指令详解

为了便于说明汇编语言指令，首先定义一些符号和缩写，这些符号和缩写都将在指令中使用。表 1 列出 C54xx 系列 DSP 指令系统中所使用的符号和缩写及其意义。表 2 详细列出 C54xx 系列 DSP 的指令系统中所使用的一些特殊符号和缩写及其意义。

表 1　指令系统符号与缩写以及其意义

符号	意　　义
A	累加器 A
ACC	累加器
ACCA	累加器 A
ACCB	累加器 B
ALU	算术逻辑单元
AR	辅助寄存器
ARx	特指某个辅助寄存器（$0 \le x \le 7$）
ARP	ST0 寄存器中的辅助寄存器指针位；该位指向当前辅助寄存器（AR）
ASM	ST1 寄存器中的 5 位累加器移位方式位（$-16 \le ASM \le 15$）
B	累加器 B
BRAF	ST1 寄存器中的块循环有效标志位
BRC	块循环计数器
BITC	该 4 位的值决定位测试指令对指定的数据存储器值的哪一位进行测试（$0 \le BITC \le 15$）
C16	ST1 寄存器中的双 16 位/双精度算术选择方式位
C	ST0 寄存器中的进位位
CC	2 位条件代码（$0 \le CC \le 3$）
CMPT	ST1 寄存器中的兼容方式位
CPL	ST1 寄存器中的编译方式位
cond	表示条件执行指令所使用的条件
[d]，[D]	延迟方式
DAB	D 数据总线
DAR	DAB 地址寄存器
dmad	16 位立即数表示的数据存储器地址（$0 \le dmad \le 65535$）
dmem	数据存储器操作数
DP	ST0 中的 9 位数据页指针位（$0 \le DP \le 511$）
dst	目的累加器（A 累加器或 B 累加器）

Dst	目的累加器的反　if dst=A，then Dst=B if dst=B　then Dst=A
dst_	目的累加器的反　if dst=A，then dst_=B if dst=B　then dst_=A
EAB	E 地址总线
EAR	EAB 地址寄存器
extpmad	23 位立即数表示的程序存储器地址
FRCT	ST1 寄存器中的分数方式位
H	十六进制数据
h	十六进制数据
hi(A)	累加器 A 的高端（位 32～16）
HM	ST1 寄存器中的保持方式位
IFR	中断标志寄存器
INTM	ST1 寄存器中的中断屏蔽位
K	少于 9 位的短立即数
k3	3 位立即数（$0 \le k3 \le 7$）
k5	5 位立即数（$-16 \le k5 \le 15$）
k9	9 位立即数（$0 \le k9 \le 115$）
lk	16 位长立即数
Lmem	使用长字寻址 32 位单数据存储器操作数
mmr MMR	存储器映射寄存器，AR0～AR7 或 SP
MMRx MMRy	存储器映射寄存器，AR0～AR7 或 SP
n	紧跟 XC 指令的字数，n=1 或 2
N	指定在 RSBX、SSBX 和 XC 指令中修改的状态寄存器 N=0，状态寄存器 ST0 N=1，状态寄存器 ST1
OVA	ST0 寄存器中的累加器 A 的溢出标志
OVB	ST0 寄存器中的累加器 B 的溢出标志
OVdst	目的累加器（A 或 B）的溢出标志
OVdst_	目的累加器反（A 或 B）的溢出标志
Ovsrc	源累加器（A 或 B）的溢出标志
OVM	ST1 寄存器中的溢出方式位
PA	16 位立即数表示的端口地址

PAR	程序地址寄存器
PC	程序指针
pmad	16 位立即数表示的程序存储器地址（$0 \leq pmad \leq 65535$）
pmem	程序存储器操作数
PMST	处理器方式状态寄存器
prog	程序存储器操作数
[R]	四舍五入选项
md	四舍五入
RC	循环计数器
RTN	在指令 RETF[D]中使用的快速返回寄存器
REA	块循环尾地址寄存器
RSA	块循环开始地址寄存器
SBIT	该 4 位的值指明在指令 RSBX、SSBX 和 XC 中修改的状态寄存器位数（$0 \leq SBIT \leq 15$)
SHFT	4 位移位数（$0 \leq SHFT \leq 15$)
SHIFT	5 位移位数（$-16 \leq SHIFT \leq 15$)
Sind	使用间接寻址的单数据存储器操作数
Smem	16 位单数据存储器操作数
SP	堆栈指针
SPC	程序页指针
src	源累加器（A 或 B）
ST0	状态寄存器 0
ST1	状态寄存器 1
SXM	ST1 寄存器中的符号扩展方式位
T	暂存器
TC	ST0 寄存器中的测试／控制标志位
TOS	堆栈栈顶
TRN	过渡寄存器
TS	T 寄存器的 5～0 位确定的移位数（$-16 \leq TS \leq 31$）
uns	无符号的
XF	ST1 寄存器中的外部标志状态位
XPC	程序计数器扩展寄存器

Xmem	在双操作数指令和一些单操作数指令中，使用的 16 位双数据存储器操作数
Ymem	在双操作数指令中使用的 16 位双数据存储器操作数

表 2　指令系统中特殊符号与缩写以及其意义

符号	意义	符号	意义
‖	并行执行	|	或运算
&	与运算	^	异或运算
＋＋	增 1	——	减 1
= =	相等	=	赋值于
[x]	[x]可选项	#	表示一个立即数
x→y	x 的内容送到 y	$\overline{x}$	x 的补码
|x|	x 的绝对值	0x	十六进制数据
>>	右移	<<	左移
×	乘法	*	指针指向的地址

TMS320C54xx 系列 DSP 的指令一共有 129 条，按功能分为如下几类：算术指令、逻辑指令、程序控制指令、存储和装入指令以及循环指令。下面根据每种指令功能的不同，划分成小组列表的形式以便于查询，并就常用指令的用法作简要的说明，详细的说明请参考附录中的指令详解。

1　算术指令

算术运算指令可分为加法指令（ADD）、减法指令（SUB）、乘法指令（MPY）、乘加指令（MAC）、乘减指令（MAS）、双数据或者双精度指令（DADD、DSUB）以及特殊操作指令（ABDST、FIRS、SQDST）。其中大部分指令只需要一个指令周期，个别指令需要 2～3 个指令周期。

① 加法指令

语　法	表 达 式	注　释
ADD Smem, src	src = src + Smem	Smen 加到累加器中
ADD Smem, TS, src	src = src + Smem << TS	Smen 移位后加到累加器中
ADD Smem, 16, src [, dst]	dst = src + Smem << 16	Smen 左移 16 位后加到累加器中
ADD Smem [, SHIFT], src [, dst]	dst = src + Smem << SHIFT	Smen 左移 SHIFT 位后加到累加器中
ADD Xmem, SHFT, src	src = src + Xmem << SHFT	Xmen 左移 SHIFT 位后加到累加器中
ADD Xmem, Ymem, dst	dst = Xmem << 16 + Ymem << 16	Xmen 和 Ymen 都左移 16 位后相加，并存到累加器中

ADD # lk [, SHFT], src [, dst]	dst = src + #lk << SHFT	长立即数移动 SHFT 位后加到 src，存放到 dst
ADD # lk, 16, src [, dst]	dst = src + #lk << 16	长立即数左移 16 位后加到 src，存放到 dst
ADD src [, SHIFT] [, dst]	dst = dst + src << SHIFT	src 移动 SHIFT 位后加到 dst
ADD src, ASM [, dst]	dst = dst + src << ASM	src 移动 ASM 位后加到 dst
ADDC Smem, src	src = src + Smem + C	Smen 和进位位加到累加器
ADDM # lk, Smem	Smem = Smem + #lk	把长立即数加到存储器中
ADDS Smem, src	src = src + uns(Smem)	无符号数 Smen 加到累加器

② 减法指令

语 法	表 达 式	注 释
SUB Smem, src	src = src – Smem	从累加器中减去 Smem
SUB Smem, TS, src	src = src – Smem << TS	从累加器中减去 Smem 移动 TS 位后的值
SUB Smem, 16, src [, dst]	dst = src – Smem << 16	从累加器中减去 Smem 移动 16 位后的值
SUB Smem [, SHIFT], src [, dst]	dst = src – Smem << SHIFT	从累加器中减去 Smem 移动 SHIFT 位后的值
SUB Xmem, SHFT, src	src = src – Xmem << SHFT	从累加器中减去 Xmem 移动 SHFT 位后的值
SUB Xmem, Ymem, dst	dst = Xmem << 16 – Ymem << 16	Xmem 和 Ymem 分别左移 16 位，再相减，结果存到累加器中
SUB # lk [, SHFT], src [, dst]	dst = src – #lk << SHFT	从 src 中减去长立即数移动 SHFT 位后的值，结果存到 dst 中
SUB # lk, 16, src [, dst]	dst = src – #lk <<16	从 src 中减去长立即数移动 16 位后的值，结果存到 dst 中
SUB src[, SHIFT] [, dst]	dst = dst – src << SHIFT	从 dst 中减去 src 移动 SHIFT 位后的值
SUB src, ASM [, dst]	dst = dst – src << ASM	从 dst 中减去 src 移动 ASM 位后的值
SUBB Smem, src	src = src – Smem – $\overline{C}$	从累加器中减去 Smem 和进位位的反
SUBC Smem, src	If (src – Smem << 15) ≥ 0 src = (src – Smem << 15) << 1 + 1 Else src = src << 1	条件减法

SUBS Smem, src	src = src – uns(Smem)	从累加器中减去无符号数 Smem

③ 乘法指令

语 法	表达式	注 释
MPY Smem, dst	dst = T * Smem	T 寄存器与 Smen 相乘，结果存放在累加器中
MPYR Smem, dst	dst = rnd(T * Smem)	T 寄存器与 Smen 相乘，结果进行四舍五入运算后，存放在累加器中
MPY Xmem, Ymem, dst	dst = Xmem * Ymem, T = Xmem	两数据存储器操作数 Xmen 和 Ymen 相乘，结果存放在累加器中；同时 Xmen 送到 T 寄存器中

MPY Smem, # lk, dst	dst = Smem * #lk
	T = Smem
		长立即数与 Smem 相乘，结果存放在累加器中；同时 Smen 送到 T 寄存器中

MPY # lk, dst	dst = T * #lk	长立即数与 T 寄存器相乘，结果存放在累加器中
MPYA dst	dst = T * A(32～16)	累加器 A 的高 16 位与 T 寄存器相乘，结果存放在目的累加器中
MPYA Smem	B = Smem * A(32～16) T = Smem	累加器 A 的高 16 位与 Smem 相乘，结果存放在累加器 B 中；同时 Smen 送到 T 寄存器中
MPYU Smem, dst	dst = uns(T) * uns(Smem)	无符号数 T 寄存器与无符号数 Smen 相乘，结果存放在累加器中
SQUR Smem, dst	dst = Smem * Smem T = Smem	Smen 的平方，结果存放在累加器中；同时 Smen 送到 T 寄存器中
SQUR A, dst	dst = A(32～6) * A(32～16)	累加器 A 的高 16 位的平方值，结果存放在目的累加器中

④ 乘加和乘减指令

语 法	表达式	注 释
MAC Smem, src	src = src + T * Smem	T 寄存器与 Smen 相乘，结果加到累加器
MAC Xmem, Ymem, src [, dst]	dst = src + Xmem * Ymem T = Xmem	两数据存储器操作数 Xmen 和 Ymen 相乘，结果加到累加器中；同时 Xmen 送到 T 寄存器中
MAC # lk, src [, dst]	dst = src + T * #lk	长立即数与 T 寄存器相乘，结果加到累加器中
MAC Smem, # lk, src [, dst]	dst = src + Smem * #lk	长立即数与 Smem 相乘，结果加

	T = Smem	到累加器中；同时 Smen 送到 T 寄存器中
MACR Smem, src	src = rnd(src + T * Smem)	T 寄存器与 Smen 相乘，结果加到累加器，再进行四舍五入运算
MACR Xmem, Ymem, src [, dst]	dst = rnd(src + Xmem * Ymem) T = Xmem	两数据存储器操作数 Xmen 和 Ymen 相乘，结果加到累加器中，并执行四舍五入运算；同时 Xmen 送到 T 寄存器中
MACA Smem [, B]	B = B + Smem * A(32～16) T = Smem	累加器 A 的高 16 位与 Smen 相乘，结果加到累加器 B；同时 Smen 送到 T 寄存器中
MACA T, src [, dst]	dst = src + T * A(32～16)	累加器 A 的高 16 位与 T 寄存器相乘，结果加到累加器
MACAR Smem [, B]	B = rnd(B + Smem * A(32～16)) T = Smem	累加器 A 的高 16 位与 Smen 相乘，结果加到累加器 B，并执行四舍五入运算；同时 Smen 送到 T 寄存器中
MACAR T, src [, dst]	dst = rnd(src + T * A(32～16))	累加器 A 的高 16 位与 T 寄存器相乘，结果加到累加器，并执行四舍五入运算
MACD Smem, pmad, src	src = src + Smem * pmad T = Smem (Smem + 1) = Smem	Smen 与 Pmad 相乘，并加到累加器中；同时 Smen 送到 T 寄存器中和下一个 Smen 单元
MACP Smem, pmad, src	src = src + Smem * pmad T = Smem	Smen 与 Pmad 相乘，并加到累加器中；同时 Smen 送到 T 寄存器中
MACSU Xmem, Ymem, src	src = src + uns(Xmem) * Ymem T = Xmem	带符号数 Ymem 与无符号数 Xmem 相乘，结果加到累加器中，同时 Xmen 送到 T 寄存器中
MAS Smem, src	src = src – T * Smem	累加器减去 T 寄存器与 Smen 的乘积
MASR Smem, src	src = rnd(src – T * Smem)	累加器减去 T 寄存器与 Smen 的乘积，并执行四舍五入运算
MAS Xmem, Ymem, src [, dst]	dst = src – Xmem * Ymem T = Xmem	累加器减去 Xmem 与 Ymen 的乘积，同时 Xmen 送到 T 寄存器中
MASR Xmem, Ymem, src [, dst]	dst = rnd(src – Xmem * Ymem) T = Xmem	累加器减去 Xmem 与 Ymen 的乘积，再执行四舍五入运算，同时 Xmen 送到 T 寄存器中
MASA Smem [, B]	B = B – Smem * A(32～16)	累加器 B 减去 Smen 与累加器 A 的高 16 位的乘积，同时 Smen 送

	T = Smem	到 T 寄存器中
MASA T, src [, dst]	dst = src – T * A(32～16)	目的累加器减去 T 寄存器和累加器 A 的高 16 位的乘积
MASAR T, src [, dst]	dst = rnd(src – T * A(32～16))	目的累加器减去 T 寄存器和累加器 A 的高 16 位的乘积，并执行四舍五入运算
SQURA Smem, src	src = src + Smem * Smem T = Smem	Smen 平方后加到累加器，同时 Smen 送到 T 寄存器中
SQURS Smem, src	src = src – Smem * Smem T = Smem	累加器减去 Smen 的平方，同时 Smen 送到 T 寄存器中

⑤ 双操作数指令

语 法	表 达 式	注 释
DADD Lmem, src [, dst]	If C16 = 0 dst = Lmem + src If C16 = 1 dst(39～16) = Lmem(31～16) +src(31～16) dst(15～0) = Lmem(15～0) + src(15～0)	双精度数据 Lmen 和 src 加法；双 16 位数据 Lmen 和 src 加法
DADST Lmem, dst	If C16 = 0 dst = Lmem + (T << 16 + T) If C16 = 1 dst(39～16) = Lmem(31～16) + T dst(15～0) = Lmem(15～0) – T	双精度数据 Lmen 带移位的 T 寄存器加法；双 16 位数据 Lmen 和 T 寄存器的高 16 位数据的加法，低 16 位数据 Lmen 和 T 寄存器的减法
DRSUB Lmem, src	If C16 = 0 src = Lmem – src If C16 = 1 src(39～16) = Lmem(31～16) – src(31～16) src(15～0) = Lmem(15～0) – src(15～0)	双精度数据 Lmen 和 src 的减法；双 16 位数据 Lmen 和 src 的减法
DSADT Lmem, dst	If C16 = 0 dst = Lmem – (T << 16 + T) If C16 = 1 dst(39～16) = Lmem(31～16) – T dst(15～0) = Lmem(15～0) + T	双精度数据 Lmen 和带移位的 T 寄存器的减法；双 16 位数据的高 16 位数据 Lmen 和 T 寄存器的减法，低 16 位数据 Lmen 和 T 寄存器的加法
DSUB Lmem, src	If C16 = 0 src = src – Lmem If C16 = 1 src (39～16) = src(31～16) – Lmem(31～16) src (15～0) = src(15～0) – Lmem(15～0)	双精度数据 src 和 Lmen 的减法；双 16 位数据 src 和 Lmen 的减法
DSUBT Lmem, dst	If C16 = 0 dst = Lmem – (T << 16 + T) If C16 = 1	双精度数据 Lmen 和带移位的 T 寄存器的减法；双 16 位数据的 Lmen 和 T 寄

	dst(39～16) = Lmem(31～16) – T dst(15～0) = Lmem(15～0) – T	存器的减法

⑥ 特殊数学运算的指令

语 法	表 达 式	注 释
ABDST Xmem, Ymem	B = B + |A(32–16)| A = (Xmem – Ymem) << 16	累加器 A 高 16 的绝对值加到累加器 B，同时 Xmem 和 Ymen 相减，右移 16 位送到累加器 A 中
ABS src [, dst]	dst = |src|	累加器的绝对值
CMPL src [, dst]	dst = ~src	求累加器值的反码
DELAY Smem	(Smem + 1) = Smem	存储器延迟操作
EXP src	T = number of sign bits (src)	求累加器的指数
FIRS Xmem, Ymem, pmad	B = B + A * pmad A = (Xmem + Ymem) << 16	对称有限冲激响应滤波器
LMS Xmem, Ymem	B = B + Xmem * Ymem A = A + Xmem << 16 + 2^15	求最小均方值
MAX dst	dst = max(A, B)	求累加器的最大值
MIN dst	dst = min(A, B)	求累加器的最小值
NEG src [, dst]	dst = –src	求累加器的反值
NORM src [, dst]	dst = src << TS dst = norm(src, TS)	归一化运算
POLY Smem	B = Smem << 16 A = rnd(A(32–16) * T + B)	求多项式的值
RND src [, dst]	dst = src + 2^{15}	求累加器的四舍五入值
SAT src	saturate(src)	对累加器的值做饱和计算
SQDST Xmem, Ymem	B = B + A(32–16) * A(32–16) A = (Xmem – Ymem) << 16	求两点之间距离的平方

2 逻辑指令

逻辑指令包括与指令（AND）、或指令（OR）、异或指令（XOR）、移位指令（ROL）以及测试指令（BITF）。根据操作数的不同，这些指令需要 1～2 个指令周期。

① 与指令

语 法	表 达 式	注 释
AND Smem, src	src = src & Smem	Smem 和累加器相与
AND # lk [, SHFT], src [, dst]	dst = src & #lk << SHFT	长立即数移动 SHFT 位后和累加器相与

AND # lk, 16, src [, dst]	dst = src & #lk << 16	长立即数左移 16 位后和累加器相与
AND src [, SHIFT] [, dst]	dst = dst & src << SHIFT	源累加器移 SHIFT 位后和目的累加器相与
ANDM # lk, Smem	Smem = Smem & #lk	Smen 和长立即数相与

② 或指令

语 法	表 达 式	注 释
OR Smem, src	src = src | Smem	Smem 和累加器相或
OR # lk [, SHFT], src [, dst]	dst = src | #lk << SHFT	长立即数移动 SHFT 位后和累加器相或
OR # lk, 16, src [, dst]	dst = src | #lk << 16	长立即数左移 16 位后和累加器相或
OR src [, SHIFT] [, dst]	dst = dst | src << SHIFT	源累加器移 SHIFT 位后和目的累加器相或
ORM # lk, Smem	Smem = Smem | #lk	Smen 和长立即数相或

③ 异或指令

语 法	表 达 式	注 释
XOR Smem, src	src = src ^ Smem	Smem 和累加器相异或
XOR # lk [, SHFT,], src [, dst]	dst = src ^ #lk << SHFT	长立即数移动 SHFT 位后和累加器相异或
XOR # lk, 16, src [, dst]	dst = src ^ #lk << 16	长立即数左移 16 位后和累加器相异或
XOR src [, SHIFT] [, dst]	dst = dst ^ src << SHIFT	源累加器移 SHIFT 位后和目的累加器相异或
XORM # lk, Smem	Smem = Smem ^ #lk	Smen 和长立即数相异或

④ 移位指令

语 法	表 达 式	注 释
ROL src	—	累加器带进位位循环左移
ROLTC src	—	累加器带 TC 位循环左移
ROR src	—	累加器带进位位循环右移
SFTA src, SHIFT [, dst]	dst = src << SHIFT （算术移位）	累加器算术移动 SHIFT 位
SFTC src	if src(31) = src(30) then src = src << 1	累加器条件移位
SFTL src, SHIFT [, dst]	dst = src << SHIFT（逻辑移位）	累加器逻辑移位

⑤ 测试指令

语 法	表 达 式	注 释
BIT Xmem, BITC	TC = Xmem(15 – BITC)	测试指定 BITC 位
BITF Smem, # lk	TC = (Smem && #lk)	测试立即数和 Smen 位
BITT Smem	TC = Smem(15 – T(3–0))	测试由 T 寄存器的低 4 位指定位
CMPM Smem, # lk	TC = (Smem = = #lk)	比较 Smen 和立即数是否相等
CMPR CC, ARx	—	辅助寄存器 ARx 与 AR0 相比较

3 程序控制指令

程序控制指令包括：分支指令（B、BC）、调用指令（CALL）、中断指令（INTR、TRAP）、返回指令（RET）、重复指令（RPT）、堆栈操作指令（FRAME、POPD）以及混合程序控制指令（IDLE、NOP）。这些指令根据情况不同分别需要 1～6 个指令周期。

① 分支指令

语 法	表 达 式	注 释
B[D] pmad	PC = pmad(15～0)	程序指针 PC 无条件转移到 pmad 指向的地址
BACC[D] src	PC = src(15～0)	程序指针 PC 无条件转移到 src 指向的地址
BANZ[D] pmad, Sind	if (Sind ≠ 0) then PC = pmad(15～0)	满足条件程序指针 PC 转移到 pmad 指向的地址
BC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then PC = pmad(15～0)	满足条件程序指针 PC 转移到 pmad 指向的地址
FB[D] extpmad	PC = pmad(15～0) XPC = pmad(22～16)	程序指针 PC 无条件转移到 pmad 的低 16 位指向的地址，同时远程程序指针无条件转移到 pmad 的高 7 位指向的地址
FBACC[D] src	PC = src(15～0) XPC = src(22～16)	程序指针 PC 无条件转移到累加器的低 16 位指向的地址，同时远程程序指针无条件转移到累加器的高 7 位指向的地址

② 调用指令

语 法	表 达 式	注 释
CALA[D] src	– –SP PC + 1[3] = TOS PC = src(15–0)	堆栈指针减 1，可延时的无条件调用累加器所指向的子程序
CALL[D] pmad	– –SP	堆栈指针减 1，可延时的无条件

	PC + 2[4] = TOS PC = pmad(15–0)	调用 pmad 所指向的子程序
CC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then – –SP PC + 2[4] = TOS PC = pmad(15–0)	可延时的有条件调用 pmad 所指向的子程序，堆栈指针减 1
FCALA[D] src	– –SP PC + 1 [3] = TOS PC = src(15–0) XPC = src(22–16)	堆栈指针减 1，可延时的远程无条件调用 src 所指向的子程序
FCALL[D] extpmad	– –SP PC + 2[4] = TOS PC = pmad(15–0) XPC = pmad(22–16)	堆栈指针减 1，可延时的远程无条件调用 pmad 所指向的子程序

③ 中断指令

语 法	表 达 式	注 释
INTR K	– –SP + + PC = TOS PC = IPTR(15～7) + K << 2 INTM = 1	软件中断，程序指针中断到由 K 指定的中断矢量地址，堆栈指针减 1，同时关闭总中断
TRAP K	– –SP + + PC = TOS PC = IPTR(15～7) + K << 2	软件中断，程序指针中断到由 K 指定的中断矢量地址，堆栈指针减 1，但不关闭总中断

④ 返回指令

语 法	表 达 式	注 释
FRET[D]	XPC = TOS ++ SP PC = TOS ++SP	可延时的远程返回，TOS（堆栈顶单元）弹到 XPC，堆栈加 1；TOS 再弹到 PC，堆栈加 1
FRETE[D]	XPC = TOS ++ SP PC = TOS ++SP INTM = 0	可延时的远程返回，TOS（堆栈顶单元）弹到 XPC，堆栈加 1；TOS 再弹到 PC，堆栈加 1；同时打开总中断
RC[D] cond [, cond [, cond]]	if (cond(s)) then PC = TOS, ++SP	可延时的有条件返回，堆栈指针加 1
RET[D]	PC = TOS	可延时的无条件返回，堆栈指针

	++SP	加 1
RETE[D]	PC = TOS ++SP INTM = 0	可延时的无条件返回，堆栈指针加 1，且允许中断
RETF[D]	PC = RTN ++SP INTM = 0	可延时的快速无条件返回，堆栈指针加 1，且允许中断

⑤ 重复指令

语 法	表 达 式	注 释
RPT Smem	RC = Smem	循环执行下一条指令，循环次数为 Smen 的内容
RPT # K	RC = #K	循环执行下一条指令，循环次数为短立即数
RPT # lk	RC = #lk	循环执行下一条指令，循环次数为长立即数
RPTB[D] pmad	RSA = PC + 2[4] REA = pmad BRAF = 1	可延迟的块循环

RPTZ dst, # lk	RC = #lk
	dst = 0
		循环执行下一条指令，循环次数为长立即数，并对累加器清 0

⑥ 堆栈操作指令

语 法	表 达 式	注 释
FRAME K	SP = SP + K	堆栈指针偏移立即数值
POPD Smem	Smem = TOS ++SP	把数据从栈顶弹入到数据存储器，堆栈指针加 1
POPM MMR	MMR = TOS ++SP	把数据从栈顶弹入到存储器映射寄存器，堆栈指针加 1
PSHD Smem	– –SP Smem = TOS	堆栈指针减 1，把数据存储器值压入堆栈
PSHM MMR	– –SP MMR = TOS	堆栈指针减 1，把存储器映射寄存器值压入堆栈

⑦ 其他程序控制指令

语 法	表 达 式	注 释
IDLE K	idle(K)	保持空闲状态直到中断产生
MAR Smem	If CMPT = 0，修改 ARx 的值	修改当前辅助寄存器的值

	If CMPT = 1 and ARx ≠ AR0, then 修改 ARx, ARP = x If CMPT = 1 and ARx = AR0, then 修改 AR(ARP)	
NOP	空操作	程序插入一个等待时间片
RESET	软件复位	软件复位
RSBX N, SBIT	STN (SBIT) = 0	状态寄存器复位
SSBX N, SBIT	STN (SBIT) = 1	状态寄存器复位
XC n , cond [, cond[, cond]]	If (cond(s)) then 执行下面的 n 条指令（n = 1 或 2）	条件执行

4 装入和存储指令

装入和存储指令包括：一般的装入和存储指令（LD、ST）、条件存储指令（CMPS、SACCD）、并行的读取和乘法指令（LD ‖ MAC）、并行的读取和存储指令（ST ‖ LD）、并行的存储和乘法指令（ST ‖ MAC）、并行的读取和加减指令（LD ‖ ADD,LD ‖ SUB）以及其它读取类型和存储类型指令（MVDD、PORTW、READA）。这些指令根据情况不同分别需要 1～5 个指令周期。

① 装入指令

语 法	表 达 式	注 释
DLD Lmem, dst	dst = Lmem	将 Lmen 装入累加器
LD Smem, dst	dst = Smem	将 Smen 装入累加器
LD Smem, TS, dst	dst = Smem << TS	将 Smen 移动 TS 位后装入累加器
LD Smem, 16 , dst	dst = Smem << 16	将 Smen 移动 16 位后装入累加器
LD Smem [, SHIFT], dst	dst = Smem << SHIFT	将 Smen 移动 SHIFT 位后装入累加器
LD Xmem, SHFT, dst	dst = Xmem << SHFT	将 Xmen 移动 SHFT 位后装入累加器
LD # K, dst	dst = #K	将短立即数装入累加器
LD # lk [, SHFT], dst	dst = #lk << SHFT	将短立即数移动 SHFT 位后装入累加器
LD # lk, 16, dst	dst = #lk << 16	将短立即数移动 16 位后装入累加器
LD src, ASM [, dst]	dst = src << ASM	将源累加器移动 ASM 位后装入目的累加器
LD src [, SHIFT], dst	dst = src << SHIFT	将源累加器移动 SHIFT 位后装入目的累加器
LD Smem, T	T = Smem	将 Smen 装入 T 寄存器
LD Smem, DP	DP = Smem(8～0)	将 Smen 的低 9 位装入数据页指针 DP
LD # k9, DP	DP = #k9	将 9 位的短立即数装入数据页指针 DP
LD # k5, ASM	ASM = #k5	将 5 位的短立即数装入 ASM
LD # k3, ARP	ARP = #k3	将 3 位的短立即数装入 ARP

LD Smem, ASM	ASM = Smem(4～0)	将 Smen 的低 5 位装入 ASM
LDM MMR, dst	dst = MMR	将 MMR 装入累加器中
LDR Smem, dst	dst = rnd(Smem)	将 Smen 装入累加器中，并带四舍五入运算
LDU Smem, dst	dst = uns(Smem)	把不带符号的存储器值装入到累加器中
LTD Smem	T = Smem (Smem + 1) = Smem	将 Smen 装入 T 寄存器中，并送入下一个 Smen 单元

② 存储指令

语 法	表 达 式	注 释
DST src, Lmem	Lmem = src	将累加器的值存放到 Lmem 中
ST T, Smem	Smem = T	将 T 寄存器的值存放到 Smem 中
ST TRN, Smem	Smem = TRN	将 TRN 的值存放到 Smem 中
ST # lk, Smem	Smem = #lk	将长立即数存放到 Smem 中
STH src, Smem	Smem = src << –16	将累加器的值移动－16 位后存放到 Smem 中
STH src, ASM, Smem	Smem = src << (ASM – 16)	将累加器的值移动(ASM – 16)位后存放到 Smem 中
STH src, SHFT, Xmem	Xmem = src << (SHFT – 16)	将累加器的值移动位后存放到 Xmem 中
STH src [, SHIFT], Smem	Smem = src << (SHIFT – 16)	将累加器的值移动(SHFT – 16)位后存放到 Smem 中
STL src, Smem	Smem = src	将累加器的值存放到 Smem 中
STL src, ASM, Smem	Smem = src << ASM	将累加器的值移动 ASM 位后存放到 Smem 中
STL src, SHFT, Xmem	Xmem = src << SHFT	将累加器的值移动 SHFT 位后存放到 Xmem 中
STL src [, SHIFT], Smem	Smem = src << SHIFT	将累加器的值移动 SHIFT 位后存放到 Smem 中
STLM src, MMR	MMR = src	将累加器的值存放到 MMR 中
STM # lk, MMR	MMR = #lk	将长立即数的值存放到 MMR 中

