计算机体系之变迁

（征询意见稿）

第三分册

张　修编

中国科学院计算技术服务社

一九八一年七月

第三分册目录

第四章　统一与反统一之争（单机体系结构到多机体系结构）

早在1961年美国就研制成了第一台试验性的空间领行集成电路计算机。但普遍认为应把1964年IBM公司发表360系列作为计算机进入第三代的标志。360系列不仅使用了集成电路，而且它将大小若干种机器形成具有统一体系结构的系列，又拥有大量的软件，使它在国际计算机市场上取得很大成功。进入七十年代以后该公司又发表了IBM 370系列。它是360系列的改进。以后IBM公司又陆续发表了3033系列（1977年）和4300系列（1979年），但它们主要是元件和工艺上的革新，体系结构只有轻微的变化。在360和370系列的影响下，其他计算机厂家也宣布研制系列机。在日本和欧洲国家，这一行动是和计算机工业的改组同时进行的。1972年在日本政府的促进下，六家日本计算机公司改组成三个集团，分别生产三个主要的计算机系列。富士通——日立集团研制M系列，目前以FACOM M-200和HITAC-M200H为最大。日本电气——东芝集团研制ACOS77系列。三菱——冲电气集团研制COSMO系列。1968年在英国成立了ICL公司，除继续生产1900系列外，近年又推出2000系列。法国在1967年成立了CII公司。西德政府则从1967年起大力资助西门子公司。1973年CII公司、西门子公司和荷兰的菲利浦公司曾联合生产7.700系列。但不久即因联合失败而告终。7.700系列成了西门子公司的产品。最近它又宣布了7.300系列。苏联和东欧国家在1972年搞了一个РЯД—I系列，在1978年又搞了

一个РЯД—Ⅱ系列。此外，专门研制与IBM公司兼容的计算机而与其抗衡的美国阿姆德尔公司继470系列以后，在80年底又发表了580系列。

所有这些系列都不及IBM 360和370系列的成功，而且有些不过是这两个系列的仿制。

体系结构仍在继续发展，而且有些系列机中的机种也采结了某些新的设计思想。

在第一、二代计算机中未受重视的微程序设计思想，由于第三代技术条件的成熟而取得进展。1972年研制的B1700和1971年研制的QM-1分别是垂直微指令和水平微指令的代表。沿着这个途径发展，就是下一章分绍的固件工视和动态体系结构计算机。

在存储层次的控制上，虚拟存储器的思想由于1972年被IBM 370/158和168所采用，受到很大重视。隐含存储器的工作方式，1969年首先在IBM 360/85中实现。1972年日水研制的超高性能计算机则将隐含存储器扩大应用于多处理机之中。

联想存储器也因半导体存储器的制成而获得实用的条件。著名的使用联想存储器的计算机有1976年部分制成的PEPE和1972年制成的STARAN。

在提高可靠性方面，采用了多种将故障自动定位的诊断措施和提高系统可靠性的冗余技术。本章第六节介绍的STAR计算机就是这样一台计算机。它是1961年制成的。

在发展单指令流单数据流的单机结构的同时，单指令流多数据流和多指令流多数据流的结构被大型机和超大型机的设计所采用。1972年制成的著名的ILLIACIV就是一台单指令流多数据流的阵列计算机。1976年由ICL公司制成与2900系列连接的分布式阵列处理机DAP和1979年由巴勒斯公司制成的科学处理机BSP是这一思想的进一步发展。

流水线工作方式在第三代计算机继续发展。IBM 360/91和195在第三章第二节中已做过介绍。它的下一步就是向量计算机。向量数据的处理都是用专门的部件。当这个部件处于次要地位时，称其为数列部件。如1976年为FACOM 230/75制造的数列部件AP和1979年制成的HITAC M200H中的IAP。若将向量处理作为计算机的主要功能，则成为向量计算机。如STAR-100（1974年）、TI ASC（1972年）和CRAY-I（1976年）。1980年出现的大型计算机CYBER 203是STAR-100的进一步发展。

