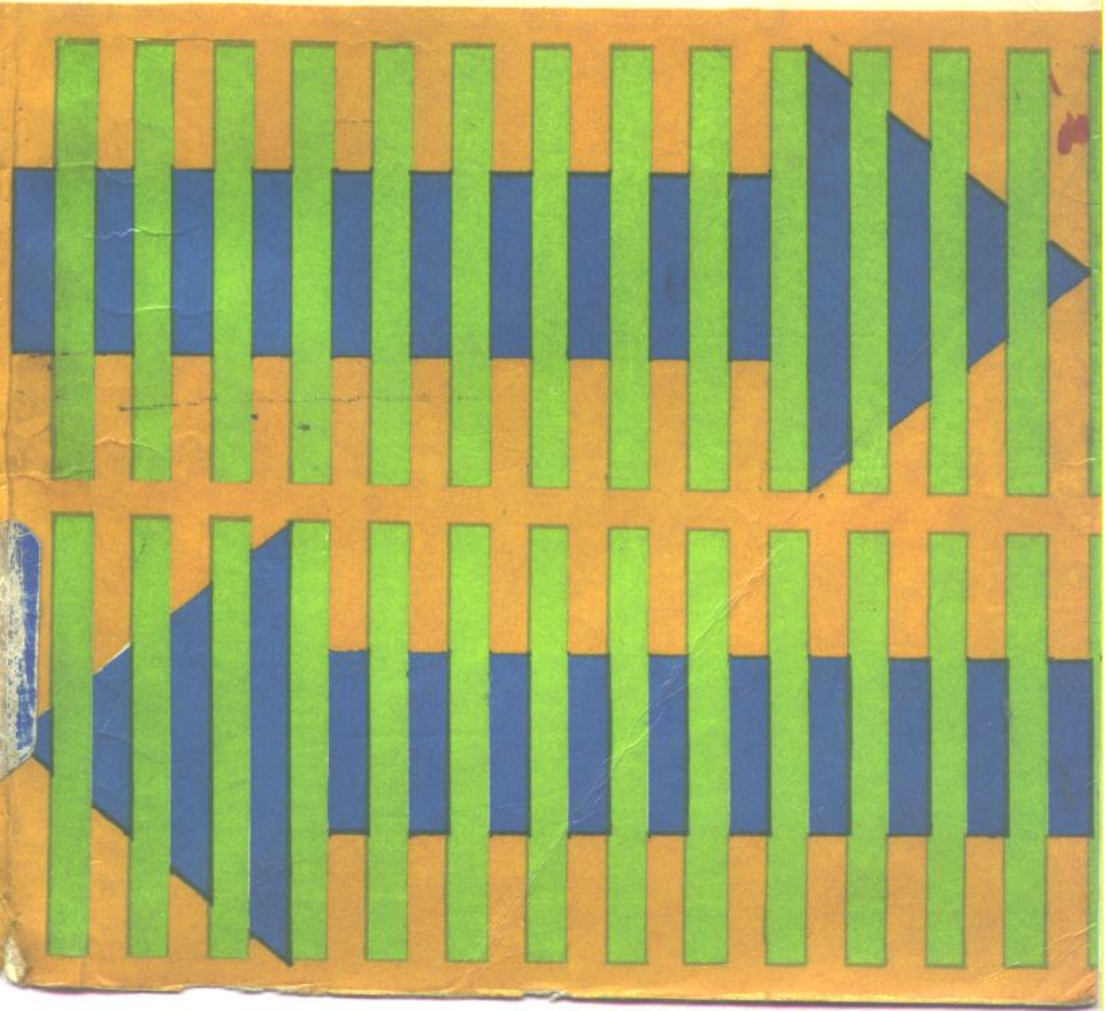
微型计算机系统开发

林国璋 编著

化学工业出版社



微型计算机系统开发

林国璋　编著

电子工业出版社

内 容 提 要

本书以目前广为流行的IBM-PC/XT微型计算机为例，深入地讨论了微型计算机系统开发的原理和技术。这是一本面向开发的理论与实际紧密结合的教科书，作者由浅入深，逐步展开，系统而又深入地讨论了微型计算机的基本结构（包括硬件结构和操作系统结构），软硬件开发的基本技术、屏幕支持系统、图形支持系统、磁盘支持系统、汉字支持系统以及串行通讯支持系统等重要课题。本书可作为大专院校计算机专业或其它专业的教科书，也可以作为从事计算机系统和应用开发的工程技术人员的参考书。