③ 条件存储指令

语 法	表 达 式	注 释
CMPS src, Smem	If src(31～16) > src(15～0) then Smem = src(31～16) If src(31～16) ≤ src(15～0) then Smem = src(15～0)	比较，选择并存储最大值
SACCD src, Xmem, cond	If (cond) Xmem = src << (ASM – 16)	有条件存储累加器的值

SRCCD Xmem, cond	If (cond) Xmem = BRC	有条件存储块循环计数器
STRCD Xmem, cond	If (cond) Xmem = T	有条件存储 T 寄存器的值

④ 并行装入和存储指令

语 法	表 达 式	注 释
ST src, Ymem || LD Xmem, dst	Ymem = src << (ASM －16) || dst = Xmem << 16	累加器移动(ASM －16)后装入 Ymen，同时并行 Xmen 移动 16 位后装入目的累加器
ST src, Ymem || LD Xmem, T	Ymem = src << (ASM – 16) || T = Xmem	累加器移动(ASM －16)后装入 Ymen，同时并行 Xmen 装入 T 寄存器

⑤ 并行装入和乘法指令

语 法	表 达 式	注 释
LD Xmem, dst || MAC Ymem, dst_	dst = Xmem << 16 || dst_ = dst_ + T * Ymem	Xmem 移动 16 位后送到累加器，同时并行执行 T 寄存器乘 Ymen，并将乘积加到目的累加器
LD Xmem, dst || MACR Ymem, dst_	dst = Xmem << 16 || dst_ = rnd(dst_ + T * Ymem)	Xmem 移动 16 位后送到累加器，同时并行执行 T 寄存器乘 Ymen，并将乘积加到目的累加器，最后对结果进行四舍五入运算
LD Xmem, dst || MAS Ymem, dst_	dst = Xmem << 16 || dst_ = dst_ – T * Ymem	Xmem 移动 16 位后送到累加器，同时并行执行 T 寄存器乘 Ymen，并从目的累加器中减去该乘积
LD Xmem, dst || MASR Ymem, dst_	dst = Xmem << 16 || dst_ = rnd(dst_ – T * Ymem)	Xmem 移动 16 位后送到累加器，同时并行执行 T 寄存器乘 Ymen，并从目的累加器中减去该乘积，最后对结果进行四舍五入运算

⑥ 并行存储和加减指令

语 法	表 达 式	注 释
ST src, Ymem || ADD Xmem, dst	Ymem = src << (ASM－16) || dst = dst_ + Xmem << 16	Xmen 移动 16 位后加上源累加器，结果存到目的累加器中；同时执行，src 移动(ASM－16)位后送到 Ymen 中
ST src, Ymem || SUB Xmem, dst	Ymem = src << (ASM – 16) || dst = (Xmem << 16) – dst_	Xmen 移动 16 位后减去源累加器，结果存到目的累加器中；同

		时执行，src 移动(ASM－16)位后送到 Ymen 中

⑦ 并行存储和乘法指令

语 法	表 达 式	注 释
ST src, Ymem || MAC Xmem, dst	Ymem = src << (ASM – 16) || dst = dst + T * Xmem	Src 移动(ASM－16)位后送到 Ymen 中，同时并行执行 T 寄存器乘 Ymen，并将乘积加到目的累加器
ST src, Ymem || MACR Xmem, dst	Ymem = src << (ASM – 16) || dst = rnd(dst + T * Xmem)	Src 移动(ASM－16)位后送到 Ymen 中，同时并行执行 T 寄存器乘 Xmen，并将乘积加到目的累加器，最后对结果进行四舍五入运算
ST src, Ymem || MAS Xmem, dst	Ymem = src << (ASM – 16) || dst = dst – T * Xmem	Src 移动(ASM－16)位后送到 Ymen 中，同时并行从累加器中减去 T 寄存器和 Xmen 的乘积
ST src, Ymem || MASR Xmem, dst	Ymem = src << (ASM – 16) || dst = rnd(dst – T * Xmem)	Src 移动(ASM－16)位后送到 Ymen 中，同时并行从累加器中减去 T 寄存器和 Xmen 的乘积，最后对结果进行四舍五入运算
ST src, Ymem || MPY Xmem, dst	Ymem = src << (ASM – 16) || dst = T * Xmem	Src 移动(ASM－16)位后送到 Ymen 中，同时将 T 寄存器和 Xmen 的乘积送到累加器中

⑧ 其它装入和存储指令

语 法	表 达 式	注 释
MVDD Xmem, Ymem	Ymem = Xmem	将 Xmen 送到 Ymen
MVDK Smem, dmad	dmad = Smem	将 Smen 送到 dmad
MVDM dmad, MMR	MMR = dmad	将 dmad 送到 MMR
MVDP Smem, pmad	pmad = Smem	将 Smen 送到 pmad
MVKD dmad, Smem	Smem = dmad	将 dmad 送到 Smen
MVMD MMR, dmad	dmad = MMR	将 MMR 送到 dmad
MVMM MMRx, MMRy	MMRy = MMRx	将 MMRx 送到 MMRy
MVPD pmad, Smem	Smem = pmad	将 pmad 送到 Smen
PORTR PA, Smem	Smem = PA	从 PA 端口把数据读到 Smen
PORTW Smem, PA	PA = Smem	将 Smen 写到 PA 端口
READA Smem	Smem = A	把由累加器 A 寻址的程序存储器单元的值读到数据单元中去

WRITA　Smem	A = Smem	把数据单元中的值写到由累加器 A 寻址的程序存储器中

5　单个循环指令

C54xx 系列 DSP 有一些单个循环指令，它们将引起下一指令被重复执行。重复执行的次数由单个循环指令中的一个操作数决定，并等于该操作数的内容加 1 后的值，且该操作数的值被存储在 16 位的重复计数寄存器（RC）中。RC 的值只能由单个循环指令中的操作数决定，其最大值是 65536。当下一条指令被重复执行时，绝对程序或数据地址将自动加 1。当重复指令被解码时，所有中断（除复位中断外）均被屏蔽，直到下一条指令被重复执行完毕。重复的功能体现在如乘加或块移动指令这些指令中，也可以用来增加指令的执行速度。下列指令是单个循环指令：

名 称	说　明
FIRS	有限冲击响应滤波器
MACD	累加器的值进行乘运算和移动（带延时）
MACP	累加器的值进行乘运算和移动（不带延时）
MVDK	数据空间到数据空间的移动
MVDM	数据空间到 MMR 寄存器的移动
MVDP	数据空间到程序空间的移动
MVKD	数据空间到数据空间的移动
MVMD	MMR 寄存器到数据空间的移动
MVPD	程序空间到数据空间的移动
READA	从程序空间读到数据空间
WRITA	写数据空间到程序空间

对单个数据存储器操作数指令而言，如果有一个长的偏移地址或绝对地址，指令不可被循环执行。下面的指令就不能使用 RPT 或 RPTZ 循环指令来执行。

符 号	说　明
ADDM	长立即数加到数据存储器
ANDM	数据存储器与长立即数相与
B[D]	无条件跳转
BACC[D]	跳转到累加器地址
BANZ[D]	跳转到非 0 的辅助寄存器
BC[D]	条件转移
CALA[D]	调用累加器地址
CALL[D]	无条件调用
CC[D]	条件调用
CMPR	和辅助寄存器相比较

DST	长字（32 位）存储
FB[D]	无条件远程跳转
FBACC[D]	远程跳转至由累加器所指定的位置
FCALA[D]	远程调用子循环，地址由累加器指定
FCALL[D]	无条件远程调用
FRET[D]	远程返回
FRETE[D]	中断使能并从中断中远程返回
IDLE	待机指令
INTR	中断
LD　ARP	调用辅助寄存器指针（ARP）
LD　DP	调用数据页指针（DP）
MVMM	MMR 之间的移动
ORM	数据存储器与长立即数相与
RC[D]	条件返回
RESET	软件复位
RET[D]	无条件返回
RETF[D]	中断返回
RND	累加器求余
RPT	重复执行下一条指令
RPTB[D]	块重复
RPTZ	重复下一条指令并清除累加器
RSBX	状态寄存器中的位清 0
SSBX	状态寄存器中的位置 1
TRAP	软件中断
XC	条件执行
XORM	长立即数和数据存储器相异或

逐条指令的详细说明如下：

1. ABDST　Xmem，Ymem

操作数　Xmem，Ymem： 双数据存储操作数

执行　　(B) +| (A(32～16))|→ B

((Xmem) — (Ymem)) << 16 → A

状态位　被 OVM、 FRCT 以及 SXM 位影响

影响 C、 OVA 以及 OVB 位

说明

计算两变量 Xmem 和 Ymem 的差。同时将累加器 A 的高端（位 32～16）的绝对值加

到累加器 B 中。 Xmem 减去 Ymem 的差左移 16 位然后存到累加器 A 中。如果分数方式位（FRCT 位）为 1，该绝对值将自动乘 2。

例 ABDST *AR3+， *AR4+

解释

计算 55h－AAh＝FFABh，将 FFABh 送到累加器 A 的高端，并做符号扩展；同时计算累加器 A 高端的绝对值|ABCDh|=5433h，将结果加到累加器 B 的低端。

2. ABS src [，dst]

操作数 src，dst： A（累加器 A） B（累加器 B）

执行 | (src)|→ dst

状态位 OVM 位按照以下方式影响该指令运行结果：

如果 OVM = 1，80 0000 0000h 的绝对值是 00 7FFF FFFFh

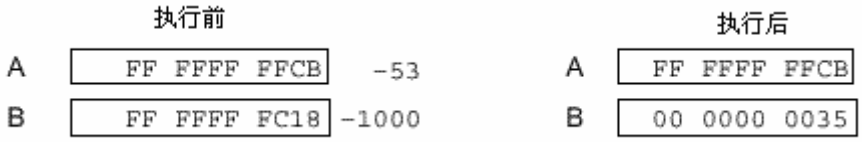
如果 OVM = 0，80 0000 0000h 的绝对值是 80 0000 0000h

影响 C 以及 OVdst 位（如果 dst = src，影响 OVsrc 位）

说明

计算 src 的绝对值，然后存在 dst 中，如果没有定义 dst，结果直接存在 src 中。

例 ABS A， B



3. ADD

（1） ADD Smem， src

（2） ADD Smem，TS，src

（3） ADD Smem，16，src [，dst]

（4） ADD Smem [，SHIFT]，src [，dst]

（5） ADD Xmem，SHFT，src

（6） ADD Xmem，Ymem，dst

（7） ADD # lk [，SHFT]，src [，dst]

（8） ADD # lk，16，src [，dst]

（9） ADD src [，SHIFT]，[，dst]

（10）ADD　 src，ASM [，dst]

操作数

–32 768 ≤ lk ≤ 32 767

–16 ≤ SHIFT ≤ 16

0 ≤ SHFT ≤ 15

执行（1）(Smem) + (src)→src

（2）(Smem) << (TS) + (src)→src

（3）(Smem) << 16 + (src)→dst

（4）(Smem) [<< SHIFT] + (src)→dst

（5）(Xmem) << SHFT + (src) →src

（6）((Xmem) + (Ymem)) << 16→dst

（7）lk << SHFT + (src) →dst

（8）lk << 16 + (src) →dst

（9）(src or [dst]) + (src) << SHIFT→dst

（10）(src or [dst]) + (src) << ASM→dst

状态位　 被 SXM 位和 OVM 位影响

影响 C 位和 OVdst 位 (如果 dst = src，影响 OVsrc 位)

说明

把一个 16 位的数加到选定的累加器中，或一个采用双数据存储器操作数寻址的操作数 Xmen。这个 16 位的数可以为以下情况之一：

（1） 单数据存储器操作数（Smem）；

（2） 双数据存储器操作数（Xmem 和 Ymem）；

（3） 一个 16 位的立即操作数（#lk）；

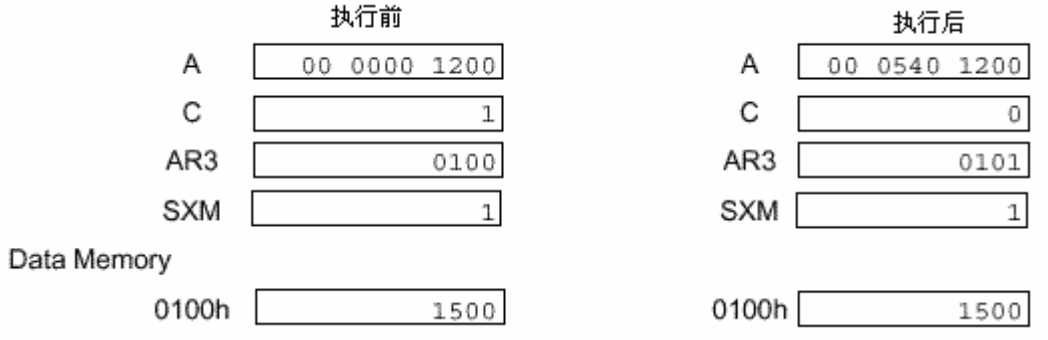
（4） 累加器中的移位后得到的数。

如果定义了 dst，结果就存在 dst 中；否则结果存在 src 中。大部分第二操作数需要移位。左移时低位必定添 0；右移时高位的符号扩展有两种情况，分别为：

（1）如果 SXM=0 进行符号扩展；

（2）如果 SXM=1 不进行符号扩展，直接添０。

例　 ADD　*AR3+，　14，　A



解释

将 AR3 指向的地址单元 0100h 的内容 1500h 左移 14 位，得到 5400000h 加上 1200h，将结果 5401200h 存在累加器 A 中。

4. ADDC　 Smem，src

操作数　　Smem： 单数据存储操作数

src： A（累加器 A） B（累加器 B）

执行 (Smem) + (src) + (C) → src

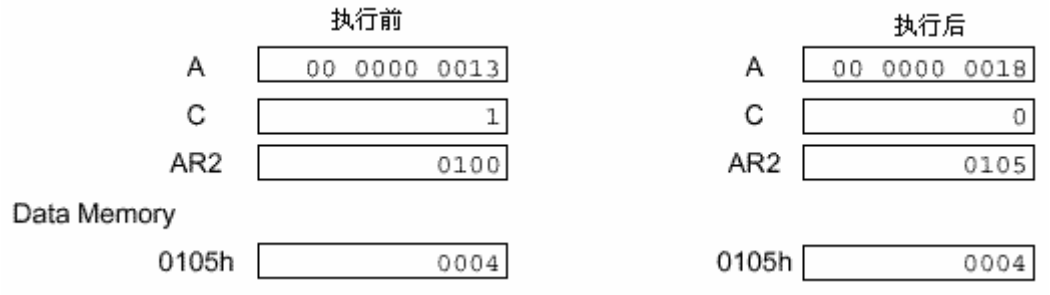
状态位 被 OVM 位和 C 位影响

影响 C 位和 OVsrc 位

说明

将 16 位单数据存储器操作数 Smem 和进位位(C)的值加到 src 中。无论 SXM 位的值是多少都不进行符号扩展。

例 ADDC *+AR2(5)， A



解释

将 AR2 加上 5 后指向的地址单元内容，并进位加到累加器 A 中，此指令计算 0013h＋0004h＋1h＝0018h。

5. ADDM #lk， Smem

操作数 Smem： 单数据存储操作数

–32 768 ≤ lk ≤ 32 767

执行 #lk + (Smem) → Smem

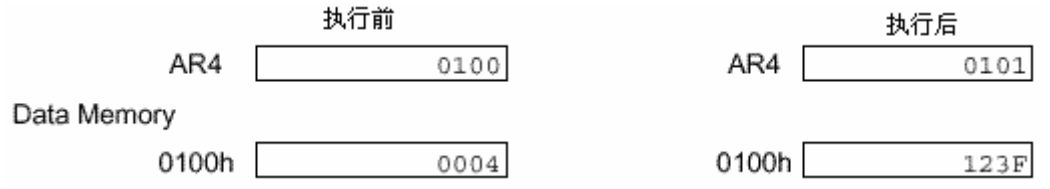
状态位 被 OVM 位和 SXM 位影响

影响 C 位和 OVA 位

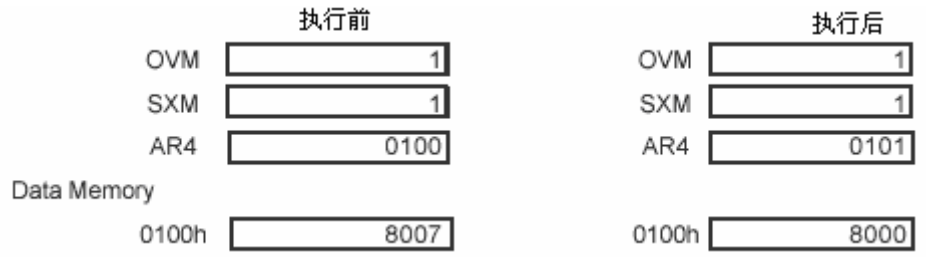
说明

16 位单数据存储器操作数 Smem 与 16 位立即数 lk 相加，结果存在 Smem 中。该指令不能循环执行。

例 ADDM 0123Bh， *AR4+



例 ADDM 0FFF8h， *AR4+



解释

指令运行前 OVM 和 SXM 都为 1，对运行结果做溢出保护和符号扩展。立即数 0FFF8h＋8007h＝17FFFh，由于做符号扩展，这两个数都是负数，结果超过 16 位范围，做溢出保护，设置成负的最大值，所以最终结果为 8000h。如果将 SXM 改为 0，最终结果为 7FFFh。

6. ADDS Smem， src

操作数 Smem： 单数据存储操作数

src： A (累加器 A) B (累加器 B)

执行 uns(Smem) + (src)→src

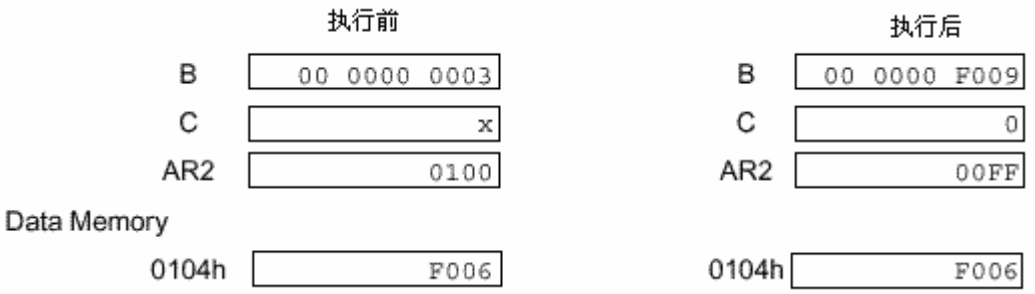
状态位 被 OVM 位影响

影响 C 位和 OVsrc 位

说明

把无符号的 16 位单数据存储器操作数 Smem 加到 src 中。无论 SXM 为何值都不进行符号扩展。

例 ADDS *AR2–， B



解释

此指令运行后进位位 C 变为 0，表示该指令运行后未产生进位。

7. AND

（1） AND Smem，src

（2） AND # lk [，SHFT]，src [，dst]

（3） AND # lk，16，src [，dst]

（4） AND src [，SHIFT]，[，dst]

操作数 Smem： 单数据存储操作数

src： A (累加器 A) B (累加器 B)

–16 ≤ SHIFT ≤ 15

0 ≤ SHFT ≤ 15

0 ≤ lk ≤ 65 535

执行

（1） (Smem) AND (src) →src

（2） lk << SHFT AND (src)→ dst

（3） lk << 16 AND (src)→ dst

（4） (dst) AND (src) << SHIFT → dst

状态位 无

说明

该指令功能是使以下几种数据和 src 相与：

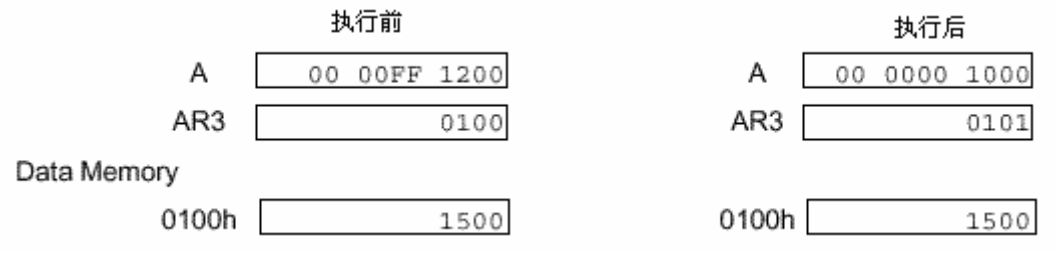
（1） 一个 16 位单数据存储器操作数 Smem；

（2） 一个 16 位立即数 lk

（3） 源或目的累加器（src 和 dst）

如果确定了移位，操作数先进行移位，然后再进行与操作。如果左移，低位添 0，高位不进行符号扩展；如果右移，高位添 0。

例 AND *AR3+， A



8. ANDM # lk， Smem

操作数 Smem： 单数据存储操作数

0 ≤ lk ≤ 65 535

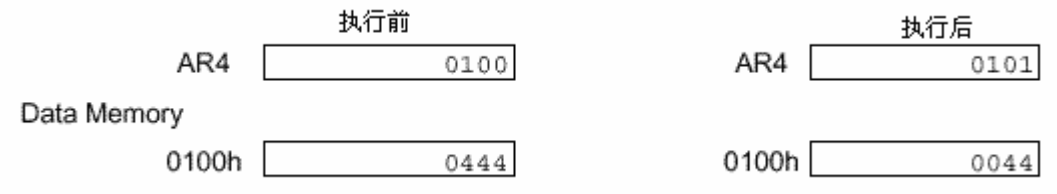
执行 lk AND (Smem) → Smem

状态位 无

说明

16 位单数据存储器操作数 Smem 和一个 16 位长立即数 lk 相与，结果存在 Smem 所在单元中。该指令不可循环执行。

例 ANDM #00FFh， *AR4+



9. B[D] pmad

操作数 0 ≤ pmad ≤ 65 535

执行 pmad → PC

状态位 无

说明

程序指针指向指定的程序存储器地址（pmad），该地址可以是符号或一个数字。如果是延迟转移（指令带有后缀 D），紧接着该指令的两条单字指令或一条双字指令从程序存储器中取出先执行。该指令不能循环执行。

例 BD 1000h

10. BACC[D] src

操作数 src： A (累加器 A) B (累加器 B)

执行　　(src(15～0))→PC
状态位　　无
说明

程序指针指向 src 的低 16 位所确定的地址。如果是延迟转移（指令带有后缀 D），紧跟着该指令的两条单字或一条双字指令从程序存储器中取出先执行。该指令不能循环执行。

例　BACC　A

11. BANZ[D] pmad， Sind

操作数　　Sind： 单间接寻址操作数

0 ≤ pmad ≤ 65 535

执行　　If ((ARx) ≠ 0)

Then

pmad → PC

Else

(PC) + 2 → PC

状态位　　无

说明

如果当前的辅助寄存器 ARx 不为 0，程序指针转移到指定的地址（pmad），否则程序指针加 2。如果是延迟转移（指令带有后缀 D），紧接着该指令的两条单字指令或一条双字指令从程序存储器中取出先执行。该指令不能循环执行。

例　BANZ　2000h，　*AR3–

12. BC[D]　　pmad，　cond [，　cond [，　cond]]

操作数　0 ≤ pmad ≤ 65 535

执行　　If (cond(s))

Then　　pmad → PC

Else　　(PC) + 2 → PC

状态位　影响 OVA 位或 OVB 位

表 3　指令的条件代码所对应的条件

条　件	说　明	条　件	说　明
BIO	BIO 为低	NBIO	BIO 为高
C	C=1	NC	C=0
TC	TC=1	NTC	TC=0
AEQ	（A）= 0	BEQ	（B）= 0

ANEQ	（A）≠0	BNEQ	（B）≠0
AGT	（A）> 0	BGT	（B）> 0
AGEQ	(A) ≥0	BGEQ	(B) ≥ 0
ALT	(A) < 0	BLT	(B) < 0
ALEQ	(A) ≤0	BLEQ	(B) ≤ 0
ALV	A 溢出	BOV	B 溢出
ANOV	A 没有溢出	BNOV	B 没有溢出
UNC	无条件		

说明

如果满足特定的条件，程序指针转移到程序存储器地址（pmad）上。如果是延迟转移（指令带有后缀 D），紧接着该指令的两条单字指令或一条双字指令从程序存储器中取出先执行。但如果条件满足，那么这两个字将从流水中冲掉，程序从 pmad 开始执行；如果条件不满足，PC 加 2 且紧接着该指令的两个字继续执行。

程序指针在改变之前可对多个条件进行测试。指令的条件代码见附录表 1。指令可测试相互独立的条件或者关联的条件；但多个条件只能出自同一组的不同类中。条件的分组和分类见附录表 2。

例 BC 2000h，AGT

表 4 指令的条件代码的分组和分类

第一组		第二组		
A 类	B 类	A 类	B 类	C 类
EQ NEQ LT LEQ GT GEQ	OV NOV	TC NTC	C NC	BIO NBIO

13. BIT Xmem， BITC

操作数 Xmem： 双数据存储操作数

0 ≤ BITC ≤ 15

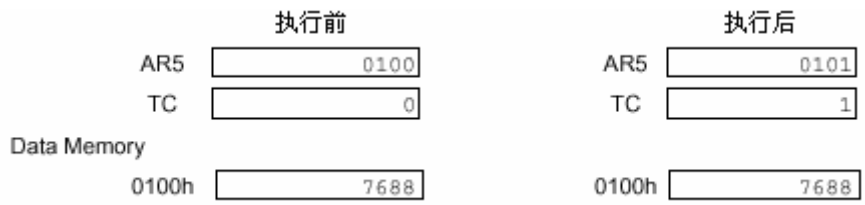
执行 (Xmem(15 – BITC)) → TC

状态位 影响 TC 位

说明

把双数据存储器操作数 Xmem 的指定位复制到状态寄存器的 ST0 的 TC 位。

例 BIT *AR5+， 15-12



解释

指令中的（15－12）表示测试第 12 位。

14. BITF Smem， #lk

操作数 Smem： 单数据存储操作数

0 ≤ lk ≤ 65 535

执行 If ((Smem) AND lk) = 0

Then

0 → TC

Else

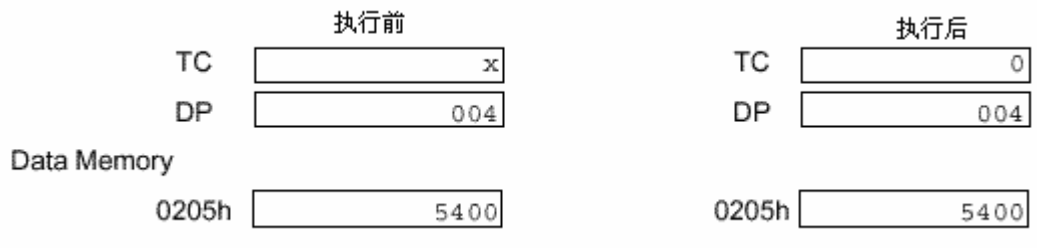
1 → TC

状态位 影响 TC 位

说明

测试单数据存储器值 Smem 中指定的某些位，假如指定的一位或多位为 0，状态寄存器 ST0 的 TC 位清 0，否则该位置 1。lk 常数在测试一位或多位时起屏蔽作用。

例 BITF 5， 00FFh



解释

页指针 DP 的值为 4，表示第四页地址单元。指令中的数值 5 表示第四页的第 5 个地址单元，即 205h。

15. BITT Smem

操作数 Smem： 单数据存储操作数

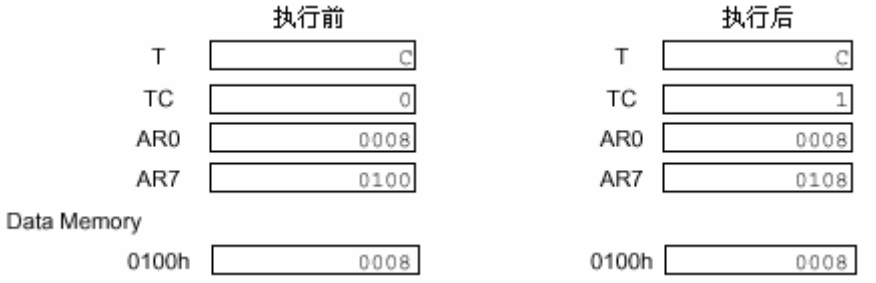
执行 (Smem (15 – T(3～0)))→TC

状态位 影响 TC 位

说明

把单数据存储器 Smem 的指定位复制到 ST0 的 TC 位，T 寄存器的低四位(位 3～0)值确定了被复制的位代码，高 12 位对应位地址。

例 BITT *AR7＋0



16. CALA[D] src

操作数 src： A (累加器 A) B (累加器 B)

执行　　非延时

(SP) – 1→SP

(PC) + 1→TOS

(src(15～0)) →PC

延时

(SP) – 1→SP

(PC) + 3→TOS

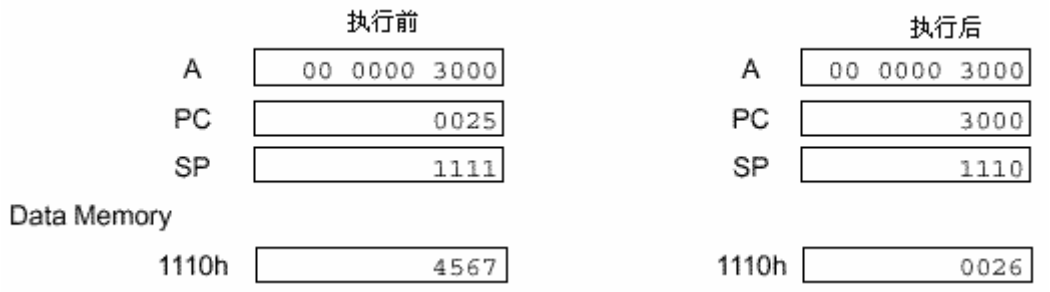
(src(15～0)) →PC

状态位　　无

说明

程序指针转移到 src 的低位所确定的地址单元，返回地址压入栈顶。如果是延迟调用，紧接着该指令的两条单字指令或一条双字指令从程序存储器中取出来先执行。该指令不可循环执行。

例　CALA　A



17. CALL[D] pmad

操作数　　0 ≤ pmad ≤ 65 535

执行　　非延时

(SP) － 1 → SP

(PC) + 2 → TOS

pmad → PC

延时

(SP) － 1 → SP

(PC) + 4 → TOS

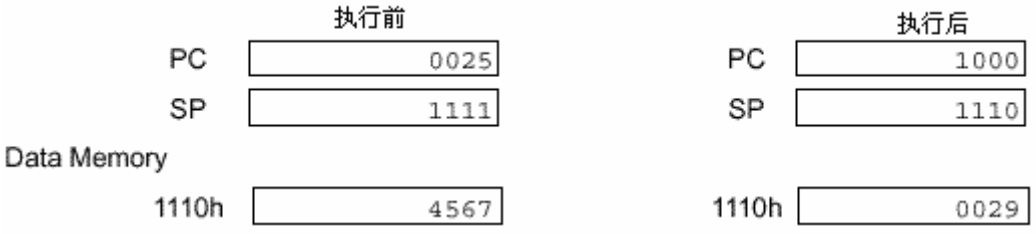
pmad → PC

状态位　无

说明

程序指针指向确定的程序存储器地址（pmad），在 pmad 装入 PC 之前返回地址压入栈顶保存。如果是延迟调用，紧接着该指令的两条单字指令或一条双字指令从程序存储器中取出来先执行。该指令不能循环执行。

