



南开大学
Nankai University

南 开 大 学

计 算 机 学 院

编译原理实验报告

定义你的编译器、汇编编程

曹倚飞 2011745

王禹曦 2011920

年级：2020 级

专业：计算机科学与技术

指导教师：王刚

2022 年 10 月 13 日

摘要

本小组根据所选择的 sysY 语言特性,使用 CFG 进行文法设计,描述 sysY 语言子集,设计几个 sysY 程序来体现所选择的语言特性,通过编写等价的 ARM 汇编代码,用汇编器生成可执行程序、调试通过、运行得到正确的结果。

关键字: sysY 语言 CFG ARM 汇编编程

目录

一、 定义编译器	1
(一) sysY 语言特性选择	1
(二) 任务分工	1
(三) CFG 设计	1
1. sysY 语言终结符集合 V_T	1
2. sysY 语言非终结符集合 V_N	2
3. 开始符号 S	2
4. 产生式 P	3
二、 ARM 汇编编程	5
(一) 支持的语言特性	5
(二) 任务分工	6
(三) 设计 sysY 程序	6
(四) 编写 ARM 汇编代码	6
(五) 结果验证	9
1. 生成可执行程序	9
2. 执行并验证	9
三、 总结	9

一、 定义编译器

上下文无关文法（CFG）是一种用于描述程序设计语言语法的表示方式，有四个元素组成：

- 终结符集合 V_T ：有时也称为“词法单元”，终结符号是该文法所定义的语言的基本符号的集合；
- 非终结符集合 V_N ：有时也称为“语法变量”。每个非终结符号表示一个终结符号串的集合；
- 开始符号 S ：指定一个非终结符号为开始符号；
- 产生式 P ：包括一个成为产生式头部的非终结符号，一个箭头，一个成为产生式右部的由终结符及非终极符组成的序列，主要用来表示某个语法构造的某种书写形式。

因此，我们可以使用上下文无关文法对所选择的 sysY 语言特性进行描述。

（一） sysY 语言特性选择

经过小组讨论并结合实际情况，为尽可能多地体现 sysY 语言特性，我们选择要设计的语法包括：标识符、数值常量、编译单元、变量、常量、函数、语句、表达式等等。

（二） 任务分工

根据所选择的内容，合理分配任务：

曹倚飞负责：数值常量、变量、函数、表达式的 CFG 设计；

王禹曦负责：标识符、常量、编译单元、语句的 CFG 设计。

（三） CFG 设计

1. sysY 语言终结符集合 V_T

标识符 (identifier):

$$\begin{aligned}
 identifier &\rightarrow identifier_nondigit \\
 &\quad | \quad identifier \ identifier_nondigit \\
 &\quad | \quad identifier \ identifier_digit \\
 identifier_nondigit &\rightarrow _ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z \\
 identifier_digit &\rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
 \end{aligned}$$

数值常量 (IntConst):

```

integer_const  -> decimal_const
                |  octal_const
                |  hexadecimal_const
decimal_const  -> nonzero_digit
                |  decimal_const digit
octal_const    -> 0|octal_const octal_digit
hexadecimal_const -> hexadecimal_prefix hexadecimal_digit
                |  hexadecimal_const hexadecimal_digit
hexadecimal_prefix -> 0x|0X
nonzero_digit   -> 1|2|3|4|5|6|7|8|9
octal_digit     -> 0|1|2|3|4|5|6|7
hexadecimal_digit -> 0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f|A|B|C|D|E|F

```

2. sysY 语言非终结符集合 V_N

编译单元	CompUnit	表达式	Exp
声明	Decl	条件表达式	Cond
常量声明	ConstDecl	左值表达式	LVal
基本类型	BType	基本表达式	PrimaryExp
常数定义	ConstDef	数值	Number
常量初值	ConstInitVal	一元表达式	UnaryExp
变量声明	VarDecl	单目运算符	UnaryOp
变量定义	VarDef	函数实参表	FuncRParams
变量初值	InitVal	乘除模表达式	MulExp
函数定义	FuncDef	加减表达式	AddExp
函数类型	FuncType	关系表达式	RelExp
函数形参表	FuncFParams	相等性表达式	EqExp
函数形参	FuncFParam	逻辑与表达式	LAndExp
语句块	Block	逻辑或表达式	LOrExp
语句块项	BlockItem	常量表达式	ConstExp
语句	Stmt		

