寻址方式表2-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 寻址方式 | 定义 | 例子 | 使用范围 | 备注 |
| 寄存器寻址 | 直接用寄存器名称作为操作数进行寻址 | MOV A,R0  MOV P1，A  就是A到P1口  ADD A,R0 | 当前工作寄存器组R0~R7  A  B  C  DPTR |  |
| 直接寻址 | 操作数为片内RAM单元的地址 | MOV A,3AH  MOV A,P1  MOV A,90H | 包括32个工作寄存器组在内片内RAM低128字节  SFR | SFR（只能用直接寻址） |
| 立即数寻址 | 直接一字节或两字节的数，用#来标明  如：MOV DPTR,#DATA16 | MOV DPTR,#2000H  MOV R0，#31H | ROM |  |
| 寄存器间接寻址 | 用给出寄存器内的内容作为地址所指向的寄存器中内容为操作数，用@表明 | MOV A,@R0  MOV A,@DPTR | R0或R1可以间接寻址片内RAM低128字节地址和片外RAM的低128字节地址。  SP（仅是PUSH和POP）  间接寻址64KB片外RAM。 | 不可以SFR  片外RAM可以利用R0，R1，DPTR |
| 变址寻址 | 以PC或DPTR中的内容作为基地址，以A中内容作为地址偏移量  两者相加得到操作数地址 | MOVC A,@A+DPTR | 只能访问ROM  范围为64KB | 只能读取  不能写入  多用于查表 |
| 相对寻址 | 以PC的值为源地址  与给出的偏移量rel相加得到目的地址 | JC rel  如果CY=1，  （PC）+2+rel  SJMP 55H | ROM  256字节范围内 | rel范围是-128~127  正向：rel=目的地址-源地址-2  反向：rel=（目的地址-（源地址+2））的补码=FEH-|地址差| |

寻址方式表2-2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 寻址方式 | 定义 | 例子 | 使用范围 | 备注 |
| 位寻址 | 对某一位寻址bit | CLR bit | 片内RAM中20H~2FH共128位  字节地址可以被8整除的SFR | 可以直接用位地址  也可以用寄存器名字加位数表  如PSW.3 |

89C51 指令系统可分为5 大类:

数据传送指令(28 条) ;

算术运算指令(24 条) ;

逻辑运算及移位指令(25 条) ;

控制转移指令(17 条) ;

位操作指令或布尔操作(17 条)。

传送指令2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 例子 | 备注 |
| 数据传送 | 移动MOV | 目的地址为累加器A  4个 | MOV A,Rn  MOV A,direct  MOV A,@Ri  MOV A,#data |  | 只影响PSW的P标志位 |
| 目的地址为寄存器Rn  3个 | MOV Rn,A  MOV Rn,direct  MOV Rn,#data | MOV R5,A  MOV R5,70H  MOV R5,#A3H | 没有MOV Rn,Rn |
| 目的地址为直接地址  5个 | MOV direct,A  MOV direct,Rn  MOV direct,direct  MOV direct,@Ri  MOV direct,#data |  | MOV direct,direct  前为源地址  后为目的地址 |
| 目的地址为间接地址  3个 | MOV @Ri,A  MOV @Ri,direct  MOV @Ri,#data |  | 图1-1 |
| 16位立即数  1个 | MOV DPTR,#data16 | MOV DPTR,#1234H | dataH到DPH  dataL到DPL  唯一的16位立即数传送 |
| MOVC | 查找表  2个 | MOVC A,@A+DPTR  MOVC A,@A+PC |  | (PC)+1到PC后，（（A）+（PC））到A  图1-2 |

传送指令2-2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 例子 | 备注 |
| 数据传送 | MOVX | 累加器A与片外RAM之间传送  4个 | MOVX A,@Ri  MOVX A,@DPTR  MOVX @Ri,A  MOVX @DPTR,A |  | 前两个使/RD=0  后两个使/WR=0  @Ri寻址片外RAM 0~255字节  @DPTR寻址片外RAM 0~64KB |
| 栈指令PUSH | 入栈  1个 | PUSH direct |  | （SP）+1→(SP)  (direct) →（SP） |
| 栈指令  POP | 出栈  1个 | POP direct |  | ((SP)) →direct  (SP)-1→(SP)  先进后出 |
| 交换指令XCH | 字节交换  3个 | XCH A,Rn  XCH A,direct  XCH A,@Ri |  |  |
| 交换指令XCHD | 半字节交换  1个 | XCHD A,@Ri |  | 低位互换，高位不变 |