例　　CALLD　1000h



18. CC[D] pmad， cond [， cond [， cond]]

操作数 0 ≤ pmad ≤ 65 535

执行 非延时

If (cond(s))

Then (SP) － 1 → SP

(PC) + 2 → TOS

pmad → PC

Else

(PC) + 2 → PC

延时

If (cond(s))

Then (SP) － 1 → SP

(PC) + 4 → TOS

pmad → PC

Else

(PC) + 2 → PC

状态位 影响 OVA 位或 OVB 位

说明

当满足确定的条件时，程序指针指向程序存储器地址(pmad)，实现程序跳转；如果不满足条件时，程序指针加 2。如果是延迟调用，该指令的后两个字指令先取出执行，且不会影响被测试的条件。该指令的判断条件和 BC[D]指令的使用完全一样。

例 CC 2222h， AGT

19. CMPL src [， dst]

操作数 src， dst： A (累加器 A) B (累加器 B)

执行 ($\overline{\text{src}}$) → dst

状态位 无

说明

计算 src 的反码（逻辑反），结果存放在 dst 中；若没有指定 dst，则存放在 src 中。

例 CMPL A， B

20. CMPM Smem， # lk

操作数 Smem： 单数据存储操作数

–32 768 ≤ lk ≤ 32 767

执行 If (Smem) = lk

Then 1 → TC

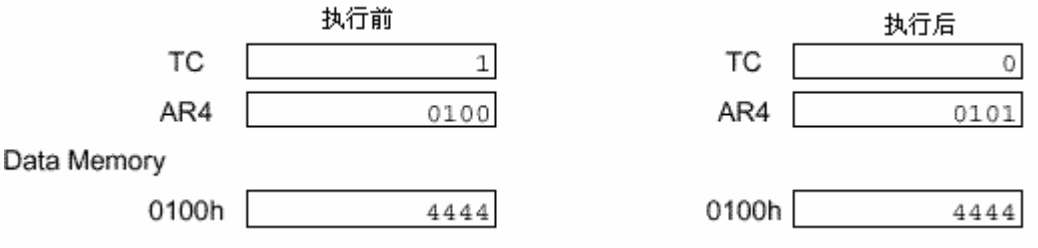
Else 0 → TC

状态位 影响 TC 位

说明

比较 16 位单数据存储器操作数 Smem 和 16 位常数 lk 是否相等。若相等，ST0 寄存器的 TC 位置 1，否则该位清 0。

例 CMPM *AR4+， 0404h



21. CMPR CC， ARx

操作数 0 ≤ CC ≤ 3

ARx： AR0～AR7

执行 If (cond)

Then 1 → TC

Else 0 → TC

状态位 影响 TC 位

说明

比较指定的辅助寄存器（ARx）和 AR0 的值，然后根据结果决定 TC 的值。比较的方式由条件代码 CC 的值决定，见附录表 3。若条件满足 ST0 寄存器的 TC 位置 1，否则该位清 0。所有的条件都以无符号数运算。

表 5 条件代码的说明

条 件	条 件 代 码（CC）	说 明
EQ	00	测试 ARx 是否等于 AR0
LT	01	测试 ARx 是否小于 AR0
GT	10	测试 ARx 是否大于 AR0
NEQ	11	测试 ARx 是否不等于 AR0

例　CMPR　2，　AR4

22.　CMPS　　src，　Smem

操作数　　src：　A (累加器 A)　　B (累加器 B)

Smem：　单数据存储操作数

执行　　If ((src(31～16)) > (src(15～0)))

Then　　(src(31～16))→Smem

(TRN) << 1→TRN

0→TRN(0)

0→TC

Else　　(src(15～0))→Smem

(TRN) << 1→TRN

1→TRN(0)

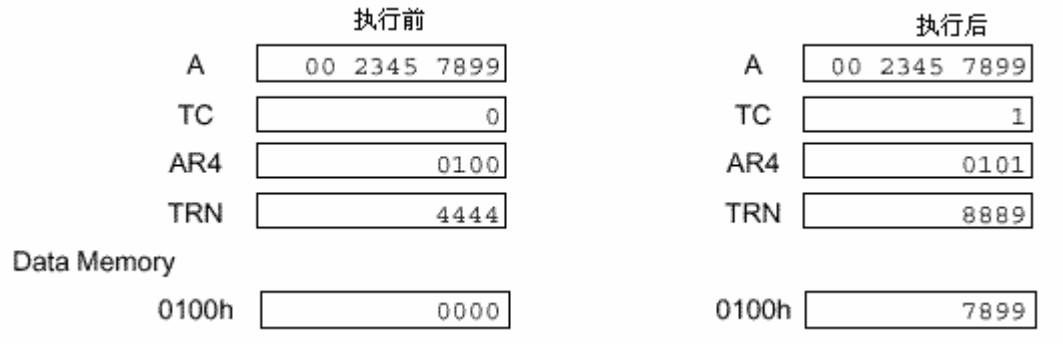
1→TC

状态位　　影响 TC 位

说明

比较位于累加器的高端和低端的两个 16 位二进制补码值的大小，把较大值存在单数据存储器单元 Smem 中。如果是累加器的高端（31～16 位）较大，暂时寄存器（TRN）左移一位，最低位添 0；TC 位清 0。反之，若累加器的低端（15～0 位）较大，暂时寄存器左移一位，最低位添 1；TC 位置 1。该指令不遵从标准的流水线操作。比较是在读操作数阶段完成。因而，累加器的值是指令执行前一个阶段的值。暂时寄存器和 TC 位是在指令执行阶段被修改。

例 CMPS　A，　*AR4+



解释

累加器 A 的低 16 位 7899h 大于高 16 位 2345h，所以将低 16 位 7899h 送到 AR4 指向的地址单元 100h 中。同时，TRN 寄存器的值左移一位并加 1，相当于 4444h×2＋1＝8889h，最后将 TC 位置 1。

23.　DADD　Lmem，　src [，　dst]

操作数　Lmem：　长数据存储操作数

src， dst： A (累加器 A)　B (累加器 B)

执行　If C16 = 0

Then　(Lmem) + (src) → dst

Else　(Lmem(31～16)) + (src(31～16)) → dst(39～16)

(Lmem(15～0)) + (src(15～0)) → dst(15～0)

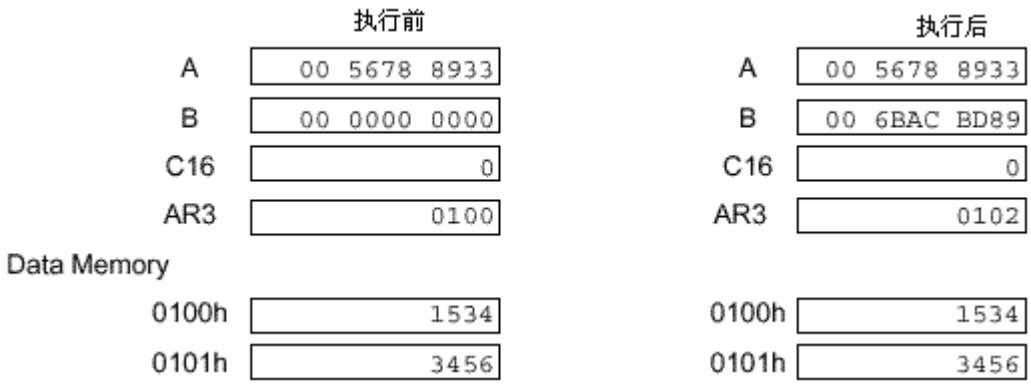
状态位　只有在 C16=0 时被 SXM 位和 OVM 位影响

影响 C 位和 OVdst 位

说明

源累加器的内容加到 32 位长数据存储器操作数 Lmem 中。如果定义了目的累加器，结果存在目的累加器中，否则存在源累加器中。C16 的值决定了指令执行的方式，当 C16=0 时，指令以双精度方式执行，40 位源累加器的值加到 Lmem 中，饱和位和溢出位根据运算的结果变化。当 C16=1 时，指令以双 16 位方式执行，SRC 的高端（位 31～16）与 Lmem 的高 16 位相加；SRC 的低端（位 15～0）与 Lmem 的低 16 位相加。饱和位和溢出位在此方式下不改变，无论 OVM 的状态是什么，结果不进行饱和运算。

例　DADD　*AR3+， A， B



解释

AR3 指向的地址单元 100h 和下一个地址单元 101h 组成一个 32 位的数据，将此数据加上累加器 A，结果存在累加器 B 中。实际运行为，15343456h＋56788933h＝6BACBD89h。由于是长数据操作，AR3 执行后增加 2。

24. DADST　Lmem， dst

操作数　Lmem： 长数据存储操作数

dst： A (累加器 A)　B (累加器 B)

执行　If　C16 = 1

Then　(Lmem(31～16)) + (T) → dst(39～16)

(Lmem(15～0)) － (T) →dst(15～0)

Else　(Lmem) + ((T) + (T) << 16) → dst

状态位　只有在 C16=0 时被 SXM 位和 OVM 位影响

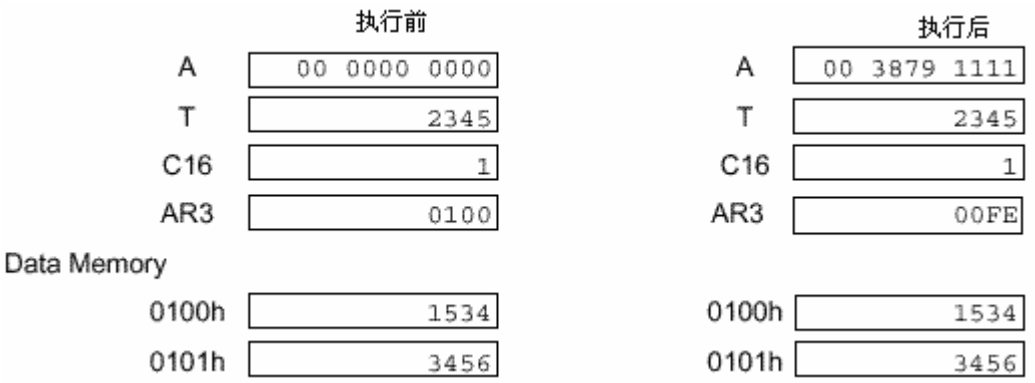
影响 C 位和 OVdst 位

说明

该指令把 T 寄存器的值加到 32 位长数据存储器操作数 Lmem 中。C16 的值决定了指令执行的方式。当 C16=0 时，指令以双精度方式执行，T 寄存器的值和其左移 16 位得到的值相结合所组成的 32 位数据加到 Lmem 中，结果存在目的累加器中；当 C16=1 时，指令以双 16 位方式执行，Lmem 的高 16 位与 T 寄存器的值相加，存在目的累加器的前 24

位，同时从 Lmem 的低 16 位减去 T 寄存器的值，结果存在目的累加器的低 16 位。无论 OVM 位的状态是什么，结果都不进行饱和运算。注意：该指令仅当 C16 置 1 时（双 16 位方式）才有意义。

例　DADST　*AR3–，　A



解释

AR3 指向的地址单元 100h 和下一个地址单元 101h 组成一个 32 位的数据，将此数据的高 16 位加上 T 寄存器的值，结果存在累加器 A 的高 16 位。实际运行为，1534h＋2345h＝3879h；然后再将此数据的低 16 位减去 T 寄存器的值，结果存在累加器 A 的高 16 位。实际运行为，3456h－2345h＝1111h。由于是长数据操作，AR3 执行后增加 2。

25. DELAY　Smem

操作数　　Smem： 单数据存储操作数

执行　　(Smem) → Smem + 1

状态位　　无

说明

该指令把单数据存储器单元 Smem 的内容复制到紧接着的较高地址单元中去。数据复制完后，原单元内容保持不变。该功能在数字信号处理应用中实现一个 Z 延迟是相当有用的。这种延迟操作数在 LTD 和 MACD 等指令中都可以见到。

例　DELAY　*AR3

26. DLD　Lmem，　dst

操作数　　Lmem： 长数据存储操作数

dst： A (累加器 A)　B (累加器 B)

执行　　If　C16=0

Then　(Lmem) → dst

Else　(Lmem(31～16)) → dst(39～16)

(Lmem(15～0)) → dst(15～0)

状态位　　被 SXM 位影响

说明

该指令是把一个 32 位的长操作数 Lmem 装入目的累加器 dst 中。C16 的值决定了指令执行的方式，当 C16=0 时，指令以双精度方式执行，Lmem 装入到目的累加器 dst 中；当 C16=1 时，指令以双 16 位方式执行，Lmem 的高 16 位装入目的累加器 dst 的高 24 位，同时 Lmem 的低端装入目的累加器的低端。

例　DLD　*AR3+，　B

27. DRSUB　Lmem，　src

操作数　　Lmem：　长数据存储操作数

src：　A (累加器　A)　　B (累加器　B)

执行　　If　C16 = 0

Then　　(Lmem)　－　(src) → src

Else　　(Lmem(31～16))－(src(31～16)) → src(39～16)

(Lmem(15～0))－(src(15～0)) → src(15～0)

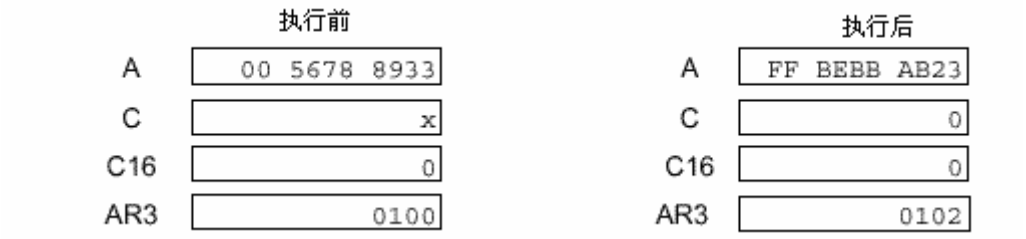
状态位　　被 SXM 位和 OVM 位影响

影响 C 位和 OVsrc 位

说明

该指令是把一个 32 位的长操作数 Lmem 减去 src 的内容，结果存在 src 中。C16 的值决定了指令执行的方式。当 C16=0 时，指令以双精度方式执行；当 C16=1 时，指令以双 16 位方式执行，Lmem 的高 16 位减去 src 的高端，结果存放在 src 的高 24 位，同时 Lmem 的低端减去 src 的低端，结果存放在 src 的低端。在这种方式下，不管 OVM 的状态如何，都不进行饱和运算。

例　DRSUB　*AR3+，　A



28. DSADT　Lmem，　dst

操作数　　Lmem：　长数据存储操作数

dst：　A (累加器　A)　　B (累加器　B)

执行　　If　C16 = 1

Then (Lmem(31～16)) － (T) → dst(39～16)
(Lmem(15～0)) － (T) → dst(15～0)
Else (Lmem) – ((T) + (T << 16)) → dst

状态位 只有 C16 = 0 时被 SXM 位和 OVM 位影响
影响 C 位和 OVdst 位

说明

该指令是把一个 32 位的长操作数 Lmem 减去 T 寄存器的值，结果装入 dst 中。C16 的值决定了指令执行的方式。当 C16=0 时，指令以双精度方式执行，T 寄存器的值与其左移 16 位后得到的值连在一起组成的 32 位的数与 Lmem 相减，结果存在 dst 中；当 C16=1 时，指令以双 16 位方式执行，Lmem 的高 16 位减去 T 寄存器的值，结果存在 dst 的高端，同时把 T 寄存器的值加到 Lmem 的低端，结果存在 dst 的低端。在这种方式下，不管 OVM 的状态如何，都不进行饱和运算。

例 DSADT *AR3+， A

	执行前	执行后
A	00 0000 0000	FF F1EF 1111
T	2345	2345
C	0	0
C16	0	0
AR3	0100	0102
Data Memory		
0100h	1534	1534
0101h	3456	3456

解释

C16 为 0，首先左移 T 寄存器 16 位，然后和 T 寄存器组成一个 32 位的数据，为 23452345h。将 AR3 指向的 32 位数 15343456h－23452345h＝F1EF1111h，做符号扩展后存在累加器 A 中。

29. DST src， Lmem

操作数 src： A (累加器 A) B (累加器 B)
Lmem： 长数据存储操作数

执行 (src(31～0)) →Lmem

说明

把源累加器的内容存放在一个 32 位的长数据储存器单元中 Lmem 中。

例 DST B， *AR3+

	执行前	执行后
B	00 6CAC BD90	00 6CAC BD90
AR3	0100	0102
Data Memory		
0100h	0000	6CAC
0101h	0000	BD90

30. DSUB Lmem， src

操作数　　Lmem： 长数据存储操作数

src： A (累加器 A)　 B (累加器 B)

执行　　If　C16 = 0

Then　　(src) － (Lmem) →src

Else　　(src(31～16)) － (Lmem(31～16)) → src(39～16)

(src(15～0))－ (Lmem(15～0)) → src(15～0)

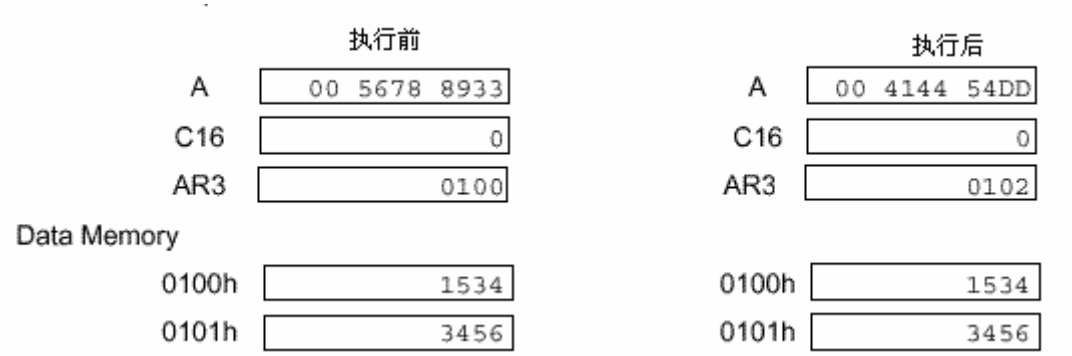
状态位　只有在 C16 = 0 时被 SXM 位和 OVM 位影响

影响 C 位和 OVsrc 位

说明

该指令是从源累加器中减去 32 位长数据存储器操作数 Lmem 的值，结果存在 src 中，C16 的值决定了指令执行的方式。当 C16=0 时，指令以双精度方式执行；当 C16=1 时，指令以双 16 位方式执行，从源累加器 src 的高端减去 Lmem 的高端，结果装入 src 的高 24 位，同时从源累加器 src 的低端减去 Lmem 的低端，结果装入 src 的低端。

例　DSUB　*AR3+，　A



解释

C16 为 0，执行 56788933h－15343456h＝414454DDh。

31. DSUBT　Lmem，　dst

操作数　Lmem： 长数据存储操作数

dst： A (累加器 A)　 B (累加器 B)

执行　　If C16 = 1

Then　　(Lmem(31～16)) － (T) → dst(39～16)

(Lmem(15～0)) － (T) → dst(15～0)

Else　　(Lmem)－((T) + (T << 16)) → dst

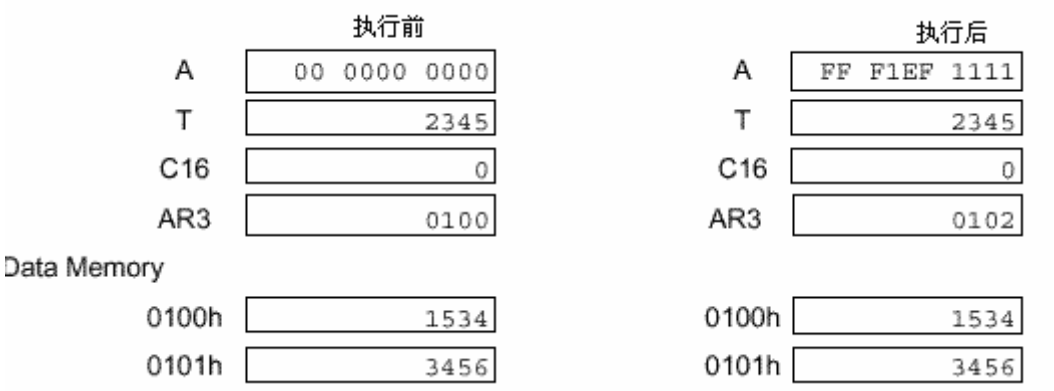
状态位　只有在 C16 = 0 时被 SXM 位和 OVM 位影响

影响 C 位和 OVdst 位

说明

该指令是从 32 位的长数据存储器操作数 Lmem 减去 T 寄存器的值，结果存在 src 中。C16 的值决定了指令执行的方式。当 C16=0 时，指令以双精度方式执行，T 寄存器的值与其左移 16 位的值连在一起组成的一个 32 位数与 Lmem 相减，结果装入 dst 中。当 C16=1 时，指令以双 16 位方式执行，从 Lmem 的高 16 位值中减去 T 寄存器的内容，结果存在目地累加器的高端，同时从 Lmem 的低端中减去 T 寄存器的值，结果存放在目的累加器 dst 的低端。在这种方式下，不管 OVM 的状态如何，都不进行饱和运算。

例　DSUBT　*AR3+，　A



32. EXP src

操作数 src： A (累加器 A) B (累加器 B)

执行 If (src) = 0

Then 0 → T

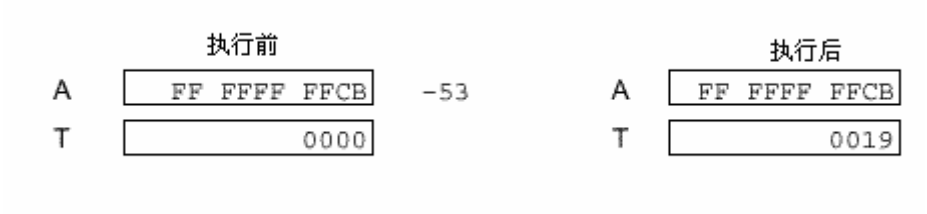
Else (src 的引导位数)－8 →T

状态位 无

说明

计算指数值并把结果存在 T 寄存器中，该值是一个范围在－8 到 31 之间的二进制补码值。指数值是通过源累加器 src 的引导位数然后减去 8 得到的，引导位数等于消除 40 位源累加器 src 中除符号位以外的有效位所需要左移的位数。指令执行后，源累加器 src 保持不变。

例 EXP A



33. FB[D] extpmad

操作数 0 ≤ extpmad ≤ 7F FFFF

执行 (pmad(15～0)) → PC

(pmad(22～16)) → XPC

状态位 无

说明

程序指针指向远程程序地址，该地址的页由 pmad 的位 22～16 决定，页中的位置由 pmad 的位 15～0 所确定，pmad 是程序中的符号或一个具体的数值。如果是延迟转移，紧接着该指令的两条单字指令或一条双字指令先执行。

例 FB 012000h

解释

该指令专门为远程跳转所使用，远程程序指的是 DSP 外部的程序或者数据空间，一般为扩展的空间，指令中程序地址由 XPC 给出。如果用户的实际系统中没有外扩的 RAM 单元，不会使用该指令。

34. FBACC[D] src

操作数 src： A (累加器 A) B (累加器 B)

执行 (src(15～0)) → PC

(src(22～16)) → XPC

状态位 无

说明

该指令是把源累加器 src 的位 22～16 装入 XPC，并让程序指针指向 src 的低端（位 15～0）所确定的 16 位地址。如果是延迟转移，紧接着该指令的两条单字指令或一条双字指令先执行。

例 FBACC A

解释

该指令和远程跳转指令一样，除非用户将程序存放在外扩的 RAM 单元，否则不会使用该指令。

35. FCALA[D] src

操作数 src： A (累加器 A) B (累加器 B)

执行 非延时

(SP) – 1 → SP

(PC) + 1 → TOS

(SP) – 1 → SP

(XPC) → TOS

(src(15～0)) → PC

(src(22～16)) → XPC

延时

(SP) – 1 → SP

(PC) + 3 → TOS

(SP) – 1 → SP

(XPC) → TOS

(src(15～0)) → PC

(src(32～16)) → XPC

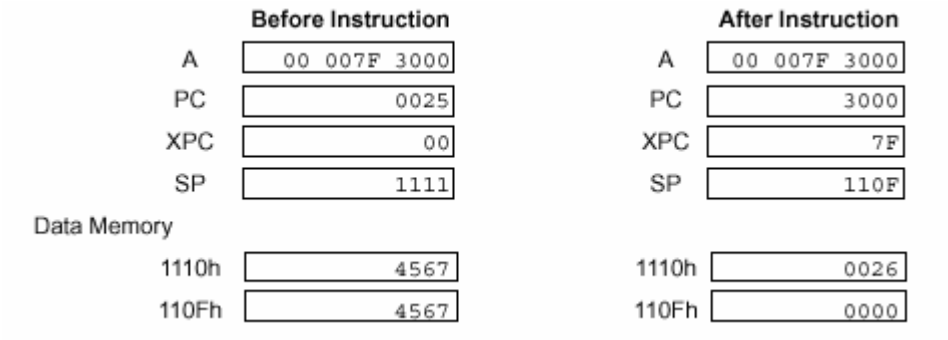
状态位 无

说明

把源累加器 src 的位 22～16 装入远程程序指针（XPC），src 的低端位 15～0 装入程序

指针（PC）。如果是延迟调用，紧接着该调用指令的两条单字指令或一条双字指令先执行。

例　　FCALA　A



解释

将累加器 A 的高端 7Fh 存到 XPC，低端 3000h 存到 PC。其中数据空间的 1110h 和 110Fh 单元的内容改变，是因为执行前后 XPC 指向的页不同，执行前后的 1110h 和 110Fh 实际上已经是完全不同的 RAM 单元。实际中，执行后的 1110h 和 110Fh 肯定是在用户外扩的 RAM 单元。该指令为远程调用指令，和远程跳转指令一样，除非用户将程序存放在外扩的 RAM 单元，否则不会使用该指令。

36. FCALL[D]　extpmad

操作数　0 ≤ extpmad ≤ 7F FFFF

执行　　非延时

(SP) － 1 → SP
(PC) + 2 → TOS
(SP) – 1 → SP
(XPC) → TOS
(pmad(15～0)) → PC
(pmad(22～16)) → XPC

延时

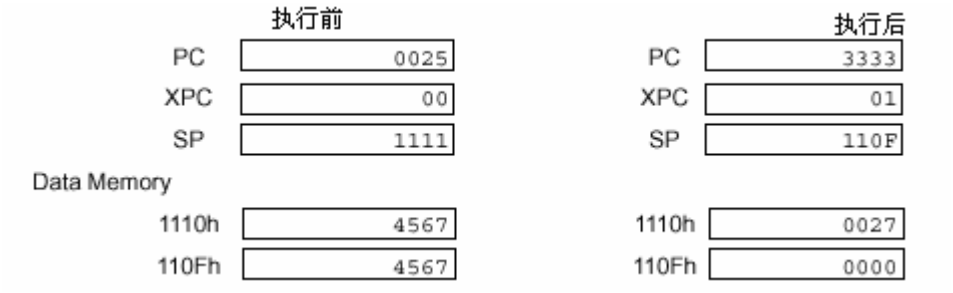
(SP) – 1 → SP
(PC) + 4 → TOS
(SP) – 1 → SP
(XPC) → TOS
(pmad(15～0)) → PC
(pmad(22～16)) → XPC

状态位　无

说明

pmad 的位 22～16 装入远程程序指针（XPC），pmad 的低端位 15～0 装入程序指针（PC）。如果是延迟调用，紧接着调用指令的两条单字指令或一条双字指令先执行。

例　　FCALL　013333h



37. FIRS Xmem， Ymem， pmad

操作数　Xmem， Ymem： 双数据存储操作数

0 ≤ pmad ≤ 65 535

执行　pmad → PAR

While (RC) ≠ 0

(B) + (A(32～16)) × (由 PAR 中的内容所决定的 Pmem 的地址单元) → B

((Xmem) + (Ymem)) << 16 → A

(PAR) + 1 → PAR

(RC) － 1 → RC

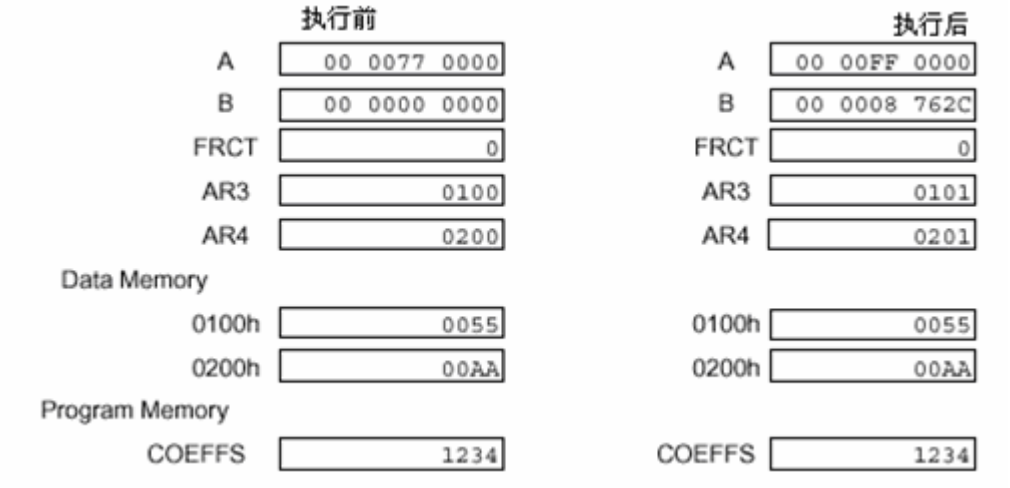
状态位　被 SXM 位、FRCT 位和 OVM 位影响

影响 C 位、OVA 位和 OVB 位

说明

累加器 A 的高端（位 32～16）和由 pmad（存放在程序地址寄存器 PAR 中）寻址的 Pmem 值相乘，结果加到累加器 B 中，同时，存储器操作数 Xmem 和 Ymem 相加，结果左移 16 位，然后装入到累加器 A 中，在下一个循环中 pmad 加１。一旦循环流水正常工作，该指令就成为单周期指令。

例　FIRS *AR3+， *AR4+， COEFFS



解释

将累加器 A 的高端乘上 COEFFS 确定数值，执行 77h×1234h＝8762Ch，加到累加器 B 中。然后执行 AR3 和 AR4 指向的地址单元内容相加，为 55h＋AAh＝FFh，左移 16 位，存在累加器 A 中。该指令一般用于滤波器运算，可以快速实现滤波器的一次乘加操作，循环执行该指令，可以快速实现两个 n 阶的向量乘法。

