微机原理与接口技术+第四版+钱晓捷主编+课后习题答案

“微机原理与接口技术”习题解答

第1章 微型计算机系统

〔习题1.1〕简答题

〔解答〕

① 处理器每个单位时间可以处理的二进制数据位数称计算机字长。

② 总线信号分成三组，分别是数据总线、地址总线和控制总线。

③ PC机主存采用DRAM组成。

④ 高速缓冲存储器Cache是处理器与主存之间速度很快但容量较小的存储器。

⑤ ROM-BIOS是“基本输入输出系统”，操作系统通过对BIOS的调用驱动各硬件设备，用户也可以在应用程序中调用BIOS中的许多功能。

⑥ 中断是CPU正常执行程序的流程被某种原因打断、并暂时停止，转向执行事先安排好的一段处理程序，待该处理程序结束后仍返回被中断的指令继续执行的过程。

⑦ 主板芯片组是主板的核心部件，它提供主板上的关键逻辑电路。

⑧ MASM是微软开发的宏汇编程序。

⑨ 指令的处理过程。处理器的“取指—译码—执行周期” 是指处理器从主存储器读取指令（简称取指），翻译指令代码的功能（简称译码），然后执行指令所规定的操作（简称执行）的过程。

⑩ 机器语言层，即指令集结构。

（学生很多认为是：汇编语言层。前4章主要涉及汇编语言，但本书还有很多处理器原理等内容）

〔习题1.2〕判断题

① 错 ② 错 ③ 对 ④ 错 ⑤ 对

⑥ 错 ⑦ 错 ⑧ 对 ⑨ 错 ⑩ 错

〔

〔解答〕

① Central Processing Unit，中央处理单元，处理器

② 1MB，4GB

③ 216，64KB

④ EXE，COM（BAT老师讲的）

⑤ Instruction Set Architecture

⑥ 目录

⑦ MMX，SSE3

⑧ 64

⑨ IBM，DOS

⑩ PCI

〔习题1.4〕

说明微型计算机系统的硬件组成及各部分作用。

〔解答〕

CPU：CPU也称处理器，是微机的核心。它采用大规模集成电路芯片，芯片内集成了控制器、运算器和若干高速存储单元（即寄存器）。处理器及其支持电路构成了微机系统的控制中心，对系统的各个部件进行统一的协调和控制。

存储器：存储器是存放程序和数据的部件。

外部设备：外部设备是指可与微机进行交互的输入（Input）设备和输出（Output）设备，也称I/O设备。I/O设备通过I/O接口与主机连接。

总线：互连各个部件的共用通道，主要含数据总线、地址总线和控制总线信号。

〔习题1.5〕

什么是通用微处理器、单片机（微控制器）、DSP芯片、嵌入式系统？

〔解答〕

通用微处理器：适合较广的应用领域的微处理器，例如装在PC机、笔记本电脑、工作站、服务器上的微处理器。

单片机：是指通常用于控制领域的微处理器芯片，其内部除CPU外还集成了计算机的其他一些主要部件，只需配上少量的外部电路和设备，就可以构成具体的应用系统。
DSP芯片：称数字信号处理器，也是一种微控制器，其更适合处理高速的数字信号，内部集成有高速乘法器，能够进行快速乘法和加法运算。
嵌入式系统：利用微控制器、数字信号处理器或通用微处理器，结合具体应用构成的控制系统。
〔习题1.6〕
综述Intel 80x86系列处理器在指令集方面的发展。
〔解答〕
8086奠定了基本的16位指令集，80286提供了保护方式的各种指令，80386将指令集全面提升为32位，80486融入了浮点数据处理指令，奔腾系列陆续增加了多媒体指令MMX、SSE、SSE2和SSE3，最新的奔腾4处理器还支持64位指令集。
题外话：大家可以通过阅读相关资料、查询互联网获得更加详细的发展情况。可以考虑组织成一篇或多篇论文。
〔习题1.7〕
区别如下概念：助记符、汇编语言、汇编语言程序和汇编程序。
〔解答〕
助记符：人们采用便于记忆、并能描述指令功能的符号来表示机器指令操作码，该符号称为指令助记符。
汇编语言：用助记符表示的指令以及使用它们编写程序的规则就形成汇编语言。
汇编语言程序：用汇编语言书写的程序就是汇编语言程序，或称汇编语言源程序。
汇编程序：汇编语言源程序要翻译成机器语言程序才可以由处理器执行。这个翻译的过程称为“汇编”，完成汇编工作的程序就是汇编程序（Assembler）。
〔习题1.8〕
区别如下概念：路径、绝对路径、相对路径、当前目录。系统磁盘上存在某个可执行文件，但在DOS环境输入其文件名却提示没有这个文件，是什么原因？
〔解答〕
路径：操作系统以目录形式管理磁盘上的文件，文件所在的分区和目录就是该文件的路径。
绝对路径：从根目录到文件所在目录的完整路径称为“绝对路径”。是保证文件唯一性的标示方法。
相对路径：从系统当前目录到文件所在目录的路径称为相对路径。
当前目录：用户当前所在的目录就是当前目录。
指明的路径不正确，或者执行了另外一个同名的文件。
〔习题1.9〕
什么是摩尔定律？它能永久成立吗？
〔解答〕
每18个月，集成电路的性能将提高一倍，而其价格将降低一半。（1965年，Intel公司的创始人之一摩尔预言：集成电路上的晶体管密度每年将翻倍。现在这个预言通常表达为：每隔18个月硅片密度（晶体管容量）将翻倍；也常被表达为：每18个月，集成电路的性能将提高一倍，而其价格将降低一半。）
不能。由于电子器件的物理极限在悄然逼近，摩尔定律不会永远持续。
〔习题1.10〕
冯·诺依曼计算机的基本设计思想是什么？
〔解答〕
采用二进制形式表示数据和指令。指令由操作码和地址码组成。
将程序和数据存放在存储器中，计算机在工作时从存储器取出指令加以执行，自动完成计算任务。这就是“存储程序”和“程序控制”（简称存储程序控制）的概念。
指令的执行是顺序的，即一般按照指令在存储器中存放的顺序执行，程序分支由转移指令实现。
计算机由存储器、运算器、控制器、输入设备和输出设备五大基本部件组成，并规定了5部分的基本功能。
〔习题1.11〕
计算机系统通常划分为哪几个层次？普通计算机用户和软件开发人员对计算机系统的认识一样吗？
〔解答〕
最上层是用户层。

第5层是高级语言层。
第4层是汇编语言层。
第3层是操作系统层。
第2层是机器语言层。
第1层是控制层。
第0层是数字电路层。
普通计算机用户和软件人员对计算机系统的认识并不一样。普通计算机用户看到的计算机，也就是我们最熟悉的计算机，属于用户层，而软件人员看到的属于高级语言层或是汇编语言层。
〔习题1.12〕
什么是系列机和兼容机？你怎样理解计算机中的“兼容”特性？例如，你可以用PC机为例，谈谈你对软件兼容（或兼容性）的认识，说明为什么PC机具有如此强大的生命力？
〔解答〕
系列机是指在一个厂家生产的具有相同计算机结构，但具有不同组成和实现的一系列（Family）不同档次、不同型号的机器。
兼容机是指不同厂家生产的具有相同计算机结构（不同的组成和实现）的计算机。
兼容是一个广泛的概念，包括软件兼容、硬件兼容、系统兼容等。其中软件兼容是指同一个软件可以不加修改地运行于体系结构相同的各档机器，结果一样但运行时间可能不同。软件兼容可从机器性能和推出时间分成向上（向下）和向前（向后）兼容。例如32位PC机就陆续增加了对浮点处理指令、多媒体指令等的支持。在保证向后兼容的前提下，不断改进其组成和实现，延续计算机结构的生命，才使得PC机具有如此强大的生命力。
〔习题1.13〕
英特尔公司最新Intel 80x86处理器是什么？请通过查阅相关资料（如英特尔公司网站），说明其主要特点和采用的新技术。
〔解答〕
酷睿2多核处理器。
〔习题1.14〕
说明高级语言、汇编语言、机器语言三者的区别，谈谈你对汇编语言的认识。
〔解答〕
高级语言与具体的计算机硬件无关，其表达方式接近于所描述的问题，易为人们接受和掌握，用高级语言编写程序要比低级语言容易得多，并大大简化了程序的编制和调试，使编程效率得到大幅度的提高。而汇编语言是为了便于理解与记忆，将机器指令用助记符代替而形成的一种语言。汇编语言的语句通常与机器指令对应，因此，汇编语言与具体的计算机有关，属于低级语言。它比机器语言直观，容易理解和记忆，用汇编语言编写的程序也比机器语言易阅读、易排错。机器语言的每一条机器指令都是二进制形式的指令代码，计算机硬件可以直接识别。高级语言程序通常也需要翻译成汇编语言程序，再进一步翻译成机器语言代码。
〔习题1.15〕
为了更好地进行编程实践，请进入Windows操作系统下的控制台环境（或MS-DOS模拟环境），练习常用命令。
第2章 处理器结构
〔习题2.1〕简答题
〔解答〕
① ALU是算术逻辑运算单元，负责处理器所能进行的各种运算，主要是算术运算和逻辑运算。
② 取指是指从主存取出指令代码通过总线传输到处理器内部指令寄存器的过程。8086分成总线接口单元和指令执行单元，可以独立操作。在执行单元执行一条指令的同时，总线接口单元可以读取下一条指令，等到执行时不需要进行取指了，所以称为预取。
③ Pentium采用分离的Cache结构，一个用做指令Cache，一个用做数据Cache。
④ 堆栈的存取原则是先进后出（也称为后进先出）操作方式存取数据。
⑤ 标志寄存器主要保存反映指令执行结果和控制指令执行形式的有关状态。
⑥ 执行了一条加法指令后，发现ZF＝1，表明运算结果为0。
⑦ 没有。
⑧ 汇编语言的标识符大小写不敏感，即表示字母大小写不同、但表示同一个符号。
⑨ 不会。
⑩ 指令的操作数需要通过存储器地址或I/O地址，才能查找到数据本身，故称数据寻址方

式。
〔习题2.2〕判断题

〔解答〕
① 错 ② 对 ③ 对 ④ 对 ⑤ 错
⑥ 对 ⑦ 对 ⑧ 错 ⑨ 对 ⑩ 对
〔习题2.3〕填空题

〔解答〕
① 32，DX，DH
② 16
③ 段地址，偏移地址，EIP，IP
④ 00100110，0
⑤ 73C00H，73800H
⑥ EBX，ECX，ESI，EDI，EBP，ESP
⑦ 实地址，64KB
⑧ ASM，目标模块，FLAT
⑨ 立即数寻址、寄存器寻址和存储器寻址
⑩ DS，SS
〔习题2.4〕
处理器内部具有哪3个基本部分？8086分为哪两大功能部件？其各自的主要功能是什么？
〔解答〕
处理器内部有ALU、寄存器和指令处理三个基本单元。
8086有两大功能部件：总线接口单元和执行单元。
总线接口单元：管理着8086与系统总线的接口，负责处理器对存储器和外设进行访问。8086所有对外操作必须通过BIU和这些总线进行。
执行单元EU：负责指令译码、数据运算和指令执行。
〔习题2.5〕
8086怎样实现了最简单的指令流水线？
〔解答〕
8086中，指令的读取是在BIU单元，而指令的执行是在EU单元。因为BIU和EU两个单元相互独立、分别完成各自操作，所以可以并行操作。也就是说，在EU单元对一个指令进行译码执行时，BIU单元可以同时对后续指令进行读取；这就是最简单的指令流水线技术。
〔习题2.6〕
什么是标志？什么是IA-32处理器的状态标志、控制标志和系统标志？说明状态标志在标志寄存器EFLAGS的位置和含义。
〔解答〕
标志：用于反映指令执行结果或控制指令执行形式的一个或多个二进制数位。例如，有些指令执行后会影响有关标志位；有些指令的执行要利用相关标志。
状态标志：用来记录程序运行结果的状态信息。
控制标志：DF标志，控制字符串操作的地址方向。
系统标志：用于控制处理器执行指令的方式。
状态标志在标志寄存器EFLAGS中的位置和含义如下：

31	11	10	9	8	7	6	5	4	3
2	1	0							
……	OF				SF	ZF		AF	
PF		CF							

〔习题2.7〕
举例说明CF和OF标志的差异。
〔解答〕
进位标志CF表示无符号数运算结果是否超出范围，超出范围后加上进位或借位，运算结果仍然正确；溢出标志OF表示有符号数运算结果是否超出范围，如果超出范围，运算结果已经不正确。
例1：3AH ＋ 7CH＝B6H
无符号数运算：58＋124＝182，范围内，无进位。

有符号数运算：58＋124＝182，范围外，有溢出。
例2：AAH + 7CH＝①26H
无符号数运算：170＋124＝294，范围外，有进位。
有符号数运算：－86＋124＝28 ，范围内，无溢出。
〔习题2.8〕
什么是8086中的逻辑地址和物理地址？逻辑地址如何转换成物理地址？请将如下逻辑地址用物理地址表达（均为十六进制形式）：
① FFFF：0 ② 40：17 ③ 2000：4500 ④ B821：4567
〔解答〕
物理地址：在处理器地址总线上输出的地址称为物理地址。每个存储单元有一个唯一的物理地址。
逻辑地址：在处理器内部、程序员编程时采用逻辑地址，采用“段地址：偏移地址“形式。某个存储单元可以有多个逻辑地址，即处于不同起点的逻辑段中，但其物理地址是唯一的。
逻辑地址转换成物理地址：逻辑地址由处理器在输出之前转换为物理地址。将逻辑地址中的段地址左移二进制4位（对应16进制是一位，即乘以16），加上偏移地址就得到20位物理地址。
① FFFFH:0＝FFFF0H
② 40H:17H＝00417H
③ 2000H:4500H＝24500H
④ B821H:4567H＝BC777H
〔习题2.9〕
IA-32处理器有哪三类基本段，各是什么用途？
〔解答〕
IA-32处理器有代码段、数据段、堆栈段三类基本段。
代码段：存放程序的指令代码。程序的指令代码必须安排在代码段，否则将无法正常执行。
数据段：存放当前运行程序所用的数据。程序中的数据默认是存放在数据段，也可以存放在其他逻辑段中。
堆栈段：主存中堆栈所在的区域。程序使用的堆栈一定在堆栈段。
〔习题2.10〕
什么是平展存储模型、段式存储模型和实地址存储模型？
〔解答〕
平展存储模型下，对程序来说存储器是一个连续的地址空间，称为线性地址空间。程序需要的代码、数据和堆栈都包含在这个地址空间中。
段式存储模型下，对程序来说存储器由一组独立的地址空间组成，独立的地址空间称为段。通常，代码、数据和堆栈位于分开的段中。
实地址存储模型是8086处理器的存储模型。它是段式存储模型的特例，其线性地址空间最大为1MB容量，由最大为64KB的多个段组成。
〔习题2.11〕
什么是实地址方式、保护方式和虚拟8086方式？它们分别使用什么存储模型？
〔解答〕
实地址方式：与8086具有相同的基本结构，只能寻址1MB物理存储器空间，逻辑段最大不超过64KB；但可以使用32位寄存器、32位操作数和32位寻址方式；相当于可以进行32位处理的快速8086。实地址工作方式只能支持实地址存储模型。
保护方式：具有强大的段页式存储管理和特权与保护能力，使用全部32条地址总线，可寻址4GB物理存储器。保护方式通过描述符实现分段存储管理，每个逻辑段可达4GB。处理器工作在保护方式时，可以使用平展或段式存储模型。
虚拟8086方式：在保护方式下运行的类似实方式的运行环境，只能在1MB存储空间下使用“16位段”。 处理器工作在虚拟8086方式时，只能使用实地址存储模型。
〔习题2.12〕
汇编语句有哪两种，每个语句由哪4个部分组成？
〔解答〕
汇编语句有两种：执行性语句（处理器指令）、说明性语句（伪指令）。
每个语句有：标号、指令助记符、操作数或参数、注释4个部分组成。
〔习题2.13〕
汇编语言程序的开发有哪4个步骤，分别利用什么程序完成、产生什么输出文件。
〔解答〕

汇编语言程序的开发有4个步骤：
编辑：用文本编辑器形成一个以ASM为扩展名的源程序文件。
汇编：用汇编程序将ASM文件转换为OBJ模块文件。
连接：用连接程序将一个或多个目标文件链接成一个EXE或COM可执行文件。
调试：用调试程序排除错误，生成正确的可执行文件。
〔习题2.14〕
MASM汇编语言中，下面哪些是程序员可以使用的正确的标识符。
FFH，DS，0xvab，Again，next，@data，h_ascii，6364b，.exit，small
〔解答〕
FFH，Again，next，h_ascii
〔习题2.15〕
给出IA-32处理器的32位寻址方式和16位寻址方式的组成公式，并说明各部分作用。
〔解答〕
① 32位存储器寻址方式的组成公式
32位有效地址 ＝ 基址寄存器＋（变址寄存器×比例）＋位移量
其中的4个组成部分是：
• 基址寄存器任何8个32位通用寄存器之一；
• 变址寄存器除ESP之外的任何32位通用寄存器之一；
• 比例可以是1，2，4或8（因为操作数的长度可以是1，2，4或8字节）；
• 位移量可以是8或32位有符号值。
② 16位存储器寻址方式的组成公式
16位有效地址 ＝ 基址寄存器＋变址寄存器＋位移量
其中基址寄存器只能是BX或BP，变址寄存器只能是SI或DI，位移量是8或16位有符号值。
〔习题2.16〕
说明下列指令中源操作数的寻址方式？假设VARD是一个双字变量。
（1）mov edx,1234h
（2）mov edx,vard
（3）mov edx,ebx
（4）mov edx,[ebx]
（5）mov edx,[ebx+1234h]
（6）mov edx,vard[ebx]
（7）mov edx,[ebx+edi]
（8）mov edx,[ebx+edi+1234h]
（9）mov edx,vard[esi+edi]
（10）mov edx,[ebp*4]
〔解答〕
① 立即数
② 直接
③ 寄存器
④ 寄存器间接
⑤ 寄存器相对
⑥ 寄存器相对
⑦ 基址变址
⑧ 相对基址变址
⑨ 相对基址变址
⑩ 带比例寻址
〔习题2.17〕
使用本书配套的软件包（或者按照本书说明）创建MASM开发环境，通过编辑例题2-1和例题2-2程序、汇编连接生成可执行程序和列表文件，掌握汇编语言的开发。
第3章 数据处理
〔习题3.1〕简答题
〔解答〕
① 没有。使用二进制8位表达无符号整数，257没有对应的编码。
② 字符“’F’”的ASCII码就是数值46H，所以没有区别。
③ 汇编程序在汇编过程中对数值表达式计算，得到一个确定的数值，故称数值表达式为常量。

