

# lab2

---

PB17030889 吉志远

## 1. 汇编代码

### C语言

```
typedef int i16;
typedef unsigned int u16;
i16 func(i16 n, i16 a, i16 b, i16 c, i16 d, i16 e, i16 f){ //Lots of arguments, hah?
    i16 t = GETC() - '0' + a + b + c + d + e + f;
    if(n > 1){
        i16 x = func(n - 1, a, b, c, d, e, f);
        i16 y = func(n - 2, a, b, c, d, e, f);
        return x + y + t - 1;
    }else{
        return t;
    }
}
i16 main(void){
    i16 n = GETC() - '0';
    return func(n, 0, 0, 0, 0, 0, 0);
}
__Noreturn void __start(){
    /**/
    u16 __R0=main()
    HALT();
}
```

### 汇编代码

```
.ORIG x3000
;;interface
;;using R0 as argument 'n'
;;R0 return value

;;__start() begins here:

JSR main
ADD R0,R3,#0    ;R0=main()
HALT
;;main() begins
main
ST R1,SAVE1
ST R2,SAVE2
ST R4,SAVE4
ST R5,SAVE5
```

```

ST R6,SAVE6
LEA R6,STACK ;R6 is location for return address stack
LEA R5,TEMP ;R5 is location for n stack
LEA R4,T ;R4 is location for t stack
LEA R2,RESULT ;R2 is return value

STR R7,R6,#0 ;store return __start address
ADD R6,R6,#1
IN ; char R0 = GETC()
LD R3,ASCII
ADD R1,R0,R3 ;R1=GETC()-'0',e.g.R1 = n
JSR func
ADD R2,R2,#-1
LDR R3,R2,#0 ;now R3 is return value

ADD R6,R6,#-1
LDR R7,R6,#0 ;load return __start address
LD R1,SAVE1
LD R2,SAVE2
LD R4,SAVE4
LD R5,SAVE5
LD R6,SAVE6
RET

;;func()
;;R3 作为返回值
;;R0 为传入参数
func
STR R7,R6,#0 ;save return address in stack
ADD R6,R6,#1 ;increase stack pointer
IN ;input t
LD R3,ASCII
ADD R0,R0,R3 ;t = GETC()-'0'
ADD R1,R1,#-1 ;test n=n-1
BRnz else ;n<=1?jump to else, return t
STR R1,R5,#0 ;save n-1 to temp n stack
ADD R5,R5,#1
STR R0,R4,#0 ;save t to temp t stack
ADD R4,R4,#1
JSR func ;call func(n-1)
;;restore R1(n) from R5(temp)
ADD R5,R5,#-1
LDR R1,R5,#0 ;load n-1
ADD R1,R1,#-1 ;get func(n-2)
JSR func
;;
ADD R2,R2,#-1
LDR R1,R2,#0 ;R1 = y e.g. func(n-2)
ADD R2,R2,#-1
LDR R3,R2,#0 ;R3 = x e.g. func(n-1)
ADD R3,R3,R1 ;sum x and y, store to R3
ADD R4,R4,#-1
LDR R1,R4,#0 ;R1=t

```

```
ADD R3,R3,R1      ;x+y+t
ADD R3,R3,#-1     ;return x+y+t-1
BRnzp Done

else
;;recurse base, just return t
ADD R3,R0,#0      ;R3=t
Done
;;deal with jump back
ADD R6,R6,#-1     ;decrement stack
LDR R7,R6,#0      ;load return address
STR R3,R2,#0      ;save Result value
ADD R2,R2,#1      ;increment result stack
RET
;;
SAVE1 .BLKW 1
SAVE2 .BLKW 1
SAVE4 .BLKW 1
SAVE5 .BLKW 1
SAVE6 .BLKW 1
ASCII .FILL xFFD0 ;数字0的补码表示
STACK .BLKW #60
TEMP .BLKW #60
T .BLKW #60
RESULT .BLKW #60
.END
```

验证

编写实现相同功能的c代码文件，使用相同输入：在0~9内取n值，然后随机取t至满足程序需要，然后对比运行结果如下：

测试数据

输入数据序列测试	C	lc3
1 2	2	x0002
3 4 5 6 7 8	28	x001c
5 1 2 3 4 9 6 7 8 9 0 1 2 3 4 5	57	x0039

第二组数据测试截图：

The LC-3 tools distribution is free s  
Public License, and you are welcom  
of it under certain conditions. The  
tools) specifies those conditions. 1  
the LC-3 tools distribution, as desc  
distributed with the tools).

