第 3 章　HC08/S08/RS08 CPU

HC08 CPU 有 5 个寄存器、16 种寻址方式、140 条基本指令(加上寻址方式可形成 270 条具体指令),这是学习 08 汇编语言的基础。本章从程序员角度介绍 HC08 CPU 的编程结构、寻址方式、指令系统。为方便学习,对 HC08 CPU 的 270 条具体指令进行了统一编号。由于 HCS08 CPU 指令集是 HC08 CPU 扩充,因此给出了其扩充指令。RS08 CPU 指令集是 HCS08 CPU 指令集的精简版,本章也将给出简要介绍。本章还给出了 08 汇编语言的格式、伪指令。

08 汇编语言是学习 08 系列 MCU 的重要基础之一,学好汇编语言有助于进行底层编程、有助于理解编程结构。轻视汇编语言学习与编程,是不正确的学习方法。实际上,学习 08 汇编语言并不困难,关键是掌握其要领。这些基本要领是:掌握 CPU 内部 5 个寄存器的使用方法、理解 16 种寻址方式、记住常用指令、进行编程实践。

本章的重点有 4 个方面:

(1) HC08 CPU 的寄存器 A、X、HX、CCR、PC、SP。主要理解 HX 作为变址寄存器的用法,理解 SP 采用递减结构。

(2) HC08 CPU 的寻址方式。理解 CPU 的寻址方式对理解和记忆汇编指令很有帮助。

(3) HC08 CPU 的指令系统。虽然短时间内难以记住指令系统,但是当作英语单词进行一定的记忆,对随后看懂程序很有好处。建议重点记忆指令系统简表。

(4) 08 汇编语言基础。必须了解汇编语言的结构和掌握基本伪指令,才能进行汇编入门。对于汇编程序的基本入门,是学习嵌入式系统的基础之一。一点汇编程序都不了解,难于真正理解嵌入式系统的底层开发。

对于 HCS08 CPU 和 RS08 CPU,要求基本了解其与 HC08 CPU 的区别。HC08 CPU 和 HCS08 CPU 差别较小,指令、寻址方式基本相同,HCS08 CPU 只是比 HC08 CPU 多了几条指令。RS08 CPU 是 HCS08 CPU 的简化版本,其 CPU 变化较大,寻址方式、指令系列都有所精简及优化。

3.1　HC08 CPU 基本构成

M68HC08 系列单片机的各种型号的 CPU 均使用 HC08 CPU,有时也将 HC08 CPU 称为 CPU08,它具有如下主要特点:

(1) 目标代码与 M68HC05 系列单片机向上兼容。

(2) 具有 16 位堆栈指针 SP。

(3) 具有 16 位变址寄存器 HX。

(4) 8MHz CPU 内部总线频率。

(5) 64KB 程序/数据存储器空间。

(6) 16 种寻址方式。

(7) 不经过累加器 A 的存储器之间数据直接传送。

(8) 快速 8 位×8 位乘法指令、快速 16 位与 8 位相除指令,增强的 BCD 指令。

(9) 模块化结构,可扩展的内部总线定义可使寻址范围超过 64KB。

(10) 低功耗的 STOP、WAIT 模式。

下面主要对 CPU 中的寄存器功能进行介绍。从编程角度来说,它代表 CPU 的结构,即寄存器是程序员能够直接"看到"的 CPU 结构。至于算术逻辑部件(ALU)的功能与一般计算机含义相同,在此略。

HC08 CPU 中有 5 个寄存器:累加器 A、变址寄存器 HX、堆栈指针 SP、程序计数器 PC 和条件码寄存器 CCR。这 5 个寄存器与端口部件的寄存器不同,它们可以被指令直接执行或被控制器直接控制,不需要像端口部件寄存器那样通过存储器映射方式进行读/写。5 个寄存器的结构示意图如图 3-1 所示。

1) 累加器 A(accumulator)

