全国高等教育自学考试系列辅导教材

汇编语言程序设计
复习与考试指导

计算机及应用专业（专科）

孙琦　郑硕　　编写
姚君遗　　审

高等教育出版社

内 容 提 要

本书是全国高等教育自学考试“计算机及应用专业”系列辅导教材之一，是专科“汇编语言程序设计”课程的复习与考试指导教材。它依据考试大纲“识记、领会、识别、应用”的四个层次，紧密结合自考考生特点，以典型例题的形式对每一章节的内容进行透彻的分析与解答，每一章后配有自测试题，供考生及时检验学习效果；同时本书还提供了四套模拟试卷，全面涵盖本课程的考核知识点，以利考生熟悉考试过程，从而进入良好的考试状态并顺利通过考试。

本辅导教材除适用于“计算机及应用”专业的考生外，还适用于“计算机信息管理”、“计算机通信工程”和“计算机应用及教育”等专业相同课程的考生，也可以作为工程技术人员、社会读者的学习参考用书。

图书在版编目（CIP）数据

汇编语言程序设计复习与考试指导/孙琦，郑硕编写.
北京：高等教育出版社，2002.1
ISBN 7－04－010464－4

Ⅰ.汇… Ⅱ.①孙…②郑… Ⅲ.汇编语言－程序设计－高等教育－自学考试－自学参考资料
Ⅳ.TP313

中国版本图书馆CIP数据核字（2001）第088515号

汇编语言程序设计复习与考试指导
孙琦 郑硕 编写

出版发行 高等教育出版社
社 址 北京市东城区沙滩后街55号 邮政编码 100009
电 话 010-64054588 传 真 010-64014048
网 址 http:// www.hep.edu.cn
http:// www.hep.com.cn

经 销 新华书店北京发行所
印 刷 北京市鑫鑫印刷厂

开 本 787×1092 1/16 版 次 2002年1月第1版
印 张 13.75 印 次 2002年1月第1次印刷
字 数 320 000 定 价 17.60元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

序

高等教育自学考试正式实施已经 20 周年了。20 年来，每年都有数以百万计的人参加考试，目前全国自考在籍考生已达 1600 多万人，显示了其在高等教育总体格局中的地位。近年来，随着计算机技术及应用水平的不断提高，报考“计算机及应用”专业的考生逐年增加。

为了满足自学考生的需求，高等教育出版社根据全国高等教育自学考试指导委员会 1999 年重新修订的“计算机及应用专业教学大纲”的教学基本要求和相应新教材的出版，组织编写了“高等教育自学考试计算机及应用专业系列辅导教材”。在这一系列辅导教材中，包括“计算机及应用”专业专科和独立本科段开考的共 15 门课程的辅导教材。

这套辅导教材是根据教材的编写思想和教学大纲的基本要求，从自考生的学习水平及自考生以自学为主的特点出发，为应试考生提供既包含自考教材中的考核知识点，又包含各知识点相互关系的讲解；既指出课程中的学习重点，又进行课程难点的分析；既提供多种典型例题，又详细地进行分析解答；各章后面附有大量的自测试题，以帮助考生测试各章内容的掌握情况，全书后面包含有多套模拟试卷，综合、全面测试考生对本课程的理解和掌握情况；使考生既学到了知识，又逐渐进入考试的实战状态，达到考试中应答自如、顺利通过考试的目的。本系列辅导教材并不单纯以解答教材中的例题为目的，而是通过作者在自考辅导过程中积累的典型例题，逐层分析、解答，从而使考生全面掌握本课程的知识。由此，也构成了本系列辅导教材例题丰富、实用性强的特点。

在组织编写这套系列辅导教材时，高等教育出版社选择的作者中，有些是自学考试教材的编写者，有些是多年来一直从事自学考试辅导教学工作的出色教师，他们非常了解自学考生的特点，富有自学考试的辅导经验。

本系列辅导教材适用于“计算机及应用”专业专科和独立本科段考生的自学及教师上课辅导，也适用于“计算机信息管理”专业、“计算机通信工程”专业、“计算机应用及教育”专业相同课程的考生，也可以作为工程技术人员、社会读者的学习用书。

全国高等教育自学考试指导委员会
电子电工与信息类专业委员会副主任
陈国良
2001 年 7 月

前　言

汇编语言是直接面对微机硬件的语言，通过它可直接使用 CPU 中的寄存器，或直接按地址控制对存储器及输入/输出设备进行读或写。本课程中所涉及的汇编语言程序设计仅限于单纯用汇编语言编写程序，基本上属于纯软件的内容。

