微机原理与接口技术

吴瑞坤　主编

厦门大学出版社
XIAMEN UNIVERSITY PRESS
国家一级出版社
全国百佳图书出版单位

微机原理与接口技术

吴瑞坤　主编

厦门大学出版社
XIAMEN UNIVERSITY PRESS
国家一级出版社
全国百佳图书出版单位

图书在版编目（CIP）数据

微机原理与接口技术/吴瑞坤主编．—厦门：厦门大学出版社，2014.11
ISBN 978-7-5615-5289-6

Ⅰ.①微… Ⅱ.①吴… 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（2014）第 255882 号

厦门大学出版社出版发行
（地址：厦门市软件园二期望海路 39 号　邮编：361008）
总 编 办 电 话：0592-2182177　传真：0592-2181253
营销中心电话：0592-2184458　传真：0592-2181365
网址：http://www.xmupress.com
邮箱：xmup@xmupress.com
沙县四通彩印有限公司印刷
2014 年 11 月第 1 版　2014 年 11 月第 1 次印刷
开本：787 × 1092　1/16　印张：17
字数：420 千字　印数：1~2 000 册
定价：35.00 元
本书如有印装质量问题请直接寄承印厂调换

内容简介

本书从微型计算机系统出发，较全面介绍了8086/8088系统微型计算机的组成及接口技术。主要内容包括：8086/8088微处理器结构，8086/8088系统的存储器结构、工作模式和总线周期，CPU与存储器的连接，8086/8088 CPU的指令系统，汇编语言程序设计和常用的DOS功能调用，输入输出技术与模拟数字通道接口，中断技术与可编程中断控制器，可编程并行接口芯片8255A和可编程串行接口芯片8251A，可编程定时/计数器8253A等。

本书结构严谨，内容选择与章节安排结合了编者多年的理论与实验教学的经验，力求通俗易懂，每章都配有丰富的实例。可作为高等院校电子信息类、计算机类、电气自动化及相关专业教材，也可作为从事微机软、硬件工作的工程技术人员的参考用书。

前 言

“微机原理与接口技术”是电子信息、计算机技术、自动控制、通信工程等专业的专业基础必修课。它主要讲授微型计算机的基本工作原理、系统组成、指令系统、汇编语言程序设计及接口技术，介绍通用可编程接口芯片，说明工作原理和基本应用。本书内容兼顾硬件和软件两个方面。通过本课程学习，学生可为今后在计算机自动检测控制与计算机信息处理等相关领域的研究开发打下良好基础。学习本课程，要求学生达到以下目的，即掌握微型计算机的基本原理、基本组成和系统结构，深入理解微处理器与存储器结构、指令系统、汇编语言程序设计、中断技术、输入/输出接口技术与模拟及数字通道接口技术，熟练掌握基本的软件编程方法和可编程硬件接口技术。

本书在内容组织方面，首先是微机系统组成，包括 8086/8088 CPU 结构、8086/8088 系统的工作模式和总线周期，以及 8086/8088 系统的存储器结构、半导体存储器与 CPU 的连接问题等。其次是关于 8086/8088 CPU 指令系统及汇编语言程序设计方法、DOS 功能调用等。最后是关于接口技术，包括输入/输出接口的编址方法、数字通道接口、A/D转换器及 D/A 转换器接口、中断技术与可编程中断控制器、可编程并行接口和串行接口芯片、可编程定时/计数器等。本书综合考虑这门课程课时少的特点，以满足教学需要为要求进行编写，在注重系统性、完整性和可实践性的前提下，尽可能做到内容精简。本书各章节都有完整的应用实例，这些实例有些是从实验中来的，方便读者理解和学习本课程，同时每章都配有思考与练习。本书对工程应用也具有一定的参考作用。

作者根据十几年的教学经验，并查阅了大量的相关资料完成此书的编写，但由于水平有限，书中难免有错误或不妥之处，敬请同行和读者批评指正。

作 者

2014 年 10 月于福建师范大学福清分校

目 录

第 1 章　微型计算机系统

1.1　微型计算机的组成及工作过程

1.1.1　微型计算机的组成

微型计算机在基本结构和基本功能上与计算机大致相同，但由于采用了具有特定功能的大规模和超大规模集成电路组件，微型计算机在系统结构上有着简单、规范和易于扩展的特点。微型计算机结构框图如图 1-1 所示。