累加器 A 是 8 位通用寄存器,用来存放操作数和运算结果。数据读取时,累加器 A 用于存放从存储器读出的数据;数据写入时,累加器 A 用于存放准备写入存储器的数据。在执行算术、逻辑操作时,累加器首先存放一个操作数,执行完毕时累加器存放操作结果。累加器 A 是指令系统中最灵活的一个寄存器,各种寻址方式均可对之寻址。复位时,累加器的内容不受影响。

2) 变址寄存器 HX(index pointer)

HC08 CPU 的变址寄存器 HX 是 16 位寄存器,H 是高 8 位,X 是低 8 位,可单独使用。变址寄存器 HX 主要用于变址寻址方式中确定操作数的地址,也可以用来存放临时数据,作为一般寄存器使用。复位时,高 8 位被清零。

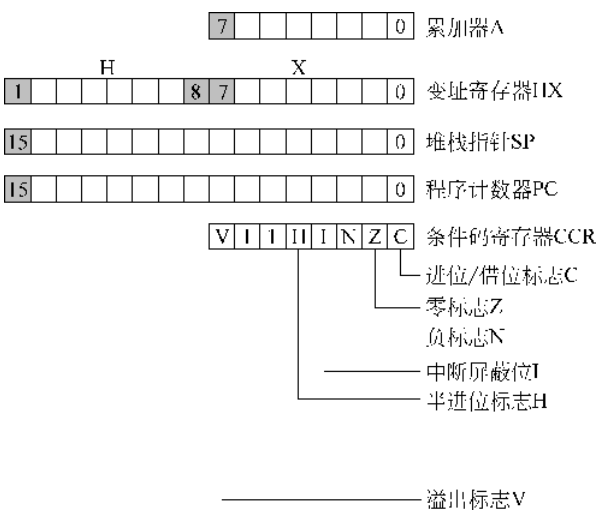

图 3-1　HC08 CPU 的寄存器

3) 堆栈指针 SP(stack pointer)

SP 是指向下一个栈地址的 16 位寄存器,堆栈指针 SP 采用递减的结构,即进栈时 SP 减 1,出栈时 SP 加 1。复位时,SP 的初值为 $00FF(前置符号 $ 表示十六进制数,下同)。栈

指针复位指令(RSP)可将SP的低8位置为$FF,而不影响高8位。

在堆栈指针作为8位或16位的偏移量寻址方式中,SP充当变址寄存器功能,CPU利用SP的内容确定操作数的条件地址。

4) 程序计数器PC(program counter)

程序计数器PC也是16位的,可寻址范围达64KB。PC存放下一条指令的地址,在执行转移指令时存放转移地址,在执行中断指令时存放中断子程序入口地址。复位时,程序计数器PC装入地址$FFFE和$FFFF中的内容。一般地,地址$FFFE和$FFFF中的内容是复位的入口地址,这样,复位后,程序就能够从复位入口地址开始执行程序。复位入口地址也称为复位向量地址或复位向量地址(reset vector address),意味着复位状态过后,PC指向该处,从这里执行程序。

5) 条件码寄存器CCR(condition code register)

条件码寄存器CCR是8位的寄存器(如图3-1所示),其中5位(除了中断屏蔽位I)用于指示执行完指令的结果状态,这些位可由程序来测试。第6位(D6)和第5位(D5)永远为1,其余位有具体的含义,分别介绍如下:

(1) V(D7)——溢出标志位(overflow flag)。

当二进制补码溢出时置位。有符号跳转指令BGT、BGE、BLE和BLT使用该标志。

(2) H(D4)——半进位标志位(half-carry flag)。

执行加法指令(ADD)和带进位加法指令(ADC)时,如果相加的结果的低4位向上产生进位,即累加器D3向D4有进位,则CPU将半进位标志H置"1"。该标志对于一般运算是没有用的,但在二进制编码的十进制(BCD)数据运算中则很有用,由于BCD码是以4位二进制数来表示一位十进制数,所以在BCD码算术运算中,半进位标志H记录的是一位十进制数的进位。在进行十进制调整(DAA)时,利用H和C的状态来判断是否调整。