大家知道，对于计算机软件方面的课程都有这样的特点：在学习把握课程基本内容及考试要求的同时，需要大量地阅读和上机编写程序练习，只有大量地阅读和亲自编写程序，才能熟悉和掌握其中的规律及技巧，也只有当阅读和亲自编程积累到一定的量时，才会对此方面的问题达到应对自如的程度。对于广大参加高等教育自学考试的朋友，由于课程学习和上机练习的条件远不如脱产学习的在校学生，且不能直接面对参加出题的教师，参加考试的难度将是非常大的。我们作为从事本门课教学多年的教师，现根据“汇编语言程序设计”课程自学考试大纲的要求，将在本门课程教学中涉及到的常见问题及多年积累的典型例题及习题，经整理并配备相应的答案后，愿意提供给广大参加自学考试的朋友们，希望能对你们练习与提高提供一些帮助。

本书第一章至第五章由北京航空航天大学孙琦老师编写，第六章至第八章由中国人民大学郑硕老师编写。书稿完成后，高等教育自学考试本门课程原教材的作者、合肥工业大学的姚君遗教授于百忙之中对本书进行了最终审阅，姚教授对本书提出了很多宝贵的意见，在此特对姚教授致以衷心的感谢。

因本书编写时间较为仓促及作者水平有限，书中错误之处在所难免，敬请广大读者批评指正。

编　者

2001 年 10 月

目　录

第一章　基 础 知 识

1.1　必考知识点

1．计算机系统的基本概念和基本组成，应达到“识记”的层次

（1）有关计算机硬件的基本概念和计算机系统基本组成，如中央处理器、存储器、输入/输出设备；总线（地址总线、数据总线、控制总线）；CPU 执行指令的过程；CPU 到存储器取数据的过程。

（2）计算机系统的软件，如系统软件（操作系统）和应用软件；文本编辑程序、翻译程序、链接程序；源程序、目标程序、可执行程序；软件调试工具。

2．8086 汇编语言编程的硬件模型，应达到“综合应用”的层次

（1）8086 CPU 的组成，如通用寄存器组 AX、BX、CX、DX；段寄存器组 CS、DS、ES、SS；地址指令寄存器组 IP、SI、DI、SP、BP；地址加法器——存储器物理地址的合成；算术逻辑单元 ALU；程序状态字 PSW（标志寄存器 FR）；8086 复位时各寄存器的状态。

（2）8086 的存储器组织，如存储单元的地址和内容；字节数据、字数据在存储器中的存放；存储器的分段使用；堆栈段数据存取的特点（对 8086/8088）。

（3）输入/输出设备，如输入/输出设备的寻址空间等。

3．汇编语言程序设计的特点和应用，应达到“识记”的层次

（1）汇编语言的概念和特点，如程序、指令、机器语言（二进制指令代码）；汇编语言与机器语言的关系。

（2）汇编语言在微型计算机中的应用，如汇编语言与计算机硬件的关系——汇编语言是直接控制硬件的语言；汇编语言在微型计算机中的应用。

1.2　重点与难点解析

1.2.1　8086/8088 CPU 与存储器及输入/输出设备的连线

8086/8088 CPU 与存储器及输入/输出设备的连线分为：地址总线、数据总线和控制总线。

图 1-1 中只画出了地址总线、数据总线和控制总线与存储器的连接，实际上，地址总线、数据总线和控制总线同样也连到了输入/输出设备上。

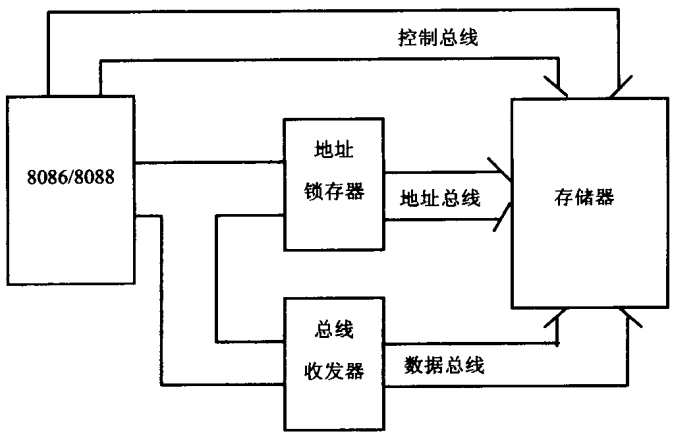

图 1-1　8086/8088 CPU 与存储器连线图

8086/8088 CPU 是根据程序进行工作的，控制 CPU 工作的程序和 CPU 要处理的数据都放在存储器（Memory）中，所以，一个能正常工作的微型计算机系统除了有 CPU 外，必须还得有存储器。CPU 与存储器之间通过三类连线连接起来，即地址总线、数据总线和控制总线。

