



南开大学
Nankai University

南 开 大 学

计 算 机 学 院

定义你的编译器 & 汇编编程

吴帅达 宋先锋

年级：2021 级

专业：计算机科学与技术

指导教师：李雨森 王刚

2023 年 10 月 10 日

摘要

确定了我们要实现的 SysY 语言特性，给出其形式化定义。设计并完成了两个 SysY 程序，编写等价的 ARM 汇编程序，用汇编器生成可执行程序，调试通过、能正常运行得到正确结果。

关键字：SysY 语言特性、ARM 汇编

目录

一、 编译器所支持的 SysY 语言特性	1
二、 形式化定义	1
(一) SYsY 语言文法	1
(二) SysY 语言的终结符特征	2
三、 ARM 汇编	2
(一) 汇编编程练习 1	2
(二) 汇编编程练习 2	4
四、 思考题	5
五、 分工	6

一、 编译器所支持的 SysY 语言特性

准备实现的 SysY 语言特性:

数据类型: int、float

变量声明、常量声明及初始化

赋值 (=)、表达式语句、语句块

运算符: +、-、*、/、

数组、函数

输入输出, 实现连接 SysY 运行时库

二、 形式化定义

(一) SYsY 语言文法

SysY 语言的文法采用扩展的 Backus 范式 (EBNF, Extended Backus-Naur Form) 表示, 其中:

符号 [...] 表示方括号内包含的为可选项;

符号... 表示花括号内包含的为可重复 0 次或多次的项;

终结符或者是单引号括起的串, 或者是 Ident、InstConst、floatConst 这样的记号。

SysY 语言的文法表示如下, 其中 CompUnit 为开始符号:

编译单元 CompUnit \rightarrow [CompUnit] (Decl | FuncDef)

声明 Decl \rightarrow ConstDecl | VarDecl

常量声明 ConstDecl \rightarrow 'const' BType ConstDef ';' ConstDef ';' ;

基本类型 BType \rightarrow 'int' | 'float'

常数定义 ConstDef \rightarrow Ident '[' ConstExp ']' '=' ConstInitVal

常量初值 ConstInitVal \rightarrow ConstExp

| " [ConstInitVal ';' ConstInitVal] "

变量声明 VarDecl \rightarrow BType VarDef ';' VarDef ';' ;

变量定义 VarDef \rightarrow Ident '[' ConstExp ']' | Ident '[' ConstExp ']' '=' InitVal

变量初值 InitVal \rightarrow Exp | " [InitVal ';' InitVal] "

函数定义 FuncDef \rightarrow FuncType Ident '(' [FuncFParams] ')' Block

函数类型 FuncType \rightarrow 'void' | 'int' | 'float'

函数形参表 FuncFParams \rightarrow FuncFParam ';' FuncFParam

函数形参 FuncFParam \rightarrow BType Ident '[' ']' '[' Exp ']']

语句块 Block \rightarrow " BlockItem "

语句块项 BlockItem \rightarrow Decl | Stmt

语句 Stmt \rightarrow LVal '=' Exp ';' | [Exp] ';' | Block | 'if' '(' Cond ')' Stmt ['else' Stmt] | 'while' '(' Cond ')' Stmt | 'break' ';' | 'continue' ';' | 'return' [Exp] ';' ;

表达式 Exp \rightarrow AddExp 注: SysY 表达式是 int/float 型表达式条件表达式 Cond \rightarrow LOrExp

左值表达式 LVal \rightarrow Ident '[' Exp ']'

基本表达式 PrimaryExp \rightarrow '(' Exp ')' | LVal | Numbr

数值 Number \rightarrow IntConst | floatConst

一元表达式 UnaryExp \rightarrow PrimaryExp | Ident '(' [FuncRParams] ')' | UnaryOp UnaryExp

单目运算符 UnaryOp \rightarrow '+' | '-' | '!' 注: '!' 仅出现在条件表达式中

函数实参表 FuncRParams \rightarrow Exp ';' Exp

乘除模表达式 $\text{MulExp} \rightarrow \text{UnaryExp} \mid \text{MulExp} \text{ '}' \text{ '}' \mid \text{ '}'$
 加减表达式 $\text{AddExp} \rightarrow \text{MulExp} \mid \text{AddExp} \text{ '}' \text{ '}' \mid \text{ '}'$
 关系表达式 $\text{RelExp} \rightarrow \text{AddExp} \mid \text{RelExp} \text{ '}' \text{ '}' \mid \text{ '}' \text{ '}' \mid \text{ '}' \text{ '}'$
 相等性表达式 $\text{EqExp} \rightarrow \text{RelExp} \mid \text{EqExp} \text{ '}' \text{ '}' \mid \text{ '}' \text{ '}'$
 逻辑与表达式 $\text{LAndExp} \rightarrow \text{EqExp} \mid \text{LAndExp} \text{ '}' \text{ '}' \mid \text{ '}'$
 逻辑或表达式 $\text{LOrExp} \rightarrow \text{LAndExp} \mid \text{LOrExp} \text{ '}' \text{ '}' \mid \text{ '}'$
 常量表达式 $\text{ConstExp} \rightarrow \text{AddExp}$

