PDP—11小型计算机程序设计

北京航空学院706教研室翻印

一九八三年三月

目　录

第二章　程序设计初步

第三章　PDP—11的组织和结构

第四章 程序设计技术

第五章 数据结构介绍

第六章 I／O 程序设计

第七章 系统软件

第八章 操作系统

前　言

当一个人使用独立的小型计算机时，往往会产生某种自我陶醉之感。整个计算机处于他的绝对控制之下，自以为他什么都能干，什么都不妨试一试，这种感觉不禁油然而生。我敢说，研制和使用第一批计算机的老前辈们一定也有过同样的感觉。据我所知，今天的系统程序员们大多还有这样的感觉。

也许有人会说，这种感觉已属往事。如今到处都是“不开放”的计算机，再谈什么自我陶醉，恐怕不大合适吧！他们还会指出，普通计算机用户十分满足于用高级语言来编写程序，他们关心的主要是计算机的程序设计。如果允许他们对计算机硬件、软件和系统设计的许多方面一无所知，那末他们是相当高兴的。

本书确实不是为这些人写的。有些人曾为计算机所陶醉，这种感觉促使他们要探索隐藏在计算机内部的奥妙，他们才是本书的对象。这样的计算机用户（系统程序员、系统设计师、计算机科学家、电气工程师、应用专家）渴望懂得一些计算机组织和汇编语言程序设计方面的知识，我这本书正是为他们编写的。

前几年我曾用许多不同的计算机给各种水平的学生讲授计算机组织和汇编语言程序设计，教学实践使我得以编写这本书。在讲授过程中，我总是力求阐明计算机硬件和软件的基本概念，特别是那些对计算机系统及其组织的重大课题有所影响的思想。为此目的，我曾尝试过各种方法，总是发现，若想讲清提出的概念，使用一台假想的计算机远不如使用一台真实的计算机。

小型计算机价格便宜、容易买到，用来作为教学用的“真实计算机”似乎最好不过。这些机器具有以前只有较大系统才具备的特点（如，重迭操作、中断、多寄存器、内部乘除功能等），现已广泛地普及，并成为许多系统不可分割的一部分。所以说，这种机器十分适合于本书的需要。我假定本书的读者都学过面向过程的语言的基本课程。目前FORTRAN已被普遍接受，而且具备各种语言的共同思想，因此我将用这种语言来说明许多例子。事实上，用一种高级语言（FORTRAN）来说明一种低级语言（PAL—11）中的同样思想，这种作法的意义不仅局限于以此例彼。而且由于程序设计的原理和技术在很大程度上独立于语言的实现，各种各样的语言都可以用来清楚地说明一些新思想。结果，给本书读者带来的好处是，在学习PDP—11程序设计某些细节的同时，还能十分自然地建立起数字计算机和程序设计的基本概念。

为什么我把PDP—11选作我的真实机器呢？这是几个原因造成的：（a）它使用普遍，（b）它有许多特点，（c）为它已配备了为数众多的系统。我企图首先提出一种概念，而后就PDP—11加以说明，而不是反其道而行之。我相信，没有计算机的用户和有不同种类计算机的用户，能够弄懂我所提出的概念，并把它结合到自己的实践中去。

本书首先讨论计算机原理，叙述的方式相当独立于任何特定的计算机。这部分的题目包括计算机器和计算机组织、简单的记数系统及其概念、逻辑操作和计算机构筑块。第2章论的是程序设计原理，包括框图和程序的编址、符号式编码、寄存器及其用法、以及数字计算机常用的各种编址方式。

第3章讨论PDP—11的组织与结构。为了说明计算机是怎样操作的，我举了一些例子。第4章研究程序设计技术，包括与位置无关的程序设计、子例行程序和对等例行程序、重入性和递归、算术运算的程序设计、浮点数表示和字符处理。

在计算机中信息的表示十分重要，因此我专门开辟了一章（第5章），介绍数据结构。这章以FORTRAN数组的讨论作为开场白，而后又描述了堆栈、堆架、队列、列表、杂凑法，以及用来表示和存贮数据的压缩技术，並且用了一些FORTRAN程序和汇编语言程序解释了这些概念。

