User's Manual

32

R32C/100 系列

用户手册 软件篇

瑞萨单片机
M16C 族 / R32C/100 系列

瑞萨电子
www.renesas.com

Rev.1.00 2010.08

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

本手册的符号

为了详细说明单片机的运行，本手册按照一定的规则进行记载。
本手册使用以下的符号。

分类	记述	内容
符号	IMM	表示立即数 （Immediate）。
	IMMEX	表示在运算时进行符号扩展 （Extension）的立即数。
	abs	表示绝对寻址的绝对 （Absolute）值。
	dsp	表示相对寻址的位移量 （Displacement）。
	[ ]	表示间接寻址。
	label	表示程序计数器相对寻址使用的标号 （Label）。
	bit	表示位号。
	R0LB、R0B、R2R0B 等	带 “B”的寄存器符号表示组 1 寄存器。
数值	000b	最后带 “b”的数值表示 2 进制数。
	0000h	最后带 “h”的数值表示 16 进制数。
	100	只有数字的数值表示 10 进制数 （BCD）。
位长度指定	#IMM:8 等	下划线的部分表示操作数的有效位数。
	:3	表示有效位长度为 3 位。
	:4	表示有效位长度为 4 位。
	:8	表示有效位长度为 8 位。
	:16	表示有效位长度为 16 位。
	:24	表示有效位长度为 24 位。
	:32	表示有效位长度为 32 位。
指令格式	MOV:S 等	下划线的部分是指定指令格式的符号。
	:G	表示一般格式。
	:Q	表示快速格式。
	:S	表示短格式。
	:Z	表示零格式。
运算长度	MOV.W 等	下划线的部分是指定指令运算长度的符号。
	.B	指定字节 （8bit）运算。
	.W	指定字 （16bit）运算。
	.L	指定长字 （32bit）运算。
	.BW	指定字节到字的转换。
	.BL	指定字节到长字的转换。
	.WL	指定字到长字的转换。
转移距离指定	JMP.A 等	下划线的部分是指定转移相对距离的有效位数的符号。
	.S	表示 3 位 PC 前方相对，有效值为 1 ～ 8。
	.B	表示 8 位 PC 相对，有效值为 –128 ～ +127。
	.W	表示 16 位 PC 相对，有效值为 –32768 ～ +32767。
	.A	表示 24 位 PC 相对，有效值为 –8388608 ～ +8388607。

分类	记述	内容
操作		原则上遵循 C 语言的语法规则。以下说明本手册中使用的记述。
	=	这是赋值运算符，将右边的值赋给左边。
	–	表示一元运算符的负号或者二元运算符的 “减”。
	+	表示二元运算符的 “加”。
	*	表示指针运算符或者二元运算符的 “乘”。
	/	表示二元运算符的 “除”。
	%	表示二元运算符的 “模”。
	~	表示一元位运算符的 “非”。
	&	表示二元位运算符的 “与”。
	|	表示二元位运算符的 “或”。
	^	表示二元位运算符的 “异或”。
	(float)	这是类型转换运算符。
	;	表示语句的结束。
	{ }	表示多个语句的开始和结束，能在 { } 内记述多个语句。
	if (表达式) 语句 1 else 语句 2	表示 if 语句。评估表达式，当表达式是真时，执行语句 1 ； 当表达式是假时，执行语句 2。
	for (语句 1; 表达式 ; 语句 2) 语句 3	表示 for 语句。在执行语句 1 后评估表达式，当表达式是真时，执行语句 3。在执行语句 3 后执行语句 2，然后评估表达式。
	do 语句 while (表达式);	表示 do 语句，在表达式为真的期间执行语句。与表达式的真伪无关，至少执行 1 次语句。
	while (表达式) 语句	表示 while 语句，在表达式为真的期间执行语句。
	==、 !=	这是比较运算符，分别表示 “等于”和 “不等于”。
	>、 <	这是比较运算符，分别表示 “大于”和 “小于”。
	>=、 <=	这是比较运算符。给 “>”、“<” 加上 “==” 的条件。
	&&、 ||	这是逻辑运算符。左侧的条件和右侧的条件的逻辑运算分别为 “与”和 “或”的逻辑运算。
	true	不是 C 语言的符号。当用 *Cnd* 指定的条件成立时，为 “1”。
可选择的 src/dest	#IMMEX	能指定 #IMMEX:8 或者 #IMMEX:16。
	#IMM	能根据运算长度指定 #IMM:8、 #IMM:16 或者 #IMM:32。 不包括 #IMM:3 和 #IMM:4。

目　录

R32C/100 系列指令一览表

表 1　　字母 - 地址速查表 （1 / 5）

助记符	内容	记载功能的页	记载指令码 / 周期数的页
ABS	绝对值	35	169
ADC	带进位的加法运算	36	169
ADCF	进位标志的加法运算	37	170
ADD	不带进位的加法运算	38	171
ADDF	浮点加法运算	40	173
ADSF	符号标志的加法运算	41	174
AND	逻辑与	42	175
BCLR	位清除	44	176
BITINDEX	位变址	45	177
BM*Cnd*	条件位传送	46	177
BMC	C 标志为 “1” 时	46	177
BMEQ	等于时	46	177
BMGE	等于或者带符号的大于时	46	177
BMGEU	大于等于时	46	177
BMGT	带符号的大于时	46	177
BMGTU	大于时	46	177
BMLE	等于或者带符号的小于时	46	177
BMLEU	小于等于时	46	177
BMLT	带符号的小于时	46	177
BMLTU	小于时	46	177
BMN	为负时	46	177
BMNC	C 标志为 “0” 时	46	177
BMNE	不等于时	46	177
BMNO	O 标志为 “0” 时	46	177
BMNZ	Z 标志为 “0” 时	46	177
BMO	O 标志为 “1” 时	46	177
BMPZ	大于等于 “0” 时	46	177
BMZ	Z 标志为 “1” 时	46	177
BNOT	位取反	47	178
BRK	调试中断	48	178
BRK2	调试中断 2	49	178
BSET	位置位	50	179
BTST	位测试	51	179
BTSTC	位测试和清除	52	180
BTSTS	位测试和置位	53	180
CLIP	钳制	54	181
CMP	比较	55	181
CMPF	浮点比较	57	183

表 1　　字母 - 地址速查表 （2 / 5）

助记符	内容	记载功能的页	记载指令码 / 周期数的页
CNVIF	整数 → 浮点数的转换	58	184
DADC	带进位的 10 进制加法运算	59	185
DADD	10 进制加法运算	60	186
DEC	递减	61	187
DIV	带符号的除法运算	62	188
DIVF	浮点除法运算	63	189
DIVU	不带符号的除法运算	64	190
DIVX	带符号的除法运算	65	191
DSBB	带借位的 10 进制减法运算	66	192
DSUB	10 进制减法运算	67	193
EDIV	带符号的除法运算	68	194
EDIVU	不带符号的除法运算	69	194
EDIVX	带符号的除法运算	70	191
EMUL	带符号的乘法运算	71	195
EMULU	不带符号的乘法运算	72	196
ENTER	栈帧的生成	73	196
EXITD	栈帧的释放	74	197
EXITI	中断栈帧的释放	75	197
EXTS	符号扩展	76	197
EXTZ	零扩展	77	199
FCLR	标志清除	79	201
FREIT	从高速中断的返回	80	201
FSET	标志置位	81	201
INC	递增	82	202
INDEX*Type*	变址	83	202
INDEXB	对第 1 个和第 2 个操作数进行加法运算	83	203
INDEX1	对第 1 个操作数进行加法运算	83	202
INDEX2	对第 2 个操作数进行加法运算	83	203
INT	INT 指令中断	84	204
INTO	上溢中断	85	204
J*Cnd*	条件转移	86	204
JC	标志为 “1” 时	86	204
JEQ	等于时	86	204
JGE	等于或者带符号的大于时	86	204
JGEU	大于等于时	86	204
JGT	带符号的大于时	86	204
JGTU	大于时	86	204
JLE	等于或者带符号的小于时	86	204
JLEU	小于等于时	86	204
JLT	带符号的小于时	86	204

表 1　　字母 - 地址速查表 （3 / 5）

助记符	内容	记载功能的页	记载指令码 / 周期数的页
JLTU	小于时	86	204
JN	为负时	86	204
JNC	C 标志为 “0” 时	86	204
JNE	不等于时	86	204
JNO	O 标志为 “0” 时	86	204
JNZ	Z 标志为 “0” 时	86	204
JO	O 标志为 “1” 时	86	204
JPZ	大于等于 “0” 时	86	204
JZ	Z 标志为 “1” 时	86	204
JMP	无条件转移	87	205
JMPI	间接转移	88	206
JSR	子程序转移	89	206
JSRI	间接子程序转移	90	207
LDC	向专用寄存器的传送	91	208
LDCTX	上下文恢复	92	209
LDIPL	中断优先级的设定	94	209
MAX	最大值的选择	95	210
MIN	最小值的选择	96	211
MOV	传送	97	212
MOVA	有效地址传送	99	216
MOV*Dir*	4 位数据传送	100	216
MOVHH	从 src 的高位传送到 dest 的高位	100	216
MOVHL	从 src 的高位传送到 dest 的低位	100	216
MOVLH	从 src 的低位传送到 dest 的高位	100	216
MOVLL	从 src 的低位传送到 dest 的低位	100	216
MUL	带符号的乘法运算	101	217
MULF	浮点乘法运算	102	218
MULU	不带符号的乘法运算	103	219
MULX	带舍入的乘法运算	104	220
NEG	符号取反	106	221
NOP	空操作	107	221
NOT	逻辑取反	108	221
OR	逻辑或	109	222
POP	寄存器 / 存储器恢复	110	223
POPC	专用寄存器恢复	111	223
POPM	多个寄存器恢复	112	223
PUSH	寄存器 / 存储器 / 立即数保存	113	224
PUSHA	有效地址保存	115	225
PUSHC	专用寄存器保存	116	226
PUSHM	多个寄存器保存	117	226

表 1　　字母 - 地址速查表 （4 / 5）

助记符	内容	记载功能的页	记载指令码 / 周期数的页
REIT	从中断的返回	118	226
RMPA	乘加运算	119	227
ROLC	带进位的左循环	120	227
RORC	带进位的右循环	121	227
ROT	循环	122	228
ROUND	浮点数 → 整数的转换	123	229
RTS	从子程序的返回	124	230
SBB	带借位的减法运算	125	230
SC*Cnd*	条件设定	126	231
SCC	标志为 “1”时	126	231
SCEQ	等于时	126	231
SCGE	等于或者带符号的大于时	126	231
SCGEU	大于等于时	126	231
SCGT	带符号的大于时	126	231
SCGTU	大于时	126	231
SCLE	等于或者带符号的小于时	126	231
SCLEU	小于等于时	126	231
SCLT	带符号的小于时	126	231
SCLTU	小于时	126	231
SCN	为负时	126	231
SCNC	C 标志为 “0”时	126	231
SCNE	不等于时	126	231
SCNO	O 标志为 “0”时	126	231
SCNZ	Z 标志为 “0”时	126	231
SCO	O 标志为 “1”时	126	231
SCPZ	大于等于 “0”时	126	231
SCZ	Z 标志为 “1”时	126	231
SCMPU	字符串比较	127	232
SHA	算术移位	128	232
SHL	逻辑移位	129	233
SIN	字符串输入	131	235
SMOVB	反向字符串传送	132	235
SMOVF	正向字符串传送	133	235
SMOVU	字符串传送	134	236
SOUT	字符串输出	135	236
SSTR	字符串保存	136	236
STC	从专用寄存器的传送	137	237
STCTX	上下文保存	138	238
STNZ	带条件的传送	140	238
STOP	停止	141	238

表 1　　字母 - 地址速查表 （5 / 5）

助记符	内容	记载功能的页	记载指令码 / 周期数的页
STZ	带条件的传送	142	239
STZX	带条件的传送	143	239
SUB	不带借位的减法运算	144	240
SUBF	浮点减法运算	145	241
SUNTIL	字符串搜索	146	242
SWHILE	字符串搜索	147	242
TST	测试	148	242
UND	未定义指令中断	149	243
WAIT	等待	150	243
XCHG	交换	151	244
XOR	逻辑异或	152	244

表 2　　功能 - 地址速查表 （1 / 3）

功能	助记符	内容	记载功能的页	记载指令码 / 周期数的页
传送	MOV	传送	97	212
	MOVA	有效地址传送	99	216
	MOVDir	4 位数据传送	100	216
	POP	寄存器 / 存储器恢复	110	223
	POPM	多个寄存器恢复	112	223
	PUSH	寄存器 / 存储器 / 立即数保存	113	224
	PUSHA	有效地址保存	115	225
	PUSHM	多个寄存器保存	117	226
	STNZ	带条件的传送	140	238
	STZ	带条件的传送	142	239
	STZX	带条件的传送	143	239
	XCHG	交换	151	244
位处理	BCLR	位清除	44	176
	BITINDEX	位变址	45	177
	BMCnd	条件位传送	46	177
	BNOT	位取反	47	178
	BSET	位置位	50	179
	BTST	位测试	51	179
	BTSTC	位测试和清除	52	180
	BTSTS	位测试和置位	53	180
移位	ROLC	带进位的左循环	120	227
	RORC	带进位的右循环	121	227
	ROT	循环	122	228
	SHA	算术移位	128	232
	SHL	逻辑移位	129	233
算术运算	ABS	绝对值	35	169
	ADC	带进位的加法运算	36	169
	ADCF	进位标志的加法运算	37	170
	ADD	不带进位的加法运算	38	171
	ADSF	符号标志的加法运算	41	174
	CLIP	钳制	54	181
	CMP	比较	55	181
	DEC	递减	61	187
	DIV	带符号的除法运算	62	188
	DIVU	不带符号的除法运算	64	189
	DIVX	带符号的除法运算	65	191
	EDIV	带符号的除法运算	68	194
	EDIVU	不带符号的除法运算	64	194
	EDIVX	带符号的除法运算	65	195

表 2　　功能 - 地址速查表 （2 / 3）

功能	助记符	内容	记载功能的页	记载指令码 / 周期数的页
算术运算	EMUL	带符号的乘法运算	71	195
	EMULU	不带符号的乘法运算	72	196
	EXTS	符号扩展	76	197
	EXTZ	零扩展	86	199
	INC	递增	77	202
	MAX	最大值的选择	95	210
	MIN	最小值的选择	96	211
	MUL	带符号的乘法运算	101	217
	MULU	不带符号的乘法运算	103	219
	MULX	带舍入的乘法运算	104	220
	NEG	符号取反	106	221
	SBB	带借位的减法运算	125	230
	SUB	不带借位的减法运算	144	240
10 进制运算	DADC	带进位的 10 进制加法运算	59	185
	DADD	10 进制加法运算	60	186
	DSBB	带借位的 10 进制减法运算	66	192
	DSUB	10 进制减法运算	67	193
浮点数运算	ADDF	浮点加法运算	40	173
	CMPF	浮点比较	57	183
	CNVIF	整数 → 浮点数的转换	58	184
	DIVF	浮点除法运算	63	189
	MULF	浮点乘法运算	102	218
	ROUND	浮点数 → 整数的转换	123	229
	SUBF	浮点减法运算	145	241
乘加运算	RMPA	乘加运算	119	227
逻辑运算	AND	逻辑与	42	175
	NOT	逻辑取反	108	221
	OR	逻辑或	109	222
	TST	测试	148	242
	XOR	逻辑异或	152	244
转移	JCnd	条件转移	86	204
	JMP	无条件转移	87	205
	JMPI	间接转移	88	206
	JSR	子程序转移	89	206
	JSRI	间接子程序转移	90	207
	RTS	从子程序的返回	124	230

表 2　　功能 - 地址速查表 （3 / 3）

功能	助记符	内容	记载功能的页	记载指令码 / 周期数的页
字符串	SCMPU	字符串比较	127	232
	SIN	字符串输入	131	235
	SMOVB	反向字符串传送	132	235
	SMOVF	正向字符串传送	133	235
	SMOVU	字符串传送	134	236
	SOUT	字符串输出	135	236
	SSTR	字符串保存	136	236
	SUNTIL	字符串搜索	146	242
	SWHILE	字符串搜索	147	242
专用寄存器的操作	FCLR	标志清除	79	201
	FSET	标志置位	81	201
	LDC	向专用寄存器的传送	91	208
	POPC	专用寄存器恢复	111	223
	PUSHC	专用寄存器保存	116	226
	STC	从专用寄存器的传送	137	237
中断	BRK	调试中断	48	178
	BRK2	调试中断 2	49	178
	FREIT	从高速中断的返回	80	201
	INT	INT 指令中断	82	204
	INTO	上溢中断	85	204
	LDIPL	中断优先级的设定	94	209
	REIT	从中断的返回	118	226
	UND	未定义指令中断	149	243
高级语言的支持	ENTER	栈帧的生成	73	196
	EXITD	栈帧的释放	74	197
	EXITI	中断栈帧的释放	75	197
OS 的支持	LDCTX	上下文恢复	92	209
	STCTX	上下文保存	138	238
其他	INDEXType	变址	83	203
	NOP	空操作	107	221
	SCCnd	条件设定	126	231
	STOP	停止	141	238
	WAIT	等待	150	243

1. 概要

1.1 R32C/100 系列的特点

R32C/100 系列是以嵌入式设备为对象而开发的单片机。

此系列有适合 C 语言的指令，并且将常用指令分配在 1 字节的操作码，因此无论是使用汇编语言还是使用 C 语言，都能开发出存储容量小而且效率高的程序。另外，有很多以 1 个时钟周期执行的指令，实现了更高速的运算处理。

此系列有对应丰富的寻址方式的 108 种指令的指令集，能进行寄存器 - 寄存器、寄存器 - 存储器、存储器 - 存储器之间的运算，并且能进行以位和 4 位数据为对象的运算。

另外，由于内置了乘法运算器和浮点运算器，所以能进行高速运算。

(1) 特点

(a) 寄存器结构

数据寄存器	32 位 ×4 个 （可用作 16 位寄存器，其中 2 个还可用作 8 位寄存器）
地址寄存器	32 位 ×4 个
基址寄存器	32 位 ×2 个

(b) 有特点的指令集

适合 C 语言的指令 （栈帧操作）	： ENTER、 EXITD 等
不区分寄存器和存储器的指令	： MOV、 ADD、 SUB 等
强大的位处理指令	： BCLR、 BTST、 BSET 等
4 位传送指令	： MOVLL、 MOVHL 等
常用的 1 字节指令	： MOV、 ADD、 SUB、 JMP 等
高速的 1 个周期指令	： MOV、 ADD、 SUB 等
定点乘法运算指令	： MULX
浮点运算指令	： ADDF、 SUBF、 MULF、 DIVF 等

(c) 4G 字节的线性地址空间

对应转移距离的相对转移指令

(d) 高速的指令执行时间

最短 1 个周期指令 ： 108 种指令中有 36 种指令为 1 个周期指令。

(2) 性能 （运行频率为 100MHz 的情况）

寄存器之间的传送	10ns
寄存器 - 存储器之间的传送	20ns
寄存器之间的加减法运算	10ns
8 位 ×8 位寄存器之间的运算	20ns
16 位 ×16 位寄存器之间的运算	20ns
32 位 ×32 位寄存器之间的运算	20ns
16 位 ÷16 位寄存器之间的运算	150ns
32 位 ÷32 位寄存器之间的运算	220ns

1.2 地址空间

地址空间如图 1.1 所示。

地址 00000000h ～地址 000003FFh 和地址 00040000h ～地址 0004FFFFh 为 SFR （专用功能寄存器）区。

地址 00000400h ～地址 0003FFFFh、地址 00060000h ～地址 0007FFFFh 和地址 FFE00000h ～地址 FFFFFFFFh 为存储区。在产品种类的展开中， RAM 区从地址 00000400h 向高地址方向扩展， ROM 区从地址 FFFFFFFFh 向低地址方向扩展。但是，地址 FFFFFFDCh ～地址 FFFFFFFFh 为固定向量区。

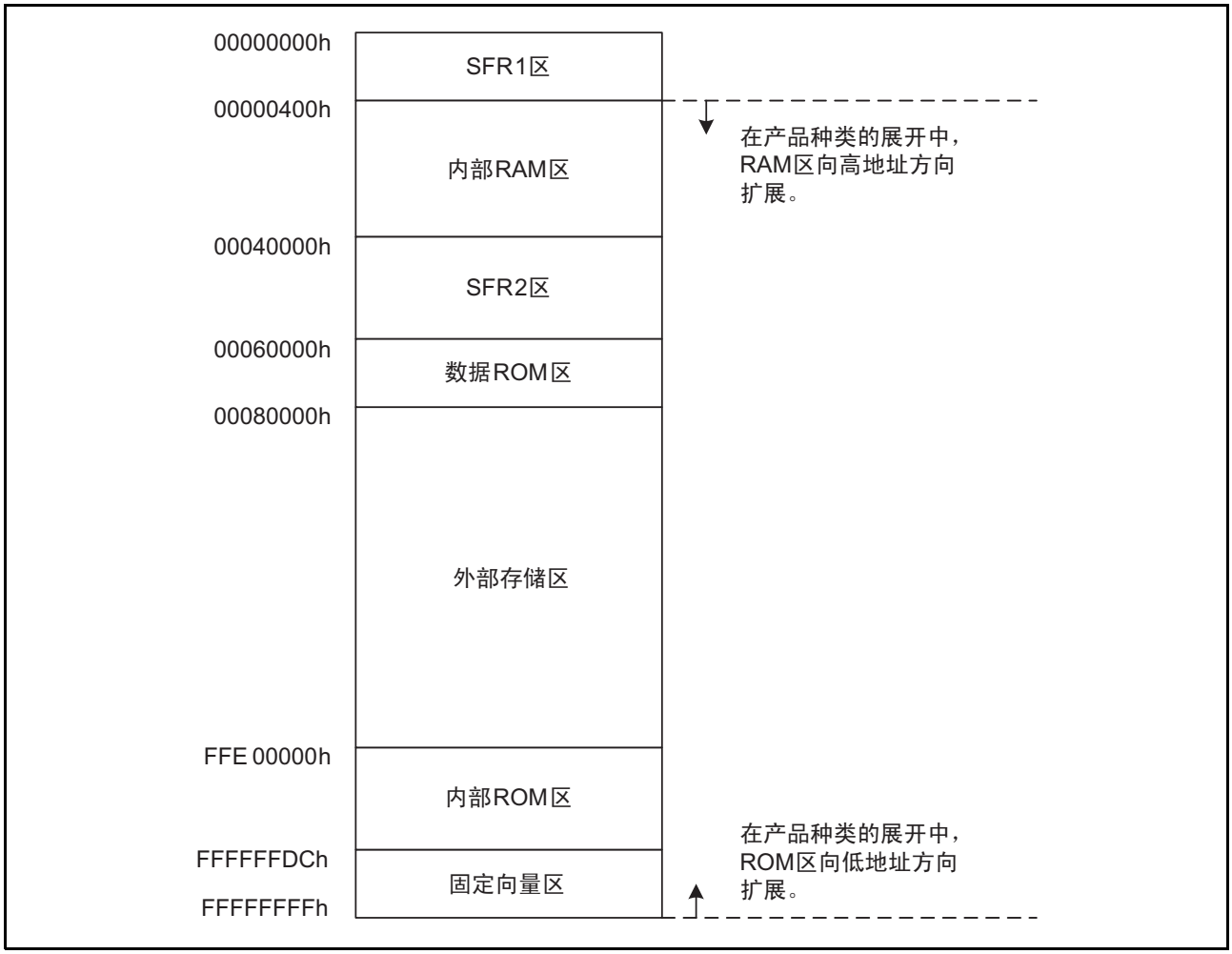


图 1.1 地址空间

1.3 寄存器结构

中央处理器有如图 1.2 所示的寄存器。其中，R2R0、R3R1、R6R4、R7R5、A0、A1、A2、A3、SB、FB 共 10 个寄存器有 2 组，由寄存器组指定标志（标志寄存器的 B 标志）指定。

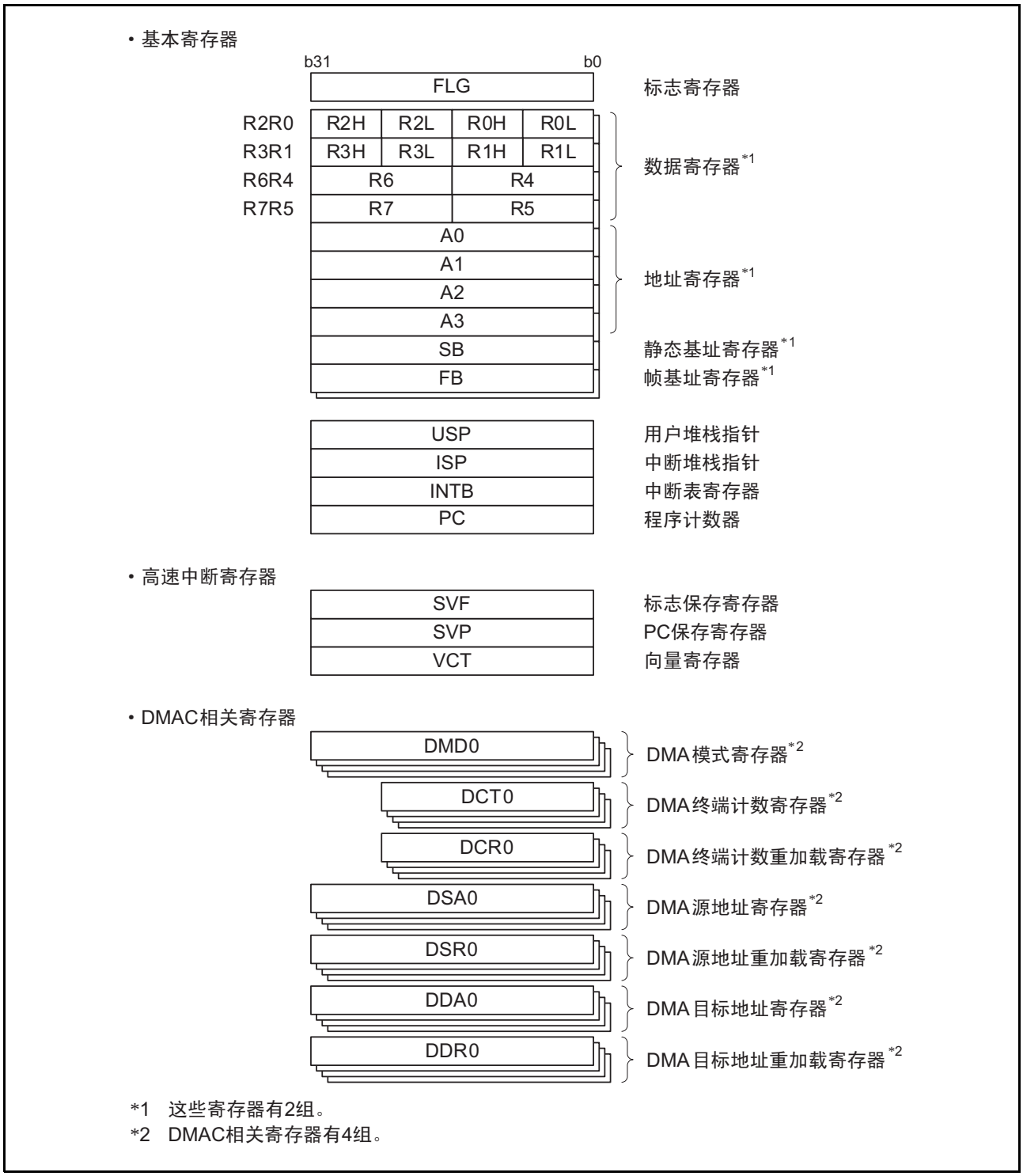


图 1.2 中央处理器的寄存器结构

1.3.1 基本寄存器

基本寄存器有以下 9 种：

(1) 数据寄存器 （R2R0/R3R1/R6R4/R7R5/R0/R0H/R0L/R1/R1H/R1L/R2/R2H/R2L/R3/R3H/R3L/R4/R5/R6/R7）

数据寄存器 （R2R0/R3R1/R6R4/R7R5）由 32 位构成，主要用于传送、算术和逻辑运算。

能将 R2R0/R3R1/R6R4/R7R5 的高位 （R2/R3/R6/R7）和低位 （R0/R1/R4/R5）分别用作 16 位数据寄存器，也能将 R2R0/R3R1 的高位 （R2H/R3H）、中高位 （R2L/R3L）、中低位 （R0H/R1H）和低位 （R0L/R1L）分别用作 8 位数据寄存器。

(2) 地址寄存器 （A0/A1/A2/A3）

地址寄存器 （A0/A1/A2/A3）由 32 位构成，具有和数据寄存器同等的功能，用于地址寄存器间接寻址和地址寄存器相对寻址。

(3) 静态基址寄存器 （SB）

静态基址寄存器 （SB）由 32 位构成，用于 SB 相对寻址。

(4) 帧基址寄存器 （FB）

帧基址寄存器 （FB）由 32 位构成，用于 FB 相对寻址。

(5) 程序计数器 （PC）

程序计数器 （PC）由 32 位构成，指示下一条要执行的指令的地址。

(6) 中断表寄存器 （INTB）

中断表寄存器 （INTB）由 32 位构成，指示中断向量表的起始地址。

(7) 用户堆栈指针 （USP） / 中断堆栈指针 （ISP）

堆栈指针有用户堆栈指针 （USP）和中断堆栈指针 （ISP） 2 种，都由 32 位构成。

能通过堆栈指针指定标志 （U 标志）转换要使用的堆栈指针 （USP/ISP）。

堆栈指针指定标志 （U 标志）为标志寄存器 （FLG）的 bit7。

必须给 USP 和 ISP 设定 4 的倍数，这样能提高执行效率。

(8) 标志寄存器 （FLG）

标志寄存器 （FLG）由包括 16 位保留位的 32 位寄存器构成，用作以 1 位为单位的标志。各标志的功能请参照 “1.3.4 标志寄存器 （FLG）”。

1.3.2 高速中断寄存器

高速中断寄存器是为高速响应中断而设置的寄存器，有以下 3 个寄存器。在中断响应顺序中保存寄存器时，使用这些寄存器来代替堆栈区，就能实现高速响应中断。

(1) 标志保存寄存器 （SVF）

标志保存寄存器 （SVF）由 16 位构成，在发生高速中断时用于保存标志寄存器 （FLG）。

(2) PC 保存寄存器 （SVP）

PC 保存寄存器 （SVP）由 32 位构成，在发生高速中断时用于保存程序计数器。

(3) 向量寄存器 （VCT）

向量寄存器 （VCT）由 32 位构成，指示发生高速中断时的转移目标地址。

1.3.3 DMAC 的相关寄存器

DMAC 的相关寄存器有以下 7 种：

(1) DMA 模式寄存器 （DMD0/DMD1/DMD2/DMD3）

DMA 模式寄存器 （DMD0/DMD1/DMD2/DMD3）由 32 位构成，是设定 DMA 的传送模式等的寄存器。

(2) DMA 终端计数寄存器 （DCT0/DCT1/DCT2/DCT3）

DMA 终端计数寄存器 （DCT0/DCT1/DCT2/DCT3）由 24 位构成，是对 DMA 传送次数进行计数的寄存器。

(3) DMA 终端计数重加载寄存器 （DCR0/DCR1/DCR2/DCR3）

DMA 终端计数重加载寄存器 （DCR0/DCR1/DCR2/DCR3）由 24 位构成，是设定 DMA 终端计数寄存器初始值的寄存器。

(4) DMA 源地址寄存器 （DSA0/DSA1/DSA2/DSA3）

DMA 源地址寄存器 （DSA0/DSA1/DSA2/DSA3）由 32 位构成，是保持 DMA 传送源地址的寄存器。

(5) DMA 源地址重加载寄存器 （DSR0/DSR1/DSR2/DSR3）

DMA 源地址重加载寄存器 （DSR0/DSR1/DSR2/DSR3）由 32 位构成，是设定 DMA 源地址寄存器初始值的寄存器。

(6) DMA 目标地址寄存器 （DDA0/DDA1/DDA2/DDA3）

DMA 目标地址寄存器 （DDA0/DDA1/DDA2/DDA3）由 32 位构成，是保持 DMA 传送目标地址的寄存器。

(7) DMA 目标地址重加载寄存器 （DDR0/DDR1/DDR2/DDR3）

DMA 目标地址重加载寄存器 （DDR0/DDR1/DDR2/DDR3）由 32 位构成，是设定 DMA 目标地址寄存器初始值的寄存器。

1.3.4 标志寄存器 （FLG）

标志寄存器 （FLG）的结构如图 1.3 所示。

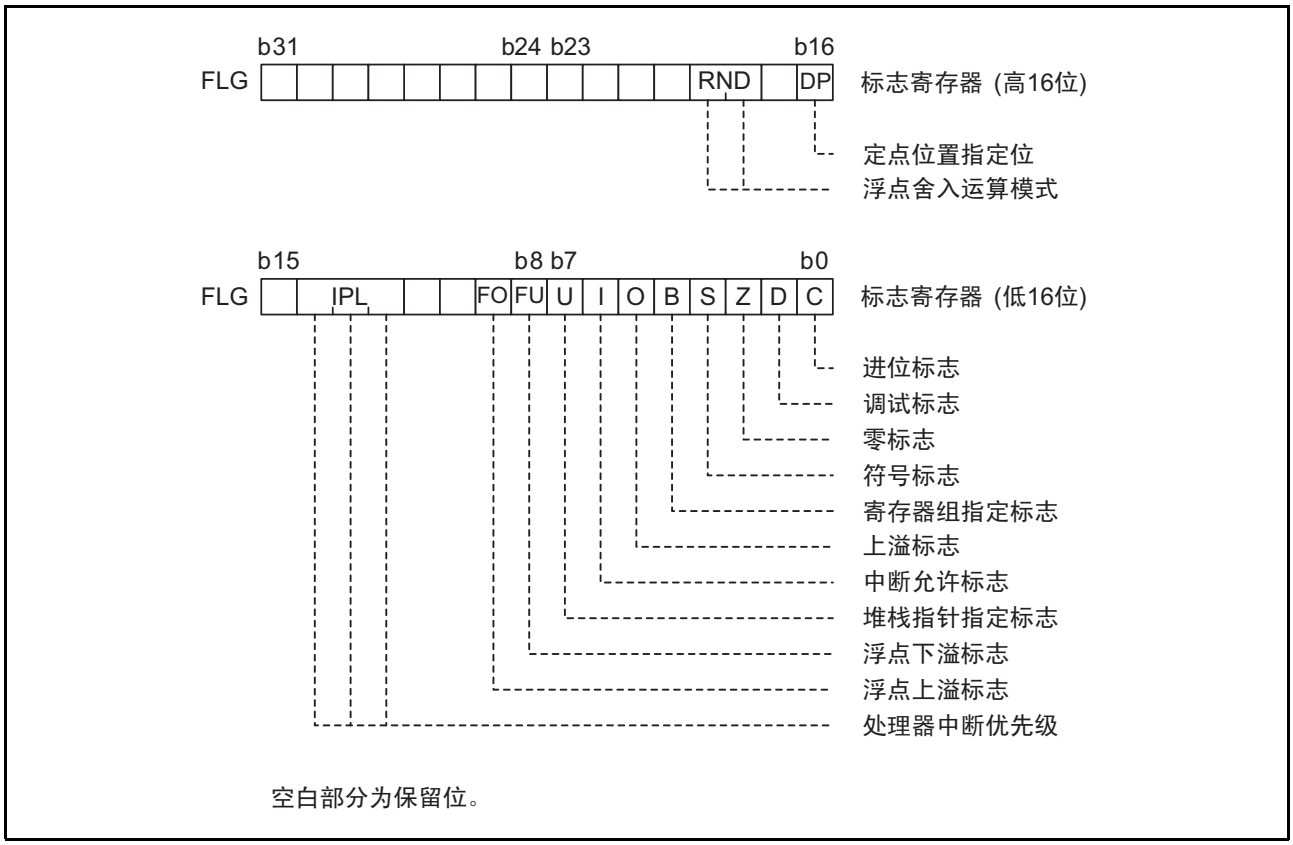


图 1.3 标志寄存器 （FLG）的结构

各标志的功能如下所示：

(1) bit0：进位标志 （C 标志）

保存由算术逻辑单元产生的进位、借位和移出位等。

(2) bit1：调试标志 （D 标志）

这是允许单步中断的标志。
如果将此标志置 “1”，就在指令执行后产生单步中断。一旦接受中断，此标志就变为 “0”。

(3) bit2：零标志 （Z 标志）

当运算结果是 “0” 时，此标志为 “1”，否则为 “0”。

(4) bit3：符号标志 （S 标志）

当运算结果是负时，此标志为 “1”，否则为 “0”。

(5) bit4：寄存器组指定标志 （B 标志）

选择寄存器组。当此标志为 “0” 时，指定寄存器组 0；当此标志为 “1” 时，指定寄存器组 1。

(6) bit5：上溢标志 （O 标志）

当运算结果发生上溢时，此标志为 “1”，否则为 “0”。

(7) bit6：中断允许标志 （I 标志）

这是允许可屏蔽中断的标志。
当此标志为 “0” 时，禁止中断；当此标志为 “1” 时，允许中断。
一旦接受中断，此标志就变为 “0”。

(8) bit7：堆栈指针指定标志 （U 标志）

当此标志为 “0” 时，指定中断堆栈指针 （ISP）；当此标志为 “1” 时，指定用户堆栈指针 （USP）。
当接受硬件中断或者执行软件中断序号 0 ～ 127 的 INT 指令时，此标志变为 “0”。

(9) bit8：浮点下溢标志 （FU 标志）

当浮点运算结果小于最小规格化数 （下溢）时，此标志为 “1”，否则为 “0”。
在操作数的数据既不是规格化数也不是 “0”（非法输入值）的情况下，此标志也为 “1”。

(10) bit9：浮点上溢标志 （FO 标志）

当浮点运算结果大于最大规格化数 （上溢）时，此标志为 “1”，否则为 “0”。
在操作数的数据既不是规格化数也不是 “0”（非法输入值）的情况下，此标志也为 “1”。

(11) bit10 ～ bit11：保留区

(12) bit12 ～ bit14：处理器中断优先级 （IPL）

处理器中断优先级 （IPL）由 3 位构成，指定 0 级～ 7 级的 8 个级别的处理器中断优先级。
如果有请求的中断的中断请求级高于处理器中断优先级 （IPL），就允许该中断。
如果将处理器中断优先级 （IPL）设定为 “7 级”（111b），就禁止全部的中断。

(13) bit15：保留区

(14) bit16：定点位置指定位 （DP 位）

这是指定定点数的小数点位置的位，也是指定从定点乘法运算结果中取出哪个部分作为最终运算结果的位。
用于 MULX 指令。

(15) bit17：保留区

(16) bit18 ～ bit19：浮点舍入运算模式 （RND）

浮点舍入运算模式 （RND）由 2 位构成，指定浮点运算结果的舍入模式。

RND	舍入模式	说明
00b	最接近的值	向可表现的最接近的值舍入。当原来的值为可表现的 2 个值的中间值时，将 bit0 舍入为 “0”。
01b	保留	不能设定。
10b	–∞ 方向	将运算结果向接近 “–∞” 的方向舍入。
11b	零方向	将运算结果向接近 “0” 的方向舍入。

(17) bit20 ～ bit31：保留区

1.3.5 寄存器组

由数据寄存器 （R2R0/R3R1/R6R4/R7R5）、地址寄存器 （A0/A1/A2/A3）、帧基址寄存器 （FB）和静态基址寄存器 （SB）构成的寄存器组有 2 组。通过标志寄存器 （FLG）的寄存器组指定标志 （B 标志）进行寄存器组的转换。

寄存器组的结构如图 1.4 所示。

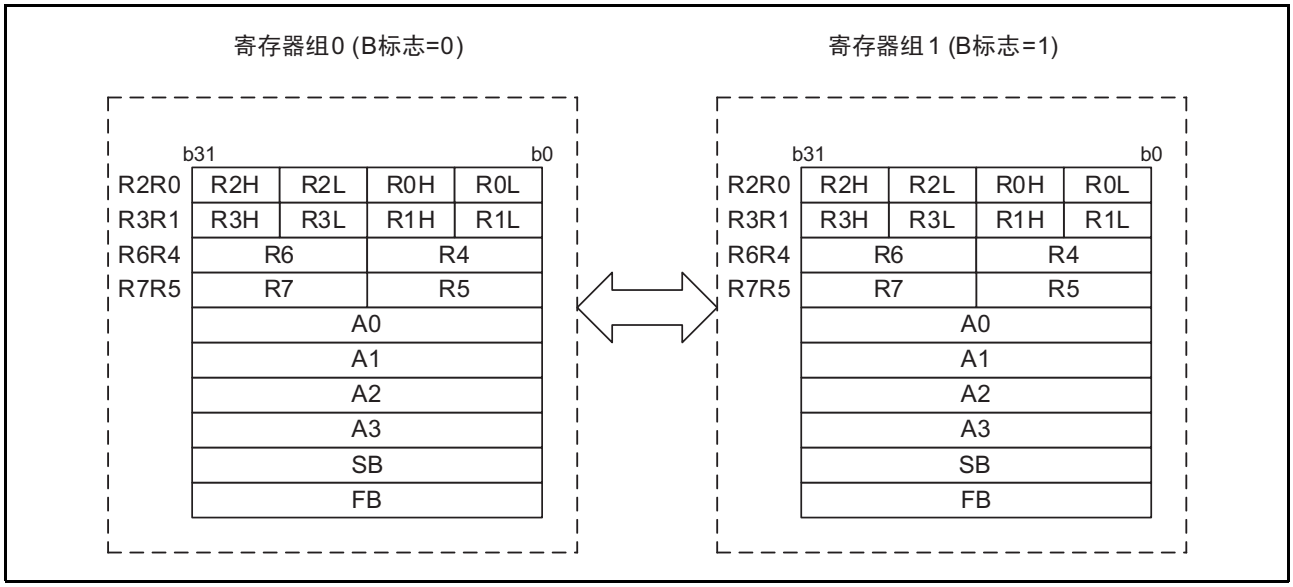


图 1.4 寄存器组的结构

R32C/100 系列提供有能在寄存器组指定标志为 “0”的状态下存取寄存器组 1 的寻址方式。

1.3.6 复位解除后的内部寄存器值

复位解除后的各寄存器的值如下所示：

- 数据寄存器 （R2R0/R3R1/R6R4/R7R5） ：00000000h
- 地址寄存器 （A0/A1/A2/A3） ：00000000h
- 静态基址寄存器 （SB） ：00000000h
- 帧基址寄存器 （FB） ：00000000h
- 中断表寄存器 （INTB） ：00000000h
- 用户堆栈指针 （USP） ：00000000h
- 中断堆栈指针 （ISP） ：00000000h
- 标志寄存器 （FLG） ：00000000h
- DMA 模式寄存器 （DMD0/DMD1/DMD2/DMD3） ：00000000h
- DMA 终端计数寄存器 （DCT0/DCT1/DCT2/DCT3） ：不定值
- DMA 终端计数重加载寄存器 （DCR0/DCR1/DCR2/DCR3） ：不定值
- DMA 源地址寄存器 （DSA0/DSA1/DSA2/DSA3） ：不定值
- DMA 源地址重加载寄存器 （DSR0/DSR1/DSR2/DSR3） ：不定值
- DMA 目标地址寄存器 （DDA0/DDA1/DDA2/DDA3） ：不定值
- DMA 目标地址重加载寄存器 （DDR0/DDR1/DDR2/DDR3） ：不定值
- 标志保存寄存器 （SVF） ：不定值
- PC 保存寄存器 （SVP） ：不定值
- 向量寄存器 （VCT） ：不定值

1.4 数据类型

数据类型为整数型、 10 进制数型、定点型、浮点型、位型、字符串型共 6 种。

1.4.1 整数型

整数有带符号整数和不带符号整数，带符号整数的负值用 2 的补码表现。

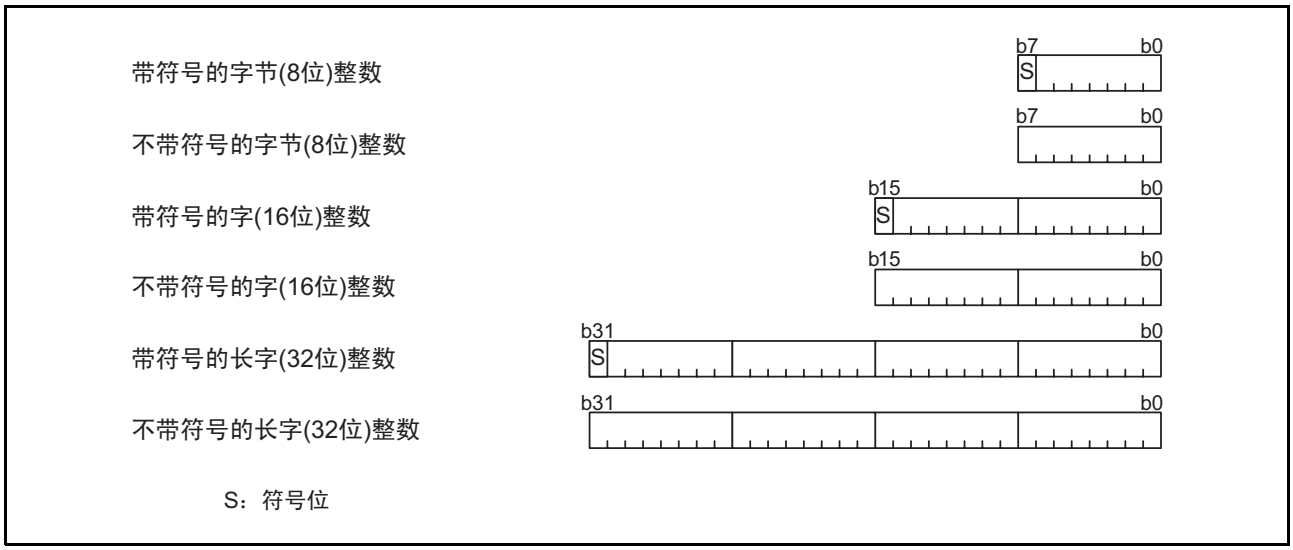


图 1.5 整数型

1.4.2 10 进制数型

能处理二进制编码的十进制数 （BCD）。 10 进制数能用于 DADC、 DADD、 DSBB、 DSUB 共 4 种指令。

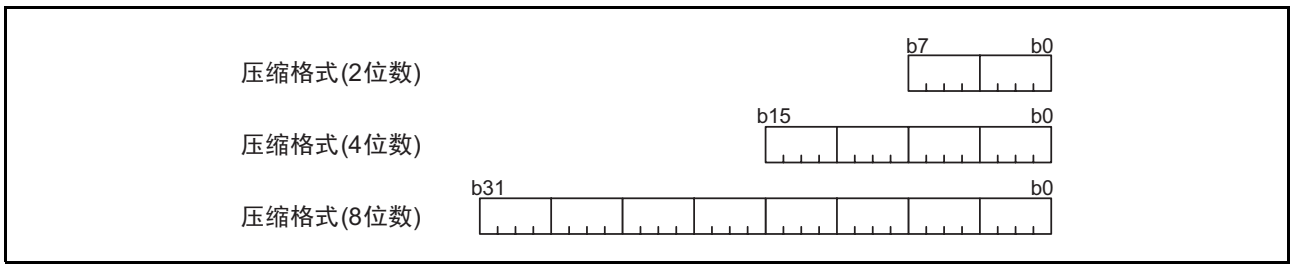


图 1.6 10 进制数型

1.4.3 定点型

通常，通过整数乘法运算指令和移位指令进行定点数的乘法运算，但是 R32C/100 系列配置有定点专用乘法运算指令 MULX。 MULX 指令支持的定点数格式如图 1.7 所示，当 DP 为 “1” 时，没有 8 位定点数。

