高等教育规划教材

微型计算机原理及应用技术

第 3 版

朱金钧　麻新旗　等编著

机 械 工 业 出 版 社

本书以 Intel 86 系列微处理器为背景,从微处理器的角度介绍了Intel 86系列微处理器的结构、工作原理、指令系统及汇编语言、程序设计等内容;从微型机系统组成的角度介绍了存储器结构、中断系统和接口技术;从应用角度介绍了典型的微型机系统及工业 PC 系统,并引入适量的可直接引用的编程实例。书中采用软硬件结合的方法,全面介绍了微型计算机系统的组成原理及应用。

本书内容丰富,注重系统性、先进性和实用性,可作为高等学校计算机及电类有关专业教材或技术人员培训教材,也适合于广大从事微型计算机科研、生产、教学和应用开发的科技人员自学或参考。

本书配有电子教案,需要的教师可登录 www. cmpedu. com 免费注册,审核通过后下载,或联系编辑索取(QQ:2966938356,电话:010 - 88379739)。

图书在版编目(CIP)数据

微型计算机原理及应用技术/朱金钧等编著. —3 版. —北京: 机械工业出版社, 2015. 1
高等教育规划教材
ISBN 978-7-111-48266-6

Ⅰ. ①微… Ⅱ. ①朱… Ⅲ. ①微型计算机 - 高等学校 - 教材 Ⅳ. ①TP36

中国版本图书馆 CIP 数据核字(2014)第 271219 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)
责任编辑: 和庆娣 责任校对: 张艳霞
责任印制:
印刷(装订)

2015 年 1 月第 3 版 · 第 1 次
184mm × 260mm · 21. 5 印张 · 532 千字
0001 - 册
标准书号: ISBN 978-7-111-48266-6
定价: 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

电话服务
服务咨询热线:(010)88379833
读者购书热线:(010)88379649

网络服务
机 工 官 网:www. cmpbook. com
机 工 官 博:weibo. com/cmp1952
教育服务网:www. cmpedu. com
金 书 网:www. golden - book. com

出 版 说 明

当前,我国正处在加快转变经济发展方式、推动产业转型升级的关键时期。为经济转型升级提供高层次人才,是高等院校最重要的历史使命和战略任务之一。高等教育要培养基础性、学术型人才,但更重要的是加大力度培养多规格、多样化的应用型、复合型人才。

为顺应高等教育迅猛发展的趋势,配合高等院校的教学改革,满足高质量高校教材的迫切需求,机械工业出版社邀请了全国多所高等院校的专家、一线教师及教务部门,通过充分的调研和讨论,针对相关课程的特点,总结教学中的实践经验,组织出版了这套“高等教育规划教材”。

本套教材具有以下特点:

1)符合高等院校各专业人才的培养目标及课程体系的设置,注重培养学生的应用能力,加大案例篇幅或实训内容,强调知识、能力与素质的综合训练。

2)针对多数学生的学习特点,采用通俗易懂的方法讲解知识,逻辑性强、层次分明、叙述准确而精炼、图文并茂,使学生可以快速掌握,学以致用。

3)凝结一线骨干教师的课程改革和教学研究成果,融合先进的教学理念,在教学内容和方法上做出创新。

4)为了体现建设“立体化”精品教材的宗旨,本套教材为主干课程配备了电子教案、学习与上机指导、习题解答、源代码或源程序、教学大纲、课程设计和毕业设计指导等资源。

5)注重教材的实用性、通用性,适合各类高等院校、高等职业学校及相关院校的教学,也可作为各类培训班教材和自学用书。

欢迎教育界的专家和老师提出宝贵的意见和建议。衷心感谢广大教育工作者和读者的支持与帮助!

机械工业出版社

前　言

本书是高等院校计算机及相关专业非常重要的专业基础课教材，同时也是计算机科学工作者研发计算机有关项目的得心应手的参考书，书中很多实例来自作者的科研成果，并有大量最新计算机资料供读者查阅。本书从八五级计算机、电子、自动化等专业开始使用，结合二十多年的教学实践，历经三次结构性的改写和多次调整，本书日益成熟。此次改版主要的是把近年来微处理器等新产品、新技术补充进来，使本书成为名副其实的具备最新计算机资料的专业基础教科书。

本书以 8086 系统为基础，系统地讲述了微型计算机的 CPU 结构、指令系统与汇编语言程序设计、存储器、接口设计和总线等一系列技术。书中大部分内容是作者多年来从事教学实践和科学研究的工作总结，同时对大量有关微型计算机技术和应用的文献资料进行了提炼和综合。本书力求做到以下几点。

1）重点突出、层次清楚、由浅入深、循序渐进。例如：在第 1.3 节中补充了计算机组织与结构的核心内容，为后续章节的学习奠定了基础，便于读者顺利地入门并学习微型计算机的工作原理。全书共分为 9 章，每一章均可作为一个相对独立的层次，从计算机基础知识到计算机系统组成，从 CPU 到指令系统、汇编语言设计，从接口设计到计算机应用，引导读者由浅入深、循序渐进地学习。

2）理论与实践相结合。计算机科学本身是一门软硬相结合的科学，需要大量实践才能掌握，故在汇编语言程序设计、接口设计等章节中，本书给出了一些可以直接引用的程序实例和接口设计的软硬件设计实例，便于读者学习和使用。

3）既强调基础，又突出先进性。本书以 16 位的 8086CPU 作为范本，追踪 32 位系列主流高档微型计算机的技术发展，介绍了从 80286 到 Pentium 4 的结构和特点和多核微处理器。

4）在文字叙述上力求语言精炼、通俗易懂，并将书中不能详述的内容收入附录，便于读者查询。

由于学时限制，在教学中建议第 3 章和第 9 章简讲，但不会影响知识的连贯性。全书由朱金钧统稿，第 1～3 章由麻新旗编写，第 4、5 章由张会莉编写，第 6 章和附录由周万珍编写，第 7、9 章由朱金钧、朱薇编写，第 8 章由薛增涛编写。

本书在编写过程中得到多位同志的帮助，在此深表谢意。

由于作者水平有限，书中难免存在疏漏和不妥之处，敬请读者批评指正。

编　者

目　　录

第 1 章　计算机基础知识

电子数字计算机是 20 世纪最重大的科技成就之一。自 1946 年第一台电子计算机问世以来,计算机得到迅速发展,并已广泛应用于工农业生产、科学研究、国防及人们日常工作和生活的各个领域。伴随人类进入 21 世纪,以高科技为支撑的信息化社会已经到来,以“信息”为主导的新兴产业正在全球经济领域掀起一场空前的革命。“知识”是这场革命的直接推动力,而计算机及其应用技术则是知识经济的基础。随着信息化时代的到来,计算机技术的进一步发展和应用必将对社会发展和人类文明产生更大的促进作用,对社会政治、经济、文化和人类生活的各个方面产生巨大而深远的影响。

1.1　计算机发展概述

本节主要讲述计算机的发展历史,分析计算机的主要特点和分类,指出计算机的应用范畴和发展趋势。

1.1.1　计算机发展概况

“计算”是人类生活中的一项重要活动。人类祖先在史前时期就知道用石块和贝壳计数。随着人类文明的发展,人类创造了简单的计算工具,如我国在唐宋时期就开始使用算盘。经过长期艰苦的努力和探索,科学家们发明了机械式计算器、继电器式计算机,在 1946 年终于研制成功第一台电子数字计算机 ENIAC。

在推动计算机发展的诸多因素中,电子元器件的发展是最活跃的因素。因此,人们常常把计算机的发展以所采用的电子元器件为标志进行划分。

第一代(1945 ~ 1958 年):电子管计算机。采用水银延迟线作内存,磁鼓作外存;体积大、耗电多、运算速度慢。最初只能使用二进制表示的机器语言,到 20 世纪 50 年代中期才出现汇编语言。这个时期,计算机主要用于科学计算和军事方面,应用很不普遍。

第二代(1958 ~ 1964 年):晶体管计算机。内存主要采用磁芯,外存大量采用磁盘,输入/输出设备有了较大改进;体积显著减小、可靠性提高、运算速度可达每秒百万次;软件方面出现了高级程序设计语言和编译系统。计算机开始广泛应用于以管理为目的的信息处理。

第三代(1964 ~ 1971 年):固体组件计算机。主要采用中、小规模集成电路;运算速度达每秒千万次,可靠性大大提高,体积进一步缩小,价格大大降低;软件方面进步很大,有了操作系统,开展了计算机语言的标准化工作并提出了结构化程序设计方法,出现了计算机网络。计算机应用开始向社会化发展,其应用领域和普及程度迅速扩大。

第四代(1971 年至今):大规模集成电路计算机。大规模集成电路的出现使计算机发生了巨大的变化,特别是出现了微处理器,从而推出了微型计算机。微型计算机的出现和发展是计算机发展史上的重大事件,使得计算机在存储容量、运算速度、可靠性和性能价格比等方面都比上一代计算机有了较大突破。各种系统软件、应用软件大量推出,功能配置空前完善,充分发挥了计算机的功能,把计算机的发展和应用带入到了一个全新的时代,计算机已经应用到几乎所有的领域,成为人类社会活动中不可缺少的工具。

微型计算机的发展过程,也就是微处理器的发展过程,自 1971 年第一个微处理器出现以来,其发展已经经历了 4 代,目前正处在第 5 代微处理器发展阶段,每一代的性能都提高了近一个数量级。几乎每两年就有一个质的变化,目前仍在向多功能、多媒体方向发展。

1971 ~ 1973 年为第一代微处理器,代表产品为 Intel4004 和 Intel8008。前者为 4 位机,后者为 8 位机。集成度约为 2000 个等效晶体管/片,时钟频率为 1MHz,指令周期为 20μs。

1973 ~ 1975 年为第二代微处理器,代表产品有 Intel8080、M6800,字长为 8 位,集成度为 5000 个晶体管/片,时钟频率为 2MHz,指令周期为 2μs。

1975 ~ 1977 年为第三代微处理器,代表产品有 Intel8085、Z80、M6802 等,字长为 8 位,集成度为 10000 个晶体管/片,时钟频率为 2.5 ~ 5MHz,指令周期为 1μs。

1978 ~ 1980 年为第四代微处理器,代表产品有 Intel8086、M6809、Z8000 等,字长为 16 位,集成度约为 30000 个晶体管/片,时钟频率为 5MHz,指令周期小于 0.5μs。

1980 年之后为第 5 代微处理器,代表产品有 80286、Motorola68010 等,字长为 16 位,集成度达 100000 个晶体管/片,时钟频率为 10MHz,指令周期约为 0.2μs。1983 年之后又出现了 Intel80386、Motorola68020 等微处理器,字长为 32 位,时钟频率为 16MHz 以上,集成度高达 15 ~ 500 000 个晶体管/片,指令周期为 0.1μs。其后又出现了 Intel80486、Pentium 系列,集成度达数百万管/片以上,时钟频率高达上千兆赫兹,指令周期只有几 ~ 几十纳秒。在这些处理器的芯片上已经包含大容量的高速缓冲存储器(CACHE),原来属于大型机的存储管理技术已经移植到芯片上。

1.1.2 计算机的主要特点

电子计算机是一种不需要人的直接干预,能够高速、自动地进行算术和逻辑运算的电子装置。存储程序与程序控制是计算机的重要工作原则,是它能够高速自动运算的基础。

存储程序是把计算过程表示为由许多条命令(指令)组成的命令序列(程序),与数据一起预先存入计算机的存储器内。只要发出运行命令,计算机就会按照规定的顺序一条一条地取出指令和执行指令,自动地完成预定的信息处理任务。由于在程序执行过程中不需要人的干预,因此,计算机能自动高速地执行程序。人们可以把任何信息处理任务分解成一系列基本的算术和逻辑操作,并按照执行的先后顺序把它们组成程序,存入计算机并使之执行,因此计算机可以完成任何信息处理任务,具有很强的通用性。

计算机处理的各种符号,包括数值、文字、符号、图像、声音等,都通过数字化编码技术,用数字量表示,所以计算机运算非常准确。同时,计算机可以对这些数字量进行各种大小关系的比较判断,并根据比较结果决定下一步的处理,这就是“条件转移”概念,即计算机的逻辑性。

综上所述,计算机具有 5 大特点:自动性、高速性、准确性、逻辑性和通用性。

1.1.3 计算机的分类

电子计算机的分类方法多种多样。从原理上,可分为处理连续变化信号的模拟计算机和处理离散信号的数字计算机;从用途上,可分为专用计算机和通用计算机。人们现在常说的“计算机”实际上是指通用电子数字计算机,并可分成巨型机、大型机、小型机和微型机等。微型机按字长又可分为 4 位、8 位、16 位、32 位和 64 位机,按结构可分为位片机、单片机、单板机和微机系统等。

1.1.4 计算机的应用

计算机的应用形式和应用领域千变万化、日新月异,已深入到人类社会生活的各个领域,从

国防技术、航空航天技术、核能技术、管理信息系统、科学研究、工业设计和仿真、生产过程控制、多媒体与信息高速公路技术、文化教育、医疗等,到智能仪表、家用电器等,无一不是计算机信息处理与控制的应用领域。按照计算机应用的性质和形式,可分为数值计算、数据处理(包括办公自动化、数据库应用系统)、生产自动化(包括过程控制、计算机辅助系统——如计算机辅助设计 CAD、辅助制造 CAM、辅助测试 CAT、辅助工程 CAE、计算机集成制造系统 CIMS 等)、计算机模拟、人工智能、计算机网络应用、远程教育等。

1.1.5 计算机技术的发展趋势

当今计算机的发展趋势主要有以下几方面。

1)两极化。即巨型计算机和微型计算机。前者代表着计算机科学技术的发展水平,主要应用于大型领域如国防尖端技术、航空航天技术等。后者则反映了计算机的应用普及程度。多媒体技术是当前微型计算机研究的热点,并行处理则是巨型计算机的核心技术。

2)多媒体技术。多媒体技术是将数值、文字、声音、图形、图像、视频等多种媒体信息进行综合处理的技术,是当前微型计算机研究和开发的热点,其中的关键技术是音频和视频数据的压缩/解压缩、多媒体数据的通信传输,以及各种多媒体信息设备的接口技术。

3)网络化。网络技术是计算机技术与通信技术的结合,是今后计算机应用的主流。Internet 的迅速发展与广泛应用,使人类进入了信息化时代,信息的快速获取和共享已成为影响经济发展与社会进步的重要因素之一。

4)智能化。应用人工智能技术,使计算机模仿人的推理、思维、联想、学习等功能,并具有声音识别、图像识别能力,在某种程度上具备人的智能,这种智能计算机是未来计算机发展的总方向。

5)非冯·诺依曼体系结构的计算机是现代计算机技术研究的另一个焦点。冯·诺依曼体系结构的"存储程序和程序控制"原理表现为"集中顺序控制"这一串行机制,已成为限制计算机性能提高的障碍。要从根本上提高计算机性能,就必须突破冯·诺依曼体系结构的限制,方法之一就是采用并行处理技术。

目前计算机科学家正在研究新一代计算机,如神经网络计算机、生物计算机、光子计算机等。

1.2 运算基础

计算机的基本功能是对数的运算和处理。计算机中,通过数字化编码技术对所表示的数值、文字、符号及控制信息等进行数字编码,这种数字化表示方法不仅要符合人的自然习惯,同时要满足机器中所用器件、线路的工作状态以及数据可靠传输与易于校验纠错等方面的要求。一个具有两种不同的稳定状态且能相互转换的器件,就可以用来表示一位二进制数,由于表示二进制的器件易于制造且工作可靠,并且二进制数的运算规则也最简单,因此目前计算机中均采用二进制数来表示各种信息及进行信息处理。

1.2.1 进位计数制

按进位的方法进行计数称为进位计数制。数是客观事物的量在人头脑中的反映,可用不同的数制来度量。同一个量用不同的数制度量的结果不同。在日常生活中,人们最熟悉、最常用的是十进制、七进制(星期)、十二进制和六十进制(时间)等。在计算机中,常用二进制和十六进制。

一个 R 进制数具有以下主要特点：

1）具有 R 个不同的数字符号：0、1、2、…、R－1。

2）逢 R 进一。

任一 R 进制数 S 可用其若干个数字符号的组合来表示，如 $a_{n-1}a_{n-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-m}$（这种书写方法称为位置表示法），其中 n 为小数点前的位数，m 为小数点后的位数，a_i 是 R 进制的一个数字符号，R 称为基数。上述 R 进制数 S 可用多项式（称为按权展开式）表示：

$$\begin{aligned}S &= a_{n-1}a_{n-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-m}\\ &= a_{n-1}\times R^{n-1}+a_{n-2}\times R^{n-2}+\cdots+a_1\times R^1+a_0\times R^0+a_{-1}\times R^{-1}+\cdots+a_{-m}\times R^{-m}\end{aligned}$$

显然，位置表示法中，每个数码 a_i 所代表的数值等于该数码乘以一个与所在数位有关的常数 R^i，如 R^1、R^0、R^{-1}、R^{-2}等，这些常数称为位权，简称“权”。显然，同一个数码所处位置不同，其权也不同，代表的数值大小也不同，这正是位置表示法的含义。

1. 十进制数

特点：1）具有 10 个不同的数字符号，即 0～9。

2）逢十进一。

一个十进制数可以用它的按权展开式表示。例如：

$$(758.75)_{10}=7\times10^2+5\times10^1+8\times10^0+7\times10^{-1}+5\times10^{-2}$$

其中用$(\ )_{10}$的形式表示括号中的数是十进制数，其他进制数也用类似的形式表达。

2. 二进制数

特点：1）具有 2 个不同的数字符号，即 0 和 1。

2）逢二进一。

一个二进制数可以用它的按权展开式表示。例如：

$$\begin{aligned}(10110.101)_2 &= 1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}\\ &= (22.625)_{10}\end{aligned}$$

3. 十六进制数

特点：1）具有 16 个不同的数字符号，即 0～9 和 A～F。

2）逢十六进一。

一个十六进制数可以用它的按权展开式表示。例如：

$$(1AF.4)_{16}=1\times16^2+10\times16^1+15\times16^0+4\times16^{-1}=(431.25)_{10}$$

表 1-1 给出了上述 3 种进位计数制的对应关系。

表 1-1　3 种数制对照表

十　进　制	二　进　制	十六进制	十　进　制	二　进　制	十六进制
0	0000	0	9	1001	9
1	0001	1	10	1010	A
2	0010	2	11	1011	B
3	0011	3	12	1100	C
4	0100	4	13	1101	D
5	0101	5	14	1110	E
6	0110	6	15	1111	F
7	0111	7	16	10000	10
8	1000	8	17	10001	11

4. 各种数制之间的转换

1）二进制、十六进制转换成十进制。采用按权展开式计算求和的方法，如前例。

2）十进制转换成二进制、十六进制。整数部分采用除基取余法，小数部分采用乘基取整法。

【例 1-1】十进制数 22.625 转换为二进制数。

整数部分：

$2 \lfloor 22$

$2 \lfloor 11$ …余 0 （低位）

$2 \lfloor 5$ …余 1

$2 \lfloor 2$ …余 1

$2 \lfloor 1$ …余 0

0 …余 1 （高位）

小数部分：

$$\begin{array}{r} 0.625 \\ \times \quad 2 \\ \hline \boxed{1}.25 \end{array}$$ …取整数 1 （高位）

$$\begin{array}{r} \times \quad 2 \\ \hline \boxed{0}.5 \end{array}$$ …取整数 0

$$\begin{array}{r} \times \quad 2 \\ \hline \boxed{1}.0 \end{array}$$ …取整数 1 （低位）

所以：$(22)_{10} = (10110)_2$　　$(0.625)_{10} = (0.101)_2$

故：$(22.625)_{10} = (10110.101)_2$

【例 1-2】十进制数 430.25 转换为十六进制数。

整数部分：

$16 \lfloor 430$

$16 \lfloor 26$ …余 14→E（低位）

$16 \lfloor 1$ …余 10→A

0 …余 1（高位）

小数部分：

$$\begin{array}{r} 0.25 \\ \times \quad 16 \\ \hline \boxed{4}.0 \end{array}$$ …取整数 4

故：$(430.25)_{10} = (1AE.4)_{16}$

注意：① 整数部分转换，每次只求整数商，将余数作为转换结果的一位，重复对整数商除基数，一直除到商为 0 为止；② 小数部分转换，每次把乘积的整数取走作为转换结果的一位，对剩下的小数继续进行乘法运算。对某些数可以乘到积的小数为 0（如上述两例），这种转换结果是精确的；对某些数（如 0.3）永远不能乘到积的小数为 0，这时要根据精度要求，取适当的结果位数即可，这种转换结果是不精确的。

3）二进制与十六进制间的相互转换。因为 $2^4 = 16$，所以每 1 位十六进制数对应 4 位二进制数（参见表 1-1），因此，只要用 4 位二进制数代替对应的 1 位十六进制即可完成十六进制到二进制的转换。

例如：十六进制数　1　A　E　.　4

↓　↓　↓　↓

0001　1010　1110　.　0100

即 $(1AE.4)_{16} = (110101110.01)_2$

反之，若要将二进制数转换为十六进制数，只要以小数点为分界，分别向左和向右每 4 位二进制位分为一组（若最高位或最低位不够 4 位则补 0），对应转换为十六进制数即可。

例如：二进制数　110101110.01

↓

0001　1010　1110　.　0100

↓　↓　↓　↓

十六进制数　1　A　E　.　4

即 $(110101110.01)_2 = (1AE.4)_{16}$

5. 二进制数的算术运算

二进制算术运算与十进制的运算方法基本相同,但在二进制运算时,逢二进位,借一当二。

【例 1-3】10100 + 1101 = 100001

```
  10100
+  1101
------
 100001
```

【例 1-4】100001 - 10100 = 1101

```
 100001
-  10100
------
   1101
```

【例 1-5】1101 × 1011 = 10001111

```
   1101
×  1011
------
   1101
  1101
 0000
+1101
--------
 10001111
```

【例 1-6】11100 ÷ 101 = 101……11

```
      101 ……商
101 ) 11100
      101
      ---
       1000
        101
        ---
         11   ……余数
```

6. 二进制数的逻辑运算

逻辑运算又称为布尔运算,是计算机中二进制的基本运算。常用的逻辑运算有逻辑与运算(AND)、逻辑或运算(OR)、逻辑非运算(NOT)和逻辑异或运算(XOR)。其运算规则如表 1-2 所示。

表 1-2　二进制数的运算规则

a	b	NOT a	NOT b	a AND b	a OR b	a XOR b
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	1	0

【例 1-7】10100101 AND 10001011 = 10000001

```
  10100101
AND 10001011
------------
  10000001
```

【例 1-8】10100101 OR 10001011 = 10101111

```
  10100101
OR 10001011
-----------
  10101111
```

【例 1-9】NOT 10100101 = 01011010

```
NOT 10100101
------------
  01011010
```

【例 1-10】10100101 XOR 10001011 = 00101110

```
  10100101
XOR 10001011
------------
  00101110
```

1.2.2　二进制编码

如上所述,在计算机中,数是用二进制表示的。而计算机还应该能够识别和处理各种文字信息,如大小写英文字母、标点符号、运算符号等,这些符号又怎样表示呢? 由于计算机的基本物理器件是具有两个状态的器件,所以各种文字信息也只能用若干位的二进制编码组合来表示。

1. 二进制编码的十进制数(BCD 码—Binary Coded Decimal)

因为二进制数实现容易、可靠,且运算规律简单,所以在计算机内部采用二进制数。但是,二进制数不直观,在计算机进行输入和输出与人交换信息时,通常还是采用十进制数。为了转换直观方便,要将十进制数用二进制编码来表示,这就是 BCD 码。

1 位十进制数需要用 4 位二进制编码,编码的表示方法很多,较常用的是 8421BCD 码,表 1-3列出了 8421BCD 码和十进制数的对应关系。

表 1-3　8421BCD 码和十进制数的对应关系

十　进　制	8421BCD 码	十　进　制	8421BCD 码
0	0000	6	0110
1	0001	7	0111
2	0010	8	1000
3	0011	9	1001
4	0100	10	0001 0000
5	0101	11	0001 0001

BCD 码是十进制数,有 10 个不同的数字符号,且是逢十进位的;但它的每一位是用4 位二进制编码来表示的,因此称为二进制编码的十进制数。BCD 码比较直观,例如十进制数 65 用 BCD 码书写为 01100101,BCD 码 01001001.0111 表示的十进制数为 49.7。

所以,只要熟悉了 BCD 码的 10 个编码,就可以很容易地实现十进制与 BCD 码之间的转换。

虽然 BCD 码是用二进制编码方式表示的,但它与二进制之间不能直接转换,要用十进制作为中间桥梁,即先将 BCD 码转换为十进制数,然后再转换为二进制数;反之亦然。

2. 字母与字符的编码

如上所述,字母和字符也必须按照特定的规则,用二进制编码才能在机器中表示。编码可以有各种方式,目前微机中最普遍采用的是 ASCII 码(American Standard Code for Information Interchange,美国标准信息交换码),编码表见附录 C。

ASCII 码采用 7 位二进制编码,故可表示 $2^7 = 128$ 个字符,其中包括数码 0 ~ 9 以及英文字母等可打印的字符。从表中可以看到,数码 0 ~ 9 用 0110000 ~ 0111001 来表示。因微型机字长或内存单元通常是 8 位,所以通常把最高位用作奇偶校验位,但在机器中表示时,一般认为是 0,故用一个字节(8 位二进制数)来表示一个字符的 ASCII 码值。于是 0 ~ 9 的 ASCII 码为 00110000B ~ 00111001B 即 30H ~ 39H,大写字母 A ~ Z 的 ASCII 码为 41H ~ 5AH,小写字母 a ~ z 的 ASCII 码为 61H ~ 7AH。

另外,在计算机中,汉字编码采用国标码(GB 18030 - 2005),它采用单、双、四字节混合编码,每个字节的最高位为 1,并以此来区分汉字和 ASCII 码。

1.2.3　带符号数的表示

上面提到的二进制数没有涉及符号问题,故是一种无符号数的表示。但是在实际应用中,数当然会有正负,那么计算机中如何表示数的符号呢? 通常规定数据的最高位为符号位,即若是字长为 8 位,则 D_7 位是符号位,$D_6 \sim D_0$ 为数字位。并规定符号位用 0 表示正,用 1 表示负。

如:$X = (00001100)_2$,则 X 的值为 +12;$Y = (10001100)_2$,则 Y 的值为 -12。

在计算机中,带符号数有 3 种表示法——原码、反码和补码。用原码、反码、补码表示的数称为机器数,机器数对应的数学值称为真值。

1. 原码

如上所述,正数的符号位用 0 表示,负数的符号位用 1 表示,数值位保持不变,这种表示法称为原码。原码的定义:

若 $X \geqslant +0$,则 $[X]_{原} = X$;

若 $X \leqslant -0$,则 $[X]_{原} = 2^{n-1} - X$;其中 n 为原码的位数。

例如,设用 8 位表示原码,则$[+124]_{原} = 01111100$,$[-124]_{原} = 11111100$。

其中,最高位 D_7 为符号位,后 7 位 $D_6 \sim D_0$ 是数值位。如果字长是 16 位,则 D_{15}是符号位,$D_{14} \sim D_0$ 是数值位。在用原码表示时,8 位二进制原码真值范围为 $-127 \sim +127$;16 位二进制原码真值范围为 $-32767 \sim +32767$。8 位二进制原码的表示见表 1-4。

原码表示法简单易懂,而且与真值转换方便。但若是两个异号数相加(或两个同号数相减)就需要做减法。为了简化运算器设计(把减法运算转换为加法运算),引入了反码和补码。

表 1-4　8 位二进制原码的表示

十进制数	二进制数	原　码	反　码	补　码
-128	-10000000	——	——	10000000
-127	-1111111	11111111	10000000	10000001
-126	-1111110	11111110	10000001	10000010
…	……	……	……	……
-2	-0000010	10000010	11111101	11111110
-1	-0000001	10000001	11111110	11111111
-0	-0000000	10000000	11111111	00000000
+0	+0000000	00000000	00000000	00000000
+1	+0000001	00000001	00000001	00000001
+2	+0000010	00000010	00000010	00000010
…	……	……	……	……
+126	+1111110	01111110	01111110	01111110
+127	+1111111	01111111	01111111	01111111

2. 反码

反码的定义:

若 $X \geqslant +0$,则 $[X]_{反} = X$;

若 $X \leqslant -0$,则 $[X]_{反} = 2^n - 1 + X$;其中 n 为反码的位数。

显然,正数的反码表示与原码相同,最高位为符号位,其余位为数值位。

例如:$X = +4$,则$[X]_{反} = [X]_{原} = 00000100$。

而负数的反码应当表示为该数的原码除符号位外其余位按位取反。

例如:$X = -4$, 则$[X]_{原} = 10000100$,$[X]_{反} = 11111011$。

$X = -31$,则$[X]_{原} = 10011111$,$[X]_{反} = 11100000$。

负数的反码表示与原码有很大的区别:最高位相同,仍是“1”,但数据位的值完全相反,这一点要特别注意。8 位二进制反码的表示见表 1-3,它有以下特点:

1)“0”有两种表示方法:$[+0]_{反} = 00000000$,$[-0]_{反} = 10000000$。

2) 8 位二进制反码真值范围为 $-127 \sim +127$,16 位反码真值范围为 $-32767 \sim +32767$。

3) 当一个带符号数用反码表示时,最高位为符号位。若符号位为 0,说明该数是正数,后面各位即是其真值;若符号位为 1,说明该数为负数,其真值为后面各位按位取反。例如:已知 $[X]_{反} = 00000101$,则 $X = +5$;$[Y]_{反} = 11111110$,则 $Y = -1$。

反码表示对计算机的结构有特殊要求,现在很少采用。

3. 补码

1) 模的概念。在钟表上,指针正拨 12 小时或倒拨 12 小时,其时间值是相等的,即在钟表上 $X + 12 = X - 12 \pmod{12}$。对钟表来说,它的模为 12。

对于 n 位二进制计数器,其计数范围为 0 ~ (2^n-1),在该计数器上加 2^n 或减 2^n 结果是不变的,人们称 2^n 为 n 位计数系统的模。

由上述可见,一个计数系统,某数加(或减)其模,结果不变。

2) 补码的引入。在钟表上,如果现在的时间是 6 点整,而钟表却指着 8 点整,快了 2 小时,校准的方法是正拨 10 小时或倒拨 2 小时,结果都正确,即:

$$8+10=6(\text{mod } 12)\text{顺拨} \qquad 8-2=6(\text{mod } 12)\text{倒拨}$$

即加 10 和减 2 效果相同,那么可以用加 10 运算来代替减 2 运算。

在数学上,如 a 和 b 满足:a MOD M =(nM + b)MOD M(其中 n 为正整数,M 为模),则称 a 和 b 对 M 同余,也称 a 和 b 互为补数(以 M 为模)。

在时钟上,+10 与(-2),+7 与(-5),+6 与(-6)对 12 同余或称互为补数。

引入补数的概念后就可以将减法转化为加法来计算。

3) 补码的求法。对 n 位二进制数,模为 2^n,则 $[X]_{补}=(2^n\times i+X)\ \text{MOD}\ 2^n$,i 为正整数。

补码的定义:

若 $X\geqslant +0$,则 $[X]_{补}=X$(即取 i=0);