如果我们没有办法给计算机提供数据，或者从中取出结果，那末即使我们会编程序，也是毫无意义的。所以第6章就专门来谈I/O程序设计，从基本细节说起，一直说到中断驱动的设备。另外，为了有效地利用I/O设备，並帮助程序员编写要求输入输出的程序，计机厂家往往提供一种系统软件——IOPS（输入输出程序设计系统），这章最后一节对IOPS作了较为详细的说明。这样一来就十分自然地过渡到下一章的主题——系统软件上了。

第7章简要地介绍了小型计算机几乎都有的各类系统软件。讨论的题目包括编辑程序的功能、宏汇编程序的灵活性、联结与装配过程，以及联机动态排错设施。

本书具有一个异乎寻常的特点，就是论述了操作系统（第8章）。在讨论了计算机组织、汇编语言程序设计、硬件特点和系统软件之后，本章作了个总结，並把前面提出的思想和概念统一起来。首先我说明了，为什么要有操作系统。然后考虑一个通用系统需要哪些功能，如何提供，这样就会形成一个初步的理解。

为了加深理解，巩固记忆，本书补充了好些例子和习题，还列举了一批参考文献，供进一步学习之用。这些例子和习题来处很多，其中包括我在编写本书期间提出的课堂设计。

我打算让这本入门性的教科书作为一级阶梯，使读者得以向计算机科学中更高深的专题攀登。因此我力求给读者提供进一步学习计算机系统所必要（而充分）的知识。我的目的并不是要教会读者怎样为PDP—11计算机编程序，怎样操纵它；尽管我相信，读者将会熟练地使用它。我希望，读者学过本书之后，确实能感到自己掌握了计算机部件的一些基本思想和计算机系统组织的一些原理。

Richard H. Eckhouse

第一章　计算机基础

1.1.介　绍

在过去的二十五年中，计算机的发展，剧烈地改变着我们的世界，可以予见，今后将引起更为巨大的变化。

今天的通用数字计算机，比起早先的计算机，快得多、小得多、更可靠、造价更低。新型工艺、多种多样的结构，以及快速的存贮器，都给计算机带来极大的影响，然而在许多使用了计算机的新方面，已出现了更为重大的突破。

第一批大型电子计算机一般都用作能力极强的计算器，能够解决以前无法解决的复杂的数学问题。近些年来，计算机程序员开始把计算机用于非数值应用中，如：控制系统、通讯、人工智能、图形识别和数据处理。在这些方面，计算机以极高的速度处理大量的数据。

1.1.1. 计算机提出的问题

有人说过，可以为计算机编制程序，来解决任何能够定义的问题。此处“定义”一词极为重要，它的意思是：问题的求解过程可以分解成一些步，每步又能编写成一个计算机指令序列。有些问题是很难定义的，如自然语言的翻译。几年前，人们还认为，可以编写出能把法语翻译成英语的计算机程序。实际上，把一批法语单词翻译成意义相近的英语单词，是非常容易的。但是由于每个单词和词组都有多种不同的意义，所以很难正确地翻译句子。

虽说自然语言不适用于计算机，但是程序设计语言（如BASIC和FORTRAN）具有精确定义的结构和语法，大大简化了同计算机的通讯。程序设计语言是针对问题的，包含了人们熟悉的词句，因此人们经过不长时间的学习，就可以用程序设计语言来编写程序。因为计算机厂家大都采用标准的程序设计语言，並在他们的机器上实现了，所以编好了的一个程序总能在许多计算机上执行。小型计算机的程序员往往用 FORTRAN 语言来解决科学和工程问题，而用 BASIC 来完成较短的数值计算。另外还有许多种程序设计语言，可用于程控机床、计算机排版、作曲、数据采集、计算机辅助教学等领域。将来可能出现更多的新型程序设计语言。每发展一种新语言，都使用户更容易把计算机的能力应用到他的特定问题和任务中。

什么人能作程序员呢？在计算机程序设计初期，程序员差不多都是数学家。然而正如本书所述，程序设计差不多只需要些算术操作和逻辑操作方面的基本技能。也许程序设计最基本的要求是逻辑推理的能力。

