第 1 章 单片机概述

本章教学目标

- 了解单片机的概念、特点及其发展；
- 了解单片机的分类及应用领域；
- 掌握单片机的基本结构及各部件功能；
- 掌握各进制之间的转换。

本章重点内容

- 掌握单片机基本结构；
- 熟练掌握二进制与十六进制之间的转换。

1.1 单片机概念

什么是单片机？单片机是指在一块硅片上集成了微处理器、存储器及各种输入/输出接口的芯片，即单片机就是一块集成芯片，且这块芯片具有计算机的属性，所以还被称为单片微型计算机，简称单片机。单片机只有外加所需的输入、输出设备，才可以实现单片机应用系统。

1.2 单片机的特点

（1）具有高性价比。单片机之所以被广泛应用，其中一个很大的原因就是其具有高性能、低价格、安装调试等显著特点。

（2）集成度高、体积小、可靠性高。单片机是将各个功能部件集成在一块芯片上，对

信息的传输及对存储器和 I/O 口的访问一般是在单片机内部进行的，所以不易受外界干扰，具有较高的可靠性。

1.3 单片机的发展

1976 年 Intel 公司推出了 MCS－48 系列 8 位单片机，因其体积小、功能全、价格低得到了广泛的应用。

由于 MCS－48 系统的成功应用，单片机系列及单片机应用技术迅速发展，世界各地厂商已相继研制出大约 50 个系列 300 多个品种的单片机产品。代表产品有 Intel 公司的 MCS－51系列单片机(8 位机)、Motorala 公司的 MC68O1 系列单片机、Zilog公司的 Z－8 系列单片机等。单片机的应用领域不断扩大，除了在工业控制智能化仪器仪表、通信及家用电器等领域应用外，在智能化、高档电子玩具产品中也大批采用单片机作为核心控制部件。

此后，在 8 位单片机的基础上，又推出了超 8 位单片机。其功能进一步加强。同时 16 位单片机也相继问世，代表产品有 Intel 公司的 MCS－96 系列单片机。

单片机正朝着高性能和多品种的方面发展。然而，由于应用领域大量需要的仍是 8 位单片机，因此，各大公司纷纷推出高性能、大容量、多功能的新型 8 位单片机。目前，市场上的主流产品是51 系列兼容机、由 STC 公司推出的高性价比的 STC89 系列单片机(带负载能力最强)和 Atmel 公司生产的 AT89 系列单片机。由于 51 系列兼容单片机使用方便、灵活且能满足绝大多数应用领域的需要，因而可以肯定，现在及今后相当长的一段时期内，51 系列兼容单片机仍将占据单片机应用的主导地位。

1.4 单片机的分类

1. 按生产厂商分

美国的英特尔(Intel)公司、摩托罗拉(Motorola)公司、国家半导体(NS)公司、Atmel 公司、微芯片(Microchip)公司、洛克威尔(Rockwell)公司、莫斯特克公司(Mostek)、齐洛格(Zilog)公司、仙童(Fairchid)公司、德州仪器(TI)公司，日本的电气(NS)公司、东芝(Toshiba)公司、富士通(Fujitsu)公司、松下(Panasonic)公司、日立(Hitachi)公司、日电(NEC)公司、夏普(Sharp)公司等。

2. 根据应用领域分

(1) 工控型/家电型。工控型的单片机主要是面向测控，要求寻址范围大，运算能力强。家电型的单片机要求体积小、价格低，外围器件少，使用方便。

(2) 总线型/非总线型。总线型单片机是指单片机设有并行总线，用以扩展并行外围

器件。非总线型单片机是指单片机通过串行口与外围器件连接，或直接把外围器件、外设接口集成在片内。

（3）通用型/专用型。通用型单片机，它的应用范围宽，如 Intel 公司的 MCS－51 系列产品 8031、80C51 等通过不同的外围扩展就可以用在不同的设备中。专用型单片机是专门为某一产品设计生产的，如电子体温计、计费电度表等。

3. 按字长分

（1）4 位单片机。4 位单片机的控制功能较弱，CPU 一次只能处理 4 位二进制数，这类单片机常用于计算器、各种形态的智能单元以及作为家用电器中的控制器。典型产品有 NEC 公司的 UPD75 系列、NS 公司的 COP400 系列、松下公司的 MN1400 系列、洛克威尔公司的 PPS/1 系列，富士通公司的 MB88 系列、夏普公司的 SM 系列、东芝公司的 TMP47 系列等。

（2）8 位单片机。8 位单片机和 4 位单片机相比，不仅具有较大的存储容量和寻址范围，而且中断源、并行 I/O 接口和定时器/计数器等个数都有了不同程度的增加，并集成有全双工串行通信接口。在指令系统方面，普遍增设了乘除指令和比较指令。特别是 8 位机中的高性能增强型单片机，除片内增加了 A/D 和 D/A 转换器外，还集成有定时器捕捉/比较寄存器、监视定时器(Watchdog)。总线控制部件和晶体振荡电路等。这类单片机由于其片内资源丰富和功能强大，主要应用在工业控制、智能仪表、家用电器和办公自动化系统中。代表产品有 Inter 公司的 MCS－48 系列和 MCS－51 系列、Microchip 公司的 PIC16C 系列和 PIC7C 系列以及 PIC1400 系列、Motorola 公司的 M68HC05 系列和 M68HC11 系列、Zilog 公司的 Z8 系列、荷兰 Philips 公司的 80C51 系列(同 MCS－51 兼容)、Atmel 公司的 AT89 系列(同 MCS－51 兼容)、NEC 公司的 UPD78 系列等。

（3）16 位单片机。16 位单片机是在 1983 年以后发展起来的。这类单片机的特点是：CPU 是 16 位的，运算速度普遍高于 8 位单片机，有的单片机的寻址能力高达 1MB，片内含有 A/D 和 D/A 转换电路，支持高级语言。这类单片机主要用于过程控制、智能仪表、家用电器以及作为计算机外部设备的控制器等。典型产品有 Intel 公司的 MCS－96/98 系列、Motorola 公司的 M68HC16 系列、TI 公司的 MSP430 系列等。其中，以 MSP430 系列最为突出。它集成了较丰富的片内外设：看门狗(WDT)、模拟比较器 A、定时器 A(Timer_A)、定时器 B(Timer_B)、串口 0、1(USART0、1)、硬件乘法器、液晶驱动器、10 位/12 位 ADC、12C 总线直接数据存取(DMA)、端口 O(P0)、端口 1～6(P1～P6)、基本定时器(Basic Timer)等。

（4）32 位单片机。32 位单片机的字长为 32 位，是单片机的顶级产品，具有极高的运算速度。随着家用电子系统的新发展，32 位单片机的市场前景良好。继 16 位单片机出现后不久，几大公司先后推出了代表当前最高性能和技术水平的 32 位单片机系列。32 位单片机具有极高的集成度，内部采用新颖的 RISC(精简指令系统计算机)结构，CPU 可与其他微控制器兼容，主频频率可达 33MHz 以上，指令系统进一步优化，运算速度可动态改变，设有高级语言编译器，具有性能强大的中断控制系统、定时/事件控制系统、同步/异步通信控制系统。代表产品有 Intel 公司的 MS－80960 系列、Motorola 公司的 M68300 系

列、Hitachi 公司的 Super H(简称 SH)系列等。

4. 按制造工艺分

(1) HMOS 工艺。高密度短沟道 MOS 工艺，具有高速度、高密度的特点。

(2) CHMOS(或 HCMOS)工艺互补的金属氧化物的 HMOS 工艺，是 CMOS 和 HMOS 的结合，具有高密度、高速度、低功耗的特点。Intel 公司产品型号中若带有字母“C”，Motorola公司产品型号中若带有字母“HC”或“L”，通常为 CHMOS 工艺。

1.5 单片机的应用

基于其易操作、易编程等特点，而受到广泛应用，单片机主要应用于以下几个常见的生活领域：

(1) 家电电器。例如在电视、空调、冰箱、洗衣机、家用防盗报警器等产品中，单片机使其本身具有更多的功能，进而实现智能控制。

(2) 智能玩具。由于单片机的价格低廉、功能强大，被广泛应用于智能型玩具的控制，例如发声玩具、玩具机器人、遥控电动车等。

(3) 智能测量设备。使用单片机强大的可编程和可扩展能力，可以设计新一代的智能化仪表，如各种数字万用表、示波器等。

(4) 自动测控系统。在自动控制和测量领域，可以采用单片机设计各种数据采集系统、自适应控制系统等。例如温度的自动控制、压力的自动感应、电压电流的数据采集和分析等。

由于其具备可重复编程等强大功能，单片机可适用于一切需要智能控制的场合。

1.6 微机的基本结构

单片机所特有的结构和资源反映了单片机的性能，也是单片机程序设计的基础。下面介绍一下 8051 单片机的基本结构。

1. 中央处理器

中央处理器(CPU)是整个单片机的核心部件。CPU 主要由算术逻辑部件、控制器和专用寄存器三部分组成。它负责控制、指挥和调度整个单元系统的工作，完成运算和控制输入/输出功能等操作。

2. 程序存储器

程序存储器(ROM)用于存放用户程序、原始数据或表格等。8051 单片机共有 4096 个 8 位 ROM。有些增强型的单片机提供了更大的程序存储器，有些甚至还采用 Flash 程序存储器。

3. 数据存储器

数据存储器(RAM)可存放读写的数据、CPU 运算的中间结果或用户定义的字型表等。8051 单片机内部有 128 个 8 位用户数据存储单元和 128 个专用寄存器单元。专用寄存器只能用于存放控制指令数据，只能访问而不能存放用户数据，对于一些新推出的单片机，其内部 RAM 单元可能更多，例如 AT89S52 单片机内部有 256 个 RAM 数据存储单元。

4. 定时器/计数器

定时器/计数器用于单片机硬件的定时或者计数。一般包含两个 16 位的可编程定时器/计数器，以实现定时或计数功能。它也可以产生中断，从而在程序中控制程序的转向。部分新推出的单片机可能拥有更多的定时器/计数器。

5. 并行 I/O 口

单片机的并行 I/O 口主要用于和外部设备进行并行的输入/输出通信，以便于处理外部的输入和将运算结果反馈到外部设备。

6. 全双工串行口 UART

全双工串行通信口 UART 主要用于与其他设备间的串行数据传送。单片机均内置一个全双工串行通信口，该串行口既可以用作异步通信收发器，也可以当同步移位器使用。部分新推出的单片机可能拥有更多的全双工串行口。

7. 中断系统

8051 单片机具备较完善的中断功能，包含两个外部中断、两个定时器/计数器中断和一个串行中断。8051 单片机的中断系统具有两级的优先级别选择。部分新推出的单片机可能拥有更多的中断源。

8. 时钟振荡电路

时钟振荡电路主要用于为单片机提供 CPU 时钟源。单片机可以采用内部时钟振荡电路或者由外部提供时钟源，其最大工作频率根据单片机型号的不同而有所差别，例如 AT89S52 单片机的时钟振荡频率为 0 ~ 33MHz。

1.7 码制与数制

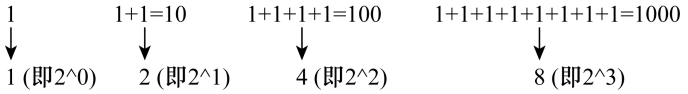
单片机和计算机一样只能识别“0”和“1”，即机器语言。任何命令或信息都是以二进制信息进行存储的。日常生活中常用十进制进行计数，而机器语言中用 0 和 1 构成 8 个字符串为一个字节，比如 00001011 表示字母 B，为了缩短字符串长度，我们习惯用十进制或十六进制。所以我们要学会各种进制之间的转换。

1.7.1　二进制

十进制数大家应该都不陌生，“逢十进一，借一当十”是十进制数的特点。有了十进制数的基础，我们学习二进制数便非常容易了，二进制以“逢二进一”的原则。为了便于区别不同进制的数据，一般情况下可在数据后面跟一后缀：二进制数用“B”作后缀，如二进制

数可表示为(110)B、(110.11)B、10110B 等，十六进制数用“H”作后缀，如 3A5H。十进制数用“D”作后缀，如 39D 或 39。

根据位权表示法，每一位二进制数在其不同位置表示不同的值。例如：



对于 8 位二进制数(由低位到高位分别用 D0～D7 表示)，其各位所对应权值为

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
D7	D6	D5	D4	D3	D2	D1	D0

对于任何二进制数，可按位权展开求和得到与之相应的十进制数，则有

(01)B = 0 * 2^1 + 1 * 2^0 = (1)D

(11)B = 1 * 2^1 + 1 * 2^0 = (3)D

(100)B = 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = (4)D

(111)B = 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = (7)D

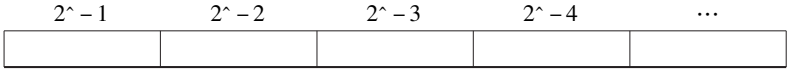
(1110)B = 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = (14)D

(11111)B = 1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = (31)D

例如，二进制数 10110111，按位权展开求和计算可得

(10101010)B = 1 * 2^7 + 0 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = (183)D

对于含有小数的二进制数，从小数点右边第一位小数开始向右各位的权值分别为



例如，二进制数 10110.101，按位权展开求和计算可得

(10110.101)B = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0 + 1 * 2^-1 + 0 * 2^-2 + 1 * 2^-3

= (22.625)B

必须指出：在计算机中，一个二进制数(如 8 位、16 位或 32 位)既可以表示数值，也可以表示一种符号的代码，还可以表示某种操作(即指令)，计算机在程序运行时按程序的规则自动识别，这就是本节开始所讲的一切信息都是以二进制数据进行存储的。

1.7.2 十六进制数

十六进制数是学习和研究计算机中二进制数的一种比较方便的工具。十六进制与二进制大同小异，不同之处就是十六进制是“逢十六进一，借一当十六”。还有一点特别之处需要注意，十进制的 0～15 表示成十六进制分别为 0～9，A，B，C，D，E，F，即十进制的 10 对应十六进制的 A，11 对应 B……15 对应 F。我们一般在十六进制数的最后面加上后缀 H，表示该数为十六进制数，如 AH，DEH 等。这里的字母不区分大小写，在 C 语言编程时要写成 0xa，0xde，在数的最前面加上“0x”表示该数为十六进制数。十进制数与十六进制数之间的转换在这里不再讲解，大家可参考十进制数与二进制数之间的转换规则。

