**《微控制器原理、接口与应用》**

**2019年冬季学期课堂作业（2019.11.25）**

**1、简述ARM920T的工作模式，简述ARM920T寄存器，并对比说明ARM状态下寄存器集和Thumb状态下寄存器集。**

ARM920T有7种工作模式

除了用户模式外，其他模式可以自由地访问系统资源和改变模式，称为特权模式, FIQ，IRQ、超级用户、异常中止和未定义这5种模式称为异常模式，分别对应特定的异常出现时进入的模式每种特权模式都有某些附加的通用或状态寄存器，减少模式切换时对现场的保护.

**2、简述ARM处理器寻址方式。**

立即数寻址

寄存器寻址

寄存器间接寻址

寄存器偏移寻址

基址变址寻址

多寄存器寻址

堆栈寻址

**3、分类列出ARM伪指令，并简单说明该伪指令功能。**

* + **(1) 符号定义伪指令**

(1) 全局变量声明: GBLA、GBLL和GBLS

(2) 局部变量声明: LCLA、LCLL和LCLS

(3) 变量赋值: SETA、SETL和SETS

(4) 为一个通用寄存器列表定义名称: RLIST

(5) 为一个协处理器的寄存器定义名称: CN

(6) 为一个协处理器定义名称: CP

(7) 为一个VFP寄存器定义名称: DN和SN

(8) 为一个FPA浮点寄存器定义名称: FN

* + **(2) 数据定义伪指令,用于数据表定义、文字池定义、数据空间分配等**

(1) 声明一个文字池: LTORG 。

(2) 定义一个结构化的内存表的首地址: MAP

(3) 定义结构化内存表中的一个数据域: FIELD

(4) 分配一块内存空间，并用0初始化: SPACE

(5) 分配一段字节的内存单元，并用指定的数据初始化: DCB

(6) 分配一段字的内存单元，并用指定的数据初始化: DCD和DCDU

(7) 分配一段字的内存单元，将每个单元的内容初始化为此单元相对于静态基址寄存器的偏移量: DCDO

(8) 分配一段双字的内存单元，并用双精度浮点数据初始化: DCFD和DCFDU

(9) 分配一段字的内存单元，并用单精度浮点数据初始化: DCFS和DCFSU

(10) 指定内存单元存放的是代码，而不是数据: DCI

(11) 分配一段双字的内存单元，并用64位整数数据初始化: DCQ和DCQU

(12) 分配一段半字的内存单元，并用指定的数据初始化: DCW和DCWU

* + **(3) 报告伪指令**

(1) 断言错误: ASSERT

(2) 汇编诊断信息显示: INFO

(3) 设置列表选项: OPT

(4) 插入标题: TTL和SUBT

* + **(4) 汇编控制伪指令,** **用于条件汇编、宏定义、重复汇编控制等**

(1) 条件汇编控制: IF ELSE 和ENDIF

(2) 宏定义: MACRO和MEND

(3) 重复汇编: WHILE及WEND

* + **(5) 杂项伪指令,**

(1) 条件汇编控制: IF ELSE 和ENDIF

(2) 宏定义: MACRO和MEND

(3) 重复汇编: WHILE及WEND

(1) 边界对齐: ALIGN

(2) 段定义: AREA

(3) 指令集定义: CODE16和CODE32

(4) 汇编结束: END

(5) 程序入口: ENTRY

(6) 常量定义: EQU

(7) 声明一个符号可以被其他文件引用: EXPORT和GLOBAL

(8) 声明一个外部符号: IMPORT和EXTERN

(9) 包含文件: GET和INCLUDE

(10) 包含不被汇编的文件: INCBIN

(11) 保留符号表中的局部符号: KEEP

(12) 禁止浮点指令: NOFP

(13) 指示两段之间的依赖关系: REQUIRE

(14) 堆栈8字节对准: REQUIRE8和PRESERVE8

(15) 给特定的寄存器命名RN

