel — a declarative, general-purpose programming language.

GOMscript — an interpreter for an object-oriented, C++-like, language.

Gorby — a small, stack-based scripting language.

Guile — includes an embeddable Scheme interpreter, several graphics options.

GOOPS — an object-oriented extension to Guile.

Gwydion Dylan — an implementation of a dynamic language strongly resembling Dylan.

HASKELL — a general purpose, purely functional programming language.

Hugs — a functional programming system based on Haskell.

Glasgow Haskell Compiler — an implementation of Haskell functional language.

ICI — a general purpose interpretive programming language.

Icon — programming language with features for processing text and data structures.

Jcon — a Java-based Icon implementation.

Internet C++ and ICVM — an open alternative to Java and C-Sharp, and its virtual machine.

Inferno — a network operating system and programming environment.

iPP — a platform independent preprocessor written entirely in Java.

iScript — a platform independent scripting language written entirely in Java.

J — high level language emphasized on functional programming and array processing.

Jacl — a Tcl interpreter for Java.

Java & JDK — a powerful object oriented language and development environment.

CACAO — 64 bit just-in-time (JIT) compiler for Java on Alpha processor.

DynamicJava — a Java source interpreter written in Java.

EPP — an extensible Java source-to-source pre-processor.

GCJ — a GCC front end for compiling Java source and bytecode.

GJ — a generic Java language extension.

Guavac — a standalone compiler for the Java programming language.

Harissa — a compiler from Java bytecode to C and a Java interpreter.

Homebrew Decompiler — a decompiler for Java class files.

Instant Converter — converts BASIC source code to Java source code.

J-Mate — Java development environment.

j2c — a translator from java . class to C program.

Jad — a fast Java decompiler.

JavaParty — an extension of Java with some important new features.

Japhar — Hungry Programmer’s Java VM.

jas — a Java bytecode assembler.

Jasmin — a Java assembler interface.

JavaCC (Java Compiler Compiler) — a Java parser generator.

Jester — an extension to Java to program reactive systems with Esterel constructs.

Jikes — faster Java compilers that adheres to languages and VM specifications.

JUMP — a bytecode compiler to compile JAVA source code.

Linux JDK — IBM’s port of Java 2 Platform to Linux.

Kaffe — a virtual machine design to execute Java bytecode.

KOPI — a completely open source Java compiler.

JOLT — implementation of Sun’s Java language and tools.

NetRexx — a human-oriented language designed to be an alternative to Java.

PIZZA — a substantial companion to Java.

Snacc — A Java stub compiler for ASN. 1 specifications.

Toba — a Java-to-C translator.

TowerJ — a high performance compiler for server-side Java.

TurboJ — a high perfromacne Java compiler.

TYA — a JIT-compiler for Linux JDK.

WingDis — a Java decompiler.

JEL — a library for evaluating a simple single line expressions in Java.

LDL — a language development laboratory.

LIFE — an experimental language proposing to integrate logic/functional/OO programming.

Lisp — a high-level language, especially popular for artificial intelligence.

Allegro CL — a Common Lisp system for application development and deployment.

CLISP Common Lisp — a popular Common Lisp implementation.

CMU Common Lisp — CMU implementation of Lisp.

Elisp (Emacs Lisp) — the language used to extend emacs.

EusLisp — an integrated programming system for the research on intelligent robots.

GCL — implementation of Lisp that used to be known as Kyoto Common Lisp.

Graphic Lisp — portable Common Lisp language written in ANSI-C.

iLisp — a small, multiplatform implementation of Lisp language.

Linux Poplog — developing CLisp, Prolog, Pop-11 and standard ML.

LISP DEBUG — source level debugger with GUI for LISP program.

Screamer — an extension of Common Lisp with nondeterministic programming.

Simple Lisp — a simple lisp interpreter with symbolic math library for tensors.

Steel Bank Common Lisp — a development environment for Common Lisp.

xbvl — a Lisp dialect.

Lua — a simple, yet powerful, language for extending applications.

Marlais — an object-oriented dynamic language.

Maxtal Interscript — an advanced literate programming system.

