南開大學
Nankai University

南　　开　　大　　学

定义编译器、实现汇编编程及熟悉辅助工具

王思宇

年级：2021 级

专业：信息安全

蒋薇

年级：2021 级

专业：计算机科学与技术

2023 年 10 月 7 日

摘要

通过本次实验，我们确定了要实现的编译器支持哪些 SysY 语言特性，然后学习教材第 2 章及第 2 章讲义中的 2.2 节、参考 SysY 中巴克斯瑙尔范式定义给出其形式化定义，并且我们用上下文无关文法描述 SysY 语言子集。之后我们设计了 SysY 程序（“预备工作 1”给出的阶乘或斐波那契），并编写了等价的 ARM 汇编程序，然后用汇编器生成可执行程序，且调试通过、能正常运行得到正确结果。

关键字：SysY、CFG、上下文无关文法、ARM 汇编

目录

一、 第一部分

我们合作确定了我们的编译器支持的 SysY 语言特性，并参考 SysY 中巴克斯瑙尔范式定义，用上下文无关文法描述了 SysY 语言子集。并根据所选 SysY 语言特性设计了斐波拉契数列和阶乘程序，包含 putarray() 和 getint() I/O 操作，函数调用、算数运算等十余种 SysY 语言特性，自主编写等价的 ARM 汇编程序并进行优化，通过解决遇到的栈帧调整、函数调用等困难对 ARM 汇编有了基本掌握，最后用汇编器生成可执行程序，调试通过并得到正确结果。

(一) 编译器支持的 SysY 语言特性

1）支持 int 数据类型

2）支持变量声明、常量声明，常量、变量的初始化

3）支持以下语句：赋值语句、表达式语句（表达式可以为空）、语句块、if 分支语句、while 循环语句、return 返回语句

4）支持的运算类型：算术运算（+、-、*、/、%，其中 +、-都可以是单目运算符）、关系运算（==，>，<，>=，<=，!=）和逻辑运算（&&（与）、||（或）、!（非））

5) 支持注释: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‘*/’。

6）支持输入输出

7）函数、语句块：包括函数声明、函数调用、变量、常量作用域，即在函数中、语句块（嵌套）中包含变量、常量声明的处理，break、continue 语句

8）支持数组的声明和数组元素的访问

9）支持浮点数常量识别、变量声明、存储、运算

(二) 上下文无关文法 CFG 描述 SysY 语言子集

我们使用上下文无关文法描述上述所选取的 SysY 语言子集，对其进行形式化定义。

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。一般来说，上下文无关文法可以通过 (VT ,VN ,P,S) 这个四元式定义：

（1）一个终结符号集合 VT，它们有时也称为“词法单元 token”。终结符号是该文法所定义的语言的基本符号的集合。终结符是由单引号引起的字符串或者是标识符 (Ident) 和数值常量 (IntConst)。

终结符一般是: 小写字母（a,b,c,…）、运算符（=，+，-,!，/，%，>,<,>=,<=,…）、标点符号（,(,),…）、数字（0，1，…，9）、粗体字符串（id，if，…）

SysY 语言中标识符 Ident 的规范如下 (identifier)：标识符 identifier → identifier-nondigit| identifier identifier-nondigit| identifier digit

identifier_nondigit → _ | 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 | 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->0|1|2|3|4|5|6|7|8|9

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。

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）：

整型常量:

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->0|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

运算符： =, +, -, !, *, /, %, <, >, <=, >=, ==, !=, , ||

关键字： void, int, const, Ident, if, while, break, continue, return, else, IntConst

基本符号： ;, [,], , , (,), //, /*, */

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）一个非终结符号集合 Vn，它们有时也称为“语法变量”。每个非终结符号表示一个终结符号串的集合。非终结符即一些语法变量，定义了我们所需要的一些中间状态，是源程序到终结符之间的过渡。

（3）一个产生式集合 P，其中每个产生式包括一个称为产生式头或左部的非终结符号，一个箭头，和一个称为产生式体或右部的由终结符号及非终结符号组成的序列。产生式主要用来表示某个语法构造的某种书写形式。如果产生式头非终结符号代表一个语法构造，那么该产生式体就代表了该构造的一种书写方式。产生式： ->

（4）指定一个非终结符号为开始符号 S。表示该文法中最大的语法成分，S 至少在产生式左部出现一次。

CompUnit 为开始符号:

编译单元：CompUnit → CompUnit Decl|CompUnit|Decl|FuncDef|

声明：Decl → ConstDecl| VarDecl

1. 一个 SysY 程序由单个文件组成，文件内容对应 EBNF 表示中的 CompUnit。在该 CompUnit 中，必须存在且仅存在一个标识为‘main’、无参数、返回类型为 int 的 FuncDef(函数定义)。main 函数是程序的入口点，main 函数的返回结果需要输出。

