《openEuler内核编程》

课程讲稿

第5章 第2讲

寄存器

软件所制

第4章 第2讲 寄存器

**学时：1**学时

**教学目的：**学习CPU中的寄存器知识，了解了硬件知识之后就更容易理解后面中断相关的东西了……

**课程时间线：**

章节简介

计算机体系结构

x86下的寄存器

指令执行

5min

15min

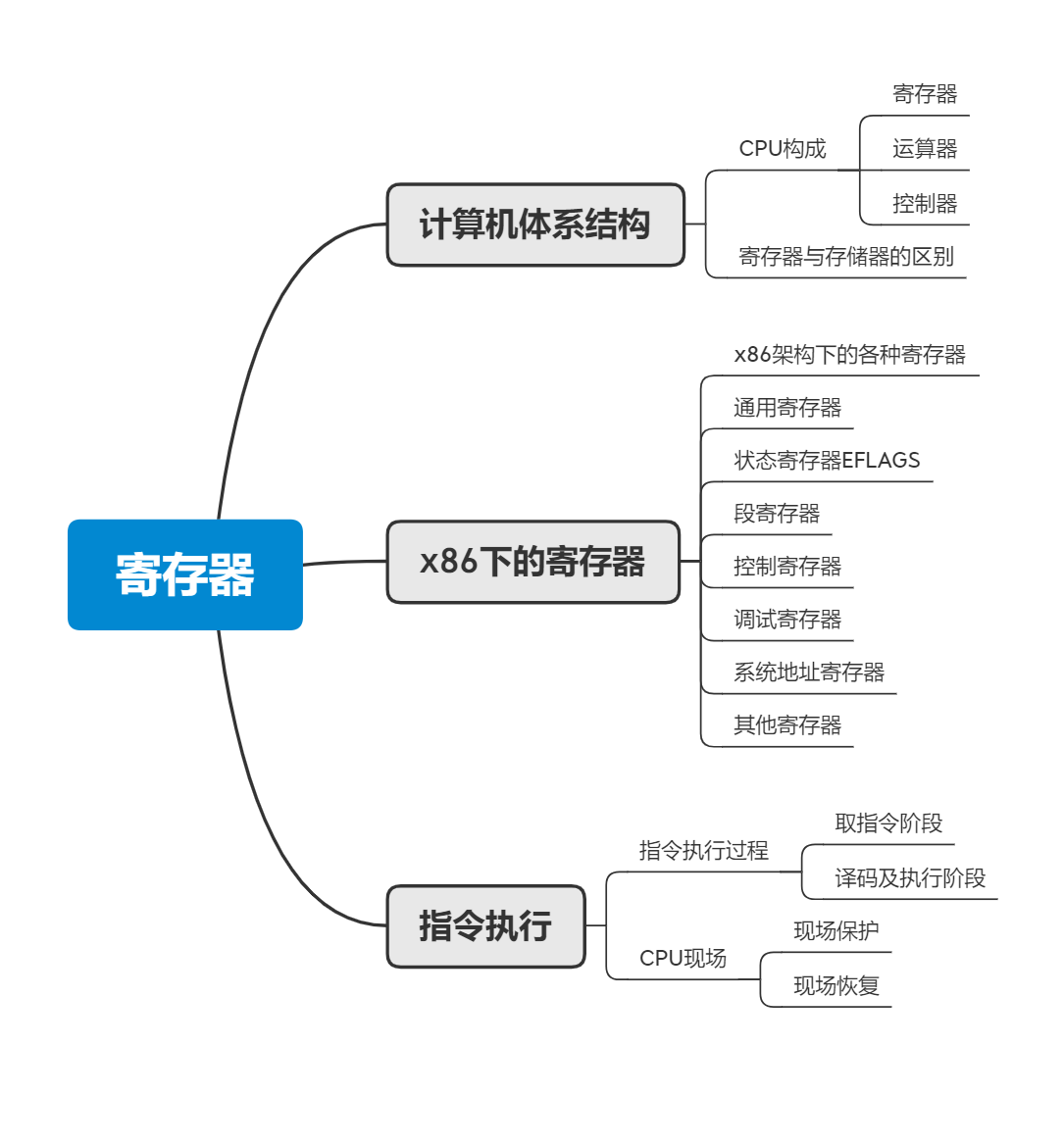
35min

45 min

**课外参考读物：**

《深入理解LINUX内核》

**知识框图：**

****

**PPT讲稿：**

1. 今天我们来学习CPU中寄存器的知识。
2. 中断机制是操作系统中与硬件结合非常紧密的部分，这就需要同学们掌握一些硬件特别是寄存器相关的知识。
3. 本节中我们将分为三块内容来介绍中断与异常基础知识，分别是计算机体系结构、x86架构下的寄存器和指令执行过程，接下来我们先来看看计算机体系结构
4. 学过计算机组成原理的大概都看得懂这张图，一般来说计算机都是由以上这及部分组成。而最基本的部分叫做主机。真正的计算以控制发生在中央处理器里面，内存储器用作存放所需进行处理的数据和所要运行的机器码。机器码存放在内存中，而执行发生在中央处理器中。
5. 中央处理器由运算器、控制器和寄存器组成，简单描述一下cpu的工作原理：控制器在机器码序列的指导下，对运算器、寄存器以及其它的部件发出各种命令，命令它们进行响应的操作，这些操作主要就是对cpu中的各个寄存器进行读写操作。
6. 寄存器其实就是cpu内部的存储部件，cpu内部用作存储的部件还有cache，不过cache只是用作补偿cpu与内存之间的速度差。并且寄存器会直接参加运算，而cache不直接参加运算。Cpu中的寄存器用于保存数据以及记录当前程序的一些状态标志，因此可以说所有寄存器的值就反映了当前程序的运行状态。
7. 同是用作存储数据的作用，内存与寄存器到底有什么区别呢？它们之间的区别就如同ppt上所讲。
8. 我们再来看看cpu中的运算器和控制器。控制器就是cpu内部的大boss，它可以对其他的所有部件发号施令，其内部也有很多寄存器来用于辅助发出命令：
   1. 指令寄存器用于保存CPU正在执行的指令的机器码，控制器内部的指令译码器会将当前的指令翻译成控制序列来对其他部件发出命令
   2. 程序计数器保存着下一条要执行指令的地址。
   3. 地址寄存器用于保存CPU所要访问的那一块内存的首地址。
   4. 指令译码器对IR内的指令进行分析，生成微操作序列。
9. 运算器用于对数据进行运算：
   1. 算术逻辑单元就是一系列基础运算实际进行的地方，它进程操作的对象就是运算器内部的其他寄存器（例如累加寄存器）；
   2. 累加寄存器，主要功能就是存储所要计算或者计算完成的数据，提供给算术逻辑单元；
   3. 从内存读取过来的数据会暂时存放在数据缓冲寄存器中，相当于内存和CPU之间做数据传送的一个中转站。
   4. 状态条件寄存器用于保存当前运算过程的一些状态，例如是否有进位或结果是否为零。
10. 学完一些寄存器的通用知识，我们再来看看x86架构下的寄存器
11. 我们拿x86架构来介绍各个主要寄存器的作用，在x86下的寄存器的作用主要分为以下几种：
    1. 通用寄存器：可以灵活使用，例如用于计算、访存或记录标志。
    2. 标志寄存器：用于记录当前进程的状态。