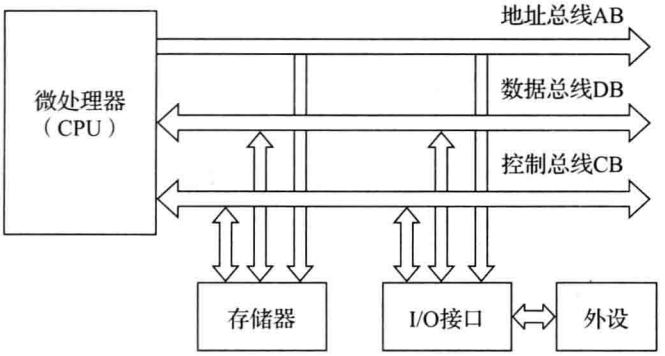

图 1-1　微型计算机结构框图

微型计算机由微处理器、存储器、输入/输出(I/O)接口电路等组成，连接这些功能部件的是三总线，即数据总线、地址总线和控制总线。

1. 微处理器即中央处理器

亦称 CPU(Central Processing Unit)。微处理器是把运算器和控制器这两部分功能部件集成在一个芯片上的超大规模集成电路。微处理器是微型计算机的核心部件，它的基本功能是按指令的要求进行算术和逻辑运算，暂存数据以及控制和指挥其他部件协调工作。

2. 存储器

微型计算机的内存储器采用集成度高、容量大、体积小、功耗低的半导体存储器。根据能否写入信息，内存储器分为随机存取存储器(RAM)和只读存储器(ROM)两类。随机存取存储器又称读写存储器，存储器中的信息按需要可以读出和写入，断电后，其中储存的信息自动消失。它用于存放当前正在使用的程序和数据。只读存储器的信息在一般情况下只能读出，不能写入和修改，断电后原信息不会丢失，是非易失性存储器，主要用来存放固定的程序和数据。

3. 输入/输出接口电路

介于计算机和外部设备之间的电路称为输入/输出接口电路，它具有对数据的缓存作用，使外部设备的速度与计算机速度相适配；具有对信号的变换作用，使各种电气特性不同的外部设备与计算机相连接；还具有连接作用，使外部设备的输入/输出与计算机操作同步。目前微型计算机的接口普遍采用大规模集成电路芯片，大多数接口芯片是可编程的，用命令来灵活地选择接口功能和工作模式。

4. 总线

总线是一组公共的信息输送线，用于连接计算机的各个部件。位于芯片内部的总线称内部总线，与此相对应，连接微处理器与存储器、I/O 接口之间的总线称为系统总线。微型计算机的系统总线分为数据总线、地址总线和控制总线三组。数据总线用于传送数据信息，它是双向总线，数据信息可朝两个方向传送，数据总线用于实现微处理器、存储器和 I/O 接口之间的数据交换。地址总线用于传送内存地址和 I/O 接口地址。控制总线则传送各种控制信号和状态信号，使微型计算机各部件协调工作。

1.1.2 微型计算机的工作过程

微型计算机的工作过程是执行程序的过程，而程序又是由指令序列组成的，因此执行程序的过程就是逐条地执行指令。计算机每执行一条指令，都包含着两个基本的步骤，即取指令和执行指令。

程序在运行前要先由输入设备及操作系统调入到内存储器中，当机器进入运行状态后，首先将第一条指令在内存中的地址赋给程序计数器 PC(或指令指针 IP)，之后就进入取指令阶段，CPU 按照指定的地址读内存，此时读出的必为指令。此指令经过译码后，由控制器发出相应的控制信息，使运算器按照指令规定的操作去执行。一条指令执行完后，又开始下一条指令，重复上述过程，直至遇到暂停指令或某种使程序执行暂停的意外情况才会结束。

为了进一步说明微型计算机的工作过程，下面具体讨论一个模型机怎样执行一段简单的程序。如计算“2+6=?”。这虽然是一个很简单的加法运算，但是计算机无法理解，必须由人通过编写程序，来指定计算机完成这项工作。