十六进制数有 16 个数字符号，其中 0～9 与十进制数相同，剩余 6 个为 A～F。分别

表示十进制数的 10 ~ 15。十六进制数的计数原则是“逢十六进一”。也称其基数为十六，整数部分各位的权值由低位到高位分别为 16^0、16^1、16^2……

例如：(12)H = 1 * 16^1 + 2 * 16^0 = (18)D

(2A) = 2 * 16^1 + 10 * 16^0 = (42)D

1.7.3　不同进制之间的转换

计算机中的数只能用二进制表示，十六进制数是读写方便的需要且能缩短二进制的长度，日常生活中使用的是十进制数，因此，计算机必须根据需要对各种进制的数据进行转换。

(1) 二进制数转换为十进制数

任意二进制数均可按权值展开，将其转化为十进制数。

例如：10111B = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 23D

10111.011 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 + 0 * 2^ - 1 + 1 * 2^ - 2 + 1 * 2^ - 3

= 23.375D

(2) 十进制数转换为二进制数

十进制数转换为二进制数，可将整数部分和小数部分分别进行转换，然后合并。其中整数部分可采用“除 2 取余法”进行转换。小数部分可采用“乘 2 取整法”进行转换。

(3) 二进制数与十六进制相互转换

在计算机进行输入、输出时，常采用十六进制数。十六进制数可看做二进制数的简化表示。

因为 2^2 = 16，所以 4 位二进制数相当于 1 位十六进制数，二进制、十进制、十六进制对应数的转换关系见表 1 - 1。

表 1 - 1　十进制数和二进制数之间的转换表

十进制	二进制	十进制	二进制
0	0	8	1000
1	1	9	1001
2	10	10	1010
3	11	11	1011
4	100	12	1100
5	101	13	1101
6	110	14	1110
7	111	15	1111

我们在进行单片机编程时常常会用到其他较大的数，这时我们用 Windows 系统自带的计算器，可以非常方便地进行二进制、八进制、十进制、十六进制之间的任意转换，如图 1－1 所示。

图 1－1　Windows 自带的计算器

关于十进制、二进制与十六进制数之间的转换，我们要熟练掌握 0～15 之间的数，因为在以后的单片机 C 语言编程当中，我们要大量使用它们。一般的转换规律是，先将二进制数转换成十进制数，再将十进制数转换为十六进制数，若大家现在记忆不牢，也可以在以后的学习中边学边加深记忆。二进制、十进制、十六进制 0～15 的数的转换列表如表1－2所示。

表 1－2　十、二、十六进制数的转换表

十进制	二进制	十六进制	十进制	二进制	十六进制
0	0	0	8	1000	8
1	1	1	9	1001	9
2	10	2	10	1010	A
3	11	3	11	1011	B
4	100	4	12	1100	C
5	101	5	13	1101	D
6	110	6	14	1110	E
7	111	7	15	1111	F

本章小结

本章简单介绍了单片机概念、特点、分类及应用领域，常用二进制、十进制、十六进制之间的转换。为读者后续的学习打下基础。

本章参考文献

[1] 田希晖等. C51 单片机技术教程. 北京：人民邮电出版社，2007.
[2] 顾晖等. 微机原理与接口技术—基于 8086 和 Proteus 仿真(第 2 版). 北京：电子工业出版社，2015.
[3] 张毅刚等. 单片机原理与应用—C51 编程 + Proteus 仿真. 北京：高等教育出版社，2012.
[4] 林立等. 单片机原理与应用—基于 Proteus 和 KeilC(第 2 版). 北京：电子工业出版社，2013.
[5] 李群芳等. 单片微机计算机与接口技术(第 4 版). 北京：电子工业出版，2012.

本章习题

1.1　简述什么是单片机，主要特点有哪些?
1.2　单片机主要应用在哪些领域?
1.3　把下列十进制数转换为二进制数和十六进制数。
（1）100　　（2）48　　（3）56.5　　（4）0.65
1.4　把下列十六进制数转换为二进制数和十进制数。
（1）20H　　（2）2AF　　（3）0F2.AH　　（4）12FAH

第 2 章 MCS－51 单片机硬件结构与工作原理

本章教学目标

- 熟悉单片机内部结构基本组成单元，掌握单片机的 RAM、ROM、中断系统、定时器/计数器、串口、并口、时钟电路等基本硬件结构；
- 了解单片机的引脚功能，熟悉 P0、P1、P2、P3 口的基本功能；
- 掌握单片机寄存器及寄存器的基本功能；
- 了解单片机内部机器周期、指令周期及指令时序；
- 熟悉单片机复位方式、程序执行方式、节点工作方式。

本章重点内容

- 重点掌握 MCS－51 单片机的硬件结构。

2.1 MCS－51 单片机的硬件结构

单片机是把一些作为控制所必需的基本部件都集成在一块芯片上。按照功能划分主要有：微处理器(CPU)、数据存储器(RAM)、程序存储器(ROM/EPROM)、中断系统、定时器/计数器、串行口、并行 I/O 口及时钟电路。各个部件之间信息的交换由总线完成。

各部件主要功能如下：

1. 微处理器(CPU)

中央处理器(CPU)是整个单片机的核心部件。CPU 主要由算术逻辑部件，控制器和专用寄存器三部分组成。它负责控制、指挥和调度整个单元系统的工作，完成运算和控制输入/输出功能等操作。

2. 数据存储器(RAM)

MCS－51 内部数据存储器共有 128 个单元，地址空间为 00H～7FH，用以存放和读取

数据，但不能存放程序指令，可直接寻址和间接寻址。按功能分为三个区域，如图2－1所示。

图 2－1　MCS－51 内部 RAM 的配置

（1）工作寄存器区

地址为 00H～1FH 的 32 个单元是四组通用工作寄存器区，每组有 8 个 8 位工作寄存器，寄存器名用 R0、R1、R2、R3、R4、R5、R6、R7 表示。工作寄存器和内部 RAM 地址对照关系如表 2－1 所示。

表 2－1　工作寄存器和 RAM 地址对照表

0 组		1 组		2 组		3 组	
地址	寄存器	地址	寄存器	地址	寄存器	地址	寄存器
00H	R0	08H	R0	10H	R0	18H	R0
01H	R1	09H	R1	11H	R1	19H	R1
02H	R2	0AH	R2	12H	R2	20H	R2
03H	R3	0BH	R3	13H	R3	21H	R3
04H	R4	0CH	R4	14H	R4	22H	R4
05H	R5	0DH	R5	15H	R5	23H	R5
06H	R6	0EH	R6	16H	R6	24H	R6
07H	R7	0FH	R7	17H	R7	25H	R7

用户可以通过对程序状态字寄存器 PSW 中的 RS1、RS0 的设置来选择当前工作寄存器组。当工作寄存器组发生切换后，原工作寄存器组的各寄存器的内容被屏蔽保护起来，利用这一特性可以方便地完成现场的快速保护任务。

单片机复位后自动选中第 0 组工作寄存器组。即 RS1，RS0＝00，这时内部 RAM 的 00H 与 R0 指的是同一单元，如果编程时不需要使用这么多工作寄存器组时，也可以把不用部分当作通用存储区。

（2）位寻址区

地址为 20H～2FH 的 16 个字节单元是位寻址区，共 128 位，位地址为 00H～7FH。如表 2－2 所示。该位寻址区既可以按字节操作，也可以按位操作。当某些信息只需用一个二进制位来表示时，用位操作就显得比较方便，MCS－51 系列单片机提供了这一片位寻址区，其中每一位都有地址，可以对它置“1”或清“0”；如果编程时不需要用这么多位时，

也可以把不用部分当作通用存储区。

表 2－2　MCS－51 位地址表

字节地址	位地址							
	D7	D6	D5	D4	D3	D2	D1	D0
20H	07H	06H	05H	04H	03H	02H	01H	00H
21H	0FH	0EH	0DH	0CH	0BH	0AH	09H	08H
22H	17H	16H	15H	14H	13H	12H	11H	10H
23H	1FH	1EH	1DH	1CH	1BH	1AH	19H	18H
24H	27H	26H	25H	24H	23H	22H	21H	20H
25H	2FH	2EH	2DH	2CH	2BH	2AH	29H	28H
26H	37H	36H	35H	34H	33H	32H	31H	30H
27H	3FH	3EH	3DH	3CH	3BH	3AH	39H	38H
28H	47H	46H	45H	44H	43H	42H	41H	40H
29H	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H
2AH	57H	56H	55H	54H	53H	52H	51H	50H
2BH	5FH	5EH	5DH	5CH	5BH	5AH	59H	58H
2CH	67H	66H	65H	64H	63H	62H	61H	60H
2DH	6FH	6EH	6DH	6CH	6BH	6AH	69H	68H
2EH	77H	76H	75H	74H	73H	72H	71H	70H
2FH	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H

（3）通用 RAM 区

地址为 30H～7FH 的 80 个单元是 RAM 区，可用于存放数据，也可作为堆栈存储区域。只能进行字节寻址。

3. 程序存储器(ROM/EPROM)

程序存储器用于存放编好的程序和表格常数。程序存储器通过 16 位程序计数器 PC 寻址，寻址能力为 64KB。

80C51 的 64KB 程序存储器空间，片内为 4KB，地址从 0000H～0FFFH；片外最多可扩至 64KB，地址从 1000H～FFFFH，片内外是统一编址的。

在 MCS－51 的指令系统中，同外部程序存储器打交道的指令仅有两条，即

MOVC A，@A＋DPTR

MOVC A，@A＋PC

两条指令的功能将在下一章中详细介绍。

（1）片内与片外程序存储器的选择

EA 引脚接高电平

CPU 首先执行片内程序存储器中的程序。当 PC 的值超过 0FFFH 后，系统会自动地转

向片外程序存储器读取程序，外部程序存储器从 1000H 开始编址。

EA 引脚接低电平(接地)

CPU 只能从片外程序存储器中读取程序(无论片内是否有程序存储器)，外部程序存储器从 0000H 开始编址，对于片内不带 ROM 的单片机，如 8031，使用时必须使 EA＝0，以便能够从外部扩展 EPROM 中读取程序。

(2) 程序存储器低端的几个特殊单元

MCS－51 单片机复位后，程序存储器 PC 的内容为 0000H，故系统必须从 0000H 单元开始取指令，执行程序。程序存储器中的 0000H 地址是系统程序的启动地址，这一点要牢牢记住。一般在该单元存放无条件转移指令，跳向用户设计的主程序的起始地址。

另外，在程序存储器中，还有 40 个特殊单元 0003H～002AH。它们被均匀地分为 5 段(每段有 8 个单元)，用作 5 个中断服务程序的入口地址。如表 2－3 所示。

表 2－3　中断矢量地址

中断源	中断服务程序入口地址
外部中断 0	0003H
定时/计数器 0 溢出	000BH
外部中断 1	0013H
定时/计数器 1 溢出	001BH
串行口	0023H

由于两个中断入口间隔仅有 8 个单元，存放中断服务程序往往是不够用的。通常在这些中断入口地址处都放一条无条件跳转指令。

4. 中断系统

为提高单片机的实时处理外部或内部随机发生的事件，MCS－51 单片机具有 5 个中断源，即有 5 种情况发生时，单片机会去处理中断程序。

5. 定时/计数器

在单片机的使用过程中，为了精确地计时或者对外部事件进行计数，通过在单片机内部设置定时/计数器部件。51 单片机内部共有 2 个 16 位可编程定时/计数器。

6. 串行口

用来进行串行通讯，提供对数据各位按序一位一位地传输。MCS－51 中的串行口是一个全双工通信口，即能同时发送和接收数据。

7. 并行 I/O 口

用于单片机实现对外部设备控制，51 系列单片机有 4 个 8 位的并行双向 I/O 口，即 P0～P3 口。每个口既可以作为输入口，也可以作为输出口。

8. 时钟电路

时钟电路的作用是产生时钟信号(为脉冲信号)，能够使单片机按一定的时序来工作。通过外接石英晶体为 CPU 产生所谓的“心脏”。常用晶振的频率有 6MHz、11.0592MHz、12MHz。允许最高外接晶振频率为 12MHz。

2.2 引脚介绍

MCS－51 单片机采用 40 引脚双列直插封装(DIP)形式，对于 CHMOS 单片机除采用 DIP 封装形式外，还采用方形封装工艺。由于受到引脚数目的限制，所以有一些引脚具有第二功能。

图 2－2 是 MCS－51 的引脚图和逻辑符号，在单片机的 40 条引脚中，有 2 条专用于主电源的引脚，2 条外接晶体的引脚，4 条控制和其他电源复用的引脚，32 条输入输出的引脚。

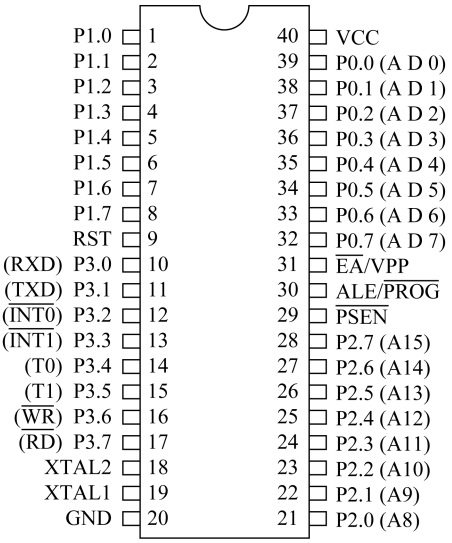


图 2－2　MCS－51 的引脚图

下面说明这些引脚的名称和功能。

1. 主电源引脚 V_{CC}和 V_{SS}

V_{CC}：接＋5V 电源。

V_{SS}：接地。

2. 时钟电路引脚 XTAL1 和 XTAL2

XTALl：接外部晶体的一端。在单片机内部，它是反相放大器的输入端，该放大器构成了片内振荡器。在采用外部时钟电路时该引脚必须接地。

XTAL2：接外部晶体的另一端。在单片机内部，接上述振荡器的反相放大器的输出端，振荡器的频率是晶体振荡频率。若采用外部时钟电路时，该引脚输入作为外部时钟的输入端。

3. 控制信号引脚 RST/V_{dd}、ALE/PROG、PSEN 和 EA V_{pp}

RST/V_{dd}：复位电源输入端。单片机上电后，只要在该引脚上输入 24 个振荡周期(2 个机器周期)宽度以上的高电平就会使单片机复位；在主电源 V_{CC}掉电期间，该引脚可接上－5V 备用电源。当 V_{CC}掉到低于规定的电平，而 V_{dd}在其规定的电压范围内时，V_{dd}就向片

内 RAM 提供备用电源，以保持片内 RAM 中的信息不丢失，复电后能继续正常运行。

ALE/$\overline{\text{PROG}}$：地址锁存使能输出/编程脉冲输入端。当 CPU 访问外部存储器时，ALE 的输入作为外部锁存地址的低位字节的控制信号；当不访问外部存储器时，ALE 却仍以 1/6 的时钟振荡频率固定的输出正脉冲。因此，它可用作对外输出的时钟或用于定时。

