x86 处理器架构

本章重点是与 x86 汇编语言相关的底层硬件。有说法认为，汇编语言是直接与机器交流的理想软件工具。如果是真的，那么汇编程序员就必须非常熟悉处理器的内部结构与功能。本章将讨论指令执行时处理器内部发生的一些基本操作，以及操作系统如何加载和执行程序，并通过样本主板布局来了解 x86 系统的硬件环境，最后还讨论了在应用程序与操作系统之间，层次化输入输出是如何工作的。本章所有主题为开始编写汇编语言程序提供了硬件基础。

2.1 一般概念

本章描述了 x86 处理器系列架构，以及从程序员角度看到的主机系统。其中包括了所有的 Intel IA-32 和 Intel 64 处理器，如奔腾（Intel Pentium）和酷睿双核（Core-Duo）处理器，还包括了高级微设备（AMD）处理器，如速龙（Athlon）、弈龙（Phenom）、皓龙（Opteron）和 AMD64。汇编语言是学习计算机如何工作的很好的工具，它需要读者具备计算机硬件的工作知识。为此，本章的概念和详细信息将帮助程序员理解自己所写的汇编代码。

本章在所有处理器都使用的概念与 x86 处理器特点之间进行了平衡。程序员将来可能要面对各种类型的处理器，因此，本章呈现的是通用概念。同时，为了避免对机器架构形成肤浅的认知，本章也关注 x86 处理器的特性，以便具备汇编编程的坚实基础。

> 若希望了解更多 Intel IA-32 架构，请参阅《Intel 64 与 IA-32 架构软件开发手册》的卷 1：基础架构（*Intel 64 and IA-32 Architectures Software Developer'S Manual*, Volume 1：Basic Architecture）。该文档可以从 Intel 网站免费下载（www.intel.com）。

2.1.1 基本微机设计

图 2-1 给出了假想机的基本设计。中央处理单元（CPU）是进行算术和逻辑操作的部件，包含了有限数量的存储位置——寄存器（register），一个高频时钟、一个控制单元和一个算术逻辑单元。

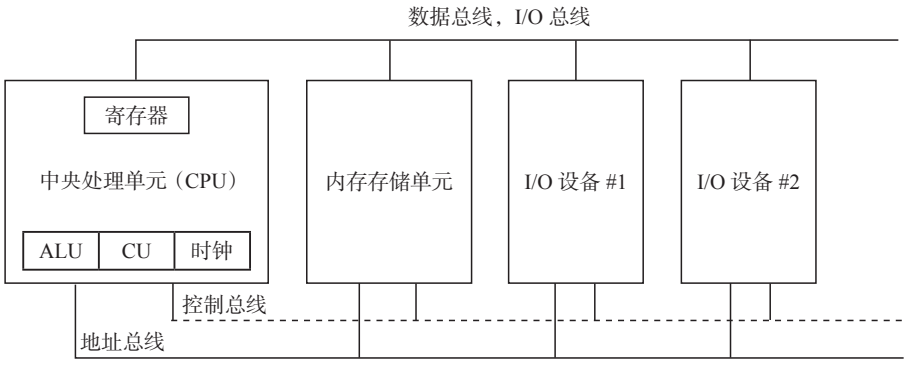


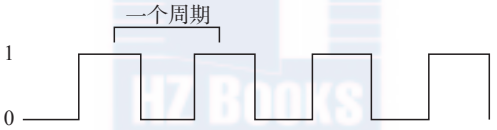
图 2-1 微计算机框图

- 时钟（clock）对 CPU 内部操作与系统其他组件进行同步。
- 控制单元（control unit，CU）协调参与机器指令执行的步骤序列。
- 算术逻辑单元（arithmetic logic unit，ALU）执行算术运算，如加法和减法，以及逻辑运算，如 AND（与）、OR（或）和 NOT（非）。

CPU 通过主板上 CPU 插座的引脚与计算机其他部分相连。大部分引脚连接的是数据总线、控制总线和地址总线。内存存储单元（memory storage unit）用于在程序运行时保存指令与数据。它接受来自 CPU 的数据请求，将数据从随机存储器（RAM）传输到 CPU，并从 CPU 传输到内存。由于所有的数据处理都在 CPU 内进行，因此保存在内存中的程序在执行前需要被复制到 CPU 中。程序指令在复制到 CPU 时，可以一次复制一条，也可以一次复制多条。

总线（bus）是一组并行线，用于将数据从计算机一个部分传送到另一个部分。一个计算机系统通常包含四类总线：数据类、I/O 类、控制类和地址类。数据总线（data bus）在 CPU
33 和内存之间传输指令和数据。I/O 总线在 CPU 和系统输入 / 输出设备之间传输数据。控制总线（control bus）用二进制信号对所有连接在系统总线上设备的行为进行同步。当前执行指令在 CPU 和内存之间传输数据时，地址总线（address bus）用于保持指令和数据的地址。

时钟 与 CPU 和系统总线相关的每一个操作都是由一个恒定速率的内部时钟脉冲来进行同步。机器指令的基本时间单位是机器周期（machine cycle）或时钟周期（clock cycle）。一个时钟周期的时长是一个完整时钟脉冲所需要的时间。下图中，一个时钟周期被描绘为两个相邻下降沿之间的时间：



时钟周期持续时间用时钟速度的倒数来计算，而时钟速度则用每秒振荡数来衡量。例如，一个每秒振荡 10 亿次（1GHz）的时钟，其时钟周期为 10 亿分之 1 秒（1 纳秒）。

执行一条机器指令最少需要 1 个时钟周期，有几个需要的时钟则超过了 50 个（比如 8088 处理器中的乘法指令）。由于在 CPU、系统总线和内存电路之间存在速度差异，因此，需要访问内存的指令常常需要空时钟周期，也被称为等待状态（wait states）。

2.1.2 指令执行周期

一条机器指令不会神奇地一下就执行完成。CPU 在执行一条机器指令时，需要经过一系列预先定义好的步骤，这些步骤被称为指令执行周期（instruction execution cycle）。假设
34 现在指令指针寄存器中已经有了想要执行指令的地址，下面就是执行步骤：