    3. 段寄存器：在x86下的内存访问方式是段页式，段寄存器用于保存段描述符。
    4. 控制寄存器：用于控制和确定处理器的操作模式以及当前执行任务的特性。
    5. 调试寄存器：在调试的时候使用，例如设置断点的位置等。
    6. 系统地址寄存器：用于存放非常重要的数据结构的地址，例如全局描述符表的首地址，这样可以快速访问这些表。
    7. 其他：例如程序计数器
12. 下面我们对这些寄存器进行更加具体的讲解。通用寄存器的“通用”体现在它们的功能多样性，在32位下的通用寄存器可以只使用16位也可以使用32位，在32位下时，一部分通用寄存器不仅可传送数据、暂存数据保存算术逻辑运算结果，而且也可作为指针寄存器。
13. 累加器，既可以用作存放基础运算的源数据，也可以用来存放计算后的返回值，由于x86的特殊设计，如果在计算中直接使用累加器会需要更少的时间。有一种访存方式叫做基质寻址，这就需要一个寄存器来存放这个基址，而这个特殊的寄存器就叫做基址寄存器。另外还有一个计数寄存器，有时候要访问内存的多个字节或者多次循环，就要用这个寄存器来存放循环计数。
14. 再来看看剩下的两个通用寄存器。数据寄存器在功能上与累加器有重叠的地方，它是存放参与运算的默认操作数的寄存器，也可以存放io端口地址。最后的源/目标索引寄存器就更加神通广大了，它们两个通常是配合起来使用。用来存放地址，可以实现多种寻址方式。也可以用来存放数据，实现通用操作。
15. 接下来就是状态寄存器了，它主要用于提供程序的状态及进行相应的控制，这对程序的正确执行至关重要。可以看看它主要存放了三类信息，一类是当前指令执行结果的各种状态信息，一类是控制信息（例如是否允许中断等），另外一类是DF标志，用于决定指令计数器PC的增长方向。
16. 下面这两张ppt就是介绍标志寄存器中每个标志位的作用。
17. 这里面要讲一下IF标志位，IF标志用于控制当前CPU是否响应外部中断。当IF为1时则屏蔽了全部外部中断，这时我们所讲的可屏蔽中断都不会被响应，而异常和非屏蔽中断不受这个影响。
18. 在各个架构的CPU中都有一个指向下一条指令地址的寄存器，在ARM下叫做程序计数器PC，在x86下叫做指令指针寄存器EIP，它们都有一个类似的功能，就是在读出当前的指令之后会自动对其内部存储的值进行增减，以指向下一条指令的地址。但是这个值到底是加还是减则要受到当前程序内存布局的限制，有的下一条指令是在高位，有的是在低位。这时就需要有一个标志来控制PC或者EIP的增长方向，这个标志位就是状态寄存器里面的DF标志。
19. 接下来再看看与虚拟地址变换息息相关的段寄存器。在x86架构下的Linux的数据存放是段页式存放的。在内核保存了一些段表，例如全局描述符表，表当中的表项叫做段描述符，用于选择当前的段。另外还有一个概念叫做段选择子，用于在全局描述符表当中选择对应的段描述符，而段寄存器中存放的就是段选择子。Linux中存放了三种段，代码段、数据段和栈段。访问代码时就使用CS寄存器里的段选择子来选择对应的代码段、访问数据段时就使用DS/ES/FS/GS寄存器里的段选择子来选择对应的数据段。
20. 此外还有控制寄存器，用于控制和确定处理器的操作模式以及当前执行任务的特性，例如当前是否处于x86下的保护模式、是否开启了页机制等。在整个操作系统的运行过程中需要有很多的标志位来确定运行所需遵循的规则，这一功能和状态寄存器有点类似，只不过状态寄存器是保存当前进程或指令的状态，而控制寄存器保存的是整个操作系统的状态。

