第 3 章
指令系统与汇编语言程序设计

S12X 系列单片机具有强大的指令系统，以高速 16 位 S12X CPU 核为基础。S12X CPU 指令集完全兼容以前的 CPU12 指令集，并增加了一些新指令，支持 16 位数据通道支持高效快速的算术操作，支持奇数字节指令，包括很多单字节指令能更有效地利用 ROM 空间，指令缓冲队列技术可以使 CPU 能够在最短时间内寻址多字节的机器码；寻址方式灵活多样，特别是具有强大的变址寻址能力，允许堆栈指针 SP 和程序计数器 PC 作为变址寄存器，通过累加器 A、B、D 提供偏移量以及自动调整变址指针等，这大大增强了寻址的灵活性。此外，S12X CPU 指令类别齐全、功能丰富，包含数据传送、算术运算、逻辑运算、位操作、移位、控制、全局地址读写、特殊等类别，共有多达几百条的不同功能的指令。

3.1 CPU 寄存器

在 MCU 的程序设计中，除了需要处理、操作 RAM 数据外，还要使用到大量的各种用途的寄存器。为了以示区分，本书将 S12X MCU 的寄存器分为两个类别：I/O 寄存器和 CPU 寄存器（如图 3-1 所示）。

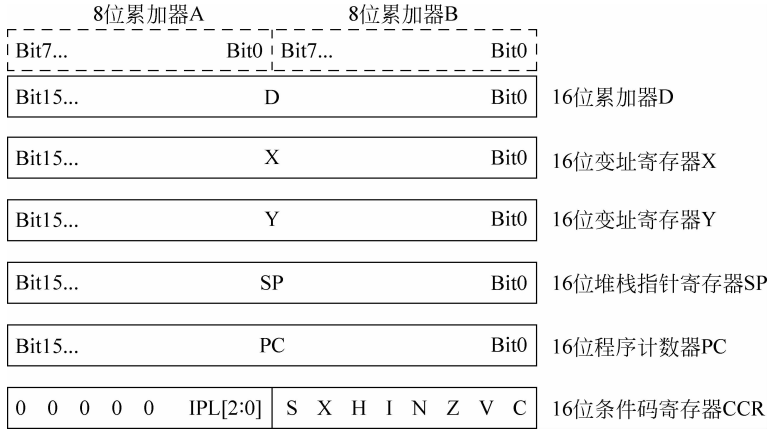

图 3-1 CPU 寄存器

I/O 寄存器就是 S12X 的众多 I/O 接口、功能模块的几百个寄存器，占用 2K 字节的存储器映射空间，各个寄存器功能固定、各有侧重，都有各自的有自明含义的寄存器名，用于对 MCU 的各个资源模块的管理、设置或写入/读出操作。I/O 寄存器也可称为功能寄存器。

CPU 寄存器专指 S12X CPU 内部寄存器，简称 CPU 寄存器，只有 D、X、Y、SP、PC 和 CCR 这 6 个寄存器（可以看成是 7 个）。它们直接与 CPU 的 ALU 相连，具有比 RAM 存储器更快的读/写速度。CPU 寄存器在程序代码中频繁使用，用于直接参与操作、暂存中间数

据、存放操作结果、作为寻址指针等。编程时需要涉及这些 16 位 CPU 寄存器，如图 3-1 所示，分别具体说明如下。

1. 累加器 D(A、B，Accumulator)

S12X CPU 有两个 8 位累加器：累加器 A 和累加器 B。累加器 A 和累加器 B 合起来就是一个 16 位累加器 D，其高 8 位在累加器 A，低 8 位在累加器 B。累加器 A、B 是指令系统中最灵活、最常用的寄存器，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。累加器 D、A、B 实际上是同一个寄存器。累加器 D(A、B)主要用于存放操作数和运算结果，对于大多数算术运算指令，累加器用做目的寄存器。

2. 变址寄存器 X、Y (Index Register)

