实验 2 ARM 和 C 相互调用实验

实验目的

- 学会使用 ADS1.2 调试信息窗口来分析判断调试过程和结果；
- 学会在 ADS1.2 环境中编写、编译与调试汇编和 C 语言相互调用的程序。

实验设备

- 硬件：PC 机
- 软件：ADS1.2+WINXP。

实验内容

- 1.使用 C 语言编写函数 str_copy(const char* src,char* dest)函数，把 src 字符串的内容拷贝到 dest，并在汇编代码中调用；
- 2.使用汇编语言编写函数 str_copy(const char* src,char* dest), 把 src 字符串的内容拷贝到 dest，并在 C 程序中调用；

实验原理

1. ARM 过程调用 ATPCS（ARM ）

ATPCS 是一系列用于规定应用程序之间相互调用的基本规则，这此规则包括：

- 支持数据栈限制检查；
- 支持只读段位置无关（ROPI）；
- 支持可读/写段位置无关（RWPI）；
- 支持 ARM 程序和 Thumb 程序的混合使用；
- 处理浮点运算。

使用以上规定的 ATPCS 规则时，应用程序必须遵守如下：

- 程序编写遵守 ATPCS；
- 变量传递以中间寄存器和数据栈完成；
- 汇编器使用-apcs 开关选项。

关于其他 ATPCS 规则，用户可以参考 ARM 处理器相关书籍或登录 ARM 公司网站。

程序只要遵守 ATPCS 相应规则，就可以使用不同的源代码编写程序。程序间的相互调用最主要的解决参数传递问题。应用程序之间使用中间寄存器及数据栈来传递参数，其中，第一个到第四个参使用 R0-R3，多于四个参数的使用数据栈进行传递。这样，接收参数的应用程序必须知道参数的个。

但是，在应用程序被调用时，一般无从知道所传递参数的个数。不同语言编写的应用程序在调用可以自定义参数传递的约定，使用具有一定意义的形式来传递，可以很好地解决参数个数的问题。用的方法是把第一个或最后一个参数作为参数个数（包括个数本身）传递给应用程序。

ATPCS 中寄存器的对应关系如图 2-9 所示

ARM 寄存器		ATPCS 别名	ATPCS 寄存器说明
R0 – R3	<==>	a1 – a4	参数/结果/scratch 寄存器 1-4
R4	<==>	v1	局部变量寄存器 1
R5	<==>	v2	局部变量寄存器 2
R6	<==>	v3	局部变量寄存器 3
R7	<==>	v4、wr	局部变量寄存器 4 Thumb 状态工作寄存器
R8	<==>	v5	ARM 状态局部变量寄存器 5
R9	<==>	v6、sb	ARM 状态局部变量寄存器 6 RWPI 的静态基址寄存器
R10	<==>	v7、sl	ARM 状态局部变量寄存器 7 数据栈限制指针寄存器
R11	<==>	v8	ARM 状态局部变量寄存器 8
R12	<==>	ip	子程序内部调用的临时(scratch)寄存器
R13	<==>	sp	数据栈指针寄存器
R14	<==>	lr	链接寄存器
R15	<==>	PC	程序计数器

图 0-1 ATPCS 中寄存器的对应关系

2. main() 函数与__main()

不使用标准 C 库函数时，可以直接跳转到 main()，当使用标准 C 库函数时，应跳转到__main()，详细解释参考理论课 PPT。

实验步骤

实验内容 1：

1) 新建工程：
2) 编辑源文件：
3) 添加源文件：
4) 配置。RO_Base 设置为 0X8000
5) 编译生成目标代码：

6) 调试运行（注意：第一次使用 AXD 调试器时，应先对其进行配置，参考理论课 PPT）

7) 单步执行程序并观察和记录寄存器与 memory 的值变化。

数据记录表

指令/C 语句	指令或 C 语句执行后变化情况					
	寄存器		存储器		变量	
	R0	R1	R0 所指向的连续存储器	R1 所指向的连续存储器	src	dest

参考源程序

SY2_1s.S 源程序：

```
    IMPORT str_copy
    AREA SY2_1_DATA,DATA,READWRITE
src_addr DCB "source",0
dest_addr SPACE 10
    AREA SY2_1_CODE,CODE,READONLY
    LDR R0,=src_addr
    LDR R1,=dest_addr
    BL str_copy
LOOP
    B LOOP
    END
```

SY2_1c.c 源程序：

```
void str_copy(const char* src,char* dest)
{
    int i=0;
    char c;
    c=src[i];
    while(c!=0)
    {
        dest[i]=c;
        i++;
        c=src[i];
    }
}
```

实验内容 2：

1) 新建工程：
2) 编辑源文件：
3) 添加源文件：
4) 配置。RO_Base 设置为 0X8000
5) 编译生成目标代码：
6) 调试运行（注意：第一次使用 AXD 调试器时，应先对其进行配置，参考理论课 PPT）
7) 单步执行程序并观察和记录寄存器与 memory 的值变化。

数据记录表

指令/C 语句	指令或 C 语句执行后变化情况						
	寄存器			存储器		变量	
	R0	R1	R2	R0 所指向的连续存储器	R1 所指向的连续存储器	src	dest

参考源程序

SY2_2s.S 源程序：

```
EXPORT str_copy
    AREA SY2_2CODE,CODE,READONLY
str_copy
    LDRB a3,[a1],#1    ;a1=R0,a3=R2 参数传递
    STRB a3,[a2],#1    ;a2=R1，参数传递
    CMP a3,#0
    BNE str_copy
    MOV PC,LR
    END
```

SY2_2c.c 源程序：

```
extern str_copy(const char* src, char* dest);
int main()
{
    const char* src="source";
    char dest[10];
    str_copy(src,dest);
    return 1;
}
```