④ 不能。数值500大于一个字节所能表达的数据量，所以不能为字节变量赋值。
⑤ 源、目标寄存器位数不同，不能用该指令进行数据交换。
⑥ 前者在指令执行时获得偏移地址，是正确的；但后者的OFFSET只能在汇编阶段获得偏移地址，但此时寄存器内容是不可知的，所以无法获得偏移地址。
⑦ INC，DEC，NEG和NOT指令的操作数既是源操作数也是目的操作数。
⑧ 大小写字母转换利用它们的ASCII码相差20H。
⑨ 加减法不区别无符号数和有符号数，但根据运算结果分别设置标志寄存器的CF和OF标志，可利用CF和OF进行区别。
⑩ 逻辑与运算规则类似二进制的乘法，所以称其为逻辑乘。
〔习题3.2〕判断题

〔解答〕
① 对　② 对　③ 对　④ 错　⑤ 错
⑥ 对　⑦ 错　⑧ 错　⑨ 对　⑩ 对
〔习题3.3〕填空题
〔解答〕
① BYTE，OFFSET
② 97，61，小写字母a
③ 0DH（13），0AH（10）
④ 8843H
⑤ DWORD，4，WORD PTR XYZ
⑥ 3
⑦ 78894111
⑧ 0，0，0
⑨ 0123456788765432H，83H
⑩ 4
〔习题3.4〕
下列十六进制数表示无符号整数，请转换为十进制形式的真值：
① FFH　② 0H　③ 5EH　④ EFH
〔解答〕
① 255
② 0
③ 94
④ 239
〔习题3.5〕
将下列十进制数真值转换为压缩BCD码：
① 12　② 24　③ 68　④ 99
〔解答〕
① 12H
② 24H
③ 68H
④ 99H
〔习题3.6〕
将下列压缩BCD码转换为十进制数：
① 10010001　② 10001001　③ 00110110　④ 10010000
〔解答〕
① 91
② 89
③ 36
④ 90
〔习题3.7〕
将下列十进制数用8位二进制补码表示：
① 0　② 127　③－127　④ －57
〔解答〕
① 00000000
② 01111111

③ 10000001
④ 11000111
〔习题3.8〕
进行十六进制数据的加减运算，并说明是否有进位或借位：
① 1234H＋7802H
② F034H＋5AB0H
③ C051H－1234H
④ 9876H－ABCDH
〔解答〕
① 1234H＋7802H＝8A36H，无进位
② F034H＋5AB0H＝4AF4H，有进位
③ C051H－1234H＝BE1DH，无借位
④ 9876H－ABCDH＝ECA9H，有借位
〔习题3.9〕
数码0～9、大写字母A～Z、小写字母a～z对应的ASCII码分别是多少？ASCII码0DH和0AH分别对应什么字符？
〔解答〕
数码0～9对应的ASCII码依次是30H～39H。
大写字母A～Z对应的ASCII码依次是：41H～5AH。
小写字母a～z对应的ASCII码依次是：61～7AH。
ASCII码0DH和0AH分别对应的是回车和换行字符。
〔习题3.10〕
设置一个数据段，按照如下要求定义变量或符号常量：
① my1b为字符串变量：Personal Computer
② my2b为用十进制数表示的字节变量：20
③ my3b为用十六进制数表示的字节变量：20
④ my4b为用二进制数表示的字节变量：20
⑤ my5w为20个未赋值的字变量
⑥ my6c为100的常量
⑦ my7c表示字符串：Personal Computer
〔解答〕

```
my1b    byte 'Personal Computer'
my2b    byte 20
my3b    byte 14h
my4b    byte 00010100b
my5w    word 20 dup(?)
my6c    = 100
my7c    equ <Personal Computer>
```

〔习题3.11〕
定义常量NUM，其值为5；数据段中定义字数组变量DATALIST，它的头5个字单元中依次存放-10，2，5和4，最后1个单元初值不定。
〔解答〕

```
num     equ 5
datalist        byte -10,2,5,4,?
```

〔习题3.12〕
从低地址开始以字节为单位，用十六进制形式给出下列语句依次分配的数值：

```
        byte 'ABC',10,10h,'EF',3 dup(-1,?,3 dup(4))
        word 10h,-5,3 dup(?)
```

〔解答〕

```
41 42 43 0A 10 45 46 FF 00 04 04 04 FF 00 04 04 04 FF 00 04 04 04
10 00 FB FF 00 00 00 00 00 00
```

〔习题3.13〕
设在某个程序中有如下片段，请写出每条传送指令执行后寄存器EAX的内容：

```
        ; 数据段
        org 100h
varw    word 1234h,5678h
```

```
varb    byte 3,4
vard    dword 12345678h
buff    byte 10 dup(?)
mess    byte 'hello'
        ; 代码段
        mov eax,offset mess
        mov eax,type buff+type mess+type vard
        mov eax,sizeof varw+sizeof buff+sizeof mess
        mov eax,lengthof varw+lengthof vard
```

〔解答〕

① EAX＝0114H
② EAX＝0006H
③ EAX＝0013H
④ EAX＝0003H

〔习题3.14〕

按照如下输出格式，在屏幕上显示ASCII表：

```
   | 0 1 2 3 4 5 6 7 8 9 A B C D E F
---+--------------------------------
20 |   ! " # ...
30 | 0 1 2 3 ...
40 | @ A B C ...
50 | P Q R S ...
60 |  ` a b c ...
70 | p q r s ...
```

表格最上一行的数字是对应列ASCII代码值的低4位（用十六进制形式），而表格左边的数字对应行ASCII代码值的高4位（用十六进制形式）。编程在数据段直接构造这样的表格、填写相应ASCII代码值（不是字符本身），然后使用字符串显示子程序DISPMSG实现显示。

〔解答〕

```
        include io32.inc
        .data
table   byte '   |0 1 2 3 4 5 6 7 8 9 A B C D E F',13,10
        byte '---+--------------------------------',13,10
        byte '20
|',20h,20h,21h,20h,22h,20h,23h,20h,24h,20h,25h,20h,26h,20h,27h,20h,28h,20h,29h,2
0h
        byte 2ah,20h,2bh,20h,2ch,20h,2dh,20h,2eh,20h,2fh,20h,13,10
        byte '30
|',30h,20h,31h,20h,32h,20h,33h,20h,34h,20h,35h,20h,36h,20h,37h,20h,38h,20h,39h,2
0h
        byte 3ah,20h,3bh,20h,3ch,20h,3dh,20h,3eh,20h,3fh,20h,13,10
        byte '40
|',40h,20h,41h,20h,42h,20h,43h,20h,44h,20h,45h,20h,46h,20h,47h,20h,48h,20h,49h,2
0h
        byte 4ah,20h,4bh,20h,4ch,20h,4dh,20h,4eh,20h,4fh,20h,13,10
        byte '50
|',50h,20h,51h,20h,52h,20h,53h,20h,54h,20h,55h,20h,56h,20h,57h,20h,58h,20h,59h,2
0h
        byte 5ah,20h,5bh,20h,5ch,20h,5dh,20h,5eh,20h,5fh,20h,13,10
        byte '60
|',60h,20h,61h,20h,62h,20h,63h,20h,64h,20h,65h,20h,66h,20h,67h,20h,68h,20h,69h,2
0h
        byte 6ah,20h,6bh,20h,6ch,20h,6dh,20h,6eh,20h,6fh,20h,13,10
        byte '70
|',70h,20h,71h,20h,72h,20h,73h,20h,74h,20h,75h,20h,76h,20h,77h,20h,78h,20h,79h,2
0h
        byte 7ah,20h,7bh,20h,7ch,20h,7dh,20h,7eh,20h,7fh,20h,13,10
```

```
        byte 0
        .code
start:
        mov eax,offset table
        call dispmsg
        exit 0
        end start
```

〔习题3.15〕

数据段有如下定义，IA-32处理器将以小端方式保存在主存：

```
var     dword 12345678h
```

现以字节为单位按地址从低到高的顺序，写出这个变量内容。并说明如下指令的执行结果：

```
        mov eax,var     ; EAX＝__________
        mov bx,var      ; BX＝__________
        mov cx,var+2    ; CX＝__________
        mov dl,var      ; DL＝__________
        mov dh,var+3    ; DH＝__________
```

可以编程使用十六进制字节显示子程序DSIPHB顺序显示各个字节进行验证，还可以使用十六进制双字显示子程序DSIPHD显示该数据进行对比。

〔解答〕

小端方式采用“低对低、高对高”，即低字节数据存放在低地址存储单元、高字节数据存放在高地址存储单元。以字节为单位按地址从低到高的顺序，var变量的内容：78H、56H、34H、12H。

```
        ; EAX＝12345678H
        ; BX＝5678H
        ; CX＝1234H
        ; DL＝78H
        ; DH＝12H
```

〔习题3.16〕

使用若干MOV指令实现交互指令“XCHG EBX,[EDI]”功能。

〔解答〕

```
        push eax        ; 可以没有
        mov eax,ebx
        mov ebx,[edi]
        mov [edi],eax
        pop eax ; 可以没有
```

〔习题3.17〕

假设当前ESP＝0012FFB0H，说明下面每条指令后，ESP等于多少？

```
        push eax
        push dx
        push dword ptr 0f79h
        pop eax
        pop word ptr [bx]
        pop ebx
```

〔解答〕

ESP＝0012FFACH
ESP＝0012FFAAH
ESP＝0012FFA6H
ESP＝0012FFAAH
ESP＝0012FFACH
ESP＝0012FFB0H

〔习题3.18〕

已知数字0～9对应的格雷码依次为：18H、34H、05H、06H、09H、0AH、0CH、11H、12H、14H；请为如下程序的每条指令加上注释，说明每条指令的功能和执行结果。

```
        ; 数据段
table   byte 18h,34h,05h,06h,09h,0ah,0ch,11h,12h,14h
        ; 代码段
```

```
        mov ebx,offset table
        mov al,8
        xlat
```

为了验证你的判断，不妨使用本书的I/O子程序库提供的子程序DISPHB显示换码后AL的值。如果不使用XLAT指令，应如何修改？

〔解答〕

```
        ; 数据段
table   byte 18h,34h,05h,06h,09h,0ah,0ch,11h,12h,14h     ; 定义格雷码表
        ; 代码段
        mov ebx,offset table     ; EBX＝格雷码表首地址
        mov al,8                 ; AL＝8
        xlat            ; AL＝12H（8的格雷码）
```

不使用XLAT指令：

```
        mov ebx,offset table     ; EBX＝格雷码表首地址
        mov eax,0
        mov al,8                 ; AL＝8
        mov al,[eax+ebx]                 ; AL＝12H（8的格雷码）
```

〔习题3.19〕

请分别用一条汇编语言指令完成如下功能：

（1）把EBX寄存器和EDX寄存器的内容相加，结果存入EDX寄存器。

（2）用寄存器EBX和ESI的基址变址寻址方式把存储器的一个字节与AL寄存器的内容相加，并把结果送到AL中。

（3）用EBX和位移量0B2H的寄存器相对寻址方式把存储器中的一个双字和ECX寄存器的内容相加，并把结果送回存储器中。

（4）将32位变量VARD与数3412H相加，并把结果送回该存储单元中。

（5）把数0A0H与EAX寄存器的内容相加，并把结果送回EAX中。

〔解答〕

```
        ① add edx,ebx
        ② add al,[ebx+esi]
        ③ add [bx+0b2h],cx
        ④ add varw,3412h
        ⑤ add eax,0a0h
```

〔习题3.20〕

分别执行如下程序片断，说明每条指令的执行结果：

（

〔解答〕

（1）

```
        ; EAX＝80H
        ; EAX＝83H，CF＝0，SF＝0
        ; EAX＝103H，CF＝0，OF＝0
        ; EAX＝106H，CF＝0，ZF＝0
```

（2）

```
        ; EAX＝100
        ; EAX＝300，CF＝0
```

（3）

```
        ; EAX＝100
        ; EAX＝44，CF＝1（包含256的进位含义：256＋44＝300）
```

（4）

```
        mov al,7fh       ; AL＝7FH
        sub al,8         ; AL＝77H，CF＝0，SF＝0
        sub al,80h       ; AL＝F7H，CF＝1，OF＝1
        sbb al,3         ; AL＝F3H，CF＝0，ZF＝0
```

〔习题3.21〕

给出下列各条指令执行后AL值，以及CF、ZF、SF、OF和PF的状态：

```
        mov al,89h
        add al,al
```

```
        add al,9dh
        cmp al,0bch
        sub al,al
        dec al
        inc al
```

〔解答〕

```
        mov al,89h      ; AL＝89H       CF      ZF      SF      OF      PF
        add al,al       ; AL＝12H       1       0       0       1       1
        add al,9dh      ; AL＝0AFH      0       0       1       0       1
        cmp al,0bch     ; AL＝0AFH      1       0       1       0       1
        sub al,al       ; AL＝00H       0       1       0       0       1
        dec al  ; AL＝0FFH      0       0       1       0       1
        inc al  ; AL＝00H       0       1       0       0       1
```

〔习题3.22〕

有两个64位无符号整数存放在变量buffer1和buffer2中，定义数据、编写代码完成EDX.EAX←buffer1－buffer2功能。

〔解答〕

```
        ; 数据段
buffer1 qword 67883000h
buffer2 qword 67762000h
        ; 代码段
        mov eax,dword ptr buffer1
        mov edx,dword ptr buffer1+4
        sub eax,dword ptr buffer2
        sbb edx,dword ptr buffer2+4
```

〔习题3.23〕

分别执行如下程序片断，说明每条指令的执行结果：

〔解答〕

（1）

```
        ; ESI＝9CH
        ; ESI＝80H
        ; ESI＝FFH
        ; ESI＝01H
```

（2）

```
        ; EAX＝1010B（可以有前导0，下同）
        ; EAX＝0010B，CF＝1
        ; EAX＝0100B，CF＝0
        ; EAX＝0000B，CF＝0
```

（3）

```
        ; EAX＝1011B（可以有前导0，下同）
        ; EAX＝101100B，CF＝0
        ; EAX＝10110B，CF＝0
        ; EAX＝10111B，CF＝0
```

（4）

```
        ; EAX＝0，CF＝0，OF＝0
        ;  ZF＝1，SF＝0，PF＝1
```

〔习题3.24〕

3.24 给出下列各条指令执行后AX的结果，以及状态标志CF、OF、SF、ZF、PF的状态。

```
        mov ax,1470h
        and ax,ax
        or ax,ax
        xor ax,ax
        not ax
        test ax,0f0f0h
```

〔解答〕

```
        mov ax,1470h    ; AX＝1470H     CF  OF  SF      ZF      PF
```

```
        and ax,ax          ; AX＝1470H     0       0       0       0       0
        or ax,ax           ; AX＝1470H     0       0       0       0       0
        xor ax,ax          ; AX＝0000H     0       0       0       1       1
        not ax   ; AX＝FFFFH      0        0       0       1       1
        test ax,0f0f0h   ; AX＝0F0F0H     0       0       1       0       1
```

〔习题3.25〕
逻辑运算指令怎么实现复位、置位和求反功能？
〔解答〕
AND指令同“0”与实现复位，OR指令同“1”或实现置位，XOR同“1”异或实现求反。
〔习题3.26〕
说明如下程序段的功能：

```
        mov ecx,16
        mov bx,ax
next:   shr ax,1
        rcr edx,1
        shr bx,1
        rcr edx,1
        loop next
        mov eax,edx
```

〔解答〕
将AX的每一位依次重复一次，所得的32位结果保存于EAX中。
〔习题3.27〕
编程将一个64位数据逻辑左移3位，假设这个数据已经保存在EDX.EAX寄存器对中。
〔解答〕

```
        ; 代码段
        mov ecx,3
again:  shl eax,1
        rcl edx,1
        loop again
```

〔习题3.28〕
编程将一个压缩BCD码变量（例如92H）转换为对应的ASCII码，然后调用DISPC子程序（在输入输出子程序库中）显示。
〔解答〕

```
        ; 数据段
bcd     byte 92h
        ; 代码段
        mov al,bcd
        shr al,4
        add al,30h
        call dispc
        mov al,bcd
        and al,0fh
        add al,30h
        call dispc
```

〔习题3.29〕
以MOVS指令为例，说明串操作指令的寻址特点，并用MOV和ADD等指令实现MOVSD的功能（假设DF＝0）。
〔解答〕
MOVS指令的功能是：
ES:[EDI]←DS:[ESI]；ESI←ESI±1/2/4，EDI←EDI±1/2/4
由此可看出串操作指令的寻址特点：
源操作数用寄存器ESI间接寻址，默认在DS指向的数据段，但可以改变；目的操作数用寄存器EDI间接寻址，只能在ES指向的附加数据段；每执行一次串操作，源指针ESI和目的指针EDI将自动修改：±1（字节），±2（字）或±4（双字）。指针的增量和减量控制由DF标志确定，DF＝0，进行增量；DF＝1，进行减量。

```
        push eax
```

```
        mov eax,[esi]
        mov es:[edi],eax
        add esi,4
        add edi,4
```