两个 16 位变址寄存器 X、Y 主要用于寻址操作，扮演为地址指针；它们也可用于临时存放数据并参与运算。在变址寻址方式下，变址寄存器的内容加上 5 位、9 位或者 16 位的值或者加上一个累加器中的内容得到指令操作的有效地址。寄存器 X、Y 内容不受复位影响。

3. 堆栈指针寄存器 SP(Stack Pointer)

S12X CPU 的 16 位堆栈指针寄存器 SP 主要用于堆栈管理，服务于中断和子程序调用。堆栈指针 SP 采用递减结构，即进栈时 SP 减 1，出栈时 SP 加 1，堆栈内容后进先出。堆栈指针 SP 总是指向堆栈区的顶部。SP 也可在 8 位或 16 位的偏移量寻址方式中作为变址寄存器。SP 的内容不受复位影响。

4. 程序计数器 PC(Program Counter)

程序计数器 PC 是一个 16 位指针，寻址范围 64KB，其内容始终指向程序序列中下一条将要执行的指令地址，包括在执行转移指令时存放转移地址、在执行中断时存放中断子程序入口地址。用户可以读取 PC 指针，但不能直接写入 PC 指针。复位后，PC 自动回到默认状态。PC 是特殊的寄存器，一般不能挪作他用，但也可以像 SP 一样，作为变址寄存器使用。

5. 条件码寄存器 CCR(Condition Code Register)

在 S12 CPU 中它是 8 位寄存器，在 S12X CPU 中它已被扩展成了 16 位寄存器。为了兼容以前指令，16 位 CCR 寄存器也可以看成是两个 8 位寄存器：CCRH 和 CCR。

CCR 也称程序状态寄存器，包括 5 个算术特征位：H、N、Z、V、C，它们反映上一条指令执行结果的特征；还有 3 个 MCU 控制位：STOP 指令控制位 S 和中断屏蔽位 X、I，这 3 位通常由软件设定，控制 S12X CPU 的操作。其具体定义如下：

条件码寄存器——CCRH:CCR

复位默认值：0000 0000 1101 0000B

读写	Bit15～Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
R	0	IPL[2:0]			S	X	H	I	N	Z	V	C
W												

IPL[2:0]：记录当前最新中断的较高优先级级别，当 RTI 指令后自动将入栈的原先的中断优先级级别恢复。是 S12X CPU 较以前 S12 CPU 所增加的。

S：停止模式(STOP 指令)禁止位。该位置 1 将禁止 CPU 执行 STOP 指令。

X：XIRQ 中断屏蔽位。该位置 1 将屏蔽来自 XIRQ 引脚的中断请求，复位默认值为 1。

H：辅助进位标志位。该位 BCD 操作时累加器 A 的 Bit3 向 Bit4 进位。

I：中断屏蔽位。该位置 1 将屏蔽所有的可屏蔽中断源，复位默认值为 1。

N：负标志位。当操作结果为负时，该位置 1。

Z：零标志位。当操作结果为 0 时，该位置 1。

V：溢出标志位。当操作结果出现补码溢出时，该位置 1。

C：进位/借位标志位。当加法运算产生进位或者减法运算产生借位时，该位置 1。一些测试、跳转、移位操作或者直接针对 C 的指令也可改变 C 的值。

3.2　寻 址 方 式

寻址方式是指 CPU 在执行指令时确定操作数所在的单元地址的方式。在 MCU 中，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，CPU 所需的操作数可能来自寄存器、指令代码或存储单元，CPU 在执行指令时(NOP 指令除外)，都要先找到操作数的地址，从中得到内容，然后再完成相应的动作。显然，寻址方式越多，指令系统的功能就越强、灵活性越大。

S12X CPU 指令共可综合为 9 种寻址方式(Addressing Mode)。

1. 隐含寻址(Inherent Addressing，INH)

指令本身已经隐含了操作数所在地址的寻址方式。指令的操作数隐含在助记符中或无需操作数，这类指令一般为单字节指令。例如 ROLA、PSHB、INX 等指令，操作数隐含是 CPU 寄存器 A、B、X 中；NOP 指令无需操作数。

