高·校·教·育·教·学·课·程·建·设

汇编语言程序设计教程

Assembly Language Program Design

刘臣奇 主编

沈 阳 出 版 社

高校教育教学课程建设

汇编语言程序设计教程

Assembly Language Program Design

主　　编　刘臣奇

副 主 编　王　罡　张林家

沈阳出版社

前　言

“汇编语言程序设计”是高等院校计算机、自动化等专业学生的必修课程之一。它不仅是计算机原理、操作系统等其它课程的必要先修课程，而且对于训练学生掌握计算机工作原理、程序设计技术、熟练上机操作和程序调试技术有着重要作用。

汇编语言是运行速度最快、编程能力最强的语言，同时也是较难掌握的语言之一。但是计算机厂商为了便于用户掌握和使用方便，特地为用户开发了很多底层控制子程序（也叫软中断调用口）。这使用户可以方便地利用这些软中断调用口去控制计算机的各个部件，同时也使汇编语言的学习变得容易一些。

如何学习汇编语言：

学习汇编语言的首要工作是，了解数据的表示方法和基本的硬件结构，因此本书将这两部分安排在第一和第二章说明。另外，我们在第三章说明了编写、汇编、连接、执行汇编语言的基础知识，有了这些概念后，就开始了汇编语言程序设计。从第四章到第九章，我们分门别类地将常用的指令做了说明，学会了指令，那也只是一个基础，学习汇编语言最终目的是能灵活地利用系统资源 DOS 和 BIOS，以达到用软件取代硬件之目的。因此，从第十章开始，我们将常用的 DOS 和 BIOS 功能调用做了综合性的简介。限于本书课时的安排，功能调用的内容不能详细展开，只将原理做了说明。

通过程序实例来掌握汇编语言，这是学习汇编语言的一种好方法，通过实验来验证具体实例，这是学习汇编语言的捷径。同时，通过实验也加深了对所学内容的理解，巩固了所学的知识。对于汇编语言的上机实验，编者非常强调，尤其对于初学者。希望读者掌握且能灵活使用 DEBUG 调试程序。因为对于较大一些的程序没有 DEBUG 跟踪几乎是不可想象的，不论是你自己编写程序，还是读别人的程序。本书是在 MASM 6.0 宏汇编的环境下所做的实验，全部采用了简化段定义的源程序格式，这对于初学者非常有益——简洁明了，每个实例都通过了实验。书中的源程序不是在 WORD 下打印的，都是直接从源程序中拷贝过来的，这就避免一些不必要的错误，分号“；”后表示的是注释，由于时间的关系，个别的没有写注释，较难的程序还写了流程图，这对于程序的理解是有帮助的。

本书第一章由邹国红编写，第二章由冯艳君编写；第四章、第五章、第七章由王罡编写；第六章、第八章、第十三章由张林家编写；第三章、第九章、第十章、第十一章、第十二章、第十四章及实验部分由刘臣奇编写，全书由刘臣奇统稿。

本书的初稿曾多次在院内使用，电子教案也曾获得 1 等奖。这次对原稿进行了再次的修正，在编写过程中，得到了杨福义老师的大力支持与帮助，在此表示衷心感谢，书中个别习题摘自于互联网，作者不详，在此一并对他们表示感谢。

编者在编写本书的过程中尽了很大的努力，但限于水平，错误及不妥之处在所难免，恳请读者批评指正。

编　者

2005 年 2 月

内 容 简 介

本书为汇编语言程序设计教材。本书是以 Intel 80X86/Pentium 系列微处理器为背景，系统地阐述了 PC 机汇编语言程序设计的方法和技术。

全书共分十四章。第一章为基础知识部分，包括数制、码制、基本数据类型等基础知识；第二章为微型处理机的内部结构部分，包括微型处理机的组织结构及功能结构，以及寻找操作数的寻址方式等基本知识；第三章为编写、汇编、连接以及执行汇编语言的基础知识，包括伪操作和汇编语言的格式等基础知识；第四章为数据的传送，包括常用的传送数据指令；第五章为简单的四则运算，包括汇编语言的加，减，乘、除运算以及这些运算的限制；第六章为分支与循环，包括程序设计的基本结构以及跳转指令等；第七章为位运算指令，包括逻辑、移位及大小循环等运算指令；第八章为字串指令及其应用；第九章为子程序和宏调用及其应用，包括子程序和宏程序的作用、定义的格式及其异同等；第十章~第十四章为中断及其系统资源的调用包括基本字符的输入与输出、显示系统与 BIOS 功能调用、键盘的 BIOS 服务例程及其磁盘目录及文件管理系统等。