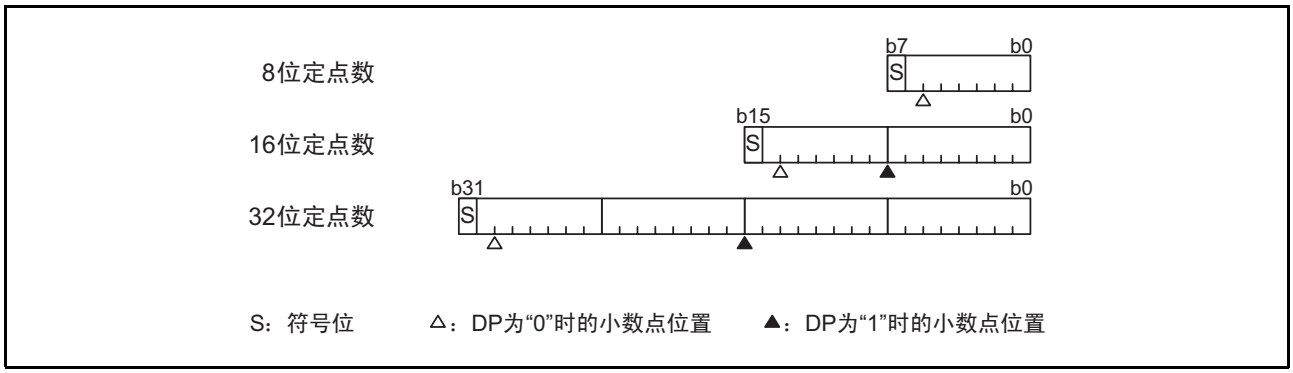


图 1.7 定点型

1.4.4 浮点型

R32C/100 系列支持由 IEEE754 规定的单精度浮点型。

浮点型能用于浮点运算指令 ADDF、 CMPF、 CNVIF、 DIVF、 MULF、 ROUND、 SUBF 共 7 种指令。

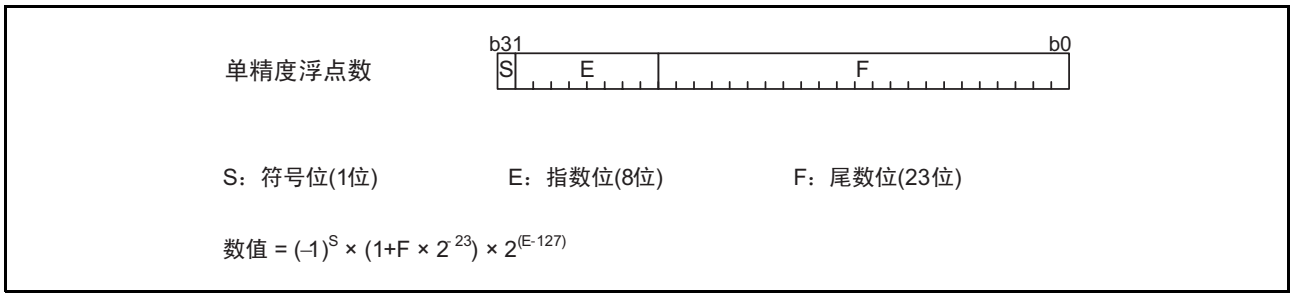


图 1.8 浮点型

浮点型支持以下的数值：

- 0<E<255 （规格化数—Normal Numbers）
- E=0 并且 F=0 （零—Signed Zero）

浮点型不支持以下的数值：

- E=0 并且 F>0 （非规格化数—Subnormal Numbers）
- E=255 并且 F=0 （无穷大—Infinity）
- E=255 并且 F>0 （非数值—NaN: Not-a-Number）

1.4.5 位型

位型能用于 BCLR、 BSET、 BNOT、 BTST、 BM*Cnd*、 BTSTS、 BTSTC 共 7 种指令。

寄存器的位由寄存器名和 0 ～ 7 的位号指定，能指定的寄存器只有 8 位长的寄存器 （R0L/R0H/R1L/R1H/R2L/R2H/R3L/R3H）。

同样，存储器的位也由对象地址和 0 ～ 7 的位号指定，能用于指定地址的寻址方式为绝对 （仅 BTST:S）、地址 0 相对、地址寄存器间接、地址寄存器相对、 SB 相对、 FB 相对共 6 种。

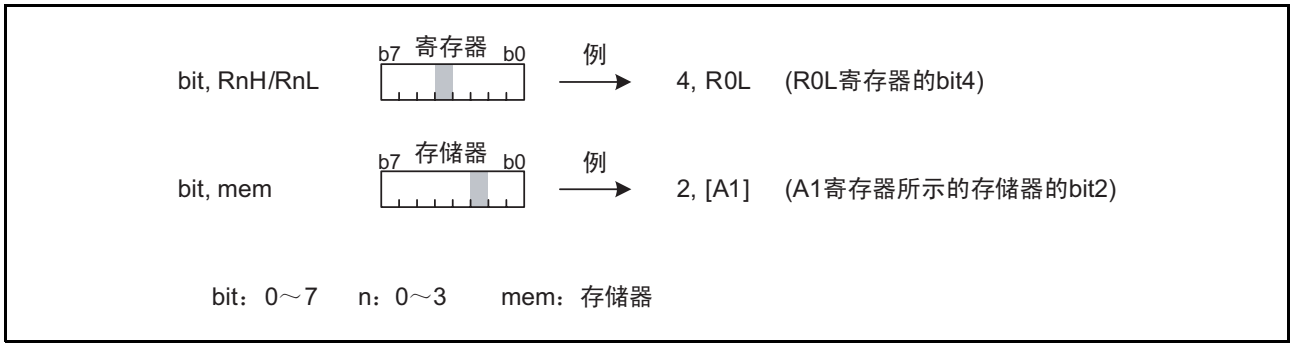


图 1.9 位型

1.4.6 字符串型

字符串型是指将字节 （8 位）、字 （16 位）和长字 （32 位）中的任意个同种数据连续排列的数据类型。

能使用字符串的指令有 SMOVB、SMOVF、SMOVU、SCMPU、SIN、SOUT、SSTR、SUNTIL、SWHILE 共 9 种指令。

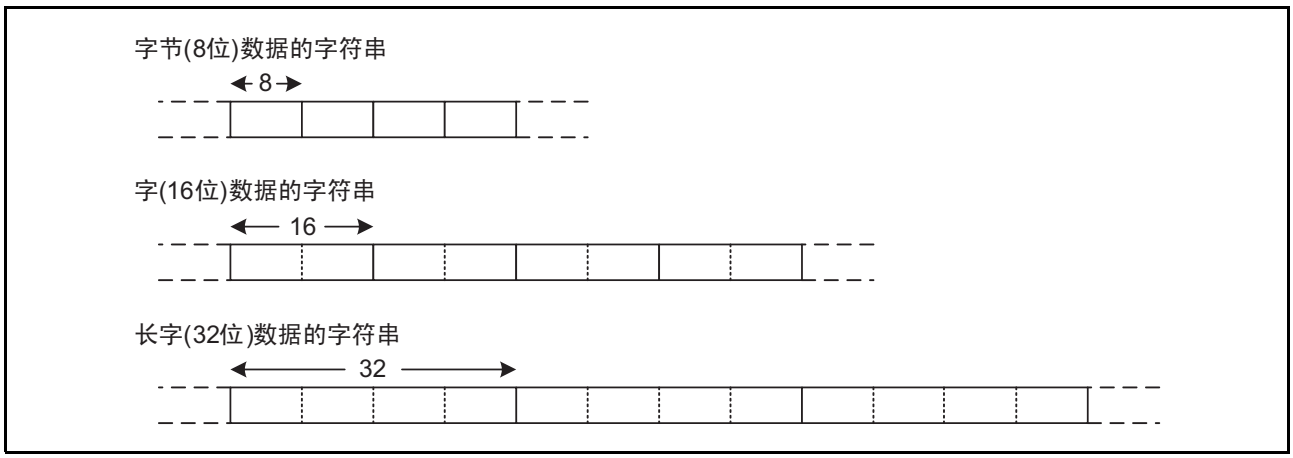


图 1.10 字符串型

1.5 数据排列

1.5.1 寄存器的数据排列

寄存器的数据长度和位号的关系如图 1.11 所示。

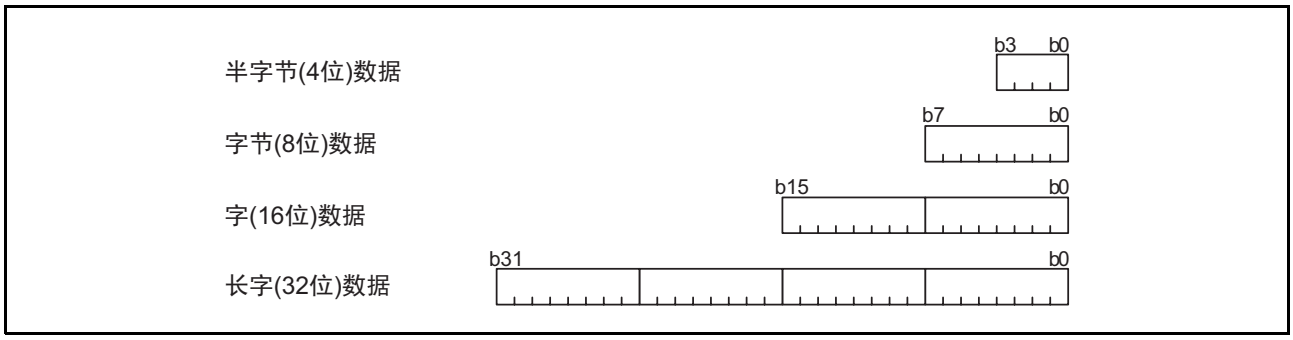


图 1.11 寄存器的数据排列

1.5.2 存储器的数据排列

存储器的数据排列如图 1.12 所示。

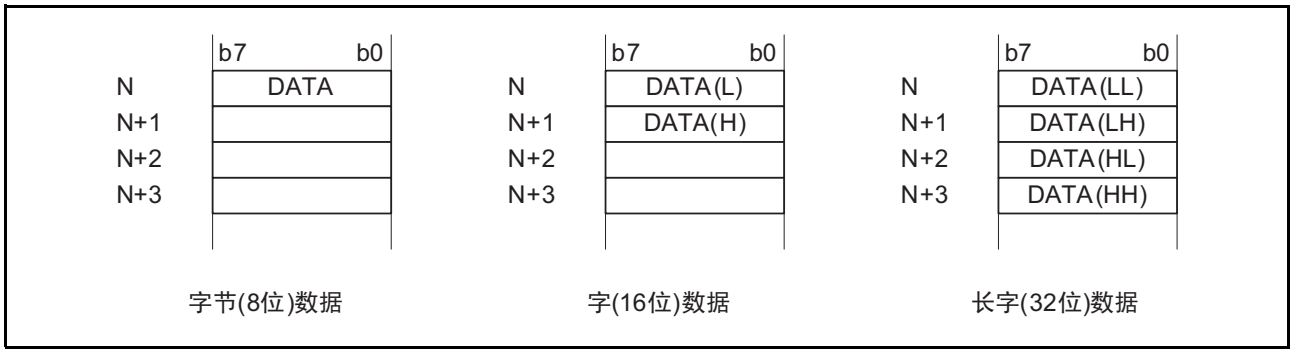


图 1.12 存储器的数据排列

1.6 指令格式

R32C/100 系列的指令结构如图 1.13 所示，1 条指令最短为 1 字节，最长为 14 字节。由操作码部分决定指令的操作、操作数的种类、操作数的个数、以及指令的长度。

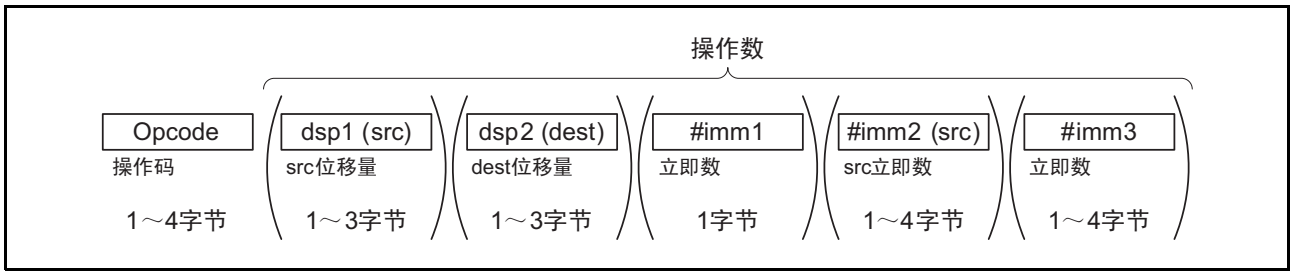


图 1.13 指令格式

R32C/100 系列定义了使常用指令字节数少的指令码，并且通过限定能使用的寻址方式，能给部分指令选择指令字节数更少的格式。

指令格式有一般格式、快速格式、短格式、零格式共 4 种格式。

各种格式的特点如下所示：

(1) 一般格式 （:G）

一般格式的操作码为 2 ～ 3 字节，包含操作信息、 source （以下省略为 src）和 destination （以下省略为 dest）的寻址方式信息。

指令码由操作码 （2 ～ 3 字节）、 src 码 （0 ～ 4 字节）和 dest 码 （0 ～ 3 字节）构成。

(2) 快速格式 （:Q）

快速格式的操作码为 1 ～ 2 字节，包含操作信息、立即数和 dest 的寻址方式信息，但是限制了操作码中包含的立即数是能用 3 位或者 4 位表现的数值。

指令码由包含立即数的操作码 （1 ～ 2 字节）和 dest 码 （0 ～ 3 字节）构成。

(3) 短格式 （:S）

短格式的操作码为 1 ～ 2 字节，包含操作信息、 src 和 dest 的寻址方式信息，但是限制了能使用的寻址方式。

指令码由操作码 （1 ～ 2 字节）、 src 码 （0 ～ 4 字节）和 dest 码 （0 ～ 2 字节）构成。

(4) 零格式 （:Z）

零格式的操作码为 1 字节，包含操作 （以及立即数）信息和 dest 的寻址方式信息，但是立即数被固定为“0”，还限制了能使用的寻址方式。

指令码由操作码 （1 字节）和 dest 码 （0 ～ 2 字节）构成。

1.6.1 操作码

操作码由 1 ～ 3 字节长的位串构成。当操作数的寻址方式为 “间接指令寻址”或者 “组 1 寄存器直接”时，在操作码的起始位置附加 8 位的位串，变为 2 ～ 4 字节长。

1.6.2 操作数

操作数由 dsp1、dsp2、#imm1、#imm2、#imm3 的各字段构成，根据指令和选择的寻址方式决定是否省略各字段。

将 src 码分配到 dsp1 字段或者 #imm2 字段， dest 码分配到 dsp2 字段。在扩展操作码时，使用 #imm1 字段，但是对于持有 2 个 src 码的指令 （CLIP、 STZX），将 src1 分配到 #imm2 字段、 src2 分配到 #imm3 字段。

(1) dsp1 字段

dsp1 字段由 1 ～ 3 字节构成，只在 src 的寻址方式为绝对或者相对 （堆栈指针相对除外）时有此字段。

(2) dsp2 字段

dsp2 字段由 1 ～ 3 字节构成，只在 dest 的寻址方式为相对 （程序计数器相对除外）时有此字段。

(3) #imm1 字段

#imm1 字段由 1 字节构成，只有部分指令有此字段。

BM*Cnd* 指令	给 #imm1 字段分配 8 位立即数，用低 4 位指定条件码。
LDC/STC 指令	通过分配给 #imm1 字段的 8 位立即数，指定 DMAC 相关寄存器和 VCT 寄存器。在存取 SB、 FB、 FLG、 SP、 ISP、 SVF、 SVP、 INTB 时没有此字段。
MOV dsp:8[SP]	分配到 #imm1 字段的 8 位立即数表示对堆栈指针的位移量 dsp:8。
PUSHM/POPM 指令	给 #imm1 字段分配 8 位立即数，并且在为 “1” 的位的位置指定要保存 / 恢复的寄存器。
ROT/SHA/SHL 指令	分配到 #imm1 字段的 8 位立即数表示移位的位数。

(4) #imm2 字段

#imm2 字段由 1/2/4 字节构成，只在 src 的寻址方式为立即数和符号扩展立即数时有此字段。

(5) #imm3 字段

#imm3 字段由 1/2/4 字节构成，只有部分指令有此字段。

CLIP/STZX 指令	给 src 操作数指定 2 个立即数。将第 1 个立即数设定到 #imm2 字段、第 2 个立即数设定到 #imm3 字段。

1.7 向量表

向量表中有中断向量表，中断向量表中有固定向量表和可变向量表。

1.7.1 固定向量表

固定向量表是分配在固定地址 （地址 FFFFFFDCh ～ FFFFFFFFh）的中断向量表，如图 1.14 所示。
中断向量表由各向量为 4 字节的 9 个中断向量构成，给各向量设定中断程序的起始地址。

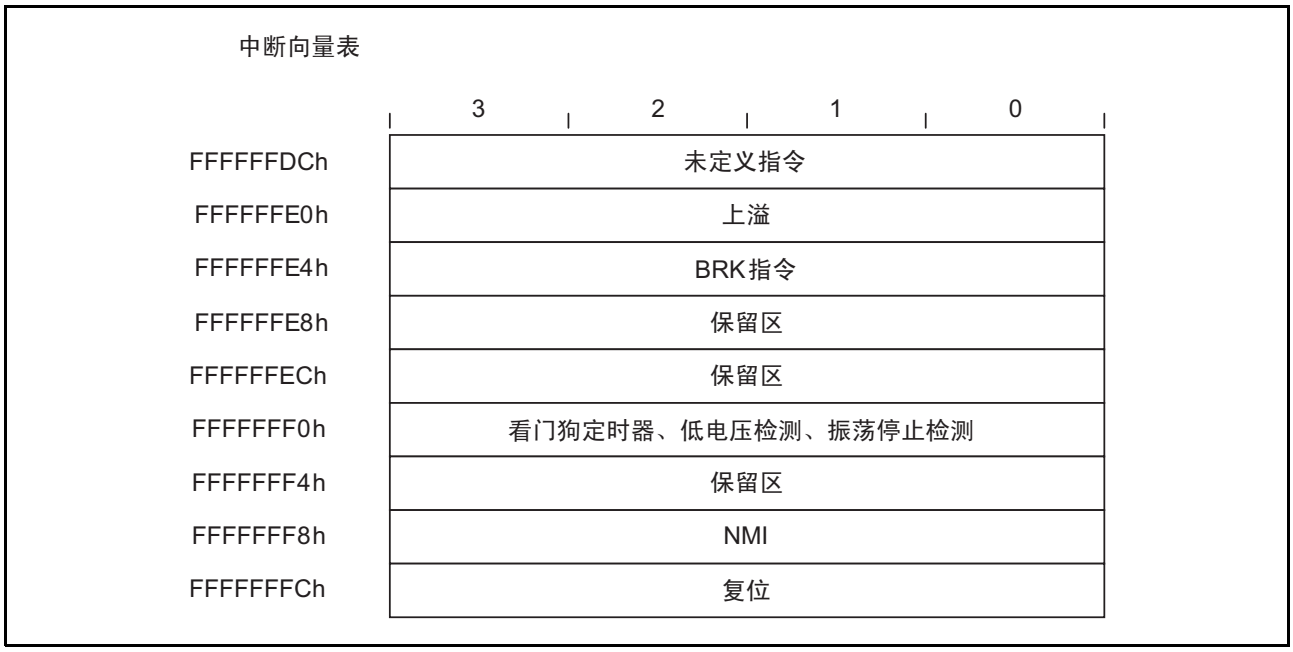


图 1.14 固定向量表

1.7.2 可变向量表

可变向量表是能改变表的分配地址的向量表，是以中断表寄存器 （INTB）的内容所示的值为起始地址（IntBase）的 1K 字节的中断向量表，如图 1.15 所示。

可变向量表由各向量为 4 字节的 256 个中断向量构成，给各向量区设定中断处理程序的起始地址。每个向量都有软件中断序号 （0 ～ 255）， INT 指令使用此软件中断序号。

在可变向量表中还分配了内部外围功能的中断 （外围功能中断），从软件中断序号 0 开始分配了外围功能中断的中断向量，但是外围功能中断的中断向量的个数因产品的种类而不同。因此，建议在使用 INT 指令中断时，从软件中断序号 255 开始使用。

用于 INT 指令中断的堆栈指针 （SP）因软件中断序号而不同。

对于软件中断序号 0 ～ 127，在接受中断请求时保存堆栈指针指定标志 （U 标志），并且将 U 标志置“0”，然后在选择中断堆栈指针 （ISP）后执行中断响应顺序。在从中断处理程序返回时，恢复接受中断请求前的 U 标志。

对于软件中断序号 128 ～ 255，不转换堆栈指针。

外围功能中断与软件中断序号无关，在接受中断请求时选择中断堆栈指针 （ISP）。

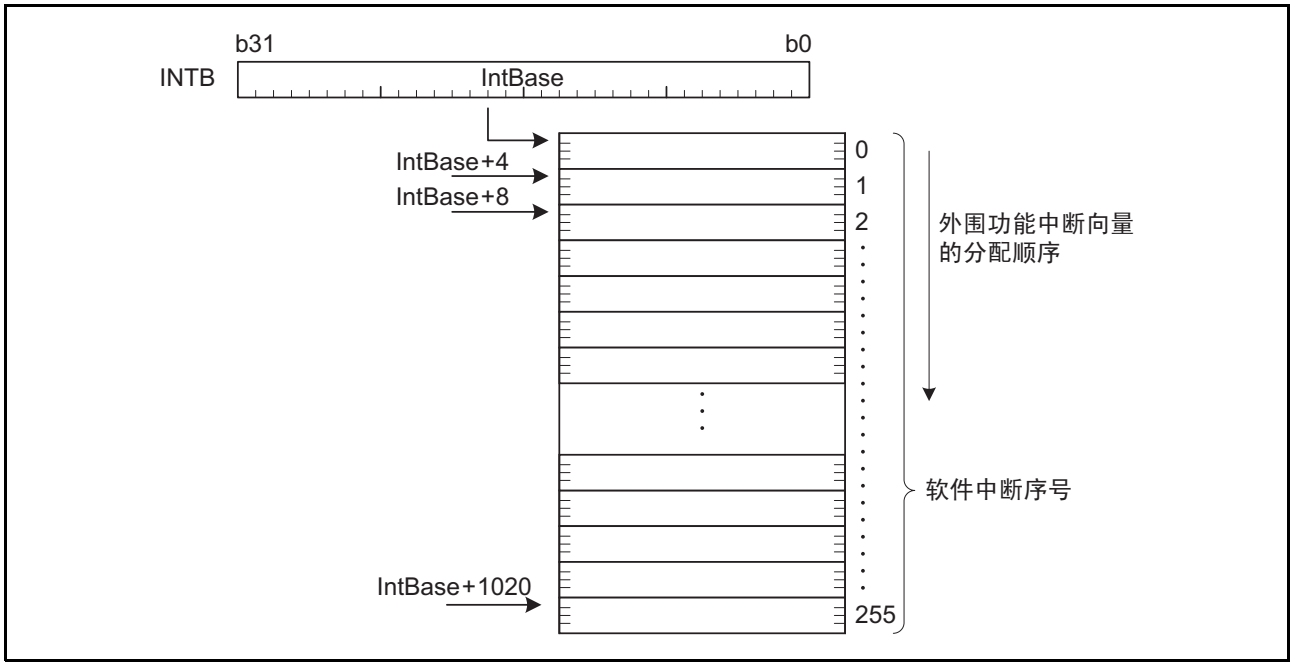

b31
b0
INTB
IntBase
IntBase+4
IntBase+8
IntBase+1020
0
1
2
255
外围功能中断向量
的分配顺序
软件中断序号

图 1.15 可变向量表

2. 寻址方式

2.1 寻址方式

在本章中，按各寻址方式对表示寻址方式的符号和操作进行说明。寻址方式有以下所示的 5 种类型：

(1) 一般指令寻址

这是存取通用寄存器和存储器的最一般的寻址。

- 寄存器直接
- 立即数
- 符号扩展立即数
- 地址0相对
- 地址寄存器间接
- 地址寄存器相对
- SB相对
- FB相对

(2) 间接指令寻址

这是根据存储器中的信息进行存储器存取的寻址，能用于支持一般指令寻址的大多数指令。

- 地址0相对间接
- 地址寄存器间接的间接
- 地址寄存器相对间接
- SB相对间接
- FB相对间接

(3) 扩展指令寻址

这是为了对一般指令寻址不能直接存取的寄存器和存储器进行存取以及缩短代码长度而被扩展的寻址。

- 组1寄存器直接
- 短立即数
- 堆栈指针相对

(4) 特殊指令寻址

这是存取专用寄存器、标志和存储器以及用于转移的寻址，只支持部分指令。

- 专用寄存器直接
- FLG直接
- 绝对
- 程序计数器相对

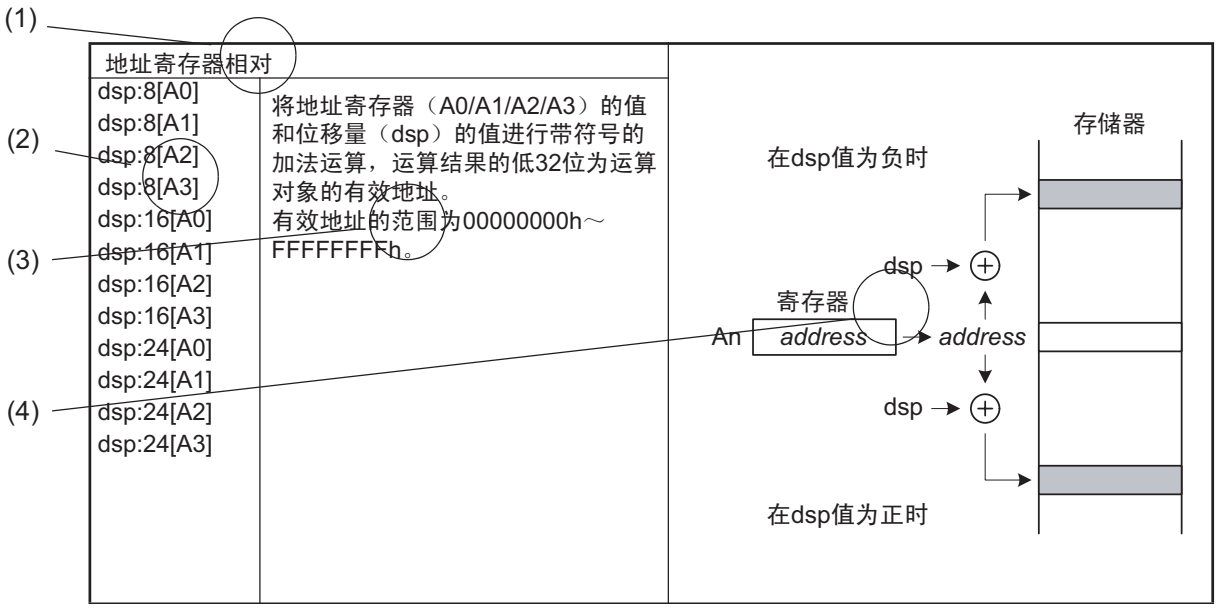
(5) 位指令寻址

这是以位为单位存取通用寄存器和存储器的寻址。

- 寄存器直接
- 绝对
- 地址0相对
- 地址寄存器间接
- 地址寄存器相对
- SB相对
- FB相对

2.2 本章的阅读方法

通过以下的例子说明本章的阅读方法：



(1) 名称

这是寻址的名称。

(2) 符号

这是表示寻址方式的符号。

“:3”、“:4”、“:8”、“:16”、“:24”、“:32”表示前面的值的有效位数。在手册中需要明确记述了有效位数，而通常情况下是不需要记述的，请用数值或者符号替换 “dsp”等符号。

(3) 解说

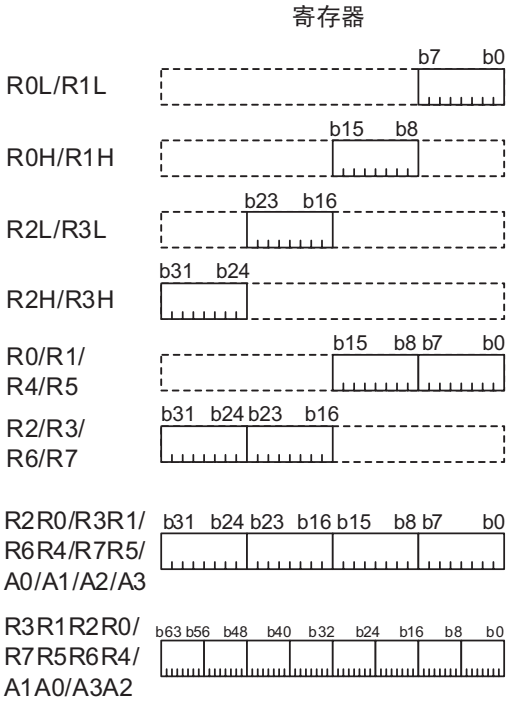
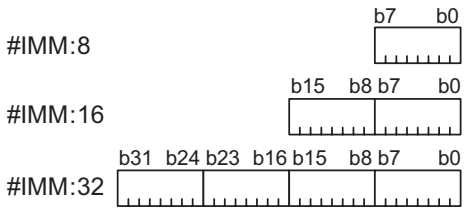
说明操作和有效地址的范围。

(4) 操作图

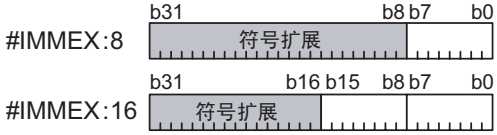
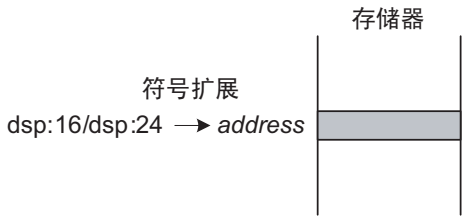
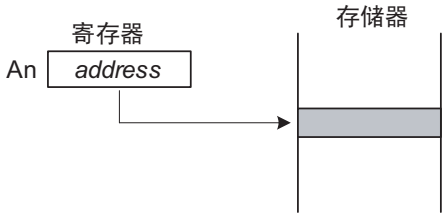
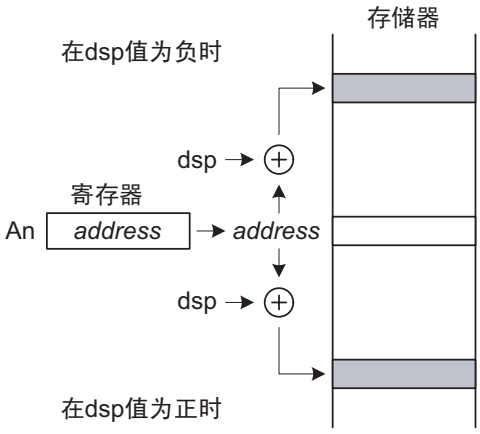
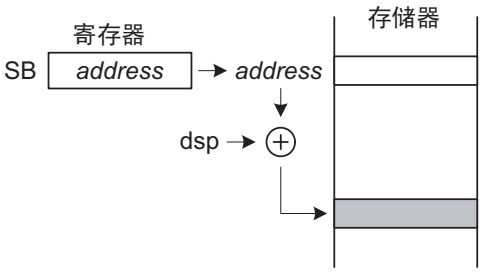
用图对操作进行说明。

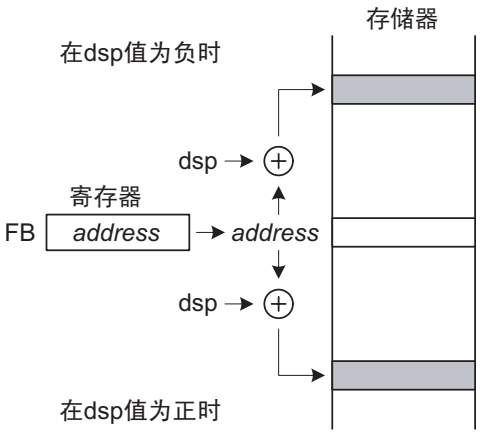
2.3 一般指令寻址

这是存取通用寄存器和存储器的最一般的寻址，能用于大多数指令。

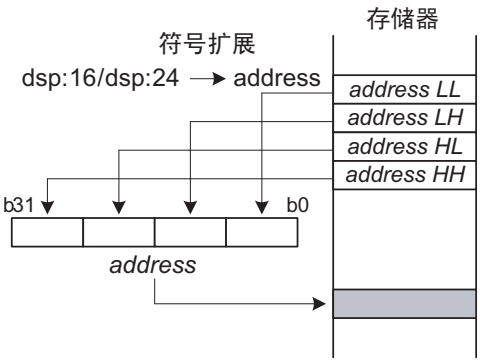
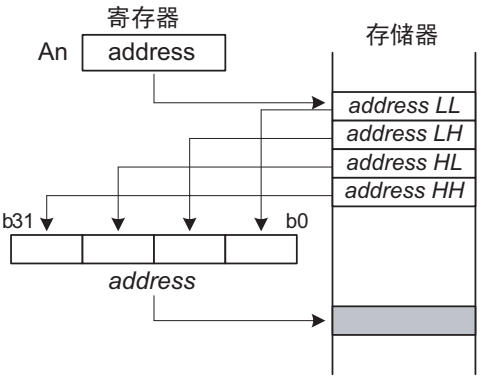
<table>
<tr><td colspan="2">寄存器直接</td><td rowspan="2"> 
</td></tr>
<tr><td>R0L
R0H
R1L
R1H
R2L
R2H
R3L
R3H
R0
R1
R2
R3
R4
R5
R6
R7
R2R0
R3R1
R6R4
R7R5
A0
A1
A2
A3
R3R1R2R0
R7R5R6R4
A1A0
A3A2</td><td>指定的寄存器为运算对象。</td></tr>
<tr><td colspan="2">立即数</td><td rowspan="2"> 
</td></tr>
<tr><td>#IMM
#IMM:8
#IMM:16
#IMM:32</td><td>#IMM 所示的立即数为运算对象。</td></tr>
</table>

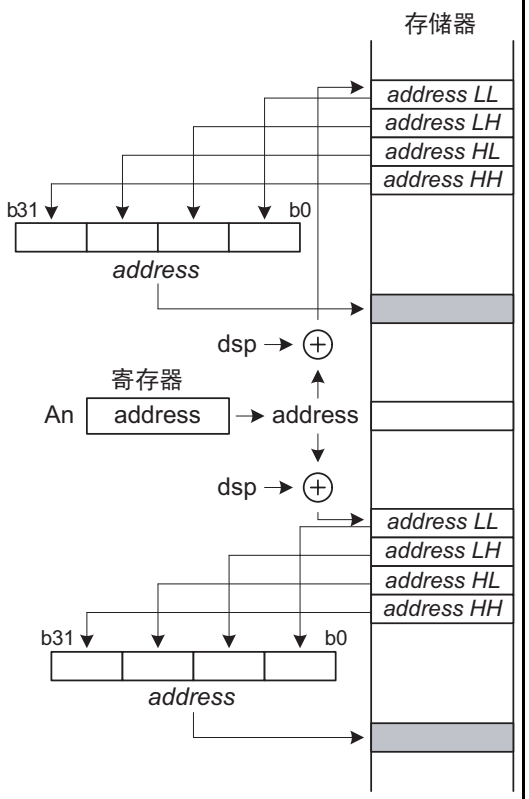
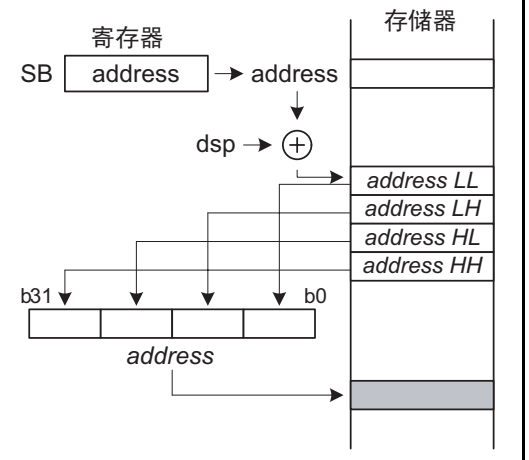
<table>
<tr><td colspan="2">符号扩展立即数</td><td rowspan="2">给长度指定符指定“.W”
 

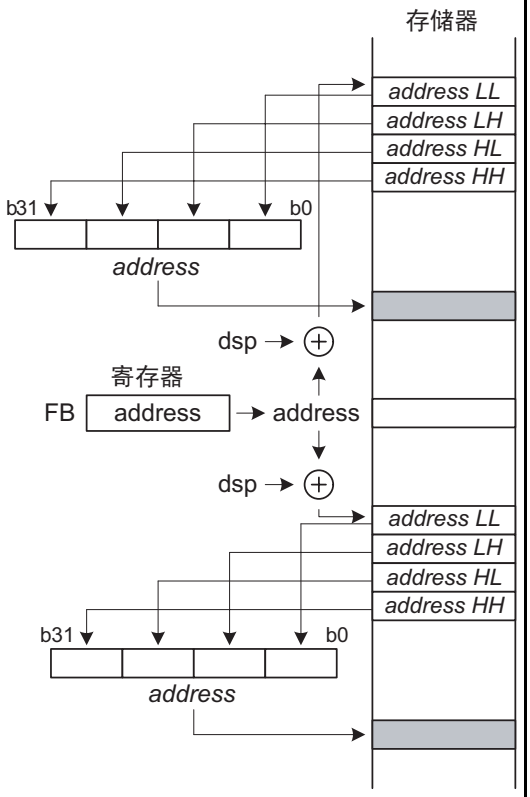
给长度指定符指定“.L”
 
</td></tr>
<tr><td>#IMMEX
#IMMEX:8
#IMMEX:16</td><td>将 #IMMEX 所示的立即数进行符号扩展，扩展为长度说明符指定的位长的结果为运算对象。</td></tr>
<tr><td colspan="2">地址 0 相对</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:16
dsp:24</td><td>将位移量 （dsp）所示的值进行符号扩展，扩展结果为运算对象的有效地址。
有效地址的范围在 dsp:16 时为 00000000h ～ 00007FFFh 和 FFFF8000h ～ FFFFFFFFh，在 dsp:24 时为 00000000h ～ 007FFFFFh 和 FF800000h ～ FFFFFFFFh。</td></tr>
<tr><td colspan="2">地址寄存器间接</td><td rowspan="2"> 
</td></tr>
<tr><td>[A0]
[A1]
[A2]
[A3]</td><td>地址寄存器 （A0/A1/A2/A3）的值为运算对象的有效地址。
有效地址的范围为 00000000h ～ FFFFFFFFh。</td></tr>
<tr><td colspan="2">地址寄存器相对</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:8[A0]
dsp:8[A1]
dsp:8[A2]
dsp:8[A3]
dsp:16[A0]
dsp:16[A1]
dsp:16[A2]
dsp:16[A3]
dsp:24[A0]
dsp:24[A1]
dsp:24[A2]
dsp:24[A3]</td><td>将地址寄存器 （A0/A1/A2/A3）的值和位移量 （dsp）的值进行带符号的加法运算，运算结果的低 32 位为运算对象的有效地址。
有效地址的范围为 00000000h ～ FFFFFFFFh。</td></tr>
<tr><td colspan="2">SB 相对</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:8[SB]
dsp:16[SB]
dsp:24[SB]</td><td>将静态基址寄存器 （SB）的值和位移量 （dsp）的值进行不带符号的加法运算，运算结果的低 32 位为运算对象的有效地址。
有效地址的范围为 00000000h ～ FFFFFFFFh。</td></tr>
</table>

FB 相对		
dsp:8[FB] dsp:16[FB]	将帧基址寄存器 （FB）的值和位移量 （dsp）的值进行带符号的加法运算，运算结果的低 32 位为运算对象的有效地址。 有效地址的范围为 00000000h ～ FFFFFFFFh。	

2.4 间接指令寻址

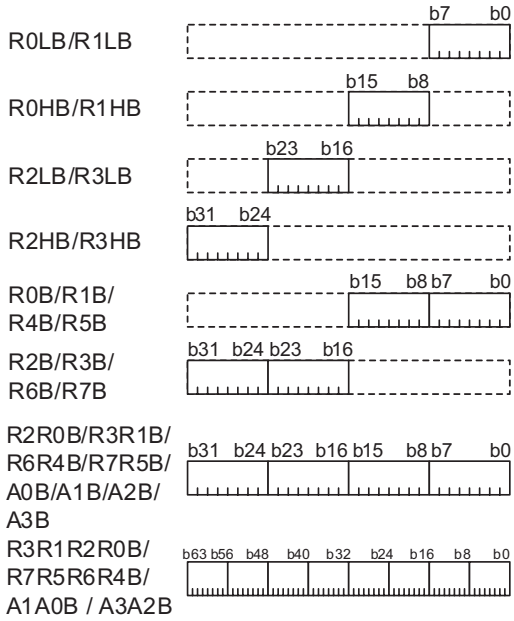
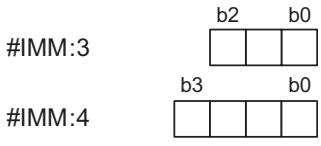
地址 0 相对间接		
[dsp:16] [dsp:24]	地址 0 相对寻址指定的地址中的 4 字节的值为运算对象的有效地址。 有效地址的范围为 00000000h ～ FFFFFFFFh。	
地址寄存器间接的间接		
[[A0]] [[A1]] [[A2]] [[A3]]	地址寄存器间接指定的地址中的 4 字节的值为运算对象的有效地址。 有效地址的范围为 00000000h ～ FFFFFFFFh。	

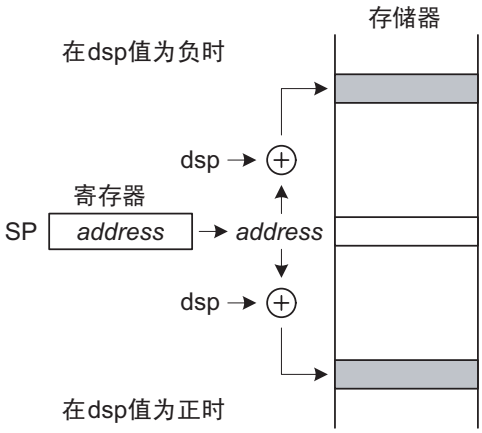
地址寄存器相对间接		
[dsp:8[A0]] [dsp:8[A1]] [dsp:8[A2]] [dsp:8[A3]] [dsp:16[A0]] [dsp:16[A1]] [dsp:16[A2]] [dsp:16[A3]] [dsp:24[A0]] [dsp:24[A1]] [dsp:24[A2]] [dsp:24[A3]]	地址寄存器相对指定的地址中的 4 字节的值为运算对象的有效地址。 有效地址的范围为 00000000h ～ FFFFFFFFh。	
SB 相对间接		
[dsp:8[SB]] [dsp:16[SB]] [dsp:24[SB]]	SB 相对指定的地址中的 4 字节的值为运算对象的有效地址。 有效地址的范围为 00000000h ～ FFFFFFFFh。	

FB 相对间接		
[dsp:8[FB]] [dsp:16[FB]]	FB 相对指定的地址中的 4 字节的值为运算对象的有效地址。 有效地址的范围为 00000000h ～ FFFFFFFFh。	

2.5 扩展指令寻址

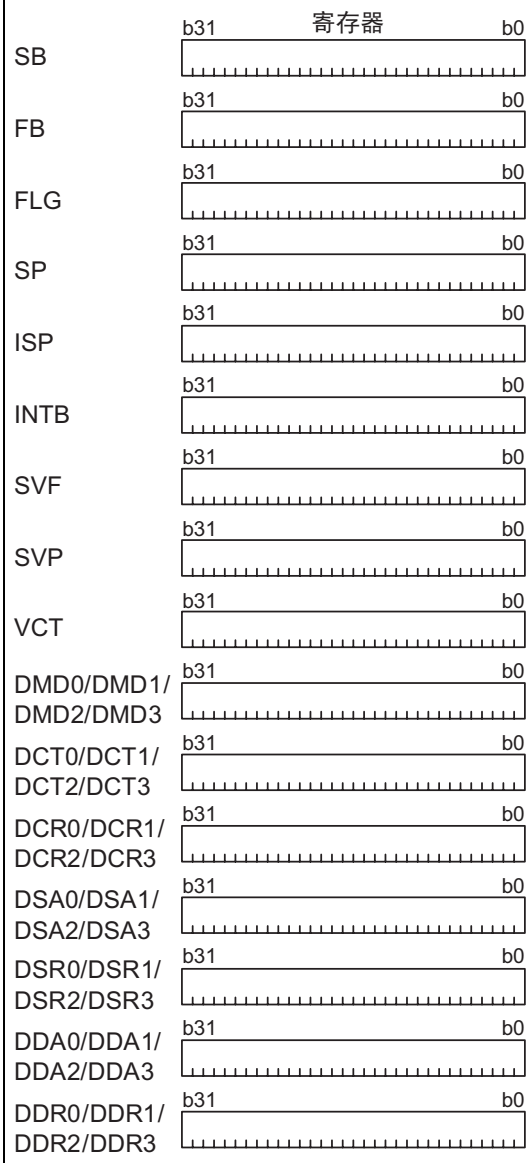
<table>
<tr><td colspan="2">组 1 寄存器直接</td><td rowspan="2">

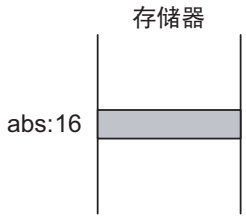
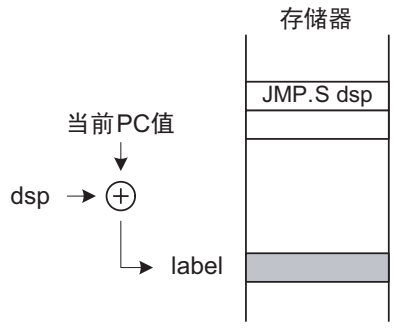
寄存器
 
</td></tr>
<tr><td>R0LB
R0HB
R1LB
R1HB
R2LB
R2HB
R3LB
R3HB
R0B
R1B
R2B
R3B
R4B
R5B
R6B
R7B
R2R0B
R3R1B
R6R4B
R7R5B
A0B
A1B
A2B
A3B
R3R1R2R0B
R7R5R6R4B
A1A0B
A3A2B</td><td>指定的组 1 寄存器为运算对象。
与 B 标志的值无关，总是能指定组 1 寄存器直接。</td></tr>
<tr><td colspan="2">短立即数</td><td rowspan="2"> 
</td></tr>
<tr><td>#0
#IMM:3
#IMM:4</td><td>零或者 #IMM 所示的立即数为运算对象。用于零格式或者快速格式。</td></tr>
</table>

