高等学校教材

计算机系统实践教程

张吉锋　刘恩林　魏廷德

国防工业出版社

计算机系统实践教程

张吉锋　刘恩林　魏廷德

国防工业出版社

内 容 简 介

计算机系统实践教学，是当前教学工作中的一个薄弱环节。本教程针对这个教学环节，吸收了全国主要院校的经验，经过综合分析，按照软、硬件逻辑功能等效的观点，合理组织内容，编写成33个实验，形成了一个较为完整的实践教材，内容包括：基本概念、存储体系、I/O系统、流水技术、计算机性能测试、互连网络、并行处理结构、主从结构多机系统和计算机网络等。

本教程所选进的各实验内容，经过有关院校的实践，证明是完全可行的，对培养学生理论联系实际和分析问题与解决问题的能力将收到良好效果。

本书为高等学校计算机科学技术各专业高年级学生的统编教材，也可供计算机、自动控制等有关工程技术人员自学与参考。

计算机系统实践教程

张吉铮　刘恩林　魏廷德

责任编辑　周烈强

国防工业出版社出版

（北京市海淀区紫竹院南路23号）

（邮政编码：100044）

新华书店北京发行所发行　各地新华书店经售

国防工业出版社印刷厂印装

787×1092　1/16　印张16¾　388千字

1990年9月第一版　1990年9月北京第一次印刷　印数：.0001—1.300册

ISBN 7-118-00704-8/TP·90　　定价：3.35元

出 版 说 明

根据国务院关于高等学校教材工作分工的规定，我部承担了全国高等学校、中等专业学校工科电子类专业教材的编审、出版的组织工作。由于各有关院校及参与编审工作的广大教师共同努力，有关出版社的紧密配合，从 1978 ～ 1985 年，已编审、出版了两轮教材，正在陆续供给高等学校和中等专业学校教学使用。

为了使工科电子类专业教材能更好地适应“三个面向”的需要，贯彻“努力提高教材质量，逐步实现教材多样化，增加不同品种、不同层次、不同学术观点、不同风格、不同改革试验的教材”的精神，我部所属的七个高等学校教材编审委员会和两个中等专业学校教材编审委员会，在总结前两轮教材工作的基础上，结合教育形势的发展和教学改革的需要，制订了 1986～1990 年的“七五”（第三轮）教材编审出版规划。列入规划的教材、实验教材、教学参考书等近 400 种选题。这批教材的评选推荐和编写工作由各编委会直接组织进行。

这批教材的书稿，是从通过教学实践、师生反映较好的讲义中经院校推荐，由编审委员会（小组）评选择优产生出来的。广大编审者、各编审委员会和有关出版社为保证教材的出版和提高教材的质量，作出了不懈的努力。

限于水平和经验，这批教材的编审、出版工作还会有缺点和不足之处，希望使用教材的单位，广大教师和同学积极提出批评建议，共同为不断提高工科电子类专业教材的质量而努力。

电子工业部教材办公室

前　言

本教材系按电子工业部工科电子类专业教材 1986～1990 年编审出版规划，由计算机教材编审委员会征稿并推荐出版，责任编委张吉锋。

本书由上海工业大学张吉锋、国防科学技术大学刘恩林、中国矿业大学魏廷德编写，国防科学技术大学李勇教授、西安电子科技大学李学干副教授主审。

本教程的参考实验学时为 40～60 学时，内容涉及从单处理机系统到多机并行处理系统及计算机网络通信，通过较全面地研究与分析其组成，使读者进一步掌握有关计算机系统结构的一些基本概念、基本原理和基本技术，从而巩固和加深对计算机系统结构的认识。在材料选择和编写上，按计算机和系统结构下移的观点，选取具有典型系统结构的微型机、小型机系统作为实践环境；按软、硬件逻辑功能等效的观点，合理组织实验体系，使前后实验成为一个较为完整的系统。具体内容包括：基本概念，存储体系，I/O 系统，流水技术，计算机系统性能测试，互连网络，并行处理机结构，主从结构多机系统和计算机网络。

