

M68HC08 指令系统

计算机的指令系统是一套控制计算机操作的编码，称之为机器语言。计算机只能识别和执行机器语言的指令，机器语言指令是随计算机系统的不同而不同的。为了容易为人们所理解，便于记忆和使用，通常用符号指令(即汇编指令)来表示计算机的指令。这一章我们以 M68HC08 的汇编指令来分析 M68HC08 的指令系统功能和使用方法。

1 指令格式

1.1 汇编指令格式

M68HC08 汇编指令由操作码助记符字段和操作数字段所组成，指令的格式如下：

操作码 [操作数 1], [操作数 2], [操作数 3]

第一部分“操作码”助记符，由 2-5 个英文字母组成，例如 LD、MOV、DBNXZ、BRSET 等。

第二部分为“操作数”字段，根据指令功能的不同，操作数可以有一个、二个、三个或者没有(例如空操作指令 NOP)。

操作码与操作数之间以空格分隔，操作数和操作数之间用“,”号分开。指令前可以加标号，标号代表这条指令的起始存储地址，指令后可以加注释，注释以“;”号开始，一行指令以回车符结束。例如：

STRT: MOV#FF, DORA; PTA 编程为输出口中

1.2 常用伪指令

一、定位伪指令

格式：ORG 地址

“地址”可以十六进制(以\$开头)或十进制表示，该伪指令给出以下的程序或数据的起始存储地址。

例如：ORG \$C000

则该伪指令下面的程序或常数存放在\$C000 开始的存储器中。

二、定义字节常数伪指令

格式：FCB X₁、X₂、X₃、…X_n

定义程序区中的字节常数，X_i 是单字节常数也可以是用单引号括起来的字符串(这时定义 ASCII 码)，每个字节为一个字符。

例如：ORG \$8000

FCB \$3F, \$61

则在 Flash 的\$8000、\$8001 两个单元定义了两个常数\$3F、\$61。利用字节定义伪指令可以在程序存储器中定义常数表，如显示器的字形数据表等。

三、定义字常数伪指令

格式：FDB Y₁、Y₂、…Y_m

定义程序区中字(双字节)常数，高位字节在前，低位字节在后，Y_i用十六进制或十进制数表示。

例如：ORG \$FFFE

FDB \$8000

则在\$FFFE、\$FFFF 定义了一个字常数，\$80 存储在\$DDDDW 单元，而\$FFFF 单元存储\$0。利用 FDB 指令可以定义复位和中断向量。

1.3 符号说明

M68HC08 汇编程序允许使用标准的 M68HC08 汇编指令和多种伪指令(详见 § 7.1M68HC08 汇编使用方法)。

为了以下说明和分析指令的方便，我们对常用的符号作简单说明。

一、功能性符号

- () 寄存器或存储单元的内容；
- ← 数据传送的方向；
- ↑ 表示退栈；
- ↓ 表示进栈；
- & 表示逻辑或；
- ⊕ 表示逻辑异或；
- × 乘；
- ÷ 除；
- + 加；
- - 减；
- ∴ 表示连成双字节；
- 《 表示扩展为有符号的 16 位数。

二、寄存器或存储单元符号

- A 累加器；
- CCR 条件码寄存器；
- H 变址寄存器高 8 位；
- X 变址寄存器低 8 位；
- PC 程序计数器；
- PCH 程序计数器的高 8 位；
- PCL 程序计数器的低 8 位；
- SP 堆栈指针；
- M 存储器地址或数据(视寻址方式而定)
- rel 相对偏移量(表示有符号的 8 位二进制数)。

三、位标识符号

- V 溢出标志位；
- H 半进位；
- I 中断屏蔽位；
- N 负标志位；
- Z 零标志位；
- C 进位/借位。

四、对条件码寄存器影响：

- - 无影响；
- 0: 清“零”；
- ∴ 根据结果置“1”或清“零”。

五、机器码标志

- dd: \$00XX 的低 8 位地址；
- ee: 16 位偏移量地址；
- ff: 16 位偏移量低 8 位或 8 位偏移量；
- ii: 单字节立即数；
- ll: 16 位扩展寻址低 8 位；
- rr: 相对偏移量。

