高等专科学校教材

中国计算机学会大专教育学会推荐出版

80486(80X86)汇编语言程序设计

周学毛 李明钧 赵欢 田华荣 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http: // www. phei. co. cn

内 容 简 介

本书以80486(80X86)微机为背景，较全面地讲述汇编语言程序设计的基本概念、基本原理、基本方法和常用技术，强调以汇编语言程序设计为中心。

本书内容通俗易懂，简明扼要；例题实用，难点分散；程序设计技术先进，容易掌握。适于作大专计算机及相关专业教材，亦适合作非计算机专业本科教材及自学计算机的读者使用。

丛 书 名：高等专科学校教材
书　　名：80486(80X86)汇编语言程序设计
编　　著：周学毛　李明钧　赵欢　田华荣
责任编辑：张凤鹏
特约编辑：袁　英
排版制作：电子工业出版社计算机排版室
印 刷 者：北京大中印刷厂
出版发行：电子工业出版社出版、发行　URL：http://www.phei.co.cn
　　北京市海淀区万寿路173信箱　邮编100036　发行部电话：68214070
经　　销：各地新华书店经销
开　　本：787×1092 1/16　印张：14.25　字数：364.8千字
版　　次：1997年7月第1版　1997年7月第1次印刷
书　　号：ISBN 7-5053-4059-X / G·334
定　　价：16.50元

目　录

第一章　程序设计基础

本章作为开篇，主要介绍学习汇编语言程序设计所必需的计算机软硬基础知识、计算机的系统结构、指令系统。

第一节　汇编语言程序设计

一、程序设计语言与程序设计

(一)指令与程序

指令是用于指出计算机要进行的操作和操作对象的一组代码，实际上是计算机能识别的一组二进制代码。

一台计算机全部指令的集合，构成该计算机的指令系统。指令系统是计算机基本功能的体现，不同的计算机指令系统不同。

程序是为使计算机完成工作、实现预期目的而用程序设计语言设计的一系列操作，其最终形式是指令序列。

(二)程序设计语言

程序设计语言是程序设计人员和计算机进行会话的语言，它遵循一定的规则和形式，构成程序的实现工具。

设计程序设计语言时有一基本前提：既让程序设计人员感到方便，又让计算机最终能识别。二者之间相去甚远，至今尚未得到理想的解决，兼顾二者设计了多种程序设计语言，有的语言接近机器，有的语言接近人。

1. 面向机器的机器语言和汇编语言

机器语言是以计算机硬件能直接执行和理解的指令系统为基础而形成的语言，它为特定的计算机而设计。相应机器语言编写的程序称为机器语言程序，机器语言编写的程序质量高、运行速度快、占用资源少，但不易阅读理解、编写难、调试难、通用性差。除用于编写机器的核心系统程序外，在实际中很少直接使用。

汇编语言是一种符号化了的机器语言，即用助记符号代替机器语言的二进制代码。汇编语言的设计在一定程度上克服了机器语言的不足，同时保留了机器语言的长处。

2. 面向过程的高级语言

为了更加易学易用，提高程序的通用性，设计了大量接近自然语言的程序设计语言。这些语言面向用计算机求解问题的过程，不依赖具体机器，与特定机器相分离，采用接近自然语言的词汇。典型的高级语言有 BASIC、PASCAL、C、FORTRAN、COBOL 语言等。

3. 面向对象的高级语言

程序设计语言的更高形式是面向求解问题本身的高级语言。典型的面向对象的语言有 C++、Smalltalk、面向对象的 PASCAL 语言等。

(三)算法与程序设计

设计程序设计语言的目的是用来编制程序，通过计算机对程序的执行来求解问题。

程序设计的关键是设计算法。所谓算法就是求解问题的有限步骤，具有有穷性、确定性、能行性、输入、输出五大特征。

二、汇编语言与汇编语言程序设计

(一)汇编语言与宏汇编语言

汇编语言是引进了助记符的机器语言，其语句基本上对应机器语言的指令，通常专属于某种机型或某一系列机型，能将计算机的功能全部提供给程序设计者。但也要求程序设计者在计算机内部元件上操作，对机器性能有更深的了解。学汇编语言要了解机器的硬件、指令系统、寻址方式、汇编语法等。

汇编语言一般用在系统软件的开发、扩充或修改系统设备及对速度、空间有特别要求的环境。

实际使用的往往是宏汇编语言，它包含一般汇编语言的功能，采用了高级语言的一些特性，是一种接近高级语言的汇编语言，仍属于汇编语言的范畴。

汇编语言及汇编语言程序设计是计算机专业的一门核心专业基础课，不同的机器配备的汇编语言不同。本教材介绍的是广泛应用的 PC 系列微机的 80486(80X86) 汇编语言与汇编语言程序设计。

(二)汇编程序与汇编

用汇编语言编写的程序称为汇编语言源程序，简称源程序，在 DOS 下的扩展名为 ASM。源程序在计算机中不能直接运行，必须通过一个称为汇编程序的机器语言程序将源程序翻译成相应的目标程序才能运行。

