微嵌答案 第五章

5.1 计算机中的“ISA”和“μarch”各是什么意思？两者之间有何联系？

- ISA：指令集体系结构，用来描述软件如何使用硬件的一种规范和约定
- μarch：微架构，对应于ISA 的硬件实现方式
- 相同的ISA的处理器可以有不同微架构，ISA是介于微架构硬件实现和软件的桥梁

5.2 请简述哈佛结构的主要优缺点。

- 优点：取指和数据存取可以通过两套独立的总线同时进行，从而减小指令流水线发生资源冲突的概率
- 缺点：哈佛结构较为复杂，与外设以及扩展存储器的连接难度较大

5.3 TCM与高速缓存Cache有什么区别？

TCM具有物理地址，需要占用内存空间，无cache的不可预测性

5.5 ARM指令集、Thumb指令集和Thumb-2指令集之间的主要区别是什么？

- ARM指令集是32位，
- thumb指令集只有16位
  - 虽然thumb指令集在指令功能方面不如ARM全面，但是提高了代码密度，降低系统成本
  - 另外大多数外设的接口都是8位或者16位的，利用thumb指令集可以提高传输效率
- Thumb-2指令集结合了thumb和ARM指令集的优点

5.6 MMU和MPU的功能有何异同？

- MMU：内存分页管理+分区域访问权限管理+虚拟地址VA到物理地址PA的转换，适用于多用户系统
- MPU：内存分区域访问权限管理，适用于要求对处理时间有明确要求的实时系统

MMU与MPU均用于内存管理，且基本功能大体上一致，但MMU较MPU更为先进，具有MPU所不具备的内存分页转化和虚拟地址到物理地址的转换，覆盖MPU的功能且开销更低，适用于多系统

5.9 Cortex-M系列处理器定义的存储器映射关系是固定不变的，这样做有何利弊？

利：统一的地址映射方案有助于提高基于Cortex-M设备之间的软件可移植性和代码可重用性

弊：某些内存区域使用受限，一定程度上降低了灵活性，例如程序不允许在外设、设备和系统存储器区域中执行、通过系统总线从SRAM和RAM域执行程序代码时性能会稍微低些

5.10 Cortex-M3与Cortex-M4使用两个堆栈的目的是什么？在中断响应时，程序断点和程序状态寄存器的内容保存在哪个堆栈中？

- 使用两个堆栈是为了服务于不同的操作模式和特权访问等级，处理模式总是使用MSP，线程模式可以使用MSP或PSP
- 程序断点和程序状态寄存器的内容保存在MSP中

5.11 Cortex-M3/M4的CODE区选用总线互连矩阵与总线复用器有什么区别？

- 总线矩阵：I-CODE对FLASH的取指操作与D-CODE对SRAM的数据存取操作可以同时进行
- 总线复用器：I-CODE和D-CODE对CODE区域的访问只能分时进行，数据传送不在具有并行性

5.13 Cortex-M3/M4从SRAM域读取指令执行时有什么缺点?

从SRAM区域读取指令和从CODE区域利用I-CODE读取指令相比，效率较低

5.14 I-Code和D-Code总线全部连接到同一片Flash芯片上会有什么问题?

会产生冲突，无法做到并行操作，降低效率

5.15 私有外设总线（Private Peripheral Bus，PPB）基于哪种总线协议，有何特点?

- 32位APB总线
- 特点:
  - 外部私有外设子区域0xE0040000~0xE00F FFFF
  - 在外部私有外设子区域中，有一部分空间已被ETM、TPIU和ROM表等调试组件所占用，只有0xE004 2000~0xE00F EFFF之间可用于连接外部私有设备
  - 由于内核私有区域需要特权访问权限，该总线一般是专门用于连接调试组件，不用于普通的外设，否则将会出现因特权管理导致的各种错误

5.16 如果非特权线程试图访问内核私有区域，将会导致哪一类异常？如果Cortex-M3使用了一条SIMD运算指令，结果又将如何?

- 编号4MemManage错误
- 由于cortex-M3没有浮点运算，DSP等协处理器，所以会触发用法错误（编号6）

5.17 在Cortex-M3/M4中，寄存器R0~R12有何异同？如果这些寄存器都是空闲的，你觉得首先使用哪些？为什么?

