微处理器

—工艺、结构及其应用

吉林电子编辑部

微 处 理 器

—— 工艺、結构及其应用

序　言

半导体大规模集成技术领域的最新进展已经有可能大大减少数字逻辑电路的成本和尺寸。近十年中，计算机系统结构部件已经从分立元件(例如单个的晶体管)发展到在一个半导体"芯片"上含有大量逻辑门的功能复杂的集成电路。在单个集成电路芯片上集中了大量算术与逻辑功能，这样在体积上经济上表现的优点，对于袖珍式计算器和智能终端设备制造厂家来说是特别有利的，"单片式计算机"的首次动议就来源于此。

"单片式计算机"的常规设计虽然早就有人尝试过，但是直到一九七一年，第一批商品微处理器才出现在市场上。对任何革新产品的了解并为市场所接受总是逐渐的过程，特别对于象微处理器这样复杂的产品就更是如此了。仅在几年以前，数字系统设计者才完全了解了微处理器的多方面优点；也是在几年以前，由于厂家认定了在新生产线上利用微处理器的可行性，从而使生产过程具有足够的规模。至一九七六年，已有20余种微处理器成为商品并公开发表；微处理器的生产能力为每月三万五千台，同时已为很多的计算机终端设备、实验室仪器、工业控制系统以及民用消费产品所引用。

微处理器是一种大规模集成电路（LSI）片子，它能以并行方式在程序控制下完成算术与逻辑运算。微处理器本身并不是可用的计算机，必须外加存贮和输入输出电路，使之与系统相接，还需要有控制微处理器自身操作的软件。用微处理器和其它集成电路部件来设计和研制这种微型计算机是本书的主要内容。

本书介绍了微处理器和微型计算机的工艺、结构及其应用。这是为微处理器的可能用户——打算把微处理器引入特定应用的电子设计者或计算机系统分析工作者而写的。

第一章评述与微型计算机系统设计有关的基本计算机结构。第二章讨论微型计算机系统设计的基本任务并概述微型计算机的主要部件。

第三章叙述制造微处理器的基本半导体工艺。虽然详述半导体生产工艺过程对于微型计算机系统设计者来说并不是直接关心的事，但是微型计算机的成本和实际操作特性都是以这些原理为基础的。在微处理器的制造中所使用的双极型工艺和 MOS 工艺的基本知识，会使用户对这些部件的能力和局限性有更多的了解。

第四章综述各种商品微处理器的结构。由于隔几个月就发表一些新型的微处理器而使老式微处理器成为过时的产品，所以这种综述仅在于描述在商品微处理器中可以得到的结构和功能的一般范围。

第五章是对一个比较流行的微处理器——Intel 8080 的概述。其指令系统、定时与同步以及内部寄存器的操作在很多其他微处理器中都具有代表性。

微处理器的程序编制与普通计算机的程序编制有所不同，因为对系统的硬件部分必须给予较大的关注。第六章包括系统定义流程图、编码及测试与调试等内容。在第七章中讨论微处理器的研制系统，这是设计和研制微型计算机系统样机的重要工具。

第八章和第九章介绍微处理器的可能应用，讨论的范围从简单的家用微型计算机到较为复杂的远程通信实例。

目　　录

第一章　微型计算机引论

微型计算机是利用微处理器作为系统中央控制单元的存贮程序式计算机。在我们更详细地叙述微处理器和微型计算机系统之前，首先简略综述一下数字计算机技术的一些基本概念——硬件结构，软件结构以及操作系统，是有益的。

数字计算机最一般地由四个基本单元组成：输入／输出设备、存贮器、算术／逻辑电路和控制单元。所有通用计算机，从微型计算机到大型多重处理系统都可以用这些基本单元来表示。这四个单元之间的特定设计和互连就称为计算机“组织”（总体结构）。

计算机总体结构不但包括硬件结构而且还包括软件结构，同时还应描述静态硬件设计和动态软件加工之间的相互作用。虽然计算机的总体结构有各种各样的分类和型式，但是在这里我们只考虑对微型计算机应用来说最实用的两种：

von Neumann 结构

堆栈结构

在我们详细了解了这些结构之后，我们再说明在微型计算机系统中如何实现这些结构，并提出由微处理器工艺所带来的一些比较重要的经济成果。

計算机总体结构

图 1.1 以简单框图形式表示了一台通用数字计算机的四个基本结构单元。通用数字计算机是一个信息处理装置，对提供的信息进行一系列算术与逻辑运算。利用输入／输出（即 I／O）设备将信息输入和输出计算机。在计算机内部，要处理的信息和对信息所要执行的编码操作二者都存贮在存贮器中。这种实际的处理是由控制单元定时和定序的数字逻辑电路完成的。此控制单元也使 I／O 设备及存贮器，同算术／逻辑电路协调动作。