源程序通过汇编程序翻译成目标程序的过程称为汇编。

汇编形成的目标程序其实也不能直接运行，还必须经过连接形成可执行的代码程序才能运行。

(三)汇编语言程序设计

汇编语言程序设计是用汇编语言设计汇编语言程序的过程，实际上是用汇编语言去表现和实现计算机完成的工作。包括对操作进行描述与对数据进行描述两个方面，一般采用结构化程序设计方法。

学程序设计语言不是目的，只是手段，学程序设计才是目的。

第二节　数据表示方法

一、二进制及十六进制数

计算机作为信息处理工具，数据表示问题是计算机工作的出发点，计算机内部采用二进制作为数据表示的基础，其他表示方法由二进制派生出来。

(一)二进制数(Binary)

二进制数由0、1二个数码构成,基数为2,第i位的权为2^i,运算逢二进一。

书写时在尾部加注字母B或下标2描述。

二进制数$a_na_{n-1}...a_0.b_{-1}b_{-2}...b_{-m}$的值是:

$a_n\times2^n+a_{n-1}\times2^{n-1}...+a_0+b_{-1}\times2^{-1}+a_{-2}\times2^{-2}+...+b_{-m}\times2^{-m}$,其中$a_i$和$b_i$为0,1两个数码中一个。

例如:$101011B=1\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0=43_{10}$

n位二进制数可以表示2^n种组合。4位能表示16种组合、0至15的整数,8位能表示256种组合、0至255的整数,16位能表示64×1024(64K)种组合。

(二)十六进制数(Hexadecimal)

使用二进制数在计算机中容易实现,便于存储,抗干扰性强,但对人的阅读、书写、记忆、输入等不方便,尤其位数较多时。

若干年前,程序员发现操作一般是对位的"组",而不是个别的一些位操作。最早的微处理器4位,自然想到用4位一组缩写二进制数,即十六进制数。

十六进制的数由0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F十六个数码构成,其中数码A、B、C、D、E、F对应于十进制数10、11、12、13、14、15,基数为16,第i位的权为16^i,运算逢十六进一。

书写时在尾部加注字母H或下标16。描述以字母开始的十六进制数的前面应加一数码0。其他进制数的书写加注相应的字母或下标,缺省一般指十进制数。

十六进制数$a_na_{n-1}...a_0.b_{-1}b_{-2}...b_{-m}$的值是:

$a_n\times16^n+a_{n-1}\times16^{n-1}...+a_0+b_{-1}\times16^{-1}+b_{-2}\times16^{-2}+...+b_{-m}\times16^{-m}$,其中$a_i$和$b_i$为0,1...、F十六个数码中一个。

例如:$14AFH=1\times16^3+4\times16^2+10\times16+15=5295_{10}$

汇编语言调试系统的数全部用十六进制数,汇编列表文件中代码指令及内存地址也用十六进制数表示。十六进制数是汇编语言的书写工具,应熟悉它。

(三)八进制数(Octal)

计算机中一个字节是8位,也常用八进制数来缩写二进制数。

八进制数由0、1、2、3、4、5、6、7八个数码构成,基数为8,第i位的权为8^i,运算逢八进一。

八进制数$a_na_{n-1}...a_0.b_{-1}b_{-2}...b_{-m}$的值是:

$a_n\times8^n+a_{n-1}\times8^{n-1}...+a_0+b_{-1}\times8^{-1}+b_{-2}\times8^{-2}+...+b_{-m}\times8^{-m}$,其中$a_i$和$b_i$为0、1...、7八个数码中一个。

由于字母O与数字0容易混淆,八进制亦常用尾标Q标识。

(四)数制转换

1. 非十进制数转换成十进制数

采用按权相加法,只需将各位数码与其对应权值的乘积相加即可。

例 1-1

$101101.01B=1*2^5+1*2^3+1*2^2+1+1*2^{-2}=45.25_{10}$

$345Q=3*8^2+4*8^1+5*8^0=229_{10}$

$0F2DH=15*16^2+2*16^1+13*16^0=3885_{10}$

2. 十进制数转换成非十进制数

转换时将整数部分与小数部分分别转换。

整数部分采用除基数取余法，直至商为 0。先得到的余数为低位，后得到的余数为高位。

小数部分采用乘基数取整法，直至乘积为整数或达到控制精度。

例 1-2

$57_{10}=111001B=71Q=39H$

```
2 | 57
  |---
2 | 28 ...1
  |---
2 | 14 ...0
  |---
2 | 7  ...0
  |---
2 | 3  ...1
  |---
2 | 1  ...1
  |---
  0  ...1
```

```
8 | 57
  |---
8 | 7  ...1
  |---
  0  ...7
```

```
16 | 57
   |---
16 | 3  ...9
   |---
   0  ...3
```

$0.625_{10}=0.101B=0.5Q=0.AH$

$0.625*2=1.25...1$　　$0.625*8=5...5$　　$0.625*16=10...A$

$0.25*2=0.5.....0$

$0.5*2=1.......1$

3. 二、八、十六进制数之间的转换

将二进制数转换成八进制数可按三位一组进行，转换成十六进制可按四位一组进行，每一组对应八进制或十六进制相应数码。分组时如位不够，整数部分在最左边补 0，小数部分在最右边补 0，参见表 1-1。

