ASM-51 汇编伪指令

一、 伪指令分类

1. 符号定义

SEGMENT, EQU, SET, DATA, IDATA, XDATA, BIT, CODE

2. 存储器初始化/保留

DS, DB, DW, DBIT

3. 程序链接

PUBILC, EXTRN, NAME

4. 汇编程序状态控制

ORG, END

5. 选择段的伪指令

RSEG, CSEG, DSEG, XSEG, ISEG, BSEG, USING

二、 伪指令具体说明

1. 符号定义伪指令

1) SEGMENT 伪指令

格式：段名 SEGMENT 段类型

说明：SEGMENT 伪指令说明一个段。段就是一块程序代码或数据存储器。

允许使用的段类型为：

●CODE 代码空间

●DATA 可以直接寻址的内部数据空间

●XDATA 外部数据空间

●IDATA 可以间接寻址的整个内部数据空间

●BIT 位空间

例子：（段符号用于表达式时，代表被连接段的基地址）

```
STACK       SEGMENT     IDATA
RSEG        STACK
DS          10H             ；保留 16 字节做堆栈
MOV         SP , #STACK-1 ；堆栈指针初始化
```

2) EQU 伪指令

格式： 符号名 EQU 表达式

符号名 EQU 特殊汇编符号

说明：EQU 表示把一个数值或特殊汇编符号赋予规定的名字。

一个表达式赋予一个符号，必须是不带向前访问的表达式。

例子：N27 EQU 27;

ACCUM EQU A ;定义 ACCUM 代替特殊汇编符号 A（累加器）

HERE EQU $; HERE 为当前位置计数器的值

3) SET 伪指令

格式： 符号名 SET 表达式

符号名 SET 特殊汇编符号

说明：SET 类似 EQU，区别在于可以用另一个 SET 伪指令在以后对定义过的符号重新定义。

例子：COUNT SET 0

COUNT SET COUNT+1

4) BIT 伪指令

格式： 符号名 BIT 位地址

说明： BIT 伪指令把一个地址赋予规定的符号名。该符号类型取段类型 BIT.

例子：

```
RSEG          DATA_SEG;
CONTROL:      DS          1
ALATM         BIT     CONTROL.0;
OPEN_BOARD    BIT      ALATM+1 ;下一位
RESET_BOARD   BIT     60H ; 下一个绝对的位
```

5）DATA 伪指令

格式： 符号名 DATA 表达式

说明：DATA 伪指令把片内的数据地址赋予所规定的符号名。符号段类型为 DATA.

例子：CONIN DATA SBUF;定义 CONIN 到串行口缓冲器的地址

TABLE_BASE DATA 70H ；把 TABLE_BASE 定义到位置 70H

6）XDATA 伪指令

格式： 符号名 XDATA 表达式

说明： XDATA 伪指令把片外的数据地址赋予所规定的符号名。段类型为 XDATA.

例子：

```
RSEG    XSEG1
ORG     100H
DATE    DS      5；定义 DATE 在偏离 XSEG1 的 100H 处
TIME    XDATA   DATE+5;定义 TIME 为 DATE 后面的 5 个字节.
```

7)IDATA 伪指令

格式： 符号名 IDATA 表达式

说明： IDATA 伪指令将间接的内部数据地址赋予所规定的符号名。类型为 IDATA.

例子：

```
BUFFER          IDATA     60H
BUFFER_LEN      EQU       20H
BUFFER_END      IDATA     BUFFER+BUFFER_LEN-1
```

8）CODE 伪指令

格式： 符号名 CODE 表达式

说明： CODE 伪指令把一个代码地址赋予所规定的符号名。符号段类型为 CODE。

2. 存储器初始化及保留

1) DS 伪指令

格式： 标号 DS 表达式

说明： DS 伪指令以字节为单位保留空间。可以用于除 BIT 类型段以外的任何段。

2) DBIT 伪指令

格式： 标号 DBIT 表达式

说明： DBIT 伪指令以位为单位保留空间。它仅用于 BIT 类型段。在 BIT 段中，位置计数器的基本单位是位而不是字节。

3) DB 伪指令

格式： 标号 DB 表达式

说明：DB 伪指令用字节值对代码存储器初始化。段类型为 CODE 型，其表达式列表是一个由逗号（,）分开的一个或多个字节值或串。

例子：AGE: DB ‘MARY’,0,27

ORIMES:DB 1,2,3,5

4) DW 伪指令

格式： 标号 DW 表达式

说明： DW 伪指令是用字（16 位）值对代码存储器初始化。段类型为 CODE 型。该表达式可以是一个由逗号（,）分开的一个或多个字值，字值可为绝对的或可重新定位的表达式。若在列表中用了位置计数器$，它计算出正被初始化字的代码地址。列表中的每一项以它出现在该列表中的顺序放入存储器，其高字节在前，低字节在后。