38. FRAME K

操作数　–128 ≤ K ≤ 127

执行　　(SP) ＋ K → SP
状态位　　无
说明

把一个短立即数 K 加到堆栈指针 SP 中。在编译方式（CPL=1）下的地址产生，紧接着该指令的下一条指令将直接运行，不会预留额外的时间等待堆栈处理。

例 FRAME　10h

39. FRET[D]

操作数　　无
执行　　(TOS) → XPC
(SP) + 1 → SP
(TOS) → PC
(SP) + 1 → SP
状态位　　无
说明

把栈顶单元的低 7 位数值装入远程程序指针（XPC）中，紧接着的下一个堆栈单元的 16 位数装入程序指针，堆栈指针在每一操作数完成后自动加 1。如果是延迟返回，紧接着该指令的两条单字指令或一条双字指令先执行。该指令为远程返回指令，和远程调用指令配套使用。

例 FRET

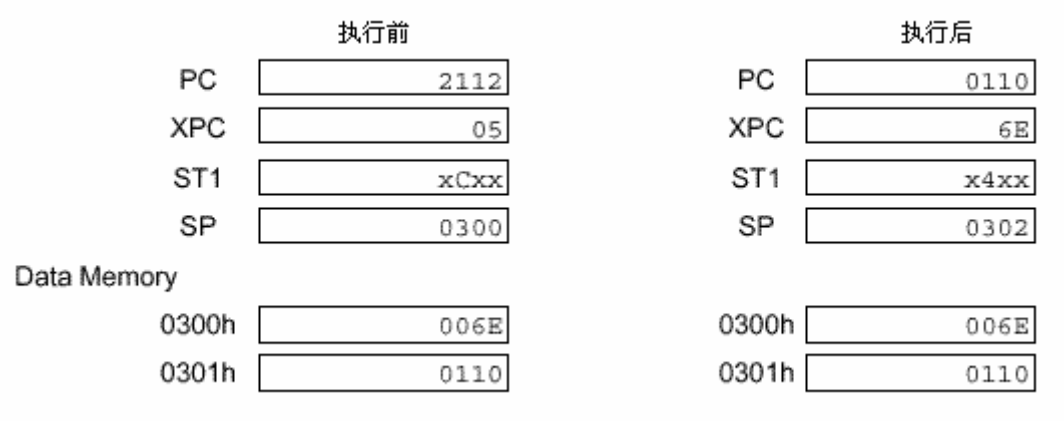
40. FRETE[D]

操作数　　无
执行　　(TOS) → XPC
(SP) + 1 → SP
(TOS) → PC
(SP) + 1 → SP
0 → INTM
状态位　　影响 INTM 位
说明

把栈顶单元的低 7 位值装入远程程序指针（XPC）中，紧接着的下一个堆栈单元的 16 位数装入程序指针，并且从新的程序指针指向的单元继续执行程序。该指令自动清除 ST1

寄存器中的中断屏蔽位（INTM 位）（清除该位表示允许中断）。如果是延迟返回，紧接着该指令的两条单字指令或一条双字指令先执行。

例 FRETE



41. IDLE K

操作数　　1 ≤ K ≤ 3

执行　　(PC) +1 → PC

状态位　　被 INTM 位影响

说明

强迫程序执行等待操作直至不可屏蔽中断产生或复位中断产生。芯片保持空闲状态（低功耗方式）。不论 INTM 如何设置，只要有一个不可屏蔽中断出现，系统就退出空闲状态，如果 INTM=1，程序继续执行紧接着 IDLE 的指令，不会响应中断；如果 INTM=０，程序转移到相应的中断服务程序。中断是通过中断屏蔽寄存器（IMR）设置。K 的值决定了可以使芯片从空闲状态中激活的中断类型：

K=1　定时器和串口等外围设备在空闲状态时仍有效，此时定时器和串口仍然有输出时钟信号。外围中断和复位以及外部中断可以激活芯片。

K=2　定时器和串口中等外围设备在空闲状态时无效，此时定时器和串口没有输出时钟信号。复位和外部中断可以激活芯片。因为在正常的设备操作条件下，中断在空闲方式下不会被锁定，所以它必须保持几个周期的低脉冲时钟才能被响应。

K=3　定时器和串口等外围设备在空闲状态时无效，锁相环 PLL 被禁止，此时芯片将彻底进入空闲状态。

42. INTR　　K

操作数　　0 ≤ K ≤ 31

执行　　(SP) － 1 → SP

(PC) + 1 → TOS

K→ PC

1→ INTM

状态位　　影响 INTM 位和 IFR 位

说明

强制程序指针指向 K 所确定的中断向量。指令允许使用软件方式执行任何中断服务程序。在指令开始执行时，程序指针加 1 并且把它压入栈顶保存，然后把 K 指定的中断向量装入程序指针，执行该中断服务程序。对中断标志寄存器（IFR）中的相应位清 0，对应的中断将被禁止（当 INTM=1 时）。中断屏蔽寄存器（IMR）不会影响该指令。注意：该指令

不能循环执行。

例 INTR 3

43. LD

（1） LD Smem， dst

（2） LD Smem， TS， dst

（3） LD Smem， 16 ， dst

（4） LD Smem [， SHIFT]， dst

（5） LD Xmem， SHFT， dst

（6） LD # K， dst

（7） LD # lk [， SHFT]， dst

（8） LD # lk， 16， dst

（9） LD src， ASM [， dst]

（10）LD src [， SHIFT]， dst

操作数 Smem： 单数据存储操作数

Xmem： 双数据存储操作数

src， dst： A (累加器 A) B (累加器 B)

0 ≤ K ≤ 255

–32 768 ≤ lk ≤ 32 767

–16 ≤ SHIFT ≤ 15

0 ≤ SHFT ≤ 15

执行（1） (Smem) → dst

（2） (Smem) << TS → dst

（3） (Smem) << 16 → dst

（4） (Smem) << SHIFT → dst

（5） (Xmem) << SHFT → dst

（6） K → dst

（7） lk << SHFT → dst

（8） lk << 16 → dst

（9） (src) << ASM → dst

（10）(src) << SHIFT → dst

状态位 在累加器载入时被 SXM 位影响

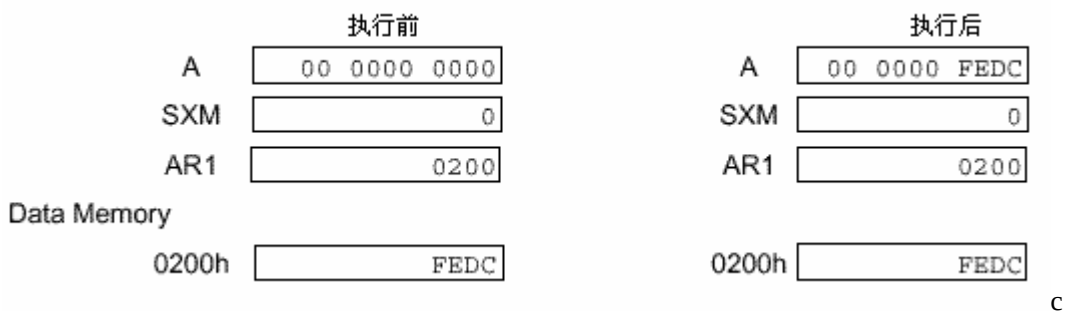
在带移位操作时被 OVM 位影响

影响 OVdst 位(如果 dst = src，影响 OVsrc 位)

说明

把一个数据存储器的值或一个立即数装入累加器，并且支持各种不同的移位。此外，该指令支持带移位的累加器到累加器的搬移。

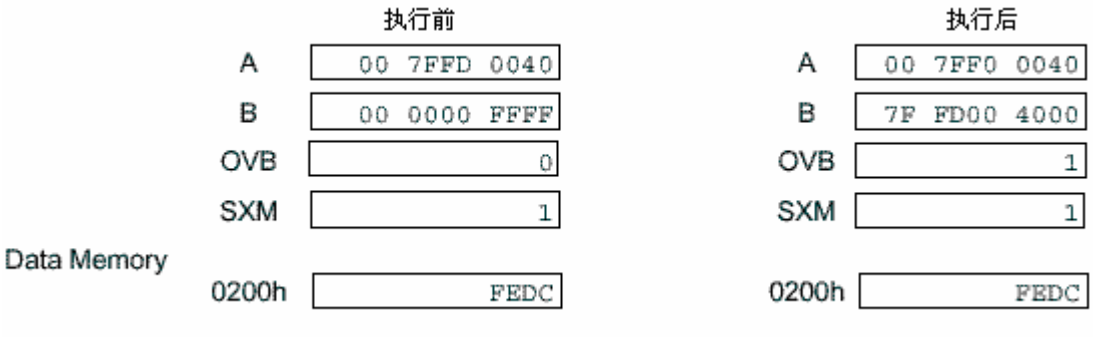
例 LD *AR1， A



c

例 LD *AR1， TS， B

例 LD A， 8， B



44. LD

（1） LD Smem， T

（2） LD Smem， DP

（3） LD # k9， DP

（4） LD # k5， ASM

（5） LD # k3， ARP

（6） LD Smem， ASM

操作数 Smem： 单数据存储操作数

0 ≤ k9 ≤ 511

–16 ≤ k5 ≤ 15

0 ≤ k3 ≤ 7

执行（1） (Smem) → T

（2） (Smem(8～0)) → DP

（3） k9 → DP

（4） k5 → ASM

（5） k3 → ARP

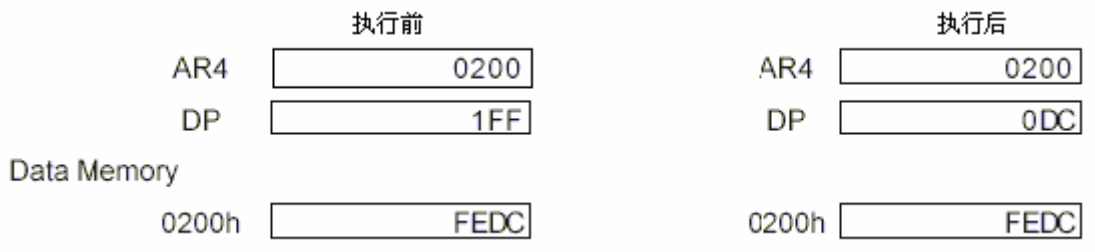
（6） (Smem(4～0)) → ASM

状态位 无

说明

把一个数据存储器的值或一个短立即数装入 T 寄存器或状态寄存器中的 DP、ASM 和 ARP 位。该指令代码为一个字，但当 Smem 采用了长偏移直接寻址或绝对地址寻址方式时，指令代码为 2 个字。

例 LD *AR4， DP



45. LDM MMR， dst

操作数 MMR： 存储映射寄存器

dst： A (累加器) B (累加器)

执行 (MMR) → dst(15～0)

00 0000h → dst(39～16)

状态位 无

说明

把存储器映射寄存器 MMR 中的值装入到目的累加器中，不论 DP 的当前内容或 ARx 的高 9 位的值是多少，都把有效地址的高 9 位清 0，强制将页指针设置为 0，该指令不受 SXM 位的影响。

例 LDM AR4， A

46. LD Xmem， dst

|| MAC[R] Ymem [， dst_]

操作数 dst： A (累加器 A) B (累加器 B)

dst_： 如果 dst = A，则 dst_ = B； 如果 dst = B，则 dst_ = A

Xmem， Ymem： 双数据存储操作数

执行 (Xmem) << 16 → dst (31～16)

If (Rounding)

Round (((Ymem) × (T)) + (dst_)) → dst_

Else　　((Ymem) × (T)) + (dst_) → dst_

状态位　　被 SXM 位、FRCT 位 OVM 和位影响

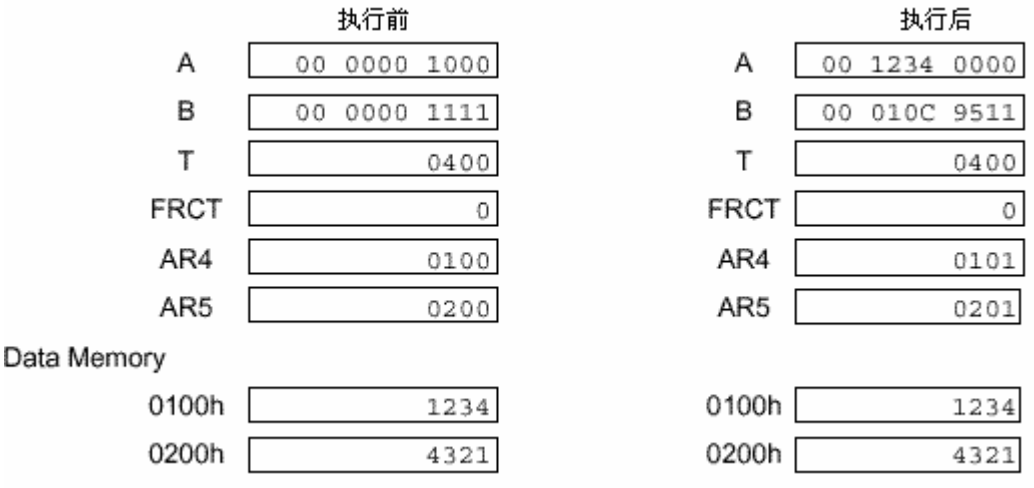
影响 OVdst_位

说明

16 位双数据存储器操作数 Xmem 左移 16 位后装入目的累加器的高端。同时并行执行一个双数据操作数 Ymem 与 T 寄存器的值相乘再把乘积加到 dst_中的操作。如果指令带有 R 后缀，则对乘积和累加器操作的结果进行四舍五入，再存在 dst 中。四舍五入的方法是：给该值加上 215，然后将结果的低端(位 15～0)清 0。

例　　LD　*AR4+，　A

　　||MAC　*AR5+，　B



例　　LD　*AR4+，　A

　　||MACR　*AR5+，　B

解释

该指令一般用于两个小数相乘，并可以对结果进行四舍五入运算。两个 16 位小数分别放在 T 寄存器和指定的单元中，结果存在累加器中。其中，累加器的低 16 位做四舍五入运算，最后余下累加器的高 16 位为两个小数相乘的结果。注意，这里的小数指的是纯小数，介于（－1，1）之间。

47. LDR Smem， dst

操作数　　Smem： 单数据存储操作数

dst： A (累加器 A)　 B (累加器 B)

执行　　(Smem) << 16 + （1 << 15） → dst(31～16)

状态位　　被 SXM 位影响

说明

把单数据存储器操作数 Smem 左移 16 位后装入目的累加器 DST 的高端(位 31～16)。Smem 通过加上 215 再对累加器的位 14～0 清 0 进行四舍五入运算，累加器的第 15 位设置为 1。

例 LDR　*AR1， A

48. LDU Smem， dst

操作数　　Smem： 单数据存储操作数

dst： A (累加器 A)　 B (累加器 B)

执行　　(Smem) → dst(15～0)

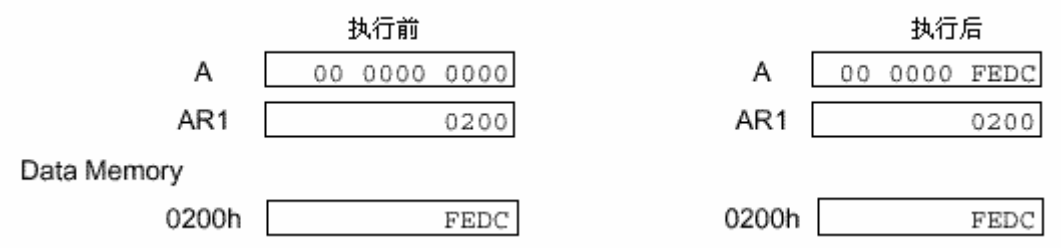
00 0000h → dst(39～16)

状态位　　无

说明

把单数据存储器 Smem 的值装入目的累加器 dst 的低端（位 15～0），dst 的保护位和高端(位 39～16)清 0。因此，数据被看成是一个无符号的 16 位数，不管 SXM 位的状态如何都不进行符号扩展。该指令代码占一个字，当 Smem 采用长偏移间接寻址或绝对地址寻址方式时就多占一个字。

例 LDU　*AR1， A



49. LMS Xmem， Ymem

操作数　　Xmem， Ymem： 双数据存储操作数

执行　　(A) + (Xmem) << 16 + 215 → A

(B) + (Xmem) × (Ymem) → B

状态位　　被 SXM 位、FRCT 位和 OVM 位影响

影响 C 位、OVA 位和 OVB 位

说明

双数据存储器操作数 Xmem 左移动 16 位后再加上 215，将结果送到累加器 A 中；同时执行 Xmem 与 Ymem 相乘，将结果送到累加器 B 中。

例 LMS *AR3+， *AR4+

解释

执行 77778888h＋550000h＋215＝77CD0888h，结果存在累加器 A 中，同时执行 55h ×AAh＋0100h＝3972h。

该指令用于最小均方值算法，该指令不会修改 T 寄存器的值，所以一般将最小均分运算所需要的误差放在 T 寄存器中，用于修改均分运算的系数。

50. LTD Smem

操作数 Smem： 单数据存储操作数

执行 (Smem) →T

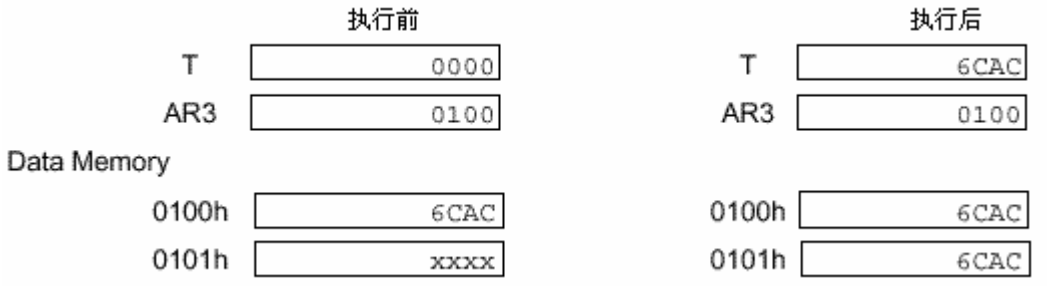
(Smem) → Smem + 1

状态位 无

说明

把一个单数据存储器单元的内容 Smem 复制到 T 寄存器和紧接着的下一个数据单元。指令执行结束后，Smem 单元的内容保持不变。该指令一般用于数字信号处理中实现一个 Z 域的时间延迟。

例 LTD *AR3



51. MAC

（1） MAC[R] Smem， src

（2） MAC[R] Xmem， Ymem， src [， dst]

（3） MAC # lk， src [， dst]

（4） MAC Smem， # lk， src [， dst]

操作数 Smem： 单数据存储操作数

Xmem， Ymem： 双数据存储操作数

src， dst： A (累加器 A) B (累加器 B)

–32 768 ≤ lk ≤ 32 767

执行（1） (Smem) × (T) + (src) → src

（2） (Xmem) × (Ymem) + (src) → dst (Xmem) → T

（3） (T) × lk + (src) → dst

（4） (Smem) × lk + (src) → dst (Smem) → T

状态位 被 FRCT 位和 OVM 位影响

影响 OVdst 位

说明

该指令是完成乘加运算，并可进行四舍五入运算（指令带有后缀 R)。结果按规定存放在 dst 或 src 中。对于指令的第 2 和第 4 种执行情况，紧接着操作码的数据储存器的值在读操作数阶段放到 T 寄存器中。

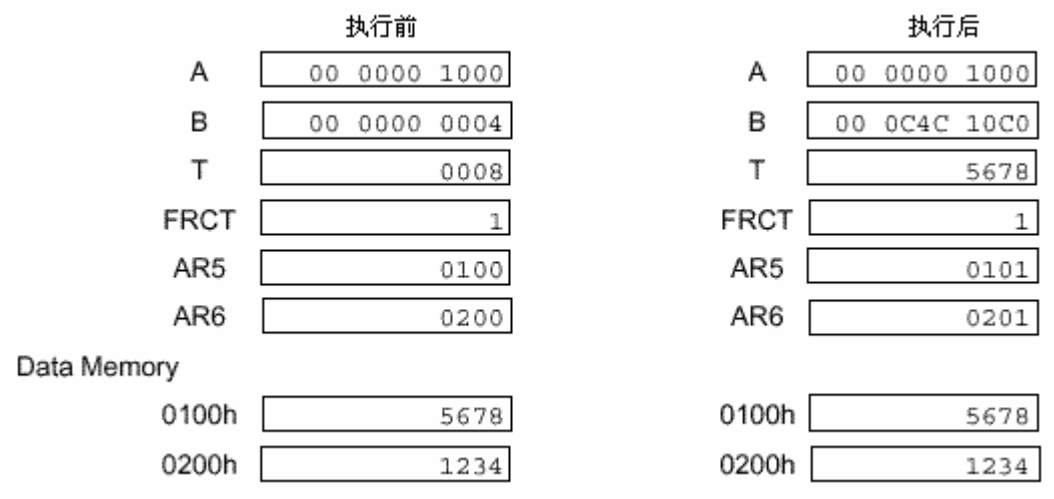
例 MAC *AR5+， A

	执行前	执行后
A	00 0000 1000	00 0048 E000
T	0400	0400
FRCT	0	0
AR5	0100	0101
Data Memory		
0100h	1234	1234

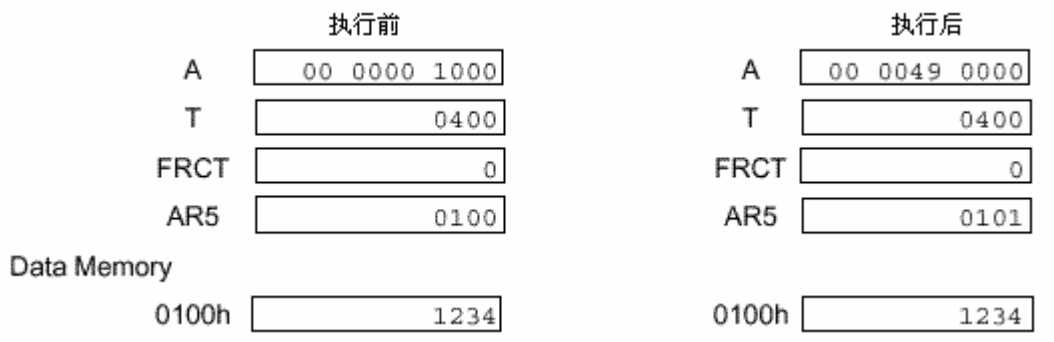
例 MAC #345h， A， B

	执行前	执行后
A	00 0000 1000	00 0000 1000
B	00 0000 0000	00 001A 3800
T	0400	0400
FRCT	1	1

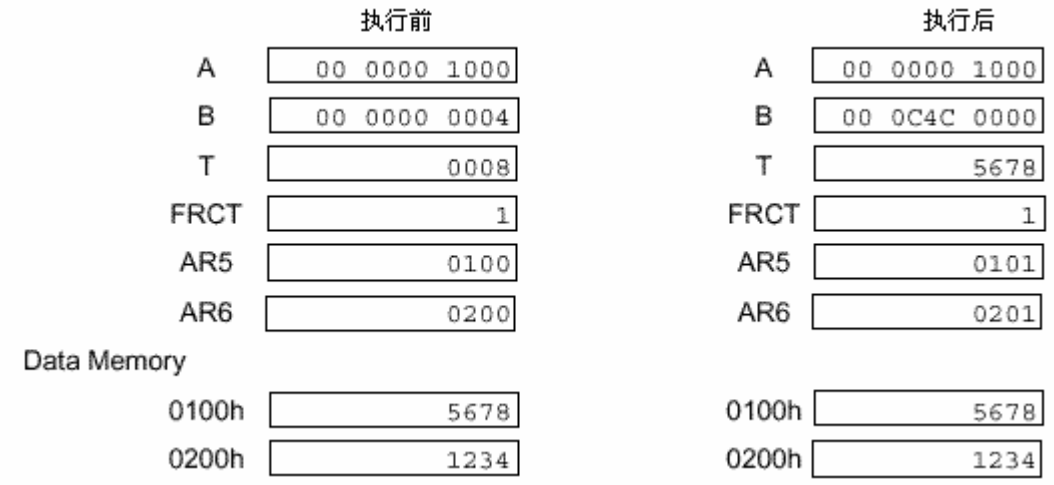
例 MAC *AR5+， *AR6+，A， B



例 MACR *AR5+， A



例 MACR *AR5+， *AR6+，A， B



52. MACA

（1） MACA[R] Smem [， B]

（2） MACA[R] T， src [， dst]

操作数 Smem： 单数据存储操作数

src， dst： A (累加器 A) B (累加器 B)

执行（1） (Smem) × (A(32～16)) + (B) → B (Smem) → T

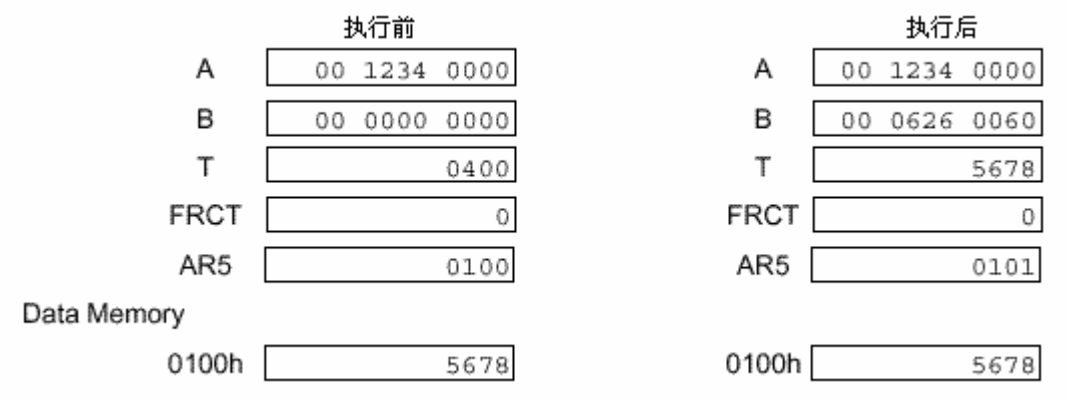
（2） (T) × (A(32～16)) + (src) → dst

状态位　被 FRCT 位和 OVM 位影响
　　影响 OVdst 位和 OVB 位

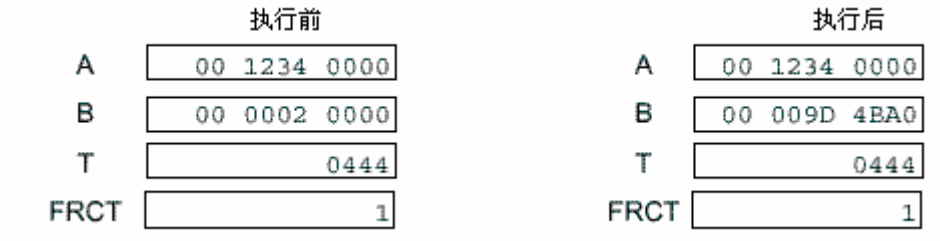
说明

累加器 A 的高端（位 32～16）与一个单数据存储器操作数 Smem 或 T 寄存器中的内容相乘，结果加到累加器 B 或源累加器中。累加器 A 的位 32～16 值用做乘法器的一个 17 位操作数。如果指令带有 R 后缀，结果进行四舍五入运算。

例　MACA　*AR5+



例　MACA　T，　B，　B



53. MACD　Smem，　pmad，　src

操作数　　Smem：　单数据存储操作数

　　src：　A (累加器 A)　　B (累加器 B)

　　0 ≤ pmad ≤ 65 535

执行　　pmad → PAR

　　If (RC) ≠ 0

　　Then

　　(Smem) × (由 PAR 中的内容所决定的 Pmem 的地址单元) + (src) → src

　　(Smem) → T

　　(Smem) → Smem + 1

　　(PAR) + 1 → PAR

　　Else

　　(Smem) × (由 PAR 中的内容所决定的 Pmem 的地址单元) + (src) → src

　　(Smem) → T

　　(Smem) → Smem + 1

状态位　　被 FRCT 位和 OVM 位影响

　　影响 OVsrc 位

说明

一个单数据存储器值 Smem 与一个程序存储器值 pmad 相乘，结果加到源累加器 src 中。此外，还把数据存储器值 Smem 装入到 T 寄存器和紧接着 Smem 地址的下一个数据单元中。循环执行该指令，则程序地址寄存器 PAR 中的程序存储器地址执行加 1 操作。循环流水一旦启动，指令就变成单周期指令。

例 MACD *AR3–， COEFFS， A

解释

执行 COEFFS 的内容和 AR3 指向的地址单元相乘，并将结果加到累加器 A 中，即 1234h×55h＋770000h＝7D0B44h。同时将 55h 分别送到 T 寄存器和 0101h 的地址单元，最后 AR3 减 1。

54. MACP Smem， pmad， src

操作数 Smem： 单数据存储操作数

src： A (累加器 A) B (累加器 B)

0 ≤ pmad ≤ 65 535

执行 (pmad) → PAR

If (RC) ≠ 0

Then (Smem) × (由 PAR 中的内容所决定的 Pmem 的地址单元) + (src) → src

(Smem) → T

(PAR) + 1 → PAR

Else (Smem) × (由 PAR 中的内容所决定的 Pmem 的地址单元) + (src) → src

(Smem) → T

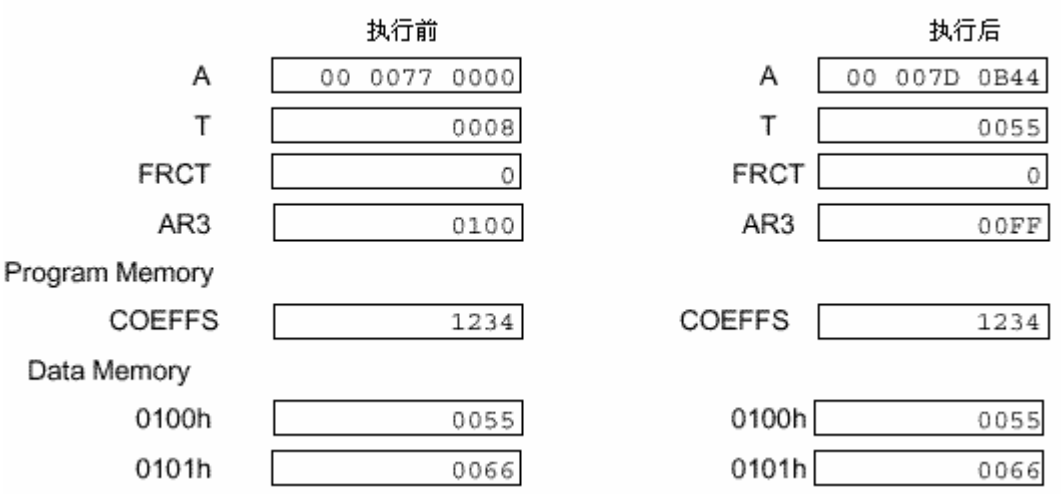
状态位 被 FRCT 位和 OVM 位影响

影响 OVsrc 位

说明

一个单数据存储值 Smem 与一个程序存储器值 pmad 相乘，结果加到源累加器 src 中。同时把数据存储器 Smem 的值复制到 T 寄存器中。循环执行该指令，则程序地址寄存器 PAR 中的程序存储器地址执行加 1 操作。该指令和 MACD 指令只差一步将 Smem 的值送到 Smem 的下一个地址单元。

