第一篇　单发射结构 RISC

第一章　微计算机技术发展综述

在讨论微计算机技术的新进展时，应该说明，现在微计算机与工作站、小型机以至大型机之间的界线已愈来愈模糊了。微处理器芯片的速度已经达到甚至超过了10年前一般大型机的中央处理器的速度；超级微机系统已经采用了小型机和大型机的体系结构，其图形功能有时也可达到1兆象素分辨率。然而，本文仍按传统的看法，以单个VLSI芯片为CPU，并以这种CPU组成的系统称为微计算机。下面仅就微计算机的微处理器芯片、微计算机系统的体系结构、微计算机系统的总线以及微计算机的操作系统等几个主要方面，来讨论近年的进展及其未来发展方向。从中可以看到先进的微处理器体系结构RISC的历史沿革和意义。

1. 微处理器芯片

自从80年代后期，RISC芯片从实验室进入工业产品市场以来，微处理器芯片的速度获得极大的提高。当微处理器沿着CISC道路发展的时候，从1979年最快的1MIPS M68000微处理器，到1987年大多数工作站都采用的速度为2MIPS的M68020，花了8年时间，速度只翻了一倍。那时INTEL 80386刚问世，速度也不过(3～4)MIPS。可是，自RISC问世以来，到1992年已经出现了DEC ALPHA为CPU的芯片，其浮点速度可达200MFLOPS。不过5年间，将微处理器芯片速度提高了50倍。据称，到了2000年，微处理器芯片可达到4000MIPS，INTEL现在已经在着手研制2000MIPS芯片。因此可以认为，RISC是20多年来计算机体系结构中一项革命性成就，它产生的影响是深刻而巨大的。由图1.1可见，RISC技术使80年代中后期的微处理器速度飞速提高，目前已经赶上了大型机中央处理器的速度。100MIPS的微型机即将以台式机型式出现在用户面前，它使微机的应用领域更上了一层台阶。意义更深刻的是：由于RISC芯片速度已经赶上大型

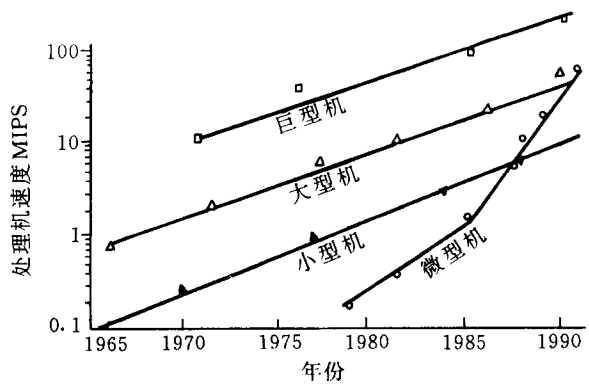


图1.1　处理机速度增长情况

机，这样就创造了一个条件，即可以把微型机、工作站、小型机和大型机建立在一个二进制兼容的共同平台 RISC/UNIX 之上，依靠不同的技术，实现不同的机型和用途。例如，加强图形功能，可成为工作站；采用双机或多机系统和扩大内存，并增强 I/O 通道能力和外存容量，即可发展成小型机。如再采用并行处理技术，即可达到目前大型机的功能。这样将使微、小、大、巨型机的界线不象传统观念那么清楚，同时会刺激计算机体系结构设计技术，如并行处理技术和图形技术向更精细和更高效能的方向提高。同时，二进制兼容的平台使软件开发更为有效，而且使微、小、大、巨型机连网和共享资源更为方便。

在 RISC 芯片不断发展的同时，CISC 芯片也取得长足的进步。50MHz 的 80486 已经投入生产，其速度也可达到 20MIPS。当然，它所要求的片上晶体管数相对较多。MOTOROLA 的 0.8μm 工艺的 68040 可达 40MHz 工作频率，晶体管数为 120 万个；0.65μm 工艺的 M68050 即将推出，其性能为 68040 的二倍，0.5μm 工艺微处理机芯片也在试制中。

CISC 芯片之所以还在不断发展，是由于要保持软件上的兼容性。可以认为：80 年代微机的不断发展使计算机成为一个大规模生产的产业，这个规模经济产业（硬件和软件产业）的基础是建立一个标准化的平台。这个平台就是 80x86/MSDOS，该平台上的规模性经济可以用以下数据（1991 年数据）来说明：

80x86 的 PC 机生产达　7000 万台

MSDOS 操作系统达　7000 万份

80x86 系统投资达　3400 亿美元

MSDOS 上应用软件达 2 亿件

可以相信，具有如此巨大经济实力支持的 80x86/MSDOS 在未来的年代中还会具有很强大的生命力。CISC 还会在发展中采用 RISC 思想，技术将要不断更新。操作系统 MSDOS 会向 Window 方向发展，CISC 80x86 会逐渐靠近 RISC 的体系结构。大家知道，精简指令系统计算机 RISC 的设计概念的精华不在于减少其指令系统中的指令数目，而是在于减少平均每条指令执行所需的周期数 CPI 值。从 8088/8087 到 80386/80387，再发展到 80486，其中每条指令执行所需的周期数也在不断减少。现在，INTEL 的 80586（现已正式定名为 Pentium）已经采用了第三代 RISC 体系结构中的超标量结构，即处理机中有多个执行部件，其指令执行并行度更高了。