1.2.2　由存储单元组成的存储器

如图 1-2 所示，存储器是由存储单元组成的，存储器与存储单元就像一座大厦与它当中的房间一样。要选中存储器中的某个存储单元工作，就要给存储器发一个地址，每个存储单元都有一个自己的地址，地址就像大厦的房间号一样。

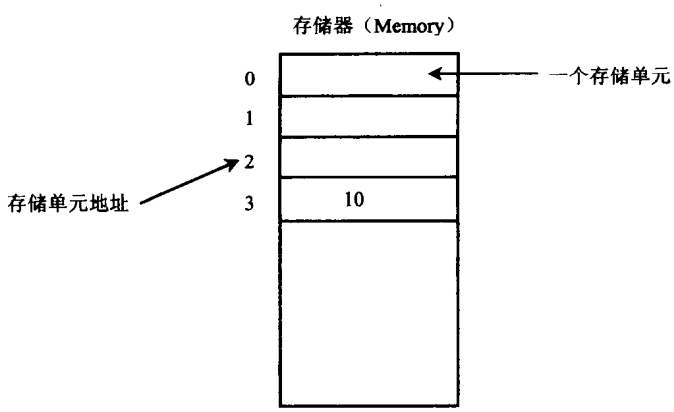

图 1-2　存储器示意图

CPU 到某个存储单元中读取数据过程是：先通过地址总线发出该存储单元的地址，选中该存储单元，然后通过控制总线发读信号，使该存储单元中的数据能输出到数据总线上，再沿着数据总线将该存储单元中的数据读入到 CPU 中。

存储单元是按 8 位二进制数编地址的，即每个存储单元中可存一个 8 位二进制数，如

果一个数据值较大，一个存储单元放不下，可用两个存储单元或更多的存储单元存放。

1.2.3 8086/8088 CPU 的内部结构

1．8086 CPU 内部的三类寄存器

（1）通用寄存器 AX、BX、CX、DX

8086 CPU 内部具有 4 个用于运算的 16 位通用寄存器 AX、BX、CX、DX。如图 1-3 所示，这 4 个 16 位的寄存器每个又可分为 2 个 8 位的寄存器使用，例如，AX 可分为 AH 和 AL 两个 8 位的寄存器，H 即 High 的英文首字母，L 即 Low 的英文首字母。

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

图 1-3 8086 通用寄存器

（2）段寄存器 CS、DS、ES、SS 及地址指针寄存器 SI、DI、SP、BP、IP

8086 CPU 内部具有 4 个给出存储器段地址的 16 位段寄存器 CS、DS、ES、SS，5 个给出存储器偏移地址的 16 位地址指针寄存器 SI、DI、SP、BP、IP。

存储器的地址都是 CPU 给出的，对于 8086/8088 微机系统，存储器的地址是由段寄存器和地址指针寄存器共同给出，段寄存器给出段地址，地址指针寄存器给出偏移地址，16 位的段寄存器和 16 位的地址指针寄存器的值经过地址加法器共同合成一个存储器的 20 位物理地址，然后通过地址总线输出给存储器。这个过程可用图 1-4 来表示。

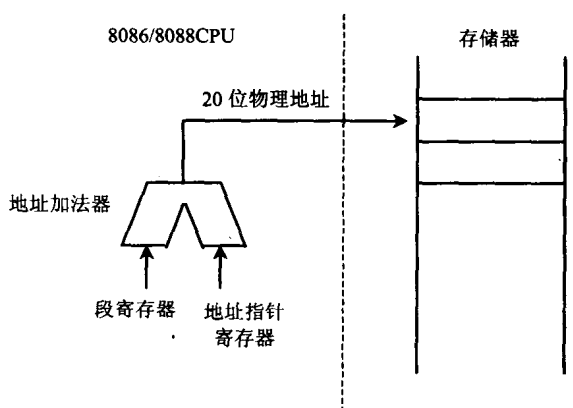

图 1-4 8086 输出存储器地址示意图

段寄存器的值与地址指针寄存器的值是如何经地址加法器合成存储器的地址呢？地址加法器并不是简单地将段寄存器的值加上地址指针寄存器的值作为存储器的地址输出，而是将 16 位的段寄存器的值先左移 4 位二进制位右补 0，得到一个 20 位的二进制数，然后再加上 16 位的地址指针寄存器的值，以此作为存储器的物理地址输出。**物理地址**就是 8086/8088 从地址总线输出给存储器的地址。例如，当前码段寄存器 CS 的值为 1111111111111111，指令指针寄存器 IP 的值为 0000000000000000，则 8086 输出的存储器码段的物理地址就是：

