

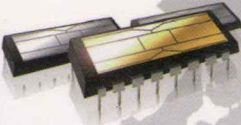


北京市高等教育精品教材立项项目



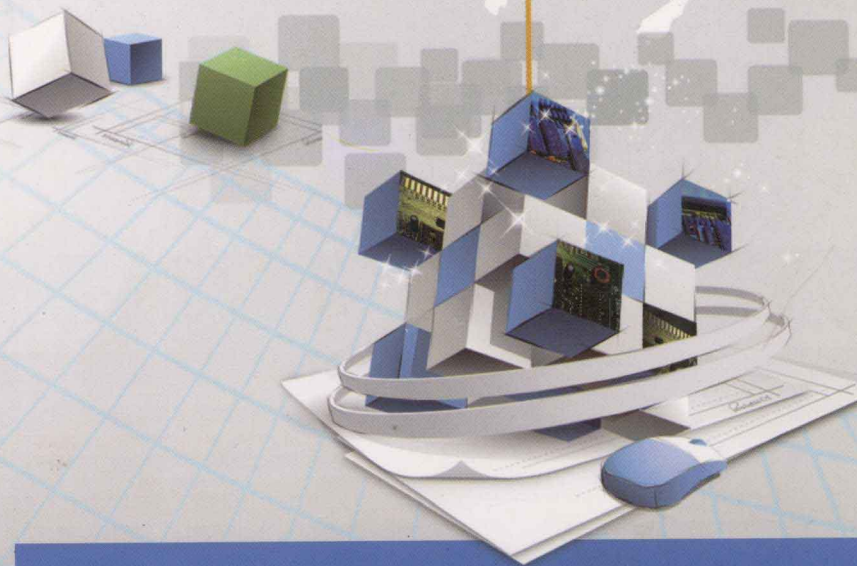
高等学校工程创新型“十二五”规划计算机教材

微机原理 与接口技术



周锋 倪晖 编著

——计算机通信国家级教学团队成果



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>



北京市高等教育精品教材立项项目

高等学校工程创新型“十二五”规划计算机教材
计算机通信国家级教学团队成果

微机原理与接口技术

周 锋 倪 晖 编著

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书是计算机通信国家级教学团队成果和北京市精品教材立项项目。

全书以 8086 处理器和 16 位微型计算机系统为实例进行讲解,共分 10 章,主要包括微机基础、微处理器的功能结构、指令系统与寻址方式、汇编语言程序设计、处理器外部特性与输入/输出、存储器、中断与定时技术、常用接口技术、外部设备接口、总线等内容。本书提供配套 PPT 课件和相关教学资源,读者可以通过华信教育资源网(www.hxedu.com.cn)免费注册下载。

本书适合作为高等学校理工科各专业微机原理与接口技术教材,也是相关技术人员或爱好者的参考书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

微机原理与接口技术 / 周锋, 倪晖编著. —北京: 电子工业出版社, 2010.9

高等学校工程创新型“十二五”规划计算机教材

ISBN 978-7-121-11602-5

I. ①微… II. ①周… ②倪… III. ①微型计算机—理论—高等学校—教材 ②微型计算机—接口—高等学校—教材 IV. ①TP36

中国版本图书馆 CIP 数据核字(2010)第 159282 号

策划编辑: 童占梅

责任编辑: 索蓉霞

印 刷: 北京市李史山胶印厂
装 订:

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×1092 1/16 印张: 19.75 字数: 505.6 千字

印 次: 2010 年 9 月第 1 次印刷

印 数: 4 000 册 定价: 32.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zits@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

前 言

电子计算机是 20 世纪最重要的科技发明之一，它对人类活动的各个领域产生了重大而深远的影响。1981 年 IBM 公司推出了个人计算机 IBM PC，在不到三十年的时间里，微型计算机以令人难以置信的速度飞速发展。微型计算机的性能越来越高而价格越来越低，应用范围越来越广，人们的日常生活中已经离不开它。

《微机原理与接口技术》课程以 Intel 80x86 微处理器和 IBM PC 系列机为主体，介绍微处理器的工作原理、微型计算机系统的组成、汇编语言程序设计和接口技术。这是一门包括软件和硬件两方面知识的课程，以具体处理器为实例，讲解处理器的构成、工作原理及程序设计，从而更深刻、更全面地理解计算机系统。其先修课程至少包括数字电路与逻辑设计、一种高级语言程序设计等课程。

