

XI`AN TECHNOLOGICAL UNIVERSITY

实验报告

实验课程名称 ARM嵌入式系统原理及其应用开发

专 业： 物联网工程

班 级： 16060616

姓 名： 田宇龙

学 号： 16060616107

实验学时：

指导教师：

成 绩：

2018 年 6 月 18 日

西安工业大学实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 专业 | 物联网工程 | 班级 | 16 | 姓名 | 田宇龙 | 学号 | 16060616107 |
| 实验课程 | 综合实验 | 指导教师 |  | 实验日期 | 2018 | 同实验者 |  |
| 实验项目 | 综合实验 | | | | | | |
| 实验设备及器材 | PC机、Keil ，ProteusARM仿真软件，windows xp | | | | | | |

**一丶实验目的**

掌握处理器启动配置过程；

掌握使用Embest IDE 辅助信息窗口来分析判断调试过程和结果，学会查找软件调试时的故障或错误；

掌握使用Embet IDE 开发工具进行软件开发与调试的常用技巧。

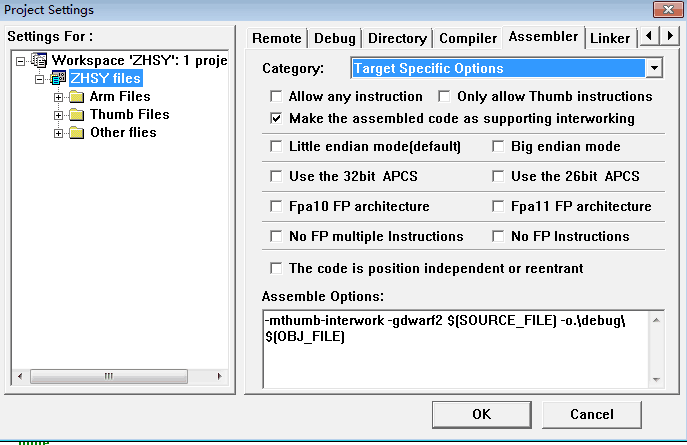
**二、实验内容**

完成一个完整的工程，要求包含启动代码，汇编函数，和C 文件，而且C 文件包含ARM 函数和Thumb 函数，并可以相互调用。

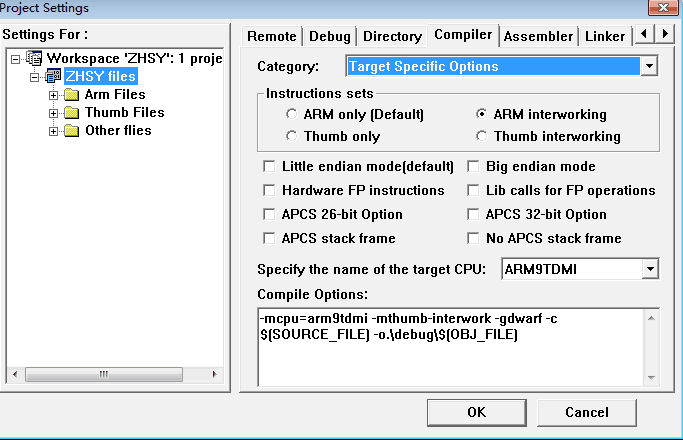
**三、实验原理**

1）. 打开实验例程目录下的ZHSY 工程，按以下步骤观察工程配置的内容：

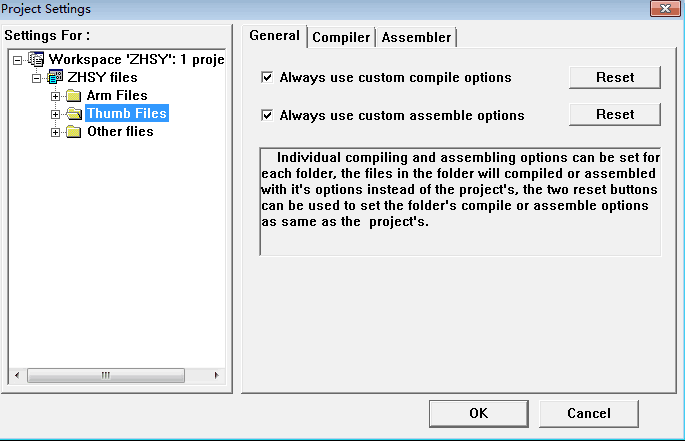
(1) 在 Assembler 页面选择 Make the assembled code as supporting, interworking 设置