计算机各专业的实践教学是当前各院校教学工作中的一个薄弱环节。近几年来，各院校都十分重视这个教学环节，做了大量工作，但仍缺少一本较为系统而完整的教材。为了适应各院校教学工作需要，推动计算机专业的实践教学，编委会组织国防科学技术大学、西安交通大学、上海交通大学、上海工业大学、复旦大学、华中理工大学、东北工学院、哈尔滨工业大学、电子科技大学、东南大学、西安电子科技大学、北京工业大学、华东师范大学等院校的有关专业教师进行了认真细致的讨论，拟定了编写大纲。本教程就是根据这个大纲，以及上述各院校所提供的、经过自己实践证明完全可行的有关实验材料和其他参考资料编写而成的。本书共提供了 33 个实验。限于教学时间上的安排，每个学校不可能全部做完所有这些实验，只能根据自己的需要与可能，选择其中的一部分。希望各院校在选择时注意后续实验能继承前面实验的结果，其中有些专用装置和软件也可以与有关院校联系，他们可以提供方便，以节省准备时间。

本实践教程的第一～四章由刘恩林编写，第五章由张吉锋、孙昱东编写，第六、七章由魏廷德编写，第八章由张吉锋、涂士亮编写，第九章由张吉锋、郭怡峰、杜轩华编写，张吉锋统编全稿。参加审阅与修改工作的还有徐良贤、吕俊、孙德文、刘景文、朱秀珍等同志，他们为本教程提出了许多宝贵意见，编者在这里谨表示诚挚的感谢。由于水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

1989 年 8 月

目　录

第一篇　单处理机系统

本篇是“计算机系统实践教程”中最基础的部分。通过实践，使读者能了解和掌握有关计算机系统结构的一些基本概念、基本原理和基本技术，从而巩固和加深对计算机系统结构的认识。组织这部分实践的理想环境是具有一台功能完善的实验机(或大型机)系统，它既能包括我们所要研究的系统结构大部分属性，又能允许读者根据实验的要求对其体系结构进行扩充、缩减或修改。然而，这需要较多的投资。目前，一般高等院校和科研单位还不具备这样的投资能力。为此，我们按如下思想组织本篇各章的实践：

（1）按计算机系统结构下移的观点，选取具有典型系统结构的微型机、小型机系统作为实践的环境。如 IBM PC/XT、DEC VAX-11/780、MICRO VAX Ⅱ 等。

（2）按软、硬件逻辑功能等效的观点，有一部分硬件实践通过软件模拟的方法完成。

（3）合理组织实验体系，使得后续实验能继承前面实验的成果。

（4）在每一章中尽可能提供多个实验项目，以供读者依据自己的条件进行选择。

本篇主要是围绕单处理机的计算机系统结构进行实验研究。

第一章 基本概念

实验1.1 软、硬件逻辑功能的等效性

一、目的要求

初步掌握计算机系统中原来由硬件实现的功能可以通过软件模拟的方法来实现这一概念，学习在用户程序中测量时间的方法，掌握软、硬件不同的实现方法在性能上不等效的概念。

二、实验原理

在比较简单的微机中，一般没有设置专门的硬件乘（除）法指令，其乘（除）法是用子程序或专门的部件实现。但随着硬件价格的降低，目前大多数微机中已设置了硬件乘（除）法指令。如IBM PC/XT就设置有无符号数的乘（除）法指令和带符号数的乘（除）法指令，用以对8位或16位无符号和带符号的数进行乘（除）法运算。

乘法和除法都是双操作数运算，在指令中应指明两个源操作数地址和结果地址。但在8088的指令中却只指明一个操作数地址，另一个源操作数地址和结果地址采取隐式指明的方法。乘法指令的隐式指明规则如图1-1所示，除法指令的隐式指明规则如图1-2所示。图中的操作数可以取自寄存器，也可以取自存储器。当取自寄存器时，可使用的16位寄存器有AX，BX、CX、DX、SI、DI、BP、SP，可使用的8位寄存器有AH、

