**Loader引导功能**

Loader引导加载程序负责检测硬件信息，处理器模式切换，向内核传递数据三部分工作，这些工作作为内核的初始化提供信息及功能支持，以便内核在完成初始化工作后能够正常运行。

**1.检测硬件信息**

Loader引导加载程序需要检测的硬件信息很多，主要通过BIOS中断服务程序来获取和检测硬件信息。由于BIOS在上电自检出的大部分信息只能在实模式下获取，而且内核运行于保护模式下，因此必须在进入内核程序前将这些信息检测出来，作为参数传递给内核程序。

**2.处理器模式**

从起初BIOS运行的实模式（real mode），到32位操作系统使用的保护模式（protect mode），再到64位操作系统使用的IA-32e模式（long mode，长模式），Loader引导加载程序必须历经这三个模式，才能使处理器运行于64位的IA-32e模式。在各个模式的切换过程中，Loader引导加载程序必须手动创建各运行模式的临时数据，并按照标准流程执行模式间的跳转。

**3.向内核传递数据**

Loader引导加载程序可向内核程序传递两类数据，一类控制信息，另一类是硬件数据信息。这些数据一方面控制内核程序的执行流程，另一方面为内核程序的初始化提供数据信息支持。

控制信息：一般用于控制内核执行流程或限制内核的某些功能。这些数据（参数）是与内核程序早已商定的协议，属于纯软件控制逻辑，如启动模式（字符界面或图形界面），启动方式（网络或本地），终端重定向（串口或显示器等）信息。

硬件数据信息通常指Loader引导加载程序检测出的硬件数据信息。Loader引导加载程序将这些数据信息多半都保存在固定的地址中，并将数据起始内存地址和数据长度作为参数传递给内核，以供内核程序在初始化时分析，配置和使用。

**内核地址空间起于物理地址0x100000处原因？**

因为1MB以下的物理地址并不全是可用内存地址空间，这段物理地址被划分成若干个子空间段，它们可以是内存空间，非内存空间以及地址空洞。随着内核体积的不断增长，未来的内核程序可能会超过1MB，因此让内核程序跳过这些纷繁复杂的内存空间，从平坦的1MB地址空间开始。

**内存地址0x7E00的作用？**

内存地址0x7E00是内核程序的临时转存空间，由于内核程序的读取操作是通过BIOS中断服务程序INT 13h实现的，BIOS在实模式下只支持上限为1MB的内存地址空间寻址，所以必须将内核程序读入到临时转存空间，然后在通过特殊方式搬运到1MB以上的内存空间中。当内核程序被转存的到最终的内存空间后，这个临时转存空间就可另作它用。

**开启A20功能的原因**

最初的处理器只有20根地址线，这使得处理器只能寻址1MB以内的物理地址空间，如果超过1MB范围的寻址操作，也只有低20位是有效地址（即对1MB取余）。随着处理器寻址能力的增强，20根地址线已经无法满足今后的开发需求，为了保证硬件平台向下兼容，便出现了一个开启或禁止1MB以上地址空间的开关。当时的8042键盘控制器上恰好有空闲的端口引脚（端口P2，引脚P21），从而使用此引脚作为功能控制开关。如果A20引脚为低电平，那么只有低20位有效。

**开启A20功能的方法**

1.操作键盘控制器，但由于键盘控制器是低速设备，以至于功能开启速度较慢。

2.A20快速门，它使用I/O端口0x92来处理A20信号线。对于不含键盘控制器的操作系统，就只能使用0x92端口进行控制。

3.使用BIOS中断服务程序INT 15h的主功能号AX=2401可开启A20地址线，AX=2400可禁止使用A20地址线，功能号AX=2403可查询A20地址线的当前状态。

4.通过读0xee端口来开启A20信号线，而写该端口会禁止A20信号线。

**为何使用保护模式，而不使用实模式？**

在实模式下，程序可以操作任何地址空间，而且无法限制程序的执行权限，尽管这种模式给设置硬件功能带来许多方便，但却给程序执行的安全性和稳定性带来了灾难性的后果，一旦程序执行错误，很可能导致整个系统的崩溃。同时实模式的寻址能力有限，故而才出现保护模式。