(3) I(D3)——中断屏蔽标志位(interrupt mask flag)。

I=1表示屏蔽中断,即禁止中断;I=0表示允许中断,即开放中断。复位时,该位被置"1",可用CLI指令开中断。中断响应时,CPU将除H寄存器以外的寄存器推入堆栈,然后执行中断服务子程序,遇到RTI指令时,从栈中恢复包括CCR在内的各寄存器。当然也包括这一位的状态。为了保持与M68HC05系列兼容,变址寄存器高字节即H寄存器,在中断时未被自动保护(CPU内部自动保护了A、X、CCR、PC),若在中断服务子程序中用到H寄存器,则需要程序对H进行保护,可用PSHH和PULH使其进栈、出栈。

(4) N(D2)——负标志位(negative flag)。

CPU进行运算过程中,如果产生负结果,则将负标志N置为"1"。该情况用于有符号位的运算,数据位的最高位D7作为符号位,如果D7=1,说明数据为负,如果D7=0,说明数据为正。如果寄存器或存储单元的D7为1,那么把寄存器或存储单元的内容送入累加器A时,就会使N=1。如果寄存器或存储单元的D7为0,那么把寄存器或存储单元的内容送入累加器A时,就会使N=0。所以,负标志位N可用于检查有关寄存器或存储单元D7位的状态。

(5) Z(D1)——零标志位(zero flag)。

CPU进行运算过程中,如果数据或运算结果为0,零标志位Z被置1,否则被清0。

(6) C(D0)——进位/借位标志(carry/borrow flag)。

当进行加法运算时,在最高位D7上有进位;或在进行减法运算时D7需要向更高位借

位，则 CPU 将进位/借位标志 C 置 1，否则清 0。一些指令如位测试、跳转、移位指令等也会影响该标志。

3.2 寻址方式

指令是对数据的操作，通常把指令中所要操作的数据称为操作数，HC08 CPU 所需的操作数可能来自：寄存器、指令代码、存储单元。而确定指令中所需操作数的各种方法称为寻址方式(addressing mode)。寻址方式越多，指令系统的功能就越强，灵活性也越大。MC68HC08 系列单片机的寻址方式共有 16 种，比 MC68HC05 单片机增加了 8 种。下面逐一介绍。

1. 内在寻址方式(INH，Inherent addressing mode)

内在寻址，是指那些在指令中已经包含了操作数所在之处。内在寻址的指令是没有明确指出操作数的指令，操作数由指令隐含，且为单字节指令。这类指令只是执行内在的数据操作。例如累加器加 1 指令 INCA、中断返回指令 RTI、停止指令 STOP 和置进位标志指令 SEC 等。

2. 立即寻址方式(IMM，Immediate addressing mode)

立即寻址，是在指令中直接给出操作数。这种指令是双字节指令，第 1 个字节是操作码，第 2 个字节是参与操作的立即数。立即寻址指令通常是对立即数和累加器内容或变址寄存器内容进行操作。下述指令属于 IMM 寻址方式。

```
LDA   # $FC          //把十六进制数 $FC 放入累加器 A 中
```

特别说明：在 Freescale(Motorola)公司单片机的指令系统中，规定在数字前加 $ 表示十六进制数，加%表示二进制数，无前缀表示十进制数。另外，指令中的数据前加 # 表示立即数。

3. 直接寻址方式(DIR，Direct addressing mode)

直接寻址指令，字长为 2 个字节，它可以对存储器的起始 256 个地址($0000～$00FF)，叫内存直接页或内存第 0 页进行访问。指令的第 1 个字节是操作码，第 2 个字节是操作数地址。在直接寻址中，CPU 自动把 $00 作为操作数高位地址，第 2 个字节作为操作数的低位地址，其寻址范围为 $0000～ $00FF。下述指令属于 DIR 寻址方式。

```
LDA   $60          //将存储单元 $60 中的数取至累加器 A
```

BRSET 和 BRCLR 是 2 条很特别的 3 字节指令，它们用直接寻址方式取得操作数，用相对寻址方式指明转移目的地址。例如：

```
BRCLR # 3, $50, A1          //若存储单元 $50 的第 3 位＝0 则转至标号 A1 处
BRSET # 3, $50, A1          //若存储单元 $50 的第 3 位＝1 则转至标号 A1 处
```

