Compiler Optimizations and Program Analysis

Unnikrishnan C

July 31, 2019

Conteúdo

Overview

Course Overview

- Part1

Course Overview

- Part1
  - SSA form, CFG, Lattices, POSET, Abstract Interpretation
  - DFA theory, LLVM tutorial

Course Overview

- Part1
  - SSA form, CFG, Lattices, POSET, Abstract Interpretation
  - DFA theory, LLVM tutorial
- Part2 (Program Analysis)

Course Overview

- Part1
  - SSA form, CFG, Lattices, POSET, Abstract Interpretation
  - DFA theory, LLVM tutorial
- Part2 (Program Analysis)
  - SSA form, CFG, Lattices, POSET, Monotonic Functions,
  - Control Flow Analysis, Data Flow Analysis
  - Some examples on the same.

Course Overview

- Part1
  - SSA form, CFG, Lattices, POSET, Abstract Interpretation
  - DFA theory, LLVM tutorial
- Part2 (Program Analysis)
  - SSA form, CFG, Lattices, POSET, Monotonic Functions,
  - Control Flow Analysis, Data Flow Analysis
  - Some examples on the same.
- Part 3 (Code Optimizations)

Course Overview

- Part1
  - SSA form, CFG, Lattices, POSET, Abstract Interpretation
  - DFA theory, LLVM tutorial
- Part2 (Program Analysis)
  - SSA form, CFG, Lattices, POSET, Monotonic Functions,
  - Control Flow Analysis, Data Flow Analysis
  - Some examples on the same.
- Part 3 (Code Optimizations)
  - Common Expression Elimination, Partial Redundancy Elimination, local value numbering
  - Introduction to Polyhedral Optimization, Optimization for cache locality
  - few other optimizations.

Course Overview

- Part1
  - SSA form, CFG, Lattices, POSET, Abstract Interpretation
  - DFA theory, LLVM tutorial
- Part2 (Program Analysis)
  - SSA form, CFG, Lattices, POSET, Monotonic Functions,
  - Control Flow Analysis, Data Flow Analysis
  - Some examples on the same.
- Part 3 (Code Optimizations)
  - Common Expression Elimination, Partial Redundancy Elimination, local value numbering
  - Introduction to Polyhedral Optimization, Optimization for cache locality
  - few other optimizations.
- Part 4

Course Overview

- Part1
  - SSA form, CFG, Lattices, POSET, Abstract Interpretation
  - DFA theory, LLVM tutorial
- Part2 (Program Analysis)
  - SSA form, CFG, Lattices, POSET, Monotonic Functions,
  - Control Flow Analysis, Data Flow Analysis
  - Some examples on the same.
- Part 3 (Code Optimizations)
  - Common Expression Elimination, Partial Redundancy Elimination, local value numbering
  - Introduction to Polyhedral Optimization, Optimization for cache locality
  - few other optimizations.
- Part 4
  - Register Allocation, Instruction Selection, Runtime Environments.

Books and references

Books

Compilers: Principles, Techniques, and Tools, Alfred Aho, Monica Lam, Ravi Sethi, Jeffrey D. Ullman, Addison-Wesley; Second Edition, 2006, ISBN-13: 978-0321486813

Reference

Data Flow Analysis: Theory and Practice, Uday P. Khedker, Amitabha Sanyal, and Bageshri Karkare, CRC Press, USA (2009).

LLVM Assignment

You need to write an LLVM Pass. Tutorial available online

Overview

Comparison of Programming Languages

Gendration	Language(s)

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
1^{st}	Machine Language

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
1^{st}	Machine Language
2^{nd}	Assembly Language

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
1^{st}	Machine Language
2^{nd}	Assembly Language
3^{st}	Fortran, C, Lisp, Java

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
1^{st}	Machine Language
2^{nd}	Assembly Language
3^{st}	Fortran, C, Lisp, Java
4^{nd}	SQL, NOMAD

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
1^{st}	Machine Language
2^{nd}	Assembly Language
3^{st}	Fortran, C, Lisp, Java
4^{nd}	SQL, NOMAD
5^{st}	Prolog

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	Language(s)

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	Language(s)
Imperative	

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
	ML, Haskell

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
Functional	ML, Haskell

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
Functional	ML, Haskell
Constraint Logic	

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
Functional	ML, Haskell
Constraint Logic	Prolog

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
Functional	ML, Haskell
Constraint Logic	Prolog
Scripting	

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
Functional	ML, Haskell
Constraint Logic	Prolog
Scripting	Python

Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
Functional	ML, Haskell
Constraint Logic	Prolog
Scripting	Python
Object Oriented	

