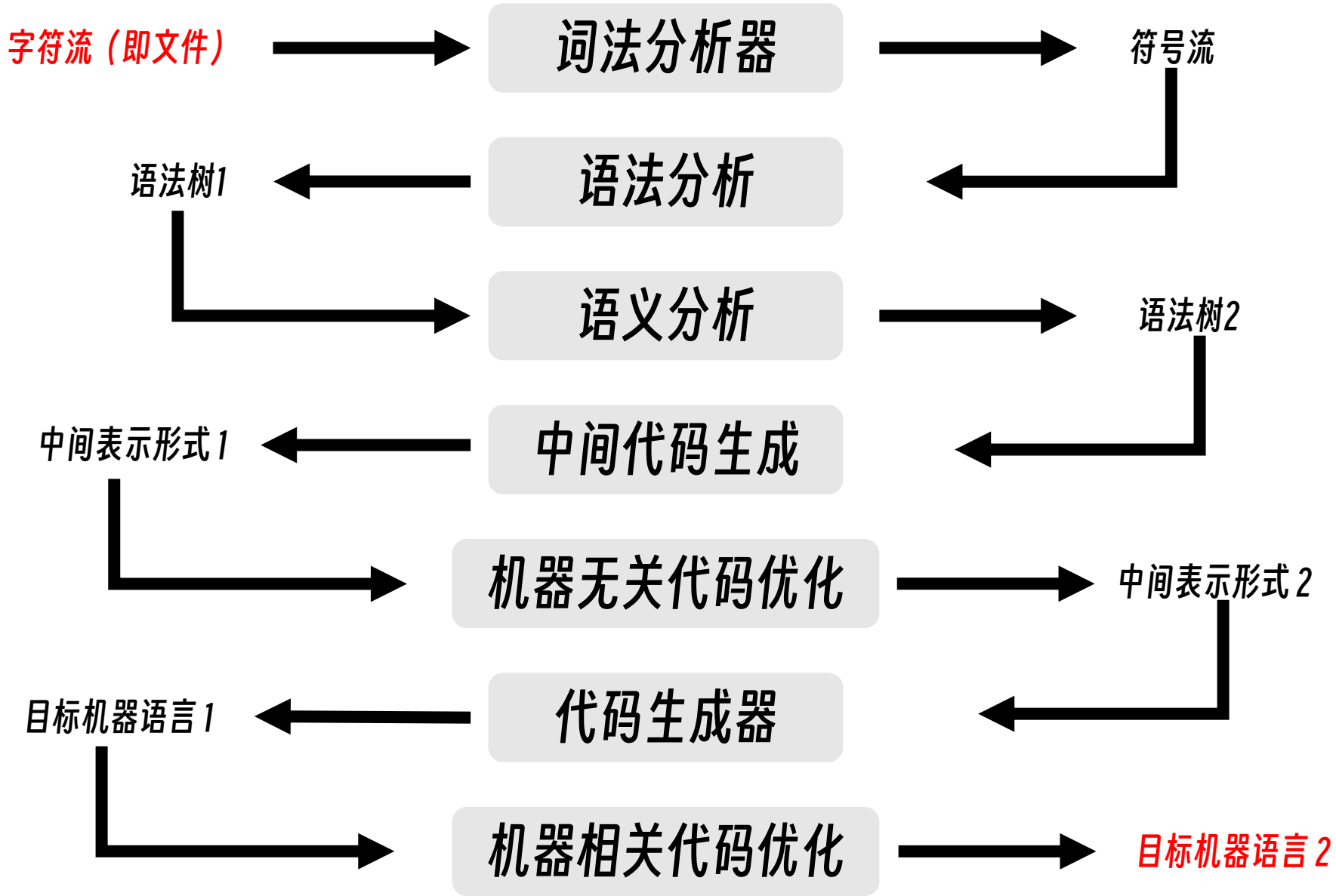
