高等学校电子信息类专业
“十三五”规划教材

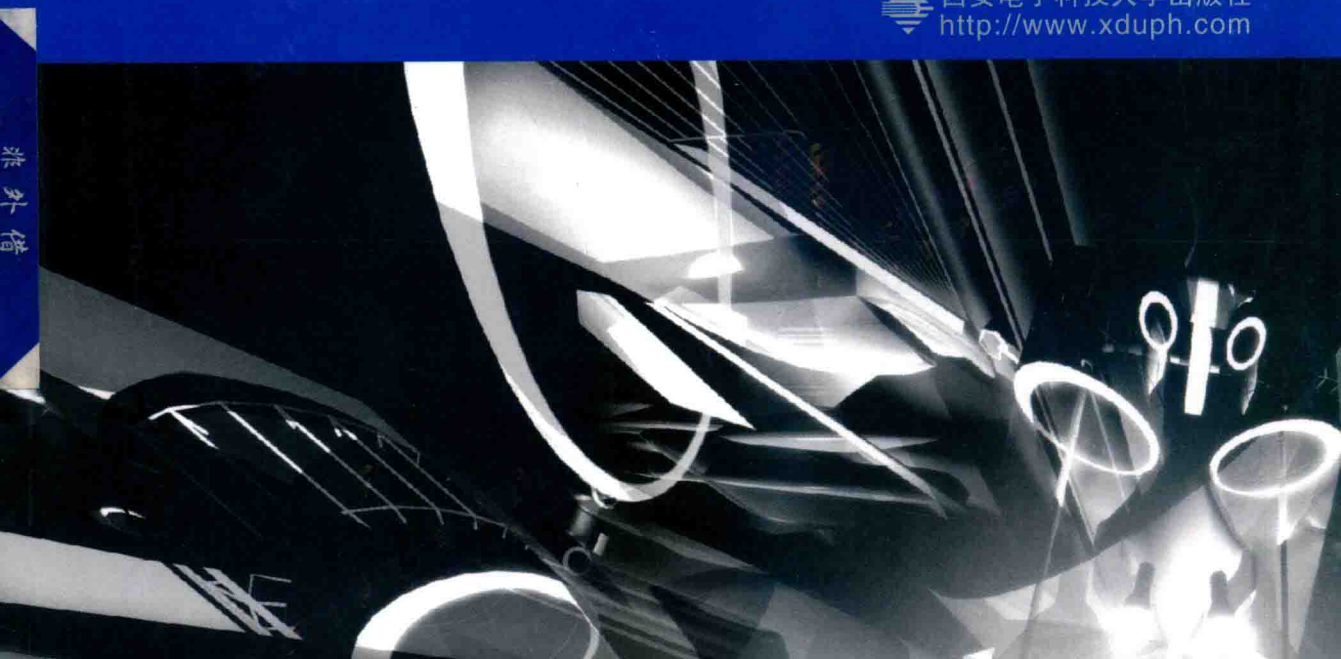
ELECTRONIC
INFORMATION SPECIALTY

计算机组成原理

——基于MIPS结构

康磊　编著

西安电子科技大学出版社
http://www.xduph.com



高等学校电子信息类专业“十三五”规划教材

计算机组成原理

——基于 MIPS 结构

康 磊 编 著

西安电子科技大学出版社

内 容 简 介

本书以 MIPS 微处理器为基础，从教学和实际应用的角度出发，讲述了计算机的基本组成和运行机制。全书中通过对 C 语言和汇编语言的比较，由浅入深地帮助读者理解高级语言和机器语言、计算机软件和硬件之间的关系，使读者对计算机的内部运行机制有一个整体的认识。本书主要内容包括计算机系统概述、运算方法和运算器、存储系统、总线技术、指令系统、中央处理器、输入/输出系统。全书语言通俗易懂、内容全面、条理清晰，突出实用性和先进性。

本书可作为高等院校电子工程、计算机工程和计算机科学专业“计算机组成原理”课程的教材，也可作为电子系统设计技术人员的参考用书。

图书在版编目(CIP)数据

计算机组成原理：基于 MIPS 结构/康磊编著. —西安：西安电子科技大学出版社，2019.4

ISBN 978-7-5606-4979-5

Ⅰ. ① 计… Ⅱ. ① 康… Ⅲ. ① 计算机组成原理 Ⅳ. ① TP301

中国版本图书馆 CIP 数据核字(2018)第 213412 号

策划编辑 云立实 刘玉芳

责任编辑 王 斌 雷鸿俊

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话 (029)88242885 88201467 邮 编 710071