多指令流多数据流的多处理机系统是逐渐形成的。1970年的UNIVAC1110 1971年的日本数据处理装置DIPS-I是比较有名的多处理机系统。1971年制成的试验装置C_{mmp}则是多计算机系统，它是由多台PDP-11组成的。上述的多处理机或多计算机系统有时称为均质系统。因为其中每台处理机或计算机在系统中的地位相同，只是在处理具体问题时有所分工。如果各个处理机的功能固定不变，甚至做成专用装置，如1970年的SYMBOL计算机，则称为功能分布系统。或功能

分布式计算机。SYMBOL还有一个特点就是它便于直接执行用高级语言编写的程序，因此也是台面向高级语言的计算机。英国曼彻斯特大学在1972年制成的MU-5也属于这一类计算机。关于这方面的功能将在下一章第三节中介绍。

位置分散，用通讯线路连接的多计算机系统，又称为计算机网络。1969年开始建立，随后不断发展扩大的美国ARPA网络是其中著名的一个。七十年代中期很多计算机厂商都宣布他们的计算机可以构成网络，并提出了不少网络体系结构其中以IBM公司在1974年提出的SNA较为有名。

本章将循著上面简述的由单指令统单数据流（SISD），到单指令流多数据流（SIMD），到多指令流多数据流（MIMD）的发展途径，介绍各种体系结构。并简短地说明上面列举的一些有代表性的机器。还有一些计算机，如和IBM 370系列兼容的AMDAHL 470/V，日本超高性能计算机的商业型号HITAC 8800以及CDC 7600改用集成电路制成的CYBER 76等，因体系结构上没有新特点，就不分绍了。

第三代计算机发展到何时结束？虽然人们往往把微型计算机的制成和发展，看做是计算机进入第四代的标志，但到目前为止，第三代计算机的生产与使用仍盛况未衰。

最后就本章的标题说几句。统一与反统一之争并不意味着，有一批人主张统一，另一批人反对统一。它反映了事物发展过程中矛盾着的面个方面。发展中需要统一，面统一中又出现新的矛盾。客观上证明了事物发展的辩证规律。

一、兼容性和系列机

第二代计算机由于元件技术的提高，应用范围的扩大，体系结构（主要是单机体系结构）进入了一个大发展时期。第三章中我们已经介绍了这个发展过程。但是这个发展也引起了一些问题。其主要原因在于机器型号过于分散，体系结构不统一。产生了哪些问题呢？

首先，应用范围的扩大，各种各样的用户都要使用计算机。他们要求计算机易于使用，但直接使用机器语言编程序是困难的，因此发展了各种软件。到六十年代中期，软件的重要性已越来越被人们所认识。然而，每换一个机种，就要重新搞一套软件，而软件的工作量很大，并且随着软件内容的不断增加，这种工作量更加日益扩大。有时两个机种只是不大的差别，但为此要修改软件的工作量，却是繁重的。也许只需修改几条指令，但如何找到这些指令，如何确定正确的修改方案，就是一个难题。特别由于软件往往由许多人编制而成，这种修改的困难就更大。因此，软件的发展强烈要求计算机作系结构的统一。

其次，应用的深度也在扩大。六十年代以来，不仅计算机应用领域扩大，而且在每个领域内应用计算机解算的问题也逐年增加。如1960年政府机关只利用计算机解决7个方面的问题，到1974年就扩大到210项。还要注意到，解题的复杂性也在增加。举飞机设计中空气动力学的计算为例，最初只能解算二维无粘方程，以后发展到解三维无粘方程，进一步发展到解二维粘性方程，现在又朝解三维粘性方程的方向努力。所用的机器规模日益扩大。但是由于体系结构不统一，每次扩大规模，

都要更换机器型号，都可能重新修改用户的解题程序。有人可能认为，第二代计算机期间，高级语言已经发展起来，用户程序多用高级语言编写，还受机器更换的影响吗？事实并非如此。虽然在六十年代初已经提出了几种高级语言并获得推广。在七十年代还逐步做到标准化。但是，在机器上具体实现时，仍要根据机器的体系结构做适当的修改。特别在输入输出格式及存储管理方面，更有各种差别。顺便指出：几种高级语言在六十年代初期提出并得到推广，这件事本身就反映出要求统一。

