第1章

机器语言简介

矛盾即是运动，即是事物，即是过程，也即是思想。

——毛泽东

计算机程序是在某种存储模式上实现的算法。所谓存储模式是指数据和对数据的一些基本处理的存在形式，这些形式可以被计算机直接或间接地识别；所谓算法是指求解可计算性问题的有穷步骤，每一步骤都可以转化为基本处理的有穷序列。程序语言提供存储模式或存储模式的创建机制，它正是在这个意义上被称为编程工具的。因此，可以说程序是用某种程序语言实现的算法。

最先提出计算机设计思想的人是图灵，他提出，应该用机器保留一些最简单的操作，然后将一个复杂的计算分解为这些操作。他设计出了第一台理论上的计算机——图灵机。实现这个思想的人是冯·诺依曼，他用计算机硬件实现了一些简单的操作，每一个操作都用一条机器指令表示。计算机指令系统构成第一个程序语言——机器语言。机器语言的特点是其指令直接关系到计算机硬件设施，因此，学习机器语言程序的同时也是在学习计算机的组成和工作过程。

1.1 计算机组成及工作过程

1. 程序存储思想和计算机组成

美国普林斯顿大学的冯·诺依曼于1945年提出的计算机体系结构设计思想，一般称为程序存储思想。计算机从1946年问世至今都是以这种思想为基本依据的。这个思想主要包含如下三点内容：

1）计算机应该采用二进制，与十进制相比，二进制的结构简单，容易实现和控制。

2）操作指令也是一种信息，和数据存储形式完全相同。

3）程序由机器指令组成，存储在计算机存储器中。每一条机器指令包含操作码和操作数两部分，前者是操作内容，后者一般是数据所在的存储单元地址，有时直接就是数据。

例如，“01H 1000H”是一条机器指令，其中01H是操作码，1000H是操作数。其具体意思是：取出地址为1000H存储单元中的数据，存入CPU的寄存器A中。

机器指令系统构成机器语言（machine language），机器指令和计算机硬件直接联系。学习机器语言的同时，也就是在学习计算机的组成和工作过程。

冯·诺依曼型计算机的典型系统结构由五个部分组成，它们是运算器、控制器、存储器、输入设备和输出设备（如图1-1所示）。

运算器进行各种算术运算和逻辑运算。控制器控制和指挥整个运算过程，使程序中的指令按要求一条一条执行。存储器存放程序及原始数据。输入设备输入指令代码和原始数据。输出

设备输出或打印计算结果。

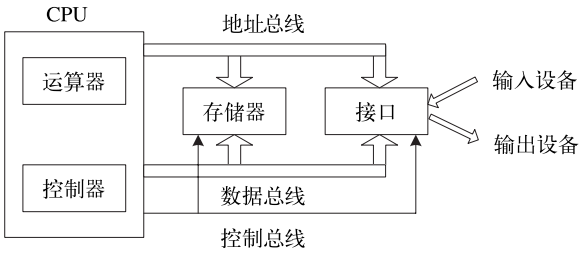

图1-1 计算机系统结构框图

2. 存储器和地址

存储器由存储单元组成，每一个存储单元称为一个字节（1B），存储器就是字节序列。一个字节有8位，存放8位二进制信息。每个字节有一个编号，称为地址（码）。

存储器的大小是指它有多少字节。每个字节相当于一座房子，有多少座房子，就应该有多少地址，反之，有多少地址，就有多少房子。地址总线的多少决定了地址的多少，也就是字节的多少。设有16根地址总线，如果一根地址总线有脉冲信号时表示1，没有脉冲信号时表示0，那么16根地址总线可以表示2^{16}个地址，这就是说存储器最多可具有2^{16}个字节。

例如，1000000000000000表示第1根地址总线有脉冲信号，其他地址总线没有脉冲信号，它代表一个二进制数表示的地址。这个地址可以用一个4位十六进制数简单地表示为8000H（H代表十六进制）。4位二进制数共有16个可能值，和1位十六进制数的16个可能值一一对应（见表1-1）。这个一一对应关系可以将一个16位二进制数简单地表示为一个4位十六进制数。使用十六进制数仅仅为了简洁。类似地，如果有32根地址总线，那么32位二进制地址可以用8位十六进制数简洁地表示。