网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com

经 销 新华书店

印刷单位 咸阳华盛印务有限责任公司

版 次 2019 年 6 月第 1 版 2019 年 6 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印张 20

字 数 475 千字

印 数 1～3000 册

定 价 45.00 元

ISBN 978-7-5606-4979-5/TP

XDUP 5281001-1

* * * 如有印装问题可调换 * * *

本社图书封面为激光防伪覆膜，谨防盗版。

前　言

“计算机组成原理”是计算机相关专业的一门专业核心基础课程。该课程主要讲述计算机的基本组成和运行机制。由于目前大多数传统教材单纯注重理论的讲解，缺乏具有针对性的具体机型的讲解，学生在学习的过程中往往面临概念多、头绪多、关系多的困境，无法把计算机中各部件有效地关联起来形成一个有机的整体的概念。

而随着计算机技术和大规模集成电路技术的发展，被称为“高效的 RISC 体系结构中最优雅的一种体系结构”的 MIPS 架构处理器已经替代了 Intel x86 和 ARM 处理器，被广泛应用于嵌入式系统，如消费电子和路由器等。因此，编者在调研国内外多所高校的“计算机组成原理”课程理论及实践教学安排的基础上，并根据教育部计算机教学指导委员会所提出的增强学生系统能力培养的目标要求编写了本书。本书以一个简单的 MIPS 架构模型机设计为主线索，使读者了解处理器中各部件的功能和相互之间的协作关系，同时通过对机器语言、汇编语言和 C 语言之间的关系的讲解，帮助读者深刻理解计算机内部的工作机制，帮助读者建立从计算机到数字电路、从高级语言到机器语言的实现过程以及软件和硬件之间的相互依存关系。

本书共分为七章，其具体内容安排如下：

第 1 章首先介绍了计算机系统的基本组成和常用概念；然后说明计算机的工作过程和性能指标，最后对 MIPS 架构做了简单说明。

第 2 章首先采用与 C 语言中的基本数据类型相对照的方式，说明了计算机中定点数、浮点数、字符和汉字的表示和存储方式；然后详细叙述了定点数和浮点数的算术逻辑运算方法和运算器的实现过程。

第 3 章首先介绍了存储器的基本知识，包括存储器的分类方法、性能指标，建立存储器体系结构的目的；然后从半导体存储元的工作原理入手，详细说明了半导体存储器的内部构成、读写方式、扩展方法以及存储器与 CPU 的连接方式；最后从提高存储器性能的角度分析了各种高速存储器和 Cache 的工作原理和性能。

第 4 章介绍了总线的基本原理，包括总线的常用结构、仲裁和通信方式，并对常用总线标准的特点做了简要说明。

第 5 章首先介绍了指令系统的一般概念，包括指令格式、寻址方式、指令类型；然后对比地介绍了 MIPS 微处理器的汇编指令，分析了 MIPS 指令的指令格式、寻址方式、指令类型的基本特点；最后通过将常用 C 语言代码转换为 MIPS 汇编指令的例子，使读者透彻理解高级语言与汇编语言、计算机软件与硬件之间的关系。

第 6 章介绍了微处理器的基本构成，并以一个 MIPS 模型机为例，详细说明了实现一个给定指令集的 CPU 的过程中典型指令的执行过程、数据通路的建立和不断完善的过程，以及采用单周期、多周期和流水线方式下控制器的工作原理和设计方法。

第 7 章介绍了主机与外设之间采用程序控制方式、中断方式、DMA 方式和通道方式

进行信息交换时的原理和方法。

本书从教学和工程角度出发，具有理论严谨、内容新颖、实用性较强等特点，力求缩小教学与实践的距离，为今后学生进行项目开发实践打下良好的基础。同时希望本书也能够对电子工程人员和高校相关专业学生有所帮助。