2. 立即寻址(Immediate Addressing，IMM)

指令的操作数在指令中立即给出，在汇编语言中用“#”号代表一个立即数。立即寻址类指令常用于给某一寄存器赋值。

例如：

```
LDDA   #$8F              ;将十六进制数 8F 立即装载到 A 中
LDX    #1234             ;将十进制数 1234 立即装载到 X 中
```

3. 直接寻址(Direct Addressing，DIR)

直接寻址又叫零页寻址，指令中直接给出操作数的地址。这种方式可以直接访问存储器空间中 $0000～$00FF 段的 256 个单元，直接寻址方式默认的地址高 8 位为 $00，指令中只需给出单字节地址。在 S12X 单片机默认的存储器地址分配中，这一段是 I/O 寄存器地

址，因此访问这些 I/O 寄存器，可以使用直接寻址。

例如：

```
LDAA   $55              ;将 8 位地址$0055 单元的内容装载到 A 中
```

4. 扩展寻址(Extended Addressing，EXT)

扩展寻址与直接寻址类似，指令中给出操作数地址，只是这时的地址是 16 位地址，可以寻址整个 64K 地址空间，寻址范围远大于立即寻址方式。

例如：

```
LDAA   $200A            ;将 16 位地址 $200A 单元的内容装载到 A 中
```

5. 相对寻址(Relative Addressing，REL)

相对寻址只用于转移指令，用于程序跳转和子程序调用。在程序中写出需要跳转的目的地址的标号，汇编语言程序会自动计算出相对转移地址并完成跳转，例如：

```
BRA    LABEL            ;无条件跳转到 LABEL 标号的地址处
BRA    *                ;无条件跳转到当前地址(*)，此语句实现原地等待
BCC    DONE             ;如果 C 标志为 0，则跳转到 DONE 标号的地址处
```

6. 变址寻址(Indexed Addressing，IDX)

变址寻址方式以变址寄存器 X、Y 或者 SP、PC 寄存器的内容为基址，再加上或减去一个偏移量，作为操作数的最终地址。这个偏移量可以是 5 位(－16～15)、9 位(－256～255，IDX1)或 16 位(－32768～32767，IDX2)常数，也可以是 0，对应指令的占用字节数分别为 2 字节、3 字节和 4 字节，功能相同。例如：

```
STD    7, X             ;5 位常数偏移量，偏移量为 7
                        ;X 寄存器内容加上 7 作为地址，2 字节的 D 内容存储到这里
                        ;其中，低地址字节存 D 的 A，高地址字节内容存 D 的 B
LDAA   0, X             ;5 位常数偏移量，偏移量为 0
                        ;X 寄存器内容作为地址，其指向单元的内容装载到 A
LDAB   - $FA, PC        ;9 位常数偏移量，偏移量为 - $FA
                        ;PC 寄存器内容减去 $FA 作为地址，其指向单元内容装载到 B
LDAA   1000, X          ;16 位常数偏移量，偏移量为 1000
                        ;X 寄存器内容加 1000 作为地址，其指向单元内容装载到 A
```

7. 累加器变址寻址(Accumulator Offset Indexed Addressing，IDX)

累加器变址寻址也简称 IDX，这种变址寻址的偏移量来自累加器 A、B 或 D，基址寄存器内容加上这个无符号偏移量构成操作数的地址。例如：

```
LDAA   D,X      ; 将 X 值加上 D 值作为地址，其指向的字节内容装载到 A
LDAD   A,Y      ; 将 Y 值加上 A 值作为地址，其指向的字内容装载到 D
```

8. 自加自减的变址寻址(Auto Pre/Post Decrement/Increment Indexed Addressing,IDX)