若 $X\leqslant -0$,则 $[X]_{补}=2^n+X$(即取 i=1);其中 n 为补码的位数。

当 $X\geqslant 0$ 时,$[X]_{补}=X$,即正数的补码为原正数不变,显然正数的原码、反码、补码相同。当 $X\leqslant 0$,$[X]_{补}=2^n-1+X+1=[X]_{反}+1$,即负数的补码等于负数的反码加 1,也就是等于负数原码除符号位外按位取反、最低位加 1。下面举例说明补码的求法与应用。

$[+3]_{补}=[+3]_{原}=[+3]_{反}=00000011$ $[-3]_{补}=[-3]_{反}+1=11111100+1=11111101$

$[+0]_{补}=[+0]_{原}=[+0]_{反}=00000000$ $[-0]_{补}=[-0]_{反}+1=11111111+1=00000000$

特别注意,$[+0]_{补}=[-0]_{补}==00000000$,即 0 的补码为 0,且只有一种表示方法。

8 位带符号数的补码表示也列在表 1-4 中,它有以下特点:

① $[+0]_{补}=[-0]_{补}=00000000$。

② 8 位二进制补码真值范围为 -128 ~ +127,16 位补码真值范围为 -32768 ~ +32767。

③ 一个用补码表示的二进制数,最高位为符号位,当符号位为“0”即正数时,其余位即为此数的二进制值;但当符号位为“1”即负数时,其余位不是此数的二进制值,其值为后面各位按位取反、最低位加 1。

例如:$[X]_{补}=00001111$,则 X = +15;$[X]_{补}=11110001$,则 X = -15;

$[X]_{补}=10000000$,则 X = -128。

当采用补码表示时,可以把减法运算转换为加法运算,即 $[X\pm Y]_{补}=[X]_{补}+[\pm Y]_{补}$。

例如:X = 64 - 9 = 64 + (-9),则 $[X]_{补}=[64]_{补}+[-9]_{补}$。

$[+64]_{补}=01000000$,$[+9]_{原}=00001001$,$[-9]_{补}=11110111$,

于是:

$$\begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ +\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1 \\ \hline \boxed{1}\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1 \end{array} \qquad \begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ -\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1 \\ \hline 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1 \end{array}$$

└→自然丢失

显然,加法和减法运算的结果完全相同。原因是对 8 位二进制数,当运算结果超过 $2^8=256$ 时,在机器上体现为又从零开始,即 8 位字长的最大数不能超过 256,也就是模是 256。正是由于此,减法运算才能转换为加法运算。

4. 补码运算的溢出及其判断方法

溢出是指运算结果超出了规定长度数据的表数范围,在此特指带符号数的补码运算溢出。例如,对字长为 n 位的补码表示的带符号数,其最高位表示符号,其余 n－1 位表示数值,其表数范围为 $-2^{n-1} \sim +2^{n-1}-1$。如果一个运算的结果超出了这个范围,就称为补码溢出(简称溢出),这时的运算结果将是错误的。

例如,对于 8 位字长的二进制补码数,其表数范围为 －128 ~ ＋127。如果运算结果超出了此范围,就会产生溢出。

例如,用 8 位二进制补码分别计算 60＋100、(－60)＋(－100)、60＋(－100)。

已知$[60]_{补}=00111100$,$[-60]_{补}=11000100$,$[100]_{补}=01100100$,$[-100]_{补}=10011100$。

$$\begin{array}{rr} [60]_{补}= & 00111100 \\ +\ [100]_{补}= & 01100100 \\ \hline & 10100000 \\ & \downarrow \\ & \text{符号} \end{array} \qquad \begin{array}{rr} [-60]_{补}= & 11000100 \\ +\ [-100]_{补}= & 10011100 \\ \hline & 101100000 \\ & \text{自然丢失} \leftarrow\!\!\lrcorner\downarrow \\ & \text{符号} \end{array} \qquad \begin{array}{rr} [60]_{补}= & 00111100 \\ +\ [-100]_{补}= & 10011100 \\ \hline & 11011000 \\ & \downarrow \\ & \text{符号} \end{array}$$

即$[60+100]_{补}=10100000$,两个正数相加,结果却为负数,显然是错误的;

$[(-60)+(-100)]_{补}=01100000$,两个负数相加,结果却为正数,显然也是错误的;

$[60+(-100)]_{补}=11011000=-40$。

前两个运算结果之所以不正确,是因为其相加结果分别为＋160 和－160,均超出了表数范围,使结果的数值部分占据了符号位,产生了溢出错误。但一个正数与一个负数相加,一定不会产生溢出错误。

判断溢出的方法很多,上例根据参加加法运算的两个数据的符号及运算结果的符号可以判断是否溢出。在计算机中,经常根据加法运算中在最高位与次高位的两个进位来判断。设 8 位二进制数的各位记为 $D_7D_6D_5\cdots D_0$,运算中两个 D_6 位的进位记为 C_6,两个 D_7 位的进位记为 C_7,用 $OV=C_7\ XOR\ C_6$(XOR 是逻辑异或运算)判别式可以判断溢出情况。如果 OV＝0,表示结果无溢出,否则当 OV＝1 时,表示结果有溢出。

请注意进位与溢出的区别。进位是指运算结果的最高位向更高位的进位,如上所述的 8 位运算中的 C_7。进位通常记做 C_y,$C_y=0$ 表示无进位,$C_y=1$ 表示有进位。而溢出是用最高位进位(即 C_y)与次高位进位的逻辑异或结果来判断的。通过上例可以看出,有进位不一定就有溢出,无进位也不一定就无溢出。同理,有溢出不一定就有进位,无溢出也不一定就无进位(请计算(－60)＋100 来验证)。可见,进位和溢出是两个不同性质的概念,不能混淆。

1.2.4 数的定点和浮点表示

在实际中,计算机所处理的数一般是带有小数点的数,它既有整数部分,又有小数部分,这就提出一个小数点放在什么位置的问题。在计算机中,通常有两种方法,即定点表示法和浮点表示法。采用定点表示法的计算机称为定点机,采用浮点表示法的称为浮点机。所谓定点与浮点,意指计算机中数的小数点位置是固定的还是浮动的。

1. 定点表示法

在定点表示法中,小数点的位置是固定不变的。一般有以下两种简单的约定:

1) 定点小数法。约定小数点在符号位之后、数值部分最高位之前,因此数据是纯小数,故又称定点小数,其格式:

符号位	数值部分(尾数)

↑—小数点位置

采用定点小数表示数据时,在机器中参与运算的数总是小于 1 的纯小数;但实际中需要计算机处理的数总有大于 1 的,这就要求机器在解题之前,选择适当的比例因子,使参加运算的所有数都缩小若干倍,化为小于 1 的数,而计算结果又必须用相应的比例增大若干倍,使其恢复为原有的数值。这就需要预先对每步计算结果给予估计(应该小于 1),否则计算机将产生“溢出”,机器无法表示即发生错误。

定点小数法比例因子的选择:例如有两个数为 010.01 和 001.100,若进行两数相加时,如 $010.01+001.100=(0.1001+0.0110)\times 2^{2}$。

该比例因子选为 2^{2},而且两数相加结果仍小于 1。如果比例因子选得太小,定点小数缩小比例不够,其结果可能有溢出(结果不小于 1),如缩小比例过大,又会损失有效精度。

2)定点整数法。约定小数点的位置固定在数值部分的最低位之后,也就是把数表示为纯整数,机器中参与运算的数都是纯整数,其格式如下:

符号位	数值部分(尾数)

小数点位置—↑

定点整数法也有比例因子的选择问题,例如上例两个数化为定点整数运算则为(010.01 + 001.100) = (01001 +00110) $\times 2^{-2}$,该比例因子选为 2^{-2}。

在计算机中,小数点实际上是不表示出来的,需要编程人员事先约定小数点的位置,即选择适当的比例因子。这种做法并不完美,因为在一个定点运算程序中的比例因子要适用于多个不同位数、值大小差异很大的浮点数据,很难同时满足在有限位数内既不扩大数的表示范围以免溢出,又可保证数的有效精度的要求。解决这个矛盾的办法,是将比例因子以恰当的形式表示在数之中,使之能根据每一个数的需要而浮动,尽可能地协调该数在表示范围与精度两方面的要求。因此,引入了数的浮点表示法。

2. 浮点表示法

在浮点表示法中,小数点的位置是不固定的或者说是可浮动的。例如,十进制数 63.8 可表示为 0.638×10^{2} 或 6.38×10^{1} 或 63.8×10^{0} 或 638×10^{-1} 等,二进制数 1101.001 可表示为 0.1101001×2^{4} 或 0.01101001×2^{5} 或 110100.1×2^{-2} 等。一般来说,任何一个二进制浮点数可表示为 $N=2^{P}\times S$。

其中,P 为阶码,S 为尾数,它们都有各自的符号位。阶码的符号位又称阶符,用 P_f 表示,阶码由 P_1、$P_2\cdots P_m$ 位组成,一般为整数;尾数的符号位又称数符,用 S_f 表示,尾数由 S_1、$S_2\cdots S_n$ 位组成,一般为纯小数。式中的 2 是阶码的“底”,它与尾数的基数应是相同的,如对二进制来说,尾数部分是二进制数码,那么阶的“底”也应取 2。

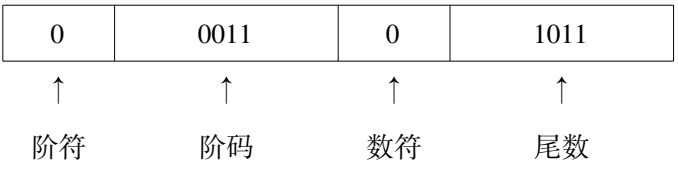
浮点数在计算机中的表示形式如下:

P_f	$P_1P_2\cdots P_m$	S_f	$S_1S_2\cdots S_n$
↑	↑	↑	↑
阶符	阶码	数符	尾数

在这种表示中,尾数和阶码可以采用原码、反码或补码形式。浮点数的表示范围主要由阶码决定,阶码规定了小数点的位置,它相当于定点表示法中的比例因子,但现在阶码成为了浮点数

的一部分。由于阶码的值是可以变化的,因此数的小数点是可以浮动的。

例如,二进制数 $N=2^{+11}\times 0.1011$,在浮点机中的表示格式:



在上述定点表示法和浮点表示法中,两者都有一个尾数"规格化"问题。由于有效数字的精度主要由尾数决定,为了充分利用尾数的有效位数,要把尾数的绝对值限定在某个范围内,规定当尾数满足 $0.5\leqslant|S|<1$ 时,即为规格化数。从而看出,所谓规格化数,即尾数的最高有效位是有效数字 1,而不是 0。对定点小数来说,用选择恰当的比例因子实现规格化;对浮点表示法,需要调整阶码的数值实现规格化。

例如:$N=2^{+11}\times 0.0101$ 是非规格化浮点数,若改写成:$N=2^{+10}\times 0.1010$,即为规格化浮点数。

3. 定点表示方法与浮点表示法的比较

1)数值表示范围。假定某机器字长 32 位,数符占 1 位,尾数占 31 位,则:

定点小数表示范围为 $-(1-2^{-31})\sim+(1-2^{-31})$;

定点整数表示范围为 $-(2^{31}-1)\sim+(2^{31}-1)$;

用浮点表示,若字长 32 位,其中阶码 8 位(包括 1 位阶符),24 位尾数(包括 1 位数符),其最大阶码为 $2^7-1=127$。

故它能表示数的范围为 $2^{-128}\times 2^{-23}\leqslant|S|\leqslant 2^{127}\times(1-2^{-23})$。

近似为 $0\leqslant|S|\leqslant 2^{127}$,即数的表示范围近似为 $-2^{127}\sim+2^{127}$。

显然,用浮点表示法比定点表示法表示数的范围大得多。尽管它的尾数只有 24 位,精度稍许降低,但它换来更大的数据表示范围,因而编制程序时,不必担心出现"溢出"错误。

2)浮点数的运算比定点数的运算复杂。由于浮点数的小数点位置隐含于阶码之中,阶码不同的两个尾数不能直接加减,需先把两个数的阶码调整到一致,这称为"对阶",然后两个尾数才能相加减。这些操作控制复杂,其硬件代价较高。

在微型机中一般采用定点表示法,是定点机。但可通过软件手段即浮点运算子程序来实现浮点运算;也可附加浮点运算的扩充部件,指令系统可借助于传送指令将数据送往浮点扩充部件中进行浮点运算。在高性能的微型机或小、中、大型计算机中,一般都有浮点运算指令并配有浮点运算部件。

1.3 计算机系统的组成及程序执行过程

计算机系统由硬件系统和软件系统两部分组成。硬件是指组成计算机的各种电子的、机械的、光磁学的物理器件和设备,它们构成了计算机的物理实体。软件则是指为了运行、管理和维护计算机而编制的各种程序及其有关的文档资料的总称。硬件是基础,软件是灵魂,两者既相互独立,又相互依存,缺一不可。硬件和软件合起来才能组成一个完整的计算机系统。

1.3.1 计算机硬件系统组成及程序执行过程

1946 年,美籍匈牙利科学家冯·诺依曼(Von Neumann)就提出了计算机体系结构设计的一些思想,包含以下 3 个基本要点:

1）采用二进制数的形式表示指令和数据。

2）将指令序列(程序)和数据预先存入计算机的存储器中;程序执行时,能自动、连续地从存储器中逐一取出指令并执行之。

3）计算机硬件由运算器、控制器、存储器、输入设备、输出设备5大部分组成。

按照这种思想设计的计算机称为冯·诺依曼型计算机,其工作原理的核心是“存储程序”和“程序控制”,即“集中顺序控制”。冯·诺依曼提出的这些基本概念奠定了现代计算机体系结构的基本框架,并由此产生了程序设计思想。尽管从计算机诞生到现在已经历了半个多世纪,计算机的体系结构已发生了很大变化,计算机的性能也有了巨大提高,但目前大多数计算机仍遵从冯·诺依曼体系结构理论。

1. 计算机硬件的基本结构

电子数字计算机一开始是作为一个计算工具出现的。若要计算机能脱离人的直接干预,自动地完成计算,它应具有哪些主要部分呢?

首先,计算机要进行计算,必须有能完成算术运算和逻辑运算的部件,称之为算术逻辑部件(Arithmetic Logic Unit,ALU),简称运算器。仅有运算器是不能够自动计算的,进行运算还必须有原始数据,这些数据到什么地方去找呢?计算结果也必须有地方保存,以备后面的计算使用。此外,要脱离人的直接干预,也必须把进行计算的顺序、过程和步骤以命令序列(程序)的形式预先存放好,以备计算机执行。因此,计算机还必须有一个专门存放原始数据、中间结果和最终结果以及程序的地方,称之为存储器。当计算机执行命令时,这是个什么命令呢?执行这个命令需要做哪些动作或操作呢?完成这些操作需要哪些部件执行哪些动作呢?因此,计算机还必须有一个能分析命令并指挥协调各部件统一行动完成命令规定的各种动作或操作的部件,称之为控制器。

有了运算器、存储器和控制器,计算机似乎可以自动完成运算了,但原始数据和程序如何进入到存储器呢?计算结果又如何通知用户呢?这就需要计算机有能与外界交换信息的输入设备和输出设备(简称I/O设备或外围设备,如键盘、鼠标、光笔、扫描仪、显示器、打印机、绘图仪、磁盘机、光驱等),以及把由运算器、存储器和控制器组成的系统与I/O设备连接起来的接口电路和其他辅助电路,这样便形成了计算机的硬件系统,如图1-1所示。

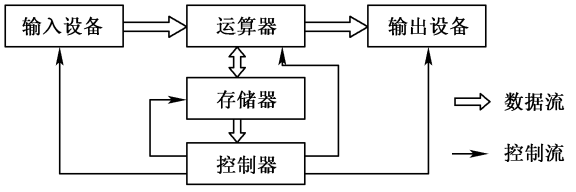


图1-1　计算机的基本硬件组成

图1-1是典型的冯·诺依曼型计算机的结构。可以看出,计算机内部有两类信息在流动,一是数据流,二是控制流。程序和数据通过输入设备送入到存储器中;控制器从存储器中逐条取出命令并加以分析,按照命令的功能规定向各个部件发出一系列的控制信号来执行这条命令;存储器把参加运算的数据送给运算器;运算器按规定的运算操作进行计算,并把结果送回到存储器中保存。最后把处理的结果通过输出设备送给外界。这就是计算机工作的基本过程。

综上所述,控制器是计算机的大脑和控制指挥中心,计算机的每一步工作都离不开控制器的控制。为了帮助读者更好地理解计算机的组成和工作过程,下面简单讨论一下运算器、存储器和

控制器。

2. 运算器、存储器和控制器

运算器主要包括能完成加、减、乘、除算术运算及逻辑运算的电路以及多个寄存器。在控制信号的指挥下,运算器完成诸如算术运算、逻辑与移位运算、暂存操作数或运算结果以及数据传送等工作。

存储器由许多存储单元组成,每一个存储单元可以存放若干个二进制位(bit),在微型机中存储单元通常以 8 位即 1 字节为单位。为了能唯一确定并找到任一个存储单元,计算机对每一存储单元都指定一个唯一的编号,称之为存储单元的地址,地址通常从 0 开始顺序编排。这里读者必须把存储单元的地址与其所存储的内容区分开来。人们可以把存储器想像成一个含有许多小抽屉的大柜子。小抽屉即相当于存储单元,每个小抽屉都有一个编号——相当于存储单元的地址,在小抽屉内放有书、本等各种东西——相当于存储单元中存放的各个数据即内容,如图 1-2 所示。

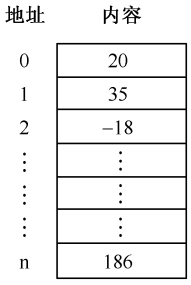


图 1-2 存储单元的地址与内容

当需要把某一单元的内容读出或把数据写入到某一单元中时,存储器就必须能根据地址找到该存储单元,这些工作要由存储器的地址译码器和读/写控制电路等来完成。

由于计算机的速度很高,每秒能执行数百万条指令,通常每条指令需要若干个操作数,因此就要求存放程序和数据的存储器容量极大、速度极高,这从性能、价格和技术上都是难以同时实现的。为解决这一问题,通常把存储器分为内存(主存)和外存(辅存)两类。内存是用高速电子线路制造的,速度极高、容量较小、价格较贵,用于存放当前正在运行的程序块和正在处理的数据块。外存主要有磁盘、磁带和光盘等,速度稍慢、容量极大、价格低廉,用于存放那些暂不急需或不使用的程序和数据,当需要时,再把外存上的程序和数据调入内存去执行和处理。通常把外存归入到 I/O 设备类,本书所讲存储器主要是指内存。

控制器主要由指令指针寄存器(IP)、指令寄存器(IR)、指令译码器(ID)和控制信号发生器组成。指令指针寄存器中存放将要执行的指令的地址,按此地址从内存取出指令,并送到指令寄存器中暂存,然后由指令译码器进行分析译码,根据译码结果由控制信号发生器发出一系列控制信号到运算器、存储器等各功能部件来完成该指令的执行。一个简单计算机的硬件电路结构框图如图 1-3 所示。

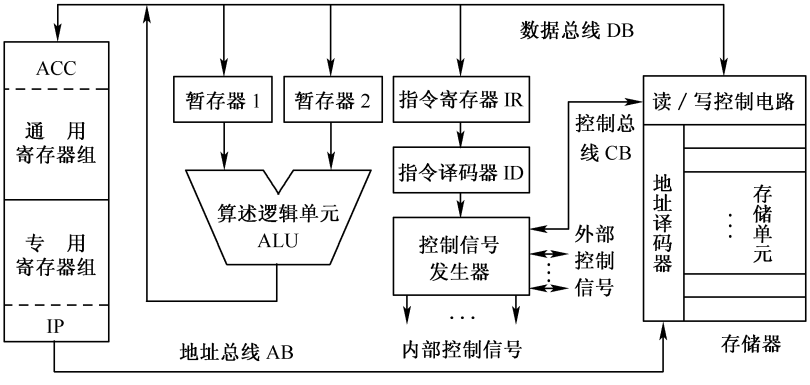


图 1-3 简单计算机的硬件电路结构框图

3. 程序执行过程

计算机的工作过程就是执行程序的过程，而程序是由一系列指令组成的，因此程序的执行过程就是执行指令序列的过程，也就是一条一条地执行指令的过程。每一条指令的执行都要先从存储器中取出指令（称为取指阶段），然后由控制器进行分析译码、发出一系列控制信号完成该指令的执行（称为分析执行阶段）。

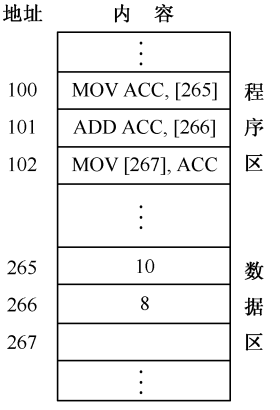


图 1-4 示例程序和数据

下面以一个简单的例子来说明计算机执行程序的主要过程。假设人们已通过输入设备把程序和数据存入到存储器中，如图 1-4 所示。本例中，假设程序段共有 3 条指令，每条指令占用 1 个存储单元。第一条指令 MOV ACC,[265]的功能是将内存地址为 256 的单元内容读出并送入 CPU 内部寄存器 ACC 中；第二条指令 ADD ACC,[266]的功能是将 ACC 的内容与 266 单元的内容相加，结果存到 ACC 中；第三条指令 MOV [267],ACC 的功能是将 ACC 的内容存到 267 单元中。

当执行该程序时，用户首先通过键盘把程序第一条指令的地址（100）送到 CPU 的指令指针寄存器（IP）中并启动执行。执行过程如下。

1）取指阶段：把 IP 中存放的指令地址送到存储器（M）：IP→M；之后，IP 自动加 1，指向下一条指令：IP + 1→IP。

存储器对地址译码，找到 100 单元，并读出其内容，送入指令寄存器（IR）：M→IR。

2）分析执行阶段：IR 把指令送入指令译码器（ID），ID 对指令译码后，由控制信号发生器产生如下一系列控制信号来执行这条指令：

① 操作数地址（265）送到存储器：addr→M。

② 读存储器，把指定单元的内容送到规定的累加器 ACC 中：M→ACC。

至此，第一条指令执行完毕。由上可以看出，一条指令的执行分成取指阶段和分析执行阶段。需要强调的是，不管是取指阶段还是分析执行阶段，每一项操作如地址传送、存储器读写、数据传送等，都是在控制器发出的控制信号指挥下按照严格的时间顺序一步一步完成的。由于在取指阶段取出指令后 IP 已自动加 1，指向了下一条指令，因此，以后的执行过程就是周而复始地重复上述的取指、分析执行阶段。

3）取指阶段：IP→M，IP + 1→IP，M→IR。

4）分析执行阶段：IR→ID，译码、执行。① addr→M；② M→暂存器 1；③ ACC→暂存器 2；④ ALU 执行“ADD”运算；⑤ ALU 结果→ACC。

5）取指阶段：IP→M，IP + 1→IP，M→IR。

6）分析执行阶段：IR→ID，译码、执行。① addr→M（送地址）；② ACC→M（送数据）；③ 存储器写操作。

至此，上述程序段执行完毕。

4. 控制器设计

如上所述，在计算机工作过程中，控制器每时每刻都在发出各种控制信号，统一指挥各部件协调一致地工作，那么控制器该如何设计呢？

用计算机解决问题的基本方法是把一个复杂的问题分解成许多小的简单的问题，这就是所谓的“自顶向下，逐步细化”的方法。在计算机内部设计中同样采取这种方法，如上例中的指令“MOV ACC,[265]”执行过程就分成了 IP→M，IP + 1→IP，M→IR，IR→ID，addr→M，M→ACC 6

个基本操作(称为微操作)。在电子线路中,这些基本操作主要是靠逻辑元器件的开门电平和接收脉冲实现的,例如微操作 IP→M,就是首先给出 IP 寄存器的开门电平,IP 的内容就送到了 AB 总线上,然后再给出存储器的地址接收脉冲,这样就把 AB 上的内容存入到存储器的地址译码器中了。可以看出,信息在电路中流动的过程即为指令的执行过程。人们把这些控制各种信息在电路中流动的基本操作控制信号称为微操作控制信号。控制器的主要功能就是根据指令译码结果,严格按时间节拍产生各种微操作控制信号。需要说明的是,执行一条指令需要产生多个微操作控制信号,同一个微操作控制信号也可能在多条不同指令执行时以及不同的时间节拍中都需产生,如前面讲到的地址送存储器这一微操作,在 3 条指令执行时均数次被用到,所以存储器的地址接收脉冲这一微操作控制信号在每条指令执行时均要产生,甚至还要产生多次。一种型号计算机通常有数百条指令,因此,微操作控制信号的产生条件非常繁多且复杂。

控制器按照其产生微操作控制信号的方式分成传统的组合逻辑控制器和微程序控制器。组合逻辑控制器是把产生每一个微操作控制信号的所有条件综合在一起,列出它的逻辑表达式并化简,然后用组合电路来实现这些逻辑表达式。组合逻辑控制器存在许多缺点:控制网络非常复杂且杂乱无章、设计正确性难以检查、难以扩充功能等。随着微电子技术的发展,微程序技术迅速发展成为设计控制器的主要方法。

众所周知,计算机工作的基本方法是把复杂问题分解成多个简单问题,用一些简单的指令编成程序来解决复杂问题。用类似的方法,可以把指令的执行分解成若干个微操作,用微指令将这些微操作编成微程序,通过执行微程序来完成这些微操作,即执行了指令。这就是微程序技术的基本原理。

微程序控制器就是根据上述原理,把执行每一条指令的微操作按步骤编成微程序,存到位于控制器内的控制存储器中。执行某一指令时,首先找到该条指令对应微程序的开始地址,逐条取出微指令,由微指令直接产生微操作控制信号,就实现了指令的执行。

微程序控制器具有很多优点,每条指令对应一个微程序,因此控制器设计相对独立和简单,而且只要检查每条指令的微程序就可以验证控制器的设计是否正确。当需要扩充系统和指令功能时,只要增加新指令的微程序就可以很方便地实现。

尽管微程序控制器采用的是微程序技术,但读者一定要明白,这样的控制器是用硬件或固件实现的,并非一般意义上的软件或程序。

1.3.2 计算机的软件系统

如上节所述,仅有硬件,计算机是不能工作的。若要计算机正确地运行以解决各种问题,就必须为它编制程序。为了运行、管理和维护计算机所编制的各种程序及其相关文档资料统称为软件。软件的种类很多,各种软件开发的目的都是为了扩大计算机的功能和方便用户使用,以及使用户编制解决各种问题的源程序更为简单、方便和可靠。

1. 系统软件

在计算机的发展初期,人们采用二进制编码来编写程序,这种二进制编码就称为机器语言。但是这种用一串 0、1 书写的机器语言程序太烦琐,不易记忆,且难理解,不便于推广使用。所以人们采用助记符来代替操作码,用符号来代替地址,这便是汇编语言阶段。汇编语言的指令比较容易理解、记忆和交流,但是汇编语言程序不能直接在机器上运行。要运行一个汇编语言程序,必须要把它翻译成机器语言程序,机器才能识别和执行。在早期阶段这种翻译工作是人工完成的,但这项工作非常烦琐。计算机正是解决这类问题的高手,人们便编制了一个称之为汇编程序

的翻译程序，把用汇编语言编写的程序交给汇编程序翻译成机器语言程序。虽然如此，由于汇编语言的语句和机器语言的指令是一一对应的，所以编制汇编语言程序仍然是一件很困难的工作。

为了使用户编程更容易，程序中的语句与实际应用更接近，而且用户不必了解具体的机器就能编写程序，出现了各种高级语言，例如：BASIC、FORTRAN、PASCAL 等。高级语言易于学习、理解和掌握，用高级语言编程比较简单，大大提高了工作效率。但是，高级语言编制的程序仍不能直接在机器上执行，为了执行该程序也需要把高级语言程序翻译成机器语言程序。这就需要各种翻译程序，例如 BASIC 的解释程序、编译程序等。

随着计算机技术的发展以及计算机的普及和推广，计算机的操作也由手工方式过渡到多道程序成批地在计算机中运行，于是出现了控制计算机中的所有资源、使多道程序能成批自动运行、充分发挥各种资源最大效能并方便用户使用的操作系统。

以上这些都是由机器的设计者提供的，这些使用和管理计算机的软件，统称为系统软件。系统软件包括：①各种语言的汇编或解释、编译程序；②机器的监控管理程序、操作系统、调试程序、故障诊断程序；③程序库。

2. 应用软件

用户用各种语言编制的解决各种问题的软件统称为应用软件。目前随着计算机的普及，各种各样的应用软件竞相推出，例如财务管理软件、银行管理软件、文字处理软件等。这些应用软件有着广阔的市场和美好的发展前景，并且各种各样的应用软件能更好地解决实际问题，使人们解决问题的能力越来越强，使用计算机越来越简单方便，自动化程度越来越高。

总之，硬件建立了计算机的物质基础，而各种软件则扩大了计算机的功能。硬件和软件只有结合起来，才能完成各种功能，才是一个完整的计算机系统。图 1-5 给出了微处理器、微型计算机和微型计算机系统的结构关系。

计算机系统硬件、软件与用户之间的关系如图 1-6 所示，软件可看做是用户与计算机硬件系统的接口，而软件之间又是逐层依赖的。

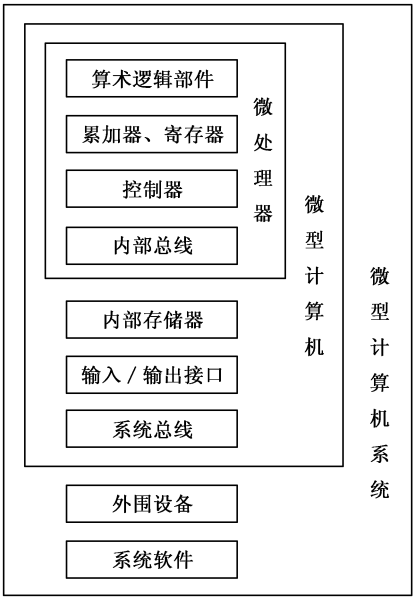


图 1-5　微处理器、微型计算机和微型计算机系统的结构关系

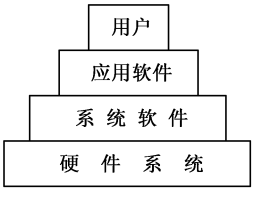


图 1-6　计算机系统硬件、软件与用户之间的关系

1.3.3 微型计算机系统的组成及特点

微型计算机系统与其他计算机系统一样,由硬件和软件两部分组成(见图 1-5)。

1. 微型机硬件结构及特点

在计算机中,把运算器和控制器合在一起称为中央处理机(Center Processing Unit,CPU)。在微型机中,CPU 通常是一个大规模集成电路芯片,也称为微处理器(Microprocessor,μP)。CPU 与内存合起来称为主机。主机、总线、接口电路与外围设备组成了微型机的硬件。

总线是微型机中连接各功能部件并传送信息的一组信号线,分为 3 类,即地址总线(Address Bus,AB)、数据总线(Data Bus,DB)和控制总线(Control Bus,CB)。总线结构是微型机的独特结构,如图 1-7 所示。

