清华大学计算机系列教材

微型计算机技术及应用

——习题与实验题集

（第二版）

戴梅萼 编著

清华大学出版社

微型计算机技术及应用

——从16位到32位

习题与实验题集

（第二版）

戴梅萼 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书是和清华大学计算机系列教材《微型计算机技术及应用——从 16 位到 32 位(第二版)》配套的习题和实验题集。每章的习题都针对了教材中相应章节的关键技术和主要内容,其中包含了一部分例题性习题。和初版相比,增加了以 80386 为对象的 32 位微处理器的原理和关键技术的有关习题,其中包括片内两级存储管理技术、虚拟存储技术、流水线技术及高速缓存技术;另外,以 MCS-8051 为对象重写了单片微型机一章的习题;还增加了一份模拟试卷及其答案。实验题的软件部分可以用任何一台 IBM PC/XT、AT、Pentium 来进行;硬件部分须另外连接实验线路来完成。书中对每个实验均给出详细线路图,只须在主机总线扩展槽和实验线路间加简单的总线驱动线路即可实现。

图书在版编目(CIP)数据

从 16 位到 32 位:习题与实验题集/戴梅萼编著.—2 版.北京:清华大学出版社,1997
清华大学计算机系列教材
ISBN 7-302-02646-7

Ⅰ.从… Ⅱ.戴… Ⅲ.微型计算机-高等学校-习题 Ⅳ.TP368-44

中国版本图书馆 CIP 数据核字(97)第 18086 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
责任编辑:贾仲良
印刷者:北京通县大中印刷厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:8.5 字数:198 千字
版 次:1997 年 9 月第 2 版 1997 年 9 月第 1 次印刷
书 号:ISBN 7-302-02646-7/TP・1364
印 数:0001～6000
定 价:8.00 元

第二版前言

本书是和《微型计算机技术及应用——从16位到32位(第二版)》完全配套的习题和实验题集。与第一版相比,主要在以下几方面作了修订:

1. 删除了习题部分第11章音频盒式磁带接口,所以,后面的章节序号依次提前。

2. 以MCS-8051为对象重写了单片微型机一章的习题。

3. 增加了习题部分第16～19章,这几章的习题主要围绕以下内容:32位微处理器的工作原理、片内两级存储管理、虚拟存储技术、流水线技术、32位微处理器指令系统特点和高速缓存技术。

4. 附上了一份模拟试卷,并给出了答案,这一点主要是考虑了许多自学者的要求。

5. 对第一版习题从文字上作了全面修改。

本书第3、13、14、17、19章由史嘉权执笔,史云凌对试卷作了解答,其余均由戴梅萼执笔。

戴梅萼

1997年5月于清华大学

前　言

《微型计算机技术及应用》一书自 1991 年 11 月出版以来，编著者收到了许多读者的热情来信，他们像相识已久的朋友一样提出了不少有益的建议，其中最普遍最一致的便是希望有一本对应的习题和实验题集。希望这本题集的出版能满足广大读者的这一要求。

本题集完全和教材《微型计算机技术及应用》一书相配套。每一章的习题针对了教材中相应章节的关键技术和主要内容。此外，题集中还包含了部分例题性习题，这类习题实际上是对教材的一种补充，它们一方面提供了程序实例以具体说明一些重要技术的使用方法，另一方面要求读者据此举一反三，去编写一个应用这些技术的另一个程序，或者编写一个更高层次的程序，等等。实验题集分为软件和硬件两部分。所有的软件实验可以在任何一台 IBM PC/XT、AT、Pentium 机上完成；12 个硬件实验则须另外连接硬件线路才能完成，也可在《TPC-1 型 16 位微机实验培训系统》上进行。TPC-1 实验系统并不是一个独立的系统，而只是一个实验台，它必须通过 62 芯总线驱动板接到 PC 机上才能使用。实验台上主要安置了 12 个硬件实验所用到的 8253、8251A、8255A、DAC0832、ADC0809 芯片及附加电路，还有小键盘、数码管、8MHz 晶振、发光二极管等器件（见附录 3）。

在本题集的编写和定稿过程中，北京计算机学院苏开娜副教授提出了许多建设性建议并作了全面审定；清华大学计算机系史嘉权副教授编写了部分章节的习题；清华大学计算机系（计九年级）学生史云凌对书中的全部程序进行了调试验证；此外，几位热心的朋友试用了整套习题，并从读者的角度提出许多宝贵意见；还有和我共同设计 TPC-1 实验系统的冯一兵高级工程师等。在此，向他们表示最真诚的谢意。