4. 扩展寻址方式(EXT，Extended addressing mode)

扩展寻址指令是 3 字节指令，可访问存储器中的任何地址。在这种指令中，第 1 个字节是操作码，第 2 个字节是操作数的高字节地址，第 3 个字节是低字节地址。扩展寻址是相对于直接寻址方式而言的，其寻址范围为 $0000～ $FFFF，比直接寻址范围大得多。下述指

令属于 EXT 寻址方式，因为它的地址已经超过了 DIR 寻址方式的范围。

```
LDA   $023D          //将存储单元 $023D 中的数取至累加器 A
```

特别说明：实际编程时，程序员不必考虑是直接寻址还是扩展寻址，汇编程序会自动识别，主要区别在于汇编产生的指令长度不一样。

5. 无偏移量变址方式(IX，Indexed，no offset addressing mode)

无偏移量变址的指令是单字节指令。在这种指令中，变址寄存器 HX 中的内容是操作数的地址。所以无偏移量变址指令可以对 $0000～$FFFF 地址进行寻址。下述指令属于 IX 寻址方式。

```
LDA   ,X          //从变址寄存器 HX 指向的存储器单元中取数到累加器 A 中
COM   ,X          //将变址寄存器 HX 指向的存储单元数据取反
```

无偏移量变址指令通常用于数据表中移动数据指针，或者用于保存频繁使用的 RAM 或 I/O 单元的地址。

特别说明：在变址方式中，在操作数位置出现的 X，实际代表变址寄存器 HX，这样书写，仅是因为 HC08 是从 HC05 发展而来，而 HC05 系列 MCU 中没有 H，下同。

6. 8 位偏移量变址方式(IX1，Indexed，8-bit offset addressing mode)

8 位偏移量变址是双字节指令，CPU 把变址寄存器 HX 的内容和指令第 2 个字节内容相加，其和便是操作数地址。下述指令属于 IX1 寻址方式。

```
STA  $80,X        //将 A 中的数存至地址为 HX＋ $80 的存储器单元中
LDA  $80,X        //将存储单元 HX＋ $80 中的数取至累加器 A
```

7. 16 位偏移量变址方式(IX2，Indexed，16-bit offset addressing mode)

该变址方式指令是 3 字节指令，可访问存储器的任何地址。CPU 将变址寄存器 HX 的内容与指令的第 2、3 字节相加，和为操作数地址。在第 2、3 字节中，第 2 字节是高位地址，第 3 字节是低位地址。下述指令属于 IX2 寻址方式。

```
STA   $280,X        //把 A 中的数存放到以 HX＋ $280 为地址的存储器单元中
LDA   $280,X        //将存储单元 HX＋ $280 中的数取至累加器 A
```

特别说明：实际编程时，程序员不必考虑是 8 位偏移量变址方式(IX1)还是 16 位偏移量变址方式(IX2)，汇编程序会自动识别，主要区别在于汇编产生的指令长度不一样。

8. 相对变址寻址方式(REL，Relative addressing mode)

相对变址只用于转移指令。当转移条件满足时，CPU 将指令中的偏移量和程序计数器 PC 的内容相加，得出转移的目的地址。如果转移条件不满足，则 CPU 执行下一条指令。转移指令中的偏移量是带符号的，且用 1 字节表示，因此转移指令的偏移量为－128～＋127。偏移量用二进制补码表示。相对变址转移指令的转移范围在下一条指令地址的－128～＋127 之间。下述指令属于 REL 寻址方式。

```
BRA   A1          //无条件转向标号 A1 处执行
BSR   L1          //调用标号 L1 处的子程序
```

特别说明：以上 8 种寻址方式在 M68HC05 系列中也具有，以下 8 种为 M68HC08 系列单片机新增的寻址方式，其中 9～12 为存储器到存储器的数据传送寻址方式，13～14 是含

有加 1 的变址寻址，15～16 为堆栈寻址。

9. 存储器：直接地址——直接地址寻址方式(DD，Direct to direct addressing mode)

