

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 | ARM汇编实验 | 指导教师 |  | 实验日期 | 2018 | 同实验者 |  |
| 实验项目 | ARM汇编指令实验 | | | | | | |
| 实验设备及器材 | PC机、Embest IDE Pro 2004集成开发环境，windows xp | | | | | | |

**一丶实验目的**

初步学会使用Embest IDE for ARM开发环境及ARM软件模拟器。

通过实验掌握简单ARM汇编指令的使用方法。

1. **实验内容**

熟悉开发环境的使用并使用ldr/str，mov等指令访问寄存器或存储单元。

使用add/sub/lsl/lsr/and/orr等指令，完成基本数学/逻辑运算。

1. **实验原理**

ARM 处理器共有 37 个寄存器：

31 个通用寄存器，包括程序计数器(PC)。这些寄存器都是 32 位的。

6 个状态寄存器。这些寄存器也是 32 位的，但是只是使用了其中的 12 位。

ARM 通用寄存器

通用寄存器（R0~R15）可分为 3 类：

不分组寄存器 R0~R7；

分组寄存器 R8~R14；

程序计数器 R15；

1) 不分组寄存器 R0~R7；

R0~R7 是不分组寄存器。这意味着在所有处理器模式下，它们每一个都访问一样的 32 位寄存器。它们是真正的通用寄存器，没有体系结构所隐含的特殊用途。

1. 分组寄存器 R8~R14

R8～R14 是分组寄存器。它们每一个访问的物理寄存器取决于当前的处理器模式。若要访问特定的物理寄存器而不依赖当前的处理器模式，则要使用规定的名字。

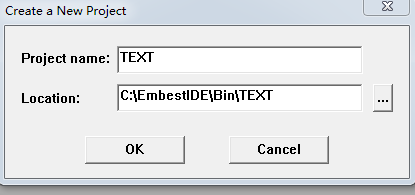
寄存器 R8~R12 各有两组物理寄存器：一组为 FIQ 模式，另一组为除了 FIQ 以外的 所有模式。寄存器 R8~R12 没有任何指定的特殊用途。只是使用 R8~R14 来简单地处理 中断。寄存器 R13,R14 各有 6 个分组的物理寄存器。1 个用于用户模式和系统模式，其它 5 个分别用于 5 种异常模式。寄存器 R13 通常用做堆栈指针，称为 SP。每种异常模式都有 自己的 R13。寄存器 R14 用作子程序链接寄存器，也称为 LR。

3) 程序计数器 R15

寄存器 R15 用做程序计数器 (PC)。

在本实验中，我们认为 ARM 核工作在用户模式 ，R0~R15 可用。

1. **实验过程**
2. 打开Embest IDE Pro软件，选择菜单项File-->New Workspace,系统弹出对话框，点击OK创建名为TEXT的新工程，并同时创建一个与工程名相同的工作区。此时在工作窗口将打开该工作区和工程。



1. 建立源文件：

点击菜单项File-->New,系统弹出一个新的、没有标题的文本编辑窗，输入光标位玉窗口中第一行，将程序所需的源文件代码输入，编辑完后，进行保存，保存文件格式为\_a.s文件。

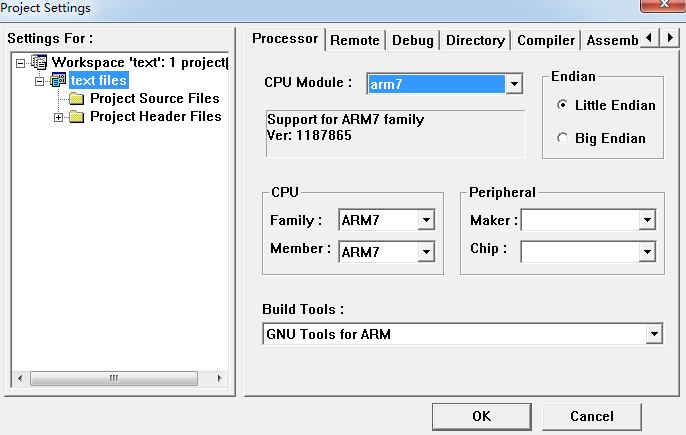
1. 添加源文件：

选择Project-->Add To Project-->File命令，弹出文件选择对话框，在工程目录下选择刚才建立的\_a.s格式的源文件



1. 基本配置：

选择菜单项Project-->Settings,弹出工程设置对话框，在工程设置对话框中，选择Processor设置对话框，选择ARM7对目标板所用处理器进行配置。