将八进制数转换成二进制数，只需将八进制数一位对应换成二进制数三位即可。同样对十六进制，只需将其一位对应换成二进制数四位即可，参见表 1-1。

表 1-1　十、二、八、十六进制对应关系表

十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
二进制	000	001	010	011	100	101	110	111								
八进制	0	1	2	3	4	5	6	7								
二进制	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

例 1-3

$1100100.11010B = \underline{001}\ \underline{100}\ \underline{100}.\ \underline{110}\ \underline{100} = 144.64Q$

$$= \underline{0110}\ \underline{0100}\ .\ \underline{1101}\ \underline{0000} = 64.DH$$

反过来即是八进制数 144.64，十六进制 64.D 转换所得的二进制数。

二、内存中的数据

数的原值称为真值，它用常规的数符正负号及常规的数码表示。

在计算机中，数的符号也用二进制来表示。连数符一起数字化了的数，称为机器数。一般用最高有效位表示数的符号，正数用 0 表示，负数用 1 表示。机器数可以用不同码制表示，常用 的有原码、补码、反码三种表示法，广泛应用的原码和补码两种表示法。

(一)数的原码表示

原码表示是一种比较直观的机器数表示方法，原码与真值的区别仅仅是符号位数字化。

+1011 采用原码表示为 01011，-1011 采用原码表示 11011。

采用原码作乘除运算比较方便，可取绝对值直接运算，而符号单独处理。但对于应用最多的加减运算，原码表示不太方便，像(−2)+3 表面作加法操作，而实际需作(3−2)减法操作。

(二)数的补码表示

补码表示能让符号位一同参与运算，将减法化成加法。

正数的补码就是本身，形式与原码相同。负数的补码符号与原码相同，其余各位取反，然后末位加 1。

+1011 采用补码表示为 01011，−1011 采用补码表示为 10101。

$[x-y]_{补}=[x]_{补}+[-y]_{补}$，符号位同时参加运算。

如何从补码求原码或真值，可按求补码同样的方法进行。

n 位补码表示的数的范围为$[-2^{n-1},2^{n-1}-1]$。

为扩大数的表示范围，可用二个字来表示一个数，这种数称为双字长数或双精度数。

在计算机中，负数一般用补码表示。

(三) 十进制数的二进制码

十进制数的二进制编码称为 BCD 码。它的引入是为解决日常习惯的十进制与机器内的二进制之间的矛盾，方便进行十进制与二进制之间的转换，最常用的是 8421 码。它对每一位十进制数码用 4 位二进制编码表示。

十进制数 1329 用 BCD 码表示为：$\underline{0001}\ \underline{0011}\ \underline{0010}\ \underline{1001}$

BCD 码在 0 至 9 的范围与纯二进制无区别，不同之处仅在于逢十进位，而 4 位纯二进制数逢十六进位，因此对大于等于 10 的数需作加 6 修正。

BCD 码有压缩 BCD 码和非压缩 BCD 码二种形式，压缩 BCD 码的每个十进制数字按四个二进制位一组存放，而非压缩 BCD 码的每个十进制数字占每个字节的低 4 位，高 4 位内容是什么不重要。

(四)字符编码

ASCII 码是美国信息交换标准代码，是最主要的字符编码方式。对字符进行编码是进行非数值处理，实现输入输出的基本手段。

字符包括字母、数字、专用字符及一些非打印字符。一个字符用 8 位来表示，其中低 7 位为字符的 ASCII 码值，最高位是校验位，7 位编码的 ASCII 码能表示 128 种字符。参见 ASCII 码表。

数字的 ASCII 码用数字加十六进制数 30 可得到。当按某个键，就产生相应的 ASCII 码并送至计算机。

（五）几点说明

在某些情况下，要处理的数据全是正数，此时保留符号位就无意义，可将最高有效位也作为数值位处理，形成无符号整数。8 位无符号整数范围是 0 至 255，16 位无符号整数范围是 0 至 65535。计算机中常用无符号整数表示地址。

数据在计算机内部采用何种方式，依赖于程序的执行情况，可用二进制、二—十进制、字符方式。无符号整数 1996 以压缩 BCD 码、未压缩 BCD 码、二进制方式表示，分别需 16、32、11 位。

数据的存储是以数据长度为单位，按字节顺序存放，用低端低地址方式来存储大于一个字节的数据，低位字节放低地址，高位字节放高地址.

数据要扩展位数，一般是对当前最高位（符号位）进行扩展。当前最高位为 0，扩展位补 0，当前最高位为 1，扩展位补 1。

第三节　计算机系统结构

计算机系统结构是指计算机主要部件的特性布局及连接，是程序设计者必须了解的计算机资源及其资源部件，如地址空间、寄存器组、寻址方式、指令系统等。典型的计算机结构包括 CPU、存储器、输入/输出系统三个主要部分，通过系统总线连接。CPU 是整个系统的核心，指令的执行、机器的工作都由其控制完成。各部分可动态构成各种规模系统。

现在广泛装备的是 PC 系列微机，其 CPU 主要为 Intel 公司所提供的 80486（80X86）系列微处理器，本教材介绍的汇编语言程序设计即建立在其基础上。

一、80X86 微处理器

80X86 是以 Intel 公司 1978 年推出的 8086 微处理器为基础，作纵横两个方向扩展发展起来的微处理器系列，其 CPU 为 8086/8088、80286、80386、80486，至目前最新的奔腾与高能奔腾中的一种，配以适当数量的支持芯片，构成功能强大的微处理器系统。