本书各章配有丰富的习题，在书的最后编者做了详细的上机实验指导。

本书不仅可供高等院校及大、中专院校作为“汇编语言程序设计”课程的教材使用，同时也适合于成教及高职的学生使用。

本书言简意赅，循序渐进，示例丰富，特别适合于初学者自学使用，只要稍有一点 PC 或汇编语言知识的人，就可以通过本书学习掌握汇编语言程序设计技术。

目　录

上机实验教程

第一章　汇编语言基础知识

本章学习目标

1：使读者了解汇编语言的特点

2：使读者熟悉基本数字，非数字（字符）的数据表示法。

3：读者可了解如何进行各数字系统的转换。

本章内容

1-1：汇编语言概述

1-2：二进制系统

1-3：十六进制系统

1-4：二进码十进制（BCD）系统

1-5：字符数据表示法

有基础的读者，可以跳过本章知识，继续下一章内容。

1-1：　汇编语言概述

1．1．1　　汇编语言的基本概念

汇编语言是机器语言的符号表示形式。在汇编语言出现之前，计算机使用机器语言来控制计算机的各种动作。所谓机器语言，就是用 0 和 1 所组成的一串二进制数所表示的命令或数据，机器的硬件可以直接识别和执行，不需要进行翻译。机器语言的特点是命令代码效率高，但不容易记忆，不利于推广和使用；程序员借助机器语言编程时，要用数值表示指令和地址，不但费时费力，而且容易出错。汇编语言将机器语言指令和地址符号化，程序员只需要记住符号名并用其编程，汇编器负责把汇编程序翻译成机器指令和正确的地址数值。

大多数程序应该也可以用高级语言编写，但是在某些情况下汇编语言也是必不可少的。汇编语言可以访问计算机的所有指令，可以利用计算机的所有特性，而高级语言就没有这样的能力，高级语言能做的事情汇编语言都能做，而汇编语言能做的，高级语言不一定能做到。例如，如果计算机有一个溢出位，汇编程序可以测试它，而一个 JAVA 程序就不能直接测试溢出位。汇编语言可以直接访问寄存器，而高级语言则不一定能访问。

1．1．2　　汇编语言的特点

汇编语言相对机器语言而言好记好用，但远不如高级语言方便、实用，而且编写同样的程序，使用汇编语言比使用高级语言花费的时间更多，调试和维护更困难。既然如此，为什么还要使用汇编语言呢？主要有两个原因：性能和对计算机的完全控制。一般而言，汇编语言具有如下特点：

1．执行速度快。一个汇编语言程序，要比高级语言程序执行得更快。速度对于某些应用来说是至关重要的。对于这些应用，单纯使用高级语言往往达不到要求，单纯使用汇编语言编写程序也并不是最好的方案，许多成功的大型应用程序往往使用的是混合编程。首先使用高级语言编写整个程序，然后测试程序的执行时间，再使用汇编语言重写其中最费时的部分。这样做的依据是在实际使用中，通常程序的大部分执行时间都花费在一小部分代码上。

2．程序短小。一个汇编语言程序，要比高级语言程序更小。在某些情况下，设备中的嵌入式处理器往往只有很少的内存，使用汇编语言可能是惟一的方法。如智能卡中有 CPU，但很难有 1MB 以上的内存，也不可能有带分页的硬盘。

3．可以直接控制硬件。某些应用程序要求能够完全控制计算机硬件，这也必须使用汇编语言。如操作系统中的低级中断和陷阱处理程序，以及许多嵌入式实时系统中的设备控制程序都属于这一类应用。