例子：ARRIVAL DW 710, ‘AM’

JUMP_TABLE DW GO_PROC,BREAK_PROC,DISPLAY_PROC;由产生地址的清单成一个跳转表。

3. 程序的链接

1) PUBLIC 伪指令

格式：PUBLIC 符号名列表

说明：PUBLIC 伪指令允许当前被汇编模块的符号为外部公用。符号名对于一个时，用逗号（,）分开。寄存器或段符号名（通过 SEGMENT 伪指令声明了的）不可以规定为 PUBLIC。

例子：PUBLIC putctrl, put_data, liner

2) EXTRN 伪指令

格式：ERTRN [段类型（符号名列表）],。。。

说明：EXTRN 伪指令列出当前模块中要访问的在其他模块中定义了的符号。这个伪指令可以出现在程序的任何地方。外部符号必须符合与上每一个符号相关的段类型。(这些类型是 CODE,XDATA,DATA,IDATA,BIT 及 NUMBER，即一个无类型的符号)

例子：EXTRN CODE(pntcrif, putstring, getnum), DATA(count, total)

EXTRN CODE(binas, ascbin), NUMBER(table-size)

3) NAME 伪指令

格式： NAME 模块名

说明：NAME 伪指令用来定义当前的程序模块，应放在该模块全部伪指令及指令代码的前面。若无 NAME 伪指令，则源文件名作为系统对其设置的名称，但不得以数字开始。

4. 汇编程序的状态控制

1） END

格式： END

说明：END 语句不得有标号，允许在它的行上出现一个注释。END 语句是程序的最后一行，否则将产生一个错误。

2） ORG

格式： ORG 表达式

说明：ORG 伪指令用来对汇编程序的位置计数器做修改，以设置一个新的程序起点。该表达式应当是一个绝对的或是可重新定位的表达式，它访问当前的段而不含向前的访问。

ORG 伪指令改变位置计数器，但它并不产生一个新段。如果当前段为绝对段，其值便是当前段的一个绝对地址，如果该段是可以重新定位的，其值是当前段基地址的偏移。

例子：ORG ($+10H) AND 0FFF0H ；将位置计数器设为下一个 16 字节边界

ORG 50 :将位置计数器设置为 50

5. 段选择伪指令

可重定位段：RSEG

格式：RSEG 段名

说明：其段名一定要在前面使用 SEGMENT 伪指令定义过

绝对段：CSEG XSEG DSEG ISEG BSEG

格式：

●CSEG［ AT 绝对地址］;在代码地址空间选择一个绝对段

●XSEG［ AT 绝对地址］;在外部数据地址空间选择一个绝对段

●DSEG［ AT 绝对地址］;在内部数据地址空间选择一个绝对段

●ISEG［ AT 绝对地址］;在间接内部数据地址空间选择一个绝对段

●BSEG［ AT 绝对地址］;在位地址空间选择一个绝对段

如果使用[AT 绝对地址]，汇编器便结束以前规定的绝对段并生成一个从该地址开始的新的绝对段。如果未规定一个绝对地址，该段类型的原来绝对段便继续生效。如果选择段类型为以前未定义的绝对段而且省略了绝对地址，则在位置 0 开始生成一个段。