表1-1 4位二进制数和1位十六进制数的对应表

二进制	十六进制	二进制	十六进制	二进制	十六进制	二进制	十六进制
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

存储器分为程序存储区、数据存储区和栈（如图1-2所示）。CPU通过地址总线发出相应的地址，选中存储器的某一字节即存储单元，然后通过数据总线读写该单元中的数据。

在存储器中特别划分出一块区域，称为栈，其特点是存取数据都在一端，该端称为栈顶（如图1-3所示）。栈的存取不需要计算地址，因此速度快，可以作为寄存器的补充。

3. CPU

运算器和控制器合称为中央处理器，简称CPU。CPU通过数据总线与存储器及接口交换数据。

在CPU中设有寄存器，它们与运算器或控制器直接相连，可以存放数据或计算的中间结果（如图1-2所示）。因为不通过地址总线和数据总线，所以寄存器的数据存取速度快。但是寄存器不能无限制地增加，否则会影响速度。

CPU中的寄存器有两类：通用寄存器和专用寄存器。通用寄存器是运算器的组成部分，用来暂存操作数及运算的中间结果。从本章的需要出发，我们重点介绍如下专用寄存器：

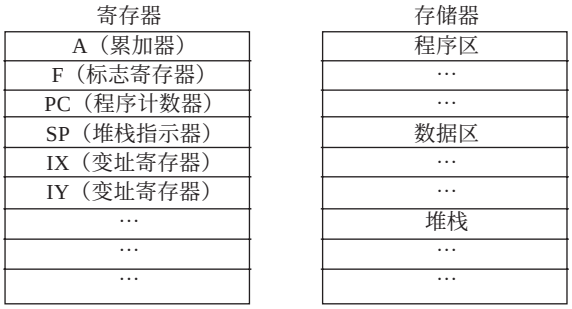

图1-2 Z80编程模型图

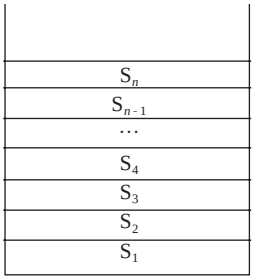

图1-3 栈结构示意图

- A是一个8位寄存器，一般称为累加器。它与运算器ALU一起完成各种运算。ALU是一个组合逻辑电路，本身不能保留信息，只有与A寄存器一起才能完成各种运算。累加器A在运算前向ALU提供操作数，运算后暂存运算结果。
- F是一个8位寄存器，一般称为标志寄存器。它与累加器A相连，记录运算结果的某些特征，以此作为控制程序流程转向的依据。
- PC为16位寄存器，一般称为程序计数器。程序是一组指令，这组指令一般都连续存放在存储器的程序存储区中。PC用来寄存指令的地址。CPU通过PC取出一条指令执行时，PC便“指向”下一条指令，即PC的值变为下一条指令的地址。比如，取出的一条指令占2个字节，取出这条指令之后，PC的值自动加2。除非遇到转移指令或子程序调用指令，否则CPU都是通过PC顺序地提取指令。
- SP为16位寄存器，一般称为堆栈指示器。SP的值始终是栈顶元素的地址，随着数据的存入和删除，SP的值自动改变。
- IX和IY是两个独立的16位变址寄存器，通常包含一个基地址（这个地址是根据需要写入的，一般在程序的首部通过赋值完成），由基地址加上偏移量（在程序运行中给出），以形成操作数的实际地址。

4. 计算机工作过程

程序是机器指令的有穷序列，机器指令联系着存储器和CPU。下面我们通过一个程序了解计算机组成原理和工作过程。以下是一个简单的求和程序：

```
y=a+b
```

把a和b单元中的数相加，结果存放在y单元。设a、b和y在存储器数据区的地址依次为3000H、3001H和3002H，如图1-5a所示。注意，a、b和y不是数据，只是符号。程序由4条指令组成（见表1-2）。这组指令在存储器程序区的地址分别为2000H、2003H、2006H和2009H，前3条指令各占3个字节，第4条指令占1个字节。一条指令的实际存储如图1-4所示。