微型计算机系统开发

林国璋 编著

责任编辑：刘 哲
封面设计：任 辉

*

[illegible]出版发行
（北京和平里七区十六号楼）
北京印刷一厂印刷
新华书店北京发行所经销

*

开本 850×1168 1/32 印张 14 3/8 插页 1 字数 461千字
1989年10月第1版 1989年10月北京第1次印刷
印 数 1—7520
ISBN 7-5025-0502-4/TP·14
定 价 8.10 元

前　言

自七十年代初期微处理器问世以来，微型计算机以两年一翻番的速度迅猛发展。今天，微型计算机的应用已经深入到社会的各个领域，对科学技术的发展和社会各行各业的现代化变革产生着深刻的影响。

近几年来，我国微型计算机的研究和应用工作也在迅速发展。尤其是16位微型计算机进入我国市场以来，在微型计算机的研究和应用上提高到了一个崭新的水平。汉字支持系统的研究与开发成果使得在我国普及计算机应用成为现实。微型计算机局部网络及其数据库管理技术的开发应用，正在使我国的行政办公、企业管理走向自动化。微型计算机图形支持系统和各种CAD、CAM软件也正在促进工程设计、产品设计和产品制造的电脑化，缩短了研制周期。并从根本上改善了产品质量和经济效益。为了充分发挥微型计算机在我国各个领域工作中的作用，需要有更多的人掌握微型计算机的开发技术，为此，作者编写了这本《微型计算机系统开发》，希望能对读者有所帮助。

本书以目前广为流行的IBM-PC微型计算机为例，讨论微型计算机系统开发的原理与技术。本书假定读者已经具备计算机原理、汇编语言程序设计的基础知识，并且对IBM-PC系列微机的软硬件有初步了解。

本书第一章讨论微型计算机的系统结构。我们将以INTEL 8086或8088CPU为主讨论三代微型机的结构特点，其中包括CPU结构、内存结构、中断结构、单处理机系统结构以及多重处理机系统结构。

第二章讨论IBM-PC的硬件结构与操作系统软件结构。本章将从用户应用与开发的角度深入讨论这类机器的硬件结构特点与

DOS 结构特点。硬件结构包括系统结构、中断结构、主要接口芯片的规划与应用；操作系统（DOS）结构包括 DOS 的层次结构、ROMBIOS、DOS 定义的中断与功能调用。

第三章讨论软件开发技术。包括程序设计技术、用开发户的.EXE文件和.COM文件的汇编语言源程序的结构要求和设计方法以及对 DOS 系统的扩充技术。程序设计技术部分扼要地介绍了模块程序设计技术、结构程序设计技术、自顶向下和自底向上的程序设计技术，说明了这几种程序设计技术之间的关系和应用。在.EXE 文件和.COM 文件的汇编语言源程序的设计部分，我们讨论了文件与操作系统之间的关系，外部文件如何正确返回系统；如何传递和使用命令随带的参数，如何从当前程序调用另一个外部文件，以及如何保证两种文件在源程序结构上的正确性。最后我们讨论了 DOS 系统的扩充技术。

第四章和第五章讨论屏幕处理。在这两章中全面地分析了 ROMBIOS 中的 CRT 驱动系统，包括字符处理、光标处理、窗口功能、卷轴功能、图形功能以及 TTY 驱动系统，并给出了全部源程序和必要的框图。读者务必认真地学习并掌握这两章的内容，这不仅因为屏幕处理在系统开发中所占地位的重要性，而是也是本书第七章汉字支持系统的必备基础。

第六章讨论磁盘 I/O 支持系统。在这一章中我们讨论了DOS 的文件管理、文件结构、磁盘格式以及磁盘 I/O 支持系统。本章着重从开发的角度讨论了文件存取的方法和磁盘处理技术。