由于水平所限，书中仍然会有错误和不足之处，敬请读者批评指正。

戴梅萼

1994 年 5 月于清华大学

目　录

习　题

第 1 章　微型计算机概述

1.1　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？

1.2　CPU 在内部结构上由哪几部分组成？CPU 应具备哪些主要功能？

1.3　累加器和其他通用寄存器相比，有何不同？

1.4　微处理器的控制信号有哪两类？

1.5　微型计算机采用总线结构有什么优点？

1.6　数据总线和地址总线在结构上有什么不同之处？如果一个系统的数据和地址合用一套总线或者合用部分总线，那么，要靠什么来区分地址和数据？

1.7　控制总线传输的信号大致有哪几种？

第 2 章　8086 微处理器

2.1　总线接口部件有哪些功能？请逐一说明。

2.2　8086 的总线接口部件由哪几部分组成？

2.3　段寄存器 CS＝1200H，指令指针寄存器 IP＝FF00H，此时，指令的物理地址为多少？指向这一物理地址的 CS 值和 IP 值是唯一的吗？

2.4　8086 的执行部件有什么功能？由哪几部分组成？

2.5　状态标志和控制标志有何不同？程序中是怎样利用这两类标志的？8086 的状态标志和控制标志分别有哪些？

2.6　8086/8088 和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？

2.7 总线周期的含义是什么？8086/8088 的基本总线周期由几个时钟组成？如一个 CPU 的时钟频率为 24MHz，那么，它的一个时钟周期为多少？一个基本总线周期为多少？如主频为 15MHz 呢？

2.8 在总线周期的 T_1、T_2、T_3、T_4 状态，CPU 分别执行什么动作？什么情况下需要插入等待状态 T_W？T_W 在哪儿插入？怎样插入？

2.9 从引腿信号上看，8086 和 8088 有什么不同？

2.10 在对存储器和 I/O 设备读写时，要用到$\overline{IOR}$、$\overline{IOW}$、$\overline{MR}$、$\overline{MW}$信号，这些信号在最大模式和最小模式时分别可用怎样的电路得到？请画出示意图。

2.11 CPU 启动时，有哪些特征？如何寻找 8086/8088 系统的启动程序？

2.12 CPU 在 8086 的微机系统中，为什么常用 AD_0 作为低 8 位数据的选通信号？

2.13 8086 和 8088 在最大模式和最小模式时，引腿信号分别有什么不同？

2.14 8086 和 8088 是怎样解决地址线和数据线的复用问题的？$\overline{ALE}$信号何时处于有效电平？

2.15 $\overline{BHE}$信号和 A_0 信号是通过怎样的组合解决存储器和外设端口的读/写操作的？这种组合决定了 8086 系统中存储器偶地址体及奇地址体之间应该用什么信号区分？怎样区分？

2.16 RESET 信号来到后，CPU 的状态有哪些特点？

2.17 在中断响应过程中，8086 往 8259A 发的两个$\overline{INTA}$信号分别起什么作用？

2.18 总线保持过程是怎样产生和结束的？画出时序图。

2.19 8086 系统在最小模式时应该怎样配置？请画出这种配置并标出主要信号的连接关系。

2.20 时钟发生器的功能是什么？画出它的线路图。

2.21 8086 在最大模式下应当怎样配置？最大模式时为什么一定要用总线控制器？总线

控制器的输入信号是什么？输出信号是什么？

2.22 在编写程序时，为什么通常总要用开放中断指令来设置中断允许标志？

2.23 T_1 状态下，数据/地址线上是什么信息？用哪个信号将此信息锁存起来？数据信息是在什么时候给出的？用时序图表示出来。

2.24 画出 8086 最小模式时的读周期时序。

2.25 8086 最多可有多少级中断？按照产生中断的方法分为哪两大类？

2.26 非屏蔽中断有什么特点？可屏蔽中断有什么特点？分别用在什么场合？

2.27 什么叫中断向量？它放在哪里？对应于 1CH 的中断向量存放在哪里？如果 1CH 的中断处理子程序从 5110H：2030H 开始，则中断向量应怎样存放？

2.28 从 8086/8088 的中断向量表中可以看到，如果一个用户想定义某个中断，应该选择在什么范围？

2.29 非屏蔽中断处理程序的入口地址怎样寻找？

2.30 叙述可屏蔽中断的响应过程，一个可屏蔽中断或者非屏蔽中断响应后，堆栈顶部四个单元中是什么内容？

2.31 一个可屏蔽中断请求来到时，通常只要中断允许标志为 1，便可在执行完当前指令后响应，在哪些情况下有例外？

2.32 在对堆栈指针进行修改时，要特别注意什么问题？为什么？

2.33 在编写中断处理子程序时，为什么要在子程序中保护许多寄存器？有些寄存器即使在中断子程序中并没有用到也需要保护，这又是为什么（联系串操作指令执行时遇到中断这种情况来回答）？