要使计算机完成一项指定的工作，一般需要这样几个步骤：

(1)仔细了解并弄清需要解决的问题，建立模型；

(2)制定方案，确定算法，必要时绘制程序流程图；

(3)编写程序；

(4)程序调试或系统测试。

本例是一个非常简单的运算，可以直接进行程序编写。编写程序首先要确定使用什么样的程序语言。目前有很多功能强大、使用方便的高级程序设计语言可供用户选用，但为了更好地说明程序在计算机中的工作过程，在这里使用能够直接对系统硬件进行操作的汇编语言来说明。

首先查阅所使用的微处理器的指令表(或指令系统)，它是某种微处理器所能执行的全部操作命令汇总。不同系列的微处理器具有不同的指令系统。

假定查到模型机的指令表中有 3 条指令可以用来解决这个问题，见表 1-1。

表 1-1　模型机指令

指令名称	助记符	机器码	指令长度	操作
数据传送	MOV A,n	10110000 n	2	把立即数 n 送累加器
加法	ADD A,n	00000100 n	2	把累加器的内容与一个常数 n 相加,结果送到累加器
停机	HLT	11110100	1	CPU 暂停运行

表 1-1 中第 1 列为指令的名称。编写程序时,写指令的全名是不方便的,因此,人们给每条指令规定了一个缩写词,或称作助记符,如上表中的第 2 列。第 3 列为机器码,机器码用二进制或十六进制形式表示,这是计算机真正能够识别的指令形式。第 4 列为指令长度,说明本指令有几个字节。最后一列说明执行一条指令时所完成的具体操作。

现在来编写“2+6=?”的程序。根据指令表提供的指令,用助记符和十进制数表示的加法运算的程序可表达为:

```
MOV   A,2          ;第一个操作数 2 送到累加器
ADD   A,6          ;把累加器的内容 2 与第 2 个数 6 相加,结果送累加器
HLT                ;停机
```

但是,此程序还有问题。因为模型机并不认识助记符和十进制数,而只认识二进制数表示的操作码和操作数。因此,必须把以上程序翻译成二进制数的形式,即用对应的机器码代替每个助记符,用相应的二进制数代替每个十进制数。

```
MOV   A,2          ;对应成二进制码:10110000(指令码)00000010(操作数)
ADD   A,6          ;对应成二进制码:00000100(指令码)00000110(操作数)
HLT                ;对应成二进制码:11110100(指令码)
```

整个程序共有 3 条指令,占用 5 个字节。由于微处理器和存储器均以字节为单位来处理与存放信息,因此,当把这段程序存入存储器时,需要占用 5 个存储单元。假设把它存放在存储器的最前面 5 个单元里,则该程序将占用从 00H 至 04H 这 5 个单元的空间。如图 1-2 所示。

需要强调指出的是,每个内存单元都具有两个和它有关的二进制数。

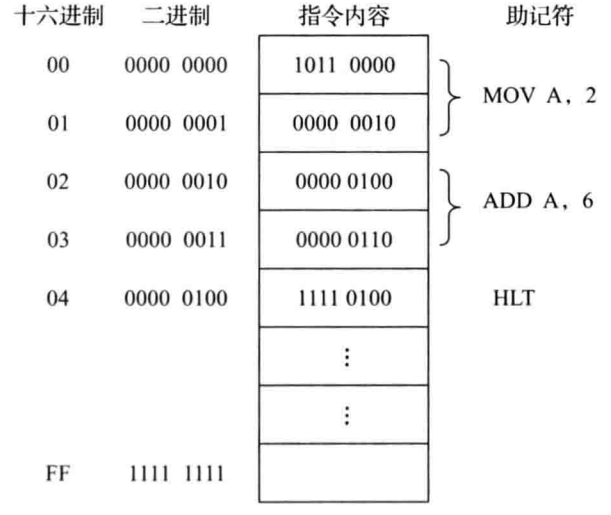

图 1-2　机器码在内存中的存放

第一个是它的地址，另一个是它的内容。切不可混淆两种数据的含义。内存单元的地址是固定的，而内存单元的内容则可以随时由于存入新的内容而改变。

当程序存入存储器并开始执行时，先将第一条指令的首地址 00H 赋给程序计数器 PC，使 PC 指向程序的第一条指令。然后就进入第一条指令的取指阶段，如图 1-3 所示。图中各编号的含义为：