1）CPU 从被称为指令队列（instruction queue）的内存区域取得指令，之后立即增加指令指针的值。

2）CPU 对指令的二进制位模式进行译码。这种位模式可能会表示该指令有操作数（输入值）。

3）如果有操作数，CPU 就从寄存器和内存中取得操作数。有时，这步还包括了地址计算。

4）使用步骤 3 得到的操作数，CPU 执行该指令。同时更新部分状态标志位，如零标志

（Zero）、进位标志（Carry）和溢出标志（Overflow）。

5）如果输出操作数也是该指令的一部分，则 CPU 还需要存放其执行结果。

通常将上述听起来很复杂的过程简化为三个步骤：取指（Fetch）、译码（Decode）和执行（Execute）。操作数（operand）是指操作过程中输入或输出的值。例如，表达式 Z=X+Y 有两个输入操作数（X 和 Y），一个输出操作数（Z）。

图 2-2 是一个典型 CPU 中的数据流框图。该图表现了在指令执行周期中相互交互部件之间的关系。在从内存读取程序指令之前，将其地址放到地址总线上。然后，内存控制器将所需代码送到数据总线上，存入代码高速缓存（code cache）。指令指针的值决定下一条将要执行的指令。指令由指令译码器分析，并产生相应的数值信号送往控制单元，其协调 ALU 和浮点单元。虽然图中没有画出控制总线，但是其上传输的信号用系统时钟协调不同 CPU 部件之间的数据传输。

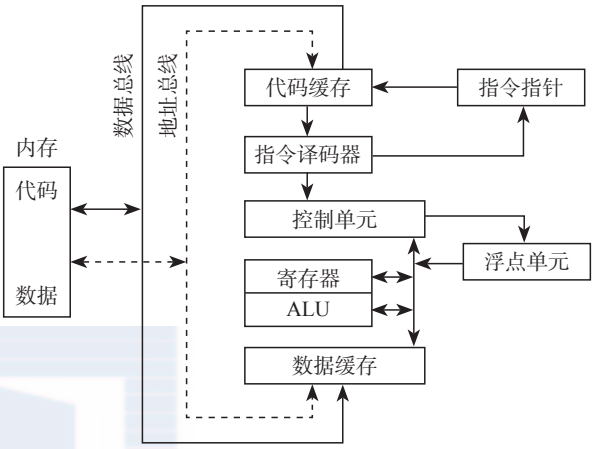


图 2-2 简化的 CPU 框图

2.1.3 读取内存

作为一个常见现象，计算机从内存读取数据比从内部寄存器读取速度要慢很多。这是因为从内存读取一个值，需要经过下述步骤：

1）将想要读取的值的地址放到地址总线上。

2）设置处理器 RD（读取）引脚（改变 RD 的值）。

3）等待一个时钟周期给存储器芯片进行响应。

4）将数据从数据总线复制到目标操作数。

上述每一步常常只需要一个时钟周期，时钟周期是基于处理器内固定速率时钟节拍的一种时间测量方法。计算机的 CPU 通常是用其时钟速率来描述。例如，速率为 1.2GHz 意味着时钟节拍或振荡为每秒 12 亿次。因此，考虑到每个时钟周期仅为 1/1 200 000 000 秒，4 个时钟周期也是非常快的。但是，与 CPU 寄存器相比，这个速度还是慢了，因为访问寄存器一般只需要 1 个时钟周期。

幸运的是，CPU 设计者很早之前就已经指出，因为绝大多数程序都需要访问变量，计算机内存成为了速度瓶颈。他们想出了一个聪明的方法来减少读写内存的时间——将大部分近期使用过的指令和数据存放在高速存储器 cache 中。其思想是，程序更可能希望反复访问相同的内存和指令，因此，cache 保存这些值就能使它们能被快速访问到。此外，当 CPU 开始执行一个程序时，它会预先将后续（比如）一千条指令加载到 cache 中，这个行为是基于这样一种假设，即这些指令很快就会被用到。如果这种情况重复发生在一个代码块中，则 cache 中就会有相同的指令。当处理器能够在 cache 存储器中发现想要的数据，则称为 cache 命中（cache hit）。反之，如果 CPU 在 cache 中没有找到数据，则称为 cache 未命中（cache miss）。

x86 系列中的 cache 存储器有两种类型：一级 cache（或主 cache）位于 CPU 上；二级

cache（或次 cache）速度略慢，通过高速数据总线与 CPU 相连。这两种 cache 以最佳方式一起工作。

还有一个原因使得 cache 存储器比传统 RAM 速度快—— cache 存储器是由一种被称为静态 RAM（static RAM）的特殊存储器芯片构成的。这种芯片比较贵，但是不需要为了保持其内容进行不断地刷新。另一方面，传统存储器，即动态 RAM（dynamic RAM），就需要持续刷新。它速度慢一些，但是价格更便宜。

2.1.4 加载并执行程序

在程序执行之前，需要用一种工具程序将其加载到内存，这种工具程序称为*程序加载器*（program loader）。加载后，操作系统必须将 CPU 指向程序的*入口*，即程序开始执行的地址。

36 以下步骤是对这一过程的详细分解。

- 操作系统（OS）在当前磁盘目录下搜索程序的文件名。如果找不到，则在预定目录列表（称为路径（path））下搜索文件名。当 OS 无法检索到文件名时，它会发出一个出错信息。
- 如果程序文件被找到，OS 就访问磁盘目录中的程序文件基本信息，包括文件大小，及其在磁盘驱动器上的物理位置。
- OS 确定内存中下一个可使用的位置，将程序文件加载到内存。为该程序分配内存块，并将程序大小和位置信息加入表中（有时称为*描述符表*（descriptor table））。另外，OS 可能调整程序内指针的值，使得它们包括程序数据地址。
- OS 开始执行程序的第一条机器指令（程序入口）。当程序开始执行后，就成为一个进程（process）。OS 为这个进程分配一个标识号（进程 ID），用于在执行期间对其进行追踪。
- 进程自动运行。OS 的工作是追踪进程的执行，并响应系统资源的请求。这些资源包括内存、磁盘文件和输入输出设备等。
- 进程结束后，就会从内存中移除。

