硬件编程接口与系统软件实现

朱春森　编著

北京大学出版社
PEKING UNIVERSITY PRESS

硬件编程接口与系统软件实现

朱春森　编著

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

硬件功能是为软件实现服务的，软件实现是基于硬件支持的。本书以软硬件结合部为切入点，完整地介绍硬件编程接口与系统软件实现，包括系统引导接口、设备配置接口、设备控制器编程接口和处理器编程接口，并在处理器硬件支持基础上说明系统软件低层实现，如存储器管理、任务管理、线程切换和处理器调度。随着设备的标准化，调用系统 BIOS 驱动设备的方法已日渐淘汰，操作系统直接驱动设备和管理电源。所以，除了介绍 IA-32 处理器、IA-32e 处理器(扩展 64 位)和安腾 64 位处理器外，本书以大量篇幅介绍设备控制器编程接口，包括键盘控制器、图形控制器、IDE 设备控制器、音频/调制解调器控制器、USB 控制器和 LAN 控制器，并附带介绍相关设备总线标准。本书还介绍了外存介质分区结构、存储介质格式规范和流行的文件系统，并给出外存管理、文件管理和文件接口示例。另外，本书结合处理器指令功能和寻址机制，介绍一些操作系统相关的高级语言程序实现内容，如变量空间与程序结构、函数申明与函数出口、函数调用与函数入口等，同时还涉及一些代码生成问题，如数据编码与存储格式、算术表达式和逻辑表达式编译，并介绍微软 32 位可执行文件格式，说明程序加载步骤。

本书适合软件专业学生和相关从业人员作为教材或参考书，也可以作为电脑发烧友导游图，协助您漫游计算机系统工作过程，从系统引导到程序运行、从高级语句到可执行代码、从键盘到处理器。

图书在版编目(CIP)数据

硬件编程接口与系统软件实现/朱春森编著. —北京：北京大学出版社，2005.8
ISBN 7-301-08932-5

I. 硬… II. 朱… III. ①硬件—接口—程序设计②软件设计 IV. ①TP303②TP311.5

中国版本图书馆 CIP 数据核字（2005）第 031838 号

书　　名：硬件编程接口与系统软件实现
著作责任者：朱春森　编著
责 任 编 辑：黄庆生　汉明
标 准 书 号：ISBN 7-301-08932-5/TP・0782
出　版　者：北京大学出版社
地　　址：北京市海淀区成府路 205 号　100871
电　　话：邮购部 62752015　发行部 62750672　编辑部 62765013
网　　址：http://cbs.pku.edu.cn
电 子 信 箱：xxjs@pup.pku.edu.cn
印　刷　者：河北滦县鑫华书刊印刷厂
发　行　者：北京大学出版社
经　销　者：新华书店
787 毫米×1092 毫米　16 开本　23.75 印张　592 千字
2005 年 8 月第 1 版　2005 年 8 月第 1 次印刷
定　　价：46.00 元

前　言

在学习计算机相关教程时总有隔靴搔痒之感，脱离硬件支持谈操作系统原理、脱离软件实现谈硬件结构，除增加词汇量外，系统软件究竟如何实现还是摸不到门。若不了解硬件为软件提供的支持，不了解软件对硬件支持的利用，只能永远躺在别人的界面上。这正是笔者编写本书的原动力，本书的编写目的不是为了解惑“那是什么”，而是为了动手“怎么实现”。因此，本书尽可能详细地说明处理器为操作系统提供的硬件支持，尽可能详尽地罗列设备控制器编程接口，并通过大量图表和代码示例演示硬件机制的应用。

本书涉及如下几方面的系统软件实现问题：

（1）在计算机系统加电复位后，处理器与系统引导代码之间如何实现交接？系统引导代码与操作系统加载程序之间如何实现握手？

（2）在操作系统接管系统后，操作系统需建立哪些系统管理数据？操作系统如何识别设备和配置设备？如何控制设备？如何管理系统存储器？

（3）在启动应用程序时，如何创建和管理任务空间？如何创建、初始化和调度线程？如何调度处理器？如何加载可执行文件？如何管理文件映像空间？

（4）在访问外存设备时，操作系统如何识别外存介质的文件系统？如何支持某个文件系统？如何管理外存块和文件目录？如何传输数据？

（5）在生成可执行文件时，如何编码数据和机器指令？如何实现高级语言语法单位？如何组织程序空间？如何链接系统库函数？可执行文件格式如何？

全书包含了硬件编程接口、系统软件实现和高级语言程序实现三个方面内容。由于硬件与软件、操作系统与应用程序存在着依赖关系，同时为了照顾读者的阅读理解过程，三个内容并没有完全分割。