第七章讨论汉字支持系统。讨论了汉字支持系统与西文 DOS 的软件界面、汉字处理系统中的数据结构、基本映射。进而，又深入地讨论了中西文兼容环境下 CRT 驱动系统、键盘驱动系统与打印机驱动系统的设计原理和程序框图。我们还讨论了系统的生成与安装方法。本章以目前较为流行的几种汉字支持系统为例，并结合作者亲身参加汉字支持系统设计的经验与体会，讨论了汉字支持系统开发与设计中的若干重要问题，如中西文兼容性问题、字典覆盖问题、字库驻留问题、重码处理问题以及输入方式的选择与转换等

问题。认真地学习本章内容，不仅可以掌握汉字支持系统的开发原理与技术，而且可以从中学到许多软件开发的基本知识和基本技能。

第八章讨论通讯支持系统。我们讨论了串行通讯的基本概念、通用异步收发器 UART 的工作原理、调制解调器的应用、 RS 232 接口的使用方法以及通讯支持系统的基本软件。 最后我们结合 IBM-PC 机的 RS232 接口，给出了一个把 PC 机作为终端使用的通讯程序例子。

微型计算机系统开发的内容涉及面广，要求读者具有较好的理论和实践基础。本书力求理论与实践结合，普及与提高结合，由浅入深，由表及里，由简到繁，使一般的读者能顺利地读完它。

由于作者水平有限，又加时间仓促，书中所述，错误难免，恳请各位读者多提宝贵意见。

作者　林国璋

一九八六年四月一日于北京工业学院

目　　录

第一章　微型计算机系统结构

本章以IBM-PC/XT系列微型计算机为例，概述微型计算机的系统结构，包括硬件结构与操作系统的软件结构。掌握微型计算机的系统结构特点，是开发微型计算机系统的必备基础。我们假定读者对于微型计算机的原理已经有了一定的基础，因此，讨论着重在结构、特点上。对于微型计算机的原理知识还缺乏了解的读者可预先阅读有关书籍。

第一节　中央处理机（CPU）结构

1.1.1 INTEL 8086的CPU结构

INTEL 8086和INTEL 8088是INTEL公司推出的两种十六位微处理器芯片。两者在结构上基本相同，但前者是全十六位微处理机，而后者是准十六位微处理机。8086或8088 CPU是十六位微处理机中的典型产品。目前广为流行的IBM-PC/XT系列微机以及各类兼容机均是采用8086或者8088芯片作为中央处理机单元。图1-1给出了8086CPU的结构方块图。

注意，INTEL 8086或8088在结构上分成两个独立的部分：执行单元EU和总线接口单元BIU。这与八位微型计算机CPU的结构完全不同。EU只完成执行指令的功能，而BIU则负责从存贮器或外部设备中读取操作码、操作数，并将结果写到目标有效地址中。因此，8086或8088的所有的总线操作均由BIU执行，故称为总线接口单元（BUS INTERFACE UNIT）。

执行单元EU包括一个16位的算术逻辑单元ALU、一个16位的反映CPU状态和运算结果标志的状态标志寄存器、一组通用寄存器、数据暂存器和执行单元的控制逻辑。8086 CPU所有的寄存器和数据传输通路都是16位数。它们之间可以通过ALU数据总线

