第 1 章 微型计算机系统

〔习题 1.1〕简答题

（1）计算机字长（Word）指的是什么？

（2）总线信号分成哪三组信号？

（3）PC 机主存采用 DRAM 组成还是 SRAM 组成？

（4）Cache 是什么意思？

（5）ROM-BIOS 是什么？

（6）中断是什么？

（7）32 位 PC 机主板的芯片组是什么？

（8）教材中 MASM 是指什么？

（9）处理器的“取指－译码－执行周期”是指什么？

（10）本课程的主要内容属于计算机系统层次结构中哪个层次？

〔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）软件与硬件的等价性原理说明软硬件在功能、性能和成本等方面是等价的。

（2）IA-64 结构是 IA-32 结构的 64 位扩展，也就是 Intel 64 结构。

（3）8086 的数据总线为 16 位，也就是说 8086 的数据总线的个数、或说条数、位数是 16。

（4）微机主存只要使用 RAM 芯片就可以了。

（5）处理器并不直接连接外设，而是通过 I/O 接口电路与外设连接。

（6）处理器是微机的控制中心，内部只包括 5 大功能部件的控制器。

（7）Windows 的模拟 DOS 环境与控制台环境是一样的。

（8）16 位 IBM PC/AT 机采用 ISA 系统总线。

（9）IA-32 处理器吸取了 RISC 技术特长。RISC 是指复杂指令集计算机。

（10）处理器进行读操作，就是把数据从处理器内部读出传送给主存或外设。

〔解答〕

① 错　② 错　③ 对　④ 错　⑤ 对

⑥ 错　⑦ 错　⑧ 对　⑨ 错　⑩ 错

〔习题 1.3〕填空题

（1）CPU 是英文_____的缩写，中文译为_____，微型机采用_____芯片构成 CPU。

（2）Intel 8086 支持_____容量主存空间，80486 支持_____容量主存空间。

（3）二进制 16 位共有_____个编码组合，如果一位对应处理器一个地址信号，16 位地址信号共能寻址_____容量主存空间。

（4） DOS 主要支持两种可执行文件，它们的扩展名分别是_____和_____。

（5）英文缩写 ISA 常表示 PC 机工业标准结构（Industry Standard Architecture）总线，也表示指令集结构，后者的英文原文是_____。

（6）Windows 的文件夹对应的专业术语是_____。

（7）Pentium 系列处理器的多媒体指令有_____，SSE，SSE2 和_____类指令。

（8）Pentium 处理器采用_____位数据总线与主存相连。

（9）最初由_____公司采用 Intel 8088 处理器和_____操作系统推出 PC 机。

（10）当前 32 位 PC 机主要采用_____总线连接 I/O 接口电路卡。

〔解答〕

① Central Processing Unit，中央处理单元，处理器

② 1MB，4GB

③ 2^{16}，64KB

④ EXE，COM

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

（1）ALU 是什么？

（2）8086 的取指为什么可以被称为指令预取？

（3）Pentium 的片上 Cache 采用统一结构还是分离结构？

（4）堆栈的存取原则是什么？

（5）标志寄存器主要保存哪方面的信息？

（6）执行了一条加法指令后，发现 ZF＝1，说明结果是什么？

（7）汇编语言中的标识符与高级语言的变量和常量名的组成原则有本质的区别吗？

（8）汇编语言的标识符大小写不敏感意味着什么？

（9）汇编语言源程序文件中，END 语句后的语句会被汇编吗？

（10）为什么将查找操作数的方法称为数据寻“址”方式？

〔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

（1）程序计数器 PC 或指令指针 EIP 寄存器属于通用寄存器。

（2）处理器的指令译码是将指令代码翻译成它代表的功能的过程，与数字电路的译码器是不同的概念。

（3）EAX 也被称为累加器，因为它使用最频繁。

（4）处理器的传送指令 MOV 属于汇编语言的执行性语句。

（5）汇编语言的语句由明显的 4 部分组成，不需要分隔符区别。

（6）80 减 90（80－90）需要借位，所以执行结束后，进位标志 CF＝1。

（7）MASM 汇编语言的注释用分号开始，但不能用中文分号。

（8）IA-32 处理器在实地址方式下，不能使用 32 位寄存器。

（9）存储器寻址方式的操作数当然在主存了。

（10）保护方式下，段基地址加偏移地址就是线性地址或物理地址。

〔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

（1）寄存器 EDX 是_____位的，其中低 16 位的名称是_____，还可以分成两个 8 位的寄存器，其中 D8～D15 部分可以用名称_____表示。

（2）IA-32 处理器在保护方式下，段寄存器是_____位的。

（3）逻辑地址由_____和_____两部分组成。代码段中下一条要执行的指令由 CS 和_____寄存器指示，后者在实地址模型中起作用的仅有_____寄存器部分。