3. 开始符号 S

编译单元 CompUnit

4. 产生式 P

根据 sysY 语言特性分别给出该语言各部分的产生式：

(1) 编译单元

编译单元	$CompUnit \rightarrow CompUnit\ Decl$
	$\quad \quad \quad \quad CompUnit\ FuncDef$
	$\quad \quad \quad \quad Decl$
	$\quad \quad \quad \quad FuncDef$
声明	$Decl \rightarrow ConstDecl;$
	$\quad \quad \quad \quad VarDecl;$

(2) 常量、变量

常量声明	$ConstDecl \rightarrow \mathbf{const}\ BType\ ConstDef$
	$\quad \quad \quad \quad ConstDecl\ ','\ ConstDef$
基本类型	$BType \rightarrow \mathbf{int}$
常数定义	$ConstDef \rightarrow Ident\{'[ConstExp]'\} = ConstInitVal$
常数初值	$ConstInitVal \rightarrow ConstExp$
	$\quad \quad \quad \quad \{'\}\}$
	$\quad \quad \quad \quad \{'InitValList'\}$
变量声明	$VarDecl \rightarrow BType\ VarDef VarDecl\ ','\ VarDef$
变量定义	$VarDef \rightarrow Ident\{'[ConstExp]'\}$
	$\quad \quad \quad \quad Ident\{'[ConstExp]'\} = InitVal$
变量初值	$InitVal \rightarrow Exp$
	$\quad \quad \quad \quad \{'\}\}$
	$\quad \quad \quad \quad \{'InitValList'\}$
	$InitValList \rightarrow ConstExp InitValList, ConstExp$

(3) 函数

函数定义	$FuncDef \rightarrow FuncType\ Ident\('')'\ Block$
	$\quad \quad \quad \quad FuncType\ Ident\('FuncFParams')'\ Block$
函数类型	$FuncType \rightarrow \mathbf{void} int$
函数形参表	$FuncFParams \rightarrow FuncFParam FuncFParams, FuncFParam$
函数形参	$FuncFParam \rightarrow BType\ Ident$
	$\quad \quad \quad \quad BType\ Ident\('')'\$
	$\quad \quad \quad \quad FuncFParam\ '['\ Exp'\]$

(4) 语句

语句块	$Block$	\rightarrow	$\{\{BlockItem\}\}$
语句块项	$BlockItem$	\rightarrow	$Decl Stmt$
语句	$Stmt$	\rightarrow	$;\mid LVal' = ' Exp';'$ $Exp';'$ $Block$ $\mathbf{if}(Cond) \quad Stmt$ $\mathbf{if}(Cond) \quad Stmt \quad \mathbf{else} \quad Stmt$ $Block$

(5) 表达式

表达式	$Exp \rightarrow AddExp$
条件表达式	$Cond \rightarrow LOrExp$
左值表达式	$LVal \rightarrow Ident LVal['Exp']$
基本表达式	$PrimaryExp \rightarrow '('Exp') LVal Number$
数值	$Number \rightarrow IntConst$
一元表达式	$UnaryExp \rightarrow PrimaryExp$ $ Ident('')$ $ Ident('FuncRParams')$ $ UnaryOp \quad UnaryExp$
单目运算符	$UnaryExp \rightarrow + - !$
函数实参表	$FuncRParams \rightarrow Exp FuncRParams','Exp$
乘除模表达式	$MulExp \rightarrow UnaryExp$ $ MulExp*UnaryExp$ $ MulExp/UnaryExp$ $ MulExp\%UnaryExp$
加减表达式	$AddExp \rightarrow MulExp$ $ AddExp+MulExp$ $ AddExp-MulExp$
关系表达式	$RelExp \rightarrow AddExp$ $ RelExp<AddExp$ $ RelExp>AddExp$ $ RelExp\leq AddExp$ $ RelExp\geq AddExp$
相等性表达式	$EqExp \rightarrow RelExp$ $ EqExp==RelExp$ $ EqExp!=RelExp$
逻辑与表达式	$LAndExp \rightarrow EqExp LAndExp\&\&EqExp$
逻辑或表达式	$LOrExp \rightarrow LAndExp LOrExp LAndExp$
常量表达式	$ConstExp \rightarrow AddExp$