(二) SysY 语言的终结符特征

1、标识符 Ident

SysY 语言中标识符 Ident 的规范如下 (identifier):

$\text{identifier} \rightarrow \text{identifier} - \text{nondigit} \mid \text{identifier} \text{ '}' \text{ '}' - \text{nondigit} \mid \text{identifier} \text{ '}' \text{ '}'$

$\text{identifier} - \text{nondigit} \rightarrow \text{a} \mid \text{b} \mid \text{c} \mid \text{d} \mid \text{e} \mid \text{f} \mid \text{g} \mid \text{h} \mid \text{i} \mid \text{j} \mid \text{k} \mid \text{l} \mid \text{m} \mid \text{n} \mid \text{o} \mid \text{p} \mid \text{q} \mid \text{r} \mid \text{s} \mid \text{t} \mid \text{u} \mid \text{v} \mid \text{w} \mid \text{x} \mid \text{y} \mid \text{z}$

$\text{A} \mid \text{B} \mid \text{C} \mid \text{D} \mid \text{E} \mid \text{F} \mid \text{G} \mid \text{H} \mid \text{I} \mid \text{J} \mid \text{K} \mid \text{L} \mid \text{M} \mid \text{N} \mid \text{O} \mid \text{P} \mid \text{Q} \mid \text{R} \mid \text{S} \mid \text{T} \mid \text{U} \mid \text{V} \mid \text{W} \mid \text{X} \mid \text{Y} \mid \text{Z}$

$\text{identifier} - \text{digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

同名标识符的约定:

全局变量和局部变量的作用域可以重叠, 重叠部分局部变量优先; 同名局部变量的作用域不能重叠;

SysY 语言中变量名可以与函数名相同。

2、注释

SysY 语言中注释的规范与 C 语言一致, 如下:

单行注释: 以序列 ‘//’ 开始, 直到换行符结束, 不包括换行符。

多行注释: 以序列 ‘/*’ 开始, 直到第一次出现 ‘*/’ 时结束, 包括结束处 ‘*/’。

3、数值常量

SysY 语言中数值常量可以是整型数 IntConst , 也可以是浮点型数 FloatConst 。整型数 IntConst 的规范如下 (对应 integer-const):

$\text{integer-const} \rightarrow \text{decimal-const} \mid \text{octal-const} \mid \text{hexadecimal-const}$

$\text{decimal-const} \rightarrow \text{nonzero-digit} \mid \text{decimal-const} \text{ digit}$

$\text{octal-const} \rightarrow 0 \mid \text{octal-const} \text{ octal-digit}$

$\text{hexadecimal-const} \rightarrow \text{hexadecimal-prefix} \text{ hexadecimal-digit} \mid \text{hexadecimal-const} \text{ hexadecimal-digit}$

$\text{hexadecimal-prefix} \rightarrow \text{ '0x' } \mid \text{ '0X' }$

$\text{nonzero-digit} \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\text{octal-digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7$

$\text{hexadecimal-digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid \text{a} \mid \text{b} \mid \text{c} \mid \text{d} \mid \text{e} \mid \text{f} \mid \text{A} \mid \text{B} \mid \text{C} \mid \text{D} \mid \text{E} \mid \text{F}$

三、 ARM 汇编

(一) 汇编编程练习 1

简单的 SysY 程序:

```

1 int main ()
2 {
3     int x=1;

```

```
4     int y=2;
5     x=x+y;
6     cout<<x;
7 }
```

ARM 汇编代码:

```
1     .arch armv7-a
2     .align 2
3     .text
4     .align 2
5     .section .rodata
6     .align 2
7 __str0:
8     .ascii "x: %d\n\0"
9     .align 2
10
11    .text
12    .align 2
13
14    .global main
15
16
17 main:
18    push    {fp, lr}
19    add     fp, sp, #4
20
21    str     fp, [sp, #-4]!
22    mov     fp, sp
23
24    sub     sp, sp, #8
25    mov     r1, #1
26    str     r1, [fp, #-4]
27    mov     r1, #2
28    str     r1, [fp, #-8]
29    ldr     r1, [fp, #-4]
30    ldr     r2, [fp, #-8]
31    add     r1, r2
32    ldr     r0, __bridge
33    bl      printf
34    mov     r0, #0
35    add     sp, fp, #0
36    ldr     fp, [sp], #4
37    pop     {fp, pc}
38
39 __bridge:
40    .word   __str0
```

