嵌入式系统设计实验报告

实验三

C 程序调用汇编子程序完成字符串排序

学　　号：**PE20060014**

姓　　名：王晨

专　　业：计算机科学与技术

指导老师：张辉

2020 年 12 月 10 日

一、 实验要求

1. 使用 C 语言完成字符串的输入输出
2. 使用 ARM 汇编语言完成排序操作
3. 从键盘输入至少 5 个字符串，每个字符串的长度不小于 10，在开发板正确输出结果。

二、 实验条件

1、 硬件条件：Macbook pro

2、 软件条件：Mac OS Big Sur 11.01

VMware fusion pro 虚拟机

Ubuntu 20.04.1 64 位

三、 实验过程

一、 C 程序调用 Arm 汇编程序混合编程的基本方法

在 C 程序中，可以根据需要调用汇编程序，若汇编代码较为简洁，则可使用直接内嵌汇编的方法；若汇编代码较为复杂，则需要将汇编代码以文件的形式加入到项目中，共同编译连接生成可执行文件。在 C/汇编混合编程中，设计汇编程序必须按照 ATPCS 的规定与 C 程序相互调用与访问，以保证程序调用时参数的正确传递。本实验中采用 GNU 风格的 ARM 汇编，具体规范可以参照 GNU AS 汇编器官方文档 。

二、C 程序代码段（完成输入输出功能）

完成字符串输入输出功能的 C 代码段如下图所示，通过以下声明外部汇编程序：

extern void strsort(char* strs[], int n);

后在 main 函数中调用 strsort()，该函数是完成字符串排序功能的外部汇编程序：

strsort(strs, STR_NUM);

排序完成后 strs[0]地址值将指向最小的字符串，strs[4]地址值将指向最大的字符串，然后依次输出 strs[0]- strs[4]即可完成字符串的从小到大输出。

```c
#include <stdio.h>
#include <stdlib.h>
#define STR_NUM 5
#define MAX_STRLEN 20
extern void strsort(char* strs[], int n);

int main(){
    char* strs[STR_NUM];
    //从终端输入5个20字符以内的字符串
    for (int i = 0; i < STR_NUM; i++) {
        strs[i] = (char *) malloc( size: sizeof(char) * MAX_STRLEN);
        scanf("%s", strs[i]);       //str[i]是char型指针，scanf字符串到str[i]内保存到地址起始值
    }
    //调用外部的汇编程序
    strsort(strs, STR_NUM);
    //输出排序后的字符串
    for (int i = 0; i < STR_NUM; i++)
        printf("%s\n", strs[i]);

    return 0;
}
```

三、ARM 汇编程序代码段（完成字符串比较和排序功能）

这部分的汇编程序，通过 strsort 和 strcmp 两个函数实现，strcmp 完成两个字符串的比较，方法是依次比较两个字符串第 i 个位置的单个字符，若两字符串未结束，则出现首次不相同时，返回两个字符的 ASCII 码的差值。如此，若 str1[i]>str2[i]，则返回值为正数，否则返回负数，相等返回 0。

strsort 函数完成字符串的排序功能，基本方法是通过调用 strcmp 完成比较后进行冒泡排序，若地址 strs1 所指字符串小于 strs0 地址所指字符串，则交换这两个地址值，使得 str0 内地址始终指向最小的字符串。