Intel 80x86 系列处理器经历了 8086、80286、80386、80486、奔腾和酷睿等处理器微体系结构的发展过程，今天我们使用的处理器已进入多核时代。为什么本课程还以化石级的 8086 处理器和 16 位微型计算机系统为主进行讲解呢？这个问题在近年来的教学过程中也经常被学生问及，相信众多的专业教师在教学中也遇到过同样的问题。

现代微处理器及微计算机系统的体系结构已发展到了一个相当复杂的程度，其处理能力和所用技术的复杂度已经大大超过了早期的大型机，而且还在不断地向前发展。但现代计算机仍然没有摆脱冯·诺依曼体系结构，技术复杂性的增加带来的是处理器性能的提升。因此，以 8086 处理器为实例，讲解微处理器的结构、指令系统、微型计算机的基本工作原理等内容，对一个初学者来讲，不失为一个最佳选择。众所周知，让初学者站在系统的角度认识计算机系统软硬件和工作过程不是一件简单的事情。如果直接讲解 32 位处理器，即使是一个最简单的 32 位 80386 处理器，难免要涉及多任务支持、虚拟存储器的管理、任务管理与切换、保护模式下的编程方法等复杂概念，这样不仅没有足够的课时，而且会使初学者感到新概念、新知识过多，难以理解。因此，基于上述考虑，本书讲述一个简单而具体的 8086 处理器，让初学者理解微处理器的基本原理、指令的执行过程，进而建立起微型计算机体系结构的整体概念，在此基础上，再引入 32 位处理器的概念及关键技术。

本书是计算机通信国家级教学团队成果和北京市精品教材立项项目。全书共分 10 章。

第 1 章微机基础，介绍微型计算机的基础知识，包括数和字符的表示、微机基本结构、软件系统以及微机系统的发展等。

第 2 章微处理器的功能结构，介绍 8086 处理器的内部结构、寄存器结构、存储器组织、I/O 组织、32 位微处理器的内部结构、保护模式下的存储器组织和任务管理等。

第 3 章指令系统与寻址方式，介绍 8086 的寻址方式、指令格式、指令系统、32 位处理器寻址方式及指令的扩展等。

第 4 章汇编语言程序设计，介绍如何进行汇编语言程序设计，包括汇编语言概述、伪指令、程序的基本结构、子程序结构、宏汇编和条件汇编等。

第 5 章处理器外部特性与输入/输出，介绍微处理器的引脚功能、工作时序、两种模式下系统的典型连接、接口的功能、端口的概念、译码电路、处理器与外设之间的数据传送方

式等。

第 6 章存储器，介绍多级存储体系、半导体存储器、内存储器的组织等。

第 7 章中断与定时技术，介绍中断的基本概念、中断系统、可编程中断控制器 8259A 及其应用、硬件中断服务程序的编写、定时与计数技术等。

第 8 章常用接口技术，介绍可编程并行接口芯片 8255A、串行接口、DMA 传输、A/D 和 D/A 转换等。

第 9 章外部设备接口，介绍键盘接口、打印机接口、显示器原理及接口等。

第 10 章总线，介绍总线的分类、ISA 总线标准、PCI 总线标准、USB 总线的体系结构和总线协议等。

本书作者是从从事计算机专业课和专业基础课的教师，在长期的教学实践活动中积累了丰富的经验，同时，还主持和参与了许多计算机应用系统的开发和研究工作，积累了丰富的工程开发经验。作者将这些经验和体会融入到本书中，并结合当前微型计算机软、硬件新技术的发展趋势，从传授知识和培养能力的目标出发，让初学者尽快掌握组成微机系统的各种硬件接口技术、汇编语言程序设计的基本方法，学会分析、设计微机系统各种接口的硬件电路及相应的驱动程序。