(16) 标记局部标号使用范围的界限: ROUT

**4、基于ARM920T处理，编写一段ARM状态的汇编指令，功能是将一个由5位十进制数字字符组成的字符串，转化为整数的代码，例如将字符串“12345”转换为数值12345。**

**5、编写一个函数，实现 S= 1 + 2 + 3 + ... + N, N作为参数存储在R0寄存器。**

**6、把下面C代码转换成汇编代码。 数组a和b分别存放在以0x4000和0x5000位起始地址的存储区内，类型为long（即32b）。**

**for （i=0； i<10； i++）**

**｛ a[i]=a[i] + b[i]; ｝**

**7、用汇编程序编写函数，实现如下功能： 1！+2！+3！+ ... + N!, 其中N作为参数存储在R0寄存器（建议先编写函数实现N！，然后再实现题目要求的函数）。**

**8、简述ARM处理器的异常中断。**

**9、简述ARM异常中断响应过程，异常中断返回过程。**

响应过程: 1. ★将下条指令的地址自动保存到相应的链接(R14)寄存器中 ★复制CPSR到相应的SPSR ★根据异常类型强制改变CPSR模式位的值 ★PC从相关的异常向量来取得异常处理程序入口地址 ★这时也可能设置中断禁止标志，以防止不可控的异常嵌套发生。当处理器处于Thumb状态时发生了异常，将自动地切换 到ARM状态

返回过程: 复制程序状态字从SPSR到CPSR 

如果在进入中断时设置了中断禁止标志，则清除它 

将链接寄存器的值调整后(由指令执行的硬件微体系结构特殊性导致) 赋给PC返回

**2019微控制器 选择题20分**

**涉及到异常对应地址、输入输出端口、一些指令的含义（附录ABC可以带进去所以不用记）、R1=0x31，R2=0x02，ADD R0 R1 R2 LSL 3，问R0的值、后面的不记得了（忘得太快哈哈）。**

**简答题 28分 l**

**1.简述ARM处理器寻址方式 (考了两次)**

**答案:**

1.立即数寻址, **操作数本身包含在指令中**，只要取出指令也就取到了操作数。这个操作数叫做立即数，对应的寻址方式叫做立即寻址

2.寄存器寻址, 就是**利用寄存器中的数值作为操作数**，也称为**寄存器直接寻址**

3.寄存器间接寻址, 就是把**寄存器中的值作为地址**，再通过这个地址去取得操作数，**操作数本身存放在存储器中**。

4.寄存器移位寻址,也叫偏移寻址, 这是ARM指令集特有的寻址方式，**它是在寄存器寻址得到操作数后再进行移位操作**，得到最终的操作数。

5.基址变址寻址, 寄存器基址变址寻址又称为**基址变址寻址**, 它**将寄存器（该寄存器一般称作基址寄存器）中的值与指令中给出的地址偏移量相加，从而得到一个地址，通过这个地址取得操作数**。

6.多寄存器寻址, 可以**一次完成多个寄存器值的传送**

7.堆栈寻址, 堆栈是一种数据结构，按先进后出（First In Last Out，FILO）的方式工作，使用堆栈指针（Stack Pointer, SP）指示当前的操作位置，堆栈指针总是指向栈顶。有四种堆栈类型.

8. 相对寻址,由PC提供基准地址, 指令中的地址码字段作为偏移量, 两者相加后得到的地址即为操作数的有效地址

**2.简述ARM920T寄存器，并对比说明ARM状态下寄存器集和Thumb状态下寄存器集**

**答案:**

ARM 共有 37 个 32 b 的寄存器 其中 3 1个是通用寄存器 . 6 个是状态寄存器 。 但在同一时间，对程序员来说并不是所有的寄存器都可见，在某一时刻寄存器是否可见(可被访问 ) ，是由处理器当前的工作状态和工作模式决定的。

通用寄存器可以分为三类， 即未分组寄存器、分组寄存器和程序计数器PC。RO~R7 是未分组寄存器。 这意味着在处理器的所有工作模式下， RO~ R7 所对应的物理寄存器地址是相同的，没有体系结构隐含的特殊用途.

