EDN微计算机设计教程

上海市仪表电讯技术情报所

EDN微计算机设计教程

目　录

EDN微计算机设计教程

第一章　微处理机与微计算机系统的定义*

任何数字系统不管是通用计算机还是专用电路，其最基本的部件是寄存器（能存储数值的物理器件，这种所存储的数值通常用数目字表示）。寄存器的最简单形式是一根能表示一位信息的导线。

较普通的寄存器由并行闩锁组成；它能表示一个从0到2^N-1范围内的数字，其中N为并行闩锁的数目。在寄存器应该记录输入时，时钟脉冲就发信号；最后锁定的数据总是出现在输出端上。

寄存器做所有的工作

所有数字系统都能设计成一组具有本系统所需特性的互连寄存器，这个系统是用有选择地将一个寄存器的内容逐步向另一寄存器传输的方法来产生的。

数据还能在它通过某种寄存器时得以变换。在最简单的情况下，考虑并行移位寄存器，不管是由复杂的多路转换器组组成的，还是由几根导线组成的，置入这类寄存器的数据在可将它往回读出之前，就得加以修正。修正的范围可随应用情况而变动。

当置入寄存器的数据“左”移一位的时候，最大有效位（MSB）出现什么情况？将MSB就近馈送回最小有效位（LSB）的位置内，称为旋转操作。在其它设计中，使MSB移出并消失。更不同的设计是清除LSB位置到零、或在某些其它情况下，把LSB留在原来的输入值上。要注意，对于寄存器的这个简单例子来说，基本上存在许许多多的微妙变化。这些微妙变化造成各种根本不同的微处理机和微计算机。

所有计算机“都是一样的”

每台计算机都包含5个必不可少的基本单元（图1）。但是，全有这些单元的每种系统并不都是计算机；计算器具有这五个单元，但却不能算是计算机。

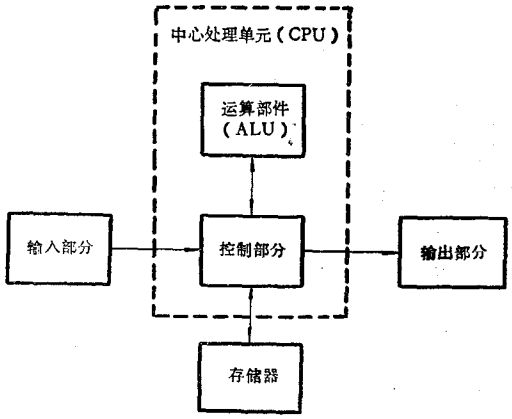

图1　所有微计算机都含有五个基本单元，而其CPU（虚线内的）通常为单片的微处理机

“运算部件”（ALU）是计算机的数据转换部分；在其输入端上出现的数据按其某种特定的计划变化。“存储设备”为以后使用提供了保存介值的手段；可将它看作巨大的中间结果。“输入部件”与“输出部件”接受供处理的数据，并将处理好的数据送出以供使用。没有输出，计算机就毫无价值。没有输入，计算机注定要反复地重复某种单一的顺序（在某种顺序控制的应用中，证明是一种有用的特性）。最后，这四个单元在“控制部分”的主持下工作。这个控制部分确定哪个寄存器在计算机内完成转移以及按什么次序转移。因此，控制部分的特性决定了整个计算机的特性（因而决定计算机的“结构格式”）。

这种按顺序的概念有什么用处呢？这在

---

* EDN μC Design Course Chapter 1: "Defining the microprocessor and microcomputer system," EDN, Vol. 21, No.21, November 1976, pp. 129～136, 杨子梅译，李政校

1936年就证实了，即只要给以充分的时间，寄存器之间转移的正确顺序就能解决任何具有收敛解的问题。这意味着，不必为每个新的应用创立专门的硬件设计。数字计算机的能力和前途就是这样：它是“一切”的机器，能解决许多问题，解决这些问题如果需要专用硬件的话就太费钱了。

计算机用户还发现，计算机不必单单用数字工作，那些数字也可代表非数字的数据。例如，数字41可代表“A”字，42可代表“B”等等。每个寄存器存有某个数据，这些数据可译成数字（即41）或另一种符号（如“A”）。因而说明了数据与信息之间的基本区别。数字41是数据。它能代表某个必须完成的操作时间数，或电阻器的值，或字符“A”。当含意附属于纯数字时，数据就变成信息。信息则是具有含意的数据。计算机处理数据；人处理信息。要了解什么计算机行、什么计算机不行的许多问题就靠这一条原则。

寄存器转移规定微计算机

微计算机由寄存器组成，而寄存器的结构特征则由能在寄存器之间完成的各种转移来形成。其次，计算机的每个分系统也由寄存器组成。

例如，存储器就是几个特征上一样的寄存器的集合（图2），每个可单独选用。因而就全部意义和用途而言，存储器可看作可选择的寄存器组。而且那些寄存器都由所采用的另一个寄存器来选择。

当对存储器组件的一个特定寄存器进行存取时，识别某一寄存器的数字地址存放在存储地址寄存器（SAR）内。SAR有N位输入容量，它由此产生2^N个输出中的一个。这种译码功能简直就是复杂的寄存器。当从存储模件读出时，选定寄存器中的数据就转移进存储数据寄存器（SDR）内。

因此，一般来说，计算机的存储组件包括寄存器的阵列（存储单元）和一对接口寄存器（SAR和SDR）。SDR的宽度称为存储组件的“字规模”。

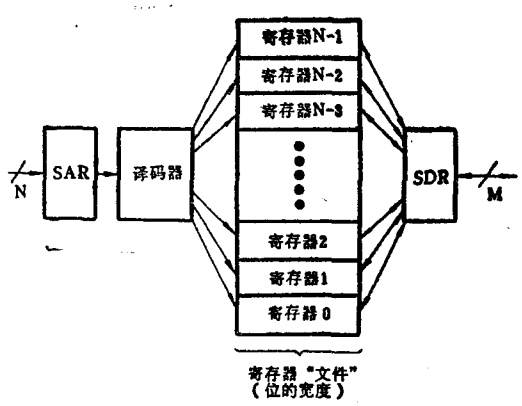

图2　任何随机寻址存储媒体的基本结构，都必须具有分别选择各个寄存器并将其内容传送至微计算机其它单元的某些手段

输入和输出皆可看作存储器结构的扩发，其主要区别在于所选定的寄存器对外界的存取能力。为了输入数据（图3），某种输入-通道-选择寄存器（IPSR）（另一种译码器）必须只选择一个寄存器。IPSR则使单个输入-数据寄存器（IDR）的内容转移到计算机的控制部分。数据的输出只不过