完成字符串比较和排序的主要汇编代码段如下：

```
.L4:
    ldr r3, [fp, #-8]        ;依次将字符串str1内的值装载入r3寄存器
    add r3, r3, #1
    str r3, [fp, #-8]
.L2:
    ldr r3, [fp, #-8]
    ldr r2, [fp, #-16]       ;将str2内对应位置的字符值装载入r2寄存器
    add r3, r2, r3
    ldrb    r2, [r3, #0]
    ldr r3, [fp, #-8]
    ldr r1, [fp, #-20]
    add r3, r1, r3
    ldrb    r3, [r3, #0]
    cmp r2, r3               ;比较r2，r3内的值
    bne .L3                  ;如果不相同，跳转到L3，进入返回值程序段
    ldr r3, [fp, #-8]
    ldr r2, [fp, #-16]
    add r3, r2, r3
    ldrb    r3, [r3, #0]
    cmp r3, #0               ;比较r3是否为结束符
    bne .L4                  ;若不是，跳转到L4继续比较下一个，否则进入返回程序段
.L3:
```

```
strsort:                             ;字符串排序程序段
    stmfd   sp!, {fp, lr}            ;保存现场
    add fp, sp, #4
    sub sp, sp, #24
    str r0, [fp, #-24]
    str r1, [fp, #-28]
    mov r3, #0
    str r3, [fp, #-8]
    b   .L6
.L10:                                ;冒泡排序
    ldr r3, [fp, #-8]
    add r3, r3, #1                   ;i++
    str r3, [fp, #-12]
    b   .L7                          ;j=i+1
.swap:                               ;swap指针程序段
    ldr r3, [fp, #-8]
    mov r3, r3, asl #2
    ldr r2, [fp, #-24]
    add r3, r2, r3
    ldr r2, [r3, #0]
    ldr r3, [fp, #-12]
    mov r3, r3, asl #2
    ldr r1, [fp, #-24]
    add r3, r1, r3
    ldr r3, [r3, #0]
    mov r0, r2
    mov r1, r3
    bl  strcmp                       ;调用strcmp函数比较r0,r1
    mov r3, r0
    cmp r3, #0
    ble .L8                          ;若相等转入L8，j++，否则执行swap
    ldr r3, [fp, #-8]
```

四、编译加载和执行

用 arm-linux-gcc 共同编译链接 C 文件和汇编文件，生成 arm 平台下的可执行文件：

```
arm-linux-gcc main.c sort.s -o main -std=c99
```

将可执行文件 main 装载到开发版的/tmp 目录下，通过 minicom 串口连接开发板，输入指令：

```
./tmp/main
```

依次输入 5 行字符串，可以看到完成了字符串的从小到大排序输出：

```
[root@FORLINX6410]# ./tmp/main
egfqgqwkbc
euiry783hj
asd99jenvee
oruoretuwe8
bmrnebnwwpp
asd99jenvee
bmrnebnwwpp
egfqgqwkbc
euiry783hj
oruoretuwe8
[root@FORLINX6410]#
```

五、实验心得

本次试验通过混合编程的方式完成了字符串排序输出，实验成功。本次实验的主要难点在于需要了解并熟悉 C 程序调用汇编子程序的基本规范、熟悉 arm 汇编的指令集。比较好的方式是先用 C 实现完整的功能，之后再将字符串比较和排序的功能部分用汇编写出，这样思路上会更加清晰。

六、附录

(1) main.c

```c
#include <stdio.h>
#include <stdlib.h>
#define STR_NUM 5
#define MAX_STRLEN 20
extern void strsort(char* strs[], int n);

int main(){
    char* strs[STR_NUM];
    //从终端输入 5 个 20 字符以内的字符串
    for (int i = 0; i < STR_NUM; i++) {
        strs[i] = (char *) malloc(sizeof(char) * MAX_STRLEN);
        scanf("%s", strs[i]);   //str[i]是 char 型指针，scanf 字符串到 str[i]内保存到地址起始值
    }
    //调用外部的汇编程序
    strsort(strs, STR_NUM);
```

```
    //输出排序后的字符串
    for (int i = 0; i < STR_NUM; i++)
        printf("%s\n", strs[i]);
    return 0;
}
```

