February 22, 2012

RV College of Engineering

Department of Computer Science

PC Based Product Development
using RV All In One Card Interface Board
in $Java^{TM}$

Instruction Manaul

By:
Samir Sheriff
Satvik N

Under the guidance of:
Prof. K. Badarinath

February 22, 2012

0.1 Introduction

A compiler is a computer program (or set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language). The most common reason for wanting to transform source code is to create an executable program.

The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language or machine code). If the compiled program can run on a computer whose CPU or operating system is different from the one on which the compiler runs, the compiler is known as a cross-compiler.

A compiler is likely to perform the following operations: lexical analysis, preprocessing, parsing, semantic analysis (Syntax-directed translation), code generation, and code optimization.

0.2 Compiler Optimization

Compiler optimization is the process of tuning the output of a compiler to minimize or maximize some attributes of an executable computer program. The most common requirement is to minimize the time taken to execute a program; a less common one is to minimize the amount of memory occupied. The growth of portable computers has created a market for minimizing the power consumed by a

0.3 Factors affecting Optimization

The machine itself
The type of optimization that can be used depend on the characteristics of the target machine. It is sometimes possible to parameterize some of these machine dependent factors, so that a single piece of compiler code can be used to optimize different machines just by altering the machine description parameters. GCC is a compiler which exemplifies this approach.
The architecture of the target CPU
Number of CPU registers: To a certain extent, the more registers, the easier it is to optimize for performance. Local variables can be allocated in the registers and not on the stack. Intermediate results can be left in registers without writing to and reading back from memory.

0.4 Static Single Assignment form

This is a specific technique of compiler optimizations which are intended to be done after transforming the program into a special form called static single assignment, in which every variable is assigned in only one place.

In compiler design, static single assignment form (abbreviated as SSA form) is a property of an intermediate representation (IR), in which each variable is assigned exactly once. Existing variables in the original IR are split into versions, new variables typically indicated by the original name with a subscript, so that every definition gets its own version.

SSA was developed by Ron Cytron, Jeanne Ferrante, Barry K. Rosen, Mark N. Wegman, and F. Kenneth Zadeck, researchers at IBM in the 1980s.