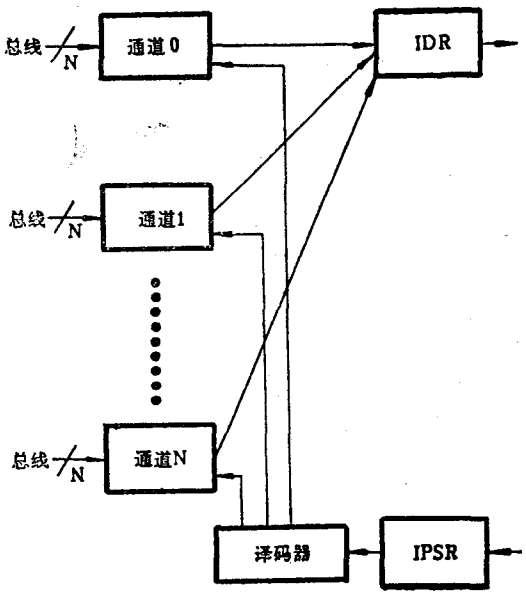

图3　输入通道都按存储寄存器的同样方法来选择。存储寄存器与输入通道寄存器之间的唯一区别，是怎样把数据存放在这里，而这对选择过程来说是次要的。输出通道的过程正好与输入通道的过程相反。

是此过程的相反过程。

寄存器的另一种集合ALU随着寄存器之间的数据转移，完成所有的内部操作。在外部，当数据送至ALU输入寄存器时，待完成的专门操作（通常表示为二进制数字）同时被置入ALU的功能寄存器内（图4）。ALU在成对的寄存器内容上典型的操作；把一个数字加在另一数字上，需要两个数据值，两者可在同时存取。

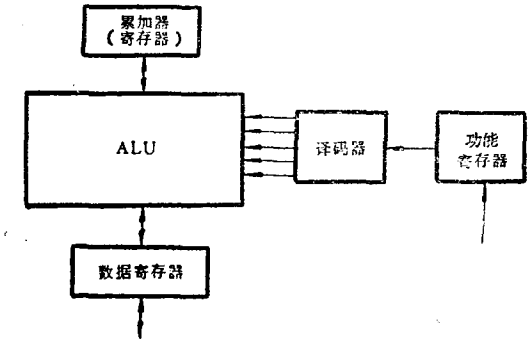

图4　尽管仍然只是寄存器的集合，但ALU却同时利用这几个来处理数据。它的位宽及它能完成的寄存器转移的种类，表示微计算机的“能力”。

通常，与ALU联系的是称作累加器的单个寄存器。累加器是一个这样的寄存器：这个寄存器已经把每次操作中的两个操作数之一的结果存入其中。累加器的规模决定计算机的字规模。8位微处理机具有8位宽的累加器。ALU一般以并行方式处理累加器内容，但不总是这样。

控制部分控制微计算机

计算机的控制部分使所有的各种寄存器都连接在一起，以完成某个所要求的一组任务。由于所有可能发生的寄存器转移都受到这个中央部分控制的影响，所以需要某种指定哪个转移该发生的方法。这种指定称为“指令”，计算机能接受的所有指令的列表定名为指令组。

一个典型的指令可以选择一个特定的输入寄存器，并接着将其内容转移至某个指定的存储寄存器。那种寄存器转移又包含其它不是直接在程序设计员控制下的、更小寄存器转移的顺序。计算机的控制部分操纵这些内寄存器转移或微指令。

通常，控制部分具有其自己的小型唯读存储器，它含有指令组中的每个指令的一个或多个字。这些“微”字又收编了专门的内部信号，而这些内部信号对于为了实现指令而需要的微小的控制个别内部寄存器转移来说，是必需的。假如能修改这个唯读存储器，则计算机具有微处理机的“可编微程序”的特性。大多数微处理机只能在制造厂里编微程序。

计算机的指令顺序带有解决特定问题的细节。这些指令存在于某种存储媒体内。大多数现代微计算机都将指令按数据形式存储在同一存储体内。

程序计数器（PC）寄存器（图5）选

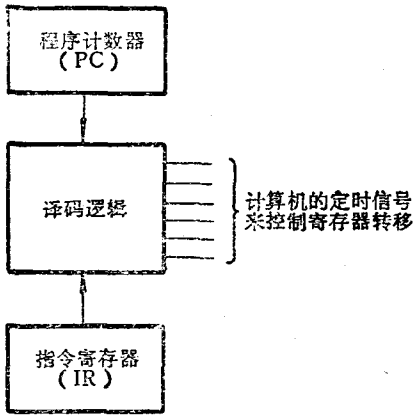

图5　微计算机的“心脏”——控制部分决定哪个动作（寄存器转移）必须出现。它的决定是根据用户输入的指令顺序和前一操作的结果而下的

择从存储器中取出并执行的特定指令。为了取出指令，首先将PC的内容存入存储地址寄存器中。当寄存器选择的物理动作（称为存取时间）发生时，程序计数器增加一位，以备首次完成时就取出所要的下一个指令。在存储器产生所选寄存器的内容以后，控制部分就将存储数据寄存器的内容转移入指令寄存器内。

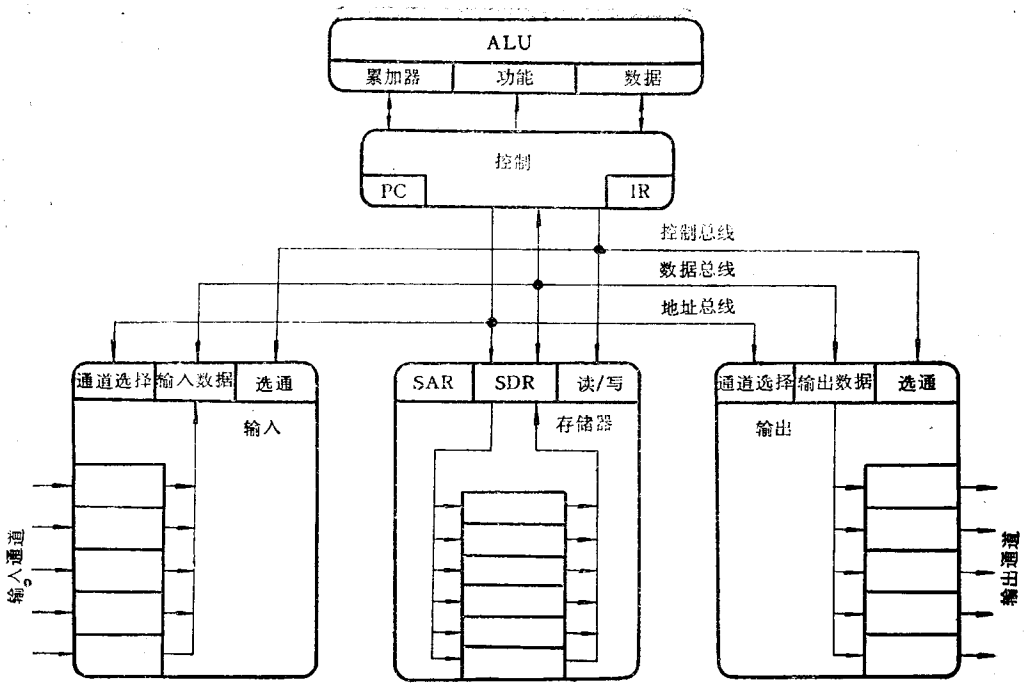

图 6　所有计算机都可表示成寄存器的集合。这个假设的计算机相当简单，没有专用的控制信号（1 位宽的寄存器）和专用寄存器，这就提高了效率和速度，而且使这一计算机与另一计算机有所不同