图1-1间接寻址



图1-2查找表

算术运算指令2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 例子 | 备注 |
| 算术运算指令 | ADD | 8位无符号二进制数和累加器A中的数相加  不加进位  4条 | ADD A,Rn  ADD A,direct  ADD A,@Ri  ADD A,#data |  | 结果存于A中  第3位进位AC置1否则为0  第7位进位CY置1，否则为0  同时影响溢出OV和奇偶P标志位  OV=7位和6位异或 |
| ADDC | 同上  不同的是结果加进位CY  4条 | ADDC A,Rn  ADDC A,direct  ADDC A,@Ri  ADDC A,#data |  | 影响标志位AC,CY,OV,P |
| SUBB | 带借位CY的减法  4条 | SUBB A,Rn  SUBB A,direct  SUBB A,@Ri  SUBB A,#data | （A）-CY-XX→(A) | 该位表示在进行有符号数的加减法时是否发生溢出；  当0V=1时，表示有符号数运算结果发生了溢出，  OV=0时，表示有符号数运算结果没有溢出。  第7位如果产生借位，会使CY置1，否则为0  当第一位相减时，首先将CY清零 |
| MUL | A与B中的两个8位无符号数相乘，结果为16位，低字节在A，高字节在B  1条 | MUL AB |  | 结果大于0FFH，OV置1 |

算术运算指令2-2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 类型 | | 助记符 | | 用途 | | 格式 | 例子 | 备注 |
| 算术运算指令 | DIV | | A与B中的两个8位无符号数相除，商在A，余数在B  1条 | | DIV AB | |  | CY和OV清零  如果B中结果为00H，OV置1 |
| INC | | 内容加1  5条 | | INC A  INC Rn  INC direct  INC @Ri  INC DPTR | |  | 不影响PSW，即使溢出也不影响  注意与ADD的区别 |
| DEC | | 内容减1  4条 | | DEC A  DEC Rn  DEC direct  DEC @Ri | |  | 同INC相同  注意没有DPTR |
| DA | | ADD或ADDC后，压缩BCD数相加结果的调整，再存于A  1条 | | DA A | | (A)=  0101011B  (56 BCD数)  (R3)=  01100111B  (67 BCD数)  ADDC A,R3  得到10111110B  （0BEH）  CY=1  DA A  高低4位都大于9，所以加66H来调整  结果为124 | 低4位大于9或第三位产生进位即AC=1  ,则低四位加6H  高4位大于9或CY=1，则高四位加6H  也就是说，判断高低4位是否大于9、AC=1和CY=1对结果加06H,60H或66H  对CY有影响  对OV无影响  不能直接用于十六进制数转换成BCD数，也不能直接用于减法，减法可先换成补码，相加后进行调整 |

逻辑操作指令1-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 例子 | 备注 |
| 逻辑操作指令 | CLR | 清零A  1条 | CLR A |  |  |
| CPL | 取反A全部内容  1条 | CPL A |  | 不影响标志位 |
| RL | 循环左移A一位  1条 | RL A |  |  |
| RR | 循环右移A一位  1条 | RR A |  |  |
| RLC | 带进位循环左移A一位  1条 | RLC A |  | 用于累加器A中内容\*2 |
| RRC | 带进位循环右移A一位  1条 | RRC A |  |  |
| SWAP | A中的高低4位内容互换  1条 | SWAP A | (A)=FAH  SWAP A  (A)=AFH |  |
| ANL | 逻辑与  6条 | ANL A,Rn  ANL A,direct  ANL A,@Ri  ANL A,#data  ANL direct,A  ANL direct,#data |  | 后两条是直接地址操作  如果是I/O端口，则可以是“读-修改-写”操作 |
| ORL | 逻辑或 | ORL A,Rn  ORL A,direct  ORL A,@Ri  ORL A,#data  ORL direct,A  ORL direct,#data |  | 后两条是直接地址操作  如果是I/O端口，则可以是“读-修改-写”操作 |
| XRL | 逻辑异或 | XRL A,Rn  XRL A,direct  XRL A,@Ri  XRL A,#data  XRL direct,A  XRL direct,#data |  | 后两条是直接地址操作  如果是I/O端口，则可以是“读-修改-写”操作 |

注：都置影响P标志位



图1-3逻辑操作

跳转指令3-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 过程 | 备注 |
| 无条件转移指令 | AJMP | 短转移指令 | AJMP addr11 | ( PC) + 2→PC , addr11→PC10～0 , (PC15～11 )不变 | 2KB范围 |
| LJMP | 长转移指令 | LJMP addr16 | addr16→PC | 64KB程序存储器的任何单元  由于需要三个字节，占用多 |
| SJMP | 相对转移（短转移）指令 | SJMP rel  SJMP 目的地址标号 | (PC) + 2→PC, ( PC) + rel→PC | rel是带符号的8位偏移字节  00H~7FH对应0~127正向转移  80H~FFH对应-128~-1反向转移  特别注意：rel=FEH时也就是-2，目的地址=PC+2-2=PC，会造成无限循环 |
| JMP | 间接转移指令 | JMP @A+DPTR | (A) + (DP TR)→PC | 对DPTR,A和标志位无影响  可散转  散转程序是分支程序的一种，可实现多分支转移。根据某种输入或运算的结果，分别转向各个处理程序段取指令 |
| 空操作指令 | NOP | 单字节指令，只是让PC+1，多用于产生机器周期延迟 | NOP |  |  |

