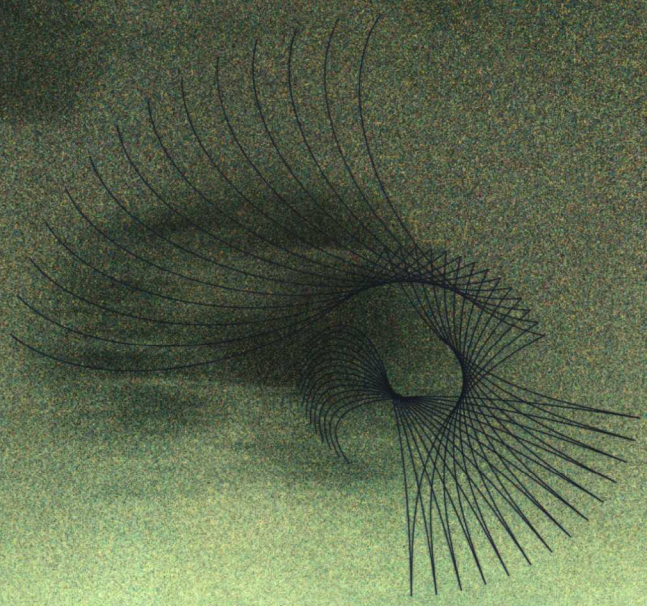
新世纪高等院校精品教材



单片微型计算机原理和应用

蔡菲娜 主编

浙江大學出版社

新世纪高等院校精品教材

单片微型计算机原理和应用

蔡菲娜　主编

浙江大学出版社

图书在版编目(CIP)数据

单片微型计算机原理和应用 / 蔡菲娜主编. —杭州：浙江大学出版社，1996.2(2001 重印)
ISBN 7-308-02646-9

Ⅰ.单... Ⅱ.蔡... Ⅲ.单片微型计算机—基本知识 Ⅳ.TP368.1

中国版本图书馆 CIP 数据核字(2001)第 073582 号

出版发行 浙江大学出版社
(杭州天目山路 34 号 邮政编码 310028)
(E-mail:zupress@mail.hz.zjcn)
(网址:http://www.zjupress.com)
责任编辑 钟仲南
排　　版 浙江大学出版社电脑排版中心
印　　刷 德清第二印刷厂
开　　本 787mm×1092mm 1/16
印　　张 15
字　　数 347 千字
版 印 次 2003 年 1 月第 2 版 2006 年 7 月第 9 次印刷
印　　数 23001—25000
书　　号 ISBN 7-308-02646-9/TP・206
定　　价 22.00 元

前　言

自本书初版发行以来，单片机的应用领域已经得到了进一步的拓宽，单片机技术有了迅速的发展。众多新型的51系列的单片机大大增强了原有单片机的功能，给单片机的应用注入了新的活力。与此同时，人们的理念也在不断地更新，人们愈来愈追求工作效率的提高和研发周期的缩短，追求用更好的开发手段对产品进行高效的开发和有效的维护，人们愈来愈关注利用高级语言快捷、便利、功能强的特点来编写单片机应用程序。现在，随着单片机存贮空间的不断扩大，运行速度的不断加快，这种良好的企望已经能够实现。

由于C语言本身既具有高级语言的特点，又具备了汇编语言的功能和可以直接对系统硬件进行操作这些特点，为结构化程序设计提供了有力的保障，因此利用C51进行程序设计目前已经成为软件开发的一个主流。本书修订时，在对前面九章进行修改的同时，删掉了原第十章内容(8098单片机)，改为新的第十章，增加了用C51对单片机进行编程的内容，正是顺应程序设计的这种发展趋势。在第十章重新编写过程中，考虑到高等院校的学生在学习本课程时已经具有C语言的基本知识，为减少篇幅，对C语言的基本语法未作系统的阐述，着重于从单片机开发应用的角度出发，在介绍了C51的基本语法特点后，针对单片机的应用系统中常用到的一些关键技术，如中断、串行口、定时器以及常用的外围接口、混合编程等作了详细的叙述，并精心选择了例题，使读者能在前九章学习的基础上尽快地掌握C51的编程方法和编程技巧。

