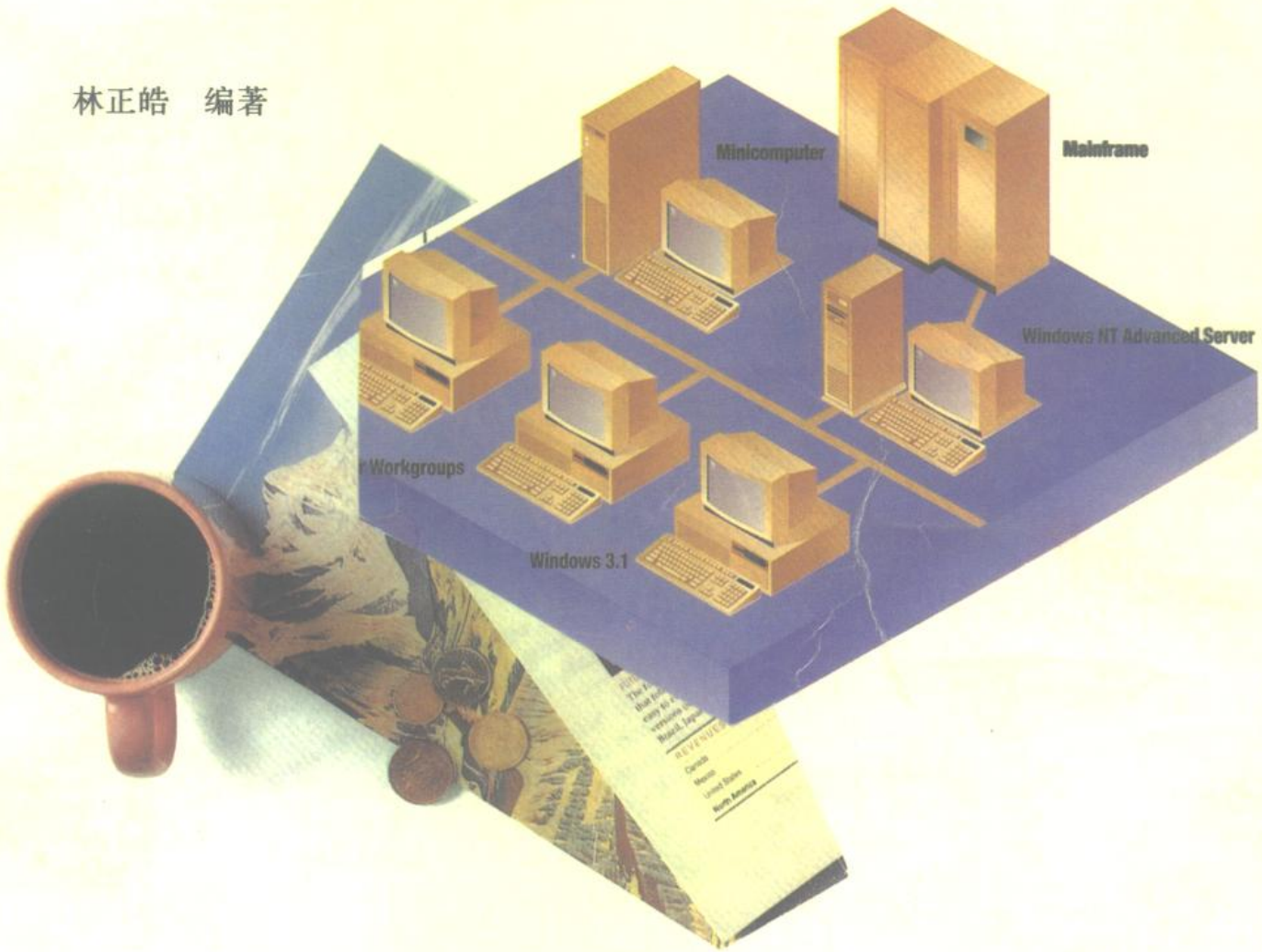
实用汇编语言入门与编程技巧

林正皓　编著



海洋出版社

实用汇编语言
入门与编程技巧

林正皓　编著
曹　康　改编
陆　明　审校

海洋出版社

1996年·北京

内 容 简 介

汇编语言是介于高级语言与机器语言之间的一种比较接近电脑硬件操作的语言。本书针对汇编语言的特点，集中论述了一些学习汇编语言必然会遇到的难点，着重讲解了汇编语言的程序开发过程和指令使用技巧。全书共分十二章，每章内容充实，章节之间联系紧密，尤其是前面的三章和最后一章是本书的精华所在，作者将丰富的经验和教学心得溶入书中，深入浅出，资料新颖，实例典型，是一本难得的学习汇编语言的辅导教材。