跳转指令3-2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 过程 | 备注 |
| 有条件转移 | JZ | A中内容全为0进行偏移跳转 | JZ rel  rel为带符号8位数-128~127共256字节  也可以是目标符号代替 |  |  |
| JNZ | A中内容不全为0进行跳转 | JNZ rel  rel为带符号8位数-128~127共256字节  也可以是目标符号代替 |  |  |
| CJNE | 比较转移指令 | CJNE A,direct,rel | (PC)+3→PC  (direct)<(A)，(PC)+rel→（PC）且0→CY | 首先PC内容加1  如果后面的比前面的小  按偏移量跳转，CY置1  如果后面的比前面的大  按偏移量跳转，CY置0  如果后面的等于前面的  不跳转，按顺序执行  CY置0  rel是+127~-128字节地址  示意如图1-4 |
| (PC)+3→PC  (direct)>(A)，(PC)+rel→（PC）且1→CY |
| (PC)+3→PC  (direct)=(A)，(PC)+rel→（PC）且0→CY |
| CJNE A,#data,rel | (PC)+3→PC  #data<(A)，(PC)+rel→（PC）且0→CY |
| (PC)+3→PC  #data>(A)，(PC)+rel→（PC）且1→CY |
| (PC)+3→PC  #data=(A)，(PC)+rel→（PC）且0→CY |

跳转指令3-3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 过程 | 备注 |
| 有条件转移 | CJNE | 比较转移指令 | CJNE Rn,#data,rel | (PC)+3→PC  #data<( Rn)，(PC)+rel→（PC）且0→CY | 首先PC内容加3  如果后面的比前面的小  按偏移量跳转，CY置1  如果后面的比前面的大  按偏移量跳转，CY置0  如果后面的等于前面的  不跳转，按顺序执行  CY置0  rel是+127~-128字节地址  示意如图1-4 |
| (PC)+3→PC  #data>( Rn)，(PC)+rel→（PC）且1→CY |
| (PC)+3→PC  #data=( Rn)，(PC)+rel→（PC）且0→CY |
| CJNE @Ri,#data,rel | (PC)+3→PC  #data<((Ri))，(PC)+rel→（PC）且0→CY |
| (PC)+3→PC  #data>((Ri))，(PC)+rel→（PC）且1→CY |
| (PC)+3→PC  #data= ((Ri))，(PC)+rel→（PC）且0→CY |
| DJNZ | 循环转移指令 | DJNZ Rn,rel | (PC)+2→PC  (Rn)-1→Rn  如果(Rn)≠0  则(PC)+rel→PC  如果(Rn)=0  结束循环，程序往下执行 | 图1-5 |
|  |  |  | DJNZ direct,rel | (PC)+2→PC  (direct)-1→direct  如果(direct)≠0  则(PC)+rel→PC  如果(direct)=0  结束循环，程序往下执行 |



图1-4 比较转移



图1-5 循环转移

条用指令1-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 过程 | 备注 |
| 调用返回指令 | ACALL | 在2KB地址空间内调用 | ACALL addr11 | (PC)+2→PC  (SP)+1→SP  (PC0~7)→(SP)  (SP)+1→SP  (PC15~8)→(SP)  addr10~0→PC10~0 | (PC15~11)不变 |
| LCALL | 在64KB的程序存储器范围内调用 | LCALL addr16 | (PC)+3→PC  (SP)+1→SP  (PC0~7)→(SP)  (SP)+1→SP  (PC15~8)→(SP)  Addr15~0→PC15~0 |  |
| RET | 子程序返回到主程序 | RET | ((SP))→PC15~8  (SP)-1→SP  ((SP))→PC7~0  (SP)-1→SP |  |
| RETI | 中断返回指令 | RETI | 同上 | 返回的同时开放中断逻辑 |