（一）8086/8088

8086CPU 是 16 位微处理器，采用 16 位数据总线。8088CPU 是准 16 位的微处理器，采用 8 位数据总线，而使用 16 位内部结构。8086 具有 6 字节指令流队列，8088 则是 4 个字节。两者指令系统相同，汇编语言一致，除了运行速度，其他方面无区别。用以装备早期及低档微机。

1. 8086CPU 结构

8086CPU 由指令执行部件 EU 与总线接口部件 BIU 两部分组成，图 1-1 为其结构示意图。

EU 部件（Execution Unit）控制和执行指令，主要由算术逻辑运算部件 ALU、EU 控制部件、8 个 16 位寄存器加标志寄存器组成。

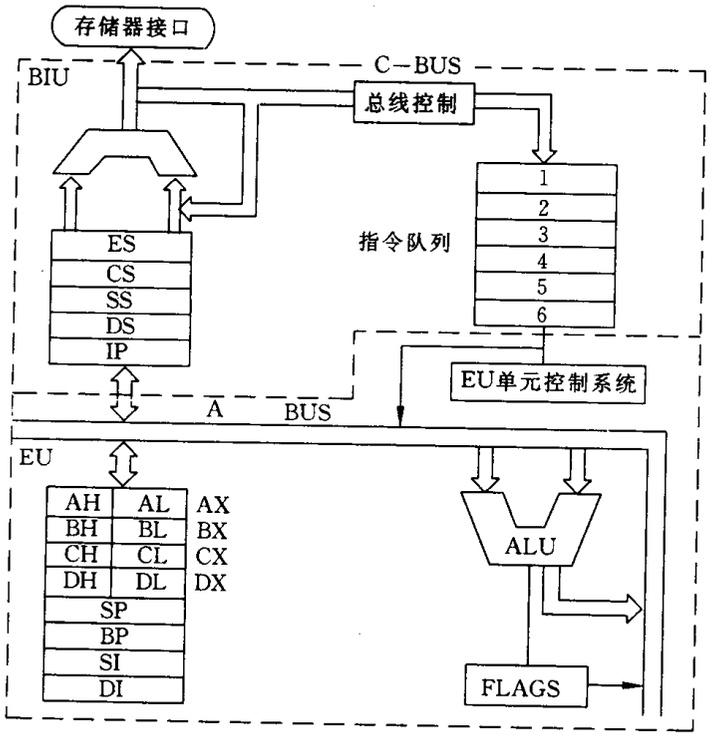

图 1-1　8086 CPU 结构示意图

BIU 部件(Bus Interface Unit)负责从存储器预取指令和数据,以及所有 EU 需要的总线操作,实现 CPU 与存储器和外设之间信息传递。主要由指令队列、指令指针寄存器、寄存器、地址加法器组成。

EU 和 BIU 能独立运行,在一条指令的执行过程中,就可取下一条指令送指令队列,实现并行操作。

2. 内部寄存器

不包括数据暂存器,内部寄存器可分为四组。

(1)数据寄存器

数据寄存器包括 AX、BX、CX、DX 四个 16 位通用寄存器。主要用于存放操作数和运算结果。其高 8 位与底 8 位也可以分开作为 8 位的寄存器使用,高 8 位用后缀 H 表示,低 8 位用后缀 L 表示。8 位寄存器有 AH、AL、BH、BL、CH、CL、DH、DL 八个。

AX、BX、CX、DX 还有各自的专用用途。

AX(Accumulator)是算术运算中主要寄存器,在乘除运算中作累加器用,指定存放被乘数与被除数。另外所有 I/O 指令使用这一寄存器与外部设备传送信息。

BX(Base)在计算存储器地址时,经常用作基址寄存器。

CX(Count)在循环和串处理指令中用作隐含的计数器,CL 还可作为移位计数器。

DX(Data)在双字长运算时与 AX 组合,存放高位字。也常用来存放 I/O 端口地址。

(2)指针及变址寄存器

指针及变址寄存器包括 SP、BP、SI、DI 四个 16 位寄存器,可像数据寄存器一样,作为存放操作数的通用寄存器,更经常的用途是存放地址,提供寻址时的偏移地址,用于形成指令要访问的存储器地址。

SP(Stack Pointer)是堆栈指针寄存器,用来指示栈顶的偏移地址。BP(Base Pointer)是基址指针寄存器,可作为堆栈的基地址。SP 或 BP 与段寄存器 SS 结合,可确定堆栈中某一存储

单元地址。

SI(Source Index)和 DI(Destination Index)是源变址寄存器和目的变址寄存器,一般与段寄存器 DS 联用,用来确定数据段中某一存储单元地址。

(3)段寄存器

段寄存器包括 CS、DS、SS、ES 四个 16 位寄存器,分别用来标识代码段、数据段、堆栈段和附加段的段地址,用以扩大可访问的存储空间,实现 CPU 对内存不同段的读写。参见稍后的存储分配方式。

