04830180-编译实习

01. 课程介绍和概述

黄　骏

jun.huang@pku.edu.cn

高能效计算与应用中心

北京大学信息科学与技术学院

理科五号楼515S

Outline

- What is a compiler
- Compiler overview
- Why study compiler
- Course overview

Outline

- What is a compiler
- Compiler overview
- Why study compiler
- Course overview

What is a Compiler

- A computer program that translates

  a source code in a one *source programming language*

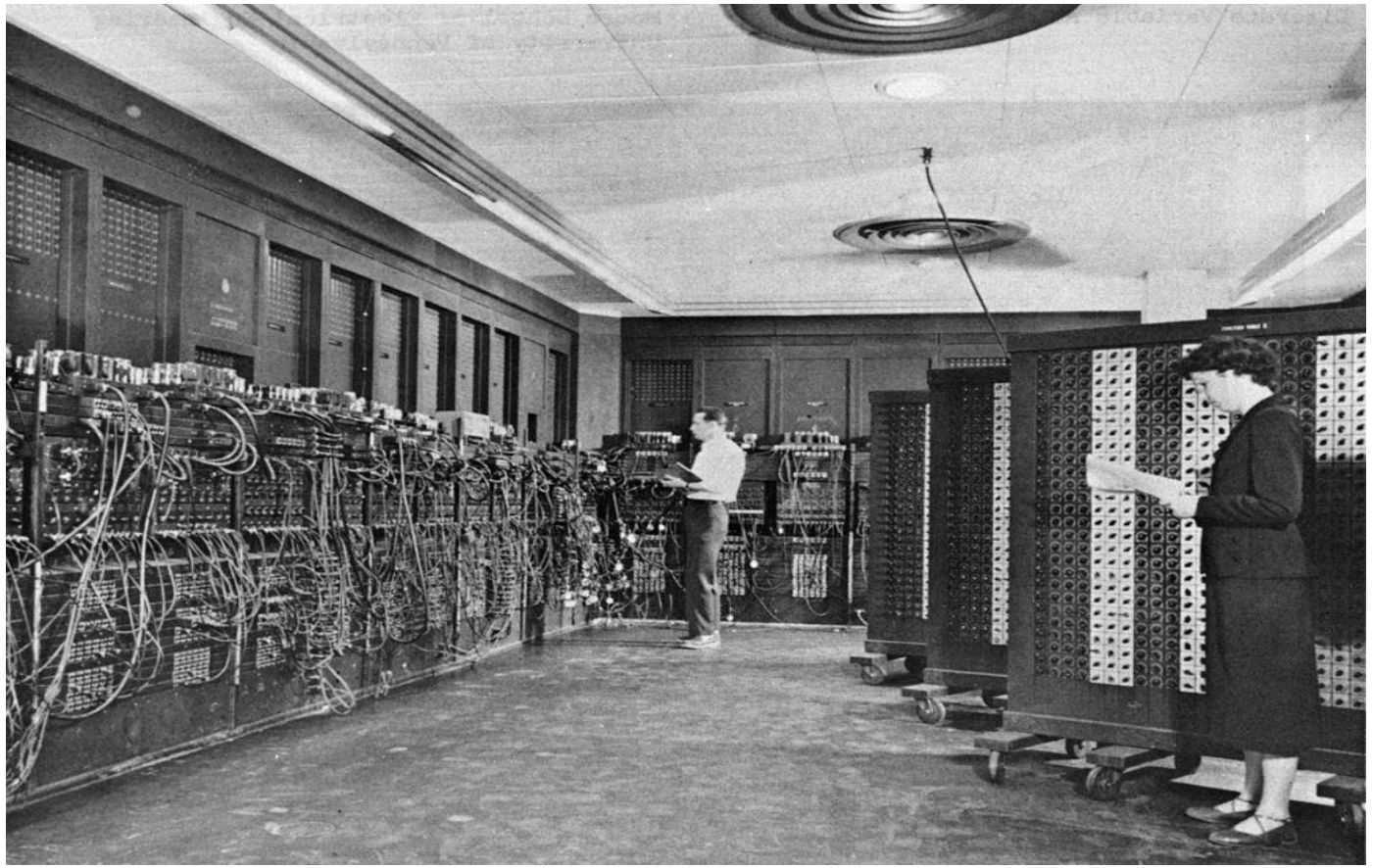
  into a *target programming language*

- The primary use is to translate

  a source code written in *high-level programming language*

  to a *lower level language* (e.g., assembly language or machine code)

What is a Compiler



世界上第一台图灵机 1946

1. Move left one location
2. Move right one location
3. Read in current location
4. Write 0 in current location
5. Write 1 in current location