可以看出，从体系结构概念来看，CISC 与 RISC 是殊途同归，都在向减少 CPI 值、向超标量与超流水线结构方向发展。然而，RISC 具有结构简单的优点，因此步伐更快一些，速度提高更快一些。INTEL 的 RISC 芯片 80860 和 80960 都已采用了超标量结构。前者用于通用机，后者适于嵌入式用途。80960 速度可达 60MIPS，但每片只有 20 美元，其性能价格比是相当高的。MOTOROLA 88100/88110、新型的 SPARC 以及 IBM 的 POWER 6000 都采用了超标量结构。POWER 6000 除了采用分开的定点运算和浮点运算多个执行部件结构外，还设置了一个转移处理部件，它把编译优化中的指令调度的部分工作用硬件来实现，做到动态的指令优化调度，使指令执行的并行度大为提高。各方资料表明，IBM 将把 POWER 6000 作为它研究与开发工作的重点。各方报道表明，POWER 6000 处理系列将在 IBM 未来各档产品系统中起到极其重要的作用。

目前采用超流水线结构方案的RISC芯片只有MIPS公司的R4000。它共有四级流水线，在提高芯片主频率的情况下，每级流水线节拍又可细分为两个小级节拍，每隔一个小级节拍取出一条指令。如果每个主周期细分的小节拍数愈多，则超流水线结构的CPI值愈小，但实际上的工作频率相应地也愈高。因此，可以认为超标量结构与超流水线结构为提高指令执行并行度的概念有所不同。前者是靠空间来换取时间，后者是以时间来换取空间。这两个方向的结构合并，可以形成超流水线的超标量结构。

计算机界普遍关心的问题是：VLSI微处理器的速度提高有没有极限？随之而来的问题是：VLSI工艺线宽会减少到什么程度？单片芯片上的晶体管数会增加到什么程度？制造快速处理器的技术发展趋势是什么？目前，象Pentium这种高速RISC的VLSI工艺是采用(0.7～0.8)μm工艺，晶体管集成度(单片)可达300万个，工作主频为50MHz。HP公司即将推出的(PA)RISC将用CMOS工艺做到90MHz(五级流水线)，这已经和ECL工艺相当。但是，要进一步提高单片上晶体管集成度，就要增加芯片面积，这将引起成品率下降，并使价格随之迅速上升。现在国外快速微处理器制造技术的一个重要趋势是采用低电压逻辑和多芯片模块(MCM)。降低电源电压可以减少电流对寄生电容的充电时间，减少逻辑电路延迟。如IBM RS/6000 41MHz的550型中采用的芯片，采用0.5μm工艺，3.6V逻辑。有的公司芯片内部逻辑用5V电压，但芯片与外界接口的输入/输出信号采用1V电平，因为这些信号线上的寄生电容比芯片内部大得多。MCM指的是把多个单芯片连接起来组成处理器，但这些多个单芯片是封装在一个组件中。如IBM的POWER 6000 RISC处理器共有700万晶体管，它分成9个单芯片(其中450万晶体管是用于cache的)，这样单芯片的集成度不必太高，从而可降低价格。但随之而产生的问题是芯片之间引线的张力、湿度问题和热敏问题。此外，芯片之间的引线的寄存电容和电感都要设法减少。引线长度减少6倍，寄生电感可减少60倍，寄生电容可减少6倍，临界数据路径的延迟可减少3倍。目前，提高单片芯片本身集成度仍是发展趋势。亚微米的工艺还在向0.5μm过渡。INTEL计划做5000万到1亿个晶体管的CPU，它的Micro 2000体系结构包括2000 MIPS的微处理器，并与80386以及MSDOS/OS 2/UNIX结构二进制兼容。其它工艺如ECL和BiMOS也在进展中。但总的来说，CMOS工艺比较适合于晶体管数较多的多个执行部件的超标量结构，而ECL与BiMOS工艺比较适合于要求主频较高但晶体管数较少的超流水线结构。

2. 微机系统体系结构

以上所述的微处理器(无论是CISC还是RISC)速度的迅速提高，使它与存储器DRAM的速度差距愈拉愈大(见图1.2)。这一趋势对微机体系结构产生深刻影响，使微机系统体系结构中的cache显得更为重要。解决问题的办法是在CPU芯片上添设片上cache。但如上所述，片上cache容量增大，芯片面积将增加很多，从而导致成品率下降和成本上升，因此目前片上cache容量最多在8K字节左右，这显然是不够的。因此，在高档微机系统中都必须增添片外的大容量cache，如64K，128K以至256K字节。这种片上和片外多层次cache是目前微机体系结构中通常遇到的问题。此外，高速CPU与高速cache的配合，又要求有高速的总线来传输数据，总线的频宽要求为160MB/s，甚至更高。这是