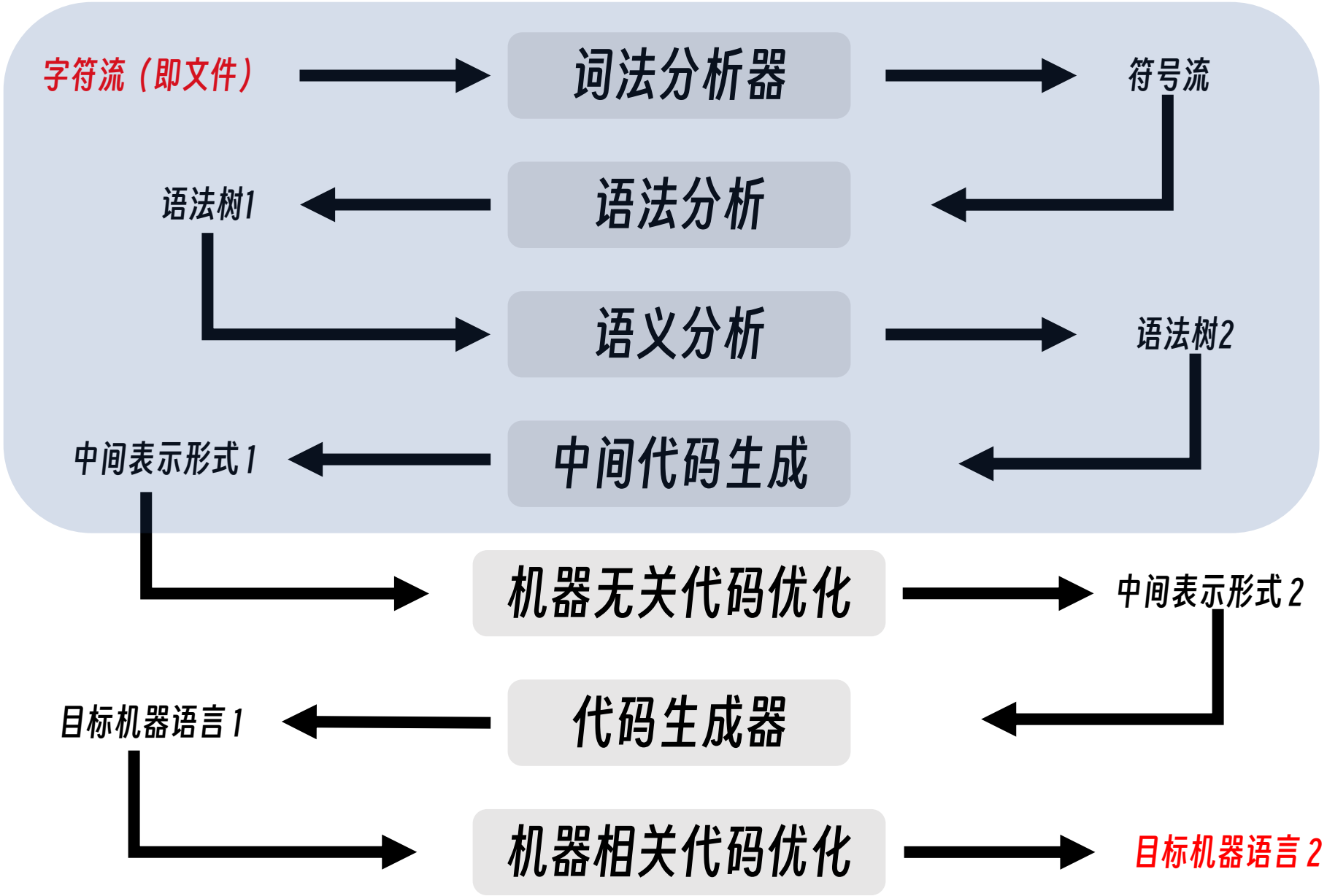
初步了解编译