Turing machine's view of computer programs

What is a Compiler

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4       //say hello
5       cout << "hello world" << endl;
6       system("PAUSE");
7       return 0;
8 }
```

Modern high-level programming languages

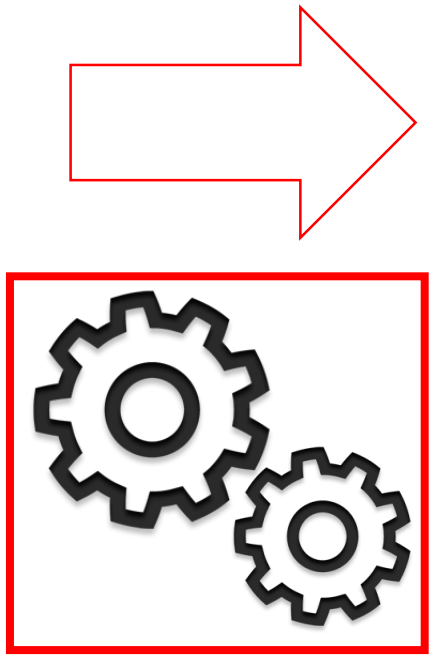
1. Move left one location
2. Move right one location
3. Read in current location
4. Write 0 in current location
5. Write 1 in current location

Turing machine's view of computer programs

What is a Compiler

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4       //say hello
5       cout << “hello world” << endl;
6       system(“PAUSE”);
7       return 0;
8 }
```

Modern high-level programming languages



Compiler

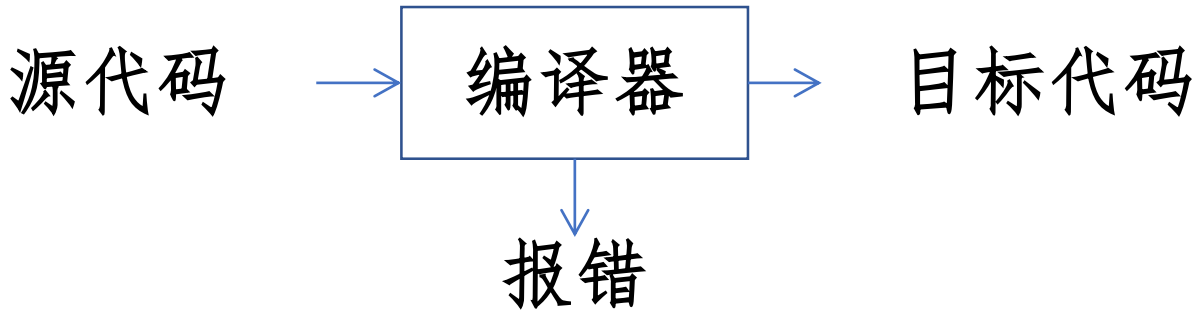
1. Move left one location
2. Move right one location
3. Read in current location
4. Write 0 in current location
5. Write 1 in current location

Turing machine's view of computer programs

Outline

- What is a compiler
- Compiler overview
- Why study compiler
- Course overview

抽象表示



- 识别正确代码
- 生成语义等价的目标代码
- 管理变量和代码的存储
- 报告错误信息

五个阶段

- 词法分析
- 语法分析
- 语义分析、中间代码生成
- 代码优化
- 生成机器代码

词法分析

- **从字符串到单词**

  扫描源程序，根据语言的**词法规则**将字符串识别为**单词**

单词：由词法规则定义的基本语法单位，包括：

(1) 关键字或保留字（if, while, for...）

(2) 标识符 （变量名、函数名等）

(3) 常数

(4) 运算符和分界符（+ - * / ; 等等）

词法分析

IAMTHEBOSS

- **从字符串到单词**

  扫描源程序，根据语言的**词法规则**将字符串识别为**单词**

单词：由词法规则定义的基本语法单位，包括：

(1) 关键字或保留字（if, while, for…）

(2) 标识符 （变量名、函数名等）

(3) 常数

(4) 运算符和分界符（+ - * / ; 等等）

词法分析

IAMTHEBOSS

- 从字符串到单词

  扫描源程序，根据语言的**词法规则**将字符串识别为**单词**

单词：由词法规则定义的基本语法单位，包括：

(1) 关键字或保留字（if, while, for...）

(2) 标识符 （变量名、函数名等）

(3) 常数

(4) 运算符和分界符（+ - * / ; 等等）

词法分析

IAMTHEBOSS

- 从字符串到单词

  扫描源程序，根据语言的**词法规则**将字符串识别为**单词**

词法规则：定义单词的结构和构成

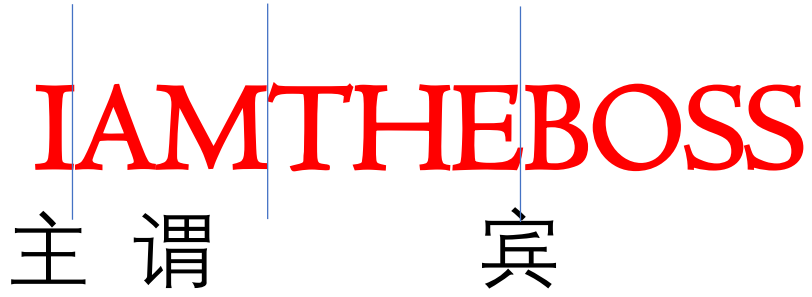
一般用正则表达式表示，比如:

letter ->(a|b|…|z|A|B|…|Z)

digit->(0|1|2|3|4|5|6|7|8|9)