由于作者水平有限，加之时间仓促，本书难免有不足之处，敬请广大读者批评指正。本书作者的电子邮箱为 kangl@xsyu.edu.com。

编　者

2018 年 12 月

目　录

第 1 章　计算机系统概述

本章介绍计算机系统的基本知识，使读者对计算机的硬件系统有一个概括性的认识，为后续章节的学习打下基础。

1.1　计算机系统的组成

1.1.1　计算机的基本概念

计算机是一种能够运行存储的程序，自动、高速处理海量数据的现代化智能电子设备。计算机系统由硬件和软件两部分组成，如图 1-1 所示。

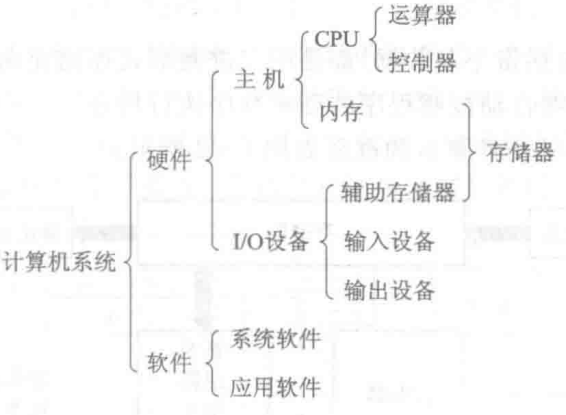


图 1-1　计算机系统的组成

硬件是指计算机系统中由电子、机械和光电元件等组成的各种物理设备的总称。这些物理装置按系统功能要求构成一个有机的整体，是计算机软件运行的物质基础。从图 1-1 可以看到，计算机硬件主要包括主机和 I/O 设备，主机和 I/O 设备的功能将在本节后续部分说明。

软件是看不见、摸不着的，它是计算机中使用的各种程序和数据文档的总称。程序是指按照特定顺序组织的计算机数据和指令序列的集合。软件存储在计算机的内存和辅助存储器中，如 RAM、ROM、磁带、磁盘、光盘、flash、U 盘等。软件是计算机系统的“灵魂”，

好的软件可以充分利用硬件资源，提高系统的工作效率。

一般来讲软件分为系统软件和应用软件。

(1) 系统软件。系统软件是指为了更加方便、高效地使用硬件资源而编写的控制和协调计算机各设备，为应用软件的开发和运行提供服务，并且无需用户干预的各种程序的集合。系统软件的主要功能是简化程序设计，提高计算机硬件的使用效率。系统软件通常包括操作系统、数据库管理系统、语言处理系统、网络管理软件和各类服务性程序。

(2) 应用软件。应用软件是指计算机用户为了解决各种问题而编写的程序。应用软件涉及广泛，按照软件的功能划分常见的有：科学计算类程序、工程设计类程序、数据处理类程序、信息管理类程序、自动控制类程序等。

系统软件和应用软件的划分界限并不是很严格，一些具有通用价值的软件对设计者而言是应用程序，但对其使用者来说就是系统程序。

1.1.2 冯·诺依曼结构计算机

1945 年 3 月，匈牙利裔美籍数学家约翰·冯·诺依曼(John von Neumann，1903—1957 年)起草了世界上第一台电子计算机 ENIAC 的设计报告初稿 EDVAC(Electronic Discrete Variable Automatic Computer)，提出了“存储程序”的思想。这份报告在计算机发展史上具有划时代的意义，奠定了计算机设计的理论基础。采用冯·诺依曼思想设计的计算机被称为冯·诺依曼结构计算机，虽然现代计算机结构更加复杂，计算能力更加强大，但基本结构仍然是基于这一原理设计的，因此，冯·诺依曼被称为“计算机之父”。

冯·诺依曼结构计算机的特点是：

(1) 计算机由五大部件构成，即计算机＝存储器＋运算器＋控制器＋输入设备＋输出设备。

(2) 所有的程序(包括指令和数据)都是用二进制形式存储在内存中的。

(3) 计算机应该能够自动按照程序的要求顺序执行指令。

典型的冯·诺依曼结构计算机的框图如图 1-2 所示。

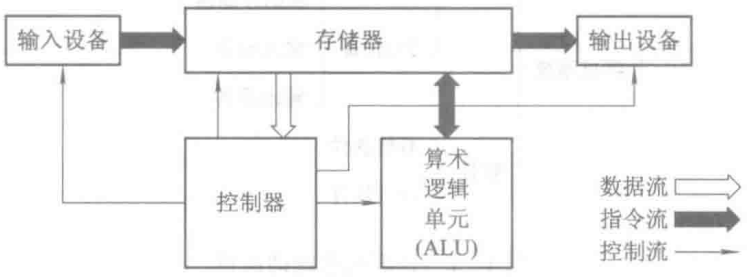


图 1-2 典型的冯·诺依曼结构计算机的框图

这里简单介绍各部件的功能，其工作原理和具体的实现过程将在后续章节详细说明。

(1) 运算器(Arithmetic and Logical Unit，ALU)：完成算术运算和逻辑运算。从提高 CPU 的效率出发，运算器内部往往有专用的寄存器运算数据，寄存器是计算机中速度最快的存储数据的电路。