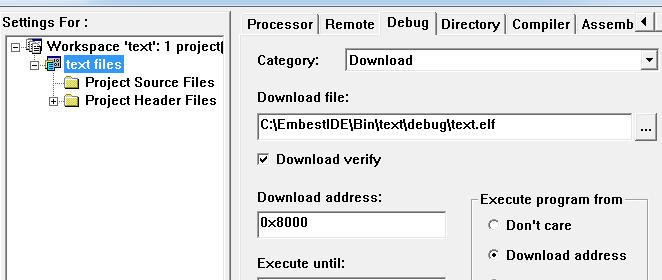
1. 生成目标代码：

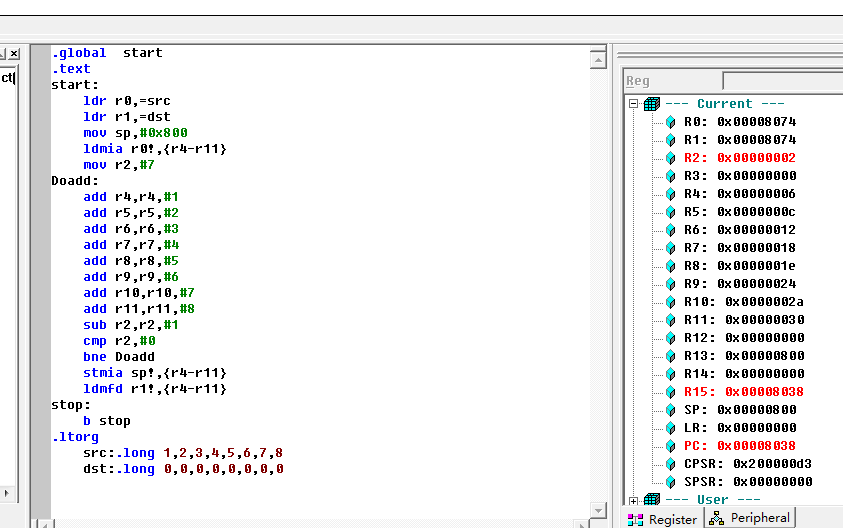
选择菜单项Build-->Build xxx\_a.s，生成目标代码。

1. 调试设置：

使用快捷键Alt+F7，弹出设置对话框，在工程设置对话框中，选择Remote设置对话框，对相应模块进行配置。

1. 选择Debug菜单Remote Connect 进行连接软件仿真器，执行Download命令下载程序，并打开寄存器窗口。打开memory窗口，观察相关内容的变化。





1. **实验程序**

编写程序循环对R4-R11进行累加8次赋值,R4-R11起始值为1-8，每次加操作后把R4-R11的内容放入SP栈中，SP初始设置为0x800，最后把R4-R11 用LDMFD指令清空赋值为0

具体程序源代码如下：

.global start ;定义全局变量START

.text ;定义文本区

start:

ldr r0,=src ;R0指向源数据的起始地址

ldr r1,=dst ;R1指向目的数据区的起始地址

mov sp,#0x800 ;将堆栈指针SP指向地址0x800

ldmia r0!,{r4-r11} ;从源数据区读取8个字的数据，放入8个寄存器并更新目标数据指针R0

mov r2,#7 ;R2=7

Doadd:

add r4,r4,#1 ;R4=R1+1

add r5,r5,#2 ;R5=R5+2

add r6,r6,#3 ;R6=R6+3

add r7,r7,#4 ;R7=R7+4

add r8,r8,#5 ;R8=R8+5

add r9,r9,#6 ;R9=R9+6

add r10,r10,#7 ;R10=R10+7

add r11,r11,#8 ;R11=R11+8

sub r2,r2,#1 R2=R2-1

cmp r2,#0 ;判断R2是否为0，若不为0则继续进行循环

bne Doadd ;循环Doadd

stmia sp!,{r4-r11} ;将R4-R11中的数据存放在SP栈中

ldmfd r1!,{r4-r11} ;对R4-R11所在寄存器进行请0

stop:

b stop

.ltorg

src:.long 1,2,3,4,5,6,7,8 ;定义源数据区

dst:.long 0,0,0,0,0,0,0,0 ;定义目标数据区

**五丶心得体会**

第一次完成ARM汇编实验，了解和认识了ARM汇编语言的一些相关知识和相关开发环境，然而在程序编码上仍需要花费较大的努力，充分理解例题思路，做到举一反三。希望下次的实验能有更丰富的收获。