在过去的十年中，计算机领域迅速扩大，使成千上万的人能够使用计算机资源，而且也给许多人带来了施展才华的机会。

1.1.2.计算机应用

计算机象别的机器一样，能够承担某些任务，比人干的更好、更有效。具体地说，它比人接受的信息更多，处理得更快。它的速度极高，人用纸和笔需几周或几个月才能完成的工作，计算机往往只用几分钟就够了。因此只要使用计算机而节省的时间足以抵销自身的价格，就应使用计算机。再有，因为计算机能够在极短的时间内处理大量的数据，所以在时间紧迫的情况下，它是解决问题的唯一手段。由于计算机速度高、容量大，在商业、工业和科研等方面使用得越来越多。计算机的应用大多可以归入两类：商业应用——要求计算机能够存放和快速检索大量的信息；科学应用——要求以较高的精度和速度来完成数字计算。这二类应用都可在通用计算机上完成。下面我们提出几个计算机应用的例子：

设计问题．对于设计工程师说来，计算机是一种十分有用的工具。例如，超音速飞机的翼型设计与许多因素有关。设计师可以使用程序设计语言，以数学方程式的形式，来描写这些因素。然后就可以用计算机来解这些方程了。

科学实验．在科学实验中，可以使用计算机保存来自各种电子检测设备的信息，并加以计算。特别是在遥测之类的系统中，信号必须很快记下来，不然就会丢失，如果使用计算机，就不难作到了。这些应用都要求迅速而准确的处理，无论条件是固定的，还是变动的。

过程控制．计算机是产品的自动生产和检查的有力工具。计算机可以操纵轧钢机、机床和其他机器，它对外界条件的反应又快又准确，这是人所不能比拟的。可以让它在制造产品的同时使用特殊的敏感元件检查产品质量，必要时适当调节机械设备。如果来料有缺欠，它可以拒绝接收，並采用下一个。

模拟式训练．在实际条件下训练大批人驾驶民航客机、控制人造卫星，或操纵宇宙飞船，显然花钱太多，又很危险，而且不现实。计算机能够为受训者模拟这些条件，並对他的动作作出反应，而且还能报告训练的成绩。这种作法既不冒险，又不使用昂贵的设备，就能使受训者接受长时间的实地训练。

上述应用往往要求对数字信息和模拟信息进行处理。模拟信息由连续的物理量所组成，容易产生，容易控制，如电压、杠杆的转动等。而数字信息由离散的数值所组成，能表示问题中的变量。通常模拟量要先转换成相等的数字量，才好完成算术运算，最后解决问题。有些计算机（如LINC—8）已把数字特点和模拟特点结合在一个计算机系统之中。

1.1.3.计算机的能力与限制

计算机同其他机器一样，若想完成一个任务，须受到指挥和控制。在编好程序，並存入计算机的内存之前，计算机一无所知，甚至如何接收输入也不知道。一台计算机无论质量多么好，也必须告诉它，让它干什么。因此说，在人们认识到计算机的能力（和限制）之前，计算机不可能充分发挥作用。计算机的能力和限制有：

1. **重复性操作**：计算机能成千上万次地完成类似的操作，不会烦燥、不会疲倦、不会粗心大意。

2. **速度**：计算机处理信息的速度极高，究竟多高，与设计师和程序员的才智直接相关。现代计算机在解决某些类型的问题时，可比熟练的数学家快一百万倍。

3. **灵活性**：通用计算机能用来解决许多类型的问题。

4. **精确性**：按照程序员的规定，计算机可使计算的结果达到令人满意的精度。

5. **直观能力**：计算机不具备直观能力。有时人们可能不经过详细的推导，实然得到问题的答案，可是计算机只能按照要求一步一步地进行。

本章其余篇幅来谈计算机的一般结构及处理数据的方式，主要包括计算机的基本组织和结构、程序设计所用的记数系统，以及计算机的算术和逻辑操作。这些内容为哪些只想大致了解计算机及其作用的用户提供了必要的背景资料，而且也是后面将要讲的机器语言和汇编语言程序设计的予备知识。