(4)控制寄存器

控制寄存器包括指令指针寄存器 IP 和标志寄存器 FLAGS 两个 16 位的寄存器。

IP 又称程序计数器,它存放当前要执行指令的地址,由于 8086 采用分段存储程序和数据,IP 实际上存放的是当前代码段的偏移量,它随程序的执行而变化,硬件上保证总是自动加 1,计算机用 IP 寄存器来控制指令序列的执行流程。

FLAGS 又称程序状态字寄存器,它的内容标志着程序执行时 CPU 的状态,8086 只使用 16 位中的 9 位,分布如下:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

9 个标志位分成状态标志和控制标志两组。状态标志位寄存着指令执行结果的状态信息,往往作为后续条件转移指令的控制条件,亦称为条件码标志位,包括以下 6 位:

OF,溢出标志位。运算结果超出系统所能表示的数的范围称为溢出,此时 OF 位置 1,否则置 0。对于无符号数的运算必须根据进位标志位 CF 决定是否溢出。

SF,符号标志位。记录运算结果的符号,结果为负时置 1,否则置 0。与运算结果最高位始终保持一致。

ZF,零标志位。运算结果为 0 时,置 1,否则置 0。

AF,辅助进位标志。运算第 3 位(半字节)产生的进位或借位时置 1,否则置 0。该标志仅用于控制十进制调整指令。

PF,奇偶标志位。运算结果中 1 的个数为偶数置 1,否则置 0。常用来检查数据在传送过程中是否发生错误。

CF,进位标志位。运算时最高位产生进位或借位时置 1,否则置 0。该标志主要用于多字节数的加减运算。

控制标志位对某些 CPU 的操作进行干预,包括以下 3 位:

DF,方向标志位。标志字符串指令处理的方向,DF 为 1 时,串处理从高地址向低地址递减进行,DF 为 0 时,串处理从低地址向高地址递增进行。

IF,中断标志位。IF 为 1 允许中断,IF 为 0 屏蔽中断。

TF,自陷标志位。TF 为 1 进入单步执行方式,以便调试程序,TF 为 0 是连续执行方式。

(二)80286

80286CPU 在 1984 年推出,采用 16 位数据总线,支持实模式与保护模式操作,在保护模式下可访问 16M RAM,用以装备中低档微机。

1. 80286CPU 结构

80286CPU 由指令执行部件 EU、指令译码部件 IU、总线部件 BU、地址部件 AU 四个部分组成。各部分能平行工作，从而减少了彼此等待时间，提高了系统的效率。图 1-2 为其结构示意图。

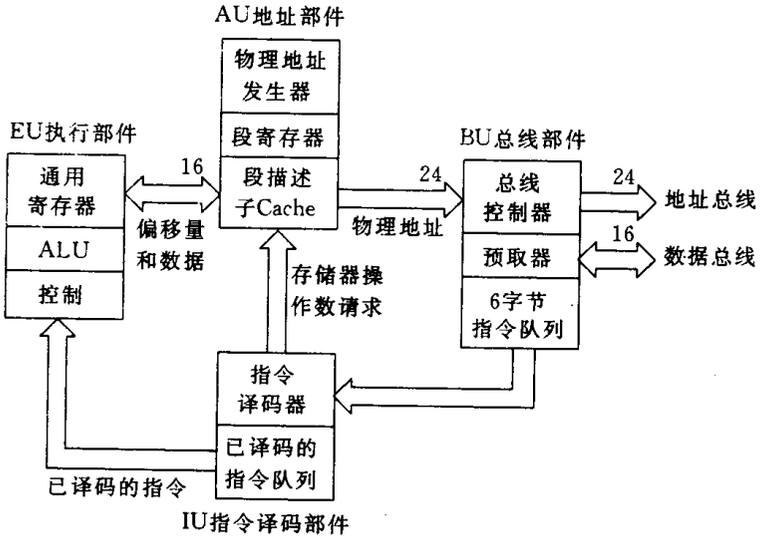

图 1-2　80286CPU 结构示意图

EU 部件与 8086 类似，但它不对指令译码。指令译码由指令译码部件 IU 完成，IU 部件可存放 3 条已译码的指令，并负责所有总线之间的通信和数据传输。AU 部件在实模式下与 8086 相应部分功能一样，在保护模式下，它支持虚拟存储方式及存储保护。

2. 内部寄存器

80286 的寄存器与 8086 的寄存器只有以下几点不同：

(1)标志寄存器增加了两个标志，占用三个标志位，用于保护模式。

IOPL、IO，特权标志，占用第 12 及 13 位，表示该任务使用的 IO 操作所处特权级。特权共 4 级，00 为最高级，11 为最低级。

NT，任务嵌套标志，占用第 14 位。NT 为 1，表示当前执行任务嵌套在另一任务中，实模式下 NT 为 0。

(2)增加了一个 16 位机器状态字寄存器 MSW。

MSW 目前使用低 4 位。其中 PE 是操作模式位，置 1 时为保护模式，否则为实模式。MP 是协处理器标志位，置 1 表示协处理器存在，否则不存在。EM 是仿真协处理器标志位，置 1 要求用软件仿真协处理器，否则不要求。TS 是任务转换标志位。