Mercury – logic programming language.

Micro Tools for Linux — development tools for Atmel AVR micro-controllers.

ML — a strict functional language

Caml Light — a small, portable implementation of the ML language.

Extended ML — a framework for specification and development of Standard ML.

Moscow ML — a light-weight implementation of Standard ML.

Objective Caml — an implementation of the Caml dialect of ML.

Standard ML of New Jersey (SML/NJ) — a compiler for the Standard ML.

mixal — language for D. E. Knuth4s hypothetical computer, the MIX 1009.

Modula-3 — a simple and safe modern systems programming language.

GPM — consistent implementation of Modula-2 on contemporary machines.

MOCKA — a Modula-2 compiler system.

SRC Modula-3 — a free implementation of Modula-3 on many operating systems.

Cambridge Modula-3 — a free implementation of Modula-3 on many OS’s.

Mozart — a distributed programming language for symbolic and reactive programming.

NCL — a natural constraint language.

NQC — a simple language with a C like syntax.

Oberon — a programming language and a modern operating system.

JACOB — a compiler for Oberon-2.

Oberon V4 for GNU/Linux — an implementation of Oberon V4 for Linux.

oo2c — Oberon-2 to ANSI-C translator.

XDS — a bilingual programming system featuring Modula-2 and Oberon-2.

OPAL — a language designed as a testbed for developing functional programs.

OZ — a concurrent constraint programming language and its interactive implementation.

PAG — a Program Analyzer Generator which makes static program analysis easy.

PAMELA — a language that supports a procedure-oriented modeling paradigm.

Pascal —

Free Pascal — a free 32-bit Pascal compiler.

gpc — GNU Pascal compiler.

NDP Pascal — a full implementation of Pascal.

PTOC — ANSI/Turbo Pascal to C/C++ converter with BGI graphics library emulation.

Perl — an interpreted language intended to be practical.

ePerl — an embedded Perl 5 language.

Perl/Tk — Access Tk facilities within Perl.

PFL — a simple functional language with database extensions.

Pike — a dynamic programming language with a syntax similar to C.

PLAN — a programming language for active networks.

Pliant — a programming language framework and a new generation of language.

PRECC — an infinite-lookahead compiler-compiler for context dependent grammars.

Prolog — high-level language based on formal logic, widedly used in artificial intelligence.

B-Prolog — a CLP system that runs Prolog and CLP(FD) programs.

BinProlog — a fairly complete and efficient Prolog compiler.

clp(FD, S) — semiring-based constraint logic programming language over finite domains.

GNU Prolog — a Prolog compiler with constraint solving over finite domains.

jProlog — a Prolog interpreter in Java.

JVProlog — implementation of Prolog based on special abstract machine.

Linux Poplog — developing Common Lisp, Prolog, Pop-11 and standard ML.

LLP — a logic programming language based on intuitionistic linear logic.

Lolli — a logic programming language based on a fragment of linear logic.

Lygon — Prolog extended with features derived from linear logic.

PM — a compiler for the logic programming language LambdaProlog.

Prolog + Logic Server — Prolog component developing in other languages.

SICStus Prolog — advanced Prolog applications development system.

SWI-Prolog — a fairly complete Edinburgh-style Prolog.

Terzo — an interpreter of lambda Prolog implemented in Standard ML of NJ.

W-Prolog — an interpreter for a Prolog like language implemented in Java.

wamcc — a Prolog Compiler which translates Prolog to C.

XSB — an extension of an Edinburgh Prolog system.

Pnuts — a scripting language totally written in Java.

Python — interpreted, interactive, object-oriented programming language.

JPython — an implementation of Python integrated with the Java platform.

Ptui — a Python editor/interpreter programmed entirely in Python.

Pyfort — creating extensions to the PYTHON using Fortran routines.

QCL — programming language and simulator for quantum computers.

R — language for statistical calculations.

Reactive-C — a C based language for reactive programming.

REBOL — a multi-platform Internet communications language.

REXX — procedural programming language.

NetRexx — a programming language designed to be an alternative to Java.