1.2.计算机与计算机组织

就基本原理而言，几乎一切通用数字计算机都有共同的基本结构，如图 1 — 1 所示。

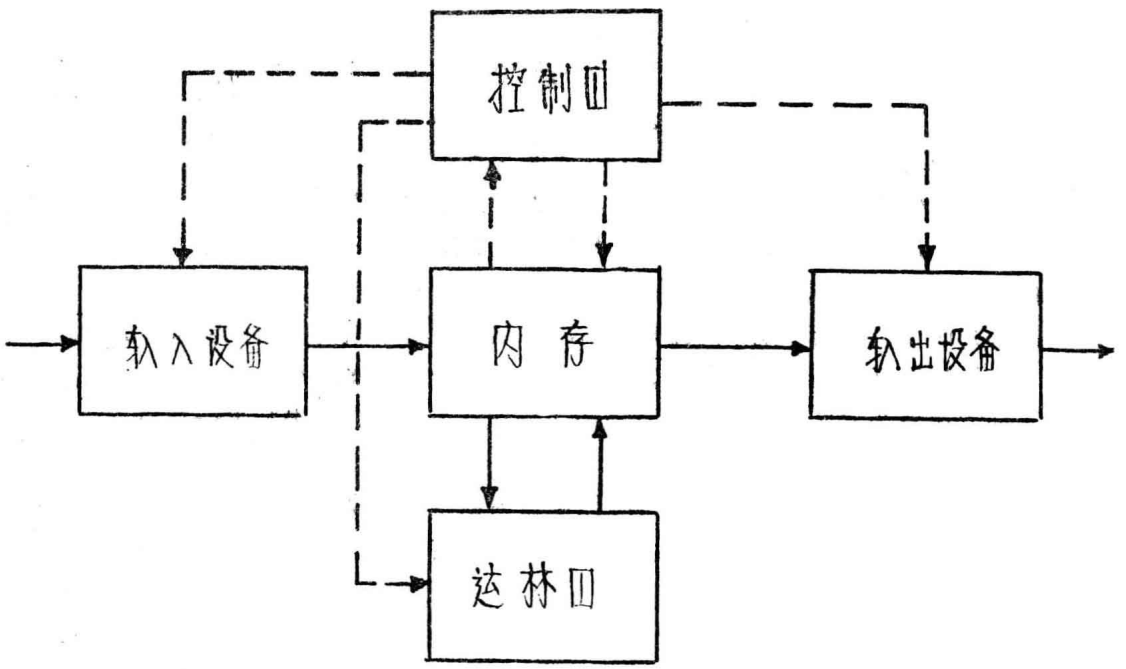

图 1—1 计林机的一般结构

1.2.1.计算机的基本部件

如果把一台机器叫作计算机，那么它必须具备某些算术运算的能力。能满足这一要求的部件叫作“运算器”。若想让运算器完成某种任务，须告诉它作什么，因此“控制器”是必不可少的。

因为算术运算是由运算器承担的，所以当它要计算同一个问题的另一部分时，必须把中

间结果保存起来，以后才能用于解决其他部分。另外，让信息随时可供控制器、运算器和计算机的其他部分使用，也是很有好处的。这一要求可由计算机的一个部件——“内存”或“主存（贮器）”来满足。

数字计算机的基本目的是以某种方式为人类服务。为此，必须有一种能把我们的愿望转达给计算机的手段，以及一种能接收计算机的计算结果的手段。完成这种功能的部件是输入和输出（I/O）设备。

1.2.2.运算器

数字计算机的运算器实际完成了计算任务。这是通过逻辑电路实现的。现代计算机使用的元件叫作“集成电路”。以前也用过开关和继电器，就其性能而论，当时还是可以采用的。但是现代计算机要追求更高的速度，只要可能，总是采用小型化的电子元件。

1.2.3.控制器

数字计算机的控制器是一个统治或开关的部件。它接受进入机器的信息，决定什么时候怎样完成操作。它要告诉运算器干什么，从哪儿取出必要的信息。当运算器完成一个计算时，它是知道的，它还要告诉运算器拿这个结果怎么办，下一步干什么。