**保护模式的特点**

在保护模式里，处理器按程序执行的级别划分为0，1，2，3四个等级（由高到低排序）。最高等级0由系统内核使用，最低等级3由应用程序使用。Linux内核目前只使用0级和3级。（等级1和等级2介于内核程序与应用程序之间，通常作为系统服务程序来使用）。保护模式还引入了分页功能。分页功能将庞大的地址空间划分为固定大小的内存页面，此举不仅便于管理，而且还缩减了应用程序的空间浪费现象。

**使用IA-32e的原因?**

在保护模式的段级保护措施中，从段结构组织的复杂性，到段间权限检测的繁琐性，再到执行的效率上，都显得臃肿，而且还降低了程序执行效率和编程的灵活性。当页管理单元出现后，段机制显得更加多余。随着硬件速度不断提升和对大容量内存的不断渴望，IA-32模式便应运而生。

**IA-32模式的优点**

简化段级保护措施的复杂性；

升级内存寻址能力；

扩展页管理单元的组织结构和页面大小；

推出新的系统调用方式；

高级可编程中断控制器。

**从实模式进入保护模式**

为了进入保护模式，处理器必须再模式切换前，在内存中创建一段可在保护模式下执行的代码以及必要的系统数据结构，只有这样才能保证模式切换顺利完成。相关系统数据结构包括IDT/GDT/LDT描述符表各一个（LDT表可选），任务状态段TSS结构，至少一个页目录和页表（如果开启分页机制）和至少一个异常/中断处理。

处理器切换到保护模式前，还必须初始化GDTR寄存器，IDTR寄存器（亦可推迟到进入保护模式后，使能中断前），控制寄存器CR1-4，MTTRS内存范围类型寄存器。

**系统数据结构：**

系统在进入保护模式前，必须创建一个拥有代码段描述符和数据段描述符的GDT（Global Descriptor Table）（第一项必须是NULL描述符），并且一定要使用LGDT汇编指令将其加载到GDTR寄存器。保护模式的栈寄存器SS，使用可读写的数据段即可，无需创建专用描述符。对于多段式操作系统，可采用LDT（Local Descriptor Table）（必须保存在GDT表的描述符中）来管理应用程序，多个应用程序可独享或共享一个局部描述符LDT。如果希望开启分页机制，则必须准备一个页目录项和页表项。

**中断和异常：**

在保护模式下，中断/异常处理程序皆由IDT（Interrupt Descriptor Table）来管理。IDT由若干个门描述符组成，如果采用中断门或者陷阱门描述符，它们可以直接指向异常处理程序；如果采用任务门描述符，则必须为处理程序准备TSS段描述符，额外的代码和数据以及任务段描述符等结构。如果处理器允许接受外部中断请求，那么IDT还必须为每个中断处理程序建立门描述符。在使用IDT表前，必须使用LIDT汇编指令将器加载到IDTR寄存器，典型的加载时机是在处理器切换到保护模式前。

**分页机制：**

CR0控制寄存器的PG标志位用于控制分页机制的开启与关闭。在开启分页机制（置位PG标志位）前，必须在内存中创建一个页目录和页表（此时页目录和页表不可使用同一物理页），并将页目录的物理地址加载到CR3控制寄存器（或称PDBR寄存器）。当上述工作准备就绪后，可同时置位控制寄存器CR0的PE标志位和PG标志位，来开启分页机制。（分页机制往往与模式切换同时进行，不能在进入保护模式前开启分页机制。

**多任务机制：**

如果希望使用多任务机制或允许改变特权级，则必须在首次执行任务切换前，创建至少一个任务状态段TSS结构和附加的TSS段描述符。（当特权级切换至0，1，2时，栈段寄存器与栈指针寄存器皆从TSS段结构中取得）在使用TSS段结构前，必须使用LTR汇编指令将其加载至TR寄存器，这个过程只能在进入保护模式后执行。此表也必须保存在全局描述符表GDT中，而且任务切换不会影响其他段描述符，LDT表，TSS段结构以及TSS段描述符的自由创建。只有处理器才能在任务切换时置位TSS段描述符的忙状态位，否则忙状态位始终保持复位状态。如果既不想开启多任务机制，也不允许改变特权级，则无需加载TR任务寄存器，也无需创建TSS段结构。