〔习题3.30〕
说明如下程序执行后的显示结果：

```
        ; 数据段
msg     byte 'WELLDONE',0
        ; 代码段
        mov ecx,(lengthof msg)-1
        mov ebx,offset msg
again:  mov al,[ebx]
        add al,20h
        mov [ebx],al
        add ebx,1
        loop again
        mov eax,offset msg
        call dispmsg
```

如果将其中语句“mov ebx,offset msg”改为“xor ebx,ebx”，则利用EBX间接寻址的两个语句如何修改成EBX寄存器相对寻址，就可以实现同样功能？
〔解答〕
显示结果：welldone
EBX寄存器相对寻址：

```
        mov al,msg[ebx]
        mov msg[ebx],al
```

〔习题3.31〕
下面程序的功能是将数组ARRAY1的每个元素加固定值（8000H），将和保存在数组ARRAY2。在空白处填入适当的语句或语句的一部分。

```
        ; 数据段
array1  dword 1,2,3,4,5,6,7,8,9,10
array2  dword 10 dup(?)
        ; 代码段
        mov ecx,lengthof array1
        mov ebx,0
again:  mov eax,array1[ebx*4]
        add eax,8000h
        mov _____________
        add ebx,__________
        loop again
```

〔解答〕

```
        mov array2[ebx*4],eax
        add ebx,1
```

〔习题3.32〕
上机实现本章的例题程序，编程实现本章的习题程序。
第4章 汇编语言程序设计
〔习题4.1〕简答题
〔解答〕
① 当同一个程序被操作系统安排到不同的存储区域执行时，指令间的位移没有改变，目标地址采用相对寻址可方便操作系统的灵活调度。
② 数据通信时，数据的某一位用做传输数据的奇偶校验位，数据中包括校验位在内的“1”的个数恒为奇数，就是奇校验；恒为偶数，就是偶校验。
③ 无符号数和有符号数的操作影响两组不同的标志状态位，故判断两个无符号数和有符号数的大小关系要利用不同的标志位组合，所以有对应的两组指令。
④ 双分支结构中两个分支体之间的JMP指令，用于实现结束前一个分支回到共同的出口作用。
⑤ 完整的子程序注释可方便程序员调用该子程序，子程序注释包括子程序名、子程序功

能、入口参数和出口参数、调用注意事项和其他说明等。
⑥ 子程序保持堆栈平衡，才能保证执行RET指令时当前栈顶的内容是正确的返回地址。主程序也要保持堆栈平衡，这样才能释放传递参数占用的堆栈空间，否则多次调用该子程序可能就致使堆栈溢出。
⑦ “传值”是传递参数的一个拷贝，被调用程序改变这个参数不影响调用程序；“传址”时，被调用程序可能修改通过地址引用的变量内容。
⑧ INCLUDE语句包含的是文本文件、是源程序文件的一部分；INCLUDELIB语句包含的是子程序库文件。
⑨ 取长补短。
⑩ Windows程序在运行时需要加载其配套的动态链接库DLL文件，当其没有被搜索到时就会提示不存在。

〔习题4.2〕判断题
〔解答〕
① 对　② 错　③ 错　④ 错　⑤ 错
⑥ 对　⑦ 对　⑧ 错　⑨ 对　⑩ 错

〔习题4.3〕填空题
〔解答〕
① 相对寻址，间接寻址，直接寻址，间接寻址
② 1256H，3280H
③ 3721H，1
④ EAH
⑤ 循环初始，循环控制
⑥ REPT1标号的地址
⑦ TEST ENDP，ENDM
⑧ EBP
⑨ PUBLIC，EXTERN
⑩ 38H 0DH 0AH

〔习题4.4〕
为了验证例题4-1程序的执行路径，可以在每个标号前后增加显示功能。例如使得程序运行后显示数码1234。
〔解答〕

```
        jmp labl1        ; 相对寻址
        nop
        mov eax,'?'
        call dispc
labl1:  mov eax,'1'
        call dispc
        jmp near ptr labl2       ; 相对近转移
        nop
        mov eax,'?'
        call dispc
labl2:  mov eax,'2'
        call dispc
        mov eax,offset labl3
        jmp eax ; 寄存器间接寻址
        nop
        mov eax,'?'
        call dispc
labl3:  mov eax,'3'
        call dispc
        mov eax,offset labl4
        mov nvar,eax
        jmp nvar         ; 存储器间接寻址
        nop
        mov eax,'?'
        call dispc
```

```
lab14:  mov eax,'4'
        call dispc
```

〔习题4.5〕

使用“SHR EAX,2”将EAX中的D1位移入CF标志，然后用JC/JNC指令替代JZ/JNZ指令完成例题4-3的功能。

〔解答〕

```
        mov eax,56h      ; 假设一个数据
        shr eax,2        ; D1位移入CF标志
        jnc nom ; D1＝0条件成立，转移
        ...      ; 余同原程序
```

〔习题4.6〕

执行如下程序片断后，CMP指令分别使得5个状态标志CF、ZF、SF、OF和PF为0还是为1？它会使得哪些条件转移指令指令Jcc的条件成立、发生转移？

```
        mov eax,20h
        cmp eax,80h
```

〔解答〕

CF＝1 ZF＝0 SF＝1 OF＝0 PF＝1

可以使得条件成立、发生转移的指令有：JC JS JP JNZ JNO

〔习题4.7〕

将例题4-4程序修改为实现偶校验。建议进一步增加显示有关提示信息的功能，使得程序具有更加良好的交互性。

〔解答〕

```
include io32.inc
        .data
msg1 byte 'Please input a character:  ',0
msg2 byte 'The ASCII code of the charater you entered is:  ',0
msg3 byte 'The code with even parity is:  ',0
        .code
start:
        mov eax,offset msg1
        call dispmsg
        call readc
        call dispcrlf
        mov ebx,eax
        mov eax,offset msg2
        call dispmsg
        mov eax,ebx
        call dispbb
        call dispcrlf
        and al,7fh
        jp next
        or al,80h
next:   mov ebx,eax
        mov eax,offset msg3
        call dispmsg
        mov eax,ebx
        call dispbb
        exit 0
        end start
```

〔习题4.8〕

在采用奇偶校验传输数据的接收端应该验证数据传输的正确性。例如，如果采用偶校验，那么在接收到的数据中，其包含“1”的个数应该为0或偶数个，否则说明出现传输错误。现在，在接收端编写一个这样的程序，如果偶校验不正确显示错误信息，传输正确则继续。假设传送字节数据、最高位作为校验位，接收到的数据已经保存在Rdata变量中。

〔解答〕

```
        ; 数据段
```

```
Rdata   byte 57h                ;保存接收的数据
error   byte 'Error !',0
        ; 代码段
        mov al,Rdata
        and al,0ffh             ;标志PF反映“1”的个数
        jp done         ;个数为偶数，正确继续
        mov eax,offset error    ;个数为奇数，显示出错
        call dispmsg
done:
```

〔习题4.9〕
指令CDQ将EAX符号扩展到EDX，即：EAX最高为0，则EDX＝0；EAX最高为1，则EDX＝FFFFFFFFH。请编程实现该指令功能。
〔解答1〕

```
        test eax,8000h  ; 测试最高位
        jz next1        ; 最高位为0（ZF＝1），转移到标号NEXT1
        mov edx,0ffffffffh      ; 最高位为1，顺序执行：设置EDX＝FFFFFFFFH
        jmp done        ; 无条件转移，跳过另一个分支
next1:  mov dx,0        ; 最高位为0转移到此执行：设置EDX＝0
done:
```

〔解答2〕
使用移位指令更好。

```
        rol eax,1
        rcr edx,1
        sar edx,31
        ror eax,1
```

〔习题4.10〕
编程，首先测试双字变量DVAR的最高位，如果为1，则显示字母“L”；如果最高位不为1，则继续测试最低位，如果最低位为1，则显示字母“R”；如果最低位也不为1，则显示字母“M”。
〔解答〕

```
        ; 数据段
dvar    dword 57h
        ; 代码段
        mov eax,dvar
        test eax,80000000h
        jnz nextl
        test eax,1
        jnz nextr
        mov al,'M'
        jmp done
nextl:  mov al,'L'
        jmp done
nextr:  mov al,'R'
done:   call dispc
```

〔习题4.11〕编写一个程序，先提示输入数字“Input Number：0～9”，然后在下一行显示输入的数字，结束；如果不是键入了0～9数字，就提示错误“Error!”，继续等待输入数字。
〔解答〕

```
        ; 数据段
inmsg   byte 'Input number(0~9): ',0
ermsg   byte 0dh,0ah,'Error! Input again: ',0
        ; 代码段
        mov eax,offset inmsg    ; 提示输入数字
        call dispmsg
again:  call readc      ; 等待按键
        cmp al,'0'      ; 数字 < 0?
```

```
        jb erdisp
        cmp al,'9'      ; 数字 > 9?
        ja erdisp
        call dispcrlf
        call dispc
        jmp done
erdisp: mov eax,offset ermsg
        call dispmsg
        jmp again
done:
```

〔习题4.12〕

有一个首地址为ARRAY的20个双字的数组，说明下列程序段的功能。

```
        mov ecx,20
        mov eax,0
        mov esi,eax
sumlp:  add eax,array[esi]
        add esi,4
        loop sumlp
        mov total,eax
```

〔解答〕

求这20个双字的和，保存在TOTAL变量，不关进心进位和溢出。

〔习题4.13〕

编程中经常要记录某个字符出现的次数。现编程记录某个字符串中空格出现的次数，结果保存在SPACE单元。

〔解答〕

```
        ; 数据段
string  byte 'Do you have fun with Assembly ?',0        ;以0结尾的字符串
space   dword ?
        ; 代码段
        mov esi,offset string
        xor ebx,ebx     ;EBX用于记录空格数
again:  mov al,[esi]
        cmp al,0
        jz done
        cmp al,20h      ;空格的ASCII码是20H
        jne next        ;不相等、不是空格，转移
        inc bx  ;相等、是空格，空格个数加1
next:   inc esi
        jmp again       ;继续循环
done:   mov space,ebx   ;保存结果
```

〔习题4.14〕

编写计算100个16位正整数之和的程序。如果和不超过16位字的范围（65535），则保存其和到WORDSUM，如超过则显示‘Overflow !’。

〔解答〕

```
        ; 数据段
array   word 2005,2008,98 dup (1394)    ; 假设100个16位正整数
wordsum word ?
error   byte 'Overflow !',0
        ; 代码段
        and ebx,0
        mov ecx,100
        xor ax,ax
again:  add ax,array[ebx*2]
        jc over
        inc ebx
        loop again
```

```
        mov wordsum,ax
over:   mov eax,offset error
        call dispmsg
```

〔习题4.15〕

在一个已知长度的字符串中查找是否包含“BUG”子字符串。如果存在，显示“Y”，否则显示“N”。

〔解答〕

```
        ; 数据段
string  byte 'If you find any error in the program, you can DEBUG it.'
count   = sizeof string
bug     byte 'BUG'
        ; 代码段
        mov ecx,count
        mov edi,offset string
L1:     mov esi,offset bug
        push edi
        mov edx,sizeof bug
LN:     mov al,[esi]
        cmp [edi],al
        jne L2
        inc esi
        inc edi
        dec edx
        jne LN
        pop edi
        mov al,'Y'
        jmp L3
L2:     pop edi
        inc edi
        loop L1
        mov al,'N'
L3:     call dispc
```

〔习题4.16〕

主存中有一个8位压缩BCD码数据，保存在一个双字变量中。现在需要进行显示，但要求不显示前导0。由于位数较多，需要利用循环实现，但如何处理前导0和数据中间的0呢？不妨设置一个标记。编程实现。

〔解答〕

```
        ; 数据段
bcd     dword 00371002h
        ; 代码段
        mov esi,bcd
        cmp esi,0
        jnz goon
        mov al,'0'
        call dispc
        jmp done
goon:   mov ecx,8
        xor ebx,ebx     ; EBX＝0，表示可能是前导0
again:  rol esi,4
        mov eax,esi
        and eax,0fh     ; EAX低4位保存当前要显示的BCD码
        cmp ebx,0       ; EBX≠0，说明不是前导0，要显示
        jnz disp        ; EBX＝0，说明可能是前导0
        cmp eax,0
        jz next ; EAX＝0，说明是前导0，不显示
        mov ebx,1       ; EAX≠0，没有前导0了，令EBX＝1≠0
```

```
disp:   add al,30h
        call dispc
next:   loop again
done:
```

〔习题4.17〕
已知一个字符串的长度，剔除其中所有的空格字符。请从字符串最后一个字符开始逐个向前判断、并进行处理。
〔解答〕

```
        ; 数据段
string  byte 'Let us have  a try !',0dh,0ah,0
        ; 代码段
        mov ecx,sizeof string
        cmp ecx,2
        jb done
        lea eax,string  ; 显示处理前的字符串
        call dispmsg
        mov esi,ecx
        dec esi
outlp:  cmp string[esi],' '      ; 检测是否是空格
        jnz next         ; 不是空格继续循环
        mov edi,esi      ; 是空格，进入剔除空格分支
        dec ecx
inlp:   inc edi
        mov al,string[edi]       ; 前移一个位置
        mov string[edi-1],al
        cmp edi,ecx
        jb inlp
next:   dec esi ; 继续进行
        cmp esi,0
        jnz outlp        ; 为0结束
        lea eax,string  ; 显示处理后的字符串
        call dispmsg
done:
```

〔习题4.18〕
第3章习题3.14在屏幕上显示ASCII表，现仅在数据段设置表格缓冲区，编程将ASCII代码值填入留出位置的表格，然后调用显示功能实现（需要利用双重循环）。
〔解答〕

```
        include io32.inc
        .data
table   byte '    |0 1 2 3 4 5 6 7 8 9 A B C D E F',13,10
        byte '---+--------------------------------',13,10
tab1    byte 6 dup(36 dup(?),13,10)

        byte 0
        .code
start:
        mov ebx,offset tab1
        mov edx,'| 02'
        mov ax,2020h
        mov esi,6
again0: mov [ebx],edx
        add ebx,4
        mov ecx,16
again1: mov word ptr [ebx],ax
        add ebx,2
        inc al
```

```
        loop again1

        add ebx,2
        add edx,1
        dec esi
        jnz again0

        mov eax,offset table
        call dispmsg
        exit 0
        end start
```

〔习题4.19〕

请按如下说明编写子程序：

子程序功能：把用ASCII码表示的两位十进制数转换为压缩BCD码

入口参数：DH＝十位数的ASCII码，DL＝个位数的ASCII码

出口参数：AL＝对应BCD码

〔解答〕

```
asctob  proc
        shl dh,4
        mov al,dh
        and dl,0fh
        or al,dl
        ret
asctob  endp
```

〔习题4.20〕

乘法的非压缩BCD码调整指令AAM执行的操作是：AH←AL÷10的商，AL←AL÷10的余数。利用AAM可以实现将AL中的100内数据转换为ASCII码，程序如下：

```
        xor ah,ah
        aam
        add ax,3030h
```

利用这段程序，编写一个显示AL中数值（0～99）的子程序。

〔解答〕

```
disp99  proc
        xor ah,ah
        aam
        add ax,3030h
        push ax
        mov al,ah
        call dispc
        pop ax
        call dispc
        ret
disp99  endp
```

〔习题4.21〕

编写一个源程序，在键盘上按一个键，将其返回的ASCII码值显示出来，如果按下ESC键（对应ASCII码是1BH）则程序退出。请调用书中的HTOASC子程序。

〔解答〕

```
        ; 代码段，主程序
again:  call readc
        cmp al,1bh
        jz done
        mov bl,al
        mov al,':'
        call dispc
        mov al,bl
        rol al,4
```

```
        call htoasc     ; 调用子程序
        call dispc      ; 显示一个字符
        mov al,bl
        call htoasc     ; 调用子程序
        call dispc      ; 显示一个字符
        call dispcrlf
        jmp again
done:
```

〔习题4.22〕

编写一个子程序，它以二进制形式显示EAX中32位数据，并设计一个主程序验证。

〔解答〕

```
        ; 代码段，主程序
        mov eax,8F98FF00H
        call dispbd     ; 调用子程序
        ; 代码段，子程序
dispbd  proc    ; 32位二进制数的输出
        push ecx
        push edx
        mov ecx,32      ; 要输出的字符个数
dbd:    rol eax,1       ; AL循环左移一位
        push eax
        and al,01h      ; 取AL最低位
        add al,30h      ; 转化成相应的ASCLL码值
        call dispc      ; 以二进制的形式显示
        pop eax
        loop dbd
        pop edx
        pop ecx
        ret
dispbd  endp
```

