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	柯骅	学号	2027405033
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	2023.05.30		

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★

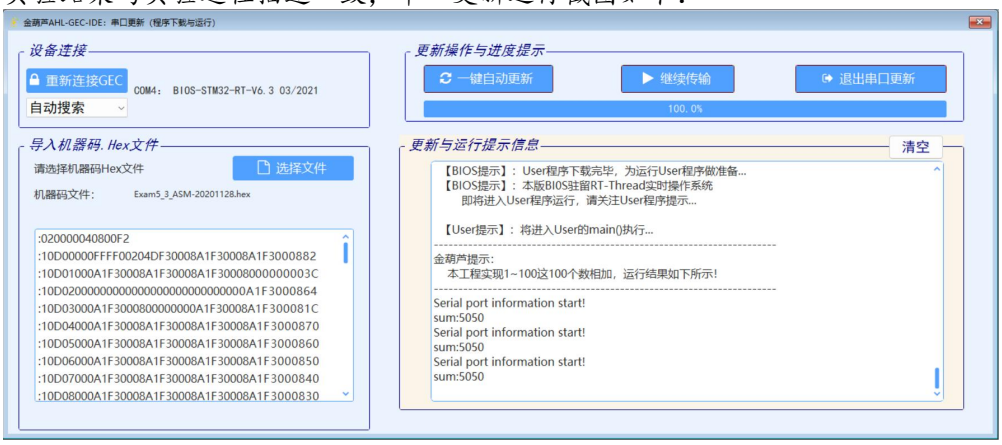
利用“Exam5_5”工程提供的一组数据，也可自行定义一组数据，采用选择排序算法对这组数据进行从小到大排序。

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一致，串口更新运行截图如下：



（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

```
1.  string_result:                          //串口输出结果前的提示信息
2.      .string "sum:%d\n"
3.  main
4.      //此处省略预处理等
5.      //计算 100 以内奇数和
6.      mov r0,#1                   //r0=1
7.      mov r1,#0                   //r1 为 sum
8.  add_loop:
9.      and r2,r0,#1                //r2=r0%2
```

教务处制

```
10.       cmp r2,#0                    //判断 r2 是否为偶数
11.       beq pass_loop                //如果为偶数，那么跳转到 pass_loop，不将 r0 加到 r1
12.       add r1,r1,r0
13.   pass_loop:
14.       add r0,r0,#1
15.       cmp r0,#100
16.       bne add_loop
17.       ldr r0, =string_result       //r0 指明字符串
18.       bl  printf                   //调用 printf
19.   .end
```

