



南開大學  
Nankai University

南 开 大 学

计 算 机 学 院

预备工作 2

---

定义编译器 & 汇编编程

---

丁屹、卢麒萱

年级：2020 级

专业：计算机科学与技术

2022 年 10 月 16 日

# 摘要

关键字：CFG 设计 ARM 汇编

# 目录

一、 总体设计及分工	1
二、 CFG 描述	1
(一) 标识符及数字	1
(二) 变量/常量声明	1
(三) 数组声明	2
(四) 赋值语句	2
(五) 复合语句	2
(六) if 分支语句	2
(七) for/while 循环语句	2
(八) 算术表达式	3
(九) 关系运算表达式	3
(十) 逻辑表达式	3
(十一) 函数	3
三、 ARM 汇编	4
(一) SysY 程序设计	4
(二) 卢麒萱的工作	4
1. 丁屹的工作	8

## 一、 总体设计及分工

经过讨论，确定我们实现的编译器 SysY 语言特性有：

1. 变量/常量声明：支持 int、float 类型
2. 数组声明：支持 int、float 类型及数组指针
3. 赋值语句
4. 复合语句
5. if 分支语句
6. for/while 循环语句
7. 算术表达式：支持加减乘除、按位与或
8. 关系运算表达式：支持不等、等于、大于、小于
9. 逻辑表达式：支持逻辑与或
10. 函数

## 二、 CFG 描述

### (一) 标识符及数字

$$\begin{aligned}
 id &\rightarrow id - nondigit \mid id \ id - nondigit \mid id \ number \\
 id - nondigit &\rightarrow \{ \_A - Z a - z \} \\
 digit &\rightarrow number \ digit \\
 number &\rightarrow \{ 0 - 9 \}
 \end{aligned}$$

id 表示标识符，以下都认为是终结符，num 表示数字字符，digit 表示数字。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；
- 变量名可以与函数名相同。

### (二) 变量/常量声明

$$\begin{aligned}
 type &\rightarrow \text{int} \mid \text{float} \\
 idlist &\rightarrow idlist, id \mid id \\
 decl &\rightarrow type \ idlist
 \end{aligned}$$

其中 type 代表变量类型，decl 代表声明语句，idlist 代表标识符列表。

### (三) 数组声明

$$\begin{aligned} arr\_decl &\rightarrow type\ arr\_idlist \\ arr\_idlist &\rightarrow arr\_idlist, id[arr\_size] \mid id[arr\_size] \mid *id \\ arr\_size &\rightarrow id \mid number \mid \epsilon \end{aligned}$$

其中  $arr\_decl$  代表数组声明语句,  $arr\_idlist$  表示数组标识符列表,  $arr\_size$  表示数组定义大小。

### (四) 赋值语句

$$\begin{aligned} unary\_expr &\rightarrow digit \mid id \\ assign\_expr &\rightarrow unary\_expr = assign\_expr \mid logical\_expr \\ unary\_expr &\rightarrow prim\_expr \mid funcname(paralist) \mid unary\_op\ unary\_expr \\ prim\_expr &\rightarrow (add\_expr) \mid Lval \mid number \\ L\_val &\rightarrow id\ [exp] \\ unary\_op &\rightarrow \{+, -, !\} \end{aligned}$$

$logical\_expr$  为逻辑表达式,  $unary\_expr$  为一元表达式,  $assign\_expr$  为赋值表达式, 这里最后一个产生式第二个右部为逻辑表达式的原因是因为逻辑表达式的逻辑与和逻辑或优先级高于赋值表达式但却低于关系表达式的比较和算术表达式的各种运算,  $prim\_expr$  为基本表达式,  $L\_val$  为左值表达式,  $unary\_op$  为单目运算符, '!' 仅出现在条件表达式中。

### (五) 复合语句

$$\begin{aligned} block &\rightarrow \{blockitem\} \\ blockitem &\rightarrow blockitem; blockitem \mid decl \mid stmt \end{aligned}$$

其中  $block$  表示一个复合语句块,  $blockitem$  表示中间内容, 由一或多条表达式或分支循环语句组成。

### (六) if 分支语句

$$stmt \rightarrow \text{if}(expr)\ stmt\ \text{else}\ stmt$$

其中  $stmt$  为语句,  $expr$  为表达式。

### (七) for/while 循环语句

$$\begin{aligned} stmt &\rightarrow \text{while}(expr)\ stmt \\ stmt &\rightarrow \text{for}(expr; expr; expr)\ stmt \end{aligned}$$