〔习题4.23〕

将例题4-16的32位寄存器改用16位寄存器，仅实现输出－215～＋215－1之间的数据。

〔解答〕

```
        ; 数据段
array   word 12345,-1234,32767,-32768,0,667
writebuf byte 6 dup(0)
        ; 代码段，主程序
        mov ecx,lengthof array
        mov ebx,0
again:
       mov ax,array[ebx*2]
       call write
       call dispcrlf
       inc ebx              ;此时ebx代表array中的第几个数
       dec ecx              ;此时ecx代表循环的次数
       jnz again
        ; 代码段，子程序
write  proc                   ;子程序开始
       push ebx
       push ecx
       push edx

       mov ebx,offset writebuf   ;ebx指向显示缓冲区
       test ax,ax
       jnz write1
       mov byte ptr [ebx],30h
```

```
        inc ebx
        jmp write5
write1:                         ;若不为0则首先判断是正是负
        jns write2              ;若为正则跳过下面两步到write2
        mov byte ptr [ebx],'-'
        inc ebx
        neg ax
write2:
        mov cx,10
        push cx                 ;将cx=10压入栈，作为退出标志
write3:                         ;write3是让eax循环除以10并把余数的ASCII码压入栈
        cmp ax,0
        jz write4
        xor dx,dx
        div cx
        add dx,30h
        push dx
        jmp write3
write4:                         ;余数的ASCII码出栈，遇到10终止并转到write5显示结果

        pop dx
        cmp dx,cx
        jz write5
        mov byte ptr [ebx],dl
        inc ebx
        jmp write4
write5:                             ;显示结果
        mov byte ptr [ebx],0
        mov eax,offset writebuf
        call dispmsg
        pop edx
        pop ecx
        pop ebx
        ret
write endp
```

〔习题4.24〕

参考例题4-17，编写实现32位无符号整数输入的子程序，并设计一个主程序验证。

〔解答〕

```
        ; 数据段
count   =10
array   dword count dup(0)
temp    dword ?
readbuf byte 30 dup(0)
errmsg  byte 'Input error,enter again!',13,10,0
msg1    byte 'Input ten unsigned numbers,each number ends with enter
key:',13,10,0
msg2    byte 'Check the numbers your inputted:',13,10,0
        ; 代码段，主程序
        mov eax,offset msg1
        call dispmsg
        mov ecx,count
        mov ebx,offset array
again:
        call read
        mov eax,temp
        mov [ebx],eax
```

```
        add ebx,4
        dec ecx
        jnz again

        mov eax,offset msg2
        call dispmsg
        mov edx,offset array
        mov ecx,count
next:
        mov eax,[edx]
        call dispuid
        call dispcrlf
        add edx,4
        dec ecx
        jnz next
        ; 代码段，子程序
read    proc
        push eax
        push ecx
        push ebx
        push edx
read0:
        mov eax,offset readbuf
        call readmsg
        test eax,eax
        jz readerr
        cmp eax,12
        ja readerr
        mov edx,offset readbuf
        xor ebx,ebx
        xor ecx,ecx
        mov al,[edx]
        cmp al,'+'
        jz read1
        cmp al,'-'
        jnz read2
        jmp readerr
read1:
        inc edx
        mov al,[edx]
        test al,al
        jz read3          ;如果为0，则说明该字符串已结束
read2:
        cmp al,'0'
        jb readerr
        cmp al,'9'
        ja readerr
        sub al,30h
        imul ebx,10      ;ebx用来存储处理过的数据
        jc readerr
        movzx eax,al
        add ebx,eax
        jnc read1
readerr:
        mov eax,offset errmsg
        call dispmsg
```

```
        jmp read0
read3:
        mov temp,ebx
        pop edx
        pop ebx
        pop ecx
        pop eax
        ret
read    endp
```

〔习题4.25〕

编写一个计算字节校验和的子程序。所谓“校验和”是指不记进位的累加，常用于检查信息的正确性。主程序提供入口参数，有数据个数和数据缓冲区的首地址。子程序回送求和结果这个出口参数。

〔解答〕

```
        ; 计算字节校验和的通用过程
        ; 入口参数：DS:EBX＝数组的段地址:偏移地址，ECX=元素个数
        ; 出口参数：AL＝校验和
        ; 说明：除EAX/EBX/ECX外，不影响其他寄存器
checksum        proc
        xor al,al       ; 累加器清0
sum:    add al,[ebx]    ; 求和
        inc ebx ; 指向下一个字节
        loop sum
        ret
checksum        endp
```

〔习题4.26〕

编制3个子程序把一个32位二进制数用8位十六进制形式在屏幕上显示出来，分别运用如下3种参数传递方法，并配合3个主程序验证它。

（1）采用EAX寄存器传递这个32位二进制数

（2）采用temp变量传递这个32位二进制数

（3）采用堆栈方法传递这个32位二进制数

〔解答〕

（1）

```
        ; 数据段
wvar    word 307281AFH
        ; 代码段，主程序
        mov eax,wvar
        call disp
        mov al,'H'
        call dispc
        ; 代码段，子程序
disp    proc
        push ebx
        push ecx
        mov ecx,8       ; 8位
dhw1:   rol eax,4
        mov ebx,eax
        and al,0fh      ; 转换为ASCII码
        add al,30h
        cmp al,'9'
        jbe dhw2
        add al,7
dhw2:   call dispc
        mov eax,ebx
        loop dhw1
        pop ecx
```

```
        pop ebx
        ret
disp    endp
```

（2）

```
        ; 数据段
wvar    word 307281AFH
temp    word ?
        ; 代码段，主程序
        mov eax,wvar
        mov temp,eax
        call disp
        mov al,'H'
        call dispc
        ; 代码段，子程序
disp    proc
        push ebx
        push ecx
        mov ecx,8       ; 8位
        mov eax,temp
dhw1:   rol eax,4
        mov ebx,eax
        and al,0fh      ; 转换为ASCII码
        add al,30h
        cmp al,'9'
        jbe dhw2
        add al,7
dhw2:   call dispc        ; 显示一个字符
        mov eax,ebx
        loop dhw1
        pop ecx
        pop ebx
        ret
disp    endp
```

（3）

```
        ; 数据段
wvar    word 307281AFH
        ; 代码段，主程序
        push wvar
        call disp
        add esp,4
        mov al,'H'
        call dispc
        ; 代码段，子程序
disp    proc
        push ebp
        mov ebp,esp
        push ebx
        push ecx
        mov ecx,8       ; 8位
        mov eax,[ebp+8]
dhw1:   rol eax,4
        mov ebx,eax
        and al,0fh      ; 转换为ASCII码
        add al,30h
        cmp al,'9'
```

```
        jbe dhw2
        add al,7
dhw2:   call dispc
        mov eax,ebx
        loop dhw1
        pop ecx
        pop ebx
        pop ebp
        ret
disp    endp
```

〔习题4.27〕
配合例题4-11的简单加密解密程序，设计一个输入密码的程序，将输入的若干字符经过适当算法得到一个字节量密码。
〔解答〕

```
;ex0427.asm
        include io32.inc
        .data
key     byte ?
msg0    byte 'Enter your password:',0
passw   byte 50 dup(0)
errormsg        byte 'Password error, input again!',13,10,0
bufnum  = 255
buffer  byte bufnum+1 dup(0)    ; 定义键盘输入需要的缓冲区
msg1    byte 'Enter messge: ',0
msg2    byte 'Encrypted message: ',0
msg3    byte 'Original messge: ',0
        .code
start:
        mov eax,offset msg0     ; 提示输入加密密码
        call dispmsg
        mov eax,offset passw    ; 设置入口参数EAX
        call readmsg    ; 调用输入字符串子程序输入密码
        mov ecx,eax
        dec ecx
        xor ebx,ebx
        mov al,passw[ebx]
again0: inc ebx
        xor al,passw[ebx]       ;使用简单的异或方法得到加密关键字
        loop again0
        mov key,al      ; 保存加密关键字

        mov eax,offset msg1     ; 提示输入字符串
        call dispmsg
        mov eax,offset buffer   ; 设置入口参数EAX
        call readmsg    ; 调用输入字符串子程序
        push eax        ; 字符个数保存进入堆栈
        mov ecx,eax     ; ECX＝实际输入的字符个数，作为循环的次数
        xor ebx,ebx     ; EBX指向输入字符
encrypt:        mov al,key      ; AL＝加密关键字
        xor buffer[ebx],al      ; 异或加密
        inc ebx
        dec ecx ; 等同于指令：loop encrypt
        jnz encrypt     ; 处理下一个字符
        mov eax,offset msg2
        call dispmsg
        mov eax,offset buffer   ; 显示加密后的密文
```

```
        call dispmsg
        call dispcrlf
        ;
again:  mov eax,offset msg0     ; 提示输入解密密码
        call dispmsg
        mov eax,offset passw    ; 设置入口参数EAX
        call readmsg    ; 调用输入字符串子程序输入密码
        mov ecx,eax
        dec ecx
        xor ebx,ebx
        mov al,passw[ebx]
again1: inc ebx
        xor al,passw[ebx]       ;使用简单的异或方法得到加密关键字
        loop again1
        cmp key,al      ; 与原加密关键字比较
        jz next ; 密码相同，则进行解密
        mov eax,offset errormsg ; 提示输入解密密码错误
        call dispmsg
        jmp again

next:   pop ecx ; 从堆栈弹出字符个数，作为循环的次数
        xor ebx,ebx     ; EBX指向输入字符
decrypt:        mov al,key      ; AL＝解密关键字
        xor buffer[ebx],al      ; 异或解密
        inc ebx
        dec ecx
        jnz decrypt     ; 处理下一个字符
        mov eax,offset msg3
        call dispmsg
        mov eax,offset buffer   ; 显示解密后的明文
        call dispmsg
```

〔习题4.28〕
设计一个简单的两个整数的加法器程序。
〔解答〕

```
;ex0428.asm
        include io32.inc
        .data
msg1    byte 'Enter the integers:',13,10,0
msg2    byte 13,10,'Enter space to continue! Enter any other key to
exit!',13,10,0
        .code
start:
        mov eax,offset msg1
        call dispmsg
        call readsid

        mov ebx,eax
        mov al,'+'
        call dispc
        call dispcrlf

        call readsid
        add ebx,eax
        mov al,'='
        call dispc
        mov eax,ebx
```

```
        call dispsid
        call dispcrlf

        mov eax,offset msg2
        call dispmsg
        call readc
        cmp al,20h
        je start

        exit 0
        end start
```

〔习题4.29〕
利用十六进制字节显示子程序DISPHB设计一个从低地址到高地址逐个字节显示某个主存区域内容的子程序DISPMEM。其入口参数：EAX＝主存偏移地址，ECX＝字节个数（主存区域的长度）。同时编写一个主程序进行验证。
〔解答〕

```
;ex0429.asm in Windows Console
        include io32.inc
        .data
var     byte 'This is a test!'
        .code
start:  ; 主程序
        mov eax,offset var
        mov ecx,sizeof var
        call dispmem
        exit 0
        ; 子程序
dispmem proc
        push ebx
        mov ebx,eax
dispm1: mov al,[ebx]
        call disphb
        mov al,' '
        call dispc
        inc ebx
        loop dispm1
        pop ebx
        ret
dispmem endp
        end start
```

〔习题4.30〕
将例题4-19分别使用子程序模块、子程序库和子程序库包含方法生成最终可执行文件。
〔习题4. 31〕
区别如下概念：宏定义、宏调用、宏指令、宏展开、宏汇编。
〔解答〕
宏定义：就是对宏进行说明，由一对宏汇编伪指令MACRO和ENDM来完成。
宏调用：宏定义之后的使用。在使用宏指令的位置写下宏名，后跟实体参数。
宏指令：使用宏时，其形式很像指令，所以称为宏指令。
宏展开：在汇编时，汇编程序用对应的代码序列替代宏指令。
宏汇编：指使用宏的方法进行汇编语言程序设计。
〔习题4. 32〕
直接使用控制台输入和输出函数实现例题4-21的功能（不使用READMSG和DISPMSG子程序）。
〔解答〕

〔习题4. 33〕
直接使用控制台输出函数实现某个主存区域内容的显示（习题4.29的功能）。可以改进显示

形式，例如每行显示16个字节（128位），每行开始先显示首个主存单元的偏移地址，然后用冒号分隔主存内容。
〔解答〕

〔习题4. 34〕
如何进行很简单的修改，使得例题4-22程序的消息窗有“OK”和“Cancel”两个按钮。
〔解答〕
将MB_OK常量定义为1，即：
```
MB_OK    equ 1
```
〔习题4. 35〕
上机实践例题4-23和例题4-24，并在创建可执行文件的过程中生成汇编语言列表文件。
〔习题4.36〕
Pentium处理器含有一个64位的时间标记计数器（Time-Stamp Counter）。该计数器每个时钟周期递增（加1）；在上电和复位后，该计数器清0。指令“RDTSC”执行后将在EDX（高32位）和EAX（低32位）返回当前的64位时间标记计数器值。利用RDTSC指令在某个函数运行前获得时间标记计数器值，然后运行该函数后，立即再次执行RDTSC指令，并将再次获得的时间标记计数器值与之前的计数值相减，得到的差值就是运行该函数需要的时钟周期数（乘以时钟周期，等于运行时间）。请利用混合编程方法显示某个函数的运行时钟周期数。
〔解答〕

第5章 微机总线
〔习题5.1〕简答题
〔解答〕
① 数据总线承担着处理器与存储器、外设之间的数据交换，既可以输入也可以输出，故其是双向的。
② 为减少引脚个数，8086采用了地址总线和数据总线分时复用。即数据总线在不同时刻还具有地址总线的功能。
③ 具有三态能力的引脚当输出呈现高阻状态时，相当于连接了一个阻抗很高的外部器件，信号无法正常输出；即放弃对该引脚的控制，与其他部件断开连接。
④ 处理器的运行速度远远快于存储器和I/O端口。处理器检测到存储器或I/O端口不能按基本的总线周期进行数据交换时，插入一个等待状态Tw。等待状态实际上是一个保持总线信号状态不变的时钟周期。
⑤ 猝发传送是处理器只提供首地址、但可以从后续连续的存储单元中读写多个数据。
⑥ 总线上可能连接多个需要控制总线的主设备，需要确定当前需要控制总线的主设备，所以需要总线仲裁。
⑦ 异步时序是由总线握手（Handshake）联络（应答）信号控制，不是由总线时钟控制。故总线时钟信号可有可无。
⑧ 单总线结构限制了许多需要高速传输速度的部件。32位PC机采用多种总线并存的系统结构。各种专用局部总线源于处理器芯片总线，以接近处理器芯片引脚的速度传输数据，它为高速外设提供速度快、性能高的共用通道。
⑨ 4个。
⑩ 即插即用技术是指32位PC机的主板、操作系统和总线设备配合，实现自动配置功能。
〔习题5.2〕判断题
〔解答〕
① 对　② 对　③ 错　④ 对　⑤ 对
⑥ 错　⑦ 对　⑧ 对　⑨ 对　⑩ 错
〔习题5.3〕填空题
〔解答〕
① 0
② 读，写
③ 存储器读，存储器读，存储器写
④ 4，2，10ns
⑤ 低有效，，0010
⑥ 寻址，数据传送

⑦ 127，1.5Mb/s，12Mb/s
⑧ 地址，数据
⑨ ，
⑩ ，I/O CH RDY

〔习题5.4〕
处理器有哪4种最基本的总线操作（周期）？
〔解答〕
存储器读、存储器写，I/O读、I/O写。
〔习题5.5〕
8086处理器的输入控制信号有RESET，HOLD，NMI和INTR，其含义各是什么？当它们有效时，8086 CPU将出现何种反应？
〔解答〕
RESET：复位输入信号，高电平有效。该引脚有效时，将迫使处理器回到其初始状态；转为无效时，CPU重新开始工作。
HOLD：总线请求，是一个高电平有效的输入信号。该引脚有效时，表示其他总线主控设备向处理器申请使用原来由处理器控制的总线。
NMI：不可屏蔽中断请求，是一个利用上升沿有效的输入信号。该引脚信号有效时，表示外界向CPU申请不可屏蔽中断。
INTR：可屏蔽中断请求，是一个高电平有效的输入信号。该引脚信号有效时，表示中断请求设备向处理器申请可屏蔽中断。
〔习题5.6〕
区别概念：指令周期、总线周期（机器周期）、时钟周期、T状态。
〔解答〕
指令周期：一条指令从取指、译码到最终执行完成的过程。
总线周期（机器周期）：有数据交换的总线操作。
时钟周期：处理器的基本工作节拍，由时钟信号产生，一个高电平和一个低电平为一个周期。
T状态：完成特定操作的一个时钟周期。由于时间上一个T状态等于一个时钟周期，所以常常将两者混为一谈。
〔习题5.7〕
总结8086各个T状态的主要功能。
〔解答〕
T1状态：总线周期的第一个时钟周期主要用于输出存储器地址或I/O地址；
T2状态：输出读/写控制信号。
T3状态：锁存地址、处理器提供的控制信号和数据在总线上继续维持有效，且T3时钟的前沿（下降沿）对READY引脚进行检测。READY信号有效，进入T4周期。
T4状态：总线周期的最后一个时钟周期，处理器和存储器或I/O端口继续进行数据传送，直到完成，并为下一个总线周期做好准备。
Tw状态：等待状态。处理器在T3前沿发现READY信号无效后，插入Tw。Tw状态的引脚信号延续T3时的状态、维持不变。
〔习题5.8〕
请解释8086（最小组态）以下引脚信号的含义：CLK，A19/S6～A16/S3，AD15～AD0，ALE，，和。默画它们在具有一个等待状态的存储器读总线周期中的波形示意。
〔解答〕
CLK：时钟输入。时钟信号是一个频率稳定的数字信号，其频率就是处理器的工作频率，工作频率的倒数就是时钟周期的时间长度。
A19/S6～A16/S3：地址/状态分时复用引脚，是一组4个具有三态能力的输出信号。这些引脚在访问存储器的第一个时钟周期输出高4位地址A19～A16，在访问外设的第一个时钟周期输出低电平无效；其他时间输出状态信号S6～S3。
AD15～AD0：地址/数据分时复用引脚，共16个引脚，用作地址总线时是单向输出信号；用作数据总线时是双向信号，具有三态输出能力。
ALE：地址锁存允许，是一个三态、输出、高电平有效的信号。有效时，表示复用引脚（AD15～AD0和A19/S6～A16/S3）上正在传送地址信号。
：访问存储器或者I/O，是一个三态输出信号，该引脚高电平时，表示处理器将访问存储器，此时地址总线A19～A0提供20位的存储器物理地址。该引脚低电平时，表示处理器将访问I/O端口，此时地址总线A15～A0提供16位的I/O地址。