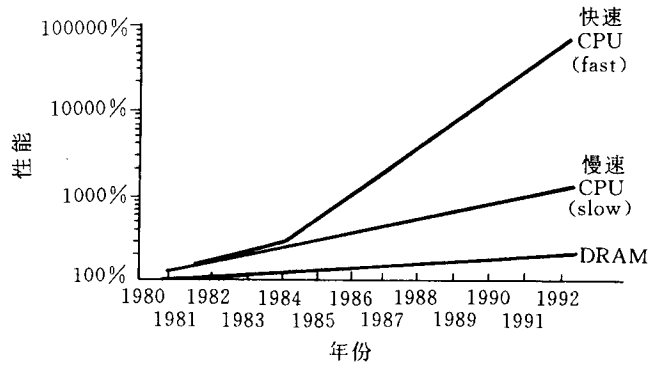


图 1.2　CPU 与 DRAM 的速度差距

目前标准的系统总线所远远不能满足的。因此现代高档微机系统都另设内部总线，这内部总线有时具有两个层次，最高速度的内部总线提供高速 CPU 与 cache 之间传输通路，次高速度的内部总线为存储管理部件与大容量 DRAM 提供传输通路。另外有两种与外围设备相连的外部总线，一种较高速的外部总线与图形卡、DMA、局部网络卡以及 SCSI 相接，另一种 8 位的较低速的外部总线联接软磁盘、键盘、串行接口、ISDN 以及音频端口。这种多层次总线结构见图 1.3。这一结构随着 CPU 速度继续提高以及多媒体环境的外围设备多样化而显得更加重要。

近年来很多有名的微机公司推出多机微机系统，一般采用四个到八个微处理器，使用各种版本的多机 UNIX 操作系统。整个系统的功能可以达到 50MIPS，甚至 100MIPS 以上，其应用领域已经渗透到传统超级小型机的领域。这种系统的体系结构如图 1.4 所示。

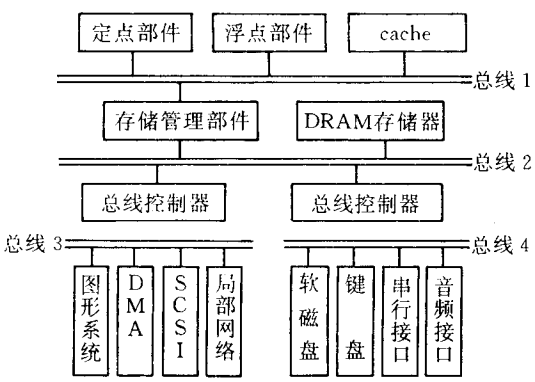
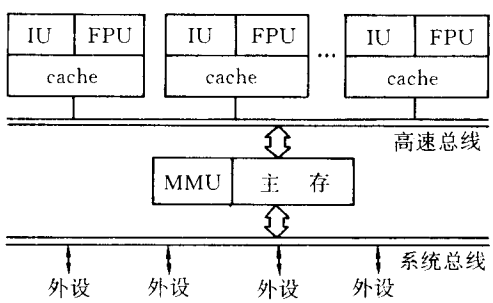


图 1.3　多层次总线体系结构



IU：整数部件　　FPU：浮点部件

cache：高速缓存　MMU：存储管理

图 1.4　多机微机系统结构框图

多机微机系统的发展潜力很大。目前一般采用 80386 CPU 或 80486 CPU，处理器和 cache 做在一块板上，cache 容量为 64KB 或 128KB。如果把 CPU 改成(40～50)MIPS 的 RISC，那么这个多机系统的功能将超过一般大型机。而且这种系统的扩展性很好，如果增

加其中的 CPU 板，选择高性能的高速总线，则系统的性能还可以升级。因此，这种系统结构很值得我国计算机界重视。其中比较关键的技术是 cache 的一致性、多机系统的处理器之间通讯协议、高速总线的结构以及多机 UNIX 系统。

3. 总线

如前所述，微机 CPU 速度的迅速提高，使高速总线和层次式总线结构设计的重要性更加突出了，这些总线结构也是国际上各大计算机公司正在努力解决的问题。过去的 AT 总线、MULTIBUS Ⅰ、VME 总线、以至 EISA 与 MCA 总线的速度都已经不够了。其出路是设计新型号总线或者另外设计特定的总线。

目前国际上已采用或正在推出的总线标准及其应用范围见图 1.5。其中 AT 总线、EISA、MCA 低档型号主要用于微机，MULTIBUS Ⅰ与Ⅱ主要用于工业控制用微机，VME 则多用于工作站。EISA、MULTIBUS Ⅱ和 VME 64 也可用在多机微机系统。高档 MCA 型号和 VME 等总线将用于未来的更高速的微机系统。