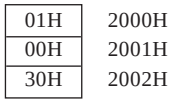

图1-4 指令01 3000H的存储

表1-2 求和程序y=a+b所包含的指令

操 作 码	操 作 数	指令含义
01H	3000H	取出地址为3000H的单元中的数据，存入寄存器A
03H	3001H	将地址为3001H的单元中的数据与寄存器A中的数据相加，结果留在A中
02H	3002H	将寄存器A中的数据存入地址为3002H的单元
00H		停机

CPU从程序计数器PC依次提取指令执行，每条指令的意义如表1-2所示。图1-5a～图1-5d演示了指令执行的结果，第4条指令执行之后程序停止。

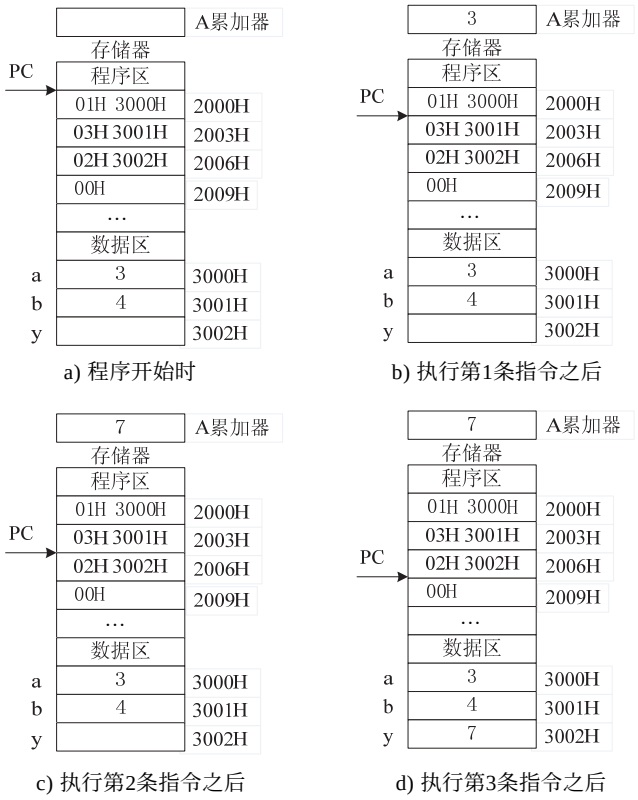

图1-5 求和程序y=a+b的执行过程示意图

1.2 计算机硬件和软件

自1946年第一台电子计算机诞生以来，计算机的发展主要经历了电子管、晶体管、集成电路和大规模集成电路四个阶段。到了第四个阶段，已经可以把计算机的中央处理单元即控制器和运算器制作在一个芯片上，称之为微处理机（或微处理器），简称为CPU（Central Processing Unit）。这就进入了微型计算机时代。

微处理机（microprocessor）：将运算器和控制器制作在一块半导体芯片上的集成电路器件，是构成微型计算机的核心，也称微处理器。

微型计算机（microcomputer）：微处理机配上存储器、输入/输出接口电路等芯片，由总线连接构成的设备。

微型计算机系统（microcomputer system）：微型计算机配上外部设备、电源和软件，构成的一个可独立工作的计算机整体。不过，人们通常所指的微型计算机，就是指微型计算机系统。

计算机仅靠硬件（hardware）是不能工作的，只有配备了程序才能运行，进而才能完成各种操作，这种程序称为软件（software）。软件通常存储在外部介质（如磁盘、磁带、卡片）或内部存储器上，它们不像硬件那样看得见、摸得着。

软件根据其功能，大致分为两类：

- 系统软件：专门由计算机设计者编写的管理计算机工作的程序，如操作系统、监控程序、调试程序、诊断程序和各种高级语言的编译程序。系统软件一般是和计算机设备一起配置的。
- 应用软件：计算机用户为了解决特定的实际问题而编写的程序或选用的已经标准化的程序（后者通常称为软件包）。