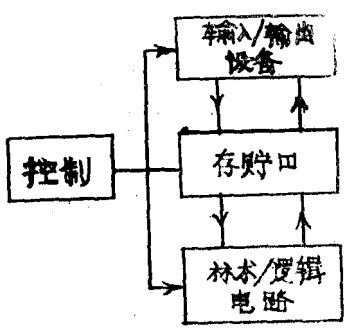

图1.1 数字计算机框图

不同类型的计算机结构都是由这种简单的功能块结构变型而来的。这种变型是通过增加存贮器或算术／逻辑电路（处理器）的数量，或者通过确定这些基本单元之间进行控制的特性而实现的。一些比较复杂的结构可以将存贮器和处理器分为不同的级别，同时要建立严密的系统控制过程，以便合理使用有限的系统资源。虽然这样的结构可以或者有时将采用微处理器来实现，但是目前应用的微型计算机系统不需要这种复杂的设计。

von Neumann 结构

von Neumann 结构是数字计算机最常用的一种结构，同时几乎是所有微处理器结构的基础。von Neumann 结构表示控制器与存贮器之间相互作用的一种形式。控制部分根据指令系列来操作，这一指令系列称为程序，存贮在存贮器中。每条指令包括两部分：操作码和操作数（图 1.2）。操作码表示由处理器执行的特定操作（算术操作如 ADD、逻辑操作如AND，以及控制操作如 JUMP），而操作数表示要处理的数据或存贮器的地址。

von Neumann 计算机是通过执行一系列与系统要完成的操作相对应的指令来运行的。

由于在控制单元和存贮器之间有很大的相互作用，所以就要象图 1.3 所示那样提供专门的硬件结构来促成数据与指令的传送。这些硬件结构是：

指令寄存器

程序计数器

累加器

算术／逻辑单元（ALU）

指令寄存器、程序计数器和累加器是图 1.1 中“存贮器”基本方框的最基本部件。这些部件是寄存器或存贮单元，其位数一般等于机器字长。指令寄存器存贮计算机将要执行的下一条指令。程序计数器存贮计算机接着要执行的下一条指令的存贮器地址。累加器存贮要处理的数据或者接受已处理过的新数据。

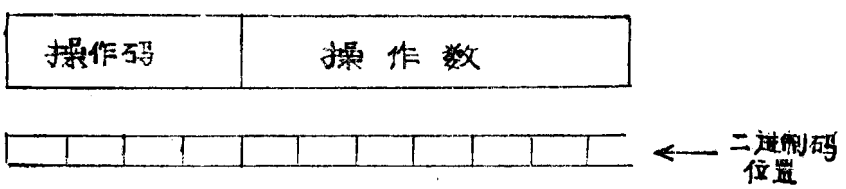

图1.2 指令字格式

ALU 根据指令指出的预定数据完成算术与逻辑运算。ALU 可表示成二个输入 A和 B、一个输出C，ALU根据输入A和输入B所提供的数据完成算术／逻辑操作，并把结果放置在输出C上。

用一条或几条总线把这些硬件结构相互连接起来。总线是在几个点之间的电连接，这些点可作为信号的源点或接收点（这可以类比乘客在不同的车站上下车）。图 1.3 所示的是一条双向总线，该总线是 n 位，n 指机器字长。微型计算机字长一般为四位、八位或十六位。

总线把这些寄存器、ALU 同存贮器及 I/O设备联接起来。在单总线结构中，每个同总线连接的设备都与其他设备共用这条总线。从而把一特定的时隙调配给每个设备，在时隙中，信息沿着总线传送到其他设备，或者从总线接收来自其他设备的信息。总线的时间多路传送由总线控制硬件来控制，并通过计算机控制单元使其同步。

