# 1.ARM处理器与中断（15道）

## 1.1 CPU的内部结构?

CPU的内部结构大致可以分为:

（1）控制单元（指令寄存器、指令译码器、操作控制器）。

（2）运算单元（算术逻辑单元）。

（3）存储单元（专用寄存器和通用寄存器）

（4）时钟。

## 1.2 CPU跟内存、虚拟内存、硬盘的关系?

（1）CPU要调用的程序和数据来自硬盘，但是CPU又不能直接读写硬盘上的系统、程序和数据，所以必须先将硬盘的内容存储在内存中，才能被CPU读写。

（2）因此内存是一个中转站，对计算机的运行速度有较大影响。

（3）当系统需要的内存空间大于实际的物理内存空间时，就需要用到虚拟内存了。虚拟内存可以将部分硬盘空间模拟成内存空间，将暂时不运行的程序和不使用的数据存储在硬盘上，需要时再将其存储到内存。

## 1.3 ARM结构处理器可分为哪几类?

| | |
|---|---|
| 嵌入式微处理器 | 32位以上的处理器，具有较高的性能。 |
| 嵌入式微控制器 | 典型代表是单片机，一般以某一种微处理器内核为核心，芯片内部集成Flash、RAM、EEPROM、总线、总线逻辑、定时/计数器、串行接口等各种必要的功能模块。 |
| 嵌入式DSP | 对系统结构和指令进行了特殊设计，使其适合于执行DSP算法，编译效率较高，执行执行速度也较快。 |

## 1.4嵌入式微处理器和DSP有什么区别?

（1）嵌入式微处理器和DSP一个偏重控制、一个偏重运算。

（2）嵌入式微处理器外围接口丰富，标准化、通用性、功耗控制等做得很好，适用于消费电子、家用电器等控制领域。

（3）DSP对系统结构和指令做了优化，能进行大量数据的快速计算，适用于音视频处理等领域。

### 1.5 ARM处理器有哪些工作状态？ARM指令和Thumb指令有什么区别？

**答案：**

（1）ARM处理器共有ARM、Thumb/Thumb-2、调试三种状态。

（2）ARM指令是32位的，较全面；Thumb指令是16位的，较精简。

**解读：**

| | |
|---|---|
| ARM状态 | 工作于32位指令状态，所有指令均为32位。 |
| Thumb状态 | 工作于16位指令状态，所有指令均为16位。 |
| Thumb-2状态 | ARM状态和Thumb状态是早期版本，近期推出的Thumb-2状态兼有16和32位指令，具有更高的性能、更低的功耗以及更少的内存占用。具有Thumb-2技术的ARM处理器无需在ARM和Thumb-2状态之间切换了。 |
| 调试状态 | 处理器停机调试。 |

### 1.6 RISC（精简指令集计算机）和CISC（复杂指令集计算机）的区别？

（1）RISC控制器多采用硬件连线控制方式，以期更快的执行速度；而CISC控制器绝大多数采用微程序控制方式。

（2）RISC只有加载和存储指令可以访问内存，数据处理指令只对寄存器的内容操作，为了加速程序的运算，RISC会设置多组寄存器，并指定特殊用途的寄存器，因此通用寄存器数量较CISC多。CISC架构允许数据处理指令对内存进行操作，因此需要的寄存器数量比较少。

（3）RISC大多指令在一个时钟周期内完成，且指令长度统一、数目少；而CISC的复杂指令通过CPU内的微码来完成，需要多个时钟周期，且指令长度不一、数目多。

（4）RISC在实现一个功能的时候，需要的指令数目多，编译器设计就更复杂；CISC的指令丰富，它的编译器可以少做很多事情。

（5）RISC较CISC更易实现指令流水线，因为其指令大多在一个时钟周期内完成。

### 1.7 ARM内部传输数据的总线有哪些?

ARM内部传输数据采用AMBA总线，共有四个版本：

（1）AMBA1：ASB、APB；

（2）AMBA2：AHB、APB；

（3）AMBA3：AHB、AXI、ATB、APB；

（4）AMBA4：AXI、ATB、ACE、APB。

### 1.8中断是嵌入式系统中重要的组成部分，这导致了很多编译开发商提供一种扩展：让标准C支持中断。具体表现是，产生了一个新的关键字__interrupt。下面的代码就使用了__interrupt关键字去定义了一个中断服务子程序(ISR)，这段代码有什么问题?

```
__interrupt double compute_area (double radius)
{
    double area = PI * radius * radius;
    printf(" Area = %f", area);
    return area;
}
```