2. CompUnit 的顶层变量/常量声明语句（对应 Decl）、函数定义（对应 FuncDef）都不可以重复定义同名标识符（Ident），即便标识符的类型不同也不允许。

3. CompUnit 的变量/常量/函数声明的作用域从该声明处开始到文件结尾。

基本类型：BType → ’int’

常量声明：ConstDecl → ’const’ BType ConstDefList’;’

ConstDefList->ConstDefList,ConstDef|ConstDefList

常数定义：ConstDef->Ident Dime’=’ConstInitVal

Dim->Dim’[’ConstExp’]’|

ConstDef 用于定义符号常量。ConstDef 中的 Ident 为常量的标识符，在 Ident 后、‘=’之前是可选的数组维度和各维长度的定义部分，在‘=’之后是初始值。Dime 存在时，表示定义数组。ConstExp 都必须能在编译时求值到非负整数。

常量初值：ConstInitVal → ConstExp|”ConstValElement”

ConstValElement->ConstValEnum|

ConstValEnum->ConstValEnum,ConstInitVal|ConstInitVal

变量声明：VarDecl->BType VarDefList';'

VarDefList->VarDefList,VarDef|VarDef

变量定义：VarDef->Ident Dim|Ident Dim '=' InitVal

Dim->Dim'['ConstExp']'|

变量初值：InitVal->Exp|"ValElement"

ValElement->ValEnum|

ValEnum->ValEnum,InitVal|InitVal

函数定义：FuncDef->FuncType Ident '('FuncFParamList')'Block

函数类型：FuncType->'void'|'int'

函数形参表：FuncFParamList->FuncFParams|

FuncFParams->FuncFParams,FuncFParam|FuncFParam

函数形参：FuncFParam->BType Ident OpArray

OpArray->Array|

Array->[]|[]Arrays;

Arrays->[Exp]Arrays|[Exp]

语句块：Block->OpBlockItems

OpBlockItems->BlockItems|

BlockItems->BlockItems BlockItem|BlockItem

语句块项：BlockItem->Decl|Stmt

语句:Stmt->LVal'='Exp';'|OpExp';'|Block|'if'(Cond) Stmt OpElse|'while'(Cond) Stmt|'break";'|'continue";'|'retu
OpExp';'

OpExp->Exp|

OpElse->'else' Stmt|

表达式：Exp->AddExp(SysY 的表达式是 int 型)

条件表达式：Cond->LOrExp

左值表达式：LVal->Ident OpArr

OpArr->Arrays|

基本表达式：PrimaryExp->'('Exp')'|LVal|Number

数值：Number->IntConst

一元表达式：UnaryExp->PrimaryExp|Ident'('OpFuncRParams')'|UnaryOp UnaryExp

单目运算符：UnaryOp->'+'|'-'|'!'

函数实参表：FuncRParams->FuncRParams,Exp|Exp

乘除模表达式:MulExp->UnaryExp|MulExp'*'UnaryExp|MulExp'/'UnaryExp|MulExp'%'UnaryExp

加减表达式：AddExp->MulExp|AddExp'+'MulExp|AddExp'-'MulExp

关系表达式:RelExp->AddExp|RelExp'<'AddExp|RelExp'>'AddExp|RelExp'<='AddExp|RelExp'>='AddExp

相等性表达式：EqExp->RelExp|EqExp'=='RealExp|EqExp'!='RealExp

逻辑与表达式：LAndExp->EqExp|LAndExp'&&'EqExp

逻辑或表达式：LOrExp->LAndExp|LOrExp'||'LAndExp

常量表达式：ConstExp->AddExp

二、 第二部分

(一) 设计 SysY 程序

1. 阶乘

阶乘 SysY 程序

阶乘

```
int factorial(int n) {
    if (n == 0) {
        return 1;
    }
    else {
        return n * factorial(n - 1);
    }
}
int main() {
    int num, result;
    putf("请输入要计算阶乘的数: ");
    num = getint();
    result = factorial(num);
    putf("%d的阶乘为: %d\n",num,result);
    return 0;
}
```

2. 斐波那契数列

斐波那契数列 SysY 程序

斐波那契数列

```

int main(){
    int a,b,i,n,t;
    a=0;
    b=1;
    i=1;
    printf("请输入斐波那契数列的n: ");
    scanf("%d",&n);
    while(i<n){
        t=b;
        b=a+b;
        printf("斐波那契数列是: %d\n",b);
        a=t;
        i=i+1;
    }
    return 0;
}
```