另外还增加有 4 个寄存器用以支持保护虚存机构。

(三)80386

80386CPU 在 1986 年推出，有 DX 与 SX 两种型号，在内部都采用 32 位总线，但外部 DX 的数据与地址总线都是 32 位，SX 的外部地址总线是 24 位，数据总线 16 位。支持实模式与保护模式操作，在保护方式下可访问 4G RAM。用以装备中档微机。

1. 80386CPU 结构

80386CPU 由指令执行部件、指令预取部件、指令译码部件、总线部件、分段部件、分页部件六个部分组成。各部件间均能并行操作，CPU 同一时间可处理多条指令的不同阶段，各部件

都能和总线部件直接通信,CPU 的能力大为提高,图 1-3 为其结构示意图。

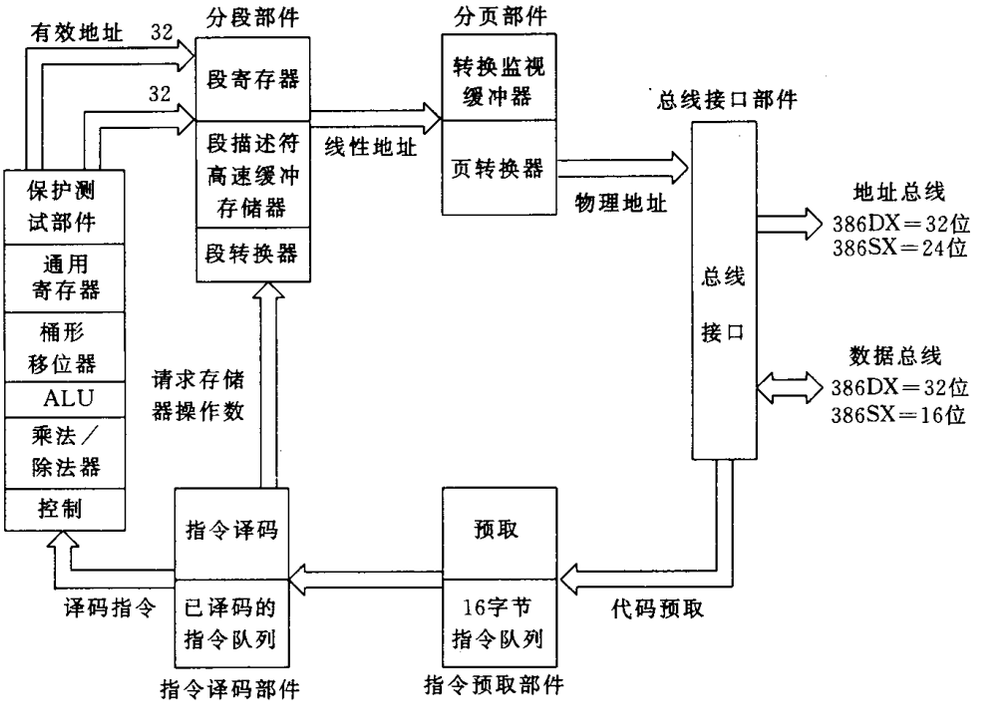

图 1-3　80386CPU 结构示意图

总线部件与 80286 的总线部件功能类似，不同的是预取指令功能从总线部件分离出来成为一个独立的处理部件。指令预取部件在 16 个字节的预取指令队列有 2 个字节空闲 时就向总线请求取指令,但优先级最低。指令译码部件与 80286 功能类似。指令执行部件负责执行指令,包含有 1 个 64 位的桶形移位器,能在 1 个时钟周期实现多位的移位。分段部件与 80286CPU 的地址部件功能类似,但加速了地址的计算速度，并且把线性地址传送给分页部件,而不是送给总线。分页部件的功能是将线性地址转换成物理地址,转换有两种情况,如果无分页特征,则该线性地址就是物理地址，如果有分页特征,则将线性地址按页存储方式转换为物理地址,页长为 4KB,最后将物理地址送总线。

2. 内部寄存器

80386 采用 32 位寄存器,包括 8 个通用寄存器 EAX、EBX、ECX、EDX、ESP、EBP 、ESI、EDI,低 16 位及其低 8 位与高 8 位可同 8086 的寄存器完全相同使用(包括名称)，但高 16 位不能单独使用。32 位寄存器的使用方法也类似,但增强通用性(包括对应 16 位寄存器),除 SP 寄存器外,均可作基址寄存器或变址寄存器使用。

80386 的指令指针寄存器 EIP 可存放 16 位或 32 位地址。标志寄存器 EFLAGS 也是 32 位的寄存器,包含 80286 标志寄存器各位的定义,并增加了 2 个标志位，重新开始标志 RF 占用第 16 位,用于控制调试异常中断。虚拟 86 模式标志占用第 17 位,置 0 为保护模式下运行,置 1 为虚拟 86 模式下操作。

另外增加 FS、GS 两个数据段寄存器,共 6 个 16 位段寄存器,可同时用于 6 个分段。还有机器状态字寄存器、保护模式管理寄存器及调试寄存器等。

二、80486 微处理器功能结构

80486 是 Intel 公司在 1989 年推出的高性能微处理器,采用完整的 32 位体系结构，CPU

芯片中集成了 8K 高速缓存、387 协处理器和存储管理部件，常用指令执行只需一个时钟周期，达 RISC 性能，增强了多处理系统，与以前 86 系列微处理器保持二进制兼容，已有软件可不作任何修改在 486 微处理器上运行。