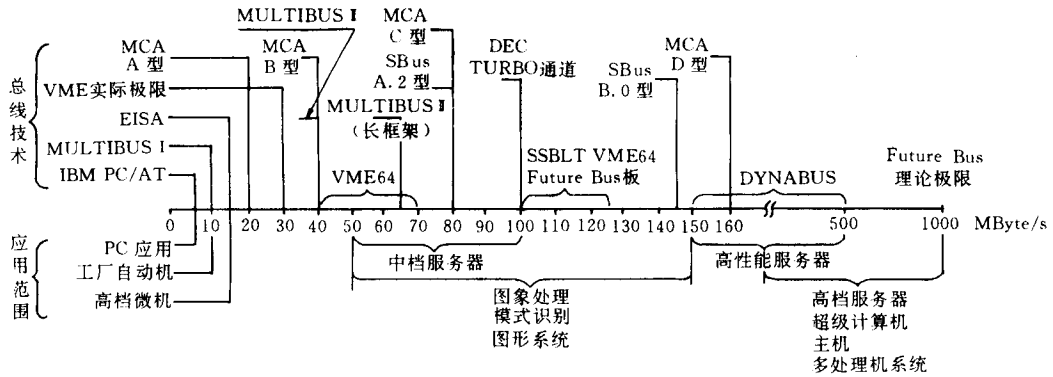


图 1.5　系统总线技术与应用范围

总线设计的新技术要考虑到 cache 一致性和信息传递协议，目前的新总线要更有效地支持多机系统，因此在这两方面做了精细的改进。此外，为了加速数据传送，除了采用成块传送以外，还有突发传送(burst transfer)功能。另外高速总线还要考虑到传输延迟、接地、干扰屏蔽等很多工程性问题。总之，在未来新微机系统中，总线设计已成为极其重要的问题，总线标准也是世界各大计算机公司激烈竞争的一个焦点。谁能掌握处理器标准、总线标准和操作系统标准，谁就能在未来 10 年中微机、工作站、小型机以至大型机各领域占统治地位。

4. 操作系统

新一代的微机操作系统要考虑的关键问题包括：

- 支持 Window 功能
- 支持多线索调度
- 支持多处理机功能

· 支持面向对象技术

未来的操作系统则将对多媒体技术提供全面支持。

下面介绍 1991 年与 1992 年推出的 32 位微机中采用的几种操作系统及性能。

(1) Microsoft 的 Windows NT(New Technology)

· 多线索

· 多任务

· 对称式多处理机

· 支持多种应用程序接口 API

· 支持多文件系统

(2) IBM 的 OS/2 2.0*

· 多线索

· 多任务

· 支持多文件系统

· 支持多种应用程序接口 API

· 表示管理功能

(3) SOLARIES (SUN 与 INTEL 联合开发)。

· 开放式窗口 GU I

· 开放式网络计算

· 面向对象的开发工具

· 支持对称式多处理机

· 多线索

(4) Mac System 7.0 (APPLE 与 IBM 联合开发)。

· 数据访问管理器

· 印刷和页面描述

· 进程之间通讯

· 多任务

· 虚拟存储

微机操作系统中的一个重要趋势是面向多厂家,即可容纳不同的操作系统的应用软件,可支持微机,也可支持工作站。此类操作系统称为跨平台(Cross Platform)操作系统。APPLE 正在开发的 PINK 操作系统是一个突出的典型。它可以在 MACINTOSH 上运行,也可以在其它机型上运行。它可以运行 MAC OS System 7.0,也可运行 OS/2 和 UNIX软件,还可在 IBM 的 RS6000 上运行,并支持多媒体功能。据称,这个 PINK 操作系统是促使 IBM 高级总裁做出与 APPLE 联盟决定的最有力的因素,它将大大加强 IBM 在软件方面的力量。

其它 Microsoft 的 NT 操作系统也在积极开发多媒体功能。

综上所述,从宏观来看,80 年代微计算机的发展,对于计算机事业以至全世界人类生活做出的最大贡献是建立了一个全世界都接受的标准化平台 80x86/MSDOS,从而使微计算机成为一项大规模经济的产业,使计算机应用深入到各个领域。

80年代后期微机史上发生了深刻的变化，即RISC诞生了。它对计算机硬件与软件技术以至对于计算机产业结构都会产生深刻的影响。因此，各大公司都在努力使自己的RISC产品成为90年代的标准。目前计算机界动荡、分化和重新结盟，其目的就是追求一个新的标准平台RISC/UNIX。这个平台上的投资和影响将比80年代的平台80x86/MS-DOS更大。可以肯定，未来的处理机芯片都将向RISC设计概念靠拢和发展；未来的操作系统也向UNIX概念靠拢和发展。

第二章　RISC 设计思想和原理

§2.1　计算机体系结构设计思想的演变

在 50 年代，计算机设计都是逐个进行的，每个计算机有它自己的指令系统，用户按照其汇编符号编写程序，当时还用到了一些程序设计自动化的技术。但是，那时还没有计算机产品系列化和软件兼容性的概念。

60 年代以后，程序设计自动化技术获得了发展，高级语言（如 FORTRAN、ALGOL）逐渐成熟。由于应用的需要，软件编写的工作量愈来愈大，这就要求用户花很多功夫编写好的软件以便在同一系列但不同档次上的计算机中都可运行。减轻软件编写工作量对计算机设计提出了新的要求。

1964 年 IBM 推出的 IBM SYSTEM 360 标志着计算机体系结构发展史上的一个重要里程碑。IBM SYSTEM 360 首次在计算机体系结构与计算机组织之间划出一道清楚的分界线：计算机体系结构是机器语言程序设计员为编写程序所必须看到的一个计算机的抽象结构。计算机组织是为实现该结构的硬件组成。