(2) 主存储器(Main Memory，MM)：用来存储程序和数据。图 1-2 中的存储器就是主存储器，也称为内存，所有 CPU 执行的程序必须加载并存储在内存中。在计算机中存储

器可以分为主存储器和辅助存储器。主存储器是半导体存储器，在掉电的情况下无法存储数据；辅助存储器通常是磁表面存储器，相比内存其速度要慢很多，可用于长期存储各种程序和文档，辅助存储器属于I/O设备。

(3) 控制器(Control Unit，CU)：是计算机的控制中心，可以根据指令的功能，顺序产生每条指令在执行过程中各部件的控制信号，保证指令能够有条不紊地按顺序执行。

(4) 输入设备：将人们需要处理的信息转化为计算机能够识别的信息形式。大家最熟悉的输入设备就是键盘，当敲击键盘上的一个按键时，这个按键所对应的字母或数字就会被自动转换成相应的二进制编码，再传送给处理器。常用的输入设备有键盘、鼠标和触摸屏等。

(5) 输出设备：将计算机处理后的结果转换为人们能够接受的信息形式。常用的输出设备有显示器、打印机和绘图仪等。

有些设备既具有输出功能又具有输入功能，如网卡和辅助存储器，因此把这类设备称为I/O设备。

通常将运算器(ALU)和控制器(CU)合称为中央处理器(Central Processing Unit，CPU)。

在图1-2中可以看到有三种类型的信息在传递，分别是数据流、指令流和控制流。指令流是指存放在主存储器里的程序中的一条条指令按顺序依次送到控制器中的形式，这个过程被称为取指令；数据流是指两个部件之间有数据传输，从图中可以看到数据可以在运算器、存储器、输入设备和输出设备之间传输；控制流是指各部件的控制信号，由于所有的控制信号都是从控制器发出的，因此控制器是保证数据能够在各部件之间正确传输的核心设备，如果把计算机比作电脑的话，控制器就是计算机的“脑”。

另外一种常见的计算机结构是哈佛结构。图1-3是哈佛结构计算机的框图。与冯·诺依曼结构相比，哈佛结构是将指令和数据分开存储的计算机，并且两个存储器具有各自独立的地址访问空间和访问总线。由于指令和数据是分开储存的，因此上一条指令执行的同时可以预先读取下一条指令，因此具有较高的执行效率。哈佛结构的主要缺点是结构相对复杂，对外围设备的连接和处理要求高，不适合外围存储器的扩展。

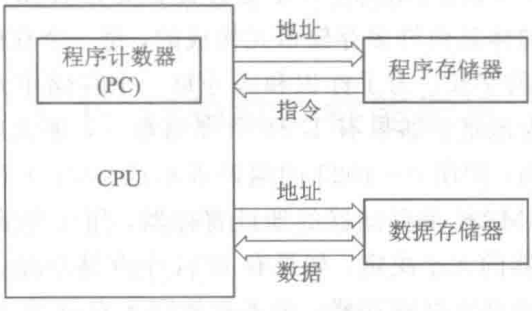


图1-3 哈佛结构计算机的框图

现在的通用处理器通常在内部采用将指令和数据分开存放的Cache，处理器外部Cache采用统一的方式。哈佛结构主要应用于嵌入式计算机中，在嵌入式应用中，系统要执行的任务相对单一，程序通常是固化在硬件中的。

1.2 计算机的工作过程

指令规定了计算机的一个基本动作，程序是完成某个特定功能的指令的有序集合。按照冯·诺依曼结构计算机的特点，计算机应该能够自动执行程序，而程序是由一条一条的指令构成的，换句话说，计算机应该能够顺序地执行每一条指令。计算机的工作过程实际上是一个周而复始的过程，即取指令、分析指令和执行指令的过程，这个过程可以用图 1-4 来描述。图中取指令后面的符号"⌐⌐"表示计算机的公共操作，是计算机为了具备中断、总线释放等功能而执行的操作。

这里以一个取数指令 Load ACC,X 为例，说明一条指令的执行过程。取数指令 Load ACC,X 的功能是将主存地址为 X 的存储单元的数据取出来，放入累加器 ACC 中。

指令的具体执行过程是由计算机的硬件完成的，而计算机的硬件结构又是千差万别的，这里我们将计算机的硬件用图 1-5 所示的细化框图来描述。为了说明计算机的工作过程，首先简要说明图中存储器、运算器和控制器的基本构成。