第 1 章从系统加电复位开始，分别介绍了系统引导过程、程序加载步骤和程序代码等内容。通过阅读本章内容，读者不仅浏览计算机系统运行过程，结合附录 A“IA-32 指令一览表”，就可以尝试阅读和编制汇编程序了。

第 2 章介绍系统结构、系统资源、系统配置端口和设备 PCI 配置空间，并着重介绍 PCI 局部总线标准，该标准定义了设备配置接口。当代主流 PC 系统的标准设备支持 PCI 协议，操作系统枚举和配置标准设备正是基于 PCI 协议。

第 3 章介绍设备控制器和局部总线主控制器编程接口，包括键盘控制器、图形控制器、IDE 控制器、AC'97 控制器(音频&调制解调器)、USB 控制器和 LAN 控制器(网卡)等。硬件编程接口基于硬件标准层，由相关总线协议、设备工业标准和控制器接口定义，独立于操作系统。硬件编程接口定义设备寄存器集、指令集和控制字，包括设备控制、状态、地址和数据端口等寄存器结构和寻址方法，以及设备通信数据帧或数据包数据结构。

第 4 章介绍 IA-32 处理器和 IA-32e 处理器编程接口。处理器同样是系统总线上一个设备，编程接口由寄存器集和指令集组成，但寄存器寻址无须通过系统总线。

第 5 章介绍 IA-32 结构中断机制，包括了中断类型、中断过程、中断处理代码入口、中断向量与处理代码入口关系，并着重介绍局部高级可编程中断控制器(APIC)编程接口，处理器调度正是基于局部 APIC 支持。中断是操作系统干预程序运行的手段，比如例外处理和线程调度，也是程序和设备请求系统服务的方法。另外，第 10 章介绍了 IA-32

处理器辅助功能。

第 6 章介绍处理器 IA-32 指令功能，在此基础上说明一些高级语言程序语法单位的实现，如 C++语言中变量、类和函数等语法单位实现问题，程序空间结构，算术表达式和逻辑表达式编译，并介绍微软 32 位可执行文件格式，解析程序加载方法。

第 7 章介绍 IA-32 结构为存储器管理提供的硬件支持，包括分段和分页映射机制、段入口和页入口结构、段保护和页保护机制。在硬件支持的基础上，本章着重说明存储器管理问题，通过图形和代码示例展示存储器管理策略，如线性地址空间管理、存储器物理空间管理、页帧淘汰策略和脏页备份信息管理等，并给出页帧分配程序、页帧回收程序和缺页故障处理程序代码。

第 8 章介绍 IA-32 结构任务切换机制，并说明通过该机制初始化线程状态方法(平面模式手工初始化效率更高)，通过局部 APIC 定时中断切换线程方法。本章通过图形和代码示例展示任务管理结构、线程切换与处理器调度策略、定时器、系统空闲时的电源管理等。另外，本章还介绍多任务多线程系统下应用程序代码结构问题。

第 9 章介绍存储介质分区结构和格式规范，介绍 FAT 和 NTFS 文件系统，并给出外存管理和文件管理示例，定义文件入口和外存块入口结构。本章还给出文件接口示例，通过文件对象管理任务空间，并映射文件线性空间与外存空间，详细说明应用程序与文件接口关系，文件接口与文件系统及设备接口关系，介绍数据异步传输策略和实现。

第 11 章介绍安腾处理器(Itanium)编程接口，包括寄存器集、指令集、中断机制和存储器管理，并简要介绍安腾结构软件协议和处理器抽象层接口。安腾结构完全摆脱 Intel 8086 结构影响，抛弃分段模式，地址空间采用平面管理模式，指令集独立于 IA-32 指令集。安腾结构采用寄存器栈技术，支持软件流水线。安腾结构中断处理机制同样不同于 IA-32 结构，中断处理代码直接放在中断向量表。