因此，对于某一个产品系列，其计算机体系结构可以是相同的，系列中每一个档次的产品的计算机组织却可以有差别。但在系列产品中，各个档次的产品在软件上具有兼容性。这样，对于不同要求的用户，可以提供不同层次、不同价格的计算机产品，但用户所用的软件却可以向上兼容，而且用户可以花较少的费用使自己的计算机向上升级，同时可保持用户投资的软件继续有效。当时，这一概念十分重要，它打破了 50 年代计算机的“手工业式”生产方式，使计算机硬件和软件都获得迅速发展，奠定了计算机发展的基础。这一概念至今仍有不可动摇的地位，仍是计算机体系结构设计者必须遵循的依据。

2.1.1　传统的计算机体系结构设计技术

60 年代中期，微程序是实现计算机系列产品的体系结构，并保证软件向上兼容的一项重要技术。在系列产品中，要求该系统中的计算机的指令系统有一个核，这个核是不变的，才能保持软件兼容。此外，在系列产品中，从低档产品到高档产品，性能逐步提高，往往要求指令系统也增加功能更强的一些指令，当然计算机组织的复杂程度也随之增加，价格也随之而增加，但这样可满足一些高要求的用户。当时，微程序技术是满足上述体系结构设计要求的有效而可行的技术。在微程序设计中，相应于上述的一个指令系统的核，微程序存储器的核心也是保持不变的；要扩展一些功能较强的指令，只要逐渐扩充微程序的存储器就可以了。因此，自 60 年代以来，几乎所有的大型机、小型机和微型机的控制部件都采用了微程序技术设计其控制指令执行的部件。

应该指出的是，微程序技术之所以风行，还有一个计算机工艺技术的背景。在 60 到 70 年代，计算机的主存储一直采用磁心体，而快速的微程序存储则采用半固定存储器，70

年代则开始采用小型半导体存储器。当时的CPU已采用很高速度的双极型半导体逻辑，它比磁心主存快(5～10)倍，而与小型快速微存储却相当。当时传统的计算机的机器周期一般相当于主存的周期，它正好等于(5～10)个微存储周期(见图2.1)。这与微程序技术中，一条指令是由(5～10)个微指令操作组成的情况是很匹配的。可见，微程序控制技术的发展是受到当时的计算机工艺技术支持的。

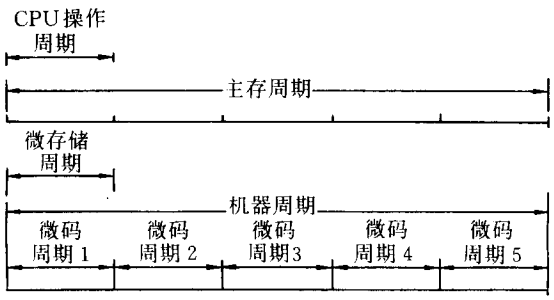


图2.1　机器周期与微码周期的匹配

此外，在60年代和70年代，计算机体系结构设计还受到一些传统思想的深刻影响，这些思想对于指令系统的选择起了很大的作用。这些传统思想认为指令系统愈丰富愈好，其理由是：

1. 指令系统愈丰富则功能愈强，而且编译程序愈好写。尤其是要增加存储器操作的指令。当时认为：在寄存器-寄存器操作指令、存储器-寄存器操作指令以及存储器-存储器操作指令等三种指令中，以第三种指令的效率最高。这种思想，在M68000微处理器设计中表现尤为突出，M68000采用了大量存储器-存储器操作指令。

我们用十分简单的例子说明：

$$B+C\rightarrow A$$

如采用寄存器-寄存器操作，要用四条指令才能完成，即

```
LOAD    B,rB
LOAD    C,rC
ADD     rB+rC→A
STORE   rA,A
```

如采用存储器-寄存器操作，则需用三条指令：

```
LOAD    B
ADD     C
STORE   A
```

但如采用存储器-存储器操作，则用一行指令就可完成，即

```
ADD     B+C→A
```

其中A、B和C是存储器单元。

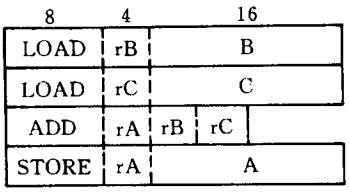
2. 指令系统愈丰富，愈可减轻软件危机。由于软件成本上升而硬件成本下降，计算机设计者想尽可能地把计算机执行高级语言的功能移到硬件上去，使机器语言很象高级语言的语句。这样可以缩小程序设计语言与机器语言之间的所谓“沟距”，从而使程序代码缩短，减少软件上耗费的愈来愈大的费用。

3. 指令系统丰富，尤其是操作步骤较多的存储器操作指令增多，可以改善体系结构的质量。60至70年代，衡量计算机体系结构质量的重要标准是“代码长度”与“存储效率”。当时存储器还采用磁心，存储容量较小，而半导体存储器也刚开始采用，价格也较贵。