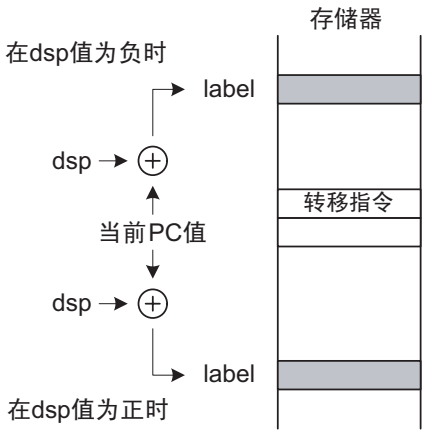
堆栈指针相对		
dsp:8[SP]	将堆栈指针 （SP）的值和位移量 （dsp）的值进行带符号的加法运算，运算结果的低 32 位为运算对象的有效地址。 有效地址的范围为 00000000h ～ FFFFFFFFh。 此寻址能用于 MOV 指令。	

2.6 特殊指令寻址

专用寄存器直接	
SB FB FLG SP ISP INTB SVF SVP VCT DMD0 DMD1 DMD2 DMD3 DCT0 DCT1 DCT2 DCT3 DCR0 DCR1 DCR2 DCR3 DSA0 DSA1 DSA2 DSA3 DSR0 DSR1 DSR2 DSR3 DDA0 DDA1 DDA2 DDA3 DDR0 DDR1 DDR2 DDR3	指定的专用寄存器为运算对象。 如果指定 SP，U 标志指示的堆栈指针就为对象。 此寻址能用于 ADD:Q、LDC、POPC、PUSHC、STC 指令。



<table>
<tr><td colspan="2">FLG 直接</td><td rowspan="2"> 
</td></tr>
<tr><td>U
I
O
B
S
Z
D
C</td><td>指定的标志为运算对象。
此寻址能用于 FCLR 指令和 FSET 指令。</td></tr>
<tr><td colspan="2">绝对</td><td rowspan="2"> 
</td></tr>
<tr><td>abs:16</td><td>abs 所示的值为运算对象的有效地址。
有效地址的范围为 00000000h ～ 0000FFFFh。
此寻址能用于 LDCTX 指令和 STCTX 指令。</td></tr>
<tr><td colspan="2">程序计数器相对</td><td rowspan="2"> 

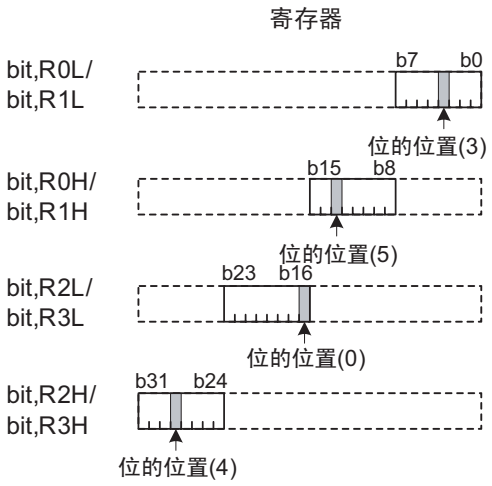
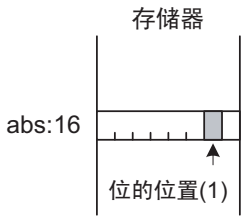
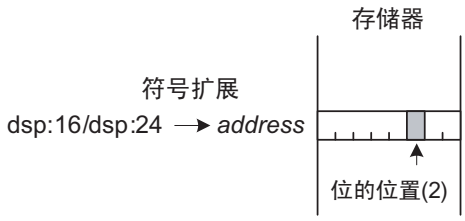
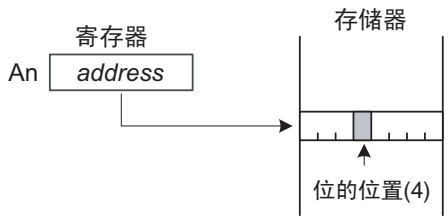
+1≤dsp≤+8
当前PC值为“转移指令的地址+1”。</td></tr>
<tr><td>label
（dsp:3）</td><td>在转移距离说明符（.length）为“.S”时，将当前程序计数器 （PC）的值和位移量 （dsp）的值进行不带符号的加法运算，运算结果的低 32 位为有效地址。
此寻址能用于 JMP 指令。</td></tr>
<tr><td>label
（dsp:8）
（dsp:16）
（dsp:24）</td><td>在转移距离说明符（.length）为“.B”、“.W”或者“.A”时，将当前程序计数器 （PC）的值和位移量 （dsp）的值进行带符号的加法运算，运算结果的低 32 位为有效地址。
此寻址能用于 JCnd、 JMP、 JSR 指令。</td><td> 

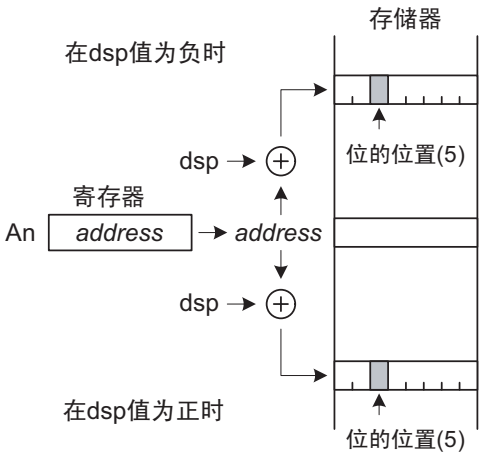
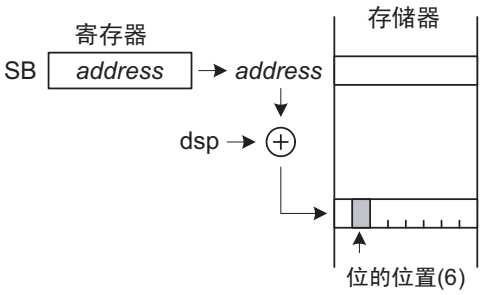
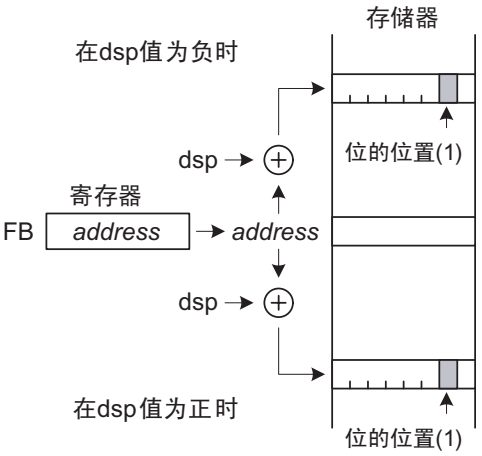
“.B” ：−128≤dsp≤+127
“.W” ：−32768≤dsp≤+32767
“.A” ：−8388608≤dsp≤+8388607
当前PC值为“转移指令的地址+1”。</td></tr>
</table>

2.7 位指令寻址

此寻址能用于以下指令：

BCLR、 BSET、 BNOT、 BTST、 BM*Cnd*、 BTSTC、 BTSTS

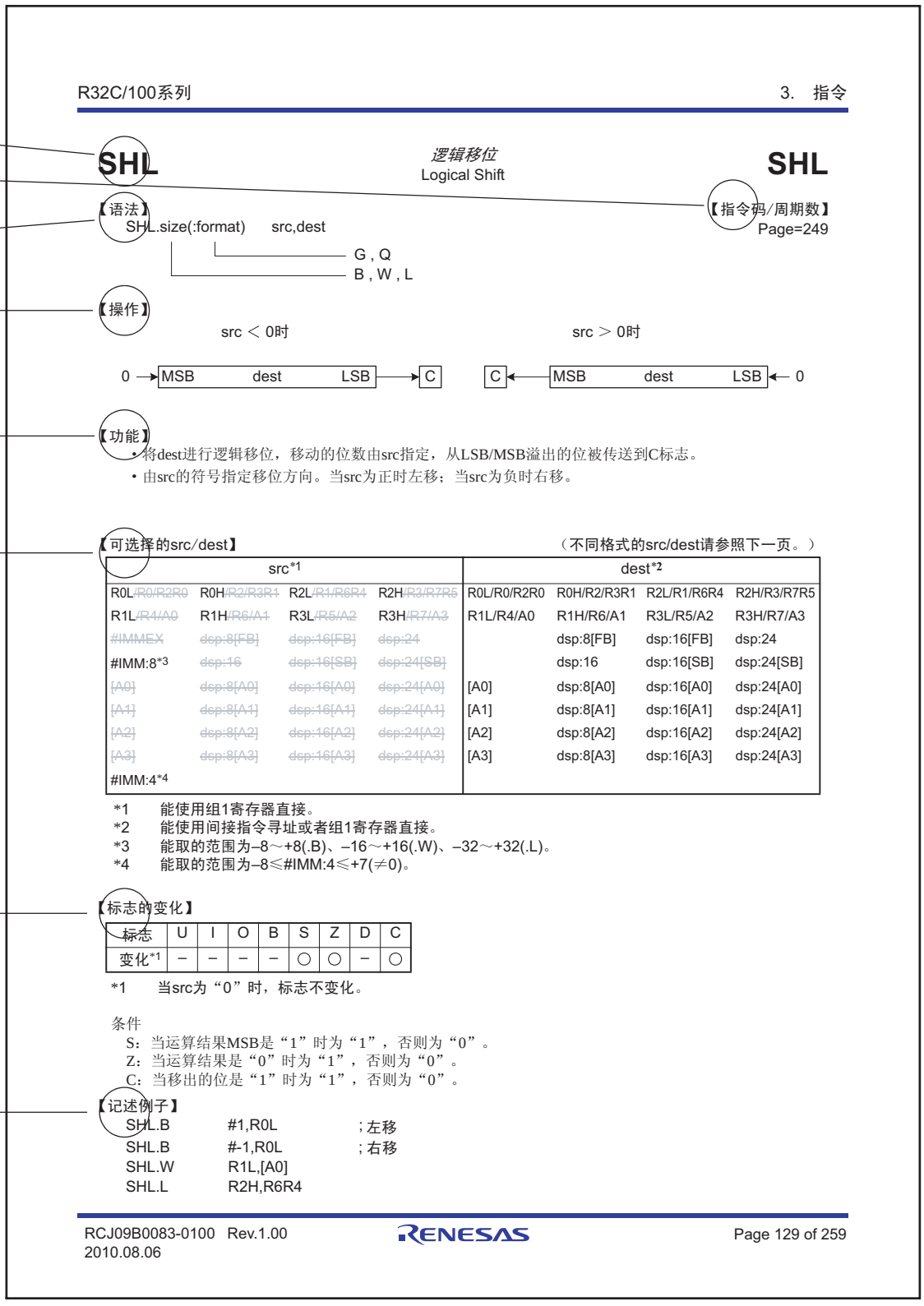
<table>
<tr><td colspan="2">寄存器直接</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,R0H
bit,R0L
bit,R1H
bit,R1L
bit,R2H
bit,R2L
bit,R3H
bit,R3L</td><td>指定的寄存器的位为运算对象。
位的位置 （bit）能指定 0 ～ 7。</td></tr>
<tr><td colspan="2">绝对</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,abs:16</td><td>abs 所示的值为运算对象的有效地址。
有效地址的范围为 00000000h ～ 0000FFFFh。
位的位置 （bit）能指定 0 ～ 7。
此寻址只能用于 BTST 指令。</td></tr>
<tr><td colspan="2">地址 0 相对</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,dsp:16
bit,dsp:24</td><td>将位移量 （dsp）所示的值进行符号扩展，扩展结果所示的地址的位为运算对象。
能指定的地址范围在 dsp:16 时为 00000000h ～ 00007FFFh 和 FFFF8000h ～ FFFFFFFFh，在 dsp:24 时为 00000000h ～ 007FFFFFh 和 FF800000h ～ FFFFFFFFh。
位的位置 （bit）能指定 0 ～ 7。</td></tr>
<tr><td colspan="2">地址寄存器间接</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,[A0]
bit,[A1]
bit,[A2]
bit,[A3]</td><td>地址寄存器 （A0/A1/A2/A3）的值所示地址的位为运算对象。
能指定的地址范围为 00000000h ～ FFFFFFFFh。
位的位置 （bit）能指定 0 ～ 7。</td></tr>
</table>

地址寄存器相对		
bit,dsp:8[A0] bit,dsp:8[A1] bit,dsp:8[A2] bit,dsp:8[A3] bit,dsp:16[A0] bit,dsp:16[A1] bit,dsp:16[A2] bit,dsp:16[A3] bit,dsp:24[A0] bit,dsp:24[A1] bit,dsp:24[A2] bit,dsp:24[A3]	将地址寄存器 （A0/A1/A2/A3）的值和位移量（dsp）的值进行带符号的加法运算，运算结果的低 32 位所示地址的位为运算对象。 能指定的地址范围为 00000000h ～ FFFFFFFFh。 位的位置 （bit）能指定 0 ～ 7。	
SB 相对		
bit,dsp:8[SB] bit,dsp:16[SB] bit,dsp:24[SB]	将静态基址寄存器 （SB）的值和位移量 （dsp）的值进行不带符号的加法运算，运算结果的低 32 位所示地址的位为运算对象。 能指定的地址范围为 00000000h ～ FFFFFFFFh。 位的位置 （bit）能指定 0 ～ 7。	
FB 相对		
bit,dsp:8[FB] bit,dsp:16[FB]	将帧基址寄存器 （FB）的值和位移量 （dsp）的值进行带符号的加法运算，运算结果的低 32 位所示地址的位为运算对象。 能指定的地址范围为 00000000h ～ FFFFFFFFh。 位的位置 （bit）能指定 0 ～ 7。	

3. 指令

3.1 本章的阅读方法

在本章中，按各指令对语法、操作、功能、可选择的 src/dest、标志的变化和记述例子进行说明。
通过以下的例子说明本章的阅读方法：



R32C/100系列 3. 指令

(1) SHL 逻辑移位 Logical Shift SHL

(2) 【指令码/周期数】 Page=249

(3) 【语法】
SHL.size(:format) src,dest
G , Q
B , W , L

(4) 【操作】
src ＜ 0时: 0 → MSB dest LSB → C
src ＞ 0时: C → MSB dest LSB → 0

(5) 【功能】
・将dest进行逻辑移位，移动的位数由src指定，从LSB/MSB溢出的位被传送到C标志。
・由src的符号指定移位方向。当src为正时左移；当src为负时右移。

(6) 【可选择的src/dest】 （不同格式的src/dest请参照下一页。）

src*1				dest*2			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM:8*3	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:4*4							

*1 能使用组1寄存器直接。
*2 能使用间接指令寻址或者组1寄存器直接。
*3 能取的范围为–8～+8(.B)、–16～+16(.W)、–32～+32(.L)。
*4 能取的范围为–8≤#IMM:4≤+7(≠0)。

(7) 【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化*1	–	–	–	–	○	○	–	○

*1 当src为“0”时，标志不变化。

条件
S：当运算结果MSB是“1”时为“1”，否则为“0”。
Z：当运算结果是“0”时为“1”，否则为“0”。
C：当移出的位是“1”时为“1”，否则为“0”。

(8) 【记述例子】

```
SHL.B   #1,R0L      ;左移
SHL.B   #-1,R0L     ;右移
SHL.W   R1L,[A0]
SHL.L   R2H,R6R4
```

RCJ09B0083-0100 Rev.1.00 2010.08.06 RENESAS Page 129 of 259

(1) 助记符

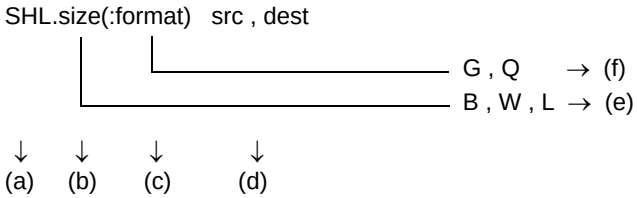
表示本页中说明的助记符。

(2) 指令码 / 周期数

表示指令码和周期数的记载页。
有关指令码和周期数，请参照此页。

(3) 语法

用符号表示指令的语法。



(a) 助记符 SHL

记述助记符。

(b) 长度说明符 .size

记述要处理的数据长度，能指定的长度如下所示：

.B 字节 （8 位）
.W 字 （16 位）
.L 长字 （32 位）

有些指令没有长度说明符。

(c) 指令格式说明符 :format

记述指令格式，也能省略。
能指定的指令格式如下所示：

:G 一般格式
:Q 快速格式
:S 短格式
:Z 零格式

有些指令没有指令格式说明符。

(d) 操作数 src, dest

记述操作数。

(e) 能指定的长度说明符

表示 (b) 中能指定的数据长度。

(f) 能指定的指令格式说明符

表示 (c) 中能指定的指令格式。

R32C/100系列 3. 指令

(1) **SHL** *逻辑移位* Logical Shift **SHL**

(2) 【指令码/周期数】 Page=249

(3) 【语法】

SHL.size(:format) src,dest

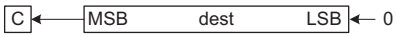
G , Q

B , W , L

(4) 【操作】

src ＜ 0时

src ＞ 0时



(5) 【功能】

・将dest进行逻辑移位，移动的位数由src指定，从LSB/MSB溢出的位被传送到C标志。

・由src的符号指定移位方向。当src为正时左移；当src为负时右移。

(6) 【可选择的src/dest】 （不同格式的src/dest请参照下一页。）

src*1				dest*2			
R0L/~~R0/R2R0~~	R0H/~~R2/R3R1~~	R2L/~~R1/R6R4~~	R2H/~~R3/R7R5~~	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/~~R4/A0~~	R1H/~~R6/A1~~	R3L/~~R5/A2~~	R3H/~~R7/A3~~	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM:8*3	~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~		dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:4*4							

*1 能使用组1寄存器直接。
*2 能使用间接指令寻址或者组1寄存器直接。
*3 能取的范围为–8～+8(.B)、–16～+16(.W)、–32～+32(.L)。
*4 能取的范围为–8≤#IMM:4≤+7(≠0)。

(7) 【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化*1	–	–	–	–	○	○	–	○

*1 当src为“0”时，标志不变化。

条件
S：当运算结果MSB是“1”时为“1”，否则为“0”。
Z：当运算结果是“0”时为“1”，否则为“0”。
C：当移出的位是“1”时为“1”，否则为“0”。

(8) 【记述例子】

```
SHL.B    #1,R0L      ; 左移
SHL.B    #-1,R0L     ; 右移
SHL.W    R1L,[A0]
SHL.L    R2H,R6R4
```

(4) 操作

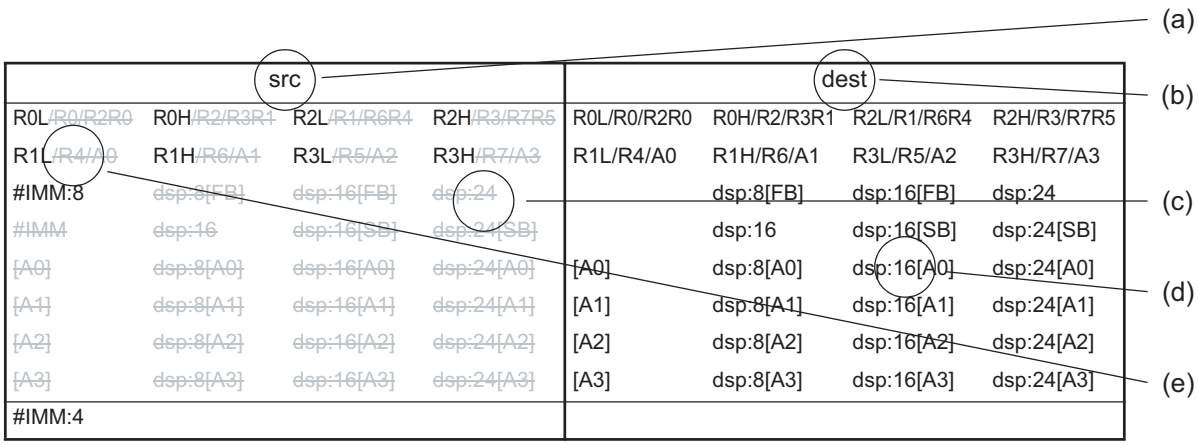
用 C 语言的记述说明指令的操作。

(5) 功能

说明指令的功能和注意事项。

(6) 可选择的 src/dest(label)

当指令有操作数时，表示能作为操作数选择的寻址方式。



src				dest			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMM:8	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:4							

(a) 能作为 src(source) 选择的项目

(b) 能作为 dest(destination) 选择的项目

(c) 不能选择的寻址

(d) 能选择的寻址

(e) 因运算长度而变化的寻址

左 （R1L） 用于运算长度为字节 （8 位）的寻址
中央 （R4） 用于运算长度为字 （16 位）的寻址
右 （A0） 用于运算长度为长字 （32 位）的寻址

(7) 标志的变化

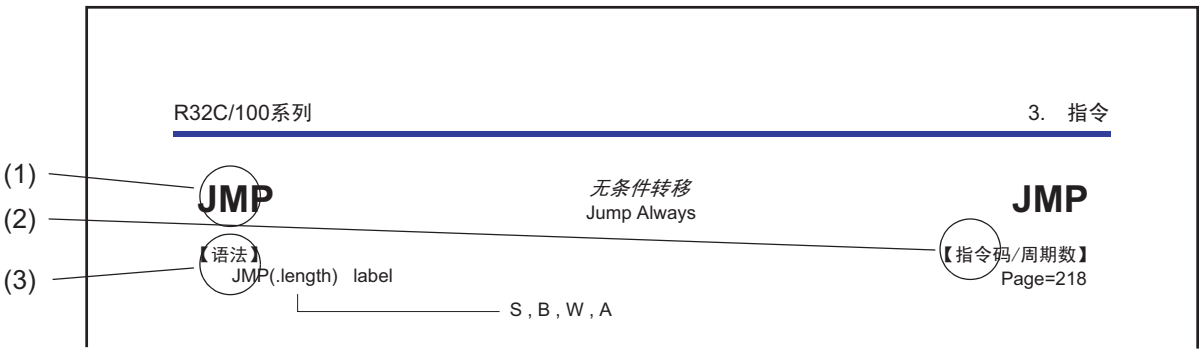
表示执行指令后的标志的变化，表中所示的符号含义如下：

“—” 不变。
“○” 因条件而变。

(8) 记述例子

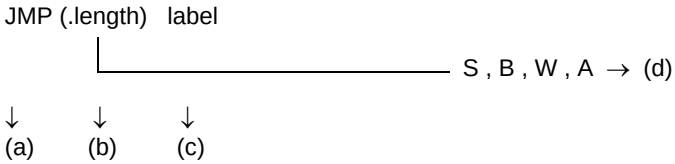
表示指令的记述例子。

通过以下的例子表示 JMP、 JMPI、 JSR、 JSRI 指令的语法：



(3) 语法

用符号表示指令的语法。



(a) 助记符 JMP

记述助记符。

(b) 转移距离说明符 .length

记述要转移的距离， JMP 指令和 JSR 指令能省略。

能指定的转移距离如下所示：

.S	3 位 PC 前方相对
.B	8 位 PC 相对
.W	16 位 PC 相对
.A	24 位 PC 相对
.L	32 位绝对

(c) 操作数 label

记述操作数。

(d) 能指定的转移距离说明符

表示 (b) 中能指定的转移距离。

3.2 指令的详细说明

从下一页开始详细说明 R32C/100 系列的各指令。

ABS

绝对值
Absolute

ABS

【语法】

ABS.size dest

B , W , L

【指令码 / 周期数】

Page=169

【操作】

```
if (dest < 0)
        dest    =    -dest;
```

【功能】

- 取 dest 的绝对值，结果保存到 dest。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：当运算前的 dest 是 –128(.B)、–32768(.W) 或者 –2147483648(.L) 时为 “1”，否则为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：为不定值。

【记述例子】

```
ABS.B       R0L
ABS.W       R2
ABS.L       R7R5
ABS.L       A0
ABS.W       mem[SB]
```

ADC

带进位的加法运算
Add with Carry

ADC

【语法】

ADC.size src,dest

B , W , L

【指令码 / 周期数】

Page=169

【操作】

dest = dest + src + C;

【功能】

- 将dest、 src和C标志相加，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、 –2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：表示不带符号的运算发生上溢。当运算结果超过 +255(.B)、 +65535(.W)、 +4294967295(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
ADC.B    #2,R0L
ADC.W    R0,R2
ADC.L    A0,R7R5
ADC.B    R3L,[A0]
ADC.W    [A1],R4
ADC.L    R2R0,[A3]
```

ADCF

进位标志的加法运算
Add Carry Flag

ADCF

【语法】
ADCF.size dest
B , W , L

【指令码 / 周期数】
Page=170

【操作】
dest = dest + C;

【功能】
- 将dest和C标志相加，结果保存到dest。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件
O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果是 “0” 时为 “1”，否则为 “0”。
C：表示不带符号的运算发生上溢。当运算结果超过 +255(.B)、 +65535(.W)、 +4294967295(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
ADCF.B    R0L
ADCF.W    R2
ADCF.L    [A3]
ADCF.W    [mem[A0]]
```

ADD

不带进位的加法运算
Add

ADD

【语法】

ADD.size(:format) src,dest

G , Q , S
B , W , L

【指令码 / 周期数】
Page=171

【操作】

dest = dest + src;

【功能】

- 将dest和src相加，结果保存到dest。

【可选择的 src/dest】 （不同格式的 src/dest 请参照下一页。）

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:3	#IMM:4			SP*2			

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 运算对象为 U 标志所示的堆栈指针。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：表示不带符号的运算发生上溢。当运算结果超过 +255(.B)、 +65535(.W)、 +4294967295(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
ADD.B     #2,R0L
ADD.W     R0,R2
ADD.L     A0,R7R5
ADD.B     R3L,[mem[A0]]
ADD.W     [[A1]],R4
ADD.L     R2R0,[[A3]]
```

【不同格式的 src/dest】

G 格式

src[*1]				dest[*1]			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMMEX:8[*3]	#IMMEX:16[*3]	#IMM:32[*3]		SP[*2*3]			

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 运算对象为 U 标志所示的堆栈指针。

*3 只能给长度说明符 （.size）指定 “.L”。

Q 格式

src	dest[*1]			
#IMM:4[*2]	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
		dsp:8[FB]	dsp:16[FB]	dsp:24
		dsp:16	dsp:16[SB]	dsp:24[SB]
	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:3[*3]	SP[*4*5]			

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 能取的范围为 –8 ≤ #IMM:4 ≤ +7。

*3 能取的范围为 #IMM:3=4, 8, 12, 16, 20。

*4 运算对象为 U 标志所示的堆栈指针。

*5 只能给长度说明符 （.size）指定 “.L”。

S 格式

src	dest[*1]			
#IMM	R0L/R0/R2R0	dsp:16	dsp:8[SB]	dsp:8[FB]

*1 能使用间接指令寻址或者组 1 寄存器直接。

ADDF

浮点加法运算
Add Floating Point

ADDF

【语法】
ADDF src,dest

【指令码 / 周期数】
Page=173

【操作】
dest = dest + src ;

【功能】
- 将src和dest进行带符号的浮点加法运算，结果保存到dest。
- 根据标志寄存器（FLG）指定的舍入模式，将运算结果进行舍入。
- 在运算结果超过最大规格化数时，根据符号，结果为正的最大值（7F7FFFFFh）或者负的最大值（FF7FFFFFh）。
- 在运算结果小于最小规格化数时，根据舍入模式，结果为零（00000000h）或者正的最小值（00800000h）或者负的最小值（80800000h）。
- 非法输入的运算结果为不定值。

【可选择的 src/dest】

src*1				dest*1			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	○	○	—	—	○	—	○	○	—	○

条件

FO：当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
FU：当非法输入或者运算结果小于最小规格化数时为 “1”，否则为 “0”。
O：当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果的全部位是 “0” 时为 “1”，否则为 “0”。
C：为不定值。

【记述例子】

```
ADDF    R2R0,R3R1
ADDF    [A1],R2R0
ADDF    mem[FB],R3R1
```

ADSF

符号标志的加法运算
Add Sign Flag

ADSF

【语法】

ADSF.size dest

B , W , L

【指令码 / 周期数】

Page=174

【操作】

dest = dest + S;

【功能】

- 将dest和S标志相加，结果保存到dest。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、 –2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：表示不带符号的运算发生上溢。当运算结果超过 +255(.B)、 +65535(.W)、 +4294967295(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
ADSF.B    R0L
ADSF.W    R2
ADSF.L    [A3]
ADSF.W    mem[A0]
```

AND

逻辑与
And Logical

AND

【语法】

AND.size(:format) src,dest

G , S
B , W , L

【指令码 / 周期数】
Page=175

【操作】

dest = dest & src;

【功能】

- 取 dest 和 src 的逻辑与，结果保存到 dest。

【可选择的 src/dest】 （不同格式的 src/dest 请参照下一页。）

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
				dsp:8[SB]			

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
AND.B     #2,R0L
AND.W     R0,R2
AND.L     A0,R7R5
AND.B     R3L,[mem[A0]]
AND.W     [[A1]],R4
AND.L     R2R0,[[A3]]
```

【不同格式的 src/dest】

G 格式

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

S 格式

src	dest*1			
#IMM	R0L/R0/R2R0	dsp:16	dsp:8[SB]	dsp:8[FB]

*1 能使用间接指令寻址或者组 1 寄存器直接。

BCLR

位清除
Clear a Bit

BCLR

【语法】
BCLR dest

【指令码 / 周期数】
Page=176

【操作】
dest = 0;

【功能】
- 将“0”保存到dest。

【可选择的 dest】

dest*1			
bit,R0L	bit,R0H	bit,R2L	bit,R2H
bit,R1L	bit,R1H	bit,R3L	bit,R3H
	bit,dsp:8[FB]	bit,dsp:16[FB]	bit,dsp:24
	bit,dsp:16	bit,dsp:16[SB]	bit,dsp:24[SB]
bit,[A0]	bit,dsp:8[A0]	bit,dsp:16[A0]	bit,dsp:24[A0]
bit,[A1]	bit,dsp:8[A1]	bit,dsp:16[A1]	bit,dsp:24[A1]
bit,[A2]	bit,dsp:8[A2]	bit,dsp:16[A2]	bit,dsp:24[A2]
bit,[A3]	bit,dsp:8[A3]	bit,dsp:16[A3]	bit,dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
```
BCLR    1,R0L
BCLR    4,R2H
BCLR    2,[A3]
BCLR    7,mem[SB]
```

BITINDEX

位变址
Index next Bit Instruction

BITINDEX

【语法】

BITINDEX.size src

B , W , L

【指令码 / 周期数】

Page=177

【操作】

【功能】

- 修改下一条位指令的寻址方式。
- 在此指令之后不能马上接受中断请求。
- 由src指定的操作数为下一条指令的src或者dest的变址值 （bit）。

【可选择的 src】

src*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
BITINDEX.B   R0L
BITINDEX.W   R2
BITINDEX.L   [A0]
BITINDEX.W   mem[A1]
```

BM*Cnd*

条件位传送
Move a Bit Conditionally

BM*Cnd*

【语法】
BM*Cnd* dest

【指令码 / 周期数】
Page=177

【操作】

```
if (true)
        dest =   1;
else
        dest =   0;
```

【功能】

- 将由*Cnd*所示条件的真假值传送到dest。真时传送 “1”，假时传送 “0”。
- *Cnd*有以下的种类：

Cnd	条件		式	*Cnd*	条件		式
GEU/C	C == 1	大于等于 / C 标志为 “1”	≤	LTU/NC	C == 0	小于 / C 标志为 “0”	＞
EQ/Z	Z == 1	等于 / Z 标志为 “1”	＝	NE/NZ	Z == 0	不等于 / Z 标志为 “0”	≠
GTU	C & ~Z == 1	大于	＜	LEU	C & ~Z == 0	小于等于	≥
PZ	S == 0	大于等于 “0”	0 ≤	N	S == 1	负	0 ＞
GE	S ^ O == 0	等于或者带符号的大于	≤	LE	(S ^ O) | Z == 1	等于或者带符号的小于	≥
GT	(S ^ O) | Z == 0	带符号的大于	＜	LT	S ^ O == 1	带符号的小于	＞
O	O == 1	O 标志为 “1”		NO	O == 0	O 标志为 “0”	

【可选择的 dest】

dest*1			
bit,R0L	bit,R0H	bit,R2L	bit,R2H
bit,R1L	bit,R1H	bit,R3L	bit,R3H
	bit,dsp:8[FB]	bit,dsp:16[FB]	bit,dsp:24
	bit,dsp:16	bit,dsp:16[SB]	bit,dsp:24[SB]
bit,[A0]	bit,dsp:8[A0]	bit,dsp:16[A0]	bit,dsp:24[A0]
bit,[A1]	bit,dsp:8[A1]	bit,dsp:16[A1]	bit,dsp:24[A1]
bit,[A2]	bit,dsp:8[A2]	bit,dsp:16[A2]	bit,dsp:24[A2]
bit,[A3]	bit,dsp:8[A3]	bit,dsp:16[A3]	bit,dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
BMC     1,R0L
BMLT    4,R2H
BMN     2,[A3]
BMNO    7,mem[SB]
```

BNOT

位取反
Not a Bit

BNOT

【语法】
BNOT dest

【指令码 / 周期数】
Page=178

【操作】
dest = ~dest;

【功能】
- 将dest取反，结果保存到dest。

【可选择的 dest】

dest*1			
bit,R0L	bit,R0H	bit,R2L	bit,R2H
bit,R1L	bit,R1H	bit,R3L	bit,R3H
	bit,dsp:8[FB]	bit,dsp:16[FB]	bit,dsp:24
	bit,dsp:16	bit,dsp:16[SB]	bit,dsp:24[SB]
bit,[A0]	bit,dsp:8[A0]	bit,dsp:16[A0]	bit,dsp:24[A0]
bit,[A1]	bit,dsp:8[A1]	bit,dsp:16[A1]	bit,dsp:24[A1]
bit,[A2]	bit,dsp:8[A2]	bit,dsp:16[A2]	bit,dsp:24[A2]
bit,[A3]	bit,dsp:8[A3]	bit,dsp:16[A3]	bit,dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
```
BNOT    1,R0L
BNOT    4,R2H
BNOT    2,[A3]
BNOT    7,mem[SB]
```

BRK

调试中断
Break

BRK

【语法】
BRK

【指令码 / 周期数】
Page=178

【操作】

地址 FFFFFFE7h 不为 “FFh” 的情况

```
SP    =  SP - 4;
*SP   =  FLG;
SP    =  SP - 4;
*SP   =  PC + 1;
PC    =  *(FFFFFFE4h);
```

地址 FFFFFFE7h 为 “FFh” 的情况

```
SP    =  SP - 4;
*SP   =  FLG;
SP    =  SP - 4;
*SP   =  PC + 1;
PC    =  *IntBase;
```

【功能】

- 发生BRK中断。
- BRK中断是非屏蔽中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	○	○	—	—	—	—	○	—

条件

- U：为 “0”。
- I：为 “0”。
- D：为 “0”。

执行 BRK 指令前的标志被压栈。

【记述例子】

```
BRK
```

BRK2

调试中断2
Break2

BRK2

【语法】
BRK2

【指令码 / 周期数】
Page=178

【操作】

```
SP    =  SP - 4;
*SP   =  FLG;
SP    =  SP - 4;
*SP   =  PC + 1;
PC    =  *(long *)0x0004C000;
```

【功能】

- 此指令是调试程序专用的指令，用户程序不能使用。
- 发生BRK2中断。
- BRK2中断是非屏蔽中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	○	○	—	—	—	—	○	—

条件

U：为 “0”。
I：为 “0”。
D：为 “0”。

执行 BRK2 指令前的标志被压栈。

【记述例子】

```
BRK2
```

BSET

位置位
Set a Bit

BSET

【语法】
BSET dest

【指令码 / 周期数】
Page=179

【操作】
dest = 1;

【功能】
- 将“1”保存到dest。

【可选择的 dest】

dest*1			
bit,R0L	bit,R0H	bit,R2L	bit,R2H
bit,R1L	bit,R1H	bit,R3L	bit,R3H
	bit,dsp:8[FB]	bit,dsp:16[FB]	bit,dsp:24
	bit,dsp:16	bit,dsp:16[SB]	bit,dsp:24[SB]
bit,[A0]	bit,dsp:8[A0]	bit,dsp:16[A0]	bit,dsp:24[A0]
bit,[A1]	bit,dsp:8[A1]	bit,dsp:16[A1]	bit,dsp:24[A1]
bit,[A2]	bit,dsp:8[A2]	bit,dsp:16[A2]	bit,dsp:24[A2]
bit,[A3]	bit,dsp:8[A3]	bit,dsp:16[A3]	bit,dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
BSET    1,R0L
BSET    4,R2H
BSET    2,[A3]
BSET    7,mem[SB]
```

BTST

位测试
Test a Bit

BTST

【语法】

BTST (:format) src

G , S

【指令码 / 周期数】

Page=179

【操作】

Z = ~src;

C = src;

【功能】

- 将src取反后的值传送到Z标志，并且将src传送到C标志。

【可选择的 src】

G 格式

src*1			
bit,R0L	bit,R0H	bit,R2L	bit,R2H
bit,R1L	bit,R1H	bit,R3L	bit,R3H
	bit,dsp:8[FB]	bit,dsp:16[FB]	bit,dsp:24
	bit,dsp:16	bit,dsp:16[SB]	bit,dsp:24[SB]
bit,[A0]	bit,dsp:8[A0]	bit,dsp:16[A0]	bit,dsp:24[A0]
bit,[A1]	bit,dsp:8[A1]	bit,dsp:16[A1]	bit,dsp:24[A1]
bit,[A2]	bit,dsp:8[A2]	bit,dsp:16[A2]	bit,dsp:24[A2]
bit,[A3]	bit,dsp:8[A3]	bit,dsp:16[A3]	bit,dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

S 格式

src
bit,abs:16

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	○	—	○

条件

Z：当 src 是 “0” 时为 “1”，是 “1” 时为 “0”。

C：当 src 是 “1” 时为 “1”，是 “0” 时为 “0”。

【记述例子】

```
BTST    1,R0L
BTST    4,R2H
BTST    2,[A3]
BTST    7,mem
```

BTSTC

位测试和清除
Test a Bit and Clear

BTSTC

【语法】
BTSTC dest

【指令码 / 周期数】
Page=180

【操作】
Z = ~dest;
C = dest;
dest = 0;

【功能】
- 将dest取反后的值传送到Z标志，并且将dest传送到C标志，然后将 “0”保存到dest。
- 不能将此指令用于SFR区。

【可选择的 dest】

dest*1			
bit,R0L	bit,R0H	bit,R2L	bit,R2H
bit,R1L	bit,R1H	bit,R3L	bit,R3H
	bit,dsp:8[FB]	bit,dsp:16[FB]	bit,dsp:24
	bit,dsp:16	bit,dsp:16[SB]	bit,dsp:24[SB]
bit,[A0]	bit,dsp:8[A0]	bit,dsp:16[A0]	bit,dsp:24[A0]
bit,[A1]	bit,dsp:8[A1]	bit,dsp:16[A1]	bit,dsp:24[A1]
bit,[A2]	bit,dsp:8[A2]	bit,dsp:16[A2]	bit,dsp:24[A2]
bit,[A3]	bit,dsp:8[A3]	bit,dsp:16[A3]	bit,dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	○	—	○

条件
Z：当 dest 是 “0”时为 “1”，是 “1”时为 “0”。
C：当 dest 是 “1”时为 “1”，是 “0”时为 “0”。

【记述例子】
```
BTSTC   1,R0L
BTSTC   4,R2H
BTSTC   2,[A3]
BTSTC   7,mem[SB]
```

BTSTS

位测试和置位
Test a Bit and Set

BTSTS

【语法】
BTSTS dest

【指令码 / 周期数】
Page=180

【操作】
Z = ~dest;
C = dest;
dest = 1;

【功能】
- 将dest取反后的值传送到Z标志，并且将dest传送到C标志，然后将 “1” 保存到dest。
- 不能将此指令用于SFR区。

【可选择的 dest】

dest*1			
bit,R0L	bit,R0H	bit,R2L	bit,R2H
bit,R1L	bit,R1H	bit,R3L	bit,R3H
	bit,dsp:8[FB]	bit,dsp:16[FB]	bit,dsp:24
	bit,dsp:16	bit,dsp:16[SB]	bit,dsp:24[SB]
bit,[A0]	bit,dsp:8[A0]	bit,dsp:16[A0]	bit,dsp:24[A0]
bit,[A1]	bit,dsp:8[A1]	bit,dsp:16[A1]	bit,dsp:24[A1]
bit,[A2]	bit,dsp:8[A2]	bit,dsp:16[A2]	bit,dsp:24[A2]
bit,[A3]	bit,dsp:8[A3]	bit,dsp:16[A3]	bit,dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	○	—	○

条件
Z：当 dest 是 “0” 时为 “1”，是 “1” 时为 “0”。
C：当 dest 是 “1” 时为 “1”，是 “0” 时为 “0”。

【记述例子】
```
BTSTS   1,R0L
BTSTS   4,R2H
BTSTS   2,[A3]
BTSTS   7,mem[SB]
```

CLIP

钳制
Clip

CLIP

【语法】

CLIP.size src1,src2,dest

B , W , L

【指令码 / 周期数】

Page=181

【操作】

```
if (dest < src1)
        dest  =  src1;
else if (dest > src2)
        dest  =  src2;
```

【功能】

- 钳制dest的值，使src1≤dest≤src2。
- 带符号进行比较。

【可选择的 src/dest】

src1 src2	dest*1			
#IMM	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
		dsp:8[FB]	dsp:16[FB]	dsp:24
		dsp:16	dsp:16[SB]	dsp:24[SB]
	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
CLIP.B      #-32,#32,R0L
CLIP.W      #1000,#4000,R2
CLIP.L      #-100000,#100000,R7R5
CLIP.W      #0,#10000,mem[A0]
```

CMP

比较
Compare

CMP

【语法】

CMP.size(:format) src,dest

G , Q, S
B , W , L

【指令码 / 周期数】

Page=181

【操作】

dest - src;

【功能】

- 根据dest减去src的运算结果，标志寄存器 （FLG）的各标志发生变化。

【可选择的 src/dest】 （不同格式的 src/dest 请参照下一页。）

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:4							

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：当不带符号的运算结果大于等于 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
CMP.B     #2,R0L
CMP.W     R0,R2
CMP.L     A0,R7R5
CMP.B     R3L,[mem[A0]]
CMP.W     [[A1]],R4
CMP.L     R2R0,[[A3]]
```

【不同格式的 src/dest】

G 格式

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

Q 格式

src	dest*1			
#IMM:4*2	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
		dsp:8[FB]	dsp:16[FB]	dsp:24
		dsp:16	dsp:16[SB]	dsp:24[SB]
	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 能取的范围为 –8 ≤ #IMM:4 ≤ +7。

S 格式

src	dest*1			
#IMM	R0L/R0/R2R0	dsp:16	dsp:8[SB]	dsp:8[FB]

*1 能使用间接指令寻址或者组 1 寄存器直接。

CMPF

浮点比较
Compare Floating Point

CMPF

【语法】
CMPF src,dest

【指令码 / 周期数】
Page=183

【操作】
dest - src ;

【功能】
- 根据dest减去src的运算结果，标志寄存器 （FLG）的各标志发生变化。
- 非法输入的运算结果为不定值。

【可选择的 src/dest】

src*1				dest*1			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	○	○	—	—	○	—	○	○	—	○

条件
FO：当非法输入时为 “1”，否则为 “0”。
FU：当非法输入时为 “1”，否则为 “0”。
O：当非法输入时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果的全部位是 “0” 时为 “1”，否则为 “0”。
C：为不定值。

【记述例子】

```
CMPF    R2R0,R3R1
CMPF    [A1],R2R0
CMPF    mem[FB],R3R1
```

CNVIF

整数 → 浮点数的转换
Convert Integer to Floating Point

CNVIF

【语法】
CNVIF src,dest

【指令码 / 周期数】
Page=184

【操作】
dest = (float) src;

【功能】
- 将src转换为单精度浮点数。
- 根据标志寄存器（FLG）指定的舍入模式，将运算结果进行舍入。

【可选择的 src/dest】

src*1				dest*1			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	○	○	—	—	○	—	○	○	—	○

条件
FO：为 “0”。
FU：为 “0”。
O：为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果的全部位是 “0” 时为 “1”，否则为 “0”。
C：为不定值。

【记述例子】
```
CNVIF    R2R0,R3R1
CNVIF    [A1],R2R0
CNVIF    mem[FB],R3R1
```

DADC

带进位的 10 进制加法运算
Add Decimal with Carry

DADC

【语法】

DADC.size src,dest

B , W , L

【指令码 / 周期数】

Page=185

【操作】

dest = dest + src + C;

【功能】

- 将dest、 src和C标志进行10进制加法运算，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	○

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：当运算结果超过 +99(.B)、 +9999(.W)、 +99999999(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
DADC.B    #12,R0L
DADC.W    R0,R2
DADC.L    A0,R7R5
DADC.B    R3L,[A0]
DADC.W    [A1],R4
DADC.L    R2R0,[A3]
```

DADD

10 进制加法运算
Add Decimal

DADD

【语法】
DADD.size src,dest
B , W , L

【指令码 / 周期数】
Page=186

【操作】
dest = dest + src;

【功能】
- 将dest和src进行10进制加法运算，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	○

条件
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果是 “0” 时为 “1”，否则为 “0”。
C：当运算结果超过 +99(.B)、 +9999(.W)、 +99999999(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
DADD.B    #12,R0L
DADD.W    R0,R2
DADD.L    A0,R7R5
DADD.B    R3L,[A0]
DADD.W    [A1],R4
DADD.L    R2R0,[A3]
```

DEC

递减
Decrement

DEC

【语法】

DEC.size dest

B , W , L

【指令码 / 周期数】

Page=187

【操作】

dest = dest - 1;

【功能】

- dest减1，结果保存到dest。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
DEC.B    R0L
DEC.W    R2
DEC.L    R7R5
DEC.L    A0
DEC.W    mem[SB]
```

DIV

带符号的除法运算
Signed Divide

DIV

【语法】

DIV.size src,dest

B , W , L

【指令码 / 周期数】

Page=188

【操作】

dest = dest / src;

【功能】

- dest除以（带符号的除法） src，商保存到dest。将商向零方向舍入。
- 如果给长度说明符（.size）指定“.B”， src和dest都以8位进行运算，并且用8位保存结果。
- 如果给长度说明符（.size）指定“.W”， src和dest都以16位进行运算，并且用16位保存结果。
- 如果给长度说明符（.size）指定“.L”， src和dest都以32位进行运算，并且用32位保存结果。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当除数（src）是“0”时，或者当运算结果超过 –128(.B) 或者 +127(.B)、–32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为“1”，否则为“0”。

【记述例子】

```
DIV.B    #12,R0L
DIV.W    R0,R2
DIV.L    A0,R7R5
DIV.B    R3L,[A0]
DIV.W    [A1],R4
DIV.L    R2R0,[A3]
```

DIVF

浮点除法运算
Divide Floating Piont

DIVF

【语法】
DIVF src,dest

【指令码 / 周期数】
Page=189

【操作】
dest = dest / src ;

【功能】
- dest除以 （带符号的除法） src，商保存到dest。
- 根据标志寄存器 （FLG）指定的舍入模式，将商进行舍入。
- 在运算结果超过最大规格化数时，根据符号，结果为正的最大值 （7F7FFFFFh）或者负的最大值 （FF7FFFFFh）。
- 在运算结果小于最小规格化数时，根据舍入模式，结果为零 （00000000h）或者正的最小值 （00800000h）或者负的最小值 （80800000h）。
- 非法输入的运算结果为不定值。

【可选择的 src/dest】

src*1				dest*1			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	○	○	—	—	○	—	○	○	—	○

条件
FO：当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
FU：当非法输入或者运算结果小于最小规格化数时为 “1”，否则为 “0”。
O：当除数 （src）是 “0” 时，或者当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果的全部位是 “0” 时为 “1”，否则为 “0”。
C：为不定值。

【记述例子】