图1-6 调用返回指令示意图

位操作指令2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 过程 | 备注 |
| 位数据传送指令 | MOV | 传送 | MOV C,bit  MOV bit,C |  | bit若为00H~7FH是片内RAM（20H~2FH单元）的128位地址  若为80H~FFH则在11个SFR中  也可以是I/O口，共32个  P0.0~P0.7, P1.0~P1.7, P2.0~P2.7,  P3.0~P3.7  本操作必须有一方是C |
| 位修正指令 | CLR | 位清零 | CLR C  CLR bit |  |  |
| SETB | 位置1 | SETB C  SETB bit |  |  |
| CPL | 位取反 | CPL C  CPL bit |  |  |
| 位逻辑运算指令 | ANL | 逻辑与 | ANL C,bit  ANL C,/bit |  | /bit是bit取反后运算，但bit内容不变 |
| ORL | 逻辑或 | ORL C,bit  ORL C,/bit |  |
| 位条件转移类指令 | JC | 检测C  若C=1则偏移转移  若C=0则顺序执行 | JC rel | (PC)+2→PC  若(C)=1  则(PC)+rel→PC  若(C)=0  则顺序执行 | 注意(PC)+2  与JZ正好相反 |
| JNC | 检测C  若C=0则偏移转移  若C=1则顺序执行 | JNC rel | (PC)+2→PC  若(C)=0  则(PC)+rel→PC  若(C)=1  则顺序执行 |

位操作指令2-2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 过程 | 备注 |
| 位条件转移类指令 | JB | 检测bit  若bit=1则偏移转移  若bit=0则顺序执行 | JB bit,rel | (PC)+3→PC  若(bit)=1  则(PC)+rel→PC  若(bit)=0  则顺序执行 | 注意(PC)+3 |
| JNB | 检测bit  若bit=0则偏移转移  若bit=1则顺序执行 | JNB bit,rel | (PC)+3→PC  若(bit)=0  则(PC)+rel→PC  若(bit)=1  则顺序执行 |
|  | JBC | 检测bit  若bit=1则偏移转移，且bit清零  若bit=0则顺序执行 | JBC bit,rel | (PC)+3→PC  若(bit)=1  则(PC)+rel→PC且0→bit  若(bit)=0  则顺序执行 | bit清零 |

注意：1.位累加器C就是进位标志位CY。

2.位地址是片内RAM字节地址20H~2FH，一共128个位，地址为00H~7FH。还有单元地址可被8整除的SFR。

3．ACC(E0H~E7H)、B(F0H~F7H)和片内RAM的128个位可做软件标志或存储位变量

4.位表达方式： （1）直接位地址，如D4H（2）点操作符号方式，如PSW.4（3）位名称方式，如RS1（4）用户自定义，用伪指令bit

伪指令2-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 例子 | 备注 |
| 伪指令 | ORG | 汇编起始命令  规定了该指令后的程序放在ROM的起始地址 | ORG 16位地址 | ORG 2000H  START：MOV A,#64H  START在2000H | ORG可多次出现 |
| END | 汇编程序结束  后面的单片机不管了 | END |  |  |
| EQU | 给数据地址，代码地址，位地址或一个立即数定义一个名字 | 字符名称 EQU项  字符名称可以是一个8位的也可以是16位的 | AA EQU R1  MOV A,AA  就是用AA代表R1  DELY EQU 07EBH  DELY就是16位的 | 1.EQU定义的字符名必须先定义后使用  DATA可以后定义先使用  2.EQU可以把一个汇编符号赋值给字符名称，DATA只能把数据赋给字符名  3.DATA可把表达式赋值给字符名称，其表达式必须是可求值的  一般DATA用来定义数据地址 |
| DATA | 数据地址赋值命令 | 字符名称 DATA 表达式 |  |
| DB | 从当前ROM地址开始，存入数据  数据可以是一字节  用逗号分开的字节串  单引号中的ASCII字符串 | DB 项或项表 | ORG 2000H  DB 0A3H  DB 26H,03H  DB ‘ABC’  得到  (2000H)=A3H  (2001H)=26H  (2002H)=03H  (2002H)=41H  (2003H)=42H  (2004H)=43H  A、B、C的ASCII码是41H,42H,43H | 8位的  只对ROM有用，不能对RAM |

伪指令2-2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类型 | 助记符 | 用途 | 格式 | 例子 | 备注 |
| 伪指令 | DW | 按照16位数据存入当前地址  高8位先存入  低8位后存入 | DW 16位数据项或项表 | ORG 1500H  DW 7234H,8AH,10H  得到  (1500H)=72H  (1501H)=34H  (1502H)=00H  (1503H)=8AH  (1504H)=00H  (1505H)=10H | 16位的  只对ROM有用，不能对RAM |
| DS | 从当前地址开始  保留表达式所代表的值个单元，以备后用 | DS 表达式 | ORG 1000H  DS 08H  DB 30H,8AH  得到  (1008H)=30H  (1009H)=8AH | 只对ROM有用，不能对RAM |
| BIT | 就是给某一位地址起个名，好记 | 字符名 BIT 位地址 | A1 BIT P1.0  P1口第0位的位地址90H起了一个A1名 |  |