id->letter(letter|digit)*

语法分析



- **从单词串到抽象语法树**

根据**语法规则**识别**语法成分**，如表达式、语句、函数等。

语法分析

IAMTHEBOSS

主 谓 宾

- 从单词串到抽象语法树

根据**语法规则**识别**语法成分**，如表达式、语句、函数等。

- 语法规则

<AssignStm> ::= <id><AssignOp><Expr>

<Expr> ::= <Expr><op><Expr>

| num

| id

a := b+2.3*4.5

<id><AsgOp><id><op><num><op><num>

语法分析

IAMTHEBOSS

主 谓 宾

- 从单词串到抽象语法树

根据**语法规则**识别**语法成分**，如表达式、语句、函数等。

- 语法规则

<AssignStm> ::= <id><AssignOp><Expr>

<Expr> ::= <Expr><op><Expr>

| num

| id

a := b+2.3*4.5

<id><AsgOp><id><op><num><op><num>

<id><AsgnOp><Exp><op><Exp>

语法分析

IAMTHEBOSS

主 谓 宾

- 从单词串到抽象语法树

根据**语法规则**识别**语法成分**，如表达式、语句、函数等。

- 语法规则

<AssignStm> ::= <id><AssignOp><Expr>

<Expr> ::= <Expr><op><Expr>

| num

| id

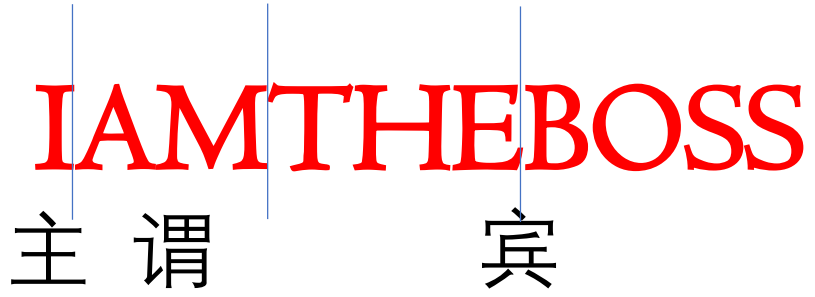