有附带功能的变址寻址,这种寻址方式提供4种方式(先加、先减、后加、后减)去自动改变变址寄存器值,加、减数值的范围是1～8,然后确定操作数的地址。变址寄存器可以是X、Y和SP,这种变址寻址对于连续数据块的操作十分方便,适合字节、字、双字、四字变量的快速定位。例如:

```
STAA   1, - SP          ;SP 寄存器先减 1,然后将 A 内容存储到 SP 指向的单元
                        ;等效于入栈指令 PSHA
LDX    2,SP +           ;SP 指向的字内容先装载到 X 寄存器,然后 SP 寄存器加 2
                        ;等效于出栈指令 PULX
MOVW   2, X + , 4, + Y  ;X 寄存器内容指向的字数据传送到 Y + 4 指向的地址单元
                        ;传送后 X 自动后加 2,传送前 Y 已经自动先加 4
```

9. 间接变址寻址(Indirect Indexed Addressing,[IDX2]/[D,IDX])

该寻址方式将变址寄存器的值加上一个16位偏移量或累加器D的值,形成一个地址,该地址中内容并不是实际操作数,该地址中所存放的内容才是最终操作数的有效地址。例如:

```
LDAA   [1000, X]        ;((1000 + X))→A
                        ;X + 1000 的地址单元内容作为地址,其指向内容装载到 A
LDAA   [D, Y]           ;((D + Y))→A
                        ;Y + D 的地址单元内容作为地址,其指向内容装载到 A
```

以上寻址方式中,后面的几种变址寻址是S12X CPU的重要、高效的寻址方式,相当于C语言中的指针操作;而间接变址寻址则相当于C语言中的“指针的指针”。变址寻址方式的通常表象是:如果指令操作码后面跟的操作数是变址寄存器X、Y或者SP、PC寄存器,那么该指令的寻址方式就是变址寻址(LEA指令除外),X、Y、SP、PC将扮演指针的作用。

另外,如果把一个16位数写入存储器,则高8位在存储器低地址处,低8位在存储器高地址处。Freescale公司的CPU对于16位、32位数据与存储器字节的对应关系都是规定为高位低地址、低位高地址,这和Intel公司的规定正好相反,Intel公司CPU系列使用高位高地址、低位低地址的方式。

3.3　指 令 概 览

S12X CPU的常用指令按照操作类别大致可以分为数据传送类指令、算术运算类指令、逻辑运算类指令、程序控制类指令、CPU控制类指令、中断类指令、全局读写指令和其他指令。每一类别中又有很多小类和多种指令,指令条数繁多,还涉及各种寻址方式,难以一一尽述,也不需要使用者全面掌握。本节内容旨在帮助使用者对S12X的一般指令集的功能、特点和使用方法有一个基础认识,各指令的使用可以在实际编程应用中临时查阅。S12X汇编指令索引总表参见本书附录A,其中左边3列分别是指令助记符、操作功能和寻址方

式，第 4 列是指令编译后形成的机器码，第 5 列是指令的执行周期数及每个周期的具体动作（每个字母代表一个时钟周期，分别表示读、写、空闲等），最右边两列是指令影响标志位的情况（△表示影响该位，－表示不影响该位，1 表示该位置 1，0 表示该清零）。

S12X CPU 的指令采用 Freescale 公司自定义的指令助记符方法，指令的英语名称上能够反映指令的基本含义，需要在使用的过程中体会理解，例如：

```
CLI     =   CLear I
LDAA    =   LoaD Accumulator A
STAB    =   STore Accumulator B
TAB     =   Transfer A to B
MOVB    =   Move Byte
BEQ     =   Branch EQual zero
PSHA    =   PuSH A
RTI     =   ReTurn of Interrupt
GLDAD   =   Global LoaD Accumulator D
…
```

3.3.1 数据传送类指令

数据传送指令包括寄存器与寄存器之间、寄存器与存储器单元之间、存储器单元与存储器单元之间的传送形式。在后续的指令表中，()表示寄存器或存储器内容，M 表示存储器地址，M:M+1 表示连续的两个地址，(M:M+1)表示 M、M+1 两个相邻存储单元的内容组成的一个字内容，(M)为高位字节内容，即高 8 位数存放在低位地址单元存储器中，H、L 分别表示高 8 位和低 8 位，其他类同。