本书由周锋、倪晖编著，整体框架由周锋负责设计。其中，第 1~4 章和第 6 章由倪晖编写，第 5 章和第 7~10 章由周锋编写。

为方便广大读者学习和参考，本书**提供配套 PPT 课件和相关教学资源**，读者可以通过**华信教育资源网（www.hxedu.com.cn）免费注册下载**。

由于作者水平有限，且时间仓促，书中错误和不足之处在所难免，恳请读者指正。反馈信息可联系 E-mail: zfeng@bupt.edu.cn。

编著者
于北京邮电大学

目 录

第 1 章 微机基础	1
1.1 计算机中的数和字符	1
1.1.1 进位制	1
1.1.2 数的机器表示	2
1.1.3 字符编码	5
1.2 微机的基本结构	5
1.2.1 冯·诺依曼结构	5
1.2.2 微机的硬件组成	6
1.3 微机的软件系统	8
1.3.1 指令和指令系统	8
1.3.2 程序和语言	8
1.3.3 微机软件系统	10
1.4 微机系统的发展	11
1.4.1 微机的诞生	11
1.4.2 微处理器的发展	11
1.4.3 微机系统的发展	12
习题 1	13
第 2 章 微处理器的功能结构	14
2.1 8086 的内部结构	14
2.2 8086 的寄存器结构	15
2.2.1 通用寄存器	16
2.2.2 段寄存器	17
2.2.3 指令指针寄存器	17
2.2.4 标志寄存器	18
2.3 8086 的存储组织	19
2.3.1 物理空间与字节序	19
2.3.2 逻辑空间	21
2.3.3 存储管理	22
2.4 8086 的 I/O 组织	24
2.5 32 位微处理器	24
2.5.1 内部结构	24
2.5.2 寄存器结构	26
2.5.3 工作模式	28
2.5.4 保护模式下的存储组织	28
2.5.5 任务管理	31
习题 2	32

第3章 指令系统与寻址方式	33
3.1 8086 寻址方式	33
3.1.1 指令	33
3.1.2 操作数	33
3.1.3 寻址方式	35
3.2 8086 指令格式	38
3.3 8086 指令系统	39
3.3.1 数据传送指令	39
3.3.2 算术运算指令	44
3.3.3 逻辑指令	50
3.3.4 串指令	53
3.3.5 控制转移指令	55
3.3.6 处理器控制指令	63
3.4 80x86 寻址方式及指令的扩充	64
3.4.1 寻址方式的扩充	64
3.4.2 指令系统的扩充	64
习题3	65
第4章 汇编语言程序设计	68
4.1 汇编语言概述	68
4.1.1 汇编语言的特点	68
4.1.2 汇编程序	68
4.1.3 汇编语言的组成	70
4.2 伪指令	72
4.2.1 程序格式	72
4.2.2 段定义	73
4.2.3 变量定义	78
4.2.4 常数、变量、标号、表达式	80
4.2.5 符号定义	86
4.2.6 地址对齐	87
4.2.7 结构定义	88
4.3 程序的基本结构	89
4.3.1 基本的程序框架	89
4.3.2 程序结构	90
4.3.3 顺序结构	91
4.3.4 分支结构	92
4.3.5 循环结构	95
4.4 子程序结构	98
4.4.1 子程序定义	98
4.4.2 子程序调用和返回	99
4.4.3 环境的保存和恢复	100

4.4.4 参数的传递	100
4.4.5 高级语言参数传递	108
4.4.6 递归	108
4.5 宏和条件汇编	111
4.5.1 宏	111
4.5.2 条件汇编	115
习题 4	117
第 5 章 处理器外部特性与输入/输出	120
5.1 处理器外部特性	120
5.1.1 8086 的引脚功能	120
5.1.2 8086 工作时序	124
5.1.3 两种模式下系统的典型连接	127
5.2 I/O 接口概述	130
5.2.1 I/O 接口的功能	130
5.2.2 I/O 接口的基本结构	131
5.2.3 I/O 端口的编址	131
5.2.4 I/O 端口地址译码	133
5.2.5 基本输入/输出接口	134
5.3 处理器与外设的数据传送方式	137
5.3.1 程序控制方式	137
5.3.2 中断方式	139
5.3.3 直接存储器访问方式	140
习题 5	141
第 6 章 存储器	143
6.1 多级存储体系	143
6.2 半导体存储器	144
6.2.1 半导体存储器的分类	144
6.2.2 RAM 的结构	146
6.2.3 现代 RAM	149
6.3 内存储器的组织	150
6.3.1 内存储器的结构	150
6.3.2 存储体的构造	151
6.3.3 存储器与 CPU 的连接	152
6.4 Cache	154
6.4.1 Cache 的基本原理	154
6.4.2 Cache 的结构	155
6.4.3 Cache 的策略	157
6.4.4 Pentium 的 Cache	158
习题 6	158