其中  $stmt$  为语句,  $expr$  为表达式。

## (八) 算术表达式

$$\begin{aligned}
 mul\_expr &\rightarrow unary\_expr \mid mul\_expr \textbf{mul\_op} unary\_expr \\
 mul\_op &\rightarrow \{*, /\} \\
 add\_expr &\rightarrow mul\_expr \mid add\_expr \textbf{add\_op} mul\_expr \\
 add\_op &\rightarrow \{+, -\}
 \end{aligned}$$

其中  $mul\_expr$  为乘除模运算表达式,  $mul\_op$  为乘除模符号,  $add\_expr$  为加减运算表达式,  $add\_op$  为加减运算符。

## (九) 关系运算表达式

$$\begin{aligned}
 rel\_expr &\rightarrow add\_expr \mid rel\_expr \textbf{rel\_op} add\_expr \\
 rel\_op &\rightarrow \{<, >, <=, >=\} \\
 equ\_expr &\rightarrow rel\_expr \mid equ\_expr \textbf{equ\_op} rel\_expr \\
 equ\_op &\rightarrow \{==, !=\}
 \end{aligned}$$

其中  $rel\_expr$  为包含大于、小于、大于等于、小于等于的关系表达式,  $equ\_expr$  为包含等于和不等于是的关系表达式。

## (十) 逻辑表达式

$$\begin{aligned}
 logical\_expr &\rightarrow lor\_expr \\
 lor\_expr &\rightarrow land\_expr \mid lor\_expr \textbf{lor\_op} land\_expr \\
 lor\_op &\rightarrow \{||\} \\
 land\_expr &\rightarrow equ\_expr \mid land\_expr \textbf{land\_op} equ\_expr \\
 land\_op &\rightarrow \{\&\&\}
 \end{aligned}$$

## (十一) 函数

$$\begin{aligned}
 funcdef &\rightarrow type \textbf{funcname} (paralist) \textbf{stmt} \\
 paralist &\rightarrow paralist, parade f \mid parade f \mid \epsilon \\
 parade f &\rightarrow type \textbf{id} \\
 funcname &\rightarrow \textbf{id}
 \end{aligned}$$

其中  $paralist$  代表参数列表,  $parade f$  代表参数声明,  $funcname$  代表函数名,  $funcdef$  代表函数声明语句。

## 三、 ARM 汇编

### (一) SysY 程序设计

为满足上述选定语言特性，我们二人分别设计了两个 SysY 程序，涵盖了我们定义的所有语言特性。这里使用了标准库的 `scanf`、`printf`，后续实现 SysY 时会替换成 `syslib` 的 I/O 函数。

### (二) 卢麒萱的工作

编写的 SysY 示例程序如下：

---

```
1  #include <stdio.h>
2
3  int a, b;
4  float c[5] = {1, 2, 3, 4, 5};
5
6  int f(int a, int b) {
7      if(a > b) return 1;
8      else return 0;
9  }
10
11 int main() {
12     scanf("%d%d", &a, &b);
13     const int d = 0;
14     {
15         int d = f(a, b);
16         if(d == 1) {
17             for(int i = 0; i < 5; i++)
18                 d = d * c[i];
19             printf("%d\n", d);
20         } else d = 1;
21     }
22     printf("%d\n", d);
23 }
```

---

编写对应 ARM 汇编代码如下：

---

```
1  .arch armv7-a
2  .fpu vfpv3-d16
3
4  .text
5  .global c
6  .bss
7  .type c, %object
8  .size c, 20
```