版 权 声 明

实用汇编语言入门与编程技巧

林正皓 编著
曹 康 改编
陆 明 审校

* * *

海洋出版社出版(北京复兴门外大街1号)
海洋出版社发行 经贸大学印刷厂印刷
开本：787×1092毫米 1/16 印张:33.5 字数：782千字
1994年3月第一版 1996年5月第二次印刷
印数：1—5000册 定价：39.00元
ISBN7－5027－3836－3/TP・238

前　言

在所有电脑语言中，不可否认的，汇编语言的确是有点不太好学，但是不好学是不是就代表它已无学习价值呢？当然答案是否定的。就目前读者所熟知的高级语言来讲（如：BASIC，PASCAL，COBOL……等等），不管您是采用那一种高级语言来开发应用程序，在设计程序时，您可能会碰到下面这几个问题而无法克服：

1. 欲完成某项特殊的工作，而语言本身并无相对应的指令可用。
2. 虽然有指令可用，但受限于指令的使用格式，而无法运用自如。
3. 程序执行的速度及效率，受制于支撑软件的系统程序，而无法提高。
4. 无法随心所欲地对电脑的周围设备加以运用及控制。
5. 电脑系统操作所需的相关知识，语言本身介绍不多。

由于有了上述这些问题的产生，您所设计出来的应用程序，在使用时或多或少会觉得有点不尽人意。因此为了让您的应用程序更具有专业化，使用时的亲和力能够提高，那学会第二种电脑语言来解决上述的那些问题，就显得格外迫切了。至于第二种电脑语言的选择，因为一般高级语言所受的限制几乎都差不多（一样无法解决上述的那些问题）。因此汇编语言就成了唯一的选择，选用汇编语言当作第二种电脑语言来学习，上述的那些问题就可以解决了。因为就汇编语言本身的特性来讲，它是属于一种比较接近电脑硬件操作的语言，凡是电脑所做的工作，我们都可以利用汇编语言指令来加以控制，因此就功能上来讲，汇编语言是所有电脑语言中最强的一个。既然功能最强，它就有学习的价值了。

然而初学的过程毕竟是辛苦的，当笔者刚接触到汇编语言时，也是和一般初学汇编语言的读者一样，程序一执行就碰到电脑死机或是莫明奇妙的问题发生，产生挫折感（这二个问题又是学习汇编语言的常客），于是会产生疑问："在高级语言可顺利解决的问题，为何用了汇编语言就变得那么麻烦呢"？因此在这里笔者想强调的一个观点就是："学汇编语言一定要胆大心细，所谓胆大就是只要电脑可以做的工作，您都可以叫它去做，所谓心细就是指使用汇编语言指令时，一定要去了解这个指令执行之前与执行之后它所能影响的范围，只要把握这二个原则，那我们就可以快快乐乐地来学汇编语言了。

本书是以入门为主，着重在于汇编语言的程序开发过程与指令的介绍，尤其是前面的三章与最后一章可说是本书的精华所在，笔者把本身的学习经验与教学心得溶入书中，配以实作的方式来引导，这样的安排可以减少读者在学习时花在自行摸索上的时间（这也正是笔者请读者不要惧怕的原因所在）。

本书承蒙松岗杨嘉水先生应允出版，谨此致谢。还有本 DAC R&D 小组邓文渊、邱文谅、黄连进、蔡思南的鼓励与协助，尤其小组的核心邓文渊的精心策划，本书才得以顺利完成，在此致以最大的谢意。

