苏州大学实验报告

院、系	计算机学院	年级专业	计算机科学与技术	姓名	张昊	学号	1927405160
课程名称	微型计算机技术					成绩	
指导教师	姚望舒	同组实验者	无		实验日期	2022 年 5 月 1 日	

实验名称：实验二：基于构件方法的汇编程序设计

一. 实验目的

（1）对构件基本应用方法有更进一步的认识，初步掌握基于构件的汇编设计与运行。

（2）理解汇编语言中顺序结构、分支结构和循环结构的程序设计方法。

（3）理解和掌握汇编跳转指令的使用方法和场合。

（4）掌握硬件系统的软件测试方法，初步理解 printf 输出调试的基本方法。

二. 实验准备

（1）硬件部分。PC 机或笔记本电脑一台、开发套件一套。

（2）软件部分。根据电子资源“..\02-Doc”文件夹下的电子版快速指南，下载合适的电子资源。

（3）软件环境。按照电子版快速指南中“安装软件开发环境”一节，进行有关软件工具的安装。

三. 实验过程或要求

（1）验证性实验

① 下载开发环境 AHL-GEC-IDE。根据电子资源下“..\05-Tool\AHL-GEC-IDE 下载地址.txt”文件指引，下载由苏州大学-Arm 嵌入式与物联网技术培训中心（简称 SD-Arm）开发的金葫芦集成开发环境（AHL-GEC-IDE）到“..\05-Tool”文件夹。该集成开发环境兼容一些常规开发环境工程格式。

② 建立自己的工作文件夹。按照“分门别类，各有归处”之原则，建立自己的工作文件夹。并考虑随后内容安排，建立其下级子文件夹。

③ 拷贝模板工程并重命名。所有工程可通过拷贝模板工程建立。例如，“\04-Soft\ Exam4_1”工程到自己的工作文件夹，可以改为自己确定的工程名，建议尾端增加日期字样，避免混乱。

④ 导入工程。在假设您已经下载 AHL-GEC-IDE，并放入“..\05-Tool”文件夹，且按安装电子档快速指南正确安装了有关工具，则可以开始运行“..\05-Tool\AHL-GEC-IDE\AHL-GEC-IDE.exe”文件，这一步打开了集成开发环境 AHL-GEC-IDE。接着单击“ ”→“ ”→导入你拷贝到自己文件夹并重新命名的工程。导入工程后，左侧为工程树形目录，右边为文件内容编辑区，初始显示 main.s 文件的内容。

⑤ 编译工程。在打开工程，并显示文件内容前提下，可编译工程。单击“ ”→“ ”，则开始编译。

⑥ 下载并运行。

步骤一，硬件连接。用 TTL-USB 线（Micro 口）连接 GEC 底板上的“MicroUSB”串口与电脑的 USB 口。

步骤二，软件连接。单击“ ”→“ ”，将进入更新窗体界面。点击“ ”查找到目标 GEC，则提示“成功连接……”。

步骤三，下载机器码。点击“ ”按钮导入被编译工程目录下 Debug 中的.hex 文件（看准生成时间，确认是自己现在编译的程序），然后单击“ ”按钮，等待程序自动更新完成。

此时程序自动运行了。若遇到问题可参阅开发套件纸质版导引“常见错误及解决方法”一节，也可参阅电子资源“..\02-Doc”文件夹中的快速指南对应内容进行解决。

⑦ 观察运行结果与程序的对应。

第一个程序运行结果（PC 机界面显示情况）见图 4-7。为了表明程序已经开始运行了，在每个样例程序进入主循环之前，使用 printf 语句输出一段话，程序写入后立即执行，就会显示在开发环境下载界面的中的右下角文本框中，提示程序的基本功能。

利用 printf 语句将程序运行的结果直接输出到 PC 机屏幕上，使得嵌入式软件开发的输出调试变得十分便利，调试嵌入式软件与调试 PC 机软件几乎一样方便，改变了传统交叉调试模式。实验

步骤和结果

（2）设计性实验

在验证性实验的基础上，自行编程实现 100 以内的奇数相加所得到的和，最后通过串口输出该结果。

（3）进阶实验★

利用“Exam5_5”工程提供的一组数据，也可自行定义一组数据，采用选择排序算法对这组数据进行从小到大排序。

四、实验结果

（1）验证性实验

请问实验结果与实验过程描述是否一致？在实验过程中是否遇到过问题？如何解决的？

实验结果与实验过程描述基本一致，示例代码是递增排序。串口更新截图：

（2）设计性实验

```
//验证性实验中求 100 以内的奇数之和的代码片段
string_add:
.string "\nsum:%d\n"       //输出100以内的奇数之和
// ……
main:
// ……
//求100以内的奇数之和
    mov r0,#1                   //r0为1...100
    mov r1,#0                   //r1为sum
add_loop:
    and r2,r0,#1                //r2=r0&1=r1%2
    cmp r2,#0                   //r2等于0（r0是偶数）不加
    beq add_pass
    add r1,r1,r0                //r1+=r0
add_pass:
    add r0,r0,#1                //r0++
```

```
    cmp r0,#100                     //r0不达到100继续循环
    bne add_loop
    ldr r0,=string_add          //输出结果
    bl printf
```

