An Overview to Compiler Design
Outline

- An Overview of Compiler Structure
- Front End
- Middle End
- Back End
Reading

Slides and Lecture Notes

Aho, Lam, Sethi, Ullman:

Chapter 1.4 ~ 1.5
Chapter 2.1 ~ 2.7
Chapter 3.1 ~ 3.5
Chapter 4.1 ~ 4.5
A Review on Compiler Structure/Design

- Overall structure
- **Front-end**: lexical and syntax analysis
- **Middle-end**: *machine independent* code analysis and (scalar, and sometimes loop nest) optimization
- **Back-end**: Machine dependent code analysis and optimization
Motivation for Compiler optimization

Source Program → COMPILER → Machine Code

Input program example:

```c
int foo()
{
    int x;
    return (x + x);
}
```
Output Assembly Code (non-optimized)

foo:
sub $sp, $sp, 8
sw $fp, 8($sp)
add $fp, $sp, 8
sw $ra, -4($fp)
add $t0, $a0 $a0
move $v0, $t0
lw $ra, -4($fp)
lw $fp, 0($fp)
add $sp, $sp, 8
jr $ra
Output Assembly Code (non-optimized)

foo:
sub $sp, $sp, 8 % Push stack frame
sw $fp, 8($sp) % Save old frame pointer
add $fp, $sp, 8 % Set new frame pointer
sw $ra, -4($fp) % Save return address
add $t0, $a0 $a0 % Addition
move $v0, $t0 % Copy return value
lw $ra, -4($fp) % Restore return address
lw $fp, 0($fp) % Restore frame pointer
add $sp, $sp, 8 % Pop stack frame
jr $ra % Jump to return address
Output Assembly Code -- Revisited (non-optimized)

foo:
sub $sp, $sp, 8  %  Push stack frame (for what?)
sw $fp, 8($sp)  %  Save old frame pointer (for what?)
add $fp, $sp, 8  %  Set new frame pointer
sw $ra, -4($fp)  %  Save return address (for what?)
add $t0, $a0, $a0 %  Addition
move $v0, $t0    %  Copy return value
lw $ra, -4($fp)  %  Restore return address
lw $fp, 0($fp)  %  Restore frame pointer
add $sp, $sp, 8  %  Pop stack frame
jr $ra           %  Jump to return address
Output Assembly Code (optimized)

foo:
add $v0, $a0, $a0  % Set result
jr $ra  % Jump to return address
Runtime Memory Organization

Entry point for procedure 1 → code for procedure 1 → code area → global/static area → stack → free space → heap

Entry point for procedure 2 → code for procedure 2

Entry point for procedure n → code for procedure n

Code memory

Address space

Runtime stack

Space for arguments (parameters)
Space for bookkeeping Information, including Return address
Space in local data
Space for local temporaries

A stack frame
Phases of a Compiler

Source program

↓

Lexical Analyzer (Scanner)

↓

Tokens

↓

Syntax Analyzer (Parser)

↓

Parse tree

↓

Semantic Analyzer

↓

Abstract Syntax Tree w/ Attributes

↓

Intermediate-code Generator

↓

Non-optimized Intermediate Code

↓

Intermediate-code Optimizer

↓

Optimized Intermediate Code

↓

Target-code Generator/Opt

↓

Target machine code
Two models of compiler structures
(Muchnick, pp. 08)
A Good Compiler Infrastructure
Needed – A modern View

Front end

Interprocedural Analysis and Optimization

Loop Nest Optimization and Parallelization

Global (Scalar) Optimization

Middle-End

Backend

Code Generation