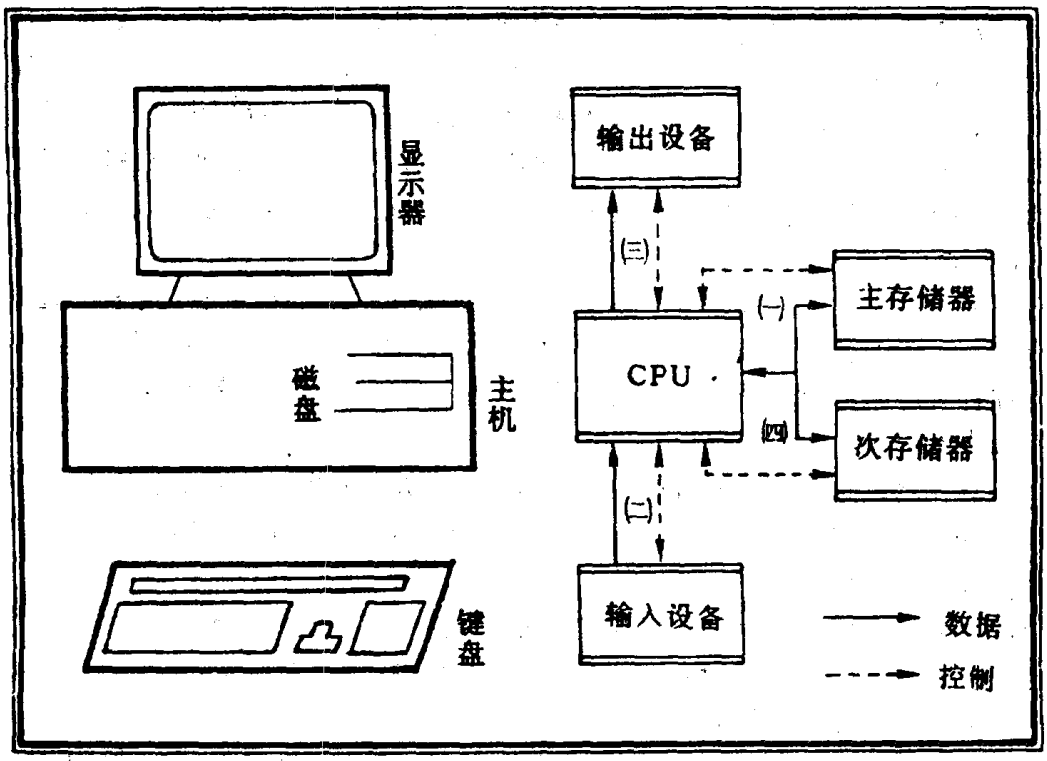
笔者学识菲薄，虽书中所述力校再三，但错误仍恐难免，还期盼您的不吝指正，谢谢。

林正皓　谨序
于台湾埔里

目　录

第一章　学习汇编语言必备的相关知识

第一节　电脑如何工作



说明：

1. 上图左边为电脑硬件设备的实体图，右边为电脑工作时的系统方块图。二者的对应关系如下：

显示器：输出设备

主　机：中央处理单元、内存、外存

键　盘：输入设备

中央处理单元及内存位于主机内部的主机板上(所以看不到)而外存指的就是使用者放进驱动器内的磁盘(包括硬盘在内)。

2. 由左页的系统方块图可知，一般电脑的硬件设备若依功能来分类的话，可分成五大单元，即中央处理单元、内存、外存、输入设备、输出设备。

3. 中央处理单元(Central Processing Unit 简称 CPU)可说是这五大单元中最重要的一个，因为所有我们要交给电脑做的工作都是要靠 CPU 来执行，而内存(Main Memory)主要是用来储存电脑将要执行的命令(含数据)的地方，换句话说就是 CPU 工作时，我们事先就要把命令或数据先储存到内存内，电脑才有办法工作(就像 BASIC 语言必须先 LOAD 才能 RUN 道理是一样的)。

4. 如上页电脑依数据进出方向的不同，可有下列四种工作模式：

（一）执行命令模式：主要是CPU与内存之间数据的进出。当CPU要执行命令时，首先它会送出一个地址给内存（即命令之所在），内存就依所送来的地址将该地址上的内容（即命令）利用原来的地址线路反送命令至CPU内。CPU在接受命令后会先判断此命令（即对命令进行解码），到底是要做什么样的工作，而通常内存在储存程序命令时，在命令之后紧接着需要所要处理的数据或储存该数据的地址，因此CPU再送出数据的所在地址，以便从内存内取得所要处理的数据，然后依命令来完成工作（例如将某一个存储器变量的内容搬到CPU内）在CPU处理完数据后，如果需要将结果存回内存的话（例如执行A＝A＋1，其中A为某个内存变量），CPU再送出储存的地址，以通知内存将结果存到所指定的地址上，等存完之后CPU再送出下一命令的所在地址给内存，以便CPU能够继续往下执行命令。

（二）输入数据模式：（例如执行BASIC语言的INPUT指令）主要是用户通过输入设备将数据送进CPU，再由CPU转存至内存内，以方便CPU往后能处理这些数据。而输入设备必须具有将数据由用户看得懂的形式（例如文字、数字、符号等）转变成电脑看得懂（电子信号）的能力。