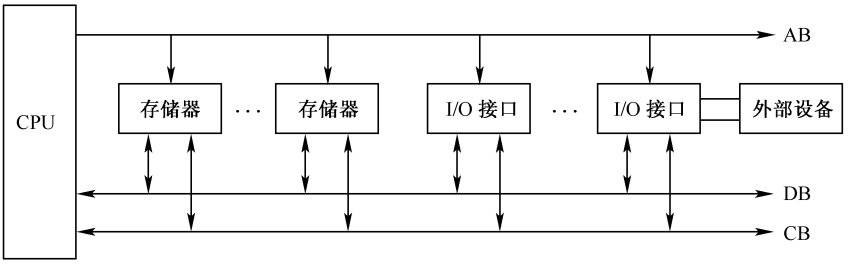


图 1-7 微型机的总线结构

有了总线结构以后,系统中各功能部件之间的相互关系变为各个部件面向总线的单一关系。一个部件或设备只要符合总线标准,就可以连接到采用这种总线标准的系统中,使系统功能能很简便地得到扩展。

数据总线用来传输数据。从结构上看,数据总线是双向的,即数据既可以从 CPU 送到其他部件,也可以从其他部件传送到 CPU。数据总线的位数(也称宽度)是微型机的一个重要指标,它和微处理器的位数相对应,反映计算机数据运算和传送的能力。和其他类型计算机一样,微型机中数据的含义也是广义的。数据总线上传送的数据不一定就是数学意义上的数值,还可以是其他编码、指令代码、状态量,或者是一个控制量。

地址总线专门用来传送地址信息。因地址总是从 CPU 送出去的,所以和数据总线不同,地址总线是单向的。地址总线的位数决定了 CPU 可以直接寻址的内存范围。比如,8 位微型机的地址总线一般是 16 位,因此,最大内存容量为 $2^{16}=64$KB;16 位微型机的地址总线为 20 位,因此,最大内存容量为 $2^{20}=1$MB。

控制总线用来传输控制信号,其中包括 CPU 送往存储器和输入/输出接口电路的控制信号,如读信号、写信号、中断响应信号、DMA 响应信号等;还包括其他部件送到 CPU 的信号,如准备好信号、中断请求信号、总线请求信号等。

2. 微型机硬件系统的组成

目前流行的微型机硬件系统一般由主机和外围设备两部分组成。

主机一般封装在主机箱中,主要包括主机板(CPU、内存槽与内存条、芯片组、总线扩展槽等)、各种接口卡(显卡、声卡、视频卡、调制解调器、网卡等)、软盘驱动器、硬盘驱动器、光盘驱动器、电源系统等。

外围设备包括输入设备和输出设备,常见的有显示器、键盘、鼠标、打印机、扫描仪、音箱等。

3. 微型机的软件系统

由于微型机体积小、价格便宜、性能相对较低,主要用在小规模应用领域,其系统软件和应用软件规模相对较小、功能相对较简单。目前常用的操作系统有 Windows 8、Windows 7、Windows NT、UNIX、Linux 等,常用的软件包有 Office、IE 浏览器,以及 VB、VC ++ 、Java 等语言处理程序和 VFP、Oracle 等数据库系统。

1.3.4 微型计算机系统的主要技术指标

1. 常用的名词术语

1) 位(bit)。位是计算机所能表示的最基本最小的数据单位。它只能有两种状态:"0"和"1",即二进制位。

2) 字(Word)。计算机中作为一个整体参与运算、处理和传送的一串二进制数,是计算机中信息的基本单位。

3) 字长(Word Length)。计算机中每个字所包含的二进制位数称为字长。字长通常等于数据总线的位数和通用寄存器的位数。8 位微处理器的字长为 8 位,32 位微处理器的字长为 32 位。

4) 字节(Byte)。8 位二进制数称为一个字节。字节的长度是固定的,但不同计算机的字长是不同的。8 位机的字长就等于 1 字节,而 16 位机的字长等于 2 字节。

5) 指令。指挥计算机进行基本操作的命令,是计算机能够识别的一组二进制编码。通常一条指令由操作码和操作数两部分组成,前者指出应该进行什么操作,后者指出参与操作的数据的来源。

6) 指令系统。计算机所能执行的全部指令的集合称为计算机的指令系统。

7) 程序。完成某一任务的指令(或语句)的有序集合称为程序。也就是根据执行过程,将对应的操作指令按一定顺序排列在一起。

2. 主要技术指标

一台计算机性能的优劣,是由它的系统结构、指令系统、硬件组成、外围设备以及软件配备齐全与否等因素决定的。只有综合各项指标,才能正确评价与衡量计算机性能的高低。下面介绍几项主要性能指标。

1) 字长。计算机的字长是计算机内部一次可以处理的二进制代码的位数。它决定着计算机的通用寄存器和加法器(运算器一次能处理的二进制码位数)、数据总线等部件的位数,因此直接影响硬件的成本。

字长是表征计算机运算精度的主要参数。字长越长,一个字所能表示的数据精度就越高。在完成同样精度的运算时,字长较长的计算机比字长较短的计算机速度快。

2) 内存储器容量。内存储器容量是衡量它存储二进制信息量大小的一个重要指标。它可以以内存的字长为单位计算,如 32768 ×16 表示有 32768 个存储单元,每个单元字长 16 位;微型计算机中的内存一般以字节(B)为单位($1B = 8bit$,$1KB = 2^{10}B = 1024B$,$1MB = 2^{10}KB = 1024KB$,$1GB = 2^{10}MB = 1024MB$),如 32MB 表示有 32M 个字节存储单元。内存容量常常有一个变化范围,同一型号微型机,实际配备的内存容量大小也有一个变化范围,这主要是由其用途、成本及价格等因素来决定的。

3) 运算速度。指令执行时间的长短反映计算机运算速度的快慢。因为执行不同指令所需的时间不同,这就产生了如何评估计算机速度问题。常用方法有以下 3 种:

① 根据各类指令在计算过程中出现的频度不同,将它的执行时间乘以不同系数,求得平均值。因此这里所说的运算速度为平均速度,常以百万条指令/秒(MIPS)作单位。

② 以执行时间最短的指令为标准来估算运算速度。

③ 用 CPU 的时钟频率(主频)来表征(微型机中常用主频来衡量运算速度)。

4) 外围设备配备。在现代计算机系统中,外围设备占重要地位。一台计算机配备多少外围设备,或者说配有多少外围设备的接口电路,对程序的研制和系统应用都有重大影响。在微型计算机系统中,外存储器容量、打印机型号、显示屏幕的分辨率以及多媒体设备、网络连接等,都是外围设备选用中要考虑的问题。

微型计算机系统的性能是由它的各个组成部分综合决定的,其中最重要的因素是主机板和CPU,包括主机板的结构类型、系统芯片组类型、系统总线的类型、位数及传输速率、CPU 的性能等。其他因素包括外设配置(如显示器的类型与分辨率)、软件配置等。因此在微型计算机选型时,应重点考虑的是 CPU 型号(字长、主频)、主板和芯片组、内存容量及其性能标准、扩展槽总线标准、高速缓存容量、显示器和显卡及其他外设接口的配备等。

1.4 习题

1. 计算机中为什么都采用二进制数而不采用十进制数?

2. 写出下列用原码或补码表示的机器数的真值。

(1) 01101101　　(2) 10001101　　(3) 01011001　　(4) 11001110

3. 填空。

(1) $(1234)_{10} = (\quad)_2 = (\quad)_{16}$

(2) $(34.6875)_{10} = (\quad)_2 = (\quad)_{16}$

(3) $(271.33)_{10} = (\quad)_2 = (\quad)_{16}$

(4) $(101011001001)_2 = (\quad)_{10} = (\quad)_{16}$

(5) $(1AB.E)_{16} = (\quad)_{10} = (\quad)_2$

(6) $(10101010.0111)_2 = (\quad)_{10} = (\quad)_{16}$

4. 已知 X = 36, Y = −136, Z = −1250,请写出 X、Y、Z 的 16 位原码、反码和补码。

5. 已知 $[X]_{补} = 01010101B$, $[Y]_{补} = 10101010B$, $[Z]_{补} = 1000111111111111B$,求 X、Y、Z 及 X + Y、Y − Z 的十进制值为多少?

6. 用 8 位补码进行下列运算,并说明运算结果的进位和溢出。

(1) 33 + 114　　(2) 33 − 114　　(3) (−33) + 114　　(4) (−33) − 114

7. 将下列十进制数表示为 8421BCD 码。

(1) 8609　　(2) 5254　　(3) 2730　　(4) 2998

8. 将下列 8421BCD 码表示为十进制数和二进制数。

(1) 01111001　　(2) 001010000101　(3) 011000000111　　(4)010110010000

9. 将下列数值或字符串表示为相应的 ASCII 码。

(1) 51　　(2) 7FH　　(3) C6H　　(4) Computer　　(5)how are you?

10. 定点数和浮点数表示方法各有什么特点?

11. 微处理器、微型计算机和微型计算机系统三者之间有什么不同?

12. 微型计算机由哪几部分组成?各部分的功能是什么?

13. CPU 在内部结构上由哪几部分组成？CPU 应具备什么功能？

14. 简述计算机执行指令和执行程序的过程。以书中的例子为例，说明在此 3 条指令执行中，哪些信号属于数据流，哪些信号属于控制流？

15. 微型计算机外部为什么采用三总线结构？

16. 数据总线和地址总线在结构和作用上有什么不同？

17. 如果某几种 CPU 的地址总线分别有 8、16、20、32 条，它们各自能寻址的存储器的容量是多少？

18. 什么是硬件？什么是软件？硬件和软件的关系如何？

19. 说明位、字节、字长的概念及它们之间的关系。

20. 计算机的发展趋势有哪些？如何看待冯·诺依曼计算机体系结构理论？

21. 说出几种型号的 CPU，并说明它们各有什么特点。

22. 说出目前流行的几种主机板的类型以及它们的性能特点。

23. 常用的外围设备有哪些？它们各有什么特点？如何衡量它们的性能？

24. 计算机软件包括哪些种类？它们有什么不同？

25. 你知道或用过哪些系统软件？它们各有什么功能特点？

26. 说出计算机的一种主要性能指标。

第 2 章　8086 微处理器及其系统

1978 年,Intel 公司推出 16 位微处理器 8086。8086CPU 具有 16 位数据总线和 20 位地址总线,数据总线与地址总线分时复用,寻址范围为 1MB。8086 的一个突出特点是多重处理能力,用 8086CPU 与 8087 数学协处理器以及 8089I/O 处理器组成的多处理器系统,可大大提高其数据处理和输入/输出能力。另外,与 8086 配套的各种外围接口芯片非常丰富,方便用户开发各种系统。8086 是构成 IBM PC 的核心。几乎在推出 8086 的同时,Intel 公司还推出了准 16 位微处理器 8088。8088 的内部结构和指令功能与 8086 完全相同,只是为了和原有的 8 位微处理器外围芯片兼容,其外部数据总线是 8 位的。

2.1　8086 微处理器简介

8086 微处理器由执行部件(EU)和总线接口部件(BIU)组成。其中,EU 部分主要完成计算机内的算术逻辑运算;BIU 部分主要完成 CPU 与存储器、I/O 设备之间传送数据、地址、状态和控制信息。

2.1.1　8086 的编程结构

8086/8088 CPU 的内部结构如图 2-1 所示,可以看出 8086CPU 由执行部件(EU)和总线接口部件(BIU)两部分组成。

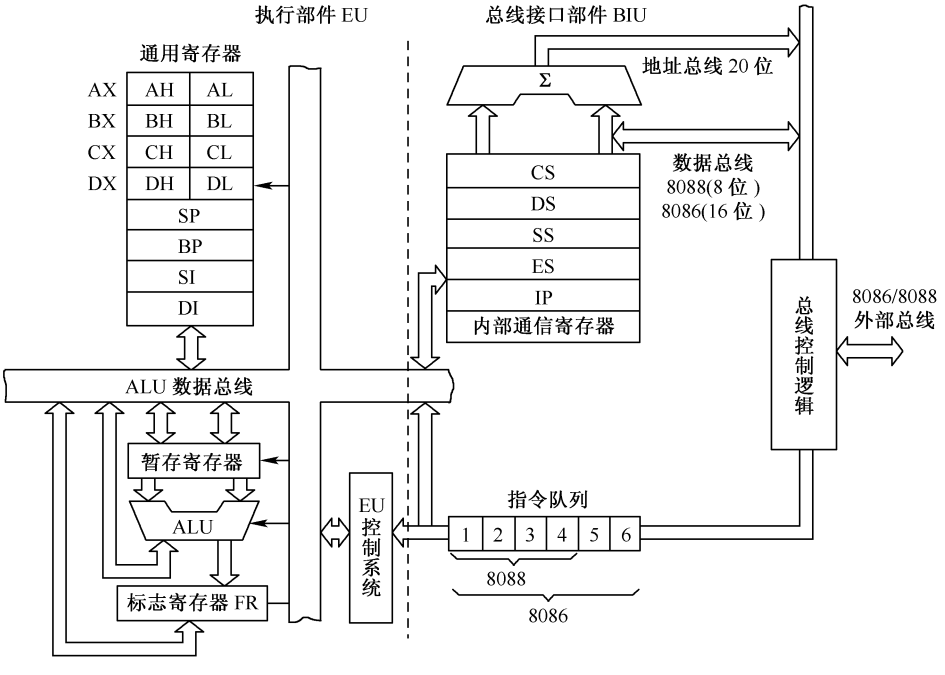


图 2-1　8086/8088 CPU 内部结构

1. 执行部件(EU)

执行部件由内部寄存器组、算术逻辑运算单元(ALU)与标志寄存器(FR)、内部控制逻辑电路3部分组成。

1) 内部寄存器组。8086共有8个16位的内部寄存器,分为两组。

① 通用数据寄存器。4个通用数据寄存器AX、BX、CX、DX均可作为16位寄存器,也可用作8位寄存器。用作8位寄存器时分别记为AH、AL、BH、BL、CH、CL、DH、DL。

- AX(AH、AL):累加器。有些指令约定以AX(或AL)为源或目的寄存器。实际上大多数情况下,8086的所有通用寄存器均可充当累加器。
- BX(BH、BL):基址寄存器。BX可用作间接寻址的地址寄存器和基地址寄存器,BH、BL可用作8位通用数据寄存器。
- CX(CH、CL):计数寄存器。CX在循环和串操作中充当计数器,指令执行后CX内容自动修改,因此称为计数寄存器。
- DX(DH、DL):数据寄存器。除用作通用寄存器外,在I/O指令中可用作端口地址寄存器,乘除指令中用作辅助累加器。

② 指针和变址寄存器。

- BP(Basic Pointer Register):基址指针寄存器。
- SP(Stack Pointer Register):堆栈指针寄存器。
- SI(Source Index Register):源变址寄存器。
- DI(Destination Index Register):目的变址寄存器。

BP、SP称为指针寄存器,用来指示相对于段起始地址的偏移量,一般用于堆栈段。SI、DI称为变址寄存器,可用作间接寻址、变址寻址和基址变址寻址的寄存器。SI一般用于数据段,DI一般用于数据段或附加段。

2) 算术逻辑单元(ALU)及标志寄存器(FR)。算术逻辑单元完成16位或8位算术逻辑运算。运算结果送上ALU内部数据总线,同时在标志寄存器中建立相应的标志。标志寄存器是一个16位寄存器,使用其中的9位作为条件标志和控制标志。条件标志(6位)根据算术逻辑运算结果由硬件自动设定,它们反映运算结果的某些特征或状态,可作为后继操作(如条件转移)的判断依据。控制标志(3位)由用户通过指令来设定,它们可控制机器或程序的某些运行过程。标志寄存器的内容如下:

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

① 条件标志(状态标志)。共6位,用于寄存程序运行的状态信息,这些标志往往用作后续指令判断的依据。

- CF(Carry Flag):进位标志,反映在运算结果的最高位有无进位或借位。如果运算结果的最高位产生了进位(加法)或借位(减法),则CF=1,否则CF=0。
- PF(Parity Flag):奇偶标志,反映运算结果中"1"的个数的奇偶性,主要用于判断数据传送过程中是否出错。若结果的低8位中有偶数个"1",则PF=1,否则PF=0。
- AF(Auxiliary Carry Flag):辅助进位标志,又称半进位标志。加减运算时,若D_3向D_4产生了进位或借位,则AF=1,否则AF=0。在BCD码运算时,该标志用于十进制调整。
- ZF(Zero Flag):零标志,反映运算结果是否为0。若结果为零,则ZF=1,否则ZF=0。

- SF(Sign Flag):符号标志,反映运算结果最高位即符号位的状态。如果运算结果的最高位为1,则SF=1(对带符号数即为负数),否则SF=0(对带符号数即为正数)。
- OF(Overflow Flag):溢出标志,反映运算结果是否超出了带符号数的表数范围。若超出了机器的表数的范围,即为产生溢出,则OF=1,否则OF=0。对于字节运算,结果范围应是-128 ~ +127,字运算的结果范围是-32768 ~ +32767。机器实际处理时判断是否溢出的方法是根据最高位的进位(CF)与次高位的进位是否相同来确定,若两者不相同,则OF=1(表示有溢出),否则OF=0(表示无溢出)。

为简化书写以字节操作为例:10001000 + 10001100。

二进制运算	十六进制运算	带符号数运算	无符号数运算	
1000 1000	88H	-120	136	
+ 1000 1100	+ 8CH	+ -116	+ 140	
1 0001 0100	1 14H	-236	276	=256+20

方框中的1表示结果超出字节部分。运算结果标志位如下:CF=1,PF=1,AF=1,ZF=0,SF=0,OF=1。因为D_7位进位为1,D_6位进位为0,所以产生溢出,OF=1。由运算结果应为-236也可以看出,显然已经超出了单字节带符号数的表示范围。因此有溢出时,运算结果对带符号数来说是错误的,程序员应做适当处理。

如果把运算看作是无符号数运算,CF=1表示最高位有进位,进位的二进制真值相当于$2^8=256$,加上本字节结果00010100B即20,显然结果是正确的。因此对于无符号数,进位位是有效结果的一部分,程序员必须在随后的运算中予以正确处理。

② 控制标志(3位)。用于控制机器或程序的某些运行过程。

- DF(Direction Flag):方向标志,用于串处理指令中控制串处理的方向。当DF=1时,每次操作后变址寄存器SI、DI自动减量,因此处理方向是由高地址向低地址方向进行。当DF=0,则SI、DI自动增量,处理方向由低地址向高地址方向进行。该标志由方向控制指令STD或CLD设置或清除。
- IF(Interrupt Flag):中断允许标志,用于控制CPU是否允许响应可屏蔽中断请求。IF=1为允许响应可屏蔽中断请求,IF=0则禁止响应可屏蔽中断请求。该标志可由中断控制指令STI或CLI设置或清除。
- TF(Trap Flag):陷阱标志,用于单步操作。TF=1时,每执行一条用户程序指令后自动产生陷阱,进入系统的单步中断处理程序。TF=0时,用户程序会连续不断地执行,不会产生单步中断。

3) 内部控制逻辑电路。它是EU的内部控制系统,主要功能为从指令队列缓冲器中取出指令,对指令进行译码,并产生各种控制信号,控制各部件的协同工作以完成指令的执行过程。控制信号如图2-1中单线所示。

2. 总线接口部件(BIU)

总线接口部件负责CPU与存储器、I/O设备之间传送数据、地址、状态及控制信息,它由段寄存器(CS、DS、SS、ES)、指令指针寄存器(IP)、地址加法器、内部暂存器(对用户透明,用户无权访问)、指令队列缓冲器及I/O控制逻辑等部分组成,分别介绍如下。

1) 段地址寄存器(CS、DS、SS、ES)。8086CPU内部数据结构是16位的,即所有的寄存器都是16位的,而外部寻址空间为1MB,即需要20位地址线。为了能用内部寄存器中的16位地址来寻址1MB空间,8086将1MB空间以16字节为一个内存节(Paragraph),共分成64K个节,如

图 2-2a 所示。节的起始地址分别为 00000H、00010H、00020H、…、FFFF0H，称为段基址。节的起始地址的后 4 位二进制数为全 0，故只需记住高 16 位地址即可，分别为 0000H、0001H、…、FFFFH，称为节的段地址。用于存放段地址的寄存器称为段寄存器，根据其主要用途，分为代码段寄存器 CS、数据段寄存器 DS、堆栈段寄存器 SS、附加段寄存器 ES。

- 代码段寄存器 CS：代码段是存放程序代码的存储区域，代码段寄存器用来存放代码段存储区域的起始地址。
- 数据段寄存器 DS：数据段是存放程序中所使用的数据的存储区域，数据段寄存器用来存放程序的数据存储区的起始地址。
- 堆栈段寄存器 SS：堆栈是按照后进先出原则组织的一段特殊存储区域，主要用于子程序调用时断点和返回地址的保存和恢复，也可用于数据的传送。堆栈段寄存器用来存放堆栈存储区的起始地址。由堆栈段寄存器 SS 与堆栈指针寄存器 SP 来确定当前堆栈指令的操作地址。
- 附加段寄存器 ES：附加段是为某些字符串操作指令存放目的操作数而设置的一个附加的数据段，附加段寄存器用来存放该附加数据段存储区域的起始地址。

对内存寻址需要找到某一个单元的实际地址（称为物理地址），它是一个 20 位的地址。物理地址的获得方法是：将段寄存器的内容左移 4 位（即 ×16），与逻辑地址（又称偏移地址或有效地址，即对段首的偏移量）相加，得到 20 位物理地址（如图 2-2b 所示）。根据寻址方式的不同，偏移地址可以来自程序计数器（IP）或其他寄存器。

程序运行期间，段寄存器内容很少变化。段地址不变，偏移地址从 0000H ~ FFFFH 变化时，对应 64KB 的空间，所以一个程序段空间最大可达 64KB。这 64KB 的空间，根据段寄存器内容的变化，可放在 1MB 空间中任意节起始位置上，从此开始的 64KB 内存空间称为一个内存段（简称为段）。各段可以互不重叠（如图 2-2c 所示），也可将某几个段安排在同一个 64KB 的空间上（如图 2-2d 所示），或使某些段重迭部分存储空间，如使代码段从 20000H 单元开始，数据段从 21000H 单元开始。

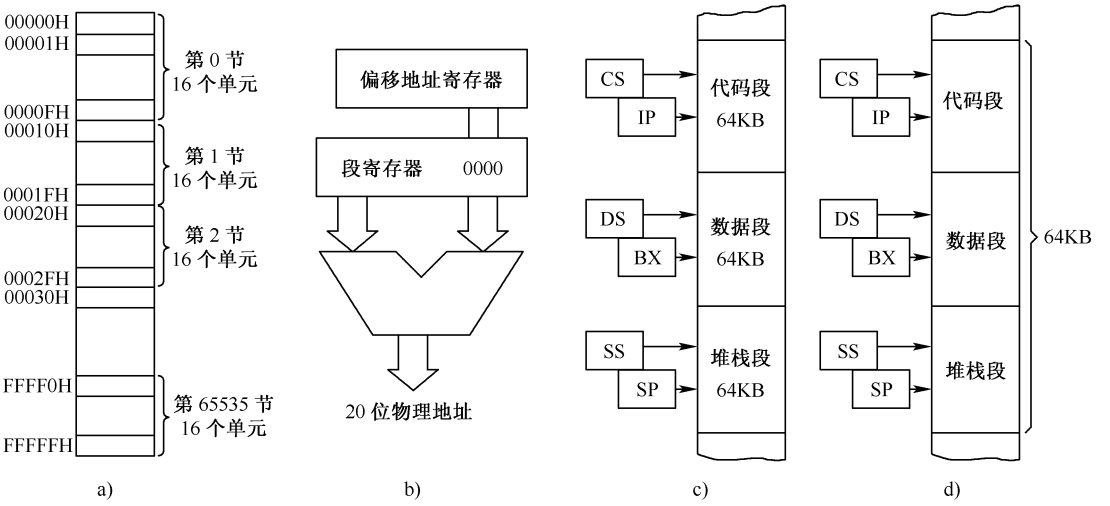


图 2-2　存储器的分段、物理地址形成及存储地址分配

a）存储器的分段　b）物理地址分配　c）存储地址分配

采用段地址的好处在于：①解决了 16 位寄存器访问大于 64KB 内存空间问题；②可以实现程序重定位，即一个不到 64KB 大小的程序可通过改变段寄存器的内容（一般由操作系统来完

成)放到1MB空间中任意段位置,从而为运行多道程序提供了方便。

2)地址加法器。用于产生20位物理地址。两个加数,一个来自段寄存器并左移4位,另一位来自IP或内部暂存器。内部暂存器的内容根据不同的寻址方式,可以通过内部总线由内部寄存器提供,也可由输入/输出控制电路从存储器中读取。

3)指令指针寄存器(IP)。又称程序计数器,是16位寄存器。IP中存放当前将要执行的指令的有效地址,每取出一条指令IP自动增量,即指向了下一条指令(亦即下次要执行的指令),因此可以说IP总是指向将要执行的指令。

4)指令队列缓冲器。一个与CPU速度相匹配的高速缓冲寄存器。8086缓冲器为6字节,8088缓冲器为4字节。在EU执行指令的同时,BIU可以从内存中取出下一条或下几条指令放到指令缓冲器中,EU执行完一条指令后,可以立即从指令缓冲器中执行下一条指令。因此取指令和执行指令可以并行进行,从而提高了总线的利用率,也提高了处理器总的处理速度。当遇到转移、调用及返回指令时,要清除指令队列缓冲器中的指令,从内存中取新的指令,以适应新的指令执行顺序。

5)输入/输出控制电路(总线控制逻辑)。它是CPU外部三总线(AB、DB、CB)的控制电路,控制CPU与其他部件交换数据、地址、状态及控制信息。

3. 总线接口部件和执行部件的管理

前已述及,由于指令队列缓冲器的存在,实现了执行部件(EU)与总线接口部件(BIU)的并行工作,因而提高了8086系统的效率。但这两部分又不是完全独立工作的,现就这两部分的动作管理简要说明如下:

当8086指令队列中有2字节空闲(8088有1字节空闲)时,总线接口部件就自动将指令从内存中预取到指令队列缓冲器中。

每当EU部件要执行一条指令时,它就从指令队列头部取出指令,后续指令自动向前推进。EU要花几个时钟周期执行指令,指令执行中若需要访问内存或I/O设备,EU就向BIU申请总线周期,若BIU总线空闲,则立即响应,若BIU正在取一条指令,则待取指令操作完成后再响应EU的总线请求。当指令队列已满,EU又没有申请总线时,则总线空闲。

遇到转移、调用及返回指令时,原先预取到指令队列中的指令已不再有用,BIU就自动清除指令队列中已有内容,从转移、调用或返回的新地址开始,重新从内存中预读取指令并填充指令队列。可见,总线接口部件与执行部件既非同步工作方式,也不是完全无关,而是互相配合工作的。

2.1.2 8086的引脚及其功能

8086芯片为40引脚双列直插封装。为解决功能与引脚数之间的矛盾,许多引脚具有多功能。其实现方法一种是总线复用,即在不同时钟周期内,引脚的作用不同,如地址、数据线的分时复用。另一种是按工作模式的不同确定引脚的功能。为便于说明引脚功能,这里先介绍8086总线周期的概念。

1. 8086总线周期

计算机是由一串脉冲控制进行工作的。这一串脉冲称为计算机的时钟,每个脉冲称为一个时钟脉冲或一个T状态。若干个时钟脉冲完成一个基本操作。一种基本操作称为一个总线周期。CPU有若干种典型操作,构成相应的总线周期,如存储器读总线周期、存储器写总线周期、I/O读总线周期、I/O写总线周期等。完成一条指令需要若干个总线周期。8086典型的时钟脉冲与总线周期如图2-3所示。可以看出,一个基本的8086总线周期一般包括T_1、T_2、T_3、T_4共4

个 T 状态。有时根据需要在 T_3 与 T_4 间插入若干个等待状态 T_W，在总线空闲时插入空闲状态 T_I。各个 T 状态的基本功能如下：

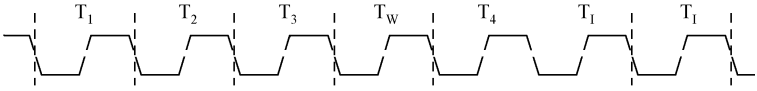


图 2-3　典型的 8086 总线周期时序

- T_1 状态：CPU 向多路复用总线上发送地址信息，指出要寻址的内存单元地址或 I/O 端口地址。这期间 CPU 还要送出地址锁存信号 ALE（正向脉冲），在 ALE 的下降沿将内存单元地址或 I/O 端口地址存入地址锁存器。
- T_2 状态：CPU 从总线上撤销地址，使总线低 16 位呈现高阻状态，为数据传输作准备。总线高 4 位（$A_{19} \sim A_{16}$）输出总线周期的状态信息，用以表示中断允许状态及正在使用的段寄存器名等。
- T_3 状态：$A_{19} \sim A_{16}$ 上状态信息不变，总线低 16 位上出现 CPU 要写出的数据或准备读入的数据。若外设或内存来不及与总线交换数据，以使在 T_4 状态下结束该总线周期，则应通过 CPU 的 READY 信号，在 T_3 前沿（下降沿）之前向 CPU 申请插入等待状态 T_W。在 T_3 及 T_W 的前沿查询 READY 线，查到为高电平则结束等待状态，进入下一状态。否则继续插入等待状态。
- T_4 状态：总线周期结束，若为总线读周期则在 T_4 前沿将数据读入 CPU。
- T_I 总线空闲周期：一个特殊的总线周期，由一个或多个 T_I 状态组成。只有与内存或 I/O 设备交换数据时，CPU 才执行总线周期。在一个总线周期之后，若不立即进入下一个总线周期，则 BIU 不执行任何总线操作，系统总线处于空闲状态，这时在总线上插入 T_I，形成总线空闲周期。

2. 8086 的引脚及功能

本节只介绍在最小模式和最大模式下的通用引脚，如图 2-4 所示，与工作模式有关的引脚在相应工作模式中介绍。

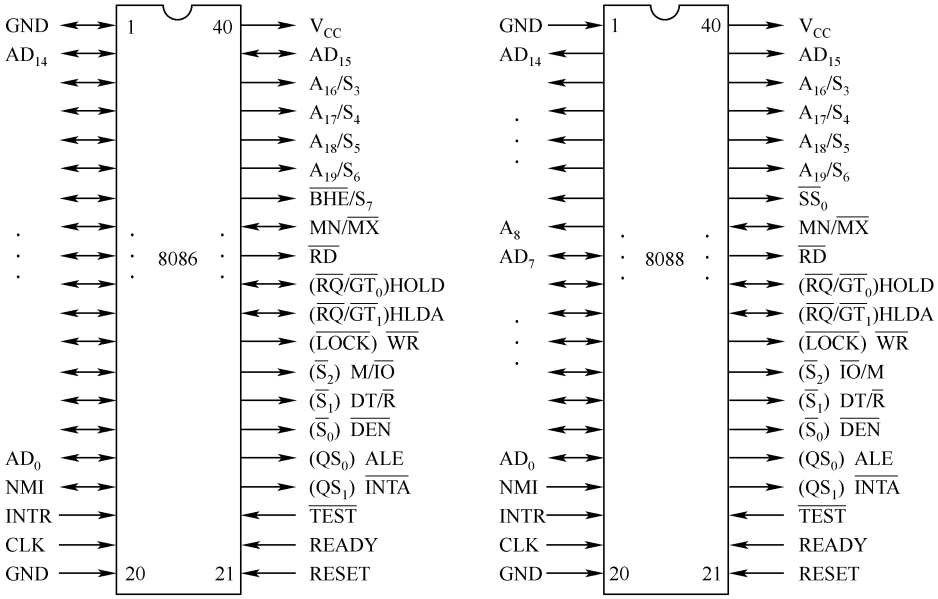


图 2-4　8086/8088 引脚分配图

注：（）中为多 CPU（最大）模式信号

1）GND(地)和 V_{CC}(电源)。V_{CC}引脚接 +5V 电源,GND 引脚接地。

2）$AD_{15} \sim AD_0$(Address Data Bus):地址/数据复用引脚,双向、三态。8086 由于采用地址/数据总线复用方式,才可能在 40 引脚的条件下提供 20 位地址总线、16 位数据总线。$AD_{15} \sim AD_0$ 在总线周期的 T_1 状态输出地址信号。在 T_2T_3 状态,若为总线读周期则先将总线浮空,然后接收从内存或外设读入的数据;若为总线写周期,则总线上为写出的数据。