结果：

（3）进阶实验★

```
//选择排序算法的代码片段
//=========================================================================
//函数名称：swap
//函数返回：无
//参数说明：r0，r1:要交换的两个数据的地址
//功能概要：将两个地址的数据交换
//=========================================================================
swap:
//（1）保存现场
      push {r0-r7,lr}              //保存现场，pc(lr)入栈
//（2）交换
      ldrb r2,[r0]              //r2=*r0
      ldrb r3,[r1]              //r3=*r1
      strb r2,[r1]              //*r1=r2
      strb r3,[r0]              //*r0=r3
//（3）恢复现场
      pop {r0-r7,pc}             //恢复现场，lr出栈到pc


//=========================================================================
//函数名称：selectionSort_up
//函数返回：无
//参数说明：r0:用于存储数据的首地址，r1:数组的长度
//功能概要：将一数组采用选择排序的方式进行升序排列，且为原地排序
```

```
//=========================================================================
selectionSort_up:
    push {r0-r7,lr}                 //保存现场，pc(lr)入栈
    //外层循环开始：每轮循环把待排序区的最小元素与待排序区的首元素交换
    mov r2, #0          //r2=0 外层循环变量
    sub r7, r1, #1      //循环终止于r7=r1-1
up_loop_outer:
    cmp r2, r7          //若r2>=r7，结束循环
    bge up_loop_outer_after
    //否则继续循环
    mov r4, r2          //最小元素的下标r4=r2
    //内层循环开始：找到最小的元素，下标存到r4
    add r3, r2, #1      //r3=r2+1 内层循环变量
up_loop_inner:
    cmp r3, r1          //若r3>=r6，结束循环
    bge up_loop_inner_after
    //否则继续循环
    ldrb r5, [r0, r3] //r5=r0[r3]
    ldrb r6, [r0, r4] //r6=r0[r4] （最小元素）
    cmp r5, r6
    //若r5>=r6，即r6=r0[r4]仍然是更小的元素，不修改
    bge up_not_change
    //若r5<r6，即r5=r0[r3]是更小的元素，修改最小元素的下标r4=r3
    mov r4, r3
up_not_change:
    add r3, r3, #1      //r3++
    b up_loop_inner         //继续循环
up_loop_inner_after:
    //内层循环结束
    //交换最小元素r0[r4]和待排序区首元素r0[r2]
    add r5, r0, r4      //r5=&r0[r4]
    add r6, r0, r2      //r6=&r0[r2]
    push {r0,r1}        //保护r0和r1
    mov r0, r5          //传递参数
    mov r1, r6
    bl swap             //调用交换函数
    pop {r0,r1}             //恢复r0和r1
    add r2, r2, #1      //r2++
    b up_loop_outer         //继续循环
up_loop_outer_after:
    //外层循环结束
    pop {r0-r7,pc}              //恢复现场，lr出栈到pc
```


教务处制

```
//=========================================================================
//函数名称：selectionSort_down
//函数返回：无
//参数说明：r0:用于存储数据的首地址，r1:数组的长度
//功能概要：将一数组采用选择排序的方式进行降序排列，且为原地排序
//=========================================================================
selectionSort_down:
    push {r0-r7,lr}              //保存现场，pc(lr)入栈
    //外层循环开始：每轮循环把待排序区的最大元素与待排序区的首元素交换
    mov r2, #0          //r2=0 外层循环变量
    sub r7, r1, #1      //循环终止于r7=r1-1
down_loop_outer:
    cmp r2, r7          //若r2>=r7，结束循环
    bge down_loop_outer_after
    //否则继续循环
    mov r4, r2          //最大元素的下标r4=r2
    //内层循环开始：找到最大的元素，下标存到r4
    add r3, r2, #1      //r3=r2+1 内层循环变量
down_loop_inner:
    cmp r3, r1          //若r3>=r6，结束循环
    bge down_loop_inner_after
    //否则继续循环
    ldrb r5, [r0, r3] //r5=r0[r3]
    ldrb r6, [r0, r4] //r6=r0[r4] （最大元素）
    cmp r5, r6
    //若r5<=r6，即r6=r0[r4]仍然是更大的元素，不修改
    bls down_not_change   //修改这里!
    //若r5>r6，即r5=r0[r3]是更大的元素，修改最大元素的下标r4=r3
    mov r4, r3
down_not_change:
    add r3, r3, #1      //r3++
    b down_loop_inner      //继续循环
down_loop_inner_after:
    //内层循环结束
    //交换最大元素r0[r4]和待排序区首元素r0[r2]
    add r5, r0, r4      //r5=&r0[r4]
    add r6, r0, r2      //r6=&r0[r2]
    push {r0,r1}        //保护r0和r1
    mov r0, r5          //传递参数
    mov r1, r6
    bl swap             //调用交换函数
    pop {r0,r1}             //恢复r0和r1
    add r2, r2, #1      //r2++
    b down_loop_outer      //继续循环
down_loop_outer_after:
```

```
    //外层循环结束
    pop {r0-r7,pc}              //恢复现场，lr出栈到pc
```

```
//main 函数中调用代码
array1:                     //定义需排序的数组
    .byte 12,15,8,14,16,2,45
    .align 1

.equ count1,7               //count1=数组array1长度，用于记录外循环次数
//……
main:
//……
//调用选择升、降序
    ldr r0,=array1
    ldr r1,=count1
    bl selectionSort_up                 //调用升序选择排序函数
    //bl selectionSort_down             //调用降序选择排序函数
```

结果：

五. 实践性问答题

（1）进阶实验中，如果要求从大到小顺序，则应修改哪些语句。

修改 label 中 up 为 down；修改内层循环中 r5, r6 比较后的转跳条件为小于等于才转跳，即：`bls down_not_change` （具体可以看上面的代码）

教务处制