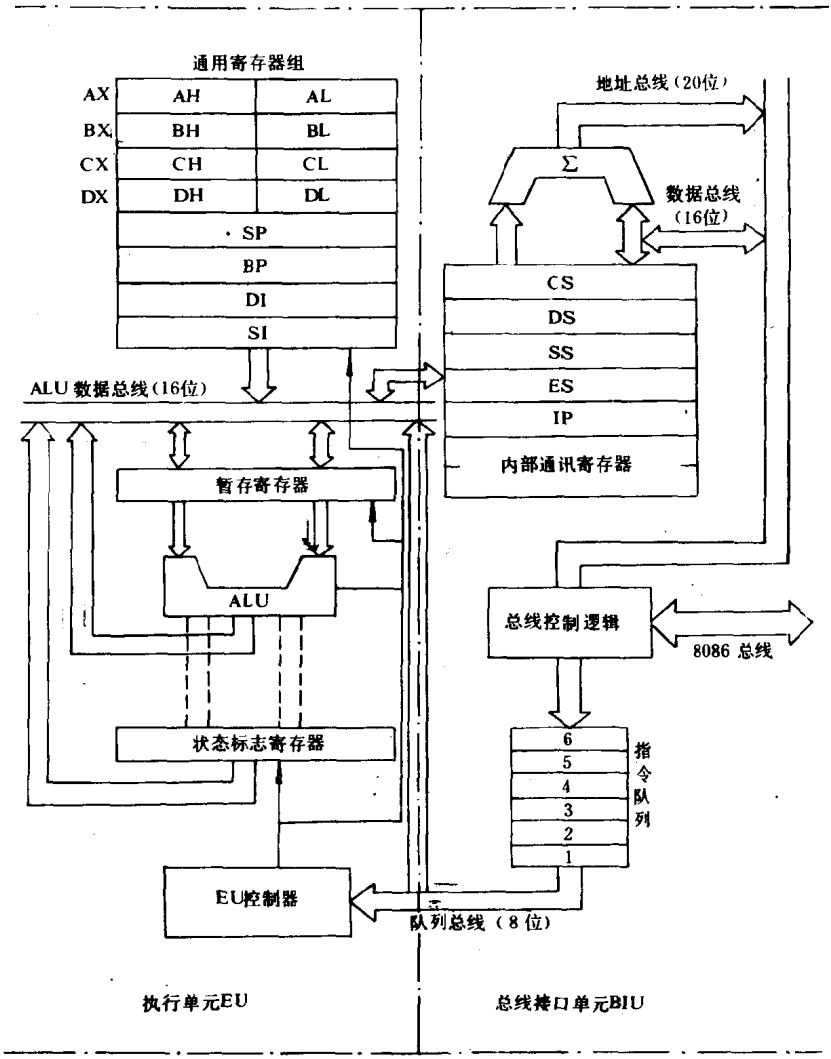


图 1-1 8086 CPU结构框图

实现快速的内部数据传送。应当注意到，EU本身不与系统总线相连，它从 BIU 的指令队列寄存器中取得指令和数据。当指令要求把数据写到存贮器或外部设备中，或需要从存贮器或外部设备中读

取数据时，由EU向BIU发出请求。BIU根据EU发出的请求完成相应的工作。EU主要完成两种操作：第一种是完成所有的算术或逻辑运算；第二种是计算出指令要求寻址的内存地址的偏移量，并把一个16位的地址偏移量送到BIU中，由BIU形成一个20位的物理地址，以寻址1 M字节的内存空间。

总线接口单元BIU包括一组内存段寄存器（CS、DS、SS和ES）、一个指令计数器IP、6个字节的指令队列寄存器、地址产生器和总线控制器。如上所述，BIU是总线接口单元，它负责全部CPU与内存或外部设备的数据交换工作。8086或8088采取了指令先行的技术，在EU执行指令的过程中，BIU始终“向前看”(looks ahead)，从存贮器中预先取出一些指令，放在指令队列之中。指令队列寄存器是能够存放6个字节代码的RAM存贮器。取来的指令按字节顺序存入队列。队列是一种先进先出的栈。在CPU执行指令的过程中，BIU并不总占用系统总线。当BIU不使用系统总线时，系统总线可以供其它部件使用。BIU在下面两种情况下自动执行取指操作：一种是当指令队列中出现两个以上的指令字节空的时候，无需由EU发出请求，BIU自动执行总线操作取指令，通常BIU一次可以取得两个指令字节；第二种情况是当程序发出转移时，BIU也自动执行取指操作。如果程序是转移到一个奇数地址时，BIU先从这个奇数地址中取得一个字节，然后再从这个奇数地址后面开始的偶数地址中，两个字节两个字节地读取。

在大多数情况下，指令队列中至少应有一个字节的指令代码，这样EU就不必等待BIU去读取指令。如果EU执行一条转移指令，那么预先存放在队列中的其它指令显然就不能再使用了。因此后续要执行的指令应当根据转移指令给出的地址重新读取。当EU执行转移指令时，向BIU发出控制信息，BIU新取得的第一条指令将直接送到EU中去执行，然后BIU将随后取得的指令填入指令队列寄存器。在转移指令之前指令队列中所存放的指令被覆盖掉。

EU在执行指令的过程中，可能要求BIU进行存贮器或I/O设备的数据存取操作。此时，BIU停止取指操作，按照EU的要求执行存贮器或I/O设备的数据存取总线操作。如果BIU在执行EU发来的总线请求以前已经准备好去取下一条指令时，EU的总线请求要等到下条指令取出之后才能执行。