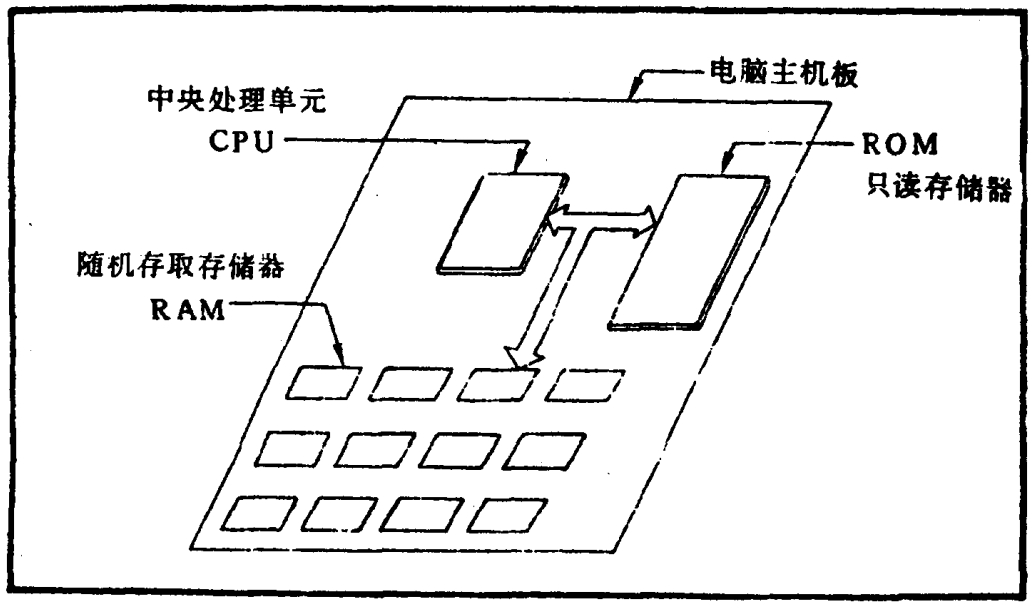
（三）输出数据模式：（例如执行BASIC语言的PRINT指令），主要是CPU从内存内取出数据，然后送到输出设备上。而输出设备必须具有将数据由电脑看得懂的形式转变成使用户看得懂的能力。

（四）外存内数据进出模式：主要是通过驱动器将磁盘上的数据读进内存或将主内存内某段数据写到磁盘上，而此时CPU只是站在协调与控制的立场来监督数据的进出。所谓外存实际上指的就是用户放进驱动器内的磁盘片，而此磁盘片实际上也算是内存的一种，可以作为数据储存的地方，不过它与内存除了在结构上不同之外，对于所储存的数据在处理的时间上也是不同的，一般放进内存内的数据（含命令）是电脑马上就要处理的，不急着处理的数据就放在磁盘片上（外存），等要处理时再放进内存内。（如果不考虑数据的转换问题（A↔电脑），也是可以将驱动器看成兼具输入及输出的设备）

通常我们所设计的应用程序（不管是用何种语言来编与），都是经过上述的四种工作模式来逐步累积完成的。（不断循环重复一直到工作结束）

在电脑的四种工作模式中，内存一直接演着非常重要的角色，没有它电脑几乎停止工作（与CPU同等重要），接下来我们就要研究内存的结构及其在学习汇编语言时如何来表示它。

第二节　内存



说明：

数据一定要有地方(即内存)才能储存，且空间愈大(即地址愈多)储存的数据就愈多，因此由上图我们可以看出，内存单元分散于整个电脑的主机板上，且依材料结构及储存数据方式的不同，内存可由随机存取内存及只读内存二种组合而成。

随机存取内存：(Random Acess Memroy 简称 RAM)是一种兼具读出数据及写入数据两种功能的内存，一般我们所设计的应用程序就是先写入(即储存)到 RMA，CPU 再从 RMA 里面读出来执行。此种内存在使用上有二个基本特性必须要记住，一为储存在某个地址上的内容随时可以被更改(换句话说就是在执行写入动作时，所写入的数据会覆盖原来的数据，原来的数据消失)，二为把电脑的电源关掉，那储存在 RAM 上面的数据都不见了(也即重新关机后是无法取回原先的数据)。

只读内存(Read Only Memory 简称 ROM)此种内存在安装到电脑主机板上之前，电脑厂商就已经将数据或命令(即程序)储存到里面了(利用特殊仪器来储存)，因此当用户打开电脑的电源，实际上电脑就已经开始在执行命令(即程序已经在内存内，不必有载入的操作，CPU 会自 ROM 内取出命令来执行)，通常储存在 ROM 内的数据并不会因为电脑的开机而消失，再开电源时数据依旧存在，且所储存数据不管在地址或内容上都是固定不变的，不可用任何命令去更改(上述 ROM 的二种特性与 RAM 不同)。

因此从上述二种内存的基本特性，我们就可知道：开机时电脑就已经开始在工作了(即执行程序)，而此程序是储存在内存的 ROM 内(程序由厂商所提供)，CPU 会自 ROM 内取出命令来执行，通常存在 ROM 内的程序其工作内容都是固定不变(因 ROM 的内容不容更改)，但毕竟只执行一个程序是不够的，我们使用电脑的目的就是为了要来解决问题(将问题程序化)，因此我们还须借助输入设备或外存将我们所设计出来的程序，由外部