80486CPU 由总线部件、高速缓存部件、指令预取部件、指令译码部件、控制部件、整数部件、浮点部件、分段部件与分页部件 9 个独立部分组成。图 1-4 为其结构示意图。

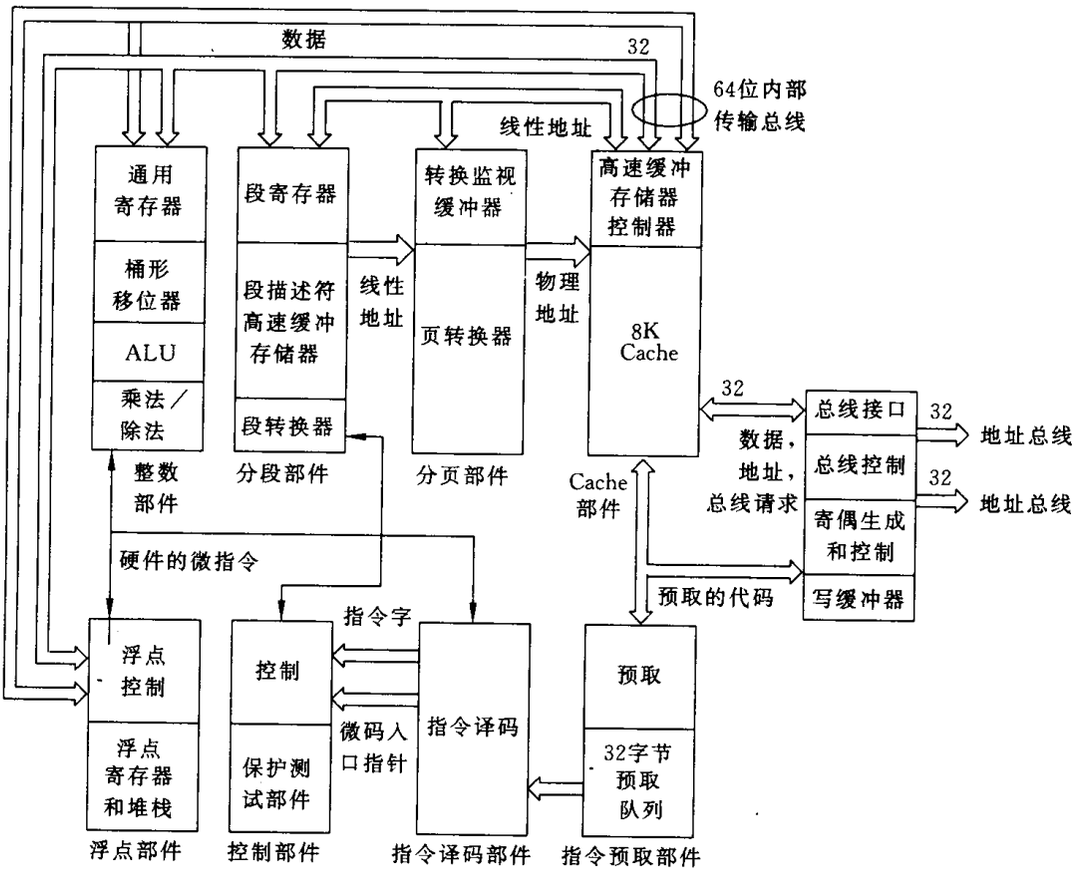

图 1-4　80486CPU 结构示意图

(1) 总线部件

负责总线界面，内部只与高速缓存部件和指令预取部件交换数据。一次从外部总线读取 16 个字节数据传送至高速缓存，预取指令、读存储器和填充高速缓冲存储器均由它控制与进行。

(2) 高速缓存部件(Cache)

高速缓存有 8K 字节，由自身控制器管理，采用通写策略，凡对总线的读写请求，均先送 Cache。读时，如 Cache 中能命中，则立即满足，不产生总线周期。写时，如果 Cache 中有此地址数据，产生 1 个总线写请求，将数据写回内存。预取指令也用 Cache 进行。

(3) 指令预取部件

有 32 字节的预取指令队列，指令平均长度为 3.2 字节，可预取 10 条指令，若 Cache 中能命中，不产生总线周期。

(4) 指令译码部件

负责指令译码。由于一个时钟周期执行一条指令，无需指令队列。

(5) 控制部件

负责执行控制性指令，并对整数部件、浮点部件和分段部件进行控制。

(6) 整数部件

即算术逻辑部件,包括算术逻辑运算器,8 个 32 位通用寄存器和 1 个 64 位桶形移位器。

(7) 浮点部件

功能与 80387 一样,为各种数值数据类型提供运算指令,与整数部件并行操作。

(8) 分段部件与分页部件

同 80386 一样,分段部件与分页部件是存储的管理部件 ,但产生的物理地址通过 Cache 部件传送给总线部件。

分段部件,Cache 部件,运算部件在内部共享两条 32 位总线,当 64 位段描述符从 Cache 传送到分段部件时,每条总线传送一半,从而实现内部 64 位总线传输。