> **提示** 不论使用哪个版本的 Microsoft Windows，按下 Ctrl-Alt-Delete 组合键，可以选择任务管理器（task manager）选项。在任务管理器窗口可以查看应用程序和进程列表。应用程序列表中列出了当前正在运行的完整程序名称，比如，Windows 浏览器，或者 Microsoft Visual C++。如果选择进程列表，则会看见一长串进程名。其中的每个进程都是一个独立于其他进程的，并处于运行中的小程序。可以连续追踪每个进程使用的 CPU 时间和内存容量。在某些情况下，选定一个进程名称后，按下 Delete 键就可以关闭该进程。

2.1.5 本节回顾

1. 中央处理单元（CPU）包含寄存器和哪些其他基本部件？
2. 中央处理单元通过哪三种总线与计算机系统的其他部分相连？
3. 为什么访问存储器比访问寄存器要花费更多的机器周期？
4. 指令执行周期包含哪三个基本步骤？
5. 指令执行周期中，如果用到存储器操作数，则还需要哪两个步骤？

2.2 32 位 x86 处理器

本节重点在于所有 x86 处理器的基本架构特点。这些处理器包括了 Intel IA-32 系列中的成员和所有 32 位 AMD 处理器。

2.2.1 操作模式

x86 处理器有三个主要的操作模式：保护模式、实地址模式和系统管理模式；以及一个子模式：虚拟 8086（virtual-8086）模式，这是保护模式的特殊情况。以下是对这些模式的简介：

37

保护模式（Protected Mode） 保护模式是处理器的原生状态，在这种模式下，所有的指令和特性都是可用的。分配给程序的独立内存区域被称为段，而处理器会阻止程序使用自身段范围之外的内存。

虚拟 8086 模式（Virtual-8086 Mode） 保护模式下，处理器可以在一个安全环境中，直接执行实地址模式软件，如 MS-DOS 程序。换句话说，如果一个程序崩溃了或是试图向系统内存区域写数据，都不会影响到同一时间内执行的其他程序。现代操作系统可以同时执行多个独立的虚拟 8086 会话。

实地址模式（Real-Address Mode） 实地址模式实现的是早期 Intel 处理器的编程环境，但是增加了一些其他的特性，如切换到其他模式的功能。当程序需要直接访问系统内存和硬件设备时，这种模式就很有用。

系统管理模式（System Management Mode） 系统管理模式（SMM）向操作系统提供了实现诸如电源管理和系统安全等功能的机制。这些功能通常是由计算机制造商实现的，他们为了一个特定的系统设置而定制处理器。

2.2.2 基本执行环境

1. 地址空间

在 32 位保护模式下，一个任务或程序最大可以寻址 4GB 的线性地址空间。从 P6 处理器开始，一种被称为*扩展物理寻址*（extended physical addressing）的技术使得可以被寻址的物理内存空间增加到 64GB。与之相反，实地址模式程序只能寻址 1MB 空间。如果处理器在保护模式下运行多个虚拟 8086 程序，则每个程序只能拥有自己的 1MB 内存空间。

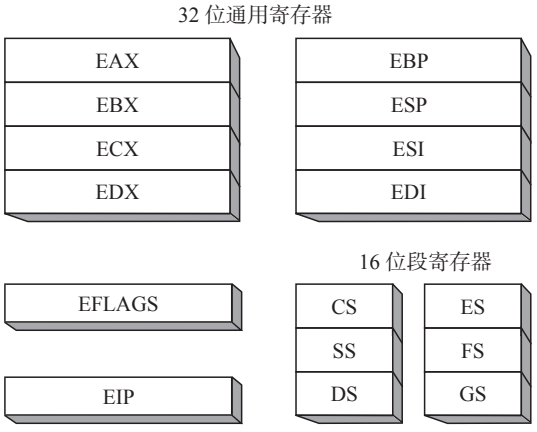


图 2-3 基本程序执行寄存器

38

2. 基本程序执行寄存器

寄存器是直接位于 CPU 内的高速存储位置，其设计访问速度远高于传统存储器。例如，当一个循环处理为了速度进行优化时，其循环计数会保留在寄存器中而不是变量中。图 2-3 展示的是基本程序执行寄存器（basic program execution registers）。8 个通用寄存器，6 个段寄存器，一个处理器状态标志寄存器（EFLAGS），和一个指令指针寄存器（EIP）。

通用寄存器 通用寄存器主要用于算术运算和数据传输。如图 2-4 所示，EAX 寄存器的低 16 位在使用时可以用 AX 表示。

一些寄存器的组成部分可以处理 8 位的值。例如，AX 寄存器的高 8 位被称为 AH，而低 8 位被称为 AL。同样的重叠关系也存在于 EAX、EBX、ECX 和 EDX 寄存器中：

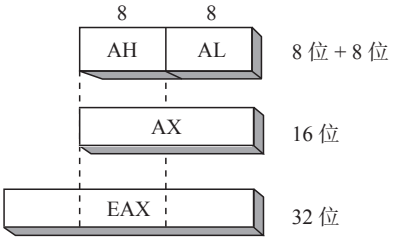


图 2-4 通用寄存器

32 位	16 位	8 位（高）	8 位（低）
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

其他通用寄存器只能用 32 位或 16 位名称来访问，如下表所示：

32 位	16 位	32 位	16 位
ESI	SI	EBP	BP
EDI	DI	ESP	SP

39

特殊用法 某些通用寄存器有特殊用法：

- 乘除指令默认使用 EAX。它常常被称为扩展累加器（extended accumulator）寄存器。
- CPU 默认使用 ECX 为循环计数器。
- ESP 用于寻址堆栈（一种系统内存结构）数据。它极少用于一般算术运算和数据传输，通常被称为扩展堆栈指针（extended stack pointer）寄存器。
- ESI 和 EDI 用于高速存储器传输指令，有时也被称为扩展源变址（extended source index）寄存器和扩展目的变址（extended destination index）寄存器。
- 高级语言通过 EBP 来引用堆栈中的函数参数和局部变量。除了高级编程，它不用于一般算术运算和数据传输。它常常被称为扩展帧指针（extended frame pointer）寄存器。

段寄存器 实地址模式中，16 位段寄存器表示的是预先分配的内存区域的基址，这个内存区域称为段。保护模式中，段寄存器中存放的是段描述符表指针。一些段中存放程序指令（代码），其他段存放变量（数据），还有一个堆栈段存放的是局部函数变量和函数参数。