附录 A“IA-32 指令一览表”列出了全部 IA-32 指令机器编码和操作内容。本书主要引用英特尔芯片手册，部分来自相关的规范标准。

软件技术，无论操作系统、开发平台或应用程序，只是一门应用技术而已，不一定需要深奥的抽象名词，需要的是朴素严谨的实现方法。读者将会发现，在熟悉了处理器和设备编程接口后，操作系统和高级语言程序实现并不是神秘的事情，远比“句柄”和“钩”等名词容易理解。许多软件企业寄希望于别人公开代码，其实，研究别人代码还不如去翻阅一下相关的总线协议和设备标准。常识告诉我们——出谜容易猜谜难，即使真想猜谜也不妨先看看谜底。读者朋友，让我们一起掀掉所谓软件核心技术的面纱吧。

处理器指令集分为特权指令集与用户指令集，系统特权代码才允许执行特权指令。出于安全性和兼容性考虑，当代操作系统连设备 I/O 端口也不让用户直接访问。如果不能访问系统寄存器和试探硬件响应，我们仍将陷于坐而论道的窘境。因此，软件专业教程需构建一个结构简洁、代码开放的操作系统，作为软件试验平台，不妨叫**教育操作系统**(EOS)，国外叫**实验操作系统**。该操作系统不要求兼容传统设备，但必须追踪最新软硬件技术，既允许开放系统权限也可转换为实用操作系统，姑且称操作系统“概念车”。希望本书能起到抛砖引玉的作用，期待有志的读者朋友打造出操作系统“概念车”。

编 者

2005 年 2 月

目　录

第 1 章　系统引导和程序代码 1
1.1　系统运行 1
1.1.1　系统引导过程 1
1.1.2　程序加载步骤 3
1.1.3　指令执行流程 4
1.2　机器指令 5
1.2.1　机器指令结构 5
1.2.2　操作数寻址模式 7
1.3　地址映射 10
1.3.1　实地址模式 10
1.3.2　保护模式 11
1.3.3　虚拟 8086 模式 13
1.3.4　系统管理模式 13
1.4　数据编码 14
1.4.1　整数编码 14
1.4.2　实数编码 15
1.4.3　数据存储格式 15
1.5　存储器组织 16
1.5.1　存储器物理地址 16
1.5.2　存储器寻址 17
1.5.3　存储器分页映射 17
1.5.4　存储器管理模式 18
1.6　汇编程序 18
1.6.1　处理器与指令集简介 18
1.6.2　汇编指令格式 19
1.6.3　变量声明 20
1.6.4　汇编程序结构 20
1.6.5　控制转移 21
1.6.6　堆栈 22
1.7　中断 23
第 2 章　系统配置接口 25
2.1　系统结构 25
2.1.1　系统总线分层结构 26
2.1.2　系统配置端口 28

2.2 系统总线与地址空间 29
2.2.1 总线组成与总线周期 29
2.2.2 系统存储器地址空间和 I/O 地址空间 30
2.2.3 流水线与并行机制 31
2.3 系统资源 31
2.3.1 I/O 中断控制器与 IRQ 通道 31
2.3.2 DMA 控制器与 DMA 通道 33
2.4 PCI 局部总线 34
2.4.1 PCI 总线协议 34
2.4.2 PCI 配置空间 36
2.4.3 PCI 设备枚举协议 37
2.4.4 AGP 局部总线 37
2.5 设备 PCI 空间配置接口 39
2.5.1 GMCH 芯片集内置设备配置接口 39
2.5.2 ICH4 芯片集内置设备配置接口 42
2.6 系统管理接口 47
2.6.1 系统控制接口 47
2.6.2 电源管理接口 48
2.6.3 系统管理总线接口 49
第 3 章 控制器编程接口 51
3.1 实时时钟和计时器 51
3.1.1 实时时钟 51
3.1.2 记时器 53
3.2 键盘鼠标控制器 54
3.2.1 键盘扫描码 54
3.2.2 鼠标数据包格式 55
3.2.3 键盘鼠标控制器接口 56
3.3 图形控制器 59
3.3.1 图形基础知识 59
3.3.2 图形控制器寄存器 61
3.3.3 图形控制器指令集 65
3.3.4 VGA/EVGA 控制器 71
3.4 IDE 控制器 76
3.4.1 ATA/ATAPI-5 标准介绍 76
3.4.2 ATA 设备寄存器 78
3.4.3 IDE 总线主控接口 79
3.4.4 ATA 设备命令集 79
3.5 AC'97 控制器 84
3.5.1 AC'97 连接标准介绍 84
3.5.2 调制解调器寄存器 86