第 7 章 中断与定时技术	160
7.1 中断的基本概念	160
7.1.1 中断	160
7.1.2 中断源及中断源识别	160
7.1.3 中断优先级	162
7.1.4 中断嵌套	162
7.1.5 中断处理过程	163
7.2 8086 的中断系统	164
7.2.1 8086 中断类型	164
7.2.2 中断向量表	166
7.2.3 8086 中断响应过程	166
7.3 可编程中断控制器 8259A	167
7.3.1 8259A 的内部结构及引脚	168
7.3.2 8259A 的工作方式	169
7.3.3 8259A 的编程	173
7.4 8259A 应用举例	180
7.5 硬件中断服务程序的编写	181
7.6 定时与计数技术	185
7.6.1 可编程定时/计数器 8253/8254	186
7.6.2 8253/8254 工作方式	188
7.6.3 8253/8254 编程	192
7.6.4 8253 /8254 的应用	195
习题 7	197
第 8 章 常用接口技术	199
8.1 可编程并行接口 8255A	199
8.1.1 8255A 的内部结构及引脚	199
8.1.2 8255A 的控制字	201
8.1.3 8255A 的工作方式	203
8.1.4 8255A 的应用	206
8.2 串行接口	211
8.2.1 串行通信基础	211
8.2.2 串行接口标准	216
8.2.3 可编程串行通信接口 Intel 8251	221
8.3 DMA 传输	229
8.3.1 DMA 传输过程	229
8.3.2 DMA 控制器 8237A	229
8.3.3 8237A 应用实例	237
8.4 数/模、模/数转换	240
8.4.1 模拟输入/输出系统	240
8.4.2 数/模转换器接口	241

8.4.3 模/数转换器接口	245
习题 8	252
第 9 章 外部设备接口	254
9.1 概述	254
9.2 键盘	254
9.2.1 键盘接口	256
9.2.2 PC 键盘及接口	260
9.3 打印机接口	262
9.4 显示器	264
9.4.1 LED 显示器	265
9.4.2 CRT 显示器及接口	268
9.4.3 液晶显示器	274
习题 9	277
第 10 章 总线	278
10.1 概述	278
10.1.1 总线的发展过程及分类	278
10.1.2 总线的传送过程	281
10.1.3 总线控制方式	282
10.1.4 总线的性能参数	286
10.2 ISA 总线	286
10.3 PCI 总线	288
10.3.1 PCI 体系的基本概念	288
10.3.2 PCI 总线的特点	290
10.3.3 PCI 总线信号定义	291
10.3.4 PCI 总线事务	294
10.4 USB 总线	295
10.4.1 概述	295
10.4.2 USB 体系结构	296
10.4.3 USB 总线协议	301
习题 10	304
参考文献	306

第1章 微机基础

1.1 计算机中的数和字符

1.1.1 进位制

进位计数制是最常用的数值表示方法。一个数由一定数目的数码排列在一起组成，每个数码的位置称为数位，每个数位上所能使用的数码的个数称为基数，每个数位上的数码所代表的数值是数码乘以该数位的位权，位权是基数的次幂。运算中，某一数位累计到基数以后向高数位进一。

数 N 使用 r 进制，即基数为 r 的进位计数制，可以表示为

$$(d_{n-1}d_{n-2}\cdots d_1d_0.d_{-1}\cdots d_{-m})_r$$

其数值是

$$N = d_{n-1} \times r^{n-1} + d_{n-2} \times r^{n-2} + \cdots + d_1 \times r^1 + d_0 \times r^0 + d_{-1} \times r^{-1} + \cdots + d_{-m} \times r^{-m}$$

式中， d_i 是 r 个数码 0、1、 $\cdots$ 、 $(r-1)$ 中任意一个； $i=n-1, n-2, \cdots, 1, 0, -1, \cdots, -m$ 是数位； r^i 为数位 i 的位权。

十进制是日常使用的计数制。十进制的基数是 10，数码是 0、1、2、3、4、5、6、7、8、9，遵循逢十进一的规则。例如：

$$100420 = 1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 4 \times 10^2 + 2 \times 10^1 + 0 \times 10^0$$

计算机中为便于存储以及计算的物理实现，普遍采用二进制。二进制的基数是 2，只有 0 和 1 两个数码，遵循逢二进一的规则。例如：

$$(101011)_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = (43)_{10}$$

在计算机中，还经常使用十六进制。十六进制的基数是 16，数码是 0~15，其中 10、11、12、13、14、15 分别用 A、B、C、D、E、F 表示，遵循逢十六进一规则。例如：

$$(5AF)_{16} = 5 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = (1455)_{10}$$