（4）进行 8 位二进制数加法：＋，8 位结果是_____，标志 PF＝_____。

（5）在实地址工作方式下，逻辑地址“7380H：400H”表示的物理地址是_____，并且该段起始于_____物理地址。

（6）IA-32 处理器有 8 个 32 位通用寄存器，其中 EAX，_____，_____和 EDX，可以分成 16 位和 8 位操作；还有另外 4 个是_____，_____，_____，和_____。

（7）IA-32 处理器复位后，首先进入是_____工作方式。该工作方式分段最大不超过_____。

（8）MASM 要求汇编语言源程序文件的扩展名是_____，汇编产生扩展名为 OBJ 的文件被称为_____文件，编写 32 位 Windows 应用程序应选择_____存储模型。

（9）除外设数据外的数据寻址方式有 3 类，分别被称为_____、_____和_____。

（10）用 EBX 做基地址指令，默认采用_____段寄存器指向的数据段；如果采用 BP，EBP 或 SP，ESP 作为基地址指针，默认使用_____段寄存器指向堆栈段。

〔解答〕

① 32，DX，DH
② 16
③ 段地址，偏移地址，EIP，IP
④ ，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

（1）使用二进制 8 位表达无符号整数，257 有对应的编码吗？

（2）字符“'F'”和数值 46H 作为 MOV 指令的源操作数有区别吗？

（3）为什么可以把指令“MOV AX,(34＋67H)*3”中的数值表达式看成是常量？

（4）数值 500，能够作为字节变量的初值吗？

（5）为什么说“XCHG EDX,CX”是一条错误的指令？

（6）都是获取偏移地址，为什么指令“LEA EBX,[ESI]”正确，而指令“MOV EBX, OFFSET[ESI]”就错误？

（7）INC，DEC，NEG 和 NOT 都是单操作数指令，这个操作数应该是源操作数还是目的操作数？

（8）大小写字母转换使用了什么规律？

（9）乘除法运算针对无符号数和有符号数，有两种不同的指令。只有一种指令的加减法如何区别无符号数和有符号数运算？

（10）逻辑与运算为什么也称为逻辑乘？

〔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

（1）对一个正整数，它的原码、反码和补码都一样，也都与无符号数的编码一样。

（2）常用的 BCD 码为 8421 BCD 码，其中的 8 表示 D3 位的权重。

（3）IA-32 处理器采用小端方式存储多字节数据。

（4）空操作 NOP 指令其实根本没有指令。

（5）堆栈的操作原则是“先进后出”，所以堆栈段的数据除 PUSH 和 POP 指令外，不允许其他方法读写。

（6）虽然 ADD 指令和 SUB 指令执行后会影响标志状态，但执行前的标志并不影响它们的执行结果。

（7）指令“INC ECX”和“ADD ECX,1”的实现功能完全一样，可以互相替换。

（8）无符号数在前面加零扩展，数值不变；有符号数前面进行符号扩展，位数加长一位、数值增加一倍。

（9）逻辑运算没有进位或溢出问题，此时 CF 和 OF 没有作用，所以逻辑运算指令如 AND、OR 等将 CF 和 OF 设置为 0。

（10）CMP 指令是目的操作数减去源操作数，CMPS 指令是源操作数减去目的操作数。

〔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

（1）定义字节变量的伪指令助记符是_____，获取变量名所具有的偏移地址的操作符是_____。

（2）计算机中有一个“”编码。如果把它认为是无符号数，它是 10 进制数_____；如果认为它是 BCD 码，则表示真值_____；又如果它是某个 ASCII 码，则代表字符_____。

（3）C 语言用“\n”表示让光标回到下一行首位，在汇编语言中需要输出两个控制字符：一个是回车，其 ASCII 码是_____，它将光标移动到当前所在行的首位；另一个是换行，其 ASCII 码是_____，它将光标移到下一行。

（4）数据段有语句“H8843 DWORD H”，代码段指令“MOV CX, WORD PTR H8843”执行后，CX＝_____。

（5）用 DWORD 定义的一个变量 XYZ，它的类型是_____，用“TYPE XYZ”会得到数值为_____。如果将其以字量使用，应该用_____说明。

（6）数据段有语句“ABC BYTE 1,2,3”，代码段指令“MOV CL, ABC+2”执行后，CL＝_____。

（7）例题 3-9 的 TAB 定义如果是“”，则显示结果是_____。

（8）指令“XOR EAX, EAX”和“SUB EAX, EAX”执行后，EAX＝_____，CF＝OF＝_____。而指令“MOV EAX, 0”执行后，EAX＝_____，CF 和 OF 没有变化。

（9）例题 3-15 程序执行结束，变量 QVAR 内容是_____；BCD 内容是_____。

（10）欲将 EDX 内的无符号数除以 16，使用指令“SHR EDX, _____”，其中后一个操作数是一个立即数。

〔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
⑧ 0，0，0
⑨ 65432H，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　②　③　④