六、源操作数标志

- OPR: 操作数, 一个或两个字节, 由操作方式确定;
- rel: 相对偏移量。

七、寻址方式

- INH: 隐含寻址;
- IMM: 8 位立即数寻址;
- DIR: 8 位直接地址寻址;
- EXT: 16 位扩展寻址;
- IX: 16 位变址无偏移量;
- IX1: 16 位变址 8 位偏移量;
- IX2: 16 位变址, 16 位偏移量;
- IX+: 16 位变址, 无偏移量, 变址寄存器加 1;
- IX1+: 16 位变址, 8 位偏移量, 变址寄存器加 1;
- rel: 8 位相对寻址;
- DD: 直接寻址;
- IMD: 立即寻址, 直接寻址;
- IX+D: 16 位变址, 直接寻址, 变址寄存器加 1;
- DIX+: 直接寻址, 16 位变址, 变址寄存器加 1;
- SP1: 堆栈指针寻址, 8 位偏移量;
- SP2: 堆栈指针寻址, 16 位偏移量。

2 寻址方式

指令给出操作数或产生操作数有效地址(EA)的方式称为寻址方式。根据寻址方式, CPU 就可以取得操作数。M68HC08 比 M68HC05 增加了 8 种寻址方式, 使程序员能选择最合适的寻址方式来优化程序, 达到缩短程序长度、提高运行速度的目的。下面分别介绍 M68HC08 指令系统的寻址方式。

一、隐含寻址

无操作数或操作数隐含在操作码字节中。

例如: NOP ; 无操作数

CLRA ; 清零累加器

一般只能访问 CPU 寄存器。

二、立即数寻址

操作数包含在指令操作码后面的一个或两个字节中。

例如: LDA # \$10 ; 80H—A

LDHX# \$8100; 81H—H, 0—X

M68HC08 指令系统中, 数字前加\$表示十六进制数, 加%表示二进制数, 无前缀表示十进制数, 指令的数据前加#表示立即数。

三、直接寻址

操作数的有效地址 EA 包含在指令操作码后续字节中。直接寻址的有效地址在指令的地址段中只指出低 8 位的值, 高 8 位固定为零, 即 EA=\$00XX。这种寻址方式只能访问零页的 I/O 寄存器或 RAM。

例如: INC \$50 ; (50H)+1→50H 单元

四、扩展寻址

扩展寻址方式中, 操作数的 16 位有效地址 EA 在指令操作码的后面两个字节中。实际

上也是一种直接寻址方式。因为有效地址为 16 位, 可以访问 64K 字节存储空间的任意单元。

例如：JMP \$C100 ; \$C100→PC, CPU 转至\$C100 执行程序

LDA \$8120; (8120H) →A

在使用 M68HC08 汇编时，用户不必考虑指令是用直接寻址还是扩展寻址。汇编程序自动根据指令中有将地址值取直接寻址或扩展寻址方式。

五、变址寻址无偏移量

这是一种间接寻址方式，操作数的有效地址 EA 的高 8 位在 H 寄存器中，低 8 位在 X 寄存器中。这种寻址方式可以访问 64K 字节存储空间的任意单元。

例如：DEC, X; (HX)-1→(HX)单元

六、变址 8 位偏移量

操作数的有效地址为(H: X)加上无符号的 8 位偏移量。这种寻址方式可用于查表，从 n 个元素的表中选择第 k 个元素，k 在变址寄存器中，表格的首地址为 8 位偏移量。