程序愈小，则存储效率愈高。

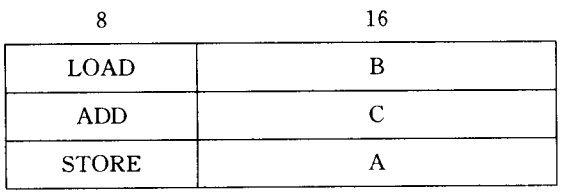
代码长度是指一段程序中指令和数据总共占用的存储总容量(以位"b"计)。图 2.2 表示三种类型操作指令所占用的存储总容量 M，M＝I＋D。其中 I 是指令占用位数，D 是数据占用位数。假定操作码为 8 位，地址码是 16 位，寄存器共 2^4＝16 个。执行 B＋C→A，数据字长 32 位。这三种不同操作指令中，以存储器-存储器操作指令的存储效率最高。

(1) 寄存器-寄存器操作指令




```
LOAD  B,rB
LOAD  C,rC
ADD   rB+rC→rA
STORE rA,A
    I=104  b
    D=96   b
    M=200  b
```

(2) 存储器-寄存器操作指令




```
LOAD  B
ADD   C
STORE A
    I=72   b
    D=96   b
    M=168  b
```

(3) 存储器-存储器操作指令

```
ADD  B+C→A
I=56 b, D=96 b, M=152 b
```

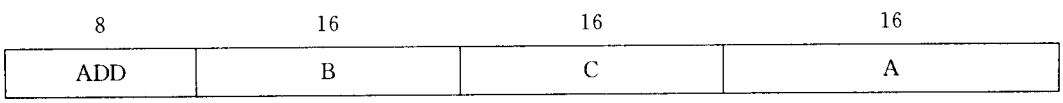


图 2.2　三种不同操作指令

当时很多研究工作者都认为应当大量采用存储器-存储器操作指令，指令系统中含有这种指令愈多，则程序愈紧凑，存储效率愈高。有的作者甚至认为不能再使用寄存器-寄存器操作指令，但实际上存储器操作是很复杂的。

4. 指令系统的复杂便于追求软件兼容。如以 INTEL 的 80x86 系列为例。8086 设计要考虑到 8085 原有的几个 8 位寄存器的指令，80286 则要考虑到与 8086 兼容的指令。如此发展，则在不断扩充其性能的同时，还要考虑与前代处理器的兼容性，其指令系统必然愈来愈庞大了。

归纳以上的情况，60 年代末期和 70 年代以至 80 年代初的计算机体系结构设计中有两个特点：

1. 把存储效率作为体系结构设计质量的重要衡量手段，大量采用存储器-存储器操作指令，甚至采用面向存储器堆栈操作的体系结构。为了充分利用内存，还采用了可变字

长的指令，即复杂指令用多个字节，简单指令用较少字节。由于存储器是以字(32 位)为基本单位组织的，若指令系统具有可变字长的指令，则程序在内存中的装配可以更加紧凑。

这种想法认为机器执行速度和程序代码大小成正比，但却没有去深入分析执行效率与计算机结构的复杂性。

2. 微程序控制占主导地位。微存储随指令系统庞大而增大，微存储容量甚至可与 cache 相比拟。

表 2.1 列出了当时很著名的几种计算机。从表中可以看出，其指令数很多，指令长度相差悬殊，有大量的存储器操作指令，微存储容量很大。

表 2.1 四种典型计算机的结构特点

机 型 (年份)	IBM 370/168 (1973)	VAX 11/780 (1978)	i APX 432 (1982)	Dorado (1978)
指令数	208	303	222	270
微存储容量(位)	420 Kb	480 Kb	64 Kb	136 Kb
指令长度(位)	16～48	16～456	6～321	8～24
工艺	ECL MSI	TTL MSI	NMOS VLSI	ECL MSI
指令操作类型	存储器-存储器 存储器-寄存器 寄存器-寄存器	存储器-存储器 存储器-寄存器 寄存器-寄存器	面向堆栈 存储器-存储器	面向堆栈
cache 容量(字节)	64 KB	64 KB	0	64 KB

2.1.2 微程序设计技术遇到的问题

80 年代计算机工艺技术，尤其是 VLSI 工艺技术取得了惊人的迅速发展，前述的传统计算机体系结构设计思想已经不能符合新工艺技术的要求，微程序设计技术遇到了下述问题。

1. 半导体存储器已完全取代了磁心存储器。主存的速度已经可以和微程序存储相比拟，不再比后者慢(5～10)倍。换句话说，一个机器周期不再等于(5～10)个微周期了。

此外，大容量半导体存储器价格日益下降，一般微机系统的主存就可达几个兆字节，因此，所谓“存储效率”已不再是体系结构设计时要考虑的重要衡量标准了，设计观念必须随计算机工艺的发展而变化。

2. 现有的一些著名计算机的指令系统过于复杂，使微存储器的微码多达 400 千位。这样微程序设计很易出错，庞大的微程序设计因而也愈来愈复杂。为了微程序设计，甚至还要写微程序的编译器和调试程序，以至形成了写微程序的“程序设计语言”。

3. 研究还表明，在微码机器中，很难做到一条指令的执行接近于一个微周期。一般来说，平均每条指令至少要(3～4)个微周期。然而，很多程序中的一些简单指令实际上只与一条微指令的操作相当，这些指令的执行完全不必靠微存储的控制部件来实现。依靠一些