另外，应当看到，在六十年代中期也具备了统一的条件。这包括两个方面：元件技术方两和设计经验方面。

在元件技术方面，主机的基本元件已由晶体管、电容、电阻等分离元件，逐步过渡到以门和触发器等逻辑结构为单位的集成电路。存储部件，输入输出部件已不再专为某种机器型号所特制，两是作为一种独立的部件研制和生产。元件技术的这一变化，也促使体系设计的统一。因为体系结构的统一，有利于这些元件和部件的标准化。

在设计经验方面，虽然当时还做不到通过严格数学推导确定参数（现在也做不到），但由于近二十年经验的积累，已经可以对各种方案做出较为一致的取舍意见。

正是在这些前提下，提出了体系结构概念和兼容性的问题。

什么是兼容性？笼统地讲，就是在一台计算机上解题的程序也能在另一台计算机上执行。这两台计算机就有兼容性。由于这个问题是在1964年提出来的。当时高级语言的使用还不如现在这样普遍和需要，所以解题程序是理解为由汇编语言或机器语言编写的。为了进一步说明

这点，当时还强调了体系结构概念。认为体系结构就是用户对计算机硬件结构的理解。

不过十几年来的发展，这个含义出现了混淆。很多用户是通过高级语言使用计算机的。对他们来说，对某种语言的理解比对某种计算机型号的理解更清楚。同时软件的比重在计算机系统中已超过硬件的比重，出现了软件体系结构这一概念。因此提出了在哪一级语言上兼容的问题。在高级语言一级上兼容，还是在汇编语言或机器语言一级上兼容？

但是，提出这个问题是违反原意的。兼容主要是解决软件发展中出现的危机和困难。如果在高级语言一级上兼容，则不但没有解决问题，反而使问题更加严重。因此，还是应当提以汇编语言或机器语言一级兼容为宜。

以上的讨论表明，兼容性这个问题，提出时就不严格，现在则更加混乱。它反映了人们有了统一要求，但如何统一还并没有全都搞清楚。既然兼容性还是一个发展变化的概念，本节中只能根据最初提出这一概念的目的，对它加以说明。

兼容性要求：

第一、两台机器的数据形式和指令形式完全一致。

第二、两台机器的输入输出接口形式完全一致。

第三、两台机器的指令系统基本一致。

这三条要求概括来讲，就是信息的内部外部表现形式和处理方式要一致。第三条要求的基本一致，理解为一台计算机上的指令系统，完全包括在另一台的指令系统之内，而另一台计算机的指令系统，则不完全

为前一台计算机具备。因此，在前一台计算机解算的问题，在后一台上同样能算，而在后一台计算机上解算的问题，则只有在一定的条件下才能由前一台计算机计算。从前一台转到后一台是向上兼容，而从后一台转到前一台是向下兼容。

最早提出并实现兼容性的是IBM公司，它所研制的IBM 360系列包括13种型号的机器，但彼此具有兼容性。七十年代发表的IBM 370系列，不仅本身十几种型号的机器之间具有兼容性，而且和360系列也具有兼容性，以后发表的303x系列和4300系列，也都和前两个系列具有兼容性。这种风气影响到美国，日本和西欧的其他计算机厂家，纷纷发表各自的系列，但都不如360和370系列影响深远。所谓系列机就是研制速度快慢不同，存储量大小不同，但彼此兼容的多种型号的计算机，供用户选择。

系列化的成功有好处也有问题，它限制了体系结构的发展。虽然在360和370系列中，以后采用了虚拟存储器，隐含存储器、分时系统、多处理机等一系列新的体系设计思想。但这些都给兼容性带来了麻烦。一方面必须修改体系结构，增加新的指令和控制方式。另一方面又受原来框框的限制，不能做较大的，但是合理的修改。这样一来，就使七十年代以后，只是在超大型机和微型机方面，体系结构有新的发展，而在中大型通用机的体系结构方面，几乎没有出现重大变化。

