7. 帮助命令 HELP

格式：HELP

作用：提供 DOS 各命令的联机帮助信息。

8. 设置路径命令 PATH

格式：PATH [[分区字母]路径[[；分区字母]路径]]

作用：为查找外部命令或文件设置检索的路径。

示例：

PATH D:\ML615;C:\　　设置搜索路径依次为：D:\ML615 和 C:\

该命令可以带有用分号隔开的多个路径。当你输入一个在当前目录里找不到的外部命令或文件时，DOS 按照你用 PATH 命令所定义的次序，寻找指定目录，然后调用命令文件。单独输入 PATH 命令可以显示已设置过的搜索路径。

计算机执行某一特定命令，其实质是执行一段完成特定功能的程序。DOS 有两种形式的命令：一个是内部命令，另一个是外部命令。内部命令的程序是建立在 DOS 系统内，它们在操作系统引导时已装入内存。外部命令的执行依赖于存储于磁盘中的命令文件。其执行过程是先把指定的命令文件读入内存，然后把控制转移到文件中的程序去执行，执行完毕后又回到 DOS 操作系统状态。命令文件是指扩展名为如下三种形式的文件：.COM、.EXE、.BAT。内部命令与外部命令在调用格式上没有多大区别，不同的是，在执行一个外部命令前，应确知此命令文件在哪个磁盘上，如果命令文件不存在，就必须找到才可以去调用执行。

如果在 Windows 操作系统下使用模拟 DOS 环境，大家也可以利用 Windows 资源管理器进行文件管理，图形化操作界面更方便。

1.3　习题解答

〔习题 1-1〕简答题（用一句话回答）

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 MASM 是指什么？
⑨ DOS 的批处理文件可以用 Windows 的记事本打开吗？
⑩ 数值协处理器和浮点处理单元是什么关系？

〔解答〕

① 微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 是微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 Microsoft 公司开发的宏汇编程序。

⑨ DOS 的批处理文件可以用 Windows 的记事本打开。

⑩ 数值协处理器和浮点处理单元都是处理浮点数据的器件，如果说有什么不同，前者多指独立的芯片、后者多指与其他处理器集成一体的功能单元。

〔习题 1-2〕判断题（判断如下论述是正确还是错误）

① 微处理器的引脚信号常被称为处理器总线。

② IA-32 微处理器是 32 位 Intel 80x86 微处理器。

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 微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 微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 环境运行于实地址方式。

⑧ 16 位 IBM PC/AT 机采用 ISA 系统总线。

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

⑩ 微处理器进行读操作，就是把数据从微处理器内部读出传送给主存或外设。

〔解答〕

① 对　② 对　③ 对　④ 错　⑤ 对

⑥ 错　⑦ 错　⑧ 对　⑨ 错　⑩ 错

〔习题 1-3〕填空题（填写数字或文字）

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

② 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，如果一位对应微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 Windows 2000 有两个模拟 DOS 的文件，一个是 COMMAND.COM，另一个是_____。

⑥ Windows 的文件夹对应的专业术语是_____。

⑦ Pentium 系列微处理器的多媒体指令有_____、SSE、SSE2 和_____指令。

⑧ Pentium 微处理器采用_____位数据总线与主存相连。

⑨ 最初由_____公司采用 8088 微处理器和_____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，微处理器

② 1MB，4GB

③ 2^{16}，64KB

④ EXE，COM

⑤ CMD.EXE

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 也称微处理器，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）。微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，它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 系列微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 微处理器还支持 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。

〔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。

〔习题 1-9〕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〕

指明的路径不正确，或者执行了另外一个同名的文件。

〔习题 1-10〕进入 MS-DOS 模拟环境，练习 DOS 常用命令。

〔解答〕

参考教材和本章的补充，具体的上机任务将在第 3 章布置。

段描述符）中，其中含有 32 位的段基地址。IA-32 微处理器编程仍使用逻辑地址，它由两部分组成：32 位的段基地址和 32 位的段内偏移地址。形成物理地址时，各个段寄存器的内容作为段选择符指向段描述符，由段描述符中取得 32 位段基地址；32 位偏移地址在代码段由 EIP 指令指针、在堆栈段由 ESP 堆栈指针、在数据段由各种 32 位主存寻址方式得到；段基地址加上偏移地址就形成了 32 位线性地址。如果不采用页式存储管理，则线性地址就是物理地址；如果还采用页式存储管理，则根据刚才得到的 32 位的线性地址先访问到需要的页目录项，再通过页目录项在页表中访问到需要的页表项，最后通过页表项就从这一页中访问到所寻址的物理单元。

2.2 习题解答

〔习题 2-1〕简答题（用一句话回答）

① ALU 是什么？
② 计算机学科中“透明”是什么含义？
③ “取指”是微处理器进行的什么操作？
④ Pentium 的片上 Cache 采用统一结构还是分离结构？
⑤ 堆栈的存取原则是什么？
⑥ 标志寄存器主要保存哪方面的信息？
⑦ 执行了一条加法指令后，发现 ZF=1，说明结果是什么？
⑧ 单步调试是怎么一回事？
⑨ 为什么称多媒体指令为 SIMD 指令？
⑩ 段描述符中的访问权字节有什么作用？

〔解答〕

① ALU 是算术逻辑运算单元，负责微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 “透明”是计算机学科中常用的一个专业术语，表示实际存在但从某个角度看好像没有。
③ 取指是指从主存取出指令代码通过总线传输到微处理器内部指令寄存器的过程。
④ Pentium 采用分离的 Cache 结构，一个用做指令 Cache，一个用做数据 Cache。
⑤ 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。
⑥ 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。
⑦ 执行了一条加法指令后，发现 ZF=1，表明运算结果为 0。
⑧ 单步调试就是逐条指令或语句进行调试，即在每条指令或语句执行结束时，程序暂停执行，调试程序显示当前运行状态。
⑨ 多媒体指令的一个关键技术是单指令多数据结构（Single Instruction Multiple Data，SIMD），所以多媒体指令常被称为 SIMD 指令。
⑩ 段描述符中访问权字节说明该段的访问权限，例如只读、读写、只执行等操作设定。

〔习题 2-2〕判断题（判断如下论述是正确还是错误）

① 程序计数器 PC 或指令指针 EIP 寄存器属于通用寄存器。
② 微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 从汇编语言这个层次看，微处理器被抽象为寄存器，因为只有通过编程寄存器才能控制微处理器。同理，主存也就被抽象为主存地址。
⑤ 堆栈段的栈顶位置用 SS 和 ESP 指示，后者具有专门用途，不属于 8 个通用寄存器。

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

⑦ 国际标准浮点格式中单精度和双精度分别是 32 位和 64 位，所以 IA-32 微处理器的浮点寄存器也是 64 位的。

⑧ IA-32 微处理器在实地址方式下，不能使用 32 位寄存器。

⑨ 在保护方式下，每个段描述符和门描述符，都是 8 个字节 64 位。

⑩ 在保护方式下，段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 错　⑧ 错　⑨ 对　⑩ 对

〔习题 2-3〕填空题（填写数字或文字）

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 微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 16 位段中起作用的仅有_____寄存器部分。

④ 进行 8 位二进制数加法：10111010＋01101100，8 位结果是_____，标志 PF=_____。

⑤ 在实方式下，逻辑地址“7380H：400H”表示_____物理地址，并且该逻辑段起始于_____物理地址。

⑥ IA-32 微处理器有 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 微处理器复位后，首先进入是_____工作方式。

⑧ 实地址方式分段最大不超过_____KB。

⑨ 在 MASM 汇编语言中，十六进制数据用后缀字母_____表示；而高级语言例如 C/C++则用前缀_____表示。

⑩ 已知 IA-32 微处理器某个段描述符为 0000B98200002000H，则该段基地址=_____，段界限=_____。

〔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

⑧ 64

⑨ H（h），0x

⑩ 00820000H，02000H

〔习题 2-4〕微处理器内部具有哪 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〕

微处理器内部有 ALU、寄存器和指令处理 3 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，负责微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？状态标志和控制标志有什么区别？画出 16 位标志寄存器 FLAGS，说明各个标志的位置和含义。

〔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。

状态标志和控制标志的区别：状态标志用来记录程序运行结果的状态信息；控制标志位可根据需要用指令设置，用于控制处理器执行指令的方式。

16 位标志寄存器 FLAGS 及各个标志的位置和含义参见教材第 25 页。

〔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：在微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：在微处理器内部、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：逻辑地址由微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 微处理器有哪 3 类逻辑段，各种逻辑段分别是什么用途？

〔解答〕

IA-32 微处理器有代码段、数据段、堆栈段 3 类逻辑段。

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）什么是实方式、保护方式和虚拟 8086 方式？

（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。

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。

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”。

（习题 2-11）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 微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 2-12）在 MASM 6.15 软件包中有一个 QH.BAT 文件，执行一次，看看有什么内容。学习过程中，有时不妨抽空阅读一下。这些是原始资料，参考价值很高。

② 首先判断教材习题 3-18 中每条指令执行后 AX 的数值，然后将其编辑成为一个完整的汇编语言源程序，接着进行汇编连接、生成列表文件和可执行文件，最后利用列表文件核对你判断的 AX 数值。你也可以利用本书配套的 I/O 子程序库，在每条 MOV 指令之后，调用其中的 DISPUIW 子程序，以十进制形式显示 AX 的数值。