2.34 一个可屏蔽中断响应时，CPU 要执行哪些读/写周期？对一个软件中断又如何？

2.35 中断处理子程序在结构上一般是怎样一种模式？

2.36 软件中断有哪些特点？在中断处理子程序和主程序的关系上，软件中断和硬件中断

有什么不同之处？

2.37 系统中有多个总线模块时，在最大模式和最小模式下分别用什么方式来传递总线控制权？

2.38 8086的存储器空间最大可以为多少？怎样用16位寄存器实现对20位地址的寻址？

2.39 IBM PC/XT系统中，哪个区域为显示缓冲区？哪个区域用来存放中断向量？在FFFF0H到FFFFFH单元中存放什么内容？

第3章 8086的寻址方式和指令系统

3.1 8086汇编语言指令的寻址方式有哪几类？用哪一种寻址方式的指令执行速度最快？

3.2 直接寻址方式中，一般只指出操作数的偏移地址，那么，段地址如何确定？如果要用某个段寄存器指出段地址，指令中应如何表示？

3.3 在寄存器间接寻址方式中，如果指令中没有具体指明段寄存器，那么，段地址如何确定？

3.4 用寄存器间接寻址方式时，BX、BP、SI、DI分别针对什么情况来使用？这四个寄存器组合间接寻址时，地址是怎样计算的？举例进行说明。

3.5 设DS=2100H，SS=5200H，BX=1400H，BP=6200H，说明下面两条指令所进行的具体操作：

```
MOV  BYTE  PTR  [BP], 2000
MOV  WORD  PTR  [BX], 2000
```

3.6 使用堆栈操作指令时要注意什么问题？传送指令和交换指令在涉及内存操作数时分别要注意什么问题？

3.7 下面这些指令中哪些是正确的？哪些是错误的？如是错误的，请说明原因。

```
XCHG    CS, AX
MOV     [BX], [1000]
XCHG    BX, IP
PUSH    CS
POP     CS
```

```
IN      BX, DX
MOV     BYTE [BX], 1000
MOV     CS, [1000]
```

3.8 8086 系统中，当对 SS 和 SP 寄存器的值进行修改时，有什么特殊规定？这样做的原因是什么？

3.9 以下是格雷码的编码表

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101

请用换码指令和其他指令设计一个程序段，实现格雷码往 ASCII 码的转换。

3.10 用加法指令设计一个简单程序，实现两个 16 位十进制数的相加，结果放在被加数单元。

3.11 为什么用增量指令或减量指令设计程序时，在这类指令后面不用进位标志 CF 作为判断依据？

3.12 用乘法指令时，特别要注意先判断用有符号数乘法指令还是用无符号数乘法指令，这是为什么？

3.13 字节扩展指令和字扩展指令用在什么场合？举例说明。

3.14 什么叫 BCD 码？什么叫组合的 BCD 码？什么叫非组合的 BCD 码？8086 汇编语言在对 BCD 码进行加、减、乘、除运算时，采用什么方法？

3.15 用普通运算指令执行 BCD 码运算时，为什么要进行十进制调整？具体讲，在进行 BCD 码的加、减、乘、除运算时，程序段的什么位置必须加上十进制调整指令？

3.16 普通移位指令和循环移位指令(带 CF 的和不带 CF 的两类)在执行操作时，有什么差别？在编制乘除法程序时，为什么常用移位指令来代替乘除法指令？试编写一个

程序段，实现将 BX 中的数除以 10，结果仍放在 BX 中。

3.17 串操作指令使用时，特别要注意和 SI、DI 这两个寄存器及方向标志 DF 密切相关。请具体就指令 MOVSB/MOVSW、CMPSB/CMPSW、SCASB/SCASW、LODSB/LODSW、STOSB/STOSW 列表说明和 SI、DI 及 DF 的关系。

3.18 用串操作指令设计实现如下功能的程序段：首先将 100H 个数从 2170H 处搬到 1000H 处，然后，从中检索相等于 AL 中字符的单元，并将此单元值换成空格符。

