Code Generation

66.648 Compiler Design Lecture (03/31//97)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- Code Generation (Chapter 9)
- Examples
- Administration

Phases of a Compiler

- Lexical Analyzer
- Parser (Syntax Analyzer)
- Semantic Analyzer
- Intermediate Code Generator
- Code generator
- Intermediate Code optimizer

Target Code

1. Assembly Language

Advantage: Simplifies Code generation due to the use of symbolic names and mnemonics

2. Relocatable Machine Language (Object Modules)

Advantage: Flexibility of separate compilation.

Disadvantage: Overhed of extra linking and loading steps, and extra complexity of object code formats.

Target Code - Contd

3. Absolute Machine Language (hard-coded address)

Advantage: Simpler and more efficient than generating object modules (also useful for dynamic code generation)

Disadvantage: inflexible --hard to reload generated code.

We will generate assembler code - the approach used in the text book.

Target Machine

Since we use assembly language as the target code, the target machine for a compiler is the virtual machine at the assembly language level.

Registers - There are conventions about using the registers. There are also floating point and multiply/divide registers. Some of the registers are used as stack pointer, frame pointer.

There could also be temporary registers.

Addressing Modes: What kind of addressing your machine permits and how many machine cycles

Target Machine

Are needed to execute?

Normal Addressing Modes are: register, stack, direct, indirect, indexed.

Instructions:

1. Load and Store
2. Arithmetic and Logical (shiftleft,shiftright)
3. Compare Instructions (These may set the condition code registers)

Target Machine

4. Unconditional Jump Instructions
5. Subroutine Jumps and returns
6. Conditional Branch Instructions.

Qn: How about I/O instructions - Usually supported by runtime libraries. These libraries will also handle math function calls, file ios etc.

Code Generation

X:=y op z

1) Find out where the result has to be stored - will

usually be a register.

2) Find out the current location of y (may be a register) - a naïve method is to store and load a register.

3) Generate the instruction op z,L (z may be a register).

Example

D := (a-b) + (a -c) + (a-c)

bytecode

iload a

iload b

isub

iload a

iload c

isub

Example

D := (a-b) + (a -c) + (a-c)

bytecode

iadd

iload a

iload c

isub

iadd

istore d

Example

D := (a-b) + (a -c) + (a-c)

Assembler Code:

mov a, r0

sub b,r0

mov a,r1

sub c,r1

add r1,r0

add r1,r0 ; mov r0,d

Code Generation

For simplicity, this code generation algorithm assumed that there is a target instruction available for each intermediate code operator.

Ways to accomplish this:

1) By restricting Intermediate code to a subset of machine instructions.

2) By translating each intermediate code as a sequence of target machine instructions. This will reduce the level intermediate code optimizations.

Issues

1) Register Assignment

How do we allocate registers to variables so that maximum utilization with least cost of execution.

2) Evaluation Order: The order in which computations are performed can affect the efficiency of target code.

Definitions

Basic Blocks: Let us consider three address codes (equivalently bytecodes)

Defn: A basic block is a sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without halt or possibility of branching except at the end.

Finding Basic Blocks

Leader is the first statement in the basic block.

Rules for the leader:

1) First statement in the program is a leader.

2) Any statement that is the target of a conditional or unconditional goto is a leader.

3) Any statement that immediately follows a goto or conditional is a leader.

Basic Block for a leader consists of a leader and all the statements but not including the next leader.

Example

Comments and Feedback

Project 3 is out. Please start working. PLEASE do not wait for the due date to come.

We are in chapter 9. Please read that chapter and read the relevant portion of Java. Please keep studying this material and work exercises.