51 单片机汇编语言教程：第 27 课-关于单片机的一些基本概念

（基于 HJ-1G、HJ-3G 实验板）

随着电子技术的迅速发展，计算机已深入地渗透到我们的生活中，许多电子爱好者开始学习单片机知识，但单片机的内容比较抽象，相对电子爱好者已熟悉的模拟电路、数字电路，单片机中有一些新的概念，这些概念非常基本以至于一般作者不屑去谈，教材自然也不会很深入地讲解这些概念，但这些内容又是学习中必须要理解的，下面就结合本人的学习、教学经验，对这些最基本概念作一说明，希望对自学者有所帮助。

一、总线：我们知道，一个电路总是由元器件通过电线连接而成的，在模拟电路中，连线并不成为一个问题，因为各器件间一般是串行关系，各器件之间的连线并不很多，但计算机电路却不一样，它是以微处理器为核心，各器件都要与微处理器相连，各器件之间的工作必须相互协调，所以就需要的连线就很多了，如果仍如同模拟电路一样，在各微处理器和各器件间单独连线，则线的数量将多得惊人，所以在微处理机中引入了总线的概念，各个器件共同享用连线，所有器件的8根数据线全部接到8根公用的线上，即相当于各个器件并联起来，但仅这样还不行，如果有两个器件同时送出数据，一个为0，一个为1，那么，接收方接收到的究竟是什么呢？这种情况是不允许的，所以要通过控制线进行控制，使器件分时工作，任何时候只能有一个器件发送数据（能有多个器件同时接收）。器件的数据线也就被称为数据总线，器件所有的控制线被称为控制总线。在单片机内部或者外部存储器及其它器件中有存储单元，这些存储单元要被分配地址，才能使用，分配地址当然也是以电信号的形式给出的，由于存储单元比较多，所以，用于地址分配的线也较多，这些线被称为地址总线。

二、数据、地址、指令：之所以将这三者放在一起，是因为这三者的本质都是一样的——数字，或者说都是一串‘0’和‘1’组成的序列。换言之，地址、指令也都是数据。指令：由单片机芯片的设计者规定的一种数字，它与我们常用的指令助记符有着严格的一一对应关系，不能由单片机的开发者更改。地址：是寻找单片机内部、外部的存储单元、输入输出口的依据，内部单元的地址值已由芯片设计者规定好，不可更改，外部的单元能由单片机开发者自行决定，但有一些地址单元是一定要有的（详见程序的执行过程）。数据：这是由微处理机处理的对象，在各种不一样的应用电路中各不相同，一般而言，被处理的数据可能有这么几种情况：

1·地址（如 MOV DPTR，#1000H），即地址1000H 送入 DPTR。

2·方式字或控制字（如 MOV TMOD，#3），3即是控制字。

3·常数（如 MOV TH0，#10H）10H 即定时常数。

4·实际输出值（如 P1口接彩灯，要灯全亮，则执行指令：MOV P1，#0FFH，要灯全暗，则执行指令：MOV P1，#00H）这里0FFH 和00H 都是实际输出值。又如用于 LED 的字形码，也是

实际输出的值。

理解了地址、指令的本质，就不难理解程序运行过程中为什么会跑飞，会把数据当成指令来执行了。

三、P0口、P2口和 P3的第二功能使用办法 开始学习时一般对 P0口、P2口和 P3口的第二功能使用办法迷惑不解，认为第二功能和原功能之间要有一个切换的过程，或者说要有一条指令，事实上，各端口的第二功能完全是自动的，不需要用指令来转换。如 P3.6、P3.7 分别是 WR、RD 信号，当微片理机外接 RAM 或有外部 I/O 口时，它们被用作第二功能，不能作为通用 I/O 口使用，只要一微处理机一执行到 MOVX 指令，就会有对应的信号从 P3.6或 P3.7 送出，不需要事先用指令说明。事实上‘不能作为通用 I/O 口使用’也并不是‘不能’而是（使用者）‘不会’将其作为通用 I/O 口使用。你完全能在指令中按排一条 SETB P3.7的指令，并且当单片机执行到这条指令时，也会使 P3.7变为高电平，但使用者不会这么去做，因为这常常这会导致系统的崩溃（即死机）。

四、程序的执行过程 单片机在通电复位后8051内的程序计数器(PC)中的值为‘0000’，所以程序总是从‘0000’单元开始执行，也就是说：在系统的 ROM 中一定要存在‘0000’这个单元，并且在‘0000’单元中存放的一定是一条指令。