指令寄存器可以是具有大量“随机”逻辑的复杂类型，也可以是译指令的唯读存储器的存储-地址寄存器。不论哪种情况，其结果是相同的：控制信号的顺序有选择的将某些寄存器的内容转移入其它寄存器内。指令的末了开始下一个“取出与执行”的操作，并且无限地循环下去。

图 6 示出了将各个单元连系在一起的一个假设的计算机，为了明确，我们大大地简化了它。例如，许多控制信号必须是与某个主时钟脉冲同步发生的。可是，我们可将这些控制信号看作一位宽的寄存器，并且象处理其它任何寄存器转移那样来处理它们的移入和移出。

另一方面，有可能将更多的具有特殊功能的寄存器加到计算机上。例如，控制部分可包括用于存储中间数据的一小组“局部”寄存器(而不涉及存储器)。这类寄存器的优点包括速度(因为不必包括存储器存取时间)和效率（局部寄存器可在指令中用几位来选择，而不是用象存储地址寄存器那么宽的地址来选择）。

连总线也可看作寄存器

计算机的诸单元的互连几乎始终利用总线概念的优点。总线事实上是具有为所有正用单元的单一格式的理想寄存器。大多数微计算机有三个总线：地址总线：数据总线与控制总线。CPU通常产生供其它系统单元使用的地址总线信号与控制总线信号，而CPU与这些其它单元之间的数据交换则采用双向数据总线。当然也可以将双向数据总线组成为两条单向总线：不过，一条双向总线使数据传输所需的引线数减到最少。

计算机的字规模可能涉及存储器中可寻址单元的宽度、累加器的宽度或数据总线的宽度。由于把重点放在计算机所有单元之间相互通信上，所以字规模的三个定义有区别，数据总线的宽度一般起支配作用。

通过分析，已经提供了哪些寄存器和数据从一个寄存器转移到另一个寄存器时什么方式最有效，可更好地了解计算机中的差别。所有寄存器的组成部分愈少计算机就愈

容易了解。同时，仅有尽可能少的寄存器转移使易于了解的计算机相对地降低了效率。反之，具有品种繁多的专用寄存器的复杂计算机往往是难以了解的。但是，一旦了解了这些计算机，它们就会很有效，这是因为这些计算机完成同样的工作量所需的指令较少。

从较大程序中取出的很小指令样品（表1）可阐明这些所有的原理。所给出的程序（即指定要做什么的指令集合）是简单的：它只能说“将17加到寻址在存储器中61处的寄存器的内容上，并将结果存入存储器存储单元70内”。这个特殊程序段由三个指令组成；在另一种计算机上，它可多可少。

寻址方式选择寄存器

如前述例子所表明的那样，程序通常是指在不同寄存器内存储的各种数据值。大多来自指令的数据基准都通过寻址存储器内某一个存储单元来完成。这称为直接寻址（图7a）。指令的地址部分（可能在第一个字内或同一指令的后一个字内），存放进存储-地址寄存器内，而存储数据寄存器则接受数据（对于一个存储器的读出而言）。

直接寻址是简单的，但在许多应用中稍费钱些。如果存储器具有多达65,536个寄存器（可能用许多8位微处理机）的话，则存储地址寄存器必须是16位宽。因此，有关存储器的每个指令必须供应一个完全的16位地址。在许多小的应中，实际存储器可以只是256个寄存器长；因而只需8位的地址值。因此，使有关存储器的每个指令都含有8个无用位。

“短缩寻址”的概念克服了这种失效。短缩地址（图7b）只示出指令的有效位；剩余的存储-地址-寄存器位或者予置（通常予置到零），或者来自另一个寄存器。假如提供了短缩寻址的话，则通常作为标准直接寻址的附属品来工作。指令中的一个或多个位将指定采用那一个寻址方式，来使控制部分可产生的适当寄存器转移出现。

短缩寻址的另一种形式指设计成控制部分的部件的局部寄存器组，这定名为“寄存器寻址”（图7c）。例如，如果控制部分

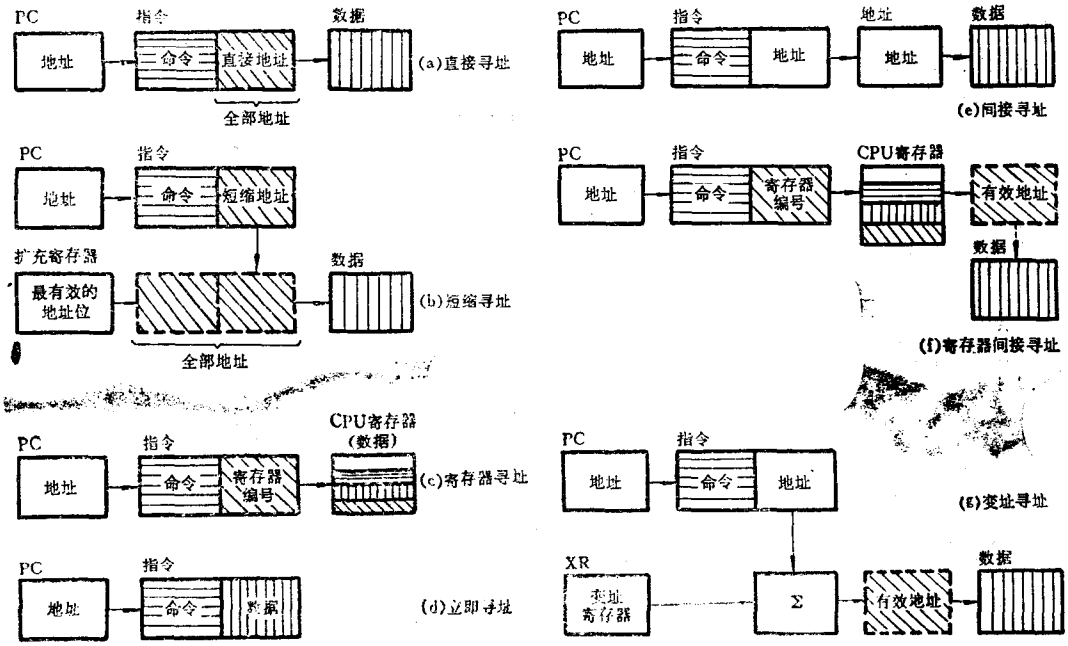

图7　在每台微计算机中，都不能包括所有的寻址方式。缺少寻址方式意味着微计算机必须执行更多的指令才能完成一个很简短的任务。

有8个专用寄存器的话，则只需选择3位中的一位来用于ALU操作中。这使指令中留有更多的位来用于指定哪个操作要完成。看来把计算机设计成仅采用局部寄存器作为ALU操作中的操作数，是相当普遍的。完全直接寻址则只将存储寄存器内容存入这些控制部分的局部存储寄存器内。