3.5 习题解答

〔习题 3-1〕简答题（用一句话回答）

① 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？
② 汇编语言的标识符大小写不敏感意味着什么？
③ 语句.STARTUP 除说明程序开始位置外，还包括一个什么关键作用？
④ 汇编语言源程序文件中，END 语句后的语句会被汇编吗？
⑤ 使用二进制 8 位表达无符号整数，257 有对应的编码吗？
⑥ DOS 的 9 号功能调用可以显示“$”符号吗？
⑦ 字符'F'和数值 46H 作为 MOV 指令的源操作数有区别吗？
⑧ 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？
⑨ 数值 500 能够作为字节变量的初值吗？
⑩ 标准 DOS 程序按什么顺序排列代码段、数据段和堆栈段？

〔解答〕

① 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没有本质的区别，要求类似。
② 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。
③ 语句.STARTUP 除说明程序开始位置外，还将 DS 指向当前程序设置的数据段。
④ END 语句后的语句不会被汇编。
⑤ 使用二进制 8 位表达无符号整数，257 没有对应的编码。
⑥ 因 DOS 的 9 号功能调用用“$”符号作为显示字符串结束标志，所以不能显示“$”符号。
⑦ 字符'F'的 ASCII 码就是数值 46H，所以没有区别。
⑧ 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。
⑨ 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。
⑩ 标准 DOS 程序从低地址到高地址依次安排代码段、数据段和堆栈段。

〔习题 3-2〕判断题（判断如下论述是正确还是错误）

① 微处理器的传送指令 MOV 属于汇编语言的执行性语句。
② 在 MASM 中可以用 EDI 作为标号或变量名。
③ 汇编语言的语句由明显的 4 部分组成，不需要分隔符区别。
④ MASM 汇编语言的注释用分号开始，但不能用中文分号。
⑤ 使用 32 位指令编写实方式和虚拟 8086 方式的 MASM 程序，应该将处理器选择伪指令书写在存储模式语句之后。
⑥ 汇编结束 END 表明程序执行到此结束。
⑦ 汇编语言不区分大小写，所以执行 ML.EXE 生成列表文件的参数/Fl，也可以表达成/fL。
⑧ 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。
⑨ 常用的 BCD 码为 8421 BCD 码，其中的 8 表示 D3 位的权重。
⑩ 汇编语言要求，所有字符串定义都必须用“$”符号作为结尾。

〔解答〕

① 对　② 错　③ 错　④ 对　⑤ 对
⑥ 错　⑦ 错　⑧ 对　⑨ 对　⑩ 错

〔习题 3-3〕填空题（填写数字或文字）

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 创建数据段可以使用_____语句，这时可以用_____预定义符号表达数据段名。

③ 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，用 TINY 模式连接后生成_____形式可执行文件，其他存储模式生成 EXE 文件。

④ 计算机中有一个 01100001 编码。如果把它认为是无符号数，它是十进制数_____；如果认为它是 BCD 码，则表示真值_____；如果它是某个 ASCII 码，则代表字符_____。

⑤ 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。

⑥ 真值－1，如果其补码用字节量表达是_____，如果用 16 位补码表示，则为_____。

⑦ 用 DD 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。

⑧ 程序中需要使用 80486 的指令，则源程序应该用_____伪指令说明。

⑨ 数据段有语句 ABC DB 1,2,3，代码段指令 MOV CL, ABC+2 执行后，CL＝_____。

⑩ 数据段有语句 H8843 DD 99008843H，代码段指令 MOV CX, WORD PTR H8843 执行后，CX＝_____。

〔解答〕

① DB，OFFSET
② .DATA，@DATA
③ ASM，目标模块，COM
④ 97，61，小写字母 a
⑤ 0DH（13），0AH
⑥ FFH，FFFFH
⑦ DWORD，4，WORD PTR XYZ
⑧ .486.（486P）
⑨ 3
⑩ 8843H

〔习题 3-4〕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 3-5〕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如下 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 3-6〕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 3-7〕实现例题 3-1，再创建一个 COM 程序完成例题 3-1 的功能。

〔解答〕

要创建 COM 类型程序，需要定义程序采用 TINY 存储模式。要将程序模板中.STACK 和.DATA 语句去掉，并将数据定义填到子程序之后（END 之前）就形成了一个 COM 类型程序的模板文件。

```
;ex307.asm for COM program
            .model tiny
            .486
            .code
            .startup
            mov dx,offset msg
            mov ah,9
            int 21h
            .exit
msg         db 'Hello, Assembly !',13,10,'$'
            end
```

〔习题 3-8〕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 3-9〕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 3-10〕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 3-11〕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 3-12〕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 3-13〕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：61H～7AH。
ASCII 码 0DH 和 0AH 分别对应的是回车和换行字符。

〔习题 3-14〕说明采用小端方式如何存放多字节数据。

〔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。

〔习题 3-15〕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〕

```
① my1b   db 'Personal Computer'
② my2b   db 20
③ my3b   db 14h
④ my4b   db 00010100b
⑤ my5w   dw 20 dup(?)
⑥ my6c   = 100
```

⑦ my7c equ <Personal Computer>

〔习题 3-16〕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    db −10,2,5,4,?
```

〔习题 3-17〕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。

```
db 'ABC',10,10h,'EF',3 dup(−1,?,3 dup(4))
dw 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 3-18〕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 AX 的内容。

```
            ;数据段
            org 100h
varw        dw 1234h,5678h
varb        db 3,4
vard        dd 12345678h
buff        db 10 dup(?)
mess        db 'hello'
            ;代码段
            mov ax,offset mess
            mov ax,type buff+type mess+type vard
            mov ax,sizeof varw+sizeof buff+sizeof mess
            mov ax,lengthof varw+lengthof vard
```

〔解答〕

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

〔习题 3-19〕使用 MLL.BAT 和 MAKE.BAT 文件简化你的汇编操作过程。打开这两个文本文件看看，前者会帮助你生成基本的列表文件，后者方便你生成带有 CodeView 调试程序的文件。它们的使用很简单：键入 MLL 或 MAKE，空格，后跟要汇编连接的文件名就可以了，不需要输入扩展名。

母如何转换为小写字母。

上机任务 3：① 掌握常用指令，上机实现教材例题 4-7～例题 4-15 程序（至少实践例题 4-9 和例题 4-12）。

② 首先完成教材习题 4-13 或习题 4-15 要求，写出你的结果。然后将其编辑成为一个完整的汇编语言源程序，汇编连接、生成可执行文件。最后在 CodeView 调试程序环境，单步执行每条指令，核对结果。也可以利用本书配套的 I/O 子程序库，在每条指令之后，调用其中的 DISPRF，显示 6 个状态标志位的结果。

③ 将教材习题 4-20 要求编写的程序上机实现。

上机任务 4：① 熟悉串操作指令，上机实现教材例题 4-16～例题 4-18 程序（至少实践例题 4-16）。

② 上机实现教材习题 4-22 要求编写的程序。

4.5 习题解答

〔习题 4-1〕简答题（用一句话回答）

① 为什么将查找操作数的方法称为数据寻“址”方式？

② 为什么说直接引用变量名表示的操作数是直接寻址方式？

③ 为什么说 XCHG EDX,CX 是一条错误的指令？

④ 都是获取偏移地址，为什么指令 LEA EBX, [ESI]是正确的，而指令 MOV EBX, OFFSET[ESI]就是错误的？

⑤ 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？

⑥ 大小写字母转换使用了什么规律？

⑦ 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？

⑧ 指令 SHL AX,4 为什么不能在 8086 和 8088 微处理器上执行？

⑨ 逻辑与运算为什么也称为逻辑乘？

⑩ 逻辑运算指令怎么实现复位、置位和求反功能？

〔解答〕

① 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。

② 因指令中直接引用变量名作操作数，其指向的偏移地址是直接寻址的有效地址。

③ 源、目标寄存器位数不同，不能用该指令进行数据交换。

④ 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。

⑤ INC、DEC、NEG 和 NOT 指令的操作数既是源操作数也是目的操作数。

⑥ 大小写字母转换利用它们的 ASCII 码相差 20H。

⑦ 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。

⑧ 在 8086 和 8088 微处理器上指令 SHL 中表示移位位数的立即数只能为 1。

⑨ 逻辑与运算规则类似二进制的乘法，所以称其为逻辑乘。

⑩ 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 4-2〕判断题（判断如下论述是正确还是错误）

① 立即数寻址方式的立即数保存在通用寄存器中。
② 存储器寻址方式的操作数肯定在主存。
③ 有效地址 EA 就是存储器操作数的偏移地址。
④ 2 号 DOS 功能（INT 21H）需要入口参数，AL＝显示字符的 ASCII 码。
⑤ 空操作 NOP 指令其实根本没有指令。
⑥ 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。
⑦ 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。
⑧ 指令 INC ECX 和 ADD ECX,1 的实现功能完全一样，可以互相替换。
⑨ 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。
⑩ 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。