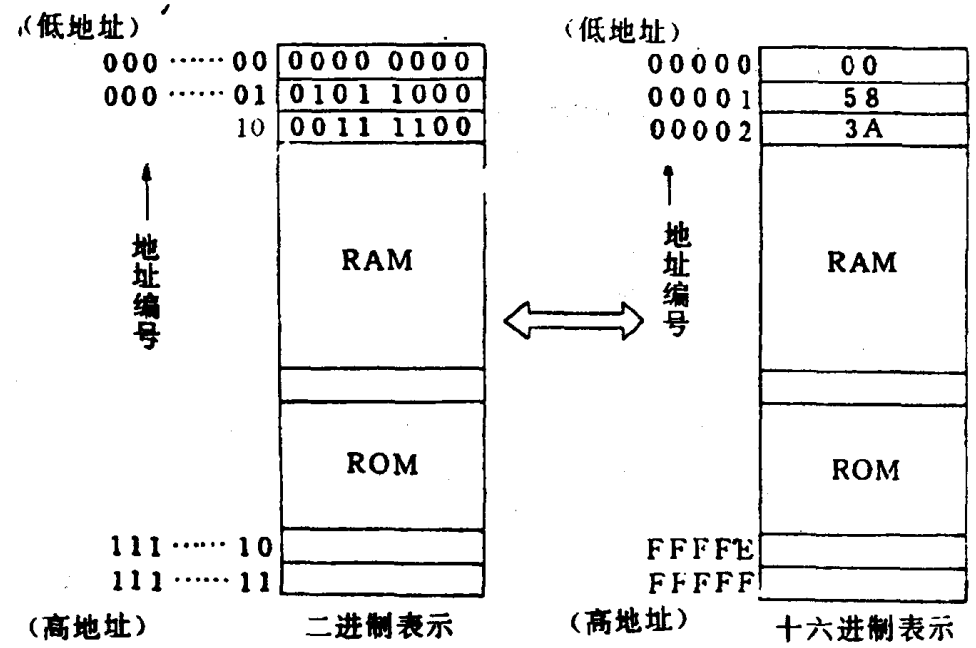
载入到电脑内，让电脑来执行，而此程序就是载入到内存的 RAM 内，利用 RAM 的特性我们就可以重复载入不同的程序到内存内来执行，而达到我们使用电脑的目的。

或许读者会问：厂商所提供的程序到底做了那些事呢？关于这一点，如果读者仔细观察的话，是不难找到答案的，实际上此程序只做二大工作，一为测试，二为载入运行系统程序（也有人称为开机程序）。当您打开电脑的电源（假设电脑设备已安置妥当，该接的线都接好了），首先显示器会出现一些信息（不同厂牌的电脑可能会有不同的显示方式），接着屏幕上有些数字会一直在变化（这就是在测试 RAM 的好坏），然后驱动器动一下，键盘指示灯闪一下，如果接有打印机且打印机的电源是打开的则打印头也会动一下，以上就是执行测试所显示出来的动作，为了确定这些设备可以用，因此才有测试的必要（注意：此时读者不妨将键盘的接头松动一下，让其接触不良，然后再重新打开电源试试看，一定无法顺利开机成功，这是因为无法使用键盘为输入设备的关系）。最后才由 A 驱动器内载入运行系统程序（必须将所谓的系统盘放入），如果 A 驱动器内没有系统启动程序，而系统配设有硬盘的话，那就会转而从 C 硬盘内载入启动系统程序，（换句话说，A 驱动器的载入优先于 C 驱动器），等启动系统程序载入到内存的 RAM 后，厂商所提供的程序其工作才算完成，紧接着电脑就开始执行刚刚所载入的运行系统程序了（CPU 转向自 RAM 内取出命令来执行）。

通常用户在购买电脑时，厂商都会随机附送一张（或二张）所谓的系统软盘（或称开机软盘），在此软盘中会有三个程序（名称为 IBMBIOS. COM、IBMDOS. COM 及 COMMAND. COM）这就是所谓的启动系统程序了，当系统测试完毕后就会将这三个程序载入到内存内，然后开始执行（此运行系统最主要是管理驱动器的操作，因此就称为磁盘操作系统 Disk Operating System 简称 DOS）。

当电脑在执行磁盘操作系统程序的过程中，此程序会先到运行系统程序所在的那个目录上寻找有没有 AUTOEXEC. BAT 这个批处理文件，如果有则执行这个批处理文件内的命令，没有则会寻问用户日期与时间。在执行完批处理文件内的命令或回答完日期与时间后，屏幕才会出现 A>（或 C>），此时才算真正的开机成功，但开机成功并不代表系统程序就已经执行完，实际上屏幕所出现的 A>就是操作系统程序用以揭示用户可以开始输入命令了（类似 BASIC 语言的提示符号 OK 一样），而用户由键盘所输入的命令，都是交给操作系统程序来帮我们执行的（换句话说就是我们所设计出来的程序到最后都是由操作系统程序来完成），等命令执行完屏幕会再度出现 A>，用户又可以再输入命令了，（套用一句程序的术语，运行系统程序是属于一个无穷循环的程序，除非关机，要不然此程序永远在执行）。