R8~ R14 是分组寄存器,它们所访问的物理寄存器取决于当前处理器的工作模式 。寄存器 R8~ R12 有两组物理寄存器 : 一组为 FIQ 模式 另 一组为除 FIQ 以外的其他模式。 寄存器 R8~ R12没有任何指定的特殊用途 。

寄存器 R 15 用做程序计数器 PC,R 1 5 用于控制程序中指令执行的顺序 。 正常运行时 .PC 指向 CPU 执行的下一条指令 。 每次取指后 PC 的值会自动修改以指向下一条指令 . 从而保证指令按一定的顺序执行 。

Thumb 状态下寄存器是 ARM 状态下寄存器的一个子集。程序员可以直接操作 8 个通用寄存器 R0~R7 。 同样可以操作程序计数器 ( PC) 、堆栈指针寄存器 (SP ) 、链接寄存器( LR) 和 CPSR 。 它们都是各个特权模式下的私有寄存器 。

在 Thumb 状态下访问高地址寄存器:在 Thumb 状态下寄存器 R8~R15( 高地址寄存 器)不是标准寄存器集 。 但是，汇编语言的程序员可以访问它们并用它们进行快速暂存 。 向 R8~ R15 写入或读出数据，可以采用 MOV 指令的某个变型 ， 从 RO~ R7(低地址寄存器)的 某个寄存器传送数据到高地址寄存器，或者从高地址寄存器传送到低地址寄存器 。 还可以 采用 CMP 和 ADD 指令，将高地址寄存器的值与低地址寄存器的值进行比较或相加 。

3.简述ARM920T中T、D、S、I分别对应？

Thumb 16b译码器

D JTAG 调试器

S 可综合版本

I 嵌入式跟踪宏单元

4.简述ARM异常中断进入过程，异常中断退出过程。

响应过程: 1. ★将下条指令的地址自动保存到相应的链接(R14)寄存器中 ★复制CPSR到相应的SPSR ★根据异常类型强制改变CPSR模式位的值 ★PC从相关的异常向量来取得异常处理程序入口地址 ★这时也可能设置中断禁止标志，以防止不可控的异常嵌套发生。当处理器处于Thumb状态时发生了异常，将自动地切换到ARM状态.

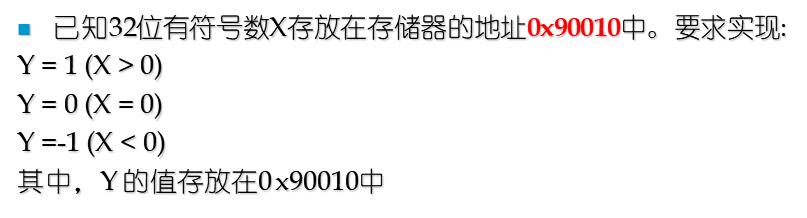
返回过程: 复制程序状态字从SPSR到CPSR 

如果在进入中断时设置了中断禁止标志，则清除它 

将链接寄存器的值调整后(由指令执行的硬件微体系结构特殊性导致) 赋给PC返回

程序题 28分 1 SWI软中断编程添加对应注释

2



3 编写一个函数，实现 S= 1 + 2 + 3 + ... + N, N作为参数存储在R0寄存器，S存储在R1寄存器。

4. 把下面C代码转换成汇编代码。 数组a和b分别存放在以0x4000和0x5000位起始地址的存储区内，类型为long（即32b）。 for （i=0； i<10； i++） ｛ a[i]=a[i] - b[i]; ｝ 编程设计1 12分

P0[1:0]对应开关1、2，P0[7:6]对应LED1、

2，编写程序使开关x按下时，对应ledx亮

## 编程设计2 12分

50个数据，首地址为SORT\_data，编写冒泡排序的程序（从小到大）

以下是微控制器这门课的吐槽和复习建议：

这门课好像是从16级开始的，上课基本是老师念ppt（和课本基本一样，很多人基本不听，没点名，建议跟着自己的节奏学，明示\*\*\*（taokeba），哈哈），中间穿插一次课程作业和课程实践的三次课，这次考试的题有几道和课程作业相似，考试覆盖范围基本上是前7章和第12章，课程实践今年占比是30%，分布为13-20（\*1.5后四舍五入），还是建议好好做吧（虽然我搞不懂没有ARM芯片硬件课程实践的意义在哪），提示一下，今年课程实践平台大多是Keil，得高分的项目都采用与proteus联合仿真（比如计算器计算开方和sin函数），最好不要纯硬件，希望下一届在这门课上取得好成绩吧。

有限的人生不应该浪费在 ,学习keil5的怎么编译上,怎么引入路径.

应该多学基础知识,学习汇编语言的精妙, 学习各种数学, 编程算法. 而这垃圾课只是在和keil5等垃圾软件 浪费十几个小时.