例如：LDA \$50, X; (HX)+\$50) →A

七、变址 16 位偏移量

操作数的有效地址为(H: X)加上无符号的 16 位偏移量，类似于变址 8 位偏移量，可用于查表和程序散转。例如：

```
FMP   KTAB, X
KTAB  JMP   PGM0
      JMP   PGM1
```

根据(HX)的不同转移到不同的地方。

八、堆栈指针 8 位偏移量

操作数的有效地址是(SP)加上无符号的 8 位偏移量。这种变址 8 位偏移量寻址方式相似。这种寻址方式主要方便对堆栈的操作。若系统中不用中断，SP 可以作为第二变址寄存器。例如：

LDA \$10, SP ; (SP)+\$100) →A

九、操作数有效地址为(SP)加上 16 位无符号的偏移量，类似于变址 16 位偏移量的寻址方式。

LDA \$100, SP ; ((SP)+\$100)→A

十、相对寻址

相对寻址只用于转移指令和子程序调用指令，转移的有效地址为当前 PC 值加上 8 位带符号的偏移量，偏移量位于操作码后面的字节中。在使用 M68HC08 汇编时程序员不必计算偏移量值，只要用标号表示转移的目标地址，汇编程序自动计算偏移量，偏移量为-128~+127，超出这个范围，汇编会报错。例如：

```
BCC   MLP1   ; C=0 转 MLP1, 相对寻址
INC   A      ; (A)+1→A
BRA   MLP2   ; 转 MLP2, 相对寻址
```

MLP1: CLRA

MLP2: STA , X

存储器单元和存储器单元之间的数据传送指令，由源操作数和目的操作数不同寻址方式组合得到以下四种寻址方式。

寻址方式	源操作数	目的操作数	例子	功能
十一、IMD	立即数寻址	直接寻址	MOV # \$80, \$40	\$80→\$40 单元
十二、DD	直接寻址	直接寻址	MOV \$80, \$40	(\$80)→\$40 单元
十三、IX+D	变址, (HX)+1	直接寻址	MOVX+, \$40	((HX))→\$40 单元, (HX)+1

LDA opr, SP								SP2	5
-------------	--	--	--	--	--	--	--	-----	---

例如: LDA # \$80 ; \$80→A

LDA PORTA ; 并行口中 A 输入信息读到 A(先用 EQU 指令定义 PORTA)

2. 所寻址的单元内容送 16 位变址寄存器的低 8 位 X, 而高 8 位 H 的内容保持不变

指令	操作	CCR						寻址方式	总线周期
		V	H	I	N	Z	C		
LDA # opr								IMM	2
LDA opr								DIR	3
LDA opr								EXT	4
LDA opr, X	A←(M)	0	—	—	↕	↕	—	IX2	4
LDA opr,X								IX1	3
LDA,X								IX	2
LDA opr, SP								SP1	4
LDA opr, SP								SP2	5

3. 所寻址的两个单元的 16 位数据送变址寄存器 HX

指令	操作	CCR						寻址方式	总线周期
		V	H	I	N	Z	C		
LDA # opr	H : X ←	0	—	—	↕	↕	—	IMM	2
LDA opr	(M: M+1)							DIR	3

这是两条 16 位数据传送指令, 主要用于对变址寄存器 HX 赋值。例如:

LDHX # \$80 ; 0→H, \$80→X

LDHX \$8212 ; (\$8212) →H, (\$8213) →X

二、CPU 寄存器存储命令

1. 累加器 A 的内容存储到所寻址的单元

例如: STA PORTA ; (A)输出到并行口不 PORTA

STA \$80, X ; (A)存储到地址为(HX)+\$80 的单元

2. 变址寄存器低 8 位 X 的内容存储到所寻址的单元

3. 16 位变址寄存器的内容存储到所寻址的两个单元

例如: STHX \$80; (H)→\$80 单元(X)→\$81 单元

三、堆栈操作指令

1. 进栈指令

M68HC08 的进栈操作是数据先进栈, 然后栈指针减 1, 因此 SP 指向下一次进栈的地

址单元。不影响标志位。

2. 退栈指令

HC08 的退栈操作是栈指针 SP 先加 1，然后取(SP)指出栈单元内容。不影响标志位。

四、CPU 寄存器之间的的数据传送指令

由上面所列指令及其功能可以看出除 TAP 指令外，这类指令不影响标志位。

五、存储器单元之间的数据传送指令

这类指令在编程时应注意格式：MOV 源操作数，目的操作数。例如：

MOV \$50, \$60 ; (\$50)→\$60 单元

MOV # \$80, \$60; 立即数\$80→\$60 单元

3.3 算术运算指令

一、加法指令

1. 不带进位加法指令 ADD

累加器 A 和所寻址的单元内容相加，运算结果送累加器 A。影响标志位 OV、H、N、Z、C，运算结果满足下列条件时置“1”相应的标志位，否则清零。

OV: (A).7, (M).7=1，而结果最高位(RESULT7，以 R7 表示)为 0，负数相加溢出；或 (A).7=0(M).7=0 而结果最高位 R7 为 1，正数相加溢出；

H: 相加结果低 3 位向高 4 位进位时置“1”；

N: 相加结果 R7 为 1 时置“1”；

Z: 相加结果为零时置“1”；

C: 相加时最高产生位进位时置“1”。

2. 带进位加法 ADC

累加器 A、进位标志 C、和所寻址的单元内容相加，结果送累加器 A。对标志位的影响和 ADD 指令相同，ADC 指令用于实现多字节加法。

3. 十进制调整指令

这条指令对累加器中由上一条加法指令(加数和被加数均为压缩 BCD 码)所得的 8 位结果(在 A)进行十进制调整，使 A 的内容为 BCD 码。调整规则如下：

a. (A).3~(A).0 大于 9 或 H=1 则(A)+\$06→A，否则(A).3~(A).0 不变；

b. (A).7~(A).4 大于 9 或 C=1 则(A)+\$60→, 否则(A).7~(A).4 不变;

c. 结束。

例如: (A)=\$56 (\$50)=\$67 执行指令:

ADD \$50 ; (A)=\$BD, (C)=0(A).7~(A).4>9, (A).3~(A).0>9

DAA ; 经调整之后(A)=23, (C)1

4. 16 位变址寄存器加上 8 位带符号立即数指令

这条指令将补码表示的 8 位立即数扩展为 16 位带符号数并和(HX)相加, 运算结果送 HX, 不影响标志位。

5. 16 位堆栈指针加上 8 位带符号立即数指令

这条指令将补码表示的 8 位立即数扩展为 16 位有符号数并和(SP)相加, 结果送 SP, 不影响标志。

6. 加 1 指令

加 1 指令不影响标志位 C 和 H, 例如: (A)=\$FF。

ADD #1 ; 结果(A)=0, (C)=1, (H)=1

INCA ; 结果(A)=0, (C)和(H)不变

二、减法指令

1. 不带借位减法指令

A 的内容减去所寻址单元内容, 结果送累加器 A, 影响 V、N、Z、C 标志位。满足下列条件置“1”, 否则清零。

V: (A).7=1(M).7=0 而结果 R7=0, 为负数减正数溢出; 或者(A).7=0(M).7=1 而结果 R7=1, 为正数减负数溢出。这时 1→V。

N: 相减结果 R7=1 时, 1→N

Z: 相减结果为零时, 1→Z;

C: 相减时最高位产生借位时, 1→C。

2. 带借位减法指令

A 的内容减去 C 和所寻址单元的内容结果送 A, 对标志位的影响和 SUB 指令产生的影响相同。

3. 减 1 指令

对所寻址单元的内容减去 1。

减 1 指令不影响标志位 H 和 C。

4. 累加器 A 的比较指令 CMP

A 和所寻址的单元内容相比较, 实际上是 A 减去所寻址单元内容, 只影响标志位, 而不传送减法的结果, 即不影响 A 和寻址单元的内容, 对标志位影响和减法指令产生的影响相同。