(二) 编写等价的 ARM 汇编程序

1. 阶乘

阶乘汇编代码

```
@数据段
.data
input_num:
    .asciz "请输入要计算阶乘的数: "
format:
    .asciz "%d"
result:
    .asciz "%d的阶乘为: %d\n"

@代码段
.text

factorial:
    str lr,[sp,#-4]!@将lr（返回链接寄存器）压入栈中
    str r0,[sp,#-4]!@将r0压入栈中, r0中保存参数n

    cmp r0,#0  @比较r0中n和0的值
    bne fact   @如果r0不等于0跳转到fact
    mov r0, #1  @如果r0等于0, r0=1
    b fact_exit  @跳转到返回函数

fact:
    sub r0,r0,#1
    bl factorial @调用factorial函数, 结果保存在r0中
    ldr r1,[sp]  @将sp地址中的值保存到r1中
    mul r0,r1

fact_exit:
    add sp,sp,#4  @恢复栈的状态
    ldr lr,[sp],#4  @加载源lr的寄存器内容重新到lr寄存器中
    bx lr  @退出factorial函数

.global main
main:
    str lr,[sp,#-4]!  @保存lr寄存器到栈中
    sub sp,sp,#4  @留出空间用于输入的参数

    ldr r0,address_of_input_num   @传入参数
    bl printf   @调用输出函数

    ldr r0,address_of_format @输入格式化字符串参数
    mov r1,sp
    bl scanf
```

```
44
45      ldr r0,[sp] @输入的参数n保存到r0中
46      bl factorial    @调用factorial函数
47
48      mov r2,r0    @将计算结果保存到r2寄存器中，作为printf的第三个参数
49      ldr r1,[sp] @读入的整数，作为printf的第二个参数
50      ldr r0,address_of_result  @作为printf的第一个参数
51      bl printf  @调用printf函数
52
53      add sp,sp,#4  @恢复sp指针寄存器
54      ldr lr,[sp],#4      @弹出lr
55      bx lr        @退出
56
57  @桥接全局变量的地址
58  address_of_input_num:
59      .word input_num
60  address_of_result:
61      .word result
62  address_of_format:
63      .word format
64
65      .section .note.GNU-stack,"",%progbits
```

在该计算阶乘的程序中，体现出了 SysY 语言的如下几个特性：函数的编写与调用；变量的声明与赋值；putf 和 getint 等运行时库的调用；if 条件判断语句和基本的运算表达式使用。

（1）arm 汇编代码与 x86 类似，可以分为若干个节，如.data 数据节，.text 代码节等。代码段的最后一般需要声明桥接全局变量的地址。

（2）编写阶乘函数 factorial 的具体代码内容前，需要先使用 str 指令保存 lr 寄存器和函数参数的值，通常 r0-r3 用作保存参数，lr 寄存器用于保存返回地址。函数返回时，需要恢复栈状态，并用 ldr 指令恢复 lr 寄存器的值，最后的返回值保存在 r0 寄存器中。

（3）函数 factorial 的主要代码实现与 if-else 分支语句、递归调用相关。对于分支语句来说，需要使用 cmp,bne 这两个指令和相关代码标签实现，先判断参数 r0 与 0 的值，若不等则跳转到 fact 代码段，否则跳转到 fact_exit 代码段。递归调用需要在 factorial 函数中使用 bl 指令再次调用该函数，此时其参数为 r0 -1。

2. 斐波那契数列

斐波那契数列汇编代码

```
1       @数据段,全局变量及常量声明
2       .data
3   a:
4       .word 0
5   b:
6       .word 1
7   i:
8       .word 1
9   n:
```

```
    .word 0
t:
    .word 0


@代码段
    .text
    .align 4
result:
    .asciz "fibo: %d \n"
    .align 4
input_str:
    .asciz "请输入斐波那契数列的n: "

input_num:
    .asciz "%d"
    .align 4

@主函数
    .global main
    .type main,%function
main:
    push {fp, lr} @保存返回地址栈基地址

input:
    adr r0,input_str @读取字info字符串地址
    bl printf @调用printf函数输出
    mov r8,lr
    adr r0,input_num
    sub sp,sp,#4 @留出一个4字节的空间
    mov r1,sp
    bl scanf
    ldr r2,[sp,#0]
    ldr r1,addr_n0
    str r2,[r1] @保存n到对应地址中
    add sp,sp,#4
    mov lr,r8

params:
    mov r0,r2
    ldr r4,addr_i0
    ldr r4,[r4] @变量i
    ldr r3,addr_b0
    ldr r3,[r3] @变量b
    ldr r6,addr_a0
    ldr r6,[r6] @变量a

loop1:
```

```
58      mov r5,r3 @ t= b
59      add r3,r3,r6 @ b = a + b
60      push {r0,r1,r2,r3}
61      adr r0,result @ 准备printf函数
62      mov r1,r3
63      bl printf  @调用 printf函数
64      pop {r0,r1,r2,r3}
65
66      mov r6,r5 @ a = t
67      add r4,#1 @ i = i + 1
68      cmp r4,r0 @判断i与n大小关系
69      ble loop1 @i < n跳转loop1继续循环
70
71  end:
72      pop {pc}
73
74  @桥接全局变量的地址
75  addr_a0:
76      .word a
77  addr_b0:
78      .word b
79  addr_i0:
80      .word i
81  addr_t0:
82      .word t
83  addr_n0:
84      .word n
85
86      .section .note.GNU-stack,"",%progbits
```