指令指针 指令指针（EIP）寄存器中包含下一条将要执行指令的地址。某些机器指令能控制 EIP，使得程序分支转向到一个新位置。

EFLAGS 寄存器 EFLAGS（或 Flags）寄存器包含了独立的二进制位，用于控制 CPU 的操作，或是反映一些 CPU 操作的结果。有些指令可以测试和控制这些单独的处理器标志位。

设置标志位时，该标识位 =1；清除（或重置）标识位时，该标志位 =0。

控制标志位 控制标志位控制 CPU 的操作。例如，它们能使得 CPU 每执行一条指令后进入中断；在侦测到算术运算溢出时中断执行；进入虚拟 8086 模式，以及进入保护模式。

程序能够通过设置 EFLAGS 寄存器中的单独位来控制 CPU 的操作，比如，方向标志位和中断标志位。

状态标志位 状态标志位反映了 CPU 执行的算术和逻辑操作的结果。其中包括：溢出位、符号位、零标志位、辅助进位标志位、奇偶校验位和进位标志位。下述说明中，标志位的缩写紧跟在标志位名称之后：

- 进位标志位（CF），与目标位置相比，无符号算术运算结果太大时，设置该标志位。
- 溢出标志位（OF），与目标位置相比，有符号算术运算结果太大或太小时，设置该标志位。
- 符号标志位（SF），算术或逻辑操作产生负结果时，设置该标志位。
- 零标志位（ZF），算术或逻辑操作产生的结果为零时，设置该标志位。 40
- 辅助进位标志位（AC），算术操作在 8 位操作数中产生了位 3 向位 4 的进位时，设置该标志位。
- 奇偶校验标志位（PF），结果的最低有效字节包含偶数个 1 时，设置该标志位，否则，清除该标志位。一般情况下，如果数据有可能被修改或损坏时，该标志位用于进行错误检测。

3. MMX 寄存器

在实现高级多媒体和通信应用时，MMX 技术提高了 Intel 处理器的性能。8 个 64 位 MMX 寄存器支持称为 SIMD（单指令，多数据，Single-Instruction，Multiple-Data）的特殊指令。顾名思义，MMX 指令对 MMX 寄存器中的数据值进行并行操作。虽然，它们看上去是独立的寄存器，但是 MMX 寄存器名实际上是浮点单元中使用的同样寄存器的别名。

4. XMM 寄存器

x86 结构还包括了 8 个 128 位 XMM 寄存器，它们被用于 SIMD 流扩展指令集。

浮点单元 浮点单元（FPU，floating-point unit）执行高速浮点算术运算。之前为了这个目的，需要一个独立的协处理器芯片。从 Intel486 处理器开始，FPU 已经集成到主处理器芯片上。FPU 中有 8 个浮点数据寄存器，分别命名为 ST（0），ST（1），ST（2），ST（3），ST（4），ST（5），ST（6）和 ST（7）。其他控制寄存器和指针寄存器如图 2-5 所示。

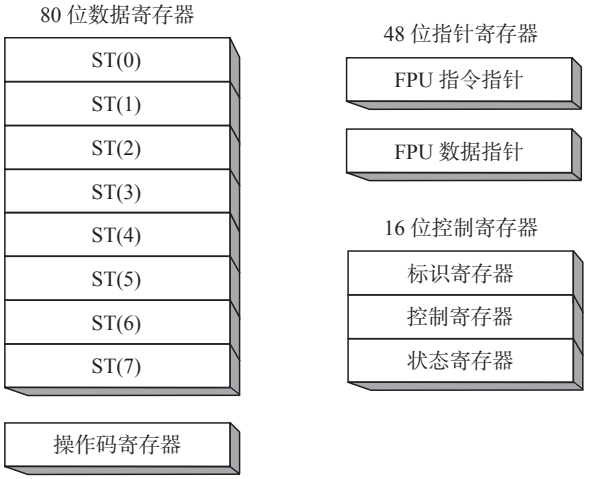


图 2-5 浮点单元寄存器

2.2.3 x86 内存管理

x86 处理器按照 2.2.1 节中讨论的基本操作模式来管理内存。保护模式是最可靠、最强
41 大的，但是它对应用程序直接访问系统硬件有着严格的限制。

在实地址模式中，只能寻址 1MB 内存，地址从 00000H 到 FFFFFH。处理器一次只能运行一个程序，但是可以暂时中断程序来处理来自外围设备的请求（称为中断（interrupt））。应用程序被允许访问内存的任何位置，包括那些直接与系统硬件相关的地址。MS-DOS 操作系统在实地址模式下运行，Windows 95 和 98 能够引导进入这种模式。

在保护模式中，处理器可以同时运行多个程序，它为每个进程（运行中的程序）分配总共 4GB 的内存。每个程序都分配有自己的保留内存区域，程序之间禁止意外访问其他程序的代码和数据。MS-Windows 和 Linux 运行在保护模式下。

在虚拟 8086 模式中，计算机运行在保护模式下，通过创建一个带有 1MB 地址空间的虚拟 8086 机器来模拟运行于实地址模式的 80x86 计算机。例如，在 Windows NT 和 2000 下，当打开一个命令窗口时，就创建了一个虚拟 8086 机器。同一时间可以运行多个这样的窗口，并且窗口之间都是受到保护的。在 Windows NT，2000 和 XP 系统中，某些需要直接使用计算机硬件的 MS-DOS 程序不能运行在虚拟 8086 模式下。

实地址模式和保护模式的更多细节将在第 11 章中进行详述。

2.2.4 本节回顾

1. x86 处理器的 3 个基本操作模式是什么？
2. 给出 8 个 32 位通用寄存器的名称。
3. 给出 6 个段寄存器的名称。
4. ECX 寄存器的特殊用途是什么？

2.3 64 位 x86-64 处理器

本节重点关注所有使用 x86-64 指令集的 64 位处理器的基本架构细节。这些处理器包括 Intel 64 和 AMD64 处理器系列。指令集是已讨论的 x86 指令集的 64 位扩展。以下为一些基本特征：

1）向后兼容 x86 指令集。

2）地址长度为 64 位，虚拟地址空间为 2^{64} 字节。按照当前芯片的实现情况，只能使用地址的低 48 位。

3）可以使用 64 位通用寄存器，允许指令具有 64 位整数操作数。

4）比 x86 多了 8 个通用寄存器。

5）物理地址为 48 位，支持高达 256TB 的 RAM。

另一方面，当处理器运行于本机 64 位模式时，是不支持 16 位实模式或虚拟 8086 模式的。（在传统模式（legacy mode）下，还是支持 16 位编程，但是在 Microsoft Windows 64 位版本中不可用。）