4．可以方便地编译。编译器可以产生供编程者使用的汇编程序或者自己执行汇编过程。因此，为了理解编译器的工作原理，必须首先理解汇编语言。

5．辅助计算机工作者掌握计算机体系结构。研究汇编语言可以使人们清楚地了解，实际计算机的体系结构，特别是对于学习计算机体系结构的学生，编写汇编语言是在结构层上理解计算机的惟一途径。

1-2： 二进制系统

由于计算机内部采用两种状态的装置，因而使得它的结构较为简单，而且它的可靠性也大为提高。通常我们使用0与1来代表这二种状态，如果我们想要表示一个较大的数字，只需将多个0或1串接起来就可以了。

每一个单一的0或1称为一个位（bit）。位是计算机数据的最小单位，虽然一个位不能提供很多信息，但是若干位串接起来，所能做的事情却令人惊讶！，n位即可形成2^n个不同的数字，每多加一位，就使得可形成的数字多增加一倍。

在计算机内部，任何数据，如字母、数字、标点符号等都是利用这些位串来代表。而同样位字串，我们根据不同的数字格式来解释，可以获得不同的结果。

一位若是在ON状态其值为1，若在是OFF状态其值为0。这看起来好像是个限制，正如十进制只能代表0到9的数位。但是，若你考虑合并十进制数字去形成大于9的数，你也可以合并二进制数去形成大于1的数，因此这就等于没有限制，且可以表示任何大小的数了。

而一个二进制数字的数值是根据每个位的“相对位置”和“1”位存在的位置而定的。下面的8位数含有8个1：

位　　：　1　1　1　1　1　1　1　1

位对应值：　128　64　32　16　8　4　2　1

最右边的位代表数值1，而它左边的下一个位代表2，再下一位则代表4，依此类推，最后一个位代表128。以此数为例，这些位的总数值为1+2+4+……+128=255。

对于一个二进制数01000001，其值为1加上64，所以是65。

在计算机的专有名词中，8位（bit）的数据单位称为一个字节（byte），使用8位，即一个字节可以表示由0到255个数字。因为字节是最基本的数据处理单位，所以计算机的存贮容量均以字节数来表示。

2^{10}等于1024，1024值通常用字母“K”来表示，因此一计算机有256K的存贮容量，即表示此计算机有256×1024（或262144）字节的存贮容量。2^{20}等于1024×1024，此值通常用字母“M”来表示，一计算机容量是1M即表示此计算机有1024×1024字节存贮容量。

1．2．1 二进制的算术运算

计算机是以二进制方式运算的，所以作为汇编语言程序员必须熟悉二进制格式与算术运算。下面是四种二进制加法的基本格式：

0+0=0

0+1=1

1+0=1

1+1=0　　进位1

对于以上二进制格式了解了以后，我们就可真正处理二进制 1000001 与 00101010 的相加了。（若以 10 进制的观点而言，这是 65 和 42 的相加）。

二进制		十进制
01000001		65
+ 00101010	+	42
01101011		107

1．2．2　带符号的数字

到目前为止，我们所讨论的都是二进制无符号的数，因此若一字节所有 8 位皆为 0，该值为 0，若全部是 1，该字节的值为 255。

但是当一个字节包含正负号时，则只有 7 位有效位（从 0 到 6）表示数据，最高有效位（位 7，又称符号位）表示该数的正负号，若该数大于或等于零，则符号位为 0，若该数小于零，则符号位是 1。图 1-2 有符号数或无符号数的数字表示方法：

计算机中的数字类型（8 位）		7	6	5	4	3	2	1	0	数的范围
无符号：　使用所有的位数来表示数字										0~255d
有符号	+数：使用较低的 7 位来表示数字	0								–128d~+127d
	–数：使用 2 的补码方法表示数字	1								

图 1-1　有符号数或无符号数的数字表示方法

在计算机系统中，所有表示负数的方法均采用 2 的补码来表示。也就是说，要将一个二进制数字表示成负数，必须把所有位数取反（即 0 变 1，1 变 0）并将取反后的结果加上 1。现以 65 来做说明：