（1）在斐波那契额数列程序中，涉及到 I/O 操作、变量的声明与赋值、while 循环、算数运算等 SysY 语言特性，arm 汇编程序中有较多的交互输出语句，使用过程中涉及栈帧的调整和寄存器的保存与恢复。

（2）调用函数时通过寄存器 R0-R3 来传递参数。

（3）汇编代码中利用 _bridge 标签，“桥接”了在 C 代码中隐性的全局变量的地址

(三) 汇编器生成可执行程序

将编写的汇编代码用下列指令进行测试：

arm-linux-gnueabihf-gcc example.S -o example.out

qemu-arm -L /usr/arm-linux-gnueabihf ./example.out

1. 阶乘

```
army@DESKTOP-8P7QBT0:~/lab2$ arm-linux-gnueabihf-gcc fac.S -o fac.out
army@DESKTOP-8P7QBT0:~/lab2$ qemu-arm -L /usr/arm-linux-gnueabihf ./fac.out
请输入要计算阶乘的数: 6
6的阶乘为: 720
```

2. 斐波那契数列

```
army@DESKTOP-8P7QBT0:~/lab2$ arm-linux-gnueabihf-gcc fib.S -o fib.out
army@DESKTOP-8P7QBT0:~/lab2$ qemu-arm -L /usr/arm-linux-gnueabihf ./fib.out
请输入斐波那契数列的n: 10
fibo: 1
fibo: 2
fibo: 3
fibo: 5
fibo: 8
fibo: 13
fibo: 21
fibo: 34
fibo: 55
fibo: 89
```

三、 思考

如果不是人“手工编译”，而是要实现一个计算机程序 (编译器) 来将 SysY 程序转换为汇编程序，应该如何做 (这个编译器程序的数据结构和算法设计) ？注意: 编译器不能只会翻译一个源程序，而是要有能力翻译所有合法的 SysY 程序。

词法分析作为语法分析的调用接口，调用一次返回应该单词。

利用上下文无关文法将不同语言特性的程序语句符号化。

利用语法制导翻译，数据结构采用语法分析树的形式，将语义动作嵌入树的节点中。

语法分析树的构造采用预测分析法，根据下一输入单词进行首单词比对，选择候选式；对终结符进行匹配 match；对每一个非终结符编写递归函数；左递归改为右递归。

设计语法制导定义/翻译模式实现 SysY 程序到汇编程序的翻译。

为每个阶段构造符号表，以键值对的形式存储 ID 和对应的值。

穷举所有 SysY 程序 (无穷无尽) 是不可能的，怎么办? 每个语言特性如何翻译。

每个语言特性仍然有无穷多个合法的实例 (a=1，b=2.0，$\cdot$)，怎么办? 符号化，语法制导翻译。

尝试设计语法制导定义/翻译模式实现简单的 SysY 程序到汇编程序的翻译，并通过 Bison 进行实验。

函数栈的增长:

ARM 架构中，函数栈在内存中是从高地址向低地址增长的。在函数调用时，参数和局部变量被压入栈中，栈指针（SP）减小。在函数返回时，栈中的数据被弹出，栈指针（SP）增加。

使用寄存器：合理利用通用寄存器（R0-R12）进行数据操作和传递。栈操作：使用堆栈指针寄存器（SP）进行栈操作，包括压栈（push）和弹栈（pop）操作。跳转与分支：使用跳转指令（B、BL）进行无条件跳转和分支指令（BEQ、BNE 等）进行条件分支。存储和加载：使用存储指令（STR、STMFD 等）将数据存储到内存中，使用加载指令（LDR、LDMFD 等）从内存中加载数据。异常处理：对于异常情况，使用异常处理指令（SWI）触发软件中断，并编写相应的异常处理程序。

小组分工:

首先共同确定了本学期构建的编译器实现的特性，共同完成 CFG 设计工作，并将完成结果合并后进行讨论与改进，完善了适合多种书写方式的变量和常量的声明及初始化。

在 arm 汇编编程部分，一人负责斐波那契程序的 SysY 编写和 arm 汇编编写，一人负责阶乘程序的 SysY 和 arm 汇编编写，过程中遇到的函数调用、栈帧转换等困难经过一起讨论得以解决，最后的思考题共同完成。

参考文献 [1] [2]

参考文献

[1] Jeffrey D.Ullman Alfred V.Aho, Ravi Sethi. *Compilers: Principles,Techniques,and Tools.* 1986.

[2] 助教. 实验指导书. 2023.