```
DIVF    R2R0,R3R1
DIVF    [A1],R2R0
DIVF    mem[FB],R3R1
```

DIVU

不带符号的除法运算
Unsigned Divide

DIVU

【语法】

DIVU.size src,dest

B , W , L

【指令码 / 周期数】

Page=190

【操作】

dest = dest / src;

【功能】

- dest除以（不带符号的除法） src，商保存到dest。将商向零方向舍入。
- 如果给长度说明符（.size）指定“.B”， src和dest都以8位进行运算，并且用8位保存结果。
- 如果给长度说明符（.size）指定“.W”， src和dest都以16位进行运算，并且用16位保存结果。
- 如果给长度说明符（.size）指定“.L”， src和dest都以32位进行运算，并且用32位保存结果。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当除数（src）是“0”时为“1”，否则为“0”。

【记述例子】

```
DIVU.B    #12,R0L
DIVU.W    R0,R2
DIVU.L    A0,R7R5
DIVU.B    R3L,[A0]
DIVU.W    [A1],R4
DIVU.L    R2R0,[A3]
```

DIVX

带符号的除法运算
Signed Divide extra

DIVX

【语法】

DIVX.size src,dest

B , W , L

【指令码 / 周期数】

Page=191

【操作】

dest = dest / src;

【功能】

- dest除以（带符号的除法） src，商保存到dest。将商向–∞方向舍入。
- 如果给长度说明符 （.size）指定 “.B”， src和dest都以8位进行运算，并且用8位保存结果。
- 如果给长度说明符 （.size）指定 “.W”， src和dest都以16位进行运算，并且用16位保存结果。
- 如果给长度说明符 （.size）指定 “.L”， src和dest都以32位进行运算，并且用32位保存结果。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当除数 （src）是 “0” 时，或者当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
DIVX.B    #12,R0L
DIVX.W    R0,R2
DIVX.L    A0,R7R5
DIVX.B    R3L,[A0]
DIVX.W    [A1],R4
DIVX.L    R2R0,[A3]
```

DSBB

带借位的 10 进制减法运算
Subtract Decimal with Borrow

DSBB

【语法】

DSBB.size src,dest

B , W , L

【指令码 / 周期数】

Page=192

【操作】

dest = dest - src - ~C;

【功能】

- 从dest减去 （10进制减法） src和C标志被取反的值，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	○

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：当运算结果大于等于 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
DSBB.B    #12,R0L
DSBB.W    R0,R2
DSBB.L    A0,R7R5
DSBB.B    R3L,mem[A0]
DSBB.W    [A1],R4
DSBB.L    R2R0,[A3]
```

DSUB

10 进制减法运算
Subtract Decimal

DSUB

【语法】

DSUB.size src,dest

B , W , L

【指令码 / 周期数】

Page=193

【操作】

dest = dest - src;

【功能】

- 从dest减去 （10进制减法） src，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	○

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：当运算结果大于等于 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
DSUB.B    #12,R0L
DSUB.W    R0,R2
DSUB.L    A0,R7R5
DSUB.B    R3L,[A0]
DSUB.W    mem[A1],R4
DSUB.L    R2R0,[A3]
```

EDIV

带符号的除法运算
Extended Signed Divide with Remainder

EDIV

【语法】

EDIV.size src,dest

B , W , L

【指令码 / 周期数】

Page=194

【操作】

destL （商） = dest / src;
destH （余数） = dest % src;

【功能】

- dest除以 （带符号的除法） src，商保存到dest的低位，余数保存到dest的高位。将商向零方向舍入，余数的符号和被除数 （dest）的符号相同。
- 如果给长度说明符 （.size）指定 “.B”， src就以8位、 dest就以16位进行运算，并且分别用8位保存商和余数。能给dest指定R0 （R0H:R0L）、 R1 （R1H:R1L）、 R2 （R2H:R2L）、 R3 （R3H:R3L）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.W”， src就以16位、 dest就以32位进行运算，并且分别用16位保存商和余数。能给dest指定R2R0 （R2:R0）、 R3R1 （R3:R1）、 R6R4 （R6:R4）、 R7R5 （R7:R5）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.L”， src就以32位、 dest就以64位进行运算，并且分别用32位保存商和余数。能给dest指定R3R1R2R0 （R3R1:R2R0）、 R7R5R6R4 （R7R5:R6R4）、 A1A0 （A1:A0）、 A3A2 （A3:A2）共4种寄存器。

【可选择的 src/dest】

src*1				dest*1	
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0/R2R0/R3R1R2R0	R2/R3R1/R7R5R6R4
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1/R6R4/A1A0	R3/R7R5/A3A2
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]		
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]		
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]		
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]		

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当除数 （src）是 “0” 时，或者当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、 –2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
EDIV.B      #12,R0          ; R0H:R0L ÷ 12
EDIV.W      R0,R3R1         ; R3:R1 ÷ R0
EDIV.L      A0,R7R5R6R4     ; R7R5:R6R4 ÷ A0
EDIV.W      [A1],R2R0       ; R2:R0 ÷ [A1]
```

EDIVU

不带符号的除法运算
Extended Unsigned Divide with Remainder

EDIVU

【语法】

EDIVU.size src,dest

B , W , L

【指令码 / 周期数】

Page=194

【操作】

destL （商） = dest / src;
destH （余数） = dest % src;

【功能】

- dest除以 （不带符号的除法） src，商保存到dest的低位，余数保存到dest的高位。
- 如果给长度说明符 （.size）指定 “.B”， src就以8位， dest就以16位进行运算，并且分别用8位保存商和余数。能给dest指定R0 （R0H:R0L）、 R1 （R1H:R1L）、 R2 （R2H:R2L）、 R3 （R3H:R3L）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.W”， src就以16位， dest就以32位进行运算，并且分别用16位保存商和余数。能给dest指定R2R0 （R2:R0）、 R3R1 （R3:R1）、 R6R4 （R6:R4）、 R7R5 （R7:R5）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.L”， src就以32位， dest就以64位进行运算，并且分别用32位保存商和余数。能给dest指定R3R1R2R0 （R3R1:R2R0）、 R7R5R6R4 （R7R5:R6R4）、 A1A0 （A1:A0）、 A3A2 （A3:A2）共4种寄存器。

【可选择的 src/dest】

src*1				dest*1	
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0/R2R0/R3R1R2R0	R2/R3R1/R7R5R6R4
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1/R6R4/A1A0	R3/R7R5/A3A2
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]		
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]		
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]		
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]		

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当运算结果商超过 8 位 (.B)、 16 位 (.W)、 32 位 (.L)，或者当除数 （src）是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
EDIVU.B     #12,R0          ; R0H:R0L ÷ 12
EDIVU.W     R0,R3R1         ; R3:R1 ÷ R0
EDIVU.L     A0,R7R5R6R4     ; R7R5:R6R4 ÷ A0
EDIVU.W     [A1],R2R0       ; R2:R0 ÷ [A1]
```

EDIVX

带符号的除法运算
Extended Signed Divide extra with Remainder

EDIVX

【语法】

EDIVX.size src,dest

B , W , L

【指令码 / 周期数】

Page=195

【操作】

destL （商） = dest / src;
destH （余数） = dest % src;

【功能】

- dest除以 （带符号的除法） src，商保存到dest的低位，余数保存到dest的高位。将商向–∞方向舍入，余数的符号和除数 （src）的符号相同。
- 如果给长度说明符 （.size）指定 “.B”， src就以8位， dest就以16位进行运算，并且分别用8位保存商和余数。能给dest指定R0 （R0H:R0L）、 R1 （R1H:R1L）、 R2 （R2H:R2L）、 R3 （R3H:R3L）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.W”， src就以16位， dest就以32位进行运算，并且分别用16位保存商和余数。能给dest指定R2R0 （R2:R0）、 R3R1 （R3:R1）、 R6R4 （R6:R4）、 R7R5 （R7:R5）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.L”， src就以32位， dest就以64位进行运算，并且分别用32位保存商和余数。能给dest指定R3R1R2R0 （R3R1:R2R0）、 R7R5R6R4 （R7R5:R6R4）、 A1A0 （A1:A0）、 A3A2 （A3:A2）共4种寄存器。

【可选择的 src/dest】

src*1				dest*1	
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0/R2R0/R3R1R2R0	R2/R3R1/R7R5R6R4
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1/R6R4/A1A0	R3/R7R5/A3A2
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]		
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]		
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]		
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]		

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当除数 （src）是 “0” 时，或者当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、 –2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
EDIVX.B     #12,R0          ; R0H:R0L ÷ 12
EDIVX.W     R0,R3R1         ; R3:R1 ÷ R0
EDIVX.L     A0,R7R5R6R4     ; R7R5:R6R4 ÷ A0
EDIVX.W     [A1],R2R0       ; R2:R0 ÷ [A1]
```

EMUL

带符号的乘法运算
Extended Signed Multiply

EMUL

【语法】

EMUL.size src,dest

B , W , L

【指令码 / 周期数】

Page=195

【操作】

destH:dest = dest * src;

【功能】

- 将src和dest进行带符号的乘法运算，结果保存到destH:dest。
- 如果给长度说明符 （.size）指定 “.B”， src和dest都以8位进行运算，并且用16位保存结果。能给dest指定R0L （R0H:R0L）、 R1L （R1H:R1L）、 R2L （R2H:R2L）、 R3L （R3H:R3L）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.W”， src和dest都以16位进行运算，并且用32位保存结果。能给dest指定R0 （R2:R0）、 R1 （R3:R1）、 R4 （R6:R4）、 R5 （R7:R5）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.L”， src和dest都以32位进行运算，并且用64位保存结果。能给dest指定R2R0 （R3R1:R2R0）、 R6R4 （R7R5:R6R4）、 A0 （A1:A0）、 A2 （A3:A2）共4种寄存器。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	~~R0H/R2/R3R1~~	R2L/R1/R6R4	~~R2H/R3/R7R5~~
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	~~R1H/R6/A1~~	R3L/R5/A2	~~R3H/R7/A3~~
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24				
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]				
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]				
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]				
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]				
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]				

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
EMUL.B    #12,R0L      ; R0L× 12 → R0H:R0L (=R0)
EMUL.W    R0,R1        ; R1 × R0 → R3:R1 (=R3R1)
EMUL.L    A0,R6R4      ; R6R4 × A0 → R7R5:R6R4
EMUL.W    [A1],R0      ; R0 × [A1] → R2:R0 (=R2R0)
```

EMULU

不带符号的乘法运算
Extended Unsigned Multiply

EMULU

【语法】

EMULU.size src,dest

B , W , L

【指令码 / 周期数】

Page=196

【操作】

dest = dest * src;

【功能】

- 将src和dest进行不带符号的乘法运算，结果保存到dest。
- 如果给长度说明符 （.size）指定 “.B”， src和dest都以8位进行运算，并且用16位保存结果。能给dest指定R0L （R0H:R0L）、 R1L （R1H:R1L）、 R2L （R2H:R2L）、 R3L （R3H:R3L）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.W”， src和dest都以16位进行运算，并且用32位保存结果。能给dest指定R0 （R2:R0）、 R1 （R3:R1）、 R4 （R6:R4）、 R5 （R7:R5）共4种寄存器。
- 如果给长度说明符 （.size）指定 “.L”， src和dest都以32位进行运算，并且用64位保存结果。能给dest指定R2R0 （R3R1:R2R0）、 R6R4 （R7R5:R6R4）、 A0 （A1:A0）、 A2 （A3:A2）共4种寄存器。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	~~R0H/R2/R3R1~~	R2L/R1/R6R4	~~R2H/R3/R7R5~~
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	~~R1H/R6/A1~~	R3L/R5/A2	~~R3H/R7/A3~~
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24				
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]				
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]				
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]				
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]				
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]				

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
EMULU.B   #12,R0L      ; R0L × 12 → R0H:R0L (=R0)
EMULU.W   R0,R1        ; R1 × R0 → R3:R1 (=R3R1)
EMULU.L   A0,R6R4      ; R6R4 × A0 → R7R5:R6R4
EMULU.W   [A1],R0      ; R0 × [A1] → R2:R0 (=R2R0)
```

ENTER

栈帧的生成
Enter and Create Stack Frame

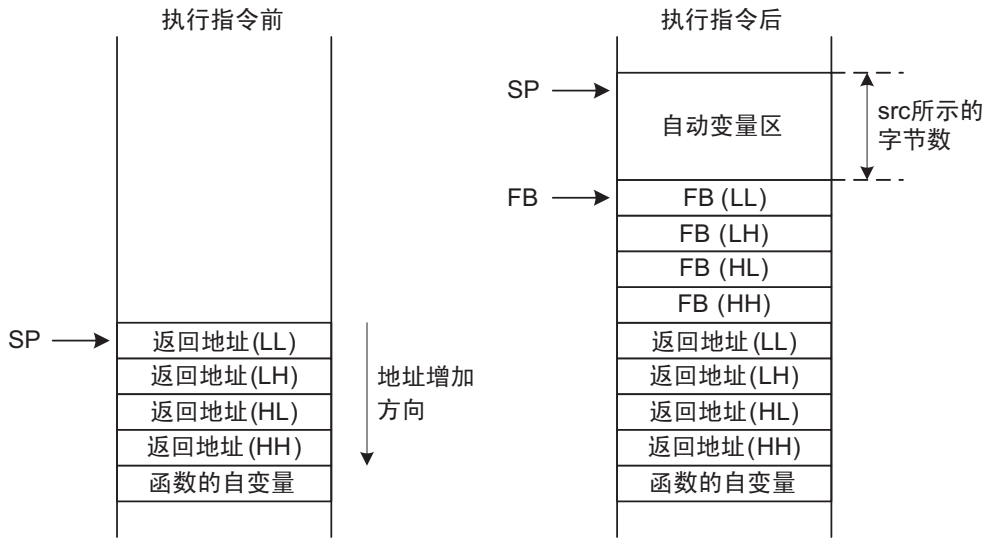
ENTER

【语法】
ENTER src

【指令码 / 周期数】
Page=196

【操作】
SP = SP - 4;
*SP = FB;
FB = SP;
SP = SP - src;

【功能】
- 生成栈帧， src 是指栈帧的大小。
- 在子程序的起始位置执行 ENTER 指令前后的堆栈区状态如下所示：



【可选择的 src】

src	
#IMM:8*1	#IMM:16*1

*1 必须给 #IMM 设定 4 的倍数。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
ENTER #12

EXITD

栈帧的释放
Exit and Deallocate Stack Frame

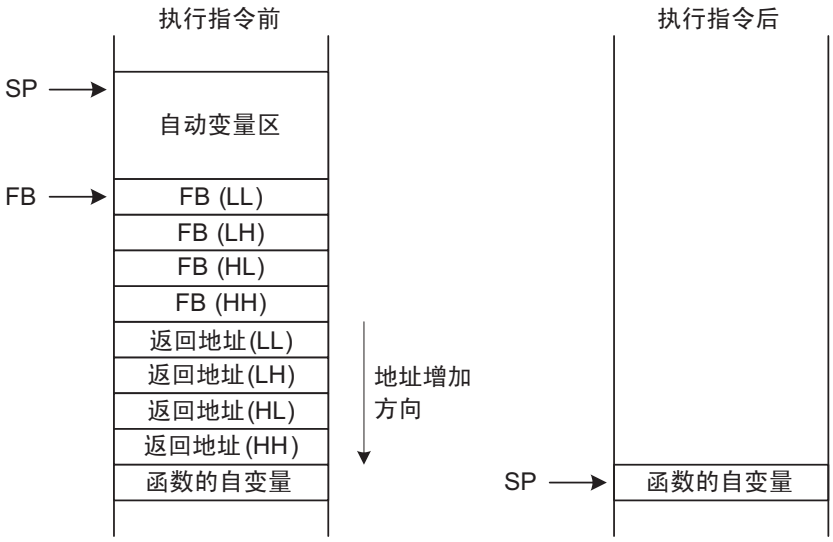
EXITD

【语法】
EXITD

【指令码 / 周期数】
Page=197

【操作】
SP = FB;
FB = *SP;
SP = SP + 4;
PC = *SP;
SP = SP + 4;

【功能】
- 释放栈帧，并且从子程序返回。
- 此指令必须和ENTER指令配对使用。
- 在子程序的最后执行EXITD指令前后的堆栈区状态如下所示：



【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
EXITD

EXITI

中断栈帧的释放
Exit Interrupt and Deallocate Stack Frame

EXITI

【语法】
EXITI

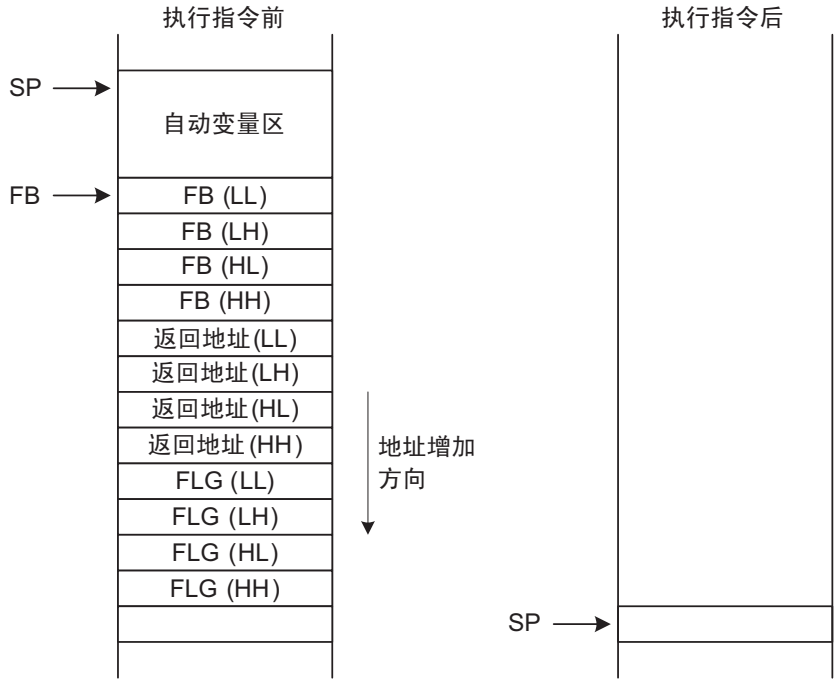
【指令码 / 周期数】
Page=197

【操作】

```
SP  =  FB;
FB  =  *SP;
SP  =  SP + 4;
PC  =  *SP;
SP  =  SP + 4;
FLG =  *SP;
SP  =  SP + 4;
```

【功能】

- 释放栈帧，并且从中断程序返回。
- 此指令必须和ENTER指令配对使用。
- 在中断程序的最后执行EXITI指令前后的堆栈区状态如下所示：



【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	○	○	○	○	○	○	○	○

*1 全部为发生中断前的值（被压栈的值）。

【记述例子】

```
EXITI
```

EXTS

符号扩展
Sign Extend

EXTS

【语法】

EXTS.size src,dest

BW , BL , WL

【指令码 / 周期数】

Page=197

【操作】

dest = (short | long) src;

【功能】

- 将src符号扩展，结果保存到dest。
- 在长度说明符 （.size）为 “.BW” 时，将8位的src符号扩展为16位，结果保存到dest。
- 在长度说明符 （.size）为 “.BL” 时，将8位的src符号扩展为32位，结果保存到dest。
- 在长度说明符 （.size）为 “.WL” 时，将16位的src符号扩展为32位，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/~~R2R0~~	R0H/R2/~~R3R1~~	R2L/R1/~~R6R4~~	R2H/R3/~~R7R5~~	~~R0L~~/R0/R2R0	~~R0H~~/R2/R3R1	~~R2L~~/R1/R6R4	~~R2H~~/R3/R7R5
R1L/R4/~~A0~~	R1H/R6/~~A1~~	R3L/R5/~~A2~~	R3H/R7/~~A3~~	~~R1L~~/R4/A0	~~R1H~~/R6/A1	~~R3L~~/R5/A2	~~R3H~~/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM*2	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 能取的范围为 –128 ≤ IMM ≤ +127(.BW, .BL)、 –32768 ≤ IMM ≤ +32767(.WL)。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
EXTS.BW     R0L,R1
EXTS.BL     R2L,R6R4
EXTS.WL     R3,[A0]
```

EXTZ

零扩展
Zero Extend

EXTZ

【语法】

EXTZ.size(:format) src,dest

- size: G , S
- format: BW , BL , WL

【指令码 / 周期数】

Page=199

【操作】

dest = (unsigned short | unsigned long) src;

【功能】

- 将src零扩展，结果保存到dest。
- 在长度说明符 （.size）为 “.BW” 时，将8位的src零扩展为16位，结果保存到dest。
- 在长度说明符 （.size）为 “.BL” 时，将8位的src零扩展为32位，结果保存到dest。
- 在长度说明符 （.size）为 “.WL” 时，将16位的src零扩展为32位，结果保存到dest。

【可选择的 src/dest】 （不同格式的 src/dest 请参照下一页。）

src*1				dest*1			
R0L/R0/~~R2R0~~	R0H/R2/~~R3R1~~	R2L/R1/~~R6R4~~	R2H/R3/~~R7R5~~	~~R0L~~/R0/R2R0	~~R0H~~/R2/R3R1	~~R2L~~/R1/R6R4	~~R2H~~/R3/R7R5
R1L/R4/~~A0~~	R1H/R6/~~A1~~	R3L/R5/~~A2~~	R3H/R7/~~A3~~	~~R1L~~/R4/A0	~~R1H~~/R6/A1	~~R3L~~/R5/A2	~~R3H~~/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM*2	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 能取的范围为 0 ≤ IMM ≤ +255(.BW, .BL)、 0 ≤ IMM ≤ +65535(.WL)。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：总是为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
EXTZ.BW     R0L,R1
EXTZ.BL     R2L,R6R4
EXTZ.WL     R3,[A0]
EXTZ.WL     R0,A0
```

【不同格式的 src/dest】

G 格式

src*1				dest*1			
R0L/R0/~~R2R0~~	R0H/R2/~~R3R1~~	R2L/R1/~~R6R4~~	R2H/R3/~~R7R5~~	~~R0L~~/R0/R2R0	~~R0H~~/R2/R3R1	~~R2L~~/R1/R6R4	~~R2H~~/R3/R7R5
R1L/R4/~~A0~~	R1H/R6/~~A1~~	R3L/R5/~~A2~~	R3H/R7/~~A3~~	~~R1L~~/R4/A0	~~R1H~~/R6/A1	~~R3L~~/R5/A2	~~R3H~~/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

S 格式

src*1*2				dest*1			
~~R0L~~/R0/~~R2R0~~	~~R0H~~/R2/~~R3R1~~	~~R2L~~/R1/~~R6R4~~	~~R2H~~/R3/~~R7R5~~	A0	A1	A2	A3
~~R1L~~/R4/~~A0~~	R1H/R6/~~A1~~	~~R3L~~/R5/~~A2~~	~~R3H~~/R7/~~A3~~				
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24				
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]				
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]				
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]				
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]				
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]				

*1 只能给长度说明符 （.size）指定 “.WL”，因此 src 固定为 16 位， dest 固定为 32 位。

*2 能使用间接指令寻址或者组 1 寄存器直接。

FCLR

标志清除
Clear a Flag

FCLR

【语法】
FCLR dest

【指令码 / 周期数】
Page=201

【操作】
dest = 0;

【功能】
- 将 “0”保存到dest。

【可选择的 dest】

dest							
U	I	O	B	S	Z	D	C

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	*1	*1	*1	*1	*1	*1	*1	*1

*1 被指定的标志为 “0”。

【记述例子】
```
FCLR    I
FCLR    S
FCLR    O
```

FREIT

从高速中断的返回
Return from Fast Interrupt

FREIT

【语法】
FREIT

【指令码 / 周期数】
Page=201

【操作】
FLG = SVF;
PC = SVP;

【功能】
- 从高速中断寄存器恢复接受高速中断请求时保存的PC和FLG，并且从中断程序返回。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	○	○	○	○	○	○	○	○

*1 全部为发生中断前的值 （被保存到 SVF 的值）。

【记述例子】
FREIT

FSET

标志置位
Set a Flag

FSET

【语法】
FSET dest

【指令码 / 周期数】
Page=201

【操作】
dest = 1;

【功能】
- 将“1”保存到dest。

【可选择的 dest】

dest							
U	I	O	B	S	Z	D	C

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	*1	*1	*1	*1	*1	*1	*1	*1

*1 被指定的标志为“1”。

【记述例子】
```
FSET    I
FSET    S
FSET    O
```

INC

递增
Increment

INC

【语法】

INC.size dest

B , W , L

【指令码 / 周期数】

Page=202

【操作】

dest = dest + 1;

【功能】

- 给dest加1，结果保存到dest。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
INC.B   R0L
INC.W   R2
INC.L   R7R5
INC.L   A0
INC.W   mem[SB]
```

INDEX*Type*

变址
Index

INDEX*Type*

【语法】

INDEX*Type*.size src

B , W , L

【指令码 / 周期数】

Page=203

【操作】

【功能】

- 修改下一条指令的寻址方式。
- 在此指令之后不能马上接受中断请求。
- 用于存取数组。
- *Type*的种类如下所示。
- 详细内容请参照 “3.3 变址指令”。

Type	功能
B	在下一条指令有 2 个一般格式的操作数时，修改 src 和 dest。
1	在下一条指令只有 1 个一般格式或者短格式的操作数时，修改该操作数；在有 2 个一般格式的操作数时，修改 src。
2	在下一条指令有 2 个一般格式的操作数时，修改 dest。

【可选择的 src】

src*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
INDEX1.B    R0L
INDEXB.W    R2
```

INT

INT 指令中断
Interrupt

INT

【语法】
INT src

【指令码 / 周期数】
Page=204

【操作】
```
SP    =  SP - 4;
*SP   =  FLG;
SP    =  SP - 4;
*SP   =  PC + 2;
PC    =  *(IntBase + src * 4);
```

【功能】
- 发生由src指定序号的软件中断。
- 中断序号 （src）的范围为0≤src≤255。
- 在src小于等于127时， U标志为 “0”，使用ISP。
- 在src大于等于128时，使用U标志所示的堆栈指针。
- 通过INT指令发生的中断是非屏蔽中断。

【可选择的 src】

src
#IMM:8*1

*1 #IMM:8 是软件中断序号，能取的范围为 0 ≤ IMM:8 ≤ 255。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	○	○	—	—	—	—	○	—

*1 执行指令前的标志被压栈。

条件

U：当软件中断序号小于等于 127 时为 “0”，否则不变。
I：为 “0”。
D：为 “0”。

【记述例子】
```
INT     #0
```

INTO

上溢中断
Interrupt on Overflow

INTO

【语法】
INTO

【指令码 / 周期数】
Page=204

【操作】

```
SP    =  SP - 4;
*SP   =  FLG;
SP    =  SP - 4;
*SP   =  PC + 1;
PC    =  *(0xFFFFFFE0);
```

【功能】

- 在O标志为 “1” 时，发生上溢中断。
- 在O标志为 “0” 时，不发生中断。
- 上溢中断是非屏蔽中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	○	○	—	—	—	—	○	—

*1 执行指令前的标志被压栈。

条件

U：为 “0”。
I：为 “0”。
D：为 “0”。

【记述例子】

```
INTO
```

JCnd

条件转移
Jump Conditionally

JCnd

【语法】
JCnd label

【指令码 / 周期数】
Page=204

【操作】
if (true)
PC = label; (= PC + dsp)

【功能】
- 判断由*Cnd*所示条件的真假值并且进行转移。真时转移，假时不转移。
- *Cnd*有以下的种类：

Cnd	条件		式	*Cnd*	条件		式
GEU/C	C == 1	大于等于 / C 标志为 “1”	≤	LTU/NC	C == 0	小于 / C 标志为 “0”	＞
EQ/Z	Z == 1	等于 / Z 标志为 “1”	＝	NE/NZ	Z == 0	不等于 / Z 标志为 “0”	≠
GTU	C & ~Z == 1	大于	＜	LEU	C & ~Z == 0	小于等于	≥
PZ	S == 0	大于等于 “0”	0 ≤	N	S == 1	负	0 ＞
GE	S ^ O == 0	等于或者带符号的大于	≤	LE	(S ^ O) | Z == 1	等于或者带符号的小于	≥
GT	(S ^ O) | Z == 0	带符号的大于	＜	LT	S ^ O == 1	带符号的小于	＞
O	O == 1	O 标志为 “1”		NO	O == 0	O 标志为 “0”	

【可选择的 label】

label
PC*1+ dsp:8*2

*1 PC 为有 JCnd 指令的地址 +1。

*2 dsp:8 的范围为 –128 ≤ dsp ≤ 127。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
JC label1
JLT label2

JMP

无条件转移
Jump Always

JMP

【语法】

JMP(.length) label

S , B , W , A

【指令码 / 周期数】

Page=205

【操作】

PC = label; (= PC + dsp)

【功能】

- 转移到label。

【可选择的 label】

.length	label	
.S	PC*1+dsp:3	1 ≤ dsp:3 ≤ 8
.B	PC*1+dsp:8	–128 ≤ dsp:8 ≤ 127
.W	PC*1+dsp:16	–32768 ≤ dsp:16 ≤ 32767
.A	PC*1+dsp:24	–8388608 ≤ dsp:24 ≤ 8388607

*1 PC 值为有 JMP 指令的地址 +1。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
JMP.B    label1
JMP.A    label2
```

JMPI

间接转移
Jump Indirectly

JMPI

【语法】

JMPI.length src

W , L

【指令码 / 周期数】

Page=206

【操作】

转移距离说明符 （.length）为 “.W”的情况

PC = PC*1 + src;

转移距离说明符 （.length）为 “.L”的情况

PC = src;

*1 PC 值为 JMP 指令地址。

【功能】

- 对src的内容进行相对转移或者绝对转移。
- 如果给转移距离说明符 （.length）指定 “.W”，就转移到将有JMPI指令的地址和src的内容相加 （带符号的加法）后的地址。在src为存储器时，需要2字节的存储器容量。
- 如果给转移距离说明符 （.length）指定 “.L”，就转移到src的内容所示的地址。在src为存储器时，需要4字节的存储器容量。

【可选择的 src】

src*1			
~~R0L~~/R0/R2R0	~~R0H~~/R2/R3R1	~~R2L~~/R1/R6R4	~~R2H~~/R3/R7R5
~~R1L~~/R4/A0	~~R1H~~/R6/A1	~~R3L~~/R5/A2	~~R3H~~/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
JMPI.W    R0
JMPI.L    A2
JMPI.W    mem[A0]
```

JSR

子程序转移
Jump to Subroutine

JSR

【语法】
JSR(.length) label
W , A

【指令码 / 周期数】
Page=206

【操作】
SP = SP - 4;
*SP = (PC + n)*1;
PC = label; (= PC + dsp)

*1 (PC+n) 为 JSR 指令的下一条指令的地址。

【功能】
- 转移到label所示的子程序。

【可选择的 label】

.length	label	
.W	PC*1+ dsp:16	–32768 ≤ dsp:16 ≤ 32767
.A	PC*1+ dsp:24	–8388608 ≤ dsp:24 ≤ 8388607

*1 PC 值为有 JSR 指令的地址 +1。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
```
JSR.W subroutine1
JSR.A subroutine2
```

JSRI

间接子程序转移
Jump to Subroutine Indirectly

JSRI

【语法】

JSRI.length src

W , L

【指令码 / 周期数】

Page=207

【操作】

转移距离说明符 （.length）为 “.W” 的情况

SP = SP - 4;
*SP = (PC + n)*1;
PC = PC*2 + src;

转移距离说明符 （.length）为 “.L” 的情况

SP = SP - 4;
*SP = (PC + n)*1;
PC = src;

*1 (PC+n) 为 JSRI 指令的下一条指令的地址。

*2 PC 值为有 JSRI 指令的地址。

【功能】

- 对src的内容进行相对子程序转移或者绝对子程序转移。
- 如果给转移距离说明符 （.length）指定 “.W”，就转移 （子程序转移）到将有JMPI指令的地址和src的内容相加 （带符号的加法）后的地址。在src为存储器时，需要2字节的存储器容量。
- 如果给转移距离说明符 （.length）指定 “.L”，就转移 （子程序转移）到src内容所示的地址。在src为存储器时，需要4字节的存储器容量。

【可选择的 src】

src*1			
~~R0L~~/R0/R2R0	~~R0H~~/R2/R3R1	~~R2L~~/R1/R6R4	~~R2H~~/R3/R7R5
~~R1L~~/R4/A0	~~R1H~~/R6/A1	~~R3L~~/R5/A2	~~R3H~~/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
JSRI.W    R0
JSRI.L    A2
JSRI.W    mem[A0]
```

LDC

向专用寄存器的传送
Load into Control Register

LDC

【语法】
LDC src,dest

【指令码 / 周期数】
Page=208

【操作】
dest = src;

【功能】
- 将src传送到dest。

【可选择的 src/dest】

src*1				dest			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	SB	FB	FLG	SP
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	ISP	SVF	SVP	INTB
#IMMEX*2	dsp:8[FB]	dsp:16[FB]	dsp:24	DSA0	DSA1	DSA2	DSA3
#IMM*3	dsp:16	dsp:16[SB]	dsp:24[SB]	DDA0	DDA1	DDA2	DDA3
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	DCT0	DCT1	DCT2	DCT3
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	DSR0	DSR1	DSR2	DSR3
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	DDR0	DDR1	DDR2	DDR3
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	DCR0	DCR1	DCR2	DCR3
				DMD0	DMD1	DMD2	DMD3
				VCT			

*1 能使用间接指令寻址或者组 1 寄存器直接。
*2 #IMMEX不能用作DMAC相关寄存器和VCT的src。然而，如果指定#IMM:8或者#IMM:16，就在零扩展后进行传送。
*3 #IMM:8 和 #IMM:16 只能用作 DMAC 相关寄存器和 VCT 的 src。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1

*1 dest 只在 FLG 时发生变化。

【记述例子】

```
LDC     #00800000h,SB
LDC     R6R4,DSA1
LDC     A1,DMD2
LDC     mem[A0],DCT1
```

LDCTX

上下文恢复
Load Context

LDCTX

【语法】
LDCTX src,dest

【指令码 / 周期数】
Page=209

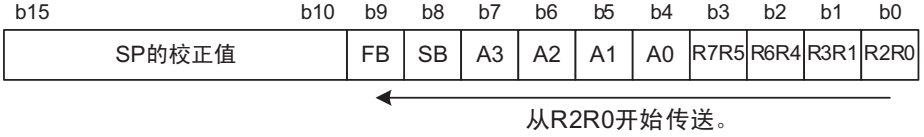
【操作】

```
for (i=0 ; i<n*1 ; i++) {
        register (dest[src])   =   *SP;
        SP    =    SP + 4;
}
```

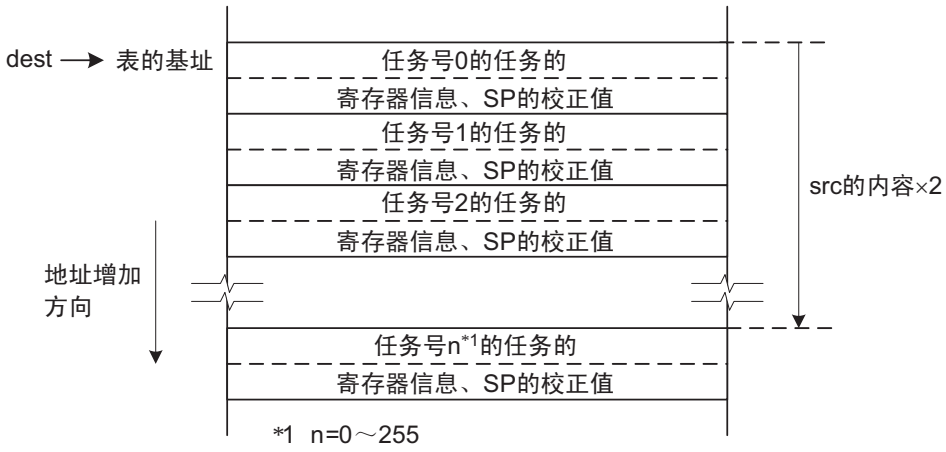
*1 n 是要恢复的寄存器个数。

【功能】

- 从堆栈区恢复任务的上下文。
- 必须给 src 设定保存任务号的 RAM 地址，给 dest 设定表数据的起始地址。
- 根据任务号，从表数据中指定需要的寄存器信息，并且根据此寄存器的信息将堆栈区的数据传送到各寄存器，然后给堆栈指针 （SP）加上 SP 的校正值。必须给 SP 的校正值设定要传送的寄存器的合计字节数，并且不能给表数据设定 “0000h”。
- 要传送的寄存器的信息由以下内容构成。当位为 “1” 时，表示要传送的寄存器；当位为 ” 0” 时，表示不要传送的寄存器。



- 表数据由以下的内容构成。 dest 所示的地址为表的基址，在以 2 倍 src 内容为偏移量的地址中保存的字数据的 bit0～9 表示寄存器的信息， bit10～15 表示堆栈指针的校正值。



【可选择的 src/dest】

src	dest
abs:16	dsp:24

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
LDCTX       ram,rom_tbl
```

LDIPL

中断优先级的设定
Load Interrupt Priority Level

LDIPL

【语法】
LDIPL src

【指令码 / 周期数】
Page=209

【操作】
IPL = src;

【功能】
- 将src传送到IPL。

【可选择的 src】

src
#IMM:3

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
```
LDIPL #2
```

MAX

最大值的选择
Select Maximum Value

MAX

【语法】

MAX.size src,dest

B , W , L

【指令码 / 周期数】

Page=210

【操作】

```
if (src > dest)
        dest  =  src;
```

【功能】

- 将src和dest进行带符号的比较，大的值保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
MAX.B     #-50,R1L
MAX.W     mem[A1],R2
MAX.L     R7R5,[A0]
```

MIN

最小值的选择
Select Minimum Value

MIN

【语法】

MIN.size src,dest

B , W , L

【指令码 / 周期数】

Page=211

【操作】

```
if (src < dest)
        dest  =  src;
```

【功能】

- 将src和dest进行带符号的比较，小的值保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
MIN.B     #-50,R1L
MIN.W     mem[A1],R2
MIN.L     R7R5,[A0]
```

MOV

传送
Move

MOV

【语法】

MOV.size(:format) src,dest

G , Q , Z , S
B , W , L

【指令码 / 周期数】
Page=212

【操作】

dest = src;

【功能】

- 将src传送到dest。

【可选择的 src/dest】 （不同格式的 src/dest 请参照下一页。）

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:4	dsp:8[SP]			dsp:8[SP]	dsp:8[SB]		

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当 src 的 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当 src 是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
MOV.B:Z     #0,R0L
MOV.W       R0,R2
MOV.L       A0,R7R5
MOV.B       R3L,[mem[A0]]
MOV.W       [[A1]],R4
MOV.L       R2R0,[[A3]]
```

【不同格式的 src/dest】

G 格式

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
dsp:8[SP] *2*3				dsp:8[SP]*2*4			

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 运算对象是 U 标志所示的堆栈指针。不能同时给 src 和 dest 选择 dsp:8[SP]。

*3 如果给 src 选择 dsp:8[SP]，就不能给 dest 选择间接指令寻址。

*4 如果给 dest 选择 dsp:8[SP]，就不能给 src 选择 #IMMEX 或者 #IMM。

Q 格式

src	dest*1			
#IMM:4*2	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
		dsp:8[FB]	dsp:16[FB]	dsp:24
		dsp:16	dsp:16[SB]	dsp:24[SB]
	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 能取的范围为 –8 ≤ #IMM:4 ≤ 7。

S 格式

src*1				dest*1			
#IMM				R0L/R0/R2R0	dsp:16	dsp:8[SB]	dsp:8[FB]
R0L/R0/R2R0*2				dsp:16	dsp:8[SB]	dsp:8[FB]	
R0H/R2/R3R1	dsp:16	dsp:8[SB]	dsp:8[FB]	R0L/R0/R2R0*2			
dsp:16	dsp:8[SB]	dsp:8[FB]		A0*2*3			

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 不能指定组 1 寄存器直接。

*3 只能给长度说明符 （.size）指定 “.L”。

Z 格式

src	dest*1			
#0	R0L/R0/R2R0	dsp:16	dsp:8[SB]	dsp:8[FB]

*1 能使用间接指令寻址或者组 1 寄存器直接。

MOVA

有效地址传送
Move Effective Address

MOVA

【语法】
MOVA src,dest

【指令码 / 周期数】
Page=216

【操作】
dest = &src;

【功能】
- 将src的有效地址传送到dest。

【可选择的 src/dest】

src				dest*1			
~~R0L/R0/R2R0~~	~~R0H/R2/R3R1~~	~~R2L/R1/R6R4~~	~~R2H/R3/R7R5~~	~~R0L/R0/~~R2R0	~~R0H/R2/~~R3R1	~~R2L/R1/~~R6R4	~~R2H/R3/~~R7R5
~~R1L/R4/A0~~	~~R1H/R6/A1~~	~~R3L/R5/A2~~	~~R3H/R7/A3~~	~~R1L/R4/~~A0	~~R1H/R6/~~A1	~~R3L/R5/~~A2	~~R3H/R7/~~A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24		~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]		~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~
~~[A0]~~	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~
~~[A1]~~	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~
~~[A2]~~	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~
~~[A3]~~	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~

*1 能指定组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
```
MOVA    mem[FB],A0
MOVA    mem[A1],R7R5
```

MOV*Dir*

4 位数据传送
Move Nibble Data

MOV*Dir*

【语法】
MOV*Dir* src,dest

【指令码 / 周期数】
Page=216

【操作】

Dir	操作
HH	H4:des = H4:src
HL	L4:dest = H4:src
LH	H4:dest = L4:src
LL	L4:dest = L4:src

【功能】

- 在*Dir*为HH时，将src （8位）的高4位传送到dest （8位）的高4位。
- 在*Dir*为HL时，将src （8位）的高4位传送到dest （8位）的低4位。
- 在*Dir*为LH时，将src （8位）的低4位传送到dest （8位）的高4位。
- 在*Dir*为LL时，将src （8位）的低4位传送到dest （8位）的低4位。
- 必须给src或者dest指定R0L。

【可选择的 src/dest】

src*1			
R0L/~~R0/R2R0~~	R0H/~~R2/R3R1~~	R2L/~~R1/R6R4~~	R2H/~~R3/R7R5~~
R1L/~~R4/A0~~	R1H/R6/A1	R3L/~~R5/A2~~	R3H/~~R7/A3~~
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
R0L/~~R0/R2R0~~*2			

dest*1			
R0L/~~R0/R2R0~~*2			
R0L/~~R0/R2R0~~	R0H/~~R2/R3R1~~	R2L/~~R1/R6R4~~	R2H/~~R3/R7R5~~
R1L/~~R4/A0~~	R1H/~~R6/A1~~	R3L/~~R5/A2~~	R3H/~~R7/A3~~
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。
*2 不能指定组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
MOVHH   R0L,[A0]
MOVHL   mem[A2],R0L
```

MUL

带符号的乘法运算
Signed Multiply

MUL

【语法】

MUL.size src,dest

B , W , L

【指令码 / 周期数】

Page=217

【操作】

dest = dest * src;

【功能】

- 将src和dest进行带符号的乘法运算，结果保存到dest。
- 在长度说明符 （.size）为 “.B”时， src和dest都以8位进行运算，并且用8位保存结果。
- 在长度说明符 （.size）为 “.W”时， src和dest都以16位进行运算，并且用16位保存结果。
- 在长度说明符 （.size）为 “.L”时， src和dest都以32位进行运算，并且用32位保存结果。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
MUL.B    #3,R0L
MUL.W    R2,R3
MUL.L    [A3],R6R4
```

MULF

浮点乘法运算
Multiply Floating Point

MULF

【语法】
MULF src,dest

【指令码 / 周期数】
Page=218

【操作】
dest = dest * src;

【功能】
- 将src和dest进行带符号的浮点乘法运算，结果保存到dest。
- 根据标志寄存器 （FLG）指定的舍入模式，将运算结果进行舍入。
- 在运算结果超过最大规格化数时，根据符号，结果为正的最大值 （7F7FFFFFh）或者负的最大值 （FF7FFFFFh）。
- 在运算结果小于最小规格化数时，根据舍入模式，结果为零 （00000000h）或者正的最小值 （00800000h）或者负的最小值 （80800000h）。
- 非法输入的运算结果为不定值。

【可选择的 src/dest】

src*1				dest*1			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	○	○	—	—	○	—	○	○	—	○

条件

FO：当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
FU：当非法输入或者运算结果小于最小规格化数时为 “1”，否则为 “0”。
O：当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果的全部位是 “0” 时为 “1”，否则为 “0”。
C：为不定值。

【记述例子】

```
MULF    R2R0,R3R1
MULF    [A0],R2R0
MULF    mem[FB],R3R1
```

MULU

不带符号的乘法运算
Unsigned Multiply

MULU

【语法】

MULU.size src,dest

B , W , L

【指令码 / 周期数】
Page=219

【操作】

dest = dest * src;

【功能】

- 将src和dest进行不带符号的乘法运算，结果保存到dest。
- 在长度说明符 （.size）为 “.B”时， src和dest都以8位进行运算，并且用8位保存结果。
- 在长度说明符 （.size）为 “.W”时， src和dest都以16位进行运算，并且用16位保存结果。
- 在长度说明符 （.size）为 “.L”时， src和dest都以32位进行运算，并且用32位保存结果。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当运算结果超过 8 位 (.B)、 16 位 (.W)、 32 位 (.L) 时为 “1”，否则为 “0”。

【记述例子】

```
MULU.B    #3,R0L
MULU.W    R2,R3
MULU.L    [A3],R6R4
```

MULX

带舍入的乘法运算
Signed Multiply and Round

MULX

【语法】

MULX.size src,dest

B , W , L

【指令码 / 周期数】

Page=220

【操作】

```
short | long | long long      tmp;

tmp*1   =   (short | long | long long)dest * src;
if (DP==0)
        dest    =   (char | short | long)(tmp >> n*2);
else
        dest    =   (short | long)(tmp >> m*3);
```

*1 tmp：临时寄存器

*2 长度说明符 （.size）是 “.B” 时为 6，是 “.W” 时为 14，是 “.L” 时为 30。

*3 长度说明符 （.size）是 “.W” 时为 8，是 “.L” 时为 16。

【功能】

- 将src和dest进行带符号的乘法运算，根据标志寄存器 （FLG）的bit16 （DP位），进行移位和四舍五入，结果保存到dest。
- 如果给长度说明符 （.size）指定 “.B”， src和dest就都以8位进行运算，并且用8位保存结果。
- 如果给长度说明符 （.size）指定 “.W”， src和dest就都以16位进行运算，并且用16位保存结果。
- 如果给长度说明符 （.size）指定 “.L”， src和dest就都以32位进行运算，并且用32位保存结果。
- 能用于src和dest为定点数的乘法运算 （参照下一页的图）。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24				
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]				
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]				
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]				
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]				
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]				