如何看待系列机的出现？系列机是通用的单机体系结构的一次总结和归纳，自此以后，体系结构朝向多机系统和专用系统的方向发展。

最后，列出IBM 360和370系列各种机器的指标。

IBM360／370系列性能表

机器型号	制成年月	存储容量（位组）	存取周期（微秒）	隐存容量（位组）	加法时间（微秒）	通道传输率（位组／秒）
360/20	1974	4-32K	1.0	—	58.00	0.160M
360/22	1971	24-64K	1.5	—	30.00	0.170M
360/25	1968	16-48K	0.9	—	45.00	0.600M
360/30	1964	24-64K	1.5	—	30.00	0.170M
360/40	1964	32-256K	2.5	—	11.88	0.800M
360/44	1966	8-64K	1.000	—	1.75	1.000M
360/50	1964	128-521K	2.0	—	4.00	1.000M
360/65	1965	256-2000K	0.75	—	1.40	1.300M
360/67	1965	256-1000K	0.75	—	1.60	1.300M
360/75	1965	256-1000K	0.75	—	0.70	1.300M
360/85	1969	512-4096K	0.96	16-32K	0.16	1.200M
360/90	1967	512-1000K	0.75	—	0.18	1.200M
360/195	1968	1000-4000K	0.756	32K	0.054	1.500M

370/115	1973	64-384K	0·480	—	14·50	0·029M
370/125	1972	96-512K	0·480	—	9·80	0·029M
370/135	1971	96-512K	0·770	—	4·20	2·600M
370/138	1976	512-1000K	0·715	—	4·20	2·600M
370/145	1970	160-2000K	0·540	—	2·10	1·850M
370/148	1976	1000-2000K	0·405	—	—	1·850M
370/155	1970	256-2000K	2·070	8K	0·99	1·500M
370/158	1972	512-6000K	0·920	8K	0·80	1·500M
370/165	1971	512-3000K	2·000	8K	0·16	3·000M
370/168	1972	1000-8000K	0·320	8K	0·15	3·000M
370/195	1971	1000-4000K	0·756	32K	—	1·500M

参考文献

1、IBM System/360 Principles of Operation, IBM

2、IBM System/370 Principles Of Operation, IBM

二、微程序设计

微程序设计这个概念是早在五十年代初，由英国曼彻斯特大学的威尔克斯教授提出来的。他在1951年发表的，题为“设计自动计算机的最好途径”一文中，写到：

“……控制的实质就是，在机器中提供脉冲，驱使与运算和控制寄存器相联的门进行操作的部件。机器中这一部件的设计者往往要先以某种方式画出框图，直到他找出一种安排能满足他的要求而且相当节约为止。我想建议一种方法，可以做到控制系统化，从而也减少了复杂性。

由机器程序中一条指令所引起的每一个操作，都涉及到一系列步骤。它可能包括从存储器到控制或运算寄存器的传送或相反，还包括从一个寄存器到另一个的传送。每一步骤都要向与控制和运算寄存器相联的某些传输线中发送脉冲，而我称之为“微操作”。因此，每一个真正的机器操作都是由一系列微操作或者说由微操作组成的“微程序”实现的。

图中（见图1）示出用什么方法产生执行微操作的脉冲。启动微操作的定时脉冲进入译码树，并按照寄存器R中的数值发送一个输出。它通过一个整流矩阵A，并按照整流器的排列使这一矩阵的某些输出线上出现脉冲。这些脉冲驱使与控制和运算寄存器相联的门动作，从而执行正确的微操作。从译码树来的脉冲也送到矩阵B，并使这个矩阵的某些输出线上也出现脉冲。这些脉冲经过很短的延迟线又回到寄存器R，并使其中的数值发生变化。结果是下一个进入译码树的启动脉冲将显现在不同的出口并因此执行不同的微操作。由此看来，矩阵A中的每一行整

流器对应于执行一个机器操作所需序列中的一条微指令。”〔1〕

威尔克斯非常明确地表述了他的思想。但尽管以后也做了一些工作，十多年中都未获得实际的应用。其主要原因是没有足够快的存储器存储这些微指令，而使它的存取速度与主机控制逻辑的速度相当。