〔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
②
③
④

〔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 b
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
datalistbyte -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 h
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,20h
        byte '30 |',30h,20h,31h,20h,32h,20h,33h,20h,34h,20h,35h,20h,36h,20h,37h,20h,38h,20h,39h,20h
        byte '40 |',40h,20h,41h,20h,42h,20h,43h,20h,44h,20h,45h,20h,46h,20h,47h,20h,48h,20h,49h,20h
        byte '50 |',50h,20h,51h,20h,52h,20h,53h,20h,54h,20h,55h,20h,56h,20h,57h,20h,58h,20h,59h,20h
        byte '60 |',60h,20h,61h,20h,62h,20h,63h,20h,64h,20h,65h,20h,66h,20h,67h,20h,68h,20h,69h,20h
        byte '70 |',70h,20h,71h,20h,72h,20h,73h,20h,74h,20h,75h,20h,76h,20h,77h,20h,78h,20h,79h,20h
        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 h
```

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＝H
        ; 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：

（1）

```
        mov eax,80h      ; EAX＝__________
        add eax,3        ; EAX＝__________, CF＝__________, SF＝__________
```

```
        add eax,80h       ; EAX=_____, CF
=_____, OF=_____
        adc eax,3         ; EAX=_____, CF
=_____, ZF=_____
```

(2)

```
        mov eax,100       ; EAX=_____
        add ax,200        ; EAX=_____, CF
=_____
```

(3)

```
        mov eax,100       ; EAX=_____
        add al,200        ; EAX=_____, CF
=_____
```

(4)

```
        mov al,7fh        ; AL=_____
        sub al,8          ; AL=_____, CF
=_____, SF=_____
        sub al,80h        ; AL=_____, CF
=_____, OF=_____
        sbb al,3          ; AL=_____, CF
=_____, ZF=_____
```

〔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 h
buffer2 qword h
        ; 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：

(1)

```
        mov esi,b         ; ESI=_____H
        and esi,80h       ; ESI=_____H
        or esi,7fh        ; ESI=_____H
        xor esi,0feh      ; ESI=_____H
```

(2)

```
        mov eax,1010b     ; EAX=_____B
        shr eax,2         ; EAX=_____B,
CF=_____
        shl eax,1         ; EAX=_____B,
CF=_____
        and eax,3         ; EAX=_____B,
CF=_____
```

(3)

```
        mov eax,1011b     ; EAX=_____B
        rol eax,2         ; EAX=_____B,
CF=_____
        rcr eax,1         ; EAX=_____B,
CF=_____
        or eax,3          ; EAX=_____B,
CF=_____
```

(4)

```
        xor eax,eax       ;EAX=_____, CF
=_____, OF=_____
                          ;  ZF=_____, SF
=_____, PF=_____
```

〔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＝B，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

（1）是什么特点决定了目标地址的相对寻址方式应用最多？

（2）什么是奇偶校验？

（3）为什么判断无符号数大小和有符号大小的条件转移指令不同？

（4）双分支结构中两个分支体之间的 JMP 指令有什么作用？

（5）为什么特别强调为子程序加上必要的注释？

（6）子程序采用堆栈传递参数，为什么要特别注意堆栈平衡问题？

（7）参数传递的“传值”和“传址”有什么区别？

（8）INCLUDE 语句和 INCLUDELIB 有什么区别？

（9）混合编程有什么优势？

（10）运行 Windows 程序，有时为什么会提示某个 DLL 文件不存在？

〔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

（1）指令指针或者还包括代码段寄存器值的改变将引起程序流程的改变。

（2）JMP 指令对应高级语言的 GOTO 语句，所以不能使用。

（3）因为条件转移指令 Jcc 要利用标志作为条件，所以也影响标志。

（4）JA 和 JG 指令的条件都是“大于”，所以是同一个指令的两个助记符。

（5）控制循环是否结束只能在一次循环结束之后进行。

（6）介绍 LOOP 指令时，常说它相当于 DEC ECX 和 JNZ 两条指令。但考虑对状态标志的影响，它们有差别。LOOP 指令不影响标志，而 DEC 指令却会影响除 CF 之外的其他状态标志。

（7）CALL 指令用在调用程序中，如果被调用程序中也有 CALL 指令，说明出现了嵌套。

（8）子程序需要保护寄存器，包括保护传递入口参数和出口参数的通用寄存器。

（9）利用 INCLUDE 包含的源文件实际上只是源

程序的一部分。

（10）导入库文件和静态子程序库文件的扩展名都是 LIB，所以两者性质相同。

〔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

（1）JMP 指令根据目标地址的转移范围和寻址方式，可以分成四种类型：段内转移、_____，段内转移、_____和段间转移、_____，段间转移、_____。

（2）假设在平展存储模型下，EBX＝1256H，双字变量 TABLE 的偏移地址是 20A1H，线性地址 32F7H 处存放 3280H，执行指令“JMP EBX” 后 EIP＝_____，执行指令“JMP TABLE[EBX]”后 EIP＝_____。

（3）“CMP EAX,3721H”指令之后是 JZ 指令，发生转移的条件是 EAX＝_____，此时 ZF＝_____。

（4）小写字母“e”是英文当中出现频率最高的字母。如果某个英文文档利用例题 4-11 的异或方法进行简单加密，统计发现密文中字节数据“8FH”最多，你判断该程序采用的字节密码可能是_____。

（5）循环结构程序一般有三个部分组成，它们是_____，循环体和_____部分。

（6）例题 4-14 程序中的 RET 指令，如果用 POP EBP 指令和 JMP EBP 指令替换，此时 EBP 内容是_____。

（7）过程定义开始是“TEST PROC”语句，则过程定义结束的语句是_____。宏定义开始是“DISP MACRO”语句，则宏定义结束的语句是_____。

（8）利用堆栈传递子程序参数的方法是固定的，例如寻址堆栈段数据的寄存器是_____。

（9）MASM 汇编语言中，声明一个共用的变量应使用_____伪指令；而使用外部变量要使用_____伪指令声明。

（10）调用 ReadConsole 函数时，用户在键盘上按下数字 8，然后回车，则键盘缓冲区的内容依次是_____。

〔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
labl4:  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,h
        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 h
        ; 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－2^{15}～＋2^{15}－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
checksumproc
        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
checksumendp
```

