第 5 章作业

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

5.4 什么是饱和运算？试举例说明饱和运算的作用。

饱和运算是指当计算发生溢出时，计算结果等于最大或最小的可表示范围，避免出现更大的误差。

例子：单字节无符号数进行运算，0xF1+0x35 应该是等于 0x126，但由于结果大于 255，那么饱和运算的结果就是 0xFF。

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。

Memory Management Unit 和 Memory Protection Unit，均是用于内存管理，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。

5.7 Cortex-R5、R7、R8 和 R52 处理器中，采用异构双核或者异构多核结构的主要目的是什么？

使得上述处理器可以在单处理器情况下实现锁步技术。

5.8 除了可以选配 FPU 以外，Cortex-M4 与 Cortex-M3 在指令功能上还有哪些不同？

Cortex-M4 支持 Cortex-M3 的所有指令，但额外增加了 DSP 扩展功能，例如：

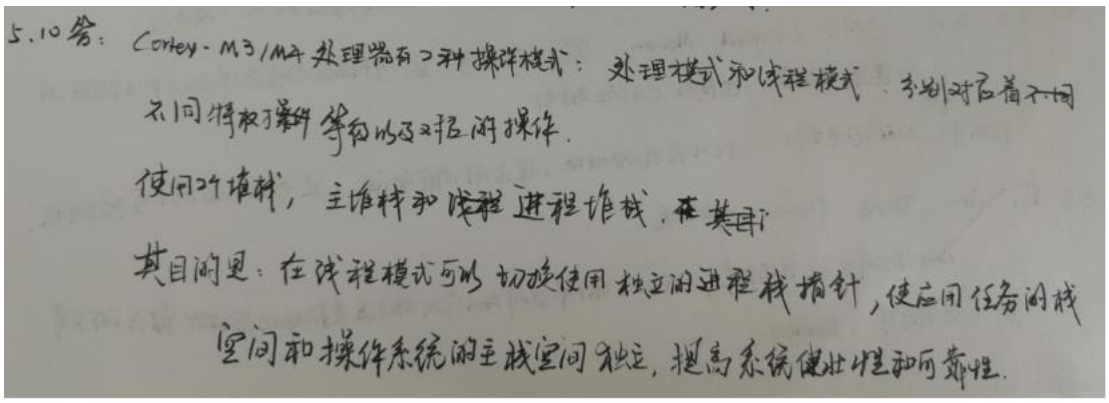
（1）增加了支持 8 位和 16 位数据的 SIMD 指令，允许对多个数据同时进行并行处理；

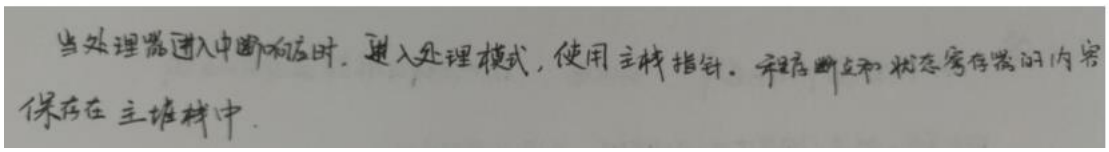
（2）支持多个（包括 SIMD 在内的）饱和运算指令，避免在出现上溢出和下溢出时计算结果出现较大的畸变；

（3）支持单周期 16 位、双 16 位以及 32 位的乘加（MAC）运算。

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。

5.12 有些芯片制造商将所有的数据集中存储在 SRAM 区，试分析这种方案的利弊。

利：将数据统一存放到由系统总线连接的片上 SRAM 中，利用系统总线与 I-Code 总线的并行性进行数据传送。减少一块 SRAM，CODE 区只需使用相对简单和低成本的总线复用器，降低成本。

弊：占用系统总线资源，浪费了小部分存储空间。

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。

5.16 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 4 MemManage 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。

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。

R0~R7 是 8 个低位寄存器，R8~R12 是 5 个高位寄存器，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。优先低位寄存器，低位使用较频繁，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。

同：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, #0x 0000 0044 或者 MOV R0, #0b 010001

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。

补充 1：假设某 MCU 实际上有 16 个外部中断，试分析当 AIRCR 中位[10:8]分别为 3、4 和 5 时，分组优先级和抢占优先级的数量（可画图表示）

AIRCR 中位[10:8]取 3：

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
分组优先级				未使用			

AIRCR 中位[10:8]取 4：

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
分组优先级			子优先级	未使用			

AIRCR 中位[10:8]取 5：

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
分组优先级		子优先级		未使用			

补充 2：假设某 MCU 实际上有 32 个外部中断，如果向量表需要重定位，试分析向量表合法的起始位置是哪些（说明需要对齐的边界）。如果向量表重定位到 SRAM 区，请写出 VTOR 中一种合理的数值

32+16=48，48×4=192，大于 192 的最小的 2 的整数次幂是 256，因此向量表起始地址是 0x100 的整数倍，如 0x0000 0000、0x0000 0100、0x0000 0200
VTOR 的值为 0x2000 0100