*1 能使用间接指令寻址或者组 1 寄存器直接。

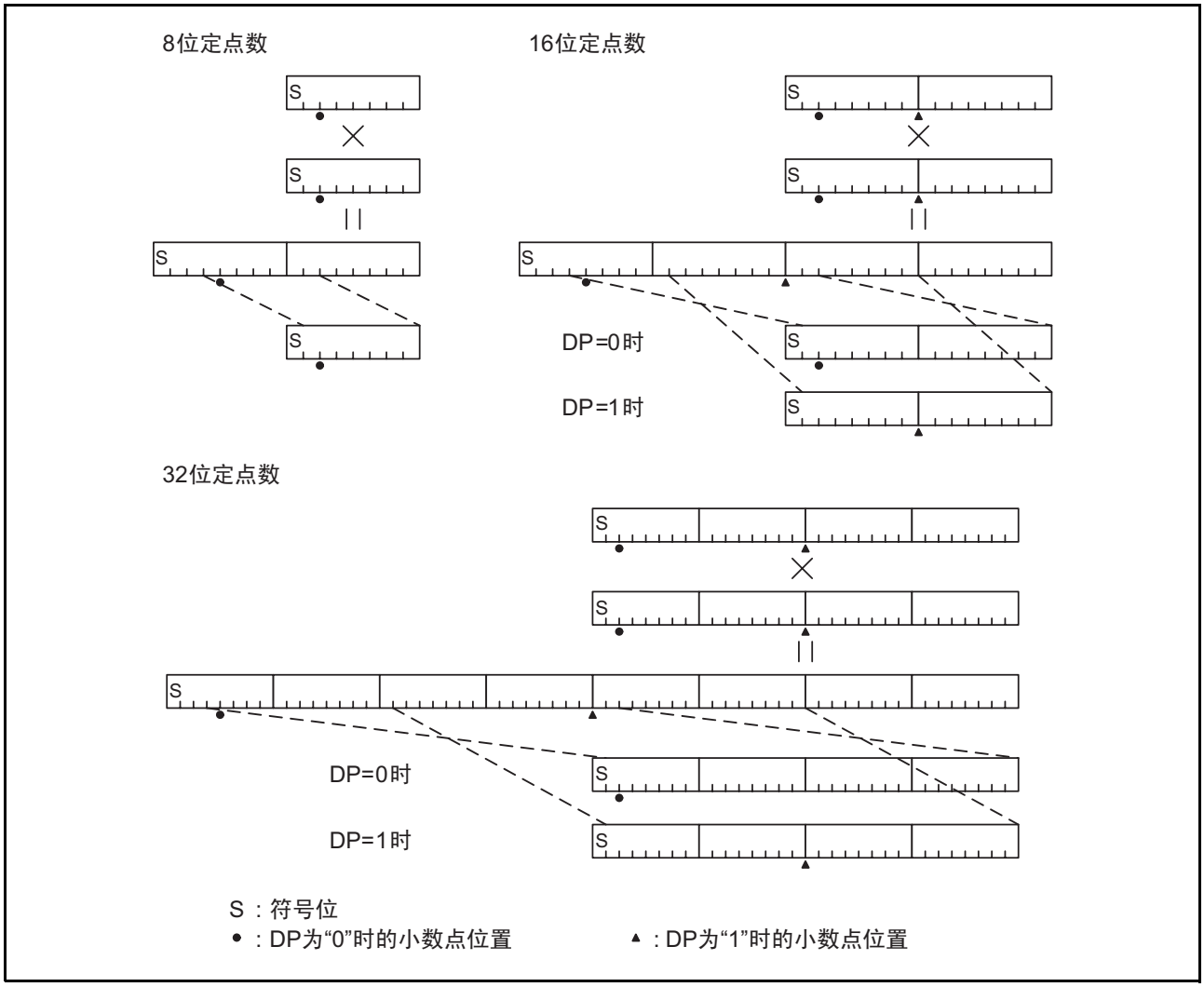


图 3.1 定点乘法运算的应用例子

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

【记述例子】

```
MULX.B    #12h,R0L      ; (R0L × 12h)>>6→R0L
MULX.W    R0,R1         ; (R1× R0)>>14 or 8→R1
MULX.L    A0,R6R4       ; (R6R4 × A0)>>30 or 16→R6R
```

NEG

符号取反
Negate

NEG

【语法】
NEG.size dest
B , W , L

【指令码 / 周期数】
Page=221

【操作】
dest = -dest;

【功能】
- 将dest的符号取反（取2的补数），结果保存到dest。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件
O：当运算前的 dest 是 –128(.B)、 –32768(.W) 或者 –2147483648(.L) 时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果是 “0” 时为 “1”，否则为 “0”。
C：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
NEG.B   R0L
NEG.W   R3
NEG.L   [A0]
```

NOP

空操作
No Operation

NOP

【语法】
NOP

【指令码 / 周期数】
Page=221

【操作】
PC = PC + 1;

【功能】
- 给PC加1。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
NOP

NOT

逻辑取反
Logical Complement

NOT

【语法】

NOT.size dest

B , W , L

【指令码 / 周期数】

Page=221

【操作】

dest = ~dest;

【功能】

- 将dest逻辑取反，结果保存到dest。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
NOT.B   R0L
NOT.W   R3
NOT.L   [A0]
```

OR

逻辑或
Or Logical

OR

【语法】

OR.size src, dest

B , W , L

【指令码 / 周期数】

Page=222

【操作】

dest = dest | src;

【功能】

- 取dest和src的逻辑或，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
OR.B    #2,R0L
OR.W    R0,R2
OR.L    A0,R7R5
OR.B    R3L,[mem[A0]]
OR.W    [[A1]],R4
OR.L    R2R0,[[A3]]
```

POP

寄存器 / 存储器恢复
Pop Data off the Stack

POP

【语法】

POP.size dest

B , W , L

【指令码 / 周期数】

Page=223

【操作】

dest = *SP;
SP = SP + 4*1;

*1 即使长度说明符 （.size）为 “.B”或者” .W”，也给 SP 加 4。

【功能】

- 将退栈的数据传送到dest。
- 使用的堆栈指针为U标志所示的堆栈指针。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
POP.B   R0L
POP.W   R3
POP.L   R6R4
```

POPC

专用寄存器恢复
Pop Contorl Register off the Stack

POPC

【语法】
POPC dest

【指令码 / 周期数】
Page=223

【操作】
dest = *SP;
SP = SP + 4;

【功能】
- 将退栈的数据传送到dest所示的专用寄存器。
- 使用的堆栈指针为U标志所示的堆栈指针。

【可选择的 dest】

dest			
SB	FB	FLG	SP*1
ISP	SVF	SVP	INTB

*1 对象为 U 标志所示的堆栈指针。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	*1	*1	*1	*1	*1	*1	*1	*1

*1 dest 只在 FLG 时发生变化。

【记述例子】
```
POPC    SB
POPC    SVF
```

POPM

多个寄存器恢复
Pop Registers off the Stack

POPM

【语法】
POPM dest

【指令码 / 周期数】
Page=223

【操作】

```
for (i=0 ; i< n*1 ; i++) {
        registers (dest)  =  *SP;
        SP  =  SP + 4;
}
```

*1 n 是要恢复的寄存器个数。

【功能】

- 将由dest选择的寄存器进行一次性的退栈。
- 使用的堆栈指针为U标志所示的堆栈指针。
- 退栈的顺序如下：

【可选择的 dest】

dest*1*2			
R2R0	R3R1	R6R4	R7R5
A0	A1	A2	A3
SB			

*1 能给 dest 指定多个寄存器。

*2 能使用组 1 寄存器直接，但是不能混合指定寄存器直接和组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
POPM        R6R4,A3,SB
POPM        R2R0,R3R1,A0,A1
```

PUSH

寄存器 / 存储器 / 立即数保存
Push Data on the Stack

PUSH

【语法】

PUSH.size(:format) src

G , S
B , W , L

【指令码 / 周期数】
Page=224

【操作】

SP = SP - 4*1;
*SP = src;

*1 即使长度说明符（.size）为“.B”或者“.W”，也从 SP 减去 4。在“.B”时高 24 位为不定值；在“.W”时高 16 位为不定值。

【功能】

- 将src压栈。
- 使用的堆栈指针为U标志所示的堆栈指针。

【可选择的 src】 （不同格式的 src 请参照下一页。）

src*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
PUSH.B    R0L
PUSH.W    #1000
PUSH.L    [mem[A0]]
PUSH.W    #FF80h
```

【不同格式的 src】

G 格式

src*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

S 格式

src	
#IMMEX	#IMM

PUSHA

有效地址保存
Push Effective Address on the Stack

PUSHA

【语法】
PUSHA src

【指令码 / 周期数】
Page=225

【操作】
SP = SP - 4;
*SP = &src;

【功能】
- 将src的有效地址压栈。
- 使用的堆栈指针为U标志所示的堆栈指针。

【可选择的 src】

src			
~~R0L/R0/R2R0~~	~~R0H/R2/R3R1~~	~~R2L/R1/R6R4~~	~~R2H/R3/R7R5~~
~~R1L/R4/A0~~	~~R1H/R6/A1~~	~~R3L/R5/A2~~	~~R3H/R7/A3~~
~~#IMMEX~~	dsp:8[FB]	dsp:16[FB]	dsp:24
~~#IMM~~	dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
```
PUSHA    mem[FB]
PUSHA    mem[A0]
```

PUSHC

专用寄存器保存
Push Contorl Register on the Stack

PUSHC

【语法】
PUSHC src

【指令码 / 周期数】
Page=226

【操作】
SP = SP - 4;
*SP = src;

【功能】
- 将dest所示的专用寄存器压栈。
- 使用的堆栈指针为U标志所示的堆栈指针。

【可选择的 src】

src			
SB	FB	FLG	SP*1
ISP	SVF	SVP	INTB

*1 对象为 U 标志所示的堆栈指针。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
```
PUSHC SB
PUSHC SVF
```

PUSHM

多个寄存器保存
Push Registers on the Stack

PUSHM

【语法】
PUSHM src

【指令码 / 周期数】
Page=226

【操作】

```
for (i=0 ; i< n*1 ; i++) {
        SP    =   SP - 4;
        *SP   =   src;
}
```

*1 n 是要保存的寄存器个数。

【功能】

- 将由src选择的寄存器进行一次性的压栈。
- 使用的堆栈指针为U标志所示的堆栈指针。
- 压栈的顺序如下：

【可选择的 src】

src*1*2			
R2R0	R3R1	R6R4	R7R5
A0	A1	A2	A3
SB			

*1 能给 src 指定多个寄存器。

*2 能使用组 1 寄存器直接，但是不能混合指定寄存器直接和组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
PUSHM     R6R4,A3,SB
PUSHM     R2R0,R3R1,A0,A1
```

REIT

从中断的返回
Return from Interrupt

REIT

【语法】
REIT

【指令码 / 周期数】
Page=226

【操作】
PC = *SP;
SP = SP + 4;
FLG = *SP;
SP = SP + 4;

【功能】
- 恢复在接受中断请求时压栈的PC和FLG，并且从中断程序返回。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	*1	*1	*1	*1	*1	*1	*1	*1

*1 为堆栈中的值。

【记述例子】
REIT

RMPA

乘加运算
Repeat Multiply and Accumulation

RMPA

【语法】

RMPA.size

B , W , L

【指令码 / 周期数】

Page=227

【操作】

```
While (R7R5 != 0)*1 {
        R4:R3R1:R2R0    =   R4:R3R1:R2R0 + *A0 * *A1;
        A0      =   A0 + n*2;
        A1      =   A1 + n*2;
        R7R5    =   R7R5 - 1;
}
```

*1 如果在给 R7R5 设定 “0”后执行，就忽视此指令。

*2 在给长度说明符 （.size）指定了 “.B”时为 “1”，指定了 “.W”时为 “2”，指定了 “.L”时为 “4”。

【功能】

- 进行A0为被乘数地址、 A1为乘数地值、 R7R5为次数的带符号的乘加运算，结果保存到R3R1:R2R0的64位。
- 能给R7R5设定的最大值为 “00020000h”（131072）。
- R4和R6用作工作寄存器，乘加运算的中途结果累计在R4:R3R1:R2R0中。
- 指令结束时的地址寄存器内容以及R4和R6的内容为不定值。
- 必须在执行指令前给R3R1:R2R0设定初始值，并且必须在R3R1:R2R0为负时给R4设定 “FFFFh”，为正时给R4设定 “0000h”。
- 如果在指令执行过程中发生中断请求，就停止运算而接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	—	—	—	—

条件

O：当运算结果超过 $2^{63}-1$ 或者 -2^{63} 时为 “1”，否则为 “0”。

【记述例子】

```
RMPA.W
```

ROLC

带进位的左循环
Rotate the bits to the Left with Carry

ROLC

【语法】

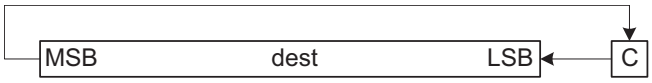
ROLC.size dest

B , W , L

【指令码 / 周期数】

Page=227

【操作】



【功能】

- 包含C标志将dest循环左移1位。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	○

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果是 “0” 时为 “1”，否则为 “0”。
C：当移出的位是 “1” 时为 “1”，否则为 “0”。

【记述例子】

```
ROLC.B    R0L
ROLC.W    [A0]
ROLC.L    R2R0
```

RORC

带进位的右循环
Rotate the bits to the Right with Carry

RORC

【语法】

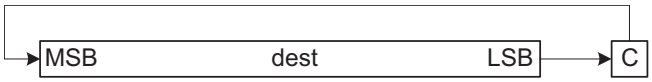
RORC.size dest

B , W , L

【指令码 / 周期数】

Page=227

【操作】



【功能】

- 包含C标志将dest循环右移1位。

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	○

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果是 “0” 时为 “1”，否则为 “0”。
C：当移出的位是 “1” 时为 “1”，否则为 “0”。

【记述例子】

```
RORC.B    R0L
RORC.W    [A0]
RORC.L    R2R0
```

ROT

循环
Rotate the bits

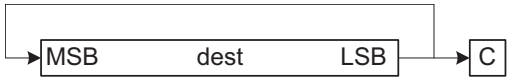
ROT

【语法】
ROT.size src,dest
B , W , L

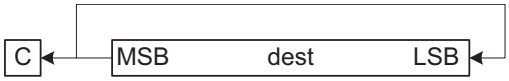
【指令码 / 周期数】
Page=228

【操作】

src ＜ 0 时



src ＞ 0 时



【功能】

- dest进行循环移位，移动的位数由src指定，从LSB/MSB溢出的位被传送到MSB/LSB和C标志。
- 由src的符号指定循环的方向。当src为正时左循环；当src为负时右循环；当src为 “0”时不循环移位。

【可选择的 src/dest】

src*1*2				dest*1			
R0L/~~R0/R2R0~~	R0H/~~R2/R3R1~~	R2L/~~R1/R6R4~~	R2H/~~R3/R7R5~~	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/~~R4/A0~~	R1H/~~R6/A1~~	R3L/~~R5/A2~~	R3H/~~R7/A3~~	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM:8	~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~		dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用组 1 寄存器直接。
*2 能取的范围为 –8 ～ +8(.B)、 –16 ～ +16(.W)、 –32 ～ +32(.L)。
*3 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	—	—	—	—	○	○	—	○

*1 当 src 为 “0” 时，标志不变化。

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果是 “0” 时为 “1”，否则为 “0”。
C：当移出的位是 “1” 时为 “1”，否则为 “0”。

【记述例子】

```
ROT.B    #1,R0L      ; 左循环
ROT.B    #-1,R0L     ; 右循环
ROT.W    R1L,[A0]
ROT.L    R0H,R6R4
```

ROUND

浮点数 → 整数的转换
Round Floating Point to Integer

ROUND

【语法】
ROUND src,dest

【指令码 / 周期数】
Page=229

【操作】
dest = (long) src;

【功能】
- 将src转换为整数，结果保存在dest。
- 根据标志寄存器（FLG）指定的舍入模式，将运算结果进行舍入，并且将结果钳制在–2147483648～+2147483647的范围。
- 当src为非法输入值时，运算结果为不定值。

【可选择的 src/dest】

src*1				dest*1			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	○	○	—	—	○	—	○	○	—	○

条件

FO：当非法输入或者运算结果超过 –2147483648 或者 +2147483647 时为 “1”，否则为 “0”。
FU：当非法输入时为 “1”，否则为 “0”。
O：当非法输入或者运算结果超过 –2147483648 或者 +2147483647 时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果的全部位是 “0” 时为 “1”，否则为 “0”。
C：为不定值。

【记述例子】

```
ROUND   R2R0,R3R1
ROUND   [A1],R2R0
ROUND   mem[FB],R3R1
```

RTS

从子程序的返回
Return from Subroutine

RTS

【语法】
RTS

【指令码 / 周期数】
Page=230

【操作】
PC = *SP;
SP = SP + 4;

【功能】
- 从子程序返回。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
RTS

SBB

带借位的减法运算
Subtract with Borrow

SBB

【语法】

SBB.size src,dest

B , W , L

【指令码 / 周期数】

Page=230

【操作】

dest = dest - src - ~C;

【功能】

- 从dest减去src和C标志被取反 （借位）的值，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：当不带符号的运算结果大于等于 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
SBB.B    #2,R0L
SBB.W    R0,R2
SBB.L    A0,R7R5
SBB.B    R3L,[A0]
SBB.W    [A1],R4
SBB.L    R2R0,[A3]
```

SCCnd

条件设定
Store Condition Conditionally

SCCnd

【语法】

SC*Cnd*.size dest

B , W , L

【指令码 / 周期数】

Page=231

【操作】

```
if (true)
        dest  =  1;
else
        dest  =  0;
```

【功能】

- 给dest设定由*Cnd*所示条件的真假值。真时设定 “1”，假时设定 “0”。
- *Cnd*有以下的种类：

Cnd	条件		式	*Cnd*	条件		式
GEU/C	C == 1	大于等于 / C 标志为 “1”	≤	LTU/NC	C == 0	小于 / C 标志为 “0”	＞
EQ/Z	Z == 1	等于 / Z 标志为 “1”	＝	NE/NZ	Z == 0	不等于 / Z 标志为 “0”	≠
GTU	C & ~Z == 1	大于	＜	LEU	C & ~Z == 0	小于等于	≥
PZ	S == 0	大于等于 “0”	0 ≤	N	S == 1	负	0 ＞
GE	S ^ O == 0	等于或者带符号的大于	≤	LE	(S ^ O) | Z == 1	等于或者带符号的小于	≥
GT	(S ^ O) | Z == 0	带符号的大于	＜	LT	S ^ O == 1	带符号的小于	＞
O	O == 1	O 标志为 “1”		NO	O == 0	O 标志为 “0”	

【可选择的 dest】

dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
	dsp:8[FB]	dsp:16[FB]	dsp:24
	dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
SCC.B       R0L
SCNE.W      [A0]
SCGT.L      A3
```

SCMPU

字符串比较
Compare Strings until not equal

SCMPU

【语法】

SCMPU.size

B , W

【指令码 / 周期数】

Page=232

【操作】

```
unsigned char *A1, *A0, tmp0, tmp1;

do {
        tmp0    =   *A0++;
        tmp1    =   *A1++;
} while (tmp0 == tmp1 && tmp0 != '\0');
```

tmp0、 tmp1：临时寄存器

【功能】

- 以地址增加方向，将A0所示的比较源地址的字符串和A1所示的比较目标地址的字符串进行比较，比较到结果不同或者检测到Null字符 ‘\0’（=00h）为止。
- 长度说明符 （.size）为 “.B”和为 “.W”时的操作相同。
- 指令结束时的地址寄存器 （A0和A1）为不定值。
- 如果在指令执行过程中发生中断请求，就停止运算而接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：为不定值。

S：为不定值。

Z：当双方的字符串相同时为 “1”，否则为 “0”。

C：当 (*A0 – *A1) 的不带符号的运算结果大于等于 “0”时为 “1”，否则为 “0”。

【记述例子】

SCMPU.W

SHA

算术移位
Arithmetic Shift

SHA

【语法】

SHA.size src,dest

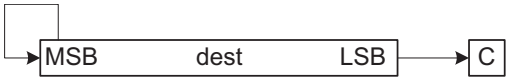
B , W, L

【指令码 / 周期数】

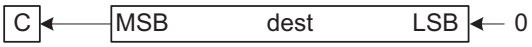
Page=232

【操作】

src ＜ 0 时



src ＞ 0 时



【功能】

- 将dest进行算术移位，移动的位数由src指定，从LSB/MSB溢出的位被传送到C标志。
- 由src的符号指定移位方向。当src为正时左移；当src为负时右移；当src为 “0” 时不移位。

【可选择的 src/dest】

src*1*2				dest*3			
R0L/~~R0/R2R0~~	R0H/~~R2/R3R1~~	R2L/~~R1/R6R4~~	R2H/~~R3/R7R5~~	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/~~R4/A0~~	R1H/~~R6/A1~~	R3L/~~R5/A2~~	R3H/~~R7/A3~~	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM:8	~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~		dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用组 1 寄存器直接。

*2 能取的范围为 –8 ～ +8(.B)、–16 ～ +16(.W)、–32 ～ +32(.L)。

*3 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	—	—	○	—	○	○	—	○

*1 当 src 为 “0” 时，标志不变化。

条件

O：左移时，如果运算结果 MSB 和移出的位是全部相同的值 （在移位中符号没有发生变化），为 “0”，否则为 “1”。右移时为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：当移出的位是 “1” 时为 “1”，否则为 “0”。

【记述例子】

```
SHA.B    #3,R0L      ; 左移
SHA.B    #-3,R0L     ; 右移
SHA.W    R1L,[A0]
SHA.L    R2H,R6R4
```

SHL

逻辑移位
Logical Shift

SHL

【语法】

SHL.size(:format) src,dest

G , Q
B , W, L

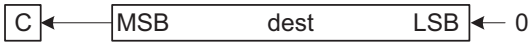
【指令码 / 周期数】

Page=233

【操作】

src ＜ 0 时

src ＞ 0 时



【功能】

- 将dest进行逻辑移位，移动的位数由src指定，从LSB/MSB溢出的位被传送到C标志。
- 由src的符号指定移位方向。当src为正时左移；当src为负时右移。

【可选择的 src/dest】 （不同格式的 src/dest 请参照下一页。）

src*1				dest*2			
R0L/~~R0/R2R0~~	R0H/~~R2/R3R1~~	R2L/~~R1/R6R4~~	R2H/~~R3/R7R5~~	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/~~R4/A0~~	R1H/~~R6/A1~~	R3L/~~R5/A2~~	R3H/~~R7/A3~~	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM:8*3	~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~		dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
#IMM:4*4							

*1 能使用组 1 寄存器直接。
*2 能使用间接指令寻址或者组 1 寄存器直接。
*3 能取的范围为 –8 ～ +8(.B)、 –16 ～ +16(.W)、 –32 ～ +32(.L)。
*4 能取的范围为 –8 ≤ #IMM:4 ≤ +7(≠ 0)。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	—	—	—	—	○	○	—	○

*1 当 src 为 “0” 时，标志不变化。

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果是 “0” 时为 “1”，否则为 “0”。
C：当移出的位是 “1” 时为 “1”，否则为 “0”。

【记述例子】

```
SHL.B    #1,R0L      ; 左移
SHL.B    #-1,R0L     ; 右移
SHL.W    R1L,[A0]
SHL.L    R2H,R6R4
```

【不同格式的 src/dest】

G 格式

src*1				dest*2			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM:8	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用组 1 寄存器直接。

*2 能使用间接指令寻址或者组 1 寄存器直接。

Q 格式

src	dest*1			
#IMM:4*2	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
		dsp:8[FB]	dsp:16[FB]	dsp:24
		dsp:16	dsp:16[SB]	dsp:24[SB]
	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

*2 能取的范围为 –8 ≤ #IMM:4 ≤ +7(≠ 0)。

SIN

字符串输入
Input Strings

SIN

【语法】

SIN.size

B , W , L

【指令码 / 周期数】

Page=235

【操作】

```
while (R7R5 != 0)*1 {
        *A1     =   *A0;
        A1      =   A1 + n*2;
        R7R5    =   R7R5 - 1;
}
```

*1 如果在给 R7R5 设定 “0”后执行，就忽视此指令。

*2 在给长度说明符 （.size）指定了 “.B”时为 “1”，指定了 “.W”时为 “2”，指定了 “.L”时为 “4”。

【功能】

- 以地址增加方向，将A0所示的固定传送源地址的字符串传送到A1所示的传送目标地址，传送次数由R7R5指定。
- 给A0设定传送源地址、 A1设定传送目标地址、 R7R5设定传送次数。
- 能给R7R5设定的最大值为 “00FFFFFFh”。
- 指令结束时的A1表示最后传送的数据的下一个地址。
- 如果在指令执行过程中发生中断请求，就在传送1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

SIN.W

SMOVB

反向字符串传送
Move Strings Backward

SMOVB

【语法】

SMOVB.size

B , W , L

【指令码 / 周期数】

Page=235

【操作】

```
while (R7R5 != 0)*1 {
        *A1     =   *A0;
        A0      =   A0 - n*2;
        A1      =   A1 - n*2;
        R7R5    =   R7R5 - 1;
}
```

*1 如果在给 R7R5 设定 “0”后执行，就忽视此指令。

*2 在给长度说明符 （.size）指定了 “.B”时为 “1”，指定了 “.W”时为 “2”，指定了 “.L”时为 “4”。

【功能】

- 以地址减少方向，将A0所示的传送源地址的字符串传送到A1所示的传送目标地址，传送次数由R7R5指定。
- 给A0设定传送源地址、 A1设定传送目标地址、 R7R5设定传送次数。
- 能给R7R5设定的最大值为 “00FFFFFFh”。
- 指令结束时的地址寄存器 （A0和A1）表示最后传送的数据的下一个地址。
- 如果在指令执行过程中发生中断请求，就在传送1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
SMOVB.W
```

SMOVF

正向字符串传送
Move Strings Forward

SMOVF

【语法】
SMOVF.size
B , W , L , Q

【指令码 / 周期数】
Page=235

【操作】

```
while (R7R5 != 0)*1 {
        *A1     =   *A0;
        A0      =   A0 + n*2;
        A1      =   A1 + n*2;
        R7R5    =   R7R5 - 1;
}
```

*1 如果在给 R7R5 设定 “0” 后执行，就忽视此指令。

*2 在给长度说明符 （.size）指定了 “.B” 时为 “1”，指定了 “.W” 时为 “2”，指定了 “.L” 时为 “4”，指定了 “.Q” 时为 “8”。

【功能】

- 以地址增加方向，将A0所示的传送源地址的字符串传送到A1所示的传送目标地址，传送次数由R7R5指定。
- 给A0设定传送源地址、 A1设定传送目标地址、 R7R5设定传送次数。
- 能给R7R5设定的最大值为 “00FFFFFFh”。
- 指令结束时的地址寄存器 （A0和A1）表示最后传送的数据的下一个地址。
- 如果在指令执行过程中发生中断请求，就在传送1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
SMOVF.W
```

SMOVU

字符串传送
Move Strings While Unequal to Zero

SMOVU

【语法】

SMOVU.size

B , W

【指令码 / 周期数】

Page=236

【操作】

```
unsigned char *A1, *A0, tmp0;

do {
        tmp0   =   *A0++;
        *A1++  =   tmp0
} while (tmp0 != '\0');

tmp0：临时寄存器
```

【功能】

- 以地址增加方向，将A0所示的传送源地址的字符串传送到A1所示的传送目标地址，传送到检测到Null字符 ‘\0’（=00h）为止。传送在传送Null字符后结束。
- 给A0设定传送源地址、 A1设定传送目标地址。
- 长度说明符 （.size）为 “.B”和为 “.W”时的操作相同。
- 指令结束时的地址寄存器 （A0和A1）为不定值。
- 如果在指令执行过程中发生中断请求，就在传送1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
SMOVU.W
```

SOUT

字符串输出
Output Strings

SOUT

【语法】

SOUT.size

B , W , L

【指令码 / 周期数】

Page=236

【操作】

```
while (R7R5 != 0)*1 {
        *A1     =   *A0;
        A0      =   A0 + n*2;
        R7R5    =   R7R5 - 1;
}
```

*1 如果在给 R7R5 设定 “0”后执行，就忽视此指令。

*2 在给长度说明符 （.size）指定了 “.B”时为 “1”，指定了 “.W”时为 “2”，指定了 “.L”时为 “4”。

【功能】

- 以地址增加方向，将A0所示的传送源地址的字符串传送到A1所示的固定传送目标地址，传送次数由R7R5指定。
- 给A0设定传送源地址、 A1设定传送目标地址、 R7R5设定传送次数。
- 能给R7R5设定的最大值为 “00FFFFFFh”。
- 指令结束时的A0表示最后传送的数据的下一个地址。
- 如果在指令执行过程中发生中断请求，就在传送1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

SOUT.W

SSTR

字符串保存
Store Strings

SSTR

【语法】

SSTR.size

B , W , L , Q

【指令码 / 周期数】

Page=236

【操作】

```
while (R7R5 != 0)*1 {
        *A1     =   R0L/R0/R2R0/R3R1R2R0*2;
        A1      =   A1 + n*3;
        R7R5    =   R7R5 - 1;
}
```

*1 如果在给 R7R5 设定 “0”后执行，就忽视此指令。

*2 在给长度说明符 （.size）指定了 “.B”时为 R0L，指定了 “.W”时为 R0，指定了 “.L”时为 R2R0，指定了 “.Q”时为 R3R1R2R0。

*3 在给长度说明符 （.size）指定了 “.B”时为 “1”，指定了 “.W”时为 “2”，指定了 “.L”时为 “4”，指定了 “.Q”时为 “8”。

【功能】

- 以地址增加方向，将R0L/R0/R2R0/R3R1R2R0的字符串保存到A1所示传送目标地址，传送次数由R7R5指定。
- 给A1设定传送目标地址、 R7R5设定传送次数。
- 能给R7R5设定的最大值为 “00FFFFFFh”。
- 指令结束时的A1表示最后写入的数据的下一个地址。
- 如果在指令执行过程中发生中断请求，就在传送1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
SSTR.W
```

STC

从专用寄存器的传送
Store from Control Register

STC

【语法】

STC src,dest

【指令码 / 周期数】

Page=237

【操作】

dest = src;

【功能】

- 将src传送到dest。

【可选择的 src/dest】

src				dest*1			
SB	FB	FLG	SP	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
ISP	SVF	SVP	INTB	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
DSA0	DSA1	DSA2	DSA3		dsp:8[FB]	dsp:16[FB]	dsp:24
DDA0	DDA1	DDA2	DDA3		dsp:16	dsp:16[SB]	dsp:24[SB]
DCT0	DCT1	DCT2	DCT3	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
DSR0	DSR1	DSR2	DSR3	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
DDR0	DDR1	DDR2	DDR3	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
DCR0	DCR1	DCR2	DCR3	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
DMD0	DMD1	DMD2	DMD3				
VCT							

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
STC     SB,R2R0
STC     FLG,[A0]
STC     DMD2,A1
STC     DCT1,mem[A2]
```

STCTX

上下文保存
Store Context

STCTX

【语法】
STCTX src,dest

【指令码 / 周期数】
Page=238

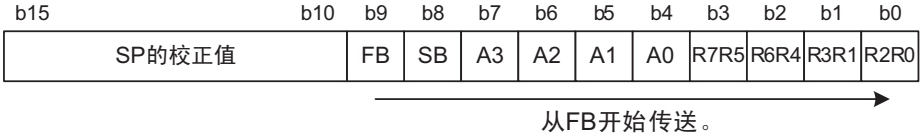
【操作】

```
for (i=0 ; i< n*1 ; i++) {
        SP    =   SP - 4;
        *SP   =   registers(dest[src]);
}
```

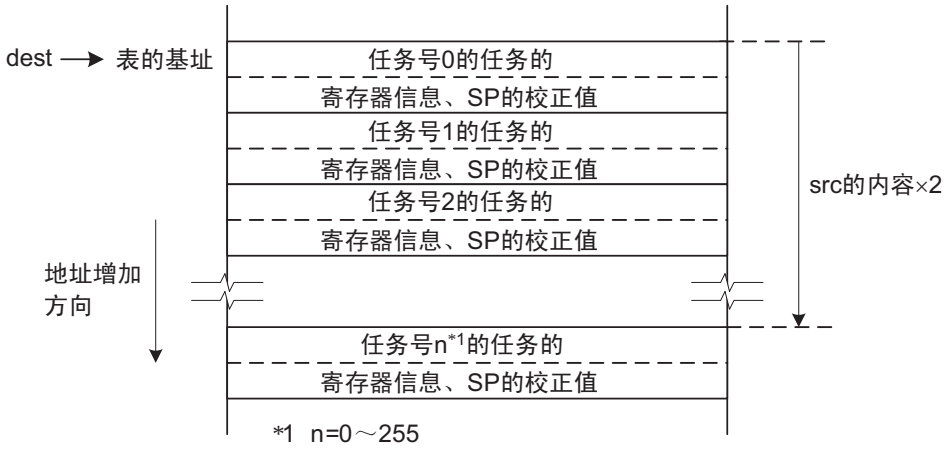
*1 n 是要保存的寄存器个数。

【功能】

- 将任务的上下文保存到堆栈区。
- 必须给 src 设定保存任务号的 RAM 地址、给 dest 设定表数据的起始地址。
- 根据任务号，从表数据中指定需要的寄存器信息，并且根据此寄存器的信息将各寄存器传送到堆栈区，然后从堆栈指针 （SP）减去 SP 的校正值。必须给 SP 的校正值设定要传送的寄存器的合计字节数，并且不能给表数据设定 “0000h”。
- 要传送的寄存器信息由以下内容构成。当位为 “1” 时，表示要传送的寄存器；当位为 “0” 时，表示不要传送的寄存器。



- 表数据由以下的内容构成。 dest所示的地址为表的基址，在以 2 倍 src 内容为偏移量的地址中保存的字数据的 bit0～9 表示寄存器的信息， bit10～15 表示堆栈指针的校正值。



【可选择的 src/dest】

src	dest
abs:16	dsp:24

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
STCTX		ram,rom_tbl
```

STNZ

带条件的传送
Store on Not Zero

STNZ

【语法】

STNZ.size src,dest

B , W , L

【指令码 / 周期数】

Page=238

【操作】

```
if (Z==0)
        dest  =  src;
```

【功能】

- 当Z标志为 “0”时，将src传送到dest；当Z标志为 “1”时， dest不变化。

【可选择的 src/dest】

src				dest*1			
~~R0L/R0/R2R0~~	~~R0H/R2/R3R1~~	~~R2L/R1/R6R4~~	~~R2H/R3/R7R5~~	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
~~R1L/R4/A0~~	~~R1H/R6/A1~~	~~R3L/R5/A2~~	~~R3H/R7/A3~~	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~		dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
STNZ.B      #1,R0L
STNZ.W      #5,[A0]
STNZ.L      #1000h,R6R4
```

STOP

停止
Stop

STOP

【语法】
STOP

【指令码 / 周期数】
Page=238

【操作】

【功能】

- 停止程序的执行。当接受优先级高于中断优先级选择位（用于从停止模式/等待模式的返回）的中断，或者发生复位时，开始程序的执行。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
STOP

STZ

带条件的传送
Store on Zero

STZ

【语法】

STZ.size src,dest

B , W , L

【指令码 / 周期数】

Page=239

【操作】

```
if (Z==1)
    dest = src;
```

【功能】

- 当Z标志为 “1”时，将src传送到dest；当Z标志为 “0”时，否则dest不变。

【可选择的 src/dest】

src				dest*1			
~~R0L/R0/R2R0~~	~~R0H/R2/R3R1~~	~~R2L/R1/R6R4~~	~~R2H/R3/R7R5~~	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
~~R1L/R4/A0~~	~~R1H/R6/A1~~	~~R3L/R5/A2~~	~~R3H/R7/A3~~	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~		dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
STZ.B    #1,R0L
STZ.W    #5,[A0]
STZ.L    #1000h,R6R4
```

STZX

带条件的传送
Store according to Zero Flag

STZX

【语法】

STZX.size src1,sec2,dest

B , W , L

【指令码 / 周期数】

Page=239

【操作】

```
if (Z==1)
        dest  =  src1;
else
        dest  =  src2;
```

【功能】

- 当Z标志为 “1”时，将src1传送到dest；当Z标志为 “0”时，将src2传送到dest。

【可选择的 src/dest】

src1 src2				dest*1			
~~R0L/R0/R2R0~~	~~R0H/R2/R3R1~~	~~R2L/R1/R6R4~~	~~R2H/R3/R7R5~~	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
~~R1L/R4/A0~~	~~R1H/R6/A1~~	~~R3L/R5/A2~~	~~R3H/R7/A3~~	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
~~#IMMEX~~	~~dsp:8[FB]~~	~~dsp:16[FB]~~	~~dsp:24~~		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	~~dsp:16~~	~~dsp:16[SB]~~	~~dsp:24[SB]~~		dsp:16	dsp:16[SB]	dsp:24[SB]
~~[A0]~~	~~dsp:8[A0]~~	~~dsp:16[A0]~~	~~dsp:24[A0]~~	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
~~[A1]~~	~~dsp:8[A1]~~	~~dsp:16[A1]~~	~~dsp:24[A1]~~	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
~~[A2]~~	~~dsp:8[A2]~~	~~dsp:16[A2]~~	~~dsp:24[A2]~~	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
~~[A3]~~	~~dsp:8[A3]~~	~~dsp:16[A3]~~	~~dsp:24[A3]~~	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
STZX.B    #1,#2,R0L
STZX.W    #5,#10,[A0]
STZX.L    #1000h,#4000h,R6R4
```

SUB

不带借位的减法运算
Subtract

SUB

【语法】

SUB.size src,dest

B , W , L

【指令码 / 周期数】

Page=240

【操作】

dest = dest - src;

【功能】

- 从dest减去src，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的运算发生上溢。当运算结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

C：当不带符号的运算结果大于等于 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
SUB.B    #2,R0L
SUB.W    R0,R2
SUB.L    A0,R7R5
SUB.B    R3L,[A0]
SUB.W    [A1],R4
SUB.L    R2R0,[A3]
```

SUBF

浮点减法运算
Subtruct Floating Piont

SUBF

【语法】
SUBF src,dest

【指令码 / 周期数】
Page=241

【操作】
dest = dest - src ;

【功能】
- 从dest减去src，结果保存到dest。
- 根据标志寄存器 （FLG）指定的舍入模式，将运算结果进行舍入。
- 在运算结果超过最大规格化数时，根据符号，结果为正的最大值 （7F7FFFFFh）或者负的最大值 （FF7FFFFFh）。
- 在运算结果小于最小规格化数时，根据舍入模式，结果为零 （00000000h）或者正的最小值 （00800000h）或者负的最小值 （80800000h）。
- 非法输入的运算结果为不定值。

【可选择的 src/dest】

src*1				dest*1			
~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5	~~R0L/R0~~/R2R0	~~R0H/R2~~/R3R1	~~R2L/R1~~/R6R4	~~R2H/R3~~/R7R5
~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3	~~R1L/R4~~/A0	~~R1H/R6~~/A1	~~R3L/R5~~/A2	~~R3H/R7~~/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	FO	FU	U	I	O	B	S	Z	D	C
变化	○	○	—	—	○	—	○	○	—	○

条件

FO：当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
FU：当非法输入或者运算结果小于最小规格化数时为 “1”，否则为 “0”。
O：当非法输入或者运算结果超过最大规格化数时为 “1”，否则为 “0”。
S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。
Z：当运算结果的全部位是 “0” 时为 “1”，否则为 “0”。
C：为不定值。

【记述例子】

```
SUBF    R2R0,R3R1
SUBF    [A1],R2R0
SUBF    mem[FB],R3R1
```

SUNTIL

字符串搜索
Search Equal String

SUNTIL

【语法】

SUNTIL.size
B , W , L

【指令码 / 周期数】
Page=242

【操作】

```
while (*A0 != R0L/R0/R2R0*1 && R7R5 != 0*2) {
        A0++;*3
        R7R5    =    R7R5 - 1;
}
```

*1 在给长度说明符 （.size）指定了 “.B”时为 R0L，指定了 “.W”时为 R0，指定了 “.L”时为 R2R0。

*2 即使在给 R7R5 设定 “0”后执行，也只进行 1 次比较，指令结束时的 A0 和标志为不定值。

*3 在给长度说明符 （.size）指定了 “.B”时加 1，指定了 “.W”时加 2，指定了 “.L”时加 4。

【功能】

- 以地址增加方向，从A0所示的比较目标地址搜索和R0L/R0/R2R0的内容相同的数据，搜索到出现相同的数据为止，比较次数由R7R5指定。
- 给A0设定比较目标地址、 R7R5设定比较次数。
- 能给R7R5设定的最大值为 “00FFFFFFh”。
- 根据 “*A0 – R0L/R0/R2R0”的运算结果，标志发生变化。
- 指令结束时的A0表示相同数据的地址，如果没有相同数据，就表示下一个数据的地址。
- 指令结束后的R7R5为 “初始值 - 比较次数”。
- 如果在指令执行过程中发生中断请求，就在比较1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的比较发生上溢。当比较结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当比较结果 MSB 是 “1”时为 “1”，否则为 “0”。

Z：当相同时为 “1”，否则为 “0”。

C：当不带符号的比较结果大于等于 “0”时为 “1”，否则为 “0”。

【记述例子】

```
SUNTIL.W
```

SWHILE

字符串搜索
Search Unequal String

SWHILE

【语法】

SWHILE.size
　　└──────── B , W , L

【指令码 / 周期数】
Page=242

【操作】

```
while (*A0 == R0L/R0/R2R0*1 && R7R5 != 0*2) {
        A0++;*3
        R7R5    =    R7R5 - 1;
}
```

*1　在给长度说明符 （.size）指定了 “.B” 时为 R0L，指定了 “.W” 时为 R0，指定了 “.L” 时为 R2R0。

*2　即使在给 R7R5 设定 “0” 后执行，也只进行 1 次比较，指令结束时的 A0 和标志为不定值。

*3　在给长度说明符 （.size）指定了 “.B” 时加 1，指定了 “.W” 时加 2，指定了 “.L” 时加 4。

【功能】

- 以地址增加方向，从A0所示的比较目标地址搜索和R0L/R0/R2R0的内容不相同的数据，搜索到出现不同的数据为止，比较次数由R7R5指定。
- 给A0设定比较目标地址、 R7R5设定比较次数。
- 能给R7R5设定的最大值为 “00FFFFFFh”。
- 根据 “*A0 – R0L/R0/R2R0” 的运算结果，标志发生变化。
- 指令结束时的A0表示不同数据的地址，如果有相同数据，就表示下一个数据的地址。
- 指令结束后的R7R5为 “初始值 - 比较次数”。
- 如果在指令执行过程中发生中断请求，就在比较1个数据后接受中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	○	—	○	○	—	○

条件

O：表示带符号的比较发生上溢。当比较结果超过 –128(.B) 或者 +127(.B)、 –32768(.W) 或者 +32767(.W)、–2147483648(.L) 或者 +2147483647(.L) 时为 “1”，否则为 “0”。

S：当比较结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当全部相同时为 “1”，否则为 “0”。

C：当不带符号的比较结果大于等于 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
SWHILE.W
```

TST

测试
Test Logical

TST

【语法】

TST.size src,dest

B , W , L

【指令码 / 周期数】

Page=242

【操作】

dest & src;

【功能】

- 根据取得的dest和src的逻辑与结果，标志寄存器 （FLG）的各标志发生变化。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
TST.B    #2,R0L
TST.W    R0,R2
TST.L    A0,R7R5
TST.B    R3L,mem[A0]
TST.W    [A1],R4
TST.L    R2R0,[A3]
```

UND

未定义指令中断
Undefined Instruction Interrupt

UND

【语法】
UND

【指令码 / 周期数】
Page=243

【操作】

```
SP    =   SP - 4;
*SP   =   FLG;
SP    =   SP - 4;
*SP   =   PC + 1;
PC    =   *(long *)0xFFFFFFDC;
```

【功能】

- 发生未定义指令中断。
- 未定义指令中断是非屏蔽中断。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化 *1	○	○	—	—	—	—	○	—

*1 执行指令前的标志被压栈。

条件

U：为 “0”。
I：为 “0”。
D：为 “0”。

【记述例子】

```
UND
```

WAIT

等待
Wait

WAIT

【语法】
WAIT

【指令码 / 周期数】
Page=243

【操作】

【功能】

- 停止程序的执行。当接受优先级高于中断优先级选择位（由于从停止模式/等待模式的返回）的中断或者发生复位时，开始程序的执行。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】
WAIT

XCHG

交换
Exchange

XCHG

【语法】

XCHG.size src,dest

B , W , L

【指令码 / 周期数】

Page=244

【操作】

```
tmp0 = src;
src  = dest;
dest = tmp0;
```

tmp0：临时寄存器

【功能】

- 将src和dest的内容进行交换。

【可选择的 src/dest】

src*1				dest*2			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
					dsp:8[FB]	dsp:16[FB]	dsp:24
					dsp:16	dsp:16[SB]	dsp:24[SB]
				[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
				[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
				[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
				[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用组 1 寄存器直接。

*2 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	—	—	—	—

【记述例子】

```
XCHG.B    R0H,R0L
XCHG.W    R0,R2
XCHG.L    A0,R7R5
XCHG.B    R3L,[A0]
XCHG.W    [A1],R4
XCHG.L    R2R0,[A3]
```

XOR

逻辑异或
Exclusive Or Logical

XOR

【语法】

XOR.size src,dest

B , W , L

【指令码 / 周期数】

Page=244

【操作】

dest = dest ^ src;

【功能】

- 取dest和src的逻辑异或，结果保存到dest。

【可选择的 src/dest】

src*1				dest*1			
R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5	R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3	R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
#IMMEX	dsp:8[FB]	dsp:16[FB]	dsp:24		dsp:8[FB]	dsp:16[FB]	dsp:24
#IMM	dsp:16	dsp:16[SB]	dsp:24[SB]		dsp:16	dsp:16[SB]	dsp:24[SB]
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 能使用间接指令寻址或者组 1 寄存器直接。

【标志的变化】

标志	U	I	O	B	S	Z	D	C
变化	—	—	—	—	○	○	—	—

条件

S：当运算结果 MSB 是 “1” 时为 “1”，否则为 “0”。

Z：当运算结果是 “0” 时为 “1”，否则为 “0”。

【记述例子】