控制单元完成计算机所有其他单元的管理和同步。计算机系统一般是同步时序数字

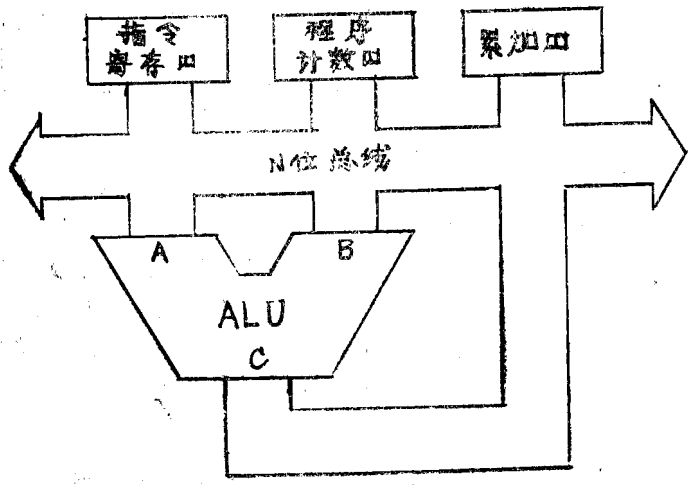

图1.3 双向总线结构

电路，根据在整个系统中提供的标准时钟信号使它们同步。计算机的每个部件都设计成在予定数目的时钟脉冲之后就发生时序转换。

图1.4简单表示了 von Neumann 计算机的基本部件。时钟信号发生器有二相输出ϕ_1和ϕ_2，计算机的基本操作部件是ALU，这已在图1.3中讨论过了。指令寄存器把下一条指令，提供给指令译码器，指令译码器能对二进制码格式进行解释，并对机器周期控制单元和 ALU 提供适当的指令指示。机器周期控制单元对这些指令译码，将正被执行的指令类型的外部信息，提供给控制输出单元和存贮器。这样，就能使其他的系统部件与 ALU 的操作同步。存贮器可以是读／写存贮器，也可以是计算机内部寄存器，而且还必须接受命令，以表明它需要提供数据进行处理。最后，示出了各种控制输入单元，以便提供机器周期的外部控制，如停止操作或中断操作。

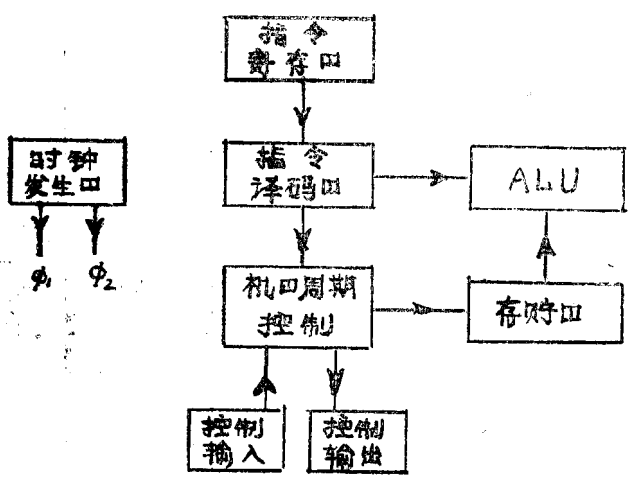

图1.4 von Neumann结构

堆 栈 结 构

计算机的结构也可以围绕着用户编程序的语言来组成。这样的结构有很多种，有的称为“可重新组合结构”或称为“可编译结构”，可以指望，这些结构是将来微处理器的重要应用领域。然而目前，在这些结构中最有批产意义的是“堆栈结构”，这样称呼是由于为了处理系统信息而使用了后进先出堆栈的缘故。

早在六十年代初期就研制了堆栈结构，以便通过提供一个直接执行指令的手段——堆栈来增加计算的效能。堆栈仅仅是个暂时存贮的装置，或者是单独的缓冲存贮器，或者是主存贮器的特定部分，用于依次存贮信息。如果新的信息进入堆栈，那么已经在堆栈中的信息，在堆栈中向下移动。如果信息从堆栈中移出，那么信息又以相反顺序向上移动。这样，放在栈顶的最后信息就是最先出来的信息。图 1.5 已经很详细地说明了这种后进先出下推堆栈的概念，这里“推”系指信息进入到堆栈的过程，“退栈”系指从堆栈中移出信息。