例子：

```
BSEG          AT    70H
DECIMAL_MODE DBIT  1
CHAT_MIDE    DBIT  1
```

6.USING 伪指令

格式： USING 表达式

说明： USING 伪指令通知汇编器其后面的代码所使用的寄存器组。其表达式是一个数（0-3 之间），它指向四个寄存器组之一。

例子：

```
USING 3
PUSH AR2 ； 压入第三组的第二个寄存器
USING 1
PUSH AR2 ;压入第一组的第二个寄存器
```

51 单片机汇编指令详解

指令格式	功能简述	字节数	周期
一、数据传送类指令			
MOV A, Rn	寄存器送累加器	1	1
MOV Rn, A	累加器送寄存器	1	1
MOV A, @Ri	内部 RAM 单元送累加器	1	1
MOV @Ri, A	累加器送内部 RAM 单元	1	1
MOV A, #data	立即数送累加器	2	1
MOV A, direct	直接寻址单元送累加器	2	1
MOV direct, A	累加器送直接寻址单元	2	1
MOV Rn, #data	立即数送寄存器	2	1
MOV direct, #data	立即数送直接寻址单元	3	2
MOV @Ri, #data	立即数送内部 RAM 单元	2	1
MOV direct, Rn	寄存器送直接寻址单元	2	2
MOV Rn, direct	直接寻址单元送寄存器	2	2
MOV direct, @Ri	内部 RAM 单元送直接寻址单元	2	2
MOV @Ri, direct	直接寻址单元送内部 RAM 单元	2	2
MOV direct2, direct1	直接寻址单元送直接寻址单元	3	2
MOV DPTR, #data16	16 位立即数送数据指针	3	2
MOVX A, @Ri	外部 RAM 单元送累加器(8 位地址)	1	2

MOVX @Ri, A	累加器送外部RAM单元(8位地址)	1	2
MOVX A, @DPTR	外部RAM单元送累加器(16位地址)	1	2
MOVX @DPTR, A	累加器送外部RAM单元(16位地址)	1	2
MOVC A, @A+DPTR	查表数据送累加器(DPTR为基址)	1	2
MOVC A, @A+PC	查表数据送累加器(PC为基址)	1	2
XCH A, Rn	累加器与寄存器交换	1	1
XCH A, @Ri	累加器与内部RAM单元交换	1	1
XCHD A, direct	累加器与直接寻址单元交换	2	1
XCHD A, @Ri	累加器与内部RAM单元低4位交换	1	1
SWAP A	累加器高4位与低4位交换	1	1
POP direct	栈顶弹出指令直接寻址单元	2	2
PUSH direct	直接寻址单元压入栈顶	2	2
二、算术运算类指令			
ADD A, Rn	累加器加寄存器	1	1
ADD A, @Ri	累加器加内部RAM单元	1	1
ADD A, direct	累加器加直接寻址单元	2	1
ADD A, #data	累加器加立即数	2	1

ADDC A, Rn	累加器加寄存器和进位标志	1	1
ADDC A, @Ri	累加器加内部 RAM 单元和进位标志	1	1
ADDC A, #data	累加器加立即数和进位标志	2	1
ADDC A, direct	累加器加直接寻址单元和进位标志	2	1
INC A	累加器加 1	1	1
INC Rn	寄存器加 1	1	1
INC direct	直接寻址单元加 1	2	1
INC @Ri	内部 RAM 单元加 1	1	1
INC DPTR	数据指针加 1	1	2
DA A	十进制调整	1	1
SUBB A, Rn	累加器减寄存器和进位标志	1	1
SUBB A, @Ri	累加器减内部 RAM 单元和进位标志	1	1
SUBB A, #data	累加器减立即数和进位标志	2	1
SUBB A, direct	累加器减直接寻址单元和进位标志	2	1
DEC A	累加器减 1	1	1
DEC Rn	寄存器减 1	1	1
DEC @Ri	内部 RAM 单元减 1	1	1
DEC direct	直接寻址单元减 1	2	1
MUL A B	累加器乘寄存器 B	1	4

DIV A B	累加器除以寄存器 B	1	4
三、逻辑运算类指令			
ANL A, Rn	累加器与寄存器	1	1
ANL A, @Ri	累加器与内部 RAM 单元	1	1
ANL A, #data	累加器与立即数	2	1
ANL A, direct	累加器与直接寻址单元	2	1
ANL direct, A	直接寻址单元与累加器	2	1
ANL direct, #data	直接寻址单元与立即数	3	1
ORL A, Rn	累加器或寄存器	1	1
ORL A, @Ri	累加器或内部 RAM 单元	1	1
ORL A, #data	累加器或立即数	2	1
ORL A, direct	累加器或直接寻址单元	2	1
ORL direct, A	直接寻址单元或累加器	2	1
ORL direct, #data	直接寻址单元或立即数	3	1
XRL A, Rn	累加器异或寄存器	1	1
XRL A, @Ri	累加器异或内部 RAM 单元	1	1
XRL A, #data	累加器异或立即数	2	1
XRL A, direct	累加器异或直接寻址单元	2	1
XRL direct, A	直接寻址单元异或累加器	2	1
XRL direct, #data	直接寻址单元异或立即数	3	2
RL A	累加器左循环移位	1	1
RLC A	累加器连进位标志左循环移位	1	1