汇报人：皮昊旋


字符流（即文件）
词法分析器
符号流
语法树1
语法分析
语义分析
语法树2
中间表示形式1
中间代码生成
机器无关代码优化
中间表示形式2
目标机器语言1
代码生成器
机器相关代码优化
目标机器语言2


字符流（即文件）
词法分析器
符号流
语法树1
语法分析
语义分析
语法树2
中间表示形式1
中间代码生成
机器无关代码优化
中间表示形式2
目标机器语言1
代码生成器
机器相关代码优化
目标机器语言2

分析部分

综合部分

词法分析

对读入的字符流，将其组织生成有意义的词素
每个词素（token）又可以以词法单元的格式输出，即<token-name, attribute-value>

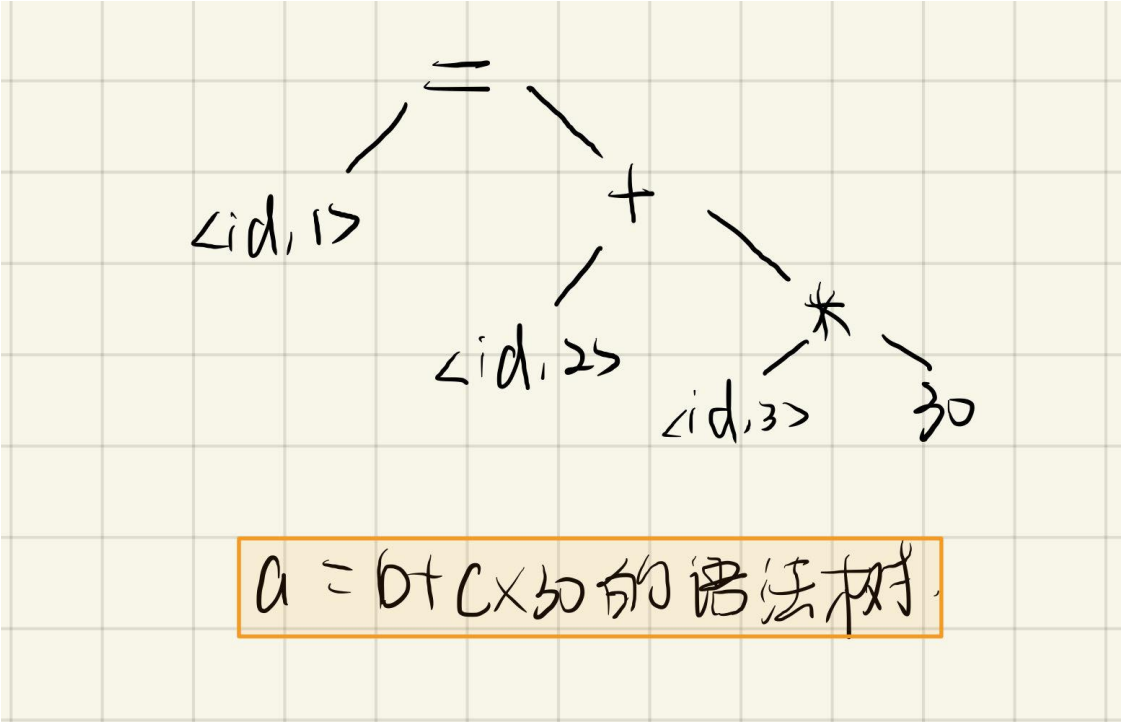
$$a = b + c * 30$$

<id,1> <=> <id,2> <+> <id,3> <*> <number,4>

语法分析

符号流 → **语法分析** → 语法树1

<id,1> <=> <id,2> <+> <id,3> <*> <number,4>

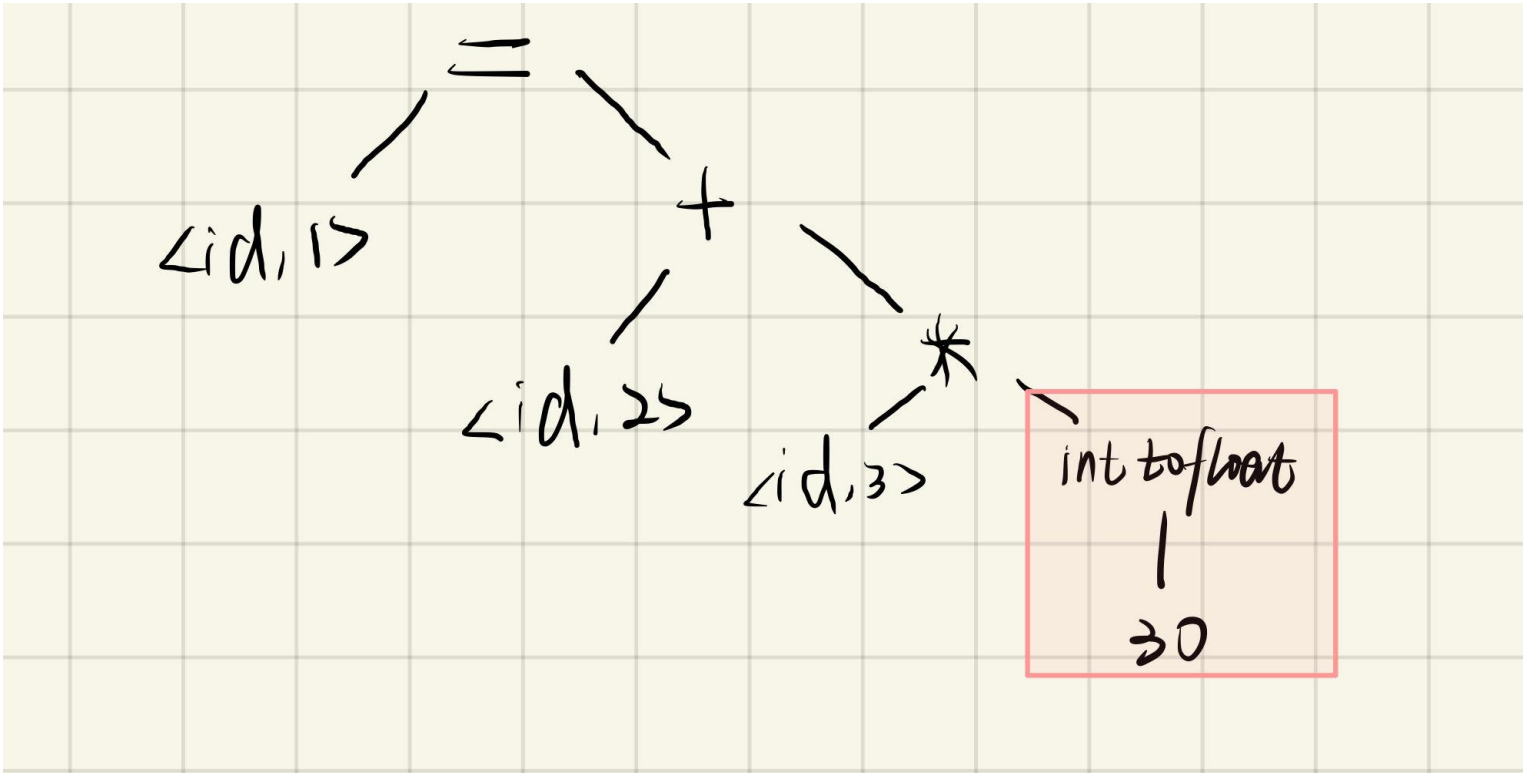


语义分析

语法树1 → **语义分析** → 语法树2

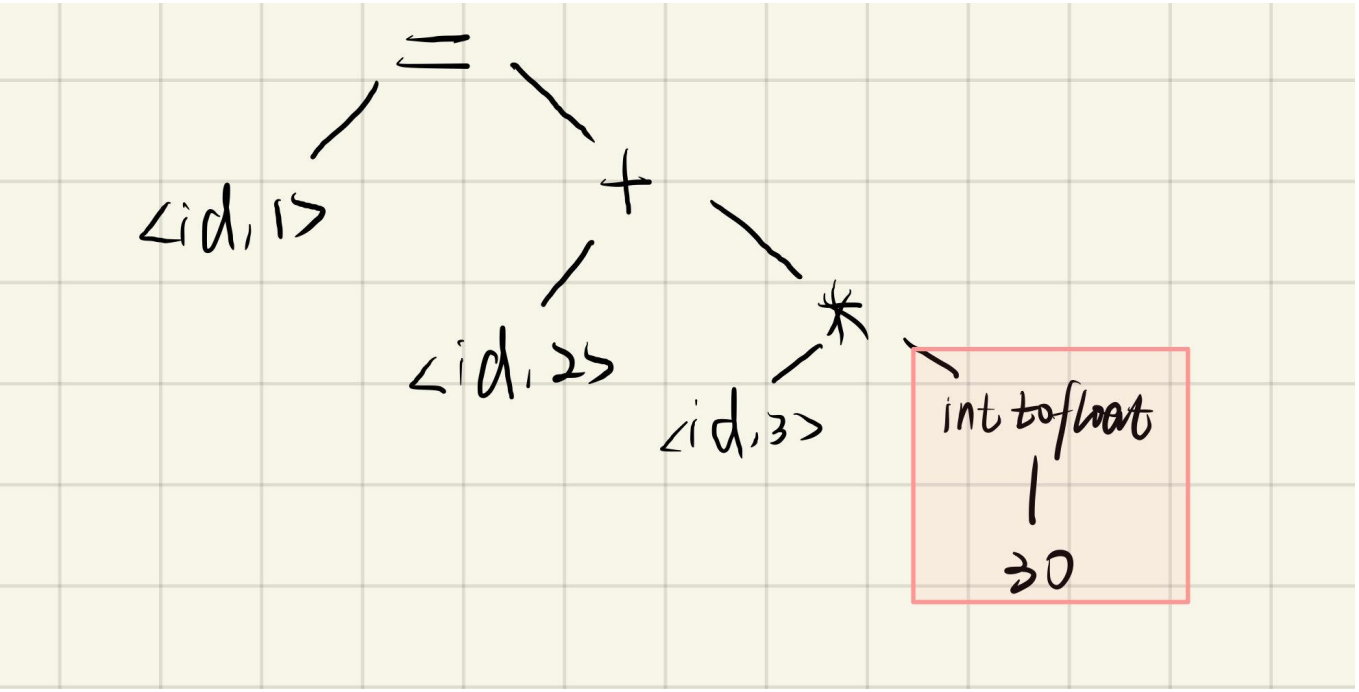
$$a = b + c * 30$$