```
XOR.B    #2,R0L
XOR.W    R0,R2
XOR.L    A0,R7R5
XOR.B    R3L,mem[A0]
XOR.W    [A1],R4
XOR.L    R2R0,[A3]
```

3.3 变址指令

在本节中说明各变址指令。

变址指令是对应数组的指令，给接着变址指令执行的指令的 src 和 dest 所示的地址加上（不带符号的加法）变址指令的 src 内容，相加的结果为有效地址。

能修改的范围为 0 ～ FFFFFFFFh 的 4GB。变址指令的 src 范围根据下一条要执行的指令长度说明符发生以下的变化。

下一条要执行的 指令长度说明符	变址指令的长度说明符	src 的范围	被加的值
.B	.B	0 ～ FFh	src 的内容
	.W	0 ～ FFFFh	
	.L	0 ～ FFFFFFFFh	
.W	.B	0 ～ FFh	src 的内容 ×2
	.W	0 ～ FFFFh	
	.L	0 ～ 7FFFFFFFh	
.L	.B	0 ～ FFh	src 的内容 ×4
	.W	0 ～ FFFFh	
	.L	0 ～ 3FFFFFFFh	

在变址指令之后不能马上接受中断请求。

变址指令有以下 4 种：

INDEXB
INDEX1
INDEX2
BITINDEX

3.3.1 INDEXB.size src

INDEXB （Index Both Operands）指令给下一条要执行的指令的 src 和 dest 所示的地址加上（不带符号的加法） INDEXB 指令的 src 内容，相加的结果为有效地址。

必须给接着 INDEXB 指令执行的指令的 src 和 dest 选择存取存储器的寻址方式。

例 1：下一条指令的长度说明符为 “.B” 的情况

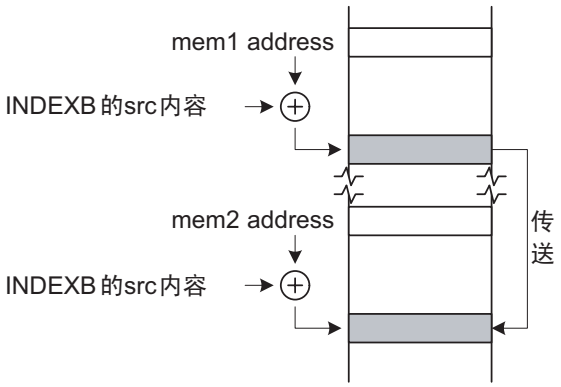
```
INDEXB.W    src
MOV.B:G     mem1, mem2
```

存储器

C语言的操作说明：

```
short       src;
char        mem1[], mem2[];

mem2[src] = mem1[src];
```



例 2：下一条指令的长度说明符为 “.W” 的情况

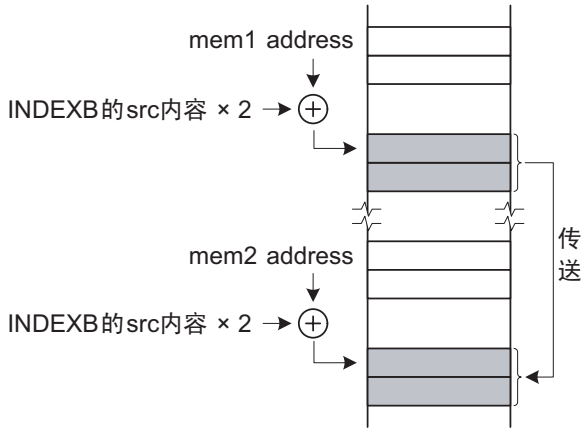
```
INDEXB.W    src
MOV.W:G     mem1, mem2
```

存储器

C语言的操作说明：

```
short       src;
short       mem1[], mem2[];

mem2[src] = mem1[src];
```



例 3：下一条指令的长度说明符为 “.L” 的情况

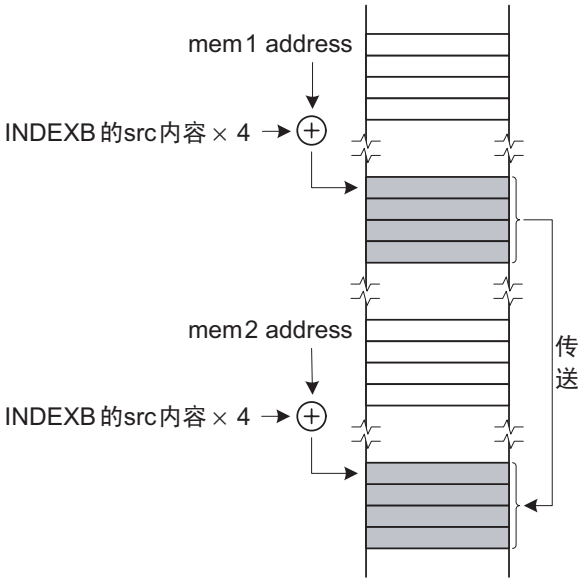
```
INDEXB.W    src
MOV.L:G     mem1, mem2
```

存储器

C语言的操作说明：

```
short       src;
long        mem1[], mem2[];

mem2[src] = mem1[src];
```



由 INDEXB 指令修改的指令：

ADC、 ADD:G*1*2、 ADDF、 AND:G*1、 CMP:G*1、 CMPF、 CNVIF、 DADC、 DADD、 DIV、 DIVF、 DIVU、 DIVX、 DSBB、 DSUB、 EXTS*3、 EXTZ:G*1*3、 MAX、 MIN、 MOV:G*1*4、 MUL、 MULF、 MULU、 OR、 ROUND、 SBB、 SUB、 SUBF、 TST、 XOR指令的src和dest

*1 只能使用 G 格式。

*2 SP 不能用于 ADD 指令的 dest。

*3 对于 src(mem1)， INDEXB 的 src 倍数取决于符号扩展 / 零扩展前的长度；对于 dest(mem2)，INDEXB 的 src 倍数取决于符号扩展 / 零扩展后的长度。

*4 dsp:8[SP] 不能用于 MOV 指令的 src 和 dest。

上述以外的指令不能用于 INDEXB 指令的下一条指令。

3.3.2 INDEX1.size src

INDEX1 （Index 1st Operand）指令给下一条要执行的指令的第一操作数所示的地址加上 （不带符号的加法） INDEX1 指令的 src 内容，相加的结果为有效地址。

必须给接着 INDEX1 指令执行的指令的第一操作数选择存取存储器的寻址方式。

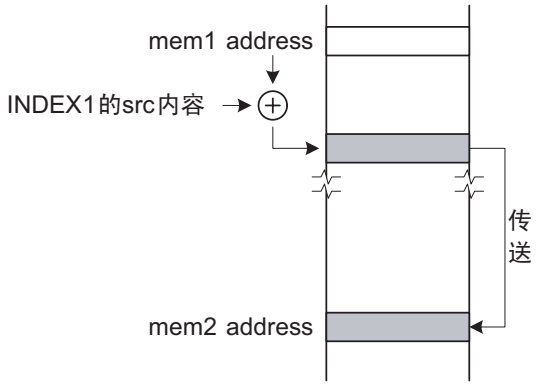
例 1：下一条指令的长度说明符为 “.B” 的情况

```
INDEX1.W    src
MOV.B:G     mem1, mem2
            |
            存储器
```

C语言的操作说明：

```
short       src;
char        mem1[], mem2;

mem2 = mem1[src];
```



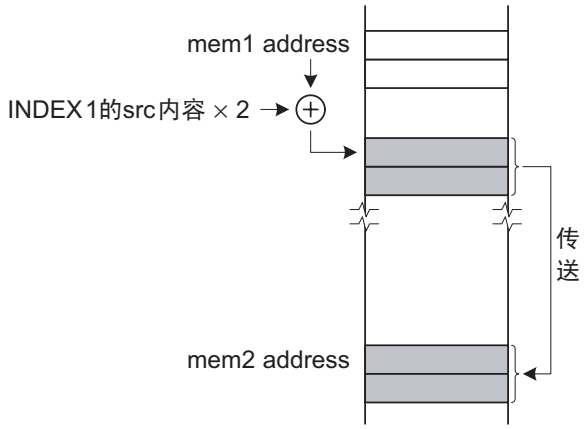
例 2：下一条指令的长度说明符为 “.W” 的情况

```
INDEX1.W    src
MOV.W:G     mem1, mem2
            \
            存储器
```

C语言的操作说明：

```
short       src;
short       mem1[], mem2;

mem2 = mem1[src];
```



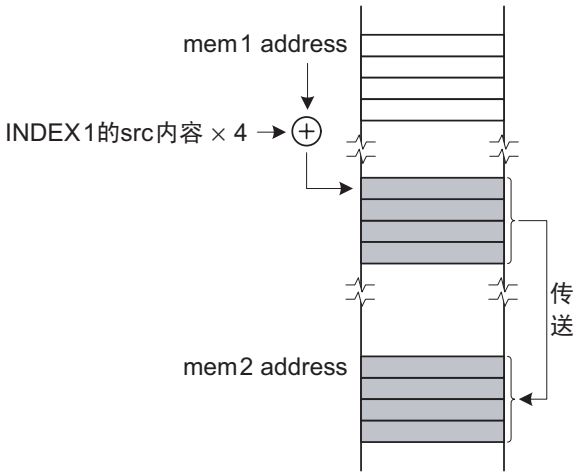
例 3：下一条指令的长度说明符为 “.L” 的情况

```
INDEX1.W    src
MOV.L:G     mem1, mem2
            \
            存储器
```

C语言的操作说明：

```
short       src;
long        mem1[], mem2;

mem2 = mem1[src];
```



由 INDEX1 指令修改的指令：

ADC、ADD:G*1*2、ADDF、AND:G*1、BTST:G*1、CMP:G*1、CMPF、CNVIF、DADC、DADD、DIV、DIVF、DIVU、DIVX、DSBB、DSUB、EDIV、EDIVU、EDIVX、EMUL、EMULU、EXTS*3、EXTZ*3、JMPI、JSRI、LDC、MAX、MIN、MOV:G*1*4、MOV:S*5、MOV*Dir**6、MUL、MULF、MULU、OR、PUSH、ROUND、SBB、SUB、SUBF、TST、XOR指令的src

ABS、ADCF、ADD:Q*2、ADD:S、ADSF、AND:S、BCLR、BM*Cnd*、BNOT、BSET、BTSTC、BTSTS、CLIP、CMP:Q、CMP:S、DEC、INC、MOV:G*7、MOV:Q、MOV:S*8、MOV:Z、MOV*Dir**9、NEG、NOT、POP、ROLC、RORC、ROT、SC*Cnd*、SHA、SHL、STC、STNZ、STZ、STZX、XCHG指令的dest

*1 只能使用 G 格式。

*2 SP 不能用于 ADD 指令的 dest。

*3 INDEX1 的 src 倍数取决于符号扩展 / 零扩展前的长度。

*4 dsp:8[SP] 不能用于 MOV 指令的 src。

*5 能用于 MOV:S src,R2R0/R0/R0L 和 MOV:S.L src,A0。

*6 能用于 MOV*Dir* src,R0L，长度说明符作为 “.B” 处理。

*7 能用于 MOV:G dsp:8[SP],dest。

*8 能用于 MOV:S R2R0/R0/R0L,dest 和 MOV:S #IMM,dest。

*9 能用于 MOV*Dir* R0L,dest，长度说明符作为 “.B” 处理。

上述以外的指令不能用于 INDEX1 指令的下一条指令。

3.3.3 INDEX2.size src

INDEX2 （Index 2nd Operand）指令给下一条要执行的指令的第二操作数所示的地址加上 （不带符号的加法） INDEX2 指令的 src 内容，相加的结果为有效地址。

必须给接着 INDEX2 指令执行的指令的 dest 选择存取存储器的寻址方式。

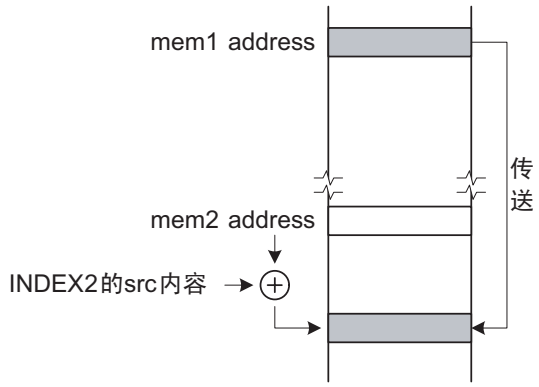
例 1：下一条指令的长度说明符为 “.B” 的情况

```
INDEX2.W   src
MOV.B:G    mem1, mem2
                 |
              存储器
```

C语言的操作说明：

```
short   src;
char    mem1, mem2[];

mem2[src] = mem1;
```



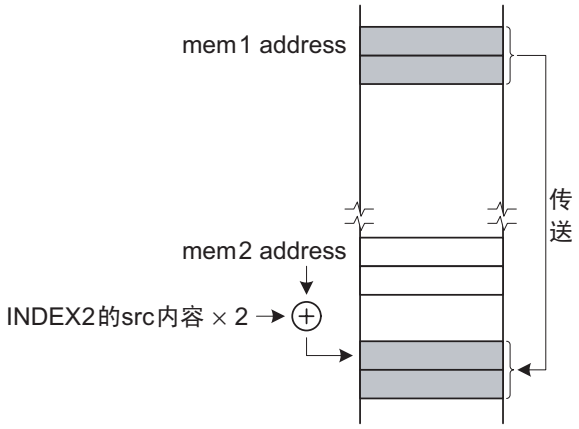
例 2：下一条指令的长度说明符为 “.W” 的情况

```
INDEX2.W    src
MOV.W:G     mem1, mem2
                 |
              存储器
```

C语言的操作说明：

```
short       src;
short       mem1, mem2[];

mem2[src] = mem1;
```



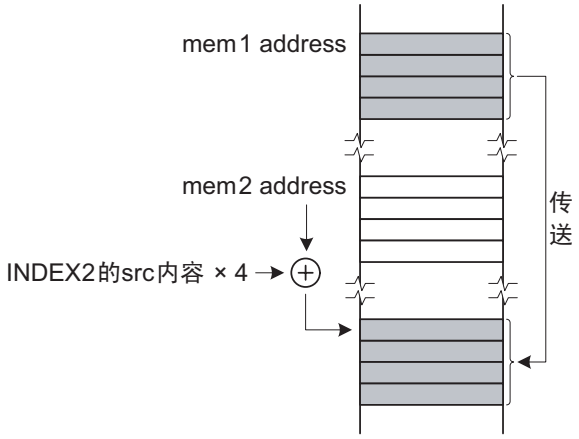
例 3：下一条指令的长度说明符为 “.L” 的情况

```
INDEX2.W    src
MOV.L:G     mem1, mem2
                 |
              存储器
```

C语言的操作说明：

```
short       src;
long        mem1, mem2[];

mem2[src] = mem1;
```



由 INDEX2 指令修改的指令：

ADC、 ADD:G*1*2、 ADDF、 AND:G*1、 CMP:G*1、 CMPF、 CNVIF、 DADC、 DADD、 DIV、 DIVF、 DIVU、 DIVX、 DSBB、 DSUB、 EXTS*3、 EXTZ:G*1*3、 MAX、 MIN、 MOV:G*1*4、 MUL、 MULF、 MULU、 OR、 ROUND、 SBB、 SUB、 SUBF、 TST、 XOR 指令的 dest

*1 只能使用 G 格式。

*2 SP 不能用于 ADD 指令的 dest。

*3 INDEX2 的 src 倍数取决于符号扩展 / 零扩展后的长度。

*4 dsp:8[SP] 不能用于 MOV 指令的 src 和 dest。

上述以外的指令不能用于 INDEX2 指令的下一条指令。

3.3.4 BITINDEX.size src

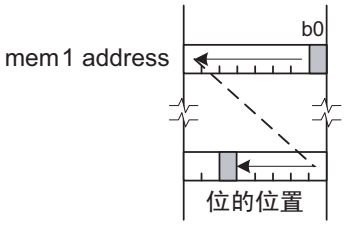
BITINDEX 指令 （Bit Index）将从下一条要执行的指令的操作数所示的地址的 bit0 到分离 BITINDEX 指令的 src 所示的位数的位作为对象。

接着 BITINDEX 指令执行的指令必须是位指令，并且必须给操作数选择存取存储器的寻址方式。

例 :

