Master

Compiler 2016

Zihao Ye
ACM Class 2014

May 11, 2016

About name

Master Compiler was named after a role in Doctor Who: The Master.

Properties

It's a simple(slow) & naive(not efficient) implementation of a superset of Mx* in which member function/method is permitted.

Structure

Procedures

- Parser: From source code to AST.
- From AST to Linear IR.
- From Linear to CFG(with a subtle optimization).
- Register allocation.
- From IR to MIPS.

Front End

Parser

ANTLR: source code -> CST.

AST

Convert CST to AST:

Firstlistener: Initialize, and collect all class names.

Secondlistener: Collect all function names and arguments.

Thirdlistener: Construct the whole AST.

Front End

Parser

ANTLR: source code -> CST.

AST

Convert CST to AST:

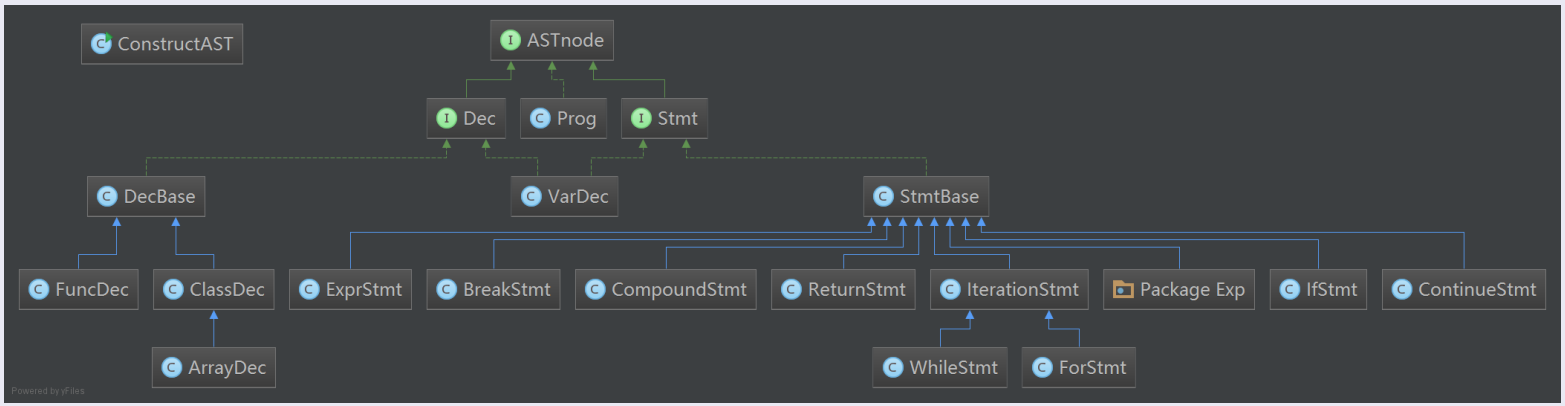
Firstlistener: Initialize, and collect all class names.

Secondlistener: Collect all function names and arguments.

Thirdlistener: Construct the whole AST.

Front End

Structure of AST



Front End

Some special AST nodes:

Type: ClassDec is used to distinguish different types.(Builtin classs such as int, bool, string are defined in the Utility.h)

Symbol: VarDec is used to distinguish different symbols.

Front End

Structure of Symbol Table

Symbol table(called Scope) is nested in AST nodes.

```
public class Scope {
    Scope previous = null;
    public Map<String, Dec> dict =
            new IdentityHashMap<>();

    public Scope(Scope previous) {
        this.previous = previous;
    }

    public Dec lookUp(String name) {
        Dec ret = dict.get(name);
        if (previous == null)
            return ret;
        if (ret == null)
            return previous.lookUp(name);
        else return ret;
    }

    public Dec lookUpInThisScope(String name) {
        return dict.get(name);
    }

    public void addEntry(String name, Dec symbol) {
        dict.put(name, symbol);
    }
}
```

Back End

Linear IR & CFG

- IR design: a three-address IR with infinite virtual registers(LLIR).
- Generate linear IR: Prog.emit();
- Convert linear IR to CFG: divide linear IR into basicblocks and collect all basicblocks in the same function together.

Back End

A trivial optimizations

- Since there are some basicblocks like:

```
__ifFalse0:
j __end0
```

combining these labels become necessary. By using the same method as union-find set, we can replace each label with the nearest one that really make sense.

Liveness Analysis & Register Allocation

Interference graph :

Iteration algorithm, instruction by instruction.

Coloring :

Greedy algorithm(color the vertex with minimum degree each step).

Liveness Analysis & Register Allocation

- $v0, $v1 are used to store temporary variables of Three-address code.
- $a0, $a1, $a2 are used as first 3 function arguments.
- Most of other machine registers($ra, $at, $0, exclusively)ar used in register allocation.
- MIPS instructions like 'move $r0, $r0' will be eliminated.

Example

Member function:

```
class Report {
    string reportname;
    void init(string name) {
        this.reportname = name;
    }
    void print() {
        println(this.reportname);
    }
}
int n = 100;

class Professor {
    Report report;
    string attitude;
    void init(Report report, int attitude) {
        this.report = report;
        if (attitude == 0) {
            this.attitude = "Accept!";
        }
        if (attitude == 1) {
            this.attitude = "Excited!";
        }
        if (attitude > 1) {
            this.attitude = "Sunny!";
        }
    }
    void print() {
        println(this.attitude);
    }
}

int main() {
    Report reportOfProfessorJiang = new Report;
    reportOfProfessorJiang.init("Trends in the Development of Energy Resources and Major Approaches to Their Economization.");
    reportOfProfessorJiang.print();
    Professor[] aHundredProfessorInSJTU = new Professor[n];
    int i;
    for (i = 0; i < 100; i++) {
        aHundredProfessorInSJTU[i] = new Professor;
        aHundredProfessorInSJTU[i].init(reportOfProfessorJiang, i % 3);
    }
    for (i = 0; i < 100; i++) {
        aHundredProfessorInSJTU[i].print();
    }
    return 0;
}
```

Example

Member function:

```
Trends in the Development of Energy Resources and Major Approaches to Their Economization.
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
Accept!
Excited!
Sunny!
```

Thanks

- I would like to thank all of our TAs for their hard work.
- I would like to thank Lequn Chen for his generous help & programs for unit test.
- I would like to thank Zhijian Liu for some of his suggestions.
- I would like to thank Tianyao Chen and Yurong You for their builtin functions.