<id,1> <=> <id,2> <+> <id,3> <*> <number,4>



中间代码生成

语法树2 → **中间代码生成** → 中间表示形式1




```
t1 = inttofloat(number)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

中间代码生成

为了保证代码的正确性

并且易于翻译成目标机器语言

大多采用三地址代码的方式表示中间代码

三地址代码

result = arg1 op arg2

t1 = inttofloat(number)
*t2 = id3 * t1*
t3 = id2 + t2
id1 = t3

代码优化


```
t1 = inttofloat(number)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

```
t1 = id3 * inttofloat(number)
id1 = id2+t1
```

注意：优化不能改变三地址代码的规则，即赋值运算符的右侧只能有一个操作符，因此并不能将其合并成一个语句

代码生成
代码优化

中间表示形式2 → 代码生成器 → 目标机器语言2

```
t1 = id3 * inttofloat(number)
id1 = id2+t1
```

→

```
LDF    R2, id3
MULF   R2, R2, #30.0
LDF    R1, id2
ADDF   R1, R2
STF    id1, R1
```

不会编译的编译器的4+1个函数

Next()函数--分析部分

Program()函数--分析部分

Expression()函数--分析部分

Eval()函数--综合部分

Main()函数

Next()函数
用于词法分析，逐字符读取文件（字符流）内容

Next()函数 - - 分析部分

Program()函数 - - 分析部分

Expression()函数 - - 分析部分

Eval()函数 - - 综合部分

```
int token;
char *src;

void next()
{
    token = *src++;
    return;
}
```

Next()函数 - - 分析部分

Program()函数 - - 分析部分

Expression()函数 - - 分析部分

Eval()函数 - - 综合部分

program()函数
语法分析的入口，分析整个程序

```
void program()
{
    next(); // get next token
    while (token > 0)//不能识别中文
    {
        printf("token is: %c\n", token);
        next();
    }
}
```