二、 ARM 汇编编程

(一) 支持的语言特性

经过小组讨论并结合实际情况，为尽可能多地体现 sysY 语言特性，我们设计的程序包括：int 型变量、一维数组、变量声明，用到赋值语句、表达式语句（基本算术运算、关系运算）、if 语句、while 语句、return 语句、函数调用（有返回值）等。

(二) 任务分工

根据所选择的内容，合理分配任务：

曹倚飞负责：int 型变量、变量声明、表达式语句、while 语句、return 语句等。

王禹曦负责：一维数组、赋值语句、if 语句、return 语句、有返回值函数调用等。

(三) 设计 sysY 程序

sysY 语言

```
1 #include<stdio.h>
2
3 int arrMax(int arr[5])
4 {
5     int max = arr[0];
6     int i = 0;
7     while(i<5)
8     {
9         if(arr[i]>max)
10        {
11            max = arr[i];
12        }
13        i = i + 1;
14    }
15
16    return max;
17 }
18 int main()
19 {
20     int arr[5] = {2,1,4,5,3};
21     int max = arrMax(arr);
22     printf("数组中最大值为:%d\n",max);
23
24     return 0;
25 }
```

(四) 编写 ARM 汇编代码

学习并编写上述 sysY 语言程序等价 ARM 汇编代码：

ARM 汇编代码

```
1 .text
2 .align 1
3 .global arrMax
4 .type arrMax, %function
5 arrMax:
6     push {r7} @ 保存r7
7     @ 扩展栈，r7赋值为sp
```



```

8      sub     sp, sp, #20
9      add     r7, sp, #0
10     @ 向栈中存数, 依次是arr,max,i
11     str     r0, [r7, #4]
12     ldr     r3, [r7, #4]
13     ldr     r3, [r3]
14     str     r3, [r7, #8]
15     movs    r3, #0
16     str     r3, [r7, #12]
17     b       .L2
18 .L4:    @ while循环体内
19     @ 执行if(arr[i]>max)
20     ldr     r3, [r7, #12]
21     lsls    r3, r3, #2
22     ldr     r2, [r7, #4]
23     add     r3, r3, r2
24     ldr     r3, [r3]          @ arr[i] => r3
25     ldr     r2, [r7, #8]      @ max => r2
26     cmp     r2, r3            @ max,arr[i] 比较
27     bge     .L3              @ max>=arr[i] 就 跳 .L3 (跳过 if
                               语句)
28     @ 条件成立; 执行max = arr[i];
29     ldr     r3, [r7, #12]
30     lsls    r3, r3, #2
31     ldr     r2, [r7, #4]
32     add     r3, r3, r2
33     ldr     r3, [r3]          @ arr[i] => r3
34     str     r3, [r7, #8]      @ r3 => max
35 .L3:
36     @ 条件不成立; 执行i++
37     ldr     r3, [r7, #12]
38     adds    r3, r3, #1
39     str     r3, [r7, #12]     @ 存入i
40 .L2:
41     @ 执行判断i<5?
42     ldr     r3, [r7, #12]
43     cmp     r3, #4
44     @ 继续循环, 跳到.L4
45     ble     .L4
46     @ 终止循环, 将max值返回
47     ldr     r3, [r7, #8]
48     mov     r0, r3
49     @ 恢复栈, r7的值
50     adds    r7, r7, #20
51     mov     sp, r7
52     ldr     r7, [sp], #4
53     bx      lr
54     .size   arrMax, .-arrMax