3.5.3 音频编解码器寄存器 86
3.5.4 AC'97 主控接口 88
3.6 USB 控制器 89
3.6.1 USB 总线协议介绍 89
3.6.2 USB 设备配置 91
3.6.3 USB 主控寄存器 92
3.6.4 USB 总线事务调度 93
3.7 LAN 控制器 98
3.7.1 网络简介 98
3.7.2 LAN 控制器寄存器 99
3.7.3 LAN 主控事务接口 102
第 4 章 IA-32 处理器 105
4.1 基本寄存器与基本指令集 105
4.1.1 通用寄存器 105
4.1.2 段寄存器 106
4.1.3 指令指针寄存器 107
4.1.4 状态标志寄存器 108
4.1.5 控制寄存器 109
4.1.6 存储器管理寄存器 110
4.1.7 基本指令集 112
4.2 FPU 寄存器与指令集 112
4.2.1 浮点寄存器栈 112
4.2.2 FPU 标志寄存器 113
4.2.3 FPU 控制寄存器 113
4.2.4 FPU 状态寄存器 114
4.2.5 操作数指针、指令指针和操作码寄存器 115
4.2.6 FPU 状态保存与复原 115
4.2.7 FPU 指令集 115
4.3 MMX 寄存器与 MMX 指令集 116
4.3.1 MMX 寄存器 116
4.3.2 MMX 指令集 117
4.3.3 MMX 与 FPU 指令混用问题 117
4.4 XMM 寄存器与 SSE 指令 118
4.4.1 XMM 寄存器和 MXCSR 寄存器 118
4.4.2 SSE 例外 119
4.4.3 SSE/SSE2 指令集 120
4.5 IA-32e 处理器 123
4.5.1 存储器寻址 124
4.5.2 寄存器 124
4.5.3 指令集 125

第 5 章 中断机制 126
5.1 中断类型 126
5.1.1 中断源 126
5.1.2 例外源及分类 126
5.1.3 可屏蔽与不可屏蔽中断 127
5.2 中断向量 127
5.2.1 中断入口 128
5.2.2 中断优先权 129
5.2.3 例外错误码 130
5.3 中断处理机制 130
5.3.1 中断描述符表 130
5.3.2 中断处理硬件机制 131
5.3.3 中断返回硬件机制 131
5.3.4 IA-32e 处理器中断机制 132
5.4 局部 APIC 控制器 132
5.4.1 局部 APIC 介绍 133
5.4.2 局部 APIC 寄存器 134
5.4.3 局部向量表 136
5.4.4 发布中断命令 138
5.4.5 中断消息接收与提交 140
5.4.6 APIC 总线优先权仲裁与总线消息格式 142
5.5 I/O APIC 控制器 143
第 6 章 高级语言程序实现 146
6.1 变量 146
6.1.1 变量空间 147
6.1.2 数据结构 148
6.1.3 类 150
6.2 赋值和算术运算 151
6.2.1 数据传递 151
6.2.2 整数运算 153
6.2.3 浮点数运算 153
6.2.4 算术表达式编译 155
6.3 控制转移 156
6.3.1 转移指令 157
6.3.2 堆栈切换 159
6.4 条件转移 160
6.4.1 条件编码与状态标记 160
6.4.2 比较指令与条件执行指令 160
6.4.3 逻辑表达式编译 161
6.5 函数 164