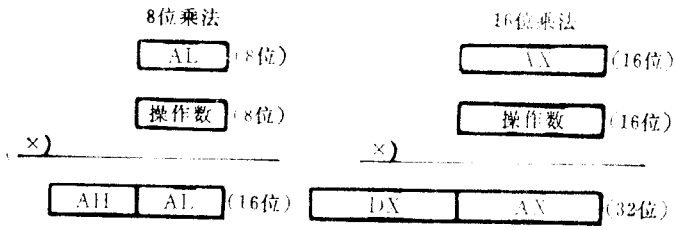

图1-1 8088的乘法指令

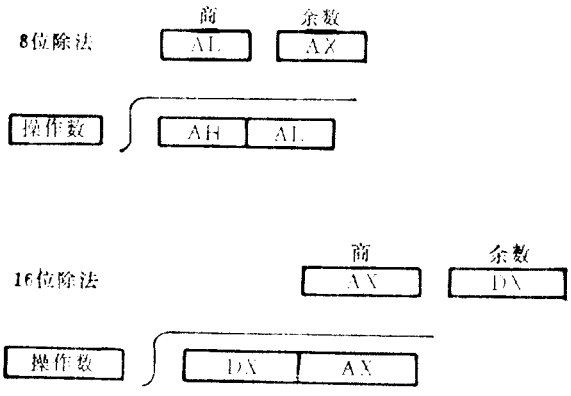

图1-2 8088的除法指令

AL、BH、BL、CH、CL、DH、DL。从图中可以看出，两个8位数相乘，其结果是一个16位数；两个16位数相乘，其结果是一个32位数。在除法操作中，如果除数是8位，则要求被除数是16位；如果除数是16位，则要求被除数是32位。若被除数的位数不够，在进行除法以前，应预先把被除数扩展到所需要的位数。值得注意的是，对于带符号的数，这种扩展应该保持扩展数的值(包括符号位)不变，因此应该是带符号位的扩展。如01101000B应扩展为0000000001101000B，而11011011B应扩展为1111111111011011B。在8088指令系统中，CBW和CWD指令就可用来进行扩展。对于带符号数的除法，IBM PC/XT中采用让余数的符号和被除数符号相同的方法处理。

乘(除)法运算也可以不利用8088的乘(除)法硬件指令，而用汇编语言编写与乘(除)法功能等效的子程序来实现。图1-3示出两个8位无符号数乘法的流程图。程序1-1示出计算65536次乘法的完整的源程序SMUL.ASM，该程序运行结果如数据1-1所示。