a := b+2.3*4.5

<id><AsgOp><id><op><num><op><num>

<id><AsgnOp><Exp><op><Exp>

<id><AsgnOp><Exp>

语法分析



- 从单词串到抽象语法树

根据**语法规则**识别**语法成分**，如表达式、语句、函数等。

- 语法规则

<AssignStm> ::= <id><AssignOp><Expr>

<Expr> ::= <Expr><op><Expr>

| num

| id

a := b+2.3*4.5

<id><AsgOp><id><op><num><op><num>

<id><AsgnOp><Exp><op><Exp>

<id><AsgnOp><Exp>

<AsgnStm>

语法分析

- 从单词串到抽象语法树

根据**语法规则**识别**语法成分**，如表达式、语句、函数等。

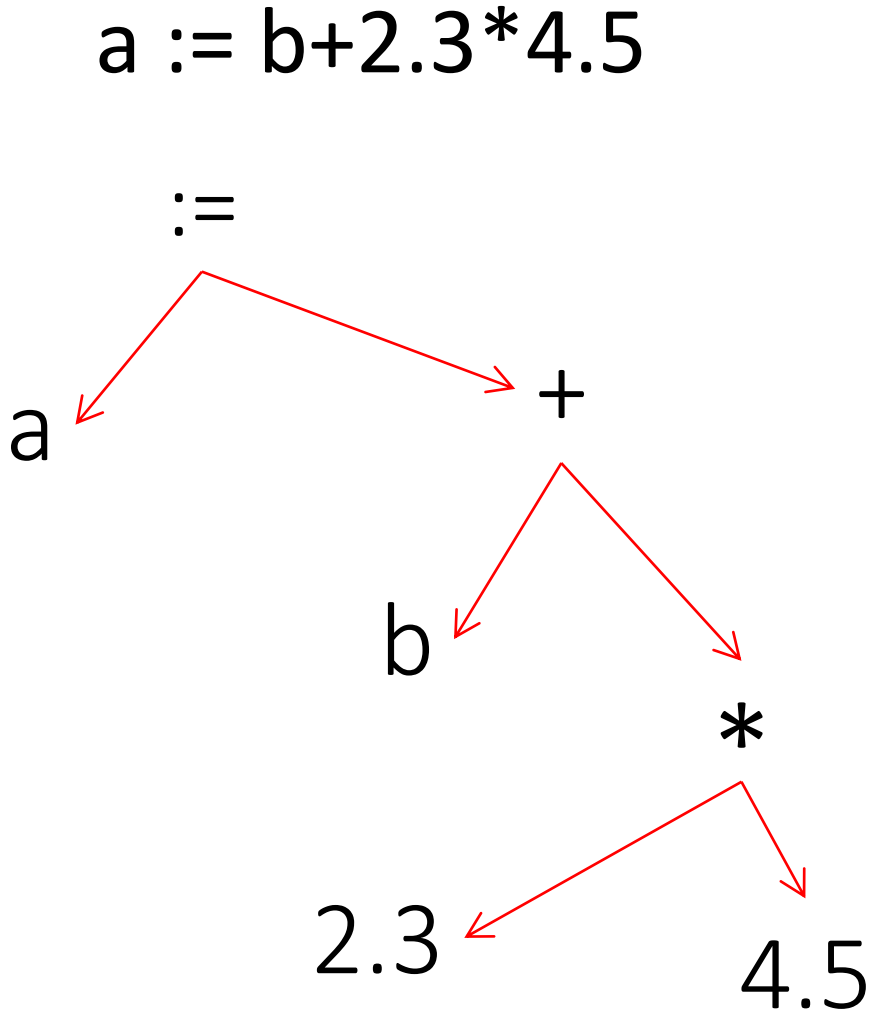
- 语法规则

<AssignStm> ::= <id><AssignOp><Expr>

<Expr> ::= <Expr><op><Expr>

| num

| id

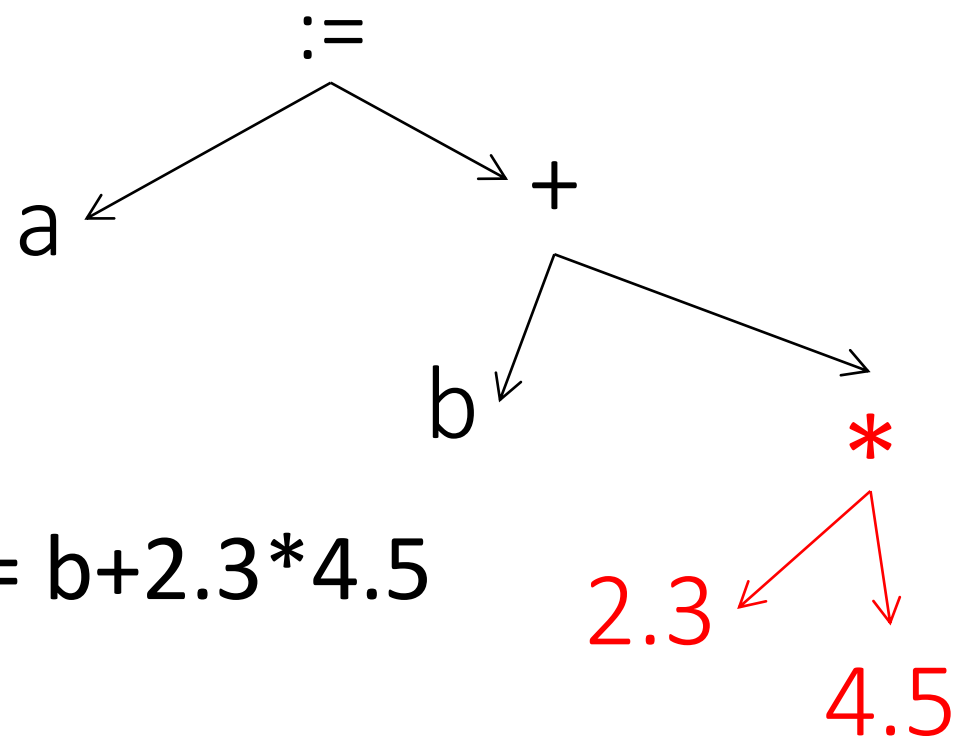


语义分析、中间代码生成

- 任务
  - 检查语义正确性（**类型检查**）: **She is a boy (主宾类型不匹配)**
  - 语义等价地翻译
- **中间代码：介于源语言和目标语言之间的中间形式**
  - 目的：便于优化和编译程序的移植
  - 形式：四元式（三地址指令）等