本书在修订后，仍保留了第一版的特点，就是：

1. 本书的体系结构采用难点分散、深入浅出的方法，逐步引出新概念，逐步上台阶，使学习者学习起来不感到很困难。编者通过多年来的教学实践，证明这样的编排是合适的，取得了良好的教学效果。

2. 由于单片机这一课程是集硬件、软件于一体的一门综合性计算机课程，而学习的目的又重在应用，因此在程序设计中列举的例子比较注重实际应用，对于应用较多的硬件接口作了较详细的分析。书中列举的软硬件设计很多取自于编者多年来的科研成果以及实际产品的开发。

本书是针对高等院校的电子信息、通信工程、计算机、机械电子工程等专业的学生编写的，也可供相关的科技工作者作为参考。

全书由蔡菲娜任主编，对全书各章内容进行统编和修订，并且编写了其中的第六、七、九、十章。蒋钦参加了第一、二、五、八章的编写，刘继清参加了第三、四章的编写。李圣良对部分程序作了验证，陈梓城任本书第一版的主审。本书在修订过程中，还得到了刘勤贤的帮助和支持，在此一并表示诚挚的感谢。由于编者水平有限，书中难免有不足和错误之处，恳请大家批评指正。

编　者

2003年1月于浙江工业大学

目　录

□第一章

微型计算机基础知识

计算机是微电子学与计算数学相结合的产物。微电子学的基本元件及其集成电路形成计算机的硬件基础，而计算数学的计算方法与数据结构则为计算机的软件基础。电子计算机最基本的功能是进行数据运算。在电子计算机中，数字和符号都是用电子元件的不同状态表示的。由于计算机的这个特点，提出了一系列的问题：对参与运算的数有哪些要求？它们是如何表示的？计算机中数的表示法与常用的数的表示法有什么关系？本章中的叙述，将逐个回答这些问题。

第一节　计算机中的数制与码制

1.1.1　十进制数

人们最常用的数制是十进制，它是由 0、1、2、3、4、5、6、7、8、9 十个不同的符号来表示数值的，这十个符号就是数字。我们常用的十进制是采用位置记数法数制，也就是每个数字的位置决定了它的值或者权。例如数 212.48 的小数点左边第一位 2 代表个位，其数值为 2×10^0；左边第二位 1 代表十位，其数值为 1×10^1；左边第三位 2 代表百位，其数值为 2×10^2；小数点右边第一位 4 代表 1/10 位，即表示 4×10^{-1}；小数点右边第二位 8 代表 1/100 位，即表示 8×10^{-2}。从以上分析可以看出，同样的数字在不同的位置代表的值或权是不一样的。数 212.48 按权展开为：

$$212.48=2\times10^2+1\times10^1+2\times10^0+4\times10^{-1}+8\times10^{-2}$$

采用位置记数法的数制有三个重要特征：

1. 数码的个数等于基数。如十进制的基数是 10，它有十个不同的数码 0～9。

2. 最大的数码比基数小 1。

3. 每个数位有一定的位值——“权”，它是基数的某次幂，该幂次由每个数所在的位置决定。

我们所说的十进制计数方式，本质上讲就是每位计满时，向高位进一，即“逢十进一”。

1.1.2　二进制数

在日常生活中用十进制计数，并非天经地义的事，只不过是为了表示数的方便。众所周知，人有十个手指头。同样，在计算机中，为了表示数方便，采用二进制计数法。按照位置记数法的三个特点，在二进制中有：

1. 数码的个数等于基数 2，即只有二个数码 0、1。

2. 最大的数码是 1。

3. 每个数位有一定的位值——“权”，它是基数 2 的某次幂，如二进制数 1011.11 按权展开有：

$1011.11=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

二进制计数方式，本质上讲就是每位计满 2 时向高位进一，即“逢二进一”。

十进制和二进制的共同特点是进位，前者是“逢十进一”，后者是“逢二进一”。十进制数的小数点向右移一位，数值就扩大十倍；小数点向左移一位，数值就缩小十倍。同样，二进制数的小数点向右移一位，数值就扩大 2 倍；小数点向左移一位，数值就缩小 2 倍。判断十进制数是奇数还是偶数，只要看个位就行了。二进制数也有类似性质，若个位数是 1，则该数为奇数，若个位数是 0，则该数为偶数。如 01、11 等是奇数，10、100 等是偶数。