当处理象表1中例举的“17”那样的常数时，通常表明在指令中而不是在可寻址的存储寄存器中有直接可供使用的数据是有益的。这省掉了在直接寻址期间的一个额外存储器存取周期。指令内的数据合并称为“立即寻址”（图7d）。

另一种普遍形式是“间接寻址”（图7e）。在复杂级中，为了加快存取数据，立即寻址只包含一个对存储器的访问；直接寻址采用一个访问以获得由第二个访问所用的地址；而间接寻址则采用三个这样的访问。那就是说，指令指定了将要找出所需数据的地址上的那个地址。

这种寻址方式的一个特殊变形是“寄存器间接寻址”（图7f）。在这种形式中，指令访问存有存储器地址的控制部分的寄存器。因而指令通过控制部分的专用寄存器，间接地访问存储器。

“变址”这个最复杂的寻址方案（图7g），它兼有任一个其它寻址方式。在变址寻址中，专用寄存器（变址寄存器）的内容自动地加在另一地址上（按指令的一部分而提供），产生一个新的“有效”地址。变址

表　1

用户指令	微指令操作(寄存器转移)	注　　释
将“17”置入累加器	SAR←PC	寻址该指令
	PC←PC+1	增量程序计数器
	IR←SDR	取出指令
	SAR←PC	取出指令的第二个字(包括“17”)
	PC←PC+1	
	ACC←SDR	将数据存入累加器内
将存储单元61的内容加到累加器	SAR←PC	寻址该指令
	PC←PC+1	增量程序计数器
	IR←SDR	取出指令
	SAR←PC	取出指令的第二个字(包括“61”)
	PC←PC+1	
	SAR←SDR	寻址保存数据的存储单元
	ALU←SDR	取得数据
	FUNCTION←ADD	使ALU开始操作
将累加器中的目前结果存入存储单元70内	SAR←PC	寻址该指令
	PC←PC+1	增量程序计数器
	IR←SDR	取出指令
	SAR←PC	取出指令的第二个字(包括“70”)
	SAR←SDR	寻址存储单元以接受此结果
	SDR←ACC	将数据存入存储器

注：
Y←X意思是将寄存器X的内容置入寄存器Y内。

SAR：存储地址寄存器
SDR：存储数据寄存器
ACC：累加器寄存器
ALU：ALU操作数输入寄存器
FUNCTION：ALU功能寄存器
PC：控制部分的程序计数器
IR：控制部分的指令寄存器

寻址使指令能保持不修改，然而在不同的时间上访问存储器的各不相同的部分。如果在计算机的指令组中没有出现的话，变址可用完成的地址计算，然后通过计算的地址间接地访问数据的办法，以某些执行时间为代价来合成。

什么是微处理机与微计算机？

在清楚了解了基本概念之后，我们就可更精确地给微处理机下定义。目前的微处理机实际上就是用单片集成电路封装（或者几片封装）制成的计算机之ALU部分和控制部分。注意，微处理机一般不是整个计算机（它没有输入、输出及存储器），尽管也存在明显的例外。更恰当地说，具有输入、输出和存储器的单片处理机应称为“微计算机”。

一般说来，微计算机都是以微处理机为基础的计算机（因此，它们具有5个基本单元）。事实上，每种装有微处理机的产品就是微计算机。但是，当适当地编过程序且仅作为产品的一个组成部分时，微处理机就将失去其本身独特的性质。

目前市场上实现微计算机的许多方法中最有效的，一般是以单片NMOS或PMOS微处理机为基础的。可是，某些“薄片”结构在设计中更具灵活性（图8）。“薄片”方式把CPU的各种寄存器排列成存储器阵列；每片集成电路为各个寄存器保留几位（典型的2位或4位）。于是可将几块集成电路级连在一起，形成宽的寄存器。可是，这种通常比单片型更有效的器件具有高级得多的工程结构和略为高些的制造成本。

只有硬件还构不成微计算机

寄存器的内容是数值，即使是在那些数值是用于计算机本身的指令的情况下，仍然如此。计算机系统的这个理想部分——指令顺序，称为“程序”；它使特殊组合的硬件完成某项专门任务。改变程序时，同一硬件就能完成许多不同的功能。

因此，许多以微处理机为基础的产品设

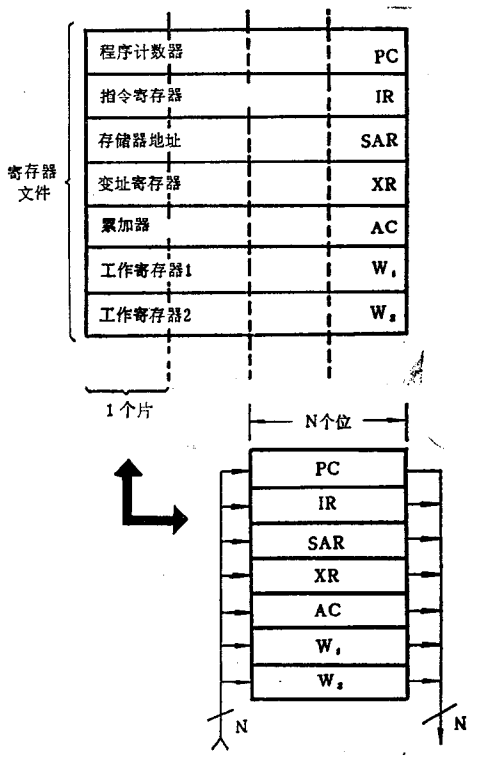

图8　位薄片结构使设计者能按基本薄片宽度的倍数制造任何“规格”的微计算机。它们的优点来源于最佳化的字长和按规格改制的可编微程序指令系统。

计事实上是难以捉摸的。程序（按其各种不同形式，从写好的说明书直到在芯片某处所存储的位）、文件编制(电子和程序两方面)及其它辅助项目全都构成计算机软件的组成部分。

厂商为用户方便提供的程序称为“系统软件”。一般，系统软件产生应用软件。

编程序是一个设计过程

编程序是一种同工程一样的复杂任务，要求有严谨的方法，来将需要译成明确的指令，以供合适的计算机使用。但是，编程序远远不是只将某些密码符号打录在冲孔纸带上，它至少包括5个主要步骤：