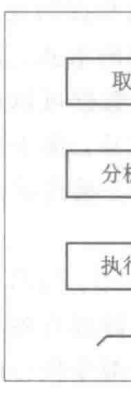


图 1-4 计算机的工作过程

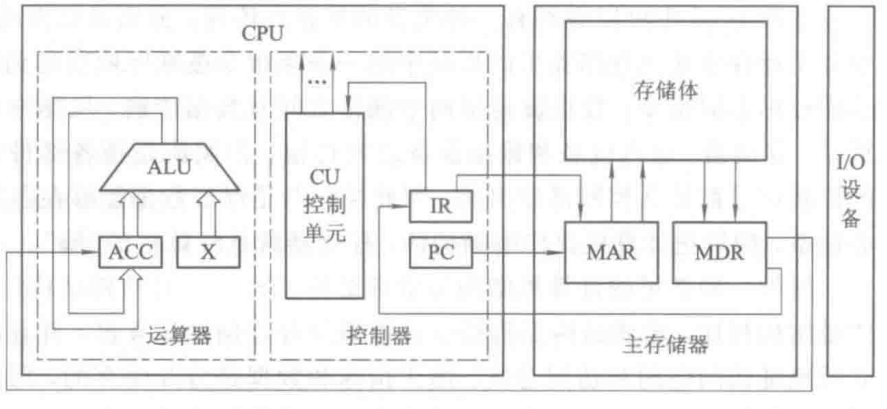


图 1-5 计算机硬件的细化框图

存储器用来存放指令和数据。图 1-5 中的存储器是由存储体和两个寄存器 MDR 和 MAR 组成的，其中，存储体是由许多存储单元构成的，每一个存储单元可以存储 n 位的二进制信息，n 就是存储器的字长。为了标识和区分每一个存储单元，对每一存储单元进行了编号，这个编号又称为地址。如果有 1024 个存储单元，那么就可以用 10 位二进制数($2^{10}=1024$)对其进行编码，即用 0～1023 的编码表示这 1024 个存储单元，每个单元的编号就称为存储单元地址。MAR 是存储器的地址寄存器，用于存放要访问的存储单元的地址，MAR 的位数由存储器的大小决定，如果有 1024 个存储单元，那么 MAR 就是一个 10 位的寄存器。MDR 是存储器数据寄存器，用来存放写入存储单元或从存储单元读出的数据，MDR 的位数就是存储器的字长。

运算器是执行算术、逻辑和移位运算的。图 1-5 中的运算器是由 ALU 和两个数据寄存器 X、ACC 组成的，运算器的两个输入数据来自 X 和 ACC，运算结果又回存到 ACC 中。

控制器由控制单元 CU、程序计数器 PC 和指令寄存器 IR 等组成。PC 实际上是个程序指针，也就是一个地址，这个地址指示将要执行指令的地址。由于指令是存放在主存储器

中的，因此主存储器中 PC 指示的存储单元中存放的就是将要执行的下一条指令。IR 是用于暂存当前正在执行指令机器码的寄存器，这个寄存器为指令译码和指令执行提供相关信息。CU 是整个计算机的控制中心，一条指令完整执行的各个环节所需要的控制信号都是从这里产生的。

下面我们以取数指令 Load ACC,X 为例，说明如图 1-4 所示的机器中指令的执行过程。假设此刻 PC 的内容等于 Y，也就是说，该指令存放在主存中地址为 Y 的单元中。

1. 取指令

取指令阶段的基本任务是将指令从内存中取出来存入 IR。这个阶段的具体操作工程如下：

(1) PC→MAR，即控制器将 PC 的内容送至存储器的 MAR，MAR 的值为 Y。

(2) 读存储器 M(MAR)→MDR，即将主存储器中地址为 MAR 的存储单元的内容送入 MDR，此时由于是取指令时间段，因此 MDR 中存放的代码就是将要执行的指令代码，即 Load ACC,X 指令的机器码。

(3) PC+1→PC，PC 指向下一条指令。

(4) MDR→IR，将 MDR 中的指令送入 IR，IR 中存放 Load ACC,X 的机器码。

2. 分析指令

对 IR 中的操作码和地址码进行分析，控制器获知这是一条取数指令，与这条指令相关的设备是 ALU 中的 ACC 和存储器中地址为 X 的存储单元。控制单元 CU 就会根据指令的功能要求，按照时间顺序依次正确的产生取数指令执行时所需要的全部控制命令，保证取数指令功能的正确性。