：读控制，也是一个三态、输出低电平有效信号。有效时，表示处理器正在从存储单元或I/O端口读取数据。
：写控制，是一个三态、输出低电平有效信号。有效时，表示处理器正将数据写到存储单元或I/O端口。
〔习题5.9〕
区别如下总线概念：芯片总线、局部总线、系统总线；并行总线、串行总线；地址总线、数据总线、控制总线；ISA总线、PCI总线。
〔解答〕
芯片总线：是指大规模集成电路芯片内部，或系统中各种不同器件连接在一起的总线；用于芯片级互连。
局部总线：位于处理器附件的器件相互连接的总线，相对于芯片总线。
系统总线：通常是指微机系统的主要总线。
并行总线：采用并行传输方式的总线。
串行总线：将多位数据按二进制位的顺序在数据线上逐位传送的总线。
地址总线：实现地址信息互连和交换的一组导线。
数据总线：实现数据信息互连和交换的一组导线。
控制总线：控制协调处理器和内存、外设交互信息的一组导线。
ISA总线：即IBM PC/AT总线，以处理器80286引脚形成的总线，分成支持8位操作的前62信号和扩展16位操作的后36信号。
PCI总线：外设部件互连总线，不仅适用于IA-32处理器，也适用其它处理器，支持32位和64位操作，广泛用于32位通用微型计算机中。
〔习题5.10〕
什么是同步时序、半同步时序和异步时序？
〔解答〕
同步时序：总线操作的各个过程由共用的总线时钟信号控制。
半同步时序：总线操作仍由共用的总线时钟信号控制，但慢速模块可以通过等待信号让快速模块等待。
异步时序：总线操作需要握手（Handshake）联络（应答）信号控制，总线时钟信号可有可无。
〔习题5.11〕
EISA总线的时钟频率是8MHz，每2个时钟可以传送一个32位数据，计算其总线带宽。
〔解答〕
（32×8）÷（2×8）＝16MBps
〔习题5.12〕
PCI总线有什么特点？
〔解答〕
PCI总线与处理器无关，具有32位和64位数据总线，有＋5V和＋3.3V两种设计，采用集中式总线仲裁、支持多处理器系统，通过桥（Bridge）电路兼容ISA/EISA总线，具有即插即用的自动配置能力等一系列优势。
〔习题5.13〕
PCI总线操作如何插入等待状态？
〔解答〕
主设备利用IRDY#信号无效、从设备利用TRDY#信号无效要求对方等待，即插入等待状态。
〔习题5.14〕
什么是USB总线支持的“热插拔”，这个特性有什么意义？
〔解答〕
“热插拔”是在PC机正常工作状态进行插入或拔出。这个特性可以使用户随时连接USB设备。
〔习题5.15〕
简述USB总线的主要特征？
〔解答〕
使用方便、扩充能力强。
支持多种传输速度、适用面广。
低功耗、低成本、占用系统资源少。
〔习题5.16〕
USB总线的集线器有什么作用？主机上是否需要集线器？

〔解答〕
集线器是专门用于提供额外USB接入点的USB设备。
主机需要集线器，被称为根集线器。
〔习题5.17〕
USB总线协议支持哪几种数据传输方式？简述之。
〔解答〕
USB的数据传输有4种：
控制传输——在USB设备初次安装时，USB系统软件使用控制传输方式设置USB设备参数、发送控制指令、查询状态等。
批量传输——对于打印机、扫描仪等设备需要传输大量数据，可以使用批量传输方式连续传输一批数据。
中断传输——该方式传输的数据量很小，但需要及时处理，以保证实时性，主要用于键盘、鼠标等设备上。
同步传输——该方式以稳定的速率发送和接收信息，保证数据的连续和及时，用于数据传输正确性要求不高而对实时性要求高的外设，例如麦克风、喇叭、电话等。
第6章 存储系统
〔习题6.1〕简答题
〔解答〕
① 因为各种存储器件在容量、速度和价格方面存在矛盾。速度快，则单位价格高；容量大，单位价格低，但存取速度慢。故存储系统不能采用一种存储器件。
② Cache中复制着主存的部分内容。当处理器试图读取主存的某个字时，Cache控制器首先检查Cache中是否已包含有这个字。若有，则处理器直接读取Cache，这种情况称为高速命中；若无，则称为高速缺失。
③ 标签存储器保存着该数据所在主存的地址信息。
④ 主存块与Cache行之间的对应关系称“地址映射”， Cache通过地址映射确定一个主存块应放到哪个Cache行组中。
⑤ 写入策略用于解决写入Cache时引起主存和Cache内容不一致性的问题。
⑥ 存取时间是指从读/写命令发出，到数据传输操作完成所经历的时间；存取周期表示两次存储器访问所允许的最小时间间隔。存取周期大于等于存取时间。
⑦ 虚拟存储器是由操作系统利用辅助存储器、以磁盘文件形式建立的、在主存储器与辅助存储器之间的一个存储器。
⑧ DRAM芯片容量大、芯片小，高集成度，引脚数量少。故DRAM芯片将地址引脚分时复用，即用一组地址引脚传送两批地址。第一批地址称行地址，第二批地址称列地址。
⑨ 译码电路中只有部分地址线参与译码会造成地址重复，也就是一个存储单元占有多个存储器地址。
⑩页表项的P位称为存在位（Present），表示该页面是否在物理存储器中。
〔习题6.2〕判断题
。
〔解答〕
① 错 ② 对 ③ 对 ④ 对 ⑤ 对
⑥ 错 ⑦ 错 ⑧ 对 ⑨ 错 ⑩ 对
〔习题6.3〕填空题
〔解答〕
① 8，1024，1024，1024，1024，240
② 8KB，4
③ 随机存取存储器，丢失，只读存储器，读取，不会丢失
④ 8，13，8
⑤ 2
⑥（UV-）EPROM，Flash Memory
⑦ 58000H，5FFFFH，32KB
⑧ 32，4，64，8
⑨ 直接映射，组合相关映射，全相关映射，2路组合相关映射
⑩ 00820000H，02000H
〔习题6.4〕
举例说明存储访问的局部性原理。
〔解答〕

处理器访问存储器时，无论是读取指令还是存取数据，所访问的存储单元在一段时间内都趋向于一个较小的连续区域中，这就是存储访问的局部性原理。
例如，求平均值的函数。

```
long mean(long d[], long num)
{
        long i,temp=0;
        for(i=0; i<num; i++) temp=temp+d[i];
        temp=temp/num;
        return (temp);
}
```

函数中的变量temp体现了时间局部，因为每次循环都要使用它。顺序访问数组d[]的各个元素（相邻存放在主存），体现了空间局部。循环体内的指令顺序存放，依次读取执行体现了空间局部；同时重复执行循环体，又体现了时间局部。
〔习题6.5〕
简述存储系统的层次结构及各层存储部件特点。
〔解答〕
为解决容量、速度和价格的矛盾，存储系统采用金字塔型层次结构，单位价格和速度自上而下逐层减少，容量自上而下逐层增加。
存储系统的各层存储部件自上而下依次是：CPU寄存器、高速缓存、主存存储器(RAM/ROM)，辅助存储器如磁盘、光盘等。CPU寄存器、高速缓存器集成在CPU芯片上，对用户来说，是透明的，它们用于暂存主存和处理器交互的数据，以减少频繁读取主存而影响处理器速度；主存储器则可和处理器直接交换数据，而辅助存储器必须经过主存存储器，才可与处理器进行数据交换。
〔习题6.6〕
在半导体存储器件中，什么是SRAM、DRAM和NVRAM?
〔解答〕
SRAM是静态读写存储器芯片，它以触发器为基本存储单元，以其两种稳定状态表示逻辑0和逻辑1。
DRAM是动态读写存储器芯片，它以单个MOS管为基本存储单元，以极间电容充放电表示两种逻辑状态，需要不断刷新保持信息正确。
NVRAM多指带有后备电池的SRAM芯片，这种芯片采用CMOS制造工艺设计以减少用电。
〔习题6.7〕
SRAM芯片的片选信号有什么用途？对应读写控制的信号是什么？
〔解答〕
片选信号：片选有效时，才可以对该芯片进行读/写操作；无效时，数据引脚呈现高阻状态、与系统数据总线隔离，并可降低内部功耗。
读控制信号：在芯片被选中的前提下，若有效，则芯片将允许地址信号选择的存储单元内的数据输出到数据引脚上。
写控制信号：在芯片被选中的前提下，若有效，则芯片将数据引脚上的数据写入地址信号选择的存储单元内。
〔习题6.8〕
DRAM为什么要刷新，存储系统如何进行刷新？
〔解答〕
DRAM以单个MOS管为基本存储单元，以极间电容充放电表示两种逻辑状态。由于极间电容的容量很小，充电电荷自然泄漏会很快导致信息丢失，所以要不断对它进行刷新操作、即读取原内容、放大再写入。
存储系统的刷新控制电路提供刷新行地址，将存储DRAM芯片中的某一行选中刷新。实际上，刷新控制电路是将刷新行地址同时送达存储系统中所有DRAM芯片，所有DRAM芯片都在同时进行一行的刷新操作。
刷新控制电路设置每次行地址增量，并在一定时间间隔内启动一次刷新操作，就能够保证所有DRAM芯片的所有存储单元得到及时刷新。
〔习题6.9〕
什么是掩摸ROM、OTP-ROM、EPROM、EEPROM和Flash ROM?
〔解答〕
掩膜ROM：通过掩膜工艺、将要保存的信息直接制作在芯片当中，以后再也不能更改。
OTP-ROM：该类芯片出厂时存储的信息为全“1”，允许用户进行一次性编程，此后便不能更

改。
EPROM：一般指可用紫外光擦除、并可重复编程的ROM。
EEPROM：也常表达为E2PROM，其擦除和编程（即擦写）通过加电的方法来进行，可实现“在线编程”和“在应用编程”
Flash ROM：是一种新型的电擦除可编程ROM芯片，能够很快擦除整个芯片内容。
〔习题6.10〕
请给出教材图6-7中138译码器的所有译码输出引脚对应的地址范围。
〔解答〕
～的地址范围依次是：
E0000H～E3FFFH，E4000H～E7FFFH，E8000H～EBFFFH，EC000H～EFFFFH，F0000H～F3FFFH，F4000H～F7FFFH，F8000H～FBFFFH，FC000H～FFFFFH。
〔习题6.11〕
什么是存储器芯片的全译码和部分译码？各有什么特点？
〔解答〕
全译码：使用全部系统地址总线进行译码。特点是地址唯一，一个存储单元只对应一个存储器地址（反之亦然），组成的存储系统其地址空间连续。
部分译码：只使用部分系统地址总线进行译码。其特点：有一个没有被使用的地址信号就有两种编码，这两个编码指向同一个存储单元，出现地址重复。
〔习题6.12〕
区别如下各个主存名称的含义：常规主存，扩展主存，扩充主存；上位主存区UMA和上位主存块UMB，高端主存区HMA，影子主存。
〔解答〕
常规主存：8088和8086提供20个地址线A19～A0，寻址1MB的存贮空间，其中，最低640KB的系统RAM区被称为常规主存或基本主存。
扩展主存：IA-32处理器在1MB之后的主存空间都作为RAM区域使用，被称为扩展主存。
扩充主存：处理器不可以直接访问，利用“体交换技术”实现处理器访问。
上位主存区UMA：在常规主存其后384KB（A0000H～FFFFFH）主存称为上位主存区UMA。
上位主存块UMB：上位主存区UMA没有被使用部分，被开辟为上位主存块UMB。
高端主存区HMA：在实方式下，通过控制A20开放，程序可以访问的1MB之后的64KB区域。
影子主存：PC机启动后可以将ROM-BIOS映射到RAM中，这部分用作ROM-BIOS、并被操作系统设置为只读的RAM区域。
〔习题6.13〕
开机后，微机系统常需要检测主存储器是否正常。例如，可以先向所有存储单元写入数据55H（或00H）、然后读出看是否还是55H（或00H）；接着再向所有存储单元写入数据AAH（或FFH）、然后读出看是否还是AAH（或FFH）。利用两个二进制各位互反的“花样”数据的反复写入、读出和比较就能够识别出有故障的存储单元。利用获得的有故障存储单元所在的物理地址，如果能够分析出该存储单元所在的存储器芯片，就可以实现芯片级的维修。试利用汇编语言编写一个检测常规主存最高64KB（逻辑地址从9000H：0000H到9000H：FFFFH）的程序，如果发现错误请显示其逻辑地址。
〔解答〕

```
        ; 代码段
        mov ax,9000h
        mov ds,ax
        mov ah,55h      ; 先用55H
        push ax
again:  mov bx,0
        mov al,ah
again1: mov [bx],al     ; 写入
        dec bx
        jnz again1
again2: mov al,[bx]     ; 读出
        cmp al,ah       ; 检测
        jz next2
        dispcrlf
        push ax
        mov ax,ds
```

```
        call disphw     ; 显示段地址
        mov al,':'
        call dispc
        mov ax,bx
        call disphw     ; 显示偏移地址
        pop ax
next2:  dec bx
        jnz again2
        pop ax
        cmp ah,0aah     ; 后用0AAH
        jz done
        mov ah,0aah
        jmp again
done:
```

〔习题6.14〕
什么是LRU替换算法？80486片内Cache中，如果3个替换算法位B2B1B0＝010，则将替换哪个Cache行，并给出你的判断过程。
〔解答〕
LRU算法是近期最少使用、即选择最长时间未被使用的数据块进行替换的算法。
B0＝0，说明最近访问了L2/L3行，所以应该替换L0或L1行。B1＝1，说明最近访问了L0行，所以应该替换L1。因为LRU算法是选择最长时间未被访问的Cache行进行替换。
〔习题6.15〕
高速缓冲存储器Cache的写入策略是解决什么问题的？有哪两种写入策略，各自的写入策略是怎样的？
〔解答〕
写入策略用于在写命中时Cache与主存内容保持一致。
直写式写入策略指处理器对Cache写入的同时，将数据也写入到主存，这样来保证主存和Cache内容一致。它简单可靠。
回写Cache只有在行替换时才可能写入主存，写入主存的次数，会少于处理器实际执行的写入操作数。回写Cache的性能要高于直写Cache，但实现结构略为复杂。
〔习题6.16〕
80486片上8KB Cache的标签存储器为什么只需要21位？
〔解答〕
80486片上Cache共有8KB容量，采用4路组合地址映射方式。对于4GB容量的主存来说，以Cache路为单位，可以分成4GB÷2KB＝232÷211＝221个Cache路。这样每个Cache行只要设计一个21位的标签存储器，记录该Cache行映射到哪个主存的Cache路。再结合直接映射的组号就可以明确该Cache行对应哪个主存块。
〔习题6.17〕
高速缓存的写入操作有几个很近似的英文词汇，它们分别表示什么含义？
（1）Write Through　　（2）Write Back
（3）Write Around　　（4）Fetch on Write
〔解答〕
（1）Write Through：写命中时的直写策略。
（2）Write Back：写命中时的回写策略。
（3）Write Around：写未命中时的不写分配法，即绕写法。
（4）Fetch on Write：写未命中时的写分配法，即写时取法。
〔习题6.18〕
区别如下高速缓存中的概念：
（1）主存数据块Block　　（2）高速缓存行Line
（3）高速缓存组Set　　（4）高速缓存路Way
〔解答〕
（1）主存数据块Block：高速缓存与主存间的数据传送以数据块（Block）为单位，例如B个字。主存数据块Block是主存中连续的B个字数据。
（2）高速缓存行Line：指高速缓存中包含B个字的一个单元。
（3）高速缓存组Set：组合相关映射将多个Cache行作为一个组（Set）。
（4）高速缓存路Way：组合相关映射将所有组中同位置Cache行称为一路（Way）。