3.19 在使用条件转移指令时，特别要注意它们均为相对转移指令，请解释“相对转移”的含义。如果要往较远的地方进行条件转移，那么，程序中应该怎样设置？

3.20 带参数的返回指令用在什么场合？设栈顶地址为 3000H，当执行 RET 0006 后，SP 的值为多少？

3.21 用循环控制指令设计程序段，从 60H 个元素中寻找一个最大值，结果放在 AL 中。

3.22 中断指令执行时，堆栈的内容有什么变化？中断处理子程序的入口地址是怎样得到的？

3.23 中断返回指令 IRET 和普通子程序返回指令 RET 在执行时，具体操作内容有什么不同？

3.24 断点中断是指怎样一种中断？在程序调试中有什么作用？断点中断指令有什么特点？设置断点过程对应了一种什么操作？这种操作会产生什么运行结果？

3.25 HLT 指令用在什么场合？如 CPU 在执行 HLT 指令时遇到硬件中断并返回后，以下应执行哪条指令？

3.26 总线封锁指令用在什么场合？以飞机订票系统为例说明总线封锁指令的作用（设飞机订票系统为一个多处理器系统，每个处理器都是平等的）。

3.27 设当前 SS＝2010H，SP＝FE00H，BX＝3457H，计算当前栈顶地址为多少？当执行 PUSH BX 指令后，栈顶地址和栈顶 2 个字节的内容分别是什么？

3.28 在 DS 段中有一个从 TABLE 开始的由 160 个字符组成的链表，设计一个程序，实现对此表进行搜索，找到第一个非 0 元素后，将此单元和下一单元清 0。

3.29 下面的程序段将 ASCII 码的空格字符填满 100 个字节的字符表。阅读这一程序段，画出流程，并说明使用 CLD 指令和 REP STOSB 指令的作用，再指出 REP STOSB 指令执行时和哪几个寄存器的设置有关？

```
        MOV     CX, SEG TABLE          ;TABLE 为字节表表头
        MOV     ES, CX
        MOV     DI, OFFSET TABLE       ;DI 指向字节表
        MOV     AL,' '                 ;空格符送 AL
        MOV     CX,64H                 ;字节数
        CALL    FILLM                  ;调用填数子程序
        ⋮                              ;后续处理
        ⋮
FILLM:  JCXZ    EXIT                   ;CX 为 0 则退出
        PUSH    DI                     ;保存寄存器
        PUSH    CX
        CLD                            ;方向标志清零
        REP     STOSB                  ;重复填数
        POP     CX
        POP     DI
EXIT:   RET
```

3.30 以下程序段将一个存储块的内容复制到另一个存储块，进入存储段时，SI 中为源区起始地址的偏移量，DI 中为目的区起始地址的偏移量，CX 中为复制的字节数。阅读此程序段并具体说明 REP MOVSB 指令使用时与哪些寄存器有关？

```
        PUSH    DI                     ;保存寄存器
        PUSH    SI
        PUSH    CX
        CMP     DI,SI                  ;看源区和目的区的地址哪个高
        JBE     LOWER                  ;如目的区地址较低，则转移
        STD                            ;目的区地址高，则设方向标志为 1
        ADD     SI,CX                  ;从最后一个字节开始复制
        DEC     SI                     ;调整源区地址
        ADD     DI,CX
        DEC     DI                     ;调整目的区地址
        JMP     MOVEM
LOWER:  CLD                            ;从第一个字节开始复制
MOVEM:  REP     MOVSB
        POP     CX
        POP     SI
        POP     DI
        RET
```

3.31 下面的程序段实现对两个存储区中的字进行比较。如找到一对不同的字，则退出，此时，ZF 标志为 0，DI 指向此字；如两个存储区中所有字均一一相同，则退出程序时，CX 中值为 0，ZF 标志为 1。阅读这一程序段，并仿此设计一个比较字节块的程序段。

```
MATT:   MOV    SI,OFFSET SOURCE      ;源区首址
        MOV    DI,OFFSET TARGET      ;目的区首址
        MOV    CX,NUMBER
        JCXZ   EUIT                  ;如 CX 为 0,则结束
        PUSH   CX                    ;保存有关寄存器
        PUSH   SI
        PUSH   DI
        CLD                          ;清方向标志
        REPE   CMPSW                 ;比较
        JZ     MATCH                 ;ZF 标志为 1,则转移
        PUSHF                        ;ZF 标志为 0,则 DI 指向此字
        SUB    DI,2
        POPF
        JMP    EXIT                  ;退出
MATCH:  POP    DI                    ;恢复寄存器
        POP    SI
        POP    CX
EXIT:   RET
```