1.1.2 寄存器结构

以汇编语言为开发工具的程序设计师十分关心CPU的寄存器结构。汇编语言程序所涉及的操作数及其目标与寄存器直接相关。

8086或8088 CPU中有13个16位的寄存器和9个1位的状态标志寄存器。根据其不同的用途，我们将其分成四个组，其中三个组是能够由程序进行存取操作的。这三组寄存器分别称为数据寄存器组、指针与变址寄存器组和内存段寄存器组。这三组中的每组都有4个16位的寄存器。另外一组寄存器包括一个16位的指令计数器(指令指针)和一个9位的状态标志寄存器。状态标志寄存器也可以通过程序进行存取。指令指针由CPU自动调整，指向下一条指令的地址（偏移值）。图1-2示出了8086或8088 CPU中的寄存器结构。

数据寄存器一般用来存放数据。它们分别为AX、BX、CX和DX。每个16位的数据寄存器又都可以分别以高8位和低8位的八位寄存器使用并兼容于8位的8080系统。所以，这组寄存器有时称为H&L组。在大多数情况下，我们可以任意使用数据寄存器组进行算术逻辑运算。但是一般我们总是把AX作为累加器使用，把BX作为基址寄存器使用，把CX作为计数寄存器使用，把DX作为数据暂存器使用。

指针和变址寄存器主要用于寻址内存操作数。其中SP是堆栈指针寄存器，或称当前堆栈栈顶指针。BP称为基址指针寄存器。一般来说，这两个指针寄存器用于存取位于当前堆栈段中的数据。如果不加说明，SP总是表示当前堆栈的栈顶，而BP总是表示当前堆栈的偏移量。换句话说，堆栈操作指令使用SP作为堆栈操作

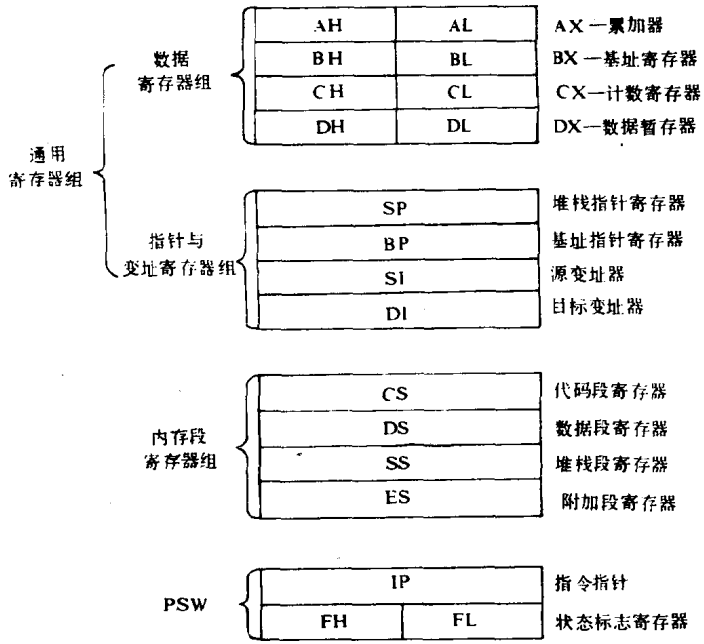


图 1-2 8086/8088 CPU的寄存器结构

数指针，而其它的数据操作指令使用 BP 作为堆栈操作数指针。SI和DI通常用来作为源操作数和目标操作数的指针。在串操作指令中，SI总是寻址当前数据段(DS段)，而DI总是寻址当前附加段(ES段)，SI和DI同时被自动修改，指向下一个地址。但是在一般的数据存取指令中，SI和DI均寻址当前数据段(DS)，除非你使用段超越前缀。

内存段寄存器组用来存放段的起始地址。CS用来给出当前的代码段（Code segment），CPU执行的指令是从代码段中取得的。SS用来给出程序当前使用的堆栈段（Stack Segment），堆栈操作的执行地址在这个段中。DS指出了程序当前使用的数据段(Data Segment)，一般来说，程序涉及到的变量和数据存放在这个段中。ES指出了程序当前使用的附加段（Extra Segment)，附