---

```
9  a:
10  .space 4
11  .global b
12  .type b, %object
13  .size b, 4
14  b:
15  .space 4
16  .global c
17  .data
18  .type c, %object
19  .size c, 20
20  c:
21  @ 浮点数 1-5 在内存中存放的地址按照 “IEEE 754 标准” 单精度格式编码的浮点数, 对应的十进制值
22  .word 1065353216
23  .word 1073741824
24  .word 1077936128
25  .word 1082130432
26  .word 1084227584
27  .text
28  .global f
29  .type f, %function
30  @ 函数部分
31  f:
32  sub sp, sp, #12
33  str r0, [sp, #4]
34  str r1, [sp]
35  ldr r2, [sp, #4]
36  ldr r3, [sp]
37  cmp r2, r3
38  ble .L1
39  movs r3, #1
40  b .end
41  .L1:
42  movs r3, #0
43  .end:
44  mov r0, r3
45  adds sp, sp, #12
46  mov pc, lr
47  @ 读入字符串
48  .str0:
49  .ascii "%d%d\000"
50  .align 2
51  .str1:
52  .ascii "%d\012\000"
```

```
53  .text
54  .global  main
55  .syntax unified
56  .thumb
57  .type  main, %function
58  @ 主函数
59  main:
60      push  {fp, lr}
61      sub   sp, sp, #16
62      add   fp, sp, #0
63      ldr   r3, ._bridge
64  .LPIC0:
65      add   r3, pc
66      mov   r2, r3
67      ldr   r3, ._bridge+4
68  .LPIC1:
69      add   r3, pc
70      mov   r1, r3
71      ldr   r3, ._bridge+8
72  .LPIC2:
73      add   r3, pc
74      mov   r0, r3
75      bl    __isoc99_scanf
76      movs  r3, #0
77      str   r3, [fp, #4]
78      ldr   r3, ._bridge+12
79  .LPIC3:
80      add   r3, pc
81      ldr   r2, [r3]
82      ldr   r3, ._bridge+16
83  .LPIC4:
84      add   r3, pc
85      ldr   r3, [r3]
86      mov   r1, r3
87      mov   r0, r2
88      bl    f
89      str   r0, [fp, #12]
90      ldr   r3, [fp, #12]
91      cmp   r3, #1
92      bne   .L5
93      movs  r3, #0
94      str   r3, [fp, #8]
95      b     .L6
96  .L7:
```




```
97     ldr r3, [fp, #12]
98     vmov s15, r3
99     vcvtf32.s32 s14, s15
100    ldr r2, ._bridge+20
101    .LPIC5:
102    add r2, pc
103    ldr r3, [fp, #8]
104    lsls r3, r3, #2
105    add r3, r3, r2
106    vldr.32 s15, [r3]
107    vmul.f32 s15, s14, s15
108    vcvts32.f32 s15, s15
109    vmov r3, s15
110    str r3, [fp, #12]
111    ldr r3, [fp, #8]
112    adds r3, r3, #1
113    str r3, [fp, #8]
114    .L6:
115    ldr r3, [fp, #8]
116    cmp r3, #4
117    ble .L7
118    ldr r1, [fp, #12]
119    ldr r3, ._bridge+24
120    .LPIC6:
121    add r3, pc
122    mov r0, r3
123    bl printf
124    b .L8
125    .L5:
126    movs r3, #1
127    str r3, [fp, #12]
128    .L8:
129    ldr r1, [fp, #4]
130    ldr r3, ._bridge+28
131    .LPIC7:
132    add r3, pc
133    mov r0, r3
134    bl printf
135    movs r3, #0
136    mov r0, r3
137    adds fp, fp, #16
138    mov sp, fp
139    pop {fp, pc}
140    @ 桥接隐性全局变量的地址
```

```

141  _bridge:
142      .word  b-(.LPIC0+4)
143      .word  a-(.LPIC1+4)
144      .word  .str0-(.LPIC2+4)
145      .word  a-(.LPIC3+4)
146      .word  b-(.LPIC4+4)
147      .word  c-(.LPIC5+4)
148      .word  .str1-(.LPIC6+4)
149      .word  .str1-(.LPIC7+4)
150      .size  main, .-main

```

测试代码正确性, 编译静态库 `sylib.a` 后, 通过 `arm-linux-gnueabi-gcc sample.arm.s sylib.a -o sample.arm` 进行链接生成可执行代码, 再使用 `qemu` 运行, 测试如图2所示。



```

root at DESKTOP-IKOEBCQ in ~/src/compile_principles/ex2 10.16 22-10-16 - 16:49:38
o arm-linux-gnueabi-gcc sample.arm.s sylib.a -o sample.arm
/usr/lib/gcc-cross/arm-linux-gnueabi/12/../../../../arm-linux-gnueabi/bin/ld: warning: /tmp/ccUhtjvca.o: missing .note.GNU-stack section implies executable stack
/usr/lib/gcc-cross/arm-linux-gnueabi/12/../../../../arm-linux-gnueabi/bin/ld: NOTE: This behaviour is deprecated and will be removed in a future version of the linker
root at DESKTOP-IKOEBCQ in ~/src/compile_principles/ex2 10.16 22-10-16 - 17:01:43
o qemu-arm -L /usr/arm-linux-gnueabi/ ./sample.arm
6 5
120
0
root at DESKTOP-IKOEBCQ in ~/src/compile_principles/ex2 10.16 22-10-16 - 17:01:50
o qemu-arm -L /usr/arm-linux-gnueabi/ ./sample.arm
4 5
0