例 MACP *AR3–， COEFFS， A



55. MACSU Xmem， Ymem， src

操作数　　Xmem， Ymem： 双数据存储操作数

src： A (累加器 A)　 B (累加器 B)

执行　　unsigned(Xmem) × signed(Ymem) + (src) → src

(Xmem) → T

状态位　　被 FRCT 位和 OVM 位影响

影响 OVsrc 位

说明

一个无符号的数据存储器 Xmem 与一个有符号的数据存储器值 Ymem 相乘，结果加到源累加器 src 中。同时，在读操作数阶段把这个 16 位无符号的数 Xmem 存到 T 寄存器中，由 Xmem 寻址的数据从 D 总线上获得，由 Ymem 寻址的数据从 C 总线上获得。

例　MACSU　*AR4+， *AR5+， A

56. MAR Smem

操作数　　Smem： 单数据存储操作数

执行　　If （兼容方式打开） (CMPT = 1)， then：

If (ARx = AR0 或者 ARx 为空)　AR(ARP) 被修改

ARP 不被修改

Else　　ARx 被修改

x → ARP

Else （兼容方式关闭） (CMPT = 0)

ARx 被修改

ARP 不被修改

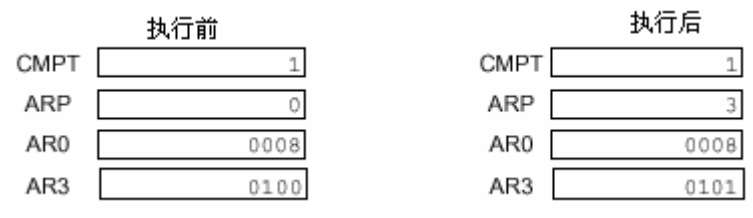
状态位　　被 CMPT 位影响

如果 CMPT = 1 影响 ARP 位

说明

修改由 Smem 所确定的辅助寄存器的内容。在兼容方式下(CMPT=1)，指令会修改 ARx 的内容以及辅助寄存器指针（ARP）的值；在非兼容方式下(CMPT=0)，指令只修改辅助寄存器的值，而不改变 ARP。该指令代码占一个字，但当 Smem 采用长偏移间接寻址或绝对寻址方式时就会多占一个字。

例　MAR　*AR3+



57. MAS

（1） MAS[R]　Smem， src

（2） MAS[R]　Xmem， Ymem， src [， dst]

操作数　　Smem： 单数据存储操作数

Xmem， Ymem： 双数据存储操作数

src， dst： A (累加器 A)　B (累加器 B)

执行　　（1） (src) – (Smem) × (T) → src

（2） (src) － (Xmem) × (Ymem) → dst

(Xmem) → T

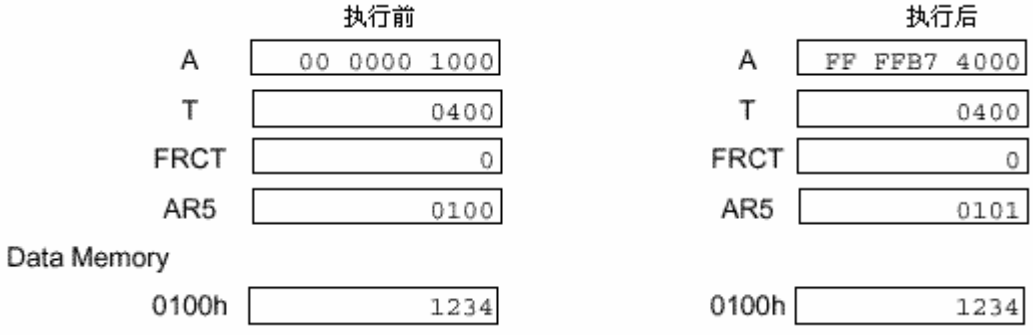
状态位　　被 FRCT 位和 OVM 位影响

影响 OVdst 位 (如果 dst = src，影响 OVsrc 位)

说明

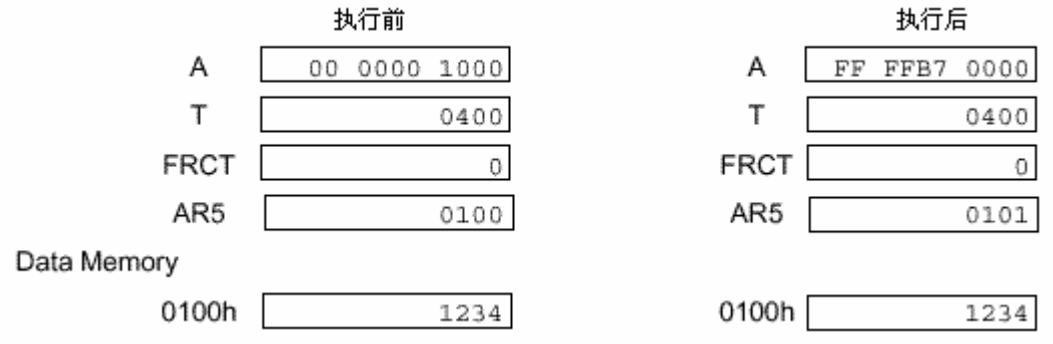
一个存储器操作数与 T 寄存器的内容相乘，或者是两个存储器操作数相乘，再从源累加器 src 或目的累加器 dst 中减去该乘积，结果存放在 src 或 dst 中。Xmem 在读操作数阶段装入到 T 寄存器中。如果指令带有 R 后缀，结果将会进行四舍五入运算。

例　　MAS　*AR5+， A

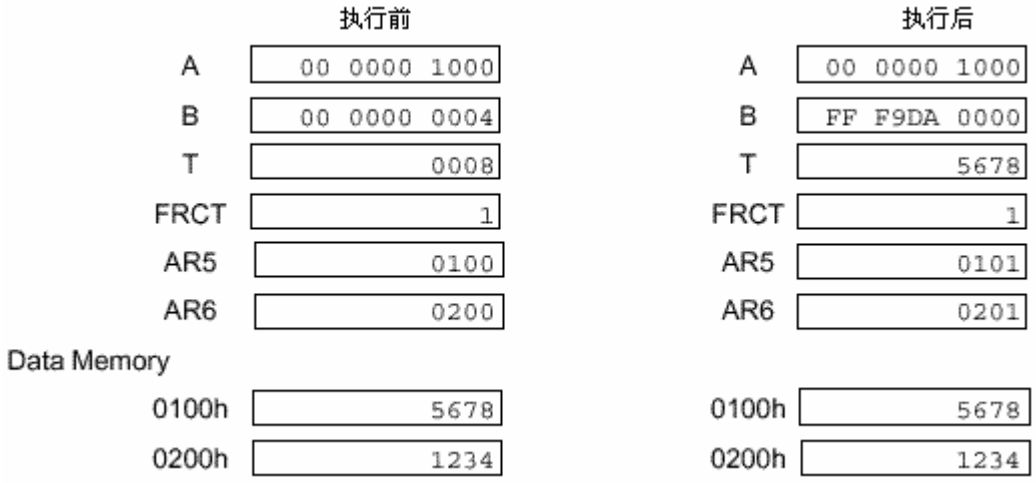


例　　MAS　*AR5+， *AR6+， A， B

例　　MASR　*AR5+，　A



例　　MASR　*AR5+，　*AR6+，　A，　B



58.　MASA

（1）　MASA　　Smem [，　B]

（2）　MASA[R]　T，　src [，　dst]

操作数　　Smem：　单数据存储操作数

src，　dst：　A (累加器 A)　B (累加器 B)

执行　　（1）　(B)－(Smem) × (A(32～16)) → B　　(Smem) → T

（2）　(src)　－　(T) × (A(32～16)) → dst

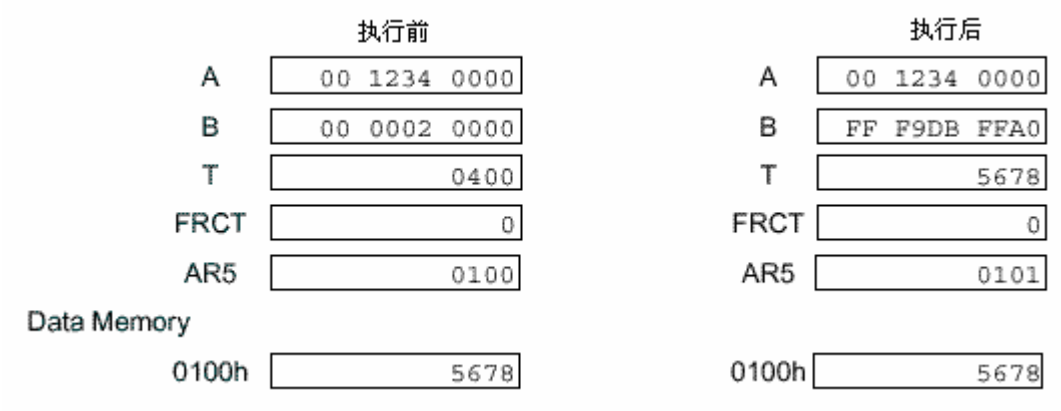
状态位　　被 FRCT 位和 OVM 位影响

影响 OVdst 位

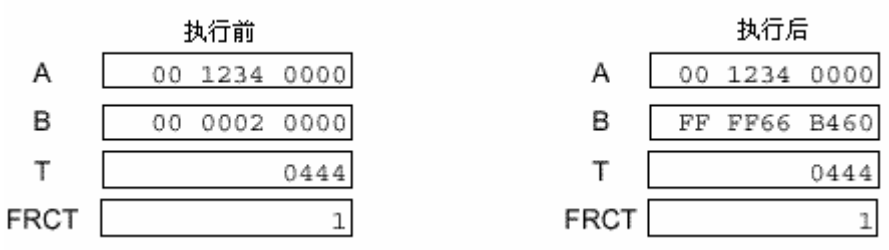
说明

累加器 A 的高端(位 32～16)与一个单数据存储器操作数 Smem 或 T 寄存器相乘，再从累加器 B 或源累加器 src 中减去该乘积，结果存放在累加器 B 或 dst 中（未定义 dst 的时候存到 src）。在读操作数阶段把 Smem 装入 T 寄存器时，如果指令带有 R 后缀，结果进行四舍五入运算。

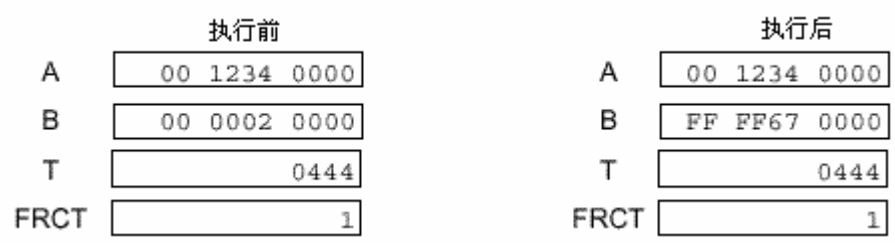
例　MASA　*AR5+



例　MASA　T，B



例　MASAR　T，B



59. MAX dst

操作数　dst： A (累加器 A)　　B (累加器 B)

执行　If (A ＞ B)

Then　(A) → dst

0 → C

Else　(B) → dst

1 → C

状态位　影响 C 位

说明

比较两累加器的内容，并把较大的一个值存放在目的累加器 dst 中。如果最大值是在

累加器 A 中，进位位 C 被清 0，否则置为 1。

例　MAX　A

60. MIN　dst

操作数　dst： A (累加器 A)　B (累加器 B)

执行　If (A < B)

Then　(A) → dst

0 → C

Else　(B) → dst

1 → C

状态位　影响 C 位

说明

比较两累加器值的大小，把较小值存放在目的累加器 dst 中。如果较小值为累加器 A，进位位 C 被清 0，否则置为 1。

例　MIN　A

61. MPY

（1） MPY[R]　Smem， dst

（2） MPY　Xmem， Ymem， dst

（3） MPY　Smem， # lk， dst

（4） MPY　# lk， dst

操作数　Smem： 单数据存储操作数

Xmem， Ymem： 双数据存储操作数

dst： A (累加器 A)　B (累加器 B)

–32 768 ≤ lk ≤ 32 767

执行（1） (T) × (Smem) → dst

（2） (Xmem) × (Ymem) → dst

(Xmem) → T

（3） (Smem) × lk → dst

(Smem) → T

（4） (T) × lk → dst

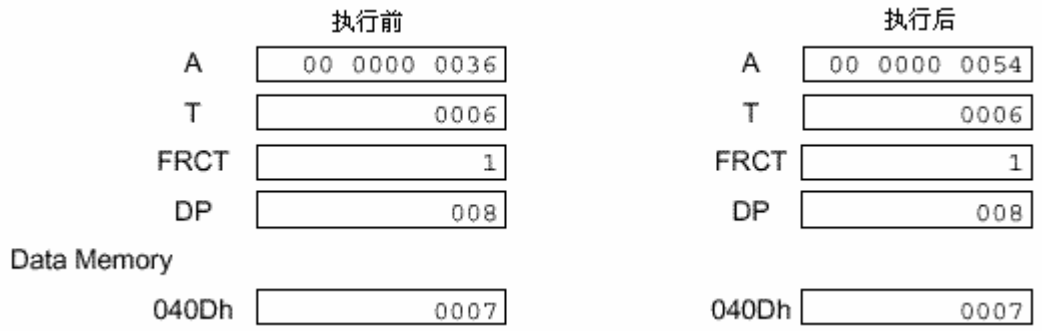
状态位　被 FRCT 位和 OVM 位影响

影响 OVdst 位

说明

T 寄存器的值或者一个数据存储器值，与另一个数据存储器值或者一个立即数相乘，结果存放在目的累加器 dst 中。在读操作数阶段把 Smem 或 Xmem 的值装入 T 寄存器中。如果指令带有 R 后缀，结果进行四舍五入运算。

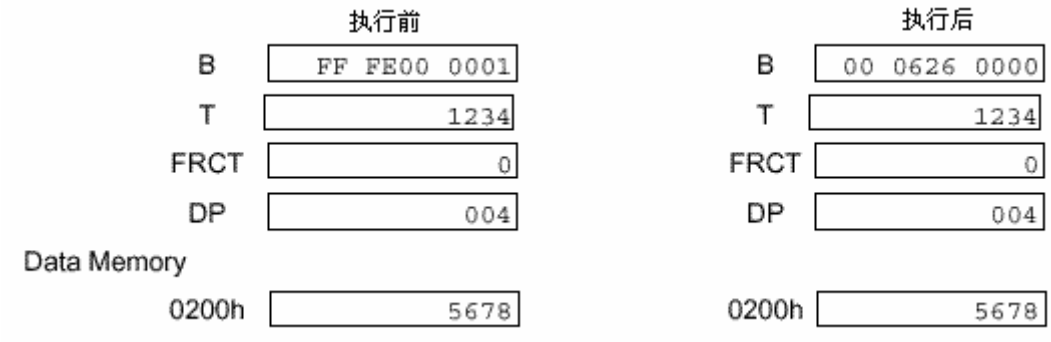
例 MPY 13， A



例 MPY *AR2–， *AR4+0%， B

例 MPY #0FFFEh， A

例 MPYR DATA0， B



62. MPYA

（1） MPYA Smem

（2） MPYA dst

操作数 Smem： 单数据存储操作数

dst： A (累加器 A) B (累加器 B)

执行 （1） (Smem) × (A(32～16)) → B

(Smem) → T

（2） (T) × (A(32～16)) → dst

状态位 被 FRCT 位和 OVM 位影响

影响 OVdst 位

说明

累加器 A 的高端（位 32～16）与一个单数据存储器操作数 Smem 或 T 寄存器相乘，结果存放在目的累加器 dst 或累加器 B 中。在读操作数期间把单数据存储器操作数 Smem 装入 T 寄存器中。

例 MPYA *AR2

63. MPYU Smem， dst

操作数 Smem： 单数据存储操作数

dst： A (累加器 A) B (累加器 B)

执行 unsigned(T) × unsigned(Smem) → dst

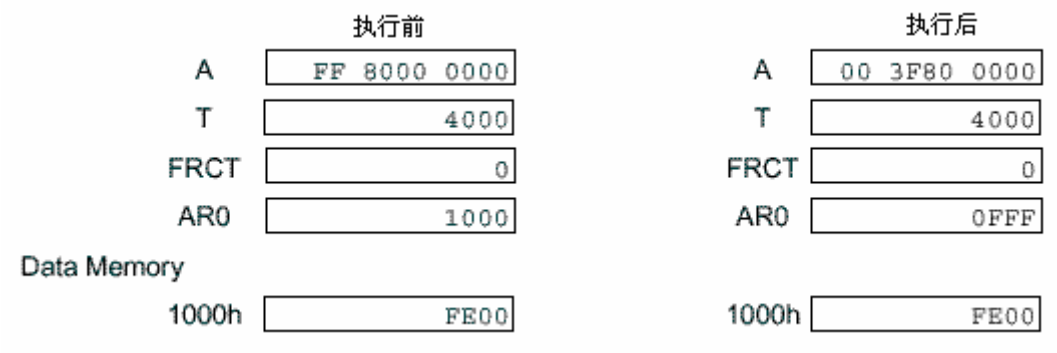
状态位 被 FRCT 位和 OVM 位影响

影响 OVdst 位

说明

无符号的 T 寄存器值与无符号的单数据存储器操作数 Smem 相乘，结果存放在目的累加器 dst。乘法器对于该指令来说相当于是其两个操作数的最高位都为 0 的一个带符号的 17×17 位的乘法器。该指令在计算两个 32 位数相乘得到一个 64 位乘积的多精度乘法时相当有用。

例 MPYU *AR0–， A



64. MVDD Xmem， Ymem

操作数　　Xmem， Ymem： 双数据存储操作数

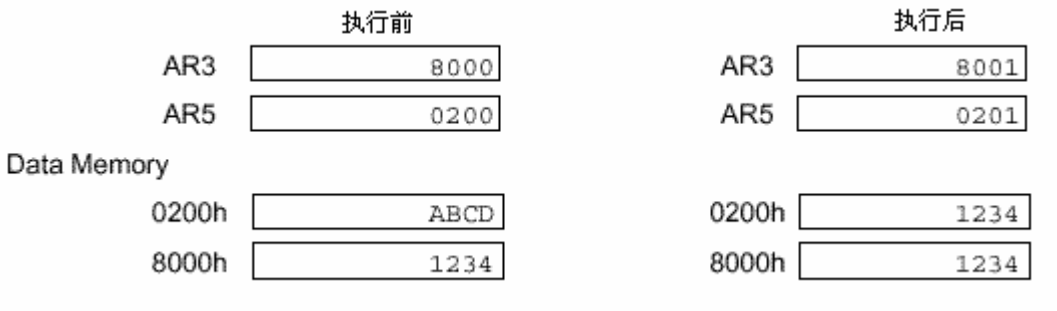
执行　　(Xmem) → Ymem

状态位　　无

说明

将 Xmem 寻址的数据存储器单元的内容复制到 Ymem 寻址的数据存储器单元中。

例　MVDD　*AR3+，　*AR5+



65. MVDK Smem， dmad

操作数　　Smem： 单数据存储操作数

0 ≤ dmad ≤ 65 535

执行　　(dmad) → EAR

If (RC) → 0

Then　　(Smem) →（由 EAR 中的内容所决定的 Dmem 的地址单元）

(EAR) + 1 → EAR

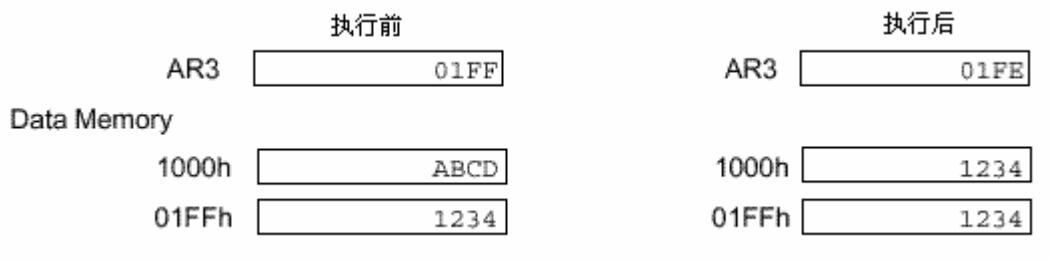
Else　　(Smem) →（由 EAR 中的内容所决定的 Dmem 的地址单元）

状态位　　无

说明

把一个单数据存储器操作数 Smem 的内容复制到一个通过 dmad（地址由 EAB 地址寄存器的 EAR 决定）寻址的数据存储器单元。可以循环执行该指令来实现数据存储器中数据块的移动。注意：实际被转移的数据个数比循环计数器中设置的值大 1。

例　　MVDK　*AR3–，　1000h



66. MVDM dmad， MMR

操作数 MMR： 存储映射寄存器

0 ≤ dmad ≤ 65 535

执行 dmad → DAR

If (RC) → 0

Then (由 DAR 中的内容所决定的 Dmem 的地址单元) → MMR

(DAR) + 1 → DAR

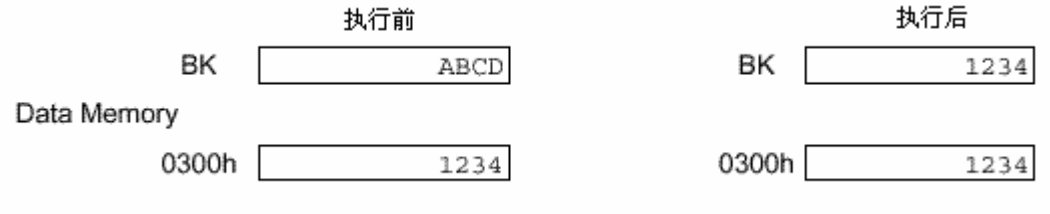
Else (由 DAR 中的内容所决定的 Dmem 的地址单元) → MMR

状态位 无

说明

把数据从一个数据存储器单元 dmad（dmad 的值由 DAB 地址寄存器的 DAR 决定）复制到一个存储器映射寄存器 MMR 中。可以循环执行该指令来实现数据存储器中数据块的移动。

例 MVDM 300h， BK



67. MVDP Smem， pmad

操作数 Smem： 单数据存储操作数

0 ≤ pmad ≤ 65 535

执行 pmad → PAR

If (RC) ≠ 0

Then (Smem) →(由 PAR 中的内容所决定的 Pmem 的地址单元)

(PAR) + 1 → PAR

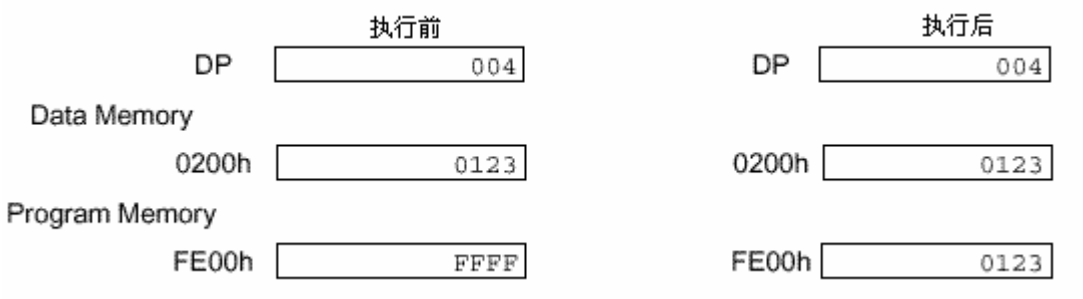
Else (Smem) → (由 PAR 中的内容所决定的 Pmem 的地址单元)

状态位 无

说明

把 16 位单数据存储器操作数 Smem 复制到一个由 16 位立即数 pmad 寻址的程序存储器单元中。可以循环执行该指令来实现数据存储器中数据块的移动。

例 MVDP 0， 0FE00h



68. MVKD dmad， Smem

操作数　　Smem： 单数据存储操作数

0 ≤ dmad ≤ 65 535

执行　　dmad → DAR

If (RC) ≠ 0

Then　(由 DAR 中的内容所决定的 Dmem 的地址单元)→ Smem

(DAR) + 1 → DAR

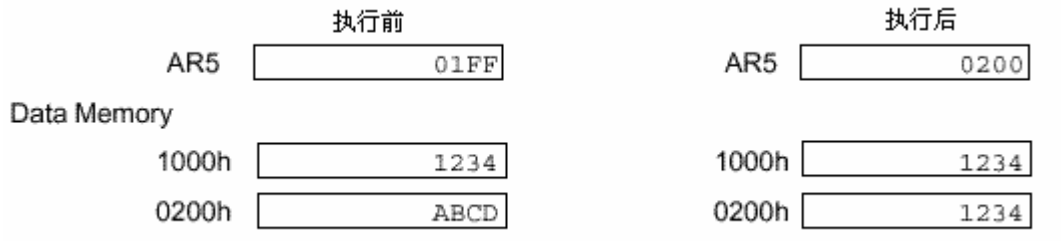
Else　(由 DAR 中的内容所决定的 Dmem 的地址单元) → Smem

状态位　　无

说明

把数据从一个数据存储器单元转移到另一个数据存储器单元。源数据存储器单元由一个 16 位立即数 dmad 寻址，目的数据存储器单元由 Smem 寻址。可以循环执行该指令来实现数据存储器中数据块的移动。

例　MVKD　1000h，　*+AR5



69. MVMD MMR， dmad

操作数　　MMR： 存储映射寄存器

0 ≤ dmad ≤ 65 535

执行　　dmad → EAR

If (RC) ≠ 0

Then　(MMR) →(由 EAR 中的内容所决定的 Dmem 的地址单元)

(EAR) + 1 → EAR

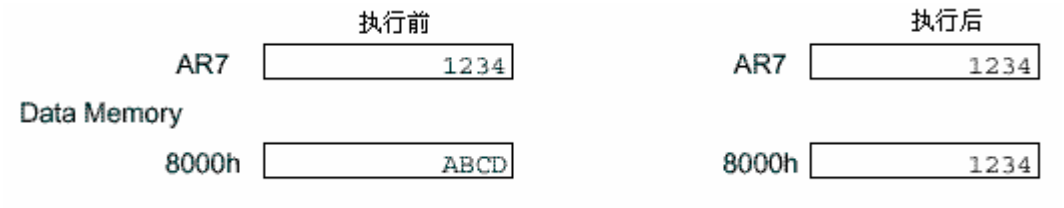
Else　(MMR) → (由 EAR 中的内容所决定的 Dmem 的地址单元)

状态位　　无

说明

把数据从一个存储器映射寄存器 MMR 转移到一个数据存储器中。可以循环执行该指令来转移数据存储器中数据块移动。

例　MVMD　AR7，　8000h



70. MVMM MMRx， MMRy

操作数 MMRx： AR0～AR7， SP
MMRy： AR0～AR7， SP

执行 (MMRx) →MMRy

状态位 无

说明

把存储器映射寄存器 MMRx 中的内容转移到另一个存储器映射寄存器 MMRy 中。MMRx 和 MMRy 只可能为 9 种操作数：AR0～AR7 和 SP。读 MMRx 的操作在译码阶段执行，写 MMRy 的操作在访问阶段执行。注意：该指令不能循环执行。

例 MVMM SP， AR1

71. MVPD pmad， Smem

操作数 Smem： 单数据存储操作数
0 ≤ pmad ≤ 65 535

执行 pmad → PAR
If (RC) ≠ 0
Then (由 PAR 中的内容所决定的 Pmem 的地址单元) → Smem
(PAR) + 1 → PAR
Else (由 PAR 中的内容所决定的 Pmem 的地址单元) → Smem

状态位 无

说明

把一个 16 位立即数 pmad 寻址的程序存储器内容复制到一个由 Smem 寻址的数据存储器单元。可以循环执行该指令来实现数据存储器中数据块的移动。

例 MVPD 2000h， *AR7–0

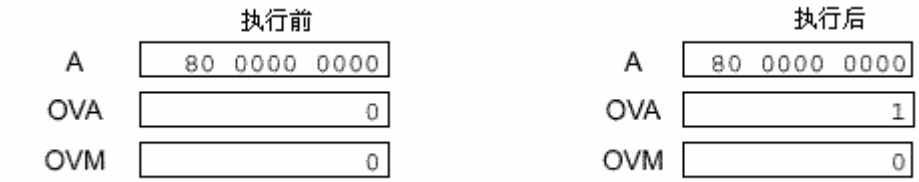
72. NEG src [， dst]

操作数　　src， dst： A (累加器 A)　B (累加器 B)
执行　　(src) ×(–1) → dst
状态位　　被 OVM 位影响
　　影响 C 位和 OVdst 位 (如果 dst = src，影响 OVsrc 位)
说明

计算源累加器的二进制补码，并把结果存放在 dst 或 src 中。只要累加器值不为 0。指令就对进位位 C 清 0；反之，如果累加器值为 0，进位位置为 1。

如果累加器的值为 80 0000 0000h，因为 80 0000 0000h 的补码超过了累加器允许的最大值，该运算将产生溢出。如果 OVM=1，目的累加器 dst 赋值为 00 7FFF FFFFh；如果 OVM=0，目的累加器 dst 赋值为 80 0000 0000 h。

例　　NEG　A



例　　NEG　A

73. NOP

操作数　无
执行　　无
说明

该指令除了程序指针执行加１操作以外不执行任何操作。这在建立流水和执行延迟方面比较有用。

74. NORM　src [， dst]

操作数　　src， dst ： A (累加器 A)　B (累加器 B)
执行　　(src) << TS → dst
状态位　　被 SXM 位和 OVM 位影响
　　影响 OVdst 位(如果 dst = src，影响 OVsrc 位)
说明

对源累加器 src 中的有符号数进行归一化，结果存放在 dst 或者 src 中。归一化实际上就是将源累加器的值按照指定的位数进行移位操作，移位数由 T 寄存器的位 5～0 确定，这 6 位数值组成一个有符号的整数，有效的移位数是－16～31。定点数的归一化是通过寻找符号扩展数的数量级把这个数分成数值部分和指数部分。该指令允许累加器的单周期归一化，指令 EXP 就是一例。

例　　NORM　A

例　　NORM　B，　A

解释

T 寄存器的位 5～0 为 39h，高位第 6 位为 1，表示负数，取反加 1 后为 7，所以实际上是右移 7 位，21 0A0A 0A0Ah>>7=00 4214 1414h。

75. OR

（1） OR　Smem，　src

（2） OR　# lk [，　SHFT]，　src [，　dst]

（3） OR　# lk，　16，　src [，　dst]

（4） OR　src [　，　SHIFT]，　[，　dst]

操作数　　src，　dst　：　A (累加器 A)　　B (累加器 B)