3.32 下面的程序段实现在 TABLE 为起始地址的 100 个字符长度的表中检索“$”字符。请分析这一程序段，然后说明 REPNE SCASB 指令的具体执行过程。

```
START:  MOV    CX,SEG TABLE          ;表段地址送 ES
        MOV    ES,CX
        MOV    DI,OFFSET TABLE       ;表偏移量送 DI
        MOV    AL,'$'                ;检索的关键字
        MOV    CX,64H                ;检索的字节数
        PUSH   DI                    ;保存起始地址
        CLD                          ;清除方向标志
        REPNE  SCASB                 ;检索
        JNZ    NFOUN                 ;如未找到,则转移
        SUB    DI,1                  ;找到,则指向此字符
        JMP    EXIT
NFOUN:  POP    DI                    ;恢复起始地址
EXIT:   RET
```

3.33 下面的程序段实现两个 32 位不带符号数的相乘，被乘数在 DX 和 AX 寄存器中，乘数在 CX 和 BX 寄存器中，最后的 64 位乘积在 DX，CX，BX，AX 中。图 1 说明乘

法过程。读懂程序段和附图，并自己设计一个程序，实现一个 16 位数和一个 32 位非符号数相乘。

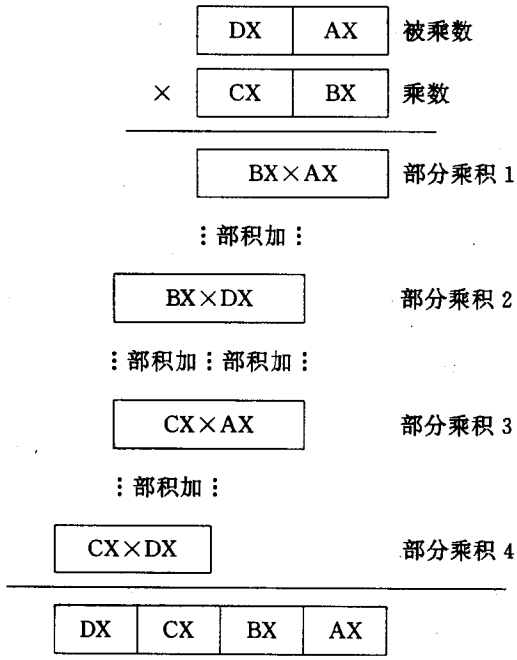

图 1　两个 32 位非符号数相乘流程图

```
STAT:   JMP   MUL64
HI0     DW    ?
LO0     DW    ?
HI1     DW    ?
LO1     DW    ?
HI2     DW    ?
LO2     DW    ?
HI3     DW    ?
LO3     DW    ?
HI4     DW    ?
LO4     DW    ?
MUL64:  MOV   HI0,DX      ;保存被乘数
        MOV   LO0,AX
        MUL   BX          ;得部分乘积 1
        MOV   HI1,DX      ;保存部分乘积 1
        MOV   LO1,AX
        MOV   AX,HI0      ;得部分乘积 2
        MUL   BX
        MOV   HI2,DX      ;保存部分乘积 2
```

```
        MOV   LO2,AX
        MOV   AX,LO0          ;得部分乘积 3
        MUL   CX
        MOV   HI3,DX          ;保存部分乘积 3
        MOV   LO3,AX
        MOV   AX,HI0          ;得部分乘积 4
        MUL   CX
        MOV   HI4,DX          ;保存部分积 4
        MOV   LO4,AX
        MOV   AX,LO1          ;乘积的低 16 位在 AX 中
        MOV   BX,HI1          ;乘积的次低 16 位在 BX 中
        ADD   BX,LO2
        ADC   HI2,0
        ADD   BX,LO3
        MOV   CX,HI2          ;乘积的次高 16 位在 CX 中
        ADC   CX,HI3
        ADC   HI4,0
        ADD   CX,LO4
        MOV   DX,HI4          ;乘积的高 16 位在 DX 中
        ADC   DX,0
        RET
```

3.34 下面的程序实现两个 32 位带符号数的乘法，其中调用了题 3.33 中的非符号数相乘的程序 MUL64，结果放在 DX、CX、BX、AX 四个寄存器中，进入程序时，DX、AX 中为被乘数，CX、BX 中为乘数。读懂程序后再设计一个 16 位带符号数和 32 位带符号数相乘程序。

```
MULS64: MOV   [1000],0        ;1000 单元作为负数标志
        CMP   DX,0            ;被乘数为负数吗?
        JNS   CHKK            ;否，则转 CHKK
        NOT   AX              ;是，则取补码
        NOT   DX
        ADD   AX,1
        ADC   DX,0
        NOT   [1000]          ;负数标志置 1
CHKK:   CMP   CX,0            ;乘数为负数吗?
        JNS   GOMUL           ;否，则转 GOMUL
        NOT   BX              ;是，则取补码
        NOT   CX
        ADD   BX,1
        ADC   CX,0
        NOT   [1000]          ;将负数标志取反
```