码段存储器物理地址 = CS 左移 4 位右补 0+IP
= 1111111111111110000B+0000000000000000B
= 1111111111111110000B
= FFFF0H

因为将一个二进制数左移一位右补 0，相当于将这个二进制数将乘以 2，所以将段寄存器的值左移 4 位二进制位右补 0 从结果来看，相当于将段寄存器的值乘以 16（4 次乘以 2），或相当于将段寄存器的值乘以十六进制数 10H（左移 4 位二进制位，右补 4 个 0）。

用公式来表示 8086/8088 输出的存储器的地址为：

存储器物理地址=段寄存器值*16+偏移地址

或

存储器物理地址=段寄存器值*10H+偏移地址

（3）程序状态字寄存器 PSW（Program Status Word）

8086 CPU 内部具有 1 个 16 位的程序状态字寄存器 PSW，程序状态字 PSW 又称为标志寄存器 FR（Flag Register），用于指示每条运算指令执行后累加器的状态和对 CPU 的控制作用。

8086 的程序状态字虽然是一个 16 位的寄存器，但只有其中的 9 位有意义，如图 1-5 所示。

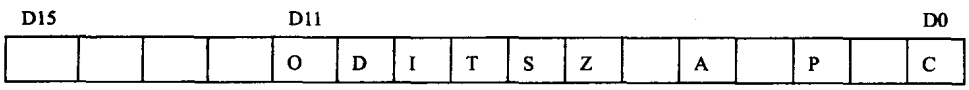

图 1-5　8086 的程序状态字寄存器

其中标记有英文字母的位是有意义的位（有些书中用相应英文字母后加 F 的方法表示，如其中的 D0 位又称为 CF 位），而空白位均是无意义的位。这有意义的 9 个标志位称为：

OF 位——溢出标志位
DF 位——方向标志位
IF 位——中断允许标志位
TF 位——单步跟踪标志位
SF 位——符号标志位
ZF 位——零标志位
AF 位——辅助（半）进位标志位
PF 位——偶个 1 标志位
CF 位——进位标志位

在这有意义的 9 位中，有 6 位受运算指令执行结果的影响，它们是：OF、SF、ZF、AF、PF 和 CF 位。另 3 位 DF、IF 和 TF 位不受运算结果影响，表示某种控制作用可由指令设置为 1 或 0。对于受运算结果影响的 6 个标志位，如果某一位为 1，表示运算结果有相应的状态存在，如果其为 0，表示运算结果无相应的状态存在。例如，经过某个运算后，溢出标志位 OF 位变为 1，表示这个运算的结果产生了溢出，即数据大小超出了累加器所能存放下的数据的

范围。

2. **算术逻辑单元**ALU（Arithmetic Logic Unit）

计算机中的所有运算都是在 CPU 的 ALU 中进行的，只是不同型号的 CPU，其 ALU 的功能不同而已。只要进行运算就要经过 ALU，而只要经 ALU 运算就会影响程序状态字 PSW（标志寄存器）的状态。

算术逻辑单元 ALU 与程序状态字 PSW 的关系见图 1-6 所示。ALU 符号中的 A 和 B 位置表示两个准备要经过 ALU 进行运算的数，ALU 下方引出一箭头指向 A 位置，表示运算的结果存在 A 中。ALU 右侧的一个箭头指向 PSW，表示只要经 ALU 运算，就会影响 PSW（对 8086/8088 CPU，影响其中的 6 位）。