在了解了 RAM、ROM 和 DOS 的作用后，接下来我们再来看看在学习汇编语言的过程中，对于内存我们还需要了解些什么呢？请看下图所示：



说明：

1. 实际上内存单元是分散于整个电脑的主机板上，但为了说明方便起见，我们用上面左边的图来表示它(将所有的内存元件集中起来，按其在主机板上的顺序排列，然后赋于一个地址编号)，这样的表示方式，在说明系统的操作上跟实际情形是不会有冲突的。

2. 就目前所谓的个人电脑来讲，实际上其操作就是属于二进制的装置，也就是针对其内部电路的每一条线(或每一个接点)来讲，都是工作在二种状态之下，即有电或是没有电，且随着电脑的运行，每一条线或是每一个接点，只能在这二者之间做变化。通常为了方便表示，有电我们就用 1 来代表，没有电就用 0 来代表，而每一个 0 或 1 我们称为位(Bit，这是表示电脑数据的最小单位)，但必竟用一个 Bit 只能表示出二种不同的数据，很显然这是不够用的，因此我们合并 8 个 Bit(称为字节 Byte)就可以表示 256 种不同的数据了(即 $2^8=256$，其中底数 2 代表每个 Bit 有二种状态的意思)，同理合并 16 个 Bit(称为字 Word)就可表示 65536 种不同的数据(即 $2^{16}=65536$)，因此 Bit、Byte、Word 的关系可用下列等式来表示。

1 Byte＝8 Bit

1 Word＝2 Byte＝16 Bit

3. 从上图中我们可以很清楚地看出，内存是以 8 个 Bit(即 1 个 Byte)为一个单位来编排地址的，换句话说就是内存 1 个地址可以存 8 个 Bit 的内容(即 1 个 Byte)。通常我们所说的文件长度就是以 Byte 为单位，这样当把文件载入到内存时，二者才能配合得天衣

无缝。

4. 当CPU要执行命令时，必须先送出命令的所在地址给内存，这样CPU才有办法取得存在内存内的命令，但因电脑是属于一种二态的装置，因此所送出的地址结构也是由一连串的0和1所组成，上页左图中有关地址的编号即是根据此原理来编排的(地址顺序有高低之分)。

5. 一般我们比较熟悉的数值系统就是十进制(也就是说所使用的数字为0～9，逢10便产生进位)，而电脑内部所表示的数据都是由0和1这二个字所组成，因此我们可以这么说，电脑所使用的是二进制的数值系统(逢2便产生进位)，但毕竟使用一连串的0和1在表示上还相当麻烦，是不是有更简便的表示方式呢？答案是有的，即用十六进制。我们将上页左图中不管是地址编号或地址上的内容都以4个Bit为一个单位，将其转成十六进制后，就可得到右图了(往后有关内存的表示都采用右图)，而其转换换请看下面表格。

十进制	二进制	十六进制
00	0 0 0 0	0
01	0 0 0 1	1
02	0 0 1 0	2
03	0 0 1 1	3
04	0 1 0 0	4
05	0 1 0 1	5
06	0 1 1 0	6
07	0 1 1 1	7
08	1 0 0 0	8
09	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F

上述表格为十六进制的一位数与二进制的四位数二者之间的一个转换表，再用我们所熟悉的十进制来表示其大小，由表中可以看出十六进制的0～9都与十进制的相同，只有在超过拾的部分是用英文的字母顺序来表示(大小写都可以)，因十六进制逢16就产生

进位，因此在F之后的数为10(含成壹零，不可念成拾，以免跟十进制混用)，至于转换的演算规则请看下列的举例示范：

(一) 二进制→十进制

$$\begin{array}{cccc} 1 & 1 & 0 & \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

$1=1\times2^3+1\times2^2+0\times2^1+0\times2^0$

$=1\times8+1\times4+0\times2+1\times1=8+4+0+1$

$=13$

(二) 十进制→二进制

除数 ↓　被除数 ↓　余数 ↓

$$\begin{array}{r|l} 2 & 13 \\ 2 & 6 \quad \cdots\cdots 1 \\ 2 & 3 \quad \cdots\cdots 0 \\ 2 & 1 \quad \cdots\cdots 1 \\ & 0 \quad \cdots\cdots 1 \end{array}$$

利用连除法将十进制的值用2除，一直到商等于0为止，写答案时由最后一个余数往前写，即：

$13=1101$

(三) 十进制→十六进制

$$\begin{array}{r|l} 16 & 760 \\ 16 & 47 \quad \cdots\cdots 8 \\ 16 & 2 \quad \cdots\cdots 15(F) \\ & 0 \quad \cdots\cdots 2 \end{array}$$