$\overline{\text{PSEN}}$：外部程序存储器读选通信号。CPU 在访问外部程序存储器期间，每个机器周期中信号两次有效。但在此期间，每当访问外部数据存储器时，这两次有效的信号不出现，可以驱动 8 个 LSTTL 负载。

$\overline{\text{EA}}/V_{\text{pp}}$：外部访同允许/编程电源输入，当输入高电平时，CPU 执行程序在低 4KB（0001H－0FFFH）地址范围内，访问片内程序存储器；在程序计数器 PC 的值超过 4KB 的地址时将自动转向执行片外程序存储器的程序；当输入低电平时 CPU 仅访问片外程序存储器。因此，对于 8031 来说，由于片内无程序存储器，因此 EA 必须接低电平。

在对 8751EPROM 编程时，此引脚接 +21V 的编程电压 V_{pp}来电擦除。

4. 输入/输出（I/O）引脚 P0、P1、P2 和 P3

P0.0－P0.7：P0 口是 8 位双向 I/O 端口。在访问片外存储器时，它分时提供低 8 位地址和作 8 位双向数据总线。在 EPROM 编程时，从 PO 口输入指令字节；在验证程序时，则输出指令字节（验证时，要外接上拉电阻）。PO 口能以吸收电流的方式驱动 8 个 LSTTL 负载。

P1.0－P1.7：P1 是 8 位准双向 I/O 端口。在 EPROM 编程时，它输入低 8 位地址。P1 口能驱动 4 个 LSTTL 负载。

P2.0－P2.7：P2 是 8 位准双向 I/O 端口。当 CPU 访问外部存储器时，它输出高 8 位地址。在对 EPROM 编程和验证程序时，它输入高 8 位地址。P2 口可驱动 4 个 LSTTL 负载。

P3.0－P3.7：P3 是 8 位准双向 I/O 端口。它是一个复用功能口。作为第一功能使用时，为普通 I/O 口，其功能和操作方向与 P1 口相同。作为第二功能使用时，各口线的第二功能见表 2－4。P3 口每一引脚均独立定义为第一功能的输入/输出或第二功能。P3 口能驱动 4 个 LSTTL 负载。

表 2－4　各口线的第二功能表

口线	第二功能
P3.0	RXD（串行口输入）
P3.1	TXD（串行口输出）
P3.2	$\overline{\text{INT0}}$（外部中断 0 输入）
P3.3	$\overline{\text{INT1}}$（外部中断 1 输入）
P3.4	T0（定时器 0 的外部输入）
P3.5	T1（定时器 1 的外部输入）
P3.6	$\overline{\text{WR}}$（外部数据存储器“写”信号输出）
P3.7	$\overline{\text{RD}}$（外部数据存储器“读”信号输出）

可以看出，P3 口的第二功能包含：串行输入输出、外部中断控制、定时器外部输入控制及外部存储器写控制。由于单片机没有专设的控制信号引脚，单片机在进行上述操作时所需要的控制信号必须由 P3 口提供，P3 口的第二功能相当于 PC 中的 CPU 的控制线引脚。

综上所述，由于 P0、P1、P2 及 P3 口的内部结构不同，因而功能也不相同，它们的带负载能力和接口要求也不尽相同，在使用时应注意以下几方面。

（1）P0～P3 口都是准双向 I/O 口，即 CPU 在读取数据时，必须先向相应端口的锁存器写入“1”。各端口名称与锁存器名称在编程时相同，均可用 P0～P3 表示。当系统复位时，P0～P3 口锁存器全为“1”，CPU 可直接对其进行读取数据。

（2）P0 口每一输出位可直接驱动 8 个 LS 型 TTL 负载（若超过则输出位应通过驱动电路与负载连接），P0 口可作为通用输入、输出口使用，此时，若要驱动 NMOS 或其他拉电流负载时，需外接上拉电阻，才能使该位高电平输出有效。

在单片机进行外部存储器扩展时，P0 口必须作为地址/数据复用总线使用，此时，不必外接上拉电阻，P0 口也不能作为通用 I/O 口使用。

（3）P1、P2、P3 口的输出均接有内部上拉电阻，输出端无须外接上拉电阻，每一位输出可以驱动 4 个 LS 型 TTL 电路（若超过则输出位应通过驱动电路与负载连接）。

（4）P0、P2、P3 口在无系统扩展时可以作为通用 I/O 口使用。应当注意，当 CPU 访问由系统扩展的外部存储器时，CPU 将自动把外部存储器的地址线信号（16 位）送 P0、P2 口，作为地址总线（P0 口输出低 8 位地址，P2 口输出高 8 位地址），向外部存储器输出 16 位存储单元地址。在控制信号（含 P3.6、P3.7）的作用下，该地址低 8 位被锁存后，P0 口自动切换为数据总线，这时经 P0 口可向外部存储器进行读、写数据操作。此时，P0 口为地址/数据复用口、P2 口不再作为通用 I/O 口、P3.7 或 P3.6 作为读或写控制信号输出。

（5）P3 口若不需要作为第二功能口时，则自动作为通用 I/O 口使用。当仅需要 P3 口的某些位作第二功能使用时，则另一些位宜作为位处理的 I/O 口使用。

2.3 复位电路

单片机的复位电路使单片机进入复位状态。通过复位操作完成单片机的初始化，也可以使处于死机状态下的单片机程序重新开始运行。图 2－3 是一个复位电路接到了单片机的 9 引脚即 RST（Reset）复位引脚上，这个复位电路如何起作用后边再讲，现在着重讲一下复位对单片机的作用。单片机复位一般分为以下三种情况：上电复位、手动复位和程序自动复位。

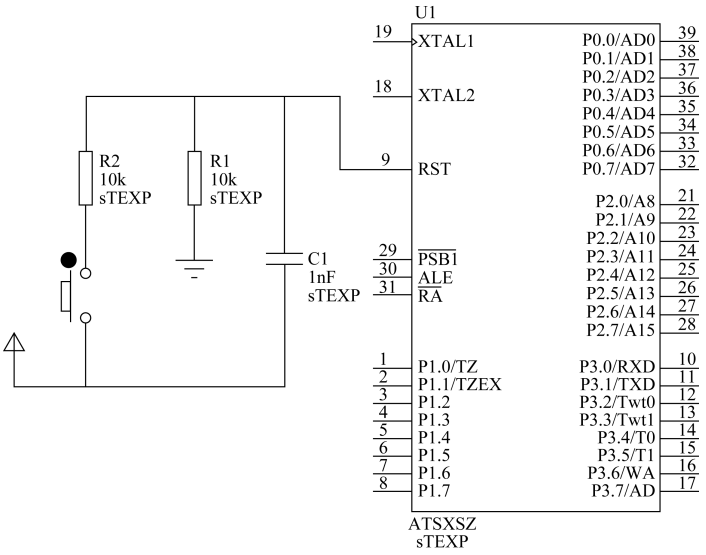


图 2－3　单片机复位电路

假如单片机程序有 100 行，当某一次运行到第 50 行的时候，突然停电了，这个时候单片机内部有的区域数据会丢失掉，有的区域数据可能还没丢失。那么下次打开设备的时候，人们希望单片机能正常运行，所以上电后，单片机要进行一个内部的初始化过程，这个过程就可以理解为上电复位，上电复位保证每次从一个固定的相同的状态开始工作。这个过程跟打开计算机电源的过程是一致的。

当程序运行时，如果受到意外干扰而导致程序死机，或者程序跑飞的时候，就可以按下一个复位按键，让程序重新初始化、重新运行，这个过程就叫作手动复位，最典型的就是计算机的重启按钮。

当程序死机或者跑飞的时候，单片机往往有一套自动复位机制。比如看门狗，具体应用以后再了解。在这种情况下，如果程序长时间失去响应，单片机看门狗模块会自动复位重启。还有一些情况是程序故意重启复位单片机。

电源、晶振、复位构成了单片机最小系统的要素，也就是说，一个单片机具备了这三个条件，就可以运行下载的程序了，其他的比如 LED 小灯、数码管、液晶等设备都是属于单片机的外部设备，即外设。最终完成用户想要的功能就是通过对单片机编程来控制各种各样的外设而实现的。

2.4　寄存器

2.4.1　PSW 寄存器

PSW(Program Status Word)全称为程序状态字标志寄存器，是一个 8 位寄存器，位于单片机内的特殊功能寄存器区，字节地址 D0H，用来存放运算结果的一些特征。如有无进

位、错位等，使用汇编程序时 PSW 寄存器很有用，但用 C 语言时，编译器会自动控制该寄存器，很少人为操作它，大家只需要做简单了解即可。其每一位如表 2－5 所示。

表 2－5　PSW 寄存器

	D7	D6	D5	D4	D3	D2	D1	D0	
PSW	CY	AC	F0	RS1	RS0	OV	－	P	D0H

（1）CY－进位标志位，它表示运算是否有进位（或借位）。如果操作结果在最高位有进位（加法）或者借位（减法），则该位为 1，否则为 0。

（2）AC－辅助进位标志，又称半进位标志，它指两个 8 位数运算低四位是否有半进位，即低四位相加（或相减）是否进位（或借位），如有 AC 为 1，否则为 0。

（3）F0－由用户使用的一个状态标志位，可用软件使它置 1 或清 0，也可由软件来测试它，以控制程序的流向。

（4）RS1，RS0－4 组工作寄存器区选择控制位，在汇编语言中，这两位用来选择 4 组工作寄存器区中的哪一组作为当前工作寄存区。

（5）OV－溢出标志位，反映带符号数的运算结果是否有溢出。有溢出时，此位为 1，否则为 0。

（6）P－奇偶标志位，反映累加器 ACC 内容的奇偶性，如果 ACC 中的运算结果有偶数个 1（如 11001100B，其中有 4 个 1），则 P 为 0，否则 P 为 1。

2.4.2　中断允许寄存器 IE

中断允许寄存器用来设定各个中断源的打开和关闭，IE 在特殊功能寄存器中，字节地址 A8H，位地址（由低位到高位）分别是 A8H～AFH。该寄存器可进行位寻址，即可对该寄存器的每一位进行单独操作。单片机复位时 IE 全部被清 0，各位定义见表 2－6。

表 2－6　寄存器位定义

位序号	D7	D6	D5	D4	D3	D2	D1	D0
位符号	EA	——	ET2	ES	ET1	EX1	ET0	EX0
位地址	AFH	——	ADH	ACH	ABH	AAH	A9H	A8H

EA—全局中断允许位。

EA＝1，打开全局中断控制，在此条件下，由各个中断控制位确定相应中断的打开或关闭。

EA＝0，关闭全部中断。

——无效位。

ET2 定时器/计数器 2 中断允许位。

ET2＝1，打开 T1 中断。

ET2＝0，关闭 T2 中断。
ES－串行口中断允许位。
ES＝1，打开串行口中断。
ES＝0，关闭串行口中断。
ET1 定时器/计数器 1 中断允许位。
ET1＝1，打开 T1 中断。
ET1＝0，关闭 T1 中断。
EX1 外部中断 1 中断允许位。
EX1＝1，打开外部中断 1 中断。
EX1＝0，关闭外部中断 1 中断。
ET0 定时器/计数器 0 中断允许位。
ET0＝1，打开 T0 中断。
ET0＝0，关闭 T0 中断。
EX0 外部中断 0 中断允许位。
EX0＝1，打开外部中断 0 中断。
EX0＝0，关闭外部中断 0 中断。

2.4.3　中断优先级寄存器 IP

中断优先级寄存器在特殊功能寄存器中，字节地址为 B8H，位地址(由低位到高位)分别是 B8H～BFH，IP 用来设定各个中断源属于两级中断中的哪一级。该寄存器可进行位寻址，即可对该寄存器的每一位进行单独操作。单片机复位时 IP 全部被清 0，各位定义见表 2－7。

表 2－7　寄存器位定义

位序号	D7	D6	D5	D4	D3	D2	D1	D0
位符号	——	——	——	PS	PT1	PX1	PT0	PX0
位地址	——	——	——	BCH	BBH	BAH	B9H	B8H

——无效位。
PS 串行口中断优先级控制位。
PS＝1，串行口中断定义为高优先级中断。
PS＝0，串行口中断定义为低优先级中断。
PT1 定时器/计数器 1 中断优先级控制位。
PT1＝1，定时器/计数器 1 中断定义为高优先级中断。
PT1＝0，定时器/计数器 1 中断定义为低优先级中断。
PX1 外部中断 1 中断优先级控制位。
PX1＝1，外部中断 1 定义为高优先级中断。

PX1 = 0，外部中断 1 定义为低优先级中断。

PT0 定时器/计数器 0 中断优先级控制位。

PT0 = 1，定时器/计数器 0 中断定义为高优先级中断。

PT0 = 0，定时器/计数器 0 中断定义为低优先级中断。

PX0 外部中断 0 中断优先级控制位。

PX0 = 1，外部中断 0 定义为高优先级中断。

PX0 = 0，外部中断 0 定义为低优先级中断。

在 51 单片机系列中，高优先级中断能够打断低优先级中断以形成中断嵌套，同优先级中断之间，或低级对高级中断则不能形成中断嵌套。若几个同级中断同时向 CPU 请求中断响应，在没有设置中断优先级情况下，按照默认中断级别响应中断，在设置中断优先级后则按设置顺序确定响应的先后顺序。

2.5 时　序

8051 单片机内部有个高增益反向放大器，用于构成振荡器，反向放大器的输入端为 XTAL1，输出端为 XTAL2，分别是 8051 的 19 和 18 脚。在 XTAL1 和 XTAL2 之间接一个石英晶体及两个电容就可以构成稳定的自激振荡器，如图 2 - 11 所示。晶体振荡器的振荡信号经过片内时钟发生器进行 2 分频，向 CPU 提供两相时钟信号 P1 和 P2。时钟信号的周期被称为状态时间 S，它是振荡周期的 2 倍，在每个状态的前半周期 P1 信号有效，在每个状态的后半周期 P2 信号有效，CPU 就以这两相时钟信号为基本节拍指挥单片机各部分协调工作。

CPU 执行一条指令所需要的时间是以机器周期为单位的。8051 单片机的一个机器周期包括 12 个振荡周期，分为 6 个 S 状态：S1 ~ S6。每个状态又分为 2 拍，即前面介绍的 P1 和 P2 信号。因此一个机器周期中的 12 个振荡周期可表示为 S1P1，S1P2，S2P1，…，S6P1，S6P2。

当使用 12MHz 的晶体振荡器时，一个机器周期为 1μs。CPU 执行一条指令通常需要 1 ~ 4个机器周期，指令的执行速度与其需要的机器周期数直接有关，所需机器周期数越少，速度越快，8051 单片机只有乘、除两条指令，需要 4 个机器周期，其余均为单周期或双周期指令。