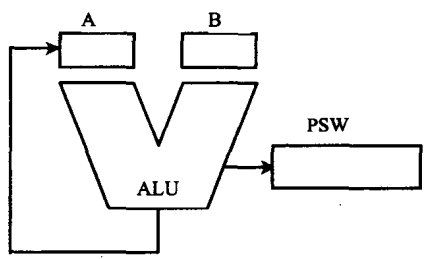

图 1-6 算术逻辑单元 ALU 与程序状态字

A 和 B 两个数经 ALU 运算并将结果存于 A 中，用公式可表示为：

A = A 运算 B

因为 A 中存有运算结果，所以，PSW 就反映 A 中运算结果的状态。A 和 B 位置可以是两个通用寄存器，或是一个通用寄存器和一个存储器中的数据。

当运算结果产生溢出时，PSW 的 OF 位为 1；当运算结果未产生溢出时，OF 位为 0。

当运算结果为负数时，PSW 的 SF 位为 1；当运算结果为正数时，SF 位为 0。

当运算结果为零时，ZF 位为 1；运算结果非零时，ZF 位为 0。

当运算结果的 D3 位向 D4 位产生进位或借位时，AF 位为 1；反之，AF 位为 0。

当运算结果的 D0 位到 D7 中有偶数个 1 时，PF 位为 1；否则（有奇数个 1 时），PF 位为 0。

当运算结果的最高位有进位或有借位时，CF 位为 1，无进位或借位时，CF 位为 0。

【例 1-1】 设有通用寄存器 AL=10000000B，BL=10001000B，AL 与 BL 通过 ALU 相加，结果放在 AL 寄存器中，写出相加后 PSW 的状态。

答：

```
    10000000    AL
+   10001000    BL
-------------------
  1 00001000    AL
```

注意：因 AL 寄存器是 8 位的寄存器，因此产生的进位无法放在 AL 中，AL 只能保存运算结果的低 8 位，所以 AL=00001000B。

由于相加结果在最高位产生了进位 1，故程序状态字的 CF 位被置 1。CF 位的作用是：当两个八位数运算时，CF 充当第 9 位（即 D8 位）；当两个十六位数运算时，CF 充当第 17 位（即 D16 位）。

相加结果 AL 寄存器中的数含有 1 个 1（奇数个 1），不是含有偶数个 1，故偶个 1 标志位 PF 变为 0。

相加结果 AL 寄存器的 D3 位没有向 D4 位产生进位，故半进位标志位 AF 变为 0。

相加结果 AL 寄存器中存的数为 08H，不为 0，故零标志位 ZF 位变为 0。

相加结果 AL 寄存器中数的 D7 位为 0，CPU 认为是正数，故符号标志位 SF 变为 0。

对本例，在相加前，AL 和 BL 寄存器中数的 D7 均为 1，CPU 认为是两个负数。相加后 AL 寄存器中数（运算结果）的 D7 位为 0，CPU 认为是正数。两个负数相加结果本应为负数，但本例却变成了正数，这是因为 8 位的 AL 寄存器放不下运算的完整结果，因此产生溢出，所以，程序状态字的溢出标志位 OF 位变为 1。

3．地址加法器

地址加法器的作用是将 8086/8088 中的段寄存器的值左移 4 位二进制位右补 0，再加上偏移地址，得到一个存储器的物理地址，然后作为存储器的地址从地址总线输出。前面已有叙述。

1.2.4　8086/8088 的存储器结构

1．存储单元的地址和内容

对于微型计算机，它只能存储二进制数，其它进制数（如十进制、八进制数、十六进制等）必须首先转换成二进制数才能储存在微机中。

存储器是以 8 位二进制数为单位进行编址的，即每 8 位二进制数分配一个地址，每个地址称为一个存储单元（又称一个字节）。一个存储器是由许多存储单元组成的。图 1-7 是一个存储单元的示意图。

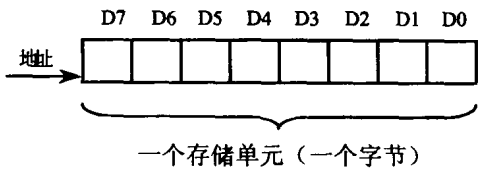

图 1-7　存储单元示意图

图中存储单元的最低位称为 D0 位，最高位称为 D7 位。

【例 1-2】 设有一个十进制数 59 放在地址为 1000H 的存储单元中，试说明它的存放形式。

答： 首先将十进制数 59 化成二进制数：

59=111011B

二进制数 111011 尾部的 B 为标识符，表示左面的数据是二进制数。

这个以二进制数形式表示的数（59）就可以放到地址为 1000H 的存储单元中了。要注

意的是，数据存在存储器中不是占一个存储单元就是占两个存储单元，如果未占满一个存储单元，则高位补 0。59 在存储单元中的存放如图 1-8 所示。

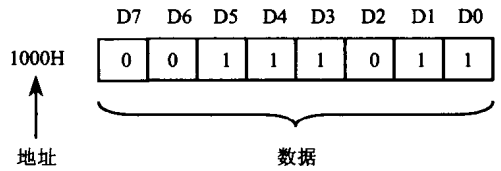

图 1-8 59 放在 1000H 存储单元中

【例 1-3】 设有一个十六进制数 1234H 放在 2000H 和 2001H 存储单元中，说明它的存放形式。

答：十六进制数 1234H 的二进制形式为：

1234H=0001001000110100B

放在存储器中应占两个存储单元。一个数据放在存储器中，如果一个存储单元放不下，需要占多个存储单元时，则总是低位数据放在低地址中，高位数据放在高地址中。对于 1234H 化成的二进制数，低位数据 34H 对应的二进制数 00110100B 放在低地址 2000H 地址中，高位数据 12H 对应的二进制数 00010010B 放在高地址 2001H 地址中。1234H 在存储器中的存放形式如图 1-9 所示。