3. 执行指令

(1) IR(X)→MAR，将 IR 中存储单元的地址 X 送至存储器的 MAR。

(2) 读存储器 M(MAR)→MDR，这个操作和取指令阶段(2)的操作是一样的，但是由于不是在取指令时间，因此这个阶段取出来的是数据。

(3) MDR→ACC，将 MDR 中的数据送往 ACC，这是指令功能规定的。至此完成了取数指令的功能。

这条指令执行完毕后，在进行公共操作处理后会继续下一条指令的执行，顺序执行取指令、分析和执行指令，直至程序结束，机器会自动停机。

1.3 计算机的性能指标

由于计算机硬件的千差万别以及软件系统的规模差异，要综合评价计算机的性能是很复杂的事情，这里仅讨论硬件的性能指标。

对于单个计算机而言，如果其性能用时间来度量，那么响应时间越短的计算机，其性能也就越好。响应时间是指计算机完成某种任务所需的总时间，包括硬盘访问、内存访问、I/O 活动、操作系统开销和 CPU 执行时间等。在通常情况下，可以从以下几个方面对计算机的性能进行评价。

1. 字长

一般来说，把计算机在同一时间内能够并行处理的一组二进制位数称为一个计算机的

“字长”。这个指标与数据的存储、传输和运算器的位数有关。为了提高 CPU 效率，存储器字长、总线字长、运算器中寄存器的位数都是相互关联的。字长越长的数据其表示范围就越大，一次运算的数据位数就越多，计算机的速度就越快，其硬件成本也就越高。

与字长相关的单位有：b(bit，表示一位二进制)；B(Byte，字节，表示 8 位二进制)。如果一个 16 位机要计算 32 位或 64 位的数据，要用软件编程的方法通过多次 16 位计算来实现，这样虽然比较慢，但是可以降低硬件成本。

2. 存储容量

存储容量主要包括主存储器容量(简称主存容量)和外存储器容量。常用的容量单位有 K、M、G、T、P，具体关系如下：

$$1\ \text{KB}=2^{10}\ \text{B},\ 1\ \text{MB}=2^{20}\ \text{B},\ 1\ \text{GB}=2^{30}\ \text{B},\ 1\ \text{TB}=2^{40}\ \text{B};\ 1\ \text{PB}=2^{50}\ \text{B}$$

(1) 主存储器是指可以直接与 CPU 进行数据交换的存储器。需要执行的程序就存放在主存中，主存的容量越大，计算机的数据处理速度就越快。主存容量是指主存中存放二进制代码的位数，即：

主存容量＝主存储器单元总数×存储单元字长

(2) 外存储器容量通常是指硬盘容量。外存可以存放各种程序和数据，外存的容量越大，可存储的信息就越多。

3. 运算速度

衡量计算机运算速度要根据其执行程序所需时间的长短，通常情况下一个程序的执行时间由下面的公式决定：

一个程序的 CPU 执行时间＝一个程序的时钟周期数×时钟周期　　(1-1)

从式(1-1)可以看出，只要减少程序的 CPU 时钟周期数或者缩短时钟周期，都可以缩短程序的运行时间。

由于程序的功能不同，程序的规模差异也很大，因此不同程序的执行时间没有可比性。

常用的衡量运算速度的指标很多，这里只介绍几个常见的指标。

1) 主频

主频是 CPU 时钟信号的频率，是时钟周期的倒数。时钟频率越高，式(1-1)中的时钟周期就越短，执行每个操作的时间就越短，程序的执行时间也就越短。主频的计算公式如下：

$$f_{cpu}=\frac{1}{t_{cpu}} \tag{1-2}$$

式中，f_{cpu}是 CPU 的频率；t_{cpu}是时钟周期。

2) 指令的执行速度

指令的执行速度是指每秒执行的指令条数。

由于每个程序的功能和复杂程度都不一样，因此每个程序的执行时间都是有差异的。程序是由不同的指令序列构成的，每一条指令的复杂程度不同，其执行时间也不同，因此需要计算出指令的平均执行时间 T_m。T_m的计算公式如下：

$$T_m=\sum_{i=1}^{n} f_i \times t_i \tag{1-3}$$

式中，f_i表示指令系统中第 i 条指令出现的概率；t_i表示第 i 条指令的指令周期(这条指令从取出到指令执行完毕的完整时间)。

有了指令的平均执行时间 T_m 就可以算出计算机每秒执行指令的条数了，也就是常说的运算速度 V_m 了。运算速度的单位是 MIPS(Million Instructions per Second，每秒百万条指令)，其计算公式如下：

$$V_m=\frac{1}{T_m} \tag{1-4}$$

3) CPI(Cycle per Instruction)