3）$A_{19}/S_6 \sim A_{16}/S_3$(Address/Status):地址/状态复用引脚,输出、三态。在总线周期的 T_1 状态,$A_{19} \sim A_{16}$输出最高 4 位地址信号;在 $T_2T_3T_WT_4$ 状态,$S_6 \sim S_3$ 输出状态信息,其意义如下:

S_6 为 0 表示 CPU 占用总线。S_5 表明中断允许标志 IF 的设置情况。如 IF = 1,为允许可屏蔽中断,则 $S_5 = 1$;如 IF = 0 则为禁止可屏蔽中断,则 $S_5 = 0$。S_4、S_3 指明正在使用的段寄存器,其组合情况见表 2-1。

表 2-1　S_4、S_3 代码组合的意义

S_4	S_3	意　义
0	0	正在使用 ES
0	1	正在使用 SS
1	0	正在使用 CS
1	1	正在使用 DS

注意,当系统总线处于“总线保持响应”状态,即 CPU 让出总线控制权时,则 $A_{19}/S_6 \sim A_{16}/S_3$、$AD_{15} \sim AD_0$ 等三态总线均为高阻状态或称为浮空状态。

4）$\overline{BHE}/S_7$(Bus High Enable/Status):高 8 位数据线允许/状态复用引脚,输出、三态。在 T_1 状态,$\overline{BHE}/S_7$ 引脚输出$\overline{BHE}$信号,表示高 8 位数据线 $D_{15} \sim D_8$ 上数据是否有效(低电平则数据有效)。在其他 T 状态,$\overline{BHE}/S_7$ 信号输出状态信号 S_7,S_7 实际上未使用。$\overline{BHE}$与 A_0 合起来可向总线上存储器接口传送表 2-2 所示信息。

表 2-2　$\overline{BHE}$与 A_0 信号的意义

$\overline{BHE}$	A_0	操　作	所用的数据线
0	0	从偶地址开始读/写 1 个字	$AD_{15} \sim AD_0$
1	0	从偶地址读/写 1 个字节	$AD_7 \sim AD_0$
0	1	从奇地址开始读/写 1 个字节	$AD_{15} \sim AD_8$
0	1	从奇地址开始读/写 1 个字(第 1 个总线周期)	$AD_{15} \sim AD_8$
1	0	(第 2 个总线周期)	$AD_7 \sim AD_0$

注:1. 从奇地址读写一个字节,$\overline{BHE} = 0$,说明高 8 位数据线上数据有效。

2. 从奇地址读写一个字,需要占用两个总线周期。第 1 个总线周期数据出现在高 8 位数据线上,第 2 个总线周期数据出现在低 8 位数据线上。

5）NMI (Non - Maskable Interrupt):非屏蔽中断请求引脚,输入。该信号为一个由低到高的上升沿有效信号。非屏蔽中断请求不受中断允许标志 IF 的影响。一旦接收到 NMI 信号,CPU 处理完一条指令后,即进入非屏蔽中断的响应和处理。

6）INTR (Interrupt Request):可屏蔽中断请求引脚,输入、高电平有效。CPU 在每条指令的最后一个时钟脉冲对 INTR 信号进行采样,当 INTR 为高电平时,如果 CPU 的中断允许标志 IF = 1,则在当前指令结束后即响应中断请求,进入中断处理。

7）$\overline{RD}$(Read)：读信号，输出、三态、低电平有效。在 CPU 执行读操作时，在 T_2、T_3、T_W 期间有效。到底是读存储器还是读 I/O 端口取决于 M/$\overline{IO}$，若 M/$\overline{IO}$为高则为读存储器，M/$\overline{IO}$为低则为读 I/O 端口。

8）CLK（Clock）：时钟输入引脚。8086 要求时钟占空比为 1/3，即一个周期中 1/3 为高电平、2/3 为低电平。CPU 的所有操作均是在时钟的同步下进行的。

9）RESET（Reset）：复位引脚，输入、高电平有效。8086 要求复位信号至少维持 4 个时钟周期的高电平，以完成 CPU 内部寄存器的复位操作。复位信号一出现，CPU 就立即结束当前的操作，进入复位操作，即将标志寄存器（FR）、IP、DS、SS、ES 清 0，指令队列清空，CS 置为 FFFFH，当复位信号降为低电平后，CPU 从 CS：IP 开始取出指令并执行它，即系统复位后执行的第一条指令的物理地址为 FFFF0H。系统程序一般在该物理地址单元放一条转移指令，转到引导程序的入口。

10）READY（Ready）：准备好引脚，输入、高电平有效。当 CPU 执行总线读写周期访问存储器或 I/O 设备时，如果存储器或 I/O 设备的读写速度较慢，来不及在 T_4 状态结束数据传输，就需要设计一个硬件电路，在 T_3 之前向 CPU 提供一个低电平 READY 信号。CPU 在 T_3 前沿查询 READY 线，当查到为低电平时，则在 T_3 之后插入一个 T_W 等待状态，并在 T_W 前沿继续查询 READY 线，直到 READY 升为高电平，才进入 T_4 状态，完成数据传送过程。

11）$\overline{TEST}$(Test)：测试引脚，输入、低电平有效。$\overline{TEST}$和 WAIT 指令配合使用，执行 WAIT 指令时，CPU 暂停执行程序，进入空转状态等待。当$\overline{TEST}$引脚接收到一个低电平信号时，CPU 就结束等待，继续向下执行指令。$\overline{TEST}$和 WAIT 配合，可以实现 CPU 与外设同步工作。

12）MN/$\overline{MX}$(Minimum/Maximum Mode Control)：最小/最大模式控制引脚，输入。该引脚接 +5V 时，8086 工作在最小模式，该引脚接地，则 8086 工作于最大模式。

2.2 8086 系统的存储器组织及 I/O 组织

8086 系统的存储器结构是由 1MB 的存储空间分成两个 512KB 的物理存储体。一个存储体由偶地址识别，另一个存储体由奇地址识别。本节主要介绍读/写 1 字节或 1 个字时的完成方法，8086 存储器的编址方法，8086 系统的地址栈用区域和系统的 I/O 组织。

2.2.1 8086 系统的存储器组织

1. 8086 存储器的结构

8086 系统中将 1MB 存储空间分成两个 512KB 的物理存储体。一个存储体由偶数地址组成，另一个存储体由奇数地址组成，用 A_0 位来区分两个存储体。对任何一个存储体的访问只需要 19 位地址码（A_{19} ~ A_1）。存储体地址空间分配情况如图 2-5 所示。每个存储体与总线的连接如图 2-6 所示。由图可以看出，用$\overline{BHE}$和 A_0 的组合来选择存储体，其组合关系及操作情况已在表 2-2 中介绍。现就几种读写情况作进一步说明。

1）从偶地址读写 1 字节（$\overline{BHE}$ A_0 = 10）。如图 2-7a 所示，AD_{15} ~ AD_8 上的数据被忽略，字节内容通过 AD_7 ~ AD_0 传送。

2）从奇地址读写 1 字节（$\overline{BHE}$ A_0 = 01）。如图 2-7b 所示，在 AD_{15} ~ AD_8 上传送的数据有效，AD_7 ~ AD_0 上数据被忽略。

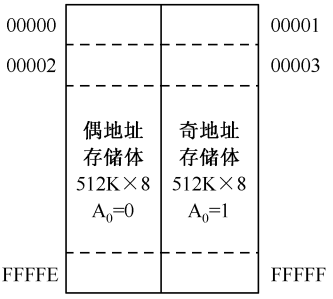


图 2-5　存储体地址空间分配

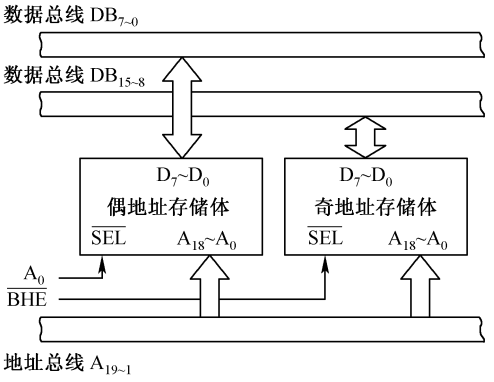


图 2-6　存储体与总线的连接

3）从偶地址开始读写 1 个字（$\overline{BHE}$ A_0 = 00）。如图 2-7c 所示，在 AD_{15} ~ AD_0 上传送的数据同时有效。

以上 3 种情况读写操作都是在一个总线周期中完成的。

4）从奇地址开始读写 1 个字。这种操作要占用两个总线周期，如图 2-7d 所示。第一个总线周期$\overline{BHE}$ A_0 = 01，从奇地址读写低字节，在 AD_{15} ~ AD_8 上传送的数据有效。第二个总线周期 $\overline{BHE}$ A_0 = 10，从偶地址读写高字节，在 AD_7 ~ AD_0 上传送的数据有效。可以看出，从奇地址开始读写一个字，CPU 要花费更多的时间，所以为提高 CPU 的利用率，字变量应尽量分配到偶地址开始的存储单元。

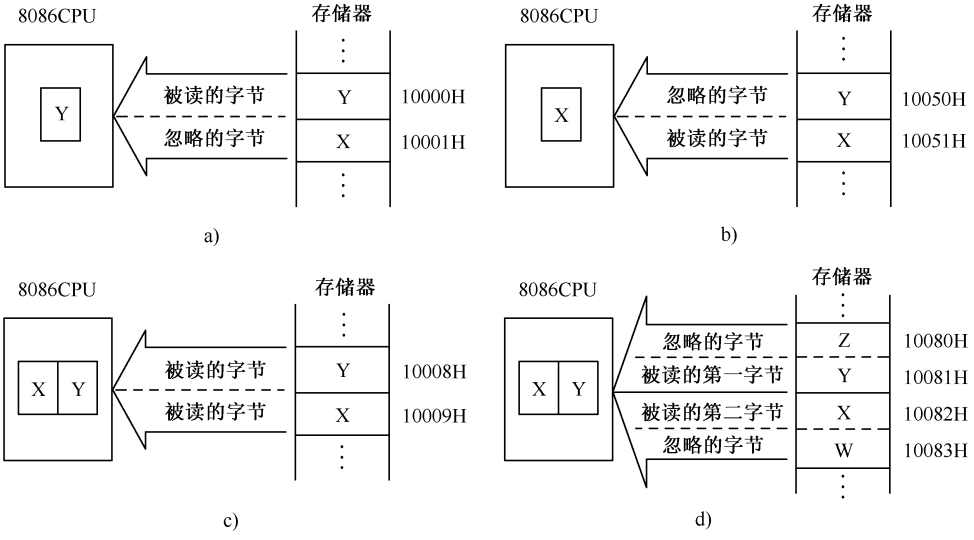


图 2-7　读存储器操作过程示意图

a）从偶地址读 1 字节　b）从奇地址读 1 字节　c）从偶地址读 1 个字　d）从奇地址读 1 个字

2. 8086 系统存储器的地址

1）物理地址的计算公式：物理地址 = 段地址 ×16 + 偏移地址。段地址由段寄存器（CS、DS、SS、ES）提供，偏移地址由 IP、SP、BP、SI、DI 等寄存器或存储器提供，也可通过计算而得。不同指令有不同的组合方式。

2）段地址的引入，为程序在内存中浮动创造了条件，一般用户程序只涉及偏移地址。段地

址在程序装入内存时由操作系统分配,所以一个程序可在内存中任何一个逻辑段(64KB 空间)中运行。

3) 同一物理地址可以由不同的段地址和偏移地址表示。

例如:CS = 2000H　IP = 1000H　物理地址 = 21000H

　　CS = 2100H　IP = 0000H　物理地址 = 21000H

3. 8086 系统内存地址的一些专用区域

0000H ~ 003FFH 1KB 空间用于存放中断向量表,可存放 256 个中断服务程序的入口地址,每个地址占 4 字节。

B0000H ~ B0FFFH 4KB 为单色显示器显示缓冲区,存放屏幕当前显示字符的 ASCII 码。

B8000H ~ BBFFFH 16KB 彩色显示器显示缓冲区,存放当前屏幕像素代码。

FFFF0H 启动地址,一般用来存放一条无条件转移指令,转到系统初始化程序。

2.2.2　8086 系统的 I/O 组织

8086 系统有专用的输入(IN)、输出(OUT)指令,用于外设端口(即外设接口中的内部寄存器)的寻址。I/O 端口与内存分别独立编址。I/O 端口使用 16 位地址 A_{15} ~ A_0,I/O 端口地址范围为 0000H ~ FFFFH,可寻址空间为 64KB。在以 8086 为 CPU 的 PC/XT 微机中,只使用了 10 位有效端口地址 A_9 ~ A_0,共 1KB 空间。其中,A_9 用于指明外设端口是否在系统板上。A_9 = 0 是系统板上的 512 个端口,A_9 = 1 是 I/O 通道上的 512 个端口。

PC/XT 微机系统已占用的端口地址见表 2-3,其余的端口地址用户可以使用。

表 2-3　PC/XT 系统占用的端口地址

地址号		用途
系统板	000H ~ 00FH	DMA 控制器 8237A 占用
	020H ~ 02FH	中断控制器 8259A 占用
	040H ~ 043H	定时器/计数器 8253A 占用
	060H ~ 063H	并行外围接口芯片 8255A 占用
	0A0H ~ 0AFH	NMI 屏蔽寄存器
	0C0H ~ 1FFH	保留
I/O 通道	2F8H ~ 2FFH	异步通信端口(第二个)
	300H ~ 31FH	试验板
	378H ~ 37FH	并行打印机接口
	3B0H ~ 3BFH	单色显示器/打印机适配器
	3D0H ~ 3DFH	彩色显示器/图形打印机适配器
	3F0H ~ 3F7H	软盘适配器
	3F8H ~ 3FFH	异步通信端口

2.3　8086 系统的工作模式

8086 系统的工作模式有两种:最小模式和最大模式。最小模式的显著特点是系统中只有一个微处理器——8086/8088,所有总线的控制信号都由它发出。最大模式的系统中有两个或两个以上的微处理器,以 8086/8088 为主处理器,其他处理器为协处理器,总线控制信号由主/协处理器共同发出。

2.3.1　最小模式和最大模式的概念

为了适应各种应用场合的要求,8086/8088CPU 在设计中提供了两种工作模式,即最小模式

和最大模式。实际机器中究竟工作在哪一种模式,根据需要由硬件连接决定。

1. 最小模式

如果系统中只有一个微处理器 8086(或 8088),所有总线控制信号由它产生,则系统中总线控制逻辑信号可减少到最小,因此称这种系统为最小模式系统。

2. 最大模式

如果系统中包括两个以上处理器,其中一个为 8086/8088 作为主处理器,其他处理器作为协处理器,这样的系统称为最大模式系统。一般多用于复杂的大中型系统。与 8086 协同工作的协处理器有 8087 和 8089 两种,分别为数学协处理器和输入/输出协处理器。

8087 协处理器主要用于数值运算。它由硬件实现高精度的数值运算,因此处理速度快,不仅可进行整数、浮点数运算,还可处理三角函数等多种超越函数。

8089 是输入/输出协处理器,它有一套专门用于输入/输出操作的指令系统,可直接为I/O设备服务。

配置协处理器的系统,主处理器不用处理费时的复杂运算和 I/O 操作,因此可大大提高主处理器的运行效率。

2.3.2 最小模式系统

当系统只有一个微处理器 8086 时,将 MN/$\overline{MX}$引脚接向 +5V,构成最小模式系统,其原理如图 2-8 所示。

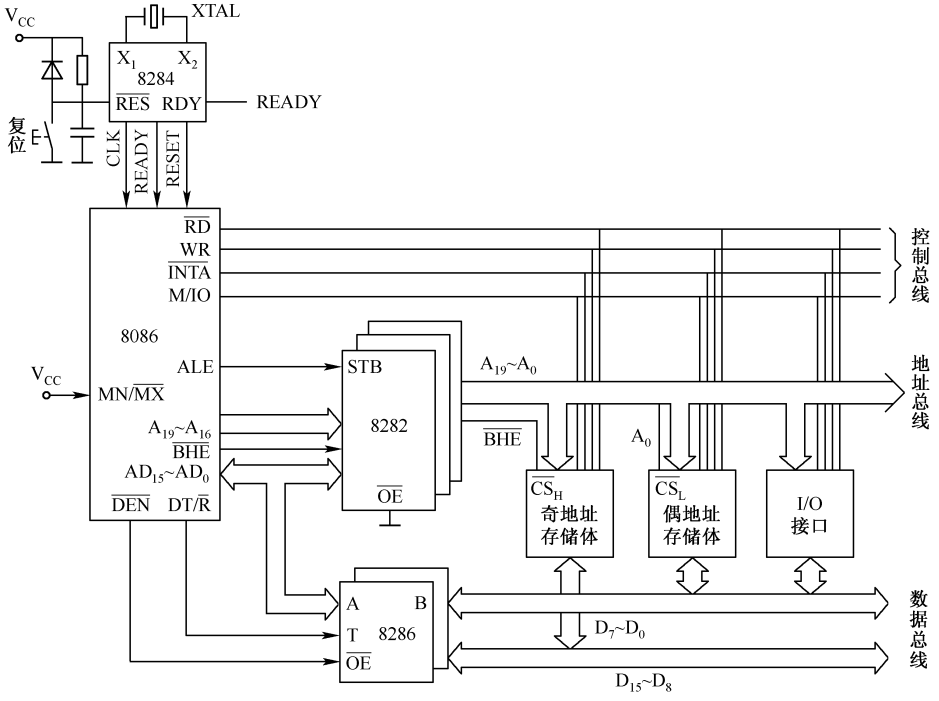


图 2-8 8086 单 CPU 模式(最小模式)系统

1. 最小模式系统的典型配置

一片 8284A 时钟发生器产生系统所需要的时钟信号 CLK,同时对外部 READY 信号和系统复位信号 RESET 进行同步,其输出送向 8086 相应引脚。三片 8282(或 74LS373)地址锁存器用

于 20 位地址和$\overline{BHE}$信号锁存,使得整个总线读写周期期间地址信号始终有效,以支持 8086CPU 地址/数据总线分时复用的工作方式。两片 8286 总线驱动器(又称总线收发器或总线驱动器),当系统所连存储器和外设较多时,为了提高数据总线的驱动能力,可以接入 8286 芯片。

2. 8284A 时钟发生器与 8086 的连接

8284A 时钟发生器与 8086 的连接如图 2-9 所示。外接晶振和电容接向 X1、X2 输入端。F/C 频率/晶振输入选择端接地,由 8284 内部振荡器产生自激振荡,在 CLK 端输出频率为1/3 晶振频率、占空比为 1/3 的时钟脉冲。上电复位及按钮复位信号送 8284 $\overline{RES}$输入端,经 8284 同步后送 8086 RESET 端。来自内存或外设的等待请求信号送 RDY1,经 8284 同步后送 8086 READY 输入端。PCLK 引脚输出占空比为 1/2、频率为 1/2 CLK 的时钟脉冲,用作外部时钟,8284 的其他不使用端接地。

3. 地址锁存器 8282 与 8086 的连接

8086 地址总线与数据总线是分时复用的,高 8 位数据有效信号也是复用信号。在 T_1 状态,总线上输出 20 位地址信号及$\overline{BHE}$信号,而在 $T_2 \sim T_4$ 状态,总线用于数据传送,$\overline{BHE}$信号也失效。为了正确地交换数据,地址信号及$\overline{BHE}$信号在 $T_2 \sim T_4$ 期间必须保持,所以需要设一组地址锁存器(3 片 8282),用于锁存地址及$\overline{BHE}$信号(如图 2-10 所示)。

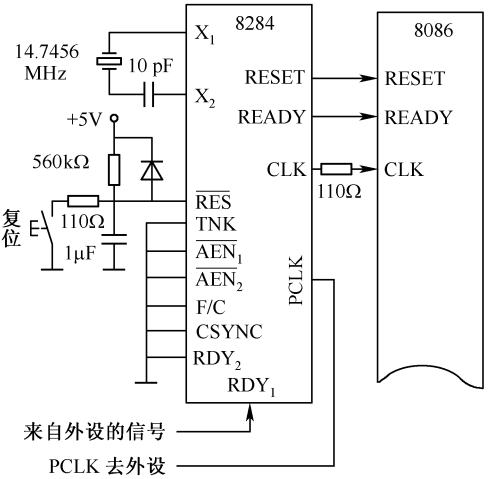


图 2-9　8284 时钟发生器与 8086 的连接图

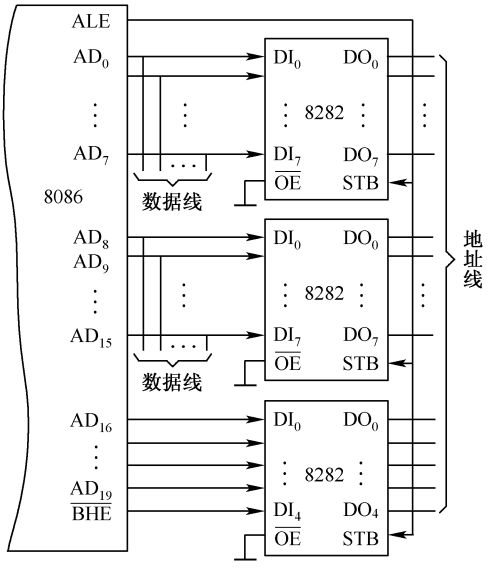


图 2-10　8282 锁存器与 8086 的连接

在 T_1 状态,CPU 送出地址锁存允许信号 ALE,将 ALE 接向 8282 的选通输入端 STB。当 ALE =1 时,8282 输出跟随输入变化,用 ALE 的下降沿将总线上已经稳定的地址信号锁入 8282。由于地址等信号要一直保持有效,所以 8282 的输出允许端$\overline{OE}$直接接地。

4. 总线驱动器 8286 与 8086 的连接

8286 与 8086 的连接如图 2-11b 所示。8286 为双向总线驱动器,其内部结构如图 2-11a 所示,$\overline{OE}$为输出使能端,当$\overline{OE}$ =1 时,控制门关,A_i 与 B_i 断开,且均呈现高阻状态。当$\overline{OE}$ =0 时,控制门开,8286 接通,数据传送方向受 T 端控制。当 T =1 时,数据从 A_i 传向 B_i,当 T =0 时,数据

从 B_i 传向 A_i。

8286 与 8086 连接时，两片 8286 的 $A_0 \sim A_7$ 分别接向 8086 的 $AD_{15} \sim AD_8$ 和 $AD_7 \sim AD_0$。$\overline{OE}$ 端接向 8086 的数据使能端$\overline{DEN}$。T 端接向 8086 的数据发送/读入控制端 DT/$\overline{R}$。当$\overline{DEN}$有效时，若 DT/$\overline{R}$ = 1，数据从 A 传向 B，即 8086 向外部发送数据；若 DT/$\overline{R}$ = 0 时，数据从 B 传向 A，即 8086 接收外部传来的数据。

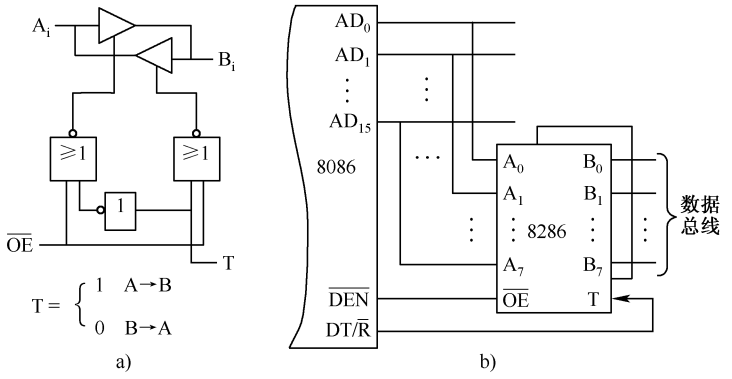


图 2-11　8286 总线驱动器与 8086 的连接

5. 其他控制信号

现在介绍 8086 引脚中与最小模式有关的控制信号（见图 2-4）。

1）M/$\overline{IO}$（Memory/Input & Output）及$\overline{WR}$（Write）信号，输出、三态。这两个信号与$\overline{RD}$信号组合起来决定系统中数据传输的方向。M/$\overline{IO}$、$\overline{RD}$及$\overline{WR}$信号的组合方式及对应功能见表 2-4。

表 2-4　M/$\overline{IO}$、$\overline{RD}$及$\overline{WR}$信号的组合方式及对应功能

M/$\overline{IO}$	$\overline{RD}$	$\overline{WR}$	功　能
0	0	1	I/O 读
0	1	0	I/O 写
1	0	1	存储器读
1	1	0	存储器写
X	0	0	无效组合
X	1	1	非读/写状态

2）HOLD（Hold Request）：总线保持请求信号，输入、高电平有效。当系统中其他部件需要占用总线时，通过该引脚向 CPU 发出申请，如 DMA 控制器就是通过这种方式申请总线的。

3）HLDA（Hold Achknowledge）：总线请求响应信号，输出、高电平有效。CPU 在每个时钟周期都要检测 HOLD 引脚，当检测到 HOLD 信号，并且 CPU 允许其他部件占用总线，则在当前总线周期的 T_4 状态从 HLDA 引脚发出总线请求响应信号，同时 CPU 的所有三态总线进入浮空状态。总线申请部件接到 HLDA 有效信号后即可接管总线进行操作，直到操作完成、撤销 HOLD 信号，CPU 才重新接管总线。

4）$\overline{INTA}$（Interrupt Acknowledge）：中断响应信号，输出、三态、低电平有效。外设申请可屏蔽中断，CPU 响应后，在两个连续的中断响应周期发出两个$\overline{INTA}$负脉冲信号。第一个$\overline{INTA}$通知外设，中断请求已被允许（主要用于中断优先权排队），第二个$\overline{INTA}$信号通知中断请求被响应的外

设将其中断类型码送到数据总线上。

5）ALE（Address Latch Enable）：地址锁存允许信号，输出、高电平有效。该信号用于向地址锁存器 8282 提供地址锁存信号。在任何一个总线周期的 T_1 状态，ALE 输出一个正名称，其高电平表示当前在地址/数据复用总线上输出的是地址信息，用其下降沿将地址信息锁存于地址锁存器。

6）$\overline{DEN}$（Data Enable）：数据允许信号，输出、三态、低电平有效。该信号为总线收发器 8286 提供一个控制信号，表示 CPU 当前准备发送或接收一个数据，8286 将它作为输出允许信号$\overline{OE}$。

7）DT/$\overline{R}$（Data Transmit/Receive）：数据收发控制信号，输出、三态。该信号用来控制总线收发器 8286 的数据传送方向。当它为高电平时表示为数据发送，为低电平时表示为数据接收。

2.3.3 最大模式系统

将 MN/$\overline{MX}$引脚接地就构成了 8086CPU 的最大工作模式，下面介绍最大模式下 8086 的有关引脚信号、8288 总线控制器、最大模式的系统配置。

1. 最大模式下的有关引脚信号

1）QS_0、QS_1（Instruction Queue Status）：指令队列状态信号，输出。这两个信号合起来提供前一个时钟状态（指总线周期的前一个状态）中指针队列的情况，以便于外部对 8086 内部指令队列的动作进行跟踪。QS_0、QS_1 组合的意义见表 2-5。

表 2-5 QS_0、QS_1 代码组合的意义

QS_0	QS_1	意　义
0	0	无操作
0	1	从指令队列的第 1 个字节取走代码
1	0	队列为空
1	1	除第 1 个字节外，还取走了后续字节中的代码

2）$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$（Bus Cycle Status）：总线周期状态信号，输出。这些信号组合起来，可以指出当前总线周期中数据传输过程的类型。最大模式下，总线控制器 8288 就是利用这些状态产生对存储器和 I/O 接口的控制信号。其组合意义见表 2-6。

表 2-6 $\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$代码组合的意义

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	对 应 操 作	8288 发出的控制命令
0	0	0	发中断响应信号	$\overline{INTA}$
0	0	1	读 I/O 端口	$\overline{IORC}$
0	1	0	写 I/O 端口	$\overline{IOWC}$　$\overline{AIOWC}$
0	1	1	暂停	
1	0	0	取指令	$\overline{MRDC}$
1	0	1	读内存	$\overline{MRDC}$
1	1	0	写内存	$\overline{MWTC}$　$\overline{AMWC}$
1	1	1	无源状态	

对$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$来讲，在前一个总线周期的 T_4 状态和本总线周期的 T_1、T_2 中，至少有一个为低电平，每种情况都对应一种总线操作过程，称为有源状态。在 T_3 和 T_W 状态，且 READY 为高电平时，$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$均为高电平，它表示一个总线周期即将结束，新的总线周期还未开始，这种状态称为无源状态。在 T_4 状态，$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$中任何一个信号由 1 变为 0，则说明新的总线周期开始。

3）$\overline{\text{LOCK}}$(Lock)：总线封锁信号，输出、低电平有效。当$\overline{\text{LOCK}}$为低电平时，系统中其他部件不得占用总线。$\overline{\text{LOCK}}$信号由指令前缀 LOCK 产生。一旦使用该前缀，在执行其后的指令期间总线被封锁，其他部件不能占用总线而打断本指令的执行。本指令执行完，$\overline{\text{LOCK}}$信号失效。在中断响应过程中 CPU 会自动产生$\overline{\text{LOCK}}$信号，以防止其他总线部件占用总线而打断中断响应过程。

4）$\overline{\text{RQ}}/\overline{\text{GT}}_1$、$\overline{\text{RQ}}/\overline{\text{GT}}_0$(Request/Gtant)：总线请求/允许信号，双向。这两个引脚可供 CPU 以外的两个处理器分别通过一条$\overline{\text{RQ}}$申请总线，主 CPU 响应后，在同一条线上输出总线允许信号($\overline{\text{GT}}$)。可见，申请与允许信号在同一引脚上是分时双向传输的，$\overline{\text{RQ}}/\overline{\text{GT}}_0$ 的优先级高于$\overline{\text{RQ}}/\overline{\text{GT}}_1$。

2. 8288 总线控制器

在最大模式系统中要用到总线控制器 8288，它根据 CPU 提供的$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$信号产生各种总线控制信号。为便于理解最大模式系统的组成，这里先介绍 8288 总线控制器的组成及工作原理。8288 逻辑框图如图 2-12 所示。