串口更新运行结果如下：

（3）进阶实验★

//选择排序算法的代码片段

//selectionSort.s

```
1.  //=========================================================================
2.  //函数名称：selectionSort_up
3.  //参数说明：r0：存放数组的首地址
4.  //          r1：数组的长度
5.  //          r2：数组的长度 r1-1，表示外层循环变量 i/r3 的最大值
6.  //          r3：外层循环变量 i
7.  //          r4：内层循环变量 j
8.  //          r5：内层循环找到的待排序区间的最小元素的下标
9.  //          r6：临时变量，存放 r0[r4]
10. //          r7：临时变量，存放 r0[r5]
11. //功能概要：将 r0 所指向的数组，长度为 r1，进行由小到大的选择排序
12. //=========================================================================
13. selectionSort_up:
14.     push {r0-r7,lr}          //入栈保存数据现场
15.     //for(i=0;i<=r1-1;i++)   外层循环开始
16.     mov r3, #0           //r3：外层循环变量 i ,初始值为 0
17.     sub r2, r1, #1       //r2：数组的长度 r1-1，表示外层循环变量 i/r3 的最大值
```

```
loop_up_outer:
    cmp r3, r2              //比较 r3/i 是否达到了 r2，
    bge loop_up_outer_done  //如果达到了，则排序完成，退出
    mov r5, r3          //r5:内层循环找到的待排序区间的最小元素的下标，初始值为 r3/i
    //for(j=i+1;j<=r1;j++)  内层循环开始
    add r4, r3, #1      //r4：内层循环变量 j
loop_up_inner:
    cmp r4, r1           //比较 r4/j 是否达到了 r1，
    bge loop_up_inner_done //如果达到了，则已找到待排序区间里最小的元素，并将其下标存到了 r5 中
    ldrb r6, [r0, r4]   //r6：临时变量，存放 r0[r4]
    ldrb r7, [r0, r5]   //r7：临时变量，存放 r0[r5]
    cmp r6, r7
    bge loop_up_keep //若 r6>=r7，即 r0[r5]仍然是更小的元素，不修改
    mov r5, r4       //若 r6< r7，即 r0[r4]是更小的元素，修改最小元素的下标 r5=r4
loop_up_keep:
    add r4, r4, #1      //r4/j++
    b loop_up_inner
loop_up_inner_done:
    //r4/j->r1，已找到待排序区间里最小的元素，并将其下标存到了 r5 中
    //交换最小元素 r0[r5]和待排序区首元素 r0[r3]
    add r6, r0, r5  //r6=&r0[r5]
    add r7, r0, r3  //r7=&r0[r3]
    push {r0,r1}        //保护 r0 和 r1
    mov r0, r6          //传递参数
    mov r1, r7
    bl swap             //调用交换函数
    pop {r0,r1}         //恢复数据
    add r3, r3, #1      //r3/i++
    b loop_up_outer
loop_up_outer_done:
    //r3/i->r2，外层循环结束
    pop {r0-r7,pc}              //恢复数据现场

//=============================================================================
//函数名称：swap
//参数说明：r0，r1:要交换的两个数据的地址
//功能概要：将 r0,r1 两个地址所指向的两个数据交换
//=============================================================================
swap:
      push {r0-r7,lr}             //入栈保存数据现场
      //将 r0 与 r1 交换
      ldrb r2,[r0]                //r2=*r0
      ldrb r3,[r1]                //r3=*r1
      strb r2,[r1]                //*r1=r2
      strb r3,[r0]                //*r0=r3
```

```
63.         pop {r0-r7,pc}                  //出栈恢复数据现场
```

//main.s:

```
1.      ldr r0,=string_first_2      //串口输出数据前的提示信息
2.      bl printf
3.      ldr r0,=uart_bef
4.      bl printf
5.      mov r7,#0
6.  //输出排序前的数组
7.  loop_bub_bef:
8.      ldr r2,=array1                  //r2=获取数组的首地址
9.      ldrb r1,[r2,r7]
10.     ldr r0,=string_control
11.     bl printf
12.     add r7,r7,#1                    //r7=r7+1，每次取完数后，地址+1
13.     ldr r6,=count1
14.     cmp r7,r6                       //比较 r7 和数组长度大小
15.     blt loop_bub_bef                //若 r7<数组长度，则没有输出完，则继续输出
16.     bge ctn
17. ctn:
18. //进行选择排序
19.     ldr r0,=array1
20.     ldr r1,=count1
21.     bl selectionSort_up             //调用选择排序函数
22.     ldr r0,=uart_aft                //排序后的信息
23.     bl printf
24.     mov r7,#0                   //需要移动的相对地址数
25.
26. loop_bub_aft:                   //串口输出数据提示信息前的显示标签
27.     ldr r2,=array1              //r2=获取数组的首地址
28.     ldr r0,=string_control
29.     ldrb r1,[r2,r7]
30.     bl printf
31.     add r7,r7,#1                //r7=r7+1
32.     ldr r6,=count1
33.     cmp r7,#7
34.     blt loop_bub_aft            //没有输出完，则继续输出
```

教务处制

串口更新运行结果如下：

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。

在从小到大排序时，我们需要在每次内层循环中寻找到待排序序列中最小的数，并记录下它的下标。

在从大到小排序时，我们只需要将维护条件修改成找到待排序区间的最大的数，即将如下代码：

```
1.    bge loop_up_keep
```

修改为：

```
1.    bls loop_up_keep
```

即可，运行结果如下：

教务处制