```
BITINDEX.B/W    src
BSET            3, mem1
```

位指令 无效 存储器



由 BITINDEX 指令修改的指令：

BTST:G*1指令的src

BCLR、 BMCnd、 BNOT、 BSET、 BTSTC、 BTSTS指令的dest

*1 只能使用 G 格式。

上述以外的指令不能用于 BITINDEX 指令的下一条指令。

3.3.5 能接着变址指令执行的指令

能接着各变址指令执行的指令如下表所示：

变址指令	有效指令	
INDEXB.B/W/L	ADC、 ADD:G*1*2、 ADDF、 AND:G*1、 CMP:G*1、 CMPF、 CNVIF、 DADC、 DADD、 DIV、 DIVF、 DIVU、 DIVX、 DSBB、 DSUB、 EXTS、 EXTZ:G*1、 MAX、 MIN、 MOV:G*1*3、 MUL、 MULF、 MULU、 OR、 ROUND、 SBB、 SUB、 SUBF、 TST、 XOR 上述指令的 src 和 dest	
INDEX1.B/W/L	ADC、 ADD:G*1*2、 ADDF、 AND:G*1、 BTST:G*1、 CMP:G*1、 CMPF、 CNVIF、 DADC、 DADD、 DIV、 DIVF、 DIVU、 DIVX、 DSBB、 DSUB、 EDIV、 EDIVU、 EDIVX、 EMUL、 EMULU、 EXTS、 EXTZ、 JMPI、 JSRI、 LDC、 MAX、 MIN、 MOV:G*1*4、 MOV:S*5、 MOV*Dir**6、 MUL、 MULF、 MULU、 MULX、 OR、 PUSH、 ROUND、 SBB、 SUB、 SUBF、 TST、 XOR 上述指令的 src	ABS、 ADCF、 ADD:Q*2、 ADD:S、 ADSF、 AND:S、 BCLR、 BM*Cnd*、 BNOT、 BSET、 BTSTC、 BTSTS、 CLIP、 CMP:Q、 CMP:S、 DEC、 INC、 MOV:G*7、 MOV:Q、 MOV:S*8、 MOV:Z、 MOV*Dir**9、 NEG、 NOT、 POP、 ROLC、 RORC、 ROT、 SC*Cnd*、 SHA、 SHL、 STC、 STNZ、 STZ、 STZX、 XCHG 上述指令的 dest
INDEX2.B/W/L	ADC、 ADD:G*1*2、 ADDF、 AND:G*1、 CMP:G*1、 CMPF、 CNVIF、 DADC、 DADD、 DIV、 DIVF、 DIVU、 DIVX、 DSBB、 DSUB、 EXTS、 EXTZ:G*1、 MAX、 MIN、 MOV:G*1*3、 MUL、 MULF、 MULU、 OR、 ROUND、 SBB、 SUB、 SUBF、 TST、 XOR 上述指令的 dest	
BITINDEX.B/W/L	BTST:G*1 上述指令的 src	BCLR、 BM*Cnd*、 BNOT、 BSET、 BTSTC、 BTSTS 上述指令的 dest

*1 只能使用 G 格式。

*2 SP 不能用于 ADD 指令的 dest。

*3 dsp:8[SP] 不能用于 MOV 指令的 src 和 dest。

*4 dsp:8[SP] 不能用于 MOV 指令的 src。

*5 能用于 MOV:S src,R0L/R0/R2R0 和 MOV:S.L src,A0。

*6 能用于 MOV*Dir* src,R0L。

*7 能用于 MOV:G dsp:8[SP],dest。

*8 能用于 MOV:S R0L/R0/R2R0,dest 和 MOV:S #IMM,dest。

*9 能用于 MOV*Dir* R0L,dest。

3.3.6 寻址方式

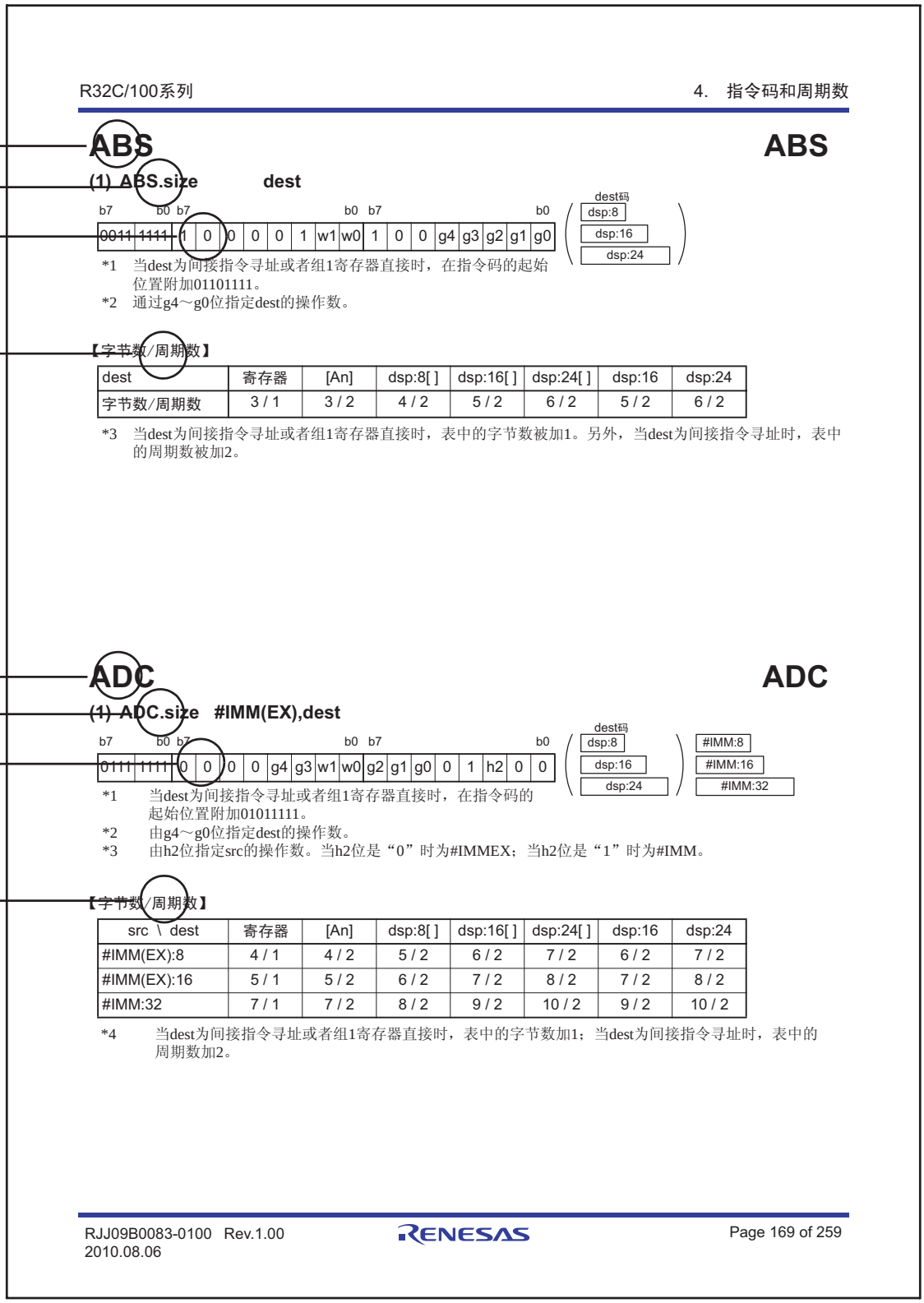
能接着变址指令执行的指令的有效寻址方式如下表所示，各指令能使用间接寻址方式。

src				dest			
[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]	[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]	[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]	[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]	[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]
	dsp:8[FB]	dsp:16[FB]			dsp:8[FB]	dsp:16[FB]	
	dsp:8[SB]	dsp:16[SB]	dsp:24[SB]		dsp:8[SB]	dsp:16[SB]	dsp:24[SB]
		dsp:16	dsp:24			dsp:16	dsp:24

4. 指令码和周期数

4.1 本章的阅读方法

在本章中，按各操作码对指令码和周期数进行说明。
通过以下的例子说明本章的阅读方法：



(1) 助记符

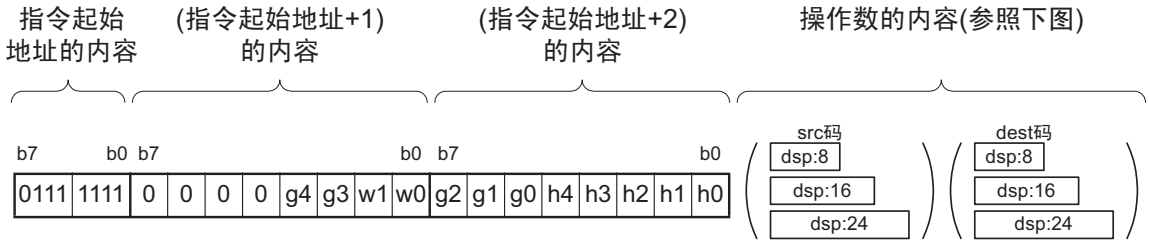
表示本页中说明的助记符。

(2) 语法

用符号表示指令的语法。

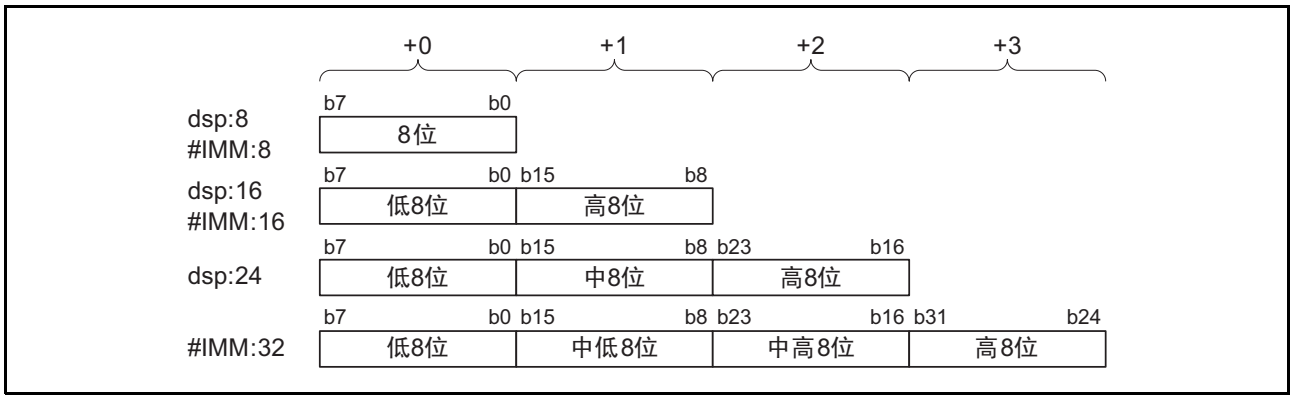
(3) 指令码

表示指令码。（）内的内容根据选择的 src/dest 而被省略。



有关 w1 位、 w0 位、 g4 ～ g0 位和 h4 ～ h0 位等的内容，请参照 “4.2 寻址”。

操作数的内容 （上述例子中 (指令起始地址 +3) 以及 (指令起始地址 +3) 以后的部分）如下分配：



(4) 字节数 / 周期数表

表示在执行本指令时所需的周期数和指令的字节数。

因为周期数记载的是最短周期数，所以在以下情况下有可能增加：

- 取到指令队列缓冲器中的字节数少时
- 写周期和读周期重叠并且等待操作数的存取时
- 存取SFR区或者外部存储区时
- 对总线周期插入了等待时

斜线的左侧是字节数，右侧是周期数。

4.2 寻址

4.2.1 运算长度的指定

运算长度由指令码中的 w1 位和 w0 位指定。

表 4.1 运算长度的指定

w1	w0	运算长度 *1	#IMMEX 的操作数和运算长度
0	0	字节	8 位立即数、字
0	1	字	8 位立即数、长字
1	0	长字	16 位立即数、长字
1	1	保留	保留

*1 2 个操作数指令时，如果 src 为 #IMMEX， w1 位和 w0 位遵守 #IMMEX 的规则。

4.2.2 操作数的指定

操作数的指定方法因指令和指令格式而不同。
按寻址方式表示操作数的指定方法如下所示。

(1) 一般指令寻址・一般格式

一般格式中使用的操作数 gen1 和 gen2 分别由 g4 ～ g0 位和 h4 ～ h0 位指定。

表 4.2 一般格式中的操作数指定

g4/h4	g3/h3	g2/h2		gen1/gen2			
			g1/h1	0	0	1	1
			g0/h0	0	1	0	1
0	0	0		R0L/R0/R2R0	R0H/R2/R3R1	R2L/R1/R6R4	R2H/R3/R7R5
0	0	1		R1L/R4/A0	R1H/R6/A1	R3L/R5/A2	R3H/R7/A3
0	1	0		#IMMEX*1	dsp:8[FB]	dsp:16[FB]	dsp:24
0	1	1		#IMM*1	dsp:16	dsp:16[SB]	dsp:24[SB]
1	0	0		[A0]	dsp:8[A0]	dsp:16[A0]	dsp:24[A0]
1	0	1		[A1]	dsp:8[A1]	dsp:16[A1]	dsp:24[A1]
1	1	0		[A2]	dsp:8[A2]	dsp:16[A2]	dsp:24[A2]
1	1	1		[A3]	dsp:8[A3]	dsp:16[A3]	dsp:24[A3]

*1 不能给 dest 指定 #IMMEX 和 #IMM。

(2) 一般指令寻址・快速格式

快速格式中使用的操作数 #IMM:4 由指令码中的 q3 ～ q0 位指定。

表 4.3 快速格式中的操作数指定 （1）

q3	q2	q1	q0	IMM:4	q3	q2	q1	q0	IMM:4
0	0	0	0	0	1	0	0	0	-8
0	0	0	1	1	1	0	0	1	-7
0	0	1	0	2	1	0	1	0	-6
0	0	1	1	3	1	0	1	1	-5
0	1	0	0	4	1	1	0	0	-4
0	1	0	1	5	1	1	0	1	-3
0	1	1	0	6	1	1	1	0	-2
0	1	1	1	7	1	1	1	1	-1

ADD 指令的快速格式中使用的 #IMM:3 由指令码中的 q2 ～ q0 位指定。

表 4.4 快速格式中的操作数指定 （2）

q2	q1	q0	IMM:3
0	0	0	保留
0	0	1	4
0	1	0	8
0	1	1	保留
1	0	0	12
1	0	1	16
1	1	0	20
1	1	1	保留

(3) 一般指令寻址・短格式

短格式中使用的操作数 gen0 由指令码中的 s1 位和 s0 位指定。

表 4.5 短格式中的操作数指定

s1	s0	gen0
0	0	R0L/R0/R2R0 (R0H/R2/R3R1)*1
0	1	dsp:16
1	0	dsp:8[FB]
1	1	dsp:8[SB]

*1 R0H/R2/R3R1 只用于 MOV:S src,R0L/R0/R2R0 的情况。

(4) 寄存器直接

在只能使用寄存器直接寻址的 EDIV、EDIVU、EDIVX、EMUL、EMULU、EXTZ:S、MOVA 指令的 dest 以及在 ROT、SHA、SHL、XCHG 指令的 src 使用的操作数 greg 由 q2 ～ q0 位指定，各指令使用以下寄存器：

表 4.6 EDIV、EDIVU、EDIVX 指令的操作数指定

q2	q1	q0	greg
0	0	0	R0/R2R0/R3R1R2R0
0	0	1	—
0	1	0	R2/R3R1/R7R5R6R4
0	1	1	—
1	0	0	R1/R6R4/A1A0
1	0	1	—
1	1	0	R3/R7R5/A3A2
1	1	1	—

表 4.7 EMUL 指令和 EMULU 指令的操作数指定

q2	q1	q0	greg
0	0	0	R0L/R0/R2R0
0	0	1	—
0	1	0	R2L/R1/R6R4
0	1	1	—
1	0	0	R1L/R4/A0
1	0	1	—
1	1	0	R3L/R5/A2
1	1	1	—

表 4.8 EXTZ:S 指令的操作数指定

q2	q1	q0	greg
0	0	0	—
0	0	1	—
0	1	0	—
0	1	1	—
1	0	0	A0
1	0	1	A1
1	1	0	A2
1	1	1	A3

表 4.9 MOVA、 MULX、 ROT、 SHA、 SHL、 XCHG 指令的操作数指定

q2	q1	q0	greg
0	0	0	R0L/R0/R2R0
0	0	1	R0H/R2/R3R1
0	1	0	R2L/R1/R6R4
0	1	1	R2H/R3/R7R5
1	0	0	R1L/R4/A0
1	0	1	R1H/R6/A1
1	1	0	R3L/R5/A2
1	1	1	R3H/R7/A3

(5) 专用寄存器直接 （CPU）

专用寄存器直接寻址中使用的操作数 creg 由 q2 ～ q0 位指定。

表 4.10 专用寄存器直接寻址中的操作数指定

q2	q1	q0	creg
0	0	0	SVP
0	0	1	INTB
0	1	0	SVF
0	1	1	FLG
1	0	0	SP
1	0	1	ISP
1	1	0	FB
1	1	1	SB

(6) 专用寄存器直接 （DMAC 和 VCT）

在将 DMAC 相关的专用寄存器和向量寄存器 （VCT）指定为操作数时，要在指令码的最后附加 #IMM:8，各寄存器由 #IMM:8 的位列指定。

表 4.11 专用寄存器直接寻址中的寄存器指定 （1）

#IMM:8 （16 进制）	寄存器
03h	VCT
00h ～ 07h （03h 除外）	通道 0 的 DMA 寄存器
08h ～ 0Fh （0Bh 除外）	通道 1 的 DMA 寄存器
10h ～ 17h （13h 除外）	通道 2 的 DMA 寄存器
18h ～ 1Fh （1Bh 除外）	通道 3 的 DMA 寄存器

表 4.12 专用寄存器直接寻址中的寄存器指定 （2）

#IMM:8 的低 3 位			寄存器	
0	0	0	DSAn*1	DMA 源地址寄存器
0	0	1	DDAn*1	DMA 目标地址寄存器
0	1	0	DCTn*1	DMA 终端计数寄存器
0	1	1	保留	
1	0	0	DSRn*1	DMA 源地址重加载寄存器
1	0	1	DDRn*1	DMA 目标地址重加载寄存器
1	1	0	DCRn*1	DMA 终端计数重加载寄存器
1	1	1	DMDn*1	DMA 模式寄存器

*1 n 为通道号。

(7) 程序计数器相对・短格式

程序计数器相对・短格式中使用的操作数 dsp3 由 q2 ～ q0 位指定。

表 4.13 程序计数器相对・短格式中的操作数指定

q2	q1	q0	dsp3
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

(8) 位指令寻址

位指令寻址中使用的操作数 bit 由 b2 ～ b0 位指定。

表 4.14 位指令寻址中的位的位置指定

b2	b1	b0	bit
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

(9) FLG 直接

FLG 直接寻址中使用的操作数 flg 由 q2 ～ q0 位指定。

表 4.15 标志直接寻址中的操作数指定

q2	q1	q0	flg
0	0	0	C
0	0	1	D
0	1	0	Z
0	1	1	S
1	0	0	B
1	0	1	O
1	1	0	I
1	1	1	U

4.2.3 条件的指定

条件由指令码中的 c3 ～ c0 位指定。

表 4.16 条件码的指定

c3	c2	c1	c0	条件	c3	c2	c1	c0	条件
0	0	0	0	LTU/NC	1	0	0	0	GEU/C
0	0	0	1	LEU	1	0	0	1	GTU
0	0	1	0	NE/NZ	1	0	1	0	EQ/Z
0	0	1	1	PZ	1	0	1	1	N
0	1	0	0	NO	1	1	0	0	O
0	1	0	1	GT	1	1	0	1	LE
0	1	1	0	GE	1	1	1	0	LT
0	1	1	1	保留	1	1	1	1	保留

4.3 指令码 / 周期数

从下一页开始详细说明 R32C/100 系列的指令码和周期数。

ABS

ABS

(1) ABS.size dest

b7		b0	b7						b0	b7							b0
0011	1111	1	0	0	0	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ADC

ADC

(1) ADC.size #IMM(EX),dest

b7		b0	b7						b0	b7							b0
0111	1111	0	0	0	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8 / #IMM:16 / #IMM:32

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM(EX):16	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
#IMM:32	7 / 1	7 / 2	8 / 2	9 / 2	10 / 2	9 / 2	10 / 2

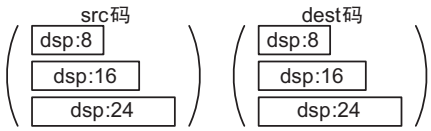
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ADC

ADC

(2) ADC.size src,dest

b7								b0 b7								b0
0111	1111	0	0	0	0	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
[An]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:8[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:16[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24[ ]	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3
dsp:16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

ADCF

ADCF

(1) ADCF.size dest

b7							b0 b7							b0
1	0	0	0	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

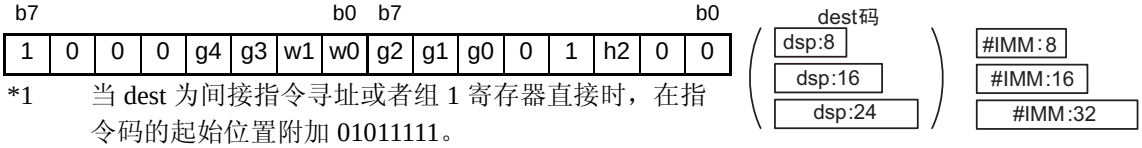
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ADD ADD

(1) ADD.size:G #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

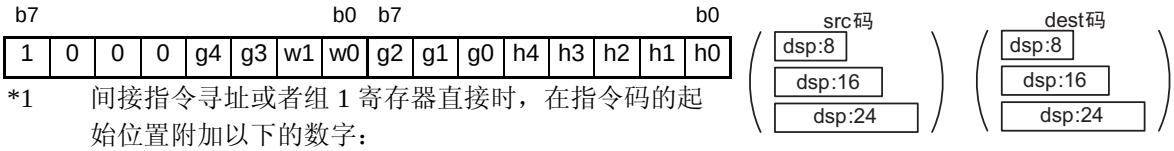
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

ADD ADD

(2) ADD.size:G src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

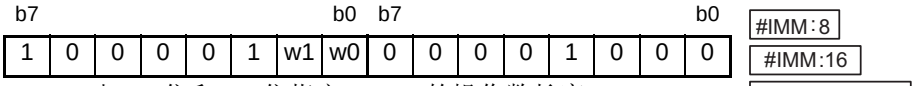
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2
[An]	2 / 2	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3
dsp:8[ ]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:16[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:16	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

ADD ADD

(3) ADD.L:G #IMM,SP



*1 由 w1 位和 w0 位指定 #IMM 的操作数长度。

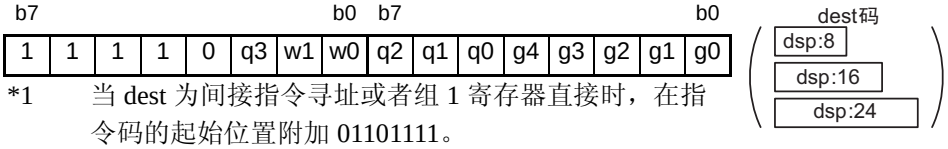
*2 与 w1 位和 w0 位无关，将 #IMM 符号扩展为 32 位。

【字节数 / 周期数】

src	#IMM:8	#IMM:16	#IMM:32
字节数 / 周期数	3 / 2	4 / 2	6 / 2

ADD ADD

(4) ADD.size:Q #IMM:4,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 q3 ～ q0 的 4 位指定 #IMM:4，由 g4 ～ g0 位指定 dest 的操作数。

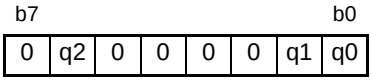
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ADD ADD

(5) ADD.L:Q #IMM:3,SP



*1 由 q2 ～ q0 位指定 #IMM:3，详细内容请参照 “4.2.2 操作数的指定”。

【字节数 / 周期数】

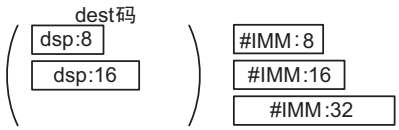
字节数 / 周期数	1 / 1

ADD

ADD

(6) ADD.size:S #IMM,dest

b7							b0
0	1	1	1	s1	s0	w1	w0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 dest 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	dsp:8[ ]	dsp:16
#IMM:8	2 / 1	3 / 2	4 / 2
#IMM:16	3 / 1	4 / 2	5 / 2
#IMM:32	5 / 1	6 / 2	7 / 2

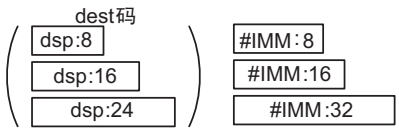
*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ADDF

ADDF

(1) ADDF #IMM(EX),dest

b7		b0	b7							b0	b7						b0
0111	1111	1	0	0	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
#IMMEX:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
#IMM:32	7 / 4	7 / 5	8 / 5	9 / 5	10 / 5	9 / 5	10 / 5

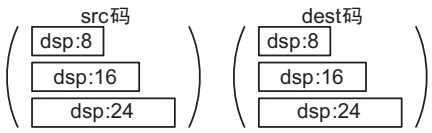
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ADDF

ADDF

(2) ADDF src,dest

b7	b0	b7							b0	b7							b0
0111	1111	1	0	0	0	g4	g3	1	0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 4	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
[An]	3 / 5	3 / 6	4 / 6	5 / 6	6 / 6	5 / 6	6 / 6
dsp:8[ ]	4 / 5	4 / 6	5 / 6	6 / 6	7 / 6	6 / 6	7 / 6
dsp:16[ ]	5 / 5	5 / 6	6 / 6	7 / 6	8 / 6	7 / 6	8 / 6
dsp:24[ ]	6 / 5	6 / 6	7 / 6	8 / 6	9 / 6	8 / 6	9 / 6
dsp:16	5 / 5	5 / 6	6 / 6	7 / 6	8 / 6	7 / 6	8 / 6
dsp:24	6 / 5	6 / 6	7 / 6	8 / 6	9 / 6	8 / 6	9 / 6

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

ADSF

ADSF

(1) ADSF.size dest

b7	b0	b7								b0	b7							b0
0011	1111	1	0	0	1	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0	

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

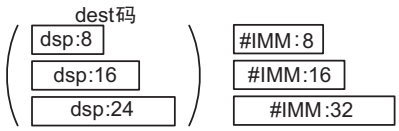
dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

AND AND

(1) AND.size:G #IMM(EX),dest

b7							b0	b7							b0
1	1	0	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

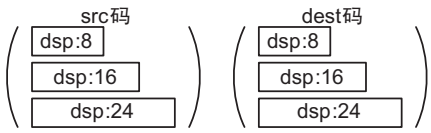
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

AND AND

(2) AND.size:G src,dest

b7							b0	b7							b0
1	1	0	0	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2
[An]	2 / 2	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3
dsp:8[ ]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:16[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:16	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3

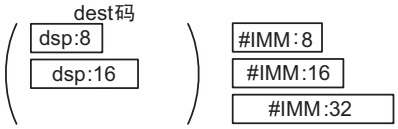
*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

AND

AND

(3) AND.size:S #IMM,dest

b7							b0
0	1	1	0	s1	s0	w1	w0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 dest 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	dsp:8[ ]	dsp:16
#IMM:8	2 / 1	3 / 2	4 / 2
#IMM:16	3 / 1	4 / 2	5 / 2
#IMM:32	5 / 1	6 / 2	7 / 2

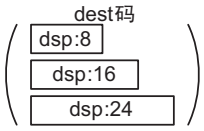
*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

BCLR

BCLR

(1) BCLR bit,dest

b7							b0	b7							b0
0	1	0	0	0	1	0	0	b2	b1	b0	g4	g3	g2	g1	g0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 b2 ～ b0 的 3 位指定位的位置，由 g4 ～ g0 位指定 dest 的操作数。

*3 运算长度固定为字节。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

BITINDEX BITINDEX

(1) BITINDEX.size src

b7 b0	b7						b0	b7							b0
0011 1111	1	1	0	1	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

*4 紧接在后的位指令的周期数加 2。

BM*Cnd* BM*Cnd*

(1) BM*Cnd* bit,dest

b7 b0	b7							b0	b7							b0
0111 1111	0	1	1	1	0	0	0	0	b2	b1	b0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

0	0	0	0	c3	c2	c1	c0

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 b2 ～ b0 的 3 位指定位的位置，由 g4 ～ g0 位指定 dest 的操作数。

*3 由 c3 ～ c0 位指定 *Cnd*，详细内容请参照 “4.2.3 条件的指定”。

*4 运算长度固定为字节。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 5	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5

*5 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

BNOT BNOT

(1) BNOT bit,dest

b7	b0	b7						b0	b7							b0
0111	1111	0	1	1	0	0	0	0	b2	b1	b0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 b2 ～ b0 的 3 位指定位的位置，由 g4 ～ g0 位指定 dest 的操作数。

*3 运算长度固定为字节。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

BRK BRK

(1) BRK

b7							b0
0	0	0	0	0	0	0	0

【字节数 / 周期数】

字节数 / 周期数	1 / 16

*1 这是可变向量时的周期数，在固定向量时为 19 个周期。

BRK2 BRK2

(1) BRK2

b7							b0
1	0	1	1	1	1	1	1

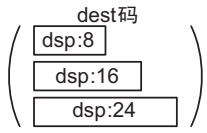
【字节数 / 周期数】

字节数 / 周期数	1 / 19

BSET BSET

(1) BSET bit,dest

b7							b0	b7							b0
0	1	0	0	1	1	0	0	b2	b1	b0	g4	g3	g2	g1	g0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 b2 ～ b0 的 3 位指定位的位置，由 g4 ～ g0 位指定 dest 的操作数。

*3 运算长度固定为字节。

【字节数 / 周期数】

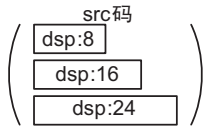
dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

BTST BTST

(1) BTST:G bit,src

b7							b0	b7							b0
0	1	0	0	1	0	0	0	b2	b1	b0	g4	g3	g2	g1	g0



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 b2 ～ b0 的 3 位指定位的位置，由 g4 ～ g0 位指定 src 的操作数。

*3 运算长度固定为字节。

【字节数 / 周期数】

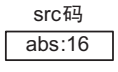
src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

BTST BTST

(2) BTST:S bit,abs:16

b7							b0
0	b2	b1	b0	0	1	1	1



*1 运算长度固定为字节。

*2 由 b2 ～ b0 的 3 位指定位的位置。

【字节数 / 周期数】

字节数 / 周期数	3 / 2

BTSTC

BTSTC

(1) BTSTC bit,dest

b7	b0	b7						b0	b7							b0
0111	1111	0	1	1	0	0	1	0	0	b2	b1	b0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 b2 ～ b0 的 3 位指定位的位置，由 g4 ～ g0 位指定 dest 的操作数。

*3 运算长度固定为字节。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

BTSTS

BTSTS

(1) BTSTS bit,dest

b7	b0	b7						b0	b7							b0
0111	1111	0	1	1	1	0	1	0	0	b2	b1	b0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 b2 ～ b0 的 3 位指定位的位置，由 g4 ～ g0 位指定 dest 的操作数。

*3 运算长度固定为字节。

【字节数 / 周期数】

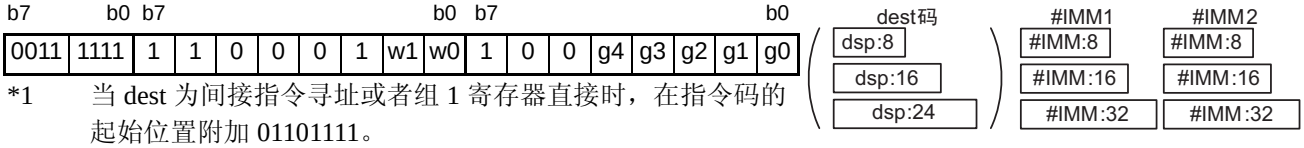
dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

CLIP

CLIP

(1) CLIP #IMM1,#IMM2,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5

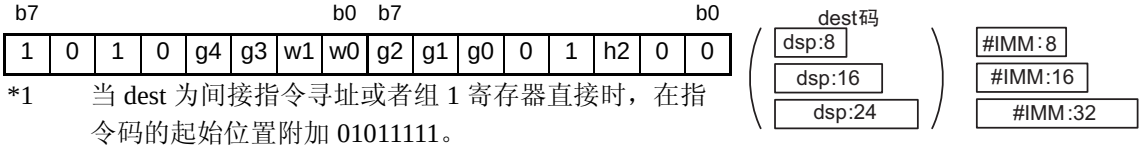
*3 当长度说明符 （.size）为 “.W” 时，表中的字节数加 2 ； 当长度说明符 （.size）为 “.L” 时，表中的字节数加 6。

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

CMP

CMP

(1) CMP.size:G #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

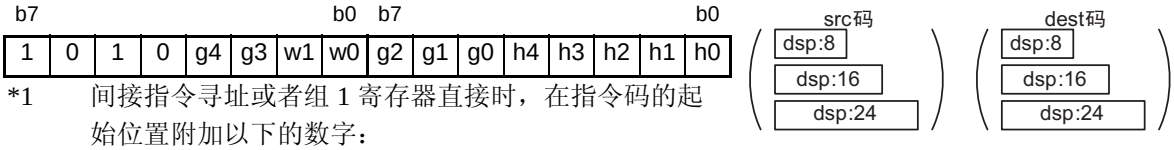
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

CMP CMP

(2) CMP.size:G src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

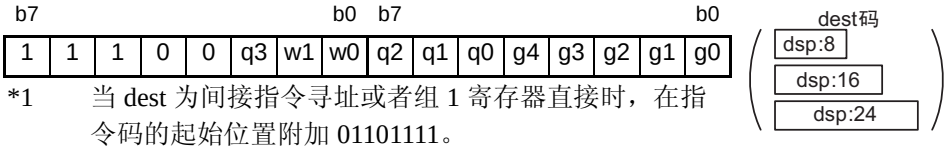
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2
[An]	2 / 2	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3
dsp:8[ ]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:16[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:16	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

CMP CMP

(3) CMP.size:Q #IMM:4,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 q3 ～ q0 的 4 位指定 #IMM:4，由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

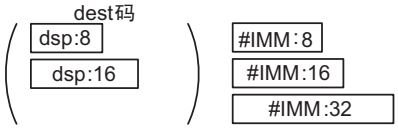
*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

CMP CMP

(4) CMP.size:S #IMM,dest

b7 b0

0	0	1	1	s1	s0	w1	w0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 dest 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	dsp:8[ ]	dsp:16
#IMM:8	2 / 1	3 / 2	4 / 2
#IMM:16	3 / 1	4 / 2	5 / 2
#IMM:32	5 / 1	6 / 2	7 / 2

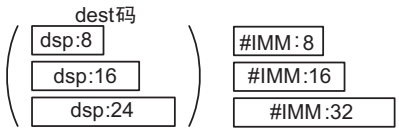
*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

CMPF CMPF

(1) CMPF #IMM(EX),dest

b7 b0 b7 b0 b7 b0

0111	1111	1	0	1	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
#IMMEX:16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
#IMM:32	7 / 3	7 / 4	8 / 4	9 / 4	10 / 4	9 / 4	10 / 4

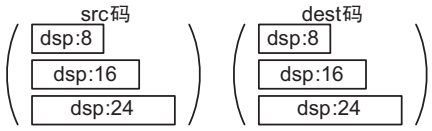
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

CMPF

CMPF

(2) CMPF src,dest

b7		b0	b7						b0	b7							b0
0111	1111	1	0	1	0	g4	g3	1	0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
[An]	3 / 4	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
dsp:8[ ]	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
dsp:16[ ]	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24[ ]	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5
dsp:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5

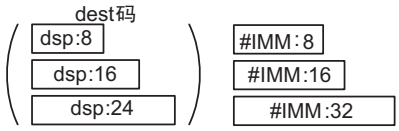
*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

CNVIF

CNVIF

(1) CNVIF #IMM(EX),dest

b7		b0	b7							b0	b7						b0
0111	1111	0	0	1	1	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
#IMMEX:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
#IMM:32	7 / 4	7 / 5	8 / 5	9 / 5	10 / 5	9 / 5	10 / 5

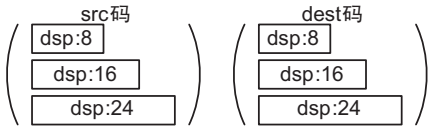
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

CNVIF

CNVIF

(2) CNVIF src,dest

b7	b0	b7							b0	b7							b0
0111	1111	0	0	1	1	g4	g3	1	0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 4	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
[An]	3 / 5	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
dsp:8[ ]	4 / 5	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
dsp:16[ ]	5 / 5	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24[ ]	6 / 5	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5
dsp:16	5 / 5	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24	6 / 5	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5

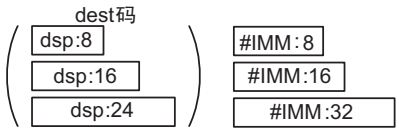
*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DADC

DADC

(1) DADC.size #IMM(EX),dest

b7	b0	b7								b0	b7							b0
0011	1111	1	0	0	1	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 6	4 / 7	5 / 7	6 / 7	7 / 7	6 / 7	7 / 7
#IMM(EX):16	5 / 6	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
#IMM:32	7 / 6	7 / 7	8 / 7	9 / 7	10 / 7	9 / 7	10 / 7

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

DADC DADC

(2) DADC.size src,dest

b7		b0	b7							b0	b7							b0
0011	1111	1	0	0	1	g4	g3	w1	w0		g2	g1	g0	h4	h3	h2	h1	h0

src码: dsp:8 / dsp:16 / dsp:24

dest码: dsp:8 / dsp:16 / dsp:24

*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111

当dest为间接指令寻址或者组1寄存器直接时：01011111

当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 6	3 / 6	4 / 6	5 / 6	6 / 6	5 / 6	6 / 6
[An]	3 / 7	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7
dsp:8[ ]	4 / 7	4 / 7	5 / 7	6 / 7	7 / 7	6 / 7	7 / 7
dsp:16[ ]	5 / 7	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
dsp:24[ ]	6 / 7	6 / 7	7 / 7	8 / 7	9 / 7	8 / 7	9 / 7
dsp:16	5 / 7	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
dsp:24	6 / 7	6 / 7	7 / 7	8 / 7	9 / 7	8 / 7	9 / 7

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DADD DADD

(1) DADD.size #IMM(EX),dest

b7		b0	b7							b0	b7							b0
0011	1111	1	0	0	0	g4	g3	w1	w0		g2	g1	g0	0	1	h2	0	0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8 / #IMM:16 / #IMM:32

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 6	4 / 7	5 / 7	6 / 7	7 / 7	6 / 7	7 / 7
#IMM(EX):16	5 / 6	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
#IMM:32	7 / 6	7 / 7	8 / 7	9 / 7	10 / 7	9 / 7	10 / 7

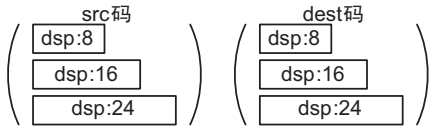
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

DADD

DADD

(2) DADD.size src,dest

b7		b0	b7							b0	b7							b0
0011	1111	1	0	0	0	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0	



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 6	3 / 6	4 / 6	5 / 6	6 / 6	5 / 6	6 / 6
[An]	3 / 7	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7
dsp:8[ ]	4 / 7	4 / 7	5 / 7	6 / 7	7 / 7	6 / 7	7 / 7
dsp:16[ ]	5 / 7	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
dsp:24[ ]	6 / 7	6 / 7	7 / 7	8 / 7	9 / 7	8 / 7	9 / 7
dsp:16	5 / 7	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
dsp:24	6 / 7	6 / 7	7 / 7	8 / 7	9 / 7	8 / 7	9 / 7

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DEC

DEC

(1) DEC.size dest

b7		b0	b7							b0	b7							b0
0011	1111	1	0	0	1	0	1	w1	w0	0	0	0	g4	g3	g2	g1	g0	

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

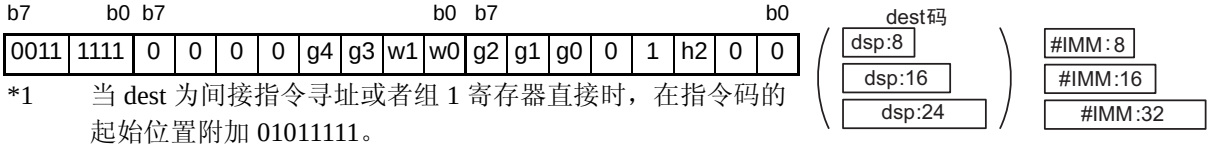
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

DIV DIV

(1) DIV.size #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

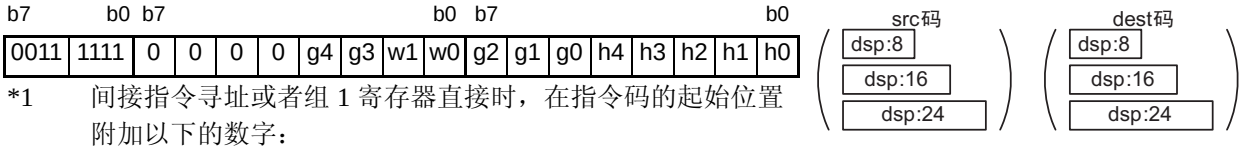
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 10	4 / 11	5 / 11	6 / 11	7 / 11	6 / 11	7 / 11
#IMM(EX):16	5 / 10	5 / 11	6 / 11	7 / 11	8 / 11	7 / 11	8 / 11
#IMM:32	7 / 10	7 / 11	8 / 11	9 / 11	10 / 11	9 / 11	10 / 11

*4 当长度说明符 （.size） 为 “.W” 时，表中的周期数加 3 ； 当长度说明符 （.size） 为 “.L” 时，表中的周期数加 10。

*5 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

DIV DIV

(2) DIV.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 10	3 / 11	4 / 11	5 / 11	6 / 11	5 / 11	6 / 11
[An]	3 / 11	3 / 12	4 / 12	5 / 12	6 / 12	5 / 12	6 / 12
dsp:8[ ]	4 / 11	4 / 12	5 / 12	6 / 12	7 / 12	6 / 12	7 / 12
dsp:16[ ]	5 / 11	5 / 12	6 / 12	7 / 12	8 / 12	7 / 12	8 / 12
dsp:24[ ]	6 / 11	6 / 12	7 / 12	8 / 12	9 / 12	8 / 12	9 / 12
dsp:16	5 / 11	5 / 12	6 / 12	7 / 12	8 / 12	7 / 12	8 / 12
dsp:24	6 / 11	6 / 12	7 / 12	8 / 12	9 / 12	8 / 12	9 / 12

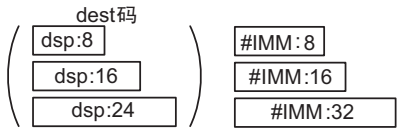
*3 当长度说明符 （.size） 为 “.W” 时，表中的周期数加 3 ； 当长度说明符 （.size） 为 “.L” 时，表中的周期数加 10。

*4 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DIVF DIVF

(1) DIVF #IMM(EX),dest

b7		b0	b7							b0	b7						b0
0111	1111	1	1	0	1	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

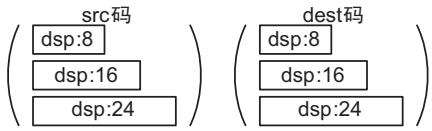
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 16	4 / 17	5 / 17	6 / 17	7 / 17	6 / 17	7 / 17
#IMMEX:16	5 / 16	5 / 17	6 / 17	7 / 17	8 / 17	7 / 17	8 / 17
#IMM:32	7 / 16	7 / 17	8 / 17	9 / 17	10 / 17	9 / 17	10 / 17

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

DIVF DIVF

(2) DIVF src,dest

b7		b0	b7							b0	b7						b0
0111	1111	1	1	0	1	g4	g3	1	0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

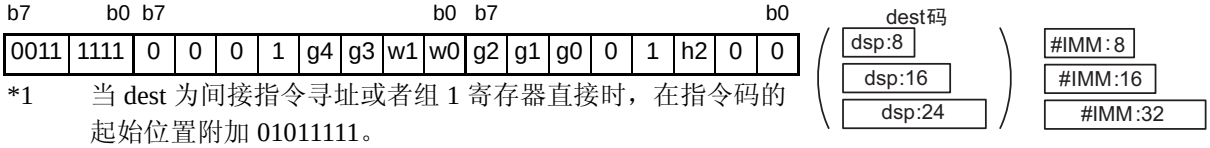
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 16	3 / 17	4 / 17	5 / 17	6 / 17	5 / 17	6 / 17
[An]	3 / 17	3 / 18	4 / 18	5 / 18	6 / 18	5 / 18	6 / 18
dsp:8[ ]	4 / 17	4 / 18	5 / 18	6 / 18	7 / 18	6 / 18	7 / 18
dsp:16[ ]	5 / 17	5 / 18	6 / 18	7 / 18	8 / 18	7 / 18	8 / 18
dsp:24[ ]	6 / 17	6 / 18	7 / 18	8 / 18	9 / 18	8 / 18	9 / 18
dsp:16	5 / 17	5 / 18	6 / 18	7 / 18	8 / 18	7 / 18	8 / 18
dsp:24	6 / 17	6 / 18	7 / 18	8 / 18	9 / 18	8 / 18	9 / 18

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DIVU DIVU

(1) DIVU.size #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

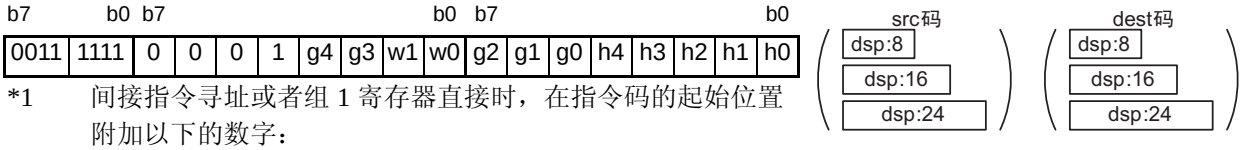
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 10	4 / 11	5 / 11	6 / 11	7 / 11	6 / 11	7 / 11
#IMM(EX):16	5 / 10	5 / 11	6 / 11	7 / 11	8 / 11	7 / 11	8 / 11
#IMM:32	7 / 10	7 / 11	8 / 11	9 / 11	10 / 11	9 / 11	10 / 11

*4 当长度说明符 （.size） 为 “.W” 时，表中的周期数加 3 ； 当长度说明符 （.size） 为 “.L” 时，表中的周期数加 10。

*5 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

DIVU DIVU

(2) DIVU.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

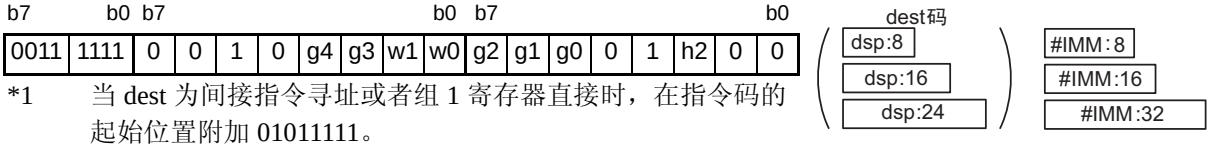
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 10	3 / 11	4 / 11	5 / 11	6 / 11	5 / 11	6 / 11
[An]	3 / 11	3 / 12	4 / 12	5 / 12	6 / 12	5 / 12	6 / 12
dsp:8[ ]	4 / 11	4 / 12	5 / 12	6 / 12	7 / 12	6 / 12	7 / 12
dsp:16[ ]	5 / 11	5 / 12	6 / 12	7 / 12	8 / 12	7 / 12	8 / 12
dsp:24[ ]	6 / 11	6 / 12	7 / 12	8 / 12	9 / 12	8 / 12	9 / 12
dsp:16	5 / 11	5 / 12	6 / 12	7 / 12	8 / 12	7 / 12	8 / 12
dsp:24	6 / 11	6 / 12	7 / 12	8 / 12	9 / 12	8 / 12	9 / 12

*3 当长度说明符 （.size） 为 “.W” 时，表中的周期数加 3 ； 当长度说明符 （.size） 为 “.L” 时，表中的周期数加 10。

*4 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DIVX DIVX

(1) DIVX.size #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

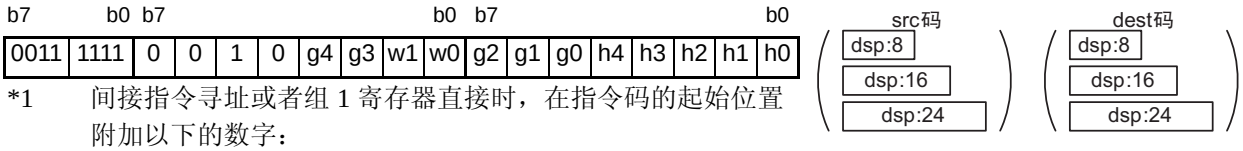
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 10	4 / 11	5 / 11	6 / 11	7 / 11	6 / 11	7 / 11
#IMM(EX):16	5 / 10	5 / 11	6 / 11	7 / 11	8 / 11	7 / 11	8 / 11
#IMM:32	7 / 10	7 / 11	8 / 11	9 / 11	10 / 11	9 / 11	10 / 11

*4 当长度说明符 （.size） 为 “.W” 时，表中的周期数加 3 ； 当长度说明符 （.size） 为 “.L” 时，表中的周期数加 10。

*5 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

DIVX DIVX

(2) DIVX.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

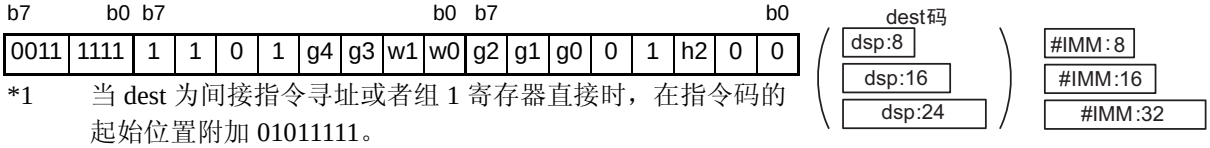
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 10	3 / 11	4 / 11	5 / 11	6 / 11	5 / 11	6 / 11
[An]	3 / 11	3 / 12	4 / 12	5 / 12	6 / 12	5 / 12	6 / 12
dsp:8[ ]	4 / 11	4 / 12	5 / 12	6 / 12	7 / 12	6 / 12	7 / 12
dsp:16[ ]	5 / 11	5 / 12	6 / 12	7 / 12	8 / 12	7 / 12	8 / 12
dsp:24[ ]	6 / 11	6 / 12	7 / 12	8 / 12	9 / 12	8 / 12	9 / 12
dsp:16	5 / 11	5 / 12	6 / 12	7 / 12	8 / 12	7 / 12	8 / 12
dsp:24	6 / 11	6 / 12	7 / 12	8 / 12	9 / 12	8 / 12	9 / 12

*3 当长度说明符 （.size） 为 “.W” 时，表中的周期数加 3 ； 当长度说明符 （.size） 为 “.L” 时，表中的周期数加 10。

*4 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DSBB DSBB

(1) DSBB.size #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

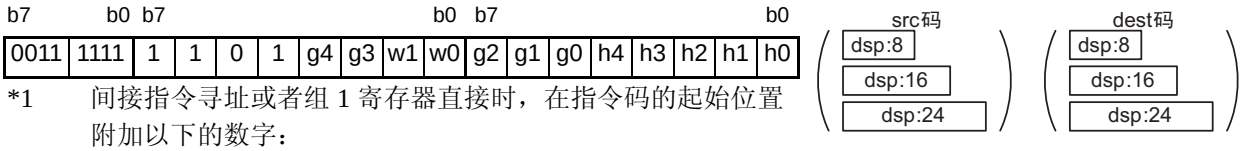
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
#IMM(EX):16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
#IMM:32	7 / 3	7 / 4	8 / 4	9 / 4	10 / 4	9 / 4	10 / 4

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

DSBB DSBB

(2) DSBB.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

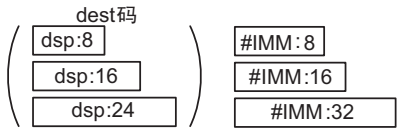
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
[An]	3 / 4	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
dsp:8[ ]	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
dsp:16[ ]	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24[ ]	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5
dsp:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

DSUB DSUB

(1) DSUB.size #IMM(EX),dest

b7		b0	b7						b0	b7							b0
0011	1111	1	1	0	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

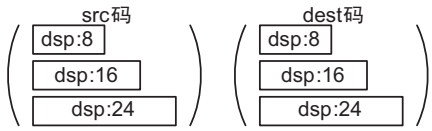
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
#IMM(EX):16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
#IMM:32	7 / 3	7 / 4	8 / 4	9 / 4	10 / 4	9 / 4	10 / 4

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

DSUB DSUB

(2) DSUB.size src,dest

b7		b0	b7						b0	b7							b0
0011	1111	1	1	0	0	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
[An]	3 / 4	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
dsp:8[ ]	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
dsp:16[ ]	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24[ ]	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5
dsp:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5

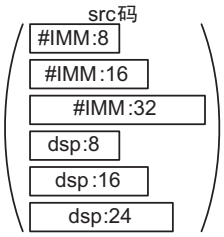
*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

EDIV

EDIV

(1) EDIV.size src,dest

b7		b0	b7						b0	b7							b0
0011	1111		0	1	1	0	0	1	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为组 1 寄存器直接时：01011111

当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的寄存器。

【字节数 / 周期数】

src	寄存器	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 16	4 / 16	5 / 16	7 / 16	3 / 16	4 / 16	5 / 16	6 / 16	5 / 16	6 / 16

*3 当长度说明符 （.size）为 “.W” 时，表中的周期数加 3 ；当长度说明符 （.size）为 “.L” 时，表中的周期数加 10。

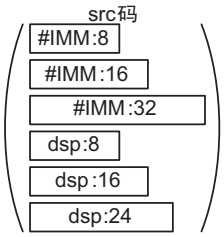
*4 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址时，表中的周期数加 2。

EDIVU

EDIVU

(1) EDIVU.size src,dest

b7		b0	b7						b0	b7							b0
0011	1111		0	1	1	1	0	1	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为组 1 寄存器直接时：01011111

当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的寄存器。

【字节数 / 周期数】

src	寄存器	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 14	4 / 14	5 / 14	7 / 14	3 / 14	4 / 14	5 / 14	6 / 14	5 / 14	6 / 14

*3 当长度说明符 （.size）为 “.W” 时，表中的周期数加 3 ；当长度说明符 （.size）为 “.L” 时，表中的周期数加 10。

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址时，表中的周期数加 2。

EDIVX

EDIVX

(1) EDIVX.size src,dest

b7		b0	b7						b0	b7							b0
0011	1111	0	1	1	0	1	1	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0

src码: #IMM:8 / #IMM:16 / #IMM:32 / dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为组 1 寄存器直接时：01011111

当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的寄存器。

【字节数 / 周期数】

src	寄存器	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 16	4 / 16	5 / 16	7 / 16	3 / 16	4 / 16	5 / 16	6 / 16	5 / 16	6 / 16

*3 当长度说明符 （.size）为 “.W” 时，表中的周期数加 3 ；当长度说明符 （.size）为 “.L” 时，表中的周期数加 10。

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址时，表中的周期数加 2。

EMUL

EMUL

(1) EMUL.size src,dest

b7		b0	b7						b0	b7							b0
0011	1111	0	1	1	0	0	0	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0

src码: #IMM:8 / #IMM:16 / #IMM:32 / dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为组 1 寄存器直接时：01011111

当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的寄存器。

【字节数 / 周期数】

src	寄存器	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 2	4 / 2	5 / 2	7 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*3 这是长度说明符 （.size）为 “.B” 或者 “.W” 时的周期数。当长度说明符 （.size）为 “.L” 时，周期数加 1。

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时，表中的字节数加 1 ；当 src 为间接指令寻址时，表中的周期数加 2。

EMULU

EMULU

(1) EMUL.size src,dest

b7		b0	b7						b0	b7							b0
0011	1111	0	1	1	1	0	0	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0

src码
- #IMM:8
- #IMM:16
- #IMM:32
- dsp:8
- dsp:16
- dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为组 1 寄存器直接时：01011111

当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的寄存器。

【字节数 / 周期数】

src	寄存器	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 2	4 / 2	5 / 2	7 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*3 这是长度说明符 （.size）为 “.B” 或者 “.W” 时的周期数。当长度说明符 （.size）为 “.L” 时，周期数加 1。

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

ENTER

ENTER

(1) ENTER #IMM:8

b7							b0
0	0	0	1	1	0	1	1

- #IMM:8

【字节数 / 周期数】

字节数 / 周期数	2 / 3

ENTER

ENTER

(2) ENTER #IMM:16

b7							b0
1	1	0	1	1	0	1	1

- #IMM:16

【字节数 / 周期数】

字节数 / 周期数	3 / 3

EXITD

EXITD

(1) EXITD

b7							b0
1	1	1	0	1	1	1	1

【字节数 / 周期数】

字节数 / 周期数	1 / 7

EXITI

EXITI

(1) EXITI

b7							b0	b7							b0
1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 8

EXTS

EXTS

(1) EXTS.size #IMM,dest

b7		b0	b7							b0	b7							b0
0011	1111		1	0	1	1	g4	g3	w1	w0	g2	g1	g0	0	1	1	0	0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8 / #IMM:16

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 w1 位和 w0 位指定 src 和 dest 的操作数长度。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

【字节数 / 周期数】

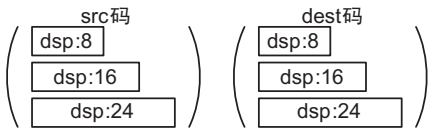
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM:8	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:16	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

EXTS EXTS

(2) EXTS.size src,dest

b7	b0	b7							b0	b7							b0
0011	1111	1	0	1	1	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

*3 由 w1 位和 w0 位指定 src 和 dest 的操作数长度。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

【字节数 / 周期数】

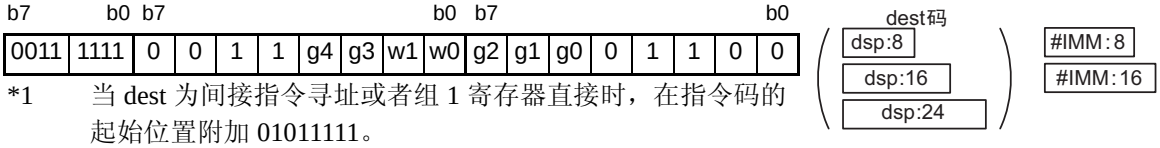
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 1	3 / 1	4 / 1	5 / 1	6 / 1	5 / 1	6 / 1
[An]	3 / 2	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
dsp:8[ ]	4 / 2	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
dsp:16[ ]	5 / 2	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
dsp:24[ ]	6 / 2	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2
dsp:16	5 / 2	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
dsp:24	6 / 2	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

EXTZ

EXTZ

(1) EXTZ.size:G #IMM,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 w1 位和 w0 位指定 src 和 dest 的操作数长度。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM:8	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:16	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2

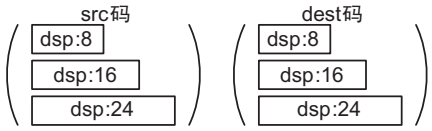
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

EXTZ

EXTZ

(2) EXTZ.size:G src,dest

b7		b0	b7							b0	b7							b0
0011	1111		0	0	1	1	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当src为间接指令寻址或者组1寄存器直接时：01101111
当dest为间接指令寻址或者组1寄存器直接时：01011111
当src和dest都为间接指令寻址或者组1寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

*3 由 w1 位和 w0 位指定 src 和 dest 的操作数长度。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 1	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
[An]	3 / 3	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:8[ ]	4 / 3	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:16[ ]	5 / 3	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24[ ]	6 / 3	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3
dsp:16	5 / 3	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24	6 / 3	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3

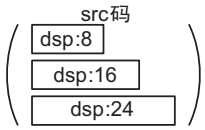
*4 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

EXTZ

EXTZ

(3) EXTZ.WL:S src,An

b7							b0	b7							b0
0	1	0	0	0	1	0	1	q2	q1	q0	g4	g3	g2	g1	g0



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

FCLR FCLR

(1) FCLR dest

b7							b0	b7							b0
0	1	0	0	1	1	0	0	q2	q1	q0	0	1	0	0	0

*1 由 q2 ～ q0 的 3 位指定 dest 的标志，详细内容请参照 “4.2.2 操作数的指定”的 “(9) FLG 直接”。

【字节数 / 周期数】

字节数 / 周期数	2 / 3

FREIT FREIT

(1) FREIT

b7							b0
1	0	0	0	1	1	1	1

【字节数 / 周期数】

字节数 / 周期数	1 / 4

FSET FSET

(1) FSET dest

b7							b0	b7							b0
0	1	0	0	1	1	0	1	q2	q1	q0	0	1	0	0	0

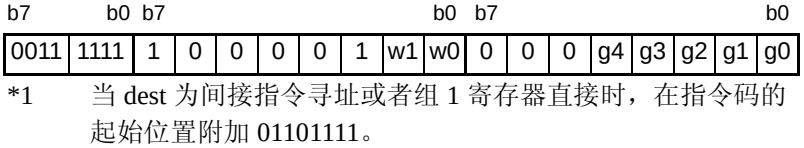
*1 由 q2 ～ q0 的 3 位指定 dest 的标志，详细内容请参照 “4.2.2 操作数的指定”的 “(9) FLG 直接”。

【字节数 / 周期数】

字节数 / 周期数	2 / 3

INC INC

(1) INC.size dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

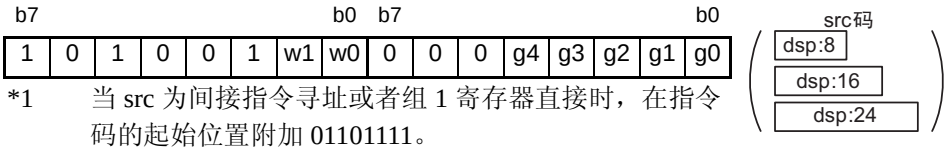
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

INDEX1 INDEX1

(1) INDEX1.size src



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 紧接在后的指令的周期数加 2。

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

INDEX2 INDEX2

(1) INDEX2.size src

b7							b0	b7							b0
1	0	0	1	0	1	w1	w0	0	0	0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 紧接在后的指令的周期数加 2。

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

INDEXB INDEXB

(1) INDEXB.size src

b7							b0	b7							b0
1	0	0	0	0	1	w1	w0	0	0	0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 紧接在后的指令的周期数加 4。当紧接在后的指令为 EXTS 指令或者 EXTZ 指令时，周期数加 5。

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

INT

INT

(1) INT #IMM:8

b7							b0
1	1	0	1	1	0	1	1

#IMM:8

【字节数 / 周期数】

字节数 / 周期数	2 / 11

INTO

INTO

(1) INTO

b7							b0
1	0	1	0	1	1	1	1

【字节数 / 周期数】

字节数 / 周期数	1 / 1(12)*1

*1 当 O 标志为 “1” 时，为 12 个时钟。

JCnd

JCnd

(1) JCnd label

b7							b0
c3	c2	c1	c0	0	0	1	1

label码

dsp:8

*1 由 c3 ～ c0 位指定 *Cnd*，详细内容请参照 “4.2.3 条件的指定”。

【字节数 / 周期数】

字节数 / 周期数	2 / 1(3)*2

*2 当转移到 label 时，周期数为 3。

JMP

JMP

(1) JMP.S label

b7							b0
1	q2	q1	q0	0	1	1	1

*1 由 q2 ～ q0 位指定 dsp3，详细内容请参照 “4.2.2 操作数的指定”的 “(7) 程序计数器相对・短格式”。

【字节数 / 周期数】