〔解答〕

① 错　② 对　③ 对　④ 错　⑤ 错
⑥ 错　⑦ 对　⑧ 错　⑨ 错　⑩ 对

〔习题 4-3〕填空题（填写数字或文字）

① 从列表文件中，能发现例题 4-1 最后一条指令执行完，EBP＝_____。
② 将例题 4-2 的最后一条错误指令按其含义改正，可以是指令_____。
③ 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。
④ 例题 4-3 最后一条指令的源操作数寻址方式是_____。
⑤ 虽然有 8 个 16 位通用寄存器，但在 16 位存储器寻址方式中只能使用其中 4 个，它们是 BP、_____、_____和_____。
⑥ 参见例题 4-4 和 4-5，为了方便逐个处理数组中的元素，可以使用寄存器间接寻址，例如将寄存器赋值数组的偏移地址；也常用寄存器相对寻址，此时寄存器通常赋值为_____。
⑦ 例题 4-8 的 TAB 定义如果是 1234567890，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 4-15 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 00000800H
② mov edi,esi（mov di,si）
③ DS，SS
④ 立即数寻址
⑤ BX，SI，DI
⑥ 0
⑦ 78894111
⑧ 0，0，0
⑨ 0123456788765432H，83H
⑩ 4

〔习题4-4〕给出IA-32微处理器的32位寻址方式和16位寻址方式的组成公式，并说明各部分的作用。

〔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4-5〕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[bx]
⑦ mov edx,[bx+di]
⑧ mov edx,[bx+di+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4-6〕如果例题4-6的数组元素都是16位，则程序应该如何修改？

〔解答〕

```
        ; 代码段修改如下：
        mov ecx,lengthof array1
        mov ebx,0
again:  mov ax,array1[ebx*2]
        add ax,8000h
        mov array2[ebx*2],ax
        add ebx,1
        loop again
```

〔习题 4-7〕在 IA-32 微处理器的实方式下，假设当前 DS＝2000H，BX＝0100H，SI＝0002H，物理地址 [20100H]～[20103H]的存储单元依次存放 12H、34H、56H 和 78H，[21200H]～[21203H]依次存放 2AH、4CH、B7H 和 65H。在上述条件下，独立执行下列每条指令后，AX 寄存器的内容是什么？

① mov ax,1200h
② mov ax,bx
③ mov ax,ds:[1200h]
④ mov ax,[bx]
⑤ mov ax,[bx+1100h]
⑥ mov ax,[bx+si]
⑦ mov ax,[bx][si+1100h]

〔解答〕

① 1200H
② 0100H
③ 4C2AH
④ 3412H
⑤ 4C2AH
⑥ 7856H
⑦ 65B7H

〔习题 4-8〕说明下面各条指令的具体错误原因。

① mov ecx,dx
② mov al,300
③ mov 20h,ah
④ mov es,1234h
⑤ mov es,ds

〔解答〕

① 类型不匹配
② 立即数超出寄存器范围
③ 立即数不能作目的操作数
④ 立即数不能传送给段寄存器
⑤ 段寄存器不能传送给段寄存器

〔习题 4-9〕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   db 18h,34h,05h,06h,09h,0ah,0ch,11h,12h,14h
        ;代码段
        mov ebx,offset table
        mov al,8
        xlat
```

为了验证读者自己的判断，不妨使用本教材的输入输出子程序库 IO.LIB 提供的子程序 DISPHB 显示换码后 AL 的值。

〔解答〕

```
        ;数据段
table   db 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）
```

〔习题 4-10〕假设当前 ESP＝00B0H，说明下面每条指令后，ESP 等于多少？

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

〔解答〕

ESP＝00AEH
ESP＝00AAH
ESP＝00A6H
ESP＝00AAH
ESP＝00ACH
ESP＝00B0H

〔习题 4-11〕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 BX 和位移量 0B2H 的寄存器相对寻址方式把存储器中的一个字和 CX 寄存器的内容相加，并把结果送回存储器中。

④ 将变量 VARW 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 4-12〕编写程序，用 32 位寄存器和 32 位加法指令实现例题 4-11。

〔解答〕

```
        ; 代码段修改如下
        mov ecx,count
        mov ebx,0
again:  mov eax,dvar1[ebx*4]
        add eax,dvar2[ebx*4]
        mov dvar3[ebx*4],eax
        inc bx
        loop again
```

〔习题 4-13〕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 4-14〕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 dq 67883000h
buffer2 dq 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 4-15〕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 4-16〕例题 4-14 将 AX 内容乘以 10，为什么先设置 EAX 等于 0?

〔解答〕

因为 AX 内容乘以 10 后结果可能超过 16 位、高位应该进入 EAX 高 16 位；所以，数据进入 AX 之前先设置 EAX 为 0，即将高 16 位清 0，再进行移位操作，才能得到正确的运算结果。

〔习题 4-17〕第 3 章开始有一个模板源程序 EXAMPLE.ASM，比较一下有和无处理器选择伪指令.486 产生的列表文件有什么不同，注意带上/Sg 参数。

〔解答〕

有.486 语句，移位指令是：SHL BX,04。

无.486 语句，移位指令是：共 4 条 SHL BX,01。

〔习题 4-18〕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 4-19〕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 4-20〕编程将一个压缩 BCD 码变量（例如 92H）转换为对应的 ASCII 码，然后用 2 号 DOS 功能调用显示。

〔解答〕

```
        ; 数据段
bcd     db 92h
        ; 代码段
        mov dl,bcd
        shr dl,4
        add dl,30h
        mov ah,2
```

```
int 21h
mov dl,bcd
and dl,0fh
add dl,30h
mov ah,2
int 21h
```

〔习题 4-21〕以 MOVS 指令为例，说明串操作指令的寻址特点。

〔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，大多数情况下 DF＝0，进行增量；有时设置 DF＝1，进行减量。

〔习题 4-22〕参考例题 4-17，编程将 DOS 屏幕内容向上滚动 2 行。

〔解答〕

```
; 代码段
cld
mov dx,0b800h
mov es,dx
mov edi,0
mov ds,dx
mov esi,2*2*80
mov ecx,23*80
rep movsw
;
mov edi,23*2*80
mov ax,0761h
mov ecx,2*80
rep stosw
```

〔习题 4-23〕参考例题 4-17，编程将 DOS 屏幕内容向下滚动 1 行。本习题需要从第 24 行最后一列的字符开始传送，所以需要使用 STD 指令设置 DF＝1，向地址减小的方向进行操作。

〔解答〕

```
; 代码段
std
mov dx,0b800h
mov es,dx
mov edi,25*2*80-2
mov ds,dx
mov esi,24*2*80-2
mov ecx,24*80
```

```
rep movsw
;
mov edi,2*80-2
mov ax,0761h
mov ecx,80
rep stosw
.exit
end
```

〔习题4-24〕说明如下程序的功能，执行结束显示什么。其中JE指令是条件分支指令（Jump if Equal），在相等（ZF=1）时，条件成立，转移到标号指定的指令；否则顺序执行下条指令。

```
                        ;数据段
0000 53 65 61 72 63 68  string    db 'Searching for a 20H'
     69 6E 67 20 66 6F
     72 20 61 20 32 30
     48
= 0013                  count     equ sizeof string
                        ;代码段
0010 FC                           cld
0011 8C D                         mov ax,ds
0013 8E C0                        mov es,ax
0015 66| BF 00000000 R            mov edi,offset string
001B B0 20                        mov al,' '  ;空格字符（ASCII码为20H）
001D 66| B9 00000013              mov ecx,count
0023 F2/ AE                       repnz scasb
0025 74 04                        je found
0027 B2 4E                        mov dl,'N'
0029 EB 02                        jmp done
002B B2 59              found:    mov dl,'Y'
002D B4 02              done:     mov ah,2
002F CD 21                        int 21h
```

〔解答〕

程序判断字符串中是否有空格字符，有则显示Y，无则显示N。本习题程序将显示Y。

〔习题4-25〕通过阅读附录A，熟悉CodeView调试程序的各个窗口功能，并进行必要的选项设置。然后，并按照教材指导尝试调试例题4-3。然后参照例题4-3的调试过程，在CodeView中调试例题4-4。注意观察大写字母组成的字符串是否转换为小写字母组成的字符串。

编和源文件包含方法实现。

② 理解子程序库方法，将例题 5-22 按照子程序库的要求生成最终可执行文件（即习题 5-30 要求）。

上机任务 5：① 参考习题 5-31 说明，熟悉本书配套使用的 I/O 子程序库，以字节、字和双字数据归类子程序，或者按照二进制、十进制、十六进制数据归类子程序，并组成项目组。

② 以项目组为单位，编写子程序，形成子程序模块。

③ 项目组长负责协调各个项目组，并编写主程序，最终生成完整的子程序库。项目组长也可以安排一个测试组，编写主程序和验证各个项目组的子程序。

5.4　习题解答

〔**习题 5-1**〕简答题（用一句话回答）

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数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 宏定义体的标号为什么还要特别声明一下？
⑨ 子程序模块与子程序库是什么关系？
⑩ INCLUDE 语句和 INCLUDELIB 有什么区别？

〔**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。
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就可能致使堆栈溢出。
⑦ “传值”是传递参数的一个副本，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 宏定义体中的标号必须用 LOCAL 伪指令声明为局部标号，否则多次宏调用将出现标号的重复定义语法错误。
⑨ 子程序模块是将子程序单独编写成一个源程序文件，经过汇编之后形成目标模块 OBJ 文件，使子程序通用和得以复用。子程序库是子程序模块的集合，存放着各子程序的名称、目标代码以

及有关定位信息等。

⑩ 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。

〔习题 5-2〕判断题（判断如下论述是正确还是错误）

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 段间转移的原位置和目的位置之间一定超出 64KB 存储空间。
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 LOOP 循环指令的目标地址只能是在同一个代码段，且不超过－128～127 之间的距离。
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 进行模块连接是将多个可执行文件（例如 EXE）连接成一个可执行文件。

〔解答〕

① 对　② 错　③ 错　④ 对　⑤ 错
⑥ 对　⑦ 对　⑧ 错　⑨ 对　⑩ 错

〔习题 5-3〕填空题（填写数字或文字）

① JMP 指令根据目标地址的转移范围和寻址方式，可以分成 4 种类型：段内转移、_____，段内转移、_____段间转移、_____，段间转移、_____。
② 假设在实方式下，DS＝2000H，BX＝1256H，字变量 TABLE 的偏移地址是 20A1H，物理地址 232F7H 处存放 3280H，执行指令 JMP BX 后 IP＝_____，执行指令 JMP TABLE[BX]后 IP＝_____。
③ CMP AX,3721H 指令之后是 JZ 指令，发生转移的条件是 AX＝_____，此时 ZF＝_____。
④ 例题 5-9 运行结果显示的信息是_____。如果将 DVAR1 的数据改为 123456789，显示结果会是_____。
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 过程定义开始是 TEST PROC 语句，则过程定义结束的语句是_____。
⑦ 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 宏定义中使用的标号要求用_____伪指令声明一下，子程序就不需要。
⑩ 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。

〔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 ERROR ! Overflow!，Correct !
⑤ 循环初始，循环控制
⑥ TEST ENDP
⑦ ENDM
⑧ EBP
⑨ LOCAL

⑩ PUBLIC，EXTERN

〔习题 5-4〕说明下列 3 个条件转移指令发生跳转的条件：