五、堆栈 堆栈是一个区域，是用来存放数据的，这个区域本身没有任何特殊之处，就是内部 RAM 的一部份，特殊的是它存放和取用数据的方式，即所谓的‘先进后出，后进先出’，并且堆栈有特殊的数据传输指令，即‘PUSH’和‘POP’，有一个特殊的专为其服务的单元，即堆栈指针 SP，每当执一次 PUSH 指令时，SP 就（在原来值的基础上）自动加1，每当执行一次 POP 指令，SP 就（在原来值的基础上）自动减1。由于 SP 中的值能用指令加以改变，所以只要在程序开始阶段更改了 SP 的值，就能把堆栈设置在规定的内存单元中，如在程序开始时，用一条 MOV SP，#5FH 指令，就时把堆栈设置在从内存单元60H 开始的单元中。一般程序的开头总有这么一条设置堆栈指针的指令，因为开机时，SP 的初始值为07H，这样就使堆栈从08H 单元开始往后，而08H 到1FH 这个区域正是8031的第二、三、四工作寄存器区，经常要被使用，这会造成数据的浑乱。不一样作者编写程序时，初始化堆栈指令也不完全相同，这是作者的习惯问题。当设置好堆栈区后，并不意味着该区域成为一种专用内存，它还是能象普通内存区域一样使用，只是一般情况下编程者不会把它当成普通内存用了。

六、单片机的开发过程 这里所说的开发过程并不是一般书中所说的从任务分析开始，我们假设已设计并制作好硬件，下面就是编写软件的工作。在编写软件之前，首先要确定一些常数、地址，事实上这些常数、地址在设计阶段已被直接或间接地确定下来了。如当某器

件的连线设计好后，其地址也就被确定了，当器件的功能被确定下来后，其控制字也就被确定了。然后用文本编缉器（如 EDIT、CCED 等）编写软件，编写好后，用编译器对源程序文件编译，查错，直到没有语法错误，除了极简单的程序外，一般应用仿真机对软件进行调试，直到程序运行正确为止。运行正确后，就能写片（将程序固化在 EPROM 中）。在源程序被编译后，生成了扩展名为 HEX 的目标文件，一般编程器能够识别这种格式的文件，只要将此文件调入即可写片。在此，为使大家对整个过程有个认识，举一例说明：

表1

```
ORG 0000H
LJMP START
ORG 040H
START：
MOV SP，#5FH ;设堆栈
LOOP：
NOP
LJMP LOOP ； 循环
END ； 结束
```

表2

```
:03000000020040BB
:0700400075815F000200431F
```

表3

```
02 00 40 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF 75 81 5F 00 02 00 43
```

表1为源程序，表2是汇编后得到的 HEX 文件，表3是由 HEX 文件转换成的目标文件，也就是最终写入 EPROM 的文件，它由编程器转换得到，也能由 HEXBIN 一类的程序转换得到。学过手工汇编者应当不难找出表3与表1的一一对应关系，值得注意的是从02 00 40后开始的一长串‘FF’，直到75 81，这是由于伪指令：ORG 040H 造成的结果。

七、仿真、仿真机 仿真是单片机开发过程中非常重要的一个环节，除了一些极简单的任务，一般产品开发过程中都要进行仿真，仿真的主要目的是进行软件调试，当然借助仿真机，也能进行一些硬件排错。一块单片机应用电路板包括单片机部份及为达到使用目的而设

计的应用电路，仿真就是利用仿真机来代替应用电路板（称目标机）的单片机部份，对应用电路部份进行测试、调试。仿真有 CPU 仿真和 ROM 仿真两种，所谓 CPU 仿真是指用仿真机代替目标机的 CPU，由仿真机向目标机的应用电路部份供给各种信号、数据，进行调试的办法。这种仿真能通过单步运行、连续运行等多种办法来运行程序，并能观察到单片机内部的变化，便于改正程序中的错误。所谓 ROM 仿真，就是用仿真机代替目标机的 ROM，目标机的 CPU 工作时，从仿真机中读取程序，并执行。这种仿真其实就是将仿真机当成一片 EPROM，只是省去了擦片、写片的麻烦，并没有多少调试手段可言。常常这是二种不一样类型的仿真机，也就是说，一台仿真机不能既做 CPU 仿真，又做 ROM 仿真。可能的情况下，当然以 CPU 仿真好。

51 实验板推荐（点击下面的图片可以进入下载资料链接）

HJ-1G	HJ-3G