作法与十进制转成二进制类似，唯在写答案时需用十六进制来表示，例如本例的余数15就要写成F，故本例的答案为：

$760=2F8$

(四) 十六进制→十进制

$$\begin{array}{ccc} 2 & F & \\ \downarrow & \downarrow & \downarrow \\ 16^2 & 16^1 & 16^0 \end{array}$$

$8=2\times16^2+F\times16^1+8\times16^0$

$=2\times256+15\times16+8\times1=512+240+8$

$=760$

(五) 二进制→十六进制

$\frac{1001}{9}\quad\frac{1101}{D}$

因二进制的四位数恰可用十六进制的一位数来表示，因此只要将二进制以四位数为一个单位来分割就可直接转换了，故本例的结果为：10011101=9D

(六) 十六进制→二进制

9　D

1 1 0 1

1 0 0 1

同理十六进制的一位数可表示成二进制的四位数，因此只要按顺序直接转换即可，本例的答案为：9D=10011101

以上所介绍的三种数值系统(二进制、十进制、十六进制)在汇编语言程序中可说是用得非常普遍。

6. 有关电脑内存的使用(不管是内存或外存)，通常都会使用到K(Kilo，千)及M(Mega，百万)这二个字，这是对内存容量的一种表示方式(以Byte为单位)，例如买电脑

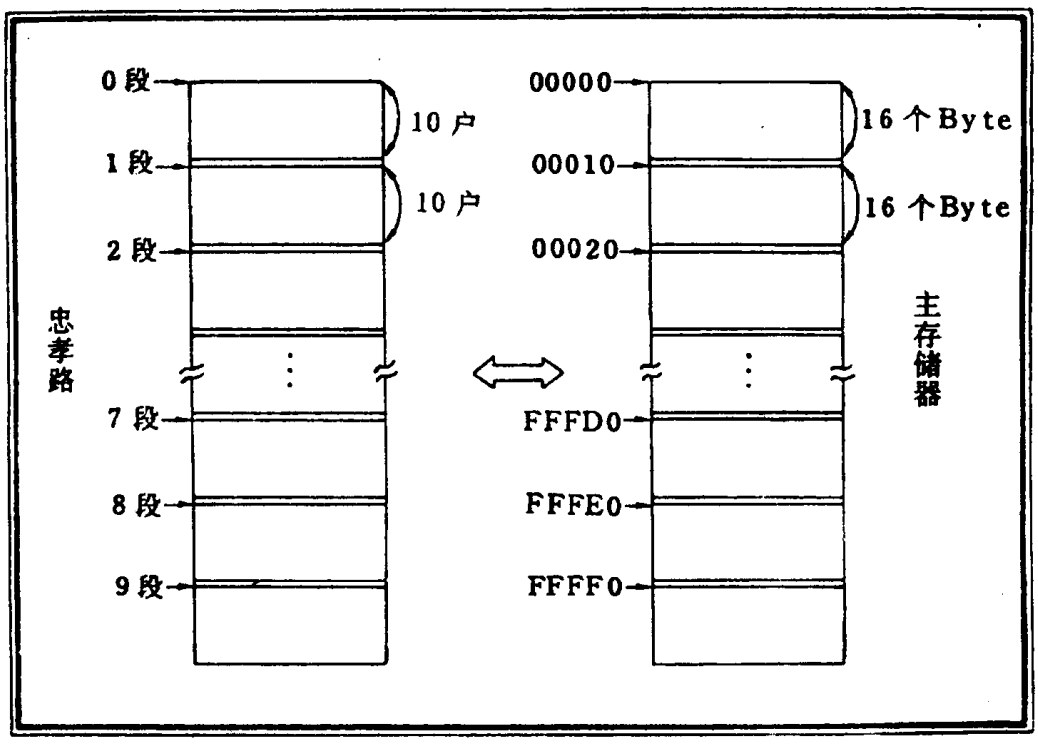
时厂商会告诉您内存的容量(或数量)为 640K 或 1M,软盘有 360K(2D)及 1.2M(2HD),硬盘有 10M、20M、40M……等等,如果我们用十进制来表示,则可得到如下的关系:

$1K=2^{10}=1024$

$1M=2^{20}=1024K=1048576$

由上述二列等式读者不难看出,电脑上所使用的千及百万是比数理上所使用的千及百万要来得大些(数理上的 $1K=10^3=1000$,$1M=10^6=1000000$,这是因为数理习惯使用十进制的关系),然而为了能使用编号来编排内存的地址,对 1K 来讲我们必须合并 10 个 Bit 才足够来编排(1M 就必须合并 20 个 Bit,这是因为电脑是属于二态装置的关系),就目前个人电脑所使用的 CPU 来讲,其所能存取的内存空间最大为 1M,因此对内存就必须合并 20Bit 才能表示其所有的地址空间(这就好比在十进制系统中,如果我们使用二位数来编号,那只能从 00 处增到 99,编一百个不同的号码而已)。