```
xor ax,1e1eh
je equal              ; ①
test al,10000001b
jnz there             ; ②
cmp cx,64h
jb there              ; ③
```

〔解答〕

① AX＝1E1EH（异或后为 0）。

② AL 的 D0 或 D7 至少有 1 位为 1。

③ CX（无符号数）＜ 64H。

〔习题 5-5〕用 JS 指令进行条件判断实现分支，完成例题 5-2 程序功能。

〔解答〕

```
          ; 代码段
          mov al,bvar     ;取出要判断的数据
          cmp al,0        ;与 0 比较
          js next4
          mov ah,0
          jmp done        ;无条件转移，跳过另一个分支
next4:    mov ah,0ffh
done:
```

〔习题 5-6〕8086 支持另一条符号扩展指令 CWD，它的功能是：如果 AX 最高位（符号位）为 0，则设置 DX＝0；如果 AX 最高位为 1，则设置 DX＝FFFFH。现在模仿例题 5-2 用一段程序实现其功能。

〔解答〕

```
          ; 数据段
wvar      db 8356h            ;假设一个数据
          ; 代码段
          mov ax,wvar         ;取出要判断的数据
          test ax,8000h       ;测试最高位
          jz next1            ;最高位为 0（ZF＝1），转移到标号 NEXT1
          mov dx,0ffffh       ;最高位为 1，顺序执行：设置 DX＝FFFFH
          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 DX＝0
done:
```

〔习题 5-7〕参照例题 5-3，实现将待发送数据加上偶校验位的程序。

〔解答〕

只需将例题 5-3 程序中的 JNP 指令改为 JP 即可。

〔习题 5-8〕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，否则就出现传输错误。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   db 57h              ;保存接收的数据
error   db 'Error !$'
        ; 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 ah,9            ;个数为奇数，显示出错
        mov dx,offset error
        int 21h
done:
```

〔习题 5-9〕编写一个程序段，在 DX 高 4 位全为 0 时，使 AX＝0；否则使 AX＝－1。

〔解答〕

```
        ; 代码段
        test dx,0f000h  ;test dh,0f0h
        jz next         ;jnz next
        mov ax,-1       ;mov ax,0
        jmp again
next:   mov ax,0        ;mov ax,0ffffh
again:  ...
```

〔习题 5-10〕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    dd 57h
        ; 代码段
        mov eax,dvar
        test eax,80000000h
        jnz nextl
        test eax,1
        jnz nextr
        mov dl,'M'
        jmp done
nextl:  mov dl,'L'
        jmp done
nextr:  mov dl,'R'
done:   mov ah,2
        int 21h
```

〔习题 5-11〕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；如果不是输入了 0～9 数字，就提示错误“Error!”，继续等待输入数字。

〔解答〕

```
            ; 数据段
inmsg       db 'Input number(0~9): ','$'
crlf        db 0dh,0ah,'$'
ermsg       db 0dh,0ah,'Error !$'
            ; 代码段
            mov dx,offset inmsg        ; 提示输入数字
            mov ah,9
            int 21h
again:      mov ah,1                   ; 等待按键
            int 21h
            cmp al,'0'                 ; 数字 <0?
            jb erdisp
            cmp al,'9'                 ; 数字 >9?
            ja erdisp
            push ax
            mov dx,offset crlf
            mov ah,9
            int 21h
            pop dx
            mov ah,2
            int 21h
            jmp done
erdisp:     mov dx,offset ermsg
            mov ah,9
            int 21h
            jmp again
done:
```

〔习题 5-12〕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，不关心进位和溢出。

〔习题 5-13〕编程求主存 0040H：0 开始的一个 64KB 物理段中共有多少个 NOP 指令（其代码是 90H），结果保存在 EAX 寄存器中。

〔解答〕

```
        ; 代码段
        mov dx,40h
        mov ds,dx
        mov ebx,0
        mov eax,ebx
        mov ecx,10000h
again:  cmp byte ptr [ebx],90h
        jnz next
        inc eax
next:   inc ebx
        loop again
```

〔习题 5-14〕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   dw 2005,2008,98 dup (1394); 假设 100 个 16 位正整数
wordsum dw ?
error   db 'Overflow !$'
        ; 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 dx,offset error
        mov ah,9
        int 21h
```

〔习题 5-15〕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  db 'If you find any error in the program, you can DEBUG it.'
count   = sizeof string
bug     db '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 dl,'Y'
        jmp L3
L2:     pop edi
        inc edi
        loop L1
        mov dl,'N'
L3:     mov ah,2
        int 21h
```

〔**习题 5-16**〕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     dd 00371002h
        ; 代码段
        mov esi,bcd
        cmp esi,0
        jnz goon
        mov dl,'0'
        mov ah,2
        int 21h
        jmp done
goon:   mov ecx,8
        xor ebx,ebx     ; EBX=0，表示可能是前导 0
again:  rol esi,4
        mov edx,esi
        and edx,0fh     ; EDX 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 edx,0
        jz next         ; EDX=0，说明是前导 0，不显示
        mov ebx,1       ; EDX≠0，没有前导 0 了，令 EBX=1≠0
disp:   add dl,30h
        mov ah,2
```

```
        int 21h
next:   loop again
done:
```

〔习题 5-17〕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  db 'Let us have   a try !',0dh,0ah,'$'
        ; 代码段
        mov ecx,sizeof string
        cmp ecx,2
        jb done
        mov ah,9                ; 显示处理前的字符串
        lea dx,string
        int 21h
        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
        mov ah,9                ; 显示处理后的字符串
        lea dx,string
        int 21h
done:
```

〔习题 5-18〕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 5-19〕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 dl,ah
        mov ah,2
        int 21h
        pop dx
        mov ah,2
        int 21h
        ret
disp99  endp
```

〔习题 5-20〕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:  mov ah,1
        int 21h
        cmp al,1bh
        jz done
        push ax
        mov ah,2
        mov dl,':'
        int 21h
        pop ax
        push ax
        rol al,4
        call htoasc      ; 调用子程序
        mov ah,2
        mov dl,al
```

```
        int 21h          ; 显示一个字符
        pop ax
        call htoasc      ; 调用子程序
        mov ah,2
        mov dl,al
        int 21h          ; 显示一个字符
        mov ah,2
        mov dl,13
        int 21h
        mov dl,10
        int 21h
        jmp again
done:
```

〔习题 5-21〕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 ASCII 码值
        mov dl,al
        mov ah,06h           ; 以二进制的形式显示
        int 21h
        pop eax
        loop dbd
        pop edx
        pop ecx
        ret
dispbd  endp
```

〔习题 5-22〕编写子程序，以二进制形式输入一个 8 位数据，并设计一个主程序验证。

〔解答〕

```
        ; 数据段
bvar    db ?
        ; 代码段，主程序
        call readbb          ; 调用子程序
```

```
            mov bvar,al
            ; 代码段，子程序
readbb      proc            ;8 位二进制数的输入
            push bx
            push cx
            push dx
            mov  bx,0       ;BX 用于存放二进制结果
            mov  cx,8       ; 限制输入字符的个数
rbb:        mov ah,06h      ; 从键盘读入字符
            mov dl,0ffh     ; 不回显
            int 21h
            jz rbb          ; 检测是否有键按下
            cmp al,'0'      ; 检测输入字符是否合法
            jb rbb           不合法则返回重新输入
            cmp al,'1'
            ja rbb
            mov ah,06h      ; 显示所输入的字符
            mov dl,al
            int 21h
            ;
            sub al,'0'      ; 对输入的字符进行转化
            shl bl,1        ;BL 的值乘以 2
            add bl,al       ;BL 和 AL 相加
            loop rbb        ; 循环输入字符
            ;
            mov al,bl       ; 把 BL 的二进制结果存放入 AL 中
            pop dx
            pop cx
            pop bx
            ret
readbb      endp
```