> **注意** 尽管 x86-64 指的是指令集，但是也可以将其看作是处理器类型。学习汇编语言时，没有必要考虑支持 x86-64 的处理器之间的硬件实现差异。

42 第一个使用 x86-64 的 Intel 处理器是 Xeon，之后还有许多其他的处理器，包括 Core i5

和 Core i7。AMD 处理器中使用 x86-64 的例子有 Opteron 和 Athlon 64。

另一个为人所知的 64 位 Intel 架构是 IA-64，后来被称为 Itanium。IA-64 指令集与 x86 和 x86-64 完全不同，Itanium 处理器通常用于高性能数据库和网络服务器。

2.3.1 64 位操作模式

Intel 64 架构引入了一个新模式，称为 IA-32e。从技术上看，这个模式包含两个子模式：兼容模式（compatibility mode）和 64 位模式（64-bit mode）。不过它们常常被看做是模式而不是子模式，因此，先来了解这两个模式。

1. 兼容模式

在兼容模式下，现有的 16 位与 32 位应用程序通常不用进行重新编译就可以运行。但是，16 位 Windows（Win16）和 DOS 应用程序不能运行在 64 位 Microsoft Windows 下。与早期 Windows 版本不同，64 位 Windows 没有虚拟 DOS 机器子系统来利用处理器的功能切换到虚拟 8086 模式。

2. 64 位模式

在 64 位模式下，处理器执行的是使用 64 位线性地址空间的应用程序。这是 64 位 Microsoft Windows 的原生模式，该模式能使用 64 位指令操作数。

2.3.2 基本 64 位执行环境

64 位模式下，虽然处理器现在只能支持 48 位的地址，但是理论上，地址最大为 64 位。从寄存器来看，64 位模式与 32 位最主要的区别如下所示：

- 16 个 64 位通用寄存器（32 位模式只有 8 个通用寄存器）
- 8 个 80 位浮点寄存器
- 1 个 64 位状态标志寄存器 RFLAGS（只使用低 32 位）
- 1 个 64 位指令指针寄存器 RIP

回顾前文，32 位标志寄存器和指令指针寄存器分别称为 EFLAGS 和 EIP。此外，还有一些在讨论 x86 处理器时提过的，用于多媒体处理的特殊寄存器：

- 8 个 64 位 MMX 寄存器
- 16 个 128 位 XMM 寄存器（32 位模式只有 8 个 XMM 寄存器）

通用寄存器

在描述 32 位处理器时介绍过通用寄存器，它们是算术运算、数据传输和循环遍历数据指令的基本操作数。通用寄存器可以访问 8 位、16 位、32 位或 64 位操作数（需使用特殊前缀）。

64 位模式下，操作数的默认大小是 32 位，并且有 8 个通用寄存器。但是，给每条指令加上 REX（寄存器扩展）前缀后，操作数可以达到 64 位，可用通用寄存器的数量也增加到 16 个：32 位模式下的寄存器，再加上 8 个有标号的寄存器，R8 到 R15。表 2-1 给出了 REX 前缀下可用的寄存器。 43

表 2-1 使用 REX 前缀后，64 位模式的操作数大小

操作数大小	可用寄存器
8 位	AL、BL、CL、DL、DIL、SIL、BPL、SPL、R8L、R9L、R10L、R11L、R12L、R13L、R14L、R15L
16 位	AX、BX、CX、DX、DI、SI、BP、SP、R8W、R9W、R10W、R11W、R12W、R13W、R14W、R15W

（续）

操作数大小	可用寄存器
32 位	EAX、EBX、ECX、EDX、EDI、ESI、EBP、ESP、R8D、R9D、R10D、R11D、R12D、R13D、R14D、R15D
64 位	RAX、RBX、RCX、RDX、RDI、RSI、RBP、RSP、R8、R9、R10、R11、R12、R13、R14、R15

还有一些需要记住的细节：

- 64 位模式下，单条指令不能同时访问寄存器高字节，如 AH、BH、CH 和 DH，以及新字节寄存器的低字节（如 DIL）。
- 64 位模式下，32 位 EFLAGS 寄存器由 64 位 RFLAGS 寄存器取代。这两个寄存器共享低 32 位，而 RFLAGS 的高 32 位是不使用的。
- 32 位模式和 64 位模式具有相同的状态标志。

2.4 典型 x86 计算机组件

本节首先通过检查典型主板配置以及围绕 CPU 的芯片组来了解 x86 如何与其他组件的集成。然后讨论内存、I/O 端口和通用设备接口。最后说明汇编语言程序怎样利用系统硬件、固件，并调用操作系统函数来实现不同访问层次的 I/O 操作。

2.4.1 主板

主板是微型计算机的心脏，它是一个平面电路板，其上集成了 CPU、支持处理器（芯片组（chipset））、主存、输入输出接口、电源接口和扩展插槽。各种组件通过总线即一组直接蚀刻在主板上的导线，进行互连。目前 PC 市场上有几十种主板，它们在扩展功能、集成部件和速度方面存在着差异。但是，下述组件一般都会出现在主板上：

- CPU 插座。根据其支持的处理器类型，插座具有不同的形状和尺寸。
- 存储器插槽（SIMM 或 DIMM），用于直接插入小型内存条。
- BIOS（基本输入输出系统，basic input-output system）计算机芯片，保存系统软件。
- CMOS RAM，用一个小型纽扣电池为其持续供电。
- 大容量插槽设备接口，如硬盘和 CD-ROMS。
- 外部设备的 USB 接口。

44

- 键盘和鼠标接口。
- PCI 总线接口，用于声卡、显卡、数据采集卡和其他输入输出设备。

以下是可选组件：

- 集成声音处理器。
- 并行和串行设备接口。
- 集成网卡。
- 用于高速显卡的 AGP 总线接口。

典型系统中还有一些重要的支持处理器：

- 浮点单元（FPU），处理浮点数和扩展整数运算。
- 8284/82C84 时钟发生器，简称时钟，按照恒定速率振荡。时钟发生器同步 CPU 和计算机的其他部分。