6.5.1 函数出口 165
6.5.2 调用协议 165
6.5.3 块结构 166
6.5.4 快速系统调用 168
6.6 可移植执行文件与通用目标文件格式 169
第 7 章 存储器管理 172
7.1 分段机制 172
7.1.1 段描述符表 172
7.1.2 段描述符数据结构 173
7.1.3 段选择符 174
7.1.4 段保护机制 175
7.1.5 IA-32e 模式段描述符 175
7.2 分页机制 176
7.2.1 分页模式 176
7.2.2 页入口与线性地址译码 177
7.2.3 地址译码后援缓冲器 181
7.3 保护机制 181
7.3.1 段限长检查 181
7.3.2 段类型检查 181
7.3.3 段特权级检查 182
7.3.4 页保护机制 183
7.3.5 指针确认 183
7.3.6 特权指令 184
7.4 调用 16 位代码 185
7.5 高速缓冲机制 186
7.5.1 高速缓存结构 186
7.5.2 地址空间存储器类型分区 187
7.5.3 高速缓存控制 189
7.5.4 设置存储器类型 191
7.6 存储器一致性机制 192
7.6.1 存储器顺序规则 192
7.6.2 存储器顺序加强和削弱机制 193
7.6.3 原子操作 193
7.6.4 自修改代码同步问题 194
7.6.5 页入口修改同步问题 195
7.7 存储器管理模式 195
7.7.1 平面模式与分段模式 195
7.7.2 分页模式 196
7.7.3 线性地址空间管理示例 196
7.7.4 存储器空间管理与内存块表 199

7.8 存储器管理示例 200
7.8.1 存储器管理数据结构 200
7.8.2 页帧分配与回收 201
7.8.3 页故障处理程序 206
7.8.4 系统内核空间和全局堆 207
第 8 章 任务管理 210
8.1 任务空间 210
8.1.1 任务管理空间 210
8.1.2 程序运行空间 211
8.2 任务状态段 212
8.2.1 任务状态段结构 212
8.2.2 任务寄存器 213
8.2.3 任务门描述符 213
8.2.4 IA-32e 模式任务状态段 214
8.3 任务切换 214
8.3.1 任务切换步骤 215
8.3.2 任务链接 217
8.3.3 运行首个任务 217
8.3.4 创建新任务 218
8.4 SIMD 状态保存 219
8.5 任务管理示例 221
8.5.1 任务管理数据结构 222
8.5.2 线程调度 223
8.5.3 定时线程与计时器 225
8.5.4 空闲线程 226
8.5.5 消息接收与焦点窗口 227
8.6 应用程序结构和消息处理函数 229
8.6.1 消息读取、识别与发布 230
第 9 章 外存管理和设备管理 232
9.1 存储介质空间结构 232
9.1.1 磁盘引导区 234
9.1.2 分区入口 234
9.2 常用文件系统 235
9.2.1 FAT 文件系统 235
9.2.2 NTFS 文件系统 236
9.2.3 系统引导区 237
9.3 外存管理示例 239
9.3.1 块表入口 239
9.3.2 文件入口 240
9.3.3 系统根目录 242

9.3.4 文件系统接口 242
9.4 文件接口示例 243
9.4.1 外存设备与端口设备 245
9.4.2 文件接口实现 245
9.4.3 设备接口 246
9.4.4 页帧加载 248
9.5 设备管理 250
9.5.1 枚举与配置 250
9.5.2 ACPI 规范简介 252
9.5.3 资源管理 254
9.5.4 设备事件 255
第 10 章 处理器辅助功能 256
10.1 模式专用寄存器 256
10.2 处理器初始化与多处理器协议 258
10.2.1 处理器复位状态 259
10.2.2 多处理器协议 259
10.3 处理器识别 261
10.4 系统管理模式 262
10.4.1 SMRAM 空间结构 263
10.4.2 系统管理中断处理入口 264
10.5 系统初始化 265
10.5.1 初始化硬件平台 265
10.5.2 初始化程序运行环境 265
10.6 升级微码块 269
10.6.1 微码块结构 269
10.6.2 微码块加载器 269
10.6.3 微码块嵌入接口 270
10.7 处理器温度监控 271
10.7.1 紧急关机温度开关 271
10.7.2 自动温控器 272
10.7.3 软件调节 272
10.8 调试机制 272
10.8.1 调试寄存器与调试例外 272
10.8.2 最后转移记录 274
10.8.3 调试信息存储 275
10.9 性能监控 276
10.9.1 P6 系列处理器性能监控 276
10.9.2 奔腾®4 处理器性能监控 279
10.10 机器检查 283
10.10.1 错误报告栏 283