- 设计
- 编码
- 编译
- 检测

·调整（排除差错）

一般说来，如果设计马虎，则完成检测与调整的时间将会很长很长（从这点上说，编程序是另一种形式的工程）。计算机程序设计需要更多的分析，这比许多工程师实际完成的工作量还要多。不仅要踏实地理解问题，而且还要解决适当算法选择方面的复杂问题。这种理解问题过程的第一部分；可以确定是否需要将两个数乘起来；可是，第二部分（算法选择）则要求作更多的分析，以便从成百个用于乘法的不同算法中挑选。

程序一旦设计好了，就可进行编码。人工编码使设计的各部分变成为选定的微计算机所适用的某些指令之顺序。尽管在这过程的部分中会出现许多差错，但是它们一般是易于发现和改正的。

编码以后，要把程序从对“作者”有意义的字母和数字译成计算机能操作的“1”和“0”。幸亏计算机程序有自动完成此编译的“汇编程序和编译程序”。第十章里将对它们之间的区别下定义，并对各自进行选择的理由作出规定。

一旦把程序编译并置入存储器内后，就要对程存进行检测。作为最终目的，在将程序投入使用或生产之前，检测必须找出程序中所有可能的差错。反复地检测只“证实”程序设计员的正确程序操作设计。前一方法要求选用切合实际的检测情况，而后一方法则允许将最简易可行的检测情况作验证之用。

一旦检测出一个差错，调整过程（排除差错）就开始了。最难检测出的设计差错约占所有差错的$\frac{2}{3}$。其余为编码差错，这些差错通常是容易找出和修正的。

一旦排除了差错，就要进行重新设计、重新编码、重新编译及重新检测。这五个编程序的步骤要反复使用，直到程序达到某种合格状态为止。

附录Ⅰ μp与μp的关系

那些可编微程序的计算机往往称为微处理机（从“可编微程序的处理机，即micro-programmable processor”缩写成的）。可是这个名词往往被理解为单片CPU的意思（“微电子处理机，即microelectronic processor”）。一个单片CPU可能是，也可能不是可编微程序的。

通常，可编微程序的处理机设计以比目前单片CPU的设计规模大得多的形式出现，因此不应有多大的混淆。然而，技术正促使单片CPU提高到更大的结构，而可编微程序处理机的设计人员却缩减下来了。

所以需要注意运用微处理机这个名词：它的真正含义可能与你所想的有所不同！在本教程中，当我们谈到微处理机时，我们指的是一般含有ALU和控制部分的单片类。

附录Ⅱ 中心处理单元

计算机的运算部件和控制部分，一直传统地做在一起。早期，容纳这两个主要计算机单元的单个大中心柜称为中心处理单元（Central-processing unit，缩写为CPU）。把输入、输出和存储器都视为其附属品。

现在通常把单片微处理机称为“CPU”。就微处理机单元而言，某些作者宁可把大型中心处理单元与微处理机区别开来，因而称后者为“MPU”。

第二章　内寄存器与外寄存器的连接*

在深入微计算机的内部结构之前，设计者必须了解这些处理机操作的环境。如第一章所表明的那样，所有计算机都由5个基本单元组成：在微处理机芯片上组合的控制部分与ALU部分、外存储器、输入、输出。完整的微计算机系统在图1中阐明。

总线操纵微计算机的通信

计算机各单元之间的通信在一些标准总线上进行，地址总线、数据总线和控制总线各一条。微计算机从存储器读出，例如通过把所需的存储单元地址存放在地址总线上，并在控制总线上建立正确的信号（例如读出信号和存储器信号均为高电平）。在适当延迟存储器-存取时间并稳定之后，它采取数据在双向数据总线上出现的形式；这是由于微处理机开始了读出操作，存储器必须按照数据指定来激励微处理机的双向数据总线。

所有微计算机都有这三种总线，虽然在形式上其差异很大。有时，必须过细地找出所有这些信号。地址总线的宽度通常是与整个存储器寻址所需的宽度一样宽；可是，某些微处理机（特别是早一代的）将芯片以外的地址总线多路转换至外闩锁内。

这些多路转换处理机需要一组供多路转换操作用的附加控制信号，但节约了引出线数目，这些引出线实际上将地址输出送到总线上。当然，多路转换系统需要额外的外部硬件来闩锁存储器地址（把微计算机的地址总线看作除了非多路转换闩锁以外的系统的一部分）。

通常，为双向操作安排了系统的数据总线，尽管先前提到过，也可用两个单向总线来做成。某些处理机与大量存储器接口，而输入/输出能力不需要额外的缓冲。但是，凡是激励能力有限的大系统或微处理机，都可能需要某种额外的缓冲。这种缓冲的结构决定了选择单向总线排列还是双向总线排列。第五章里将探讨缓冲的设想。

图1的控制总线也许是最容易想象到的。大多数系统要求有多得多的控制信号来使分立的微计算机单元同步地动作。可是，这个简单例子只提供两种基本信号：读出/写入信号和存储器/输入输出信号。在数据总线上的数据向微处理机传送时（读出），读出/写入信号达到高电平，而在输出有界数据时（写入），读出/写入信号转为低电平。存储器输入输出信号选择存储器或者I/O器件参与数据的传输。

这些脉冲的宽度以及它们之间的定时可以使微计算机设计容易或困难。

信号与主时钟脉冲“协调”

微计算机与大多数的数字计算机一样，都是按同步方式操作的。内寄存器之间的数据传输和不同微计算机单元的寄存器之间的数据传输，都是由从主时钟脉冲电路中导出的主时钟脉冲信号与控制信号来同步的。在设计良好的系统里，该技术防止了两个不同的寄存器或计算机的其它部件同时激励同一条总线。如果计算机的控制部分起“乐队指挥”作用的话，那么必然把定时图看作为“乐谱”。

定时图能极方便的表达微计算机的三条主要总线之间的相互关系。事实上，定时图对了解微计算机的操作是很重要的，以致每份规格说明书都包含一个定时图，尽管在形

---

* EDN μC Design Course Chapter 2: “Connecting internal and external registers requires “good” timing”, EDN, Vol. 21, No. 21, November 1976, pp. 147～15, 2杨子梅译，李政校