- 8259A 可编程中断控制器（PIC，Programmable Interrupt Controller），处理来自硬件设备的外部中断请求，包括键盘、系统时钟和磁盘驱动器。这些设备能中断 CPU，并使其立即响应它们的请求。
- 8253 可编程间隔定时器 / 计数器（Programmable Interval Timer/Counter），每秒中断系统 18.2 次，更新系统日期和时钟，并控制扬声器。它还负责不断刷新内存，因为 RAM 存储器芯片保持其内容的时间只有几毫秒。
- 8255 可编程并行端口（Programmable Parallel Port），使用 IEEE 并行端口将数据输入和输出计算机。该端口通常用于打印机，但是也可以用于其他输入输出设备。

1. PCI 和 PCI Express 总线架构

PCI（外部设备互联，Peripheral Component Interconnect）总线为 CPU 和其他系统设备提供了连接桥，这些设备包括硬盘驱动器、内存、显卡、声卡和网卡。最近，PCI Express 总线在设备、内存和处理器之间提供了双向串行连接。如同网络一样，它用独立的“通道”传送数据包。该总线得到显卡的广泛支持，能以较高速度传输数据。

2. 主板芯片组

主板芯片组（motherboard chipset）是一组处理器芯片的集合，这些芯片被设计为在特定类型主板上一起工作。各种芯片组具有增强处理能力、多媒体功能或减少功耗等特性。以 Intel P965 Express 芯片组为例，该芯片组与 Intel Core2 Duo 或 Pentium D 处理器一起，用于桌面系统。Intel P965 具有下述特性：

- Intel 高速内存访问（Fast Memory Access）使用了最新内存控制中心（MCH）。它可以 800MHz 时钟速度来访问双通道 DDR2 存储器。
- I/O 控制中心（Intel ICH8/R/DH）使用 Intel 矩阵存储技术（MST）来支持多个串行 ATA 设备（磁盘驱动器）。
- 支持多个 USB 端口，多个 PCI Express 插槽，联网和 Intel 静音系统技术。
- 高清晰音频芯片提供了数字声音功能。

45

如图 2-6 所示，主板厂商以特定芯片为中心来制造产品。例如，Asus 公司使用 P965 芯片组的 P5B-E P965 主板。

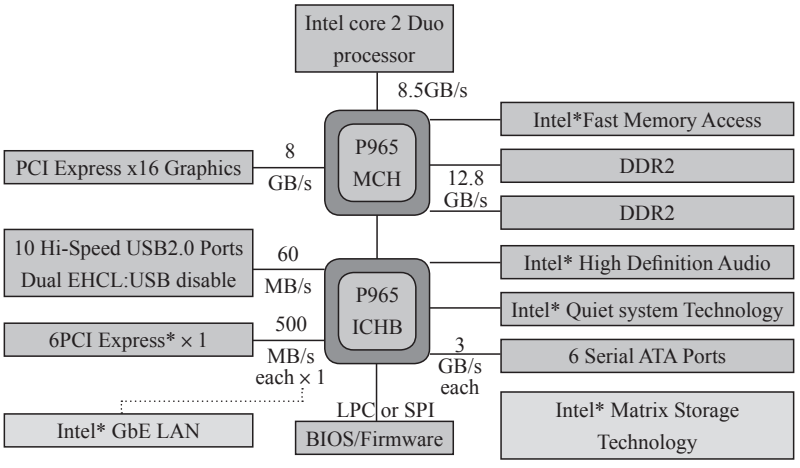


图 2-6　Intel 965 Express 芯片组框图

来源：Intel P965 Express 芯片组（产品简介），Intel 公司版权所有，已获使用权。

2.4.2 内存

基于 Intel 的系统使用的是几种基础类型内存：只读存储器（ROM）、可擦除可编程只读存储器（EPROM）、动态随机访问存储器（DRAM）、静态 RAM（SRAM）、图像随机存储器（VRAM），和互补金属氧化物半导体（CMOS）RAM：

- ROM 永久烧录在芯片上，并且不能擦除。
- EPROM 能用紫外线缓慢擦除，并且重新编程。
- DRAM，即通常的内存，在程序运行时保存程序和数据的部件。该部件价格便宜，但是每毫秒需要进行刷新，以避免丢失其内容。有些系统使用的是 ECC（错误检查和纠正）存储器。
- SRAM 主要用于价格高、速度快的 cache 存储器。它不需要刷新，CPU 的 cache 存储器就是由 SRAM 构成的。
- VRAM 保存视频数据。VRAM 是双端口的，它允许一个端口持续刷新显示器，同时另一个端口将数据写到显示器。
- CMOS RAM 在系统主板上，保存系统设置信息。它由电池供电，因此当计算机电源关闭后，CMOS RAM 中的内容仍能保留。

2.4.3 本节回顾

1. 描述 SRAM 及其最常见的用途。
46 2. 描述 VRAM。
3. 列出 Intel P965 Express 芯片组至少两个特性。
4. 写出本章描述的 4 类 RAM。
5. 8259A PIC 控制器的用途是什么？

2.5 输入输出系统

> **提示** 由于计算机游戏与内存和 I/O 有着非常密切的关系，因此，它们推动计算机达到其最大性能。善于游戏编程的程序员通常很了解视频和音频硬件，并会优化代码的硬件特性。

2.5.1 I/O 访问层次

应用程序通常从键盘和磁盘文件读取输入，而将输出写到显示器和文件中。完成 I/O 不需要直接访问硬件——相反，可以调用操作系统的函数。与第 1 章中描述的虚拟机相似，I/O 也有不同的访问层次，主要有以下三个：

- 高级语言函数：高级编程语言，如 C++ 或 Java，包含了执行输入输出的函数。由于这些函数要在各种不同的计算机系统中工作，并不依赖于任何一个操作系统，因此，这些函数具有可移植性。
- 操作系统：程序员能够从被称为 API（应用程序编程接口，Application Programming Interface）的库中调用操作系统函数。操作系统提供高级操作，比如，向文件写入字符串，从键盘读取字符串，和分配内存块。
- BIOS：基本输入输出系统是一组能够直接与硬件设备通信的低级子程序集合。BIOS