①把程序计数器 PC 的内容 00H 送到地址寄存器 AR。

②一旦 PC 的内容可靠地送入 AR 后，PC 自动加 1，即由 00H 变为 01H(注意，此时，AR 的内容并没有变化)。

③把地址寄存器 AR 的内容 00H 放在地址总线上，并送至存储器，经地址译码器译码，选中相应的 00H 单元。

④CPU 的控制器发出读命令。

⑤在读命令控制下，把所选中的 00H 单元中的内容即第一条指令的操作码 B0H 读到数据总线 DB。

⑥把读出的内容 B0H 经数据总线送数据寄存器 DR；

⑦取指阶段的最后一步是指令译码。因为取出的是指令的操作码，故数据寄存器 DR 把它送到指令寄存器 IR，然后再送到指令译码器 ID。

这就完成了第一条指令的取指阶段。

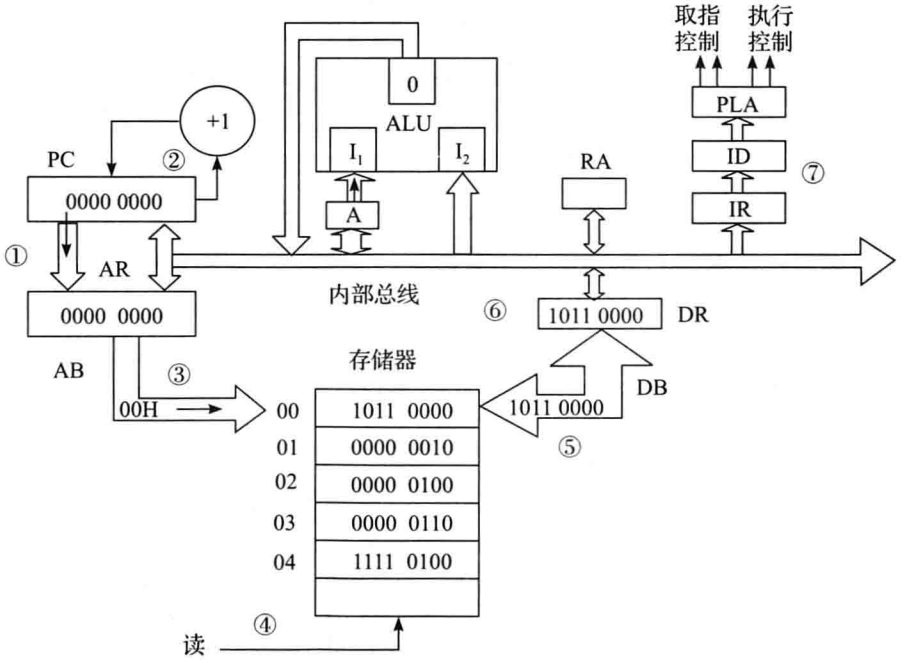

图 1-3　取第一条指令操作码的执行过程

然后转入了第一条指令的执行阶段。经过对操作码 B0H 译码后，CPU“识别”出这个操作码就是“MOV A，n”指令，即把下一个内存单元中的操作数送入累加器 A 的双字节指令，所以，执行第一条指令就必须把第二字节中的操作数取出来。取指令第二字节的过程如图 1-4 所示，图中各编号含义为：

①把 PC 的内容 01H 送到地址寄存器 AR。

②当 PC 的内容可靠地送到 AR 后，PC 自动加 1，变为 02H。但这时，AR 中的内容为 01H，并未变化。

③地址寄存器通过地址总线把地址 01H 送到存储器的地址译码器，经过译码，选中相应的 01H 单元。

④CPU 的控制器发出读命令。

⑤在读命令控制下，将选中的 01H 单元的内容 02H 读到数据总线 DB 上。

⑥通过 DB 把读出的内容送到数据寄存器 DR。

⑦因 CPU 根据该条指令具有的字节数确定，这时读出的是操作数，且指令要求把它送到累加器 A，故由数据寄存器 DR 取出的内容就通过内部总线送到累加器 A，于是第一条指令执行阶段完毕，数 02H 被取入累加器 A 中，并进行第二条指令的取指阶段。