语义分析、中间代码生成

抽象语法树



a := b+2.3*4.5

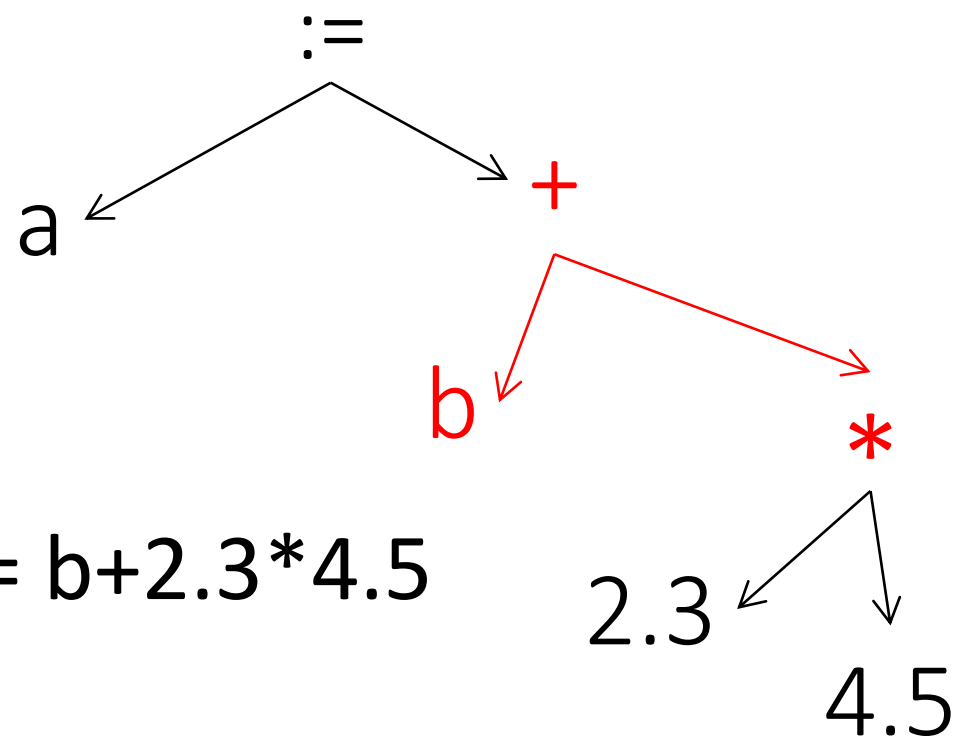
运算符	左运算对象	右运算对象	结果
*	2.3	4.5	T1

中间码的语义：

2.3*4.5 -> T1

语义分析、中间代码生成

抽象语法树



a := b+2.3*4.5

运算符	左运算对象	右运算对象	结果
*	2.3	4.5	T1
+	b	T1	T2

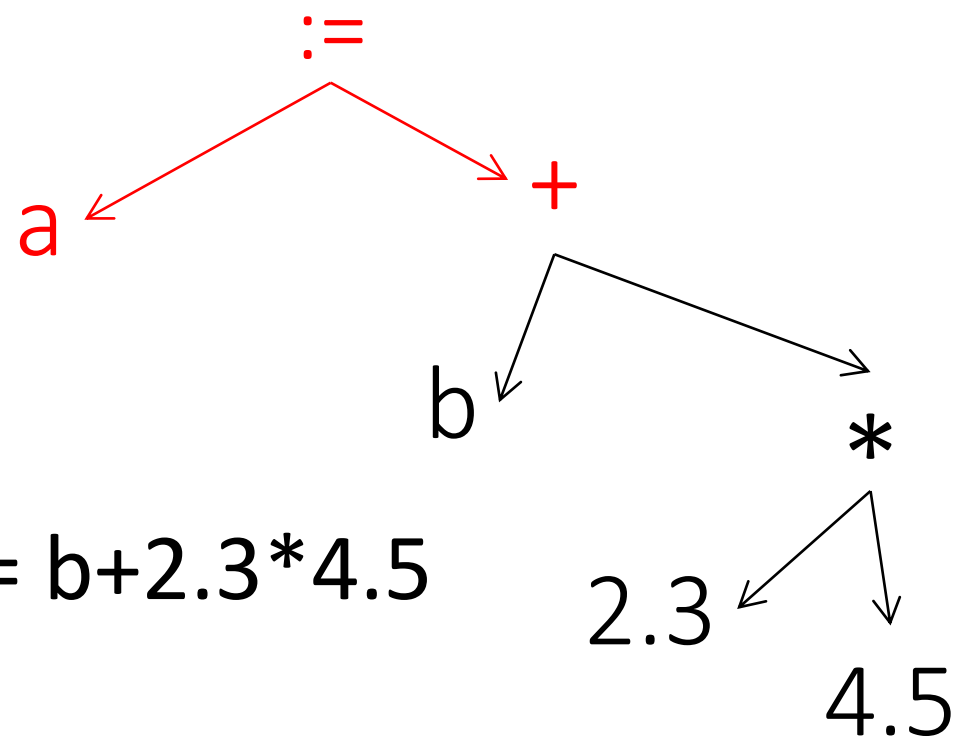
中间码的语义：

2.3*4.5 -> T1

b+T1 -> T2

语义分析、中间代码生成

抽象语法树



a := b+2.3*4.5

运算符	左运算对象	右运算对象	结果
*	2.3	4.5	T1
+	b	T1	T2
:=	a	T2	

中间码的语义：

2.3*4.5 -> T1