〔习题6.19〕
什么是段选择器、描述符、描述符表和描述符表寄存器？
〔解答〕
段选择器：保护方式下的16位段寄存器就是段选择器。
描述符：是保护方式引入的数据结构，有8个字节64位，具有段基地址、访问权限、段界限等字段。IA-32处理器利用它来实现存储管理、特权与保护。
描述符表：描述符表是存放描述符的一个特殊区域段。
描述符表寄存器：指明描述符表所在主存地址的寄存器。
〔习题6.20〕
IA-32处理器在保护方式下，段寄存器是什么内容？若DS＝78H，说明在保护方式其具体的含义。
〔解答〕
段寄存器是段选择器，包含3个域，指向一个段描述符。
DS＝78H，说明当前数据段描述符是全局描述符表中的第0FH个描述符。本次访问数据的特权级别为0，最高。
〔习题6.21〕
采用4KB分页，说明IA-32处理器将线性地址转换为物理地址的过程。
〔解答〕
通过2级查表来实现线性地址转换为位物理地址。
（1）在CR3中包含着当前任务的页目录的起始地址，将其加上线性地址最高10位A31～A22确定的页目录项的偏移量，便访问到指定的页目录项。
（2）在此页目录项中包含着指向的页表的起始地址，将其加上线性地址中间的10位A21～A12确定的页表项的偏移量，便访问到指定的页表项。
（3）在此页表项中包含着要访问的页面的起始地址，将其加上线性地址最低12位A11～A0的偏移量，就从这一页中访问到所寻址的物理单元。
第7章 输入输出接口
〔习题7.1〕简答题
〔解答〕
① 外部设备，在工作原理、驱动方式、信息格式、以及工作速度等方面彼此差别很大，与处理器的工作方式也大相径庭。所以，外设不能像存储器芯片那样直接与处理器相连，必须经过一个中间电路。
② 数据缓冲用于匹配快速的处理器与相对慢速的外设或两个功能部件速度不匹配的数据交换。
③ 处理器向接口芯片相应端口写入特定的数据，用于选择I/O芯片的工作方式或控制外设工作，该数据称命令字或控制字。
④ PC机中CMOS RAM不属于主存空间，CMOS RAM有64个字节容量，以8位I/O接口形式与处理器连接，通过两个I/O地址访问。
⑤ 在输入接口中，为避免多个设备同时向总线发送数据，需要安排一个三态缓冲器。只有当处理器选通时，才允许被选中设备将数据送到系统总线，此时其他输入设备与数据总线隔离。
⑥ 透明锁存器的控制端为有效电平时，输出随输入变化，常称为直通或透明。非透明锁存器不论其控制端为低或为高电平，输出状态都不随输入变化。
⑦ 如发光二极管、按键和开关等简单设备，它们的工作方式十分简单；相对处理器而言，其状态很少发生变化或变化很慢。这些设备与处理器交换数据时，可采用无条件传送。
⑧ 在查询程序中，当查询超过了规定的时间，设备仍未就绪时，就引发超时错误。
⑨ 远调用CALL指令利用直接或间接寻址调用另一个代码段的子程序；INT n指令利用中断向量表（地址表）的方法调用另一个代码段的中断服务程序，还有保存标志寄存器的功能。
⑩ 外部中断是由处理器外部提出中断请求引起的程序中断。相对于处理器来说，外部中断是随机产生的，所以是真正意义上的中断。
〔习题7.2〕判断题
〔解答〕
① 对 ② 对 ③ 对 ④ 错 ⑤ 错
⑥ 错 ⑦ 对 ⑧ 错 ⑨ 对 ⑩ 错
〔习题7.3〕填空题
〔解答〕
① 数字量、开关量、脉冲量

② I/O独立，输入输出（I/O）指令，直接寻址，DX寄存器间接寻址
③ I/O端口（接口，外设），处理器（主机），I/O读
④ 寄存器，I/O地址的直接寻址
⑤ I/O地址的间接寻址，寄存器
⑥ 直接存储器存取，DMA请求，总线请求，总线响应，高阻，DMAC（DMA控制器）
⑦ 除法错，2
⑧ 1，STI，CLI，0
⑨ 1KB，20H，4，F010H：2300H
⑩ IR3，IR3请求的

〔习题7.4〕
一般的I/O接口电路安排有哪三类寄存器？它们各自的作用是什么？
〔解答〕
① 数据寄存器
保存处理器与外设之间交换的数据。
② 状态寄存器
保存外设当前的工作状态信息。处理器通过该寄存器掌握外设状态，进行数据交换。
③ 控制寄存器
保存处理器控制接口电路和外设操作的有关信息。处理器向控制寄存器写入控制信息，选择接口电路的不同工作方式和与外设交换数据形式。

〔习题7.5〕
什么是I/O独立编址和统一编址，各有什么特点？
〔解答〕
独立编址是将I/O端口单独编排地址，独立于存储器地址。
统一编址是将I/O端口与存储器地址统一编排，共享一个地址空间。
端口独立编址方式，处理器除要具有存储器访问的指令和引脚外，还需要设计I/O访问的I/O指令和I/O引脚，其优点是：不占用存储器空间；I/O指令使程序中I/O操作一目了然；较小的I/O地址空间使地址译码简单。但I/O指令功能简单，寻址方式没有存储器指令丰富。
统一编址方式，处理器不再区分I/O口访问和存储器访问。其优点是：处理器不用设计I/O指令和引脚，丰富的存储器访问方法同样能够运用于I/O访问。缺点是：I/O端口会占用存储器的部分地址空间，通过指令不易辨认I/O操作。

〔习题7.6〕
简述主机与外设进行数据交换的几种常用方式。
〔解答〕
主机与外设进行数据交换的几种常用方式：
① 无条件传送方式，常用于简单设备，处理器认为它们总是处于就绪状态，随时进行数据传送。
② 程序查询方式：处理器首先查询外设工作状态，在外设就绪时进行数据传送。
③ 中断方式：外设在准备就绪的条件下通过请求引脚信号，主动向处理器提出交换数据的请求。处理器无其他更紧迫任务，则执行中断服务程序完成一次数据传送。
④ DMA传送： DMA控制器可接管总线，作为总线的主控设备，通过系统总线来控制存储器和外设直接进行数据交换。此种方式适用于需要大量数据高速传送的场合。

〔习题7.7〕
参看图7-5，编程实现以下功能：当K0键单独按下时，发光二极管L0～L7将依次点亮（L0，L1，L2，……L7），每个维持200ms；当K1键单独按下时，发光二极管L0～L7将反向依次点亮（L7，L6，L5，……L0），每个也维持200ms；在其他情况下各发光二极管均不点亮。假定有延时200ms的子程序DELAY可直接调用。
〔解答〕

```
again:  mov dx,8000h
        in al,dx
        cmp al,0feh      ; D7～D0＝11111110B ?
        jz next1         ; 单独按下K0，转移到next1
        cmp al,0fdh      ; D7～D0＝11111101B ?
        jz next2         ; 单独按下K1，转移到next2
        jmp again        ; 其它情况不点亮
next1:  mov cx,8
        mov al,1         ; 从K0开始
```

```
next11: out dx,al        ; 某个LED电亮
        call delay       ; 延时200ms
        shl al,1         ; rol al,1
        loop next11
        jmp again
next2:  mov cx,8
        mov al,80h       ; 从K7开始
next21: out dx,al        ; 某个LED电亮
        call delay       ; 延时200ms
        shr al,1         ; ror al,1
        loop next21
        jmp again
```

〔习题7.8〕
现有一个输入设备，其数据端口地址为FFE0H，状态端口地址为FFE2H。当状态标志D0＝1时，表明一个字节的输入数据就绪。请编写利用查询方式进行数据传送的程序段，要求从该设备读取100个字节保存到BUFFER缓冲区。
〔解答〕

```
        mov bx, offset buffer
        mov cx,100
again:  mov dx,0ffe2h
status: in al,dx         ; 查询一次
        test al,01h
        jz status
        mov dx,0ffe0h
        in al,dx         ; 输入一个字节
        mov [bx],al
        inc bx
        loop again       ; 循环，输入100个字节
```

〔习题7.9〕
某个字符输出设备，其数据端口和状态端口的地址均为80H。在读取状态时，当标志位D7＝0时，表明该设备闲，可以接收一个字符。请编写利用查询方式进行数据传送的程序段，要求将存放于缓冲区ADDR处的一串字符（以0为结束标志）输出给该设备。
〔解答〕

```
        mov bx,offset addr
again:  cmp byte ptr [bx],0
        jz done
status: in al,80h        ; 查询
        test al,80h
        jnz status
        mov al,[bx]
        out 80h,al       ; 输出一个字节
        inc bx
        jmp again        ; 循环
done:
```

〔习题7.10〕
以可屏蔽中断为例，说明一次完整的中断过程主要包括哪些环节？
〔解答〕
中断请求：外设通过硬件信号的形式、向处理器引脚发送有效请求信号。
中断响应：在满足一定条件时，处理器进入中断响应总线周期。
关中断：处理器在响应中断后会自动关闭中断。
断点保护：处理器在响应中断后将自动保护断点地址。
中断源识别：处理器识别出当前究竟是哪个中断源提出了请求，并明确与之相应的中断服务程序所在主存位置。
现场保护：对处理器执行程序有影响的工作环境（主要是寄存器）进行保护。
中断服务：处理器执行相应的中断服务程序，进行数据传送等处理工作。
恢复现场：完成中断服务后，恢复处理器原来的工作环境。

开中断：处理器允许新的可屏蔽中断。
中断返回：处理器执行中断返回指令，程序返回断点继续执行原来的程序。
〔习题7.11〕
什么是中断源？为什么要安排中断优先级？什么是中断嵌套？什么情况下程序会发生中断嵌套？
〔解答〕
计算机系统中，凡是能引起中断的事件或原因，被称为中断源。
处理器随时可能会收到多个中断源提出的中断请求，因此，为每个中断源分配一级中断优先权，根据它们的高低顺序决定响应的先后。
一个中断处理过程中又有一个中断请求、并被响应处理，被称为中断嵌套。
必须在中断服务程序中打开中断，程序才会发生中断嵌套。
〔习题7.12〕
明确如下中断有关的概念：中断源、中断请求、中断响应、关中断、开中断、中断返回、中断识别、中断优先权、中断嵌套、中断处理、中断服务。
〔解答〕
中断源：能引起中断的事件或原因。
中断请求：是外设通过硬件信号的形式、向处理器引脚发送有效请求信号。
中断响应：中断响应是在满足一定条件时，处理器进入中断响应总线周期。
关中断：禁止处理器响应可屏蔽中断。
开中断：允许处理器响应可屏蔽中断。
中断返回：处理器执行中断返回指令，将断点地址从堆栈中弹出，程序返回断点继续执行原来的程序。
中断识别：处理器识别出当前究竟是哪个中断源提出了请求，并明确与之相应的中断服务程序所在主存位置。
中断优先权：为每个中断源分配一级中断优先权，即系统设计者事先为每个中断源确定处理器响应他们的先后顺序。
中断嵌套：在一个中断处理过程中又有一个中断请求被响应处理，称为中断嵌套。
中断处理：接到中断请求信号后，随之产生的整个工作过程，称中断处理。
中断服务：指处理器执行相应的中断服务程序，进行数据传送等处理工作。
〔习题7.13〕
按照图7-10所示的中断查询接口与相应的流程图，编写用于中断服务的程序段。具体要求是，当程序查到中断设备0有中断请求（对应数据线D0），它将调用名为PROC0的子程序；如此，依次去查中断设备1～中断设备3，并分别调用名为PROC1～PROC3的子程序。
〔解答〕

```
        sti
        push ax
        push dx
        …
        mov dx,4000h
status: in al,dx
        test al,01h
        jnz service0
        test al,02h
        jnz service1
        test al,04h
        jnz service2
        test al,08h
        jnz service3
        …
service0:       call proc0
        jmp done
service1:       call proc1
        jmp done
service2:       call proc2
        jmp done
service3:       call proc3
```

```
        jmp done
        ……
done:   pop dx
        pop ax
        iret
```

〔习题7.14〕
什么是DMA读和DMA写？什么是DMA控制器8237A的单字节传送、数据块传送和请求传送？
〔解答〕
DMA读：存储器的数据在DMA控制器控制下被读出传送给外设。
DMA写：外设的数据在DMA控制器控制下被写入存储器。
单字节传送方式：每次DMA传送时仅传送一个字节。传送一个字节之后，DMA控制器释放系统总线，将控制权还给处理器。
数据块传送： DMA传送启动后就连续地传送数据，直到规定的字节数传送完。
请求传送：DMA传送由请求信号控制。如果请求信号一直有效，就连续传送数据；但当请求信号无效时，DMA传送被暂时中止。
〔习题7.15〕
IA-32处理器何时处于开中断状态、何时处于关中断状态？
〔解答〕
在IA-32处理器中，若IF＝1，则处理器处于开中断状态。
若IF＝0，则处理器处于关中断状态。IF＝0关中断的情况有：系统复位后，任何一个中断（包括外部中断和内部中断）被响应后，执行关中断指令CLI后。
〔习题7.16〕
简述IA-32处理器的中断工作过程。
〔解答〕
IA-32处理器获得向量号识别出中断源后，中断或异常接着的工作过程如下：
（1）将标志寄存器EFLAGS压入堆栈，保护各个标志位；将被中断指令的逻辑地址（代码段寄存器和指令指针寄存器内容）压入堆栈，保护断点。
（2）如果有错误代码，将其压入堆栈（有些异常产生错误代码，更具体地表明产生异常的原因）。实地址方式的异常不返回错误代码。
（3）根据向量号获得中断服务程序（中断或异常的处理程序）的段选择器和指令指针，分别传送给代码段寄存器CS和指令指针寄存器EIP。
（4）对于中断，要设置中断允许标志IF为0，即禁止进一步的可屏蔽中断。
（5）控制转移至中断服务程序入口地址（首地址），开始执行中断或异常处理程序。
中断服务程序最后是中断返回指令IRET。中断返回指令IRET将断点地址和标志寄存器出栈恢复，如果压入了错误代码还需要相应增量堆栈指针，于是控制又返回到断点指令继续执行。
〔习题7.17〕
IA-32处理器的中断向量表和中断描述符表的作用是什么？
〔解答〕
IA-32处理器的中断向量表和中断描述符表的作用都是获取中断服务程序的入口地址（称为中断向量），进而控制转移到中断服务程序中。
〔习题7.18〕
说明如下程序段的功能：

```
        cli
        mov ax,0
        mov es,ax
        mov di,80h*4
        mov ax,offset intproc    ; intproc是一个过程名
        cld
        mov es:[di],ax
        mov ax,seg intproc
        mov es:[di+2],ax
        sti
```

〔解答〕
设置80H号中断向量。
〔习题7.19〕
中断控制器8259A中IRR，IMR和ISR三个寄存器的作用是什么？

〔解答〕
中断请求寄存器IRR：保存8条外界中断请求信号IR0～IR7的请求状态。Di位为1表示IRi引脚有中断请求；为0表示该引脚无请求。
中断屏蔽寄存器IMR：保存对中断请求信号IR的屏蔽状态。Di位为1表示IRi中断被屏蔽（禁止）；为0表示允许该中断。
中断服务寄存器ISR：保存正在被8259A服务着的中断状态。Di位为1表示IRi中断正在服务中；为0表示没有被服务。
〔习题7.20〕
下面是IBM PC/XT机ROM-BIOS中的08号中断服务程序，请说明各个指令的作用。

```
int08h  proc
        sti
        push ds
        push ax
        push dx
        ……      ; 日时钟计时
        ……      ; 控制软驱马达
        int 1ch
        mov al,20h
        out 20h,al
        pop ax
        pop dx
        pop ds
        iret
int08h  endp
```

〔解答〕

```
int08h  proc far        ; 远过程
        sti     ; 开中断
        push ds ; 保护现场
        push ax
        push dx
        ……      ; 日时钟计时
        ……      ; 控制软驱马达
        int 1ch ; 调用1CH号中断
        mov al,20h      ; 发送EOI中断结束命令
        out 20h,al
        pop ax  ; 恢复现场
        pop dx
        pop ds
        iret    ; 中断返回
int08h  endp
```

〔习题7.21〕
编写一个程序，将例题7-5的INT 80H内部中断服务程序驻留内存。然后在调试程序中或其他程序中执行INT 80H，看能否实现其显示功能。
〔解答〕

```
        ; 代码段
        jmp start
        ; 80H内部中断服务程序：显示字符串（以0结尾）；DS：DX＝缓冲区首地址
new80h  proc    ; 过程定义
        sti     ; 开中断
        push ax ; 保护寄存器
        push bx
        push si
        mov si,offset intmsg
new1:   mov al,cs:[si]  ; 获取欲显示字符
        cmp al,0        ; 为“0”结束
        jz new2
```

```
        mov bx,0        ; 采用ROM-BIOS调用显示一个字符
        mov ah,0eh
        int 10h
        inc si  ; 显示下一个字符
        jmp new1
new2:   pop si  ; 恢复寄存器
        pop bx
        pop ax
        iret    ; 中断返回
intmsg  db 'A Instruction Interrupt !',0dh,0ah,0        ; 字符串（以0结尾）
new80h  endp    ; 中断服务程序结束
        ; 主程序
start:  mov ax,cs
        mov ds,ax       ; 设置04H中断向量
        mov dx,offset new80h
        cli
        mov ax,2580h
        int 21h
        sti
        mov eax,offset tsrmsg   ; 显示安装信息
        call dispmsg
        mov dx,offset start     ; 计算驻留内存程序的长度
        add dx,15
        shr dx,4        ; 调整为以“节”（16个字节）为单位
        mov ax,3100h    ; 程序驻留，返回DOS
        int 21h
tsrmsg  db  'INT 80H Program Installed ! ',0dh,0ah,0
```

〔习题7.22〕

完成例题7-2显示当前日期同样的功能，请获得日期数据后转换成ASCII码，保存在缓冲区、利用DISPMSG子程序显示。

〔解答〕

```
; ex0722.asm in DOS
        include io16.inc
        .data
date    byte 'Today is 20xx-yy-zz',0
        .code
start:
        mov ebx,11
        mov al,9        ; AL＝9（准备从9号单元获取年代数据）
        out 70h,al      ; 从70H的I/O地址输出，选择CMOS RAM的9号单元
        in al,71h       ; 从71H的I/O地址输入，获取9号单元的内容，保存在AL
        mov dl,al
        shr al,4        ; 转换高位BCD码为ASCII码
        add al,30h
        mov date[ebx],al        ; 保存到缓冲区
        add ebx,1
        and dl,0fh      ; 转换低位BCD码为ASCII码
        add dl,30h
        mov date[ebx],dl        ; 保存到缓冲
        add ebx,2

        mov al,8        ; AL＝8（从8号单元获取月份数据）
        out 70h,al
        in al,71h
        mov dl,al
        shr al,4        ; 转换高位BCD码为ASCII码
```

```
        add al,30h
        mov date[ebx],al          ; 保存到缓冲区
        add ebx,1
        and dl,0fh        ; 转换低位BCD码为ASCII码
        add dl,30h
        mov date[ebx],dl          ; 保存到缓冲
        add ebx,2

        mov al,7          ; AL＝7（从7号单元获取日期数据）
        out 70h,al
        in al,71h
        mov dl,al
        shr al,4          ; 转换高位BCD码为ASCII码
        add al,30h
        mov date[ebx],al          ; 保存到缓冲区
        add ebx,1
        and dl,0fh        ; 转换低位BCD码为ASCII码
        add dl,30h
        mov date[ebx],dl          ; 保存到缓冲

        mov eax,offset date       ; 显示
        call dispmsg
        exit 0
        end start
```