CPI 是每条指令执行时所花费的平均时钟周期数。CPI 是一个程序中所有指令执行所用时钟周期的平均值，通过这个值可以对相同指令集的不同实现方法进行性能比较。式(1-1)中"程序的 CPU 时钟周期数"可以转换用式(1-5)表示：

$$\text{CPU 时钟周期数}=\text{程序的指令数}\times\text{每条指令的平均时钟周期数} \tag{1-5}$$

例 1-1 计算机 A 和计算机 B 具有相同的指令集，计算机 A 的时钟周期是 250 ps，执行程序 P 的 CPI 是 2.0；计算机 B 的时钟周期是 500 ps，执行相同程序的 CPI 是 1.2。试比较两台计算机的速度。

解 设程序 P 的指令条数为 I。首先计算每台计算机的 CPU 时钟周期数：

$$\text{CPU 时钟周期数}_A=I\times 2.0,\quad \text{CPU 时钟周期数}_B=I\times 1.2$$

然后计算每台计算机的 CPU 时间：

$$\text{CPU 时间}_A=\text{CPU 时钟周期数}_A\times\text{时钟周期时间}_A=I\times 2.0\times 250\times 10^{-12}=500\times I\ \text{ps}$$

$$\text{CPU 时间}_B=\text{CPU 时钟周期数}_B\times\text{时钟周期时间}_B=I\times 1.2\times 500\times 10^{-12}=600\times I\ \text{ps}$$

因此，计算机 A 的速度更快。

4) FLOPS(Floating-point Operations per Second)

FLOPS 用来衡量计算机浮点数据的处理能力，是指每秒所执行的浮点运算次数。目前大部分处理器都有一个专门用来处理浮点数据的"浮点运算器"(FPU)。FLOPS 所测量的，实际上就是 FPU 的执行速度。目前世界最快超算的排名就是通过运行 Linpack 软件包，测量其 FLOPS 决定的。

4. 可靠性

计算机的可靠性可以用计算机平均无故障工作时间来表示，即计算机硬件运行时不发生故障的平均时间。

1.4 计算机系统的体系结构

1.4.1 计算机层次结构的划分

计算机系统是由硬件、软件组成的一个十分复杂的整体。一台计算机硬件最终能够执行的程序都是由其所能识别的指令组成的，而且只能识别 0、1 机器代码。对于同一个计算机，从不同角度看到的计算机系统的属性是不同的，如从使用人员的角度，计算机可以分为用户、应用程序开发者、操作系统程序员、硬件设计者等，每个人对计算机的使用角度和知识背景都是有很大差异的。不同人在使用计算机时，根据其目的和对硬件的了解程度可以分为多个不同的层次。最常用的层次划分方法是从使用语言的角度出发，这样就可以

把计算机系统按功能划分成如图 1-6 所示的 5 个级别层次，用户可以根据需要选择其中的一个层次，了解计算机系统的特性和工作原理，完成相应的设计任务。图 1-6 中从上到下的 5 个层次可以假设成 5 种不同的机器，指令系统之上的级别属于软件，指令系统之下的级别属于硬件。

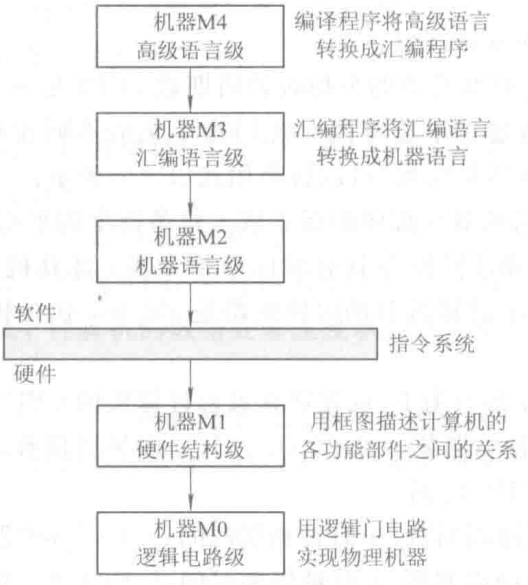


图 1-6 计算机系统层次结构示意图

随着超大规模集成电路的快速发展，许多之前由于受限于硬件成本而采用软件实现的功能都可以改为由硬件来实现，这就是软件硬化。例如，早期的许多浮点运算都是由软件开发包实现的，现在的 CPU 都支持浮点运算指令，即可用硬件完成浮点运算。