图 1.5 中所表示的堆栈仅仅指出堆栈是如何工作的，而不是表示后进先出堆栈在计

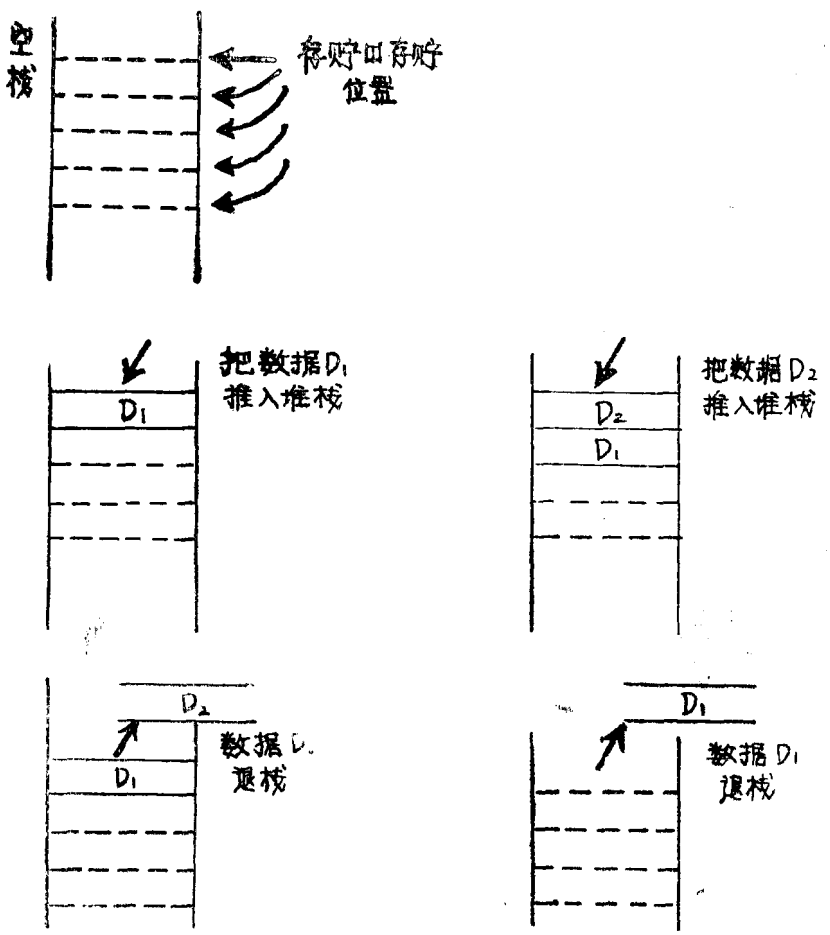

图1.5 下推堆栈的步骤

算机中实际上是如何实现的。虽然在功能上好象存贮的数据是在堆栈中“下推”或“上托”的，但是实际动作与此很不相同。

图1.6表示了在存贮器中是如何实际形成堆栈的。关键的部件是堆栈指示器。堆栈指示器是个寄存器，它含有当作堆栈“栈顶”的存贮器地址。图1.6上图表明堆栈指示器含有地址051、存贮器地址051的内容是数据D_1。当数据D_2被推到堆栈时，它就进入地址050。此时，堆栈指示器便改成读050，它表示栈顶的新地址，如图1.6下面所示。

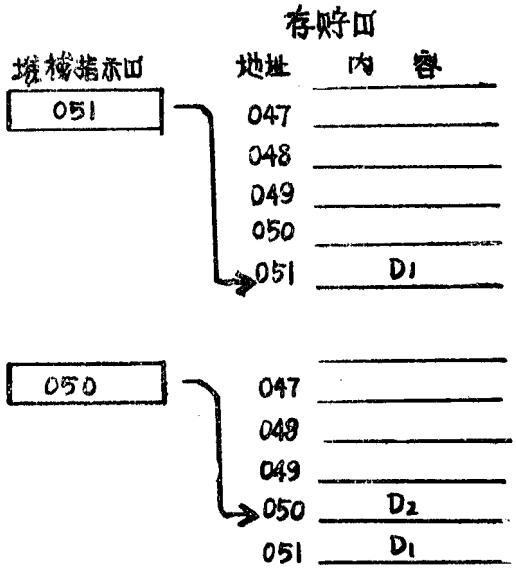

图1.6 堆栈实现

同样，当数据从堆栈上托时，可以从堆栈指示器所指出的存贮单元读出数据，同时堆栈指示器加1。

堆栈指示器所涉及的“存贮器”有可能是计算机主存贮器，也可能是较高速的缓冲存贮器。在微处理器系统中，这二种方式均可使用。

虽然以Von Neumann结构为基础的计算机系统可以实现堆栈，（正如很多微处理器系统为增强计算灵活性所作的那样，）但是在堆栈处理器中，即围绕堆栈结构的计算机中，堆栈则是基本的操作结构。

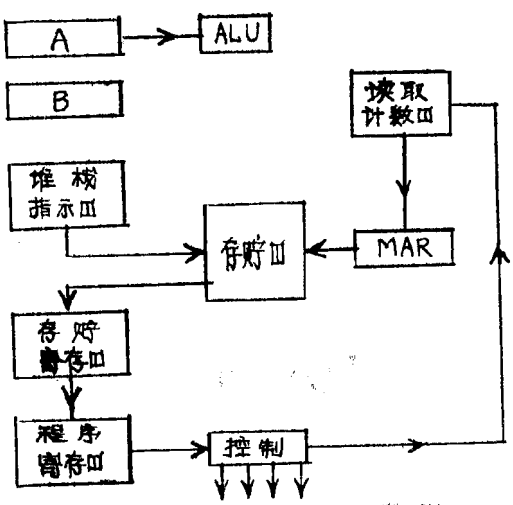

图1.7 堆栈处理器

堆栈处理器：堆栈处理器用来更有效地执行较高级的语言代码。这种特殊的方法利用了数据和指令字的特殊格式或表示方法，以供计算机执行。这种格式即所谓波兰表示法，也可以用它的变形来表示，如波兰前缀、波兰后缀和反转波兰表示法。

在堆栈计算机中，编译程序进行代码的语法分析，并把程序表示为一系列的机器操作。波兰后缀表示法实际已在

Burroughs B-5500 和 B-6500 以及 English Electric KDF-9计算机中实现。

这里作为实现堆栈结构的一个特例，讨论一下实际的堆栈在 B-6500 计算机中的实现是有益的。堆栈处理器的硬件结构如图1.7所示。与 von Neumann 计算机相似，堆栈处理器也装备了标准的存贮器和存贮器地址寄存器（MAR）。