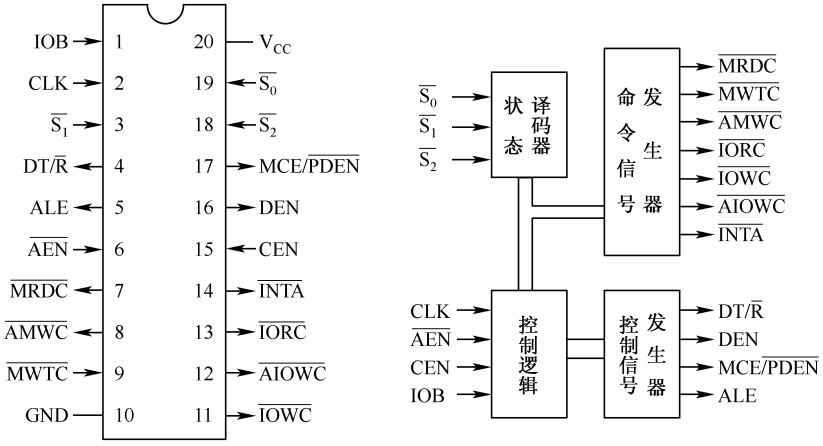


图 2-12　8288 总线控制器引脚图及结构示意图

1）$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$：来自 8086CPU 的状态信号。8288 对这些状态进行译码产生相应的总线命令信号和输出控制信号。

2）CLK：时钟输入端，通常接 8284 的 CLK 端。

3）$\overline{\text{AEN}}$：地址允许信号，输入。在系统总线方式下，当$\overline{\text{AEN}}$为低时，允许 8288 输出命令信号，$\overline{\text{AEN}}$为高时，各命令信号输出线处于高阻状态。局部总线方式下，此信号无关。

4）CEN：命令允许信号，输入。CEN 必须为高电平。如果 CEN 为低电平，则 8288 所有命令信号输出线都为无效电平。

5）IOB：总线方式控制信号，输入。8288 有两种工作方式，由 IOB 进行选择。

① IOB 为低时，8288 工作于系统总线方式(多处理器系统)。在多处理器系统中，如果存储器和 I/O 设备需被多个处理器所共享，这样的系统就需要使用总线仲裁。这种方式下，总线仲裁电路通过向 8288 的$\overline{\text{AEN}}$端送一个低电平信号来指示总线可供使用。

② IOB 为高时，8288 工作于局部总线方式(单处理器系统)。在此方式下，所有的 I/O 命令都是允许的，即处理器进行 I/O 操作期间，不需要仲裁电路送$\overline{\text{AEN}}$信号，8288 就会立即发出相应

的命令输出信号如$\overline{\text{IORC}}$等，并通过$\overline{\text{PDEN}}$和 DT/$\overline{\text{R}}$端控制总线收发器的工作。在单处理器系统中使用 8288，是为了增加控制总线的驱动能力。在多处理器系统中，如果外部设备不是共享的，而是某些外设从属于某一处理器，则也可用局部总线方式。

6）$\overline{\text{AIOWC}}$：超前 I/O 写命令，输出。在总线周期中，由该信号提前一个时钟周期发出 I/O 写命令，以便于 I/O 设备早作准备。

7）$\overline{\text{AMWC}}$：超前存储器写命令，输出。其功能与$\overline{\text{AIOWC}}$信号相似。

8）$\overline{\text{IOWC}}$：I/O 写命令，输出。指示数据总线上数据有效，可将数据写入被选中的 I/O 端口。一般用作 I/O 端口写选通信号。

9）$\overline{\text{IORC}}$：I/O 读命令，输出。通知外设端口将数据发送到数据总线上。

10）$\overline{\text{MRDC}}$、$\overline{\text{MWTC}}$：存储器读和存储器写命令，输出。它们的操作与$\overline{\text{IORC}}$、$\overline{\text{IOWC}}$作用相似，只是用于存储器读写。

11）MCE/$\overline{\text{PDEN}}$：总线主模块允许/外部数据允许双功能信号，输出。在系统总线方式中为 MCE 信号，用于中断控制器级连的控制信号。在局部总线方式中为$\overline{\text{PDEN}}$信号，作数据总线驱动器的选通信号。

12）$\overline{\text{INTA}}$、DT/$\overline{\text{R}}$、ALE 及 DEN 与 8086 最小模式的相应引脚信号功能相同，只有 DEN 信号的相位与最小模式相应引脚的相位相反。

3. 8086 最大模式下系统的典型配置

8086 最大模式下系统的典型配置如图 2-13 所示。与最小模式系统相比，增加了总线控制

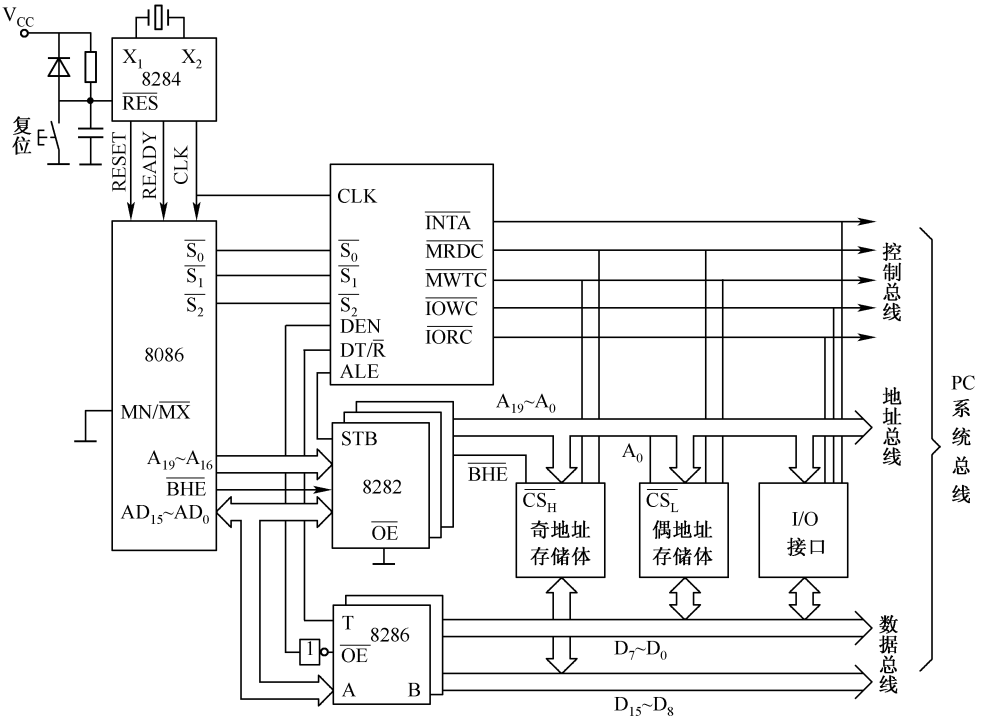


图 2-13　8086 最大模式系统结构示意图

器芯片 8288。所有总线控制信号均由$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$信号经 8288 译码处理后产生。采用 8288 可以实现多处理器配置,8288 用于协调各处理器间的工作。即使不配多处理器,增加 8288 之后也可增强控制总线的驱动能力。

8282 地址锁存器和 8286 总线驱动器的接法与最小模式下的接法相似,只是其控制信号不是来自 CPU,而是来自总线控制器 8288,即 8282 的地址锁存信号 STB 来自 8288。8282 输出使能信号$\overline{OE}$由 8288 数据输出使能信号 DEN 反相后获得。而传输方向控制信号 T 由 8288 DT/$\overline{R}$信号产生。8288 的 DT/$\overline{R}$信号与最小模式下 DT/$\overline{R}$信号意义相同。

8288 提供了单 CPU 和多 CPU 工作的可能性。当 IOB、$\overline{AEN}$接地,CEN 接 +5V 时,选择单 CPU 工作方式,这时 MCE/$\overline{PDEN}$线用作 MCE 信号。当 IOB、CEN 均接 +5V 时,则为多 CPU 工作方式,MCE/$\overline{PDEN}$线用作外部设备数据允许信号$\overline{PDEN}$。

2.4 8086 的操作时序

一个微处理器要完成预定的功能,需要执行许多操作。上电时要执行复位操作,执行一条指令要经过取指令、指令译码和执行指令的操作。执行指令时,根据指令的不同又有存储器读写操作、I/O 读写操作等。由于 8086CPU 有执行部件(EU)和总线接口部件(BIU)两部分,执行指令和总线操作是并行进行的。执行指令过程中,EU 若需要总线操作则向 BIU 申请总线周期。8086 的总线周期主要有内存或 I/O 读写总线周期、中断响应总线周期、暂停总线周期、最小模式下的总线保持周期、最大模式下的总线请求/允许总线周期等。掌握总线操作时序是理解总线操作的关键,是进行系统设计及扩展的必要知识准备。中断响应周期时序在第 7 章的中断部分叙述,本节介绍其他主要时序。

2.4.1 复位操作及时序

8086 的复位操作是通过 RESET 引脚上高电平信号实现的,RESET 引脚上的复位信号至少要保持 4 个时钟周期,上电复位时,RESET 信号应不小于 50μs。一旦 RESET 变为高电平,则立即停止指令执行,进入复位状态。若 RESET 一直持续高电平,则 8086 一直处于复位状态。复位操作的内容包括:第一,所有内部寄存器、标志寄存器 IR 及 ES、SS、DS 段寄存器清 0,指令队列缓冲器清空,指令指针寄存器(IP)清 0,CS 被置为 FFFFH,所以复位信号消失后,程序从 CS×16 + IP 即 FFFF0H 地址开始执行;第二,复位时,所有三态输出总线变为高阻状态,这些三态总线包括:AD_{15} ~ AD_0,A_{19}/S_6 ~ A_{16}/S_3,$\overline{BHE}/S_7$,$\overline{S_2}$(M/$\overline{IO}$),$\overline{S_1}$(DT/$\overline{R}$),$\overline{S_0}$($\overline{DEN}$),$\overline{LOCK}$,$\overline{WR}$,$\overline{RD}$,$\overline{INTA}$等。ALE、HLDA、QS_0、QS_1 等信号降为低电平,$\overline{RQ}/\overline{GT_0}$、$\overline{RQ}/\overline{GT_1}$ 等信号上升为高电平。

复位操作的时序如图 2-14 所示。某个时钟周期的上升沿检测到 RESET 引脚信号为高电平,则立即将内部 RESET 信号置为高电平。在内部 RESET 信号上升为高电平之后,经过 0.5 个时钟周期的不工作状态,所有三态输出总线变为高阻状态,且在 RESET 信号失效前,一直处于高阻状态。所谓不工作状态即是对总线不加控制状态,或者说是这些总线从原来状态进入高阻状态的过渡状态。

从复位过程的时序图可以看出,不管 CPU 原来做何种操作,对 RESET 信号的响应都是立即进行的。

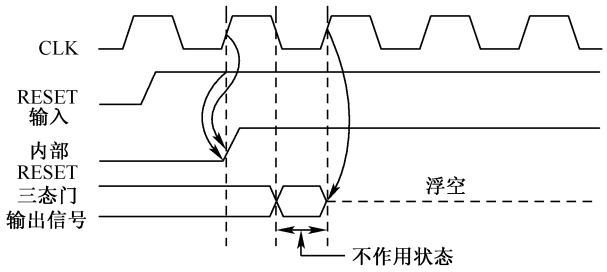


图 2-14　8086 的复位时序

2.4.2　最小模式下的总线读周期

在指令执行过程中,执行部件(EU)可能需要从内存或外设端口读取数据,因而进入总线读周期。图 2-15 给出了 CPU 从存储器或外设端口读取数据的工作时序。

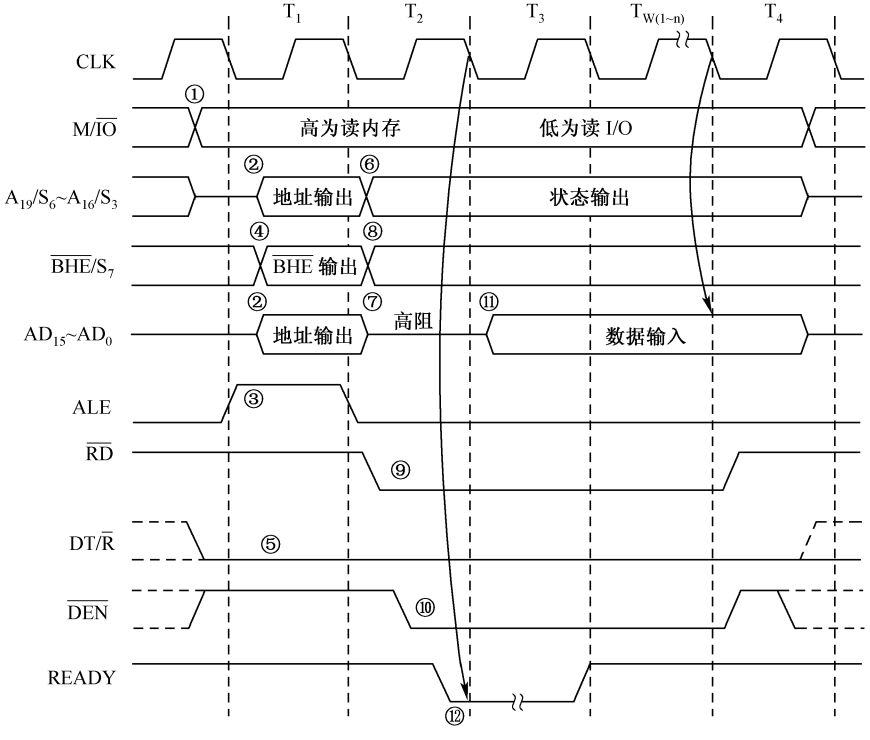


图 2-15　8086 最小模式下的读周期时序

一个总线读周期包括 4 个 T 状态,即 T_1 ~ T_4。当存储器或外设速度较慢时,可通过 READY 引脚在 T_3 和 T_4 之间插入一个或几个等待状态 T_W。下面按 T_1 ~ T_4 的顺序说明各个 T 状态总线上信号的变化。

1. T_1 状态

在进入总线读周期之前,即 T_1 前沿之前,M/$\overline{IO}$信号应该有效。若要读内存则 M/$\overline{IO}$上升为高电平,若要读 I/O 端口则 M/$\overline{IO}$下降为低电平(如图 2-15 中①所示)。A_{19} ~ A_{16}、AD_{15} ~ AD_0 线上输出地址信号(如图 2-15 中②所示),并持续一个 T 状态。

在 T_1 状态,地址锁存允许信号 ALE 有效,输出一个正脉冲。在其下降沿,将地址锁入 8282 地址锁存器(如图 2-15 中③所示)。

$\overline{BHE}$输出指示高 8 位数据线上信息是否有效(如图 2-15 中④所示)。$\overline{BHE}$信号常用作奇地址存储体的体选信号,配合地址信号 $A_{19} \sim A_1$ 实现存储器寻址,所以$\overline{BHE}$信号也应和地址信号一起由 ALE 信号锁入到地址锁存器中。当系统接有总线驱动器时,数据传输方向控制信号 DT/$\overline{R}$ 应降为低电平(如图 2-15 中⑤所示),指示现在为总线读(接收数据)。

2. T_2 状态

在 T_2 状态,地址信号消失(如图 2-15 中⑦所示),地址/数据复用总线进入高阻状态,为总线读操作作准备。$A_{19} \sim A_{17}$及$\overline{BHE}$线上输出状态信息 $S_7 \sim S_3$(如图 2-15 中⑥、⑧所示)。S_7 在 8086 系统没有用。

$\overline{DEN}$信号降为低电平,使数据总线驱动器信息收发使能(如图 2-15 中⑩所示)。

$\overline{RD}$信号有效,该信号送到内存或 I/O 端口,使它们将选中地址的数据送上数据总线(如图 2-15 中⑨所示)。

3. T_3 状态

在 T_3 状态内存或 I/O 端口将数据送上数据总线(如图 2-15 中⑪所示)。CPU 准备读入数据。在 T_3 的前沿(下降沿),CPU 查询 READY 引脚,若内存或外设工作速度较慢,来不及在基本总线周期内完成数据传送工作,则应通过逻辑电路在 T_3 前沿之前产生 READY 低电平信号;T_3 前沿若查到 READY 为低电平,则在 T_3 后自动插入一个等待状态 T_W;在 T_W 前沿继续查询 READY 信号,若 READY 仍为低电平,则继续插入 T_W,直到 READY 上升为高电平,则等待状态结束,进入 T_4 状态。

4. T_4 状态

在 T_4 前沿 CPU 将数据读入,总线读周期完成。$\overline{RD}$、DT/$\overline{R}$、$\overline{DEN}$等信号变为无效,所有三态总线变为高阻状态,为下一个总线周期作准备。

2.4.3 最小模式下的总线写周期

当 CPU 要向内存或 I/O 端口输出数据时,进入总线写周期。该周期也具有 $T_1 \sim T_4$ 基本状态,在内存或外设速度较慢的情况下,在 T_3 之后插入若干个 T_W 状态。各个 T 状态的具体操作如图 2-16 所示。

1. T_1 状态

T_1 状态的操作与总线读相同,即 M/$\overline{IO}$应在 T_1 前沿之前有效。

写内存时 M/$\overline{IO}$为高电平,写 I/O 端口时 M/$\overline{IO}$为低电平。$A_{19} \sim A_{16}$、$AD_{15} \sim AD_0$ 输出地址信号。$\overline{BHE}$有效则说明高 8 位数据线信息有效。ALE 信号用于地址锁存。与总线读不同的是 DT/$\overline{R}$上升为高电平,指示现在为总线写(数据发送),如图 2-16 中①、②、③、④、⑤所示。

2. T_2 状态

$A_{19}/S_6 \sim A_{16}/S_3$ 引脚输出状态信息 $S_6 \sim S_3$(如图 2-16 中⑥所示),$AD_{15} \sim AD_0$ 复用总线上输出要写出的数据信息,并一直保持到 T_4 状态的中间(如图 2-16 中⑦所示)。$\overline{BHE}$信号失效。$\overline{WR}$

信号降为低电平，并保持到 T_4 中部（如图 2-16 中⑨所示），该信号用于将输出数据写入选中的存储器或 I/O 端口。$\overline{DEN}$信号有效，使总线驱动器信息收发使能，它与 DT/$\overline{R}$信号配合，决定数据传输方向。

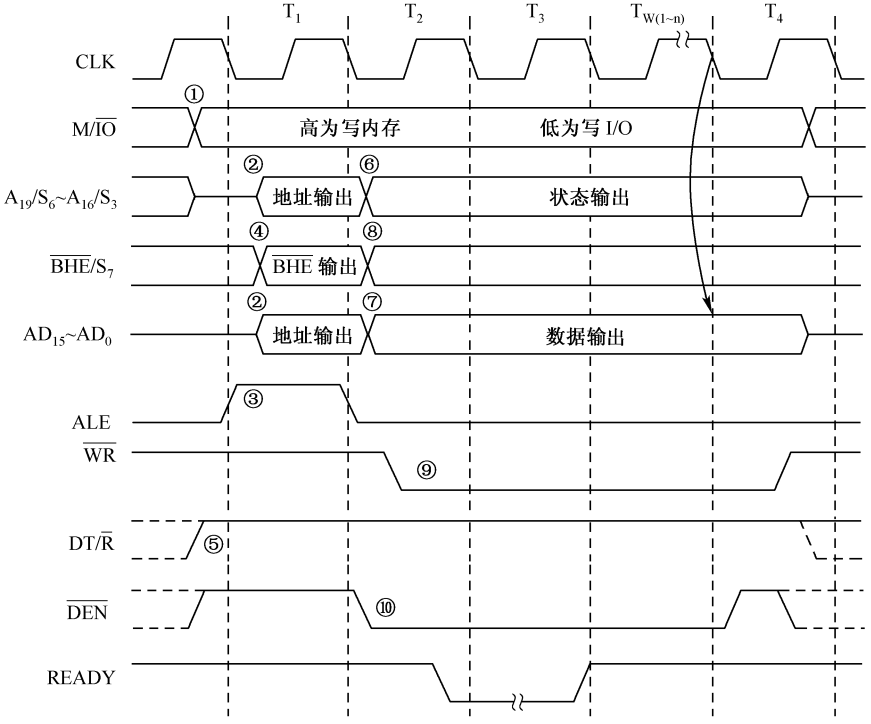


图 2-16　8086 最小模式下的写周期时序

3. T_3 及 T_W 状态

在 T_3 状态中，T_2 状态有效的信号继续保持有效，继续向外部写数据。在 T_3 的下降沿查询 READY，若内存或 I/O 端口在标准总线周期内来不及接收数据，则应通过逻辑电路在 T_3 前沿之前产生 READY 低电平信号。CPU 查到 READY 为低，则在 T_3 之后插入一个 T_W，并在 T_W 前沿继续查询 READY，直到 READY 上升为高电平，则结束等待进入 T_4 状态。

4. T_4 状态

总线写状态结束，所有控制信号变为无效状态，所有三态总线变为高阻态。

2.4.4　最大模式下的总线读周期

最大模式与最小模式的总线读周期操作在逻辑上基本相同，只是在最大模式下要同时考虑 CPU 发出的信号和总线控制器 8288 发出的信号。最大模式下，CPU 仍发出$\overline{RD}$信号，而总线控制器由$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$译码产生存储器读控制信号$\overline{MRDC}$和 I/O 读控制信号$\overline{IORC}$，如把它们看做是 M/$\overline{IO}$与$\overline{RD}$的组合信号则最大模式与最小模式读周期没有区别。而$\overline{MRDC}$和$\overline{IORC}$信号除区分出存储器与 I/O 端口外，交流特性也比较好。所以本节讨论$\overline{MRDC}$和$\overline{IORC}$作读控制信号的情况。时序图中由总线控制器产生的信号前标有 * 号。

最大模式下，每个总线周期开始前，$\overline{S_2}$ ~ $\overline{S_0}$被置为无源状态，总线控制器一旦检测到其中任

一个为0,则进入一个总线周期。最大模式下总线读周期时序如图2-17所示。

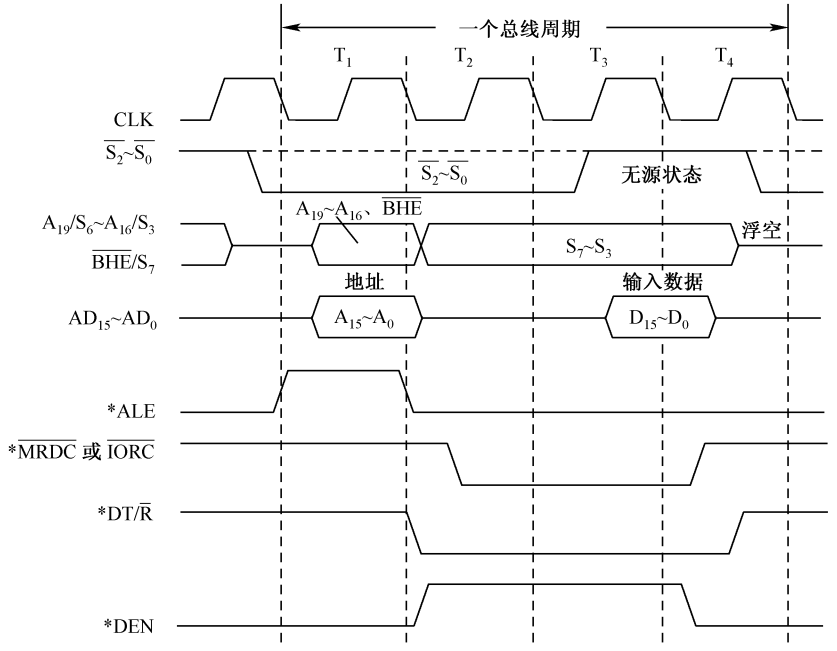


图2-17　最大模式下总线读周期时序

1. T_1 状态

CPU经$A_{19}/S_6 \sim A_{16}/S_3$、$AD_{15} \sim AD_0$送出20位地址信号及$\overline{BHE}$信号。总线控制器送出地址锁存允许信号ALE,用于地址锁存。总线控制器还使DT/$\overline{R}$降为低电平,表示当前执行总线读操作。

2. T_2 状态

CPU送出状态信号$S_7 \sim S_3$,并将地址数据/复用总线置为高阻状态,以准备数据读入。$\overline{DEN}$信号有效。总线控制器根据$\overline{S_2} \sim \overline{S_0}$的组合,产生$\overline{MRDC}$信号或$\overline{IORC}$信号送向存储器或I/O端口,用于数据读入操作,该信号持续到T_4中间。

3. T_3 状态

如果存储器或I/O端口速度足够快,这时应将数据送上数据总线,否则同样要插入等待状态。T_3状态中,$\overline{S_2} \sim \overline{S_0}$全部上升为高电平,进入无源状态,并一直继续到$T_4$。一旦进入无源状态,说明很快可以启动下一个总线周期。

4. T_4 状态

一个总线周期结束。数据从总线上撤销,数据/地址总线进入高阻状态。$\overline{MRDC}$(或$\overline{IORC}$)、DT/$\overline{R}$、$\overline{DEN}$等信号失效。$\overline{S_2} \sim \overline{S_0}$信号按照下一个总线周期的操作内容变化,准备进入下一个总线周期。

2.4.5　最大模式下的总线写周期

最大模式下CPU通过总线控制器为存储器和I/O端口提供两组写信号,一组为普通的存储器写信号$\overline{MWTC}$和普通I/O写信号$\overline{IOWC}$(这组信号相当于最小模式下M/$\overline{IO}$与$\overline{WR}$的组合信号),

另一组为提前的存储器写信号$\overline{\text{AMWC}}$和提前的 I/O 写信号$\overline{\text{AIOWC}}$。提前信号比普通信号提前一个时钟周期。最大模式下总线写周期的时序如图 2-18 所示。$\overline{S_2}$ ~ $\overline{S_0}$信号仍应在 T_1 之前置为有源状态。各个 T 状态的具体操作如下：

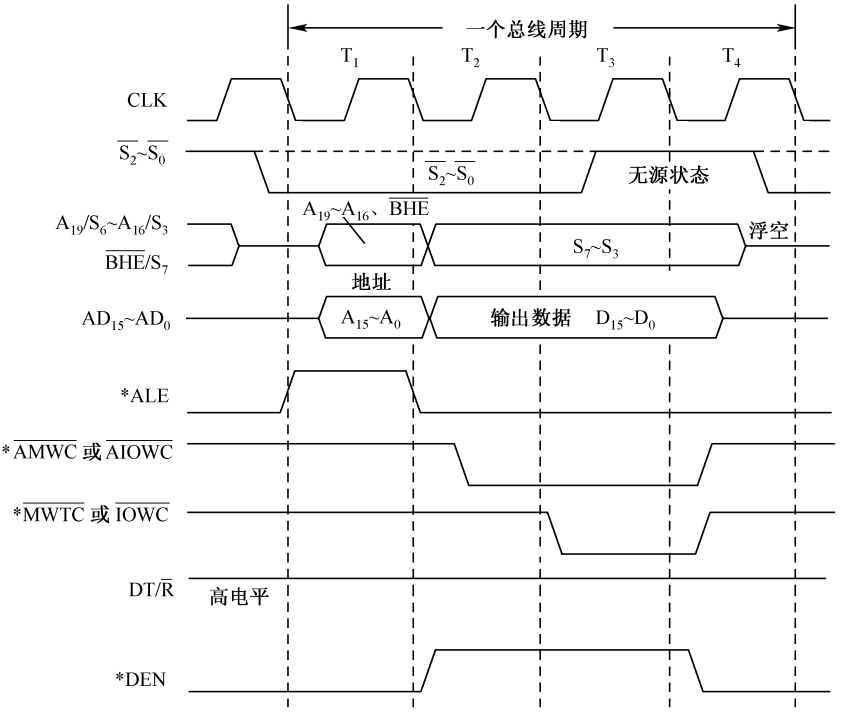


图 2-18　最大模式下的写操作时序

1. T_1 状态

A_{19}/S_6 ~ A_{16}/S_3 及 AD_{15} ~ AD_0 输出地址信号。高 8 位数据线有效信号$\overline{\text{BHE}}$输出相应电平，总线控制器使 DT/$\overline{\text{R}}$信号变为高电平，即控制总线驱动器作数据输出。总线控制器发出 ALE 信号用于地址锁存。

2. T_2 状态

总线控制器输出 DEN 高电平使数据总线驱动器使能。提前的存储器写信号$\overline{\text{AMWC}}$或I/O写信号$\overline{\text{AIOWC}}$降为低电平。S_7 ~ S_3 输出状态信号，CPU 的输出数据送到数据总线 AD_{15} ~ AD_0。T_2 状态出现的信号均持续到 T_4 中间。

3. T_3 状态

总线控制器使普通的写控制信号$\overline{\text{MWTC}}$或$\overline{\text{IOWC}}$生效。可以看出，提前的写控制信号比普通写控制信号提前一个时钟周期。慢速存储器芯片或 I/O 设备可采用提前信号作为写控制信号。T_3 状态中$\overline{S_2}$ ~ $\overline{S_0}$进入无源状态。同样在 T_3 之后，根据 READY 信号可适当插入等状态 T_W。

4. T_4 状态

总线写周期结束。A_{19}/S_6 ~ A_{16}/S_3、AD_{15} ~ AD_0 复用总线变为高阻状态。总线写控制信号$\overline{\text{AMWC}}$(或$\overline{\text{AIOWC}}$)、$\overline{\text{MWTC}}$(或$\overline{\text{IOWC}}$)等控制信号失效。$\overline{S_2}$ ~ $\overline{S_0}$状态按下一个总线周期的操作改

变,准备进入新的总线周期。

2.4.6 最小模式下的总线请求/响应周期

当系统中有多个总线驱动模块时,CPU 以外的总线驱动模块必须通过总线申请信号 HOLD 向 CPU 提出总线申请。CPU 在各种周期的每个 T 状态的上升沿查询 HOLD,当查到 HOLD 为高电平时,则在本总线周期结束后(即 T_4 或 T_I 之后)给予响应。

CPU 一旦响应其他模块的总线申请,则让出全部三态数据/地址总线、状态及控制总线(包括$\overline{RD}$、$\overline{WR}$、$\overline{INTA}$、M/$\overline{IO}$、$\overline{DEN}$及 DT/$\overline{R}$等)的控制权,即使这些引脚呈现高阻状态,并以 HLDA 作为回答信号。总线申请模块接到 HLDA 高电平信号后,即可接管总线。待总线操作完成后,再将 HOLD 降为低电平,CPU 查到 HOLD 低电平后,重新收回总线控制权。最小模式下的总线请求/响应时序如图 2-19 所示。

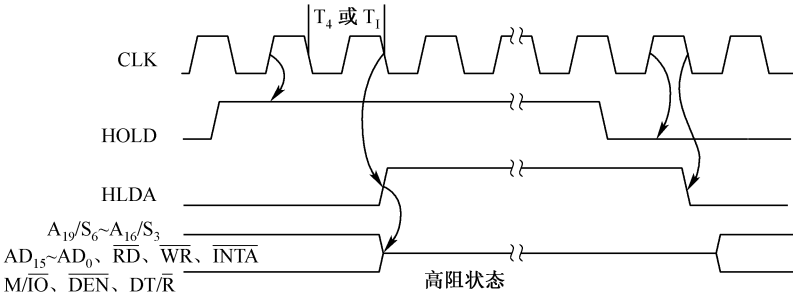


图 2-19 总线请求/响应时序

对总线请求响应周期的进一步说明:

1) 总线申请模块使 HOLD 变为高电平后,在其后的时钟脉冲上升沿 CPU 才能查到,且要等当前总线周期结束后,即在 T_4 或空闲周期 T_I 之后才能发出 HLDA 信号,让出总线,这期间会有几个时钟周期的延迟。

2) HOLD 信号直接影响总线接口部件(BIU)的工作,对执行部件 EU 的影响是间接的。即 CPU 让出总线后,EU 照样从指令队列中取出指令执行,直到需要总线操作或指令队列已空时才停下来。

3) 其他总线部件完成工作交回总线控制权后,CPU 接管总线,但并不一定马上控制总线,直到程序执行中需要总线操作时才恢复对总线的控制。为了防止总线切换过程中没有任何模块控制总线导致控制线上电平过低,控制线应接上拉电阻。

2.4.7 最大模式下的总线请求/响应周期

最大模式下也提供了 8086 CPU 与其他总线主模块(如 DMA 控制器、协处理器等)之间传递总线控制权的功能。这时不是通过 HOLD 和 HLDA 信号来联络,而是将这两条线扩充成两个双向线$\overline{RQ}$/$\overline{GT}_0$、$\overline{RQ}$/$\overline{GT}_1$,这两个双向线均称为总线请求/总线允许信号线。$\overline{RQ}$/$\overline{GT}_0$ 比$\overline{RQ}$/$\overline{GT}_1$ 有较高的优先级,但总线的使用不能嵌套。通过总线控制权的转移,各个总线主模块可以共享内存、外设端口、总线控制器、数据总线驱动器和地址锁存器等硬件资源。

最大模式下总线请求/允许的时序如图 2-20 所示。其操作过程如下:

1) CPU 外的总线主模块要使用总线时,在$\overline{RQ}$/$\overline{GT}$线上发出一个时钟周期宽的低电平总线请

求信号$\overline{RC}$。

2）CPU 在每个时钟脉冲的上升沿查询$\overline{RQ}$引脚，如查到低电平，则在下一个 T_4 状态或 T_I 状态在同一引脚上送出一个时钟周期宽的低电平允许信号$\overline{GT}$。CPU 一旦发出$\overline{GT}$，则所有三态引脚（包括地址数据线、地址/状态线等）都变为高阻状态。

3）其他总线模块接到 CPU 发出的$\overline{RQ}$/$\overline{GT}$低电平信号后，便得到了总线控制权，可以占用一个或多个总线周期。外部模块要释放总线时，只要在$\overline{RQ}$/$\overline{GT}$线上发一个时钟周期的低电平释放信号，CPU 检测到此信号，在下一个时钟周期便收回总线控制权。

关于总线请求/允许过程还有些情况要补充说明：

1）如 CPU 正在执行存储器或 I/O 端口读写周期，则总线请求信号必须在 T_2 之前到达才有效，否则要重新申请。

2）如 CPU 正在用低 8 位数据线传送数据，则总线请求无效，等到数据传输结束，外部主模块必须重新发出总线请求信号。

3）CPU 正在执行第一个中断响应周期时，总线请求无效。

4）CPU 正在执行总线封锁指令，总线请求无效。

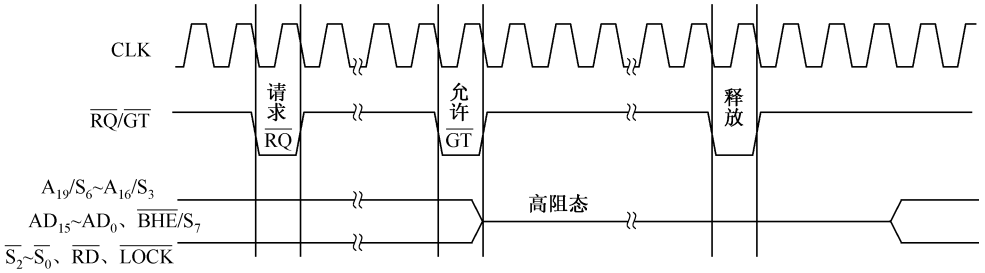


图 2-20　最大模式下总线请求/允许/释放时序

2.5　习题

1. 8086CPU 由哪两部分构成？它们的主要功能是什么？在执行指令期间，EU 能直接访问存储器吗？为什么？

2. 8086CPU 与传统的计算机相比在执行指令方面有什么不同？这样的设计思想有什么优点？

3. 8086CPU 中有哪些寄存器？各有什么用途？

4. 状态标志与控制标志有何不同？程序中是怎样利用这两类标志的？标志寄存器有哪些标志位？各在什么情况下置位？

5. 求出下列运算后各个标志的状态，并说明进位标志和溢出标志的区别。

（1）1278H + 3469H　　（2）54E3H - 27A0H

（3）3881H + 3597H　　（4）01E3H - 01E3H

6. 8086CPU 中存储器的逻辑地址和物理地址之间有什么关系？各有多少值？

7. 8086CPU 使用的存储器为什么要分段？怎样分段？为什么要设置段寄存器？有几个段寄存器？各段寄存器有什么意义？

8. 简述 A_0 与$\overline{BHE}$在 8086 系统中的应用。

9. 8086 系统中为什么要采用地址锁存器 8282？采用什么方法从分时复用地址/数据线中将数据和地址信号分离出来？

10. 8086 和 8088 CPU 的主要区别是什么？

11. 8086 系统中的存储器采用什么结构？如何与地址、数据线连接？

12. 8086 的 I/O 端口寻址范围是多少？什么是 I/O 端口与内存分别独立编址？

13. 在对存储器和 I/O 设备读/写时，要用到$\overline{IOR}$（IO 读）、$\overline{IOW}$（IO 写）、$\overline{MR}$（存储器读）、$\overline{MW}$（存储器写）信号，这些信号的作用是什么？它们在最小模式时分别可用怎样的电路得到？请画出示意图。

14. 什么是基地址和位移量？它们之间有何联系？

15. 设 CS = 1200H，IP = 0FF00H，此时指令的物理地址是多少？指向这一物理地址的 CS 和 IP 的值是唯一的吗？

16. 若 CS = 1000H，指出当前代码段可寻址的存储空间的大小和地址范围。

17. 简述 8086 单 CPU 和多 CPU 系统各自的主要特点，并说明有何差别。

18. 时钟周期、T 状态、总线周期、指令周期的定义是什么？什么情况下会出现空闲周期？

19. 8086CPU 读/写总线周期包含几个时钟周期？什么情况下需要插入 T_W 等待周期？插入 T_W 的数量取决于什么因素？

20. 8086CPU 复位后，有哪些特征？8086 系统的启动程序如何去找？

21. 8086 系统在最小模式时应该怎样配置？试画出这种配置并标出主要信号的连接关系。

22. 画出最小模式时读存储器或 I/O 设备的总线周期时序。

第3章　从8086到Pentium系列微处理器的技术发展

目前世界上大多数微型计算机系统的微处理器都属于Intel公司的8086微处理器家族。各种微机系统的兼容性(即可以运行同一目标代码的可执行程序,且得到相同的结果),都是建立在8086微处理器家族成员具有相同的基本体系结构基础之上的,即由8086微处理器和8087数学协处理器组成的体系结构。Intel 8086家族的其他成员在体系结构上有许多新的发展,但必须包括8086和8087微处理器的基本体系结构的功能,这是兼容性的基础。

自从1978年Intel公司推出16位微处理器8086之后,随着微型计算机应用领域的扩大和应用技术的深入,只能处理单任务、内存容量最大为1MB的8086/8088已不能满足应用的需要。在迅速发展的集成电路技术的支持下,Intel公司相继推出了一系列功能更强、性能更高的微处理器。1982年Intel推出了与8086向下兼容的微处理器80286。80286有16条数据线,24条地址线,寻址空间为16MB。相同时钟频率下,其执行速度可比8086快2.5倍。80286的主要目的是多任务处理,但这个目标并没有很好地实现。针对80286存在的问题,Intel公司于1985年推出了32位微处理器80386,其数据、地址总线都是32位,寻址范围达4GB,真正实现了多任务。

1989年又推出了80486,它实际上是80386的增强型,除保留80386的所有功能外,片内增加了协处理器和高速缓存(CACHE),并采用了精简指令集计算机(Reduction Instruction Set Computer,RISC)技术,相同时钟频率下80486的处理速度是80386的2倍。

继80486之后,Intel公司于1993年又推出了Pentium系列微处理器第一代产品(称为P5或80586);1995年又发布了代号为P6(Pentium Pro)的微处理器;1998年和1999年又相继推出了PentiumⅡ、Pentium Ⅲ(即奔腾2代和奔腾3代,简称奔2和奔3);2001年推出了Pentium 4(奔4),它采用了超标量双流水线结构、高性能浮点运算、独立的指令和数据高速缓存、分支指令预测、增强的64位数据总线、超线程HT()等多项新技术,使其达到甚至超过了高档工作站的性能。

表3-1给出了80X86系列微处理器的概况。

表3-1　80X86系列微处理器的概况

型　　号	发布年份/年	字长/bit	集成度/万个晶体管	主频/MHz	引脚数/个	数据总线/位	地址总线/位	寻址空间	高速缓存(Cache)
8086	1978	16	2.9	4.77	40	16	20	1MB	无
80286	1982	16	13.4	20	68	16	24	16MB	无
80386	1985	32	27.5	133	132	32	32	4GB	有
80486	1990	32	120	100	168	32	32	4GB	8KB
Pentium ~ Pentium 4	1993 ~ 2001	64	60 ~ 4200	60 ~ 3200	296 ~ 478	64	32 ~ 36	4 ~ 64GB	8KB数据,8 ~ 32KB指令,512KB二级高速缓存

本章主要介绍80386CPU,同时也对80286、80486及Pentium微处理器的改进之处和特点做简单介绍。

3.1　80286微处理器简介

尽管8086微处理器获得了巨大的成功,得到了广泛的应用,但随着应用领域的扩大和应用

技术的深入,其功能和性能已不能满足需求。8086 存在的问题主要表现在以下两个方面:首先,1MB 的存储器地址空间在很多应用中显得太小了,应该扩大;其次,8086 的硬件不适合在系统中同时运行多道程序、执行多个任务。但是,由于技术的发展,要求微处理器的硬件像小型计算机、甚至大型计算机那样,能适应多任务的要求。另外,由于已经有相当数量的软件在 8086 系统机上运行,为了不遗弃这些软件资源,要求后继开发的 CPU 与 8086 兼容。1982 年,Intel 推出了 80286。在开发时 80286 就提出了设计目标,即解决 8086 存在的上述问题。

3.1.1 80286 的特点及相对 8086 体系结构的增强点

80286 采用 68 引脚的 4 列直插式封装,具有独立的 16 条数据线和 24 条地址线,最多可寻址 16MB 的实际存储空间和 64KB 的 I/O 地址空间。

80286 对 8086 体系结构进行了很多改进,主要如下:

① 将 8086 的 BIU(总线接口部件)分成了 AU(地址部件)、IU(指令部件)和 BU(总线部件)3 个部件。CPU 内部的 4 个处理部件(EU、AU、IU、BU)可以并行操作,提高了处理速度。

② 数据线和地址线完全分离。在一个总线周期中,当有效数据出现在数据总线上时,下一个总线周期的地址已经送到地址总线,形成总线周期的流水作业。

③ 具有“实地址模式”(Real Address Mode,简称为“实模式”)和“保护虚地址模式”(Protected Virtual Address Mode,简称为“保护模式”)两种工作模式。

④ 能运行实时多任务操作系统,支持存储管理和保护功能。存储管理功能实现在实模式和保护模式两种模式下访问存储器;保护功能包括对存储器进行合法操作和对任务实现特权级的保护。

⑤ 实现了虚拟存储管理。在 8086 系统中,程序能够占有的存储器就是 CPU 能够访问的物理存储器,其大小为 1MB。80286 在保护模式下支持对虚拟存储器的访问。虚拟存储器与物理存储器是有区别的,其存储空间的大小也是不同的。所谓虚拟存储器,是指程序可以占有的存储空间,但它并不是 CPU 能够直接访问的内存的实际物理地址空间,而是由外存(如硬盘)提供的大容量的所谓虚拟地址空间。用户编写的应用程序并不受限于实际内存容量的大小,是放在虚拟存储器上的。当程序运行时,首先将要执行的程序和要存取的数据从虚拟存储器加载到物理存储器上,这就需要把程序和数据从虚拟地址空间转换到物理地址空间,这种转换称为映射。80286 中,虚拟存储器(虚拟地址空间)的大小可达 $2^{30}B = 1GB$,而物理存储器(内存地址空间)的大小只能达到 $2^{24}B = 16MB$。采用虚拟存储管理技术,就是为了解决适应多任务、多用户问题。

⑥ 与 80286 配合使用的数学协处理器是 80287。80287 基本与 8087 相同,但可以适应 80286 的两种工作模式。

3.1.2 80286 的保护模式

80286 具有两种工作模式,即实模式和保护模式。

当 80286 工作于实模式时,它的 24 条地址线中只有低 20 位地址有效,其寻址空间和寻址方法与 8086 完全相同。对于程序员来说,相当于 8086 的最大模式系统,且寄存器结构和寻址方式与 8086 相同。不同的是 20 位地址总线和 16 位数据总线不再分时复用芯片的引脚,并且增加了某些新指令。8086 的应用程序不需要修改就可以移到该方式下运行,但是运行速度更快。

80286 的另一工作模式,即保护模式提供了许多新的功能。与实模式相比,最明显的差别是

它的 24 条地址线全部有效,可寻址存储器空间扩大到 16MB。

保护模式体现了 80286 的特色,主要是对存储器管理、虚拟存储和对地址空间的保护。虽然 80286 的实存地址空间只有 16MB,但在保护模式下,可为每个任务提供多达 1GB 的虚拟存储空间和保护机制,有力地支持了多用户、多任务的操作。在保护模式下,80286 的存储管理仍然分段进行,每个逻辑段的最大长度为 64KB,但每个任务可使用的逻辑空间却高达 1GB。

在保护模式下,那些内存装不下的逻辑段,将以文件形式存在外存储器中,当处理器需要对它们进行存取操作时就会产生中断,通过中断服务程序把有关的程序或数据从外存储器调入到内存,从而满足程序运行的需要。

在保护模式下,80286 提供了保护机制,它们由硬件提供支持,一般不会增加指令的执行时间,这些保护包括:对逻辑段的操作属性(可执行、可读、只读、可写)和长度界限(1 ~ 64KB)进行检查,禁止错误的段操作。

为不同程序设置了 4 个特权级别(Privilege Level),提供若干特权级参数,可让不同程序在不同的特权级别上运行。8086 系统程序和用户程序处于同一级别,并存放在同一存储空间,所以系统程序有可能遭到用户程序的破坏。而 80286 依靠这一机制,可支持系统程序和用户程序的分离,并可进一步分离不同级别的系统程序,大大提高了系统运行的可靠性。

应该指出的是,80286 实际上并没有很好地实现多任务处理特性。在处理多任务时,经常导致程序运行非正常停止,特别是在实模式与保护模式之间进行转换时,这个问题尤其突出。因此,Intel 公司很快推出了性能更加优良的微处理器 80386。

3.2 80386 微处理器

80386 是一个与 80286 兼容的高性能全 32 位微处理器,它成功地实现了从 16 位微处理器体系结构向 32 位微处理器体系结构的过渡。

80386 微处理器采用 1.5 μm CHMOS – Ⅲ工艺,在一片芯片上集成了超过 27.5 万个晶体管。80386 在体系结构设计上进行了革命性变革,实现了全 32 位结构,硬件组织上采用硬件乘/除部件、流水线结构、指令重叠执行、高速缓存、虚拟存储等技术,并充分考虑对操作系统的支持,在芯片级实现了段页式存储管理机制、快速多任务硬件切换开关等,使 80386 达到单片小型机的水平。

3.2.1 80386 的特点及其体系结构

1. 80386 的特点

80386 是全 32 位结构,它的外部数据总线和内部数据通道,包括寄存器、ALU 和内部总线都是 32 位的,提供 32 位的存储空间寻址能力和 32 位的外部总线接口单元,能灵活处理 8 位、16 位、32 位 3 种数据类型。

80386 有以下 3 种工作模式:

其一为实模式。复位后自动进入这一模式, 其工作模式与 8086 相同,地址总线仍为 20 位,数据总线为 32 位,数据总线与地址总线是相互独立的,内部寄存器主要作为 16 位使用,也可以按 32 位使用。

保护模式又分两种。一种为虚拟 86 模式,即与 80286 的保护模式相同,主要指操作数和段内偏移地址保持为 16 位;另一种是 386 保护模式,操作数和段内的偏移地址都是 32 位,地址总

线也是 32 位,物理地址空间为 $2^{32}B = 4GB$。

80386 的硬件结构可分成 6 个逻辑单元,它们以流水线方式工作,CPU 主频可达 133MHz。其硬件设计有支持段页式存储管理部件,易于实现虚拟存储系统。在保护模式下的分段寻址体系,与操作系统相配合可以组成虚拟存储器系统,而且虚拟地址空间远远大于物理空间,一个任务的最大虚拟空间可达 $2^{46}B = 64TB$(1TB = 1024GB)。

80386 硬件支持多任务处理,用一条指令就可以实现任务切换。

80386 设置了 4 级特权级,按优先顺序依次为 0 级、1 级、2 级、3 级,前 3 级用于操作系统程序,后 1 级用于用户程序。

2. 80386 的体系结构

80386 的内部结构如图 3-1 所示。由图可以看出,80386 主要由 6 个部件组成:总线接口部件(BIU)、指令预取部件(IPU)、指令译码部件(IDU)、执行部件(EU)、段管理部件(SU)、页管理部件(PU)。为方便理解,这里将这 6 个部件按功能分成总线接口部件(BIU)、中央处理部件(CPU)和存储管理部件(MMU)3 大部分。

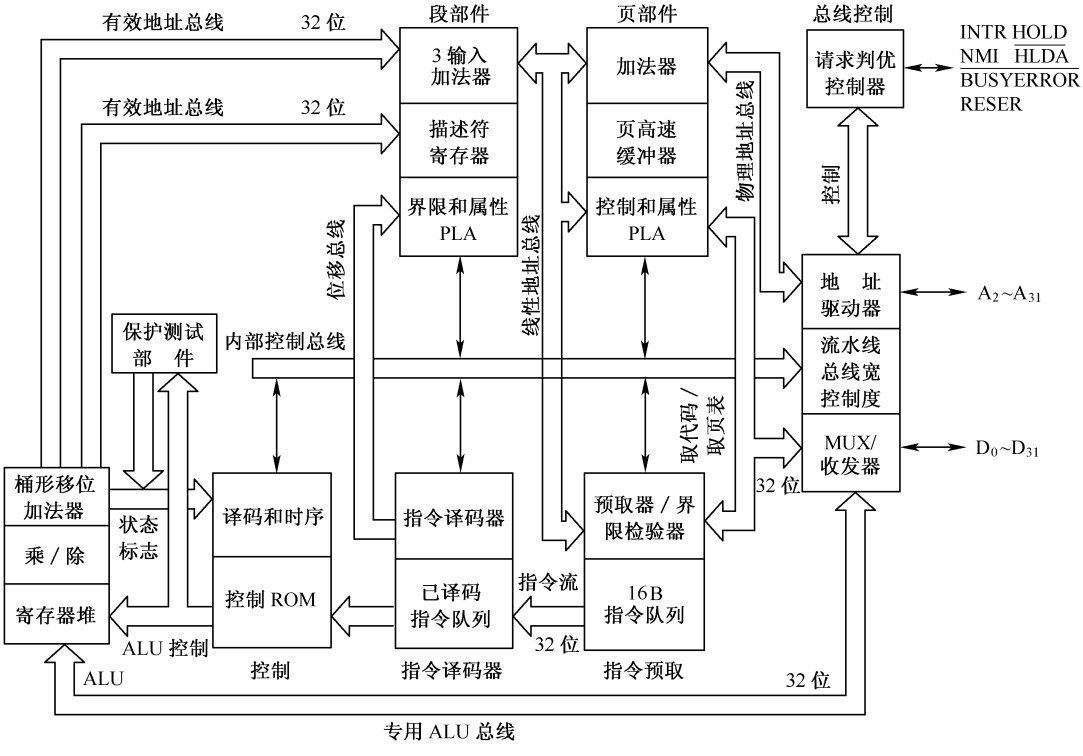


图 3-1 80386 的内部结构

1) 总线接口部件(BIU)是 80386 与外界的接口,它通过数据总线、地址总线和控制总线将外部部件与 80386 连接起来,完成对存储器和 I/O 端口的访问、80386 与 80387 数学协处理器的协调等功能。当取指令、取数据和分页部件产生请求时,总线接口部件能够响应请求,并按优先权进行排队,充分利用总线宽度传送数据。由于总线数据传送与总线地址形成可同时进行,所以总线周期只用两个时钟周期,一旦没有总线请求,总线接口部件能将下一条指令自动送到指令预取队列。

2）中央处理部件（CPU）包括指令预取部件（IPU）、指令译码部件（IDU）和执行部件（EU）。

指令预取部件是一个 16B 的指令队列寄存器（大约可存放 5 条指令），与 8086 相似，每当指令预取队列不满或发生控制转移时，指令预取部件就向总线接口部件发出总线请求，如果这时总线处于空闲状态时，它从存储器预先读取待执行的指令代码并存放到指令队列寄存器中。由于指令预取的优先级别低于数据传送等总线操作，因此，绝大部分是利用总线空闲时间预取指令。

指令译码部件对预取得指令代码译码后，送入已译码指令队列中等待执行部件执行。已译码指令队列可以容纳 3 条已经译码的指令，当已译码指令队列中有空闲字节时，译码部件就从指令队列寄存器中取出下一条指令进行译码。

执行部件主要包括寄存器、32 位的算术逻辑单元（ALU）、桶形移位加法器和乘/除运算部件以及对存储保护功能进行测试的测试部件。桶形移位加法器是一个 64 位的可实现移位、环移及位处理功能的高效移位器。指令队列中有一个字段指出本指令在微程序 ROM 中的开始地址，并由此顺序执行微指令，直到本指令执行完成。

3）存储管理部件（MMU）包括段管理部件（SU）和页管理部件（PU），它们的功能是实现存储器的段、页式管理，从而实现虚拟存储器系统和多任务处理。段管理部件的作用是应执行部件的请求，把逻辑地址转换成线性地址，在完成地址转换的同时还要执行总线周期分段的违章检验工作。页管理部件的作用是把由段管理部件或指令预取部件产生的线性地址转换成物理地址，并且要检验访问是否与页属性相符合。为了加快线性地址到物理地址的转换速度，80386 内设有一个页描述符高速缓冲存储器（TLB），可以存储 32 项页描述符。所以在地址转换期间，绝大部分情况不需要到内存中查页目录表和页表。试验证明其命中率可达 98%，因此大大加快了地址转换速度。对于 TLB 没有命中的地址转换，80386 设有硬件查表功能，这使因查表而引起的速度下降明显好转。

上述 6 个部件既能各自独立操作，也能与其他部件协调一致地并行工作。当取一条指令和执行一条指令时，每个部件都会完成一项任务或完成某一操作步骤。这样，既可以同时对不同指令进行操作，又可以同时对同一指令的不同部分并行地进行操作。例如，当总线接口部件正在执行某一条指令的写存储器操作的同时，指令译码部件可能正在对另一条指令进行译码，而执行部件却可能正在处理第 3 条指令。这样，80386 整个处理器就分成指令流水线、地址流水线和执行流水线 3 部分，这 3 部分既可协调一致工作，也可以并行工作，大大提高了处理器的工作效率。

3.2.2 80386 引脚的功能

80386CPU 共有 132 个引脚，其中 20 个用作 V_{CC}，21 个用作 GND，34 个地址线，32 个数据线，17 个控制线，其余 8 个引脚未用。下面简要说明它们的功能。

1. 数据线 $D_{31} \sim D_0$ 和总线宽度控制信号 $\overline{BS_{16}}$

这 32 条数据线是双向三态的，在总线宽度控制信号 $\overline{BS_{16}}$ 的控制下，可实现 16 位或 32 位数据传送。当 $\overline{BS_{16}}$ 为高电平时，使用 $D_{31} \sim D_0$ 进行 32 位数据传送；当 $\overline{BS_{16}}$ 为低电平时，使用 $D_{15} \sim D_0$ 进行 16 位数据传送。

2. 地址线 $A_{31} \sim A_2$ 和字节控制信号 $\overline{BE_0} \sim \overline{BE_3}$

这 34 个引脚是三态、输出引脚，由 $\overline{BE_0} \sim \overline{BE_3}$ 译码产生 A_1 和 A_0，加上 $A_{31} \sim A_2$ 32 条地址线，可寻址 4GB 物理存储器地址。只要当 $\overline{BE_0}$ 为低电平时，译码结果为 $A_1 = 0$、$A_0 = 0$，对该类地址的

内存单元数据的传送在数据总线 $D_7 \sim D_0$ 上进行；当$\overline{BE_0}$为高电平，只要$\overline{BE_1}$为低电平，译码结果为 $A_1=0$、$A_0=1$，对该类地址的内存单元数据的传送在数据总线 $D_8 \sim D_{15}$ 上进行；当$\overline{BE_0}$和$\overline{BE_1}$为高电平时，只要$\overline{BE_2}$为低电平，译码结果为 $A_1=1$、$A_0=0$，对该类地址的内存单元数据的传送在数据总线 $D_{16} \sim D_{23}$ 上进行；当$\overline{BE_0} \sim \overline{BE_2}$为高电平、$\overline{BE_3}$为低电平时，译码结果为 $A_1=1$、$A_0=1$，对该类地址的内存单元数据的传送在数据总线 $D_{24} \sim D_{31}$ 上进行。这样，通过$\overline{BE_0} \sim \overline{BE_3}$可控制选择访问内存的任一字节。

3. 协处理器接口信号

80386 可与数学协处理器 80287、80387 连接，以进行快速复杂的数学运算。为实现 80386 与 80387 协调一致地工作，需要使用下述 3 个接口信号。

1）PEREQ：协处理器向 80386 发出的请求信号，有效时表示协处理器请求与存储器之间传送数据。80386 响应该请求后，将按照指令的要求控制对存储器的读写。

2）$\overline{BUSY}$：协处理器向 80386 发出的状态信号，有效时表示协处理器正在执行指令，处于忙状态，暂时不能接收新的指令。

3）$\overline{ERROR}$：协处理器向 80386 发出的状态信号，有效时表示协处理器出错。80386 在检测到$\overline{ERROR}$信号后，将转到错误处理子程序来处理该类错误。

80386 在每个总线周期都将自动检测这 3 个信号，以决定是否能让协处理器执行下一条指令。

4. 其他信号

1）D/$\overline{C}$：数据/控制信号，输出，表示当前是数据传送周期还是控制周期。

2）$\overline{NA}$：“下一个地址”请求信号，输入，有效时则允许地址流水线进行操作。

3）$\overline{ADS}$：地址状态信号，三态输出，类似于 8086 的 ALE 信号。

4）其余如 W/$\overline{R}$、M/$\overline{IO}$、INTR、NMI、HOLD、HLDA、RESET、$\overline{READY}$、$\overline{LOCK}$、CLK 等信号与 8086 的同名信号相似，不再赘述。

3.2.3 80386 的寄存器组

80386 共有 34 个寄存器，按功能可分为通用寄存器、段寄存器、状态和控制寄存器、系统地址寄存器、调试寄存器及测试寄存器。其示意图如图 3-2 所示。

1. 通用寄存器

8 个通用寄存器和 8086 通用寄存器相同，只是扩展到 32 位，寄存器名字前加一个字符 E，即 EAX、EBX、ECX、EDX、ESI、EDI、EBP、ESP，支持 8 位、16 位和 32 位操作，用法和 8086 系统相似。例如，AX 是 EAX 的低 16 位，AH 是 AX 的高 8 位。

2. 段寄存器

6 个 16 位段寄存器包括 CS 代码段寄存器，DS 数据段寄存器，SS 堆栈段寄存器，ES、FS、GS 为 3 个附加段寄存器。在实模式下，段寄存器的用法和 8086 系统相同，只是增加了两个附加段寄存器 FS、GS。在保护模式（即支持多任务的工作模式）下，段寄存器称为段选择符（Selector），与描述符配合实现段寻址。

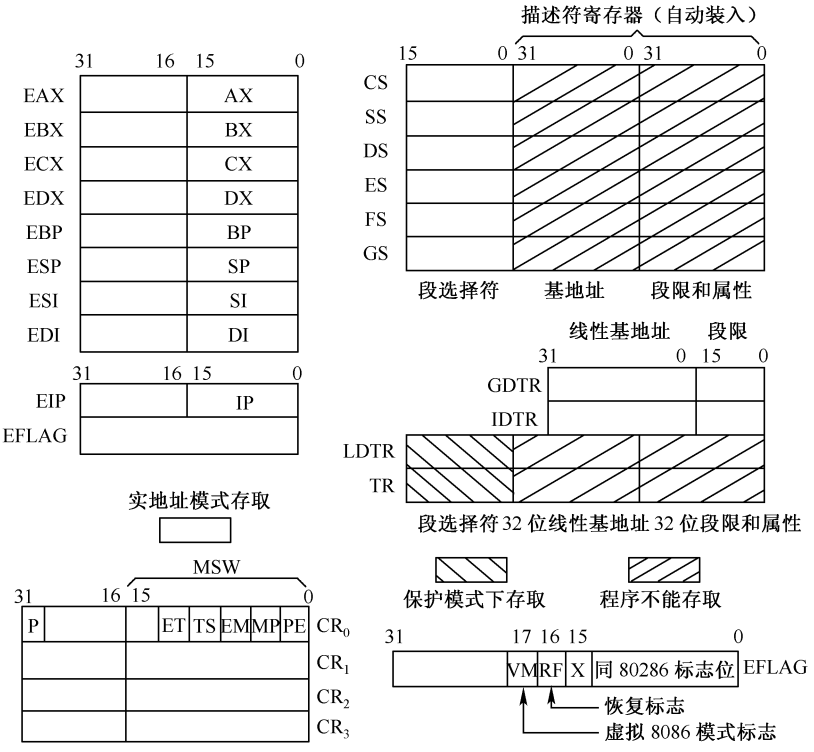


图 3-2　80386 寄存器配置

3. 段描述符寄存器

64 位的段描述符寄存器对程序员是不可见的。为了加快对内存中描述符表的查询速度，在段选择符内容装入时，段描述符同时装入段描述符寄存器。这样，只要段选择符内容不变（一般选择符内容很少变化），就不需要到内存中查描述符表，从而加快了段地址寻址的速度。描述符寄存器的内容包括段基地址、段限和段属性。段限指出本段的实际长度，与段属性一起主要用于段保护，防止不同任务进入不该进入的段进行操作。

4. 状态和控制寄存器

状态和控制寄存器由标志寄存器（EFLAGS）、指令指针寄存器（EIP）和 4 个控制寄存器 CR0 ~ CR3（Control Register）组成。

1）标志寄存器。标志寄存器的内容如图 3-3 所示。

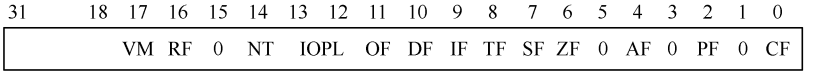


图 3-3　标志寄存器

标志寄存器 0 ~ 11 位为 6 个条件标志、3 个控制标志，其意义与 8086CPU 下的意义相同。12 ~ 17 位为新增加的标志，18 ~ 31 位未用。