1. 寄存器装载指令

寄存器装载(Load)指令将存储器的内容复制到寄存器中来，而源存储器中的内容不变。该类指令(LEA 指令除外)会自动更新 N、Z 标志位，V 标志位清零。指令如表 3-1 所示。

表 3-1 寄存器装载指令

助 记 符	功 能	操 作
LDAA	Load A	(M)→A
LDAB	Load B	(M)→B
LDD	Load D	(M:M+1)→(A:B)
LDS	Load SP	(M:M+1)→SPH:SPL
LDX	Load index register X	(M:M+1)→XH:XL
LDY	Load index register Y	(M:M+1)→YH:YL
LEAS	Load effective address into SP	Effective address→SP
LEAX	Load effective address into X	Effective address→X
LEAY	Load effective address into Y	Effective address→Y

指令使用举例如下：

```
LDAA    # $3F      ;立即数 $3F 装载到累加器 A 中,操作后 N = 0,Z = 0,V = 0
LDAB    $20FA     ;把 $20FA 单元的内容取到累加器 B 中
LDD     8, X      ;变址寄存器 X 的内容加 8 作为地址,对应单元的内容送 D 的高位(A)
                  ;下一个单元的内容送到 D 的低位(B)
LDS     $1000,Y   ;变址寄存器 Y 的内容加 $1000 作为地址,对应单元的内容送到 SP 高 8 位
                  ;下一个单元的内容送到 SP 低 8 位
LDX     A, PC     ;将(PC + A)单元的内容送 X 高 8 位,下一个单元的内容送到 X 低 8 位
LDY     2, SP +   ;将 SP、SP + 1 对应单元的内容分别送 Y 的高、低 8 位,然后 SP 内容加 2
LEAS    2, X      ;X 内容加 2 后送到 SP; 基址 + 偏移直接形成了有效地址
LEAX    B, Y      ;Y 的内容加上 B 的内容送到 X; 基址 + 偏移直接形成了有效地址
LEAY    D, SP     ;SP 的内容加上 D 的内容送到 Y; 基址 + 偏移直接形成了有效地址
```

2. 寄存器存储指令

寄存器存储(Store)指令将寄存器的内容复制到存储器中去,而源寄存器的内容不变。该类指令会自动更新 N、Z 标志位,V 标志位清零。指令如表 3-2 所示。

表 3-2　寄存器存储指令

助　记　符	功　　能	操　　作
STAA	Store A	(A)→M
STAB	Store B	(B)→M
STD	Store D	(A)→M, (B)→M+1
STS	Store SP	(SPH:SPL)→M:M+1
STX	Store X	(XH:XL)→M:M+1
STY	Store Y	(YH:YL)→M:M+1

指令使用举例如下：

```
STAA    $10         ;将累加器 A 的内容保存到 $0010 单元
STD     - $2000, X  ;将累加器 D 的 A、B 内容保存到 X - $2000、X - $2000 + 1 单元
STY     2, + SP     ;SP 内容先加 2,然后将 Y 的高、低 8 位分别保存到 SP、SP + 1 单元
```

3. 寄存器传输指令

寄存器传输(Transfer)指令复制一个寄存器内容到另一个寄存器,而源寄存器的内容不变。指令如表 3-3 所示。

S12X CPU 指令主要使用 3 个：TAB、TBA 和 TFR。TAB、TBA 指令会影响 N、Z 和 V 标志位。TFR 是一个寄存器与寄存器之间通用传输指令,TFR 指令不影响标志位。这 3 条指令与以前版本的 CPU 指令也兼容,其他指令是为了兼容以前的 S12 CPU。

TFR 指令可以在 16 位和 8 位寄存器之间传送,它的处理方法是：8 位到 8 位或 16 位到 16 位时直接传输；8 位到 16 位时高 8 位补 0 变成 16 位后传输；16 位到 8 位时舍弃高位,只传输低位。