三、80486 寄存器组

80486 寄存器组包括 80386 和 80387 的所有寄存器,可分为基本体系结构寄存器、系统级寄存器、浮点寄存器、调试和测试寄存器,增强的功能由增加相应寄存器中标志实现。其中体系结构寄存器和浮点寄存器是可被应用程序访问的,其他两类只有特权级 0 时才能访问。

(一)基本体系结构寄存器

1. 通用寄存器

如图 1-5 所示,同 80386 一样,包括 EAX、EBX、ECX、EDX、ESI、EDI、EBP 及 ESP 8 个寄存器,用以存放数据或地址,支持数据操作数 1 位、8 位、16 位、32 位以及 1 至 32 位的位字段,地址操作数有 16 位或 32 位。

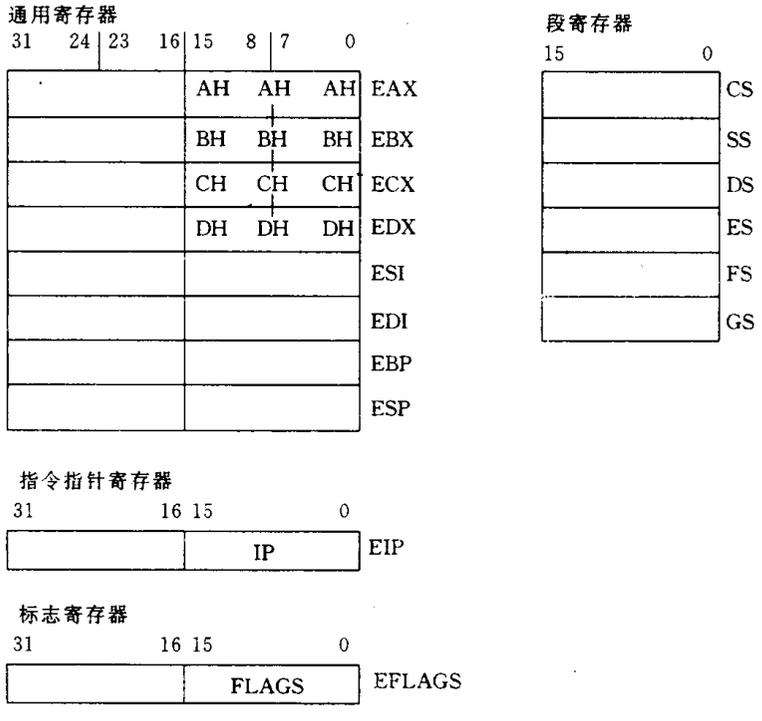

图 1-5 基本体系结构寄存器

16 位操作可用其低 16 位作为 16 位寄存器进行,低 16 位寄存器与 8086 寄存器一样,用 AX、BX、CX、DX、SI、DI、BP、SP 表示。当访问低 16 位时,高 16 位内容不变。8 位操作可以单独

访问 AX、BX、CX、DX 的低 8 位或高 8 位，用 8086 同样的方式表示。

2. 指令指针寄存器 EIP

与 80386 相同，EIP 中存放要执行指令的偏移地址。低 16 位包含有 16 位指令指针寄存器 IP。

3. 标志寄存器 EFLAGS

与 80386 相比，只增加对准检查标志位 AC，占第 18 位，表示操作数地址边界是否对准。低 16 位包含有 16 位标志寄存器 FLAGS。

4. 段寄存器

与 80386 一样，有 6 个 16 位的段寄存器，实模式下存放段地址，保护模式下存放段选择符，由选择符决定段地址，标识当前可编址的存储器段。可同时用于 6 个分段，CS 指出当前代码段，SS 指出当前堆栈段，DS、ES、FS、GS 指出当前数据段。

（二）系统级寄存器

包括三个控制寄存器 CR0、CR2 和 CR3，四个段基值寄存器 GDTR（全局描述符表寄存器）、IDTR（中断描述符寄存器）、LDTR（局部描述符表寄存器）和 TR（任务状态段寄存器），控制着高速缓存、浮点部件以及分段分页机制的操作。

（三）浮点寄存器

包括八个 80 位数据寄存器及标记字、控制、状态、指令指针、数据指针寄存器，用以完成浮点处理。

四、运行模式与存储分配方式

（一）运行模式

80486（80X86）能以实模式与保护模式两种模式运行程序，如同一种型号的两个微处理器。

1. 实模式

实模式全称为实地址存储管理操作模式，实际上是 8086 的程序运行模式，程序与数据运行在实际存储空间，无存储保护，内存 1MB，采用段式存储管理。

在实模式下，80486（80X86）除了多用几种指令，如同高速的 8086。开机时处于此模式，直到程序中明显地将它转换为保护模式。实模式是学习汇编语言的主体模式，本教材以实模式下的汇编语言程序设计为背景。

2. 保护模式

保护模式全称为保护虚地址存储管理操作模式，以实模式提供它所做的一切，程序与数据运行在虚拟存储空间，有存储保护。保护模式实际内存达 4GB，内存管理采用段式及段页式管理。

在保护模式下还有模拟 8086 模式，或称为虚拟 86 模式，保护模式下可有多个虚拟 86 模式。

实模式下运行单任务操作系统，保护模式下运行多任务操作系统。运行模式由机器状态字与标志寄存器中的相关位标识，与存储分配方式关系密切。