为了同 von Neumann 计算机中的“指令”相区别，数据和控制语句在堆栈处理器中表为“字节”。字节有三种基本形式：

操作码字节

数值调用字节

文字字节

操作码字节表示将由处理器执行的特定的算术或逻辑操作。数值调用字节指出存放所需操作的存贮器地址。文字字节为实际的操作数。

堆栈处理器执行的程序由一系列字节组成，这些字节采用波兰表示法表示要执行的算术或逻辑操作。这些字节被存贮在存贮器的连续地址中。存贮器中的第一条程序字节的地址置入读取计数器中。根据内部控制脉冲，读取计数器向存贮器地址寄存器发出信号，而使特定的程序字节从存贮器中传出。

当从存贮器中传送出程序字节时，读取计数器就会加 1，同时把程序字节存贮在存贮器寄存器中。程序字节从存贮器寄存器再传送到程序寄存器，在程序寄存器中对此程序字节译码。字节一般有个二位字段，此二位字段指出这个字节是操作码，还是数值调用或是文字字节。程序寄存器里的字节一旦译码，则一列专门的控制脉冲传送到系统的其它部件，使它们同步，以完成由译了码的字节所指出的操作。

其后的控制操作以及堆栈原理用一个简单的例子就更容易说明了。考虑下面的算术表达式求值问题：

$$x = (a+b) / (c-d)$$

a，b，c，d 是给定的一组整数。

第一步是用无括号波兰表示法解释上面的一般算术表达式。虽然可以使用任何形式的波兰表示法，但是商品计算机通常采用后缀表示法，操作码（例如 add（加）和 subtract（减）被写在一对操作数的右边。在后缀表示法中，上面的算术表达式就变为：

$$ab+cd-/$$

在波兰后缀表示法中，运算一个算术表达式或这种性质的字符串的规则如下：

1．由左至右扫描该字符串。

2．记住操作数及其出现的次序。

3．当遇到操作码时做下面的事：

a．取出在序列中的最后二个操作数。

b．根据操作码的形式对其运算。

c．在下一步运算中删除这二个操作数。

d．记住 b 步结果并把它看作序列中的最后操作数。

用波兰表示法表示字符串的另个表示方法是利用二进制树形网络。代替由左至右对

字符串进行运算的规则，现在把它们用到一串分枝上。图1.8所示的二进制树形网络形式更形象地表示了操作码的先后顺序。

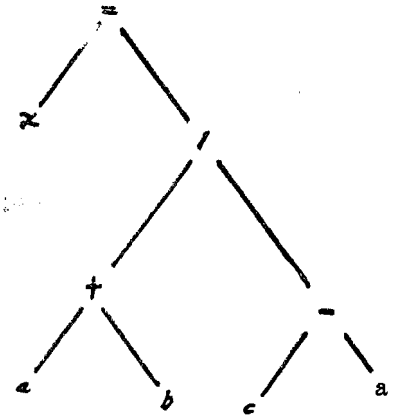

图1.8 二进制树形网络

这些操作是利用处理器的堆栈装置来实现的。图1.7所示的处理器硬件部件系指堆栈装置：A寄存器、B寄存器及堆栈指示器。A寄存器表示堆栈栈顶、B寄存器是低于A寄存器的邻近一级，堆栈指示器指出表示一条更低一级的存贮器地址。堆栈的示意图如图1.9所示。

由A寄存器、B寄存器和存贮器所决定的堆栈“下推”和“上托”动作与任何后进先出堆栈的动作相同。当一个新字节加到栈顶时，堆栈指示器就加1，B寄存器的内容被传送到堆栈指示器当时所指定的存贮单元，A寄存器的内容传送到B寄存器，而新的字节输入到A寄存器中。在退栈操作时则进行相反顺序的操作。

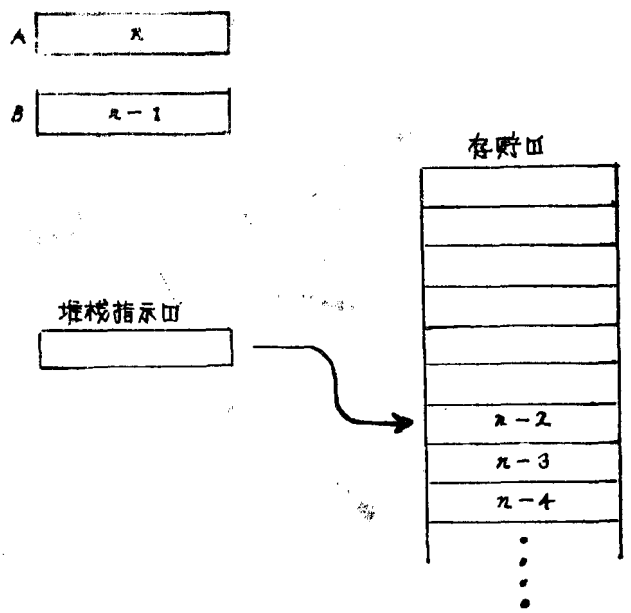

图1.9 处理器中的堆栈原理

计算机的字节串程序和相应的堆栈执行动作如下（这里，操作数存贮在存贮器中）：

程序	动作	堆栈内容 （栈顶）
1．对a进行数值调用，	a→堆栈	a）
2．对b进行数值调用，	b→堆栈	ab）
3．相加操作	$a+b=r_1$	r_1）

4．对c进行数值调用　　c→堆栈　　r_1c)