（**习题 5-23**）例题 5-18 用 32 位寄存器可以实现 32 位数据输入，如果只要求输入-2^{15}～$+2^{15}-1$之间的数据，请用 16 位寄存器实现该程序。

（**解答**）

```
            ; 数据段
count       = 10
array       dw count dup(0)
            ; 代码段，主程序
            mov cx,count
            mov bx,offset array
again:      call readsiw    ; 调用子程序，输入一个数据
            mov [bx],ax     ; 将出口参数存放到数据缓冲区
            inc bx
            inc bx
```

```
        int 21h
        pop dx
        pop ax
        ret
dpcrlf  endp
```

〔习题 5-24〕将例题 5-19 的 32 位寄存器改用 16 位寄存器，实现输出-2^{15}～$+2^{15}-1$之间的数据。

〔解答〕

```
        ; 数据段
array   dw 1234,-1234,0,1,-1,32767,-32768,5678,-5678,9000
wtemp   dw ?
        ; 代码段，主程序
        mov cx,lengthof array
        mov bx,offset array
again:  mov ax,[bx]
        mov wtemp,ax            ; 将入口参数存放到共享变量
        call dispsiw            ; 调用子程序，显示一个数据
        inc bx
        inc bx
        call dpcrlf             ; 光标回车换行以便显示下一个数据
        loop again
        ; 代码段，子程序
dispsiw proc                    ; 显示有符号十进制数的通用子程序
        push ax                 ; 出口参数：共享变量 wtemp
        push bx
        push dx
        mov ax,wtemp            ; 取出显示数据
        test ax,ax              ; 判断数据是零、正数或负数
        jnz dsiw1
        mov dl,'0'              ; 是零，显示“0”后退出
        mov ah,2
        int 21h
        jmp dsiw5
dsiw1:  jns dsiw2               ; 是负数，显示“-”
        mov bx,ax               ; AX 数据暂存于 BX
        mov dl,'-'
        mov ah,2
        int 21h
        mov ax,bx
        neg ax                  ; 数据求补（绝对值）
dsiw2:  mov bx,10
        push bx                 ; 10 压入堆栈，作为退出标志
dsiw3:  cmp ax,0                ; 数据（余数）为零，转向显示
        jz dsiw4
        sub dx,dx               ; 扩展被除数 DX.AX
```

```
            div bx              ; 数据除以 10：DX.AX÷10
            add dl,30h          ; 余数（0~9）转换为 ASCII 码
            push dx             ; 数据各位先低位后高位压入堆栈
            jmp dsiw3
dsiw4:      pop dx              ; 数据各位先高位后低位弹出堆栈
            cmp dl,10           ; 是结束标志 10，则退出
            je dsiw5
            mov ah,2            ; 进行显示
            int 2!h
            jmp dsiw4
dsiw5:      pop dx
            pop bx
            pop ax
            ret                 ; 子程序返回
dispsiw     endp

dpcrlf      proc                ; 使光标回车换行的子程序（同上一个习题）
```

〔习题 5-25〕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。传递参数方法自定。

〔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 5-26〕编制 3 个子程序把一个 16 位二进制数用 4 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 AX 寄存器传递这个 16 位二进制数。
② 采用 temp 变量传递这个 16 位二进制数。
③ 采用堆栈方法传递这个 16 位二进制数。

〔解答〕

① 采用 AX 寄存器传递这个 16 位二进制数。

```
            ; 数据段
wvar        dw 81AFH
            ; 代码段，主程序
```

```
        mov ax,wvar
        call disphw
        ; 代码段，子程序
disphw  proc
        push bx
        push cx
        mov cx,4            ;4 位
dhw1:   rol ax,4
        push ax
        and al,0fh          ; 转换为 ASCII 码
        add al,30h
        cmp al,'9'
        jbe dhw2
        add al,7
dhw2:   mov ah,2
        mov dl,al
        int 21h             ; 显示一个字符
        pop ax
        loop dhw1
        pop cx
        pop bx
        ret
disphw  endp
```

② 采用 temp 变量传递这个 16 位二进制数。

```
        ; 数据段
wvar    dw 81AFH
temp    dw ?
        ; 代码段，主程序
        mov ax,wvar
        mov temp,ax
        call disphw
        ; 代码段，子程序
disphw  proc
        push bx
        push cx
        mov cx,4            ;4 位
        mov ax,temp
dhw1:   rol ax,4
        push ax
        and al,0fh          ; 转换为 ASCII 码
        add al,30h
        cmp al,'9'
        jbe dhw2
        add al,7
```

```
dhw2:   mov ah,2
        mov dl,al
        int 21h              ; 显示一个字符
        pop ax
        loop dhw1
        pop cx
        pop bx
        ret
disphw  endp
```

③ 采用堆栈方法传递这个 16 位二进制数。

```
        ; 数据段
wvar    dw 81AFH
        ; 代码段，主程序
        push wvar
        call disphw
        add esp,2
        ; 代码段，子程序
disphw  proc
        push ebp
        mov ebp,esp
        push bx
        push cx
        mov cx,4             ; 4 位
        mov ax,[ebp+6]
dhw1:   rol ax,4
        push ax
        and al,0fh           ; 转换为 ASCII 码
        add al,30h
        cmp al,'9'
        jbe dhw2
        add al,7
dhw2:   mov ah,2
        mov dl,al
        int 21h              ; 显示一个字符
        pop ax
        loop dhw1
        pop cx
        pop bx
        pop ebp
        ret
disphw  endp
```

〔习题 5-27〕区别如下概念：宏定义、宏调用、宏名、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。

宏名：宏定义的名称，是符合语法的标识符，同一源程序中该名字定义唯一。

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 5-28〕利用 HTOASC 子程序，将例题 5-21 输入的 4 位十六进制数，再以十六进制形式输出。

〔解答〕

在代码段程序后接着编写如下：

```
        dispmsg crlf
        mov ax,wvar
        mov ecx,4       ;4 位
dhw1:   rol ax,4
        push ax
        call htoasc     ;转换为 ASCII 码
        mov ah,2
        mov dl,al
        int 21h         ;显示一个字符
        pop ax
        loop dhw1
```

〔习题 5-29〕宏和子程序有什么本质的区别。

〔解答〕

宏调用在汇编时将相应的宏定义语句复制到宏指令的位置，没有减少汇编后的目标代码，执行时不存在控制的转移与返回。执行速度没有改变。

子程序调用在执行时由主程序调用指令 CALL 实现，控制转移到子程序，子程序需要执行返回指令 RET 将控制再转移到主程序，程序没有被复制，汇编后的目标代码较短。但多次的控制转移以及子程序中寄存器保护、恢复等操作，会影响程序的执行速度。

〔习题 5-30〕将例题 5-22 按照子程序库的要求生成最终可执行文件。

〔习题 5-31〕本书利用子程序库方法为读者提供了常用的输入输出子程序，包含文件是 IO.INC，子程序库文件是 IO.LIB。子程序库中的主要子程序已经在本章例题中介绍，习题 5-21～5-24 也是其中一部分。作为一个工程项目，请读者参照附录功能说明，自己建立这个子程序库文件。

文件 IO.INC 包含有显示字符串的宏 DISPMSG，还有各个子程序的声明，例如对十六进制数的输入输出部分如下：

```
;declare procedures for inputting and outputting hexadecimal number
        extrn readhb:near,readhw:near,readhd:near
        extrn disphb:near,disphw:near,disphd:near
```

另外，读者也可以编写一个主程序（IOMAIN.ASM）来验证各个子程序。例如，验证十六进制数输入输出字量的部分程序如下：

```
        ;数据段
```

```
varw    dw ?
msghw   db 13,10,'Please input a hexadecimal number (up to 4 digits):',13,10,'$'
crlf    db 13,10,'$'
        ;代码段
        dispmsg msghw
        call readhw          ;AX=result
        mov varw,ax
        inc varw
        mov ax,varw
        dispmsg crlf
        call disphw
```

（解答）

教材中本章例题和习题包含了主要的子程序，其他可以参考附录 B。

存取操作时，需要在 T3 状态结束时插入若干个等待周期 Tw。插入 Tw 的多少取决于存储器或外设的工作速度。当存储器或外设准备好后，应使 READY 信号变为高电平，微处理器在 Tw 状态的前沿采样 READY 信号的高电平后，会结束 Tw 状态进入 T4 状态。

〔分析〕

一般说来，微处理器的运行速度是很高的，存储器和 I/O 接口往往难以满足高速的要求，故 Tw 的等待状态设置是必要的。正是为了能提供这种设置，微处理器一般都设置有相应的引脚，如 8086 的 READY 引脚。