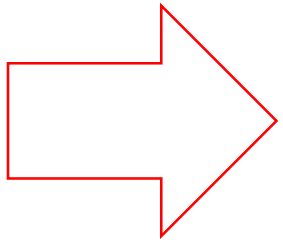
b+T1 -> T2

T2 -> a

生成机器代码

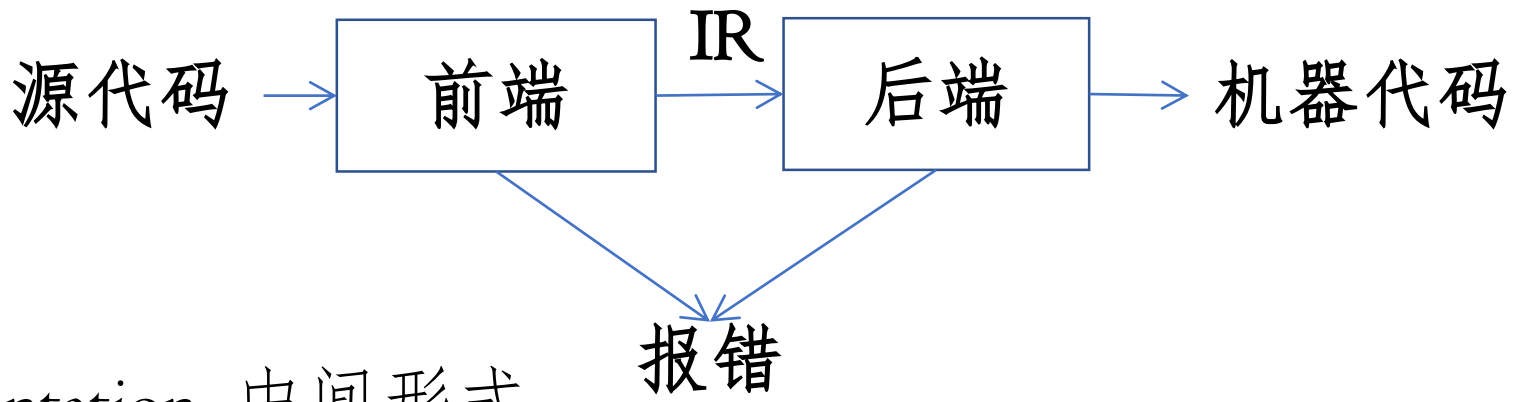
- 机器相关

运算符	左运算对象	右运算对象	结果
*	2.3	4.5	T1
+	999	T1	b



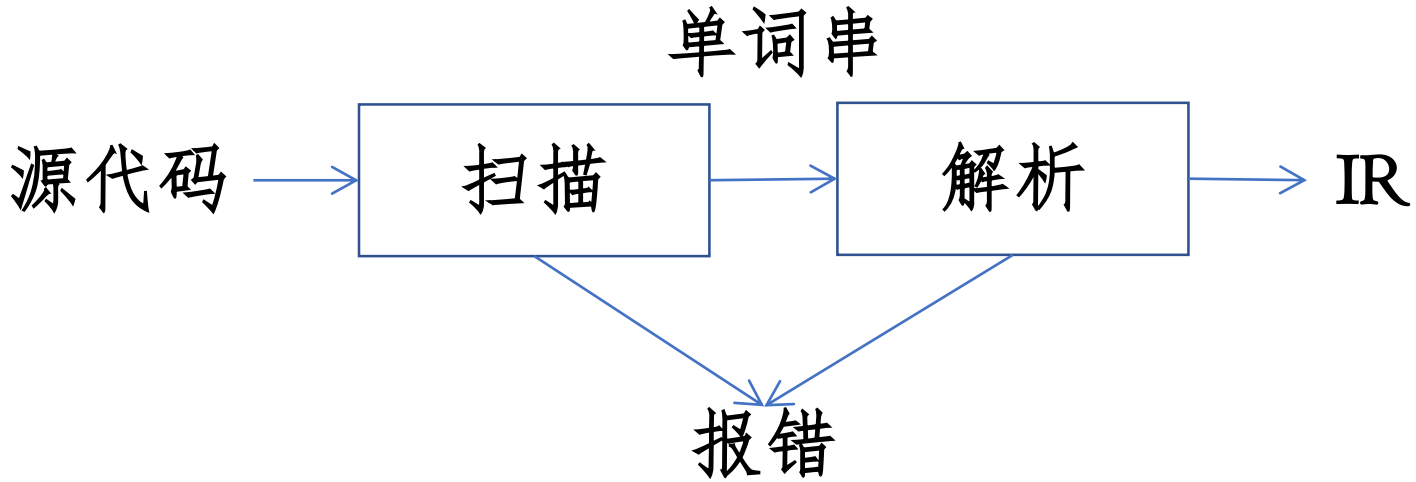
```
LOAD    2.3
MUL     4.5
STO     T1
LOAD    999
ADD     T1
STO     b
```

编译器的逻辑结构



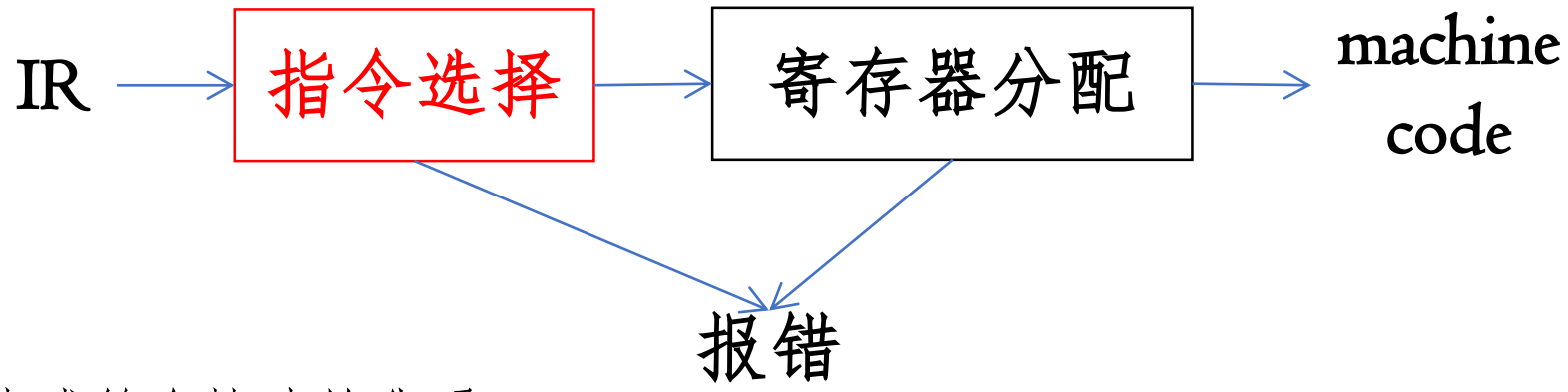
- IR： intermediate representation, 中间形式
- 前端将合法代码映射为IR
- 后端将IR映射到目标机器
- 简化编译程序的移植