二进制的很大一个优点，就是它的每个数位都可以用任何具有两个不同稳定状态的元件来表示。例如，开关的闭合和断开，晶体管的截止与导通等。只要我们规定一种状态表示“1”，另一种状态就表示“0”。

还值得指出的是：采用二进制可以节约存贮设备。例如，表示 0～999 之间的 1000 个数，十进制用了 3 位，共需 3×10＝30 个稳定状态的设备量，而用二进制数表示时，则用 10 位（实际上 $2^{10}=1024$ 个数），只需 2×10＝20 个稳定状态的设备量。

此外，采用二进制，可以使用逻辑代数这一数学工具，为计算机的设计和分析提供了方便。

1.1.3 十六进制数

十六进制的基数为“十六”，其数码共有十六个：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。十六进制的权是以 16 为底的幂。

用十六进制可以简化二进制数的书写，又便于记忆。例如：

1000B＝8H

1111B＝FH

11111001B＝F9H

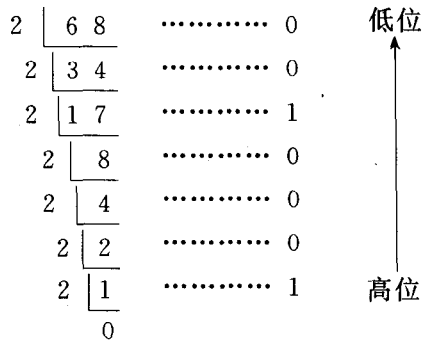
在计算机文献资料中常用数字符号加字母后缀 B、H、D 等表示采用何种进位计数制。如 101B 表示 101 是二进制数，88H 表示 88 是十六进制数，88D(D 可省略)表示 88 是十进制数，请注意区别。

1.1.4 数制之间的转换

由于我们习惯用十进制数，在研究问题的过程中，总是用十进制数来考虑和书写。当考虑成熟后，要把问题变成计算机能够“看得懂”的形式时，就得把问题中的所有十进制数转换成二进制代码，这就需要用到“十进制数转换成二进制数的方法”。在计算机运算完毕得到二进制数的结果时，又需要用到“二进制数转换为十进制数的方法”，才能把运算结果用十进制数显示出来。

一、十进制数转换成二进制数的方法

一个十进制数转换成二进制数时，通常采用“除 2 取余”的方法得到。即：将十进制数一次一次地除 2，所得到的余数（书写顺序由下至上）就是用二进制数表示的数。例如将 68D 转换成二进制数：



结果 68D=1000100B

表 1-1　几种进位制对照表

十进制	二进制	十六进制
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	$10000(2^4)$	10
32	$100000(2^5)$	20
64	$1000000(2^6)$	40
128	$10000000(2^7)$	80
256	$100000000(2^8)$	100
512	$1000000000(2^9)$	200
1024	$10000000000(2^{10})$	400

二、二进制数转换成十进制数

根据定义，只需将二进制数按权展开相加即可。例如：

$1011B=1\times2^3+0\times2^2+1\times2^1+1\times2^0=11D$

三、二进制数与十六进制数之间的转换

十六进制的数码有十六个。二进制数的每四位数，有十六种组合，可将它按次序和十六进制的十六个数码相对应，所以二进制数和十六进制数的互换非常简单。例如

1010 1101 1000 0101 B

↓ ↓ ↓ ↓

A D 8 5 H

反过来，就可以将十六进制数转换成二进制数，例如

F 9 A H

↓ ↓ ↓

1111 1001 1010 B

1.1.5 BCD 码

二进制数在计算机中容易实现，其运算规律也比较简单，因此在计算机中一般都采用二进制数字系统。但对于人们的习惯来讲，二进制数毕竟不怎么直观，因此，对于计算机的输入和输出，通常采用一种二进制编码的十进制数——BCD 码，它是十进制数，但它的十个不同的数字不是通常用的 0,1,2,…,9，而是采用 4 位二进制编码来实现，最常用的有 8-4-2-1 码，即用 0000,0001,0011,…,1001 等表示。例如 86D 的 BCD 码为 10000110。

1.1.6 ASCII 码

在微型计算机中，机器只能识别处理二进制信息，因此，字母和各种符号也必须按照某种特定的规则用二进制代码表示。目前，世界上最普遍采用的是 ASCII 码，全称为“信息交换标准代码”，这种编码在数据传输中也有广泛应用。