图 2 - 11 为几种典型的取指令和执行时序。从图中可以看到，在每个机器周期之内，地址锁存信号 ALE 两次有效，第一次出现在 S1P2 和 S2P1 期间，第二次出现在 S4P2 和 S5P1 期间。单个周期指令的执行从 S1P2 开始，此时操作码被锁存在指令寄存器内。若是双字指令，则在同一机器周期的 S4 状态读第 2 个字节。若是单字节指令，在 S4 状态仍在读，但操作无效，且程序计数器 PC 的值不等于 1。

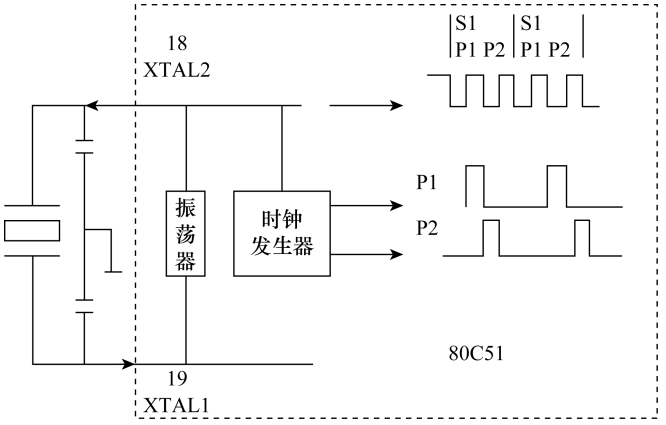


图 2－11　8051 片内振荡器内及时钟发生电路

2.6　MCS－51 的工作方式

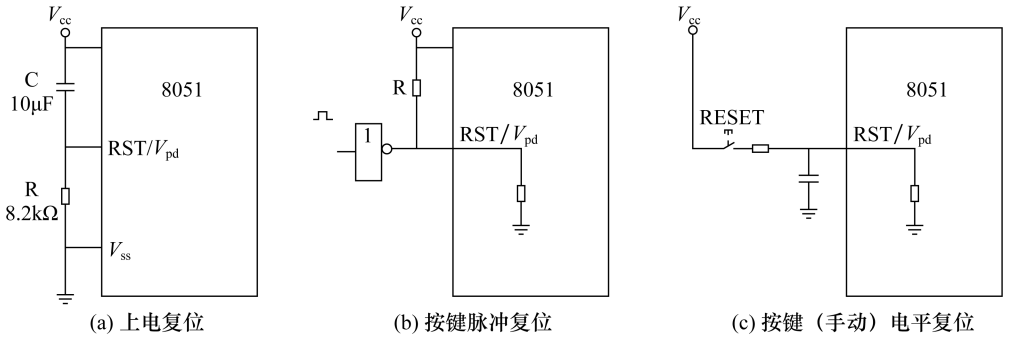
MCS－51 单片机的工作方式包括复位方式、程序执行方式、节电方式和 EPROM 的编程和校验方式。在不同的情况下，其工作方式也不相同。此处仅对前三种工作方式作简要介绍。

2.6.1　复位方式

单片机在启动运行时需要复位，使 CPU 以及其他功能部件处于一个确定的初始状态（如 PC 的值为 0000H），并从这个状态开始工作，单片机应用程序必须以此作为设计前提。

另外，在单片机工作过程中，如果出现死机，也必须对单片机进行复位，使其重新开始工作。

MCS－51 的复位电路包括上电复位电路和按键（外部）复位电路。如图 2－12 所示。



(a) 上电复位　　(b) 按键脉冲复位　　(c) 按键（手动）电平复位

图 2－12　复位电路

不管是何种复位电路，都是通过复位信号（高电平有效）由 RST/V_{pd} 引脚送入到内部的复位电路，对 MCS－51 进行复位。复位信号要持续两个机器周期（24 个时钟周期）以上，

才能使 MCS－51 单片机可靠复位。

上电复位电路利用电容充电来实现复位。在图 2－12(a)中可以看出，上电瞬间，RST/V_{pd}端的电位与 V_{cc}等电位，RST/V_{pd}为高电平，随着电容充电电流的减少，RST/V_{pd}的电位不断下降，其充电时间常数为 10＊10^－6＊8.2＊10^3＝82x10^－3s＝82ms，此时间常数足以使 RST/V_{pd}在保持高电平的时间内完成复位操作。

按键复位电路又包括按键脉冲复位和按键电平复位。图 2－12(b)为按键脉冲复位电路，由外部提供一个复位脉冲，复位脉冲的宽度应大于两个机器周期。图 2－12(c)为按键电平复位电路，按下复位按键，电容 C 被充电，RST/V_{pd}端的电位逐渐升高为高电平，实现复位操作，按键释放后，电容经内部下拉电阻放电，RST/V_{pd}端恢复低电平。

单片机复位后，其片内各寄存器的状态见表 2－8。

表 2－8　复位后内部寄存器状态

寄存器	内容	寄存器	内容
PC	0000H	TH0	00H
ACC	00H	TL0	00H
B	00H	TH1	00H
PSW	00H	TL1	00H
SP	07H	SBUF	不定
DPTR	0000H	TMOD	00H
P0～P3	0FFH	SCON	00H
IP	×××00000B	PCON(HMOS)	0×××××××B
IE	0×000000B	PCON(CMOS)	0×××0000B
TCON	00H		

由表 2－8 可以看出，单片机复位后，程序计数器 PC＝0000H，指向程序存储器 0000H 单元，使 CPU 从首地址重新开始执行程序。MCS－51 复位时，其内部 RAM 中的数据保持不变。

2.6.2　程序执行方式

程序执行方式是单片机的基本工作方式，通常可分为连续执行和单步执行两种工作方式。

1. 连续执行方式

连续执行方式就是单片机正常执行控制程序的工作方式。

被执行的程序存储在片内或片外的 ROM 中，由于单片机复位后程序计数器 PC＝0000H，因此机器在加电或按键复位后总是到 0000H 处开始连续执行程序，由于 ROM 区开始的一些存储单元的特殊作用，可以在 0000H 处放一条转移指令，以便跳转到指定的程序存储器中的任一单元去执行程序。

2. 单步执行方式

用户在调试程序时，常常要一条一条地执行程序中的每一条指令。单步执行方式就是为用户调试程序而设计出的一种工作方式，可设置一个单步执行按键，当需要单步执行程

序时，可以按下该键，每按一次可以执行一条指令。

单步执行方式是利用单片机外部中断功能实现的。其原理为：单步执行键相当于外部中断的中断源，当它被按下时，相应电路就产生一个负脉冲送到单片机的外部中断输入端（INT0 或 INT1），MCS－51 单片机便能自动执行预先设计好的具有单步执行指令的中断服务程序，从而实现单步执行的功能。

2.6.3　节电工作方式

节电工作方式是一种能减少单片机功耗的工作方式，通常有空闲方式和掉电方式两种，只有 CHMOS 型器件才有这种工作方式。CHMOS 型单片机是一种低功耗器件，正常工作时电流为 11～20mA，空闲状态为 1.7～5mA 电流，掉电时为 5～50μA。因此，CHMOS 型单片机特别适用于低功耗的场合。

CHMOS 型单片机的节电工作方式是由特殊功能寄存器 PCON 控制的。PCON 的各位定义如下：

PCON.7	PCON.6	PCON.5	PCON.4	PCON.3	PCON.2	PCON.1	PCON.0
SMOD				GF1	GF0	PD	IDL

其中，SMOD 为串行口波特率倍率控制位；GF0、CF1 为通用标志位；PD 为掉电控制位，PD＝1 进入掉电方式；IDL 为空闲控制位，IDL＝1 进入空闲方式。

当 PD 与 IDL 同时为 1 时，先进入掉电控制方式。

1. 掉电方式

单片机在运行过程中，如果发生掉电，片内 RAM 和 SPR 中的信息将会丢失。为防止信息丢失，可以把一组备用电源加到 RST/V_{pd}端，当 V_{cc}上的电压低于 V_{pd}上的电压时，备用电源通过 V_{pd}端，以低功耗保持内部 RAM 和 SFR 中的数据。

利用这种方法，可以设计一个掉电保护电路，当外部电路检测到即将发生掉电时，立即通过外部中断输入端 INT0 通知 CUP，CUP 执行中断服务程序把有关数据保存到内部 RAM 中，然后执行如下指令，将 PD 设置为 1，即可进入掉电工作方式：

```
MOV PCON，#02H
```

在掉电方式下，内部 RAM 的 00H～7FH 中的数据被保存下来，不会丢失。在掉电期间，电源 V_{cc}电压可以降到 2V，内部 RAM 耗电电流为 50μA。当电源电压恢复到 5V 后，硬件复位 10ms 后可以使单片机退出掉电方式。

80C31 复位后 SFR 重新初始化，但 RAM 中的内容保持不变。因此，若要使 80C31 在退出掉电方式后能继续执行原来的程序，就必须在掉电前预先把 SFR 中的内容保护到片内 RAM，在掉电方式退出后，从 RAM 中把被保护的数据取出，送回到 SFR，恢复 SFR 中原来的内容。

2. 空闲方式

80C31 执行如下指令可以将 IDL 设置为 1，从而进入空闲方式：

```
MOV PCON，#01H
```

进入空闲方式后，CPU 停止工作，但中断、串行口和定时/计数器可以继续工作。此时，CPU 中的 SP、PC、PSW、ACC 及 SFR 中的其他寄存器和内部 RAM 中的内容均保持不变，时钟电路继续工作，ALE 和 PSEN 变为高电平，无脉冲输出，处在无效状态。

在空用工作方式期间，如果有中断产生，单片机通过内部的硬件电路自动使 IDL = 0，CPU 从空闲方式中退出，继续执行原来的程序。

除了以上工作方式外，对于 MCS－51 中的 EPROM 型单片机(8751)，用户可以把程序写入 EPROM，并能通过暴露在紫外光下进行擦除。也可以将 EPROM 的内容读出进行校验。另外，8751 片内有一个保密位，一旦将该位写入，就可以禁止任何外部方法对片内程序存储器进行读/写操作，而且只能执行片内 EPROM 的程序，只有将 EPROM 全部擦除，保密位才能被一起擦除，以便再次建立。

本章小结

本章主要介绍了 MCS－51 单片机的硬件结构、引脚及功能、复位电路、时序的基本内容。通过学习本章，让读者初步了解单片机的基本结构。

本章参考文献

[1] 郭天祥等. C51 单片机 C 语言教程. 北京：电子工业出版社，2009.
[2] 赵建领、薛园园等. 零基础学单片机 C 语言程序设计(第二版). 北京：机械工业出版社，2012.
[3] 宋雪松等. 手把手教你学 51 单片机(C 语言版). 北京：清华大学出版社，2014.
[4] 何应俊等. 51 单片机 C 语言编程. 北京：机械工业出版社，2014.
[5] 李群芳等. 单片微机计算机与接口技术(第 4 版). 北京：电子工业出版，2012.

本章习题

2.1 在什么情况下，P3 口作为第二功能使用？
2.2 单片机最小系统包括哪些部分？各部分的功能是什么？
2.3 MCS－51 存储器结构的主要特点是什么？
2.4 举例说明 MCS－51 有哪些典型产品，它们有何区别？
2.5 中断允许寄存器 IE 各位的定义是什么？
2.6 MCS－51 单片机的 P0－P3 四个口在结构上有何异同？使用时应注意哪些事项？

第 3 章 系统指令与汇编程序设计

本章教学目标

- 了解 51 单片机指令的基本格式和各组成部分的功能；
- 了解指令系统中常用符号的书写形式及含义；
- 理解 51 单片机的 7 种寻址方式的形式、寻址范围和特点；
- 理解掌握 51 单片机的基本汇编指令的形式、功能和简单应用。

本章重点内容

- 51 单片机指令系统的寻址方式；
- 51 单片机的基本汇编指令的功能和简单应用。

3.1 单片机指令系统

计算机通过执行程序完成人们指定的操作，程序由一条条指令组成。一台计算机所能执行的全部指令集合就是该计算机 CPU 的指令系统。指令系统是决定计算机性能强弱的重要因素。

51 单片机的指令系统包括算术运算类、数据传送类、逻辑操作类、控制程序转移类、位操作类等指令。51 单片机的指令由操作码和操作数两部分组成。指令格式如下：

操作码　目的操作数，源操作数；注释

其中，操作码指明该指令所实现的操作功能，操作数部分指出了参与操作的数据来源和操作结果存放的目的单元。操作数可以是一个具体的数(立即数)，也可以是一个数据所在的空间地址。注释部分是对程序的说明，用分号“;”与指令隔开。

操作码和操作数都有对应的二进制代码，可以直接被计算机识别和执行。指令代码由若干字节组成，不同的指令对应的字节数不同。51 单片机包括单字节、双字节和三字节

三种指令。通常，指令字节数越少，所占存储单元越少，指令执行速度越快。在编程时尽量选用指令字节数少的指令。

3.1.1 寻址空间及常用符号

51 单片机指令系统中常用的符号如下：

- Rn(n=0～7)：当前选中的工作寄存器组 R0～R7。
- Ri(i=0，1)：可作为地址指针的两个工作寄存器 R0 和 R1。
- #data：8 位立即数，即包含在指令中的 8 位常数。
- #data16：16 位立即数，即包含在指令中的 16 位常数。
- direct：8 位直接字节地址(片内 RAM 和特殊功能寄存器 SFR)。
- addr11：11 位目的地址，用于 ACALL 和 AJMP 指令中。
- addr16：16 位目的地址，用于 LCALL 和 LJMP 指令中。
- rel：8 位带符号地址偏移量。地址偏移量在 -128～+127 范围内。
- bit：位地址。
- @：间接地址方式中，表示间址寄存器的符号。
- /：位操作指令中，表示对该位先取反再参与操作，但不影响该位原值。
- ()：用于注释中表示存储单元的内容。
- (( ))：用于注释中表示存储单元指出的地址单元中的内容。
- $：当前指令的地址

3.1.2 寻址方式

在计算机中，寻址方式是指令中提供操作数的方式，它可以是操作数本身，也可以是操作数所在地址。在 51 单片机中，操作数可以存放在：片内 RAM、特殊功能寄存器 SFR、外部 RAM 和程序存储器 ROM 中。为了区分不同存储空间的数据操作，采用不同的寻址方式，51 单片机的指令系统共使用了 7 种寻址方式：寄存器寻址、直接寻址、立即寻址、寄存器间接寻址、变址寻址、相对寻址、位寻址。

1. 立即寻址

指令中地址码部分给出的就是操作数本身的寻址方式就是立即寻址。该操作数称为立即数，立即数前面加“#”号，以区别直接地址，可以是 8 位数(#data，如#40H)，也可以是 16 位数(#data16，如#2180H)。立即寻址的指令多为 2 字节或 3 字节指令。

