vuoin

0 2010-05-09 23:41:38

更正一下：
四核处理器的完整启动过程如下：
1.处理器刚加电或复位，主核0立即启动（三个从核尚未启动），此时主核0的EIP指向ROM，主核0从ROM中读取BIOS指令，通过BIOS指令把磁盘中的操作系统引导程序载入ROM出口处的RAM。
2.主核0执行玩ROM中的BIOS指令进入RAM，执行RAM中的操作系统引导程序，跳至相应的地址，然后执行那个地址上的指令来改变这条跳转指令的跳转地址供从核1使用。
3.从核1延时启动，虽然此时从核1的EIP指向ROM，但ROM对从核不可读，所以执行空指令，除了EIP不停加1外什么事都不干。
4.从核1执行玩指向ROM的空指令进入RAM，执行RAM中的操作系统引导程序，执行已经被主核0改变了的跳转指令，跳入一个与主核0不同的地址，然后执行那个地址上的指令再次改变这条跳转指令的跳转地址供从核2使用。
5.从核2延时启动，虽然此时从核2的EIP指向ROM，但ROM对从核不可读，所以执行空指令，除了EIP不停加1外什么事都不干。
6.从核2执行玩指向ROM的空指令进入RAM，执行RAM中的操作系统引导程序，执行已经被从核1改变了的跳转指令，跳入一个与从核1不同的地址，然后执行那个地址上的指令再次改变这条跳转指令的跳转地址供从核3使用。
7.从核3延时启动，虽然此时从核3的EIP指向ROM，但ROM对从核不可读，所以执行空指令，除了EIP不停加1外什么事都不干。
8.从核3执行玩指向ROM的空指令进入RAM，执行RAM中的操作系统引导程序，执行已经被从核2改变了的跳转指令，跳入一个与从核2不同的地址。
至此，四个核已经分别指向不同的地址，可以同时执行不同的指令了。

确实每个核都有一套完整的寄存器。

其实多核的地址跳转分离仅仅是启动的第一步，以后还有很多事要做：
1.每个核分别执行相应地址上的操作系统指令，把各自的CR0寄存器的第0位即PE标志位设为1，从而进入保护模式。在保护模式下，指令的地址由段寄存器CS、段描述符表寄存器GDTR和LDTR、段描述符表GDT和LDT以及指令指针EIP共同决定。段描述符表寄存器GDTR和LDTR分别指定段描述符表GDT和LDT的基址，段寄存器CS中的选择子从段描述符表GDTR或LDTR中选择一项（当CS的第2位即表指示位为0时，选择全局描述符表GDT中的一项，当CS的第2位即表指示位为1时，选择局部描述符表LDT中的一项）。GDT和LDT都在RAM中，分别由全局描述符表寄存器GDTR和局部描述符表寄存器LDTR指令基址。段描述符表GDT和LDT中的被选择项指定代码段的基址，EIP指定代码段的变址。
2.每个寄存器继续分别执行相应地址上的操作系统指令，构建好各自的中断描述符表IDT，用来在中断时进入操作系统的中断处理程序。中断描述符表IDT在RAM中，由中断描述符表寄存器IDTR指定基址，由中断号指定表选项。
3.每个核分别创建操作系统代码段，即操作系统线程。

至此，操作系统的初始化完成，可以开始调度线程了。

调用操作系统的创建线程API，即可创建用户线程。

每个线程的代码段都由段描述符表指定了基址和界限，位于RAM中的不同地址，绝不允许越界，所以，只要保证多核总是执行不同的线程，就能保证多核总能执行位于不同地址的指令。
那么，怎么保证多核总是执行不同的线程呢？靠的是总线锁LOCK#信号。当某个核访问一个线程时，会同时向总线发出LOCK#信号锁住总线，阻塞其它核对这个线程的访问，直到这个核退出这个线程后解锁总线，才允许其它核访问。当多个核同时访问一个线程时，都会向总线发出LOCK#信号锁住总线，此时优先级较高的核的发出的LOCK#信号的优先级比优先级较低的核的LOCK#信号的优先级高，所以只有优先级较高的核能抢到总线锁，访问这个线程并阻塞其它核。比如核1和核2同时访问某一线程，最后执行这一线程的只有优先级较高的核1，核2被阻塞后转而去访问其它线程。

一个核不会总是执行一个线程，而是会定期切换线程。怎么切换线程呢？这是通过定时器中断来实现的。每个核都有一个定时器，会自动倒计时。在切换入一个用户线程之前，操作系统中负责切换线程的线程会把定时器设置好倒计时，当用户线程执行完定时器的倒计时后，定时器发出一个定时器中断信号，核收到定时器中断信号后，会转到中断描述符表所指定的定时器中断号的中断处理程序——即操作系统中负责切换线程的线程，然后操作系统中负责切换线程的线程又把定时器设置好倒计时，然后切换入另一个用户线程。每次用户线程之间的切换，都要经过操作系统中负责切换线程的线程。当然，与用户线程比起来，操作系统中负责切换线程的线程所需的时间是极少的。

当某个线程的执行尚未结束（如果是循环线程，则永远也不会结束），倒计时却已到，线程的寄存器状态会不会丢失呢？其实不会。因为在线程切换时，线程的寄存器状态会被保存到任务状态段TSS，任务状态段TSS也在全局描述符表GDT中，但是不是由代码段寄存器CS来选择，而是由任务寄存器TR来选择。当再次切换到这个线程中时，TSS中的任务状态又会被重新载入相应的寄存器。

任务状态段TSS保存的任务状态如下图所示:

31 ~ 15	15 ~ 0		
I/O Map Base Address	Reserved	T	100
Reserved	LDT Segment Selector		96
Reserved	GS		92
Reserved	FS		88
Reserved	DS		84
Reserved	SS		80
Reserved	CS		76
Reserved	ES		72
EDI			68
ESI			64
EBP			60
ESP			56
EBX			52
EDX			48
ECX			44
EAX			40
EFLAGS			36
EIP			32
CR3 (PDBR)			28
Reserved	SS2		24
ESP2			20
Reserved	SS1		16
ESP1			12
Reserved	SS0		8
ESP0			4
Reserved	Previous Task Link		0

在用户状态下，只要有多个用户线程，多个核就可以同时执行不同的用户线程，从而可以充分利用多核处理器的处理能力。