5．对d进行数值调用　　d→堆栈　　$r_1$cd)

6．相减操作　　c－d＝r_2　　r_1r_2)

7．相除操作　　$r_1/r_2=r_3$　　r_3)

于是，执行结果被放在栈顶，也就是在A寄存器中，在这里，结果可以在后面的操作中使用或者将其作为处理器的输出。

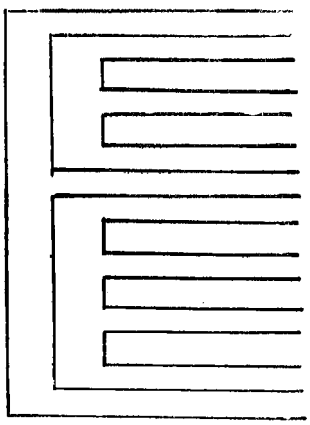

图1.10　程序块结构语言

在堆栈处理器上实现更为复杂的程序是这种基本的堆栈形成和处理技术的重复。堆栈处理器特别适用于执行“嵌套结构”语言、如ALGOL，COBOL，或PL/1。

在嵌套结构语言中，各个要执行的过程分解成为一些独立程序块的分级结构以及程序块中的子程序块，如图1.10所示。

每个程序块表示要由程序执行的特定过程。鑑于在程序块中所确定的参数和变量的不同类型，所以在各程序块间存在着差异。基本规定是，在一个程序块中，仅当参数或变量相对于这一程序块是“局部的”或“全局的”时候，才可能涉及到它。

一个参数或变量如果在一特定的程序块中被确定或说明，则称为对该程序块是“局部的”。如果一特定的程序块是一个程序块的子程序块，而参数或变量是在后一程序块中确定的，那么参数或变量相对于该特定程序块是“全局的”。

于是这个参数或变量可以用二个参数来表征：，一个是此程序或过程的词典顺序，另一个是使变量在此特定程序块或过程中作定位用的变址值或位移。因此某一变量的地址包含二部分信息：对其作了说明的程序块或过程的词典顺序L，以及使变量在特定程序块或过程中定位用的变址值或位移D。将一组显示单元或者寄存器作为指向所涉及的堆栈区段的指示字用。于是局部变量则相对于这些指示字寻址：即L表示堆栈中的一点，而D给出距离该点的位移。其主要特点在于，显示用指示字提供寻址范围，在此范围内，一地址组（L．D）由系统来解释。

嵌套结构语言，例如ALGOL，很容易在包括堆栈装置的数据处理系统上实现。堆栈包括每个ALGOL程序块的存贮区域。一个特定堆栈的每个程序块存贮区域已经与一个标记堆栈控制字（MSCW）有关。MSCW用来识别一个堆栈的特定程序块存贮区域。而且，这个程序块存贮区域中的特定数据可以通过对相应的MSCW地址相对寻址来标记。显示用寄存器指示出每个程序块存贮区域的MSCW。

数据处理系统中的堆栈装置还有二个作用。第一个作用是提供参数暂存及查询数据和程序段的手段。第二个作用是提供指示存贮程序演变过程的手段，这是由设置在存贮器中的二个清单来完成的。第一个清单是堆栈演变清单，它说明堆栈实际装入顺序。第二个清单是寻址区域清单，它系指根据所用的高级语言（如ALGOL语言）的嵌套结构规则来安排的程序块存贮区域的顺序。

微处理器结构

微处理器作为中央处理单元，也具有同大型计算机相同的基本结构。一台微处理器的基本功能结构部件是：

算术／逻辑单元

指令译码器

控制与同步单元

这些基本结构部件，在图1.11中用一个微处理器结构的大为简化的框图示出。参照这些基本结构部件，一台微处理器可以定义为一个集成电路（IC）器件，能在程序控制下以各位并行方式完成算术和逻辑运算。