控制器怎样告诉运算器让它干什么呢？这是通过解释一批指令实现的。这批指令是为控制器准备的，叫作“程序”，存放在计算机的存贮器中。存贮程序是计算机的一个特点，这一特点使它同计算器区别开。

控制器完成了当前指令规定的操作之后，必须到内存中取出下一条要执行的指令。该指令在内存中的位置可用两种方法规定。第一种方法是让上一条指令来规定下一条指令的位置。第二种方法较为常见，设立一个特殊的“程序计数器”，用来装下一条要执行的指令的位置。

不管使用什么方法，纯效果总是一样。控制器根据指令本身的那一部分或程序计数器所规定的次序每次执行一条指令。在1．2．8节里我们将讨论各种计算机指令格式，那时我们就会理解这两种方法了。

1.2.4.存贮器

存贮器用来存放供给控制器的信息（指令）和供给运算器的信息（数据）。外部存贮设备（如磁盘、磁带）经常叫作“次（极）存（贮器）”或“辅（助）存（贮器）”，而内（部）存（贮）设备（如磁芯、半导体存贮器）往往叫作“主存”。对存贮器的要求，视计算机的不同，上下相差很大。内存可以划分成一些小段，叫作“位置”（单元），每个位置可装一个数据元或一条指令。如果在一存贮设备中所有位置都同样容易地规定和到达，则把它叫作“随机存贮器”。另外一类叫作“顺序存贮器”（如磁带），在这种设备上，有些位置（在磁带开始处的位置）就比别的位置（磁带末尾处的位置）更容易到达。

1.2.5.输入设备

输入设备用来供应计算机所需要的数值，以及能指明如何在这些数值上进行操作的指

令。输入设备视机器的不同变化很大。对于小型计算机，键盘就够用了。要求快速输入的计算机把穿孔卡片用作数据输入。有的系统使用可拆卸的接插板，事先插好，就可完成某些指令。输入还可以通过穿孔纸带和磁带来完成，这两种形式在小型计算机系统中很常见。

1.2.6. 输出设备

输出设备能够记下计算机操作的结果。这些结果可以以一种永久的形式记录下来（如用打印机印出），或者用来激发一个物理动作（如调节压力的值）。许多输入用介质也可以用于输出，如纸带、穿孔卡片、磁带等。

1.2.7. 内存的构成

计算机内存是存放被处理的“信息单位”—指令和数据的地方。每个信息单位均放在内存的不同位置上，而每个位置都能存放一个这样的单位。各种内存都具有二个共同的构成上的特点：

1. 各个信息单位尺寸相同。
2. 每个信息单位都有一个与它相联系的编号或地址，可使用这个编号或地址唯一地引用它。

一个内存位置有两个特征：

1. 一个地址，就是与该内存位置相联系的一个数值。
2. 内容，就是物理地存放在这个特定内存位置上的数。

地址（名字）和内容（信息）之间的差别很重要，应注意理解。

信息单位的尺寸可以是一个二进位，叫作“位”；也可以是一些位，叫作“字符”或“字节”；还可以是更大的单位，叫作“字”。在设计计算机的时候，这个尺寸要根据计算机将来的应用种类确定下来。例如，在数据处理应用中被处理的信息，除数字之外，还包括名字和英语单词，所以最常用的是字节机器。另一方面字机器最常用于科学计算，对于这种应用，不仅参加运算的数特别多，而且要求的精确度又很高。

位、字符（字节）和字机器之间最重要的差别在于最小可编址单位的尺寸。对于位机器，是一个二进位（即是能表示0和1的信息单位）。对于字符机器，是该机器字符集中的一个字符（即是一个英文字母、一位数字，或其他特殊记号）。对于字机器，是它能表示的值域中的一个数据。

以后将会看到，字经常可以分割成固定多个字符或数码。但在真正的字机器中，我们只能引用字，而不能引用字符。所以这种分割只是为了方便，使我们能够想到，一个字“表示”了几个字符或几位数码。这里的“表示”是通过把字母“编码”为一个唯一的二进位组合而实现。在第4章中将会看到，在小型机上典型的“编码”方法。