尽管数据在存储器中是以二进制数的形式存放的，但书写时为了简单，常不像前面那样将不同进制数先化成二进制形式，再将二进制数放入存储单元的不同位中，而是直接称某存储单元放了某一个数据。例如，对例 1-3，称 2000H 地址中放入 34H，2001H 地址中放入 12H。1234H 在存储单元中的存放也简化为如图 1-10 所示。

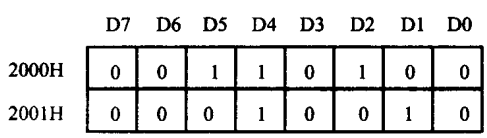

图1-9 1234H 放在 2000H 存储单元中

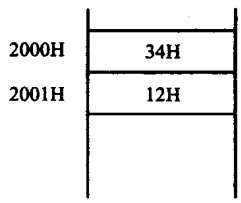

图 1-10 数据在存储单元中的存放

对例 1-3，因为 1234H 是一个整体的字数据，虽然它占地址为 2000H 和 2001H 两个存储单元存放，但实际上，只称它放在 2000H 中。一个数据占多个存储单元存放，称它的最低字节数据所占的地址为它的存放地址。

2．8086系统的存储器分段使用的概念

8086 系统的存储器分为堆栈段、数据段、附加数据段和码段使用。如图 1-11 所示。

堆栈段是用于暂存一些数据。数据段用于存放用户定义的变量等数据。附加数据段可作为数据段的补充存放用户定义的数据。码段用于存放用户编写的程序代码。

存储器是分段使用的，不同段的段地址是由不同的段寄存器给出。当 8086/8088 到堆栈段中存或取数据时，由堆栈段寄存器 SS（给出段地址）和堆栈指针寄存器 SP（给出偏移地址）共同给出存储器的物理地址，当 8086/8088 到数据段或附加数据段存或取数据时，由数

据段寄存器 DS 或附加数据段寄存器 ES 给出存储器的段地址（数据段地址的偏移地址由谁给出视 CPU 执行的指令而定，此问题将在第二章中详细介绍），当 8086 到码段去取指令的二进制代码时是由码段寄存器 CS（给出段地址）和指令指针寄存器 IP（给出偏移地址）共同给出存储器的物理地址。

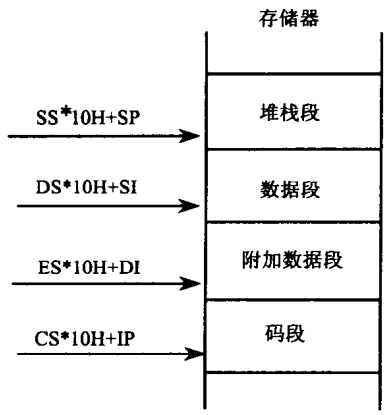

图 1-11　存储器分段使用示意图

【例 1-4】 当前码段寄存器 CS=1000H，指令指针寄存器 IP=100H。写出 8086 到存储器的什么地址去取指令（二进制代码）执行。

答： 8086 总是到码段寄存器 CS 与指令指针寄存器 IP 所给的地址中去取指令的二进制代码，然后执行。8086 当前取指令代码的存储器物理地址为：

码段物理地址= CS*10H+IP
= 1000H*10H+100H
= 10000H+100H
= 10100H

即 8086 先从地址总线输出 10100H 地址，选中 10100H 这个存储单元，然后将该存储单元中的指令二进制代码读到 8086/8088 中，然后译码执行这条指令。

3．堆栈段

对 8086/8088 系统，堆栈数据的存取都是按字（两个字节）进行的，不允许按字节或其它形式存取。

堆栈段存储器的地址是由堆栈段寄存器 SS 和堆栈指针寄存器 SP 共同给出的。每向堆栈中存入一个字数据，SP 的值就自动减 2，每从堆栈中取出一个字数据，SP 的值就自动加 2。

1.3　典型例题分析解答

【例 1-5】 8086/8088 复位后，从存储器的什么地址开始执行程序？

答： 8086/8088 复位后，除了码段寄存器 CS 的值为 FFFFH 外，其余各寄存器的值均为 0。8086/8088 的程序是放在存储器的码段当中的，码段的地址是由码段寄存器 CS 和指令指针寄存器 IP 经地址加法器合成一个 20 位的物理地址后输出给存储器的。因此，8086/8088 复位后输出的码段物理地址为：