例如：

```
MOV A，#65H          ；65H→A，机器码 7465
MOV DPTR，#1050H ；1050H→DPTR，机器码为 901050
```

第一条指令功能是把 65H 这个数本身送累加器 A。该指令为 2 字节指令。

第二条指令功能是把 1050H 这个数本身送给 DPTR，操作码后是两字节立即数，所以该指令为 3 字节指令。

2. 直接寻址

指令中地址码部分直接给出操作数所在的有效地址的寻址方式就是直接寻址。可用于直接寻址的空间有内部数据存储器 RAM 的低 128 字节(包括其中的位寻址区)和特殊功能寄存器 SFR。指令中，直接地址通常采用 direct(或 addr11 或 addr16)表示。

例如：

```
MOV A，20H          ；(20H)→A，机器码 E520
MOV 20H，30H        ；(30H)→20H，机器码 853020
```

第一条指令功能为把内部 RAM 20H 单元的内容送入累加器 A。

第二条指令功能为把内部 RAM 30H 的内容送给 20H。

对于特殊功能寄存器，既可以使用它们的地址，也可以使用它们的名字。例如：

```
MOV A，P0           ；(P0 口)→A
```

该指令是把特殊功能寄存器 SFR 中 P0 口内容送 A，它又可写成：

```
MOV A，80H
```

其中，80H 是 P0 口的地址。

3. 寄存器寻址

以指定通用寄存器的内容为操作数的寻址方式就是寄存器寻址。通用寄存器包括：寄存器组 R0～R7、内部累加器 A、通用寄存器 B、DPTR。

例如：

```
MOV A，R0           ；(R0)→A，A 和 R0 均为寄存器寻址，机器码为 E8，占 1 个字节
MOV P1，A           ；(A)→P1 口
ADD A，R0           ；(A)+(R0)→A
```

指令中给出的操作数是一个寄存器名称，该寄存器中的内容为真正被操作对象。

4. 寄存器间接寻址

寄存器间接寻址的特点是操作数的地址存放在某个寄存器中，以该寄存器的内容为地址，该地址中的内容为操作数，简称为寄存器间址。

51 单片机可用于寄存器间址的寄存器有 R0、R1，可寻址内部数据存储器 RAM 低位地址的 128 字节单元内容。如果要寻址外部数据存储器的 64KB 空间，则需要由 P2 端口提供外部 RAM 高 8 位地址，由 R0 或 R1 提供低 8 位地址，从而共同寻址整个空间。另外还可采用数据指针(DPTR)作为间址寄存器，寻址外部数据存储器 RAM 的 64KB 空间。使用时前面加@ 表示，如：@ R0、@ R1、@ DPTR。例如：

```
MOV A，@ R0         ；((R0))→A，以 R0 中内容为地址的操作数→A
MOVX A，@ R1        ；外部数据存储器 RAM(地址为 P2R1)的内容→A
MOVX @ DPTR，A      ；A→以 DPTR 内容为地址的外部数据存储器 RAM
```

5. 变址寻址

变址寻址是以某个寄存器(数据指针 DPTR 或程序计数器 PC)的内容为基地址，然后在这个基地址的基础上加地址偏移量(累加器 A 中)形成真正的操作数地址。操作数地址

为 16 位程序存储器地址。变址寻址只能对程序存储器中的数据寻址，寻址范围为 64KB。程序存储器是只读存储器，所以这种寻址操作只能读数据而不能写入。指令操作符为 MOVC。

例如：

```
MOVC A，@A+DPTR    ；((A)+(DPTR))→A
MOVC A，@A+PC      ；((A)+(PC))→A
```

6. 相对寻址

相对寻址以当前程序计数器 PC 的值加上指令中规定的偏移量 rel 而形成新的实际的转移地址。相对寻址只在相对转移指令中使用，只用于修改 PC 值。如：

```
SJMP 35H           ；当前 PC+2+35H→PC
```

该指令是 2 字节指令，因此，指令执行后，转移到当前 PC+2+35H 处执行程序。

7. 位寻址

对位地址中的内容作位操作的寻址方式称为位寻址。采用位寻址方式的指令的操作数是 8 位二进制数中的某一位。51 单片机中片内数据存储器 RAM 有两个区域可以位寻址：一个是 20H～2FH 的 16 个单元中的 128 位，另一个是字节地址能被 8 整除的特殊功能寄存器 SFR。位寻址是一种直接寻址方式，在指令中给出直接位地址。其中，特殊功能寄存器中的操作数可直接用寄存器名字加位数表示，如 PSW.3。

例如：

```
MOV C，5FH     ；(5FH)→Cy，5FH 为位地址的物理形式。
MOV C，2BH.7   ；(2BH.7)→Cy，采用第几字节单元第几位的表示法，它表示 2BH 单元中最高位。
MOV C，ACC.7   ；ACC.7→Cy，把 ACC.7 位状态送到进位标志位 Cy。
```

3.1.3 数据传送与交换指令

数据传送是一种最基本、最主要的操作。数据传送操作可以在片内数据存储器 RAM 和特殊功能寄存器 SFR 内进行，也可以在累加器 A 和片外数据存储器 RAM 之间进行。数据传送指令必须指定传送数据的源地址和目的地址，以便实现把源地址中的内容传送到目的地址中，但不改变源地址中的内容。

1. 内部数据传送指令

(1) 以累加器 A 为目的操作数的指令

这组指令是将工作寄存器 Rn(即 R0～R7)内容、直接寻址或间接寻址(Ri 为 R0 或 R1)所得的片内数据存储器 RAM 单元或特殊功能寄存器 SFR 中的内容以及立即数传送到累加器 A 中。指令格式如下：

```
汇编指令格式          说明
MOV A，Rn             ；(Rn)→A
MOV A，direct         ；dir→A
```

MOV A，@Ri　　；((Ri))→A

MOV A，#data　　；#data→A

【例 3－1】已知：R0＝25H，(25H)＝65H，执行指令 MOV A，@R0 后，A＝65H。

(2) 以 Rn 为目的操作数的指令

这组指令的功能是把源操作数所指定的内容送到当前工作寄存器组 R0～R7 中的某个寄存器。源操作数有寄存器寻址、直接寻址和立即寻址 3 种方式。指令格式如下：

汇编指令格式　　说明

MOV Rn，A　　；A→Rn

MOV Rn，direct　　；direct→Rn

MOV Rn，#data　　；#data→Rn

【例 3－2】已知(A)＝50H，(60H)＝35H，执行下列指令的意义是：

MOV R5，A　　；A→R5，(R5)＝50H

MOV R5，60H　　；(60H)→R5，(R5)＝35H

MOV R5，#75H　　；75H→R5，(R5)＝75H

(3) 以直接地址为目的操作数的指令

这组指令的功能是把源操作数所指定的内容送入由直接地址 direct 所指出的片内数据存储单元 RAM 中。源操作数有寄存器寻址、直接寻址、寄存器间接寻址和立即寻址等方式。指令格式如下

汇编指令格式　　说明

MOV direct，A　　；A→direct

MOV direct，Rn　　；(Rn)→direct

MOV direct，direct　　；direct→direct

MOV direct，@Ri　　；((Ri))→direct

MOV direct，#data　　；#data→direct

【例 3－3】已知 R0＝45H，(45H)＝20H，指令 MOV 30H，@R0 执行后，(30H)＝20H。

(4) 以间接地址为目的操作数的指令

这组指令的功能是把源操作数所指定的内容送入由间接地址 Ri 中的内容所指定的 RAM 单元。指令格式如下：

汇编指令格式　　说明

MOV @Ri，A　　；A→(Ri)

MOV @Ri，direct　　；direct→(Ri)

MOV @Ri，#data　　；#data→(Ri)

【例 3－4】已知 R0＝45H，指令 MOV@R0，#60H 执行后，(45H)＝60H。

2. 外部数据传送指令

(1) 以 DPTR 为目的操作数的指令

该指令为 16 位立即数传送指令，其功能是把 16 位常数送入 DPTR。DPTR 由 DPH 和

DPL 组成。这条指令执行的结果是，将高 8 位立即数 dataH 送入 DPH，低 8 位立即数 dataL 送入 DPL。指令格式如下：

```
汇编指令格式              说明
MOV DPTR，#data16    ；#data16→DPTR
```

【例 3－5】执行 MOV DPTR，#1234H 指令后，DPTR＝1234H。

（2）外部数据存储器与累加器 A 之间的传送指令

在 51 系列单片机中，只能通过累加器 A 与外部数据存储器 RAM 进行数据传送。这类指令可以实现外部数据存储器 RAM 和累加器 A 之间的数据传送。指令格式如下：

```
汇编指令格式              说明
MOVX A，@Ri          ；((Ri))→A
MOVX @Ri，A          ；(A)→(Ri)
MOVX A，@DPTR        ；((DPTR))→A
MOVX @DPTR，A        ；(A)→(DPTR)
```

其中，前两条指令是用 R0 或 R1 作低 8 位地址指针，P2 口输出高 8 位地址。寻址范围是 256 字节，用于访问外部 RAM 的低地址区；后两条指令以 DPTR 为片外数据存储器 16 位地址指针，寻址范围达 64KB。其功能是在 DPTR 所指定的片外数据存储器与累加器 A 之间传送数据。

【例 3－6】将片外数据存储器 RAM 中的 3040H 单元数据送入累加器 A。

```
方法一：MOV   P2，#30H        方法二：MOV   DPTR，#3040H
        MOV   R0，#40H                MOVX   A，@DPTR
        MOVX   A，@R0
```

（3）外部程序存储器 ROM 的字节传送指令

程序存储器只能读，不能写，所以共两条指令，均属于变址寻址指令，专用于查表，又称为查表指令。指令格式如下：

```
汇编指令格式              说明
MOVC A，@A＋DPTR  ；((A)＋(DPTR))→A
MOVC A，@A＋PC    ；(PC)＋1→PC，((A)＋(PC))→A
```

前一条指令以 DPTR 为基址寄存器进行查表。后一条指令，CPU 读取单字节指令“MOVC A，@A＋PC”后，PC 的内容先自动加 1，将新的 PC 内容与累加器 A 中的 8 位无符号数相加形成地址，取出该地址单元中的内容送累加器 A。

【例 3－7】已知(R0)＝02H 中，用查表的办法确定它的平方值。

```
MOV DPTR，#TABLE          ；TABLE 的地址送 DPTR
MOV A，R0                 ；A＝02
MOVC A，@A＋DPTR          ；执行完该指令，A＝(02＋DPTR)
TABLE：DB 0，1，4，9，16，25  ；定义单字节数据
```

程序中 DB 为在程序存储器中定义字节伪指令，编译时，DB 后面的数据将视为纯数

据，而不是指令。MOVC 指令把地址为(TABLE +2)单元的内容送到 A，因此该程序段执行结果为：A =4。

以上程序段也可用下列程序段代替：

```
MOV A，R0                    ；A =02
MOVC A，@A +PC               ；取完该指令 PC =TABLE，执行(2 +TABLE)→A
TABLE：DB 0，1，4，9，16，25
```

3. 堆栈操作指令

堆栈是用于暂存信息的一个存储区域，而堆栈操作指令是根据堆栈指示器 SP 中栈顶地址进行数据传送操作。在 51 单片机中，堆栈被安排在片内 RAM 的 128 字节单元中，初始化时 SP =07H，执行一次指令，完成一字节数据操作。相关指令如下：

(1) 入栈操作

```
汇编指令格式              说明
PUSH direct               ；(SP) +1→SP，(direct)→(SP)
```

入栈操作时，先将栈指针(SP) +1 指向栈顶的上一个空单元，再将直接地址(direct)寻址的单元内容送到当前 SP 所指示的堆栈单元中。

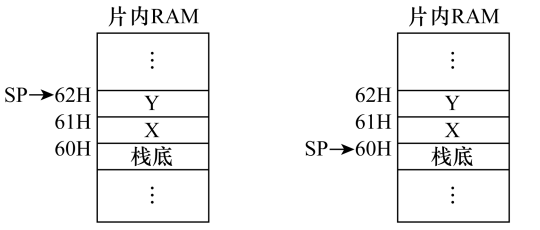
(2) 出栈操作

```
汇编指令格式              说明
POP direct                ；((SP))→direct，(SP) -1→(SP)
```

出栈操作时，先将栈指针(SP)所指示的内部 RAM(堆栈)单元中内容送入由直接地址寻址的单元中，然后(SP) -1→(SP)。

【例 3 -8】设(20H) = X，(30H) = Y，利用堆栈完成两个单元的数据交换。

```
MOV SP，#60H              ；栈底地址为 60H
PUSH 20H                  ；(SP) +1→SP，(20H)→61H，即将 X 入栈
PUSH 30H                  ；(SP) +1→SP，(30H)→62H，将 Y 入栈
POP 20H                   ；(62H)→20H，(SP) -1→(SP)，将 Y 出栈，送入 20H
POP 30H                   ；(61H)→30H，(SP) -1→(SP)，将 Y 出栈，送入 30H
```



(a) X、Y入栈后的堆栈　　(b) X、Y出栈后的堆栈

堆栈操作指令是直接寻址指令，要注意其书写格式。例如以下指令，左边的是正确的，右边的是错误的。

```
正确指令                  错误指令
PUSH ACC                  PUSH A
```

PUSH 30H　　PUSH R0

POP ACC　　POP A

POP 30H　　POP R0

4. 交换指令

(1) 字节交换指令

字节交换指令的功能是将第二操作数所指定的工作寄存器 Rn(R0 ~ R7)的内容、直接寻址或间接寻址的单元内容与累加器 A 中的内容互换。

汇编指令格式	说明
XCH A，Rn	；(A)←→(Rn)
XCH A，direct	；(A)←→(direct)
XCH A，@Ri	；(A)←→((Ri))

(2) 低半字节交换

低半字节交换指令的功能是将内部 RAM(Ri)的单元内容与累加器 A 中内容的低 4 位互换，高 4 位内容不变。该操作只影响标志位 P。

汇编指令格式	说明
XCHD A，@Ri	；($A_{0\sim3}$)←→(($Ri_{0\sim3}$))

(3) 累加器 A 的高、低半字节交换

该指令的功能是将累加器 A 的高低两个半字节交换。

汇编指令格式	说明
SWAP A	；($A_{0\sim3}$)←→($A_{4\sim7}$)

【例 3-9】(A) = D9H。执行指令“SWAP A”的结果为(A) = 9DH。

3.1.4 算术运算指令

51 单片机指令系统中算术运算指令包括加、进位加、借位减、加 1、减 1、乘、除指令。

通常，此类指令都使用累加器 A 来存放第一操作数，第二操作数可以存放在任何一个工作寄存器 Rn 或片内 RAM 单元中，也可以是指令码中的一个立即数。执行指令后的操作结果仍然保留在累加器 A 中。