〔例题 6-2〕说明 8086 空闲状态和等待状态的区别。

〔解答〕

8086 CPU 的空闲状态是指总线接口部件（BIU）既不从存储器取指令或存取数据，也不与 I/O 端口传送数据。此时 BIU 执行一系列的 Ti 状态，即空闲状态，CPU 只进行内部操作。等待状态 Tw 是指 CPU 在读/写存储器或 I/O 端口时，若存储器或 I/O 端口没有“准备就绪”，即在总线周期 T3 状态前沿检测 READY 信号为低电平时，就在下一个时钟周期插入一个 Tw 状态。等待状态用以延长总线周期，等待存储器或 I/O 端口准备就绪。

〔分析〕

本题旨在让初学者搞清空闲状态和等待状态的区别。通过以上的解答，我们可以说，总线上处于空闲状态时，CPU 的内部并不空闲；而总线上处于等待状态时，CPU 的内部则有可能是空闲状态。

6.3　习题解答

〔习题 6-1〕简答题（用一句话回答）

① 为什么称微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 什么是总线仲裁？
⑦ 异步时序为什么可以没有总线时钟信号？
⑧ 32 位 PC 机为什么采用多总线结构，而不是单总线结构？
⑨ ISA 总线为什么被设计成前 62 引脚和后 36 引脚的两个插槽？
⑩ 什么是微软宣称的即插即用技术？

〔解答〕

① 数据总线承担着微处理器与存储器、外设之间的数据交换，既可以输入也可以输出，故称其是双向的。
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 微处理器的运行速度远远快于存储器和 I/O 端口。微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 猝发传送是微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 对连接在总线上的多个需要控制总线的主设备，进行服务仲裁，称总线仲裁。

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微处理器芯片总线，以接近微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 ISA 总线的后 36 个引脚是增强原 62 引脚功能而增加的，被设计成前 62 引脚和后 36 引脚两个插槽，就可以插入 8 位接口电路卡也可插入 16 位接口电路卡。

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 6-2**〕判断题（判断如下论述是正确还是错误）

① 低电平有效是指信号为低电平时表示信号的功能。

② 微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，表明微处理器准备好交换数据了。

④ 中断响应总线周期只会发生在微处理器响应可屏蔽中断时。

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 总线独立于微处理器，所以其引脚信号多数并不与 IA-32 微处理器对应。

⑨ ISA 总线具有自动配置能力。

⑩ PCI 总线的 I/O 读写总线周期支持猝发传送。

〔**解答**〕

① 对　② 对　③ 错　④ 对　⑤ 对

⑥ 错　⑦ 对　⑧ 对　⑨ 错　⑩ 错

〔**习题 6-3**〕填空题（填写数字或文字）

① 某个微处理器具有 16 个地址总线，通常用 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 微处理器执行指令 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。如果微处理器的时钟频率为 100MHz，则每个 T 状态的持续时间为_____。

⑤ 8086 微处理器进行 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 微处理器与存储器之间采用_____（同步，异步）时序进行数据传输。微处理器与打印机之间采用_____（同步，异步）时序进行数据传输。

⑦ PCI 总线采用_____（集中，分布）总线仲裁方式。

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_____（地址，数据），第二个阶段（最少一个时钟）交换_____（地址，数据）。

⑨ Pentium 的 3 个最基本的读写控制引脚是 M/$\overline{IO}$、_____和_____。

⑩ 用于要求微处理器插入等待状态的信号在 8086 上是引脚 READY，在 Pentium 上是_____

引脚，对应ISA总线是_____信号。

〔解答〕

① 0

② 读$\overline{RD}$，写$\overline{WR}$

③ 存储器读，存储器写

④ 4，2，10ns

⑤ 低有效，$\overline{IOR}$，0010

⑥ 同步，异步

⑦ 集中

⑧ 地址，数据

⑨ D/$\overline{C}$，W/$\overline{R}$

⑩ $\overline{BRDY}$，I/O CH RDY

〔习题6-4〕微处理器有哪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6-5〕8086微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，将迫使微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微处理器申请使用原来由微处理器控制的总线。

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微处理器申请可屏蔽中断。

〔习题6-6〕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：微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6-7〕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、微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，微处理器和存储器或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 6-8〕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、M/$\overline{IO}$、$\overline{RD}$ 和 $\overline{WR}$。画出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，其频率就是微处理器的工作频率，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。

M/$\overline{IO}$：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，表示微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，表示微处理器将访问 I/O 端口，此时地址总线 A15～A0 提供 16 位的 I/O 地址。

$\overline{RD}$：读控制，也是一个三态、输出低电平有效信号。有效时，表示微处理器正在从存储单元或 I/O 端口读取数据。

$\overline{WR}$：写控制，是一个三态、输出低电平有效信号。有效时，表示微处理器正将数据写到存储单元或 I/O 端口。

波形示意如图 6-1 所示。

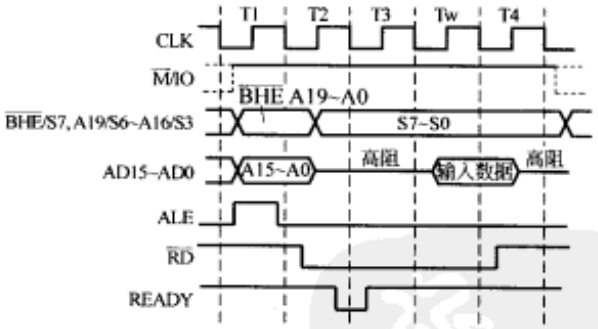

图 6-1　具有一个 Tw 的存储器读总线周期时序

〔习题 6-9〕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：位于微处理器附近的器件相互连接的总线，相当于芯片总线。

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：控制协调微处理器和内存、外设交互信息的一组导线。

ISA 总线：即 IBM PC/AT 总线，以微处理器 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 微处理器，也适用其他微处理器，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 6-10〕什么是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。

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 6-11〕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〕

$$（32\times8）\div（2\times8）=16\text{MBps}$$

〔习题 6-12〕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 6-13〕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 6-14〕自己可以着手 IO.LIB 子程序库的创建了。完善它或者添加删除子程序满足自己的要求。如果一个人感到困难，可以与其他人分工合作。

7.4 习题解答

〔习题 7-1〕简答题（用一句话回答）

① 存储系统为什么不能采用一种存储器件构成？
② 什么是高速命中和高速缺失？
③ 高速缓存 Cache 系统的标签存储器有什么作用？
④ 什么是 Cache 的地址映射？
⑤ Cache 的写入策略用于解决什么问题？
⑥ 存储器的存取时间和存取周期有什么区别？
⑦ 虚拟存储器是什么存储器？
⑧ DRAM 芯片怎么有行地址又有列地址？
⑨ 地址重复是怎么回事？
⑩ 主存有 ROM 和 RAM 区，PC 机为什么将 ROM 区设置在地址高端？

〔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。当微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，则微处理器直接读取 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、集成度高、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 PC 机采用的 80x86 微处理器复位后首先从地址高端开始执行指令，将 ROM 区设置为高端地址，是为了执行存放其中的 ROM-BIOS 程序，实现系统引导和初始化操作。

〔习题 7-2〕判断题（判断如下论述是正确还是错误）

① 存储系统的高速缓存需要操作系统的配合才能提高主存访问速度。
② 指令访问的操作数可能是 8 位、16 位或 32 位，但主存与 Cache 间却以数据块为单位传输。
③ 微机主存常以字节为单位标示存储容量，但存储器芯片习惯用二进制位为容量单位。
④ 存储器芯片的集成度高表示单位芯片面积制作的存储单元数多。
⑤ 微机大容量主存一般采用 DRAM 芯片组成。
⑥ 部分译码可以简化译码电路，不会减少可用的存储空间。
⑦ 存储系统每次给 DRAM 芯片提供刷新地址，被选中的芯片上所有单元都刷新一遍。
⑧ 存储系统的刷新地址提供给所有 DRAM 芯片。
⑨ FPM DRAM 芯片中的快页读写方式就是猝发传送方式。

⑩ ROM 芯片的烧写或擦写就是指对 ROM 芯片的编程。

〔解答〕

① 错　② 对　③ 对　④ 对　⑤ 对
⑥ 错　⑦ 错　⑧ 对　⑨ 错　⑩ 对

〔习题 7-3〕填空题（填写数字或文字）

① 计算机存储容量的基本单位：1 B（Byte）＝_____b（bits），1KB＝_____B，1MB＝_____KB，1GB＝_____MB，1TB＝_____GB＝_____B。

② 80486 片上 Cache 的容量是_____，采用_____路组合地址映射。

③ 在半导体存储器中，RAM 指的是_____，它可读可写，但断电后信息一般会_____；而 ROM 指的是_____，正常工作时只能从中_____信息，但断电后信息_____。

④ 存储结构为 8K×8 位的 EPROM 芯片 2764，共有_____个数据引脚、_____个地址引脚。用它组成 64KB 的 ROM 存储区共需_____片芯片。

⑤ 对一个存储器芯片进行片选译码时，有一个高位系统地址信号没有参加译码，则该芯片的每个存储单元占有_____个存储器地址。

⑥ 半导体_____芯片顶部开有一个圆形石英窗口。U 盘、MP3 播放器、数码相机、多媒体手机等设备一般采用半导体_____芯片构成存储器。

⑦ 在 8088 微处理器系统中，假设地址总线 A19～A15 输出 01011 时译码电路产生一个有效的片选信号。这个片选信号将占有主存从_____到_____的物理地址范围，共有_____容量。