数值 65 二进制表示：　01000001

取反：　10111110

将取反结果加 1：　10111111　（等于–65）

若要算出一个负的二进制数字的绝对值，只要重复上述的过程即可。

数值-65 二进制表示方式：　10111111

取反：　01000000

将取反结果加 1：　01000001　（等于+65）

若将+65 与-65 相加，所得结果应为 0，如下所示：

	01000001	（+65）
+	10111111	（–65）
有进位 1	00000000	

另一常用运算是减法运算。其实二进制的减法也是一件简单的事：只要将减数的正负号取反，然后将二数相加就可以了。现我们举 65 减去 42 的例子来做说明：

二进制		十进制
01000001		65
+ 11010110	+	-42
00010111		23

由上可知，所得结果 23 是正确的。

图 1-2 是 4 位带正负号数字的表示法，从此图可知 4 位可代表-8~+7 间所有整数。若将

其扩充，可知 1 字节可代表-128~+127。

二进制	十进制		二进制	十进制
0000	0		1000	-8
0001	1		1001	-7
0010	2		1010	-6
0011	3		1011	-5
0100	4		1100	-4
0101	5		1101	-3
0110	6		1110	-2
0111	7		1111	-1

图 1-2　4 位有符号数字的表示法。

1-3：　十六进制系统

对计算机而言，二进制算术运算相当好，但是对人类而言，需要较简捷的表示方法，于是我们便采用了十六进制的表示法。

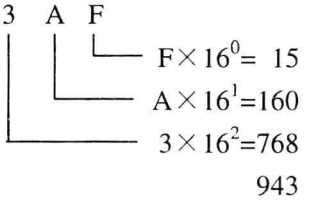
十六进制表示法是以 16 为底的数字系统，其中前面 10 个以 0 到 9 表示，最后 6 个则以 A 到 F 表示。图 1-3 是十六进制的数字表示法。

二进制	十进制	十六进制		二进制	十进制	十六进制
0000	0	0		1000	8	8
0001	1	1		1001	9	9
0010	2	2		1010	10	A
0011	3	3		1011	11	B
0100	4	4		1100	12	C
0101	5	5		1101	13	D
0110	6	6		1110	14	E
0111	7	7		1111	15	F

图 1-3　十六进制数字的表示法

由于十六进制系统是以 16 为基底，所以所有数位皆是其右邻数位的 16 倍。即最右边数位值为 16^0，其次为 16^1，其他依此类推。

例如：求十六进制的 3AF 之十进制值=943，其计算过程如下：



943

一般高级语言通常使用十进制来表示数字，汇编语言则使用十六进制来表示数据。例如，存贮单元的地址和内容、寄存器、指令码等。因此刚开始使用十六进制来设计程序时会有一点困难，但习惯以后就会变得很简单。

1-4： 二进制码十进制 BCD 系统

BCD 格式（Binary Code decimal）是计算机系统另一种数字的表示法，这种格式对于处理十进制小数点数字非常好用。

所谓 BCD 格式就是用四个二进制位，来表示一个十进制数字，这也是为什么它会被称为“二进制码十进制”。BCD 格式又可分为压缩式和非压缩式两种格式。

压缩式 BCD 格式，就是以一连串四位为一组的二进制位，来代表一个十进制数字。例如，十进制的 9502 可以写成：

```
1001  0101  0000  0010
 ↑     ↑     ↑     ↑
 9     5     0     2
```

至于非压缩 BCD 格式，每一个十进制数字，则是存放在字节的低位部分，至于高位部分存放什么东西就不重要了。因此 9502 可以表示成：

```
uuuu1001  uuuu0101  uuuu0000  uuuu0010   （U 表示我们并不关心它所放的内容）
  ↑         ↑         ↑         ↑
  9         5         0         2
```

储存或执行 BCD 运算时须特别小心，因计算机并不知道这个二进制代码是一 BCD 数字。因此要用 BCD 数字运算，尤其是算术运算，程序员须特别多做一些笔记工作。