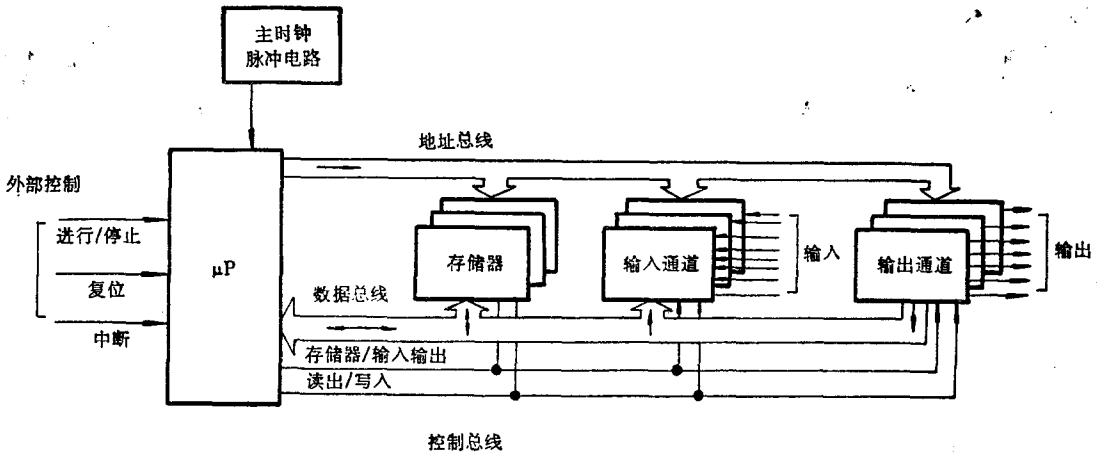

图1　典型的微计算机方案，尽管相当原始，却能选择任一个I/O通道或存储器寄存器，并从它读出或向它写入，虽然与其更复杂的“亲属”相比，效率和速度都是较低的

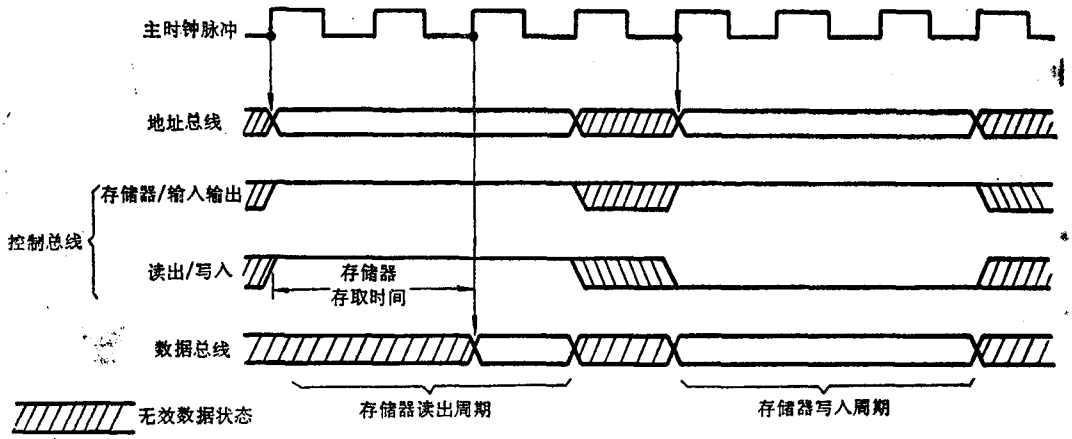

图2　定时图基本上用于了解地址总线，数据总线和控制总线之间的相互关系。它们表明各种微计算机元件之间的同步程度，并指出临界的定时范围

式上往往不必是复杂的。定时图基本上规定了在什么时侯寄存器之间传输数据。图2示出了读出和写入存储器的简单例子。

典型情况下，总线信号的所有边缘都与主时钟脉冲的某部分同步。尽管大多数系统采用多相时钟脉冲，但我们选择了单对称相位时钟脉冲。在此假设的计算机中，所有存储器操作都在连续的三个主时钟脉冲周期内发生。在某个时刻上（这里表示为主时钟脉冲的前沿），微处理机使地址总线和控制总线发挥作用。所需的存储单元规定在地址总线上，但I/O通道与存储媒体仍不理该总线，直到受到专门指定而接受地址为止。在这种情况下，选择了存储器（存储器/输入输出在地址总线时间的期间为高电平）。此外，读出/写入线规定了数据传输的方向。

对于从存储器读出而言，在出现地址信号、读出信号和存储器信号与处理机希望在数据总线上找到数据时的时刻（图2中第三个时钟脉冲的前沿）之间的时间，称为所需的存储器存取时间。整个存储器子系统（其中包括该计算机单元的所有内部缓冲、译码及控制）必须在指定的时间内完成一个读出周期。如果存储媒体发挥作用太慢的话，则为了与之适应，主时钟脉冲就必须减慢下

来。

读出系统（与读出定时图）通常比这里所示的更为复杂。由于内部的时间延迟，微处理机能在不同的时间上描述各种控制信号和地址总线信号。此外，控制信号可能不与设计进微计算机内的存储器器件完全相容。这种情况需要某种外部控制逻辑把来自微处理机的信号转换成与存储媒体相容的信号。

I/O通道与存储器选择几乎可以完全等同看待，只是I/O通道的存取时间一般较快些，这是因为I/O通道不包括大的阵列和地址译码。但是，定时的原则保持不变。

多种总线传输信号的方法

在至今使用的例子中，微处理机产生两种基本信号：读出/写入信号和存储器/输入输出信号。事实上，不同的微计算机产生不同的信号。不过，设计人员在经历了许多不同的微处理机设计以后，几乎总是采纳4线系统。其四个信号是：

/MEMW（有源-低电平存储器写入）

/MEMR(有源-低电平存储器读出）

/IOW(有源-低电平输入/输出写入）

/IOR(有源-低电平输入/输出读出）

这些信号与其逻辑意义来自普通现成芯片的实际估计法。典型的存储器和I/O部件具有有源-低电平的芯片-选择信号。同样，数据总线的缓冲器需要不同种类的控制器来保证正确的数据传送方向；它们通常大都是负逻辑输入。由于微计算机通常在双向数据总线上按各个方向单独的缓冲，故最好有按照单独的线来读出控制信号与写入控制信号。

即使是在这四种信号不正常地出现的计算机中，设计人员也已找到了最有效地包含它们的方式。例如，在8080A中，设计者可容易地导出它们，或采用一个Intel公司的辅助部件。另一方面，Motorola公司的6800则采用RD/WRT及有关的信号。然而，采用有效控制信号来产生四个标准信号，却显著地减小了用于控制操作的辅助逻辑总量。