Next( )函数 - - 分析部分

Program( )函数 - - 分析部分

Expression( )函数 - - 分析部分

Eval( )函数 - - 综合部分

expression( )函数
解析表达式，可理解为语义分析

expr -> id = arith_expr

arith_expr -> term arith_expr_tail

arith_expr_tail -> + term arith_expr_tail | - term ari

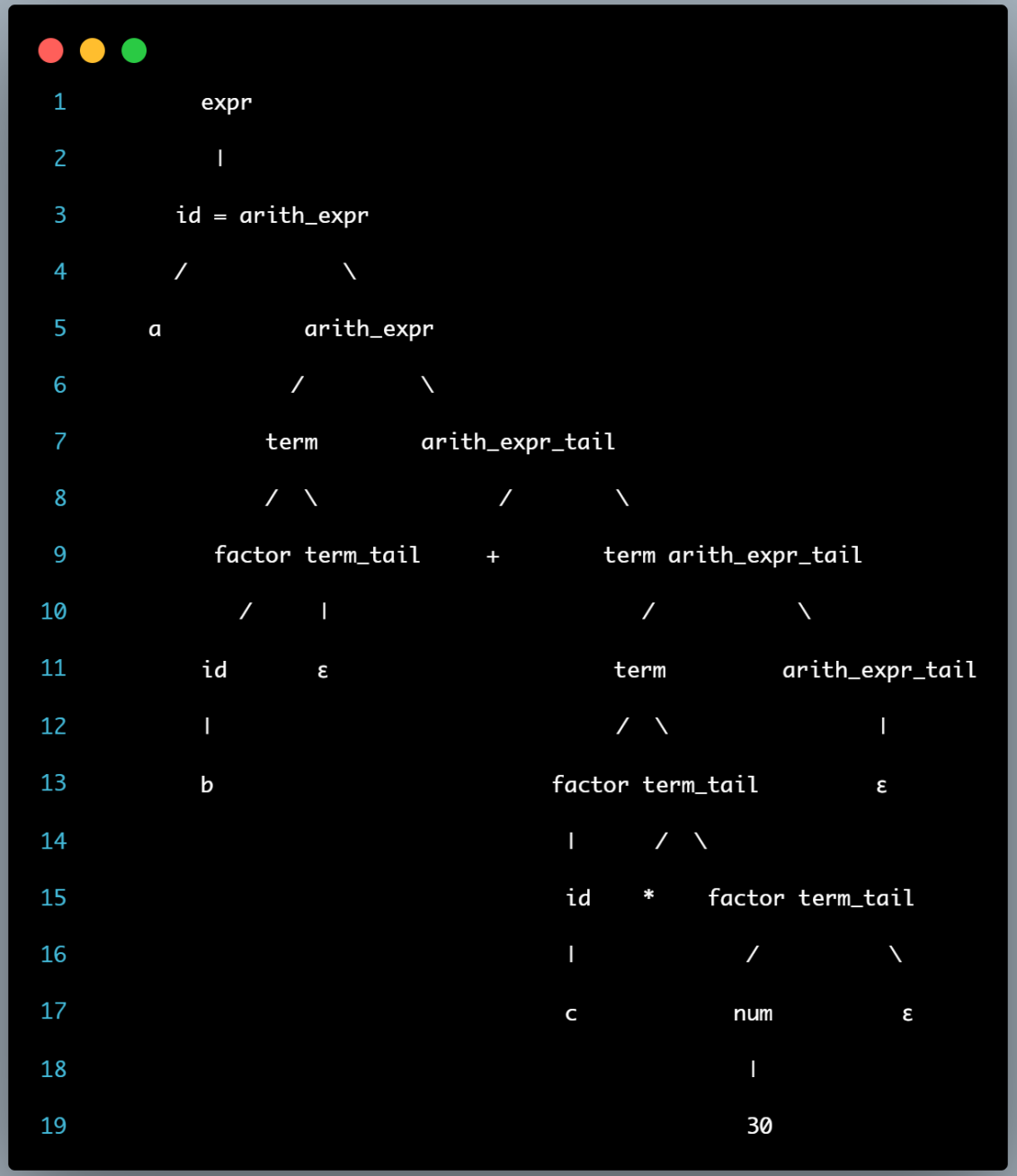
term -> factor term_tail

*term_tail -> * factor term_tail | / factor term_tail |*

factor -> id | number | (arith_expr)

id -> a | b | c

number -> 30



Next()函数 - - 分析部分

Program()函数 - - 分析部分

Expression()函数 - - 分析部分

Eval()函数 - - 综合部分

```
1  else if (op == GT)  ax = *sp++ >  ax;
2  else if (op == GE)  ax = *sp++ >= ax;
3  else if (op == SHL) ax = *sp++ << ax;
4  else if (op == SHR) ax = *sp++ >> ax;
5  else if (op == ADD) ax = *sp++ + ax;
6  else if (op == SUB) ax = *sp++ - ax;
7  else if (op == MUL) ax = *sp++ * ax;
8  else if (op == DIV) ax = *sp++ / ax;
9  else if (op == MOD) ax = *sp++ % ax;
```

Main()函数

词法分析

语法分析

语义分析（表达式解析）

目标机器代码生成

Main()函数

准备工作

①读入指令，用main函数中的内置参数

②对指令进行处理

③为编译器分配空间

④打开文件并分配空间以读取文件

⑤读入文件到src地址，封口，关闭文件

⑥进行语法分析并返回虚拟机，进行语义分析，生成机器语言

Thank you