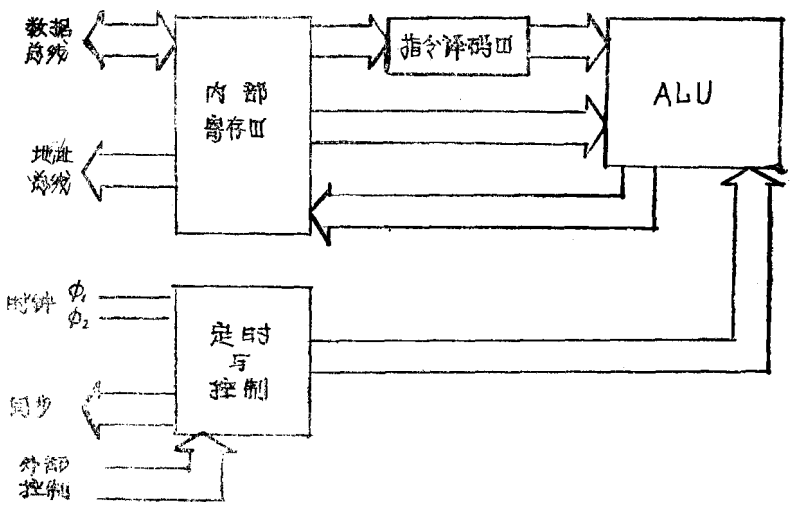

图1.11　微处理器结构

在仔细研究它的定义及框图之前，有必要讨论一下词义。微处理器一九七一年开始出现，从那时起微处理器术语一般系指上面定义的“片式计算机”的概念。早于一九七一年以及后来在少数孤立的场合中，微处理器具有不同的意思。一九七一年以前，微处理器是能够执行微指令的信息处理器，即是由微程序控制的信息处理器。微处理器一词的这种别样和陈旧的解释特别出现在学术文件上。自从一九七一年以来，微处理器在学术文件上和商业广告上都叫做集成电路“片式计算机”。

还必须注意到，有一个居世领先地位的通用计算机制造厂家已经公布了一些产品，其中包括微处理器。该公司称这种“微处理器”是先进的大规模集成的器件，其速度和密度可和主要的半导体器件制造厂家所提供的单片中央处理单元相比拟。

现在回到当今微处理器的定义上来，参照图1.11来说明其基本结构部件的功能。ALU，顾名思义，它是对存贮在微处理器的二个寄存器中的二进制数据进行算术和逻辑运算的。这些运算是用加法器以及逻辑门电路来完成的。指令译码器一般是个内部只

读存贮器，它把机器指令翻译成由微处理器执行的微指令。有些微处理器允许用户在外部只读存贮器片子上确定微指令。控制与同步部件解释微指令，以产生适当的控制和同步脉冲到系统其它部分。

微处理器一般用总线作为在系统各部件之间传送数据、地址和控制信号的手段。理由是微处理器应根据实用方便来封装，这可以使 IC 组件具有最少数目的引出脚。

微处理器某些其它的结构特点也在这里说明一下：

资源分配

存贮器的存取和传送

中断

资 源 分 配

象在任何计算机系统中的情况一样，微处理器中的一些有限资源必须由不同情况下的不同任务或用户来分配。这些资源包括总线、寄存器、I／O 引出脚和存贮器或控制程序。这些资源用分时多路传送方法来分配。微处理器的控制与同步部分确定特定的时间周期或子周期，在此周期之间允许某些操作发生。这些操作可以是处理器的内部操作，或者是有关的一些外部操作，例如从存贮器中取出数据或指令。系统设计的任务是对微处理器所给出的控制和同步信号提供译码手段，以协调外部操作。

必须认识到，由于资源分配的结果，微处理器仅在短暂的时间周期内供出信息，该周期与其它系统部件准备接受并利用此信息的时间周期不相重合。必须采用译码器和锁存器之类的辅助硬件来收集此信息并将其加到其它系统部件上。

由于资源分配以及随之而来对微型计算机系统设计的影响，产生了暂存结构的微处理器。这种结构用时间图表示，以后参照特定的微处理器还要做更详细地说明。

存贮器的存取和传送

存贮器的存取和传送结构是另外一些重要的特性，对于某些系统应用来说这些特性是关键性的。高速运算，特别是在存贮器和外部设备之间需要大量数据传送的时候，都利用存贮器存取方式，即所谓直接存贮器存取（DMA）。DMA 无须处理器的直接控制而把数据直接从予定的存贮单元传送到一台外部设备，或者从一台外部设备传送到预定的存贮单元。DMA 是用外部总线实现的，此外部总线同外部随机存取存贮器、外部设备及微处理器相连接。当微处理器不使用外部总线时，可以启动外部硬件使数据沿总线在外部设备和存贮器之间传送。

由 DMA 传送的数据量必须事先用某种方法，如用存贮器计数器来指定。起始和结尾地址也要用外部硬件来指示。唯一剩下的任务是决定外部总线何时脱离数据传送。

微处理器也象大型数字计算机一样，是在同步时序电路基础上工作的。信息处理器在预定时钟周期内完成某些操作，比如用外部总线完成读、写操作。在其他时钟周期内，处理器完成一些内部操作，由此空出总线供外部设备和存贮器使用。总线处于空闲的特定情况是用从微处理器得到的同步和状态信息来指明的。通过将同步和状态信息译码，表明总线为用户空出的信息就可以传送到外部设备和存贮器。之后，外部设备从