很简单的硬布线逻辑，这些指令可以在一个主存周期内完成(而不是几个微周期)。这里，已经孕育着 RISC 的基本思想，即一个机器周期实现一条基本指令。换句话说，从硬件角度来看，简单的基本指令就相当于过去的微指令，一个机器周期就相当于微周期，而过去庞大的微存储 ROM 完全可以相当于快速 RAM，即现代计算机中的快速主存或者 cache。

由此可见，微码技术在小规模容量情况下或者在局部应用时(如乘法、除法和浮点运算)还保持有优点。但如果微码存储过大，则微程序设计与程序设计已没有明显区别，一条微指令就已相当于一条基本指令。

2.1.3 RISC 设计思想的起源

精简指令系统计算机设计思想的起源主要有三个方面：一是由于 VLSI 工艺的迅速发展改变了传统的计算机设计思想；二是通过对指令系统运行效率的分析与统计，得出 20%-80%定律；三是重新评价一个计算机系统中硬件与软件之间复杂性的优化划分，即在系统设计时，应在硬件与软件之间取得折衷，平衡负担整个系统的复杂性。以下讨论这三方面的内容。

1. 20%-80%定律

从表 2.1 已经看出，70 年代一些著名计算机指令系统指令数已多达二百多条到三百多条。有人计算，VAX 11/780 有 303 条指令，而每条指令都还具有多种寻址方式，因此总共有多达上千种的不同的指令操作。

70 年代中后期，就有大量工作在研究这么多的指令执行的效率如何。通过静态的和动态的测试，分析哪些指令是经常使用的，哪些指令不经常使用。这样，便得出了有名的所谓“20%-80%”定律。

这条定律表明：通过大量程序实际运行的分析，一个指令系统中大约 20%的指令是在程序中经常反复使用的，其使用量大约占到整个程序中的 80%；而该指令系统中大约 80%的指令是很少使用的，其使用量只占整个程序的 20%。

这条定律在我们的研究工作中也得到了证实。我们开发了追踪指令执行并分析统计的工具，在 IBM PC 上对于 8088 指令系统做了分析，其中一项 C 编译器和 PROLOG 程序的测试分析结果见表 2.2。8088 指令操作种类大约有 100 种，其中头四种最常用的指令其程序累计百分比达 50%以上，执行时间累计达 58.65%。到了第 20 种指令，程序累计百分比已达 91.10%，超过了“20%-80%”定律中的 80%，而执行时间累计已达 97.72%。其余 80%的指令的使用频度都在 4‰以下。

统计分析工作表明，20%的基本指令使用相当频繁，如果按指令种类来划分，只有少数几种指令集中使用。我们把 8088(IBM PC/XT)的指令分成六种类型，即

① 数据传送类型
② 算术运算类型
③ 逻辑运算/位操作类型
④ 字符串处理类型
⑤ 转移类型
⑥ 处理器控制类型

表 2.2　指令系统运行的分析与统计

按使用频度排序			按执行时间排序		
指　令	占百分比	累计百分比	指　令	占百分比	累计百分比
1. MOV	24.85	24.85	1. IMUL	19.55	19.55
2. PUSH	10.36	35.21	2. MOV	17.44	36.99
3. CMP	10.28	45.49	3. PUSH	11.11	48.10
4. JMP_{cc}	9.03	54.52	4. JMP_{cc}	10.55	58.65
5. ADD	6.80	61.32	5. CMP	7.80	66.45
6. POP	4.14	65.46	6. CALL	7.27	73.72
7. RET	3.92	69.38	7. RET	4.85	78.57
8. CALL	3.89	73.27	8. ADD	3.27	81.84
9. JUMP	2.70	75.97	9. JMP	3.26	85.10
10. SUB	2.43	78.40	10. LES	2.83	87.93
11. INC	2.37	80.77	11. POP	2.61	90.54
12. LES	1.98	82.75	12. DEC	1.49	92.03
13. REPN	1.92	84.67	13. SUB	1.18	93.21
14. IMUL	1.69	86.36	14. XOR	1.04	94.25
15. DEC	1.37	87.73	15. INC	0.99	95.24
16. XOR	1.13	88.86	16. $LOOP_{cc}$	0.64	95.88
17. REPNZ	0.78	89.64	17. LDS	0.64	96.52
18. CLD	0.54	90.18	18. CMPS	0.44	96.96
19. $LOOP_{cc}$	0.52	90.70	19. MOVS	0.39	97.35
20. TEST	0.40	91.10	20. JCXZ	0.37	97.72
21. SYI	0.40	91.50	21. LODS	0.31	98.03
22. LDS	0.39	91.89	22. REPN	0.28	98.31
23. LODS	0.35	92.24	23. INT_{cc}	0.26	98.57
24. AND	0.35	92.59	24. STOS	0.25	98.82
25. SHR	0.33	92.92	25. SHR	0.23	99.05
26. STOS	0.31	93.23	26. LEA	0.12	99.17
27. MOVS	0.29	93.81	27. OR	0.11	99.28
28. JCXZ	0.29	93.81	28. REPNZ	0.11	99.39
29. SH/AL	0.27	94.08	29. AND	0.09	99.48
30. CMPS	0.27	94.35	30. TEST	0.09	99.57