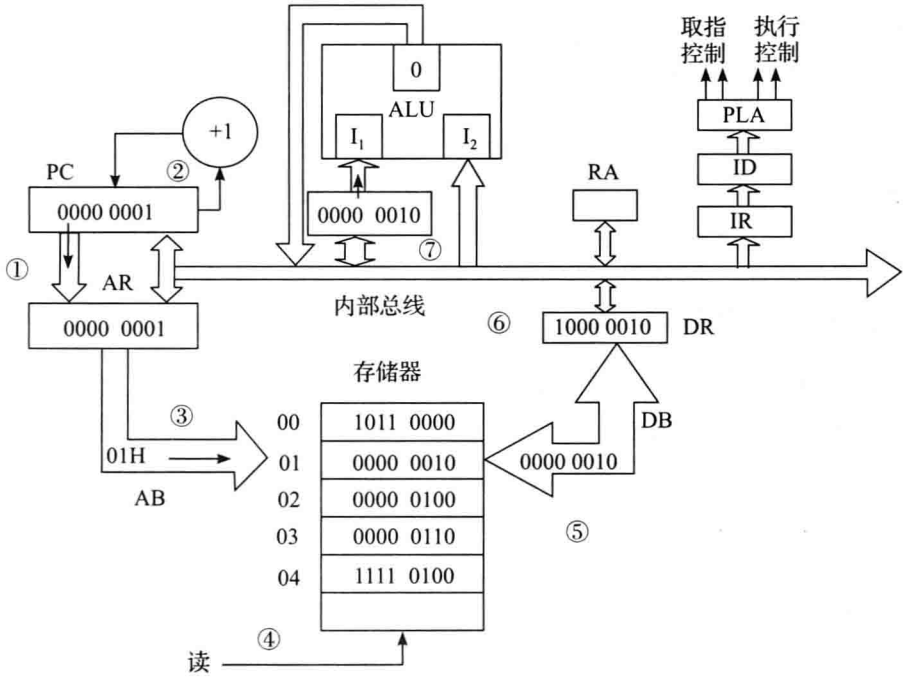

图 1-4　取第一条指令操作数示意图

取第二条指令的过程与取第一条指令的过程相同，只是在取指阶段最后一步读出的指令操作码 04H 由内部数据寄存器 DR 把它送到指令寄存器 IR，经过译码发出相应的控制信息。当指令译码器 ID 对指令译码后，CPU 就“知道”操作码 04H 表示一条加法指令。加法指令把累加器 A 中的内容作为一个操作数，另一个操作数在指令的第二字节中。所以，执行第二条指令过程中，必须取出指令的第二字节。

取第二字节操作数及执行指令的过程如下：

①把程序计数器 PC 的内容 03H 送到地址寄存器 AR；

②当把 PC 的内容可靠地送到 AR 后，PC 自动加 1；

③AR 通过地址总线把地址号 03H 送到地址译码器，经过译码，选中相应的 03H 单元；

④CPU 的控制器发出读命令；

⑤在读命令控制下，把选中的 03H 单元中的内容即数 06H 读至数据总线上；

⑥数据被数据总线传送到数据寄存器 DR；

⑦因通过对第二条指令译码时，CPU 已经确定读出的数据 06H 为操作数，且要将它与已暂存于累加器 A 中的内容 02H 相加，故数据由 DR 通过内部总线送至 ALU 的 I_2 输入端；

⑧A 中的内容送 ALU 的 I_1 输入端，然后执行加法操作；

⑨把相加的结果 08H 由 ALU 的输出端又送到累加器 A 中。

至此，第二条指令的执行阶段结束。A 中存入和数 08H，而将原有内容 02H 冲掉。接着转入第三条指令的取指阶段。

程序中的最后一条指令是 HLT。可用类似上面的取指过程把它取出。当把 HLT 指令的操作码 F4H 取到数据寄存器 DR 后，因是取指阶段，故 CPU 将操作码 F4H 送指令寄存器 IR，再送指令译码器 ID。经译码，CPU“知道”是暂停指令，于是控制器停止产生各种控制命令，使计算机停止全部操作。这时，程序已完成“2＋6＝8”的运算，并且和数 08H 已送到累加器 A 中。