⑧ 8086 和 80286 使用 16 位数据总线，主存分成偶数地址和奇数地址两个存储体。80386 和 80486 微处理器使用_____位数据总线，利用 4 个字节允许信号区别_____个存储体。Pentium 及以后的 IA-32 微处理器使用_____位数据总线，主存由_____个存储体组成。

⑨ 在 32 位 PC 机中，最低 640KB 主存空间常被称为_____主存；上位主存块 UMB 位于主存 A0000H～_____地址范围；1MB 之后的 64KB 区域被称为_____，它可以在实方式下被访问。

⑩ 8086 执行的第一条指令在物理地址_____开始的存储单元，80286 则是物理地址_____，IA-32 微处理器的起始执行物理地址为_____。它们都距离相应微处理器所支持主存空间最高端的 16 个存储单元位置，通常其中存放的是一条无条件转移 JMP 指令。

〔解答〕

① 8，1024，1024，1024，1024，2^{40}
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 常规（RAM 区系统），FFFFFH，高端主存区 HMA
⑩ FFFF0H，FFFFF0H，FFFFFFF0H

〔习题 7-4〕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。它可以分成空间局部和时间局部。在多数情况下，程序的代码段、数据段等数据被集中保存，程序多采用顺序执行、集中于某个循环或模块执行，

重复执行的循环体、反复运算的变量等。根据存储访问的局部性原理，紧邻被访问单元的地方也将被访问，刚被访问的单元也将再次被访问。

〔习题 7-5〕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，它们是透明的，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储器，才可与处理器进行数据交换。

〔习题 7-6〕区别 Cache 地址映射的概念：直接映射、相关映射和组合映射。

〔解答〕

直接映射：是将每个主存块固定地映射到某个 Cache 行。

（完全）相关映射：将一个主存块存储到任意一个 Cache 行。

组合（相关）映射：用多个相关映射的 Cache 按直接映射组合形成。

〔习题 7-7〕什么是 LRU 替换算法？

〔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。

〔习题 7-8〕Cache 系统的直写策略和回写策略有什么区别？

〔解答〕

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 7-9〕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 $4GB \div 2KB = 2^{32} \div 2^{11} = 2^{21}$ 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 7-10〕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 7-11〕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 $\overline{CS}$：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 $\overline{OE}$：在芯片被选中的前提下，若 $\overline{OE}$ 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 $\overline{WE}$：在芯片被选中的前提下，若 $\overline{WE}$ 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 7-12〕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，将存储器 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 7-13〕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 E^2PROM，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 7-14〕请给出教材图 7-9 中 74LS138 译码器的所有译码输出引脚对应的地址范围。

〔解答〕

$\overline{Y0} \sim \overline{Y7}$ 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 7-15〕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。其特点是：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 7-16〕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）、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 的存储空间，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 微处理器在 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：微处理器不可以直接访问，利用“体交换技术”实现微处理器访问。

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 4KB 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 7-17〕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〕

```
            ; 数据段
            include io.inc
crlf        db 13,10,'$'
            ; 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
            dispmsg crlf
            push ax
            mov ax,ds
            call disphw          ; 显示段地址
            mov ah,2
            mov dl,':'
            int 21h
            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 7-18〕完成 IO.LIB 子程序库的改造。

8.4 习题解答

〔习题 8-1〕简答题（用一句话回答）

① 外设为什么不能像存储器芯片那样直接与主机相连？
② 计算机两个功能部件、设备等之间为什么一般都需要数据缓冲？
③ 什么是接口电路的命令字或控制字？
④ PC 机中 CMOS RAM 属于主存空间吗？
⑤ 与系统总线连接的输入接口为什么需要三态缓冲器？
⑥ 透明锁存器和非透明锁存器有什么区别？
⑦ 什么样的外设可以采用无条件数据传送方式？
⑧ 什么是查询超时错误？
⑨ 远调用 CALL 指令和 INT n 指令有什么区别？
⑩ 为什么说外部中断才是真正意义上的中断？

〔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，与微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微处理器相连，必须经过一个中间电路。
② 数据缓冲用于匹配快速的微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 微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，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 接口形式与微处理器连接，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。只有当微处理器选通时，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；相对微处理器而言，其状态很少发生变化或变化很慢。这些设备与微处理器交换数据时，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 外部中断是由微处理器外部提出中断请求引起的程序中断。相对于微处理器来说，外部中断是随机产生的，所以是真正意义上的中断。

〔习题 8-2〕判断题（判断如下论述是正确还是错误）

① 微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。
② I/O 接口的状态端口通常对应其状态寄存器。
③ I/O 接口的数据寄存器保存微处理器与外设间交换的数据，起着数据缓冲的作用。
④ 80x86 微处理器的 64K 个 I/O 地址也像存储器地址一样分成逻辑段管理。
⑤ 指令 OUT DX,AX 的两个操作数均采用寄存器寻址方式。
⑥ 向某个 I/O 端口写入一个数据，一定可以从该 I/O 端口读回这个数据。
⑦ 程序查询方式的一个主要缺点是需要微处理器花费大量循环查询、检测时间。
⑧ 中断传送方式下，由硬件实现数据传送，不需要微处理器执行 IN 或 OUT 指令。

⑨ IA-32 微处理器保护方式用中断描述符表代替了实方式的中断向量表。

⑩ 某个外设中断通过中断控制器 IR 引脚向微处理器提出可屏蔽中断，只要微处理器开中断就一定能够响应。

〔解答〕

① 对　② 对　③ 对　④ 错　⑤ 错

⑥ 错　⑦ 对　⑧ 错　⑨ 对　⑩ 错

〔习题 8-3〕填空题（填写数字或文字）

① 计算机能够直接处理的信号是_____、_____和_____形式。

② 在 80x86 系统中，I/O 端口的地址采用_____编址方式，访问端口时要使用专门的_____指令，有两种寻址方式，其具体形式是：_____和_____。

③ 指令 IN 是将数据从_____传输到_____，执行该指令微处理器引脚产生_____总线周期。

④ 指令 IN AL, 21H 的目的操作数是_____寻址方式，源操作数是_____寻址方式。

⑤ 指令 OUT DX, EAX 的目的操作数是_____寻址方式，源操作数是_____寻址方式。

⑥ DMA 的意思是_____，主要用于高速外设和主存间的数据传送。进行 DMA 传送的一般过程是：外设先向 DMA 控制器提出_____，DMA 控制器通过_____信号有效向微处理器提出总线请求，微处理器回以_____信号有效表示响应。此时微处理器的三态信号线将输出_____状态，即将它们交由_____进行控制，完成外设和主存间的直接数据传送。

⑦ 在 IA-32 微处理器中 0 号中断被称为_____中断，有一个指令代码仅一个字节的指令中断 INTn，它是 n＝_____号中断。

⑧ IA-32 微处理器在开中断状态，其标志 IF＝_____。指令_____是开中断指令，而关中断指令是_____，关中断时 IF＝_____。

⑨ 实方式下，主存最低_____的存储空间用于中断向量表。向量号 8 的中断向量保存在物理地址_____开始的_____个连续字节空间；如果其内容依次是 00H、23H、10H、F0H，则其中断服务程序的首地址是_____。

⑩ 某时刻中断控制器 8259A 的 IRR 内容是 08H，说明其_____引脚有中断请求。某时刻中断控制器 8259A 的 ISR 内容是 08H，说明_____中断正在被服务。

〔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），微处理器（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，3

⑧ 1，STI，CLI，0

⑨ 1KB，20H，4，F010H：2300H

⑩ IR3，IR3 请求的

〔习题 8-4〕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

保存微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。微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

保存微处理器控制接口电路和外设操作的有关信息。微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 8-5〕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，微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，微处理器不再区分 I/O 口访问和存储器访问。其优点是：微处理器不用设计 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 8-6〕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：微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，主动向微处理器提出交换数据的请求。微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 8-7〕参见主教材图 8-5，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            ; 其他情况不点亮
next1:  mov cx,8
        mov al,1             ; 从 K0 开始
next11: out dx,al            ; 某个 LED 点亮
        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 点亮
         call delay             ; 延时 200ms
         shr al,1               ; ror al,1
         loop next21
         jmp again
```

〔习题 8-8〕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 8-9〕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 8-10〕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、向微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，微处理器进入中断响应总线周期。

关中断：微处理器在响应中断后会自动关闭中断。

断点保护：微处理器在响应中断后将自动保护断点地址。

中断源识别：微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：对微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：微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，恢复微处理器原来的工作环境。

开中断：微处理器允许新的可屏蔽中断。

中断返回：微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 8-11〕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。

微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 8-12〕明确如下与中断有关的概念：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、向微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，微处理器进入中断响应总线周期。

关中断：禁止微处理器响应可屏蔽中断。

开中断：允许微处理器响应可屏蔽中断。

中断返回：微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：微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微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，称为中断处理。

中断服务：指微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 8-13〕按照图 8-10 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 8-14〕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，将控制权还给微处理器。

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 8-15〕IA-32 微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 微处理器中，若 IF＝1，则微处理器处于开中断状态。