```

图 1: 汇编代码 1 测试

根据程序设定,  $a \leq b$  和  $a > b$  得到的结果均正确。

## 1. 丁屹的工作

编写的 SysY 示例程序如下:

```

1  #include <stdio.h>
2
3  int  trace[18];
4  int* ptrace      = trace;
5  float (*nonsense)[2] = trace;
6  float (*ns)[2][3]   = trace;
7  float nnss[5][4][3][2];
8
9  void fib(int n, int* a, int* b) {
10     for (int i = 0; i < n; ++i) {
11         int t = *b;
12         *b += *a;
13         trace[i] = *b;
14         *a      = t;
15     }
16 }
17

```

```

18 int main() {
19     int      a = 1, b = 1;
20     const int n      = 10;
21     char      s_fmt[19] = "%d%d";
22     const char p_fmt[19] = "%d %d %d %d %d %d\n";
23     scanf(s_fmt, &a, &b);
24     if (a > 0 && 0 < b
25         || (a >= 0 || 0 <= b) && a > -1 && -1 < b && (a != 0 || 1 == b)) {
26         // a > 0, b > 0 or a == 0, b == 1
27         fib(n, &a, &b);
28         printf(p_fmt, a + b, a - b, a * b, a / b, a & b, a | b);
29         int i = 0;
30         while (1) {
31             printf("%d ", trace[i]);
32             ++i;
33             if (i == 10) {
34                 printf("\n");
35                 break;
36             } else continue;
37             printf("unreachable!\n");
38         }
39     } else {
40         char err[] = "invalid input\n";
41         printf(err);
42         return 1;
43     }
44 }

```

编写对应 ARM 汇编代码如下:

```

1 .arch armv7-a
2 .fpu vfpv3-d16
3
4 .text
5 .global trace
6 .bss
7 .type trace, %object
8 .size trace, 72
9 trace:
10 .space 72
11
12 .global ptrace
13 .section .data.rel.local,"aw"
14 .type ptrace, %object