第8章 常用接口技术

〔习题8.1〕简答题

〔解答〕

① 方式1可以通过编程产生一个确定宽度的单稳脉冲，故称工作方式1为可编程单稳脉冲工作方式。

② 因为计数器是先减1，再判断是否为0，所以写入0实际代表最大计数值。

③ 通过控制字的D7位来区别：D7＝1，该控制字为方式控制字；否则为位控制字。

④ 8255的三种工作方式均可实现输出数据锁存，即数据输出后被保存在8255内部，可以读取出来，只有当8255再输出新一组数据时才改变。

⑤ Modem，称为调制解调器，将数字信号转换为适合在电话线路上传送的模拟信号（调制）以及将电话线路的模拟信号转换为数字信号（解调）。

⑥ 因绝大多数设备只使用RS-232C标准的其中9个信号，所以PC机上就配置9针连接器。

⑦ 两台微机进行短距离通信，可以不使用调制解调器，直接利用232C接口连接，被称为零调制解调器（Null Modem）连接。

⑧ UART表示通用异步接收发送器，主要功能是将并行数据转换为串行数据发送，以及实现串行数据转换为并行传送给处理器。

⑨ 采用多路开关，通过微型机控制，把多个现场信号分时地接通到A/D转换器上转换，达到共用A/D转换器以节省硬件的目的。

⑩ 处理器输出数据都只在输出指令OUT执行的极短时间内出现在数据总线上，慢速的外设不能及时获取，所以主机与DAC之间必须连接数据锁存器。

〔习题8.2〕判断题

〔解答〕

① 对　② 对　③ 对　④ 对　⑤ 对

⑥ 错　⑦ 错　⑧ 对　⑨ 对　⑩ 对

〔习题8.3〕填空题

〔解答〕

① 3，16，6，低，写入计数初值（并进入减1计数器），脉冲输入CLK，减法计数器，计数器的计数值减为0，高

② 5（＝1.5MHz÷300KHz），3

③ 24，PA0～PA7，PB0～PB7，PC0～PC7

④ 10110110（＝B6H，B7H）

⑤

⑥ 01H，1DH（＝30），81H，9DH（＝158）
⑦ TxD，RxD，GND
⑧ 通信线路控制（CLR），00011111B（1FH），2FBH
⑨ 10100000，01100000
⑩ 53H（＝51≈51.2＝2÷10×256）
〔习题8.4〕
8253芯片每个计数通道与外设接口有哪些信号线，每个信号的用途是什么？
〔解答〕
CLK时钟输入信号：在计数过程中，此引脚上每输入一个时钟信号（下降沿），计数器的计数值减1。
GATE门控输入信号：控制计数器工作，可分成电平控制和上升沿控制两种类型。
OUT计数器输出信号：当一次计数过程结束（计数值减为0），OUT引脚上将产生一个输出信号。
〔习题8.5〕
8253芯片需要几个I/O地址，各用于何种目的？
〔解答〕
4个，读写计数器0，1和2，及控制字。
〔习题8.6〕
试按如下要求分别编写8253的初始化程序，已知8253的计数器0～2和控制字I/O地址依次为204H～207H。
① 使计数器1工作在方式0，仅用8位二进制计数，计数初值为128。
② 使计数器0工作在方式1，按BCD码计数，计数值为3000。
③ 使计数器2工作在方式2，计数值为02F0H。
〔解答〕
①

```
        mov al,50h
        mov dx,207h
        out  dx,al
        mov al,128        ; ＝80h
        mov dx,205h
        out  dx,al
```

②

```
        mov al,33h
        mov dx,207h
        out  dx,al
        mov ax,3000h      ; 不是3000
        mov dx,204h
        out  dx,al
        mov al,ah
        out  dx,al
```

③

```
        mov al,0b4h
        mov dx,207h
        out  dx,al
        mov al,02f0h
        mov dx,206h
        out  dx,al
        mov al,ah
        out  dx,al
```

〔习题8.7〕
利用扬声器控制原理，编写一个简易乐器程序。
当按下1～8数字键时，分别发出连续的中音1～7和高音i（对应频率依次为524Hz，588Hz，660Hz，698Hz，784Hz，880Hz，988Hz和1048Hz）；
当按下其他键时暂停发音；
当按下ESC键（ASCII码为1BH），程序返回操作系统。
〔解答〕

```
        ; 数据段
table   dw 2277,2138,1808,1709,1522,1356,1208,1139
        ; 对应中音1～7和高音i的定时器记数值
        ; 代码段
        mov al,0b6h      ; 设置定时器2工作方式
        out 43h,al
again:  call readc       ; 等待按键
        cmp al,'1'       ; 判断是否为数字1～8
        jb next
        cmp al,'8'
        ja next
        sub al,30h       ; 1～8的ASCII码转换为二进制数
        sub al,1         ; 再减1，将数字1～8变为0～7，以便查表
        xor ah,ah
        shl ax,1         ; 乘以2
        mov bx,ax        ; 记数值表是16位数据，无法采用xlat指令
        mov ax,table[bx]         ; 取出对应的记数值
        out 42h,al       ; 设置定时器2的记数值
        mov al,ah
        out 42h,al
        in al,61h        ; 打开扬声器声音
        or al,03h        ; 使D1D0＝PB1PB0＝11B，其他位不变
        out 61h,al
        jmp again        ; 连续发声，直到按下另一个键
next:   push ax
        in al,61h        ; 不是数字1～8，则关闭扬声器声音
        and al,0fch      ; 使D1D0＝PB1PB0＝00b，其他位不变
        out 61h,al
        pop ax
        cmp al,1bh       ; 判断是否为ESC键（对应ASCII码1bh）
        jne again        ; 不是ESC，继续；否则程序执行结束
```

〔习题8.8〕
针对8255芯片工作方式1输出时序，说明数据输出的过程。
〔解答〕
① 中断方式下，处理器响应中断，执行输出OUT指令：输出数据给8255，发出信号。查询方式下，通过端口C的状态确信可以输出数据，处理器执行输出指令；
② 信号一方面清除INTR，另一方面在上升沿使有效，通知外设接收数据。实质上信号是外设的选通信号；
③ 信号结束后，数据从端口数据线上输出。当外设接收数据后，发出响应；
④ 信号使无效，上升沿又使INTR有效（允许中断的情况），发出新的中断请求。
〔习题8.9〕
设定8255芯片的端口A为方式1输入，端口B为方式1输出，则读取口C的数据的各位是什么含义？
〔解答〕
PC0：端口B的中断请求信号
PC1：端口B输出缓冲器满信号
PC2：端口B中断允许控制位
PC3：端口A的中断请求信号
PC4：端口A中断允许控制位
PC5：端口A输入缓冲器满信号
PC6/PC7：I/O信号
〔习题8.10〕
用8255端口A方式0与打印机接口示例中，如果改用端口B，其他不变，说明应该如何修改接口电路和程序。
〔解答〕
修改电路：将端口B的PB0～PB7接打印机的数据位DATA0～DATA7即可。

修改程序：将输出数据端口改为FFFAH即可。

〔习题8.11〕

用8255端口A方式1与打印机接口，如果改用端口B，其他不变，说明如何修改接口电路和程序。

〔解答〕

修改电路：PA0～PA7改为PB0～PB7；PC6改用PC2，PC7改用PC1，PC3改用PC0。

修改程序：

```
        mov dx,0fffeh
        mov al,84h
        out dx,al
        mov al,04h
        ; 使INTEB（PC2）为0，禁止中断
        out dx,al
        ……
        mov cx,counter  ; 打印字节数送CX
        mov bx,offset buffer    ; 取字符串首地址
        call prints     ; 调用打印子程序
        ……
prints  proc
        push ax ; 保护寄存器
        push dx
print1: mov al,[bx]     ; 取一个数据
        mov dx,0fffah
        out dx,al       ; 从端口B输出
        mov dx,0fffch
print2: in al,dx
        test al,02h     ; 检测（PC1）为1否?
        jz print2
        inc bx
        loop print1
        pop dx
        pop ax
        ret
prints  endp
```

〔习题8.12〕

有一工业控制系统，有四个控制点，分别由四个对应的输入端控制，现用8255的端口C实现该系统的控制，如本题图形。开关K0～K3打开则对应发光二极管L0～L3亮，表示系统该控制点运行正常；开关闭合则对应发光二极管不亮，说明该控制点出现故障。编写8255的初始化程序和这段控制程序。

〔解答〕

```
        ; 写入方式字
        mov al,100×00×1b        ; =81H（×表示任意，可以填写为0，也可以为1）
        mov dx,控制口地址       ; 可以假设为0FFFEH
        out dx,al
        ;加入下一段更好，使L0～L3全亮
        mov al,0fh
        mov dx,端口C地址        ; 可以假设为0FFFCH
        out dx,al
        ;控制程序段
        mov dx,端口C地址        ; 可以假设为0FFFCH
        in al,dx        ; 读入PC0～PC3
        mov cl,4
        shl al,cl       ; 左移4位
        out dx,al       ; 控制PC4～PC7
```

〔习题8.13〕

编写一个程序，每当在键盘上按下一键时，就显示其接通和断开扫描码，可以利用ESC键退

出程序执行。键盘的每个字节代码都引起一次09H号中断，这样大部分按键将产生两次中断，按下按键盘发送接通扫描码，松开按键发送断开扫描码。例如，ESC键是01H和81H。83键标准键盘以后的增加的按键可能有多个。请问主键盘区和数字小键盘区的两个回车的扫描码分别是什么？

〔解答〕

```
        ; 数据段
done    byte 0
        ; 代码段，主程序
        mov ax,3509h
        int 21h
        push es
        push bx
        cli
        push ds
        mov dx,seg new09h
        mov ds,dx
        mov dx,offset new09h
        mov ax,2509h
        int 21h
        pop ds
        in al,21h
        push ax
        and al,0fdh
        out 21h,al
        sti
start1: cmp done,1
        jne start1
        cli
        pop ax
        out 21h,al
        pop dx
        pop ds
        mov ax,2509h
        int 21h
        sti
        ; 代码段，子程序
new09h  proc
        sti
        push ax
        push bx
        in al,60h
        push ax
        in al,61h
        or al,80h
        out 61h,al
        and al,7fh
        out 61h,al
        pop ax
        cmp al,1
        je next3
        push ax
        shr al,4
        cmp al,0ah
        jb next1
        add al,7
next1:  add al,30h
```

```
        mov bx,0
        mov ah,0eh
        int 10h
        pop ax
        and al,0fh
        cmp al,0ah
        jb next2
        add al,7
next2:  add al,30h
        mov ah,0eh
        int 10h
        mov ax,0e20h     ; 输出两个空格，分隔
        int 10h
        mov ax,0e20h
        int 10h
        jmp next4
next3:  push ds
        mov ax,@data
        mov ds,ax
        mov done,1
        pop ds
next4:  mov al,20h
        out 20h,al
        pop bx
        pop ax
        iret
new09h  endp
```

利用上述程序，可以获得主键盘区的回车键的扫描码是：1C 9C。
数字小键盘区的回车键的扫描码是：ED 1C ED 9C。
〔习题8.14〕
串行异步通信发送8位二进制数01010101：采用起止式通信协议，使用奇校验和2个停止位。画出发送该字符时的波形图。若用1200 bps，则每秒最多能发送多少个数据？
〔解答〕
每个字符的位数是：1个起始位＋8个数据位＋1个奇校验位＋2个停止位＝12位，采用1200bps、即每秒1200位的传送速率，则每秒最多能发送1200÷12＝100个数据。
〔习题8.15〕
微机与调制解调器通过232C总线连接时，常使用哪9个信号线？各自的功能是什么？利用232C进行两个微机直接相连通信时，可采用什么连接方式，画图说明。
〔解答〕
常用的9个信号线及其各自的功能：
TxD：串行数据发送端。
RxD：串行数据接收端。
RTS：发送请求信号，用于通知数据通信设备准备接收数据。
CTS：清除发送， CTS信号有效响应RTS信号，即允许发送。RTS和CTS是一对用于数据发送的联络信号。
DTR：数据终端准备就绪信号
DSR：数据装置准备好信号；DTR和DSR也可用做数据终端设备与数据通信设备间的联络信号。
GND：信号地，它为所有的信号提供一个公共的参考电平。
CD：载波检测信号，当本地调制解调器接收到来自对方的载波信号时，就从该引脚向数据终端设备提供有效信号。
RI：振铃指示，当调制解调器接收到对方的拨号信号期间，该引脚信号作为电话铃响的指示、保持有效。
利用232C进行两个微机直接相连通信时，可采用教材图8-25所示连接方式。
〔习题8.16〕
8250的IIR是只读的，且高5位总是0。试分析XT机系统ROM-BIOS中下段程序的作用。如不发

生条件转移，则RS232-BASE字单元将存放什么内容？

```
        mov bx,0
        mov dx,3fah
        in al,dx
        test al,0f8h
        jnz F18
        mov RS232-BASE,3f8h
        inc bx
        inc bx
F18:    mov dx,2fah
        in al,dx
        test al,0f8h
        jnz F19
        mov RS232-BASE[bx],2f8h
        inc bx
        inc bx
F19:    ……
```

〔解答〕

ROM-BIOS中该段程序的作用是检测是否存在串行异步通信接口电路。

如果不发生条件转移，说明存在异步通信接口电路，RS232-BASE字单元存放异步通信接口电路的基地址：3F8H和2F8H。

〔习题8.17〕

首先采用自循环查询方式在本机上实现例题8-3。然后购买或制作一个用于零调制解调器连接的RS-232C电缆，修改例题8-3采用正常的查询方式实现两台微机的通信。如果在Windows的模拟DOS环境无法运行程序，则应该采用纯DOS启动微机，在实方式下运行。读者还可以改进例题8-3的功能，例如每当按下回车键才将刚输入的字符串发送给对方，本机也显示发送的信息。

〔解答〕

〔习题8.18〕

说明在模拟输入输出系统中，传感器、放大器、滤波器、多路开关、采样保持器的作用。

DAC和ADC芯片是什么功能的器件？

〔解答〕

传感器：将各种现场的物理量测量出来并转换成电信号。

放大器：放大器把传感器输出的信号放大到ADC所需的量程范围。

低通滤波器：滤波器用于降低噪声、滤去高频干扰，以增加信噪比。

多路开关：对多个模拟信号分时地接通到A/D转换器上转换，达到共用A/D转换器以节省硬件的目的。

采样保持器：对高速变化的信号，使用采样保持器可保证A/D转换期间信号不变，保证转换精度。

D/A转换器：将微机处理后的数字量转换成为模拟量（电压或电流）。

A/D转换器：将模拟量（电压或电流）转换成为数字量输入微机处理。

〔习题8.19〕

假定某8位ADC输入电压范围是－5V～＋5V，求出如下输入电压Vin的数字量编码（偏移码）：

① 1.5V　② 2V　③ 3.75V　④ －2.5V　⑤ －4.75V。

〔解答〕

① A7H　② B4H　③ E0H　④ 40H　⑤ 06H

〔习题8.20〕

ADC的转换结束信号起什么作用，可以如何使用该信号，以便读取转换结果？

〔解答〕

当A/D转换结束，ADC输出一个转换结束信号，通知主机读取结果。

有多种使用A/D转换结束信号的方法，对应的程序设计方法也不同。

查询方式：把结束信号作为状态信号经三态缓冲器送到主机系统数据总线的某一位上。主机不断查询这个状态位，发现结束信号有效，便读取数据。

中断方式：把结束信号作为中断请求信号接到主机的中断请求线上。ADC转换结束，主动向

处理器申请中断。处理器响应中断后，在中断服务程序中读取数据。
DMA传送方式：如果ADC速度足够快，可把结束信号作为DMA请求信号，采用DMA传送方式。
延时传送方法：不使用结束信号，微机延时到转换结束读取数据。
〔习题8.21〕
某控制接口电路如本题图形。需要控制时，8255A的PC7输出一个正脉冲信号START启动A/D转换；ADC转换结束在提供一个低脉冲结束信号EOC的同时送出数字量。处理器采集该数据，进行处理，产生控制信号。现已存在一个处理子程序ADPRCS，其入口参数是在AL寄存器存入待处理的数字量，出口参数为AL寄存器给出处理后的数字量。假定8255端口A，B，C及控制端口的地址依次为FFF8H～FFFBH，要求8255的端口A为方式1输入、端口B为方式0输出。编写采用查询方式读取数据，实现上述功能的程序段。
〔解答〕

```
        ; 8255A初始化
        mov al,1011000×b
        mov dx,0fffbh
        out dx,al
        ; 使PC7＝0（START为低）
        mov al,00001110b
        mov dx,0fffbh
        out dx,al
        ; 启动A/D转换
        mov al,00001111b
        mov dx,0fffbh
        out dx,al       ; 使PC7＝1（START为高）
        nop
        mov al,00001110b
        out dx,al       ; 使PC7＝0（START为低）
        ; 查询是否转换结束
        mov dx,0fffah
again:  in dx,al
        test al,20h
        ; PC5＝0（转换未结束，继续检测）
        jz again
        ; PC5＝1（转换结束）
        mov dx,0fff8h   ; 输入数据
        in al,dx
        call adprcs     ; 处理数据
        mov dx,0fff9h
        out dx,al       ; 输出数据
```