**答案：**

（1）对于单片机来说，ISR不能传递参数。

（2）对于单片机来说，ISR不能有返回值。

（3）在许多的处理器/编译器中，浮点运算一般都是不可重入的。此外，ISR应该是短而有效率的，在ISR中做浮点运算是不明智的。

（4）与第三点一脉相承，printf()经常有重入和性能上的问题。

**解读：**重入一般可以理解为一个函数在同时多次调用，例如操作系统在进程调度过程中，或者单片机、处理器等的中断的时候会发生重入的现象。满足下面条件之一的多

数是不可重入函数：

（1）使用了静态数据结构。

（2）调用了malloc或free。

（3）调用了标准I/O函数，标准IO库很多实现都以不可重入的方式使用全局数据结构。

（4）进行了浮点运算。许多的处理器/编译器中，浮点一般都是不可重入的（浮点运算大多使用协处理器或者软件模拟来实现）。

**1.9简述处理器中断处理的过程。**

处理器在中断处理的过程中，一般分为以下几个步骤：中断请求 -> 中断响应 -> 保护现场 -> 中断服务 -> 恢复现场 -> 中断返回。

**1.10保护现场、恢复现场的过程?**

（1）保护现场

①首先将R0~R7（快速中断）或R0~R12（其他中断）PUSH进堆栈中保存，并将堆栈指针保存在对应中断模式的R13（SP）中。

②把即将执行的下一条指令（PC-4）的地址保存到对应中断模式的R14（LR）中；把CPSR的值保存到对应中断模式的SPSR中，以实现对处理器当前状态、中断屏蔽及各标志位的保护。

③设置CPSR的相应位：设置CPSR[4:0]的5位以进入相应的工作模式；CPSR[5] = 0切换到ARM状态；设置CPSR[7] = 1禁止IRQ中断；如果进入复位模式或FIQ模式，还要设置CPSR[6] = 1以禁止FIQ中断。

④给程序计数器PC强制赋值，转入中断向量地址，执行相应的中断服务程序。

（2）恢复现场

①将原来保存在堆栈中的R0~R12或R0~R7 POP出栈，赋值给相应的寄存器。

②将LR的值赋值给PC，将SPSR的值赋值给CPSR中，恢复被中断的程序状态。

**1.11复位中断与其他中断有什么不同?**

（1）当中断产生后，复位中断立即中止当前指令的执行，其余情况都是当处理器完成当前指令后，再去响应中断。

（2）如果是复位中断，系统自动从0x00000000开始重新执行程序，无需中断返回。

### 1.12什么是中断向量？什么是中断嵌套?

（1）中断向量：中断服务子程序的入口地址。

（2）中断嵌套：中断系统正在执行一个中断服务程序时，有另一个优先级更高的中断源提出请求，这时会暂停当前正在执行的级别较低的中断源的服务程序，处理级别更高的中断源。处理完毕后再返回到被中断了的中断服务程序。

### 1.13外部中断请求（IRQ）和快速中断请求（FIQ）的异常向量地址分别为?

0x00000018、0x0000001C。

### 1.14中断的优缺点?

（1）优点：实现CPU和I/O设备的并行，提高CPU的利用率和系统的性能。

（2）缺点：中断处理过程需要保护现场、恢复现场，整个过程需要一定的时间和空间开销。如果中断的频率太高，会降低系统的性能。

### 1.15中断服务程序能不能有参数和返回值?

（1）在单片机裸机程序中，中断服务程序既不能有参数，也不能有返回值。

（2）但是在带操作系统的嵌入式系统中，中断服务程序可以有参数，也可以有返回值。



帖子还没人回复，快来抢沙发!