代码测试结果:

```
sxf@sxf-virtual-machine:~/桌面$ arm-linux-gnueabi-gcc test.S -o test.out
sxf@sxf-virtual-machine:~/桌面$ qemu-arm -L /usr/arm-linux-gnueabihf ./test.out
x: 3
```

图 1

输出结果与预期一致

(二) 汇编编程练习 2

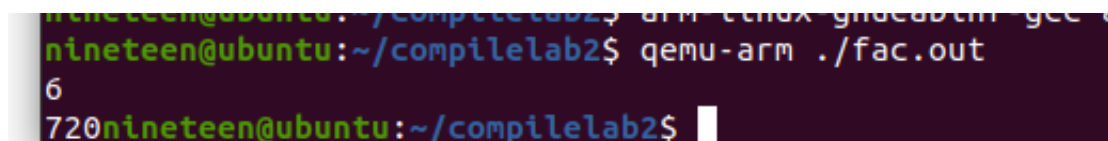
简单的 SysY 程序:

```
1 #include<stdio.h>
2 int main()
3 {
4     int i, n, f;
5     scanf("%d",&n);
6     i = 2;
7     f = 1;
8     while (i <= n)
9     {
10        f = f * i;
11        i = i + 1;
12    }
13    printf("%d",f);
14 }
```

ARM 汇编代码:

```
1 .arch armv7-a
2 .syntax unified
3 .thumb
4 .thumb_func
5
6 .data
7 input_prompt:
8     .asciz  "请输入要计算阶乘的数: "
9 input_num:
10    .word  0
11 scanf_fmt:
12    .asciz  "%d"
13 result_fmt:
14    .asciz  "%d的阶乘为: %d\n"
15
16 .text
17 .global main
18
19 main:
20     @ 保存寄存器状态
21     push    {r4, lr}
22
```

```
23  @ 输入要计算阶乘的数
24  ldr    r0, =input_prompt
25  bl     printf
26
27  ldr    r0, =scanf_fmt
28  ldr    r1, =input_num
29  bl     scanf
30
31  @ 初始化循环计数器和结果
32  mov     r4, #1      @ 计数器（用于阶乘计算）
33  mov     r5, #1      @ 结果
34
35  compute_factorial:
36  @ 比较计数器和输入值
37  cmp     r4, r1
38  bgt     print_result
39
40  @ 计算阶乘
41  mul     r5, r5, r4   @ r5 = r5 * r4
42  add     r4, r4, #1   @ 计数器加1
43  b       compute_factorial
44
45  print_result:
46  @ 输出结果
47  ldr     r0, =result_fmt
48  ldr     r1, =input_num
49  ldr     r2, =r5      @ 阶乘结果
50  bl     printf
51
52  @ 恢复寄存器状态并退出
53  pop     {r4, lr}
54
55  @ 退出程序
56  mov     r7, #1      @ 使用0x1退出程序
57  swi     0x0
```



```
nineteen@ubuntu:~/compilelab2$ qemu-arm ./fac.out
6
720
nineteen@ubuntu:~/compilelab2$
```

图 2

四、 思考题

1. 词法分析器：这个阶段将源代码分解为一个一个的标记，例如关键字、标识符、常量、运算符等。可以使用有限自动机来实现。

2. 语法分析器：这个阶段将标记转换为语法树或抽象语法树。可以使用递归下降分析、LL(1) 分析器或者 LR 分析器来实现。
3. 语义分析器：在此阶段，编译器检查程序是否符合语言的语义规则，例如类型检查、作用域分析等。这可能需要构建符号表来跟踪标识符的声明和使用，并进行类型检查。
4. 中间代码生成：这一步将 AST 或语法树转化为中间表示。中间表示可以是三地址代码、四地址代码、静态单赋值形式等。这个阶段还可以进行优化，例如常量折叠、无用代码消除等。
5. 目标代码生成：这个阶段将中间表示转换为目标汇编代码。这需要根据目标平台的体系结构和寄存器分配算法来生成有效的汇编代码。
6. 优化器：优化器可以在中间代码生成和目标代码生成之间插入，也可以在生成目标代码后进行。它负责执行各种优化，例如循环优化、内联函数、数据流分析等。
7. 链接器：如果程序由多个源文件组成，链接器负责将它们组合成一个可执行文件，并解决外部引用和地址重定向。

五、 分工

小组讨论后得出所选取的语言特性，汇编练习 1 由宋先锋完成，汇编练习 2 由吴帅达完成，实验报告由两人共同撰写。