10.10.2 MC 错误编码 284
10.10.3 MC 例外处理 285
第 11 章 安腾处理器 286
11.1 安腾处理器简介 286
11.2 安腾寄存器 287
11.2.1 通用寄存器 287
11.2.2 浮点寄存器 288
11.2.3 判断寄存器、转移寄存器和指令指针 289
11.2.4 当前帧标记寄存器 289
11.2.5 处理器状态寄存器 289
11.2.6 应用寄存器 290
11.2.7 控制寄存器 291
11.2.8 处理器标识寄存器 292
11.2.9 调试断点寄存器 292
11.2.10 性能监控寄存器 292
11.3 代码结构和指令集 293
11.3.1 机器代码结构与汇编指令格式 293
11.3.2 访存指令与读存指令提前机制 295
11.3.3 运算指令 297
11.3.4 比较指令与判断寄存器 300
11.4 转移指令与软件协议 301
11.4.1 循环指令与软件流水线 302
11.4.2 调用/返回指令与寄存器帧 303
11.4.3 寄存器保存协议 304
11.4.4 寄存器栈备份空间 305
11.4.5 存储器栈与参数传递协议 306
11.5 存储器管理 307
11.5.1 地址译码和页保护机制 307
11.5.2 页入口插入 308
11.5.3 程序空间结构 310
11.6 中断处理机制 312
11.6.1 中断类型与中断向量表 313
11.6.2 中断块和中断命令端口 314
11.6.3 中断和中断返回 315
11.6.4 中断处理程序 316
11.7 I/O 端口 317
11.8 处理器抽象层 318
11.8.1 固件地址空间 318
11.8.2 安腾系统引导 319
11.8.3 硬件事件处理 320

11.8.4 处理器抽象层程序 320
附录 A IA-32 指令一览表 322
A.1 汇编指令操作数符号 322
A.2 机器指令操作数符号 322
A.3 IA-32 指令集一览表 323
附录 B 操作码映射表 356
附录 C 英文缩写词 362
参考文献 364

第 1 章　系统引导和程序代码

在每个指令边界，处理器寄存器集和系统存储器当前值构成了设备状态。在执行每条指令之前，设备状态是已知的。处理器将根据设备状态和指令要求修改设备状态，程序的运行过程就是设备状态转换过程。在启动应用程序时，操作系统负责设置应用程序的初始状态。当多条并行指令访问设备状态时，硬件或软件负责保障数据一致性，如封锁总线或串行化访存指令。

在系统复位时，硬件逻辑电路把处理器和系统设备置为已知状态(叫硬布线)，为系统引导代码提供预定的设备状态和引导接口。系统引导代码负责初始化基本 IO 设备，以满足引导时交互需要，安装可能存放操作系统的设备，如硬盘和光盘驱动器，然后拷贝和运行操作系统加载器。当操作系统接管系统后，系统初始化程序识别和配置系统设备及外部设备，创建或加载系统数据，安装系统服务和中断处理代码，最后创建和启动首个任务。

程序由代码和数据组成，代码存储空间叫**代码段**，数据存储空间叫**数据段**，两者构成程序的运行空间。机器指令由操作类型、操作对象和操作方向等编码域组成，处理器结构标准不同，机器指令结构和编码也就不同，相应地汇编程序指令结构也不同。存储器中的数据编码由行业标准定义，与结构标准无关，但寄存器中的数据格式与结构标准相关。

在指令执行过程中，处理器从内存读取指令，并根据指令要求访问内存数据。存储器单元叫**存储器操作数**，地址译码器按“段基址+有效地址”计算存储器操作数的线性地址，段偏移量就是**有效地址**。若没启用处理器分页机制，线性地址等于存储器物理地址，否则线性地址将通过页入口映射到存储器物理地址。段基址由操作系统设置，有效地址由指令确定。机器指令可以直接编码有效地址，或编码有效地址所在的寄存器。有效地址的生成方式叫**存储器寻址模式**，IA-32 结构支持 24 种寻址模式。习惯上，存放存储器操作数地址的寄存器或存储器单元称**指针**，有效地址叫**近指针**，若包含段选择符或段基址叫**远指针**。

1.1　系 统 运 行

1.1.1　系统引导过程

一旦按下电源按钮或复位开关，处理器复位引线 RESET#将被置位(叫**冷启动**)，计算机开始启动，处理器在完成自检后运行 ROM 中的系统引导代码。系统引导主要任务是搜索和安装操作系统加载程序，下面给出基于 IA-32 结构系统启动过程。