〔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 AFH
        ; 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 AFH
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 AFH
        ; 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)
errormsgbyte    '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

（1）为什么称处理器的数据总线是双向的？

（2）8086 的地址和数据总线为什么要分时复用？

（3）具有三态能力的引脚输出高阻意味着什么？

（4）总线周期中的等待状态是个什么工作状态？

（5）猝发传送是一种什么传送？

（6）总线数据传输为什么要进行总线仲裁？

（7）异步时序为什么可以没有总线时钟信号？

（8）32 位 PC 机为什么采用多级总线结构，而不是单总线结构？

（9）USB 总线有几个导线组成？

（10）什么是微软宣称的即插即用 PnP（Plug-and-Play）技术？

〔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

（1）低电平有效是指信号为低电平时候表示信号的功能。

（2）处理器读取存储器操作数时和读取代码时，都发生存储器读的总线操作。

（3）8086 准备好 READY 引脚输出给存储器或外设有效信号，表明处理器准备好交换数据了。

（4）8086 总线周期的 T1 状态发出地址，属于总线操作的寻址阶段。

（5）存储器单元以一个字节为基本单元，所以 Pentium 对应每 8 个数据总线引脚有一个奇偶校验信号。

（6）PCI 总线和 USB 接口都支持热插拔。

（7）ISA 总线仅支持 8 位和 16 位数据传输，PCI 总线还支持 32 位和 64 位数据传输。

（8）PCI 总线独立于处理器，所以其引脚信号多数并不与 IA-32 处理器对应。

（9）USB 总线结构中主机包含有根集线器。

（10）支持 USB 2.0 版本的 USB 设备一定能够以高速（480Mb/s）传输数据。

〔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

（1）某个处理器具有 16 个地址总线，通常可以用 A_____表达最低位地址信号，用 A15 表达最高地址信号。

（2）8086 有 3 个最基本的读写控制信号，它们是 M /$\overline{IO}$，_____和_____。

（3）8086 预取指令时，在其引脚上将产生_____总线操作；执行指令“MOV AX, [BX]”时，在其引脚上将产生_____总线操作；执行指令“MOV [BX], AX”时，在其引脚上将产生_____总线操作。

（4）8086 无等待的总线周期由_____个 T 状态组成，Pentium 无等待的总线周期由_____个 T 状态组成。如果处理器的时钟频率为 100MHz，则每个 T 状态的持续时间为_____。

（5）8086 处理器进行 I/O 读操作时，其引脚 M /$\overline{IO}$ 为低，引脚 $\overline{RD}$ 为_____；ISA 总线的_____引脚低有效说明进行 I/O 读操作。PCI 总线用 C/BE[3::0]#引脚编码为_____表示 I/O 读总线周期。

（6）占用总线进行数据传输，一般需要经过总线请求和仲裁、_____、_____和结束 4 个阶段。

（7）USB 总线理论上最多能够连接_____个 USB 设备，USB 2.0 支持低速_____、全速_____和高速 480Mb/s 三种速率。

（8）PCI 总线共用数据和地址信号，所以数据传输需要两个阶段：第一个阶段（一个时钟）提供

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/848105027135007003