1.2.8. 编址方案

当我们把图1—1中的两个部件—运算器和控制器放在一起考虑时，它们就形成了计算机的“中央处理机”（CPU）。CPU的任务是从内存中取出下一条要执行的指令，再取出这条指令所需要的操作数，最后执行这条指令。为完成这些任务，提供给CPU的指令字必

须规定：

1. 操作码（如：加、减、乘等）。
2. 操作数和结果的地址。
3. 下一条要执行的指令的地址。

因为算术操作大都要求两个操作数和一个结果，所以每条指令字除了操作码域之外，实际还要求四个域。图1—2给出了这种指令形式的一种可能的格式。

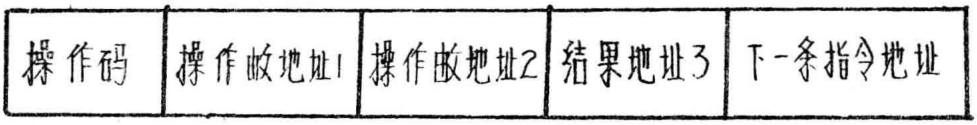

图1-2，三加一指令格式

1.2.8.1.三加一地址机器

采用图1—2那种指令格式的计算机叫作“三加一（3＋1）地址机器”。这种机器的指令可用符号表示成

ADD　　X，Y，Z，W

表示操作码　　内存位置的符号化名字

其意义是“将内存位置X的内容同内存位置Y的内容相加，结果存入内存位置Z中，再从内存位置W处取出下一条指令。”假定（X）表示“内存位置X的内容，”那末就不必说上面的话了，该ADD指令的意义可表述为

（Z）←（X）＋（Y）

下一条指令←（W）

此处箭头（←）读作“成为”或“使…等于”。这种表述形式叫作“插入记法”，因为要完成的操作插在相应操作数中间。

现在让我们来考虑计算FORTRAN表达式

A＝（B＊C）－（D＊E）

所用的计算机指令序列。为用符号化格式和插入记法表示该序列，我们有

符号化格式	插入记法
I1：MUL B，C，T1，I2	I1：（T1）←（B）＊（C），下一条指令←（I2）
I2：MUL D，E，T2，I3	I2：（T2）←（D）＊（E），下一条指令←（I3）
I3：SUB T2，T1，A，I4	I3：（A）←（T1）－（T2），下一条指令←（I4）
I4：序列后的下一条指令	I4：序列后的下一条指令

此处内存位置T1和T2表示用来暂存中间运算结果的内存位置。以后各节我们将用符号化格式和插入记法来描写各条指令。

我们仔细研究，就会发现，3 + 1 地址机器暴露了两个重要的事实。第一，尽管我们举出的例子用连续的内存位置来装各条指令字，但在这种机器结构中实无此必要。第二，因为每条指令都有三个明显的地址，所以不需要设立内部的计算机硬件，来装算术操作的结果。因此说，每个操作本身是完整的，可在一个指令周期内完成。

1.2.8.2.三地址机器

请注意上节后面的第一个注释，似乎有必要提出这样一个问题：为什么不需要把指令顺序地安排在连续的内存位置上？因为大多数程序都是顺序编写的，所以希望计算机从一个位置到另一个位置顺序地执行指令，是十分自然的。其实，这个问题是由于计算机内存的历史演变过程造成的。对于磁心存贮器来说，可以随机地存取各内存位置，而对于各个位置，存取时间没有差别。在磁心存贮器出现之前，内存是用汞延迟线或磁鼓作成，显然不能随机存取。在这类设备上，一般说来，下一个顺序位置不如别处某个位置存取得快。因此规定下一条指令位于最易取得的内存位置（未必是下一个顺序位置）上，就能编出一个存取时间为最少的程序，它比严格顺序的程序执行得快一些。（这种机器的典型例子是IBM650，尽管它使用的是1 + 1 编址方案，而不是3 + 1 的。）