Smem　：　单数据存储操作数

0　≤　SHFT　≤　15

–16　≤　SHIFT　≤　15

0　≤　lk　≤　65 535

执行（1） (Smem) OR (src(15～0)) → src

src(39～16) 不变

（2） lk << SHFT OR (src) → dst

（3） lk << 16 OR (src) → dst

（4） (src or [dst]) OR (src) << SHIFT → dst

状态位　　无

说明

两个操作数进行相或运算，结果可按指定的方式移位，并存放在 dst 或者 src 中。如果左移，低位添 0，高位不进行符号扩展。如果右移，高位直接添 0。

例　OR　A，　+3，　B

76. ORM　# lk，　Smem

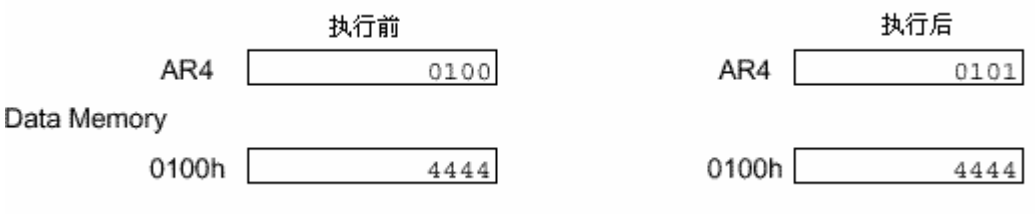
操作数　　Smem：　单数据存储操作数

0　≤　lk　≤　65 535

执行　　lk OR (Smem) → Smem
状态位　　无
说明

单数据存储器操作数 Smem 与 16 位立即数相或，结果存放在 Smem 中。该指令实现的是存储器到存储器的操作。注意：该指令不能循环执行。

例 ORM　0404h，　*AR4+

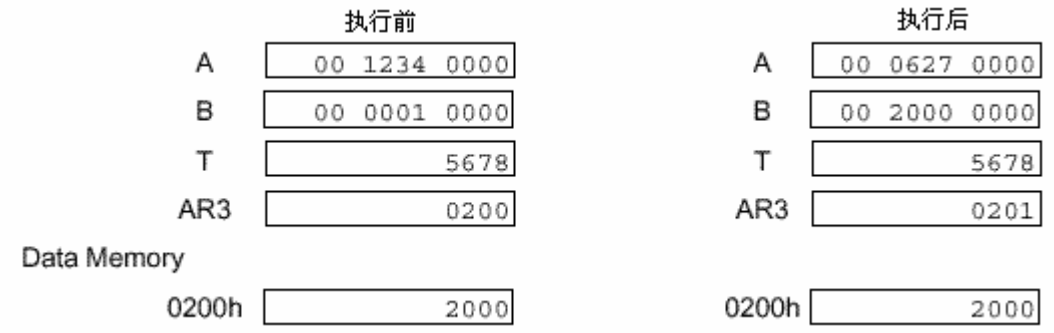


77. POLY　　Smem

操作数　Smem　：　单数据存储操作数
执行　　Round (A(32～16) × (T) + (B)) → A
　　(Smem) << 16 → B
状态位　被 FRCT 位、OVM 位和 SXM 位影响
　　影响 OVA 位
说明

单数据存储器操作数 Smem 的内容左移 16 位，结果存放在累加器 B 中。同时并行执行，累加器 A 的高端（位 32～16）与 T 寄存器的值相乘，乘积加到累加器 B 中，再对此运算的结果进行四舍五入运算，把最后结果存放在累加器 A 中。该指令在多项式计算中为实现每一单项只执行一个周期是很有用的。

例　POLY　*AR3+



解释

执行 5678h×1234h＋1 0000h＝627 0060h，将 627 0060h 进行四舍五入得到 627 0000h，存到累加器 A 中；同时执行将 AR3 指向的地址单元左移 16 位后送到累加器 B；最后 AR3 自动加 1。该指令一般用于两个纯小数乘法。

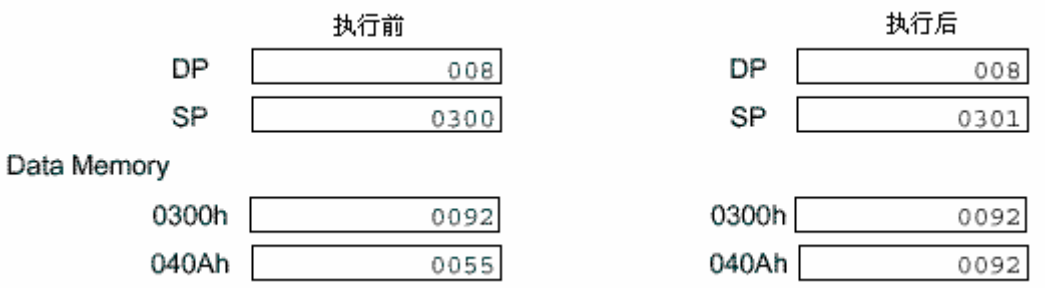
78. POPD　　Smem

操作数　Smem：　单数据存储操作数
执行　　(TOS) → Smem
　　(SP) ＋ 1 → SP
状态位　无

说明

把由堆栈指针 SP 寻址的数据存储器单元的内容转移到由 Smem 确定的数据存储器单元中，然后堆栈指针 SP 执行加 1 操作。

例 POPD 10



解释

SP 的值为 300h，表示将 300h 单元的内容送到 Smem 中。此处 Smem 由 DP 指定，DP 的值为 8，表示数据在第 8 页，第 8 页的首地址为 400h，加上指令中的十进制常数 10，得到 40Ah。

79. POPM MMR

操作数 MMR： 存储映射寄存器

执行 (TOS) → MMR

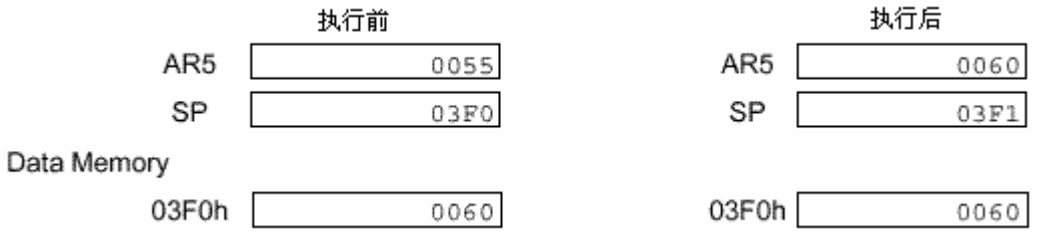
(SP) ＋ 1 → SP

状态位 无

说明

把由堆栈指针 SP 寻址的数据存储器单元的内容转移到指定的存储器映射寄存器 MMR 中，然后堆栈指针 SP 执行加 1 操作。

例 POPM AR5



80. PORTR PA， Smem

操作数 Smem： 单数据存储操作数

0 ≤ PA ≤ 65 535

执行 (PA) → Smem

状态位 无

说明

从一个外部 I/O 端口 PA（地址为 16 位立即数）读入 16 位数据到指定的数据存储器单元 Smem 中。$\overline{IS}$ 信号变为低电平来表明 DSP 在访问 I/O 口；$\overline{IOSTRB}$ 和 READY 的时

序和读外部数据存储器的时序相同。
例 PORTR 05， 60h

解释

该指令和下一条的 PORTW 指令，是专门对外部 I/O 空间的读写操作，一般用于外扩的 I/O 元件（AD、DA 等）读写。该指令执行时，地址总线和数据总线同时变化，地址总线选通存储元件的片选信号（在外扩多个元件必须使用），数据总线将数据读出或者写入。同时 $\overline{IS}$ 、 $\overline{IOSTRB}$ 以及 READY 引脚的信号都会相应变化，以便适合各种不同元件的时序要求。

81. PORTW Smem， PA

操作数 Smem： 单数据存储操作数
0 ≤ PA ≤ 65 535
执行 (Smem) → PA
状态位 无
说明

将指定的数据存储器单元 Smem 中的 16 位数据写到外部 I/O 端口 PA， $\overline{IS}$ 信号变为低电平表明 DSP 在访问 I/O 口； $\overline{IOSTRB}$ 和 READY 的时序与读外部数据存储器的时序相同。
例 PORTW 07h， 5h

82. PSHD Smem

操作数 Smem： 单数据存储操作数
执行 (SP) －1 → SP
(Smem) → TOS
状态位 无

说明

堆栈指针 SP 执行减 1 操作后，把存储器单元 Smem 的内容压入到堆栈指针 SP 指向的数据存储器单元中去。

例 PSHD *AR3+

83. PSHM MMR

操作数 MMR： 存储映射寄存器

执行 (SP) － 1 → SP

(MMR) → TOS

状态位 无

说明

堆栈指针 SP 执行减 1 操作后，再把存储器映射寄存器 MMR 中的内容压入到堆栈指针 SP 所指向的数据存储器单元中去。

例 PSHM BRC

84. RC[D] cond [， cond [， cond]]

执行 If (cond(s))

Then (TOS) → PC

(SP) + 1 → SP

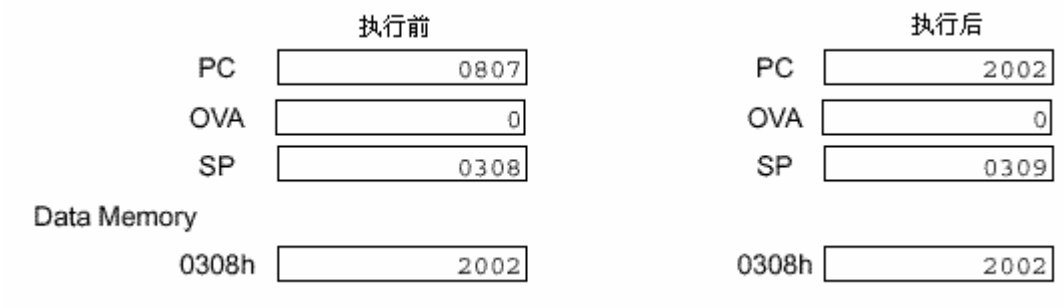
Else (PC) + 1 → PC

状态位 无

说明

当满足给出的条件时，存放在堆栈顶端的数据存储器的值弹到程序指针 PC 中，堆栈指针 SP 加 1；如果不满足条件，仅仅执行 PC 加 1 操作。如果是延迟返回，紧接着该指令的两条单字指令或一条双字指令取出先执行。先执行这两个指令字不会影响正在被测试的条件。在把程序指针转移指向另一个单元前也可对多个条件进行测试。

例 RC AGEQ， ANOV



85. READA Smem

操作数 Smem： 单数据存储操作数

执行 A → PAR

If ((RC) ≠ 0)

(由 PAR 中的内容所决定的 Pmem 的地址单元) → Smem

(PAR) + 1 → PAR

(RC) – 1 → RC

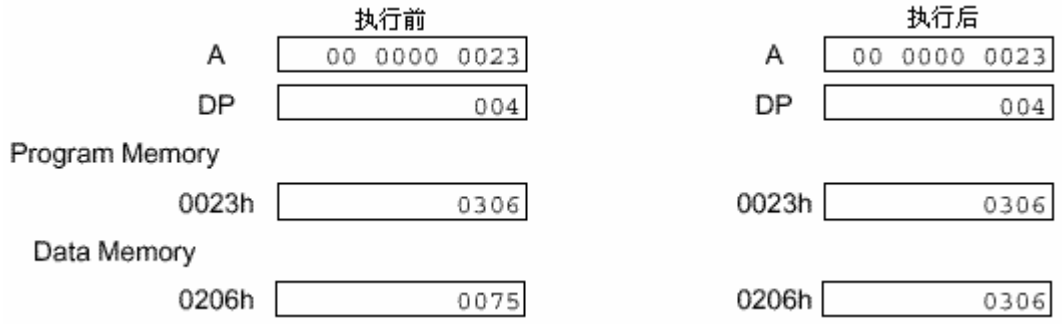
Else (由 PAR 中的内容所决定的 Pmem 的地址单元) → Smem

状态位 无

说明

把累加器 A 确定的程序存储单元中的一个字传送到一个数据存储器单元 Smem 中去。一旦建立了循环流水，指令就变成了单周期指令。可以循环执行该指令以把一块连续字（由累加器 A 确定起始地址）转移到一个连续的使用间接寻址方式的数据存储器空间中去。源和目的块不必全部都在片内或片外。

例 READA 6



解释

该指令和的 WRITA 指令，是专门对外部数据空间的读写操作，一般用于外扩的数据存储元件（RAM、FLASH、EEPROM 等）读写。该指令执行时，地址总线和数据总线同时变化，地址总线选通存储元件的地址单元，数据总线从该地址数据读出或者写入。同时 $\overline{DS}$、$\overline{PS}$、$\overline{MSTRB}$ 以及 READY 引脚的信号都会相应变化，以便适合各种不同的存储元件的时序要求。其中，地址的选择范围各种型号的 DSP 有所不同，C54xx 系列的 DSP 最高可以达到 23 位地址空间，当选择超过 16 位地址空间时，必须将地址空间数值放到累加器中。

86. RESET

操作数 无

执行　　(IPTR) << 7→PC

OVA、OVB、OVM 、ARP、DP、ASM、BRAF 位清 0

C16、FRCT、CMPT、CPL、IFR、HM 位清 0

C、TC 、SXM、XF、INTM 位置 1

说明

该指令实现了一个非屏蔽的软件复位，其在任何时候都能使用以使芯片处于可知状态。当执行这个复位指令时，就会给上面所列的各种状态位赋值。MP/$\overline{MC}$ 引脚在软件复位期间不被取样。IPTR 和外围寄存器的初始化与使用硬件初始化有所不同。该指令不受 INTM 的影响，但它对 INTM 置位以禁止中断。注意：该指令不能循环执行。

例 RESET

87. RET[D]

操作数　无

执行　　(TOS) → PC

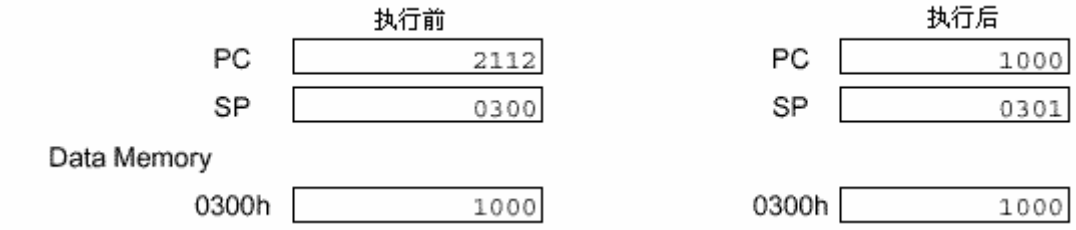
(SP) ＋ 1 → SP

状态位　无

说明

把栈顶 TOS 单元中的 16 位数据弹入到程序指针 PC 中，堆栈指针 SP 加 1。如果是延迟返回，紧接着该指令的两条单字指令或一条双字指令取出先执行。注意：该指令不能循环执行。

例 RET



88. RETE[D]

操作数　　无

执行　　(TOS) → PC

(SP) ＋ 1 → SP

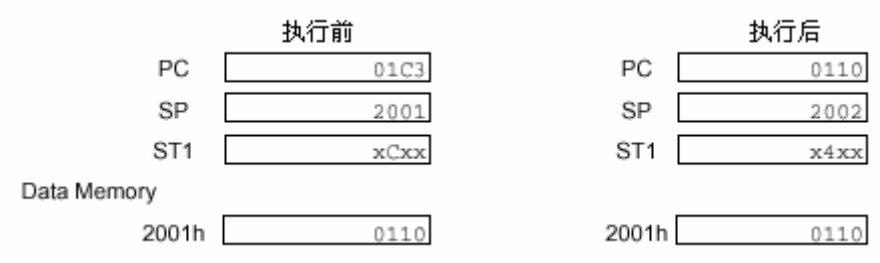
0 → INTM

状态位　　影响 INTM 位

说明

把栈顶单元 TOS 中的 16 位数据弹入到程序指针 PC 中，且从这个地址继续执行，堆

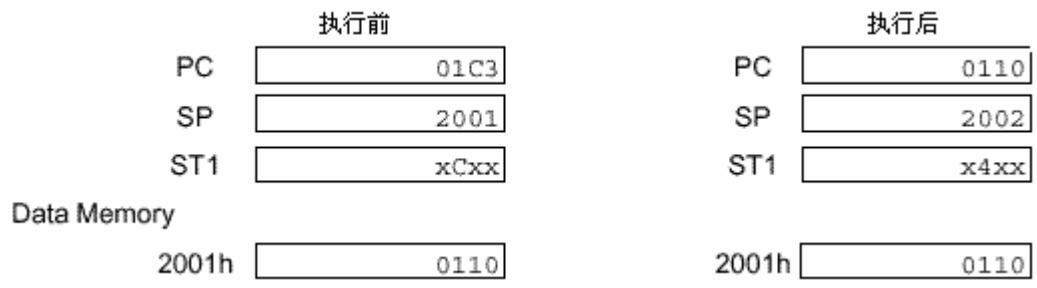
栈指针 SP 加 1。该指令自动对 ST1 中的中断屏蔽位清 0，即允许中断。如果是延迟返回，紧接着该指令的两条单字指令或一条双字指令取出先执行。注意：该指令不能循环执行。
例 RETE



89. RETF [D]

操作数 无
执行 (RTN)→PC
(SP) ＋ 1 → SP
0 → INTM
状态位 影响 INTM 位
说明

把快速返回寄存器 RTN 中的 16 位值装入到程序指针 PC 中，RTN 中保存中断服务程序返回的地址。RTN 是在返回时载入到程序指针 PC 中，然后堆栈指针 SP 自动加 1，同时对 ST1 寄存器中的中断屏蔽 INTM 位清 0（清 0 表示允许中断）。如果该指令是延迟返回，紧接着该指令的两条单字指令或一条双字指令取出先执行。注意：该指令只有在中断服务程序执行期间且没有调用其它程序时才能使用。
例 RETF



90. RND src [， dst]

操作数 src ， dst： A (累加器 A) B (累加器 B)
执行 (src) + 8000h→dst
状态位 被 OVM 位影响
说明

把 2^{15} 加到源累加器 src 中以对其进行四舍五入运算，结果存在 dst 或者 src（在没有指定 dst 的情况下）中。注意：该指令不能循环执行。
例 RND A， B

例 RND A

91. ROL src

操作数 src ： A (累加器 A) B (累加器 B)

执行 (C)→src(0)
(src(30～0))→src(31～1)
(src(31))→C
0→src(39～32)

状态位 被 C 位影响
影响 C 位

说明

源累加器 src 循环左移一位。进位位 C 的值移入 src 的最低位，src 的最高位移入 C 中，累加器保护位清 0。

例 ROL A

92. ROLTC src

操作数 src： A (累加器 A) B (累加器 B)

执行 (TC)→src(0)
(src(30～0))→src(31～1)
(src(31))→C
0→src(39～32)

状态位 被 TC 位影响
影响 C 位

说明

源累加器 src 循环左移一位，TC 的值移入 src 的最低位，src 的最高位移到进位位 C 中，累加器的保护位清 0。

例 ROLTC A

93. ROR src

操作数 src： A (累加器 A) B (累加器 B)

执行 (C)→src(31)

(src(31～1))→src(30～0)

(src(0))→C

0→src(39～32)

状态位 被 C 位影响

影响 C 位

说明

源累加器 src 循环右移一位，进位位 C 的值移入 src 的最高位，src 的最低位移到 C 中，累加器的保护位清 0。

例 ROR A

94. RPT

（1） RPT Smem

（2） RPT # K

（3） RPT # lk

操作数 Smem： 单数据存储操作数

0 ≤ K ≤ 255

0 ≤ lk ≤ 65 535

执行（1） (Smem) → RC

（2） K → RC

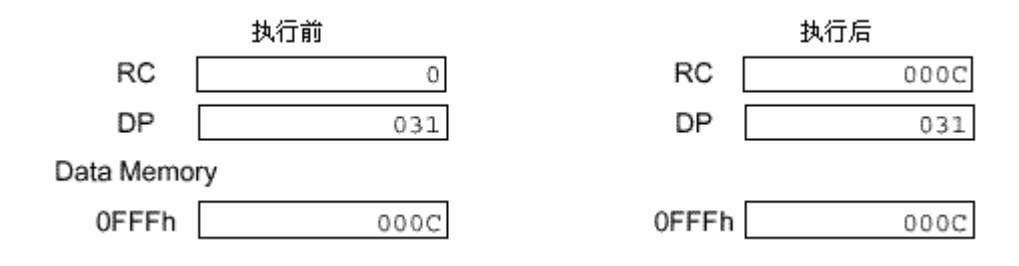
（3） lk → RC

状态位 无

说明

当指令执行时，首先把循环的次数装入循环计数器（RC），循环的次数由一个 16 位单数据存储器操作数 Smem 或一个 8 位常数 K 或者 16 位常数 lk 给定。之后，紧接着的下一条指令会循环执行。RC 在执行减 1 操作时不能被访问。注意：该指令不能循环执行，即不能套用循环。

例 RPT DAT127 ； DAT127 = 0FFFH



解释

该指令实现对下一条指令的自动循环执行。但很多指令都不能循环执行，在 CCS 仿真中编译器对这种错误不一定提示；此外，在循环执行期间，所有外部的中断都不会响应，如果有较长时间的循环，一定保证循环期间没有中断产生。

95. RPTB[D] pmad

操作数　　0 ≤ pmad ≤ 65 535

执行　　1 → BRAF

If (延时)　then

(PC) + 4 → RSA

Else　　(PC) + 2 → RSA

pmad → REA

状态位　影响 BRAF 位

说明

循环执行一段指令块，循环的次数由存储器映射的块循环计数器（BRC）确定。BRC 的值必须在指令执行之前设置。程序执行时，块循环起始地址寄存器（RSA）中装入程序指针 PC+2（如果是采用了延迟就是 PC+4）；块循环尾地址寄存器（REA）中装入程序存储器地址（pmad）。该指令执行时可以被中断。如果嵌套使用该指令，嵌套时必须保证以下两点：

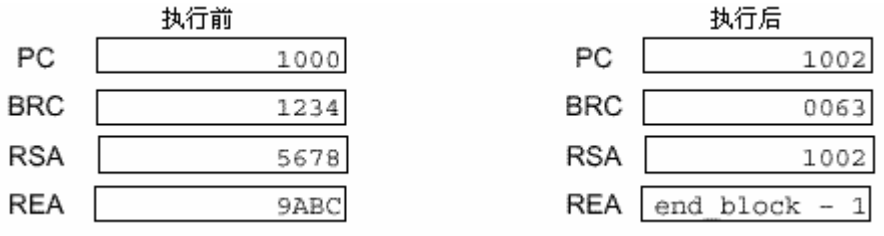
（1）BRC，RSA 和 REA 寄存器的内容都必须作适当的保存和恢复。

（2）块循环有效标志（BRAF）要正确的设置。

在带延迟的块循环中，紧接着该指令的两条单字指令和一条双字指令被取出先执行。注意，块循环可以通过对 BRAF 位清 0 来禁止。

例　ST　#99，　BRC；执行循环块 100 次

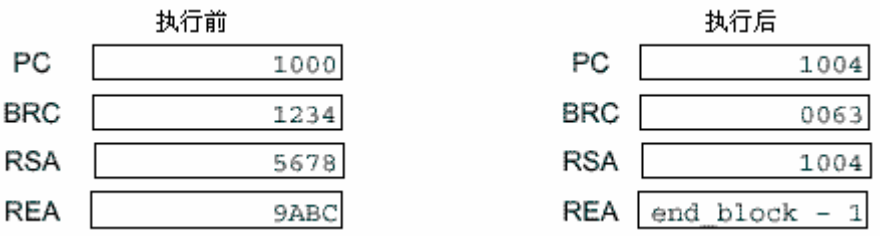
RPTB　end_block – 1



例　ST　#99，　BRC　；执行循环块 100 次

RPTBD　end_block – 1

MVDM　POINTER，　AR1



96. RPTZ dst， # lk

操作数 dst： A (累加器 A) B (累加器 B)

0 ≤ lk ≤ 65 535

执行 0 → dst

lk → RC

状态位 无

说明

对目的累加器 dst 清 0，并且循环执行下一条指令 n+1 次。其中的 n 是循环计数器（RC）中的一个 16 位常数值。

例 RPTZ A， 1023 ；重复下一条指令 1024 次

STL A， *AR2+

97. RSBX N， SBIT

操作数 0 ≤ SBIT ≤ 15

N = 0 或 1

执行 0 → STN(SBIT)

状态位 无

说明

对状态寄存器 ST0 和 ST1 的特定位清 0。N 指明了谁是被修改的状态寄存器（N＝0 表示修改 ST0 寄存器，N＝1 表示修改 ST1 寄存器），SBIT 确定被修改的位。可直接用状态寄存器中的一个特定位的名称作为操作数，这样就不需要使用 N 和 SBIT。注意：该指令不能循环执行。

例 RSBX SXM ；SXM 表明了为 ST1 寄存器的第 8 位

例 RSBX 1，8；这两条指令的执行结果完全一样

98. SACCD src， Xmem， cond

操作数 src： A (累加器 A) B (累加器 B)

Xmem： 双数据存储操作数

执行 If (cond)

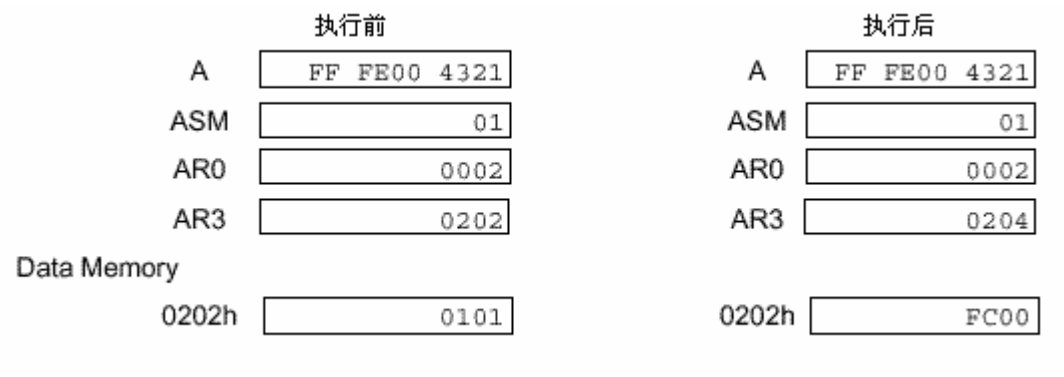
Then (src（15～0）) << (ASM – 16) → Xmem

Else (Xmem) → (Xmem)

状态位　　被 ASM 位和 SXM 位影响
说明

如果满足条件，源累加器 src 左移（ASM－16）位后存放到 Xmem 指定的存储器单元中；如果不满足条件，指令从 Xmem 中读出数据，然后又把它写回到原来的单元中去，即 Xmem 单元的值保持不变。

例 SACCD　A，　*AR3+0%，　ALT



解释

条件 ALT 表示 A 小于 0，累加器 A 满足指令条件，执行将累加器 A 的值移动（1－16）＝－15 位，即右移 15 位，并送到 0202h 地址单元。累加器右移 15 位为 FFFF FC00h，所以将低 16 位 FC00h 送到 0202h。最后，按照 AR0 的内容修改 AR3 为 0204h。

99. SAT　src

操作数　src： A (累加器 A)　 B (累加器 B)
执行　　Saturate (src) → src
状态位　影响 OVsrc 位
说明

无论 OVM 的值是什么，都把源累加器 src 的值饱和成 32 位。所谓饱和，只需要判断累加器的保护位，如果保护位为 00h 或者 FFh，累加器保持不变；如果累加器的保护位是其他数，表示累加器存储的数值超过 32 位的范围，如果是正数，强制将累加器设置成 00 7FFF FFFFh，如果是负数，强制将累加器设置成 FF 8000 0000h。

例　SAT　B

例　SAT　A

例　SAT　B

100. SFTA src， SHIFT [， dst]

操作数 src， dst A (累加器 A) B (累加器 B)

–16 ≤ SHIFT ≤ 15

执行 If SHIFT < 0

Then (src((–SHIFT) – 1)) → C

(src(39～0)) << SHIFT → dst

If SXM = 1

Then (src(39)) → dst(39－(39 + (SHIFT + 1))) [or src(39－(39 + (SHIFT + 1)))]

Else 0 → dst(39～(39 + (SHIFT + 1))) [or src(39～(39 + (SHIFT + 1)))]

Else (src(39 – SHIFT)) → C

(src) << SHIFT → dst

0 → dst((SHIFT – 1)～0) [or src((SHIFT – 1)～0)

状态位 被 SXM 位和 OVM 位影响

影响 C 位和 OVdst 位(如果 dst = src，影响 OVsrc 位)

说明

该指令对源累加器 src 进行算术移位，结果存放在 dst 或者 src。 指令的执行由 SHIFT 的值决定：

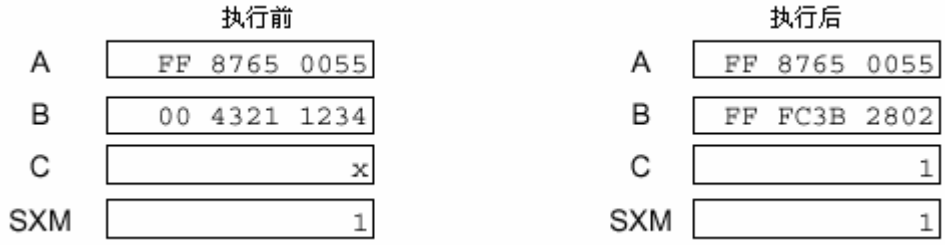
（1）如果 SHIFT 的值小于 0，那么 src((－SHIFT)－1)位复制到进位位 C。

如果 SXM 为 1，指令执行算术右移，源累加器 src 的最高位移入到 dst(39－(39+(SHIFT+1)))位。

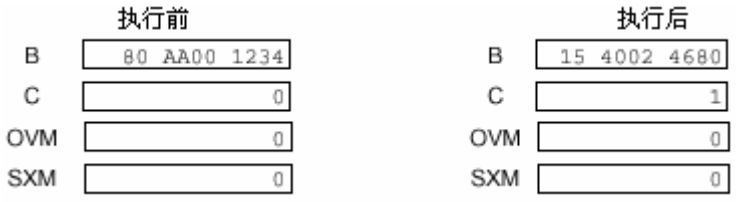
如果 SXM 为 0，dst(39～(39+(SHIFT+1)))位清 0。

（2）如果 SHIFT 的值大于是 0，那么 src(39－SHIFT)位复制到进位位 C。指令产生一个算术左移。dst((SHIFT－1)～0)位清 0

例 SFTA A， –5， B



例 SFTA B， +5



101. SFTC src

操作数　src： A (累加器 A)　B (累加器 B)

执行　If (src) = 0

Then　1→TC

Else

If (src(31)) XOR (src(30)) = 0

Then (最高两位为符号位)

0→TC

(src) << 1→src

Else (最高位为符号位)　1→TC

状态位　影响 TC 位

说明

如果源累加器 src 有两个有效符号位，就把 32 位的 src 左移 1 位，保护位保持不变，且使测试控制位 TC 清 0。如果只有一个符号位，TC 位置 1。 两个有效符号位，即高两位一致，则这两个符合位全为 0 或者全为 1；一个有效符号位，即高两位不一致，则这两个符合位中一个为 1，另一个为 0。