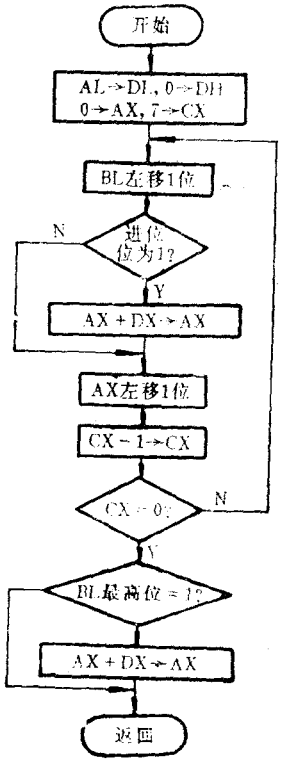

图1-3 乘法流程图

程序1-1 乘法源程序(SMUL.ASM)

```
CODE        SEGMENT
            ASSUME CS:CODE
OP1         DB 'op1(0--255)=$'
OP2         DB 'op2(0--255)=$'
RST         DB 'Result=op1*op2=$'
HD_TIME     DB 'Time of Hard_MUL is 1.01/65535 second',0dh,0ah,'$'
```

```
SF_TIME    DB 'Time of Soft_MUL is ¥'
AUT        DB '/65535 second ¥'
START      PROC FAR
           PUSH DS
           XOR AX,AX
           PUSH AX
           MOV AX,CODE
           MOV DS,AX
           MOV AX,STACK
           MOV SS,AX
           MOV DX,OFFSET OP1
           MOV AH,09H
           INT 21H
           CALL IN_NUM
           PUSH AX
           CALL CRLF
           MOV DX,OFFSET OP2
           MOV AH,09H
           INT 21H
           CALL IN_NUM
           PUSH AX
           CALL CRLF
           MOV AH,2CH
           INT 21H
           POP BX
           POP AX
           PUSH DX
           MOV CX,0FFFFH
AA1:       CALL SMUL
           LOOP AA1
           PUSH DX
           MOV AH,2CH
           INT 21H
           POP BX
           PUSH DX
           MOV AH,09H
           MOV DX,OFFSET RST
           INT 21H
           MOV DX,BX
```

```
                CALL OUT_NUM
                CALL CRLF
                MOV AH,09H
                MOV DX,OFFSET HD_TIME
                INT 21H
                MOV AH,09H
                MOV DX,OFFSET SF_TIME
                INT 21H
                POP AX
                POP BX
                CMP AL,BL
                JG AA2
                SUB AH,1
                ADD AL,100
AA2:            SUB AH,BH
                SUB AL,BL
                XOR DH,DH
                MOV DL,AH
                CALL OUT_NUM
                MOV DL,'.'
                MOV AH,02H
                INT 21H
                XOR DH,DH
                MOV DL,AL
                CALL OUT_NUM
                MOV AH,09
                MOV DX,OFFSET AUT
                INT 21H
                CALL CRLF
                RET
START           ENDP
SMUL            PROC NEAR
                PUSH AX
                PUSH BX
                PUSH CX
                XOR DH,DH
                MOV DL,AL
                XOR AX,AX
                MOV CX,7
```

```
LOOP1:  SHL BL,1
        JNC SS1
        ADD AX,DX
SS1:    SHL AX,1
        LOOP LOOP1
        TEST BL,80H
        JZ SS2
        ADD AX,DX
SS2:    MOV DX,AX
        POP CX
        POP BX
        POP AX
        RET
SMUL    ENDP
IN_NUM  PROC NEAR
        PUSH BX
        PUSH DX
        MOV AH,01
        INT 21H
        SUB AL,'0'
        MOV DL,AL
BB1:    MOV AH,01
        INT 21H
        CMP AL,0DH
        JZ BB2
        XOR AH,AH
        SUB AL,'0'
        MOV BL,10
        XCHG AL,DL
        MUL BL
        ADD DL,AL
        JMP BB1
BB2:    MOV AL,DL
        POP DX
        POP BX
        RET
IN_NUM  ENDP
OUT_NUM PROC NEAR
        PUSH AX
```

```
        PUSH BX
        PUSH CX
        PUSH DX
        XOR CX,CX
        MOV AX,DX
001:    XOR DX,DX
        MOV BX,10
        DIV BX
        ADD DX,'0'
        PUSH DX
        INC CX
        TEST AX,AX
        JNZ 001
002:    TEST CX,CX
        JZ 003
        POP DX
        MOV AH,2
        INT 21H
        DEC CX
        JMP 002
003:    POP DX
        POP CX
        POP BX
        POP AX
        RET
OUT_NUM ENDP
CRLF    PROC NEAR
        MOV AH,02
        MOV DL,0DH
        INT 21H
        MOV DL,0AH
        INT 21H
        RET
CRLF    ENDP
CODE    ENDS
STACK   SEGMENT WORD STACK 'STACK'
        DB 256 DUP(?)
STACK   ENDS
        END START
```

数据1-1　SMUL.ASM 程序运行结果

```
B>SMUL
op1(0--255)=25
op2(0--255)=4
Result=op1*op2=100
Time of Hard_MUL is 1.01/65535 second
Time of Soft_MUL is 8.46/65535 second
```

为了比较用硬件指令方法实现和采用等效的模拟程序方法实现两者的性能，需要测试其所需的时间。程序 1-1 中是采用直接测试方法来得到时间的。也可利用调用子程序的方法来测试时间，具体方法如下所示：

call time(T_1)

call time(T_2)

$T_0=T_2-T_1$

call time(T_1)

被测试程序

call time(T_2)

$T=T_2-T_1-T_0$

其中，假设计时子程序为 time，其所需的时间为 T_0。在被测试程序的前后，再分别调用计时子程序 time，计算出时间差 T_2-T_1。为了消除调用子程序本身所需的时间 T_0，还应在 T_2-T_1 中减去 T_0，从而得到被测试程序所需时间 T。

计时子程序可利用 MS-DOS 提供的软件中断 INT 21，即系统功能调用（其功能号为 2C）。使用的方法是先在 AH 中放入功能号，然后执行 INT 21，则有 CX:DX=时间。程序 1-2 给出了计时子程序清单，以供参考。

程序1-2　计时子程序

```
FRAME STRUC
SAVEBP DW ?
SAVERET DD ?
B DD ?
A DD ?
FRAME ENDS
CSEG SEGMENT 'CODE'
DGROUP GROUP DATA
        ASSUME CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
TIMER PROC FAR
        PUBLIC TIMER
        PUSH BP
        MOV BP,SP
        MOV AH,2CH
```

```
        INT 21H
        LES BX, [BP]+A
        MOV ES: [BX],CX
        LES BX, [BP]+B
        MOV ES: [BX],DX
        POP BP
        RET 8
TIMER ENDP
CSEG ENDS
        END
```