**模式切换的步骤：**

1.执行CLI汇编指令禁止可屏蔽硬件中断，对于不可屏蔽中断NMI只能借助外部电路才能禁止（模式切换必须保证在切换过程中不能产生异常和中断）

2.执行LGDT汇编指令将GDT的基地址和长度加载到GDTR寄存器

3.执行mov CR0汇编指令置位CR0控制寄存器的PE标志位。（可同时置位CR0控制寄存器的PG标志位）

4.一旦Mov CR0汇编指令执行结束，紧随其后必须执行一条远跳转（far JMP）或远调用（far CALL）指令，以切换到保护模式的代码去执行（这是典型的保护模式切换方法）

5.通过执行JMP或CALL指令，可改变处理器执行的流水线，进而使处理器加载执行保护模式的代码段。

6.如果开启分页机制，那么MOV CR0指令和JMP/CALL指令必须位于同一线性地址映射的页面内。（因为保护模式和分页机制使能后的物理地址，与执行JMP/CALL指令前的线性地址相同）至于JMP和CALL指令的目标地址则无需进行同一线性地址映射。

7.如需使用LDT，则必须借助LLDT汇编指令将GDT内的LDT段选择子加载到LDTR寄存器中

8.执行LTR汇编指令将一个TSS段描述符的段选择子加载到TR任务寄存器。处理器对TSS段结构无特殊要求，凡是可写的内存空间皆可。

9.进入保护模式之后，数据段寄存器仍旧保留着实模式的段数据，必须重新加载数据段选择子或使用JMP/CALL指令执行新任务，便可将其更新为保护模式。对于不使用数据段寄存器（DS和SS寄存器除外），可将NULL段选择子加载到其中。

10.执行LIDT指令，将保护模式下的IDT表的基地址和长度加载到IDTR寄存器。

11.执行STI指令使能可屏蔽硬件中断，并执行必要的硬件操作使能NMI不可屏蔽中断

**从保护模式到IA-32e模式**

在进入IA-32e模式前，处理器依然要为IA-32e模式准备执行代码，必要的系统数据结构以及配置相关控制寄存器。于此同时，还要求处理器只能在开启分页机制的保护模式下切换到IA-32e模式。

**系统数据结构：**

当IA-32e模式激活后，系统各描述符表寄存器（GDTR，LDTR，IDTR，TR）依然沿用保护模式的描述符表。由于保护模式的描述符表基地址是32位，这使得它们均位于低4GB线性地址空间内。在开启IA-32e模式后，系统各描述符表寄存器必须重新加载（借助LGDT，LLDT，LIDT和LTR指令）为IA-32e模式的64位描述符表。

**中断和异常：**

当软件激活IA-32e模式后，中断描述符表寄存器IDTR仍然使用保护模式的中断描述符表，那么在将IDTR寄存器更新为64位中断描述符表IDT前不能触发中断和异常，否则处理器会把32位兼容模式的中断门解释为64位中断门，从而导致不可预料的结果。使用cli指令能够禁止可屏蔽硬件中断，而NMI不可屏蔽中断，必须借助外部硬件电路才可禁止。

**开启IA-32e模式及其初始化步骤**

IA32\_EFER寄存器（位于MSR寄存器组内）的LME标志位用于控制IA-32e模式的开启与关闭，该寄存器会伴随着处理器的重启（或重置）而清零。IA-32e模式的也管理机制将物理地址扩展为四层页表结构。在IA-32e模式激活前（CR0.PG=1，处理器运行在32位兼容模式），CR3控制寄存器仅有低32位可写入数据，从而限制页表只能寻址4GB的物理内存空间，也就是说在初始化IA-32e模式时，分页机制只能使用前4GB的物理地址，一旦激活IA-32e模式，软件便可重新定位页表到物理内存空间的任何地方。