表 3-3 寄存器传输指令

助记符	功　能	操　作
TAB	Transfer A to B	(A)→B
TBA	Transfer B to A	(B)→A
TFR	Transfer register to register	(A,B,CCR,D,X,Y,SP)→A,B,CCR,D,X,Y,SP
TSX	Transfer SP to X	(SP)→X
TPA	Transfer CCR to A	(CCR)→A
TAP	Transfer A to CCR	(A)→CCR
TSY	Transfer SP to Y	(SP)→Y
TXS	Transfer X to SP	(X)→SP
TYS	Transfer Y to SP	(Y)→SP

指令使用举例如下：

```
TFR   A, X          ;将累加器 A 的内容传输到 X 的低 8 位,X 高 8 位清零
```

建议：平常使用时,尽量不使用位数不匹配的方式进行寄存器传输。

4. 寄存器交换指令

寄存器交换(Exchange)指令的作用是交换一对寄存器的内容。交换指令的运行不影响标志位。S12X CPU 的指令实际只有 EXG 一条,其他两条指令是为了兼容以前的 S12 CPU。EXG 指令中,当第一个操作数是 8 位,第二个操作数是 16 位,8 位数补 0 扩展后复制到 16 位寄存器中,而 16 位数的低 8 位复制到 8 位寄存器中,其高 8 位被舍弃。指令如表 3-4 所示。

表 3-4 寄存器交换指令

助记符	功　能	操　作
EXG	Exchange register to register	(A,B,CCR,D,X,Y,SP)←→(A,B,CCR,D,X,Y,SP)
XGDX	Exchange D with X	(D) ←→(X)
XGDY	Exchange D with Y	(D) ←→(Y)

指令使用举例如下：

```
EXG   X, SP         ;SP 与 X 内容交换
EXG   A, B          ;A 与 B 内容交换
EXG   B, X          ;B 内容送到 X 低 8 位, $00 送到 X 高 8 位,X 低 8 位送到 B
```

建议：平常使用时,尽量不使用位数不匹配的方式进行寄存器交换。

5. 存储器数据传送指令

存储器数据传送(Move)指令将源操作数(1 个字节或字)传送到目的地址,源操作数不变。直接寻址、扩展寻址、变址寻址的 6 种组合允许指定源地址和目的地址。源操作数可为立即数或存储器数,目的地址不能是立即数。存储器数据传送不影响标志位。这类指令共有两个助记符,是不经过寄存器的直接存储区数据传送的。指令如表 3-5 所示。

表 3-5　存储器数据传送指令

助记符	功　能	操　作
MOVB	Move byte (8bit)	(M1)→M2
MOVW	Move word (16bit)	(M1:M1+1)→M2:M2+1

指令使用举例如下：

```
MOVB    # $5A, 6, Y        ;将立即数 $5A 送到(Y+6)单元
MOVW    # $103F, $2011     ;将立即数 $10、$3F 分别送到( $2011)、( $2012)单元
MOVB    $103F, $2011      ;将( $103F)单元内容送到( $2011)单元
MOVB    1, X, 1, Y        ;将(X+1)内容送到(Y+1)单元
```

6. 堆栈操作指令

有关堆栈的指令分为两种：一种是堆栈指针的操作指令，用于特殊形式的数据传送，指令形式与变址寄存器 X、Y 的操作类似；另一种是堆栈操作(入栈、出栈)指令，用来从系统堆栈中保存和恢复信息，遵从“先入后出、后入先出”的堆栈原则。堆栈作为存储器单元的表现形式，本质上就是一个 RAM 存储区，因此也可以通过其他寻址方式直接访问堆栈内的数据。除了 PULC 外，其他均不影响标志位。其中堆栈操作指令如表 3-6 所示。