三、实验内容

实验者利用8088的硬件乘（除）法指令，编制计算向量C＝A＊B和C＝A/B的汇编程序，设向量长度为100，向量元素为8位带符号数或16位带符号数（即8位整数或16位整数）。然后，再利用等效的乘（除）法汇编模拟子程序完成同样计算，比较其运行结果和执行时间。

四、方法步骤

（1）熟悉Intel8088的指令系统和IBM PC的宏汇编MASM。

（2）利用整数乘法指令IMUL，编制计算C＝A＊B的汇编程序，上机运行，记录并打印结果，测试其执行时间。

（3）利用等效的乘法指令模拟子程序，编制出完成C＝A＊B的汇编程序，上机运行，记录并打印结果，测试其执行时间。

（4）对（2）和（3）的结果进行比较。分析一条乘法指令和等效的乘法子程序执行时间的差别，写出实验的结论。

（5）按（2）～（4）的方法，再计算C＝A/B，比较其运行结果和执行时间。

五、实验结果

无论是用硬件指令，还是用等效的模拟程序实现乘（除）法的功能，所得的结果值是一致的，这说明软件和硬件在逻辑功能上是等效的。但是，两种方法所需的时间却不相同。硬件的实现方法其速度显然要快得多，表明这两种方法在性能（速度是性能的一种指标）上是不等效的。需要说明的是：各个实验者测出的时间以及由此得出的软、硬件实现的时间之比很可能是不同的，这与实验者编写的汇编程序的质量有很大关系。

六、器材设备

IBM PC或PC/XT，MASM手册，8088指令系统手册。

七、思考题

1. 在什么意义上说软件和硬件是等效的？在什么意义上说两者又是不等效的？
2. 在进行计算机系统设计时，按照哪些基本原则来进行软、硬件功能的取舍？
3. 在单一指令功能模拟的基础上，请考虑如何进一步完成整个CPU功能的模拟？

八、评述

本实验是一个较简单的实验，实验者不但可以掌握软、硬件在逻辑功能上等效，而在性能上不等效的概念，而且通过该实验可使读者进一步熟悉IBM PC的上机操作方

法，从而为后续实验奠定一定的基础。该实验上机预计为 2 小时。

实验1.2 CPU 功能的软件实现

一、目的要求

本实验对于实现复杂功能的硬件（如 CPU），也可以通过软件模拟的方法进行模拟，从而使实验者进一步掌握软、硬件逻辑功能等效性概念和虚拟机的概念，并掌握由硬件实现的 CPU 和由软件实现的 CPU 之间如何进行映象，学会用一种高级语言编制模拟程序的方法，并为后续实验作好技术上的准备。

二、实验原理

从计算机系统的层次结构上看，由软件实现CPU 实际上可以看作是在物理 CPU 的上面再增加一层由软件实现的虚拟 CPU。如图 1-4 所示，此时用户可执行的机器语言程序是通过虚拟 CPU 的解释来实现，而虚拟 CPU 的每一条指令是在物理 CPU 上解释实现的。

为了实现虚拟 CPU，首先应分析硬件 CPU 的结构和功能，然后建立虚拟 CPU 和物理 CPU 的映象关系，最后选取合适的语言编写模拟程序。下面以IBM PC的 CPU—Intel8088 为例，说明建立虚拟 CPU 的原理。