1.4.2 计算机体系各层次的特点

图 1-6 中最高级是第五级，即虚拟机器 M4 高级语言级，这一级属于软件级。使用者可以用高级语言来使用机器，编写的高级语言程序需要由相应的编译程序转换为汇编语言后交由第四级机器执行。M4 级的用户只需要掌握高级语言的语法和语义，就能够通过编程使用机器，而对机器的具体机型、内部结构和指令系统几乎可以一无所知，这样为程序员带来了极大的方便，程序的可移植性就很好。例如，要计算 1～10 的和，可以采用 C 语言编写如下程序段实现：

```
int main(int argc, char * argv[])
{
  int i,sum=0;
  for(i=0;i<=10;i++)
    sum+=i;
}
```

这个程序段在实际机器上是无法执行的，需要经过编译和连接程序将其转换为机器指令才能在实际机器上运行。

第四级 M3 是汇编语言级，属于软件级，汇编语言使用汇编指令写程序。汇编指令是用一些接近人类语言的符号表示机器指令、数据或其地址，如用 ADD 表示加法操作、用

LOAD表示取数操作等，这样程序员就不必记忆毫无规律的二进制机器指令和编码了。为了能够让机器执行汇编指令，这些汇编指令必须翻译为机器指令，这个翻译工作是由汇编语言程序完成的。在通常情况下，汇编语言的语句和机器指令有一一对应的关系，因此用汇编语言编写的程序转换为机器语言时生成的冗余代码会很少，程序的执行效率很高。相比而言，高级语言转换为机器语言会生成许多冗余代码。同样计算1～10的和，采用MIPS指令的汇编语言编写的程序段如下所示：

```
.data
n:.word  10                    # n
sum:.word  19                  #  sum of 1 to n
.text
        lw $t1,n($zero)        # $t1=n
        addi $s0,$zero,0       # $s0=0
        addi $s1,$zero,1       # $s1=1
loop:   add  $s0,$s0,$t1       # $s0=$s0+$t1
        sub $t1,$t1,$s1        # $t1=$t1-1
        bne $t1,$zero,loop     # 如果 $t1≠0  则转向 loop，否则结束
        sw  $s0,sum($zero)     # sum=$s0
    syscall                    # Exit!
```

其中，.data定义了程序中所需要的数据；.text定义了程序的开始。程序中对每条语句的功能做了注解，请读者将其与C语言程序进行比对分析。

从这个程序段可以看到，使用汇编语言的程序员不但要掌握MIPS汇编语言，还需要了解寄存器的数量、寄存器的名称、数据存放的具体位置等硬件知识，这对程序员的要求是比较高的。

第三级M2是机器语言级，这一级处于软件和硬件的交界处。不同机器硬件能够识别的指令系统是不一样的，如Intel系列CPU和ARM系列CPU所能识别的指令系统是不同的。每一个机器的硬件所能够执行的指令集由其设计者决定，其指令的类别、格式和寻址方式都是千差万别的，机器语言的可移植性很差。图1-7中给出了与上面MIPS汇编语言程序对应的机器语言指令。其中，右边一列是汇编语言程序对应的机器指令的十六进制表示，左边是机器指令在内存中存放的地址，这个程序即使是经验丰富的汇编语言程序员也无法直接读懂。

Address	Code
0x00000000	0x8c092000
0x00000004	0x20100000
0x00000008	0x20110001
0x0000000c	0x02098020
0x00000010	0x01314822
0x00000014	0x1520fffd
0x00000018	0xac102004
0x0000001c	0x0000000c

图1-7 机器语言指令

第二级M1属于计算机的硬件设计级别，在这一级中，可以把计算机看成是用数据流将各功能模块连接起来的框图。这个级别的机器按照指令系统中所有指令的功能和寻址方式确定每个模块的接口和数据通路的设计。例如，一个运算器的框图可以表示为如图1-8所示的运算器框图。从图中可以看出，运算器

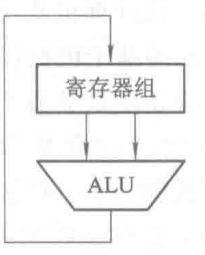


图1-8 运算器框图