由计算机制造商安装并定制，以适应机器硬件。操作系统通常与 BIOS 通信。

设备驱动程序 设备驱动程序允许操作系统与硬件设备和系统 BIOS 直接通信。例如，设备驱动程序可能接收来自 OS 的请求来读取一些数据，而满足该请求的方法是，通过执行设备固件中的代码，用设备特有的方式来读取数据。设备驱动程序有两种安装方法：（1）在特定硬件设备连接到系统之前，或者（2）设备已连接并且识别之后。对于后一种方法，OS 识别设备名称和签名，然后在计算机上定位并安装设备驱动软件。

现在，通过展示应用程序在屏幕上显示字符串的过程，来了解 I/O 层次结构（图 2-7）。该过程包含以下步骤：

1）应用程序调用 HLL 库函数，将字符串写入标准输出。

2）库函数（第 3 层）调用操作系统函数，传递一个字符串指针。

3）操作系统函数（第 2 层）用循环的方法调用 BIOS 子程序，向其传递每个字符的 ASCII 码和颜色。操作系统调用另一个 BIOS 子程序，将光标移动到屏幕的下一个位置上。

4）BIOS 子程序（第 1 层）接收一个字符，将其映射到一个特定的系统字体，并把该字符发送到与视频控制卡相连的硬件端口。

5）视频控制卡（第 0 层）为视频显示产生定时硬件信号，来控制光栅扫描并显示像素。

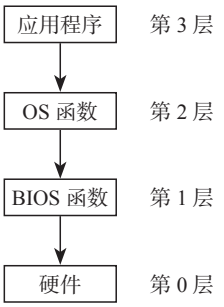


图 2-7 输入输出操作的访问层次

47

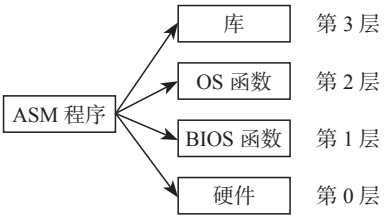


图 2-8 汇编语言访问层次

多层次编程 汇编语言程序在输入输出编程领域有着强大的能力和灵活性。它们可以从以下访问层次进行选择（图 2-8）：

- 第 3 层：调用库函数来执行通用文本 I/O 和基于文件的 I/O。例如，本书也提供了一个这样的库。
- 第 2 层：调用操作系统函数来执行通用文本 I/O 和基于文件的 I/O。如果 OS 使用了图形用户界面，它就能用与设备无关的方式来显示图形。
- 第 1 层：调用 BIOS 函数来控制设备具体特性，如颜色、图形、声音、键盘输入和底层磁盘 I/O。
- 第 0 层：从硬件端口发送和接收数据，对特定设备拥有绝对控制权。这个方式没有广泛用于各种硬件设备，因此不具可移植性。不同设备通常使用不同硬件端口，因此，程序代码必须根据每个设备的特定类型来进行定制。

如何进行权衡？控制与可移植性是最重要的。第 2 层（OS）工作在任何一个运行同样操作系统的计算机上。如果 I/O 设备缺少某些功能，那么 OS 将尽可能接近预期结果。第 2 层速度并不特别快，因为每个 I/O 调用在执行前，都必须经过好几个层次。

48

第 1 层（BIOS）在具有标准 BIOS 的所有系统上工作，但是在这些系统上不会产生同样的结果。例如，两台计算机可能会有不同分辨率的视频显示功能。在第 1 层上的程序员需要编写代码来检测用户的硬件设置，并调整输出格式来与之匹配。第 1 层的速度比第 2 层快，因为它与硬件之间只隔了一个层次。

第 0 层（硬件）与通用设备一起工作，如串行端口；或是与由知名厂商生产的特殊 I/O 设备一起工作。这个层次上的程序必须扩展它们的编码逻辑来处理 I/O 设备的变化。实模式的游戏程序就是最好的例子，因为它们常常需要取得计算机的控制权。第 0 层的程序执行速度与硬件一样快。

举个例子，假设要用音频控制设备来播放一个 WAV 文件。在 OS 层上，不需要了解已安装设备的类型，也不用关心设备卡的非标准特性。在 BIOS 层上，要查询声卡（通过其已安装的设备驱动软件），找出它是否属于某一类具有已知功能的声卡。在硬件层上，需要对特定模式声卡的程序进行微调，以利用每个声卡的特性。

通用操作系统极少允许应用程序直接访问系统硬件，因为这样做会使得它几乎无法同时运行多个程序。相反，硬件只能由设备驱动程序按照严格控制的方式进行访问。另一方面，专业设备的小型操作系统则常常直接与硬件相连。这样做是为了减少操作系统代码占用的内存空间，并且这些操作系统几乎总是一次只运行单个程序。Microsoft 最后一个允许程序直接访问硬件的操作系统是 MS-DOS，它一次只能运行一个程序。

2.5.2 本节回顾

1. 计算机系统中有 4 个输入输出层次，哪一个最具有通用性和可移植性？
2. 区分 BIOS 层输入输出的特征是什么？
3. 考虑到 BIOS 中已经有了与计算机硬件通信的代码，为什么设备驱动程序是必需的？

49

4. 在显示字符串的例子中，操作系统与视频控制卡之间存在的是哪个 I/O 层次？
5. 运行 MS-Windows 和运行 Linux 的计算机的 BIOS 可能存在不同吗？

2.6 本章小结

中央处理单元（CPU）处理算术和逻辑运算。它包含了有限数量的存储位置，即*寄存器*，一个高频时钟用于同步其操作，一个控制单元和一个算术逻辑单元。内存存储单元在计算机程序运行时，保存指令和数据。总线是一组并行线路，在计算机不同部件之间传输数据。

一条机器指令的执行可以分为一系列独立的操作，称为*指令执行周期*。3 个主要操作分别为取值、译码和执行。指令周期中的每一步都至少要花费一个系统时钟单位，即*时钟周期*。加载和执行过程描述了程序如何被操作系统定位，加载入内存，再由操作系统执行。

x86 处理器系列有三种基本操作模式：*保护模式*、*实地址模式*和*系统管理模式*。此外，还有一个*虚拟 8086 模式*是保护模式的一个特例。Intel 64 处理器系列有两种基本操作模式：*兼容模式*和 *64 位模式*。在兼容模式下处理器可以运行 16 位和 32 位应用程序。