ASCII 码是一种 8 位代码，一般最高位可用于奇偶校验，故用 7 位码来代表字符信息，共有 128 个不同的字符。其中 32 个是控制字符，如 NUL(代码是 00)为空白符，CR(代码是 0DH)为回车。96 个图形字符，如数字 0～9 的 ASCII 码为 30H～39H，字母 A～Z 的 ASCII 码为 41H～5AH。ASCII 码详见附录一。

我国于 1980 年制订了“信息处理交换用的 7 位编码字符集”，除了用人民币符号￥代替美元符号 $ 外，其余代码含义都和 ASCII 码相同。

第二节　计算机中数的运算

1.2.1 机器数的表示方法

一、机器数和真值

在计算机中，二进制数码 1 和 0 是用电子元件的两种不同状态来表示，对于一个数的符号，即正、负号，也用电子元件的两种不同状态表示。因此在计算机中符号也“数码化”了。一般约定正数的符号用 0 表示，负数的符号用 1 表示。例如：

$N_1=+1011$；$N_2=-1011$

如果用一个字节(即 8 位二进制数)来表示上述二个数，它们在计算机中的表示分别为：00001011 和 10001011，其中最高位表示符号位。

以 0 表示正数的符号，以 1 表示负数的符号，并且每一位的数值也用 0 或 1 表示，这样的数叫机器数，有时也叫机器码，而把对应于该机器数或机器码的数值叫真值。

二、原码表示方法

正数的符号位用0表示，负数的符号位用1表示，这种表示法称为原码表示法。例如：

$[+120]_{原}=01111000$

$[-120]_{原}=11111000$

对于零，可以认为它是正零，也可以认为它是负零，所以零的原码有两种表示方法：

$[+0]_{原}=00000000$

$[-0]_{原}=10000000$

八位二进制原码表示数的范围为11111111～01111111，即－127～＋127。

三、反码表示方法

在反码表示方法中，正数的反码与正数的原码相同，负数的反码由它对应正数的原码按位取反得到。例如：

$[+120]_{反}=[+120]_{原}=01111000$

$[-120]_{反}=\overline{[+120]_{原}}=\overline{01111000}=10000111$

零的反码有两种表示方式，即

$[+0]_{反}=00000000$

$[-0]_{反}=11111111$

八位二进制反码表示数的范围为－127～＋127。当符号位为0时，其余位为数的真值，当符号位为1时，其余位按位取反后才是该数的真值。

四、补码表示方法

先以钟表对时为例，说明补码的概念。假设现在的标准时间为4点整，而有一只表却已是7点，为了校准时间，可以采用两种方法：一是将时针退3格，即7－3＝4，一是将时针向前拨9格，即7＋9＝12(自动丢失)＋4，都能对准到4点。可见，减3和加9是等价的，我们把(＋9)称为(－3)对12的补码，12为模，当数值大于模12时可以丢弃12。

在字长为8位的二进制数字系统中，模为$2^8=256D$。先考察下例：

01000000	64
－00001010	－10
00110110	54
01000000	64
＋11110110	＋246
1 00110110	54

丢失

可见在字长为8位的情况下(64－10)与(64＋246)的结果是相同的，所以(－10)和246互为补数。采用补码表示数，可将减法运算转换成加法运算。现在我们看一看(－10)的补码11110110是怎样求得的。

正数的补码表示与原码相同，而负数的补码表示即为它的反码加1形式。如：

$[+4]_{原}=00000100$

$[-4]_{反}=11111011$

$[-4]_{补}=11111100$

又如：

$[+10]_{原}=00001010$

$[-10]_{反}=11110101$

$[-10]_{补}=11110110$

在补码表示法中，零的补码只有一种表示法，即$[0]_{补}=00000000$。对于八位二进制数而言，补码能表示的数的范围为$-128\sim+127$。当符号位为0时，其余位为数的真值；当符号位为1时，其余位按位取反再加1后才是数的真值。