Have fun.

--- halting the LC-3 ---

Input a character> 3

Input a character> 4

Input a character> 5

Input a character> 6

Input a character> 7

Input a character> 8

### LC-3 Simulator Interface

R0	x001C	R1	x0004	R2	x30E7
R4	x30AB	R5	x306F	R6	x3033
PC	x3006	IR	x10E0	PSR	x0200

Memory Addressx3006
ValuexF025

Next
Step
Finish
Continue
Stop
Clear All Breakpoint

```

x2FFE x0000 NOP
x2FFF x0000 NOP
x3000 xEC32 LEA R6,STACK
x3001 xEA6D LEA R5,TEMP
x3002 xE8A8 LEA R4,T
x3003 xE4E3 LEA R2,RESULT
x3004 x4802 JSR main
x3005 x10E0 ADD R0,R3,#0
B x3006 xF025 HALT
main x3007 x7F80 STR R7,R6,#0
x3008 x1DA1 ADD R6,R6,#1
x3009 xF023 IN
x300A x2627 LD R3,ASCII

```

```
~/Documents/icslab2 ➤ ./a.out
3 4 5 6 7 8
✗ ➤ ~/Documents/icslab2 ➤ echo $?
28
~/Documents/icslab2 ➤
```

## 2.初始化过程

见main函数，主要完成以下步骤

- caller-save，保存要被func用到的R1,2,4,5,6
- 使用Trap来读取字符，赋给R0作为func的参数
- 回复R1,2,4,5,6的保存值
- 准备R3作为返回值

```
;;main() begins
main
ST R1,SAVE1
ST R2,SAVE2
ST R4,SAVE4
ST R5,SAVE5
ST R6,SAVE6
```

```

LEA R6,STACK      ;R6 is location for return address stack
LEA R5,TEMP        ;R5 is location for n stack
LEA R4,T           ;R4 is location for t stack
LEA R2,RESULT      ;R2 is return value

STR R7,R6,#0       ;store return __start address
ADD R6,R6,#1
IN                 ; char R0 = GETC()
LD R3,ASCII
ADD R1,R0,R3       ;R1=GETC()-'0',e.g.R1 = n
JSR func
ADD R2,R2,#-1
LDR R3,R2,#0       ;now R3 is return value

ADD R6,R6,#-1
LDR R7,R6,#0       ;load return __start address
LD R1,SAVE1
LD R2,SAVE2
LD R4,SAVE4
LD R5,SAVE5
LD R6,SAVE6
RET 存储__start()返回地址
STR R7,R6,#0
ADD R6,R6,#1
;输入n参数
IN
LD R3,ASCII
ADD R1,R0,R3       ;R1=GETC()-'0',e.g.R1 = n

```

### 3.调用规范

如果其它函数使用了func(), 则需要遵守以下规范

- 由调用者使用R0作为传入参数n
- 由调用者保存会被func使用到的寄存器R1,2,4,5,6, 如本例中main函数
- func使用R3作为返回值

### 4.错误处理

程序会检测t是否符合0~9的数字范围, 比如, 如果用户输入了a, n等非法字符, 则会以-1作为返回值

```

;range test
LD R3,ASCII
ADD R0,R0,R3       ;t = GETC()-'0'
BRn BADIN         ;如果t的ASCII值小于48
ADD R3,R0,#-10
BRzp BADIN        ;如果t的ASCII值大于58
;;...
BADIN
AND R3,R3,#0       ;clear R3
ADD R3,R3,#-1      ;use -1 as output
RET

```

####