在存储器的 4 种数据直接传送的寻址方式中，欲传送的数据直接从源存储单元送向目的存储单元，无须寄存器中转。在本寻址方式中，源地址与目标地址由指令直接给出。只有一条指令为 DD 寻址方式：

```
MOV   N1,N2          //把内存变量 N1 赋值给内存变量 N2
```

等同于下列 2 条指令，但上述指令不影响累加器 A。

```
LDA   N1
STA   N2
```

等同于 C 语言语句：

```
N2=N1;
```

特别说明：这里设 N1、N2 为内存变量，下同。内存变量的声明方法，将在 5.5 节中讲述。

10. 存储器：直接地址——变址、变址加 1 的寻址方式(DIX＋，Direct to indexed with post increment addressing mode)

源地址为直接地址，目标地址在 HX 中。把源地址的数据存至目标地址后，HX＋1→HX。仅有一条指令为 DIX＋寻址方式。

```
MOV $80,X+          //($80)→(HX)，HX+1→HX
```

等同于下列 3 条指令，但上述指令不影响累加器 A，主要用于内存或端口数据填充一个表格。

```
LDA   $80           //($80)→A
STA   ,X            //A→(HX)
AIX   #1            //HX+1→HX
```

特别说明：操作说明栏中，($80)表示地址为 $80 的内存单元内容。(HX)表示地址为 HX 值的内存单元。若内存单元作为源操作数，表示从该内存单元中取数，若内存单元作为目的操作数，表示将数存至该内存单元。

11. 存储器：立即数——直接地址寻址方式(IMD，Immediate source to direct destination addressing mode)

该寻址方式将立即数存入直接地址，指令给出的地址即作为目的地址，在指令书写中，立即数位于直接地址之前，下述指令为 IMD 寻址方式。

```
MOV   #86,N1          //将立即数#86 赋值给内存变量 N1
```

等同于 C 语言语句：

```
N1=86;
```

12. 存储器：变址——直接地址、变址加 1 的寻址方式(IX＋D，Indexed with post increment to direct addressing mode)

HX 为源地址，直接地址为目的地址，把源地址的数据存至目标地址后，HX＋1→HX。仅有一条指令为 IX＋D 寻址方式。

```
MOV  X+,$80          //(HX)→($80),HX+1→HX
```

等同于下列 3 条指令,但上述指令不影响累加器 A,主要用于向端口传送内存数据块。

```
LDA  ,X          //(HX)→A
STA  $80         //A→($80)
AIX  #1          //HX+1→HX
```

13. 无偏移量变址、变址加 1 寻址方式(IX+,Indexed, no offset, post increment addressing mode)

在这种寻址方式与无偏移量变址寻址方式 IX 的区别在于,在本寻址方式中多了 HX+1→HX 的操作。只有一条指令属于 IX+寻址方式,该指令先完成比较操作,但不论条件是否满足,均有 HX+1→HX。通常用于寻找一段存储区中是否有与 A 中相等的数。

```
CBEQ X+,rel          //若 A=(HX)则转移,同时 HX+1→HX
```

14. 8 位偏移量变址、变址加 1 寻址方式(IX1+,Indexed, 8-bit offset, post increment addressing mode)

与上条指令的差别是:目的操作数的地址为 8 位直接地址+(HX),属于 IX+寻址方式的指令也只有一条。

```
CBEQ addr8,X+,rel          //若 A=(HX+addr8)则转移,同时 HX+1→HX
```

15. 8 位偏移量堆栈寻址方式(SP1,Stack pointer,8-bit offset addressing mode)

8 位偏移量堆栈寻址,是将指令给出的 8 位直接地址与堆栈指针 SP 相加形成操作数的地址,下列指令属于 SP1 寻址方式。

```
LDA  2,SP          //(2+SP) →A
```