RR A	累加器右循环移位	1	1
RRC A	累加器连进位标志右循环移位	1	1
CPL A	累加器取反	1	1
CLR A	累加器清零	1	1
四、控制转移类指令类			
ACCALL addr11	2KB 范围内绝对调用	2	2
AJMP addr11	2KB 范围内绝对转移	2	2
LCALL addr16	2KB 范围内长调用	3	2
LJMP addr16	2KB 范围内长转移	3	2
SJMP rel	相对短转移	2	2
JMP @A+DPTR	相对长转移	1	2
RET	子程序返回	1	2
RETI	中断返回	1	2
JZ rel	累加器为零转移	2	2
JNZ rel	累加器非零转移	2	2
CJNE A, #data, rel	累加器与立即数不等转移	3	2
CJNE A, direct, rel	累加器与直接寻址单元不等转移	3	2
CJNE Rn, #data, rel	寄存器与立即数不等转移	3	2
CJNE @Ri, #data, rel	RAM 单元与立即数不等转移	3	2
DJNZ Rn, rel	寄存器减 1 不为零转移	2	2
DJNZ direct, rel	直接寻址单元减 1 不为零转移	3	2

NOP	空操作	1	1
五、布尔操作类指令			
MOV C, bit	直接寻址位送 C	2	1
MOV bit, C	C 送直接寻址位	2	1
CLR C	C 清零	1	1
CLR bit	直接寻址位清零	2	1
CPL C	C 取反	1	1
CPL bit	直接寻址位取反	2	1
SETB C	C 置位	1	1
SETB bit	直接寻址位置位	2	1
ANL C, bit	C 逻辑与直接寻址位	2	2
ANL C, /bit	C 逻辑与直接寻址位的反	2	2
ORL C, bit	C 逻辑或直接寻址位	2	2
ORL C, /bit	C 逻辑或直接寻址位的反	2	2
JC rel	C 为 1 转移	2	2
JNC rel	C 为零转移	2	2
JB bit, rel	直接寻址位为 1 转移	3	2
JNB bit, rel	直接寻址为 0 转移		
51 单片机汇编指令集			
数据传送类指令（7 种助记符） MOV(英文为 Move)：对内部数据寄存器 RAM 和特殊功能寄存器 SFR 的数据进行传送； MOVC(Move Code)：读取程序存储器数据表格的数据传送；			

MOVX(Move External RAM)：对外部 RAM 的数据传送；

XCH(Exchange)：字节交换；

XCHD(Exchange low-order Digit)：低半字节交换；

PUSH(Push onto Stack)：入栈；

POP(Pop from Stack)：出栈；

算术运算类指令（8 种助记符）

ADD(Addition)：加法；

ADDC(Add with Carry)：带进位加法；

SUBB(Subtract with Borrow)：带借位减法；

DA(Decimal Adjust)：十进制调整；

INC(Increment)：加 1；

DEC(Decrement)：减 1；

MUL(Multiplication、Multiply)：乘法；

DIV(Division、Divide)：除法；

逻辑运算类指令（10 种助记符）

ANL(AND Logic)：逻辑与；

ORL(OR Logic)：逻辑或；

XRL(Exclusive-OR Logic)：逻辑异或；

CLR(Clear)：清零；

CPL(Complement)：取反；

RL(Rotate left)：循环左移；

RLC(Rotate Left throught the Carry flag)：带进位循环左移；

RR(Rotate Right)：循环右移；

RRC （Rotate Right throught the Carry flag)：带进位循环右移；

SWAP (Swap)： 低 4 位与高 4 位交换；

控制转移类指令（17 种助记符）

ACALL(Absolute subroutine Call) 子程序绝对调用；

LCALL（Long subroutine Call）子程序长调用；

RET（Return from subroutine）子程序返回；

RETI（Return from Interruption）中断返回；

SJMP（Short Jump）短转移；

AJMP（Absolute Jump）绝对转移；

LJMP（Long Jump）长转移；

CJNE (Compare Jump if Not Equal)比较不相等则转移；

DJNZ (Decrement Jump if Not Zero)减 1 后不为 0 则转移；

JZ (Jump if Zero)结果为 0 则转移；

JNZ (Jump if Not Zero) 结果不为 0 则转移；

JC (Jump if the Carry flag is set)有进位则转移；

JNC (Jump if Not Carry)无进位则转移；

JB (Jump if the Bit is set)位为 1 则转移；

JNB (Jump if the Bit is Not set) 位为 0 则转移；

JBC(Jump if the Bit is set and Clear the bit) 位为 1 则转移，并清除该位；

NOP (No Operation) 空操作；

位操作指令（1 种助记符）

SETB(Set Bit) 位 置 1