寄存器为 CPU 内的存储位置进行命名，其访问速度比常规内存要快很多。以下是对寄存器的简要说明：

- *通用寄存器*主要用于算术运算、数据传输和逻辑操作。
- *段寄存器*存放预先分配的内存区域的基址，这些内存区域就是段。
- *指令指针寄存器*存放的是下一条要执行指令的地址。
- *标志寄存器*包含的独立二进制位用于控制 CPU 的操作，并反映 ALU 操作的结果。

x86 有一个浮点单元（FPU）专门用于高速浮点指令的执行。

微型计算机的心脏是它的主板，主板上有 CPU、支持处理器、主存、输入输出接口、

电源接口和扩展插槽。PCI（外部设备互联）总线为 Pentium 处理器提供了方便的升级途径。大多数主板集成了若干微处理器和控制器，称为芯片组。芯片组在很大程度上决定了计算机的性能。

PC 中使用的几种基本存储器为：ROM、EPROM、动态 RAM（DRAM）、静态 RAM（SRAM）、视频 RAM（VRAM）和 CMOS RAM。

与虚拟机概念相似，输入输出是通过不同层次的访问来实现的。库函数在最高层，操作系统是次高层。BIOS（基本输入输出系统）是一组函数，能直接与硬件设备通信。程序也可以直接访问输入输出设备。 50

2.7 关键术语

32-bit mode（32 位模式）
64-bit mode（64 位模式）
address bus（地址总线）
application programming interface(API)（应用程序接口）
arithmetic logic unit(ALU)（算术逻辑单元）
auxiliary carry flag（辅助进位标志位）
basic program execution registers（基本程序执行寄存器）
BIOS(basic input-out put system)（基本输入输出系统）
bus（总线）
cache（高速缓存）
carry flag（进位标志位）
central processor unit(CPU)（中央处理单元）
clock（时钟）
clock cycle（时钟周期）
clock generator（时钟发生器）
code cache（代码 cache）
control flags（控制标志位）
control unit（控制单元）
data bus（数据总线）
data cache（数据 cache）
device drivers（设备驱动程序）
direction flag（方向标志位）
bynamic RAM（动态 RAM）
EFLAGS register（EFLAGS 寄存器）
extended destination index（扩展目的变址）
extended physical addressing（扩展物理寻址）
extended source index（扩展源变址）
extended stack pointer（扩展堆栈指针）
fetch-decode-execute（取值 – 译码 – 执行）
flags register（标志寄存器）
floating-poing unit（浮点单元）
general-purpose registers（通用寄存器）
instruction decoder（指令译码器）
instruction excution cycle（指令执行周期）
instruction queue（指令队列）
instruction pointer（指令指针）
interrupt flag（中断标志位）
Level-1 cache（1 级 cache）
Level-2 cache（2 级 cache）
machine cycle（机器周期）
memory storage unit（内存存储单元）
MMX registers（MMX 寄存器）
motherboard（主板）
motherboard chipset（主板芯片组）
operating system(OS)（操作系统）
overflow flag（溢出标志位）
parity flag（奇偶标志位）
PCI (peripheral component interconnect)（外部设备互联）
PCI express
process（过程）
process ID（过程 ID）
programmable interrupt controller(PIC)（可编程中断控制器）
programmable interval interval timer/counter（可编程间隔定时器 / 计数器）
programmable parallel port（可编程并行端口）
protected mode（保护模式）
random access memory (RAM)（随机访问存储器）
read-only memory (ROM)（只读存储器）
real-address mode（实地址模式）

registers（寄存器）
segment registers（段寄存器）
sign flag（符号标志位）
single-instruction, multiple-data(SIMD)（单指令多数据）
static RSM（静态 RAM）
51 status flags（状态标志位）
system management mode(SMM)（系统管理模式）
Task Manager（任务管理器）
virtual-8086 mode（虚拟 8086 模式）
wait states（等待状态）
XMM registers（XMM 寄存器）
zero flag（零标志位）

2.8 复习题

1. 32 位模式下，除了堆栈指针（ESP）寄存器，还有哪些寄存器指向堆栈的参数？
2. 说出至少 4 个 CPU 状态标志位。
3. 当无符号数算术运算结果超过目标位置大小时，应设置哪个标志位？
4. 当有符号数算术运算结果对目标位置而言太大或太小时，应设置哪个标志位？
5.（真 / 假）：寄存器操作数为 32 位，使用 REX 前缀，则程序可以使用 R8D 寄存器。
6. 算术或逻辑运算产生负数结果时，应设置哪个标志位？
7. CPU 的哪个部件执行浮点算术运算？
8. 32 位处理器中，浮点数据寄存器包含多少位？
9.（真 / 假）：x86-64 指令集向后兼容 x86 指令集。
10.（真 / 假）：当前 64 位芯片实现方式下，所有 64 位都可以用于寻址。
11.（真 / 假）：Itanium 指令集与 x86 完全不同。
12.（真 / 假）：静态 RAM 一般比动态 RAM 便宜。
13.（真 / 假）：加上 REX 前缀就可以使用 64 位 RDI 寄存器。
14.（真 / 假）：在原生 64 位模式下，可以使用 16 位实模式，但是不能使用虚拟 8086 模式。
15.（真 / 假）：x86-64 处理器比 x86 处理器多 4 个通用寄存器。
16.（真 / 假）：64 位的 Microsoft Windows 不支持虚拟 8086 模式。
17.（真 / 假）：DRAM 只能用紫外线擦除。
18.（真 / 假）：64 位模式下，可以使用的浮点寄存器多达 8 个。
19.（真 / 假）：总线是两端连接在主板上的塑料电缆，但没有直接位于主板上。
20.（真 / 假）：CMOS RAM 与静态 RAM 相同，也就是说，不需要额外的电源和刷新周期就可以保持它的内容。
21.（真 / 假）：PCI 接口用于显卡和声卡。
22.（真 / 假）：8259A 是一种控制器，用于处理来自硬件设备的外部中断。
23.（真 / 假）：PCI 是可编程组件接口（programmable component interface）的缩写。
24.（真 / 假）：VRAM 代表虚拟随机访问存储器。
25. 汇编语言程序在哪个（或哪些）层次上可以控制输入输出？
52 26. 为什么游戏程序常常将声音输出直接发送到声卡的硬件端口？