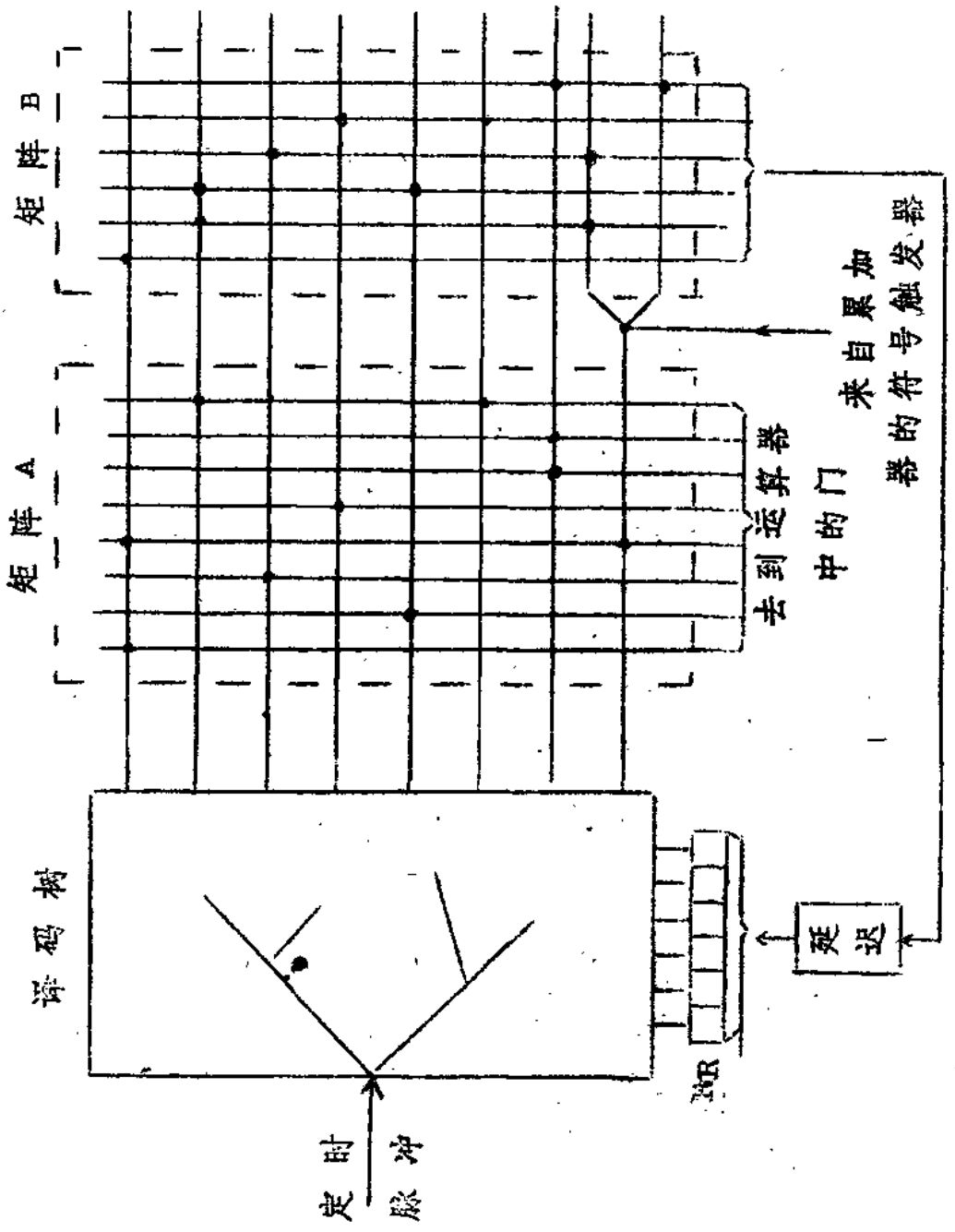

图1 威尔科斯的微程序控制模型

到六十年代中，由于用电容或电感构成的只读存储器的出现，它才有了技术基础。首先由IBM公司在360系列中使用了这种技术。在六十年代到七十年代初，主要实用于小型计算机之中，在七十年代，由于大规模集成电路的出现，它开始进入大型机的领域。