1.3 机器语言程序

现在我们假设有一台模型计算机，它具有几条简化的指令（见表1-3），然后在这个基础上进行程序设计。

表1-3 一台模型计算机的指令系统

指令名称	机器指令		说 明
	操作码	操作数	
取数	01H	N	A←(N) 取出地址为N的单元的数据，存入累加器A
存数	02H	N	(N)←A 将累加器A的数据存入地址为N的单元
加法	03H	N	A←A+(N) 将地址为N的单元中的数据和A中的相加，结果存入A
乘法	04H	N	A←A×(N) 将地址为N的单元中的数据和A中的相乘，结果存入A
比较	05H	N	A－(N) 比较两个数据（相减），结果存入寄存器F
转移	06H	N	PC←N 用地址N更新程序计数器PC的值
停机	00H		停机

例1.1 编程计算$y=ax^2+bx+c$。

算法设计：

1）分解方程：将原方程分解为$y=(ax+b)x+c$。

2）计算步骤见表1-4第1列，其中每一步可以对应一条机器指令。

3）分配存储单元：数据和程序的存储见表1-4第2列和第3列。

表1-4 例1.1算法和程序清单

算 法		存储器	指令地址	说 明
		程序区		
取数a		01H 3000H	2000H	A←(3000H)
计算$a\times x$		04H 3003H	2003H	A←A*(3003H)
计算$a\times x+b$		03H 3001H	2006H	A←A+(3001H)
计算$(a\times x+b)\times x$		04H 3003H	2009H	A←A×(3003H)
计算$(a\times x+b)\times x+c$		03H 3002H	200cH	A←A+(3002H)
计算结果存入y		02H 3004H	200fH	(3004H)←A
算法结束		00H	2012H	停机
		数据区	数据地址	
	a	5	3000H	
	b	6	3001H	
	c	7	3002H	
	x	2	3003H	
	y		3004H	

对上述计算没有采取如下步骤：

1）取数a。

2）计算$a\times x$。

3）计算$a\times x\times x$。

4）计算$a\times x\times x+b\times x$。

5）计算$a\times x\times x+b\times x+c$。

是因为表1-3中的指令系统功能有限，使步骤4）不能对应一条指令。算法设计受到了机器指令系统的限制。但是这种算法具有普遍意义，为了保留这种算法，我们扩展了机器指令系统，见表1-5。

表1-5 与子程序调用有关的指令

指令名称	机器指令		说 明
	操作码	操作数	
子程序调用	07H	N	断口地址进栈，用地址N更新程序计数器PC的值
返回主程序	08H		断口地址出栈，用来更新程序计数器PC的值
压栈	09H	T	寄存器T中的数据进栈保存
出栈	0aH	T	出栈，出栈数据进入寄存器T

下面我们结合表1-5来介绍以下一些概念：

- 入口地址。程序是一组指令，第一条指令的地址称为程序的入口地址。例如，例1.1中的2000H。
- 中断和断口地址。一个程序在执行过程中去执行另一个程序，称之为中断，主程序中断时的下一条指令的地址称为主程序的断口地址。
- 保护现场和恢复现场。在主程序中断时，如果有些寄存器中还存放着主程序的中间结果，那么就要将中间结果放入异地（比如栈）暂存，把寄存器“让给”子程序，这个过程是保护现场。待子程序执行完毕，再将主程序的中间结果放回寄存器，这个过程是恢复现场。
- 子程序调用过程。当主程序转去执行子程序时，首先将断口地址压入栈保存，再将子程序的入口地址送入PC。进入子程序后，首先保护主程序现场，然后执行子程序，子程序执行完毕后，恢复主程序现场，然后将断口地址从栈顶送回PC，过程如图1-6所示。

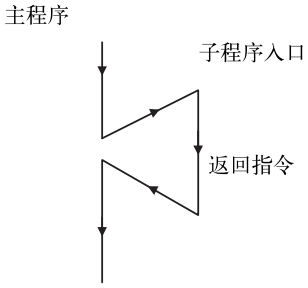

图1-6 子程序调用过程

例1.2 在增加了子程序调用指令的条件下，重新编程计算$y=ax^2+bx+c$。

首先将上述方程分解为两个简单的方程：

1）$y=ax^2+z+c$

2）$z=bx$

把求解第一个方程的程序称为主程序，求解第二个方程的程序就是子程序。算法设计和程序清单见表1-6。

表1-6 例1.2的算法和程序清单

算 法	存 储 器		指令地址	说 明
求解$y=ax^2+z+c$的算法		主程序区		
取a		01H 3000H	2000H	A←(3000H)
计算$a\times x$		04H 3003H	2003H	A←A ×(3003H)