**初始化步骤:**

1.在保护模式下，使用MOV CR0汇编指令复位CR0控制寄存器的PG标志位，以关闭分页机制。（此后的指令必须位于同一线性映射的页面内）

2.置位CR4控制寄存器的PAE标志位，开启物理地址扩展功能（PAE）。在IA-32e模式的初始化过程中，如果PAE功能开启失败，将会产生通用保护性异常（#GP）

3.将页目录（顶层页表PML4）的物理基地址加载到CR3控制寄存器中。

4.置位IA32\_EFER寄存器的LME标志位，开启IA-32e模式。

5.置位CR0控制寄存器的PG标志位开启分页机制，此时处理器会自动置位IA32\_EFER寄存器的LMA标志位。当执行MOV CR0指令开启分页机制时，其后续指令必须位于同一线性地址映射的页面内（直至处理器进入IA-32e模式后，才可以使用非同一线性地址映射页面）

**通用保护性异常（#GP）**

如果试图改变IA32\_EFER.LME，CR0.PG和CR4.PAE等影响IA-32e模式开启的标志位，处理器会进行64位模式一致性检测，以确保处理器不会进入未定义模式或不可预测的运行状态。如果一致性检测失败，处理器将会产生通用保护性异常。

1.当开启分页机制后，再试图使能或禁止IA-32e模式

2.当开启IA-32e模式后，试图再开启物理地址扩展（PAE）功能前使能分页机制

3.再激活IA-32e模式后，试图禁止物理地址扩展（PAE）

4.当CS段寄存器的L位被置位时，再试图激活IA-32e模式

5.如果TR寄存器加载的是16位TSS段结构

**CPUID指令概述：**

1.EFLAGS标志寄存器的ID标志位（第21位）表明处理器是否支持CPUID指令。如果程序可以操作（置位或复位）此标志位，则说明处理器支持CPUID指令，CPUID指令在64位模式和32位模式的执行效果相同

2.CPUID指令会根据EAX寄存器传入的基础功能号（有时还需要向ECX寄存器传入扩展功能号），查询处理器的鉴定信息和机能信息，其返回结果保存EAX，EBX，ECX和EDX寄存器中

**RDMSR和WRMSR指令的概况描述：**

1.借助RDMSR和WRMSR指令可以访问64位的MSR寄存器。在访问MSR寄存器前，必须向ECX寄存器（64位模式下，RCX寄存器的高32位被忽略）传入寄存器地址。而目标MSR寄存器则是由EDX：EAX组成的64位寄存器代表，其中EDX寄存器保存MSR寄存器的高32位，EAX寄存器保存低32位。（在64位模式下，RAX和 RDX寄存器的高32位均为0）

2.RDMSR与WRMSR指令必须在0特权级或实模式下执行，在使用者两条指令之前，应使用CPUID指令（CPUID.01h:EDX[5]=1）来检测处理器是否支持MSR寄存器组。

**80x86下内存地址：**

**1.逻辑地址（logical address）**

包含在机器语言指令中用来指定一个操作数或一条指令的地址。每个逻辑地址都由一个段（segment）和偏移量（offset）组成，偏移量指明了从段开始的地方到实际地址的距离。

**2.线性地址（linear address）或虚拟地址（virtual address）**

虚拟地址是一个32位无符号整数（如果在64位机器下应当是64位的无符号整数），可以用来表示高达4GB的地址。虚拟地址通常用16进制数字表示，值的范围从0x00000000到0xffffffff

**3.物理地址（physical address）**

用于内存芯片级内存单元寻址，它们与微处理器的地址引脚发送到内存总线上的电信号相对应。物理地址由32位或36位无符号整数表示。

**4.地址转换**

内存控制单元（MMU）通过一种分段单元的硬件电路把一个逻辑地址转换为虚拟地址；接着，第二个分页单元的硬件电路把线性地址转换为一个物理地址。