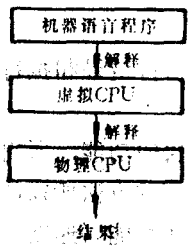

图1-4 虚拟CPU的层次

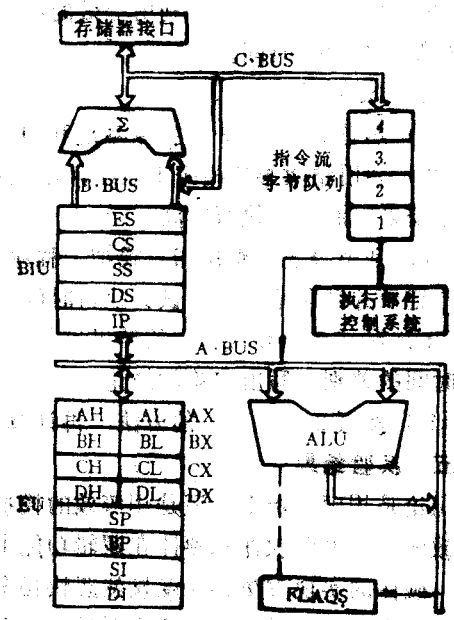

图1-5 8088 的寄存器结构和数据通路

1. 8088 的结构

图 1-5 示出 8088 的寄存器结构和数据通路。从图中可以看出，8088 的结构从功能上可划分为两大部分：总线接口单元 BIU(Bus Interface Unit)和执行单元 EU（Execution Unit)。其中，BIU 负责与存储器接口，即负责从内存的指定区域取出指令，送到指令流队列中排队。执行指令时所需的操作数，也由 BIU 从内存中取出，传送给 EU 部件去处理。Intel8088 的寻址能力为 $2^{20}=1$ MB，这 1 MB 的存储空间可以分成若干段（例如代码段，数据段，堆栈段等），每段最多可包含 64KB，各段的起始地址分别存放在相应的段寄存器中，即 CS——程序代码段，DS——数据段，SS——堆栈段，ES——附加段。各种访存地址是由相应的段寄存器内容左移 4 位后加上相应的段内位移形成的。

例如，指令的物理地址是由代码段寄存器 CS 内容左移 4 位加指令指针 IP 形成的。操作数的物理地址是由数据段寄存器 DS 或附加数据段寄存器 ES 左移 4 位加上逻辑地址（根据指令的寻址方式产生）形成的。访问堆栈时，则由堆栈段寄存器 SS 内容左移 4 位后加上堆栈指针 SP 的内容，得到 20 位的栈顶地址。执行单元 EU 由 8 个 16 位通用寄存器、1 个 16 位的算术逻辑单元 ALU 和 1 个 16 位的标志寄存器 FLAQS 组成。8 个 16位通用寄存器又可分为两组：数据寄存器 AX～DX，指针和变址寄存器 SP、BP、SI、DI。数据寄存器又可作为 8 个 8 位寄存器使用（分别用符号 AH、AL、BH、BL、CH、CL、DH、DL 表示）。指针和变址寄存器中有堆栈指针 SP、基地址指针 BP、源操作数变址寄存器 SI 和目的操作数变址寄存器 DI，由这些寄存器按指令中寻址方式的要求，形成操作数的逻辑地址。16位的标志寄存器 FLAQS 实际上只用到 9 位，用以表示 CPU 的状态标志和控制标志，如图 1-6 所示。

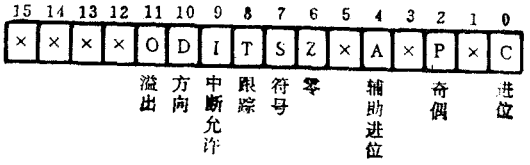

图1-6 FLAQS 寄存器

2. 8088 CPU 在 IBM PC 内存上的映象

为了用软件模拟 Intel 8088 CPU 的功能，应该在 IBM PC 的内存中建立起 8088 的映象，即 8088 指令系统中能访问到的寄存器都应映象到内存的相应单元中。例如，用内存的 8 个相邻字节单元代表 8088EU 中的 8 个 8 位数据寄存器，用内存 8 个相邻的字单元代表 8088EU 中的 8 个 16位寄存器，用内存 4 个相邻的字单元代表 8088BIU 中的 4 个段寄存器，用内存的 1 个字单元代表 8088 的指令指针 IP，用内存的 4 个字节单元代表 8088 的指令流字节队列，用内存的若干位单元分别表示标志寄存器中的状态和控制标志等等。

在 IBM PC 的内存中建立起 8088 CPU 的各寄存器的映象后，则原来 8088 指令系统中对各寄存器的操作，现在变成对相应的存储单元进行操作，原来 8088 指令系统中对存储单元的操作，现在在虚拟 8088 CPU 中仍然是对存储单元进行操作。为了减少模拟的工作量，我们没有进行存储器的映象，而使物理 CPU 和虚拟 CPU 的用户程序空间保持一致。

3. 模拟原理

虚拟 CPU 应能模拟物理 CPU 解释用户目标程序的全过程。如果用户目标程序是可执行的二进制代码文件（如.EXE 文件），则虚拟 CPU 是通过重复执行下列步骤来解释一个.EXE 文件的：

（1）从.EXE 文件中取下一条指令，送到虚拟 CPU 的指令流字节队列中；

（2）指令译码；

（3）形成操作数地址，取操作数；

（4）执行指令；

（5）写回结果；