IOPL：两位宽的特权级字段，支持保护模式。IOPL 指出要执行的 I/O 指令的特权级。如果当前的特权级别数值上小于等于 IOPL，则 I/O 指令可以执行，否则将发生一个保护。

NT：嵌套任务控制位。NT = 0，IRET 执行一般的中断返回；NT = 1，用一个任务转换代替一般的中断返回。

RF:重启动控制。RF=0 调试故障被接收;RF=1 则忽略调试故障。成功地执行完一条指令,CPU 将 RF 清 0;接收到一个非调试故障 CPU 将 RF 置 1,即忽略该故障。

VM:虚拟 8086 方式位。该位为 1,处理器工作于虚拟 8086 方式。该位为 0,处理器工作于一般的保护模式。

2）控制寄存器 CR_0 ~ CR_3,如图 3-4 所示。

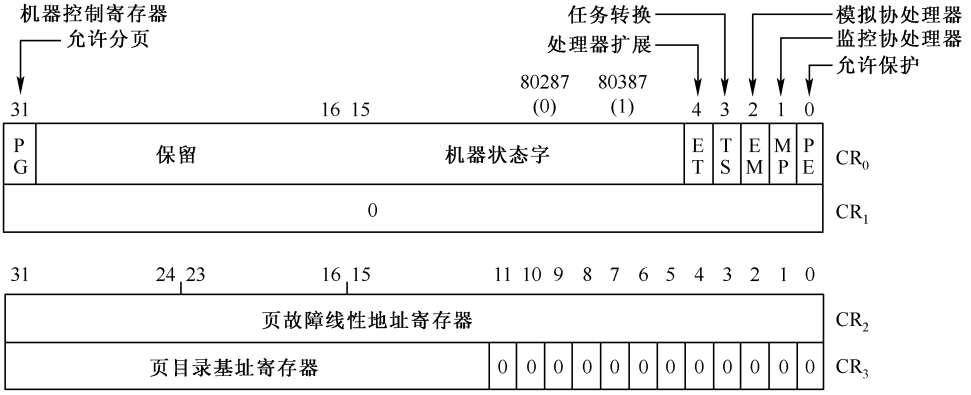


图 3-4　80386 的控制寄存器组

CR_0:0 ~ 15 位称为机器状态字。定义了 6 个标志,其余位保留。

- CR_0 的第 0 位 PE(Protection Enable)为允许保护位,第 31 位 PG(Paging Enable)为允许分页位。组合起来的 4 种操作方式见表 3-2。

表 3-2　PE 与 PG 的组合

PG	PE	方　式
0	0	实模式,8086 操作
0	1	保护模式
1	0	未定义
1	1	“分页保护”环境,允许保护方式下,所有设施都能分页

- CR_0 中的第一位是监控协处理器 MP(Monitor Coprocessor)。它与 TS 位一起决定 WAIT 操作是否产生一个“协处理器不能使用”的出错信号。
- CR_0 中的第 2 位是模拟协处理器 EM(Emulate Coprocessor)。当 EM=1 时,所有协处理器的操作码都产生一个“协处理器不能使用”的出错信号。当 EM=0,所有协处理器操作码均能在 80287 或 80387 上执行。
- CR_0 中的第 3 位是任务转换位 TS(Task Switched)。当一个任务完成后自动将 TS 置 1,随着 TS 置 1,协处理器操作码将会引起一个协处理器不能使用的陷阱。
- CR_0 的第 4 位为处理器扩充类型位 ET(Processor Extention Type)。ET=0 表示系统使用 80287 协处理器;ET=1 表示系统使用 80387 协处理器。

CR_1:未定义控制寄存器,作备用。

CR_2:页保护线性地址寄存器,保护最后出现页故障的 32 位线性地址。

CR_3:页目录基址寄存器,保存页目录表的物理基地址。

5. 系统地址寄存器

80386 有 4 个系统地址寄存器,用来保护操作系统需要的保护信息和地址转换表信息,定义

目前正在执行任务的环境、地址空间和中断向量空间,其名字与作用如下。

1）GDTR:48 位全局描述符表寄存器,用于保存全局描述符表 32 位线性基地址和 16 位全局描述符表界限。

2）IDTR:48 位中断描述符表寄存器,用于保存中断描述符表 32 位线性基地址和 16 位界限。

3）TR:16 位局部描述符表寄存器,用于保存局部描述符表段的选择符。

4）LDTR:16 位局部描述符表寄存器,用于保存局部描述符表段的选择符。

6. 调试寄存器

80386 为调试提供了硬件支持。芯片内设有 $DR_0 \sim DR_7$ 8 个调试寄存器,如图 3-5a 所示。$DR_0 \sim DR_3$ 保存 4 个线性断点地址。DR_4、DR_5 为备用寄存器,目前尚未定义。DR_6 为断点状态寄存器,通过该寄存器的标志可检测事故并进入事故处理程序或禁止进入事故处理程序。DR_7 为断点控制寄存器,用来规定断点字段的长度、段点访问类型、"允许"断点和"允许"所选择的调试条件。

调试寄存器主要为系统程序设计人员准备。

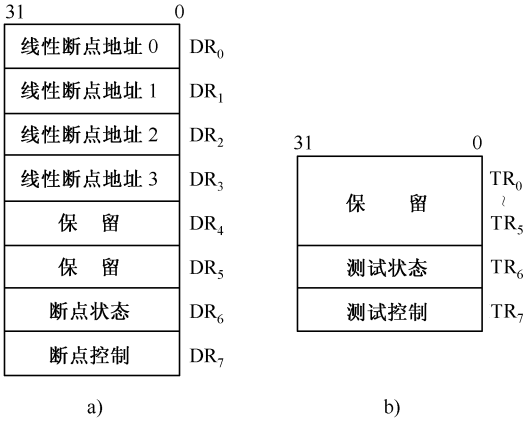


图 3-5 80386 的调试与测试寄存器

a）调试寄存器 b）测试寄存器

7. 测试寄存器

80386 有 8 个 32 位的测试寄存器,如图 3-5b 所示。其中,$TR_0 \sim TR_5$ 保留备用,TR_6、TR_7 用于控制对转换后备缓冲器(TLB)中 RAM 和 CAM(内容可寻址寄存器)的测试,TR_6 是测试命令寄存器,TR_7 为测试数据寄存器,其中保存测试结果的状态。

3.2.4 80386 的工作模式

80386 有 3 种工作模式:实地址模式(real address mode,简称实模式)、保护虚拟地址模式(protected virtual address mode,简称保护模式)、虚拟 8086 模式(virtual 8086 mode,简称虚拟 86 模式)。

1. 实模式

80386 加电启动或复位后自动进入这一模式。实模式的主要功能是初始化 80386,为建立保护模式做准备。在实模式下,80386 的工作模式与 8086 相似,可保持 80386 与 8086 兼容:地址总线仍为 20 位,不用虚拟地址的概念,存储器最大容量仍为 1MB,其寻址机制、存储器管理均与 8086 相同;数据总线为 32 位,数据总线与地址总线是相互独立的,内部寄存器主要作为 16 位使用,操作数默认长度是 16 位,也可以按 32 位使用,这时要在指令加上越权访问前缀;中断处理结构与 8086 相同;80386 具有 4 级特权级,程序运行在最高级(0 级)上,除少数几条指令外,80386 的绝大部分指令均可在实模式下执行。

2. 保护模式

保护模式是 80386 最常用的工作模式,通常在 80386 加电启动或复位后首先进入实模式,完成初始化工作后立即进入保护模式。所谓保护,主要是对存储器的保护,即对存储器中存放的程序和数据的保护。80386 运行在保护模式下,可实现对多任务、多道程序的复杂管理,也只有在保护模式下,80386 才能够真正发挥其强大的功能。

在保护模式下,采用虚拟存储器的概念,存储空间可使用虚拟地址空间、线性地址空间、物理地址空间。通过存储器管理部件,操作系统可以将磁盘等外存设备映射到内存,使程序员可使用的虚拟地址空间(逻辑地址空间)大大超过实际内存的物理地址空间(虚拟地址空间可达64TB)。程序指令的操作数和段内的偏移地址都是32位,地址总线也是32位,物理地址空间为2^{32}B=4GB,但对内存单元的访问要通过一种称为描述符的数据结构才能实现。80386具有4级特权级,可实现程序与程序之间、用户程序与操作系统之间的隔离和保护,为多任务操作系统提供了有效的支持。

3. 虚拟86模式

在虚拟86模式下,不用虚拟地址的概念,存储器最大容量仍为1MB,其寻址机制与8086相同。但存储管理机制与8086不同,它把1MB的存储空间分为256个页面,每页4KB。这时,当多道程序同时运行时,可以使其中的一个或多个任务使用虚拟86模式,并使某一个任务占用存储器的某些页面,而另一个任务占用存储器的另外一些页面,这样就可将多个任务分别转换到物理存储器的不同存储位置,实现了多任务同时运行。在虚拟86模式下,程序运行在最低特权级(3级)上,这时80386的一些特权指令是不能使用的。

80386的上述3种工作模式可以相互转换。在实模式下,通过LMSW或数据传送指令,将控制寄存器CR_0的第0位(即PE,允许保护控制位)置为1,即可进入保护模式。通过数据传送指令,将PE置为0,即可从保护模式返回到实模式。在保护模式下,通过执行IRETD指令或进行任务转换,可以进入虚拟86模式。通过中断操作,可以从虚拟86模式转换到保护模式。

3.2.5 80386的存储管理

多用户/多任务的操作系统必须提供存储管理手段,以便把系统的存储器有效地分配给不同的任务。操作系统还应提供有效的保护手段,来隔离和保护每一个任务。80386片上MMU采用页和段两种存储管理机制,为操作系统完成上述两方面的任务提供了有力支持。本节的目的在于简单地剖析80386两级存储管理与保护机制。

1. 逻辑地址与段选择符

在介绍地址转换时曾经提到,程序提供的逻辑地址包括偏移地址和段选择符两部分。段选择符的格式如图3-6所示。其中RPL为本程序段的特权级,其取值为0~3。T_1为表指示器,T_1=1指向属于某一任务的局部段描述符表;T_1=0指向属于系统的全局段描述符表。段描述符指针提供一个偏移地址,指向段描述符表中的一个表项,即一个段描述符。

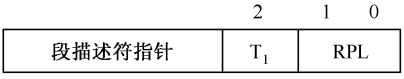


图3-6 段选择符的格式

2. 段描述符的格式及保护功能

段描述符的格式如图3-7所示。每个描述符占8字节(64位),描述符的内容是分段放置的。例如段限共占20位,分0~15位、16~19位两部分放置,可指示本段大小,最大为1MB空间。段限、存取控制和语义控制的内容可用于存储保护。

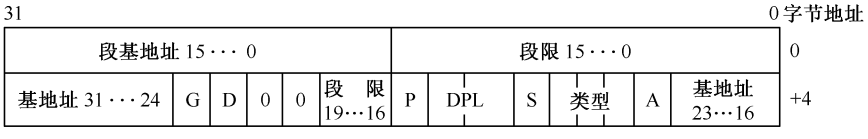


图3-7 段描述符的格式

段限的保护作用。一个应用程序产生的偏移地址大小只能在段限之内,如果偏移地址的大小大于段限,则出现一个异常保护,防止本程序段进入其他程序的工作区。存取控制字段包括A、类型、S、DPL、P 等项。其中,A 用于存储管理,说明本段是否已被访问;类型指明本段的类型,如代码段、数据段等;S 用于指明本段为系统段或非系统段;DPL 指明本段的特权级;P 指示本段是否在内存中。特权级也可用于存储保护,运行在某一特权级上的程序只能访问不高于本身特权级的段。特权级的转换要通过专用描述符提供。

段描述符中,G、D 位称为语义控制位。G = 1,段长以页面为单位;G = 0,段长以字节为单位。D = 1 为 32 位代码段,即运行于 32 位机方式;D = 0 为 16 位代码段,即运行于 16 位机方式。

3. 分页存储管理及保护

分页存储管理有两级表结构,一个为页表目录表,一个为页表。两个表均占 4KB 空间,每个表项都是 4B,所以,一个页表目录表映射为 1024 个页表,每个页表映射为 1024 页。两个表的表项分别称为页目录描述符和页表描述符,其格式如图 3-8 所示。页目录描述符和页表描述符的内容相同。P 为存在位,表示页或页表是否真正装入内存,P = 1 表示已在内存;P = 0 表示不在内存。W/R 为写/读位,W/R = 1 表示可读写出;W/R = 0 表示只读。U/S 为用户监控位,U/S = 1 表示用户占用;U/S = 0 表示监控占用。A 位和 D 位可以跟踪一页的使用情况。对一页进行读出,自动将 A 置 1,而写一页时将 D 置 1。通过查询 A、D 位,操作系统可处理页的替换。

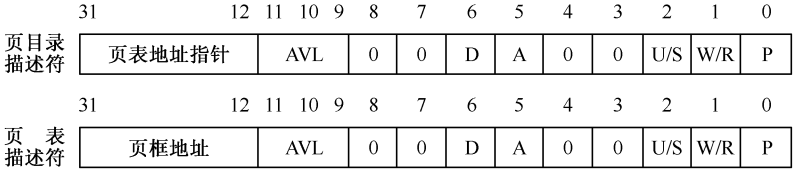


图 3-8　页目录描述符和页表描述符格式

分页机制也可以在页一级提供保护,根据页表目录和页表描述符中 U/S 和 W/R 位的情况决定监控程序和用户程序的存取权限,其关系见表 3-3。只有两个描述符 U/S、W/R 位都是 1 时才允许用户读/写。

表 3-3　用户/监控页保护

页目录描述符		页描述符		用　户	监　控
U/S	W/R	U/S	W/R		
0	0	0	0	无操作	
1	0	0	1	无操作	读/写
1	0	1	0	只　读	读/写
1	0	1	1	只　读	读/写
1	1	0	0	无操作	读/写
1	1	0	1	无操作	读/写
1	1	1	0	只　读	读/写
1	1	1	1	读/写	读/写

80386 提供了强有力的存储器寻址机构,这主要由片上的存储管理单元(MMU)完成。地址转换的过程如图 3-9 所示。用户程序中所使用的地址称为逻辑地址,80386 支持高达 64TB 的逻辑地址空间,而 80386 可寻址的物理空间为 4GB。由逻辑地址找物理地址的过程,称为地址转换。逻辑地址由两部分组成:低 32 位为偏移地址,可指向 4GB 空间中的任何地址;高 16 位为选择符,指向段描述符表(段描述符表由操作系统管理)的一个表项,即一个段描述符。段描述符

给出一个段基地址,该段基址与偏移地址相加,产生线性地址。当不采用分页机制时,该线性地址就可用作存储器的物理地址,即出现在地址总线上的地址。

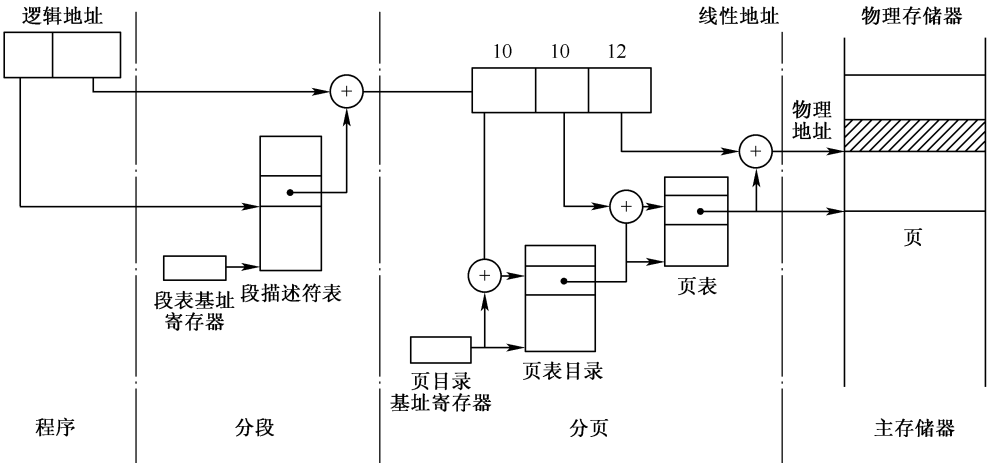


图 3-9　二级分页地址转换

当采用分页机制时,线性地址通过分页机构再转换成物理地址。线性地址共 32 位,分成 3 部分。高 10 位为页表目录索引,中间 10 位为页表索引,低 12 位为一页内的索引,即一页内的相对地址。页表目录索引选中页表目录中 1024 个页表中的一个。页表目录的基地址在页目录基址寄存器 CR_3 中,页表索引选中页表中的一个表项,即选中页在物理空间的基地址,该基地址与页内索引(即指针)相加,获得最后输出在地址总线上的物理地址。

3.2.6　80386 中断

与 8086 相同,80386 系统也可以处理 256 种类型的中断源,中断类型号也为 0 ~ 255 号。引起 80386 中断的中断源也可分为硬件中断和软件中断,其中,硬件中断是指外部设备通过 INTR 或 NMI 引脚向 80386 请求的中断;软件中断是指由于 80386 执行 INT 指令或 CPU 在执行指令时产生异常故障或协处理器执行指令时产生异常故障引起的中断。

80386 的中断管理机制在实模式和保护模式下是不同的。在实模式下,80386 处理中断的方法与 8086 完全相同。另外,除类型号为 0 ~ 4 的中断外,还增加了如下类型号为 5 ~ 9、12、13、16 共 8 个专用中断:类型 5 为 BOUND 指令故障中断,类型 6 为非法指令故障中断,类型 7 为无协处理器中断,类型 8 为 IDT(中断描述符表)小故障中断,类型 9 为浮点段溢出故障中断,类型 12 为堆栈段溢出故障中断,类型 13 为一般段溢出故障中断,类型 16 为协处理器异常故障中断。其余问题在此不再讨论,请读者参考“第 7 章 输入/输出和中断”。

在保护模式下,中断描述符表寄存器(IDTR)存放中断描述符表(IDT)的基地址和界限值。中断描述符表(IDT)类似于实模式下的中断向量表,但其中的每一个描述符占用 8 字节,除了提供中断服务程序的入口地址外,还包含了更多的中断信息。另外,除类型号为 0 ~ 7、12、17 的中断与实模式下的中断相似外,在保护模式下,又增加或修改了如下 5 个中断:类型 8 为双故障中断(即在进行一个异常故障中断处理时,又产生了另外的异常故障中断,这种中断循环情况容易造成死机,采用双故障中断就是打破中断循环),类型 9 留作备用,类型 10 为任务状态段无效故障中断,类型 11 为段不存在异常故障中断,类型 13 为保护异常故障中断,类型 14 为页故障中断。

3.3 80486 微处理器简介

Intel486 微处理器是第一个采用 RISC(精简指令集计算机)技术的 80X86 系列微处理器,它通过减少不规则的控制部分,缩短了指令的执行时间。在 486 芯片上包括整数处理单元、浮点处理单元、存储管理单元和高速缓存。由于许多单元集成在单片上,许多信号只在芯片内以芯片能达到的高速传送,而不是在印制板上传输,所以大大提高了信号处理的速度。它还大大减少了电路板的面积,降低了成本,简化了系统设计。

由于时钟速度和一些专门处理,486 的性能比 386 可高 2 ~4 倍。具体体现在:

1)虽然 486 和 386 都是 32 位总线宽度,由于 486 总线使用 3 ×1 时钟频率(即 486 内部时钟与外部晶振频率相同)、成组(burst)总线周期、高速缓存周期及高速缓存失效周期、8 位数据总线等方式,使其总线速度远快于 386。

2)成组总线周期能在一个时钟周期内传送 32 位数据,而 386 至少要两个时钟周期。

3)执行指令所需时钟周期数少于 386,486 的指令大部分能在一个时钟周期内完成。

下面就 486 微处理器的特点及内部结构作简单介绍。

3.3.1 80486 的主要特点

1)兼容性。486 在二进制代码的水平上与以前的 80X86 处理器完全兼容。

2)全 32 位的整数处理器。该处理器使用 ALU 单元和 8 个通用寄存器完成 8 位、16 位、32 位宽度的全部算术逻辑运算。

3)独立的 32 位地址、数据总线,可直接寻址 4GB 的物理地址空间。

4)单时钟周期执行。许多指令在一个时钟周期内执行完成。

5)片上浮点处理单元支持 32 位、64 位和 80 位的浮点运算,在二进制上与 8087、80287、80387 兼容。

6)片上存储管理单元。支持存储分段和分页机制,地址管理和存储空间保护机构可以保证内存的可靠性与完整性,这在多任务系统虚拟存储环境下是必不可少的。

7)带有高速缓存支持系统的片上高速缓存。片上 8KB 高速缓存可用于存储指令和数据。读写高速缓存像读写内部寄存器一样快,总线只用于跟踪、更新内部高速缓存。

8)外部 CACHE 控制。回写和擦除(Flush)控制整个外部 CACHE,使得处理器在多 CPU 环境下能跟踪 CACHE。

9)指令流水线。取指令、执行指令、地址转换工作并行进行,这使得大部分指令能在一个时钟周期内完成。

10)成组周期(Burst Cycle)。成组传送使得每个时钟周期内能读入双字数据。由于这个功能,内部 CACHE 和指令预取缓冲器能够很快被充满。

11)写缓冲器。由于有写缓冲器的存在,CPU 在内部可以连续输出数据,而不管这些数据是否已经从数据总线上输出了。

12)总线背关(Bus Backoff)。如果在 486CPU 一个总线周期的中间另外的处理器需要占用总线,486CPU 可以浮空其总线,当总线空闲后再重新开始其总线周期。

13)指令重新执行。当访问存储器意外失败后,指令可继续执行,这个功能对于请求虚拟存储应用十分重要。

14）总线宽度动态可变。外部控制器可动态地将总线宽度改为 8 位、16 或 32 位。

3.3.2 80486 的内部结构

486 微处理器的内部结构如图 3-10 所示。有 9 个功能单元并行工作，这些单元包括总线接口单元、高速缓存（CACHE）、指令预取单元、指令译码单元、控制单元、整数（数据通路）单元、浮点单元、分段单元和分页单元。其内部结构除高速缓存、浮点单元外和 386 非常相似。32 位外部信号通过总线接口单元进入处理器内部。在内部总线接口单元和 CACHE 间双向传送 32 位地址。32 位数据通过 CACHE 和总线接口部件送上外部数据总线。和 CACHE 紧耦合的指令预取单元同时接收从总线接口单元传送来的预取指令，CACHE 还接收指令操作数和其他数据。指令预取单元也可以从 CACHE 存取指令，指令预取单元中保存 32B 的指令队列等待执行。

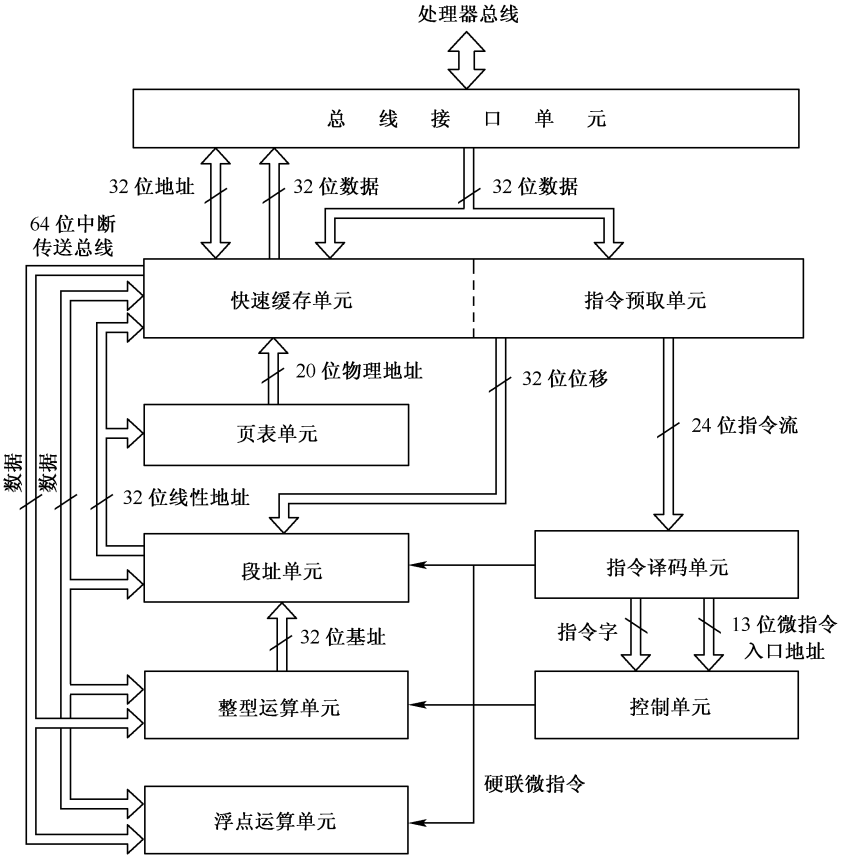


图 3-10 80486 的内部结构

当内部请求指令或数据从 CACHE 得到满足时，处理器的外部总线空闲下来，只有内部操作需要总线存取时，才调用总线接口部件。许多内部操作对外部是透明的，指令译码单元将指令译码成低级控制信号和微代码入口指针。控制单元执行微代码，并控制整数单元、浮点单元和段单元。计算结果放入整数单元和浮点单元的内部寄存器或 CACHE 中。内部存储地址保存在整数单元中。CACHE 与整数单元、段机构和浮点单元共享两组 32 位数据总线，这两组数据总线可合在一起用作 64 位内部传输总线。当 64 位段描述符从 CACHE 传向段单元时，其中 32 位通过一组数据总线，另外 32 位通过整数单元。因此，64 位同时到达段单元。地址由段单元和页单元产

生逻辑地址,由段单元转换并通过 32 位线性地址总线送向页单元和 CACHE。页单元将线性地址转换成物理地址,通过 20 位物理地址总线送向 CACHE。

下面先介绍指令流水线,然后逐项介绍 9 个内部单元。

1. 指令流水线

并不是每条指令都要调用所有的内部单元,当一条指令需要几个单元参与时,各个单元则根据指令执行的不同阶段与其他单元并行工作。虽然每条指令是顺序执行的,但在任一时刻,在处理器内部总有几条指令处于不同的执行阶段,这就称为指令流水线(Instruction Pipelining)。指令预取、指令译码、微代码执行、整数操作、浮点操作、分段、分页、CACHE 管理和总线接口操作等都是同时进行的。图 3-11 表示对于一条指令这些并行操作的各个阶段:取指令、二阶段译码、执行和执行结果写回寄存器。每个时钟周期可完成一个阶段。内部流水线操作使 486 处理器在性能上超过许多压缩指令系统计算机(RISC)。在 486 处理器中,数据可以由一条指令从 CACHE 装入而在下一个时钟周期中由下一条指令使用。这个优点来自第一阶段译码,它在执行周期之前先进行存储器存取操作。因为大多数编译程序和应用程序往往使紧跟着装入指令之后的指令对装入的数据进行操作,这种方法优化了程序代码的执行。

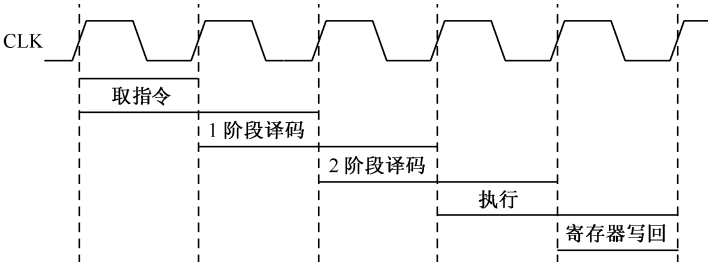


图 3-11　80486 的内部流水线

2. 总线接口单元

总线接口单元用于数据传输、指令预取和处理器内部单元与外部系统的控制功能。在内部,总线接口单元与 CACHE 及指令预取单元通过 3 个 32 位总线进行通信(如图 3-10 所示)。在外部,总线接口单元提供处理器总线信号。除周期定义信号外,所有外部总线周期(存储器读、指令预取、高速缓存的装入等)都具有相同的总线定时。总线接口单元具有以下结构特点:

1)地址收发和驱动。地址信号 $A_2 \sim A_{31}$ 与相应的字节使能信号发送到处理器的总线上。高 28 位地址信号是双向的,使得外部逻辑驱动 CACHE 失效的地址能够进入处理器。

2)数据总线收发。$D_0 \sim D_{31}$ 数据信号发向数据总线或从总线上接收 $D_0 \sim D_{31}$。

3)总线宽度控制。外部数据总线可使用 32 位、16 位和 8 位的总线宽度,由外部逻辑的两个输入信号指定要使用的总线宽度。总线宽度可按总线周期改变。

4)写缓冲。可对多达 4 个总线写请求进行缓冲,因此使许多内部操作可连续执行而不用等待上一个总线写周期完成。

5)总线周期和总线控制。支持大量的总线周期选择与控制,包括成组传送、非成组传送、总线仲裁(总线请求、总线回答、总线伪封锁和总线背封)、浮点错符号指示、中断和复位等。两个软件控制的输出使页缓冲能按总线周期进行,一个输入和一个输出信号用于控制成组读传送。

6)奇偶性的产生和控制。对输出信号加上奇偶性,对读入信号进行奇偶校验。

7)CACHE 控制。支持高速缓存控制和一致性操作。3 个输入信号使得外部系统能控制存储

在 CACHE 中数据的一致性。两个专用的总线周期使处理器可控制外部 CACHE 的一致性。

3. 高速缓冲(CACHE)单元

CACHE 单元存储当前读入的指令、操作数及其他数据的副本。当处理器请求的信息已经在 CACHE 中时(称 CACHE 命中),则不需要总线周期。当处理器请求信息不在 CACHE 中(称 CACHE 未命中),这个信息以一个或多个 16B 的形式读入 CACHE(称 CACHE 填充)。当产生一个内部写请求时,CACHE 被写入,并通过 CACHE 写向存储器,这称为 CACHE 写通过。

如图 3-10 所示,CACHE 通过两个 32 位总线传输数据,CACHE 在 32 位总线上接收线性地址,相应的物理地址送上 20 位总线。CACHE 和指令预取紧密相连,CACHE 中的 16B 指令组能迅速地送入指令预取单元。每个时钟周期都可对 CACHE 进行存取。当 CACHE 的功能和写通过功能被禁止时,CACHE 可用作高速 RAM。

4. 指令预取单元

当指令执行中不使用总线周期时,指令预取单元就通过总线接口单元预取指令。用指令预取的方法,处理器很少需要等待指令读入。指令预取单元每次依次读入 16B 的指令组,起始地址就由指令预取单元产生。预取单元直接和分页单元相连(图 3-10 中未表示出)。16B 的指令组同时被读入 CACHE 和预取单元,预取单元中的指令队列存放 32B 的指令组。每条指令从队列中取出,其操作码部分送入指令译码单元,偏移地址部分送入段单元,在段单元中进行寻址计算。如果指令中遇到循环指令,则预取单元从 CACHE 复制先前执行指令的副本。

预取单元具有总线存取最低的优先级。假如是 0 等待的存储器存取,从来不会因预取单元延迟指令的执行。当下一条指令不是顺序执行指令时(如跳转指令、任务切换、异常或中断),预取单元则被清除。