〔习题8.22〕
图8-19c矩阵键盘还可以使用反转方法识别按键。首先，将行线作为控制线接一个输出端口，将列线作为检测线接一个输入端口。CPU通过输出端口将行线（控制线）全部设置为低电平，然后从输入端口读取列线（检测线）。如果此时有键被按下，则必定会使某列线为“0”。然后，将行线和列线的作用互换，即将列线作为控制线接输出端口，行线作为检测线接输入端口。并且，将刚才读得的列值从列线所接端口输出，再读取行线的输入值，那么，闭合键所在的行线值必定为“0”。这样，当一个键被按下时，必定可以读得一对唯一的行值和列值。
能够采用反转法识别按键需要一个条件：连接行线和列线的接口电路必须支持动态改变输入、输出方式。图8-19c使用8255A，其3个端口可以编程改变。请编写扫描程序。
〔解答〕

第9章 处理器性能提高技术
〔习题9.1〕简答题
〔解答〕
（1）传统上提高计算机性能的方法是采用复杂的、功能强大的指令，即复杂指令集计算机结构CISC。精简指令集计算机结构RISC从根本上打破了这个观念，使用简单指令，更便于硬件实现高性能。

（2）分支预测技术是用于解决指令流水线技术中存在的控制相关，即转移指令导致性能降低的问题。
（3）浮点数据采用规格化形式可以表达更大、更精确的数据，也避免编码的多样性。
（4）浮点数据编码无法表达任意精度的数据，所以需要舍入。但整数编码表达的数据都是精确的数据。
（5）多媒体指令的一个突出特点是一条指令同时处理多组数据，即单指令多数据SIMD。
〔习题9.2〕判断题
〔解答〕
① 对　② 对　③ 错　④ 错　⑤ 错
〔习题9.3〕填空题
〔解答〕
① Complex Insruction Set Computer，复杂指令集计算机，Reduced，精简的，CISC
② 指令预取（PF）、指令译码1（D1）、指令译码2（D2）、指令执行（EX）和回写（WB）
③ ±2－126，±（2－2－23）×2127，下溢，上溢
④ 32，8，23
⑤ CEH，7FH
〔习题9.4〕
通过处理器性能公式，说明影响程序执行时间的三个方面。
〔解答〕
处理器执行时间＝IC×CPI×T
其中：　IC为程序的指令条数
　　CPI为执行每条指令所需的平均时钟周期数
　　T为每个时钟周期的时间，也就是时钟频率的倒数
〔习题9.5〕
什么是简单指令和复杂指令，结合RISC处理器，说明把指令分为简单和复杂的原因。
〔解答〕
简单指令是指计算机基本的、常用的指令，往往其功能也比较简单。而复杂指令是指功能强大的指令，但往往不常用。
计算机大部分时间是在执行简单指令，复杂指令的使用频度都比较低。对一个CISC结构的指令系统而言，只有约20％的指令被经常使用，其使用量约占整个程序的80％；而该指令系统中大约80％的指令却很少使用，其使用量仅占整个程序的20％；而且使用频度较高的指令通常是那些简单指令。
〔习题9.6〕
RISC技术有哪些方面的主要特色？
〔解答〕
指令条数较少。
寻址方式简单。
面向寄存器操作。
指令格式规整。
便于使用先进的流水线技术。
……
〔习题9.7〕
什么是指令流水线？80486采用哪几级流水线，各级的主要操作分别是什么？
〔解答〕
指令流水线是多条指令重叠执行的一个处理器实现技术（在一条指令还没有执行结束就开始后续指令）。
Intel 80486采用5级指令流水线：
① PF步骤——指令预取（Prefetch）。处理器总是从高速缓存读取一个Cache行，为16个字节，平均包括5条指令。
② D1步骤——指令译码1（Decode Stage 1）。指令译码分成了2个步骤，D1步骤对所有操作码和寻址方式信息进行译码。
③ D2步骤——指令译码2（Decode Stage 2）。D2步骤将每个操作码扩展为ALU的控制信号，并进行较复杂的存储器地址计算。
④ EX步骤——指令执行（Execute）。EX步骤完成ALU操作和Cache存取。
⑤ WB步骤——回写（Write Back）。WB步骤更新在EX步骤得到的寄存器数据和状态标志。
〔习题9.8〕

影响流水线效率的主要指令相关有哪三个方面？
〔解答〕
资源冲突、数据相关，控制相关
〔习题9.9〕
已知BF600000H是一个单精度规格化浮点格式数据，它表达的实数是什么？
〔解答〕
BF600000H＝1011 1111 0110 0000 0000 0000 0000 0000 B
BF600000H＝1 01111110 11000000000000000000000 B
符号位为1，表示负数。
指数编码是01111110，表示指数＝126－127＝－1。
有效数字部分是11000000000000000000000，表示有效数＝1.11 B＝1.75。
所以，这个实数为：－1.75×2-1＝－1.75×0.5＝－0.875。
〔习题9.10〕
实数真值28.75如果用单精度规格化浮点数据格式表达，其编码是什么。编程将单精度浮点数据的编码显示出来。
〔解答〕
28.75＝0001 1100.11B＝1.110011B ×24
于是，符号位＝0。
指数部分是4，8位阶码为10000011（＝4＋127＝131）。
有效数字部分是11001100000000000000000。
这样，28.75表示成单精度浮点数为：

```
        0 10000011 11001100000000000000000 B
        ＝0100 0001 1110 0110 0000 0000 0000 0000 B
        ＝41E60000H
```

程序如下：

```
        ; 数据段
f32d    real4 28.75      ; 单精度浮点数
        ; 代码段
        mov eax,dword ptr f32d  ; 取28.75的浮点格式编码
        call disphd
```

〔习题9.11〕
解释如下浮点格式数据的有关概念：
（1） 数据上溢和数据下溢
（2） 规格化有限数和非规格化有限数
（3） NaN和无穷大
〔解答〕
（1）
数据上溢：当数据比能够表达的最小数还要小、还要接近0时，就是数据下溢。
数据下溢：当数据比能够表达的最大数还要大时，就是数据上溢。
（2）
规格化有限数：使用规格化格式表达的数据。它表达的数值是：1.XXX…XX。它的最高位恒为1，随后都是小数部分；有效数字只需要表达小数部分，隐含一个整数1。
非规格化有限数：指数编码为全0；有效数字仅表示小数部分、但不能是全0，表示的数值是：0.XXX…XX，用于表达比规格化格式不能表达的更小的实数。
（3）
NaN：指数编码是全1、有效数字编码不是全0的浮点格式编码，被称为非数NaN（Not a Number），它不是实数。
无穷大：指数编码为全1，有效数字编码为全0的浮点格式编码，用于表达大于规格化浮点数所能表达的最大数的真值。
〔习题9.12〕
什么是紧缩整型数据和紧缩浮点数据？扩展有SSE3指令的Pentium 4支持哪些紧缩数据类型？
〔解答〕
紧缩整型数据是指多个8、16、32或64位的整型数据组合形成一个整体。
紧缩浮点数据是指多个单精度或双精度浮点数据组合形成一个整体。
SSE3指令的Pentium 4支持128位紧缩数据类型，具体有：

紧缩字节、字、双字、4字整型数据；
紧缩单精度浮点数据；
紧缩双精度浮点数据。
〔习题9.13〕
SIMD是什么？举例说明MMX指令如何利用这个结构特点？
〔解答〕
SIMD表示单指令多数据（Single Instructon Multiple Data）。
例如，PADDB一条指令可以实现8组整型字节数据的求和，得到独立的8个结果。
〔习题9.14〕
什么是环绕运算和饱和运算。给出如下结果：
（1）环绕加：7F38H＋1707H
（2）环绕减：1707H－7F38H
（3）无符号饱和加：7F38H＋1707H
（4）无符号饱和减：1707H－7F38H
（5）有符号饱和加：7F38H＋1707H
（6）有符号饱和减：1707H－7F38H
〔解答〕
环绕运算：通常的算术运算，即当无符号数据的运算结果超过其数据类型界限时，它进行正常进位借位。但是，每个进位或借位并不能反映出来。
饱和运算：指运算结果超过其数据界限时，其结果被最大或最小值所替代。
（1）环绕加：7F38H＋1707H＝963FH
（2）环绕减：1707H－7F38H＝87CFH
（3）无符号饱和加：7F38H＋1707H＝963FH
（4）无符号饱和减：1707H－7F38H＝0000H（饱和）
（5）有符号饱和加：7F38H＋1707H＝7FFFH（饱和）
（6）有符号饱和减：1707H－7F38H＝87CFH
〔习题9.15〕
简单说明如下名词（概念）的含义：
（1）Load-Store结构
（2）超级指令流水线
（3）指令流水线的时空图
（4）指令相关
（5）就近舍入
〔解答〕
（1）Load-Store结构：处理器内部设置较多的通用寄存器，使多数操作（算术逻辑运算）都在寄存器与寄存器之间，只有“取数Load”和“存数Store”指令访问存储器。或者说，访问存储器只能通过Load和Store指令实现。
（2）超级指令流水线：将指令流水线的步骤（阶段）化分得更多，并加倍内部时钟频率，使紧接着的2个步骤可以重叠一部分执行，使得每个时钟可以完成多条指令的执行，进而提高指令流水线的性能。
（3）指令流水线的时空图：描绘流水线操作的时间空间图、简称时空图，其横坐标表示时间，纵坐标是指令处理的各个阶段、表示空间。
（4）指令相关：指令流水线中，指令之间存在相互依赖关系，使得下一条指令无法在设计的单位时间内执行的情况。
（5）就近舍入：浮点数据的舍入方法之一，类似“四舍五入”原则。舍入结果最接近准确值。如果上下两个值一样接近，就取偶数结果（最低位为0）。
第10章 并行处理技术
〔习题10.1〕简答题
（1）英特尔所谓的微结构对应计算机层次结构的哪个层次？
（2）新一代IA-32处理器将指令译码为微操作有什么特别的作用？
（3）乱序执行是什么含义？
（4）Flynn分类法以什么内容为分类依据？
（5）为什么说动态超标量技术为软件提供了免费“性能午餐”？
〔解答〕
（1）微结构对应计算机层次结构的控制层。
（2）IA-32处理器将指令译码为微操作可以将复杂指令转换为简单指令，便于硬件实现。

（3）乱序执行是指令的执行不一定是传统的串行顺序方式，可能会出现后面指令先执行的情况。
（4）以并行操作的指令流个数和数据流个数为分类依据。
（5）因为动态超标量技术采用硬件实现性能提高，即使软件没有改进也能提高性能。
〔习题10.2〕判断题
（1）在Pentium中，只要两条指令不存在数据相关，就能配对并行执行。
（2）指令流水线运用了时间重叠思想提高并行性。
（3）数据相关可以用寄存器重命名技术消除。
（4）Intel 64结构支持16个64位整数通用寄存器。
（5）Intel Core微结构支持超线程技术。
〔解答〕
① 错 ② 对 ③ 错 ④ 对 ⑤ 错
〔习题10.3〕填空题
（1）有两种性质的并行性，同一个时刻发生的并行性称_____，同一段时间内发生的并行性称_____。
（2）Pentium处理器中，某时刻执行一条转移指令，它在BTB的历史位为10，则预测_____分支；如果确实发生分支，则该历史位成为_____；而如果预测错误，则该历史位成为_____。如果某个转移指令的实际执行情况是分支“发生—不发生—发生—不发生—发生—不发生……”，则利用Pentium的分支预测机构，它的准确度约为百分之_____。
（3）基于IA-64结构的安腾处理器使用了_____技术，一个128位的指令束包括_____条指令，还有指示指令并行性的模板域。
（4）Intel 64结构新增8个通用寄存器，名称为_____；原来的8个通用寄存器被扩展为64位，名称为_____。
（5）多核处理器属于多处理器系统，对应Flynn分类法的_____系统。
〔解答〕
① 同时性，并发性
② 发生转移，11，01，50
③ VILW（EPIC），3
④ R8～R15，RAX～RSP
⑤ MIMD
〔习题10.4〕
对比Intel 80486指令流水线和Pentium超标量指令流水线，指出它们的异同。
〔解答〕
Pentium的超标量整数指令流水线的各个阶段类似Intel 80486，仍分成了5个步骤，但是其后3个步骤可以在它的2个流水线（U流水线和V流水线）同时进行。
〔习题10.5〕
Pentium超标量指令流水线为什么限制复杂指令、存在数据相关的指令和转移指令等不能实现配对执行？
〔解答〕
因为Pentium的两条指令流水线U和V并不是完全相同的，例如V流水线只能执行简单指令，所以不可能实现复杂指令的配对执行。
存在数据相关的指令，需要执行完前一条指令才能得到后一条指令需要的操作数，所以也无法同时进行执行。
存在转移指令，需要执行完前一条指令才能决定是否执行后一条指令，所以也不能配对执行。
〔习题10.6〕
什么是Pentium的动态分支预测和Pentium II的静态分支预测？对于如下两个程序片断，分别指出Pentium处理器第一次执行转移指令时动态分支预测的结果和Pentium II的静态分支预测的结果。
（1）单分支结构

```
        mov ecx,12
        cmp eax,7
        jne EAX_7
        mov ecx,17
EAX_7:
```

（2）循环结构

```
loopx:  add list[eax*4],5
        dec eax
        jnz loopx
```

〔解答〕

动态分支预测：Pentium维持一个分支目标缓冲器BTB，用它记录最近使用的转移指令的有关情况，并动态预测当前转移指令是否发生分支；在预测正确时就可以无延迟的执行，如果预测错误，则产生3～4个时钟的延迟。

静态分支预测：解决对于分支目标缓冲器BTB中（动态分支预测）没有记录的转移指令的分支预测问题。预测无条件转移指令发生分支。预测向前分支的条件转移指令不发生分支。预测向后分支的条件转移指令发生分支。

（1）Pentium动态分支预测：不分支。Pentium II静态分支预测：不分支

（2）Pentium动态分支预测：不分支。Pentium II静态分支预测：分支

〔习题10.7〕

说明NetBurst微结构的踪迹Cache为什么优于P6微结构的L1指令Cache？Core微结构的双核处理器共享L2 Cache为什么优于Pentium D独立使用L2 Cache？

〔解答〕

踪迹Cache存储已译码指令即微操作。存储已译码指令使得IA-32指令的译码从主要执行循环中分离出来。指令只被译码一次，并被放置于踪迹Cache，然后就象常规指令Cache一样重复使用。P6微结构的L1指令Cache仍然是存储原始指令代码的指令Cache，取指后还需要译码。Core微结构中，两个处理器共享L2 Cache，可以充分利用Cache空间为两个处理器共用。每个处理器各有独立的L2 Cache，则Cache只能为对应的处理器服务。

〔习题10.8〕

简单说明如下名词（概念）的含义：

（1）超标量技术

（2）指令级并行

（3）乱序执行

（4）寄存器重命名

（5）推测执行

（6）VLIW

（7）EPIC

（8）超线程技术

（9）SIMD

（10）多核处理器

〔解答〕

（1）超标量技术：指提高标量指令的执行性能而设计的一种处理器技术。处理器采用多条指令流水线，可以实现一个时钟周期完成多条指令的执行。

（2）指令级并行：指令是处理器执行的基本单位。指令级并行是指发掘指令之间的并行执行能力，也就是提高处理器内部操作的并行程度。

（3）乱序执行：多条指令进入处理器执行单元，指令执行的顺序不必一定按照程序顺序、可能是乱序的，即乱序执行。

（4）寄存器重命名：对于使用相同寄存器名引起的假数据相关，只要更改寄存器名，就可以消除指令相关的技术。

（5）推测执行：处理器采用分支预测技术推测指令分支路径，并按照推测结果发送和执行指令，这就是推测执行。

（6）VLIW：超长指令字，一种利用软件编译器方法提高指令并行执行能力的技术。

（7）EPIC：显式并行计算，源于VLIW技术，64位Itanium系列处理器主要采用的提高性能的技术。

（8）超线程技术：它使一个物理处理器看似有两个逻辑处理器，每个逻辑处理器维持一套完整的结构状态，共享几乎物理处理器上所有执行资源。

（9）SIMD：单指令多数据，指一条指令可以同时对多组数据进行操作（执行）。

（10）多核处理器：在一个物理封装内制作了两个或多个处理器执行核心形成的处理器。

〔习题10.9〕

简单总结新一代IA-32处理器的结构特点：

（1）Pentium的超标量指令流水线

（2）Pentium II的动态执行结构

（3）Pentium 4的超线程技术
（4）多核处理器结构
〔解答〕
（1）Pentium的超标量指令流水线：利用资源重复的思想，Pentium处理器采用超标量技术，设计了2个可以并行操作的执行单元，形成了2条指令流水线。这样，在一定条件下，Pentium允许在一个时钟周期中同时运行2条整数指令，或者运行一条浮点指令（但浮点交换指令可以与另一条浮点指令配对同时执行）。
（2）Pentium II的动态执行结构：包含3个组成部分：顺序发送前端、乱序核心和顺序退出单元，它们之间通过重排序缓冲区ROB（Re-Order Buffer）建立联系，具有采用3路超标量、12级超级流水线。
（3）Pentium 4的超线程技术：超线程技术为IA-32结构引入了同时多线程概念。它使一个物理处理器看似有两个逻辑处理器。操作系统和用户程序像传统多处理器系统一样在逻辑处理器上调度线程或进程。两个逻辑处理器的指令可以在共享的执行资源上同时保持和执行。
（4）多核处理器结构：在一个物理封装内制作了两个或多个处理器执行核心，使多个处理器耦合得更加紧密，同时共享系统总线、主存等资源，可以有效地执行多线程的应用程序。
〔习题10.10〕
追踪处理器技术最新发展，选择某个方面，做一篇新技术发展的论文。