表 3-6　堆栈操作指令

助记符	功　能	操　作
PSHA	Push A	(SP)－1→SP；(A)→M(SP)
PSHB	Push A	(SP)－1→SP；(B)→M(SP)
PSHC	Push CCR	(SP)－1→SP；(CCR)→M(SP)
PSHCW	Push CCRH:CCR	(SP)－2→SP；(CCRH:CCR)→M(SP):M(SP+1)
PSHD	Push D	(SP)－2→SP；(A:B)→M(SP):M(SP+1)
PSHX	Push X	(SP)－2→SP；(X)→M(SP):M(SP+1)
PSHY	Push Y	(SP)－2→SP；(Y)→M(SP):M(SP+1)
PULA	Pull A	(M(SP))→A；(SP)＋1→SP
PULB	Pull B	(M(SP))→B；(SP)＋1→SP
PULC	Pull CCR	(M(SP))→CCR；(SP)＋1→SP
PULCW	Pull CCRH:CCR	(M(SP):M(SP+1))→CCRH:CCR；(SP)＋2→SP
PULD	Pull D	(M(SP):M(SP+1))→A:B；(SP)＋2→SP
PULX	Pull X	(M(SP):M(SP+1))→X；(SP)＋2→SP
PULY	Pull Y	(M(SP):M(SP+1))→Y；(SP)＋2→SP

指令使用举例如下：

```
PSHB    ;累加器 B 入栈
PSHD    ;累加器 D 入栈,B 先入栈,A 后入栈
PULX    ;从堆栈中弹出 2 字节到寄存器 Y,第 1 字节送到 Y 高 8 位,第 2 字节送到 Y 低 8 位
PULC    ;从堆栈中弹出 1 字节到 CCR
```

3.3.2 算术运算类指令

S12X 算术运算指令不仅包括加、减、乘、除、比较以及十进制调整的 BCD 指令，还支持乘加运算、表插值运算等，并设置了符号扩展指令。算术指令影响标志位。

1. 加、减法指令

有符号和无符号的 8 位或 16 位加、减法运算能在寄存器之间或寄存器和存储器之间执行。带进位的加、减法指令能实现多字节的准确运算。加减运算结果都仍然保存在助记符所包含的寄存器中。指令如表 3-7 所示。

表 3-7 加、减法指令

助 记 符	功 能	操 作
加法指令		
ABA	Add B to A	(A)+(B)→A
ABX	Add B to X	(B)+(X)→X
ABY	Add B to Y	(B)+(Y)→Y
ADCA	Add with carry to A	(A)+(M)+C→A
ADCB	Add with carry to B	(B)+(M)+C→B
ADDA	Add without carry to A	(A)+(M)→A
ADDB	Add without carry to B	(B)+(M)→B
ADDD	Add to D	(A:B)+(M:M+1)→A:B
减法指令		
SBA	Subtract B from A	(A)−(B)→A
SBCA	Subtract with borrow from A	(A)−(M)−C→A
SBCB	Subtract with borrow from B	(B)−(M)−C→B
SUBA	Subtract memory from A	(A)−(M)→A
SUBB	Subtract memory from B	(B)−(M)→B
SUBD	Subtract memory from D (A:B)	(A:B)−(M:M+1)→A:B

指令使用举例如下：

```
ADDA    # $35          ;累加器 A 的内容加上立即数 $35,结果存回 A 中
ADCB    8, Y           ;累加器 B 的内容加上(Y+8)单元的内容,再加上进位位,结果存回 B 中
ADDD    $200A          ;累加器 D 的内容加上($200A)单元的内容,结果存回 D 中
SUBA    $2011          ;累加器 A 的内容减去($2011)单元的内容,结果存回 A 中
SBCB    $80, X         ;累加器 B 的内容减去(X+$80)单元的内容,再减去进位位,结果存回
                       ;B 中
```

2. 加 1、减 1 指令

加 1 和减 1 指令是优化的 8 位和 16 位加、减法操作，实际上是对寄存器 X、Y、A、B 或存储器字节的加 1、减 1 计算。这类指令通常用来实现计数，一般用于指针的调整或循环控制。它们不会影响 CCR 中的进位位 C 标志。指令如表 3-8 所示。