【例 1.1】 将十进制数 420 分别转换为二进制和十六进制。

十进制数转换为二进制或十六进制的方法是：用十进制数分别不断除以 2 或 16，记下每次相除时的余数，直到商 0 为止。过程如下：

2	214	余0= d_0
	107	余1= d_1
	53	余1= d_2
	26	余0= d_3
	13	余1= d_4
	6	余0= d_5
	3	余1= d_6
	1	余1= d_7
	0	

16	214	余6= d_0
	13	余13= d_1
	0	

所以， $(420)_{10}=(11010110)_2=(D6)_{16}$ 。

从上例可以看出，二进制的数码只有 0 和 1，数的二进制表示形式较长，在书写时不太方便，阅读也不直观。十六进制实际上可以看成是由于 $2^4=16$ 而从二进制衍生而来的，一个十六进制数码正好与 4 位二进制数码一一对应，其对应关系如表 1.1 所示。二进制和十六进制相互转换十分方便：将 4 位二进制码合在一起转为对应的十六进制数码或将一个十六进制数码转换为对应的 4 位二进制码。因此，在计算机领域经常使用十六进制的书写代替二进制的书写。在汇编语言里，十进制用后缀 D（默认为十进制）、二进制用后缀 B、十六进制用后缀 H 表示，不区分大小写。

表 1.1 十六进制数码与二进制数码对应关系

二进制	0000	0001	0010	0011	0100	0101	0110	0111
十六进制	0	1	2	3	4	5	6	7
二进制	1000	1001	1010	1011	1100	1101	1110	1111
十六进制	8	9	A	B	C	D	E	F

二进制、十六进制与十进制都遵循相同的进位计数制规则，差别只在于基数不同。所以，它们的运算方法和我们已知的十进制运算是类似的，只需用“逢二进一”、“逢十六进一”代替“逢十进一”即可。

【例 1.2】 计算 $298H+72H=?$ ， $298H\times 6CH=?$

$$\begin{array}{r} 298H \\ + \quad 72H \\ \hline 30AH \end{array}$$

注意：8+2=10 没有进位，9+7=16 才有进位。

$$\begin{array}{r} 298H \\ \times \quad 6CH \\ \hline 1F20 \\ + \quad F90 \\ \hline 11820H \end{array}$$

1.1.2 数的机器表示

计算机中的数是用二进制表示的。数码 0 和 1 由二值器件的两个稳态来表示，我们称之为二进制位（bit），它是计算机中信息存储的最小单位，一般用 b 表示；8 个相邻的二进制位构成一个字节（Byte），一般用 B 表示。由于机器资源总是有限的，决定了数的机器表示和数学表示是不同的，我们把前者称为机器数，后者称为真值。

1. 定点数

定点数是指小数点位置固定。在微机中，小数点位置固定在最低位，即机器中所有的定点数都是纯整数。

在微机中，对于定点数存在无符号数和有符号数的概念。无符号数表达的是大于等于 0 的数，在机器表示时不需要表示符号信息。有符号数存在符号信息，在机器表示时最高位用来表示符号，“1”表示负，“0”表示正。在计算机中，常用像 8 位、16 位、32 位、64 位等字节整数倍的位数进行定点数的表示，位数越多数的表示范围越大，如表 1.2 所示是无符号数和有符号数的表示方法和表示范围。

表 1.2 无符号数和有符号数的表示方法和表示范围

位 数	无 符 号 数		有 符 号 数	
8 位	8 位全部参与数的表示		最高位表示符号，其余 7 位表示绝对值	
	表示方法	表示范围	表示方法	表示范围
	00000000	0~(2 ⁸ -1=255)	0 0000000	0~(2 ⁷ -1=127)
			0 1111111	
	11111111		1 0000001	-1~(-(2 ⁷ -1)=-127)
	1 1111111			
16 位	16 位全部参与数的表示		最高位表示符号，其余 15 位表示绝对值	
	表示方法	表示范围	表示方法	表示范围
	0000000000000000	0~(2 ¹⁶ -1=65535)	0 000000000000000	0~(2 ¹⁵ -1=32767)
			0 111111111111111	
	1111111111111111		1 000000000000001	-1~(-(2 ¹⁵ -1)=-32767)
	1 111111111111111			