范例 1：将$(67.9)_{10}$转换成 BCD 码。其过程如下：

```
  6     7   .   9
0110  0111  .  1001      所以（67.9）10=（01100111.1001）BCD
```

范例 2：将（10010110.0110）$_{BCD}$转换成十进制数。其过程如下：

```
1001  0110  .  0110
 9     6    .   6      所以（10010110.0110）BCD=（96.6）10
```

BCD 码的运算规则：运算器对数据做加减运算时，都是按二进制运算规则进行处理的。这样，当将 BCD 码传送给运算器进行运算时，其结果需要修正。修正规则如下：

当两个 BCD 码相加，如果和等于或小于 1001，即 9H，则不需要修正；如果相加之和在 1010 到 1111 即 0AH～0FH 之间，则需加 6H 进行修正；如果相加时本位产生了进位，则需加 6H 加以修正。这样做的原因是：机器按二进制相加，所以四位二进制相加时，是按“逢十六进一”的原则进行运算的，而实质上是两个十进制数相加，应该按“逢十进一”的原则相加，16 与 10 相差 6，所以当和超过 9 或有进位时，都要加 6 进行修正。

范例 3： 计算 1+8，其运算过程：

```
  0001
+ 1000
------
  1001   即结果是 1001=9D，1 + 8 = 9 结论正确。
```

范例 4： 计算 5 +7，其运算过程：

```
  0101
+ 0111
------
  1100   结果>9
+ 0110   加 6 修正
------
←1 0010  结果是 0010=2D，还产生了进位，即 5 + 7 = 12，结论正确。
```

进位

范例 5：　　计算 9+9，其运算过程：

```
    1001
  + 1001
  ------
  1 0010   产生了进位
  + 0110   加 6 修正
  ------
进位 ←1 1000  结果=8D，由于有进位，故+6 修正。9+9 =18，结论正确。
```

当两个 BCD 码相减，如果差等于或小于 1001，不需要修正；如果相减时本位产生了借位，则应减 6H 加以修正。原因是：如果有借位，机器将这个借位当十六看待，而实际上应该当十看待，因此，应该将差值再减 6H 才是 BCD 码的正确结果值。

1-5：　字符数据表示法

所谓字符指的是英文字母、十进制数字、标点符号以及一些特殊符号的表示。计算机界常用的编码方式为 ASCII 码，即美国标准信息交换代码（American Standard Code for Information Inter chance），如图 1-4 所示。

$B_6B_5B_4$ / $B_3B_2B_1B_0$	000 (0)	001 (1)	010 (2)	011 (3)	100 (4)	101 (5)	110 (6)	111 (7)
0000（0）	NUL	DEL	SP	0	@	P	`	P
0001（1）	SOH	DC1	!	1	A	Q	a	q
0010（2）	STX	DC2	“	2	B	R	b	r
0011（3）	ETX	DC3	#	3	C	S	c	s
0100（4）	EOT	DC4	$	4	D	T	d	t
0101（5）	ENQ	NAK	%	5	E	U	e	u
0110（6）	ACK	SYN	&	6	F	V	f	v
0111（7）	BEL	ETB	‘	7	G	W	g	w
1000（8）	BS	CAN	(	8	H	X	h	x
1001（9）	HT	EM	)	9	I	Y	i	y
1010（A）	LF	SUB	*	:	J	Z	j	z
1011（B）	VT	ESC	+	;	K	[	k	{
1100（C）	FF	FS	,	<	L	\	l	|
1101（D）	CR	GS	-	=	M	]	m	}
1110（E）	SO	RS	.	>	N	^	n	~
1111（F）	SI	US	/	?	O	_	o	DEL

图 1-4　ASCII 字符编码

ASCII 码表有以下几个特点：

1．每个字符用 7 位基 2 码表示，其排列次序为 $B_6\,B_5\,B_4\,B_3\,B_2\,B_1\,B_0$。实际上，在计算机内部，每个字符是用 8 位（即一个字节）表示的。一般情况下，将最高位置为“0”，即 B7 为“0”。需要奇偶校验时，最高位用做校验位。