(1) **处理器复位** 代码段寄存器 CS 置 F000h，指令指针寄存器 EIP 置 0000FFF0h，而且处于实地址模式。IA-32 处理器复位后虽然处于 16 位实地址模式，即支持 20 位段基址和 16 位有效地址，但地址总线 A[31:20]强制置 1。因此，处理器将从物理地址 FFFFFFF0h 读取首条指令，系统 BIOS ROM 空间必须被映射到系统存储器地址空间的高位区。

（2）**内置自检** 各个硬件单元提交电气信号或数据包，进行相互确认和校验，本过程叫**系统内置自检**。通过系统引导菜单界面，用户可以修改NVRAM(如CMOS)设置跳过自检步骤。

（3）**仲裁处理器** 仲裁电路根据多处理器协议(MP)确定1个引导处理器(BSP)，并把总线上其余处理器设置为应用处理器(AP)。应用处理器处于暂停状态，等待引导处理器调度。

（4）**运行引导代码** 引导处理器从物理地址FFFFFFF0h，开始读取系统引导代码的首条指令。该指令通常跳转到BIOS ROM中标准I/O设备(显示器和键盘)初始化代码，以便为用户提供引导时交互界面，然后运行系统引导代码。系统引导代码按用户配置顺序(记录在NVRAM)，逐个试探引导IPL设备(初始程序加载器)，即可能存放或加载操作系统的设备，比如磁盘驱动器或网卡。系统引导接口由系统引导协议定义，存储设备和操作系统须遵守系统引导协议。下面介绍PnP BIOS和EFI两种系统引导协议的引导步骤。

① **BIOS引导协议** 引导代码初始化内置IPL设备，遍历OPROM空间(C0000h-EFFFFh)并调用PnP设备初始化代码，安装块设备即INT 13中断服务代码，建立IPL设备入口表，如果设备变化重新建立IPL优先级(存放在NVRAM)。然后调用INT 19处理程序，该程序按IPL优先级遍历IPL设备表，逐个调用设备引导程序，IPL入口包含设备类型和引导程序指针。块设备引导程序将拷贝存储介质的首个扇区(叫**引导区**)，如拷贝到内存[0000:7C00]，然后跳转到该地址，开始运行存储介质引导程序。介质引导区包含引导程序和分区表，介质引导程序也叫**主引导记录**(MBR)。MBR负责搜索分区表查找系统分区(参见9.1.1小节)，如果发现系统分区，MBR将拷贝系统引导区并跳转到首字节，否则发布INT 18h中断返回INT 19处理程序，试探下一个IPL设备。介质引导区属于PC-AT系统结构引导协议，在安装操作系统时，安装程序负责修改系统引导区，让首字节转移指令跳转到操作系统加载器。安装操作系统的磁盘分区叫**系统分区**，该分区引导区叫**系统引导区**。

② **EFI引导协议** 引导代码首先加载EFI文件映像，如引导服务代码、运行服务代码和应用代码模块，然后按候选加载对象(EFI_LOAD_OPTION)顺序表逐个引导设备。如果该表无可引导对象，引导代码将枚举移动驱动器和固定驱动器，建立临时顺序表，再逐个引导。顺序表是一个候选加载对象数组，每个对象包含设备路径、设备参数和活动属性等信息，设备路径定义了设备基本类型、子类型、路径长度、设备地址或文件路径。每个候选加载对象定义一条操作系统加载器路径，如硬盘路径“EFI\系统目录\加载器名”，移动盘默认路径“EFI\BOOT\BOOTIA32.EFI”。系统引导代码直接拷贝和执行加载器文件，这意味系统引导代码还得支持文件系统。在基于EFI标准存储介质中，首个扇区仍作为传统的引导区，以兼容既有的引导协议，第2扇区设置分区表头和分区入口，并在最后一个扇区进行备份。

（5）**运行操作系统加载器** 该程序枚举系统设备，收集设备资源需求信息，配置系统设备和PnP设备，然后依次调用系统设备和外部设备初始化代码，最后加载和运行系统初始化程序。在EFI系统中，操作系统加载器须调用ExitBootServices接口结束系统引导。

（6）**运行操作系统初始化程序** 该程序将建立全局描述符表和中断描述符表，加载系统服务模块和系统管理数据。把控制寄存器CR0位PE置1，处理器进入保护模式(EFI协议无须此步)。执行STI指令启用处理器中断机制，等待接收和检查用户口令。最后，加载和运行浏览器任务，该任务在显示系统桌面后进入“等待消息 — HLT — 处理消息”循环。