5. X 的比较指令 CPX

X 和所寻址的单元内容的相比较，只影响标志位，不影响 X 和所寻址单元内容，对标志位影响和减法指令产生的影响相同。

6. HX 的比较指令 CPHX

16 位变址寄存器 HX 内容和可寻址的两个单元内容相比较，只影响标志位，不影响 HX 和可寻址单元内容，对标志位影响和减法指令产生的影响相同。

7. 取补指令

0 减去所寻址的单元，即所寻址单元的各位取反，然后加 1。对 V、N、Z、C 标志位影响和减法指令产生的影响相同。

8. 测试指令

测试所寻址单元的内容是否为零或负数。执行所寻址单元内容减去 0 的操作，并影响 V、N、Z 标志位，不影响原来单元内容。

三、乘法指令

X 和 A 中两个无符号数相乘，积的高 8 位送 X，低 8 位送 A。清零 H 和 C 标志，不影响其他标志位。

四、除法指令

(H): (A)中无符号 16 位数除以(X)中 8 位无符号数，商→A，余数送 H。若商大于\$FF 或除数为零，置 1 进位标志 C，否则清零标志 C。

3.4 逻辑运算指令

一、单操作数简单逻辑指令

1. 清零指令

清“0”所寻址的单元，0→V、N、1→Z 不影响其他标志位。

2. 取反指令

对的寻址单元的内容按位取反。

二、双操作数逻辑运算指令

1. “逻辑与”指令

累加器 A 和所寻址的单元内容按位做逻辑与运算，结果选 A。结果最高位 R7=1 置“1”，否则清零 N，结果为零，1→Z，否则 0→Z。

按位“逻辑与”操作在两个操作数对应位都为 1 时，结果对应位才为 1，否则对应结果位为 0。例如：(A)=\$F0 则执行指令 AND# \$0F 后(A)=0。