由于 M68HC08 系列单片机的堆栈方向是向地址更低的方向堆放,那么从堆栈指针初始化处至堆栈指针处便是实际使用的堆栈空间,利用带有偏移量的堆栈寻址,可以直接取出堆栈空间中的数据,无须通过出栈操作。利用该方式取出堆栈中数据,且不改变堆栈的数据结构。主要用于子程序中开辟局部变量,这种寻址方式增强了对 C 语言的支持。在讲子程序规范时,会对此用法做详细说明。

16. 16 位偏移量堆栈寻址方式(SP2,Stack pointer,16-bit offset addressing mode)

16 位偏移量堆栈寻址,将指令给出的 16 位直接地址与堆栈指针 SP 相加形成操作数的地址,下列指令属于 SP2 寻址方式。

```
LDA  $130,SP          //将地址 $130+SP 中的数取至 A 中
```

特别说明:实际编程时,程序员不必考虑 8 位偏移量堆栈寻址还是 16 位偏移量堆栈寻址,汇编程序会自动识别,它们的主要区别在于汇编产生的指令长度不一样。

3.3 指令系统

CPU 的功能是从外部设备获得数据,通过加工、处理,再把处理结果送到 CPU 的外部世界。设计一台计算机,首先需要设计一套可以执行特定功能的操作命令,这种操作命令称

为指令。CPU 所能执行的各种指令的集合，称为该微处理器的指令系统。设计一种微处理器，一般从设计它的指令系统开始。指令系统因机器不同而不同。HC08 CPU 共有 118 个保留字，加上寻址方式形成了 270 条具体指令，比 HC05 CPU 增加了 78 条指令。为了方便学习，将这些指令分为数据传送、算术运算、逻辑运算、位操作、移位、程序控制及其他等 7 类。本节分别介绍这些指令，并对每条具体指令进行统一编号。表 3-1 列出了保留字的简明含义。

表 3-1　HC08 CPU 指令简表

类　型	保　留　字	含　义
数据传送类	LDA、LDX、LDHX	取数到 A、X、HX 中
	STA、STX、STHX	存 A、X、HX 到内存
	PSHA、PSHX、PSHH、PULA、PULX、PULH	进栈、出栈
	TAP、TPA、TAX、TXA、TXS、TSX	寄存器间传送
	MOV	存储器单元之间传送
算术运算类	ADD、ADC、SUB、SBC、MUL、DIV	加、减、乘、除
	INC、INCA、INCX、DEC、DECA、DECX	加 1/减 1
	COM、COMA、COMX、NEG、NEGA、NEGX	求反/取补
	CMP、CPX、CPHX	比较
	CLR、CLRA、CLRX、CLRH	清零
	TST、TSTA、TSAX	测试是否为 0
	AIS、AIX	SP、HX 增加
逻辑运算类	AND、ORA、EOR	与、或、异或
位操作类	BIT、BCLR、BSET、CLC、SEC、CLI、SEI	位测试、清位、置位
移位类	ASL、ASLA、ASLX、LSL、LSLA、LSLX	算术左移、逻辑左移
	ASR、ASRA、ASRX、LSR、LSRA、LSRX	算术右移、逻辑右移
	ROL、ROLA、ROLX、ROR、RORA、RORX	循环左移、循环右移
程序控制类	BCC、BCS、BHCC、BHCS、BIL、BIH、BMC、BMS	标志位测试转移
	BHI、BHS、BLO、BLS、BEQ、BNE	无符号数比较后转移
	BPL、BMI、BGE、BGT、BLE、BLT	有符号数比较后转移
	BRCLR、BRSET	位测试转移
	BRA	无条件相对转移
	CBEQ、CBEQA、CBEQX	比较相等转移
	DBNZ、DBNZA、DBNZX	减 1 不为 0 转移
	JMP	无条件绝对地址跳转
	JSR、BSR、RTS	调子程序、子程序返回
其他指令	SWI、RTI、DAA、NSA、RSP、NOP、BRN、STOP、WAIT	