CS*10H+IP=FFFFH*10H+0=FFFF0H+0=FFFF0H

即 8086/8088 复位后，从存储器的 FFFF0H 地址开始执行程序。

【例 1-6】 已知寄存器 AX=AAAAH，BX=1234H，求 AX=AX+BX。写出 AX 中运算结果及运算后程序状态字 PSW 各有关位的状态。

```
  1010101010101010      AX
+ 0001001000110100      BX
------------------------------
  1011110011011110      AX
```

即运算结果 AX=BCDEH（AH=BCH，AL=DEH）。

下面看程序状态字 PSW 的状态：

参加运算的为两个 16 位数，最低位称 D0 位，最高位称 D15 位，因最高位 D15 未向 D16 位产生进位或借位，故进位位标志 CF=0。

因运算结果 AX 寄存器的低 8 位 AL 中为 DEH，含有偶数个 1，故 PF=1。因 D3 位未向 D4 位产生进位或借位，故 AF=0。

因运算结果 AX 的内容不为 0，故 ZF=0。

因运算结果 AX 寄存器的最高位 D15 位为 1，故符号标志位 SF=1。

因参加相加的两个数最高位一个为 0，另一个为 1，故运算结果不产生溢出，OF=0。

在**例** 1-6 中有几点要特别注意，进位是指参加运算的两个数的最高位，如果参加运算的两个数是 8 位数，则有无进位要看 D7 位是否向 D8 位产生进位或借位；如果参加运算的两个数是 16 位数，则有无进位要看 D15 位是否向 D16 位产生进位或借位。实际上，两个 8 位数进行运算，CF 位充当第 9 位（即 D8 位），两个 16 位数进行运算，CF 位充当第 17 位（即 D16 位）。

PF 位不论参加运算的为多少位数，它所指示的都是低 8 位是否含有偶数个 1。

AF 位不论参加运算的为多少位数，它所指示的都是 D3 位是否向 D4 位产生进位或借位。

SF 位总是与运算结果的最高位保持相同。

OF 位是比较难确定的，可以这样来判断：看两个数运算的结果是否能在累加器中放下，如果放得下，则不产生溢出（OF=0），如果放不下，则产生溢出（OF=1）。对加法，如果是两个正数相加或是两个负数相加，则有可能产生溢出。而对于：

正数 + 负数

负数 + 正数

则肯定不会产生溢出。对减法：

正数 - 负数

负数 - 正数

可能产生溢出。而

正数 - 正数

负数 - 负数

则肯定不会产生溢出。

【例 1-7**】** 已知有一个字数据 123，存在物理地址为 120H 的存储单元中，用图表示其存放形式。

数据在计算机中都是以二进制数形式存放的。十进制 123=1111011B=7BH，因 123 是个字数据，所以，放在存储单元中为：0000000001111011B。

又因存储器是按 8 位二进制数（字节）编地址的，所以，123 放在存储单元中要占两个地址存放，即高 8 位数据 00000000B 占一个存储单元存放，低 8 位数据 01111011B 占一个

存储单元存放。

根据当一个数据占多个存储单元存放时，总是低位数据占低地址存放，高位数据占高地址存放，及它的最低 8 位数据的存放地址称为它的存放地址，由此可知，123 在存储单元中的存放形式应是 01111011 放在 120H 地址中，00000000 放在 121H 地址中。用示意图可表示如图 1-12 所示。

【例 1-8】 已知当前堆栈段寄存器 SS=1000H，堆栈指针寄存器 SP=100H，寄存器 AX=1234H。写出将 AX 寄存器值存入堆栈后，SS 及 SP 的值。