2. “逻辑或”指令

累加器 A 和所寻址单元内容按位进行逻辑或运算结果送 A。

按位逻辑或操作，只有当两个操作数对应位都为零时，结果的对应位才为零，否则为 1。
例如：(A)=\$F0，执行 ORA# \$0F 后则(A)=\$FF。

3. “逻辑异或”指令

累加器 A 和所寻址单元内容按位进行逻辑异或运算，结果送 A。

按位逻辑异或就是按位半加。当两个操作数对应位不同时结果对应为 1，相同时为 0。
例如：(A)=\$F0，若执行指令 EOR# \$A0，则(A)=\$50。

4. 位测试指令

A 和所寻址的单元内容逻辑与。但结果不传送，只影响标志位 V、N、Z，不影响 A 和所寻址单元内容。

这些指令用于测试所寻址单元的指定位为 0 还是 1。例如：(A0=\$01，执行指令 BIT\$50 后，若 Z=0 则(\$50).0=0

3.5 位操作指令

一、标志位操作指令

1. 对 CCR 寄存器的 I、C 置“1”指令

2. 对 CCR 寄存器的 I、C 清零指令

对标志位 I 的置“1”和清零就是控制 CPU 的关中和开中。

二、对零页 RAM/IO 寄存器的位操作指令

对于零页的 RAM 单元或 IO 寄存器，CPU 执行置 1 和清零指令一样地快速方便。

3.6 移位指令

一、逻辑左移指令

所寻址的单元内容左移一位，低位移入 0，高位移到 C。

二、逻辑右移指令

所寻址的单元内容右移一位，高位移入 0，低位移出至 C。

三、带进位循环左移指令

所寻址单元的内容带进位循环右移一位，C 移入最高位，最低位移出至 C。

五、算术左移指令

算术左移指令功能和逻辑左移指令操作相同

六、算术右移指令

将所寻址的单元内有符号的 8 位数右移一位，符号位保持不变，相当于除 2 操作。

七、累加器 A 半字节交换指令

累加器 A 的高 4 位和低 4 位相互交换，不影响标志位。

3.7 程序转移和控制类指令

一、无条件转移指令

1. 相对转移指令

2. 绝对转移令

绝对转移指令，将所寻址的两个单元内容作为地址送 PC，使 CPU 转到该地址开始执行程序，不影响标志位。

二、条件转移指令

1. 测试状态标志条件转移指令

这类指令，根据上一条指令执行结果，测试 CCR 的标志位状态，若条件满足，执行相对转移指令，若条件不满足，则顺序执行下一条指令，不影响标志位。

若执行有符号减法或比较指令后，CCR 相关位与实际两操作数间的关系为：如果 CPU 内部 A 或 X 的内容大于等于所寻址单元内容，则 $N \oplus V = 0$ ；如果 A 或 X 内容大于所寻址单元内容， $Z1(N \oplus V) = 0$ ；A 或 X 小于所寻址的单元内则 $N \oplus V = 1$ 。

执行无符号减法或比较指令后，CCR 相关位与实际两操作数间的关系为：如果 CPU 内部 A 或 X 的大于等于所寻址单元内容则 $C = 0$ ；A 或 X 大于所寻址单元内容 $C = 0$ ， $Z = 0$ ；A 或 X 内容小于所寻址单元内容 $C = 1$ 。

BLT、BGE、BLE、BGT 指令一般直接跟在有符号数减法或比较指令之后。

2. 测试 \overline{IRQ} 条件转移指令

这两条指令也不影响标志位。

测试所寻址单元相应位状态，若满足条件则发生转移，否则顺序执行下一条指令。只能对零页的 RAM 或 I/O 寄存器的位测试转移。不影响标志位。

4. 减 1 不为零转移

这是兼有减 1 指令和不等零转移指令功能的条件转移指令，不影响标志位。

5. 比较相等转移指令

累加器 A 或 X 和所寻址单元内容相比较，若相等，则执行一相对转移指令；若不相等，顺序执行下一条指令。不影响标志位。

三、调用子程序指令

1. 相对调用指令

该调用子程序指令功能和相对转移指令类似，只是增加 PC 进栈操作也不影响标志位。

2. 绝对调用子程序指令

这些指令功能和绝对转移指令功能相似，只是附加了 PC 值进栈操作。不影响标志位。

四、返回指令

1. 从子程序返回指令 RTS

这条指令的功能是 PC 退栈返回主程序。一般子程序必须以 RIS 结束。

2. 从中断返回指令 RTI

这条的功能和 RTS 相似，只是多了 CPU 寄存器 CCR、A、X 的退栈操作，使 CPU 从原来被中断地方继续执行程序。中断服务程序必须以 RTI 指令结束。

五、控制指令

1. 软件中断指令 SWI

CPU 执行 SWI 指令，产生不可屏蔽的软件中断，主要用于设计开发工具的监控程序。

五、控制指令

1. 软件中断指令 SWI

CPU 执行 SWI 指令，产生不可屏蔽的软件中断，主要用于设计开发工具的监控程序。

2. 堆栈指针复位指令 RSP

RSP 指令使 SPL 值为 \$FF，而 SPH 值保持不变。不影响标志。

注：为保证栈指针能正确初始化，最好使用 LDHX 和 TXS 指令，例对于把 SP 初始化为 \$00FF，可使用：LDHX# \$100 和 TXS 这两条指令。

3. 节电方式指令

4. 空操作指令

空操作指令，指令本身不执行任何操作。用在延时等程序中，以调节程序执行的时间。

上面我们简要地分析了 M68HC08 的指令系统。对汇编语言程序设计有一定基础的读者，就可以用 M68HC08 提供的指令，设计出各种应用程序。