2．ASCII 码共编码了 128 个字符，它们分别是：

①32 个控制字符。用于通信中的通信控制或对计算机设备的功能控制，编码值为 0～31D。

②间隔字符（也称空格字符）SP，编码值为 20H。

③删除控制码 DEL，编码值为 7FH。

④94 个可印刷字符（或称有形字符）。这 94 个可印刷字符编码有如下规律：

字符 0～9 这 10 个数字符的高 3 位编码都为 011，低 4 位为 0000～1001，屏蔽高 3 位的值，低 4 位正好是数据 0～9 的二进制形式。这样编码的好处是既满足正常的数值排序关系，又有利于 ASCII 与二进制之间的转换。英文字母的编码值满足 A～Z 或 a～z 正常的字母排序关系。另外，大小写英文字母编码仅是 B5 位值不相同，B5 为 1 是小写字母，这样编码有利于大小写字母之间的编码转换。

计算机中字符的传送就是借助 ASCII 码和外界作勾通的。例如，当我们在键盘上按下一个键时，这个键所代表的 ASCII 码就会传到计算机内。在此请读者注意的是，并非所有的 ASCII 字符都可以打印出来的。比如空一格、退格、换行、回车等。

例如，键入下面字符串：

DOE,

JOHN P.-50

相当与下面位的组合（用 16 进制表示）：

44	4F	45	2C	0D	0A	4A	4F	48	4E	20	50	2E	2D	35	30
D	O	E	,			J	O	H	N		P	.	-	5	0
				↑	↑					↑					
				回	换					空					
				车	行					格					

本章小节

本章介绍了汇编语言的基础知识——数码及二、十、十六进制之间的转换，此部分内容读者是应该掌握的。计算机内存储的数字以及各种编码全部是二进制的，而为了书写的方便，才使用了十六进制和八进制数。

第一章课外练习题

一、单项选择题：

1．从键盘输入的字符，在计算机内存储的是它的（　　）

（A）二进制编码　　（B）四进制编码（C）八进制编码（D）十六进制编码

2．6 位无符号二进制数能表示的最大十进制数是（　　）。

（A） 64　　（B） 63　　（C） 32　　（D） 31

3．十进制数 269 变换为十六进制数是（　　）。

（A）10B　　（B）10C　　（C）10D　　（D）10E

4．8 位的微型计算机系统是以 16 位来表示地址，则该微机系统有（　）个地址空间。

（A）255　　（B）65535　　（C）65536　　（D）1048576

5．8 位有符号二进制数能表示的最大十进制数是（　　）。

（A）256　　（B）255　　（C）128　　（D）127

6．十六进制数 88H，可表示成下面几种形式，请找出错误的表示（　　）。

（A）无符号十进制数 136　　（B）带符号十进制数－120

（C）压缩型 BCD 码十进制数 88　　（D）8 位二进制数－8 的补码表示

7．有一个数值 152，它与十六进制数 6A 相等，那么该数值是（　　）。

（A）二进制数　　（B）八进制数　　（C）十进制数　　（D）四进制数

8．7 位 ASCII 码总共可表示（　　）个符号

（A）256　　（B）127　　（C）128　　（D）255

9．4B 的字长是（　　）

（A）8 位　　（B）16 位　　（C）32 位　　（D）64 位

二、判断题（判断每题正误，对的在题后括号内划 “√”，错的划 “×”）

1．字节通常用英文单词 “Bit”来表示　　（　）。

2．目前广泛使用的 Pentium 计算机其字长为 5 个字节　　（　）。

3．存储器中将 8 个相邻的二进制位作为一个单位，这种单位称为字节　　（　）。

4．微型计算机的字长并不一定是字节的整数倍　　（　）。

三、填空题

1．8 位有/无符号整数的表示范围写成 16 进制形式为（　　）/（　　）。

2．已知：计算机中有一个 “01100001” 编码，如果把它看作是无符号数，它是十进制什么数（　　）；如果认为他是 BCD 码，则表示（　　）$_{BCD}$；认为它是某个 ASCII 码，则代表（　　）字符。