例 SFTC　A

	执行前		执行后
A	FF FFFF F001	A	FF FFFF E002
TC	x	TC	0

102. SFTL　src， SHIFT [， dst]

操作数　src， dst： A (累加器 A)　B (累加器 B)

–16 ≤ SHIFT ≤ 15

执行　If SHIFT < 0

Then　src((–SHIFT) – 1) → C

src(31～0) << SHIFT → dst

0 → dst(39～(31 + (SHIFT + 1)))

If SHIFT = 0

Then　0 → C

Else　src(31 – (SHIFT – 1)) → C

src((31 – SHIFT)～0) << SHIFT → dst

0 → dst((SHIFT – 1)～0)

状态位　影响 C 位

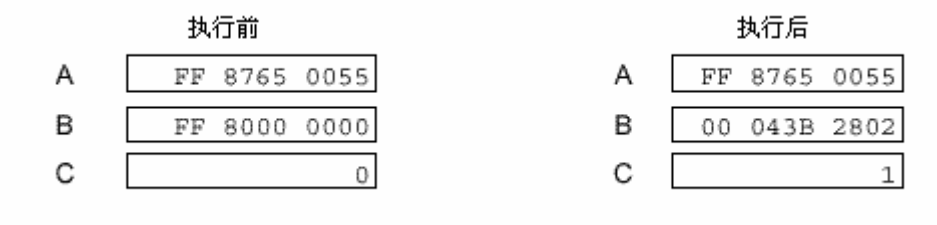
说明

对源累加器 src 进行逻辑移位，结果存放在 dst 或者 src。指令的执行由 SHIFT 的值决定：

（1）如果 SHIFT 的值小于是 0，那么 src（－SHIFT－1）复制到进位位 C。指令执行一个逻辑右移。dst（39～（31＋SHIFT＋1））位清 0。

（2）如果 SHIFT 的值大于 0，那么 src（31－（SHIFT－1）） 复制到进位位 C。指令执行一个逻辑左移。dst((SHIFT－1)～0)位清 0。

例　SFTL　A，　–5，　B



103. SQDST Xmem， Ymem

操作数 Xmem， Ymem： 双数据存储操作数

执行 (A(32～16)) × (A(32～16)) + (B) → B

((Xmem) – (Ymem)) << 16 → A

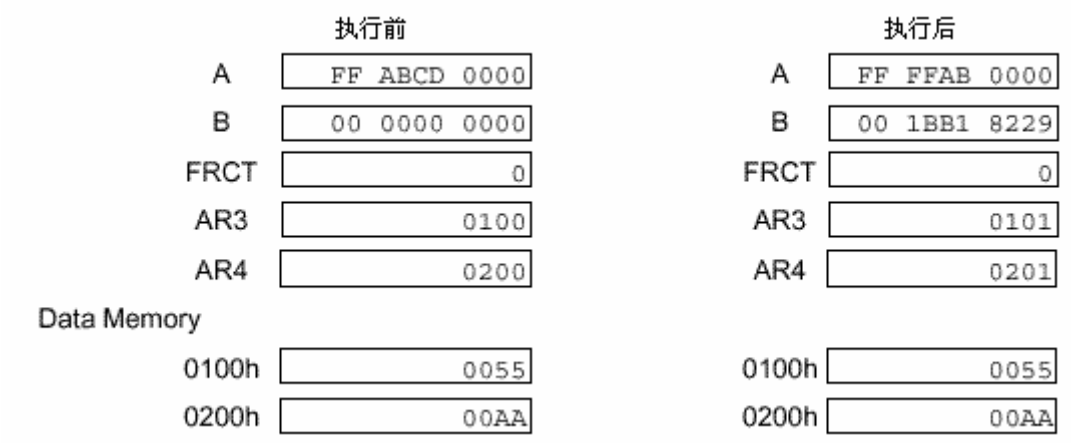
状态位 被 OVM 位、FRCT 位和 SXM 位影响

影响 C 位、OVA 位和 OVB 位

说明

累加器 A 的高端（位 32～16）平方后加到累加器 B 中。Xmem 与 Ymem 相减，结果左移 16 位后存放到累加器 A 中。

例 SQDST *AR3+， AR4+



104. SQUR

（1） SQUR Smem， dst

（2） SQUR A， dst

操作数 Smem： 单数据存储操作数

dst： A (累加器 A) B (累加器 B)

执行（1） (Smem) →T

(Smem) × (Smem) → dst

（2） (A(32～16)) × (A(32～16)) → dst

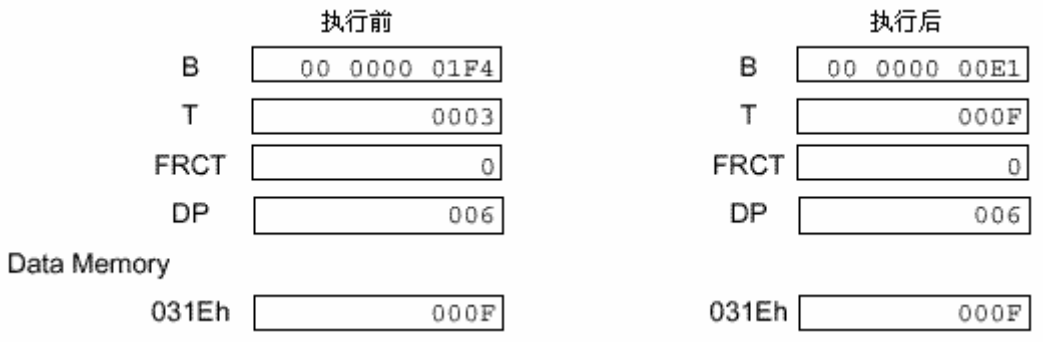
状态位 被 OVM 位和 FRCT 位影响

影响 OVsrc 位

说明

该指令为平方指令。一个单数据存储器操作数 Smem 或累加器 A 的高端（位 32～16）平方后，结果存放在目的累加器 dst 中。如果用的是累加器，T 寄存器就不受影响；如果操作数为 Smem，就要把 Smem 存入 T 寄存器中。

例 SQUR 30， B



例 SQUR A， B

105. SQURA Smem， src

操作数 Smem： 单数据存储操作数

src： A (累加器 A) B (累加器 B)

执行 (Smem) → T

(Smem) × (Smem) ＋ (src) → src

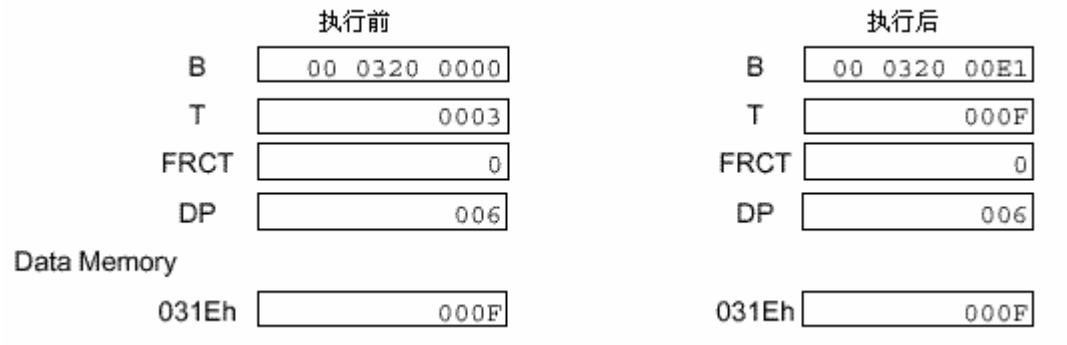
状态位 被 OVM 位和 FRCT 位影响

影响 OVsrc 位

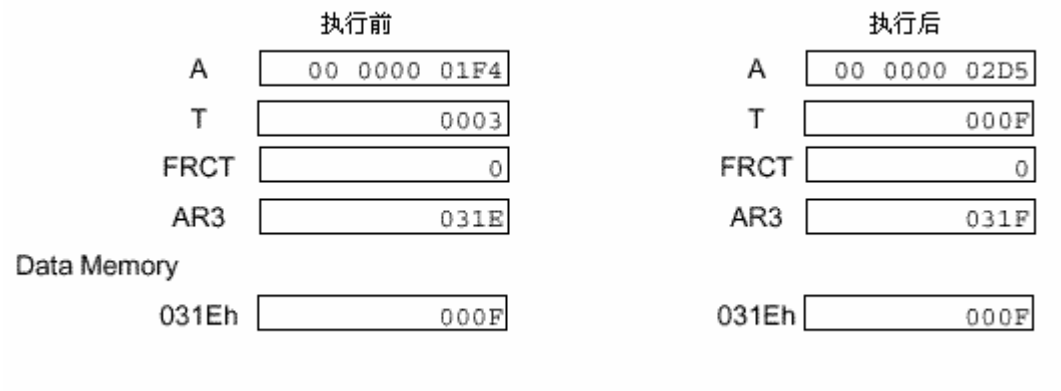
说明

该指令为平方加指令。把数据存储器值 Smem 存放到 T 寄存器中；平方 Smem，再把乘积加到源累加器 src 中。

例 SQURA 30， B



例 SQURA *AR3+， A



106. SQURS Smem， src

操作数 Smem： 单数据存储操作数

src： A (累加器 A) B (累加器 B)

执行 (Smem)→T

(src) – (Smem) × (Smem) → src

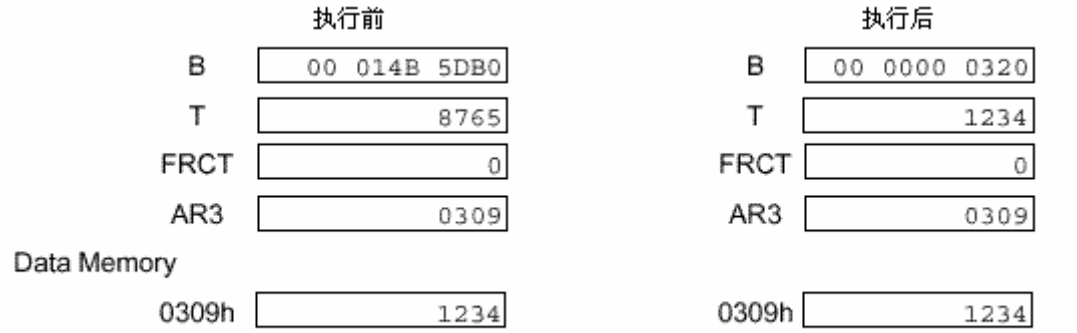
状态位 被 OVM 位和 FRCT 位影响

影响 OVsrc 位

说明

该指令为平方减指令。把数据存储器值 Smem 存放到 T 寄存器中；平方 Smem，再从源累加器 src 中减去这个平方值。

例 SQURS *AR3， B



107. SRCCD Xmem， cond

操作数 Xmem： 双数据存储操作数

执行 If (cond)

Then (BRC) → Xmem

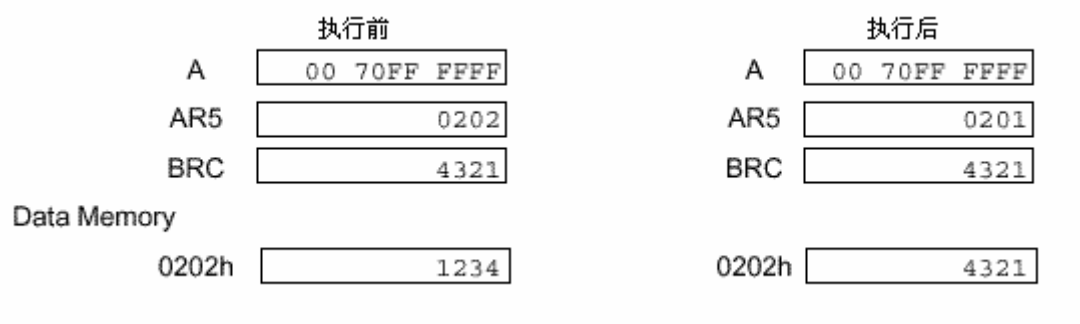
Else (Xmem) → Xmem

状态位 无

说明

如果满足条件，指令把块循环计数器（BRC）中的内容存放到 Xmem 中去，如果不满足条件，指令把 Xmem 中的内容读出再把它写回去，即 Xmem 保持不变。Xmem 内容读出并保存一次，可以动态刷新一次，适用于定时刷新外扩的动态 RAM。

例 SRCCD *AR5–， AGT



108. SSBX N， SBIT

操作数　　0 ≤ SBIT ≤ 15

N = 0 或 1

执行　　1 → STN(SBIT)

状态位　　无

说明

对状态寄存器 ST0 和 ST1 的特定位置 1。N 指明了谁是被修改的状态寄存器（N＝0 表示修改 ST0 寄存器，N＝1 表示修改 ST1 寄存器），SBIT 确定被修改的位。可直接用状态寄存器中的一个特定位的名称作为操作数，这样就不需要使用 N 和 SBIT。注意：该指令不能循环执行。

例　SSBX　SXM　；SXM 表明了为 ST1 寄存器的第 8 位

例　SSBX　1，8　；这两条指令的执行结果完全一样

109. ST

（1）　ST　　T， Smem

（2）　ST　　TRN， Smem

（3）　ST　　# lk， Smem

操作数　　Smem： 单数据存储操作数

–32 768 ≤ lk ≤ 32 767

执行（1）　(T) → Smem

（2）　(TRN) → Smem

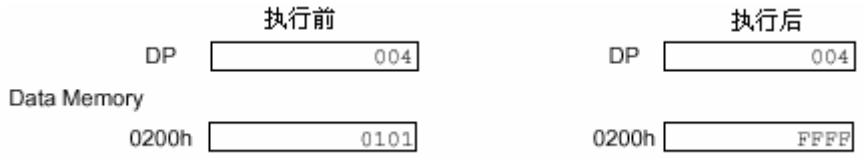
（3）　lk → Smem

状态位　　无

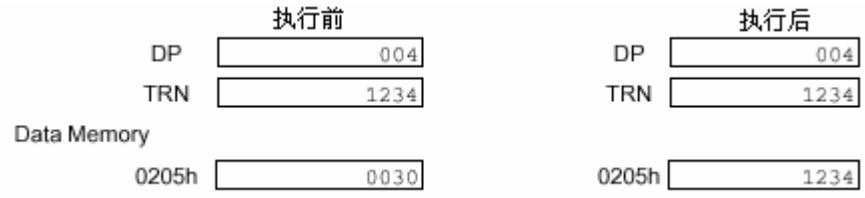
说明

把 T 寄存器的内容、过渡寄存器（TRN）的内容或一个 16 位常数 lk 存放到数据存储器单元 Smem 中去。

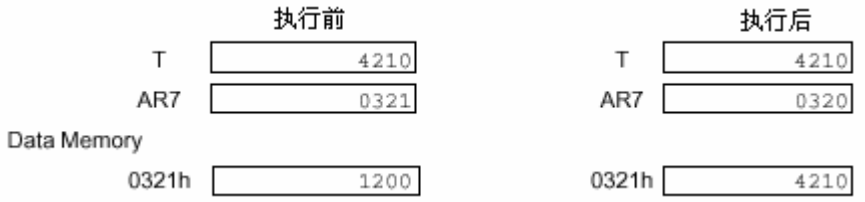
例　ST　0FFFFh， 0



例　ST　TRN，5



例　ST　T，*AR7–



110.　STH

（1）　STH　src，Smem

（2）　STH　src，ASM，Smem

（3）　STH　src，SHFT，Xmem

（4）　STH　src [，SHIFT]，Smem

操作数　　src：A (累加器 A)　B (累加器 B)

Smem：单数据存储操作数

Xmem：双数据存储操作数

0 ≤ SHFT ≤ 15

–16 ≤ SHIFT ≤ 15

执行（1）　(src(31～16)) → Smem

（2）　(src) << (ASM – 16) → Smem

（3）　(src) << (SHFT – 16) → Xmem

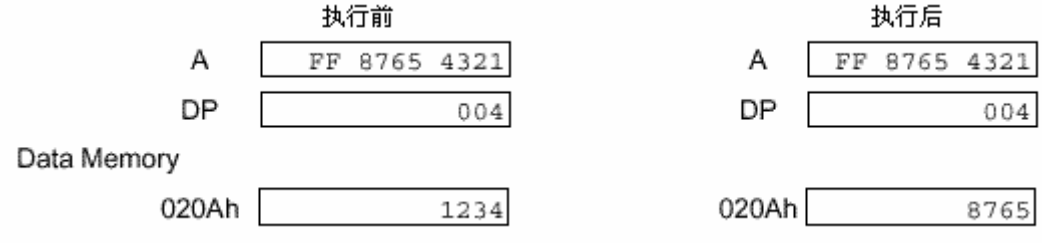
（4）　(src) << (SHIFT – 16) → Smem

状态位　　被 SXM 位影响

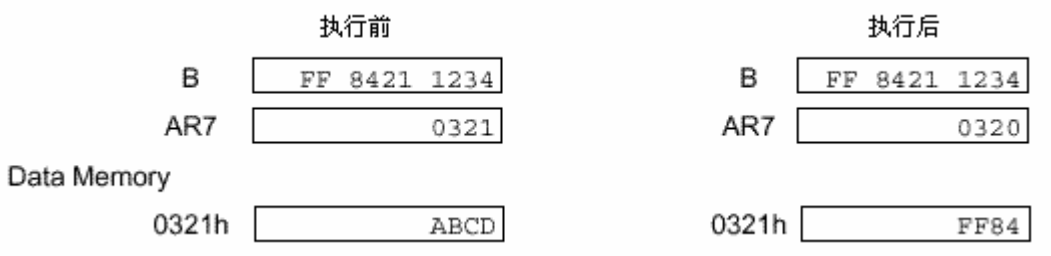
说明

把源累加器 src 的高端（位 31～16）存放到数据存储器单元 Smem 中，src 进行左移，移动位数由 ASM，SHFT 或 SHIFT 决定；然后再把移位后位 31～16 存放到数据存储器单元（Smem 或 Xmem）中。如果 SXM=0，src 的位 39 复制到数据存储器单元的最高位；如果 SXM=1，对移位后的结果进行符号扩展，再将位 39 存放到数据存储器单元的最高位。

例　STH　A，10



例　STH　B，–8，*AR7–



111. STL

（1） STL src， Smem

（2） STL src， ASM， Smem

（3） STL src， SHFT， Xmem

（4） STL src [， SHIFT]， Smem

操作数 src： A (累加器 A) B (累加器 B)

Smem： 单数据存储操作数

Xmem： 双数据存储操作数

0 ≤ SHFT ≤ 15

–16 ≤ SHIFT ≤ 15

执行（1） (src(15～0)) → Smem

（2） (src) << ASM → Smem

（3） (src) << SHFT → Xmem

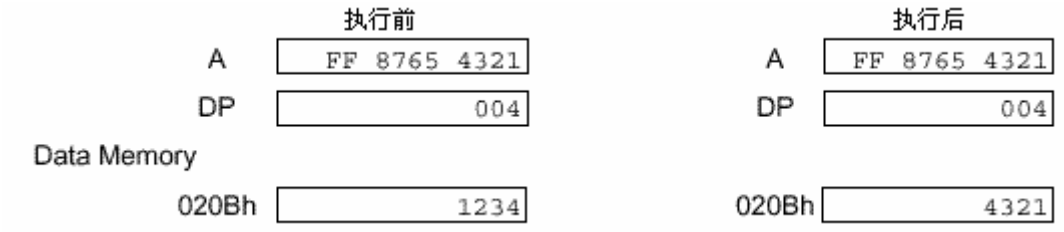
（4） (src) << SHIFT → Smem

状态位 被 SXM 位影响

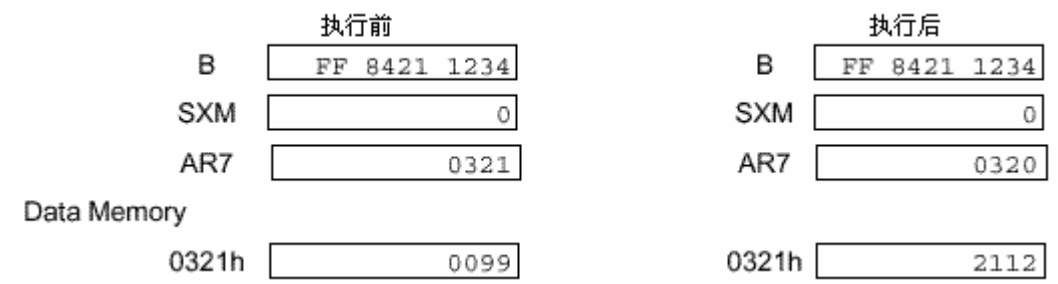
说明

把源累加器 src 的低端（位 15～0）存放到数据存储器单元 Smem 中去，src 执行左移操作，移位数由 ASM、SHFT 或 SHIFT 决定；然后把移位后位 15～0 存放到数据存储器单元 （Smem 或 Xmem）中去。当移位值为正时，低位添 0；当移位值为负时，高位做符号扩展。

例 STL A， 11



例 STL B， –8， *AR7–



112. STLM src， MMR

操作数 src： A (累加器 A) B (累加器 B)

MMR： 存储器映射寄存器

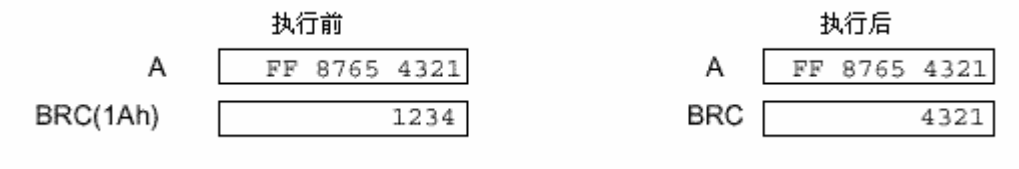
执行 (src(15～0))→MMR

状态位 无

说明

把源累加器 src 的低端(位 15～0)存放到存储器映射寄存器 MMR 中。无论 DP 的当前值或 ARx 的高 9 位是多少，有效地址的高 9 位清 0。指令允许 src 存放在数据页第 0 页中的任何一个存储器单元中而不必修改状态寄存器 ST0 中的 DP 值。

例 STLM A， BRC



例 STLM B， *AR1–

113. STM # lk， MMR

操作数 MMR： 存储器映射寄存器

–32 768 ≤ lk ≤ 32 767

执行 lk → MMR

状态位 无

说明

把一个 16 位常数 lk 存放到一个存储器映射寄存器 MMR，无论 DP 的当前值或 ARx 的高 9 位是多少，都要对有效地址的高 9 位清 0。指令允许 src 存放在数据页第 0 页中的任何一个存储器单元中，而不必修改状态寄存器 ST0 中的 DP 值。

例 STM 0FFFFh， IMR

例 STM 8765h， *AR7+

114. ST src， Ymem

|| ADD Xmem， dst

操作数　src， dst： A (累加器 A)　B (累加器 B)

Xmem， Ymem： 双数据存储操作数

dst_： 如果 dst = A， dst_ = B； 如果 dst = B， then dst_ = A

执行　(src) << (ASM－16) → Ymem

(dst_) ＋ (Xmem) << 16 → dst

状态位　被 OVM 位、SXM 位和 ASM 位影响

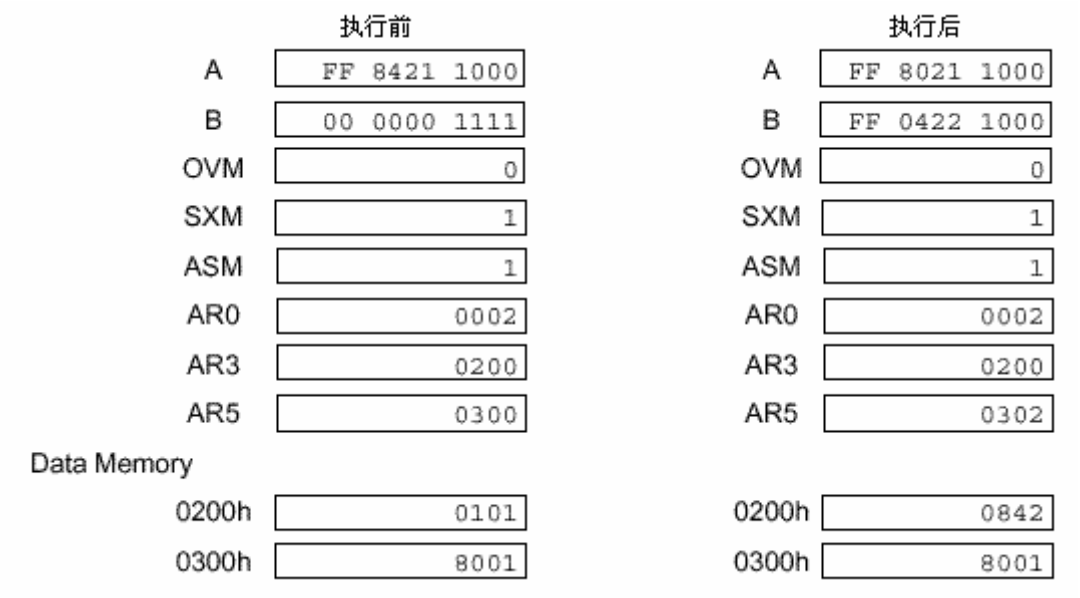
影响 C 位和 OVdst 位

说明

源累加器 src 移动由（ASM－16）所决定的位数，然后存放到数据存储器单元 Ymem 中；同时并行执行 dst_的内容与左移 16 位的存储器操作数 Xmem 相加，结果存放在 dst 中。如果 src 等于 dst，src 的值在指令执行之前就存放到 Ymem 中。

例 ST　A， *AR3

||ADD　*AR5+0%， B



115. ST　|| LD

（1） ST　src， Ymem

|| LD　Xmem， dst

（2） ST　src， Ymem

|| LD　Xmem， T

操作数　src， dst： A (累加器 A)　B (累加器 B)

Xmem， Ymem： 双数据存储操作数

执行（1） (src) << (ASM－16) → Ymem

(Xmem) << 16 → dst

（2） (src) << (ASM－16) → Ymem

(Xmem) → T

状态位　被 OVM 位和 ASM 位影响

影响 C 位

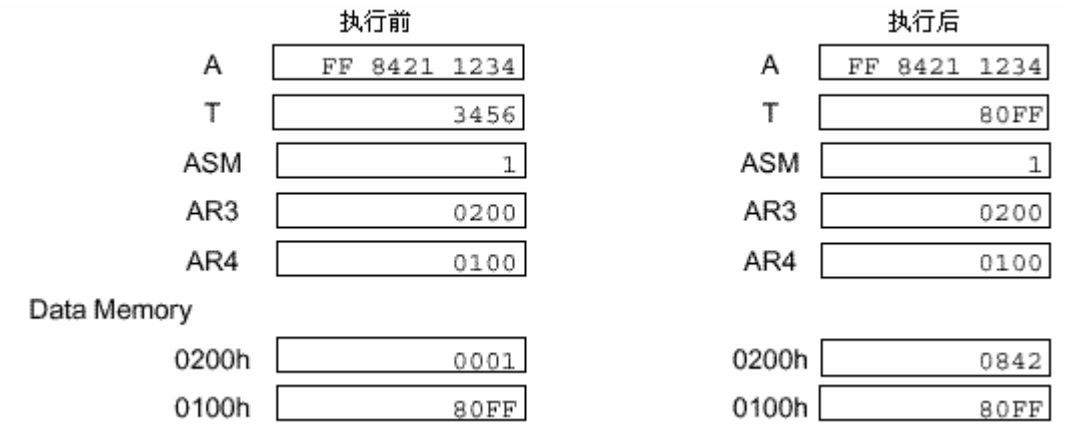
说明

源累加器 src 移动由（ASM－16）所决定的位数，然后把移位后的值存放到数据存储

器单元 Ymem 中；同时并行执行把 16 位双数据存储器操作数 Xmem 装入到目的累加器 dst 或 T 寄存器中。如果 src 等于 dst，则 src 的值在指令执行之前就存放到 Ymem 中。

例　　ST　A，　*AR3

　　||LD　*AR4，　T



116.　ST　　src，　Ymem

||MAC[R]　　Xmem，　dst

操作数　　src，　dst：　A (累加器　A)　B (累加器　B)

　　Xmem，　Ymem：　双数据存储操作数

执行　　(src（31～16）<< (ASM – 16))→Ymem

　　If (Rounding)

　　Then　　Round ((Xmem)　×　(T) + (dst)) → dst

　　Else　　(Xmem)　×　(T) + (dst) → dst

状态位　　被 OVM 位、SXM 位、ASM 位和 FRCT 位影响

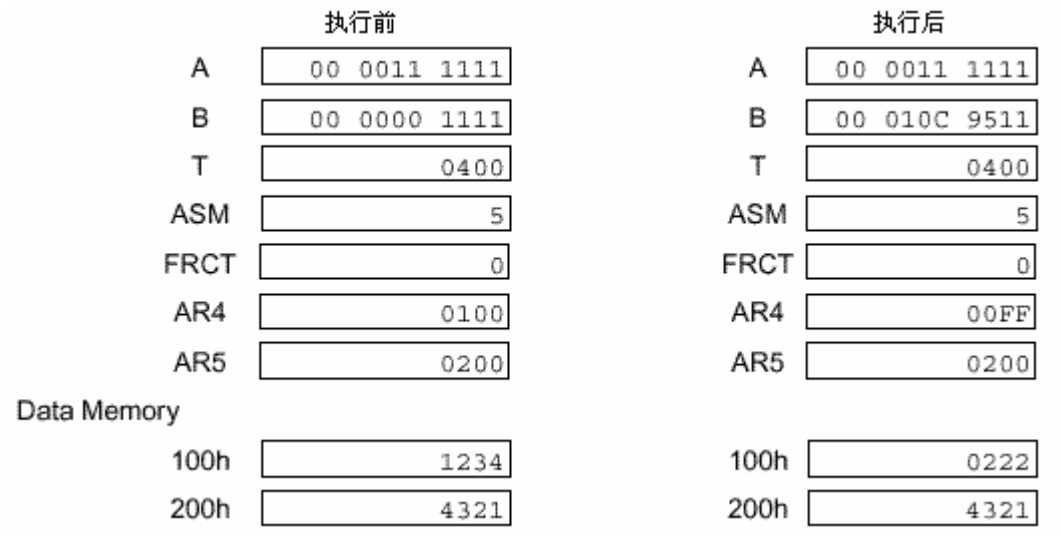
　　影响 C 位和 OVdst 位

说明

　　源累加器 src 移动由（ASM－16）所决定的位数，然后把移位后的值存放到数据存储器单元 Ymem 中；同时并行执行 T 寄存器的值与数据存储器操作数 Xmem 相乘，乘积加到目的累加器 dst（可以进行四舍五入运算）。如果 src 等于 dst，则 src 的值在指令执行之前就存放到 Ymem 中。