(2) sort.c

```
    .file   "sort.s"
    .text
    .align  2
    .global strcmp
    .type   strcmp, %function
strcmp:
    str fp, [sp, #-4]!
    add fp, sp, #0
    sub sp, sp, #20
    str r0, [fp, #-16]
    str r1, [fp, #-20]
    mov r3, #0
    str r3, [fp, #-8]
    b   .L2
.L4:
    ldr r3, [fp, #-8]          ;依次将字符串 str1 内的值装载入 r3 寄存器
    add r3, r3, #1
    str r3, [fp, #-8]
.L2:
    ldr r3, [fp, #-8]
    ldr r2, [fp, #-16]         ;将 str2 内对应位置的字符值装载入 r2 寄存器
    add r3, r2, r3
    ldrb    r2, [r3, #0]
    ldr r3, [fp, #-8]
    ldr r1, [fp, #-20]
    add r3, r1, r3
    ldrb    r3, [r3, #0]
    cmp r2, r3                  ;比较 r2，r3 内的值
    bne .L3                     ;如果不相同，跳转到 L3，进入返回值程序段
    ldr r3, [fp, #-8]
    ldr r2, [fp, #-16]
    add r3, r2, r3
    ldrb    r3, [r3, #0]
```

```
    cmp r3, #0                  ;比较 r3 是否为结束符
    bne .L4                     ;若不是，跳转到 L4 继续比较下一个，否则进入返回程序段
.L3:
    ldr r3, [fp, #-8]
    ldr r2, [fp, #-16]
    add r3, r2, r3
    ldrb    r3, [r3, #0]
    mov r2, r3
    ldr r3, [fp, #-8]
    ldr r1, [fp, #-20]
    add r3, r1, r3
    ldrb    r3, [r3, #0]
    rsb r3, r3, r2
    mov r0, r3
    add sp, fp, #0
    ldmfd   sp!, {fp}           ;还原现场
    bx  lr                      ;子程序结束，返回调用位置

    .size   strcmp, .-strcmp
    .align  2
    .global strsort
    .type   strsort, %function
strsort:                        ;字符串排序程序段
    stmfd   sp!, {fp, lr}       ;保存现场
    add fp, sp, #4
    sub sp, sp, #24
    str r0, [fp, #-24]
    str r1, [fp, #-28]
    mov r3, #0
    str r3, [fp, #-8]
    b   .L6
.L10:                           ;冒泡排序
    ldr r3, [fp, #-8]
    add r3, r3, #1              ;i++
    str r3, [fp, #-12]
    b   .L7                     ;j=i+1
.swap:                          ;swap 指针程序段
    ldr r3, [fp, #-8]
    mov r3, r3, asl #2
    ldr r2, [fp, #-24]
```

```
    add r3, r2, r3
    ldr r2, [r3, #0]
    ldr r3, [fp, #-12]
    mov r3, r3, asl #2
    ldr r1, [fp, #-24]
    add r3, r1, r3
    ldr r3, [r3, #0]
    mov r0, r2
    mov r1, r3
    bl   strcmp                          ;调用 strcmp 函数比较 r0,r1
    mov r3, r0
    cmp r3, #0
    ble .L8                              ;若相等转入 L8，j++，否则执行 swap
    ldr r3, [fp, #-8]
    mov r3, r3, asl #2
    ldr r2, [fp, #-24]
    add r3, r2, r3
    ldr r3, [r3, #0]
    str r3, [fp, #-16]
    ldr r3, [fp, #-8]
    mov r3, r3, asl #2
    ldr r2, [fp, #-24]
    add r3, r2, r3
    ldr r2, [fp, #-12]
    mov r2, r2, asl #2
    ldr r1, [fp, #-24]
    add r2, r1, r2
    ldr r2, [r2, #0]
    str r2, [r3, #0]
    ldr r3, [fp, #-12]
    mov r3, r3, asl #2
    ldr r2, [fp, #-24]
    add r3, r2, r3
    ldr r2, [fp, #-16]
    str r2, [r3, #0]
.L8:
    ldr r3, [fp, #-12]           ;j++
    add r3, r3, #1
    str r3, [fp, #-12]
.L7:
```

```
    ldr r2, [fp, #-12]
    ldr r3, [fp, #-28]
    cmp r2, r3
    blt .swap                       ;若大于 0,跳转到 swap
    ldr r3, [fp, #-8]
    add r3, r3, #1                  ;j++
    str r3, [fp, #-8]
.L6:
    ldr r2, [fp, #-8]
    ldr r3, [fp, #-28]
    cmp r2, r3
    blt .L10                        ;若大于 0,跳转到 L10
    sub sp, fp, #4
    ldmfd   sp!, {fp, pc}
    .size   strsort, .-strsort
```