（续）

算　法	存 储 器		指令地址	说　明
求解$y=ax^2+z+c$的算法		主程序区		
计算$a\times x\times x$		04H　3003H	2006H	A←A ×(3003H)
调用子程序计算$z=bx$		07H　2020H	2009H	PC←2020H，栈←200c H
计算$a\times x\times x+z$		03H　3005H	200cH	A←A +(3005H)
计算$a\times x\times x+z+c$		03H　3002H	200fH	A←A+(3002H)
计算结果存入y		02H　3004H	2012H	(3004H)←A
算法结束		00H	2015H	停机
求解$z=bx$的算法		子程序区		
寄存器A中的数据进栈（现场保护）		09H　A	2020H	栈←A
取b		01H　3001H	2023H	A←(3001H)
计算$b\times x$		04H　3003H	2026H	A←A×(3003H)
计算结果存入z		02H　3005H	2029H	(3005H)←A
栈中的数据进入寄存器A（恢复现场）		0aH　A	202cH	A←栈
返回主程序		08H	202fH	PC←200cH
		数据区	数据地址	
	a	2	3000H	
	b	3	3001H	
	c	4	3002H	
	x	5	3003H	
	y		3004H	
	z		3005H	

1.4　汇编语言

机器语言的局限性是明显的。首先，机器指令是用二进制编码的，不容易记忆和阅读。为了克服这种局限性，我们引入一种替代方法：用助记符来代替操作码，用符号代替地址。助记符是缩写的英文字符，与操作码的功能相对应；表示地址的符号即符号地址，由用户根据需要来定。这种由助记符和符号组成的指令集合称为汇编语言。汇编指令的书写格式为：

标号:操作码　操作数　;注释

其中标号可以省略。以表1-2中的机器语言程序为例，表1-7给出了相应的汇编语言程序。

表1-7　机器语言程序和汇编语言程序

机器语言程序			汇编语言程序		
	程序区				ORG 2000H ; 指定指令的起始地址
	01H 3000H	2000H		START:	LD　A,　(a) ; 把a地址单元的数据送到累加器A
	03H 3001H	2003H			ADD　A, (b) ;把累加器A的数据送到a地址单元
	02H 3002H	2006H			LD　(a), A
	00H	2006H		DONE:	HALT ;停机
	数据区				ORG　3000H ; 指定数据的起始地址
a	3	3000H		a:	DB　3　; 指定a符号地址3000H单元的数据
b	4	3001H		b:	DB　4　; 指定b符号地址3001H单元的数据
y		3002H		y:	DS　1　; 继b符号地址3001H之后指定一个字节
					END

汇编语言程序必须经过翻译，转变为机器语言程序，才能被计算机执行。我们把完成这一

翻译任务的程序叫做汇编程序，它是系统软件，一般是和计算机设备一起配置的。我们把利用汇编程序将汇编语言程序翻译为机器语言程序的过程称为汇编。把汇编语言程序称为源代码，把翻译后的机器语言程序称为目标代码，见图1-7。

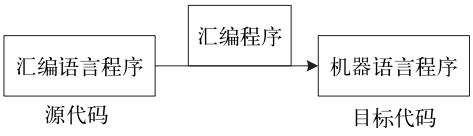

图1-7 汇编示意图

在汇编语言程序中，有些是汇编指令，有些不是。以表1-7为例，LD、ADD和HALT是汇编指令，它们经过翻译都转变为机器语言指令。而ORG、DB、DS和END不是汇编指令，仅是指示性指令，在翻译过程中起控制作用，并不转变为机器指令，我们称之为伪指令。这些伪指令的具体意义如下：定位伪指令ORG（Origin）指明的地址是其后的指令或数据块的起始地址；定义字节伪指令DB（Define Byte）为一个给定地址的存储单元定值；定义存储空间伪指令DS（Define Storage）为程序保留所需的存储空间；源程序结束伪指令END指明一个程序已经结束。