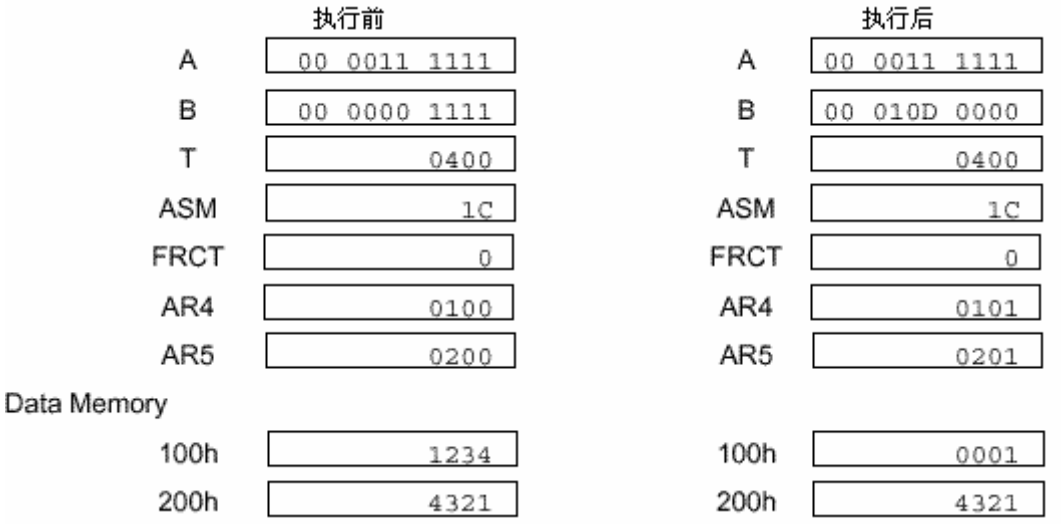
例　ST　A，　*AR4–

　　||MAC　*AR5，　B



例　ST　A，　*AR4+

　　||MACR　*AR5+，　B



117.　ST　　src，　Ymem

　　|| MAS[R]　　Xmem，　dst

操作数　　src，　dst：　A (累加器 A)　B (累加器 B)

　　Xmem，　Ymem：　双数据存储操作数

执行　　(src(31～16) << (ASM－16)) → Ymem

　　If (Rounding)

　　Then　　Round ((dst) – (Xmem) × (T))→ dst

　　Else　　(dst) – (Xmem) × (T) → dst

状态位　　被 OVM 位、SXM 位、ASM 位和 FRCT 位影响

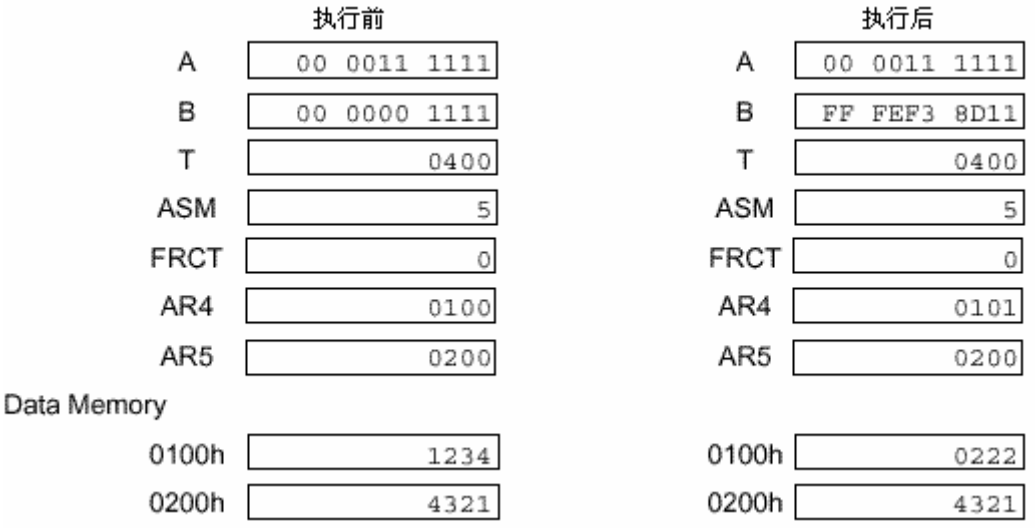
　　影响 C 位和 OVdst 位

说明

　　源累加器 src 移动由（ASM－16）所决定的位数，然后把移位后的值存到数据存储器单元 Ymem；同时并行执行 T 寄存器的值与数据存储器操作数 Xmem 相乘，乘积与目的累加器 dst 相减（可以带四舍五入运算）结果仍然存放在 dst 中。

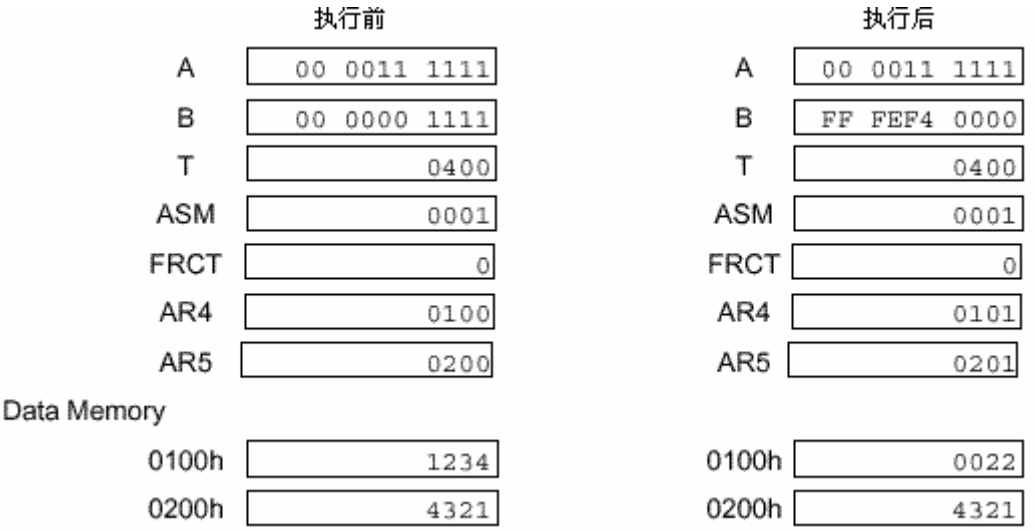
例　ST　A，　*AR4+

|| MAS *AR5， B



例 ST A， *AR4+

|| MASR *AR5+， B



118. ST src， Ymem

|| MPY Xmem， dst

操作数 src， dst： A (累加器 A) B (累加器 B)

Xmem， Ymem： 双数据存储操作数

执行 (src(31～16) << (ASM－16)) → Ymem

(T) × (Xmem) → dst

状态位 被 OVM 位、SXM 位、ASM 位和 FRCT 位影响

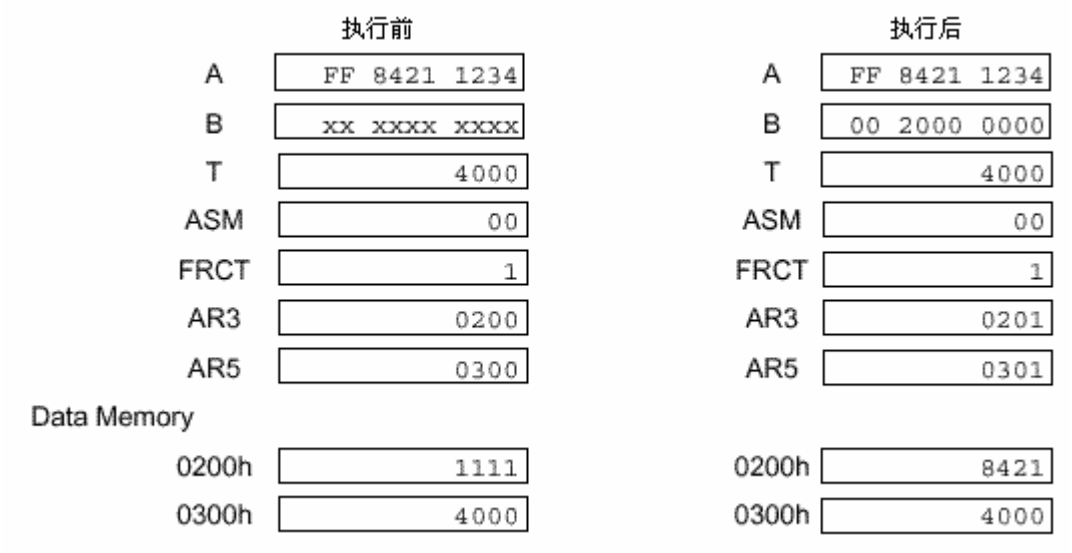
影响 C 位和 OVdst 位

说明

源累加器 src 移动由（ASM－16）所决定的位数，然后把移位后的值存放到数据存储器单元 Ymem 中；同时并行执行 T 寄存器的值与 16 位双数据存储器操作数 Xmem 相乘，乘积存放在 dst 中。

例 ST A， *AR3+

|| MPY *AR5+， B



119. ST src， Ymem

|| SUB Xmem， dst

操作数　　src， dst： A (累加器 A)　B (累加器 B)

Xmem， Ymem： 双数据存储操作数

dst_： 如果 dst = A， then dst_ = B； 如果 dst = B， then dst_ = A.

执行　　(src(31～16) << (ASM－16)) →Ymem

(Xmem) << 16 – (dst_) → dst

状态位　　被 OVM 位、SXM 位和 ASM 位影响

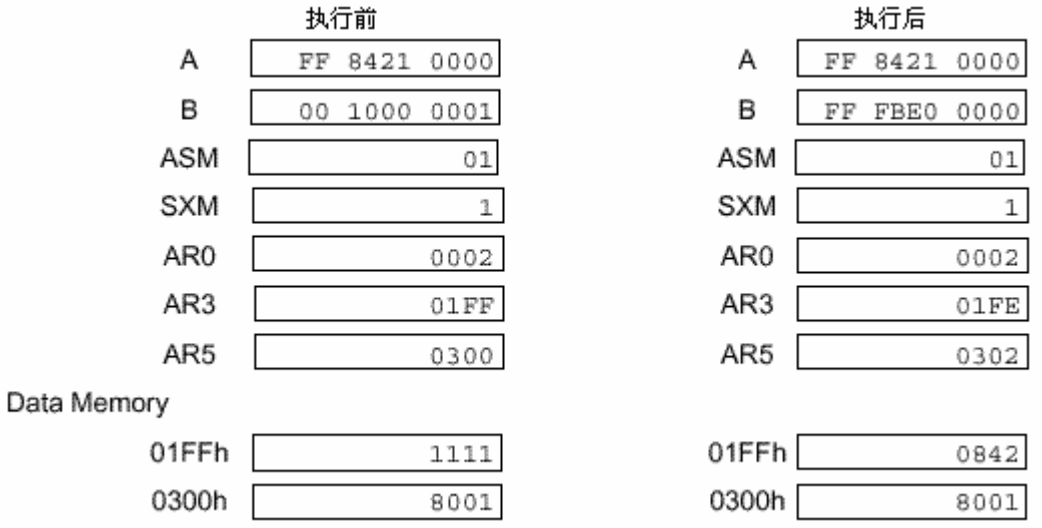
影响 C 位和 OVdst 位

说明

源累加器 src 移动由（ASM－16）所决定的位数，然后把移位后的值存放到数据存储器单元 Ymem 中；同时并行执行从左移了 16 位的双数据存储器操作数 Xmem 中减去 dst_ 的值，并把结果存放在 dst 中。

例　ST　A， *AR3–

|| SUB　*AR5+0%， B



120. STRCD Xmem， cond

操作数　　Xmem： 双数据存储操作数

执行　　If (cond)

　　(T) → Xmem

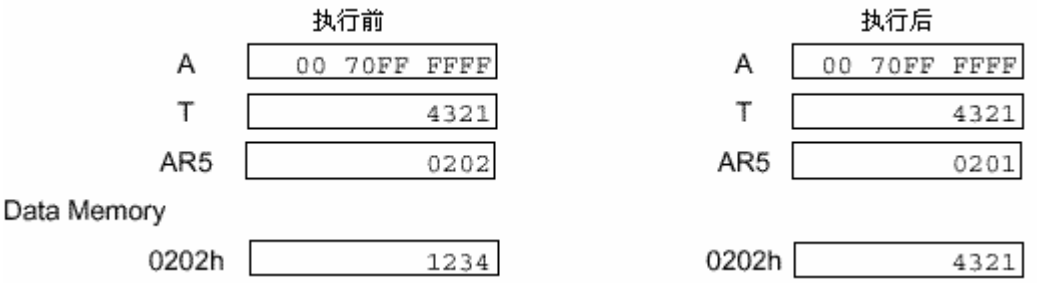
　　Else　　(Xmem) → Xmem

状态位　　无

说明

　　如果满足条件，就把 T 寄存器的值存放到数据存储器单元 Xmem 中去。如果不满足条件，指令从单元 Xmem 中读出数据然后再把它写回到 Xmem 中去，即 Xmem 中的数据保持不变。

例 STRCD　*AR5–，　AGT



121. SUB

（1） SUB　Smem，　src

（2） SUB　Smem，　TS　，　src

（3） SUB　Smem，　16　，　src [　，　dst]

（4） SUB　Smem [　，　SHIFT]　，　src [　，　dst]

（5） SUB　Xmem，　SHFT，　src

（6） SUB　Xmem，　Ymem，　dst

（7） SUB　# lk [　，　SHFT]，　src [　，　dst]

（8） SUB　# lk，　16，　src [　，　dst]

（9） SUB　src [　，　SHIFT]，　[　，　dst]

（10）SUB　src，　ASM [　，　dst]

操作数　　src，　dst： A (累加器 A)　B (累加器 B)

　　Smem： 单数据存储操作数

　　Xmem，　Ymem： 双数据存储操作数

　　–32 768 ≤ lk ≤ 32 767

　　–16 ≤ SHIFT ≤ 15

　　0 ≤ SHFT ≤ 15

执行（1） (src) – (Smem)→src

（2） (src) – (Smem) << TS→src

（3） (src) – (Smem) << 16→dst

（4） (src) – (Smem) << SHIFT→dst

（5） (src) – (Xmem) << SHFT→src

（6） (Xmem) << 16 – (Ymem) << 16→dst

（7） (src) – lk << SHFT→dst

（8） (src) – lk << 16→dst

（9） (dst) – (src) << SHIFT→dst

（10）(dst) – (src) << ASM→dst

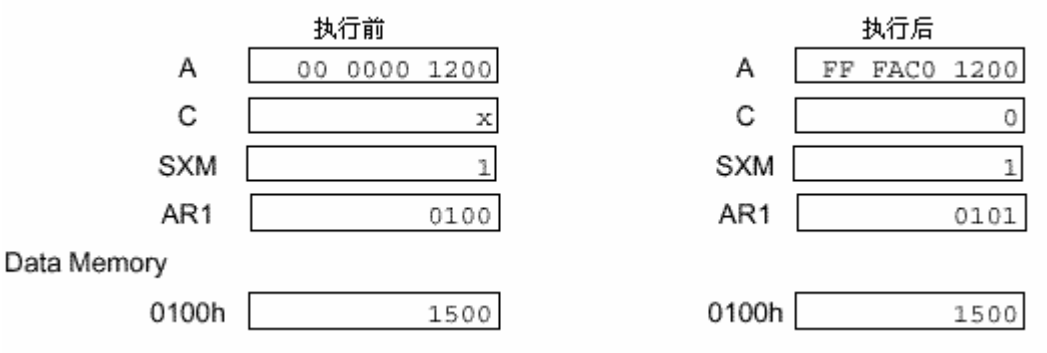
状态位　被 SXM 位和 OVM 位影响

影响 C 位和 OVdst 位(如果 dst = src，影响 OVsrc 位)

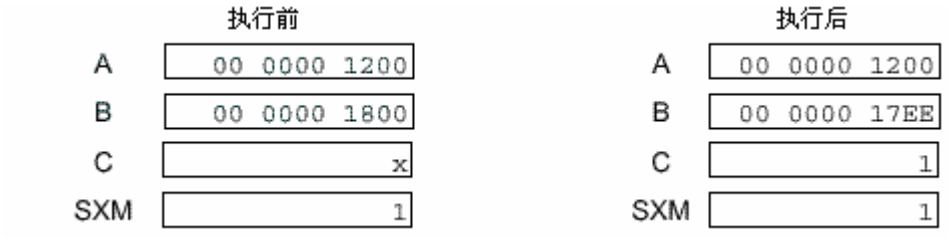
说明

从选定的累加器或采用双数据存储器寻址方式的 16 位操作数 Xmem 中减去一个 16 位的数据，该 16 位数据可以是单数据操作数、双数据操作数、立即数或源累加器移位后的数。如果指定了目的累加器，结果就放在其中，否则结果放在源累加器中。

例　SUB　*AR1+，　14，　A

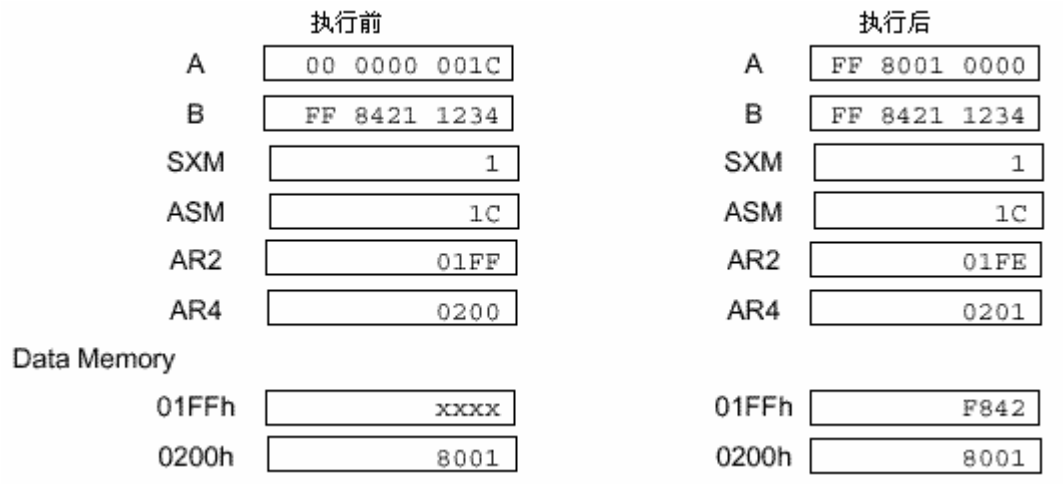


例　SUB　A，　–8，　B



例　ST　B，　*AR2–

|| LD　*AR4+，　A



122.　SUBB　　Smem，　src

操作数　src：　A (累加器 A)　　B (累加器 B)

Smem：　单数据存储操作数

执行　　(src) – (Smem) – (进位位 C 的逻辑反)→src

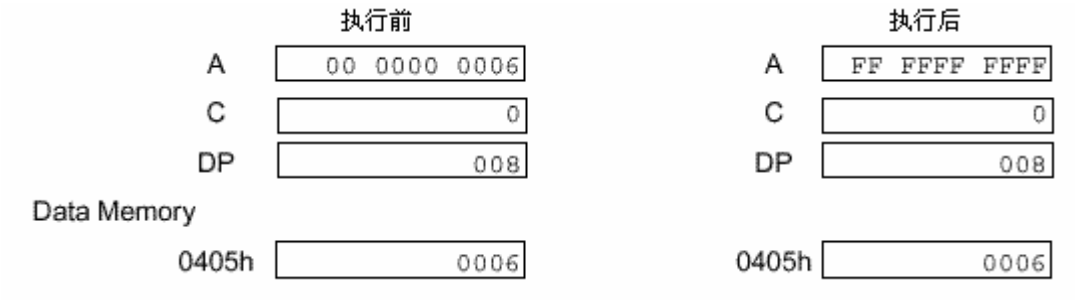
状态位　　被 OVM 位和 C 位影响

影响 C 位和 OVsrc 位

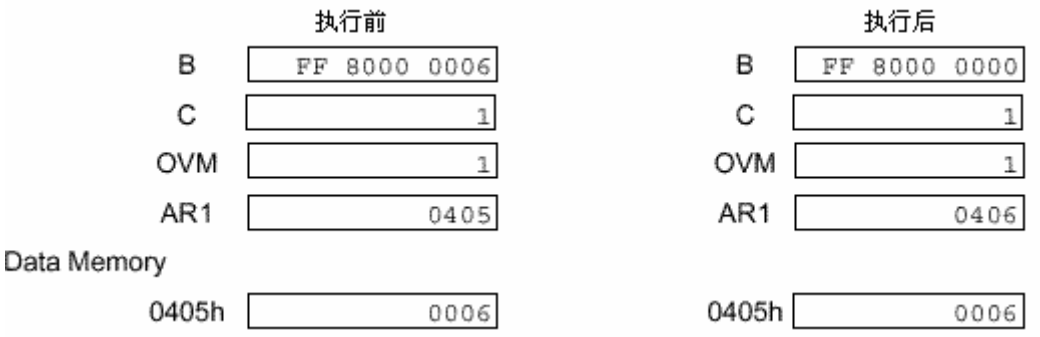
说明

从源累加器 src 中减去 16 位单数据存储器操作数 Smem 的值和进位位 C 的逻辑反。

例 SUBB 5， A



例 SUBB *AR1+， B



123. SUBC Smem， src

操作数 Smem： 单数据存储操作数

src： A (累加器 A) B (累加器 B)

执行 (src) – ((Smem) << 15)→(ALU output)

If ALU output ≥ 0

Then ((ALU output) << 1) + 1 → src

Else (src) << 1 → src

状态位 被 SXM 位影响

影响 C 位和 OVsrc 位

说明

16 位单数据存储器操作数 Smem 左移 15 位，然后再从源累加器 src 中减去移位后的值。如果结果大于 0，就左移一位且加上 1，最后存放到 src 中。否则，只把 src 的值左移一位，存放到 src 中。

除数和被除数在这条指令中都假设为正，SXM 将影响该操作：

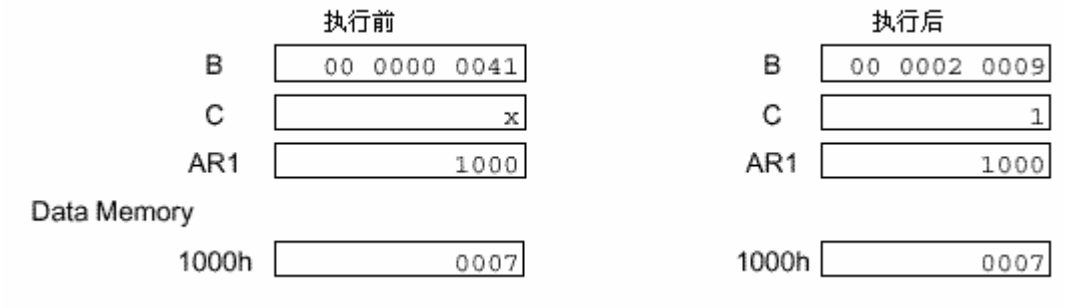
如果 SXM=1 除数的最高位必须为 0。

如果 SXM=0 任何一个 16 位除数值都可以。

src 中的被除数必须初始化为正（位 31 为 0），且在移位后也必须保持为正。该指令影响 OVA 或 OVB，但不受 OVM 影响。所以，当发生溢出时 src 不进行饱和运算。

例 RPT #15

SUBC *AR1， B



解释

该两条指令实现一个整数除法运算。C54xx 系列指令中没有专门的除法指令，所以以循环 16 次减法实现除法运算。该指令实现累加器 B 的值除以 1000h 单元的内容，即 41h÷7h，商为 9，余数为 2（十进制中为 65÷7）。指令运行后，商放在累加器 B 的低 16 位，余数放在累加器 B 的高 16 位。

124. SUBS Smem， src

操作数 Smem： 单数据存储操作数

src： A (累加器 A) B (累加器 B)

执行 unsigned(src) – (Smem)→src

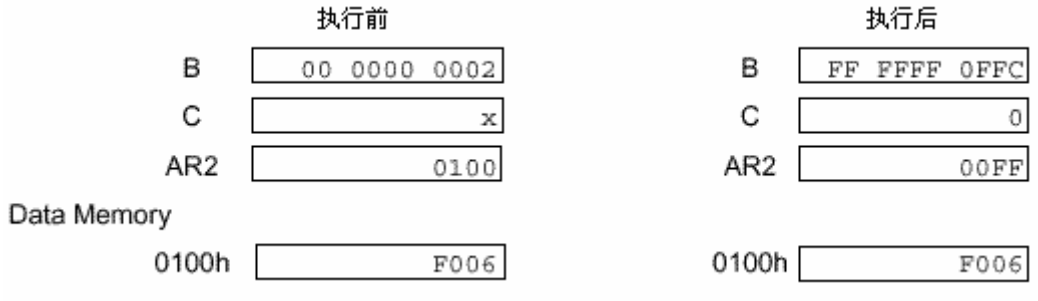
状态位 被 OVM 位影响

影响 C 位和 OVsrc 位

说明

从源累加器 src 中减去 16 位单数据存储器操作数 Smem 的值。无论 SXM 的值为多少，Smem 都被看做是一个 16 位无符号数。

例 SUBS *AR2–， B



125. TRAP K

操作数 0 ≤ K ≤ 31

执行 (SP)－1 → SP

(PC)＋1 → TOS

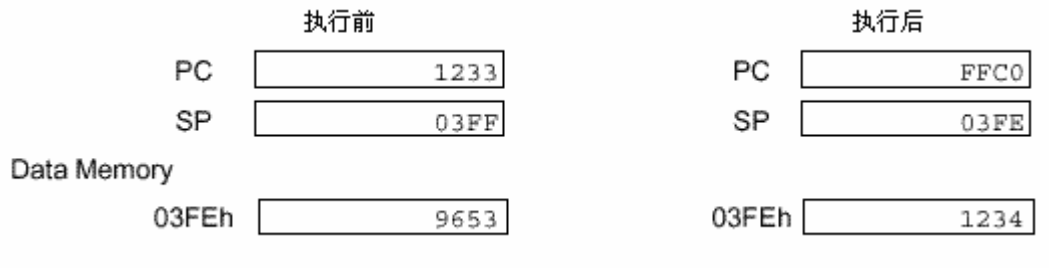
K 确定的中断矢量 → PC

状态位 无

说明

程序指针 PC 指向由 K 确定的中断矢量，指令允许使用软件来执行任何中断服务程序。指令执行时，先把 PC+1 压入由堆栈指针 SP 寻址的数据存储器单元，这使得返回指令在执行完中断服务程序后，能从堆栈 SP 所指的单元中恢复要执行的下一条指令的地址。该指令是不可屏蔽的，不受 INTM 位的影响，它也不影响 INTM 位。

例 TRAP 10h



126. WRITA Smem

操作数　　Smem： 单数据存储操作数

执行　　A→PAR If (RC) ≠ 0

　　Then　　(Smem) → (由 PAR 中的内容所决定的 Pmem 的地址单元)

　　(PAR) + 1 → PAR

　　(RC) – 1 → RC

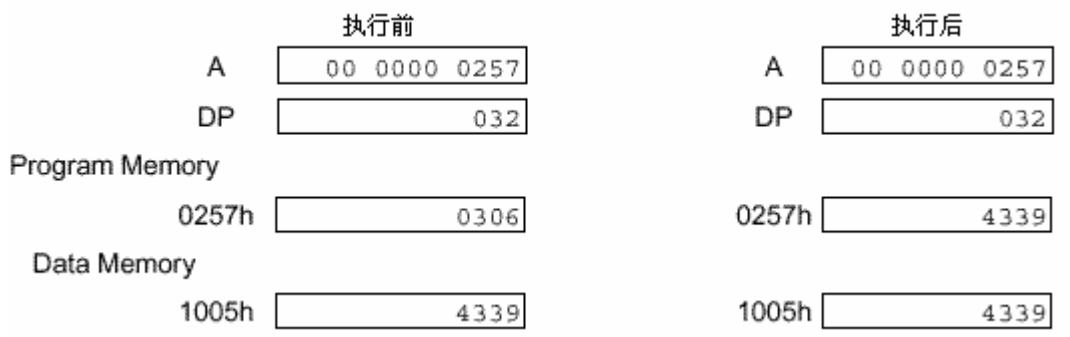
　　Else　　(Smem) →(由 PAR 中的内容所决定的 Pmem 的地址单元)

状态位　　无

说明

把一个字从 Smem 确定的数据存储器单元传送到一个程序存储器单元。程序存储器的地址是由累加器 A 确定，可以通过循环执行该指令把整块数据存储器中的内容连续转移到由 PAR 寻址的连续的程序存储器空间去。该指令和 READA 指令用法相同。

例 WRITA　5



127. XC n， cond [， cond [， cond]]

操作数　n = 1 或 2

执行　　If (cond)

　　Then　　紧接着的下 n 条指令被执行

　　Else　　紧接着执行 n 条 NOP 指令

状态位　无

说明

该指令的运行情况由 n 和所选择的条件决定。如果 n＝1 并且满足条件，就执行该指令的下一条单字指令；如果 n＝2 并且满足条件，就执行该指令的下两条单字指令或者一条双字指令；如果不满足条件，执行 n 次 nop 操作。

例　　XC　　1， ALEQ

　　MAR　*AR1+

　　ADD　 A， DAT100

128. XOR

（1） XOR Smem， src

（2） XOR # lk [， SHFT]， src [， dst]

（3） XOR # lk， 16， src [， dst]

（4） XOR src [， SHIFT] [， dst]

操作数 src， dst： A (累加器 A) B (累加器 B)

Smem： 单数据存储操作数

0 ≤ SHFT ≤ 15

–16≤ SHIFT ≤ 15

0 ≤ lk ≤ 65 535

执行（1） (Smem) XOR (src) → src

（2） lk << SHFT XOR (src) → dst

（3） lk << 16 XOR (src) → dst

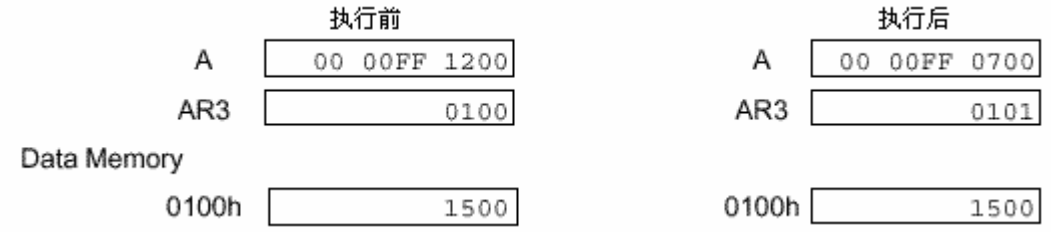
（4） (src) << SHIFT XOR (dst) → dst

状态位 无

说明

16 位单数据存储器操作数 Smem 与所选定的累加器相异或，结果存放在 dst 或 src 中。

例 XOR *AR3+， A



例 XOR A， +3， B

129. XORM # lk， Smem

操作数 Smem： 单数据存储操作数

0 ≤ lk ≤ 65 535

执行 lk XOR (Smem) → Smem

状态位 无

说明

一个数据存储器单元 Smem 的内容和一个 16 位常数 lk 相异或，结果写到 Smem 中。注意，该指令不能循环执行。

例 XORM　0404h，　*AR4–