在 51 单片机的程序状态字 PSW 寄存器中，有 4 个测试标志位：AC(辅助进、借位标志)、CY(进、借位标志)标志位、OV(溢出标志位)和 P(奇偶标志位)。算术运算结果将使 CY、AC、OV 三个标志位置位或复位，有加 1 和减 1 指令不影响这些标志位。

(1) 不带进位的加法指令

此类指令功能是将工作寄存器、内部 RAM 单元内容或立即数的 8 位无符号二进制数和累加器 A 中的数相加，运算结果仍然存放于累加器 A 中。当运算结果的第 3 位或第 7 位有进位时，分别将 AC(辅助进、借位标志)和 CY(进、借位标志)标志位置 1；否则为 0。

上述指令的执行将影响标志位 AC、CY、OV(溢出标志位)和 P(奇偶标志位)。其中，

溢出标志位 OV 只有带符号数运算时才有用。

汇编指令格式	说明
ADD A，Rn	；(A) + (Rn)→A
ADD A，direct	；(A) + (direct)→A
ADD A，@ Ri	；(A) + ((Ri))→A
ADD A，# data	；(A) + # data→A

【例 3 – 10】设(A) = 0C5H，(R0) = 0A6H。执行指令“ADD A，R0”所得和为 6BH。标志位 CY = 1，OV = 1，AC = 0。

```
    (A):   1100 0101
+   (R0):  1010 0110
-------------------
    1      0110 1011
```

溢出标志 OV 在 CPU 内部根据“异或”门输出置位，OV = C7⊕C6。

(2) 带进位的加法指令

此类指令的功能是把指令中规定的源操作数、进位标志位 CY 都与累加器 A 中的操作数相加，并将结果存放在累加器 A 中，其余的功能和上面 ADD 指令相同。这里的 CY 中的值是当前指令执行前的 CY 值，不是指令执行中形成的 CY 值。

汇编指令格式	说明
ADDC A，Rn	；(A) + CY + (Rn)→A
ADDC A，direct	；(A) + (direct) + CY→A
ADDC A，@ Ri	；(A) + ((Ri)) + CY→A
ADDC A，#data	；(A) + #data + CY→A

当运算结果第 3 和第 7 位产生进位或溢出时，分别置位 AC、CY 和 OV 标志位。本指令的执行将影响标志位 AC、CY、OV 和 P。

```
    (A):   1100 0101
+   (CY):  0000 0001
-------------------
           1100 0110
+   (R0):  1010 0110
-------------------
    1      0110 1100
```

【例 3 – 11】设(A) = 0C5H，(R0) = 0A6H，(CY) = 1。

执行指令“ADDC A，R0”得到的和 6CH 存于 A 中。

标志位 CY = 1，OV = 1，AC = 0。

(3) 带借位减法指令

带借位减法指令的功能是把累加器 A 中的操作数减去源地址所指出的操作数及指令执行前进位标志位 CY 的值，并把运算结果差值保留在累加器 A 中。

汇编指令格式	说明
SUBB A，Rn	；(A) – CY – (Rn)→A
SUBB A，direct	；(A) – CY – (direct)→A
SUBB A，@ Ri	；(A) – CY – ((Ri))→A
SUBB A，# data	；(A) – CY – #data→A

说明：

无论相减的两数是无符号数还是带符号数，减法操作总是按带符号二进制数进行，并对程序状态字 PSW 中各标志位产生影响。如果最高位在做减法时有借位，那么 CY 置 1，

否则 CY 清零；若低 4 位在做减法时向高 4 位有借位，则 AC 置 1，否则 AC 清零；若减法时最高位有借位而次高位无借位或最高位无借位而次高位有借位，则 OV 置 1，否则 OV 清零，若 OV 为 1，表示差数溢出，即破坏了正确的结果；奇偶校验标志 P 和加法时的取值相同。

在 51 单片机指令系统中，没有不带 CY 的减法指令。如果需要，可以在“SUBB”指令前用“CLR C”指令将 CY 清 0。

【例 3－12】设累加器 A 内容为 0D9H，寄存器 R2 内容为 66H，进位标志 CY＝1。

执行指令“SUBB A，R2”的结果为(A)＝72H

标志位 CY＝0，AC＝0，OV＝1。

```
(A):     1101 1001
-)(CY):  0000 0001
         1101 1000
-) (R2): 0110 0110
         0111 0010
```

(4) 乘法指令

乘法指令的功能是，把累加器 A 和寄存器 B 中两个 8 位无符号整数相乘，把计算所得的 16 位积的低 8 位字节存放在累加器 A 中，高 8 位字节存放在 B 寄存器中。本指令执行过程中将对 CY(进、借位标志)、OV(溢出标志位)和 P(奇偶标志位)三个标志位产生影响。其中，CY 为 0；P 由累加器 A 中 1 的奇偶性决定；OV 表示乘积的大小，若乘积大于 0FFH，则 OV 置 1，否则清零(即 B＝0)。

汇编指令格式说明

MUL AB ；(A)×(B)→$\begin{cases} B_{15\sim8} \\ A_{7\sim0} \end{cases}$

【例 3－13】(A)＝6AH，(B)＝3FH。

执行指令“MUL AB”结果为(B)＝1AH，(A)＝16H，表示积(BA)＝1A16H，OV＝1。

(5) 除法指令

除法指令的功能是执行 A 除以 B 的操作，并将计算所得的整数商存放到累加器 A 中，余数存放到 B 寄存器中。其中，A 和 B 的内容均为 8 位无符号整数。指令执行后，CY(进、借位标志)和 OV(溢出标志位)均被清零。若原(B)＝00H，则结果无法确定，此时 OV＝1，而 CY 仍为 0。

汇编指令格式说明

DIV AB；(A)/(B)的商→A，(A)/(B)的余数→B

(6) 加 1 指令

加 1 指令的功能是将源地址所指定的单元内容加 1。其中，前 4 条指令是 8 位操作数，第 5 条指令是 16 位操作数。只有第一条指令对奇偶标志位 P 产生影响，其余指令均不对任何标志位产生影响。若原单元内容为 FFH，加 1 后溢出为 00H，也不会影响 PSW 标志。

汇编指令格式	说明
INC A	；(A)＋1→A
INC Rn	；(Rn)＋1→Rn
INC direct	；(direct)＋1→direct

INC @ Ri　　；((Ri)) +1→(Ri)

INC DPTR　　；(DPTR) +1→DPTR

(7) 减 1 指令

减 1 指令的功能是将源地址所指定的单元内容减 1。与加 1 指令一样，只有第一条指令对奇偶标志位 P 产生影响，其余指令均不影响 PSW 标志位状态。若原单元内容为 00H，则减 1 后为 FFH，也不会影响标志位。

汇编指令格式　　说明

DEC A　　；(A) -1→A

DEC Rn　　；(Rn) -1→Rn

DEC direct　　；(direct) -1→direct

DEC @ Ri　　；((Ri)) -1→(Ri)

(8) 十进制调整指令

这是一条专用指令，用于实现 BCD 运算。通常紧跟在加法指令后，用于对执行加法后累加器 A 中的操作结果进行十进制调整。

汇编指令格式　　说明

DAA　　；将累加器 A 中二进制数相加和调整为 BCD 码

指令按下列原则进行调整：

① 若($A_{0\sim3}$) > 9 或 AC = 1，则($A_{0\sim3}$) +6→$A_{0\sim3}$。

② 同时，若($A_{4\sim7}$) > 9 或 CY = 1，则($A_{4\sim7}$) +6→$A_{4\sim7}$。

即：若和的低 4 位大于 9 或有半进位，则低 4 位加 6；若和的高 4 位大于 9 或有进位，则高 4 位加 6。指令根据相加的和与标志自行进行判断，因此，该指令应紧跟在加法指令后，至少在加指令和该指令之间不能有影响标志的指令。

3.1.5　逻辑操作指令

(1) 逻辑与运算指令

逻辑与运算指令又称为逻辑乘指令，共有以下 6 条：

汇编指令格式　　说明

ANL A，Rn　　；(A) ∧ (Rn)→A

ANL A，di rect　　；(A) ∧ (direct)→A

ANL A，@ Ri　　；(A) ∧ ((Ri))→A

ANL A，#data　　；(A) ∧ #data→A

ANL direct，A　　；(direct) ∧ (A)→direct

ANL direct，#data　　；(direct) ∧ #data→direct

其中，前 4 条指令是以累加器 A 为目标操作数寄存器的指令，可以将累加器 A 的内容和源地址所指的内容按位进行逻辑“与”，并把操作结果送回累加器 A。后两条指令是以 direct 为目标地址的逻辑与指令，它们将 direct 中的内容和源地址所指的内容按位进行逻辑

“与”，结果存入 direct 目标单元中。

（2）逻辑或运算指令

逻辑或运算指令功能是将两个指定的操作数按位进行逻辑“或”，又称为逻辑加指令，可以用来对某个存储单元或累加器 A 中的数据进行变换，使某些位变为“1”而其余位不变。其中，前 4 条指令的操作结果存放在累加器 A 中，后两条指令的操作结果存放在直接地址单元中。

汇编指令格式	说明
ORL A，Rn	；(A)∨(Rn)→A
ORL A，direct	；(A)∨(direct)→A
ORL A，@Ri	；(A)∨((Ri))→A
ORL A，#data	；(A)∨# data→A
ORL direct，A	；(direct)∨(A)→direct
ORL direct，#data	；(direct)∨#data→direct

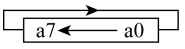
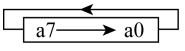
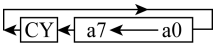
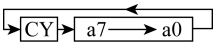
（3）逻辑“异或”指令

逻辑“异或”指令的功能是，将两个指定的操作数按位进行“异或”。可用来对某个存储单元或累加器 A 中的数据进行变换，使其中某些位取反而其余位不变。其中，前 4 条指令的结果存放在累加器 A 中，后两条指令的操作结果存放在直接地址单元中。

汇编指令格式	说明
XRL A，Rn	；(A)⊕(Rn)→A
XRL A，direct	；(direct)⊕(A)→A
XRL A，@Ri	；(A)⊕((Ri))→A
XRL A，#data	；(A)⊕# data→A
XRL direct，A	；(direct)⊕(A)→direct
XRL direct，# data	；(direct)⊕#data→direct

【例 3－14】设(A)＝0AAH，(R0)＝0FH，执行指令“XRL A，R0”后，结果为(A)＝0A5H，存于 A 中。

（4）专对 A 的指令

汇编指令格式	说明
A 清零：CLR A	；0→A，清 0 累加器 A，只影响标志位 P。
A 取反：CPL A	；$(\overline{A})$→A,对累加器 A 内容逐位取反,不影响标志位。
A 循环左移：RL A	； 
A 循环右移：RR A	； 
A 连同进位位循环左移：RLC A	； 
A 连同进位位循环右移：RRC A	； 

循环移位指令通常用于位测试、位统计、乘 2、除 2 等操作。

【例 3 -15】无符号 8 位二进制数(A) = 10101110B = AEH，CY = 0。将(A)乘 2，执行指令“RLC A”的结果为(A) = 01011100B = 5CH，CY = 1。15CH 正是 BDH 的 2 倍。

3.1.6　控制转移指令

控制转移类指令具有修改程序计数器 PC 内容的功能，以达到控制程序执行流向的目的。51 单片机的控制转移类指令包括无条件转换指令、条件转移指令、子程序调用指令及返回指令等。所有这些指令的目标地址都是在 64KB 程序存储器地址空间内。

1. 无条件转移指令

无条件转移指令是指当程序执行到该指令时，程序无条件转移到指令操作数所提供的地址处执行。无条件转移类指令包括短转移、长转移、相对转移和间接转移 4 条。

(1) 短转移指令

汇编指令格式	说明
AJMP addr11	；(PC) + 2→PC，addr11→$PC_{0\sim10}$，($PC_{11\sim15}$)不变

说明：

① 指令只提供低 11 位地址，高 5 位为原 $PC_{11\sim15}$位的值。因此，转移范围为本指令后一条指令的首字节开始的同一 2KB 地址范围内。

② 机器码形式：本指令为 2 字节指令。设 addr11 的各位是 $a_{10}a_9a_8\cdots a_2a_1a_0$，则指令的机器码为 $a_{10}a_9a_800001a_7a_6a_5a_4a_3a_2a_1a_0$。

【例 3 -16】AJMP 3216H 指令在程序存储器中首地址是 3000H，该指令是否能实现程序转移？若可以，其指令机器码是什么？

解：因为该指令首地址为 3000H，其下一条指令首址为 3002H，3002H 与转移目的地址 3216H 在同一 2KB 地址范围内，所以该指令可实现程序转移。指令的机器码为：

0100000100010110 = 4116H

(2) 长转移指令

汇编指令格式	说明
LJMP addr16	；addr16→PC

说明：

① 本指令为 3 字节指令。机器码为：00000010 a15 ~ a8 a7 ~ a0。

② 本指令提供 16 位目标地址，将指令的第 2 和第 3 字节地址码分别装入 PC 的高 8 位和低 8 位中，程序无条件转向指定的目标地址去执行。因此，本指令为 64KB 程序存储器地址空间的全范围转移指令。

(3) 间接转移指令

汇编指令格式	说明
JMP@ A + DPTR	；(A) + (DPTR)→PC

说明：

本指令的转移地址由数据指针 DPTR 的 16 位数和累加器 A 的 8 位数进行无符号数相

加形成，并直接送入 PC。指令执行过程对 DPTR、A 和标志位均无影响。

【例 3－17】已知 A = 05H，DPTR = 2000H，指令 JMP@ A + DPTR 执行后，PC = 2005H，即程序跳转到 2005H 地址处执行。

【例 3－18】已知累加器 A 中存有编号值 0～4，程序存储器中存放起始地址为 TABLE 的 2 字节短转移指令表。编程完成按照累加器 A 中的编号转去执行对应命令的程序。

```
RL A                          ; A * 2→A
MOV DPTR, #TABLE              ; 指向表首址
JMP @ A + DPTR                ; 跳转到表入口
TABLE: AJMP PROC0             ; 双字节指令
AJMP PROC 1
AJMP PROC 2
…
```

可以看出，当(A) = 00H 时，程序转移到 PROC0；当(A) = 01H 时，程序转移到 PROC1……。可见，这是一段多路转移程序，进入的方向由累加器 A 的值决定。由于 AJMP 是双字节指令，跳转前 A 中键值应先乘以 2。