在表 1.2 中，有符号数的表示方法是：最高位是符号位，其余位表示绝对值。这种表示方法称为原码。原码的表示方法常用来描述真值。在计算机中，有符号数的机器表示是补码，补码的表示方法为：

$$[X]_{\text{补}} = \begin{cases} X & X \geq 0 \\ 2^n + X = 2^n - |X| & X < 0, n \text{ 是位数} \end{cases}$$

通俗的描述是：0 和正数的补码就是本身，负数的补码是绝对值求反加 1。

【例 1.3】 求 1, 46, -1, -46 的 8 位补码。

正数的补码就是本身，所以 1 的补码就是[00000001]_补。

同理，46 的补码是[00101110]_补。

负数的补码是绝对值求反加 1，所以 -1 的补码就是[11111111]_补（00000001 求反为 11111110，再加 1 得到 11111111）。

同理，-46 的补码是[11010010]_补（00101110 求反为 11010001，再加 1 得到 11010010）。

我们都知道，时钟转一周是 12 小时，加 1 小时是将时针顺时针拨 1 格，而减 1 小时可以将时针逆时针拨 1 格，也可以顺时针拨 11（即 12-1）格，这样，在时钟上 -1 和 +11 是一样的。与时钟类似，如果使用 8 位来表示有符号数，它的一周就是 2⁸=256，所以 -1 就可以用 255（即 2⁸-1）来表示，这就是补码的含义。

再次强调：微机中有符号数的机器表示是补码。在微机中，将“求反加 1”操作称为求补操作，又称为算术求反（在数的前面加上负号）。补码具有以下特性：

$$[-X]_{\text{补}} = [X]_{\text{补}} \text{ 的求补}$$

$$[X+Y]_{\text{补}} = [X]_{\text{补}} + [Y]_{\text{补}}$$

$$[X-Y]_{\text{补}} = [X]_{\text{补}} + [-Y]_{\text{补}}$$

由第三个补码特性可以看出，补码的减法运算可以变为加法运算。这样，CPU 中就可使用加法器直接实现减法，而不需要再设置专门实现减法的部件了。

【例 1.4】用 8 位补码计算 $32-7=?$

0010 0000

- 0000 0111

0010 0000

+ 1111 1001

进位的1被丢弃 ① 0001 1001

计算中，最高位（符号位）产生的进位超出了机器资源被丢弃。
所以， $32-7=25$ 。

8 位补码和 16 位的表示范围如表 1.3 所示。一般来说， n 位补码的表示范围是：

$-2^{n-1} \leq N \leq 2^{n-1}-1$

表 1.3 补码表示范围

8 位补码表示范围		16 位补码表示范围	
0 1111111	127	0 1111111111111111	$2^{15}-1=32767$
⋮	⋮	⋮	⋮
0 0000000	0	0 0000000000000000	0
1 1111111	-1	1 1111111111111111	-1
⋮	⋮	⋮	⋮
1 0000000	-128	1 0000000000000000	$-2^{15}=-32768$

注意：有符号数在进行位数扩展（8 位扩展到 16 位、16 位扩展为 32 位等）时，正数（符号位为 0）高位补 0，负数（符号位为 1）高位应补 1，称为符号扩展。例如：

$[00000001]_{补}=[00000000\ 00000001]_{补}=1$ ， $[11111111]_{补}=[11111111\ 11111111]_{补}=-1$

2. 浮点数

任意一个带符号的二进制数 N 使用科学计数法可以表示为 $N=(-1)^S \times 2^E \times M$ ，式中， S 是数的符号； E 是指数（阶码）； M 是尾数。计算机用存储尾数和指数的方式来表达实数，这就是浮点表示法。浮点数利用指数达到了浮动小数点的效果，从而可以灵活地表达更大范围的实数。

浮点数由阶码（指数）和尾数及其符号位组成。其中，尾数用定点小数表示，它给出了实数的有效数字，尾数的位数决定了浮点数的表示精度；阶码用整数形式表示，它指明小数点在数据中的位置，阶码位数决定了浮点数的表示范围。微机使用有限的连续字节保存浮点数，单精度浮点数（32 位）和双精度浮点数（64 位）的格式定义如图 1.1 所示。