本节随后给出 HC08 CPU 指令系统的全部指令，并给出必要的解释。为了使得指令格式简明易懂，在不影响意义理解情况下，涉及 A、H、X、CCR 等寄存器的操作省略括号，而(HX)则表示 HX 所指向的存储器地址单元，(addr8)表示 8 位地址单元，(addr16)表示 16 位地址单元。书末附录给出的按字母排序的指令系统索引表中涉及 A、H、X、CCR 等寄存器的操作未省略括号，请予注意，特别是对部分指令中涉及 HX 操作，要根据寻址方式来理解。附录 B 中的指令系统索引表含有机器码、执行机器周期等信息，供必要时查阅。

3.3.1　数据传送类指令

1. 取数指令

取数指令如表 3-2 所示，其功能是取出存储器中的数放入寄存器 A、X、HX 中。同时按取出的数来改变 N、Z 标志，当取出的数为负(最高位为 1)时，则负标志位 N=1，当取出的数为 0 时，则零标志位 Z=1。对其他标志位没有影响。LD 是 Load 的简写，随后的字母是 CPU 内部寄存器名(LD 与寄存器名之间没有空格)。

表 3-2　取数指令表

编　号	指　令	操　作	寻址方式
(1)	LDA #opr8	#opr8 → A	IMM
(2)	LDA addr8	(addr8) → A	DIR
(3)	LDA addr16	(addr16) → A	EXT
(4)	LDA addr16,X	(addr16+HX) → A	IX2
(5)	LDA addr8,X	(addr8+HX) → A	IX1
(6)	LDA ,X	(HX) → A	IX
(7)	LDA addr8,SP	(addr8+SP) → A	SP1
(8)	LDA addr16,SP	(addr16+SP) → A	SP2
(9)	LDX #opr8	#opr8 → X	IMM
(10)	LDX addr8	(addr8) → X	DIR
(11)	LDX addr16	(addr16) → X	EXT
(12)	LDX addr16,X	(addr16+HX) → X	IX2
(13)	LDX addr8,X	(addr8+HX) → X	IX1
(14)	LDX ,X	(HX) → X	IX
(15)	LDX addr8,SP	(addr8+SP) → X	SP1
(16)	LDX addr16,SP	(addr16+SP) → X	SP2
(17)	LDHX #opr16	#opr16 → HX	IMM
(18)	LDHX addr8	(addr8：addr8+1) → HX	DIR

特别说明：addr8 是指 8 位地址，或指在地址处于第 0 页(地址高 8 位为 0)的变量，addr16 是指 16 位地址，或指在地址处于非第 0 页(地址高 8 位不为 0)的变量，实际编程时尽可能把常用的内存变量开辟在第 0 页，减少汇编后的程序长度。但把变量开辟在哪一页，对程序无明显影响。

LDA、LDX 是取一个字节的操作，均有 8 种寻址方式。LDHX 是取 2 个字节的操作，只有 2 种寻址方式。

```
LDHX   # $36EF          //把十六进制数 36EF 放入 HX 中
LDHX   $0058           //把 $0058、$0059 两个存储器单元的内容取到 HX 中
```

特别说明：指令 LDHX $0058 的功能是一次读取两个存储器单元 $0058、$0059 中的数放入 HX 中，其中存储器单元 $0058 中的数被放入 H 中，存储器单元 $0059 中的数被放入 X 中。数据存放时应注意这个特点。

2. 存数指令

存数指令如表 3-3 所示，其功能是将寄存器 A、X、HX 的数，存入存储器单元中。对标

志位的影响同上。ST 是 Store 的简写，随后的字母是 CPU 内部寄存器名(ST 与寄存器名之间没有空格)。

表 3-3　存数指令表

编　号	指　令	操　作	寻址方式
(19)	STA addr8	A → addr8	DIR
(20)	STA addr16	A → addr16	EXT
(21)	STA addr16,X	A → addr16＋HX	IX2
(22)	STA addr8,X	A → addr8＋HX	IX1
(23)	STA ,X	A → HX	IX
(24)	STA addr8,SP	A → addr8＋SP	SP1
(25)	STA addr16,SP	A → addr16＋SP	SP2
(26)	STX addr8	X → addr8	DIR
(27)	STX addr16	X → addr16	EXT
(28)	STX addr16,X	X → addr16＋HX	IX2
(29)	STX addr8,X	X → addr8＋HX	IX1
(30)	STX ,X	X → HX	IX
(31)	STX addr8,SP	X → addr8＋SP	SP1
(32)	STX addr16,SP	X → addr16＋SP	SP2
(33)	STHX addr8	(HX) → addr8；addr8＋1	DIR