1.2.2 补码的加减运算

当用补码表示数时，可用加法完成减法运算，因此带符号数一般都以补码形式在机器中存放和参加运算。

补码的加法公式是：

$$[X]_{补}+[Y]_{补}=[X+Y]_{补}$$

例 1.1 用补码运算求120－63。

解：$[+120]_{补}=01111000$

$[+63]_{原}=00111111$

$[-63]_{补}=11000001$

做加法

$$\begin{array}{r} 01111000 \\ +11000001 \\ \hline 1\ 00111001 \end{array}$$

丢失

相加结果为正数，即为57D。

例 1.2 用补码运算求64＋65。

解：$[64]_{补}=01000000$

$[65]_{补}=01000001$

做加法：

$$01000000+01000001=10000001$$

此时二个正数相加，其结果为负数，产生了错误的结果。这是因为：二个正数相加的结果超出了8位二进制数的范围，故产生了错误的结果。这种情况称为溢出。一般而言，若两正数相加，其结果为负数或二个负数相加其结果为正数，都表明产生了溢出。

上例若采用16位的补码来运算，即可得出正确的结果：

$[64]_{补}=0000000001000000$

$[65]_{补}=0000000001000001$

$[64]_{补}+[65]_{补}=0000000010000001$

第三节 微型计算机基本工作原理

微型计算机是计算机的微型化，它是由微处理器（也称中央处理器CPU）、存贮器（RAM、ROM）和输入/输出（I/O）接口电路三个最基本部件所组成，如图1-1所示，通过接口电路再与外围设备相连。

各基本部件通过总线交换信息。所谓总线是信息流通的公共通道，总线上的信息可以同时输送给几个部件，但不允许几个信息同时输送给总线，否则将产生信息冲突。

总线包括数据总线、控制总线和地址总线。数据总线用于CPU、存贮器、输入/输出接口之间传送数据，如从存贮器取数到CPU，把运算结果从CPU送到外部设备等。数据总线是双向的。控制总线可以是CPU发出的控制信号，也可以是其他部件输入到微处理器的信息，对于每一条控制线，其传送方向是固定的。地址总线用来传输CPU发出的地址信息，以选择需要访问的存贮器和I/O接口电路。地址总线是单向的，只能是CPU向外传送地址信息。微机采用上述三组总线的连接方式，常被称为三总线结构。