前端



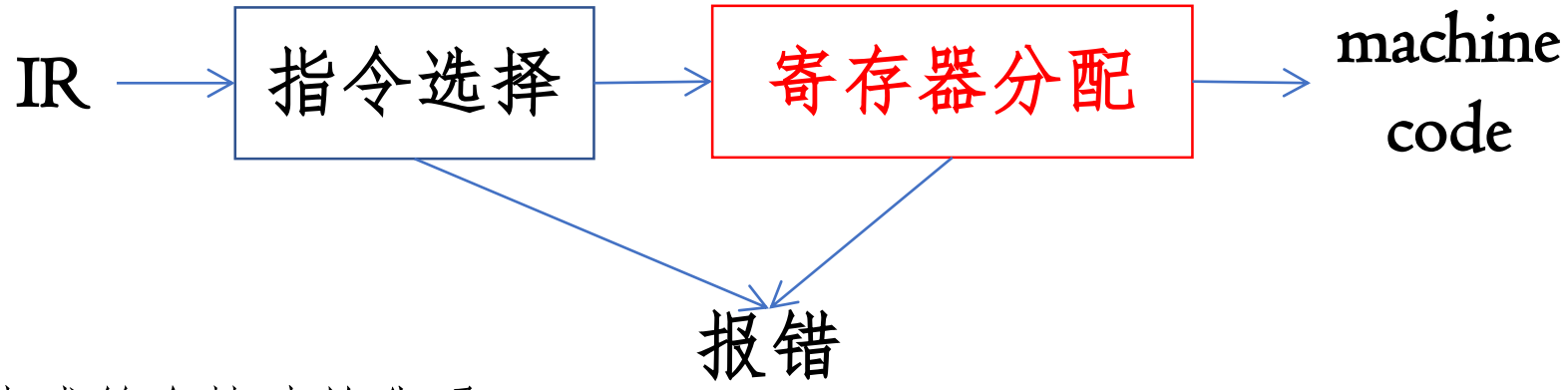
- 通常将与源程序有关的部分成为前端
  - 识别合法代码
  - 输出错误信息
  - 生成IR
  - 前端的构造已基本实现自动化

后端



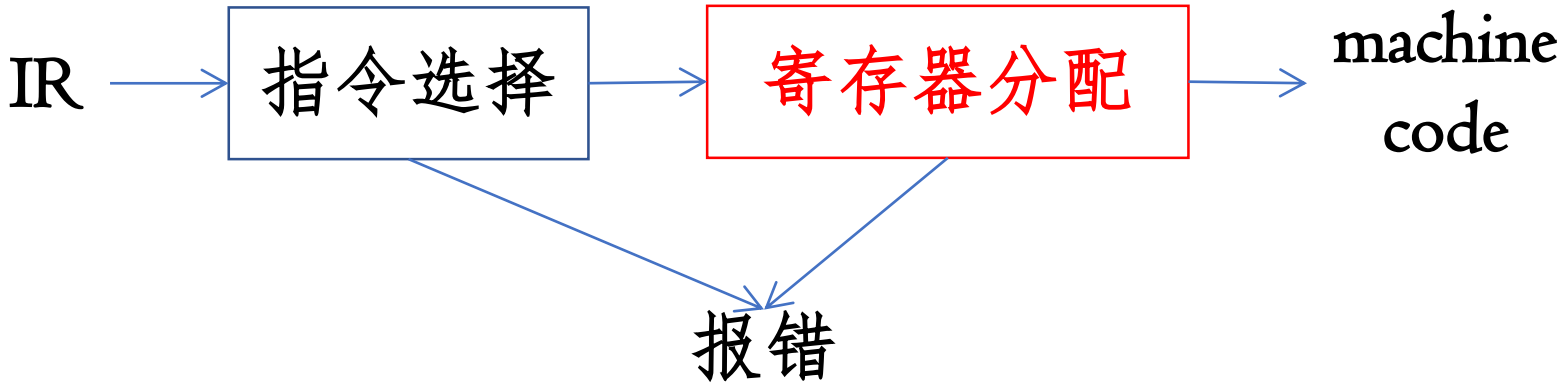
- 指令选择
  - 利用可用的寻址方式生成简介快速的代码

后端



- 指令选择
  - 利用可用的寻址方式生成简介快速的代码
- 寄存器分配
  - IR里临时变量的个数不受限制
  - 而目标机的寄存器是有限的

后端



- 指令选择
  - 利用可用的寻址方式生成简介快速的代码
- 寄存器分配
  - IR里临时变量的个数不受限制
  - 而目标机的寄存器是有限的
  - 活跃区间不同的临时变量可以被映射到同一个寄存器
  - 或将临时变量溢出至内存

```
for (int a = 0; a < 1000; a++)
        b += a + 2.3*4.5
for (int b = 0; b < 1000; b++)
        c += b ;
```

Outline

- What is a compiler
- Compiler overview
- Why study compiler
- Course overview

Why Study Compiler

- Compilers have been there for decades
  - isn't it a solved problem?
- Programming language is just a development tool
  - it's enough to stay at high-level
- I may never design a new programming language myself in my career
  - so why bothering?