从上述模型机的工作过程可以看到，微处理器 CPU 是核心部件，所以，下面从微处理器入手，介绍微型计算机系统。

1.2 8086/8088 微处理器

1.2.1 微处理器发展概述

微处理器是微型计算机的运算及控制部件，一般由算术逻辑部件(ALU)、控制部件、寄存器组和片内总线等几部分组成。

第一代微处理器是 1971 年 Intel 公司推出的 4004 和 8008，是 4 位和 8 位微处理器，采用 PMOS 工艺，只能进行串行的十进制运算，集成度达到 2000 个晶体管/片，用在各种类型的计算器中已经完全能满足需求。

第二代微处理器是 1974 年推出的 8080、M6800、Z-80 等，是 8 位微处理器，采用 NMOS 工艺，集成度达到 9000 个晶体管/片，在许多要求不高的工业生产和科研开发中已可运用。

第三代微处理器是 20 世纪 70 年代后期 Intel 公司推出 8086/8088、Motorola 公司 M68000、Zilog 公司的 Z8000 等，是 16 位微处理器，采用 HMOS 工艺，集成度达到 29000 个晶体管/片。20 世纪 80 年代之后，Intel 公司又推出 80186 及 80286，与 8086/8088 兼容。80286 是为满足多用户和多任务系统的微处理器，速度比 8086 快 5～6 倍。

第四代微处理器是 1985 年推向市场的 80386 及 M68020，是 32 位微处理器，集成度达 45 万个晶体管/片。它们时钟频率达 40 MHz，速度之快、性能之高，足以同高档小型机相匹敌。

之后，得益于大规模及超大规模集成电路技术的发展，各种高性能的微处理器新产品不断推出。如 1989 年推出 80486，1993 年推出 Pentium 及 80586，1995 年推出 Pentium Pro，1998 年推出 PentiumⅡ，1999 年推出 PentiumⅢ，2000 年推出 PentiumⅣ等更高性能的 32 位和 64 位微处理器，同时也促进了其他技术的进步。

1.2.2 8086/8088 CPU 结构

8086/8088 是 16 位微处理器，这两种微处理器的内部基本相同，但它们的外部数据总

线有所区别，8086 是 16 位数据总线，而 8088 是 8 位数据总线。在读/写一个 16 位数据字时，8088 需要两步操作，而 8086 只需要一步就能完成。