微计算机保持什么“时间”？

不同的微计算机需要不同的时钟脉冲。某些系统（象Intersil公司的6100、RCA公司的COSMAC和Fairchild公司的F8等）不需要外部时钟脉冲。晶体或RC网络把工作频率确定在某个规定的限度内。其它情况（当然是最平常的）需要一个具有非交迭脉冲的2-相时钟脉冲系统（图3）。在这种系统中，2-相时钟脉冲是不对称的，因此，这两个相位不能通过简单地除以2而产生。还有其它一些微计算机可能要求单相，至于National公司的GPC/P（与IMP-16系列）和Texas仪器公司的TMS9900则需要四相。

多数设计者着眼于上限操作频率。但是，他们也应考虑下限操作频率。某些系统（诸如CMOS等）是静态的，并且在与可产生的时钟脉冲频率一样低的时钟脉冲频率上操作。在系统调试期间，有用的静态系统可使计算机速度减慢到可监控个别机器周期的程度。

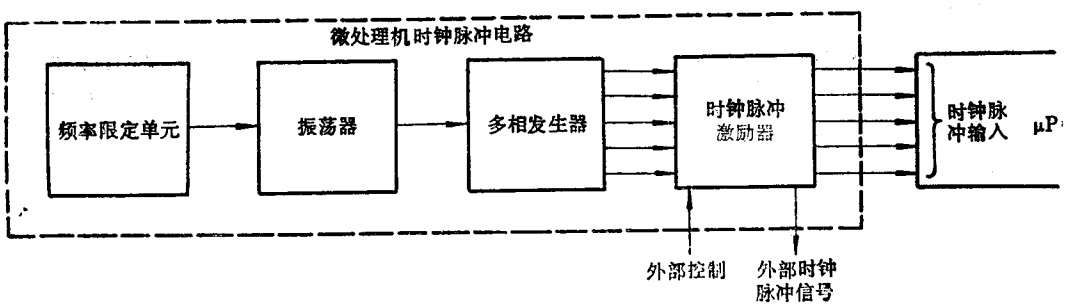

图3　大多数主时钟脉冲共享这些基本元件。要减掉什么或加上什么，必须最后取决于特定的微计算机。含有全部所需时钟脉冲电路的单片集成电路(虚线框内)可从很多厂买到，这种集成电路可用来补充专用的微处理机

但是，多数微处理机是动态的，而且要求充分快速的时钟脉冲来保证微处理机寄存器的重复更新；这就避免了数据的损失。在采用一个简单的时钟脉冲时，可证实最高操作频率与最低操作频率之间的差是重要的。主时钟脉冲频率的漂移可能超过上限操作频率，也可能超过下限操作频率，并会因此而引起某些产品的偶而失效。

多数微处理机采用晶体控制的时钟脉冲。虽然指定的操作频率可以变化（通常，某些设计者采用奇数振荡器将晶体“拉”离频率），但稳定性不能变化。按软件完成的任何定时使晶体控制成为绝对必要的。然而，在某些外部硬件（它可保持稳定的基准）控制下完成的定时，晶体控制就不必那么严格了。

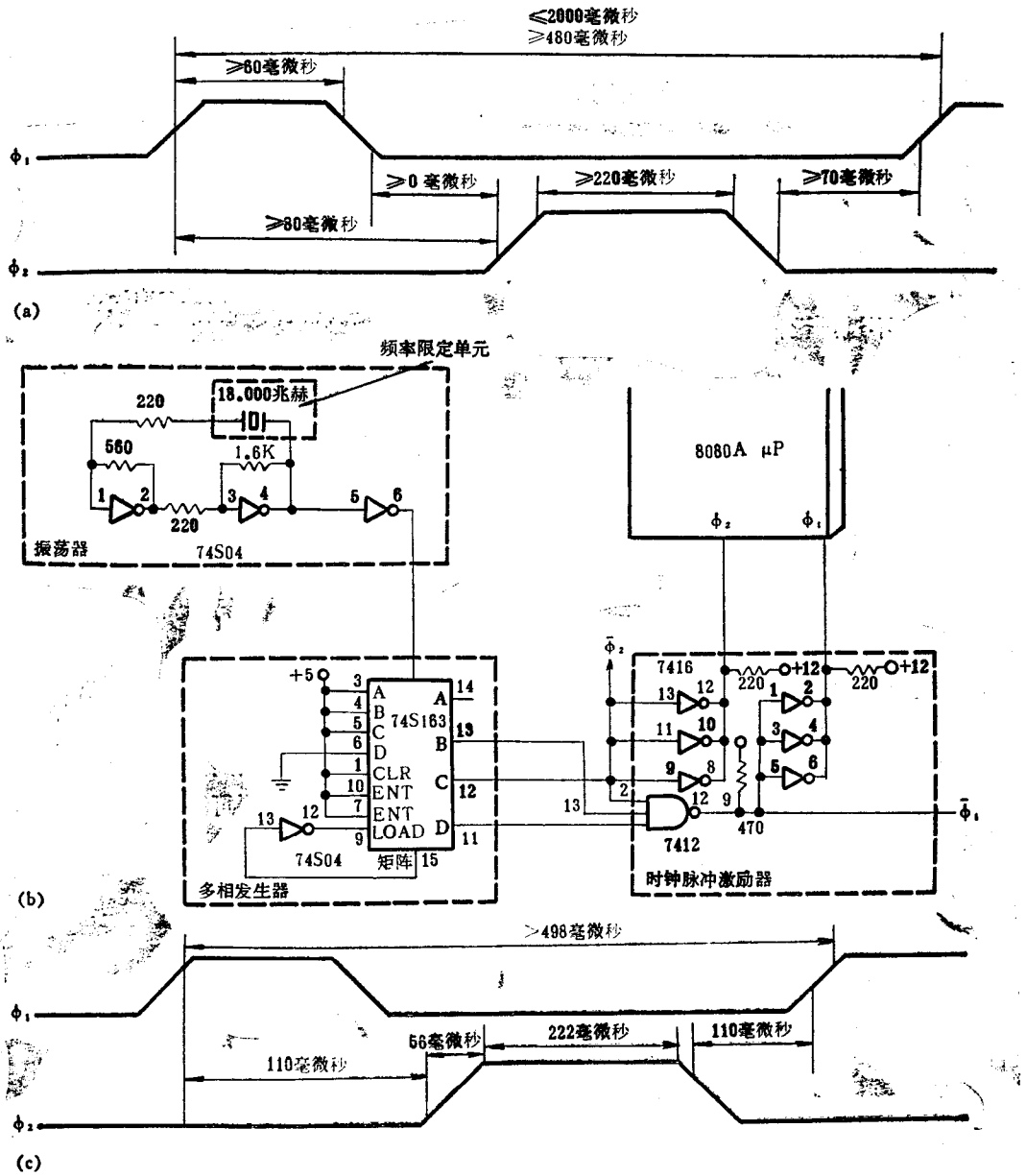

图 4　用于8080A微处理机的完整主时钟脉冲。(a) 8080A定时图，(b) 与图 3 的主要单元相一致的电路设计，(c)用于该特殊微处理机的可接受的时钟脉冲定时图