(4) 相对转移指令

汇编指令格式	说明
SJMP rel	;(PC) + 2→PC，(PC) + rel→PC

说明：

① 该指令的功能为先使程序计数器 PC 加 2(取出指令码)，然后再加上相对地址 rel 作为目标转移地址。即：目标转移地址 = 源地址 + 2 + rel。

② 指令的操作数是相对地址，rel 是 1 字节有符号数(用补码表示)，取值范围为 －128 ～ ＋127。负数表示反向转移，正数表示正向转移。

③ 该指令为双字节指令，机器码为 1000 0000 rel。

2. 条件转移指令

条件转移指令均为相对指令，是根据给出的条件进行检测，条件满足则程序转移到指定目标地址。

(1) 累加器为零(非零)转移指令

汇编指令格式	说明
JZ　rel	; A = 0 时，(PC) + rel→PC；A≠0 时，程序顺序执行。
JNZ　rel	; A≠0 时，(PC) + rel→PC；A = 0 时，程序顺序执行。

说明：

① PC 为下一条指令的首地址，PC = As + 2，As 为本指令的首地址。

② 目标地址是以下一条指令首地址为基础，加上指令中的相对偏移量 rel。

③ 相对偏移量 rel 为一个带符号的 8 位数，偏移范围为 －128 ～ ＋127。

④ 本指令不改变累加器 A 的内容，也不影响任何标志位。

（2）减一非零转移指令

汇编指令格式	说明
DJNZ　Rn，rel	；（Rn）－1→Rn，当（Rn）≠0 时，则（PC）＋rel→PC； 当（Rn）＝0 时，程序顺序执行。
DJNZ direct，rel	；（direct）－1→direct，当（direct）≠0 时，则（PC）＋rel→PC； 当（direct）＝0 时，程序顺序执行。

说明：

① 本指令有自动减 1 功能。

② DJNZ　Rn，rel 是 2 字节指令，而 DJNZ direct，rel 是 3 字节指令，所以在满足转移的条件下，前者 PC＝As＋2；后者是 PC＝As＋3，As 为本指令的首地址。

【例 3－19】有如下程序段：

```
    MOV   50H，#05H
    CLRA
LOOP：ADD   A，50H
    DJNZ   50H，LOOP
    SJMP   $
```

求程序运行后累加器 A 的值。

根据程序可知，A＝5＋4＋3＋2＋1＝15＝0FH

（3）比较条件转移指令

汇编指令格式	说明
CJNE A，direct，rel	；若（direct）≠（A），则（PC）＋rel→PC ；若（direct）＝（A），则顺序执行
CJNE A，#data，rel	；若#data≠（A），则（PC）＋rel→PC ；若#data＝（A），则顺序执行
CJNE Rn，#data，rel	；若#data≠（Rn），则（PC）＋rel→PC ；若#data＝（Rn），则顺序执行
CJNE @Ri，#data，rel	；若#data≠（（Ri）），则（PC）＋rel→PC ；若#data＝（（Ri）），则顺序执行

说明：

① 本指令为 3 字节指令，做两数相减操作，不送回结果，影响 CY 标志。

② 若第一操作数大于第二操作数，则进位标志位 CY 清 0；若第一操作数小于第二操作数，则置位进位标志位 CY；若二者相等，则往下执行。

③ 转移的目标地址为当前指令的首地址加 3 后，再加指令的第三操作数偏移量（rel）。

【例 3－20】已知 R5＝50H，指令 CJNE R5，#30H，60H 执行后，程序转移到 60H 地址单元执行，且 CY＝0。

3. 调用和返回指令

（1）短调用指令

汇编指令格式	说明
ACALL addr11	；addr11→$PC_{0\sim10}$

说明：

① 该指令功能为：先保护断点，即当前 PC（本指令的下一条指令的首地址）压入堆栈；然后将 addr11 送程序计数器的低 11 位 $PC_{0\sim10}$，PC 值的高 5 位 $PC_{11\sim15}$保持原值不变。

② 本指令提供 11 位目标地址，限定所调用的子程序入口地址必须和本调用指令在同一 2KB 范围内。

③ 本指令为双字节、双周期指令。指令机器码为 $a_{10}a_9a_8$10001$a_7a_6a_5a_4a_3a_2a_1a_0$，其中 10001 是 ACALL 指令的操作码。

【例 3－21】设 ACALL addr11 指令在程序存储器中起始地址为 102AH，试确定被调用子程序在程序存储器中的合法地址范围。

解：被调用子程序起始地址的高 5 位为断点地址的高 5 位，即 102AH＋2 的高 5 位，其余各位由 addr11 决定。Addr11 的变化范围为 00000000000B～11111111111B，因此被调用子程序的合法地址范围为 1000H～17FFH 的 2KB 区域。

（2）长调用指令

汇编指令格式	说明
LCALL addr16	；addr16→$PC_{15\sim0}$

说明：

① 该指令功能为：先保护断点，即当前 PC 压入堆栈；然后子程序的入口地址 addr16 送 PC，跳转到子程序执行。

② 本指令提供 16 位目标地址，可调用 64KB 范围内所指定的子程序。

③ 本指令为 3 字节指令。机器码为：12 addr16（00010010 addr16）。

（3）返回指令

① 汇编指令格式	说明
RET；	从调用子程序返回

指令功能为：把堆栈中的断点地址恢复到程序计数器 PC 中，从子程序返回主程序。

② 汇编指令格式	说明
RETI	；从中断服务程序返回

指令功能为：把堆栈中的断点地址恢复到程序计数器 PC 中，并恢复中断优先级状态触发器，从中断服务程序返回主程序。

4. 空操作指令

汇编指令格式	说明
NOP	；机器码 00

该指令经取指、译码后不做任何操作，而转到下一条指令。该指令除了完成 PC 内容加 1 外，不影响其他寄存器和标志位，常用来产生一个机器周期的延时。

3.1.7　位操作指令

51 单片机具有丰富的位处理功能，在其硬件结构中有个位处理机(布尔处理机)，它具有一套处理位变量的指令集，包括位传送、位运算、位置位、位清零、位控制程序转移等指令。位操作指令的操作数不是字节，而是字节中的某一位。位操作指令的操作对象是片内 RAM 的位寻址区(即 20H ~ 2FH)和 SFR 中的 11 个可以位寻址的寄存器。

(1) 位清零、置位指令

汇编指令格式	说明
CLR C	; 0→C
CLR bit	; 0→bit
SETB C	; 1→C
SETB bit	; 1→bit

(2) 位取反指令

汇编指令格式	说明
CPL C	; $(\overline{C})$→C
CPL bit	; $(\overline{bit})$→bit

(3) 位“与”、“或”指令

汇编指令格式	说明
ANL C, bit	; (C) ∧ (bit)→C
ANL C, /bit	; (C) ∧ $(\overline{bit})$→C
ORL C, bit	; (C) ∨ (bit)→C
ORL C, /bit	; (C) ∨ $(\overline{bit})$→C

(4) 位传送

汇编指令格式	说明
MOV C, bit	; (bit)→CY
MOV bit, C	; CY→bit

(5) 位转移

位转移为相对转移指令，根据位的值来决定转移方向，设 As 为下面各指令的首地址。

汇编指令格式	说明
JC　rel	; 若 CY = 1，转移(PC + rel→PC)，否则程序顺序执行
JNC　rel	; 若 CY = 0，转移(PC + rel→PC)，否则程序顺序执行
JB　bit, rel	; 若(bit) = 1，转移(PC + rel→PC)，否则程序顺序执行
JNB　bit, rel	; 若(bit) = 0，转移(PC + rel→PC)，否则程序顺序执行
JBC　bit, rel	; 若(bit) = 1，转移(PC + rel→PC)，且该位清零；否则程序顺序执行

【例 3 - 22】试通过编程把 20H 位中的内容和 50H 位中的内容相交换。

解：为了实现两个位地址内容相交换，需要设定一个暂存单元 30H。

```
MOV C，20H          ；(20H)→CY
MOV 30H，C          ；暂存于30H位
MOV C，50H          ；(50H)→CY
MOV 20H，C          ；存入20H位
MOV C，30H          ；(30H)→CY
MOV 50H，C          ；存入50H位
SJMP $
```

【例3-23】试编程完成Z=X⊕Y，设X、Y、Z代表三个位地址。

解：因位操作中无异或操作指令，位异或操作必须用位操作指令来实现。依据算式$Z=X\oplus Y=\overline{X}Y+X\overline{Y}$，程序如下：

```
MOV C，X
ANL C，/Y           ；X∧Ȳ→CY
MOV 30H，C          ；暂存于30H
MOV C，Y
ANL C，/X           ；X̄∧Y→CY
ORL C，30H          ；X̄Y+XȲ→CY
MOV X，C
SJMP $
```

3.2 汇编语言程序设计

在51单片机的应用中，汇编语言是程序设计的一个关键问题。汇编语言是面向机器的语言，用助记符代替机器指令的操作码，用地址符号或标号代替指令或操作数的地址。汇编语言产生的目标程序简短，占用存储空间小，执行速度快，常与高级语言配合使用，改善程序的执行速度和效率，弥补高级语言在硬件控制方面的不足，主要用于单片机的实时控制部分。

3.2.1 汇编语言格式

汇编语言源程序由一条条汇编语言语句构成。汇编语言中的每条语句都应当符合典型的四分段格式：

[标号段][操作码段][操作数段]；[注释段]

格式中的标号段和操作码段之间要用冒号“：”相隔；操作码段和操作数段间用分界符空格相隔；操作数与操作数之间用逗号“，”相隔；操作数段和注释段之间用分号“；”相隔。操作码段为必选项，其余各段为任选项。例如：

```
LOOP：ADD A，50H      ；(50H)+(A)→A
```

各字段具体说明如下。

1. 标号段

标号位于语句的开头，是用户定义的符号地址，以指明标号所在指令操作码在内存的地址。标号由以英文字母开始的 1 ~ 8 个字母或数字串组成，以冒号“:”结尾。标号不能与指令保留符、寄存器号以及伪指令等相同。

2. 操作码段

操作码段可以是指令的保留字(如 MOV、ADD 等)，也可以是伪指令和宏指令的助记符(如 ORG 和 END)。

3. 操作数段

操作数段用于存放指令的操作数或操作数地址。操作数个数因指令不同而不同，通常有单操作数、双操作数和无操作数 3 种情况。操作数通常有 5 种合法表示形式：

(1) 常数

操作数允许采用常数，常数可以写成二进制、十进制或十六进制等形式。常数总是要以一个数字开头(若十六进制的第一个数为“A ~ F”字符，则前面要加 0)，而数字后要直接跟一个表明数制的字母(B 表示二进制，H 表示十六进制)。

(2) 工作寄存器和特殊功能寄存器

当操作数在某个工作寄存器或特殊功能寄存器中时，允许在操作数段采用工作寄存器或特殊功能寄存器的代号表示。例如：累加器 A 和工作寄存器 R0 ~ R7。

(3) 标号名

可以在操作数段中引用的标号名包括：

赋值标号——由汇编命令 EQU 等赋值的标号可以作为操作数。

指令标号——指令标号所在指令的第一字节地址就是这个标号的值，在以后的指令操作数字段中可以引用。

(4) $

“ $ ”用来表示该转移指令操作码所在的内存地址。例如“JNB TF0， $ ”表示若 TF0 为 0，则仍执行该指令；否则顺序执行。

(5) 表达式

汇编程序允许把表达式作为操作数使用。在汇编时，计算出表达式的值，并把该值填入目标码中。例如：MOV A，SUM + 1。

4. 注释

注释字段是用于说明解释指令或程序含义的，没有汇编语言的功能部分，只用于改善程序的可读性。

3.2.2　伪指令

伪指令是指示性语句，不是真正的指令，在汇编时不会产生可执行的机器码，只是用来对汇编过程进行某种控制，如：规定汇编生成的目标代码在内存中的存放区域、指示汇编的结束。

常用伪指令有以下几条。

1. 汇编起始

格式：ORG(16 位地址或标号)

功能：常用于汇编语言源程序或数据块开头，指示汇编程序开始对源程序汇编，汇编后生成目标程序存放在指令中指定的存储单元。例如：

```
ORG 1500H          ；指示后面的程序或数据块以 1500H 为起始地址连续存放。
```

ORG 可以多次出现在程序的任何地方。当它出现时，下一条指令的地址就由此重新定位。

2. 汇编结束

格式：[标号:]END

功能：指示源程序结束。在 END 之后所有的汇编语言指令均不予以汇编。

3. 赋值

格式：字符名称 EQU(数据或汇编符号)

功能：EQU 把右边的“数据或汇编符号”赋值给左边的“字符名称”。例如：

```
ABC EQU 50H
MOV A，ABC；汇编时将 ABC 替换为 50H
```

4. 数据地址赋值

格式：字符名称 DATA(表达式)

功能：与 EQU 类似，把右边“表达式”的值赋给左边的“字符名称”。常在程序中用来定义数据地址。

两者的区别在于：

① EQU 定义的字符名必须先定义后使用，而 DATA 定义的字符名可以后定义先使用。

② 用 EQU 伪指令可以把一个汇编符号赋给一个名字，而 DATA 只能把数据赋给字符名。

③ DATA 语句中可以把一个表达式的值赋给字符名称，其中的表达式应是可求值的。

5. 字节定义

格式：[标号:]DB(字节常数、字符或表达式)

功能：指示汇编程序在程序存储器中以标号为起始地址的单元里存放的数为字节数据。例如：

```
ORG 1500H
LST：DB 20H，30H
STR：DB ‘AB’
```

经汇编后，则有：

(1500H)=20H

(1501H)=30H

(1502H)=41H

(1503H)=42H

其中，41H、42H 为 A、B 的 ASCII 编码值。

6. 字定义

格式：DW　（字常数或表达式）

功能：指示汇编程序在程序存储器中以标号为起始地址的单元里存放的数为字数据，即 16 位二进制数，数据的高 8 位先存放，低 8 位后存放。DW 常用于定义一个地址表，例如：

```
ORG 1500H
TABLE：DW 1234H，65H
```

经汇编后，则有：

（1500H）=12H　（1501H）=34H

（1502H）=00H　（1503H）=65H

7. 定义存储空间

格式：DS(数值表达式)

功能：指示汇编程序在程序存储器中保留以标号为起始地址的若干字节单元，单元个数由指令中的数值表达式决定。例如：

```
ORG 1500H
DS 05H
DB 20H
```

汇编以后，从 1500H 地址开始保留 5 个字节单元，然后使用 DB 命令在 1505H 处给内存赋值，即

（1505H）=20H

8. 位定义

格式：字符名 BIT(位地址)