AX 入栈前，堆栈段物理地址为：

SS*10H+SP=1000H*10H+100H=10100H

AX 入栈后，堆栈段段寄存器 SS 的值保持不变，而堆栈指针寄存器 SP 自动减 2，这时，堆栈的物理地址为：

SS*10H+SP=1000H*10H+（100H-2）=10000H+FEH=100FEH

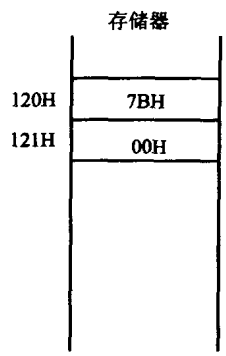

图 1-12　字数据 123 在 120H 地址中的存放

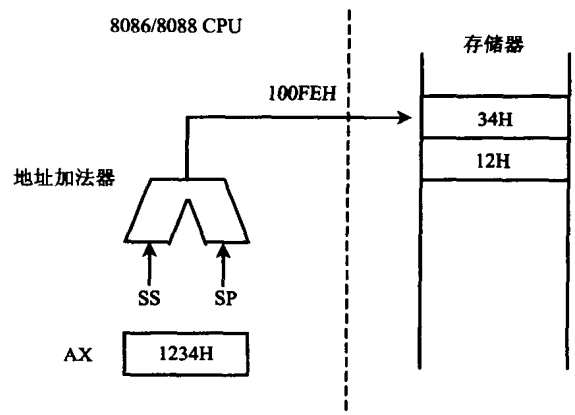

图 1-13　寄存器 AX 值入栈后情况

1.4 自 测 试 题

1.1　8086/8088 CPU 中有哪些通用寄存器，它们有什么用途？

1.2　8086/8088 CPU 中有哪些段寄存器和地址指针寄存器？它们有什么用途？

1.3　8086/8088 中的 PSW 有什么用途？它是如何反映运算结果状态的？

1.4　8086/8088 中的地址加法器和 ALU 有什么不同？

1.5　分别将十进制数 95，十六进制数 95H 转换成二进制数。

1.6　计算 100 和-100 的补码是多少？

1.7　画出字节数据 100 在地址为 200H 的存储单元单元中的存放示意图，说明 D0 位和 D7 位存储的是什么内容。

1.8　设 1234H：567H 存储单元中存储了一个字数据 2211，说明该存储单元的物理地址是多少，字数据 2211 在相应存储单元中的存放形式。

1.9　设 AL=100，BL=50，说明将 AL 的值加上 BL 的值的结果是多少，对 PSW 有关位

的影响情况如何？

1.10 设 AL=100，BL=50，说明将 AL 的值减去 BL 的值的结果是多少，对 PSW 有关位的影响情况如何？

1.11 设 AX=1000，BX=1000H，说明将 AX 的值加上 BX 的值的结果是多少，对 PSW 有关位的影响情况如何？

1.12 已知当前堆栈段寄存器 SS=1000H，堆栈指针寄存器 SP=100H，在该地址中存储的是 1234H。写出向该地址中存入一个字数据后，SS 及 SP 的值。

自测试题答案

1.1 有 AX、BX、CX、DX 四个通用寄存器，用于运算和存储数据。

1.2 有 CS、DS、ES、SS 四个段寄存器和 IP、SI、DI、SP、BP 五个地址指针寄存器。段寄存器和地址指针寄存器用于给出存储器的地址。

1.3 用于指示运算结果的状态和设置某些工作方式。

1.4 8086/8088 中的地址加法器用于将段地址和偏移地址合成为存储器的物理地址，ALU 用于算术运算和逻辑运算。

1.5 95=01011111B，95H=10010101B。

1.6 $[100]_{补码}$=01100100B，$[-100]_{补码}$=10011100B。

1.7 如图 1-14 所示，100=64H=01100100B；所以 D0 位为 0；D7 位为 0。

1.8 1234H：567H 对应的物理地址为：

1234H*10H+567H=12340H+567H=128A7H

2211 在 1234H：567H 的存放：

先将 2211 化成十六进制数：2211=08A3H（存放形式如图 1-15 所示）。

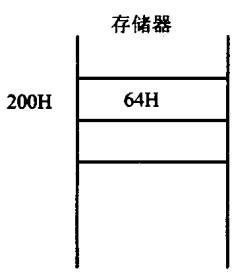

图 1-14 字数据 100 在 200H 地址中的存放

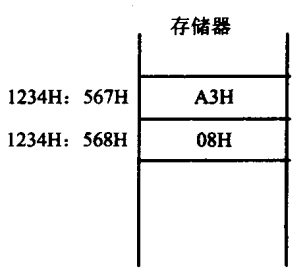

图 1-15 字数据 2211 在 1234H：567H 地址中的存放

1.9 AL+BL=64H+32H=96H=10010110B

指令执行后，CF=0，PF=1，AF=0，ZF=0，SF=1，OF=1

1.10 AL-BL=64H-32H=32H=00110010B

指令执行后，CF=0，PF=0，AF=0，ZF=0，SF=0，OF=0

1.11 AX+BX=1000+1000H=03D8H+1000H=13D8H=0001001111011000B

指令执行后，CF=0，PF=1，AF=0，ZF=0，SF=0，OF=0

1.12 SS：SP=1000H：FEH