8086/8088 CPU 的内部都是 16 位总线，都采用 16 位字进行操作与存储器寻址，两者的软件完全兼容，程序的执行也完全相同。

1. 8086/8088 CPU 的功能结构

8086/8088 CPU 的内部结构如图 1-5 所示。

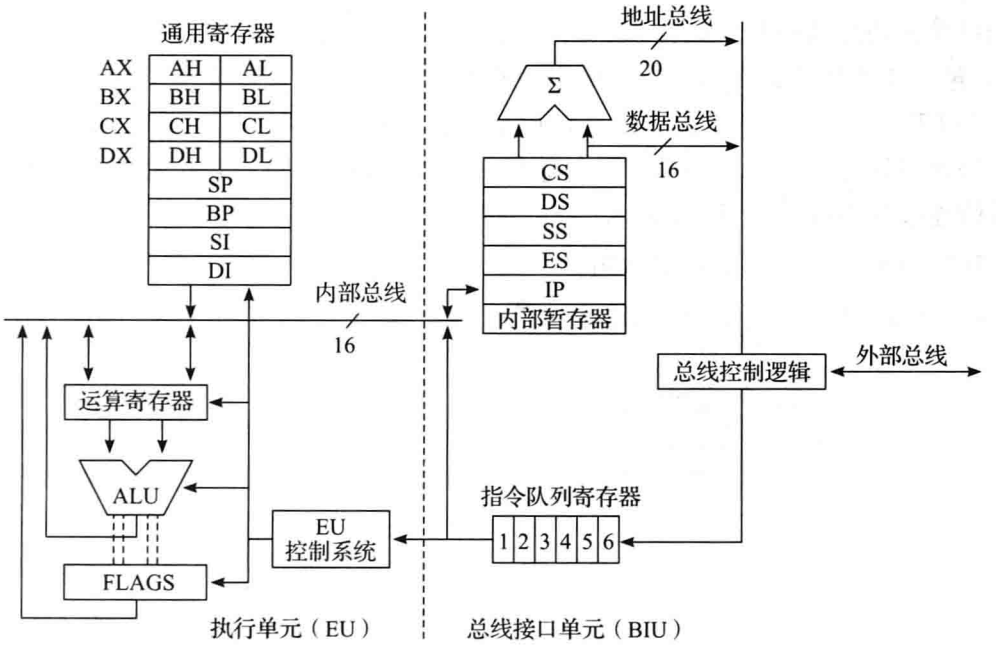

图 1-5　CPU 功能结构示意图

8086/8088 CPU 内部由两部分组成，分别为指令执行单元 EU 和总线接口单元 BIU。它们相互独立工作，大部分取指操作和执行指令的操作重叠进行，即取指令和执行指令同时进行，CPU 的工作是并行的。

(1)指令执行单元(EU)

指令执行单元(EU)的功能是负责指令的执行。EU 中的算术逻辑运算单元 ALU 可完成 16 位或 8 位的二制数运算，运算结果可通过内部总线送到通用寄存器或送往 BIU 单元的内部暂存器中，等待写入存储器。EU 内部的 16 位暂存器用来暂存参加运算的操作数，而经 ALU 运算后的结果特征置入标志寄存器 FLAGS 中保存。

EU 控制系统负责从 BIU 的指令队列中取指令，并对指令译码，根据指令要求向 EU 内部各部件发出控制命令以实现各条指令的功能。指令执行过程中如果需要访问存储器取操作数，EU 将访问地址送给 BIU，等待操作数到达，然后继续操作。在指令执行中如遇到转移指令，BIU 会将指令队列中的后继指令作废，从新的地址重新取指令。这种情况下，EU 要等待 BIU 将取到的指令装入指令队列后，才能继续执行。

(2)总线接口单元(BIU)

它负责与存储器、输入/输出(I/O)端口传送数据。BIU 通过地址加法器形成某条指令在存储器中的物理地址后，从存储器中取出该条指令的代码送入指令队列。当 8086 CPU

指令队列中空出 2 个字节(8088 CPU 指令队列空 1 个字节)时,BIU 将自动进行读指令的操作以填满指令队列。只要收到 EU 送来的操作数地址,BIU 将立即形成这个操作数的物理地址,完成读写操作。8088 CPU 的指令队列有 4 个字节,而 8086 CPU 指令队列可存放 6 个字节的指令代码。

在 BIU 单元中,有一个地址加法器,把 EU 送来的存储器的逻辑地址,即 16 位的段基址左移 4 位后加上 16 位的段内偏移地址,变换成 20 位的物理地址,进而可以寻址 1 MB 的存储空间。

BIU 单元的总线控制电路将 8086/8088 CPU 的内部总线与 CPU 引脚所连接的外部总线相连,包括 8 条数据总线(8086 为 16 条数据总线)、20 条地址总线和若干条的控制总线。

8086 CPU 与 8088 CPU 的区别:(1)指令队列长度不同,8088 只有 4 字节,8086 有 6 字节;(2)外部数据总线不同,8088 CPU 与外部交换数据的总线位数是 8 位,而 8086 与外部数据总线的连接为 16 位,所以,8088 为准 16 位微处理器。

2. 8086/8088 CPU 的寄存器结构

8086/8088 CPU 的内部寄存器结构如图 1-6 所示。

图 1-6　8086/8088 的内部寄存器结构

(1)通用寄存器

8086/8088 CPU 指令执行单元中有 8 个 16 位通用寄存器,分为两组,它们分别为数据寄存器、指示和变址寄存器。

数据寄存器有 AX、BX、CX、DX,可用来存放 16 位数据或地址,也可以把它们分成高 8 位 AH、BH、CH、DH 和低 8 位 AL、BL、CL、DL 两部分,分别作为独立的 8 位寄存器使用。它们的名称和用途如下:

AX(AL)称为累加器,在算术运算指令中用作累加器,在输入、输出指令中作为数据寄存器。

BX 称为基址寄存器。它可以用作数据寄存器,在计算存储器地址时,又可作为间址和基址寄存器。

CX 称为计数寄存器。在字符串操作、循环操作和移位操作时作为计数器。