IEEE单精度浮点数 $(-1)^S \times (1.M) \times 2^{E-127}$

符号S	阶码E	尾数M
1位	8位	23位

IEEE双精度浮点数 $(-1)^S \times (1.M) \times 2^{E-1023}$

符号S	阶码E	尾数M
1位	11位	52位

图 1.1 浮点数标准格式

1.1.3 字符编码

现代计算机不仅处理数值领域的问题，而且处理大量非数值领域的问题。非数值信息必然需要引入数字、字母、专用符号等抽象符号，这些抽象符号统称为字符。字符在计算机内部也需要以二进制形式表示，称为字符的二进制内码。字符编码规定了字符和其二进制内码的对应关系。

ASCII 码（美国国家信息交换标准字符码）是国际上普遍采用的字符编码。ASCII 码出现于 20 世纪 60 年代末，是为保证人和设备、设备和计算机之间能进行正确的信息交换所编制的统一的信息代码。微机的字符外设一般常采用 ASCII 码。ASCII 码包括 32 个通用控制字符、10 个阿拉伯数字、52 个英文大小字母及 34 个运算符、标点符号和其他专用符号，共 128 个字符，二进制编码需 7 位（占一字节，最高位置 0）。表 1.4 是标准 ASCII 码表。

表 1.4 标准 ASCII 码表

高 3 位 低 4 位	000	001	010	011	100	101	110	111
0000	NUL	DLE	(Space)	0	@	P	'	p
0001	SOH	DC1		1	A	Q	a	q
0010	STX	DC2	`	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	"	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

1.2 微机的基本结构

1.2.1 冯·诺依曼结构

虽然微型计算机发展迅速，但至今为止其硬件结构仍采用冯·诺依曼建立的经典结构。这种结构的主要特点是：微型计算机系统的硬件由五大部分组成，如图 1.2 所示。这五部分是：运算器、控制器、存储器、输入设备和输出设备。其中运算器和控制器合称为微处理器 CPU，输入/输出由多个 I/O 接口和外部设备组成。微机的各个组成部分通过总线连接起来。总线由数据总线 DB（Data Bus）、地址总线 AB（Address Bus）和控制总线 CB（Control Bus）组成。

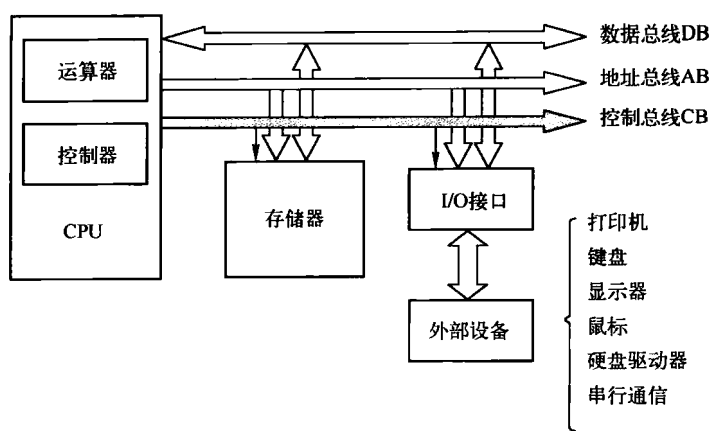


图 1.2 微机硬件结构

1.2.2 微机的硬件组成

1. 中央处理器

中央处理器（Central Processing Unit, CPU）由运算器和控制器组成，是计算机系统中的核心部件。控制器是整个计算机系统的指挥中心，负责对指令进行分析，并根据指令的要求，有序、有目的地向各个部件发出控制信号，使计算机的各部件协调一致地工作。运算器是对数据进行加工处理的部件，它在控制器的作用下与存储器交换数据，负责进行各类基本的算术运算、逻辑运算和其他操作。