7. 就目前个人电脑所使用的 CPU 来讲,其一次所能处理的数据位数最长为 16 个 Bit,最短为 8 个 Bit(这是牵涉到 CPU 内部结构的问题,在后面我们再详述),但是表示地址的长度是 20Bit,如果需要,必须针对某个地址值做运算以得到另外一个新地址时(也就是将地址看成是一个数),那只送一个地址值给 CPU 去运算,CPU 是无法完成的(不知道该怎么做,因超过所能处理的位数),那该怎么办呢? 在还没有回答这个问题之前,我们先来看一个例子(这是有助于概念上的理解),如果有一条直通的马路,如下图:

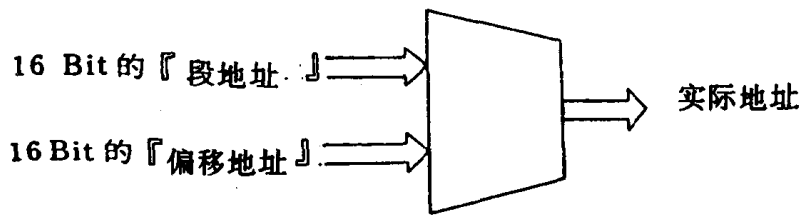


假设上图的路名为忠孝路且为了管理方便我们将此路分成十段(注意:在汇编语言内对数的起算值是由 0 开始,其它高级语言都是由 1 开始,而此处我们就是利用汇编语言的方式来编排分段值),每一分段内各住十户人家,门牌编号由 0～9 号(各分段都相同),如果有一封信写明收件地址是忠孝路 2 段 5 号,那邮差在送信时,只要从第 2 段的起始地址

开始，往下算六户就可以将信送到了。在此例中因路有分段的关系，所以我们只要各用一位数来表示分段及门牌号码就可以得到真正的地址了(没有分段就要用二位数来表示地址，因有一百户人家)而所谓的门牌号码其真正的作用就是告诉邮差先生距离所指明的分段有多远的意思，因此换一个名词来说明，我们可用差距或偏移(OFFSET)一词来解释这个门牌号码。

有了上述分段与偏移的概念之后，我们也可以将此概念应用到内存的配置上，原则上内存从最低的地址开始每隔 16 个 Byte 之处都可作为段的起始地址(如上页右图所示)，而由图中我们可以发现到每一个段的起始地址的最右边的一位数都是 0，如果将此位数忽略不计，那么段地址我们只要用四位数(即 16 个 Bit)来表示就可以了，同样的偏移地址的长度我们也是用四位数(即 16 个 Bit)来表示，(这是 CPU 内部硬件结构的规定)，因此要表示内存某一个地方的实际地址时，就跟 邮差送信一样我们只要知道该处所在的段地址及偏移地址就可以了。(因这二种地址的长度都是 16Bit ， CPU 有办法接受而加以处理)

当把段地址及偏移地址送给 CPU 之后，CPU 是如何地来将这两个 16Bit 的东西变成 20Bit 的呢？原来的 CPU 内有一个专门处理地址的硬件线路，其简图如下：(两个 16Bit 的输入端和一个 20Bit 的输出端)



实际上此硬件线路就是由一个左移位器和一个加法器所组成的，首先它会将用户所送过来的段地址先左移四个 Bit，(即左移十六进制一位数，左移后用 0 填补空位)以产生 20Bit 的段地址(上一页内存的每一个段地址中被我们所忽略的最右边那个 0，就是利用此移位器来产生的)，然后再将此 20Bit 的段地址)加上偏移地址就可得到 20Bit 的实际地址了，请看下列举例的演算：

假设段地址＝2A5B，偏移地址＝7218，求实际地址？

(一) 先将段地址左移四个 Bit(即左移十六进制一位数，左移后用 0 填补空位)可得 20Bit 的分段地址为：2A5B 0)

(二) 将此 20Bit 的段地址与偏移地址相加(使得个数对齐，且用十六进制的加法，即超过 16 便产生进位)

$$\begin{array}{r} 2\ A\ 5\ B\ 0 \\ +\quad 7\ 2\ 1\ 8 \\ \hline 3\ 1\ 7\ C\ 8 \end{array}$$

→内存的实际地址

然而用 16 个 Bit 的长度来表示段地址及偏移地址，读者算算看用 16 个 Bit 可以表示的范围有多大呢？(答案是 $2^{16}=65536=64K$)因此我们可以这么说：原则上内存最多可有