在选定基本振荡器设计之后，设计者必须选择一个适当的操作频率，并且导出所需的相关相位。非常简单而廉价的设计是通过各种不同振荡器（这些振荡器全部“拧”到适当的频率和脉宽上）来产生这些相位的。

但是，这种设计由于“拧”的过程有高的相互干扰特性，故增加了生产时间。在部件方面节约的钱(由于RC控制振荡器成本较小)比生产线调节需要所费的钱多。多数设计把主振荡器频率直除到某个公分母为止，然后组合这些信号以产生不对称的脉冲。

例如，Intel公司的8080A以500毫微秒的主时钟脉冲周期和两相来操作。图4示出在此频率时，操作所必需的时钟脉冲规范。产生这两个脉冲的最简单方法，是从18兆赫振荡器开始，并将它除以9，以达到500毫微秒周期(2兆赫)。然后，组合的逻辑（也示于图4）就产生时钟脉冲相位。聪明的设计者总能找到某个较高的操作频率（这个频率直除到产生所需比例的时钟脉冲信号）。

负载影响时钟脉冲定时与波形

微处理机在时钟脉冲输入端上出现大的动态与静态负载。实际上内部电路的每个部分都与某个或另一个时钟脉冲相位同步；要保证正确清晰的数据传送，要求仔细控制时钟脉冲的波形。为防止芯片上时钟脉冲降低，需要某些较高的功率级和电压电平。几乎每个主要半导体制造厂都制造一个时钟脉冲激励器集成电路的系列（经常略多于封装内的大功率晶体管）。例如，8080A的时钟脉冲必须摆动12伏。激励高达25微微法的电容性负载，而且仍保持50毫微秒的上升时间和下降时间。

幸运的是，时钟脉冲电路设计现在大多是学术性的。兼容的时钟脉冲激励器可从许多微处理机制造厂买到，也可以从其它来源取得。Intel公司用于3080A的8224时钟脉冲发生器和激励器，是一个16管脚的单片集成电路，它一方面接受晶体（18兆赫），而另一方面产生所有重要的时钟脉冲信号。设计者可挑选Motorola公司的K1117时钟脉冲系列供8080A使用。该器件是一种内装晶体的24管脚双列直插式的混合封装。Motorola公司的MC6870时钟脉冲系列是为6800的用户设计的。

定时可能需要不同的电平

不管时钟脉冲电路是自产的，还是从半导体公司或从其它地方买来的，设计者必须知道该时钟脉冲电路可能需要或接受的其它信号。某些微计算机需要TTL-电平时钟脉冲信号来产生用于控制总线的信号。时钟脉冲发生器应产生这些信号。此外，TTL电平时钟脉冲还需要延迟电路来使它们与微处理机时钟脉冲输入端上出现的高性能脉冲同步。

在共同功能必须与时钟脉冲同步的地方，将那些功能插入时钟脉冲电路内的办法，经证明是最佳的。对简单的同步事项来说，要考虑到时钟脉冲发生器芯片需要多少外部电路。打算与特殊微处理机一起使用的多数时钟脉冲电路，都具有早已包含共同同步问题的输入和输出。

许多时钟脉冲电路还具有“延伸”容量的特征。通常在分立的管脚上，当被激发时，它延伸时钟脉冲中的一个超出标准周期。这个功能证明，当微处理机不能为特殊存储器芯片延迟时，在动态系统中它是极为有用的。但是，当采用具有时钟脉冲延伸的慢速存储器时，恰当的时钟脉冲相位却及时地延伸，以便允许进行适当的存储器存取。当没有选择那个存储器组件时，时钟脉冲以更高的操作频率工作。当然，必须注意，要防止长期压制该“延伸”控制时钟脉冲电路，以致使CPU因未能更新寄存器文件中的动态RAM而失去其“记忆”（即寄存器内容）。

总线系统是每个微计算机所独有的

关于三种基本总线的所有一般论述，对所有流行的微计算机都适用。不过，每个微计算机是不同的，而且具有各自的一组信号和定时法则。因此，必须仔细的研究参考手

册，以便精确地确定怎样把各个处理机与所提供的信号接口。应将注意力集中在下列主要方面：

- 从存储器读出
- 写入存储器
- 从输入通道读出
- 写入输出通道
- 重要控制的同步
- 至处理机的信号

当然，如果你象寻址存储器存储单元那样寻址I/O通道的话，则不必把输入/输出定时与存储器定时分开来考虑。

来自存储器操作的读出，表示在地址总线上的地址，并且接着使某个控制信号升高（或降低），来起存储器读出的选通作用。该控制信号或者可规定控制缓冲器的数据传输（向处理机）的方向；或者规定可采用的控制线的分立方向。

写入存储器是类似的，只是为了适应某些普通RAM的独特特性，写入选通的定时与读出选通的定时有所不同。

通常，输入/输出指令产生定时信号与控制信号（这些信号与存储器使用的信号大同小异）。然而，由于通道是由简单的闩锁或缓冲器组成的，故某些处理机为输入/输出产生的定时与为存储器产生的定时有所不同，以避免不必要的等待不存在的周期时间。

许多微处理机具有必须与各种时钟脉冲相位部分同步的控制输入。例如，在某些流行的微处理机中，至CPU的RESET（复位）信号必须与这些时钟脉冲相位之一同步，并保持这些时钟脉冲周期的某个最小量。不调查研究这些要求，可能导致偶然的和非重复的设计差错。

附录　逻辑常规

所有制造厂都用“1”和“0”来表示工业标准逻辑电平。不幸的是，大多数标准到此为止。在某一制造厂将逻辑1列为正的真值时，另一个厂可能把它定为负的真值。此外，单信号线可用任意数目的加强助记的字母表来标明。

为了易于理解和一致起见，我们为该微计算机设计教程采用下列常规：

- 凡功能都用它们的英文字母拼写出来（READ，WRITE，INOUT等）
- 保持简单的缩写形式（SAR＝存储地址寄存器）
- “1”表示正的真值(有源-高电平)状态，“0”表示负的真值(有源-低电平)状态。
- 斜线把共享同一条信号线的正和负真值功能分开（READ/WRITE意味着当线为高电平时，完成读出功能，当线为低电平时，完成写入功能）。
- 有字顶横线或斜线的表示负的真值功能；没有字顶横线或斜线的则是标准的正真值功能〔/读出或$\overline{\text{读出}}$(/READ或$\overline{\text{READ}}$)** 意即在有源低电平时读出〕。

** 原文有误— 校者注。