(2) 在Compiler 页面选择ARM interworking 设置



(3) Thumb Files 文件组选择自定义C 与汇编语言文件输出的编译设置，并分别配置编译器和汇编器，



2）参考前面实验步骤进行编译链接interwork 工程，并下载调试；单步执行程序，通 过memory/register/watch/variable 窗口分析判断运行结果；

3）使用Embest IDE 工具Disassemble all 把elf 文件转换成objdump 文件，打开并观察代码的存放，并查看链接脚本文件中的相关代码段定义；同时查看 Thumb\_function 和 ARM\_function 标号定义的程序代码，并与实际源代码进行 比较，掌握通过objdump 文件和源程序文件相互查找问题的方法；

4.）使用Embest IDE 工具elf2bin 把elf 文件转换成可执行bin 文件，在IDE 中打开 后观察执行代码与 objdump 文件中的命令行进行比较，进一步加深对执行代码的 链接定位的理解；

5.）单步跟踪ARM 函数与Thumb 函数相互调用的反汇编代码，分析arm 内核的状态 切换过程

**四丶实验代码**

**arm.c 参考程序**

**extern char szArm[20];**

**static void delaya(int nTime)**

**{ int i, j, k;**

**k = 0;**

**for(i = 0; i < nTime; i++)**

**{ for(j = 0; j < 10; j++)**

**k++;**

**}**

**}**

**void arm\_function(void)**

**{**

**int i;**

**int nLoop;**

**unsigned int unRandom;**

**char \*p = "Hello from ARM world";**

**For(i =0; i < 20; i++)**

**szArm[i] = (\*p++);**

**delaya(2);**

**for( nLoop = 0; nLoop < 10; nLoop++ )**

**{ unRandom = randomnumber();**

**}**

**}**

**entry.s 参考程序**

**.equ count, 20**

**.extern thumb\_function**

**.global \_start**

**.text**

**\_start:**

**# Setup interrupt / exception vectors**

**b Reset\_Handler**

**Undefined\_Handler:**

**b Undefined\_Handler**

**SWI\_Handler:**

**b SWI\_Handler**

**Prefetch\_handler:**

**b Prefetch\_handler**

**Abort\_Handler:**

**b Abort\_Handler**

**nop**

**IRQ\_Handler:**

**b IRQ\_Handler**

**FIQ\_Handler:**

**b FIQ\_Handler**

**Reset\_Handler:**

**ldr sp, =0x0C002000**

**mov r0, #count**

**mov r1, #0**

**mov r2, #0**