加段的典型用法是用来存放处理以后的数据，在串操作指令中，附加段是不可缺少的。

指令指针 IP 和状态标志寄存器反映了程序的当前状态。IP 值表示了程序下一条指令在 CS 段中的偏移地址，而状态标志寄存器却反映了程序当前指令的执行结果和 CPU 当前所处的状态。因此这两个寄存器可以说是程序状态字 PSW (Program Status Word)。注意，8086或8088 CPU有 9 个状态标志位，如图1-3所示。

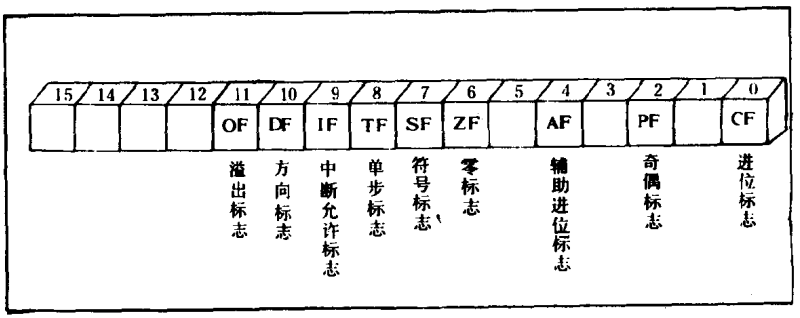


图 1-3 8086或8088的状态标志寄存器

CF 是进位标志位。CF＝1 表示指令执行后的结果在最高位上产生了一个进位或借位信号。CF＝0 则表示指令执行后没有产生这种信号。

PF 是奇偶标志位。PF＝1 表示指令执行的结果代码中有偶数个“1”；PF＝0 则表示指令执行的结果代码中有奇数个“1”。

AF 是辅助进位标志位。AF＝1 表示指令执行后从低位字节的第 4 位上产生了一个进位或借位信号；AF＝0 表示没有产生这种信号。辅助进位在十进制调整中是必要的。

ZF 是零标志位。ZF＝1 表示当前指令执行后结果为零；ZF＝0 表示当前结果不为零。

SF 是符号标志位。SF＝1 表示当前指令执行结果最高位为“1”，SF＝0 表示指令执行后结果的最高位为“0”。

OF是溢出标志位。OF＝1表示当前指令执行时产生算术溢出，OF＝0表示无溢出产生。

IF是中断允许标志位。它反映CPU的一种状态。IF＝1表示CPU允许接受外部可屏蔽中断发来的中断请求，CPU的这种状态称为中断允许状态或开中断状态；IF＝0则表示CPU不接受任何可屏蔽中断，CPU的这种状态称为中断禁止状态，或称关中断状态。注意，IF标志不会影响到非屏蔽中断和内部中断。

DF是方向标志位。DF＝1表示串操作指令将以递减数据指针的方向对数据串进行处理，DF＝0则以递增数据指针的方向处理数据串。

TF是单步标志位。它也反映CPU的一种状态。TF＝1表示处理机处于单步工作方式。在这种方式下，CPU每执行完一条指令就自动产生一个内部中断，处理机转去执行一个中断程序，这个中断称为单步中断。TF＝0时CPU正常执行程序。

第二节　存贮器结构

计算机的存贮器结构是系统开发者必须关心和掌握的另一个问题，因为程序开发中必须考虑到存贮器的分配与使用。

8086或8088 CPU能够寻址1 M字节的内存空间。8086或8088有20根地址线，给出一个20位的地址，便在1 M的地址空间中唯一地确定一个字节的地址。但是从8086或8088的CPU结构中我们看到，8086或8088 CPU只能进行16位的算术运算，能够处理的地址码也只有16位长，所有与寻址有关的寄存器，包括指令指针IP也都是16位寄存器。那么CPU如何寻址1 M字节的内存空间呢？原来8086或8088 CPU把1 M字节的地址空间逻辑上划分成任意的一些段，每个段的最大长度为64 k字节，段内的地址是连续的，而段与段之间是相互独立的。这就是CPU结构中为什么要有内存段地址寄存器的原因。根据程序的要求，由程序的每一个段规定一个基本地址，称为基址（Base Address）。段基址指定了段的开始地址。段基址必须能被16整除，即段基址地址码的最后四位应当为