本节首先说明微程序设计概念以及如何用微程序实现机器指令，然后讨论微程序计算机的若干特点，最后介绍有代表性的微程序计算机。

1、微程序设计概念及用其实现机器指令

提出微程序是作为实现机器语言的另一种方法，应把它与一般机器语言分开。假定有一台微程序计算机。这台机器的操作如下：

首先要有一个控制存储器，其中包含着各种微指令。它表示应当由机器硬件执行的最原始的操作。

然后机器按下列四步执行一条微指令：

第一步，控照控制存储器地址寄存器中的地址将一条微指令从控制存储器读到微指令寄存器。

第二步，这条微指令所要执行的微操作被译成控制信息。

第三步，形成的控制信号将驱动相应的硬件，执行机器的原始操作。

第四步，时序控制线路利用结果状态信息选择下一条应执行的微指令的地址，并将此地址送至控制存储器的地址寄存器。

重复上述过程，执行一系列微指令，即微程序。

通常用微程序解释正常的机器指令。每一条机器指令都对应着控制存储器中的一段微程序。除取指令的功能是由从控制存储器0单元开始的微程序执行外，其他各段微程序的起始地址将隐含在正常指令的操作

码之中。在指令译码时，就由译码器形成一个控制存储器的起始地址，从中调取并执行实现该指令功能的微程序。但是不能把微程序只限于实现机器指令。可以用一段微程序实现某种算法或于程序，而其效果会比用正常机器指令编写的好。在这个意义上，似乎可以把微指令看成是一种新型的独立的机器语言。

如何区分微指令和正常机器指令呢？至少有四点差别。

第一、微指令直接控制硬件的原始操作，而每一条正常的机器指令，则要执行一个操作序列，以处理由用户定义的数据结构。

第二、正常的机器指令系统都比较大，而且接近自然习贯，使人便于理解，但微指令系统通常都比较小，而且内容形式要适应机器结构。

第三、机器允许同时完成几个微操作，但程序的执行则往往要按一定顺序。

第四、微程序通常是存在一个专门的存储器（即控制存储器）中，它的速度比较快，以便和主机逻辑相匹配，但因此它的成本较高，容量也比较小。

使用微程序实现机器指令（称为存储逻辑）与用逻辑门组合实现，（称为连线逻辑）比较起来有优点，也有缺点。优点是成本低，灵活性高，容易扩充和维护，以及前面讲到的，在实现某些算法或于程序时效果好。它的主要缺点是要从控制存储器中取微指令，这要占用一定时间，而使机器数率降低。

2、微程序计算机的若于特点

在设计一台微程序计算机时，要考虑四个方面的问题，这就是：控制存储器的设计，微指令的设计，微指令的实现和微程序的能力。

控制存储器的设计

控制存储器的逻辑结构可以有下述几种形式：

第一、最简单的也是最常用的是常规存储结构，每个控制存储单元中放一条微指令。

第二、将每个控制存储单元的位数增加，做到每个单元两条指令，它减少了访问控制存储器的次数。

第三、将控制存储器分成若干组。组内单元用相对地址，组与组之间才用全地址。这便于处理转移微指令。

第四、将控制存储器分成两种结构：一种短微指令，一种长微指令。类似数据传送这类指令只需用短微指令，而一些控制硬件较多的微指令，需要的信息位多，所以构成了长微指令。访问几条短微指令才相当于访问一条长微指令。

第五、两级控制存储器。上一级是微指令，下一级叫微微指令。后者将解释微指令，就和用微指令解释正常指令一样。这样做速度自然较低，但具有更大的灵活性。

微指令的设计

在设计微指令时，主要考虑它能控制的硬件有多少。由此分成垂直微指令和水平微指令两种。垂直微指令只是执行单一的操作，如取数、加、送数、转移等。它的长度大约为12到24位。水平微指令则相反。