图 1.3 是一个简单 CPU 的模型。CPU 一般都包括控制逻辑、运算器（ALU）、寄存器、暂存器和内部总线等部件。

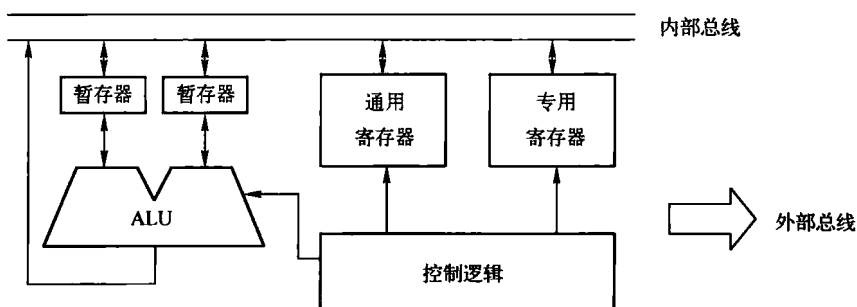


图 1.3 简单 CPU 的模型

CPU 可以完成的一个基本操作称为指令，CPU 中用来计算和控制计算机系统的一套指令的集合称为指令集或指令系统。所以，不管 CPU 内部是如何实现的，它的功能都是由指令集体现的。CPU 可以执行的指令愈多，在一定程度上代表其功能越强大，但也会带来控制复杂等负面影响。

寄存器也是 CPU 的一个重要组成部分，它是 CPU 内部的临时存储单元（与 CPU 速度相同）。寄存器既可以存放数据和地址，也可以存放控制信息或 CPU 工作的状态信息。为了减少 CPU 访问内存的次数和降低指令控制的复杂度，CPU 的许多指令都是依赖寄存器的，

所以 CPU 的寄存器结构也是 CPU 功能的一个重要体现。

字是计算机内部进行数据处理和数据传递的基本单位，其所包含的二进制位数称为字长。字长一般来说和运算器、寄存器、总线宽度是一致的，它是计算机处理精度和运算能力的反映。例如，80386 CPU 字长是 32 位，即 CPU 一次就可以处理 32 位的数据，其能力高于字长是 16 位的 8086 CPU，后者一次仅可以处理 16 位的数据。

运算速度是衡量 CPU 性能的一项重要指标。通常所说的计算机运算速度（平均运算速度），是指每秒钟所能执行的指令条数，一般用“百万条指令 / 秒”（Million Instruction Per Second, MIPS）来描述。

2. 存储器

存储器是计算机系统内最主要的记忆装置。在微机中，CPU 需要执行的程序与需要处理的数据都存放在存储器中。

不管存储器如何组织和实现，从逻辑上可以将其看做一个由存储单元构成的线性表，如图 1.4 所示。通过指定从 0 开始的序号，就可访问相应存储单元。存储单元的序号就是存储单元的地址。

在微机系统中，存储器是按字节编址的，即一个存储单元存储一个字节信息。

存储器的容量和速度对微机系统性能影响是很大的。存储容量是指存储器所能容纳二进制信息的总量，它反映了微机即时存储信息的能力。容量大，能存储的字数就多，能直接接纳和存储的程序就长，计算机的解题能力和规模就大。另一方面，计算机中大量的操作都需要与存储器打交道，因此，存储器的速度往往又是影响计算机数据处理速度的一个主要因素。存取速度通常用存取时间来衡量，指从启动一次存储器操作到完成该操作所经历的时间。

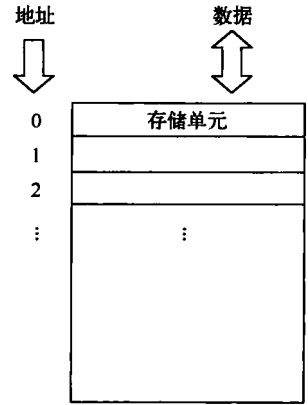


图 1.4 存储器模型

3. I/O 系统

微机的 I/O 系统负责与人或其他系统进行信息交换，由 I/O 接口和外部设备构成。

外部设备包括输入设备和输出设备。常用的输入设备是键盘、鼠标、数字扫描仪及模/数转换器等，它们的作用是获取外部信息，在获取过程中需要把外部信息变换为机器内部所能接收和识别的信息形式。输出设备的作用是把计算机处理的结果变换为人或其他机器设备所能接收和识别的信息形式，常用的输出设备有显示器和打印机等。外部设备，特别是负责与人打交道的人机设备（它们需要识别或展现人所容易理解的信息形式），在功能和实现上存在很大的差异，各式各样，种类繁多。

由于种类繁多且速度各异，所以外部设备不是直接与高速工作的主机（CPU 和存储器）相连接，而是通过 I/O 接口与主机相联系。I/O 接口负责外部设备与主机通信的转换与缓冲，屏蔽外部设备的差异。

外部设备的速度和 CPU 速度存在巨大的差距，由其引发的 I/O 瓶颈一直是微机系统需要面对的一个重要问题。