3．若[X]补=X，则 X 为 （　　）。

4．（00101011）B+（13）D+（1B）H=（　　）Q

5．已知：A=10001011，B=11011010，求 A∧B=（　　）；A∨B=（　　）。

6．已知：X=+0010011B，Y= -0111100B，试用补码完成真值（X+Y）=（　　）；（X - Y）=（　　）。

7．十进制数的（731）=（　　）BCD 码。

8．（011100110001）BCD 码=（　　）十进制数。

9．将一个 0~9 之间的数据转换为 ASCII 需加（　　）。

10．大写英文字母转换为小写英文字母需加（　　）。

11．英文字母 “A” ~ “F” 转换为十六进制数值 A~F 需要减（　　）。

四、简答题

1．什么是汇编语言？为什么要学习汇编语言？

3．把下列二进制数转换成十进制数。

① 1101.01　　② 111001.00011

③ 111.001　　④ 1010.1

5．把下列十六进制数转换成十进制数。

① A6.DC　　② 9AC.BD

③ B4A.8D　　④ 1AC.0A

6．把下列英文单词转换成 ASCII 编码的字符串。

① Water　　② Great　　③ Good　　④ After

7．回车键、空格键的 ASCII 代码及其功能是什么？

第二章　微型处理器的结构及寻址方式

本章学习目标

1：读者可通过本章获得编写汇编语言所需的硬件知识。

2：读者可了解如何使用 16 位寄存器来存取 1 兆字节（1MB）的内存空间。

3：读者可从本章获得堆栈的基本概念。

4：读者可从本章获得编程中数据的基本寻址方法

本章内容：

2-1：Intel CPU 的演化。

2-2：8086 / 8088 的内部结构。

2-3：存储器组织与分段

2-4：段的概念

2-5：堆栈的概念

2-6：8086 / 8088 数据寻址方式

现代的高级语言，由于大多具有灵活的数据结构和数据处理能力，因此在大多数情况下，对于问题的解决（problem Solving），高级语言有较大的优势。

但是当希望直接利用程序与计算机内部元件勾通，或需要精确的控制和高效能的场合时，那就需要汇编语言了。例如：设计编译程序（compiler），操作系统各个实用程序的设计（utility program），各种 I / O 控制等。此外若想更进一步增加计算机科学知识的深度与广度时，也必须具备汇编语言的知识。

由于汇编语言需要你会操作计算机内部的组件。因此，你必须懂得这些内部组件的特性和能力，即对计算机的微处理器有所了解。

2-1：　Intel CPU 的演化

过去的 50 多年，计算机的发展有了很大的进步，特别是以微处理器为基础的计算机，已从性能很差的不重要的系统，而发展成今日快速且功能繁多的系统。

最早期的微处理器是 1971 年 Intel 的 4004，这是 4 位微处理器，即表示每次它可传送 4 位数据，当要传送多位数据时，它必须执行多次的操作。

1972 年 Intel 公司推出了 8008 微处理器，这是第一种上市的 8 位微处理器。除了数据处理速度较快之外，与 4004 微处理器相比，8008 有如下优点：能选取较大的内存空间，具有堆栈（stack）及单级中断能力。

这些芯片又称为是第一代的微处理器，且都是为了某些特定应用而设计的。例如：4004 被应用于计算器（calculator），8008 则被应用于终端（terminal）。起初，这些微处理器只是被当作新奇产品而未受到重视。

直到 1974 年，Intel 公司推出了第二代的 8008，称为 8080。这时候的微处理器已具有老式大型计算机的运算功能了，又因其价格低廉，使它立刻成为计算机的宠物，也就在此时微处理器的潜力才受到大家的重视。

到了 1977 年，技术的进步，使得 Intel 公司可以制造 8080 的改良品种 8085。新改良的 8085 不仅本身具有串行输入输出能力，同时又有电源复位能力（Power-on reset，重新启动机器），向量中断以服务外围设备的请求，以及单一正 5 伏特的电源（8080 需要 2 个电源供应）。