通过局部高级可编程中断控制器(APIC)，软件能够发布INIT中断命令，通知处理器复位，这种启动方式叫**热启动**。热启动将不执行内置自检和多处理器协议，而且内部高速缓存、模式专用寄存器和浮点协处理器等设备状态均保持不变。处理器从保护模式切换回

实地址模式常采用热启动方式，如点击Windows 9x关机窗口“切换到DOS模式”菜单项。

另外，软件还可通过局部APIC发布RESET中断命令，重新启动系统。在修改静态设备资源配置后，如修改DOS的CONFIG.SYS文件、Windows系统注册表或插入非PnP设备，系统需要重新启动才能使设备配置生效，所以有的设备安装程序会提示“重新启动”。软件通过系统复位端口(0CF9h)同样可以复位系统，通过高级配置与电源管理接口(ACPI)还能够实现软关机，即设置PM1a_CNT寄存器(参见表2-27)，让系统进入S_5睡眠状态。

通过系统BIOS接口，如执行INT 19指令或转移到F000:FFF0，也可以重启系统。

1.1.2 程序加载步骤

如果操作系统支持多任务和多线程，程序加载器在启动应用程序时需要创建任务和主线程结点，并为应用程序提供子线程创建和关闭接口。任务结点用于管理程序运行的地址空间和物理空间，如页目录和文件映像，并通过任务结点管理该任务线程。线程结点用于管理程序现场状态存储空间即任务状态段，并链接程序图形界面即窗口结点(如有的话)。IA-32结构提供程序状态切换机制，自动保存旧程序的现场状态和加载新程序的现场状态，叫**任务切换**。在执行新程序时，处理器在任务状态段保存寄存器状态，再以目标任务状态段加载寄存器。在多线程系统，任务切换实际上是线程切换。线程调度程序将按一定间隔强制切换处理器线程，从而实现模拟线程并行。

程序加载器负责创建任务和线程结点，初始化任务状态段，分配程序地址空间，然后把该线程结点链接到就绪线程队列。下面介绍应用程序一般加载步骤：

(1)识别文件 当用户点击浏览器视图区的文件图标时，浏览器将以相应的文件入口为参数调用程序加载器。文件入口包含文件名、类型、地址和长度等信息，具体结构与外存介质所采用的文件系统有关，文件目录用于存放文件入口。程序加载器首先需要识别文件类型，若不是可执行文件则加载该文件默认的打开程序，而把目标文件作为数据文件。可执行文件由文件头、代码段和初始化数据段三部分组成，文件头记录了运行平台标识、文件长度、入口点、输入表和段表等信息，段表定义各个逻辑段的段类型、段长、物理地址和逻辑基址。加载器将拷贝文件头作为程序前置块，并根据段表逐段分配线性地址空间(内存空间按需分页分配)，根据输入表填写系统库函数指针。

(2)创建线程结点及任务状态段 调用内核空间分配函数请求系统主存，创建任务结点、主线程结点及任务状态段(TSS)等任务管理数据。程序加载器通过设置任务状态段初始化应用程序启动状态，把程序入口点填入任务状态段TSS.EIP域。

(3)创建段描述符 若存储器管理采用分段模式，程序加载器将创建局部描述表LDT，存放该程序相关的代码段、数据段和堆栈段描述符。每个段描述符8个字节，包含段基址、段限长、类型和特权级等内容。程序加载器根据段表为程序逐段分配线性地址空间，段线性基址=程序映像基址+段逻辑基址。段逻辑基址由段表入口定义，程序映像基址由操作系统决定。若存储器管理采用平面模式，段线性基址均置0，应用程序共享用户空间的段描述符，段描述符全部放在全局描述符表(GDT)。段描述符索引号(13位)、表选择位(LDT/GDT)和特权级(2位)构成了16位段选择符，根据段描述符位置和特权，把代码段、数据段和堆栈段选择符填入任务状态段的TSS.CS、TSS.DS和TSS.SS域。

(4)创建TSS和LDT描述符 在GDT创建任务状态段描述符，如采用分段模式还要在GDT创建LDT描述符，并把LDT选择符填入任务状态段TSS.LDTR域。操作系统初始化程序