**mov r3, #0**

**mov r4, #0**

**mov r5, #0**

**mov r6, #0**

**loop0:**

**add r1, r1, #1**

**add r2, r2, #1**

**add r3, r3, #1**

**add r4, r4, #1**

**add r5, r5, #1**

**add r6, r6, #1**

**subs r0, r0, #1**

**bne loop0**

**adr r0, Thumb\_Entry+1**

**bx r0**

**.thumb**

**Thumb\_Entry:**

**mov r0, #count**

**mov r1, #0**

**mov r2, #0**

**mov r3, #0**

**mov r4, #0**

**mov r5, #0**

**mov r6, #0**

**mov r7, #0**

**loop1:**

**add r1, #1**

**add r2, #1**

**add r3, #1**

**add r4, #1**

**add r5, #1**

**add r6, #1**

**add r7, #1**

**sub r0, #1**

**bne loop1**

**bl thumb\_function**

**b Thumb\_Entry**

**.end**

**random.s 参考程序：**

**.global randomnumber**

**.global \_\_gccmain**

**.global seed**

**.text**

**randomnumber:**

**# on exit:**

**# a1 = low 32-bits of pseudo-random number**

**# a2 = high bit (if you want to know it)**

**ldr ip, seedpointer**

**ldmia ip, {a1, a2}**

**tst a2, a2, lsr#1**

**movs a3, a1, rrx**

**adc a2, a2, a2**

**eor a3, a3, a1, lsl#12**

**eor a1, a3, a3, lsr#20**

**stmia ip, {a1, a2}**

**mov pc, lr**

**seedpointer:**

**.long seed**

**\_\_gccmain:**

**mov pc, lr**

**.data**

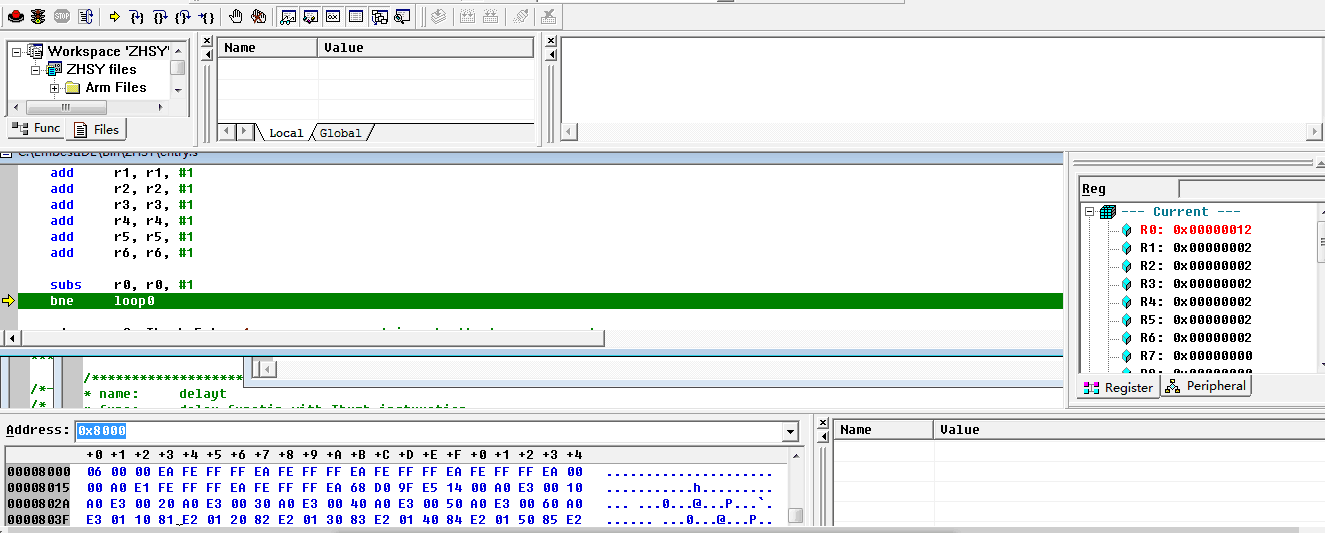
**seed:**

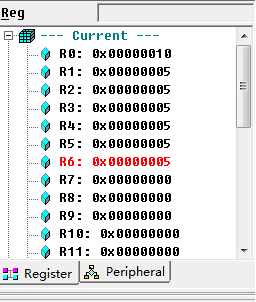
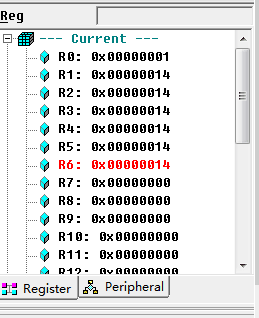
**.long 0x55555555**

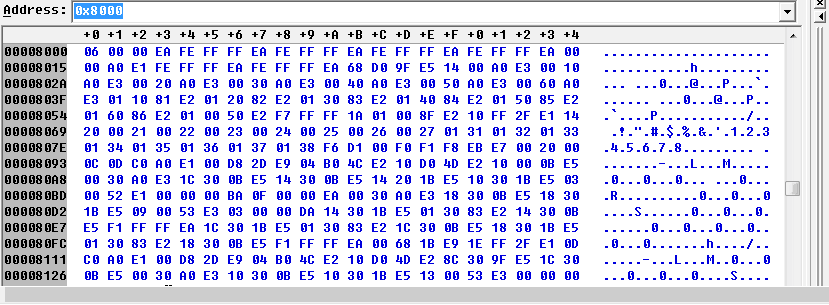
**.long 0x55555555**

**.end**

实验程序运行图如下：





**五丶心得体会**

本次实验实验较为复杂，虽然程序源代码采用指导书上的内容，但是在个人实验时仍然遇到不少问题，其中程序运行环境的配置方面便遇到不少问题，文件之间的调用关系仍然存在一些理解问题，虽然在同学的帮助下完成了本次实验，单个人认为本次实验完成不是很满意，希望接下来能够去解决相关问题，为下次实验打好基础。