机器语言的另一个局限性是对计算机系统结构的依赖，在一种机器上开发的机器语言程序在另一种机器上一般是不能执行的。汇编语言克服了这个局限性，因为汇编程序是和计算机设备一起配置的系统软件，它可以把汇编语言程序翻译为相应的计算机可执行的机器语言程序。

1.5 深入探讨——存储和算法是一对矛盾体

存储和算法是一对矛盾体。存储为了算法，算法依赖存储。存储包含着算法，因为基本操作就是算法，只不过是最简单的算法；算法包含着存储，因为算法是基本操作的有穷序列。这是存储和算法的相互依赖。

由于计算机的硬件成本不断下降，软件开发成本不断提高，人们开始扩充机器指令系统，对使用频率高、执行时间长的指令序列（即算法）用一条新的复杂机器指令来代替，使算法转变为基本操作，以缩小机器语言与高级语言的语义差别，便于高级语言的编译。由此产生了复杂指令系统计算机（Complex Instruction Set Computer，CISC）。把使用频率在80%以上、能在一个时钟周期内执行完毕、在指令系统中仅占20%的简单机器指令保留下来，消除剩余80%的复杂机器指令，改由子程序来实现，即将指令转为算法，以减少指令和寻址方式的种类，固定指令格式，优化编译，提高性能价格比。由此产生了精简指令系统计算机（Reduced Instruction Set Computer，RISC）。这是存储和算法在同一级存储模式上的相互转化。

例1.1在计算$y=ax^2+bx+c$的算法设计中，之所以要分解为$y=(ax+b)x+c$，是因为机器指令系统中没有子程序调用指令，这是存储对算法的限制。例1.2在计算$y=ax^2+bx+c$的算法设计中，引入子程序调用指令，这是算法对存储的规定。而任何限制或规定都是一种哲学意义上的否定。

机器指令是面向硬件设计的，它把对数据的基本操作等同于最简单的机械动作，使得对一个像求和一样的简单计算，都需要多条机器指令才能完成，这显然不便于算法的编写、阅读、修改和维护。这个矛盾在汇编语言中依然存在，因为汇编指令不过是符号化的机器指令。随着算法越来越复杂，它和硬件存储模式的这种矛盾越来越突出。要有效地解决这种矛盾，就要超越计算机硬件结构，建立面向算法处理对象的程序语言。我们知道，计算机最初是用于数值计算的，在数值计算中，数据和数据的基本操作是按类型划分的。例如数据分整型、实型，基本操作有加、减、乘、除等，而且基本操作直接用操作符表示。如果用操作符和操作对象构成操作符指令，那么每一个数值算法都可以表示为操作符指令的有穷序列。这种基于类型的程序语言称为高级语言，而基于硬件结构的机器语言和汇编语言称为低级语言。

高级语言中的操作符指令是计算机无法识别的，但是每一个操作符指令都可以像我们前面

的求和程序那样转换为一组对应的机器语言指令，而高级语言程序是操作符指令的有穷序列，它也就可以转换为机器语言程序。这项工作是由编译程序（也称编译器）来完成的。高级语言程序（源代码）转换为机器语言程序（目标代码）的过程称为编译。图1-8为编译示意图。

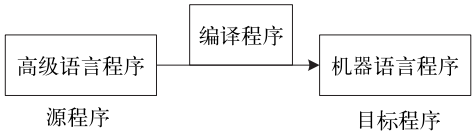

图1-8　编译示意图

操作符指令在高级语言中是基本操作，属于存储模式，而在机器语言中是算法，这是存储和算法在不同级别的存储模式上的相互转化。

低级语言的局限性在高级语言中被克服了，但是存储和算法的矛盾没有消失，只是表现为不同的形式，因为算法是越来越复杂的。后面我们在学习中继续考察这个矛盾。

习题

1. 何谓“程序存储思想”？
2. 计算机的基本组成及其主要功能分别是什么？
3. 什么是入口地址、断口地址和现场保护？
4. 分别用顺序结构和子程序调用结构两种方法编程计算$y=ax^3+bx^2+cx+d$。