而调试寄存器是在调试过程中使用的。

1. 为了快速访问到内核中重要的表结构，例如全局描述符表、局部描述符表和中断描述符表等（它们都存放于内存中），x86的cpu内特意设置了一些寄存器来存放这些百表结构的首地址。这样在访问这些表结构时，直接根据这些寄存器存放的地址就可以访问到这些表结构了。
2. 在这里有一个中断描述符表寄存器IDTR，存放的是中断描述符表的首地址，中断描述符表我们在上一章已经说过，存放的是中断服务程序的地址。这样在中断到来时，直接根据IDTR存放的地址找到IDT就可以找到对应的中断服务程序了。
3. 最后还有指令指针寄存器EIP（在ARM下叫做程序计数器PC），作用就是保存下一条指令的偏移地址，在段的基础上加上这个偏移地址就可以读取指令了。TSC则是cpu内部用于计时的寄存器。另外还有一种浮点寄存器，用于进行较复杂的浮点运算。
4. 最后我们再看看指令在计算机中是怎么执行的
5. 下面我们来看看指令的执行过程。首先进入取指令阶段，根据pc或者eip内部存放的指令地址从内存读出指令的机器码，放入指令寄存器IR。这个过程中pc或eip在DF的方向控制下自动跳到下一条指令的地址。
6. IR内部存放的是当前要执行的指令的机器码，通过控制器内部的指令译码电路翻译之后形成微操作序列，控制各部件进行相应的操作。指令执行完毕之后就进入下一个指令周期了，再次循环上述的操作。
7. 前面已经说过，cpu内部各寄存器的值共同组成的当前的进程运行的状态，而当前进程运行的状态就叫做cpu现场，其中主要就包括这几个寄存器：状态寄存器、代码段寄存器、实例指针寄存器，此外还有数据段寄存器和栈寄存器等。前面讲的系统地址寄存器，例如GDTR、IDTR等就不用保存了，因为它们在整个操作系统运行过程中都不会改变，对于每个进程都是一样的，所以不用保存。
8. 中断响应是一个比较长的过程，涉及到很多个步骤，将在后面的章节讲解。CPU在响应中断后，首先要进行现场保护。现场保护主要就是保存几个主要的寄存器的值，保存寄存器的值都是对他们进行压栈，也就是保存到内存中，同时栈指针要进行响应的移位，这样到时候中断返回的时候再按照相反的顺序将栈中保存的值返回到相应的寄存器就行了。
9. 中断返回是为了让被中断的程序继续正确的执行，因此要恢复被中断前的cpu现场，而恢复现场就是将原先的寄存器的值返回到相应寄存器。