但是因为大多数机器都有随机存取的内存，所以下一条指令的地址实在没有必要。而是设立一个指针，它叫“程序计数器”（PC），它象一个转动的箭头，总指向下一条要执行的指令。实际上PC是一个硬件寄存器，总放着当前执行的指令地址，当当前指令执行完了时，就修改它，使它指向下一条指令。〔例如，CDC 6600 就使用了一个寄存器（程序地址计数器），並执行“三地址寄存器”的运算指令。〕

因为在大多数机器中内存价格占有很大的比重，而内存价格又是每字中位数的函数，所以从每个字中去掉下条指令地址域，代之以一个硬件寄存器，显然会大大地降低价格。如果我们再去掉一个操作数地址域，还会继续降低价格，于是产生了一种新的便宜的机器，叫作“二地址机器”。

1.2.8.3.二地址机器

去掉第三个操作数地址，並不会产生严重的影响。经常不需要把算术运算的结果放在不同的地方，而把第二个操作数地址既用作源点又用作终点操作数的地址，有时反倒方便。例如，我们考虑把位置A的内容送到位置B去。对于三地址机器，需要编写一条指令如

```
ADD  A, ZERO, B        (B) ← (A) + θ
```

此处假定内存位置ZERO装有一个θ。对于两地址机器，应写作

```
SUB  B, B              (B) ← (B) - (B)
ADD  A, B              (B) ← (A) + (B)
```

然而这两条指令用的太频繁了，最好定义一条名为MOV（传送）的新指令

```
MOV      A, B
```

来代替上面的两条指令。

因为二地址机器可能比三地址机器指令多一些，而指令集太大又显得臃肿，所以务必尽量减少它的指令数目。为此要求仔细研究每条指令以及它在二地址机器的指令集中所处的地位。如此一来，我们不希望把三地址指令集简单平移到二地址指令集中（上述的MOV就是这样）。

现在不管是大型机，还是小型机都有采用二地址运算的。PDP—11、IBM1620、UNIVAC1150和IBM/360等都呈现了这种结构。

1.2.8.4.一地址机器

为进一步降低价格，我们可以从指令字中再去掉一个操作数域。结果，所余的操作数域须用作源点地址或终点地址，而大多数算术运算要有两个操作数，所以必须给一地址机器提供一个隐含（是内部硬件的一部分）的源或终点。这个隐含的操作数叫作“累加器”，它和台式计算器的累加器作用一样。

一地址机器的算术运算要用到累加器，如加到累加器中，从累加器中减去，等等，这种用法隐含在指令执行之中。为了使累加器初始化，以及保存累加器中的值，必须给一地址机器增加新指令—取累加器（LAC）和存累加器（DAC）。

下面给出一地址机器的编码示例。前已讨论的算术运算

A = (B * C) - (D * E)

在一地址机器上编码应为

LAC	D	(ACC) ← (D)
MUL	E	(ACC) ← (ACC) * (E)
DAC	T1	(T1) ← (ACC)
LAC	B	(ACC) ← (B)
MUL	C	(ACC) ← (ACC) * (C)
SUB	T1	(ACC) ← (ACC) - (T1)
DAC	A	(A) ← (ACC)

此处T1象以前一样还代表暂存位置，ACC代表累加器。细心的读者可能看出，如果第二个乘法不先作，就还需要一个暂存位置，一条DAC指令和一条LAC指令。

目前最普遍的编址方案是一地址的，例如PDP—8、PDP—15、IBM[illegible]、IBM 7090、CDC3600等机器都采用一地址指令和累加器。

1.2.8.5 通用寄存器机器

我们一旦接受了具有一个累加器的计算机的思想，就很容易把它扩大到具有多个累加器的机器上。这些累加器常用来存放刚产生的中间结果，这样一来暂存位置的需要量就减少了。假如有一台具有两个累加器的机器，它包含“取寄存器1”（LA1）、“存累加器2”（DA2）、“累加器1的内容乘以规定的内存位置的内容，结果放在累加器1中”（MP1）等指令，则FORTRAN表达式A = (B * C) - (D * E) 的编码序列可以是：

LA1	D	(ACC1) ← (D)
MP1	E	(ACC1) ← (ACC1) * (E)