STA、STX 是存 1 个字节的操作，均有 7 种寻址方式。STHX 是存 2 个字节的操作，只有 1 种寻址方式。

```
STHX $00ED          //把 HX 的内容存储到存储器 $00ED、$00EE 单元中
```

3. 堆栈操作指令

堆栈操作指令如表 3-4 所示，其功能是将 A、H、X 进栈、出栈，不改变标志位。PSH 是 Push(推)的简写，PUL 是 Pull(拉)的简写，寄存器名 A、H、X 与简写字母之间没有空格。

表 3-4　堆栈操作指令表

编　号	指　令	操　作	寻址方式
(34)	PSHA	A 进栈；SP－1 → SP	INH
(35)	PSHH	H 进栈；SP－1 → SP	INH
(36)	PSHX	X 进栈；SP－1 → SP	INH
(37)	PULA	SP＋1 → SP；A 出栈	INH
(38)	PULH	SP＋1 → SP；H 出栈	INH
(39)	PULX	SP＋1 → SP；X 出栈	INH

4. 寄存器间数据传送指令

寄存器间数据传送指令如表 3-5 所示，其功能是完成寄存器 A、X、HX、SP、CCR 之间的数据传送，不改变标志位。

表 3-5 寄存器间数据传送指令表

编 号	指 令	操 作	寻址方式
(40)	TAP	A → CCR	INH
(41)	TPA	CCR → A	INH
(42)	TAX	A → X	INH
(43)	TXA	X → A	INH
(44)	TXS	HX−1 → SP	INH
(45)	TSX	SP+1 → HX	INH

5. 存储器间数据传送指令

存储器间数据传送指令如表 3-6 所示,其功能是完成存储器单元之间数据的直接传送,对标志位的影响情况是按传送的数来改变 N、Z 标志,当传送的数为负(最高位为 1)时,则负标志位 N=1,当传送的数为 0 时,则零标志位 Z=1,对其他标志位没有影响。

表 3-6 存储器间数据传送指令表

编 号	指 令	操 作	寻址方式
(46)	MOV addr 源,addr 目的	(addr 源) → 目的 addr	DD
(47)	MOV addr,X+	(addr) → (HX); HX+1→HX	DIX+
(48)	MOV #opr8,addr	#opr8 → addr	IMD
(49)	MOV X+,addr	(HX) → addr; HX+1→HX	IX+D

3.3.2 算术运算类指令

算术类指令有加、减、乘/除、加 1/减 1、求反/取补、比较、测试等。

1. 加、减指令

加、减指令如表 3-7 所示,其第一操作数均在 A 中,第二操作数在存储器中(有 8 种寻址方式),结果均放在 A 中。根据运算结果改变 V、H、N、Z、C 标志位:

C 若有最高位的进位(减法为借位),则置位,否则清零。
V 若溢出,则置位,否则清零。
H 加法,若位 3 有进位,即半字节有进位,则置位,否则清零(减法不影响 H)。
N 若结果的最高位为 1,相当于有符号运算中的结果为负,则置位,否则清零。
Z 若结果为 0,则置位,否则清零。

加法有不带进位加法(ADD)和带进位加法(ADC)两种,减法有不带借位减法(SUB)和带借位减法(SBC)两种。

表 3-7 加、减指令表

编 号	指 令	操 作	寻址方式
(50)	ADD #opr8	A+#opr8 → A	IMM
(51)	ADD addr8	A+(addr8) → A	DIR
(52)	ADD addr16	A+(addr16) → A	EXT
(53)	ADD addr16,X	A+(addr16+HX) → A	IX2
(54)	ADD addr8,X	A+(addr8+HX) → A	IX1
(55)	ADD ,X	A+(HX) → A	IX
(56)	ADD addr8,SP	A+(addr8+SP) → A	SP1