```

```
15  .size ptrace, 4
16  ptrace:
17  .word trace
18
19  .global nonsense
20  .type nonsense, %object
21  .size nonsense, 4
22  nonsense:
23  .word trace
24
25  .global ns
26  .type ns, %object
27  .size ns, 4
28  ns:
29  .word trace
30
31  .global nnss
32  .bss
33  .type nnss, %object
34  .size nnss, 480
35  nnss:
36  .space 480
37
38  .text
39  .global fib
40  .type fib, %function
41  fib:
42      str fp, [sp, #-4]!
43      add fp, sp, #0
44      sub sp, sp, #28
45      str r0, [fp, #-16]
46      @ int n
47      str r1, [fp, #-20]
48      @ int* a
49      str r2, [fp, #-24]
50      @ int* b
51      mov r3, #0
52      str r3, [fp, #-8]
53      @ int i = 0
54      b .L0
55      @ for (int i = 0; i < n; ++i)
56  .L1:
57      ldr r3, [fp, #-24]
58      @ b
```

```
59     ldr r3, [r3]
60     @ *b
61     str r3, [fp, #-12]
62     @ int t = *b
63     ldr r2, [fp, #-20]
64     @ a
65     ldr r2, [r2]
66     @ *a
67     add r3, r2, r3
68     ldr r2, [fp, #-24]
69     str r3, [r2]
70     @ *b += *a
71     ldr r1, [r2]
72     @ *b
73     ldr r2, [fp, #-12]
74     @ *a = t
75     ldr r3, .LTR
76 .LPICO:
77     add r3, pc, r3
78     ldr r2, [fp, #-8]
79     @ i
80     str r1, [r3, r2, lsl #2]
81     @ trace[i] = *b
82     ldr r3, [fp, #-20]
83     @ a
84     ldr r2, [fp, #-12]
85     @ t
86     str r2, [r3]
87     @ *a = t
88     ldr r3, [fp, #-8]
89     add r3, r3, #1
90     str r3, [fp, #-8]
91     @ ++i
92 .L0:
93     ldr r2, [fp, #-8]
94     @ i
95     ldr r3, [fp, #-16]
96     @ n
97     cmp r2, r3
98     blt .L1
99     @ for (...; i < n; ...)
100    add sp, fp, #0
101    ldr fp, [sp], #4
102    bx lr
```

```
103     @ return
104
105 .LTR:
106 .word trace-(.LPIC0+8)
107 .size fib, .-fib
108 .align 2
109
110 .global __aeabi_idiv
111 .section .rodata
112 .align 2
113
114 .LUNMSTR0:
115 .ascii "%d \000"
116 .align 2
117
118 .LSFMT:
119 .ascii "%d%d\000"
120 .space 14
121 .align 2
122
123 .LPFMT:
124 .ascii "%d %d %d %d %d %d\012\000"
125 .align 2
126
127 .LINVSTR:
128 .ascii "invalid input\012\000"
129 .text
130 .align 2
131
132 .global main
133 .type main, %function
134 main:
135     push {r4, r5, r6, fp, lr}
136     add fp, sp, #16
137     sub sp, sp, #92
138     mov r3, #1
139     str r3, [fp, #-32]
140     @ a = 1
141     str r3, [fp, #-36]
142     @ b = 1
143     mov r3, #10
144     str r3, [fp, #-28]
145     @ n = 10
146     ldr r2, .LPIC0
```

```
147 .LPIC1:
148     add r2, pc, r2
149     @ char s_fmt[19] = "%d%d"
150     sub r3, fp, #56
151     ldm r2, {r0, r1}
152     str r0, [r3]
153     add r3, r3, #4
154     strb r1, [r3]
155     sub r3, fp, #51
156     mov r2, #0
157     str r2, [r3]
158     str r2, [r3, #4]
159     str r2, [r3, #8]
160     strh r2, [r3, #12]
161     ldr r3, .LPIC2+4
162     @ const char p_fmt[19] = "%d %d %d %d %d %d\n"
163 .LPIC2:
164     add r3, pc, r3
165     sub ip, fp, #76
166     mov lr, r3
167     ldmbia lr!, {r0, r1, r2, r3}
168     stmbia ip!, {r0, r1, r2, r3}
169     ldr r3, [lr]
170     strh r3, [ip]
171     add ip, ip, #2
172     lsr r3, r3, #16
173     strb r3, [ip]
174     sub r2, fp, #36
175     sub r1, fp, #32
176     sub r3, fp, #56
177     mov r0, r3
178     bl scanf(PLT)
179     @ scanf(s_fmt, &a, &b)
180     ldr r3, [fp, #-32]
181     cmp r3, #0
182     ble .L2
183     @ a > 0
184     ldr r3, [fp, #-36]
185     cmp r3, #0
186     bgt .L3
187     @ 0 < b
188 .L2:
189     ldr r3, [fp, #-32]
190     cmp r3, #0
```

```
191     bge .L4
192     @ a >= 0
193     ldr r3, [fp, #-36]
194     cmp r3, #0
195     blt .LINV
196     @ 0 <= b
197 .L4:
198     ldr r3, [fp, #-32]
199     cmp r3, #-1
200     ble .LINV
201     @ a > -1
202     ldr r3, [fp, #-36]
203     cmp r3, #-1
204     ble .LINV
205     @ -1 < b
206     ldr r3, [fp, #-32]
207     cmp r3, #0
208     bne .L3
209     @ a != 0
210     ldr r3, [fp, #-36]
211     cmp r3, #1
212     bne .LINV
213     @ 1 == b
214 .L3:
215     @ exec fib
216     sub r2, fp, #36
217     @ &b
218     sub r3, fp, #32
219     mov r1, r3
220     @ &a
221     ldr r0, [fp, #-28]
222     @ n
223     bl fib(PLT)
224     @ fib
225     ldr r2, [fp, #-32]
226     @ a
227     ldr r3, [fp, #-36]
228     @ b
229     add r4, r2, r3
230     @ a + b
231     sub r5, r2, r3
232     @ a - b
233     mul r6, r2, r3
234     @ a * b
```



```
235     mov r1, r3
236     @ r1 = b
237     mov r0, r2
238     @ r0 = a
239     bl __aeabi_idiv(PLT)
240     @ r0 /= r1 -> a / b
241     mov ip, r0
242     ldr r1, [fp, #-32]
243     @ a
244     ldr r2, [fp, #-36]
245     @ b
246     and r3, r1, r2
247     @ a & b
248     orr r2, r1, r2
249     @ a | b
250     sub r0, fp, #76
251     @ push p_fmt
252     str r2, [sp, #8]
253     @ push a | b
254     str r3, [sp, #4]
255     @ push a & b
256     str ip, [sp]
257     @ push a / b
258     mov r3, r6
259     @ push a * b
260     mov r2, r5
261     @ push a - b
262     mov r1, r4
263     @ push a + b
264     bl printf(PLT)
265     mov r3, #0
266     str r3, [fp, #-24]
267 .L5:
268     ldr r3, .LPICT+8
269     @ "%d "
270 .LPIC3:
271     add r3, pc, r3
272     ldr r2, [fp, #-24]
273     ldr r3, [r3, r2, lsl #2]
274     mov r1, r3
275     @ push "%d "
276     ldr r0, .LPICT+12
277 .LPIC4:
278     add r0, pc, r0
```

```

279     @ push trace[i]
280     bl printf(PLT)
281     ldr r3, [fp, #-24]
282     add r3, r3, #1
283     str r3, [fp, #-24]
284     @ ++i
285     cmp r3, #10
286     bne .L5
287     @ i != 10
288     mov r0, #10
289     bl putchar(PLT)
290     @ printf("\n") -> putchar(10)
291     mov r0, #0
292     b .L6
293     @ printf("unreachable!\n") can be deleted
294 .LINV:
295     ldr r3, .LPICt+16
296     @ "invalid input\n"
297 .LPIC5:
298     add r3, pc, r3
299     sub ip, fp, #92
300     ldm r3, {r0, r1, r2, r3}
301     stmia ip!, {r0, r1, r2}
302     strh r3, [ip]
303     add ip, ip, #2
304     lsr r3, r3, #16
305     strb r3, [ip]
306     sub r3, fp, #92
307     mov r0, r3
308     bl printf(PLT)
309     mov r0, #1
310 .L6:
311     sub sp, fp, #16
312     pop {r4, r5, r6, fp, pc}
313 .LPICt:
314     .word .LSFMT-(.LPIC1+8)
315     .word .LPFMT-(.LPIC2+8)
316     .word trace-(.LPIC3+8)
317     .word .LUNMSTRO-(.LPIC4+8)
318     .word .LINVSTR-(.LPIC5+8)
319     .size main, .-main