Object REXX — a scripting language which is easy to learn and read.

Regina — an implementation of the REXX language.

REXX/imc — REXX implementation on UNIX.

S/REXX — UNIX implementation of IBM’s SAA procedural language.

Rigal — a programming language and tool for compiler writing.

Rivl — a multimedia processing language.

Rivet — a version of Tk without the overhead of Tcl.

Ruby — an interpreted scripting language for quick and easy OO programming.

Sather-K — a modern object-oriented, imperative programming langauage.

Scheme — a high-level language and is a dialect of Lisp.

Bigloo — a Scheme interpreter and compiler.

DrScheme — a graphical environment for developing Scheme programs.

Elk — an implementation of the Scheme programming language.

Gambit — a high-performance implementation of Scheme.

Hobbit — a small optimizing scheme-to-C compiler.

Inlab-Scheme — an independent implementation of Scheme.

Kawa — a Java-based Scheme system.

LISC — a lightweight Scheme interpeter written in Java.

MIT Scheme — a Scheme compiler which generate faster code.

MrEd — GUI development environment with Scheme language.

MzScheme — a Scheme implementation which is R4RS-compilant.

OpenScheme — a Scheme interpreter/compiler/debugger conform to the R4RS.

Oscheme — Objective Scheme.

QScheme — a fast and small implementation of Scheme written in C.

RScheme — well-structured implementation of the Scheme language.

Scheme-to-C — a Scheme-to-C compiler.

Scsh — broad-spectrum programming environment embedded in R4RS Scheme.

SIOD — a small scheme interpreter with database, unix programming and cgi extensions.

STk — Scheme interpreter which can access to the Tk graphical package.

SDCC — a retargettable, optimizing ANSI-C compiler.

SeeR — a multipurpose C-like scripting library.

Shift — a programming language for describing dynamic networks of hybrid automata.

SILOON — scripting interface languages for object-oriented numerics.

Smalltalk — object-oriented programming language.

Cincom VisualWorks — a robust cross-platform Smalltalk development environment.

GNU Smalltalk — GNU Smalltalk implementation.

Smalltalk/X — implementation of the Smalltalk language.

Squeak — Squeak is a full implementation of the Smalltalk-80 system.

SNOBOL4 — a port of Macro SNOBOL4 for machines with 32-bit pointers.

Spanner — a glue/scripting/prototyping programming language.

Tcl/Tk — embeddable scripting language and graphical user interface toolkit.

Tcl Blend — a Tcl extension that provides access to Java inside Tcl.

tinySelf — an implementation of Self for Linux.

Titanium — a dialet of Java for large-scale scientific computing.

Toby — a LOGO-like language written in Java.

TXL — a language and rapid prototyping system designed to support source text analysis and transformation tasks.

UML — a language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

Yoix — scripting language uses syntax and functions similar to C and Java.

AFML — a language for modeling the structural and semantic aspects of real world.

AL — a programming language for modeling and animation.

AMPL — a modeling language for mathematical programming.

cfengine — a very high level language for administrating and configuring unix-like systems.

Cilk — an algorithmic multithreaded language.

Dynamite — a code generating language developed for sequence comparison methods.

GAMS — a high-level modeling system for mathematical programming problems.

DISGCL — an interpreter language based on plotting library DISLIN.

Glish (within AIPS++ system) — a language/environment for data acquisition/analysis.

Isaac — scientific calculator and programming language.

MAX — Xbase compiler with integrated database engine.

MetaCard — a multimedia authoring tool and GUI development environment.

MSDL — a scene description language for graphics research.

Nickle — a desk calculator language with powerful programming and scripting capabilities.

PerlDL — turn perl into an array-oriented, numerical language.

ProvideX — an object-oriented, business basic development environment.

RLaB — matrix oriented, interactive programming environment.

S-Lang — an interpreted language could be embedded into an extensible application.

Soar — a cognitive architectural framework and mode ls, and an AI programming language.

ZPL — a portable, high performance parallel programming language for computations.

References

www. programmersheaven. com

www. compiler. net

www. msn. encarta. com

www. webopedia. com