Why Study Compiler

- **Compiler is a microcosm of computer science**

Artificial intelligence	Greedy algorithms, Learning algorithms
Algorithms	Graph algorithms, Union find, Dynamic programming
Theory	DFAs, parser generators, lattice theory for analysis
Systems	Allocation and naming, locality, synchronization
Architecture	Pipeline management, hierarchy management, instruction set use

Why Study Compiler

- **Compilation is a challenging and critical problem**
  - machines are constantly changing
  - changes in architecture require changes in compilers
    - new features bring new challenges
  - compilers have primary responsibility for performance

Outline

- What is a compiler
- Why study compiler
- Compiler overview
- Course overview

课程目标

- 1-2人一组，用Java编写一个小型的MiniJava编译器
  - 输入：符合MiniJava语言规范的源程序

    MiniJava是Java的子集，不允许方法重载和嵌套类（不允许存在名字相同的方法,类中只能申明变量和方法），不支持注释...

  - 输出：能在MIPS模拟器SPIM上运行的目标代码

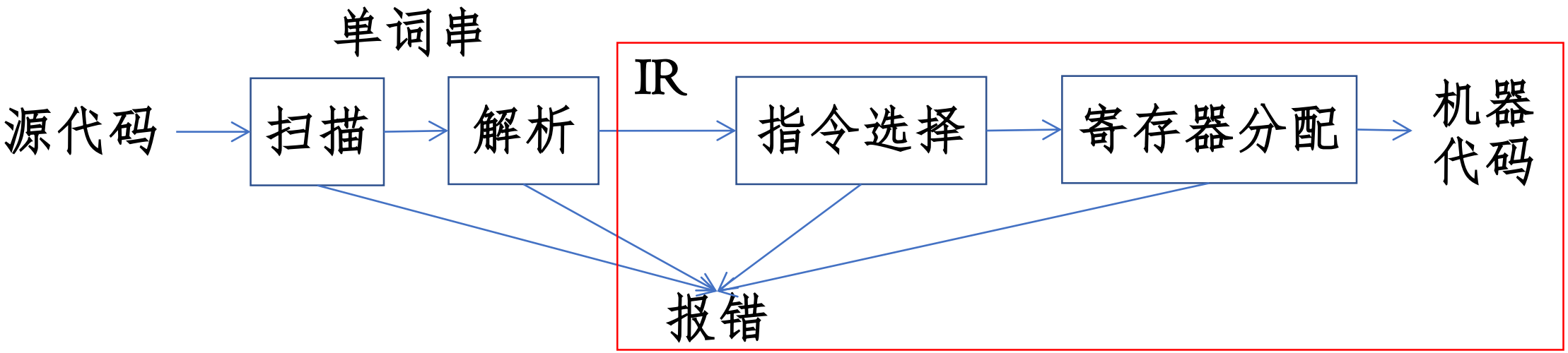
    SPIM的输入是MIPS汇编语言程序，而不是机器语言程序

- MiniC
  - 编译实习（实验班），
    - 刘先华老师，每周一7、8节，地学楼107

课程目标（续）

- 1-2人一组，实现一个小型的MiniJava编译器
  - 借助自动工具完成词法和语法分析
  - 主要完成以下关键步骤：

    类型检查，中间代码生成，寄存器分配，映射机器指令



成绩计算

- **5次平时作业: 75%**
  - 依据：准时提交，测试用例
  - TypeCheck: 15%
  - Minijava to Piglet: 25%
  - Piglet to sPiglet: 5%
  - sPiglet to Kanga: 15%
  - Kanga to MIPS: 15%
- **面测：15%**
  - 讲解代码和回答问题
- **期末报告：10%**
  - 文档和建议

Timeline

- 第1周：课程介绍、编译概述
- 第2周：MiniJava和Java
- 第3周：JavaCC和JTB
- 第4周：类型检查
- 第5、6周：答疑
- 第7周：MiniJava to Piglet (**类型检查due**)
- 第8、9、10周：答疑
- 第11周：Piglet to sPiglet (**MiniJava to Piglet due**)
- 第12周：sPiglet to Kanga (**Piglet to sPiglet due**)
- 第13、14周：答疑
- 第15周：Kanga to MIPS、系统的整合 (**sPiglet to Kanga due**)
- 第16周：答疑
- 第17、18周：面测 (**Kanga to MIPS due**)
- 6.29：**期末报告due**

辅导和检查方式

- **检查方式**
  - 电子方式提交作业源代码和程序，以及期末报告

- **辅导方式**
  - 论坛交流
  - 答疑和交流课
  - 课上作业点评
  - Office：周四和周五下午，理科五号楼515s

参考信息

http://web.cs.ucla.edu/~palsberg/course/cs132/project.html

其他材料：

工具使用样例

往届报告样例

From Java to MIPS in Five Nifty Steps

UCLA CS 132 Project

Jens Palsberg

The Hundred Hour Wood

MiniJava	grammar	minijava.jj	specification	[javac+java]	programs	Factorial.java
Piglet	grammar	piglet.jj	specification	interpreter	programs	Factorial.pg
Spiglet	grammar	spiglet.jj	specification	interpreter	programs	Factorial.spg
Kanga	grammar	kanga.jj	specification	interpreter	programs	Factorial.kg
MIPS	grammar	mips.jj	specification	SPIM	programs	Factorial.s