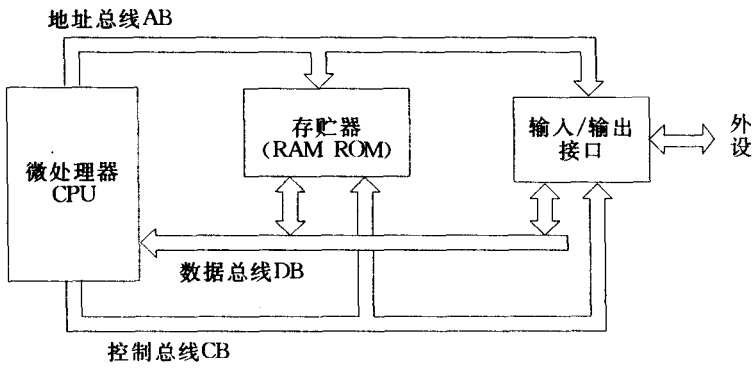


图1-1　微型计算机基本组成

1.3.1　微处理器

典型的微处理器(CPU)结构如图1-2所示。我们可以把它分为三大部分：算术逻辑运算部件、控制逻辑部件和寄存器部件，它们都挂在内部总线上，现分别叙述如下：

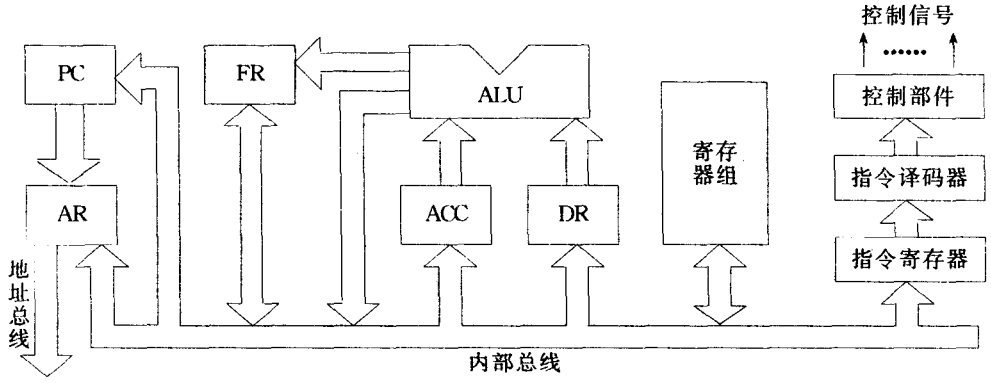


图1-2　微处理器的基本结构

1. 算术逻辑部件ALU

算术逻辑单元ALU是微机执行算术和逻辑运算的主要部件，是运算器的主要组成部件。它的基本组成是一个可控的加法/减法器。ALU有两个输入端和两个输出端。一个输入端与累加器A相连，另一个与数据寄存器DR相连。参加运算的数都要先送到这两个寄存器中，然后才能由CPU实行相应的操作。ALU的一个输出端与内部总线相连，以便把处理的结果通过总线送到累加器A中，另一个输出端与标志寄存器FR相连，以存放运算结果的某些标志状态。每执行完一条指令的状态特征，也可以由标志寄存器FR来表

征。

2. 累加器 A

累加器 A 是微机中的一个关键寄存器，凡是通过 ALU 进行算术/逻辑运算的操作，其中的一个操作数必须来自累加器 A，而运算的结果也必须通过内部总线送回到 A 中。有的微处理器只有一个累加器(如 8051)，而有的微处理器不止一个累加器(如 8098)。

3. 标志寄存器 FR

标志寄存器 FR 用于保存计算机执行完一条指令后，某些状态标志的有关信息。特别是进行了算术/逻辑运算以后，某些状态标志就会产生变化，例如，运算结果产生了进位/借位，或者产生了溢出，这些状态标志都保存在标志寄存器 FR 中。不同型号的微处理器标志位的数目及具体规定都不相同。例如 8051 单片机的标志位寄存器的名字叫程序状态字寄存器 PSW，共有六位状态标志，以后我们会详细介绍。

4. 寄存器组

微处理器的寄存器组一般分为两部分，即通用寄存器或称数据寄存器和专用寄存器。通用寄存器相当于 CPU 内部小容量的存贮器，用来暂时存放一些数据。由于寄存器在 CPU 内部，因而数据之间传送速度比较快，对于这部分寄存器的分布及作用应有一定的认识，一方面通过充分利用这些通用寄存器，可以提高运算速度，另一方面，也可简化程序设计。专用寄存器也叫特殊功能寄存器，每一种寄存器都有专门的用途，如标志寄存器 FR 就是一种特殊功能寄存器，其它还有累加器 A、堆栈指示器 SP 等等。

5. 程序计数器 PC

程序计数器 PC 用来存放下一条将要执行的指令地址。程序中的各条指令都存放在程序存贮器中，需要执行某条指令时，该指令的地址就从 PC 计数器送到地址总线，指令执行完毕后，PC 计数器自动加 1，指向下一条将要执行的指令地址。程序除了顺序执行外，也可作一定范围的跳转，这时 PC 计数器的内容就根据转移地址作较大范围的改变。

6. 指令寄存器、指令译码器、控制信号发生器

指令寄存器接收从程序存贮器取来的指令，并在整个指令执行过程中加以保存。指令译码器对指令进行译码，不同的指令产生不同的控制信号，送到控制信号发生器。控制信号发生器根据指令译码器送出的电平信号和时钟脉冲信号组合，形成各种按一定节拍变化的电平和脉冲，即各种控制信号，它可以被送到存贮器、运算器或 I/O 接口电路。

这部分电路的定时功能是由晶体振荡器产生的时钟脉冲控制的，一般每执行一条指令需要几个甚至几十个时钟周期。

1.3.2 存贮器

存贮器分为两大类：只读存贮器(ROM)和随机存取存贮器(RAM)。只读存贮器在程序执行过程中只能读出里面的信息，而不能写入新的内容。随机存取存贮器不但能随时读取已存放在各个存贮单元中的数据，而且还能随时写入新的信息。

存贮芯片内部有若干存贮单元，存贮单元所存内容称为一个字。一个字由若干位组成。8 个记忆元件的存贮单元就是一个 8 位记忆字，通常称为一个字节；16 个记忆单元组成的存贮单元就是一个 16 位记忆字，通常称为一个字。每个存贮单元都有固定地址，在存贮器内部都带有译码器，根据二进制编码译码的原理，n 根地址线可以译成 2^n 个地址号。