- R0~R7 低位寄存器(许多16位的thumb指令只能访问低位寄存器)
- R8~R12高位寄存器(可用于32位指令和少数几个16位指令)

- 相同点：都是通用寄存器
- 不同点：为了能够实现汇编程序与C语言程序的相互调用，ARM公司制定了AAPCS（ARM Architecture Procedure Call Standard）规范：R0~R3用于子程序之间的参数传递；R4~R11用于保存子程序的局部变量；R12作为子程序调用中间寄存器
- 如果这些寄存器都是空闲的，优先使用低位寄存器，低位使用较为频繁，优先低位可以降低功耗

5.19 某段程序需要跳转到0x0100 0000执行，有人写了如下两行汇编指令代码:

```
MOV    R0,     #0x0100 0000
MOV    R15,    R0
```

请问这样会有什么问题?

无论使用跳转指令还是直接写PC寄存器，写入值必须是奇数，确保其最低位是“1”，以表示其处于Thumb状态，否则将被认为试图转入ARM模式，从而导致出现错误异常

5.20 请说明特殊寄存器PRIMASK和FAULTMASK寄存器的异同。

- 异：与PRIMASK不同的是，FAULTMASK无需主动清理，当错误处理程序运行结束返回时，会自动复位FAULTMASK。PRIMASK，当最低位被置位（写入1）后，将屏蔽除复位、NMI和硬件错误以外所有的（优先级数值大于0的）系统异常和外部中断
- 同:
  - FAULTMASK硬件错误异常也被屏蔽
  - 都属于实现1位基于优先权异常/中断寄存器

5.22 某基于Cortex-M4的SOC芯片共有64级外部中断，BASEPRI寄存器的宽度共有几位？如果想屏蔽所有优先级大于16的中断，请写出对BASEPRI寄存器进行设置的汇编指令。如果想屏蔽所有优先级大于0的中断，又该如何设置?

BASEPRI:7:2共6位

```
MOV R0, #0x0000 0044 或者 MOV R0, #0b 010001(00)
MOV BASEPRI, R0

MOV R0,#0x 0000 0001
MOV PRIMASK,R0
```

5.23 有人写了一段对Cortex-M4的进程栈进行初始化的代码，其中PSP的初始值设为0x8765 4321，并且使用了如下一条语句：“MOV　PSP，R0”对PSP进行赋值（其中R0=0x8765 4321）。这样做存在哪些问题？请逐一说明。

- 由于堆栈操作是以字为单位的，所以堆栈要字对齐，而PSP的初始值并没有做到字对齐
- MSP，PSP只能用特殊寄存器指令MRS,MSR指令访问

5.25 在特权线程模式下如何切换到非特权线程模式？在非特权线程模式下能否采用类似方法切换到特权线程模式？为什么？

- 在特权线程模式下可以直接修改CONTROL寄存器nPRIV=1，就可以直接进入非特权线程模式
- 在非特权线程模式下，首先通过异常状态进入异常处理，在异常处理阶段修改CONTROL寄存器nPRIV=0，然后就进入特权线程模式

5.29 Cortex-M3存储空间的哪些区域支持位段（bit-band）操作？

SRAM区域，片上外设区域支持bit-band操作

5.31 写出利用位段操作读取0x4000 1000的第3位的代码。

```
LDR R0，=0x4202 0008
LDR R1，[R0]
```

5.32 存储器访问属性包括哪些？

- 可缓冲：当处理器继续执行下一条指令时，对存储器的写操作可以由写缓冲执行
- 可缓存：读存储器所得到的数据可被复制到缓存，下次再访问时可以从缓存中取出这个数值，从而加快程序执行
- 可执行：所得到的数据可被复制到缓存，下次再访问时可以从缓存中取出这个数值从而加快程序执行
- 可共享：这种存储器区域的数据可被多个总线主设备共用。存储器系统需要在不同总线主设备之间确保可共享存储器区域数据的一致性

5.35 Cortex-M系列处理器不会改变代码的执行顺序，因而不需要存储器屏障指令，这个观点对吗？为什么？

不正确。由于cortex-M系列存储器引入缓存，虽然存储器系统不会改变指令执行顺序，但是顺序执行的指令的存储器访问操作完成时间先后顺序不定，因此需要存储器屏障指令来保证代码的执行顺序

5.36 处理器进入异常处理子程序之前保护现场需要把哪些寄存器的值保护起来？

PSR，PC，LR，R0~R3，R12

5.38 解释Cortex-M处理器的中断优先级分组机制。

- 8位的优先级配置寄存器分为两部分：分组优先级和（组内）子优先级
- 由于分组优先级和子优先级可以配置不同宽度的组合，有效提高中断优先级配置的灵活度
- 分组优先级对应抢占优先级，在相同的分组优先级下，具有更高子优先级的异常会被优先处理

5.39 解释向量表重定位机制。

- 向量表重定位使用VTOR指示向量表的位置，保存向量表相对于存储器的偏移量
- 处理器通过修改VTOR值修改向量表的起始位置，从而实现向量表重定位