```

测试代码正确性，使用 Makefile 的目标 arm-build-run 自动构建，使用 qemu-arm 环境模拟运行，测试如图??所示

```
0|NKU-Compiler-System (2022-10-16 X)$ make arm-build-run
for i in *.arm.s; do armv7l-linux-gnueabi-gcc -o${i%%.s}.exec $i; done
qemu-arm -L /usr/armv7l-linux-gnueabi/lib -E LD_LIBRARY_PATH=/usr/armv7l-linux-gnueabi/lib fib.arm.exec
1 1
233 -55 12816 0 16 217
2 3 5 8 13 21 34 55 89 144
0|NKU-Compiler-System (2022-10-16 X)$ make arm-build-run
for i in *.arm.s; do armv7l-linux-gnueabi-gcc -o${i%%.s}.exec $i; done
qemu-arm -L /usr/armv7l-linux-gnueabi/lib -E LD_LIBRARY_PATH=/usr/armv7l-linux-gnueabi/lib fib.arm.exec
0 1
144 -34 4895 0 17 127
1 2 3 5 8 13 21 34 55 89
0|NKU-Compiler-System (2022-10-16 X)$ make arm-build-run
for i in *.arm.s; do armv7l-linux-gnueabi-gcc -o${i%%.s}.exec $i; done
qemu-arm -L /usr/armv7l-linux-gnueabi/lib -E LD_LIBRARY_PATH=/usr/armv7l-linux-gnueabi/lib fib.arm.exec
0 0
invalid input
make: *** [Makefile:24: arm-build-run] 错误 1
0|NKU-Compiler-System (2022-10-16 X)$
```

图 2: 汇编代码 2 测试

NKU