若 IF＝0，则微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 8-16〕简述 IA-32 微处理器的中断工作过程。

〔解答〕

IA-32 微处理器收到中断请求信号，获得向量号识别出中断源，响应中断过程如下：

① 将标志寄存器压入堆栈，保护各个标志位；

② 使 IF 和 TF 为 0，禁止可屏蔽中断和单步中断；

③ 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；

④ 根据向量号从中断向量表取出中断向量送代码段寄存器和指令指针寄存器；

⑤ 控制转移至中断服务程序入口地址，执行服务程序，最后是中断返回指令 IRET；

⑥ 中断返回指令 IRET 将断点地址和标志寄存器出栈恢复，控制返回到断点指令继续执行。

〔习题 8-17〕IA-32 微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 微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 8-18〕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 8-19〕

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 8-20〕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 8-21〕编写一个程序，将例题 8-4 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
```

9.4 习题解答

〔习题 9-1〕简答题（用一句话回答）

① 为什么称 8253/8254 定时器的工作方式 1 为可编程单稳脉冲工作方式？
② 为什么写入 8253/8254 定时器的计数初值为 0 却代表最大的计数值？
③ 微处理器通过 8255 的控制端口可以写入方式控制字和位控制字，8255 芯片如何区别这两个控制字呢？
④ “8255 芯片具有锁存输出数据的能力”是什么意思？
⑤ Modem（戏称“猫”）是一个什么作用的器件？
⑥ RS-232C 标准使用 25 针连接器，为什么 PC 机上常见的是 9 针连接器？
⑦ 什么是 RS-232C 的零调制解调器连接方式？
⑧ UART 器件的主要功能是什么？
⑨ 多路开关在模拟输入输出系统中起什么作用？
⑩ 微处理器为什么需要通过锁存器与数字/模拟转换器连接？

〔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，即调制解调器，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 机上就配置了 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 RS-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 微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 9-2〕判断题（判断如下论述是正确还是错误）

① 称为定时器也好，称为计数器也好，其实它们都是采用计数电路实现的。
② 计数可以从 0 开始逐个递增达到规定的计数值，也可以从规定的计数值开始逐个递减恢复到 0；前者为加法计数器，后者是减法计数器；8253/8254 芯片采用后者。
③ 32 位 PC 机中并没有 8253 或 8254 芯片，但其控制芯片组具有兼容其功能的电路。
④ 一次实现 16 位并行数据传输需要 16 个数据信号线。进行 32 位数据的串行发送只用一个数据信号线就可以。
⑤ 8255 没有时钟信号，其工作方式 1 的数据传输采用异步时序。
⑥ 调制解调器的信号调制是数字信号与模拟信号的转换，所以其转换原理与 ADC 或 DAC 器件一样。
⑦ 模拟地线和数字地线都是地线，所以一般可以随意连接在一起。

⑧ 电脑通过麦克风录音，需要 ADC 器件将音波转换为数字音频信号。

⑨ 模拟量转换为数字量一定会引入转换误差，所以一定有失真。

⑩ 当微处理器提供数字量后，DAC 器件将输出相应的模拟量，但 ADC 器件需要启动转换，隔一定时间后才能获得数字结果。

〔解答〕

① 对　② 对　③ 对　④ 对　⑤ 对

⑥ 错　⑦ 错　⑧ 对　⑨ 对　⑩ 对

〔习题 9-3〕填空题（填写数字或文字）

① 8253 芯片上有_____个_____位计数器通道，每个计数器有_____种工作方式可供选择。若设定某通道为方式 0 后，其输出引脚 OUT 为_____电平；当_____后通道开始计数，_____信号端每来一个脉冲_____就减 1；当_____，则输出引脚输出_____电平，表示计数结束。

② 假设某 8253 芯片的 CLK0 接 1.5MHz 的时钟，欲使 OUT0 产生频率为 300kHz 的方波信号，则 8253 芯片的计数值应为_____，应选用的工作方式是_____。

③ 8255 芯片具有_____个外设数据引脚，分成 3 个端口，引脚分别是_____、_____和_____。

④ 8255 芯片的 A 端口和 B 端口都定义为方式 1 输入，端口 C 上半部分定义为输出，则方式控制字是_____，其中 D0 位已经没有作用，可为 0 或 1。

⑤ 对 8255 芯片的控制寄存器写入 A0H，则其端口 C 的 PC7 引脚被用作_____信号线。

⑥ PC 机键盘上 ESC 键和字母 A 键的扫描码分别是_____和_____，断开扫描码分别是_____和_____。

⑦ 232C 用于发送串行数据的引脚是_____，接收串行数据的引脚是_____，信号地常用_____名称表示。

⑧ 欲使通信字符为 8 个数据位、偶校验、2 个停止位，则应向 8250 芯片_____寄存器写入控制字_____，其在 PC 系列机上的 I/O 地址（COM2）是_____。

⑨ 有符号数 32 的 8 位补码是 00100000，如果用 8 位偏移码是_____；有符号数−32 的 8 位补码是 11100000，如果用 8 位偏移码是_____。

⑩ 如果 ADC0809 正基准电压连接 10V，负基准电压接地，输入模拟电压 2V，则理论上的输出数字量为_____。

〔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 $\overline{OBF}$

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 9-4〕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9-5〕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9-6〕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9-7〕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524、588、660、698、784、880、988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:  mov ah,1             ; 等待按键
        int 21h
        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 9-8〕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，微处理器响应中断，执行输出 OUT 指令：输出数据给 8255，发出 $\overline{WR}$ 信号。查询方式下，通过端口 C 的状态确信可以输出数据，微处理器执行输出指令；

② $\overline{WR}$ 信号一方面清除 INTR，另一方面在上升沿使 $\overline{OBF}$ 有效，通知外设接收数据。实质上 $\overline{OBF}$ 信号是外设的选通信号；

③ $\overline{WR}$ 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 $\overline{ACK}$ 响应；

④ $\overline{ACK}$ 信号使 $\overline{OBF}$ 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 9-9〕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 9-10〕在用 8255 端口 A 方式 0 与打印机接口的示例中，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 9-11〕在用 8255 端口 A 方式 1 与打印机接口的示例中，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 9-12〕有一工业控制系统，有 4 个控制点，分别由 4 个对应的输入端控制，现用 8255 的端口 C 实现该系统的控制，如图 9-1 所示。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
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
```

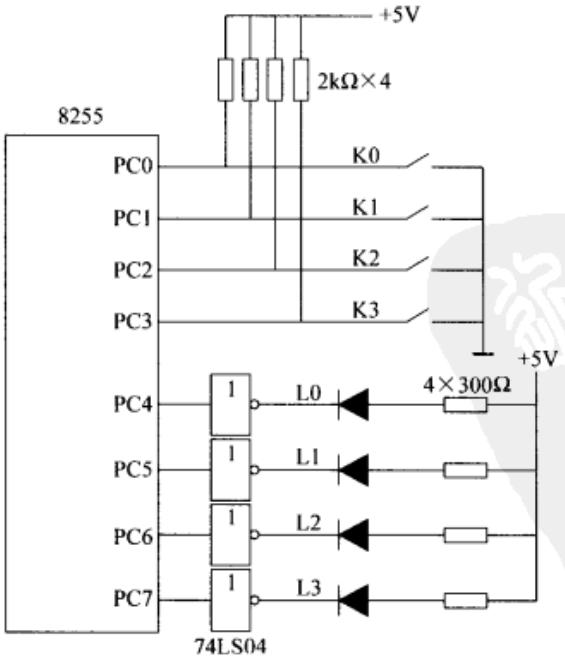

图 9-1　习题 9-12 附图

〔习题 9-13〕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            db 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
```

〔习题 9-14〕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（如图 9-2 所示），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。

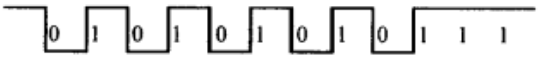

图 9-2 波形示例图

〔习题 9-15〕微机与调制解调器通过 RS-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 RS-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如下。

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 RS-232C 进行两个微机直接相连通信时，可采用教材图 9-24 所示连接方式。

〔习题 9-16〕8250 的 IIR 是只读的，且高 5 位总是 0。试分析 IBM PC/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 9-17〕首先采用自循环查询方式在本机上实现例题 9-3。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 9-4 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 9-4 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。

〔习题 9-18〕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 9-19〕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 9-20〕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，主动向微处理器申请中断。微处理器响应中断后，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，就可把结束信号作为 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。

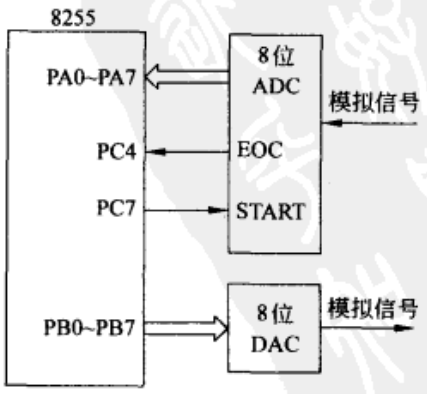

图 9-3　习题 9-21 附图

〔习题 9-21〕某控制接口电路如图 9-3 所示。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。微处理器采集该数据，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
```