```

```

55      .section      .rodata
56      .align 2
57 .LC1:
58      .ascii  "\346\225\260\347\273\204\344\270\255\346\234\200\345"
59      .ascii  "\244\247\345\200\274\344\270\272:%d\012\000"
60      .align 2
61 .LC0:
62      .word 2
63      .word 1
64      .word 4
65      .word 5
66      .word 3
67      .text
68      .align 1
69      .global main
70      .syntax unified
71      .thumb
72      .thumb_func
73      .type main, %function
74 main:
75      push    {r4, r5, r7, lr}
76      @ 扩展栈, r7赋值为sp
77      sub     sp, sp, #32
78      add     r7, sp, #0
79      ldr     r2, .L9          @ r2 = GOT-(LPIC2+4)
80 .LPIC2:
81      add     r2, pc          @ r2 = GOT
82      mov     r3, #0
83      ldr     r3, .L9+4       @ r3 = .LC0-(.LPIC0+4)
84 .LPIC0:
85      add     r3, pc          @ r3 = .LC0
86      add     r4, r7, #8      @ r7+8 => r4    栈存储首地址
87      mov     r5, r3          @ r3 => r5        字符串首地址
88      @ .LC0对应arr[0-4]放入栈
89      ldmia   r5!, {r0, r1, r2, r3}
90      stmia   r4!, {r0, r1, r2, r3}
91      ldr     r3, [r5]
92      str     r3, [r4]
93
94      add     r3, r7, #8
95      mov     r0, r3
96      bl     arrMax(PLT)      @ PLT
97      str     r0, [r7, #4]
98      ldr     r1, [r7, #4]
99      ldr     r3, .L9+8       @ r3 = .LC1-(.LPIC1+4)
100 .LPIC1:
101      add     r3, pc          @ r3 = .LC1
102      mov     r0, r3          @ r0 = r3

```

```

103      bl      printf(PLT)
104      movs    r3, #0
105      ldr     r1, .L9+12          @ r1 = _GLOBAL_OFFSET_TABLE_-(.LPIC3
                                +4)
106  .LPIC3:
107      add     r1, pc              @ r1 = GOT
108      mov     r2, #0
109  .L8:
110      @ 恢复栈结构
111      mov     r0, r3
112      adds    r7, r7, #32
113      mov     sp, r7
114
115      pop     {r4, r5, r7, pc}
116  .L10:
117      .align  2
118  .L9:
119      .word   _GLOBAL_OFFSET_TABLE_-(.LPIC2+4)
120      .word   .LC0-(.LPIC0+4)
121      .word   .LC1-(.LPIC1+4)
122      .word   _GLOBAL_OFFSET_TABLE_-(.LPIC3+4)

```

(五) 结果验证

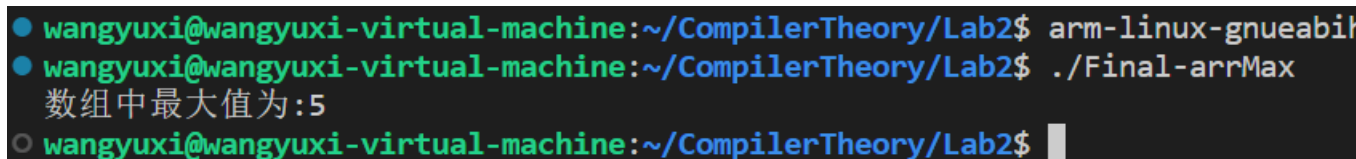
1. 生成可执行程序

执行以下命令即可得到可执行程序:

命令: `arm-linux-gnueabi-gcc Final-arrMax.S -o Final-arrMax -static`

2. 执行并验证

测试结果如图1所示:



```

● waxyuxi@waxyuxi-virtual-machine:~/CompilerTheory/Lab2$ arm-linux-gnueabi-gcc Final-arrMax.S -o Final-arrMax -static
● waxyuxi@waxyuxi-virtual-machine:~/CompilerTheory/Lab2$ ./Final-arrMax
数组中最大值为:5
○ waxyuxi@waxyuxi-virtual-machine:~/CompilerTheory/Lab2$

```

图 1: ARM 汇编代码结果验证

可以看出执行结果与源程序运行结果一致, 因此所编写的 ARM 汇编代码是完全正确的。

三、 总结

通过本次实验, 我们学习到了如何使用上下文无关文法来描述一些 sysY 语言特性, 以及 CFG 的正确设计, 对编译系统原理有了更深入的了解。学习并编写 ARM 汇编代码来体现 sysY

语言特性，为自行设计编译器奠定基础。合理分工，培养小组协作能力。

NIKU