功能：把 BIT 之后的位地址值赋给字符名。例如：A1 BIT 10H。

3.2.3　顺序程序设计

程序中没有分支、循环，也不调用子程序，程序按顺序一条一条地执行指令，称为顺序结构。

【例 3-24】在内部 RAM 的 50H(低位字节)、51H(高位字节)和 60H(低位字节)、61H(高位字节)地址单元中存有两个 16 位二进制数，试编程求这两个数的和，结果存放于 50H、51H。

```
ORG 0000H
MOV R0，#50H          ；将被加数低 8 位地址值送 R0
MOV A，@R0            ；被加数低字节内容送 A
ADD A，60H            ；低字节数相加
MOV@R0，A             ；低字节数和存于 50H 中
INC R0                ；指向被加数高位字节
MOV A，@R0            ；被加数高位字节送 A
```

```
ADDC A，61H          ；高字节数相加
MOV @R0，A           ；高字节数和存 51H 中
SJMP $
END
```

【例 3－25】在内部 RAM 的 35H(低位字节)、36H(高位字节)中存有一个 16 位二进制负数，试编程求其补码，结果存入原地址单元。

分析：负数补码 = 反码 + 1，求补过程就是取反加 1。所以，低 8 位取反后加 1；然后高 8 位取反，再加上来自低位的进位。由于 51 单片机的加 1 指令不影响标志位，所以低位加 1 要使用 ADD 指令，不能使用 INC 指令。程序如下：

```
ORG 0000H
MOV A，35H           ；16 位数低 8 位送 A
CPL A                ；求反
ADD A，#01H          ；加 1
MOV 35H，A           ；存补码低 8 位
MOV A，36H           ；取 16 位数高 8 位
CPL A                ；求反
ADDC A，#00H         ；加进位
ORL A，#80H          ；恢复负号
MOV 36H，A           ；存补码高 8 位
SJMP $
END
```

3.2.4 分支程序设计

分支程序是通过转移指令控制程序转移方向的。转移指令包括无条件转移和条件转移，因此分支程序也可分为无条件分支程序和条件分支程序两类。无条件分支程序比较简单，这里不作讨论；条件分支程序根据条件对程序的执行进行判断：若满足条件，则进行程序转移；若不满足条件，则顺序执行程序。

【例 3－26】设变量 X 存放在片内 RAM 的 XVAR 单元中，函数值 Y 存放在 YFUN 中，函数 Y 与变量 X 有如下关系式：

$$Y=\begin{cases}1 & X>1\\0 & X=0\\-1 & X<0\end{cases}$$

试编程，根据 X 的取值求出 Y，结果存放在原地址单元。

分析：用逻辑运算类指令对变量 X 的取值范围进行判断，程序流程图如图 3－1 所示。

程序如下：

```
ORG 0000H
XVAR DATA 30H
```

```
YFUN DATA 40H
START：MOV A，XVAR
JZ ZERO                ；为 0 转 ZERO
JNB ACC.7，BIG         ；x＞0 转 BIG
MOV A，#0FFH           ；x＜0，－1→A
SJMP ZERO
BIG：MOV A，#01H
ZERO：MOV YFUN，A
```

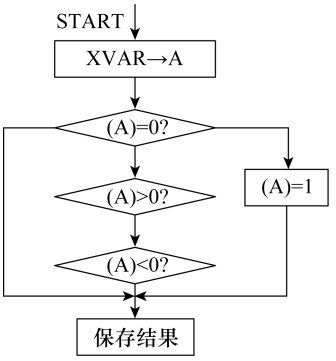


图 3－1　程序流程图

【例 3－27】在内部 RAM 的 30H 和 31H 地址单元中存放有两个无符号二进制数，试编程比较它们的大小，并将大的数存放在内部 RAM 的 BG 单元中。

分析：采用 CJNE 指令判断两数是否相等，并通过得到的 CY 标志判断两数大小，程序如下：

```
ORG 0000H
START：MOV A，30H      ；将 30H 中内容送 A
CJNE A，31H，LOOP1     ；两数比较，不相等则转 LOOP1
SJMP LOOP3
LOOP1：JC LOOP2        ；当 CY＝1，转 LOOP2
MOV BG，A              ；CY＝0，(A)＞(31H)
SJMP LOOP3
LOOP2：MOV BG，31H     ；CY＝1，(31H)＞(A)
LOOP3：END             ；程序结束
```

【例 3－28】多分支转移程序，根据 R5 的内容转向对应的目的地址，R5 的内容为 0～n，目的程序入口符号地址分别为：PRG0～PRGn。

```
    PRG0 EQU 1000H
    PRG1 EQU 1100H
    PRG2 EQU 1200H
…
ORG 0000H
JMPT：MOV A，R5
RL A                   ；AJMP 指令为 2 字节指令，A＊2→A
MOV DPTR，#JTAB
JMP@ A＋DPTR
SJMP $                 ；查表不相符，程序结束
JTAB：AJMP PRG0
AJMP PRG1
…
```

```
AJMP PRGn
END
```

说明：此程序要求 128 个转移目的地址(PRG 0 ~ PRGn)必须驻留在与绝对转移指令 AJMP 相同的一个 2KB 存储区内。RL 指令对变址部分乘以 2，因为每条 AJMP 指令占两字节。

3.2.5 循环程序设计

在程序运行时，有时需要连续重复执行某些指令，这时可以使用循环程序的方式。循环程序设计可以缩短程序长度，节省存储单元。循环程序设计主要解决的问题是循环次数的控制。有两种方法：一种是先循环处理后循环控制(即先处理后判断)；另一种是先循环控制后循环处理(即先判断后处理)。循环可以单重循环，也可以多重循环，在多重循环中，内外循环不能交叉。

【例 3 -29】在内部 RAM 中地址 20H ~ 29H 有 10 个无符号数，试编写程序求它们的和，结果存入 SUM 单元。

分析：用 R1 作为数据地址指针，R2 存放累加次数。每累加一次，数据地址指针 R1 加 1，R2 减 1，直至 R2 为 0。程序如下：

```
        ORG 0000H
        CLR A                   ; 累加器 A 清零
        MOV R2，#0AH            ; 无符号数个数 10→R2
        MOV R1，#20H            ; 数据起始地址 20H→R1
LOOP：  ADD A，@R1              ; (A) + ((R1))→A
        INC R1                  ; R1 + 1→R1，修改数据指针
        DJNZ R2，LOOP           ; 计算未完，则转 LOOP
        MOV SUM，A              ; 结果存入 SUM 单元
        SJMP $                  ; 程序结束
        END
```

【例 3 -30】已知在片内 RAM 的 50H 和 51H 单元存放有两个 BCD 码数，试编写求这两个数乘法的程序，结果存入 52H 和 53H。

分析：乘法是加法的简便计算，两数 n，m 相乘即 n 个相同的数 m 相加。用 R1 作为累加计数寄存器。

```
        MOV 53H，#00H
        MOV 52H，#00H
        CLR A
        MOV R1，#00H
LOOP：  MOV A，50H              ; 取出乘数
        ADD A，53H              ; 乘数累加
        DA A                    ; 二进制至十进制调整
```

```
MOV 53H，A              ；乘积低 8 位累加结果存入目的地址
CLR A
ADDC A，52H             ；乘积高 8 位累加并加上低 8 位进位
DA A
MOV 52H，A              ；乘积高 8 位累加结果存入目的地址
INC R1                  ；累加次数增 1
CJNE R1，51H，LOOP      ；判断累加计算是否结束
SJMP $                  ；若已经累加了(51H)次，则结束
```

【例 3－31】已知单片机的晶振为 6MHz，设计一个延时 10ms 的延时子程序。

分析：延时时间与单片机的晶振频率和程序的循环次数有关。已知晶振采用 6MHz，一个机器周期是 12/(6M) =2μs(51 单片机的机器周期由 6 个状态周期组成，即：1 个机器周期 =6 个状态周期 =12 个时钟周期)，用单循环可以实现 1ms 延时，外循环 10 次即可达 10ms 延时。

```
机器周期数
    1              MOV   R0,#0AH     ；外循环 10 次
    1        LP2:  MOV   R1,#NUM     ；内循环NUM 次
    1        LP1:  NOP
    1              NOP               ；空操作指令
    2              DJNZ  R1,LP1
    2              DJNZ  R0,LP2
                   RET
```

内循环 LP1 到指令 DJNZ　R1，LP1 的计算：

$$(1+1+2)\times 2\mu s\times NUM = 1000\mu S$$

计算得：NUM = 125 = 7DH

将 7DH 代入上面程序的 NUM，计算总的延时时间：

$$\{1+[1+(1+1+2)\times 125+2]\times 10\}\times 2\mu s = 10062\mu s = 10.062ms$$

调整 R0 和 R1 中的参数，可改变延时时间。如果需要加长延时时间，则可增加循环嵌套，如 1s 延时可用 3 重循环。这种延时方法需占用 CPU 时间，如果延时时间较长，一般采用定时器方法。

3.2.6　子程序

能被其他程序调用，在完成某种功能后能自动返回到调用程序的程序段，叫做子程序。子程序是构成单片机应用程序必不可少的部分。51 单片机有 ACALL 和 LCALL 两条子程序调用指令，可以十分方便地用来调用任何地址处的子程序。

在调用子程序时，应注意以下几点：

1. 保护现场

如果在调用子程序前主程序已经使用了某些存储单元或寄存器，在调用时，这些寄存

器和存储单元又有其他用途，就应先把这些单元或寄存器中的内容压入堆栈保护，调用完后再从堆栈中弹出以便恢复。如果有较多的寄存器要保护，应使主程序和子程序使用不同的寄存器组。

2. 设置入口参数和出口参数

调用子程序之前，主程序要按子程序的要求设置好地址单元或存储器(称为入口参数)，以便子程序从指定的地址单元或存储器获得输入数据；子程序将运算或处理后的结果存放到指定的地址单元或寄存器(称为出口参数)，主程序调用后从指定的地址单元或寄存器读取运算或处理后的结果，只有这样，才能完成子程序和主程序间的数据的正确传递。

3. 子程序中可包括对另外子程序的调用，称为子程序嵌套

【例 3－32】用程序实现 y = x! + z!，设 x，z 均在这个区间(0 ~ 5)。x 存放在 30H 单元，z 存放在 31H 单元，把 y 存入 33H 和 32H 单元。(和要求为 BCD 码)。

解：因该算式两次用到阶乘值，所以在程序中把求阶乘编为子程序。求阶乘采用查表法，主程序和子程序编写如下：

主程序：

```
ORG   0000H
MOV   SP，#50H
MOV A，30H                ；取 a
LCALL FACT                ；求 a!
MOV R1，A
MOV A，31H                ；取 b
LCALL FACT                ；求 b!
ADD A，R1                 ；求和
DA A                      ；调整
MOV 32H，A
MOV A，#0
ADDC A，#0
MOV 33H，A
SJMP $
```

子程序：

```
    ORG 0030H
FACT：MOV DPTR，#TAB
    MOVC A，@ A + DPTR    ；查阶乘表
    RET
TAB：DB 01H，01H，02H，06H，24H，64H
    END
```

本章小结

1. 51 单片机指令系统共有 7 种寻址方式，不同的存储空间寻址方式不同，使用的指令不同，必须正确区分。

2. 51 单片机指令系统包括数据传送与交换指令、算术运算指令、逻辑操作指令、控制转移指令和位操作指令。指令是程序设计的基础，应重点掌握各指令的功能、操作对象和结果，以及对标志位的影响。

3. 汇编程序设计包括四种结构：顺序程序结构、分支程序结构、循环程序结构、子程序结构。应掌握它们的设计方法，并能熟练使用查表技术简化程序的设计。

4. 伪指令是非执行指令，提供汇编程序以汇编信息，应正确使用。

本章参考文献

[1] 胡汉才编著．单片机原理及接口技术(第 2 版)．北京：清华大学出版社，2004.

[2] 李朝青编著．单片机原理及接口技术(第 3 版)．北京：北京航空航天大学出版社，2005.

[4] 田希晖等．C51 单片机技术教程．北京：人民邮电出版社，2007.

[5] 李群芳等．单片微机计算机与接口技术(第 4 版)．北京：电子工业出版，2012.

本章习题

3.1　简述 51 单片机汇编指令格式。

3.2　简述 51 单片机有哪几种寻址方式？适用于哪些寻址空间？

3.3　访问特殊功能寄存器和外部数据存储器应采用哪种寻址方式？

3.4　片内 RAM 20H～2FH 单元中的 128 个位地址与直接地址 00H～7FH 形式完全相同，如何在指令中区分出位寻址操作和直接寻址操作？

3.5　SJMP，AJMP 和 LJMP 指令在功能上有何不同？

3.6　设 A＝0，执行 MOVX A，@DPTR 与 MOVC A，A＋@DPTR 两条指令后，A 的内容是否相同，为什么？

3.7　设片内 RAM 中的(30H)＝55H，执行下列程序段

```
MOV   A，30H
MOV   R0，A
MOV   A，#10H
MOV   @R0，A
MOV   A，#40H
```

MOV 56H，A
MOV 57H，#70H
后，（A）=_____，（R0）=_____，（55H）=_____，（56H）=_____ 。

3.8 执行下列程序段
MOV A，#39H
ADD A，#85H
ADD A，57H
后，CY =_____，OV =_____，A =_____。

3.9 设 SP =50H，内部 RAM 的(20H) =35H，(21H) =10H，请完成注释中括号内容的填写。

```
PUSH 20H          ; SP = (  ), (SP) = (  )
PUSH 21H          ; SP = (  ), (SP) = (  )
POP DPL           ; SP = (  ), DPL = (  )
POP DPH           ; SP = (  ), DPH = (  )
MOV A，#00H
MOVX @ DPTR，A
```

最后执行结果为( )。

3.10 完成以下的数据传送过程。
（1）把 R0 的内容传送到 R1。
（2）片内 RAM 30H 单元的内容送 R0。
（3）片外 RAM 30H 单元的内容送 R0。
（4）片外 RAM 3000H 单元的内容送片内 RAM 30H
（5）片外 RAM 30H 单元的内容送片外 RAM 3000H。

3.11 试编写程序，将内部 RAM 的 30H、31H 和 32H 连续单元的内容依次存入 3FH、3EH 和 3DH 中。

3.12 设有两个 4 位 BCD 码，分别存放在片内 RAM 的 21H，20H 单元和 31H，30H 单元中，求它们的和，并送入 41H，40H 单元中去。

3.13 已知从内部 RAM 的 20H 单元开始存放有一组带符号数，数的个数存放在 LEN 单元。请编程统计其中正数和负数个数，结果存入内部 RAM 的 40H 和 41H 单元。

3.14 已知内部 RAM 的 30H 有一个十六进制数，请编写程序将它转换成相应 ASCII 码，结果存入 51H 和 50H，高位存入 51H。

3.15 编程计算片内 RAM 区 30H ~ 40H 的 11 个单元中数的平均值，结果存在 50H 单元中。

3.16 试编写一个 3 字节无符号数乘 1 字节的乘法程序。