我们运行了七种应用程序(F1—F7),发现其中三种类型,即数据传送、算术运算以及转移指令占用了90%以上,其余三种使用较少,见表2.3。

表2.3 各种类型指令使用率分布平均值

指令类型	应用程序							平均值
	F_1	F_2	F_3	F_4	F_5	F_6	F_7	
数据传送	34.25	35.85	28.84	20.12	25.04	24.33	34.31	30.25
算术运算	24.97	22.34	45.32	43.65	45.72	45.42	28.28	36.24
逻辑运算/位操作	3.40	4.34	7.63	7.49	6.38	3.97	4.89	5.44
字符串处理	2.42	4.22	2.72	2.01	2.10	2.35	2.10	2.86
转移指令	34.84	32.99	15.34	17.63	20.52	25.74	30.29	25.33
处理器控制	0.13	0.26	0.15	0.10	0.24	0.19	0.14	0.19

在研究指令执行效率方面,不仅要注意指令的执行频度,还要研究指令执行时间占整个程序执行时间的百分比。我们运行了十种应用程序,其中七种是快速傅里叶转换程序共有九种指令的执行时间占据了整个程序执行时间的70%左右。见表2.4。

表2.4 基本指令的执行时间百分比

类型 / 应用程序 / 基本指令	传送		乘除		过程调用	跳转		合计
	MOV	PUSH/POP	(I)MUL	(I)DIV	CALL/RET	JMP	JMP_{CC}	
FOUREA	23.1	9.9	8.4	1.2	11.3	3.1	18.2	75.2
FAST	25.8	11.2	5.8	0.3	10.2	2.5	15.6	68.4
FFTSUBS	25.6	8.8	10.0	1.5	9.8	2.2	16.2	74.1
MXFFT	23.6	5.4	12.6	3.5	8.9	2.1	19.7	73.8
MSFFT	23.6	7.7	15.1	0.4	9.3	2.4	17.1	75.6
RAFFT	26.1	8.9	8.0	3.8	10.6	2.4	18.4	78.4
FFT2D	26.9	5.1	11.7	1.3	11.7	2.7	17.4	76.8
PMPSE	25.2	4.2	15.0	0.5	10.3	2.3	19.6	76.1
CMPSE	25.8	3.4	14.8	0.5	10.5	2.3	19.9	77.3
CCSEP	22.5	8.3	12.0	0.5	10.8	2.4	18.5	70.0
平均	32		13		10.3	20.4		75.7

对于指令系统实际执行情况的分析和统计表明:不仅指令类型相当集中,而且寻址方式也相对地集中在几种方式上(见表2.5)。对于转移指令来说,转移的范围也比较集中,条件转移范围集中在偏移值为128以内,而无条件跳转的范围则集中在偏移值为512以内(约60%),但其集中程度不如条件转移,见表2.6和2.7。这些统计结果都有助于设计精简指令系统的寻址方式和指令偏置值字段。

2. 系统设计中硬件与软件方面之间折衷

70年代中后期VLSI取得显著进展,此后,人们重新评价系统设计中硬件与软件的复杂性之间应该如何取得优化的划分。这种研究对于RISC思想的形成有很大的影响。这里举一个著名DEC公司的实验的例子。DEC要把70年代广泛应用的VAX11/780用

表 2.5 寻址方式的统计

寻址方式	百 分 比
立即寻址	30.18
相对寻址	21.68
寄存器寻址	15.79
索引变址	11.58
寄存器间址	8.57
直接寻址	8.27
基址	3.92
基址索引变址	0.01

表 2.6 条件转移范围的统计

转移范围	百分比
0…1	3.94
2…3	8.45
4…7	12.24
8…15	20.55
16…64	21.67
64…127	17.99
128	0.21

表 2.7 无条件跳转范围的统计

跳转范围	百分比	累计百分比
0…127	31	31
128…255	13	44
256…511	16	60
512…1K−1	4	64
1K…2K−1	1	65
2K…4K−1	5	70
4K…8K−1	5	75
8K…16K−1	8	83
16K…32K−1	6	89
32K	11	100

VLSI 芯片实现。第一种途径是依靠 VLSI 的复杂性来实现 VAX 11 的全部指令系统。第二种途径是用较少的 VLSI 复杂性只实现 VAX 11 中的一个子集，这个子集包含有 VAX 11 的基本指令，而其余的指令则用 TRAP 转成软件子程序实现。

实践表明，VAX 11/780 中有 20%的指令占用了微程序控制部件微码的 60%。而这些指令却很少用到，在执行标准程序时只占总指令数的 0.2%。因此，用第一种途径实现 VAX 11 全部指令，需要九片 VLSI 芯片，共需 1250k 个晶体管，相当于第二种方案的复杂程度的(5～10)倍，但速度只比第二种方案快 20%。

这两种方案实现方法见表 2.8。

表 2.8 VAX11 的两种 VLSI 实现方法

项目 \ 方式	第一种方法	第二种方法
VLSI 芯片数目	9	2
微码存储容量(KB)	480	64
晶体管数	1250k	101k