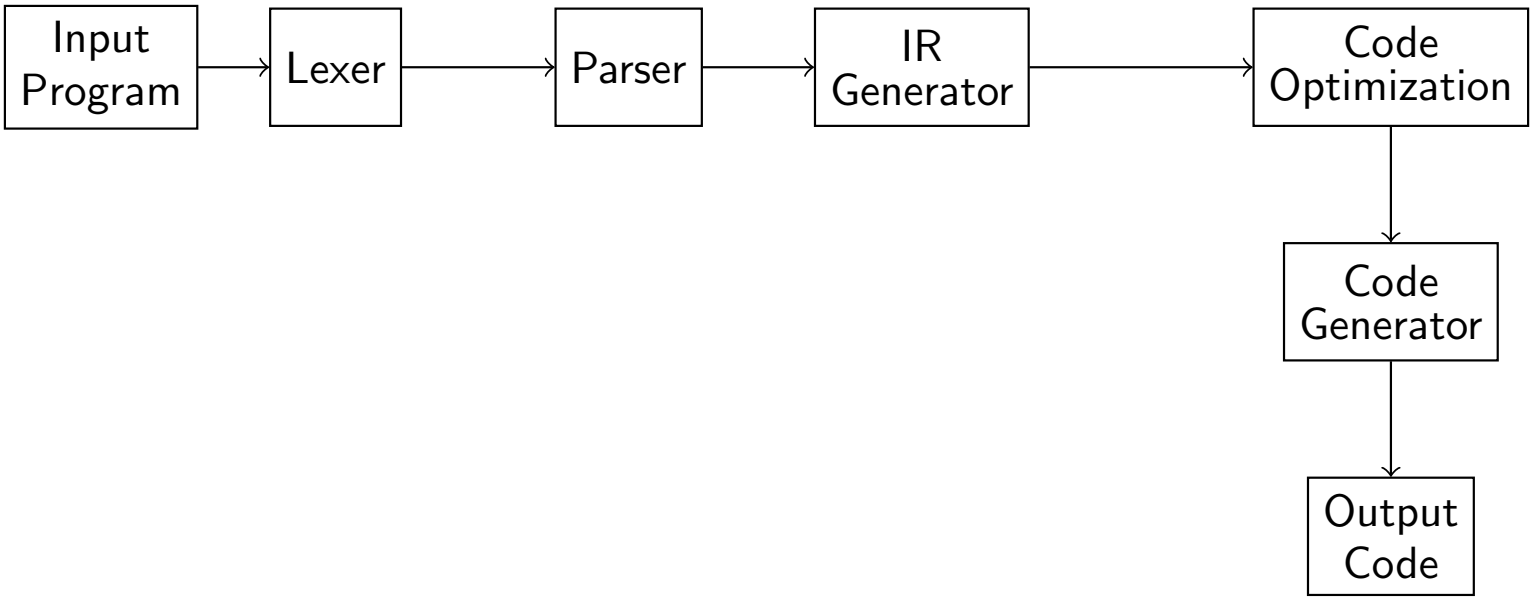
Table: Comparison of Languages

Comparison of Programming Languages

Gendration	**Language(s)**
Imperative	C, Fortran
Functional	ML, Haskell
Constraint Logic	Prolog
Scripting	Python
Object Oriented	C++

Table: Comparison of Languages

Compiler Overview



What you know

- Lexer ?
  - lex, flex

What you know

- Lexer ?
  - lex, flex
- Parser?
  - yacc, bison, antlr

What you know

- Lexer ?
  - lex, flex
- Parser?
  - yacc, bison, antlr
- IR?
  - SSA, AST

What you know

- Lexer ?
  - lex, flex
- Parser?
  - yacc, bison, antlr
- IR?
  - SSA, AST
- Code Optimization?
  - Program analysis, machine dependent and independent optimization.

What you know

- Lexer ?
  - lex, flex
- Parser?
  - yacc, bison, antlr
- IR?
  - SSA, AST
- Code Optimization?
  - Program analysis, machine dependent and independent optimization.
- Code Generation?
  - Register Allocation, Instruction Selection.

Why Program Analysis?

Code fragment

```
main(){
    for(int i=0;i<SIZE;i++){
        .....
        if (i<0) printf( "System Crash");
        .....
    }
}
```

Question

The above program will run successfully?

Why Program Analysis?

Code fragment

```
main(){
    for(int i=0;i<SIZE;i++){
        .....
        if (i<0) printf( "System Crash");
        .....
    }
}
```

Question

The above program will run successfully?

Answer

only if $SIZE < 2^{(sizeof(int)*2-1)}$

Why Program Analysis?

- Program and Errors
  - Defects in programs are very common.

Why Program Analysis?

- Program and Errors
  - Defects in programs are very common.
  - Some defects are disastrous.

Why Program Analysis?

- Program and Errors
  - Defects in programs are very common.
  - Some defects are disastrous.
  - Some defects popup occassionaly.

Why Program Analysis?

- Program and Errors
  - Defects in programs are very common.
  - Some defects are disastrous.
  - Some defects popup occassionaly.
  - Year 2000 (Y2K) problem.

Why Program Analysis?

- Program and Errors
  - Defects in programs are very common.
  - Some defects are disastrous.
  - Some defects popup occassionaly.
  - Year 2000 (Y2K) problem.
  - In 1996 first flight of Arian 5 launcher failed after 40 seconds.
  - Many defects get detected after product launch.

- Software Testing

- Software Testing
  - Software industry spends good money to find bugs.

- Software Testing
  - Software industry spends good money to find bugs.
  - null pointer, uninitialized values, array index out of bounds etc are low level errors.

- Software Testing
  - Software industry spends good money to find bugs.
  - null pointer, uninitialized values, array index out of bounds etc are low level errors.
  - Do not satisfy user requirements

- Software Testing
  - Software industry spends good money to find bugs.
  - null pointer, uninitialized values, array index out of bounds etc are low level errors.
  - Do not satisfy user requirements
  - Algorithm or Design errors.

- Software Testing
  - Software industry spends good money to find bugs.
  - null pointer, uninitialized values, array index out of bounds etc are low level errors.
  - Do not satisfy user requirements
  - Algorithm or Design errors.
  - Poor Performance.

- LLVM installation.

- LLVM installation.
- Install GCC from Source.

Compiler Optimizations and Program Analysis

Unnikrishnan C

July 31, 2019