预取单元不会取程序结束(END)之后的指令,也不会存取不存在的页。但预取可能引起某些硬件问题,例如当预取接近存储器末端时会引发中断。为了使预取不要超过某个地址单元,指令至少距那个单元有(16 +1)个单元的距离。

5. 指令译码单元

指令译码单元从指令预取单元接受指令,在两个阶段中(如图 3-11 所示)将其译码成低级控制信号和微代码入口指针(如图 3-10 所示)。大部分指令能在一个时钟周期内译码完成。在译码阶段 1 处理存储器存取,这使得两条指令的装入和执行能在两个时钟周期内完成。译码单元同时处理指令前缀字节、操作码、模式 R/M 和偏移地址。译码单元输出包括对段单元、整数单元和浮点单元的硬件微指令。当指令预取单元清除时,指令译码单元也被清除。

6. 控制单元

控制单元的功能是解释指令字和从译码单元获得的微代码入口指针。控制单元的输出可控制整数单元和浮点单元。由于段选择由指令指定,所以控制单元还控制段处理单元。控制单元包含处理器的微代码,许多指令只有一条微代码,所以平均能在一个时钟周期内执行完成。图 3-11 所示说明了执行操作在内部流水线结构中的位置顺序。

7. 整数(数据通路)单元

数据在整数单元中存储并完成 386 处理器指令及几条新增指令的所有算术逻辑运算。该单元有 8 个 32 位通用寄存器、几个专用寄存器、ALU 单元和一个桶形移位器。数据的装入、存储、加、减、移位等运算和操作都能在一个时钟内完成。

两组 32 位双向总线将整数单元和浮点单元联系起来,这些总线合起来可以传送 64 位操作数。这组总线还将处理器单元与 CACHE 联系起来,通用寄存器的内容是通过一组总线传向分

段单元并用于产生有效地址。

8. 浮点单元

浮点单元执行协处理器 387 同样的指令组。本单元包括下压寄存器堆栈和特殊硬件用以解释 32 位、64 位和 80 位的 IEEE754 标准的浮点格式。一个输出信号通向处理器总线,用于向外部系统指示浮点错误,外部系统能插入一个输入,以指示处理器忽略这个错误并继续正常的操作。

9. 分段单元

段是一个被保护的独立地址空间,分段用于加强应用程序间的隔离。分段单元将程序发出的逻辑地址转换成线性地址,并将此线性地址发向分页单元和 CACHE。当第一次存取一个段,其段描述符就被复制到处理器寄存器中。多达 6 个段描述符能够同时存于处理器的寄存器中。

10. 分页单元

分页单元用把程序和数据一部分存在存储器中、一部分存在磁盘上的方法,能够存取的数据结构远大于实际的物理空间。分页单元将线性地址分成 4KB 大小的空间,称为页。分页机构使用内存中的页表完成线性地址到物理地址的映射。当存取一个不在内存中的页时,分页机构引起一个异常错误,操作系统借处理这一错误的机会将需要的页从磁盘调入内存,必要时还可将某些页写回磁盘以释放所占用的内存空间。如果不采用分页机构,则物理地址与线性地址相同。

3.4 Pentium 微处理器简介

随着人们对图形图像、实时视频处理、语音识别、CAD/CAE/CAI、大规模财务分析和大流量客户机/服务器应用等的需求日益迫切,原有的微处理器已难胜任此类任务。由此,第五代微处理器 80586 应运而生。80586 是继 486 之后 80X86 系列的又一代新产品,随着 586 的不断改进和功能扩充,新型号芯片不断推出,人们习惯上已不再使用 586 这个名称,而称为 Pentium。Pentium 系列目前已推出了其第四代产品,称为 Pentium 4,其主频目前已达到 3.2GHz。

Pentium 芯片的集成度达到 310 万个晶体管/片以上,原来被置于片外的部件如数学协处理器和 CACHE 等可以集成到 CPU 芯片内,因而显著提高了处理速度。此外,为达此目的,Pentium 还采用了特殊 CAD 方法设计的多级金属夹层技术。虽然 Pentium 采用了许多新的设计方法,但仍与过去的 X86 系列 CPU 兼容。

3.4.1 Pentium 体系结构的特点

在 CPU 设计中,单靠增加芯片的集成度还不足以提高 CPU 的整体性能。为此,Intel 在 Pentium 的设计中采用了新的体系结构,如图 3-12 所示。Pentium 新型体系结构的特点可以归纳为以下 4 个方面。

1. 超标量流水线

超标量流水线(Superscalar)设计是 Pentium 处理器技术的核心。它由 U 和 V 两条指令流水线构成,如图 3-13 所示。每条流水线都拥有自己的 ALU、地址生成电路和数据 CACHE 的接口。这种流水线结构允许 Pentium 在单个时钟周期内执行两条整数指令,比相同频率的 486DX CPU 性能提高了一倍。与 486 流水线相类似,Pentium 的每一条流水线也分为 5 个步骤,即指令预取、指令译码、地址生成、指令执行、回写。当一条指令完成预取步骤后,流水线就可以开始对另一条指令的操作。

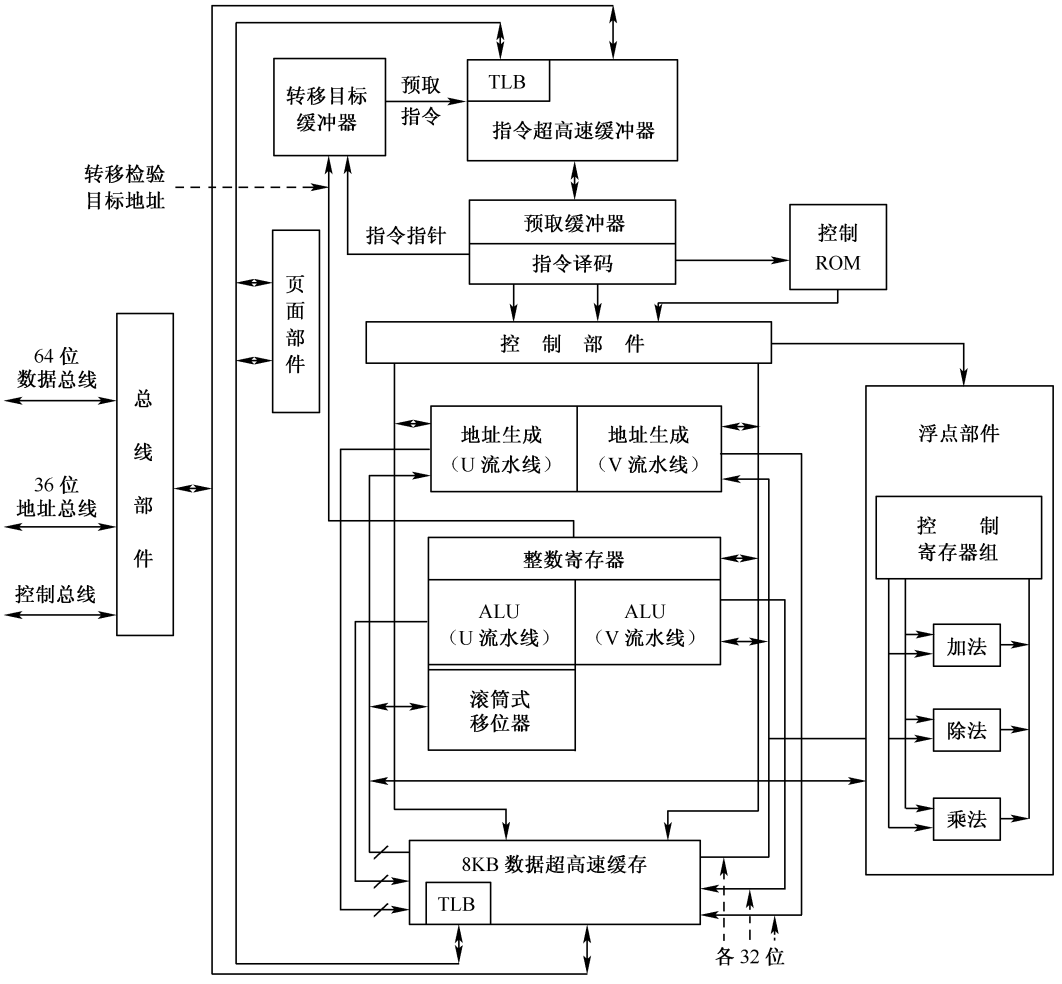


图 3-12　Pentium 体系结构示意图

但与 486 不同的是，由于 Pentium 是双流水线结构，它可以一次执行两条指令，每条流水线中执行一条。这个过程称为“指令并行”。在这种情况下，要求指令必须是简单指令，且 V 流水线总是接受 U 流水线的下一条指令。但如果两条指令同时操作产生的结果发生冲突时，则要求 Pentium 还必须借助于适当的编译工具产生尽量不冲突的指令序列，以保证其有效使用。例如下列 4 条指令：

（1）MOV AX,5　　（2）INC BX　　（3）MOV AX,5　　（4）INC AX

其中，(1)、(2)两条指令可以并行工作，而(3)、(4)两条指令则不行，因为它们都在同一个寄存器 AX 中进行操作，将引起结果冲突。

2. 独立的指令 CACHE 和数据 CACHE

80486 片内有 8 KB CACHE，而 Pentium 则为两个 8KB，一个作为指令 CACHE，另一个作为数据 CACHE，即双路 CACHE 结构，如图 3-14 所示。

图中，TLB 的作用是将线性地址翻译成物理地址。指令 CACHE 和数据 CACHE 采用32B ×8 线宽(486DX 为 16B ×8 线宽)，这是对 Pentium 64 位总线的有力支持。

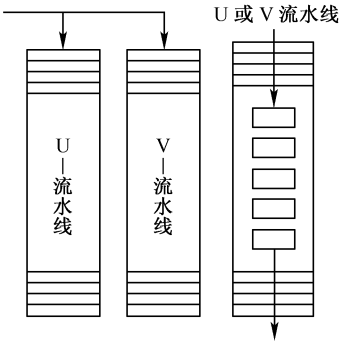


图 3-13　Pentium 超标量流水线结构

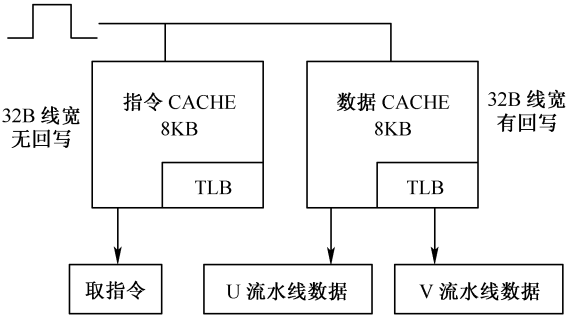


图 3-14　双路 CACHE 结构

Pentium 的数据 CACHE 有两个接口,分别通向 U 和 V 两条流水线,以便能在相同时刻对两个独立工作的流水线进行数据交换。当向已被占满的数据 CACHE 写数据时(也只有在这种情况下),将移走一部分当前使用频率最低的数据,同时将其写回主存。这个技术称为 CACHE 回写技术。由于处理器向 CACHE 写数据和将 CACHE 释放的数据写回主存是同时进行的,所以,采用 CACHE 回写技术大大节省了处理时间。

指令和数据分别使用不同的 CACHE,使 Pentium 的性能大大超过 486 微处理器。例如,流水线的第一步骤为指令预取,在这一步中,指令从指令 CACHE 中取出来,如果指令和数据合用一个 CACHE,则指令预取和数据操作之间将很可能发生冲突。而提供两个独立的 CACHE 将可避免这种冲突并允许两个同时操作。

3. 重新设计的浮点单元

Pentium 的浮点单元在 486 的基础上进行了彻底改进,其执行过程分为 8 级流水,使每个时钟周期能完成一个浮点操作(某些情况下可以完成两个)。

浮点单元流水线的前 4 个步骤与整数流水线相同,后 4 个步骤的前两步为二级浮点操作,后两步为四舍五入及写结果、出错报告。Pentium 的 CPU 对一些常用指令如 ADD、MUL 和 LOAD 等采用了新的算法,同时,用电路进行了固化,用硬件来实现,其速度的提高是显而易见的。

在运行浮点密集型程序时,66MHz Pentium 的运算速度为 33MHz 的 80486 的 5 ~ 6 倍。

4. 分支预测

循环操作在软件设计中使用十分普遍,而每次在循环当中对循环条件的判断占用了大量的 CPU 时间。为此,Pentium 提供一个称为分支目标缓冲器 BTB(Branch Target Buffer)的小 CACHE 来动态地预测程序分支,当一条指令导致程序分支时,BTB 记下这条指令和分支目标的地址,并用这些信息预测这条指令再次产生分支时的路径,预先从此处预取指令,保证流水线的指令预取步骤不会空置,BTB 机制如图 3-15 所示。

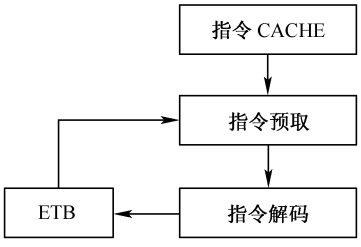


图 3-15　Pentium 的 BTB 机制

当 BTB 判断正确时,分支程序即刻得到代码,从循环程序来看,在进入循环和退出循环时,BTB 会发生判断错误,需重新计算分支地址。循环 10 次,2 次错误,8 次正确;循环 100 次,2 次错误,98 次正确。因此循环越多,BTB 的效益越明显。

3.4.2　Pentium 相对 80486 体系结构的增强点

除了以上几个特点外,Pentium 微处理器在 486 体系结构基础上,还作了一些增强性的改进,

归结为以下几点。

1. 工作频率提高

目前486系列中工作频率最高者为486DX－50，为50MHz。486DX2－50和486DX2－66内部工作频率分别为50MHz和66MHz，但由于采用的是倍速技术，其外部工作频率实际只有25MHz和33MHz。而Pentium的内部和外部工作频率一致，分别达到数百MHz甚至上千MHz。频率加快，运算速度自然也得到提高。

2. 指令固化

在Pentium中，常用指令如MOV、INC、DEC、PUSH、POP、JMP、CALL(NEAR)、NOP、SHIFT、ROT、NOT和TEST等改用硬件实现，不再使用微程序操作，使指令的运行得到进一步加快。而其他的微程序指令由于运行于双流水线上，速度也得到了提高。

3. 页尺寸增加

Pentium体系结构中，存储器中每一页的容量除了与486兼容的4KB外，还多加了一页。页尺寸的增加使程序在传送大块数据如图形的FRAME BUFFER时，避免了频繁的换页操作。

4. 增强的微指令

Pentium指令系统的微指令算法作了重大改进，使指令执行所需时钟周期相对于486大大减少。

5. 增强的总线

从图3-12可以看出，Pentium的内部总线同486一样为32位，但它通向存储器的外部总线为64位宽，在一个总线周期内，将数据传输量增加了一倍。Pentium还支持多种类型的总线周期，其中包括一种突发模式，该种模式下可以在一个总线周期装入256位数据。64位的数据总线使Pentium与主存数据交换速度达到528Mbit/s，而主频为50MHz的486，运行速度为160Mbit/s，前者是后者的3倍多。除以上特点外，Pentium在数据完整性、容错性和节电等方面也采取了一些特殊的设计方法。

从以上介绍中可以看出，Pentium作为新一代微处理器，它为微处理器体系结构和个人计算机性能引入了全新的概念，为今后微处理器和个人计算机的发展开辟了一个新的技术方向。

3.4.3 Pentium Ⅱ微处理器

Pentium Ⅱ微处理器也称为奔腾二代，简称PⅡ，是Intel公司于1997年推出的新一代微处理器。它是Pentium Pro的改进型产品，在核心结构上并没有多大变化。

PⅡ采用了一种称之为双独立总线(Dual Independent Bus，DIB)结构，即二级高速缓存总线和CPU－主内存总线。这种双独立总线结构比单总线结构的处理器在带宽处理上性能得到较大提高。

PⅡ处理器与主板的连接首次采用了Slot 1接口标准，它不再使用陶瓷封装，而是采用了一块带金属外壳的印制电路板，该印制电路板集成了处理器的核心部件以及32KB的一级高速缓存。它与一个称为单边接触卡(Single－Edge Contact，SEC)的底座相连，再套上封装外壳，形成了完整的CPU部件。SEC是Intel公司的一个创新的包装设计，它将CPU的核心部件和二级高速缓存集成组装到一个塑料及金属的卡式盒中。这些部件可以和卡式盒中的基座直接通信，使其能进行高频率操作。

PⅡ中一级高速缓存的容量较Pentium Pro提高了一倍，即指令高速缓存和数据高速缓存各为16KB。这样，由于提高了一级高速缓存的命中率，可减少访问二级高速缓存的次数，从而提高

了 CPU 的运行速度。

与 PⅡ相比,原来的 Pentium Pro 有一个很大的弱点,即在 16 位操作系统下对段寄存器的写操作速度很慢,这是由于它在运行 16 位软件时经常需要对段寄存器进行更新,而一次段寄存器的更新会使其写操作的时间大约延迟 30 个时钟周期。PⅡ由于配备了可重命名的段寄存器,即在写入段寄存器的同时进行段寄存器重新命名,因而加快了段寄存器写操作的速度。

PⅡ在两个 ALU 中增加了 MMX 指令的处理电路,可对多媒体应用中频繁使用的 8 位或 16 位数据进行并行处理。

PⅡ处理器还具有功耗控制功能,对不用的电路停止提供时钟信号,使额定功耗大幅度下降。

3.4.4 Pentium Ⅲ 微处理器

PentiumⅢ (PⅢ,奔腾Ⅲ)是 Intel 公司 1999 年推出的微处理器芯片。PⅢ仍采用了同 PII 一样的内核,制造工艺为 0.25μm 或 0.18μm 的 CMOS 技术,有 850 万个晶体管,主频最高可达 750MHz 以上。

PⅢ处理器具有片内 32KB 的一级高速缓存和 512KB 的片外二级高速缓存,可访问内存达 64GB。PⅢ对高速缓存和主存的存取操作以及内存管理更趋合理,PⅢ可预先读取期望使用的数据到高速缓存,从而提高了高速缓存的命中率,加快了运行速度。

为了进一步提高 CPU 处理数据的功能,PⅢ增加了被称为流水式单指令多数据扩展(Streaming SIMD Extension,SSE)指令集。PⅢ新增加了 70 条 SSE 指令,可以分成以下 3 组不同类型的指令。

1) 8 条内存连续数据流优化处理指令:通过采用新的数据预存取技术,减少 CPU 处理连续数据流的中间环节,提高了 CPU 处理连续数据流的效率。典型的连续数据流有数据库访问、视频数据流、音频数据流等。使用这种 CPU 之后,计算机在进行视频处理、音频处理时,速度会更快,质量会更好,视频播放更流畅,画面更清晰,色彩更鲜艳,音质也更好。

2) 50 条单指令多数据浮点运算指令:每条指令一次可处理多组浮点运算数据。以前的指令一次只能处理一对浮点运算数据,现在一次可以处理 4 对数据,从而大大提高了浮点数据处理的速度。

3) 12 条新的多媒体指令:采用改进算法,进一步提高了视频处理、图片处理的质量。

PⅢ的另一个特点是每个处理器都拥有一个唯一的 128 位序列号,从而可实现对使用该处理器的 PC 进行标识。使用处理器序列号的目的在于进一步提高信息的安全性。在互联网,尤其是在电子商务的应用中,安全性是一个突出的问题。由于 PC 使用的微处理器没有序列号,因此只要知道了用户名和密码,在任何一台 PC 上都可以访问有关信息,从而使信息安全难以保障。而有了微处理器的序列号与用户名和口令的配合,只有在用户自己的计算机上键入正确的用户名和密码后才能访问有关信息。

3.4.5 Pentium 4 微处理器

Pentium 4(P4,简称为奔 4)是 Intel 公司 2001 年推出的微处理器芯片。P4 的原始代号为 Willamette,它采用 0.18μm 铝导线微米工艺和低温 K 型半导体电介质技术,是一个具有超级深层次管线化结构的微处理器。P4 芯片有 478 个引脚,集成度达 4200 万个等效晶体管。由于 P4 内部晶体管很多,集成度很高,耗电量就很大(如主频为 1.4GHz 的 P4 耗电功率达 55W),所以它

工作时发热量也很大,必须使用较大的金属散热片,并配有散热风扇。

P4 的核心结构设计了两组独立工作的 ALU,采用双倍频算术逻辑单元构架,在一个时钟周期的上升沿和下降沿进行计算,一个时钟周期完成两次算术逻辑运算。也就是说,一个时钟周期可以执行两条算术逻辑指令,其程序执行速度大大地提高。P4 还采用了 4 倍爆发式总线(Quad Pumped Bus)技术。因为一次算术逻辑运算需要两个操作数,一个时钟周期完成两次运算就需要 4 个操作数,采用 4 倍爆发式总线技术,可以在一个时钟周期内同时传送 4 组 64 位数据,同时通过使用 850 芯片组和双通道 Dual DRDDRAM 主存,使内存的带宽提高到3.2GB/s, 以达到内存与 P4 CPU 速度的匹配。

随着多媒体应用对 CPU 的要求越来越高,P4 在 PⅢ原有的 SSE 指令集基础上,又新增加了 76 条指令。这样,P4 在增强浮点运算能力的同时,也提高了内存的效率和速度,改善了 P4 在图形和图像处理、视频编辑、数据压缩/解压等领域的应用。

P4 对 PⅢ的结构方面的一个重大改进是将指令 CACHE(I-CACHE)和数据 CACHE(D-CACHE)分开,并将 I-CACHE 直接连接到分支预测单元与执行单元,这样在执行重复性程序代码时,I-CACHE 可以减少对程序指令反复译码的频率,加快了程序的执行速度。

3.5 微处理器的发展

受多处理机系统的启示,微处理器由单核向多核发展,尤其是当前智能手机的应用越来越广泛,应用的需求也推动了微处理器日新月异的发展。

3.5.1 微处理器由单核向多核发展

在较难提高 CPU 主频的客观环境,受计算机分布式系统的启示, 计算机科学工作者设计了一个微处理器芯片,其内配置了多个功能一样的处理器核心,这就是多核微处理器。多核微处理器的问世有利地推动了并行程序、并行计算处理的发展,提高总体程序的执行速度,降低了能耗,受到人们的欢迎。多核微处理器技术使微处理器性能更高,处理信息能力更强,信息处理技术更先进。对当前网络新宠“云计算”技术的发展起了关键性作用,其 Intel 至强 E5-2600 微处理器芯片就是专为“云计算”设计的。

1. 双核微处理器

所谓双核微处理器就是在一个微处理器芯片内配置了两个相同的处理器核心,它采用的技术就是双核技术。双核技术是人们开始寻找提高微处理器性能的一个行之有效的方法。以前传统上增强微处理器性能的做法是提高 CPU 的主频。众所周知高时钟频率造成 CPU 耗电量大、产生大量热,微处理机散热成了难题。双核技术有效地解决了这个问题。增加了一个内核,微处理器内有两个 CPU 在工作,这样在每个时钟周期内微处理器执行指令数增加了一倍,从而提高了计算机处理信息的速度。由于微处理器中双核结构的引入,以及伴随 Microsoft Windos Vista 操作系统上市,硬软结合也有力地推动了虚拟技术的发展。虚拟技术的应用可以在一台物理计算机上虚拟出若干虚拟机系统,这些虚拟系统可以共享一个 PC 资源,互不干扰各自独立进行工作。这正是当今“云计算”所需求的。

双核处理器体系结构是在一个微处理器芯片中集成两个相同的核心执行部件,每个核心执行部件均配备一个自己独享的二级高速缓存储器,由各自总线连接,但是两个核心部件工作时分时共享微处理器的系统总线。

2. 多核微处理器

随着多核技术的发展,制作工艺不断创新,在一个微处理器芯片上配置 4 个、8 个或 16 个功能相同的处理器核心,这样的微处理器芯片就是 4 核、8 核或 16 核微处理器,统称多核微处理器。

多核微处理器体系结构也是在一个微处理器芯片中集成多个相同的核心执行部件,每个核心执行部件均配备一个自己独享的二级高速缓存储器,由各自总线连接。每个核心部件工作时分时共享微处理器的系统总线。这里多个核心部件工作的同步问题要比双核微处理器同步问题复杂多了。

3.5.2 微处理器发展现状

目前高性能微处理器发展有两个较明显的趋势:其一,世界各国微处理器生产厂商纷纷加强多核结构的微处理器的研制;其二,研制高性能微处理器的公司开阔视野,服务对象不仅是计算机,还扩展到移动智能设备等。市场竞争激烈,且市场份额越来越集中到少数几家。IBM 公司是最早推出多核微处理器的厂商,Power4 + 是其多核微处理器的代表作,并在"蓝色基因"巨型机中使用了自己设计生产的双核处理器芯片。2001 年 IBM 公司发布了 Power4,其芯片内配置了两个 Power3 处理器核,大约集成了 1.74 亿个晶体管。2004 年 IBM 发布了 Power5 微处理器,Power5 是双核同时多线程微处理器,每个核设计为同时多线程处理器(Simultaneous Multi - Threading,SMT),能并发执行两个线程。该芯片大约集成 2.76 亿个晶体管。2006 年 IBM 发布了 Power6 微处理器,Power6 采用了 CMOS channe 加上不同应力达到提高电子或电穴的迁移率的 DSL(Dual - Stress Line)技术,使在同等功耗下性能提高了 30%。Power6 芯片配备了两个同时多线程的处理器核,其微处理器核的频率达到 5GHz。

美国的 IBM 公司和 AMD 公司一直是全球处理器市场两大竞争对手,2008 年前两大公司几乎垄断了全球处理器市场,之后,随着移动智能设备强势上升的市场需求,微处理器急需扩充功能。高通和三星等公司崛起,并飞速发展。2013 年第一季度全球处理器厂商的排名是 Inter、高通、三星、AMD,AMD 公司的排名已落至第四位。

美国 Inter 公司技术力量雄厚,多年盘踞微处理器研发生产的首位。2005 年推出集成 17.2 亿个晶体管的双核多线程微处理器 Montecito。2006 年 Inter 推出基于 Core 构架处理器 Conroe(酷睿 2),即第二代 Core i 系列问世。例如,以 32nm 制造工艺、主频 3.6GHz、12MB 高速缓存的酷睿 i7 - 980XEE 已在流通市场上销售。一年后以 Ivy BridgeCore 为研发代号的第三代 Core i3/i5/i7 统称第三代智能酷睿处理器 Core i 又隆重推出。Core i 系列具备以下 5 方面的改进:

1)以 22 nm U3 - D 晶体管制造工艺设计,功能更强,功耗更低。

2)新一代核芯显示 CPU 支持 DXII,性能大幅提升。

3)核心指令集优化,使之同频下 CPU 性能更强。

4)高速视频同步技术的应用,使 CPU 的功能增强,进一步波击为移动智能设备服务。

5)安全性更强。

随着市场的需求不断增强,Inter 公司于 2013 年 6 月 2 日在台北 Computex 大展之前召开发布会,正式推出第四代酷睿微处理器 HaswerII。与上一代 Ivy Bridge 相比,HaswerII 微处理器采用了全新微架构,提升了每个时钟周期完成的指令数,突出了更高的能效比。2013 年第二季度 Inter 公司陆续推出 5 款酷睿 i7 微处理器和 9 款酷睿 i5 微处理器,具体型号为酷睿 i7 - 4770、i7 - 4770K、i7 - 4770S、i7 - 4770T、i7 - 4765T、i5 - 4670、i5 - 4670K、i5 - 4670S、i5 - 4670T、i5 - 4430 和 i5 - 4430S。

Inter 公司出品的酷睿处理器受到全球用户的欢迎。

美国 AMD 公司开始研发 64 位多核 CPU 的步伐是走在 Inter 公司前面。2004 年 8 月 AMD 公司就演示了 Opteron 微处理器并在 2005 年中期上市,它比早期 Inter 公司推出的双核处理器在性能和功耗上均有优势。在以后的发展中,由于在组织、技术、资金、宣传上的因素,渐渐落后,以至于 2003 年从全球第二落至第四。

尽管多核微处理器还面临着和当前计算机体系结构的匹配问题,反映在支持多线程并行应用和当前计算机体系结构一维地址和多核处理器的多维地址访存层次的匹配问题,但这些问题的出现推动了人们对多核微处理器系统及系统软件的研究。每解决一个问题,就会推动微处理器技术向前一步,微处理器的发展前景非常广阔。

3.6 习题

1. 简述 80286 的特点和保护模式的保护功能。
2. 简述 80386 的特点,80386 引脚与 8086 的区别。
3. 简述 80386CPU 寄存器的组成、特点及作用。
4. 简述 80386 的 3 种工作模式的特点和异同。
5. 什么是逻辑地址、线性地址和物理地址? 三者之间的关系是什么?
6. 简述 80486CPU 的组成及各部分的作用。
7. Pentium 微处理器采用了哪些新的技术和结构?

第4章　指令系统

指令是计算机用以控制各个部件协调动作的命令。一台机器所具有的全部指令称为机器的指令系统,它全面描述了微处理器的功能。因此,在其他条件相同时,指令系统功能越强,计算机的功能就越强。

不同的微处理器有不同的指令系统。本章将讨论 8086/8088 的寻址方式、8086/8088 指令系统以及从 80286 到 Pentium 系列微处理器的指令系统。

4.1　8086/8088 指令系统概述

8086/8088 指令系统除了拥有 8 位微处理器指令系统中全部指令的功能之外,还增加了一些功能强大的、能处理 16 位二进制数的指令以及一些数据处理指令,因此 8086/8088 指令系统具有处理 8 位和 16 位二进制数的能力。

4.1.1　8086/8088 指令系统的特点

8086 与 8088 指令系统完全相同,下面将对其具体特点加以说明。

1. 指令系统的兼容性

由于 8086/8088 指令系统包含了 8 位微处理器 8080/8085 中全部指令的功能,因而 8080/8085 的用户所编写的应用软件可以很容易地移植到 8086/8088 系统中,这是因为 Intel 公司设计的 8086/8088 指令系统与 8 位的 8080/8085 指令系统是向上兼容的。

2. 指令格式的灵活性

8086/8088 采用可变字节的指令格式,指令长度可在 1 ~ 6B 变化。指令中的操作数可以是一个字节也可以是两个字节,操作数可以是立即数、寄存器操作数和存储器操作数,而且指令中每个字节的功能都有具体规定。这样的指令既占有较少的存储空间,又便于指令的使用和记忆。

3. 寻址方式的多样性

8086 采用了多种不同的寻址方式,存储器操作数的寻址可以通过指令中给出的 16 位偏移地址直接寻址,也可以在基址或变址寄存器的内容上加上指令中给出的 8 位或 16 位偏移量作为偏移地址进行间接寻址。

4. 可对多种类型的数据进行处理

8086/8088 指令系统既可以处理 8 位或 16 位、带符号或不带符号的二进制数,又可以处理用 BCD 码表示的压缩和非压缩的十进制数,而且对 8 位或 16 位的操作数均可进行算术运算、移位和数据传送等操作,这使得信息处理的效率大大提高。

5. 可构成多处理机系统

8086/8088 指令系统拥有一组多处理机指令,具有较强的软件中断功能,对构成多处理机系统提供了方便条件。