处理器进行“周期挪用”，并在不使用处理器周期时间内沿着总线把数据传送到存贮器。“周期挪用”的概念对于采用高速外部设备的有效的微处理器系统设计来说是很重要的。

中　　断

微型计算机系统的某些应用要求处理器直接响应一个外部条件。在这种情况下，处理器必须中断目前正在执行的程序，而开始执行一个新的程序，以处理此外部条件或“中断”。根据請求服务的外部设备的编号和优先权，有若干不同类型的中断：

简单中断

矢量中断

优先中断

简单中断只表明单个外部设备請求处理器为它服务。矢量中断提供一种手段以识别从几台外部设备之一来的中断，更详细地说，表示哪台外部设备請求服务。这种表示是用一个数据段即“矢量”来完成的，此数据段示出外部设备的标记。最后，优先中断也识别从几台外部设备之一来的中断，不过还表明哪一台设备比其他设备具有优先权。仅有少数微处理器具备这种优先权或者多级处理能力。

外部条件如“中断請求”的出现，通过外部控制线通知给处理器的定时与控制部分，这些外部控制线是与该微处理器片子的预定引出脚相连接的。定时与控制部分响应这些外部控制信号，完成处理中断程序所需要的再启动及保存操作。

微处理器软件

微处理器是根据构成软件的一系列指令来工作的，软件与执行指令的硬件是对立的。软件，即机器指令，本质上是硬件部件的替换物：例如，代替二个正使用着的特定硬件部件，一条指令可以指定某操作在一个部件上执行二次。一旦硬件供给了基本逻辑功能，任何算术与逻辑运算可以由一个指令程序来完成。

用微处理器软件代替硬件部件是微处理器系统的重要优点之一。任何硬件设备可用软件模拟，并在微处理器系统中实现。使硬件设计对于给定的应用成为不实际的种种成本上的考虑，对软件设计是不适用的，正如后面提到的，很多流行的微处理器应用都包括用低成本的程序控制的微处理器系统来代替高成本的随机逻辑硬件系统。

对于一个给定的处理功能，在软件和硬件实现之间的综合取决于该特定应用，还涉及到一些经济因素，这在下章再更详细分析。这里我们必须明了，微处理器系统必须有内部软件能力，并在系统的特定应用中体现出来。

然而，在软件和硬件之间作出定量折衷是可能的。如果我们假设：一个硬件门（比如“与非”门或“或”门）等于一条或二条指令的话，那么我们就可以用一给定应用所需要的 IC 片子数与指令数目作比较。我们用一个大致的标准——平均一个 IC 含有大约10个门来计算：在一个只读存贮器中占有 2 K 位（即2048位）的一个系统程序会取代128——256个门即13——15个 IC。

这个计算也适用于具有八位指令字长的计算机，当然必须知道，软件和硬件并不是

直接互换的，同时它们各自都有在粗略计算中所没有考虑到的自身特有的开销。但为了便于比较软件和硬件，上面的标准是合理的，从中得出的结论也是有效的。

软件与硬件相比其优点是：它是一个成本不重复项目。一旦已经设计检查了系统软件，那么它就可以在任何数量系统的存贮器中复制及存贮。这样，软件的总研制费用就分散到整个系统的生产上。硬件在每个系统中都必须复制，它是个成本重复项目。既使IC本身的成本是很小的，但是如果存在着复杂的阵列，那么每个系统都要装配和测试，这就相当于大大增加了其成本。

综上所述，软件是控制微型计算机操作的指令序列，在典型的微型计算机系统中，大量的基础程序完成一些基本的系统操作，不象在大型计算机中那样，每次机器开启都要从外部存贮器输入这些程序，这是不实际的。微型计算机把这些经常使用的控制程序存贮在只读存贮器（ROM）片子上。因此，只要用其它含有不同控制程序的ROM IC来替换这个ROM IC就可以实现程序修改。微处理器系统设计的这一特点，是在微型计算机中用软件来实现的最有意义的优点之一，

有很多不同级的软件，我们应该加以讨论。

系统执行程序

程序

子程序

微程序

系统执行程序关系到系统的全部作业、任务和数据管理。特别是执行程序规定和控制计算机系统的工作方式。计算机的各种工作方式是：

专用操作

批处理

交互处理

分时操作

并行操作

根据系统执行程序的复杂性，微处理器可以用上面任一、某些或全部的工作方式来工作。在大型计算机系统中，系统执行程序可称之为监督程序、管理程序或操作系统。

程序是为执行一特定操作或功能的一组指令。一个程序通常由一些离散的独立程序即子程序组成，这些子程序完成特定的计算或算法。

某些程序是非常基本的或者是经常使用的，这样在微程序计算机中将它们编码为微指令是很值得的。可将单个复杂的用户指令翻译成一系列直接控制硬件操作的比较简单的微指令。

很多微处理器实际上是可编微程序的。第六章简略讨论了微程序编制，而在第四章说明了几种微处理器的微程序编制特点。

小型計算机与微型計算机

微型计算机与大型计算机或小型计算机不只是在结构上有区别，在数据处理中，微型计算机完全是个新的概念。首先，微处理器是个部件，即仅仅是微型计算机系