字节数 / 周期数	1 / 1

JMP

JMP

(2) JMP.B label

b7							b0
0	1	1	1	0	0	1	1

label码
dsp:8

【字节数 / 周期数】

字节数 / 周期数	2 / 3

JMP

JMP

(3) JMP.W label

b7							b0
1	0	1	0	1	0	1	1

label码
dsp:16

【字节数 / 周期数】

字节数 / 周期数	3 / 3

JMP

JMP

(4) JMP.A label

b7							b0
1	1	1	0	1	0	1	1

label码
dsp :24

【字节数 / 周期数】

字节数 / 周期数	4 / 3

JMPI

JMPI

(1) JMPI.W src

b7	b0	b7						b0	b7						b0
0111	1111	1	0	1	0	0	1	0	1	0	0	0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

JMPI

JMPI

(2) JMPI.L src

b7	b0	b7						b0	b7						b0
0111	1111	1	0	1	0	0	1	1	0	0	0	0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

JSR

JSR

(1) JSR.W label

b7							b0
1	0	1	1	1	0	1	1

label码: dsp:16

【字节数 / 周期数】

字节数 / 周期数	3 / 3

JSR JSR

(2) JSR.A label

b7							b0
1	1	1	1	1	0	1	1

label码
dsp:24

【字节数 / 周期数】

字节数 / 周期数	4 / 3

JSRI JSRI

(1) JSRI.W src

b7		b0	b7						b0	b7							b0
0111	1111	1	0	1	0	0	1	0	1	1	0	0	g4	g3	g2	g1	g0

src码
dsp:8
dsp:16
dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

JSRI JSRI

(2) JSRI.L src

b7		b0	b7						b0	b7							b0
0111	1111	1	0	1	0	0	1	1	0	1	0	0	g4	g3	g2	g1	g0

src码
dsp:8
dsp:16
dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

LDC

LDC

(1) LDC src,dest

b7	b0	b7							b0	b7							b0
0011	1111	1	1	1	0	1	0	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0

src码
- #IMM:8
- #IMM:16
- #IMM:32
- dsp:8
- dsp:16
- dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

*3 由 q2 ～ q0 位指定 dest 的专用寄存器，详细内容请参照“4.2.2 操作数的指定”的“(5) 专用寄存器直接 （CPU）”。

【字节数 / 周期数】

src	寄存器	#IMMEX:8	#IMMEX:16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 3	4 / 3	5 / 3	7 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

LDC

LDC

(2) LDC #IMM,dest

b7	b0	b7							b0	b7							b0
0011	1111	1	0	0	1	0	1	w1	w0	0	0	0	0	1	0	0	0

#IMM:8

- #IMM:8
- #IMM:16
- #IMM:32

*1 由 #IMM:8 指定 dest 的寄存器，详细内容请参照“4.2.2 操作数的指定”的“(6) 专用寄存器直接 （DMAC 和 VCT）”。

【字节数 / 周期数】

src	#IMM:8	#IMM:16	#IMM:32
字节数 / 周期数	5 / 3	6 / 3	8 / 3

LDC

LDC

(3) LDC src,dest

b7		b0	b7						b0	b7							b0
0011	1111	1	0	1	0	0	1	1	0	0	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

*3 由 #IMM:8 指定 dest 的寄存器，详细内容请参照 “4.2.2 操作数的指定”的 “(6) 专用寄存器直接（DMAC 和 VCT）”。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 3	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

LDCTX

LDCTX

(1) LDCTX abs:16,dsp:24

b7		b0	b7						b0	b7							b0
0011	1111	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0

abs:16 dsp:24

【字节数 / 周期数】

字节数 / 周期数	8 / 10+m*1

*1 m 为要传送的寄存器的个数。

LDIPL

LDIPL

(1) LDIPL #IMM:3

b7							b0	b7							b0
0	1	0	0	0	1	0	1	q2	q1	q0	0	1	0	0	0

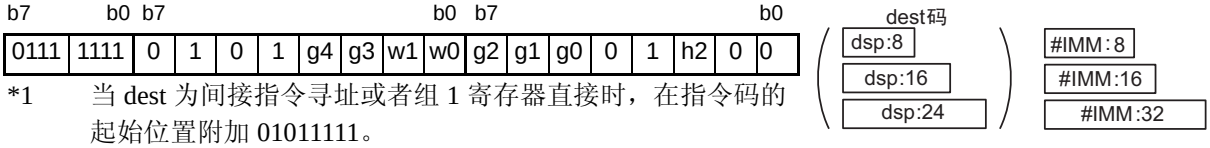
*1 由 q2 ～ q0 的 3 位指定 #IMM:3。

【字节数 / 周期数】

字节数 / 周期数	2 / 2

MAX MAX

(1) MAX.size #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

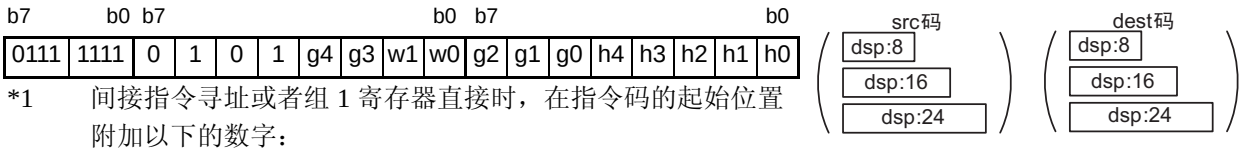
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
#IMM(EX):16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
#IMM:32	7 / 2	7 / 3	8 / 3	9 / 3	10 / 3	9 / 3	10 / 3

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

MAX MAX

(2) MAX.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
[An]	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
dsp:8[ ]	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
dsp:16[ ]	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24[ ]	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4
dsp:16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4

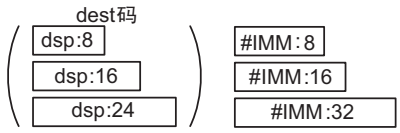
*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

MIN MIN

(1) MIN.size #IMM(EX),dest

b7 b0 b7 b0 b7 b0

0111	1111	0	1	0	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
#IMM(EX):16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
#IMM:32	7 / 2	7 / 3	8 / 3	9 / 3	10 / 3	9 / 3	10 / 3

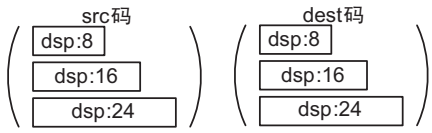
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

MIN MIN

(2) MIN.size src,dest

b7 b0 b7 b0 b7 b0

0111	1111	0	1	0	0	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

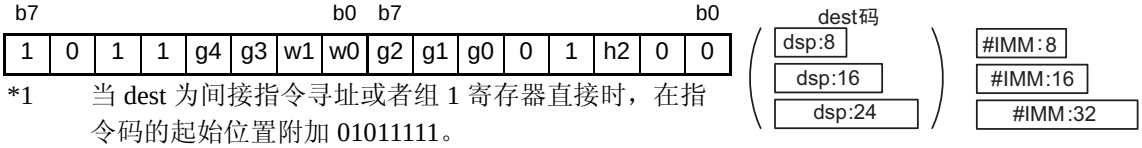
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
[An]	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
dsp:8[ ]	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
dsp:16[ ]	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24[ ]	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4
dsp:16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

MOV MOV

(1) MOV.size:G #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

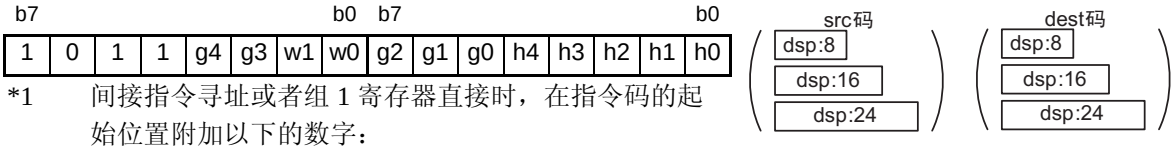
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

MOV MOV

(2) MOV.size:G src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

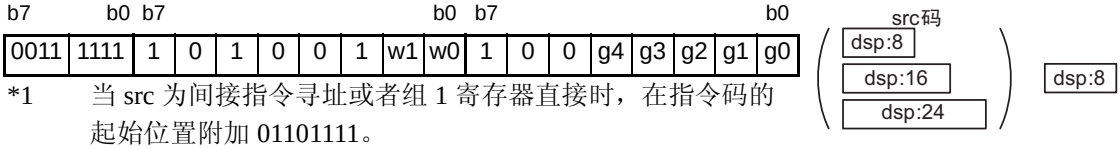
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	2 / 1	2 / 1	3 / 1	4 / 1	5 / 1	4 / 1	5 / 1
[An]	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2
dsp:8[ ]	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
dsp:16[ ]	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
dsp:24[ ]	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
dsp:16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
dsp:24	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

MOV MOV

(3) MOV.size:G src,dsp:8[SP]



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

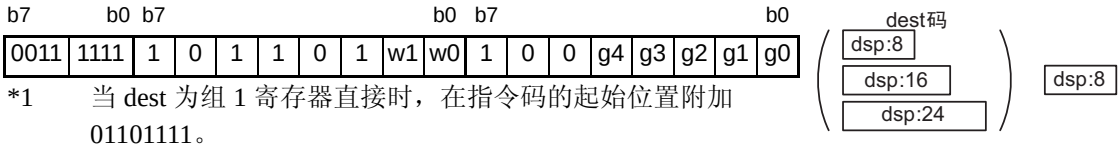
【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

MOV MOV

(4) MOV.size:G dsp:8[SP],dest



*1 当 dest 为组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

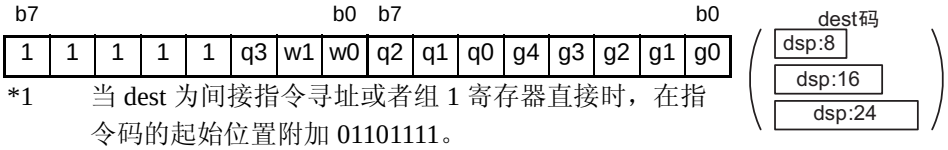
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 4	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4

*3 当 dest 为组 1 寄存器直接时，表中的字节数加 1。

MOV MOV

(5) MOV.size:Q #IMM:4,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 q3 ～ q0 的 4 位指定 #IMM:4，由和 g4 ～ g0 位指定 dest 的操作数。

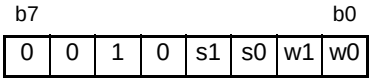
【字节数 / 周期数】

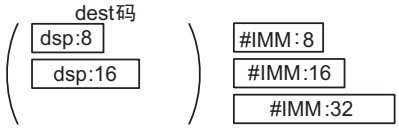
dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 1	3 / 1	4 / 1	5 / 1	4 / 1	5 / 1

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

MOV MOV

(6) MOV.size:S #IMM,dest





*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 dest 的操作数。

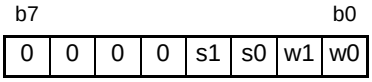
【字节数 / 周期数】

src \ dest	寄存器	dsp:8[ ]	dsp:16
#IMM:8	2 / 1	3 / 2	4 / 2
#IMM:16	3 / 1	4 / 2	5 / 2
#IMM:32	5 / 1	6 / 2	7 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

MOV MOV

(7) MOV.size:S R0L/R0/R2R0,dest



*1 当 dest 为间接指令寻址时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 dest 的操作数。

【字节数 / 周期数】

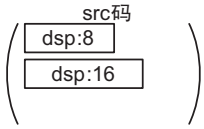
dest	dsp:8[ ]	dsp:16
字节数 / 周期数	2 / 1	3 / 1

*3 当 dest 为间接指令寻址时，表中的字节数加 1，周期数加 2。

MOV MOV

(8) MOV.size:S src,R0L/R0/R2R0

b7							b0
0	1	0	1	s1	s0	w1	w0



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	dsp:8[ ]	dsp:16
字节数 / 周期数	1 / 1	2 / 1	3 / 1

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

MOV MOV

(9) MOV.L:S src,A0

b7							b0
0	1	0	0	s1	s0	1	0

*1 当 src 为间接指令寻址时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 src 的操作数。

【字节数 / 周期数】

src	dsp:8[ ]	dsp:16
字节数 / 周期数	2 / 1	3 / 1

*3 当 src 为间接指令寻址时，表中的字节数加 1，周期数加 2。

MOV MOV

(10) MOV.size:Z #0,dest

b7							b0
0	0	0	1	s1	s0	w1	w0

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 s1 ～ s0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	dsp:8[ ]	dsp:16
字节数 / 周期数	1 / 1	2 / 1	3 / 1

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

MOVA

MOVA

(1) MOVA src,dest

b7							b0	b7							b0
0	1	0	0	1	0	0	1	q2	q1	q0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的寄存器。

*3 运算长度固定为长字。

【字节数 / 周期数】

src	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	3 / 1	4 / 1	5 / 1	4 / 1	5 / 1

*4 当 dest 为组 1 寄存器直接时，表中的字节数加 1。

MOV*Dir*

MOV*Dir*

(1) MOV*Dir* src,R0L

b7		b0	b7						b0	b7							b0
0111	1111	1	d1	0	0	0	1	0	0	d0	0	0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

*3 由 d1~d0 位指定操作 （*Dir*）。

d1	d0	*Dir*
0	0	LL
0	1	LH
1	0	HL
1	1	HH

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*4 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

MOV*Dir* MOV*Dir*

(2) MOV*Dir* R0L,dest

b7		b0	b7						b0	b7							b0
0111	1111	1	d1	0	1	0	1	0	0	d0	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 d1 ～ d0 位指定操作 （*Dir*）。

d1	d0	*Dir*
0	0	LL
0	1	LH
1	0	HL
1	1	HH

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 3	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

MUL MUL

(1) MUL.size #IMM(EX),dest

b7		b0	b7						b0	b7							b0
0011	1111	0	1	0	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM: #IMM:8 / #IMM:16 / #IMM:32

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

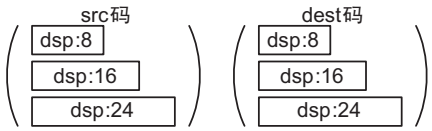
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
#IMM(EX):16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
#IMM:32	7 / 2	7 / 3	8 / 3	9 / 3	10 / 3	9 / 3	10 / 3

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

MUL MUL

(2) MUL.size src,dest

b7								b0 b7								b0
0011	1111	0	1	0	0	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

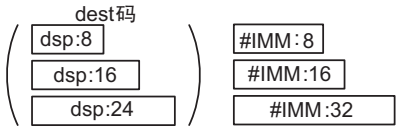
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
[An]	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
dsp:8[ ]	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
dsp:16[ ]	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24[ ]	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4
dsp:16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

MULF MULF

(1) MULF #IMM(EX),dest

b7								b0 b7								b0
0111	1111	1	1	0	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
#IMMEX:16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
#IMM:32	7 / 3	7 / 4	8 / 4	9 / 4	10 / 4	9 / 4	10 / 4

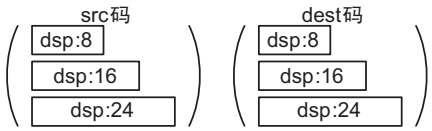
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

MULF

MULF

(2) MULF src,dest

b7		b0	b7							b0	b7							b0
0111	1111		1	1	0	0	g4	g3	1	0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
[An]	3 / 4	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
dsp:8[ ]	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
dsp:16[ ]	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24[ ]	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5
dsp:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24	6 / 4	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5

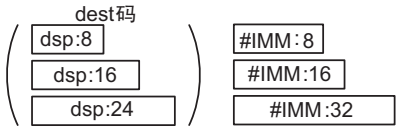
*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

MULU

MULU

(1) MULU.size #IMM(EX),dest

b7		b0	b7							b0	b7							b0
0011	1111		0	1	0	1	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

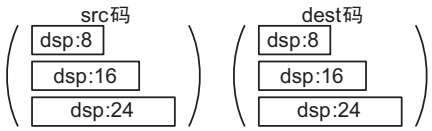
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
#IMM(EX):16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
#IMM:32	7 / 2	7 / 3	8 / 3	9 / 3	10 / 3	9 / 3	10 / 3

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

MULU MULU

(2) MULU.size src,dest

b7		b0	b7							b0	b7							b0
0011	1111		0	1	0	1	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

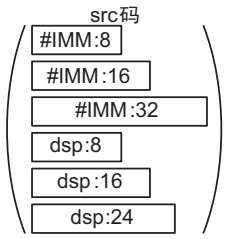
src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
[An]	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
dsp:8[ ]	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
dsp:16[ ]	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24[ ]	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4
dsp:16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24	6 / 3	6 / 4	7 / 4	8 / 4	9 / 4	8 / 4	9 / 4

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

MULX MULX

(1) MULX.size src,dest

b7		b0	b7							b0	b7							b0
0111	1111		0	1	1	0	1	0	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0



*1 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为组 1 寄存器直接时：01011111

当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 src 的操作数，由 q2 ～ q0 位指定 dest 的寄存器。

【字节数 / 周期数】

src	寄存器	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 3	4 / 3	5 / 3	7 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，或者当 dest 为组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址或者组 1 寄存器直接并且 dest 为组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

NEG NEG

(1) NEG.size dest

b7							b0	b7							b0
1	0	0	1	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

NOP NOP

(1) NOP

b7							b0
1	0	0	1	1	1	1	1

【字节数 / 周期数】

字节数 / 周期数	1 / 1

NOT NOT

(1) NOT.size dest

b7							b0	b7							b0
1	0	1	0	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

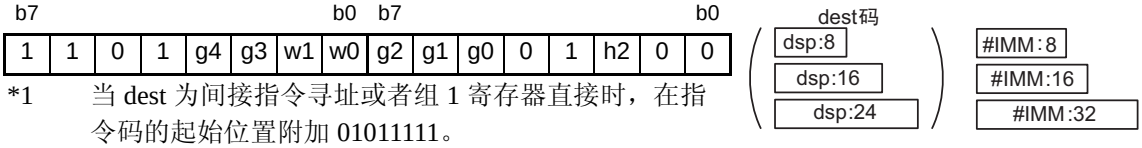
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

OR OR

(1) OR.size #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

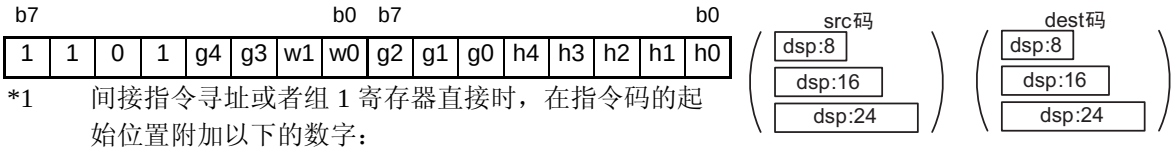
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

OR OR

(2) OR.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2
[An]	2 / 2	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3
dsp:8[ ]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:16[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:16	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

POP

POP

(1) POP.size dest

b7							b0	b7							b0
1	0	1	1	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 3	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3

*3 当 dest 为组 1 寄存器直接时，表中的字节数加 1。

POPC

POPC

(1) POPC dest

b7							b0	b7							b0
0	1	0	0	1	0	0	1	q2	q1	q0	0	1	0	0	0

*1 由 q2 ～ q0 位指定 dest 的专用寄存器。

*2 运算长度固定为长字。

【字节数 / 周期数】

字节数 / 周期数	2 / 4

POPM

POPM

(1) POPM dest

b7							b0
0	1	1	SB*1	1	0	1	1

#IMM:8

*1 在恢复 SB 寄存器时为 “1”，否则为 “0”。

*2 由 #IMM:8 指定其他的寄存器。

bit	7	6	5	4	3	2	1	0
寄存器	A3	A2	A1	A0	R7R5	R6R4	R3R1	R2R0

*3 当 dest 为组 1 寄存器直接时，在指令码的起始位置附加 01101111。

【字节数 / 周期数】

字节数 / 周期数	2 / n*4

*4 n 为要恢复的寄存器的个数。

*5 当 dest 为组 1 寄存器直接时，表中的字节数加 1。

PUSH PUSH

(1) PUSH.size:G src

b7							b0	b7							b0
1	1	0	1	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0

src码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 当 src 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为间接指令寻址时，表中的周期数加 2。

PUSH PUSH

(2) PUSH.B:S #IMM:8

b7							b0
0	0	1	0	1	0	1	1

#IMM:8

【字节数 / 周期数】

字节数 / 周期数	2 / 1

PUSH PUSH

(3) PUSH.W:S #IMM:16

b7							b0
1	0	0	0	1	0	1	1

#IMM:16

【字节数 / 周期数】

字节数 / 周期数	3 / 1

PUSH PUSH

(4) PUSH.L:S #IMM:32

b7							b0
1	1	0	0	1	0	1	1

#IMM:32

【字节数 / 周期数】

字节数 / 周期数	5 / 1

PUSH

PUSH

(5) PUSH.W:S #IMMEX:8

b7							b0
0	0	0	0	1	0	1	1

#IMM:8

【字节数 / 周期数】

字节数 / 周期数	2 / 1

PUSH

PUSH

(6) PUSH.L:S #IMMEX:8

b7							b0
0	0	1	1	1	0	1	1

#IMM:8

【字节数 / 周期数】

字节数 / 周期数	2 / 1

PUSH

PUSH

(7) PUSH.L:S #IMMEX:16

b7							b0
1	0	0	1	1	0	1	1

#IMM:16

【字节数 / 周期数】

字节数 / 周期数	3 / 1

PUSHA

PUSHA

(1) PUSHA src

b7							b0	b7							b0
0	1	0	0	1	1	0	1	0	0	0	g4	g3	g2	g1	g0

src 码
dsp:8
dsp:16
dsp:24

*1 由 g4 ～ g0 位指定 src 的操作数。

【字节数 / 周期数】

src	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	4 / 1	5 / 1	4 / 1	5 / 1

PUSHC

PUSHC

(1) PUSHC src

b7							b0	b7							b0
0	1	0	0	1	0	0	0	q2	q1	q0	0	1	0	0	0

*1 由 q2 ～ q0 位指定 src 的专用寄存器。

*2 运算长度固定为长字。

【字节数 / 周期数】

字节数 / 周期数	2 / 3

PUSHM

PUSHM

(1) PUSHM src

b7							b0
0	1	0	SB*1	1	0	1	1

#IMM:8

*1 在保存 SB 寄存器时为 “1”，否则为 “0”。

*2 由 #IMM:8 指定其他的寄存器。

bit	7	6	5	4	3	2	1	0
寄存器	R2R0	R3R1	R6R4	R7R5	A0	A1	A2	A3

*3 当 src 为组 1 寄存器直接时，在指令码的起始位置附加 01101111。

【字节数 / 周期数】

字节数 / 周期数	2 / n*4

*4 n 为要保存的寄存器的个数。

*5 当 src 为组 1 寄存器直接时，表中的字节数加 1。

REIT

REIT

(1) REIT

b7							b0
1	1	0	0	1	1	1	1

【字节数 / 周期数】

字节数 / 周期数	1 / 6

RMPA

RMPA

(1) RMPA.size

b7							b0	b7							b0
1	0	1	0	0	1	w1	w0	0	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 11+1.5m*1

*1 m 为运算次数。

ROLC

ROLC

(1) ROLC.size dest

b7	b0	b7						b0	b7							b0
0011	1111	1	1	0	0	0	1	w1	w0	0	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

RORC

RORC

(1) RORC.size dest

b7	b0	b7						b0	b7							b0
0011	1111	1	1	0	1	0	1	w1	w0	0	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

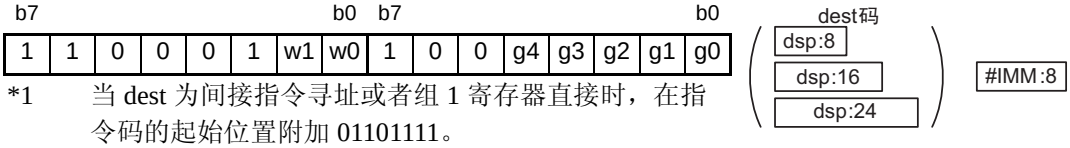
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ROT ROT

(1) ROT.size #IMM:8,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

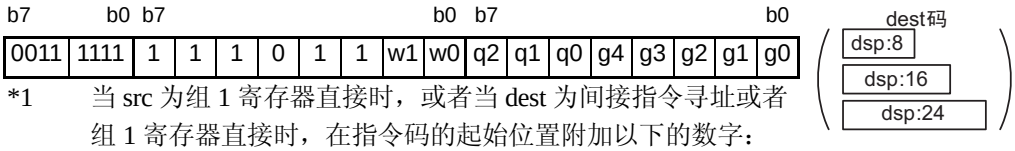
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ROT ROT

(2) ROT.size src,dest



*1 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为组 1 寄存器直接时：01011111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01101111

当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 q2 ～ q0 位指定 src 的寄存器，由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

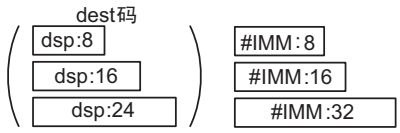
*3 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

ROUND ROUND

(1) ROUND #IMM(EX),dest

b7 b0 b7 b0 b7 b0

0111	1111	1	0	1	1	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
#IMMEX:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
#IMM:32	7 / 4	7 / 5	8 / 5	9 / 5	10 / 5	9 / 5	10 / 5

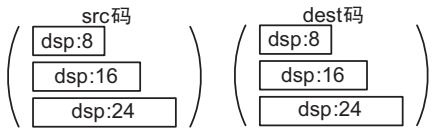
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

ROUND ROUND

(2) ROUND src,dest

b7 b0 b7 b0 b7 b0

0111	1111	1	0	1	1	g4	g3	1	0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 4	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
[An]	3 / 5	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
dsp:8[ ]	4 / 5	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
dsp:16[ ]	5 / 5	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24[ ]	6 / 5	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5
dsp:16	5 / 5	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24	6 / 5	6 / 5	7 / 5	8 / 5	9 / 5	8 / 5	9 / 5

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

RTS

RTS

(1) RTS

b7							b0
1	1	0	1	1	1	1	1

【字节数 / 周期数】

字节数 / 周期数	1 / 5

SBB

SBB

(1) SBB.size #IMM(EX),dest

b7	b0	b7							b0	b7							b0
0111	1111	0	0	0	1	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8 / #IMM:16 / #IMM:32

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM(EX):16	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
#IMM:32	7 / 1	7 / 2	8 / 2	9 / 2	10 / 2	9 / 2	10 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

SBB SBB

(2) SBB.size src,dest

b7										b0 b7							b0
0111	1111	0	0	0	1	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0

src码: dsp:8 / dsp:16 / dsp:24　　dest码: dsp:8 / dsp:16 / dsp:24

*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
[An]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:8[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:16[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24[ ]	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3
dsp:16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

SCCnd *SCCnd*

(1) SC*Cnd*.size dest

b7										b0 b7							b0
0011	1111	0	1	1	1	1	c3	w1	w0	c2	c1	c0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 c3 ～ c0 位指定 *Cnd*，详细内容请参照 “4.2.3 条件的指定”。

*3 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 2	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

SCMPU SCMPU

(1) SCMPU.size

b7							b0	b7							b0
1	0	0	1	0	1	0	w0	1	0	0	0	1	0	0	0

*1 由 w0 位指定 “.B” 或者 “.W”。当 w0 为 “0” 时，指定 “.B”；当 w0 为 “1” 时，指定 “.W”。

【字节数 / 周期数】

字节数 / 周期数	2 / 8+3m*2

*2 m 为要比较的 8 字节的字数。

*3 当比较源的起始地址不为 8 字节边界时，周期数加 1。同样，当比较目标的起始地址不为 8 字节边界时，周期数也加 1。

SHA SHA

(1) SHA.size #IMM:8,dest

b7							b0	b7							b0
1	1	0	1	0	1	w1	w0	0	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

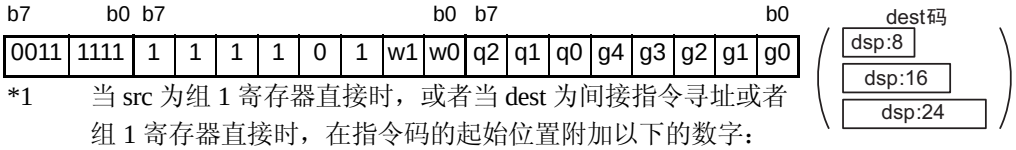
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

SHA SHA

(2) SHA.size src,dest



*1 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为组 1 寄存器直接时：01011111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01101111

当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 q2 ～ q0 位指定 src 的寄存器，由 g4 ～ g0 位指定 dest 的操作数。

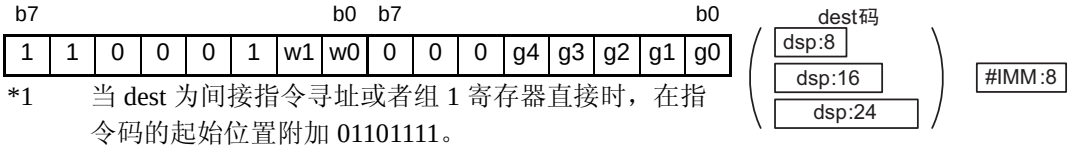
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

SHL SHL

(1) SHL.size:G #IMM:8,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

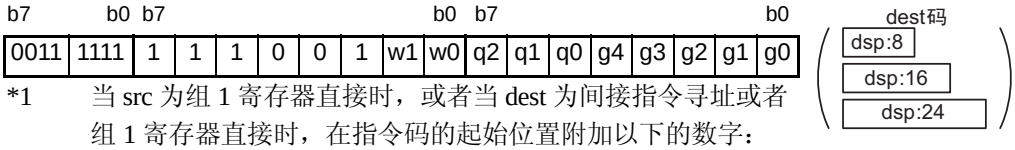
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

SHL SHL

(2) SHL.size:G src,dest



*1 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为组 1 寄存器直接时：01011111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01101111

当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 q2 ～ q0 位指定 src 的寄存器，由 g4 ～ g0 位指定 dest 的操作数。

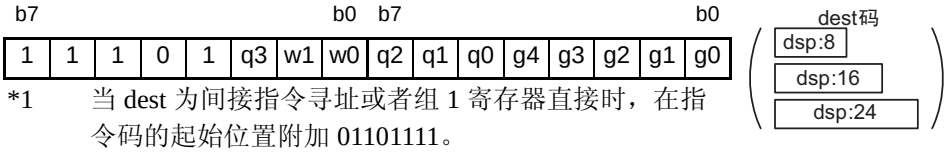
【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

SHL SHL

(3) SHL.size:Q #IMM:4,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 q3 ～ q0 的 4 位指定 #IMM:4，由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

SIN SIN

(1) SIN.size

b7							b0	b7							b0
1	1	0	0	0	1	w1	w0	0	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 3+2m*1

*1 m 为运算次数。

SMOVB SMOVB

(1) SMOVB.size

b7							b0	b7							b0
1	0	1	1	0	1	w1	w0	0	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 3+2m*1

*1 m 为运算次数。

SMOVF SMOVF

(1) SMOVF.size

b7							b0	b7							b0
1	1	0	0	0	1	w1	w0	1	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 2+2m*1

*1 m 为运算次数。

SMOVF SMOVF

(2) SMOVF.Q

b7							b0	b7							b0
1	1	0	1	0	1	1	0	1	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 4+2m*1

*1 m 为运算次数。

SMOVU SMOVU

(1) SMOVU.size

b7							b0	b7							b0
1	1	0	1	0	1	0	w0	1	0	0	0	1	0	0	0

*1 由 w0 位指定 “.B” 或者 “.W”。当 w0 为 “0” 时，指定 “.B”；当 w0 为 “1” 时，指定 “.W”。

【字节数 / 周期数】

字节数 / 周期数	2 / 5+3m*2

*2 m 为要比较的 8 字节的字数。

*3 当比较源的起始地址不为 8 字节边界时，周期数加 1。同样，当比较目标的起始地址不为 8 字节边界时，周期数也加 1。

SOUT SOUT

(1) SOUT.size

b7							b0	b7							b0
1	0	0	0	0	1	w1	w0	1	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 3+2m*1

*1 m 为运算次数。

SSTR SSTR

(1) SSTR.size

b7							b0	b7							b0
1	1	0	1	0	1	w1	w0	0	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 5+m*1

*1 m 为运算次数。

SSTR SSTR

(2) SSTR.Q

b7							b0	b7							b0
1	0	0	1	0	1	1	0	1	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 8+m*1

*1 m 为运算次数。

STC STC

(1) STC src,dest

b7		b0	b7						b0	b7							b0
0011	1111	1	1	1	1	1	0	1	0	q2	q1	q0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 q2 ～ q0 位指定 src 的专用寄存器，详细内容请参照 “4.2.2 操作数的指定”的 “(5) 专用寄存器直接 （CPU）”。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 2	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

STC STC

(2) STC src,dest

b7		b0	b7						b0	b7							b0
0011	1111	1	0	1	1	0	1	1	0	0	0	0	g4	g3	g2	g1	g0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 #IMM:8 指定 src 的寄存器，详细内容请参照 “4.2.2 操作数的指定”的 “(6) 专用寄存器直接（DMAC 和 VCT）”。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 3	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

STCTX STCTX

(1) STCTX abs:16,dsp:24

b7 b0	b7						b0	b7							b0		
0011 1111	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	abs:16	dsp:24

【字节数 / 周期数】

字节数 / 周期数	8 / 10+m*1

*1 m 为要传送的寄存器的个数。

STNZ STNZ

(1) STNZ.size #IMM,dest

b7 b0	b7							b0	b7							b0		
0111 1111	1	0	1	1	0	1	w1	w0	1	0	0	g4	g3	g2	g1	g0	dest码 dsp:8 / dsp:16 / dsp:24	#IMM:8 / #IMM:16 / #IMM:32

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 2	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2

*3 当长度说明符 （.size）为 “.W” 时，表中的字节数加 1；当长度说明符 （.size）为 “.L” 时，表中的字节数加 3。

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

STOP STOP

(1) STOP

b7 b0	b7 b0	b7 b0	b7 b0	b7 b0	b7 b0	b7 b0	b7 b0	b0
0011 1111	1010	0100	0000	1000	1100	0010	0001	0000	0000	0110	0000	0000

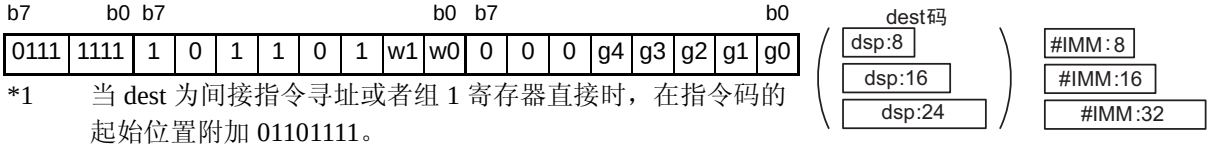
【字节数 / 周期数】

字节数 / 周期数	7 / 8

STZ

STZ

(1) STZ.size #IMM,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 2	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2

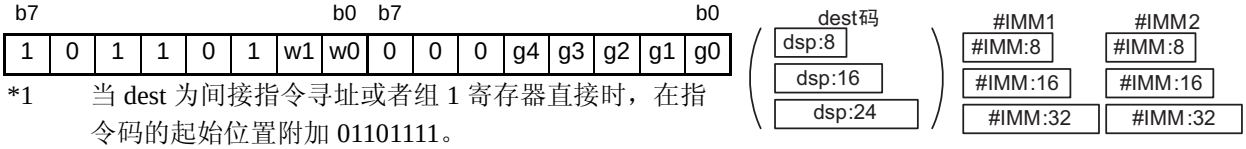
*3 当长度说明符 （.size）为 “.W” 时，表中的字节数加 1；当长度说明符 （.size）为 “.L” 时，表中的字节数加 3。

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

STZX

STZX

(1) STZX.size #IMM1,#IMM2,dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01101111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

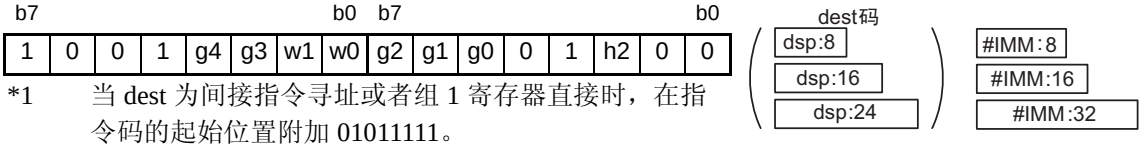
dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	4 / 4	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4

*3 当长度说明符 （.size）为 “.W” 时，表中的字节数加 2；当长度说明符 （.size）为 “.L” 时，表中的字节数加 6。

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

SUB SUB

(1) SUB.size #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

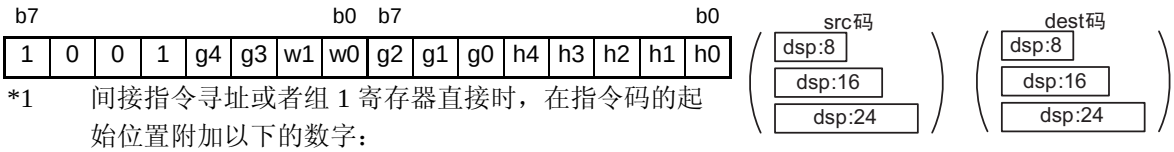
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

SUB SUB

(2) SUB.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

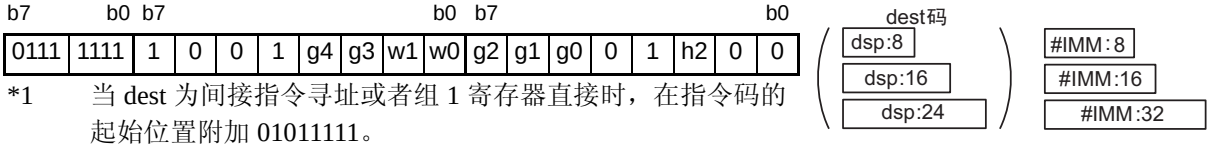
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2
[An]	2 / 2	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3
dsp:8[ ]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:16[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:16	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

SUBF SUBF

(1) SUBF #IMM(EX),dest



*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

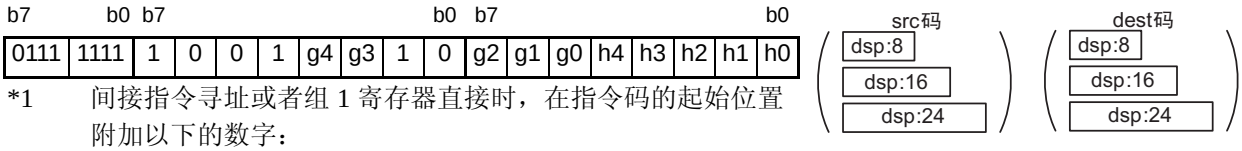
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
#IMMEX:16	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
#IMM:32	7 / 4	7 / 5	8 / 5	9 / 5	10 / 5	9 / 5	10 / 5

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

SUBF SUBF

(2) SUBF src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 4	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
[An]	3 / 5	3 / 6	4 / 6	5 / 6	6 / 6	5 / 6	6 / 6
dsp:8[ ]	4 / 5	4 / 6	5 / 6	6 / 6	7 / 6	6 / 6	7 / 6
dsp:16[ ]	5 / 5	5 / 6	6 / 6	7 / 6	8 / 6	7 / 6	8 / 6
dsp:24[ ]	6 / 5	6 / 6	7 / 6	8 / 6	9 / 6	8 / 6	9 / 6
dsp:16	5 / 5	5 / 6	6 / 6	7 / 6	8 / 6	7 / 6	8 / 6
dsp:24	6 / 5	6 / 6	7 / 6	8 / 6	9 / 6	8 / 6	9 / 6

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 src 或者 dest 为间接指令寻址时，表中的周期数加 2 ； 当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

SUNTIL SUNTIL

(1) SUNTIL.size

b7							b0	b7							b0
1	0	1	0	0	1	w1	w0	1	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 3+3m*1

*1 m 为运算次数。

SWHILE SWHILE

(1) SWHILE.size

b7							b0	b7							b0
1	0	1	1	0	1	w1	w0	1	0	0	0	1	0	0	0

【字节数 / 周期数】

字节数 / 周期数	2 / 3+3m*1

*1 m 为运算次数。

TST TST

(1) TST.size #IMM(EX),dest

b7		b0	b7							b0	b7							b0
0111	1111	0	0	1	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0

dest码: dsp:8 / dsp:16 / dsp:24

#IMM:8 / #IMM:16 / #IMM:32

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX ； 当 h2 位是 “1” 时为 #IMM。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM(EX):16	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
#IMM:32	7 / 1	7 / 2	8 / 2	9 / 2	10 / 2	9 / 2	10 / 2

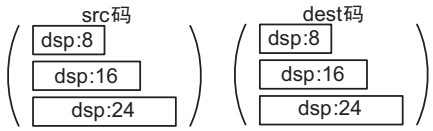
*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1 ； 当 dest 为间接指令寻址时，表中的周期数加 2。

TST

TST

(2) TST.size src,dest

b7		b0 b7					b0 b7								b0
0111	1111	0	0	1	0	g4	g3	w1	w0	g2	g1	g0	h4	h3	h2	h1	h0



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位中断 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
[An]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:8[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:16[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24[ ]	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3
dsp:16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

UND

UND

(1) UND

b7							b0
1	1	1	1	1	1	1	1

【字节数 / 周期数】

字节数 / 周期数	1 / 12

WAIT

WAIT

(1) WAIT

b7	b0 b7	b0 b7	b0 b7	b0 b7	b0 b7	b0 b7	b0
0011 1111	1010 0101	0000 1000	1100 0010	0001 0000	0000 0110	0000 0000	

【字节数 / 周期数】

字节数 / 周期数	7 / 8

XCHG

XCHG

(1) XCHG.size src,dest

b7		b0	b7							b0	b7							b0
0011	1111	1	1	1	0	0	0	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0	

dest码: dsp:8 / dsp:16 / dsp:24

*1 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为组 1 寄存器直接时：01011111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01101111

当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 q2 ～ q0 位指定 src 的寄存器，由 g4 ～ g0 位指定 dest 的操作数。

【字节数 / 周期数】

dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
字节数 / 周期数	3 / 3	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*3 当 src 为组 1 寄存器直接时，或者当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 为组 1 寄存器直接并且 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

XOR

XOR

(1) XOR.size #IMM(EX),dest

b7		b0	b7							b0	b7							b0
0011	1111	1	0	1	0	g4	g3	w1	w0	g2	g1	g0	0	1	h2	0	0	

dest码: dsp:8 / dsp:16 / dsp:24

#IMM: #IMM:8 / #IMM:16 / #IMM:32

*1 当 dest 为间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加 01011111。

*2 由 g4 ～ g0 位指定 dest 的操作数。

*3 由 h2 位指定 src 的操作数。当 h2 位是 “0” 时为 #IMMEX；当 h2 位是 “1” 时为 #IMM。

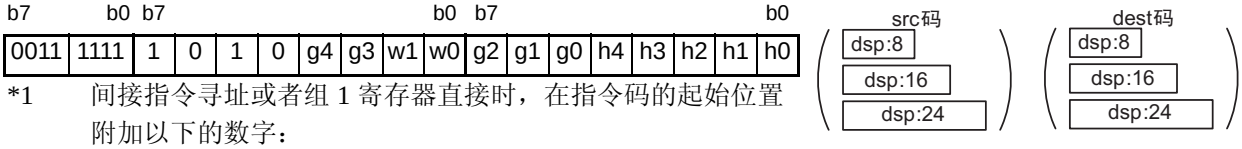
【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM(EX):16	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
#IMM:32	7 / 1	7 / 2	8 / 2	9 / 2	10 / 2	9 / 2	10 / 2

*4 当 dest 为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 dest 为间接指令寻址时，表中的周期数加 2。

XOR XOR

(2) XOR.size src,dest



*1 间接指令寻址或者组 1 寄存器直接时，在指令码的起始位置附加以下的数字：

当 src 为间接指令寻址或者组 1 寄存器直接时：01101111

当 dest 为间接指令寻址或者组 1 寄存器直接时：01011111

当 src 和 dest 都为间接指令寻址或者组 1 寄存器直接时：01001111

*2 由 g4 ～ g0 位指定 dest 的操作数，由 h4 ～ h0 位指定 src 的操作数。

【字节数 / 周期数】

src \ dest	寄存器	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
寄存器	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
[An]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:8[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:16[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24[ ]	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3
dsp:16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24	6 / 2	6 / 3	7 / 3	8 / 3	9 / 3	8 / 3	9 / 3

*3 当 src 和 dest 中的一个或者二个都为间接指令寻址或者组 1 寄存器直接时，表中的字节数加 1；当 src 或者 dest 为间接指令寻址时，表中的周期数加 2；当 src 和 dest 都为间接指令寻址时，表中的周期数加 4。

5. 中断

5.1 中断的概要

如果接受中断请求，就转移到中断向量表中设定的中断程序。必须给各中断向量表设定中断程序的起始地址，中断向量表的详细内容请参照 “1.7 向量表”。

5.1.1 中断的分类

中断的分类如图 5.1 所示，中断源 （非屏蔽）和固定向量表如表 5.1 所示。

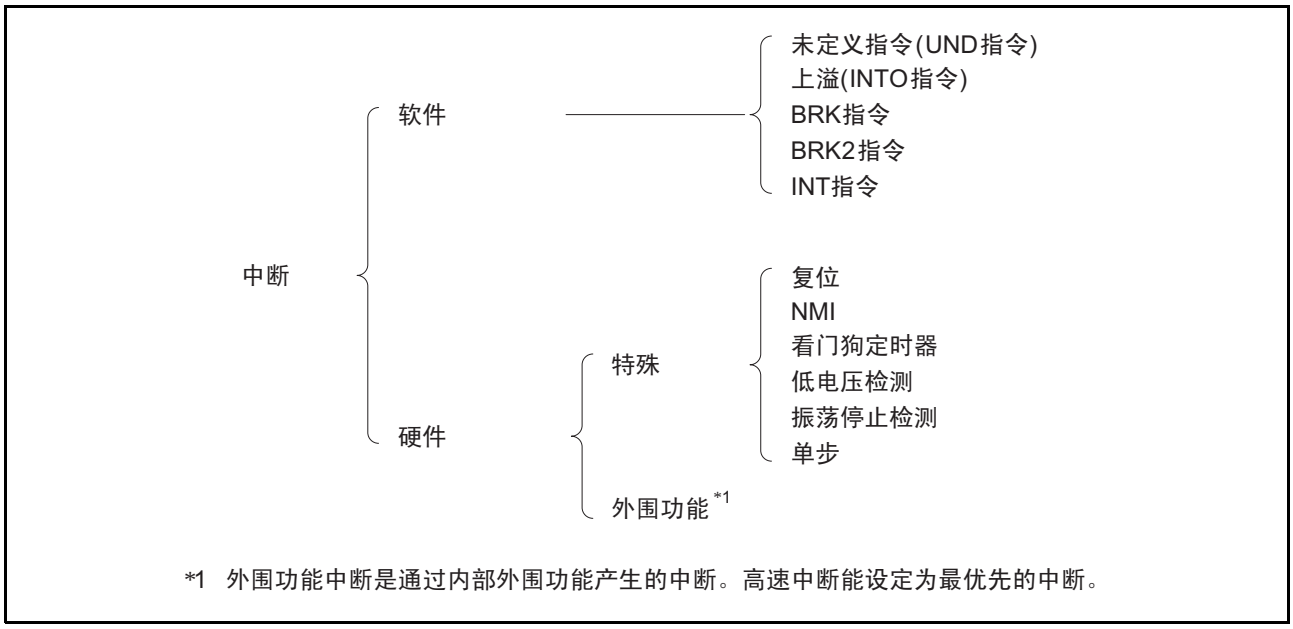


图 5.1 中断的分类

表 5.1 中断源 （非屏蔽）和固定向量表

中断源	向量表地址 地址 （L）～地址 （H）	备注
未定义指令	FFFFFFDCh ～ FFFFFFDFh	通过 UND 指令产生中断。
上溢	FFFFFFE0h ～ FFFFFFE3h	通过 INTO 指令产生中断。
BRK 指令	FFFFFFE4h ～ FFFFFFE7h	当地址 FFFFFFE7h 的内容为 “FFh” 时，从可变向量表中的向量所示的地址开始执行。
看门狗定时器 低电压检测 振荡停止检测	FFFFFFF0h ～ FFFFFFF3h	
NMI	FFFFFFF8h ～ FFFFFFFBh	通过 $\overline{\text{NMI}}$ 引脚产生外部中断。
复位	FFFFFFFCh ～ FFFFFFFFh	

根据是否可以屏蔽，中断又能分为可屏蔽中断和非屏蔽中断。

(1) 可屏蔽中断

能通过中断允许标志 （I 标志）禁止 （允许）中断，也能通过中断优先级更改中断的优先级。

(2) 非屏蔽中断

不能通过中断允许标志 （I 标志）禁止 （允许）中断，也不能通过中断优先级更改中断的优先级。

5.1.2 软件中断

软件中断通过执行指令而产生，是非屏蔽中断。
软件中断有以下 5 种中断：

(1) 未定义指令中断

如果执行 UND 指令，就产生未定义指令中断。

(2) 上溢中断

如果在上溢标志 （O 标志）为 “1”时执行 INTO 指令，就产生上溢中断。根据运算， O 标志发生变化的指令如下：

ABS、 ADC、 ADCF、 ADD、 ADDF、 ADSF、 CMP、 CMPF、 CNVIF、 DIV、 DIVF、 DIVU、 DIVX、 EDIV、 EDIVU、 EDIVX、 MUL、 MULF、 MULU、 MULX、 NEG、 RMPA、 ROUND、 SBB、 SCMPU、 SHA、 SUB、 SUBF、 SUNTIL、 SWHILE

(3) BRK 中断

如果执行 BRK 指令，就产生 BRK 中断。

(4) BRK2 中断

如果执行 BRK2 指令，就产生 BRK2 中断。
此中断是仿真器专用的中断，在用户程序中不能使用。

(5) INT 指令中断

如果指定软件中断序号 0 ～ 255 并且执行 INT 指令，就产生 INT 指令中断。因为软件中断序号 0 ～ 127 分配给外围功能中断，所以能通过执行 INT 指令，执行和外围功能中断相同的中断程序。

用于 INT 指令中断的堆栈指针 （SP）因软件中断序号而不同。对于软件中断序号 0 ～ 127，在接受中断请求时先将堆栈指针指定标志 （U 标志）压栈，然后将 U 标志置 “0”，在选择中断堆栈指针 （ISP）后执行中断响应顺序。在从中断程序返回时恢复接受中断请求前的 U 标志。对于软件中断序号 128 ～ 255，不转换堆栈指针。

5.1.3 硬件中断

硬件中断有特殊中断和外围功能中断。

只能将外围功能中断中最优先的 1 个中断设定为高速中断。

(1) 特殊中断

特殊中断是非屏蔽中断。

(a) 复位

如果给 $\overline{\text{RESET}}$ 引脚输入 “L” 电平，就产生复位。

(b) NMI 中断

如果给 $\overline{\text{NMI}}$ 引脚输入 “L” 电平，就产生 NMI 中断。

(c) 看门狗定时器中断

这是由看门狗定时器产生的中断。

(d) 低电压检测中断

这是由低电压检测电路产生的中断。

(e) 振荡停止检测中断

这是由振荡停止检测电路产生的中断。

(f) 单步中断

这是调试程序专用的中断，通常不能使用。如果将调试标志 （D 标志）置 “1”，就立即产生单步中断。

(2) 外围功能中断

外围功能中断是由内部外围功能产生的中断。因为内部外围功能因产品种类而不同，所以各种中断源也因产品种类而不同。中断向量表和 INT 指令使用的软件中断序号 0 ～ 127 相同。有关外围功能中断的详细内容，请参照硬件用户手册。

对于外围功能中断，在接受中断请求时先将堆栈指针指定标志 （U 标志）压栈，然后将 U 标志置 “0”，在选择中断堆栈指针 （ISP）后执行中断响应顺序。在从中断响应顺序返回时恢复接受中断请求前的 U 标志。

(3) 高速中断

高速中断是能高速执行中断响应的中断，只能用于外围功能中断中最优先的 1 个中断。必须使用 FREIT 指令从高速中断返回。

有关高速中断的详细内容，请参照硬件用户手册。

5.2 中断控制

以下说明可屏蔽中断的允许和禁止以及接受优先级的设定，但是在此说明的内容不适用于非屏蔽中断。

通过中断允许标志 （I 标志）、中断请求级选择位和处理器中断优先级 （IPL），允许或者禁止可屏蔽中断。中断请求位表示中断请求的有无。中断请求位和中断请求级选择位分配在各中断的中断控制寄存器，而中断允许标志 （I 标志）和处理器中断优先级 （IPL）分配在标志寄存器 （FLG）。

有关中断控制寄存器的存储器分配和寄存器结构，请参照硬件用户手册。

5.2.1 中断允许标志 （I 标志）

中断允许标志 （I 标志）控制可屏蔽中断的允许或者禁止。如果将此标志置 “1”，就允许全部的可屏蔽中断；如果置 “0”，就禁止全部的可屏蔽中断。此标志在复位解除后变为 “0”。

在 I 标志发生变化后，变化的内容反映到中断请求接受判断的时序如下：。

- 通过REIT指令、 FREIT指令更改了I标志时，就从此REIT指令、 FREIT指令开始反映。
- 通过FCLR、 FSET、 POPC、 LDC指令更改了I标志时，就从下一条指令开始反映。

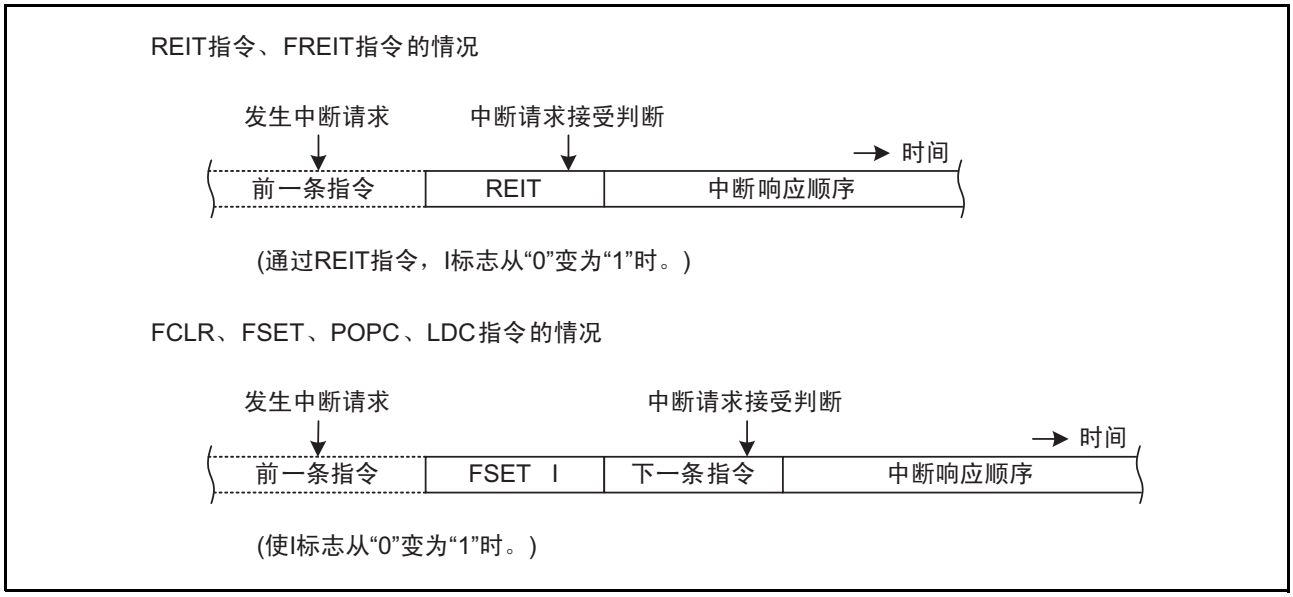


图 5.2 I 标志发生变化的内容反映到中断的时序

5.2.2 中断请求位

如果发生中断请求，中断请求位就为 “1” 并且保持到接收中断请求为止。如果接受中断请求，中断请求位就变为 “0”。

此位能通过软件置 “0”（不能写 “1”）。

5.2.3 中断请求级和处理器中断优先级 （IPL）

通过中断控制寄存器的中断请求级选择位来设定中断请求级。在发生中断请求时，将中断请求级和处理器中断优先级 （IPL）比较，如果中断请求级大于处理器中断优先级 （IPL），就允许该中断。因此，如果将中断请求级设定为 “0 级”，就禁止该中断。

中断请求级的设定以及由处理器中断优先级 （IPL）的内容允许的中断请求级分别如**表 5.2** 和**表 5.3** 所示。

表 5.2 中断请求级的设定

中断请求级选择位			中断请求级	优先级
b2	b1	b0		
1	1	1	7 级	高
1	1	0	6 级	
1	0	1	5 级	
1	0	0	4 级	
0	1	1	3 级	
0	1	0	2 级	
0	0	1	1 级	低
0	0	0	0 级 （禁止中断）	—

表 5.3 由处理器中断优先级 （IPL）的内容允许的中断请求级

处理器中断 优先级 （IPL）			允许的中断请求级
IPL_2	IPL_1	IPL_0	
1	1	1	禁止全部可屏蔽中断。
1	1	0	只允许 7 级。
1	0	1	允许 6 级或者 6 级以上。
1	0	0	允许 5 级或者 5 级以上。
0	1	1	允许 4 级或者 4 级以上。
0	1	0	允许 3 级或者 3 级以上。
0	0	1	允许 2 级或者 2 级以上。
0	0	0	允许 1 级或者 1 级以上。

能接受中断请求的条件如下所示：

- 中断允许标志 （I标志） = 1
- 中断请求位 = 1
- 中断请求级 > 处理器中断优先级 （IPL）

中断允许标志 （I 标志）、中断请求位、中断请求级选择位和处理器中断优先级 （IPL）各自独立而且不相互影响。

在更改处理器中断优先级 （IPL）或者各中断请求级后，变化的内容反映到中断的时序如下：

- 通过REIT指令、 FREIT指令更改了处理器中断优先级 （IPL）时，就从此REIT指令、 FREIT指令开始反映。
- 通过POPC、 LDC、 LDIPL指令更改了处理器中断优先级 （IPL）时，就从下一条指令开始反映。
- 通过MOV指令等更改了各中断控制寄存器的中断请求级时，就在改写寄存器的值后经过1个外围总线时钟周期后开始反映。

5.2.4 中断控制寄存器的变更

必须在不发生与中断控制寄存器对应的中断请求处，更改中断控制寄存器。如果有可能发生中断请求，就必须在禁止中断后进行更改。

在更改中断控制寄存器后立即允许中断时，为了不受指令队列的影响而在写中断控制寄存器后使中断允许标志（I 标志）置位，必须采取插入 NOP 等措施。

在禁止中断的状态下执行中断控制寄存器的改写指令的过程中，如果发生与该寄存器对应的中断请求，根据指令，中断请求位就有可能不被置位。如果因此而引起问题，就必须使用以下的指令更改寄存器：

- AND
- OR
- BCLR
- BSET

5.3 中断响应顺序

以下说明从接受中断请求到执行中断程序为止的中断响应顺序。

如果在执行指令的过程中发生中断请求，就在执行该指令后判断优先级，并且从下一个周期转移到中断响应顺序。但是，如果在执行 RMPA、SCMPU、SIN、SMOVB、SMOVF、SMOVU、SOUT、SSTR、SUNTIL、SWHILE 各指令的过程中发生中断请求，就暂时中断指令的运行，转移到中断响应顺序。

中断响应顺序按顺序执行以下运行：

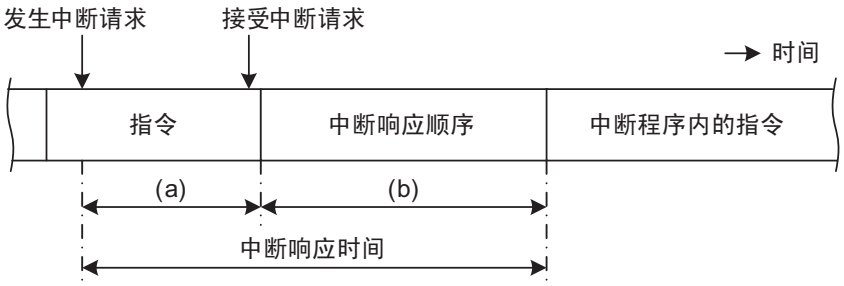
1. CPU通过返回中断应答从中断控制器取中断信息（中断序号和中断请求级），然后对应中断的请求位变为“0”。
2. 将中断响应顺序前的标志寄存器（FLG）的内容保存到CPU内部的临时寄存器*1。
3. 将中断允许标志（I标志）、调试标志（D标志）和堆栈指针指定标志（U标志）置“0”（但是，在执行了软件中断序号128～255的INT指令时， U标志不变）。
4. 将CPU内部的临时寄存器*1的内容压栈。在高速中断的情况下，保存到标志保存寄存器（SVF）。
5. 将程序计数器（PC）的内容压栈。在高速中断的情况下，保存到PC保存寄存器（SVP）。
6. 给处理器中断优先级（IPL）设定已接受中断的中断请求级。
7. 从中断向量表中取已接受中断源的向量。
8. 将取到的中断向量设定到程序计数器（PC）。

在中断响应顺序结束后，从中断程序的起始地址开始执行指令。

*1 用户不能使用。

5.3.1 中断响应时间

中断响应时间是指从发生中断请求开始到执行中断程序内的第一条指令为止的时间，由发生中断请求开始到当时正在执行的指令结束为止的时间 (a) 和执行中断响应顺序的时间 (b) 构成，中断响应时间如图 5.3 所示。



(a) 这是从发生中断请求到当时正在执行的指令结束为止的时间。
(b) 这是执行中断响应顺序的时间。

图 5.3 中断响应时间

(a) 的时间因正在执行的指令而不同。
(b) 的时间如表 5.4 所示。

表 5.4 中断响应顺序的执行时间

中断	中断向量的地址	中断响应顺序的执行时间 （内部存储器）
外围功能	4 字节边界 *1	13 个周期 *2
INT 指令	4 字节边界 *1	11 个周期
BRK 指令 （可变向量表）	4 字节边界 *1	16 个周期
未定义指令	FFFFFFDCh*3	12 个周期
上溢	FFFFFFE0h*3	12 个周期
BRK 指令 （固定向量表）	FFFFFFE4h*3	19 个周期
看门狗定时器 低电压检测 振荡停止检测	FFFFFFF0h*3	11 个周期
NMI	FFFFFFF8h*3	10 个周期
单步 BRK2 指令 DBC 中断	4 字节边界 *3	19 个周期
高速中断 *4	向量表是内部寄存器	11 个周期

*1 必须尽可能将中断向量表分配到 4 字节边界，否侧就会延长执行时间。
*2 当外围总线的等待数大于等于 3 时，就只增加 (等待数 –2) 的周期数。
*3 向量表地址固定为 4 字节边界。
*4 高速中断不受这些条件的影响。

5.3.2 处理器中断优先级 （IPL）的变化

如果接受中断请求，就给处理器中断优先级 （IPL）设定已接受中断的中断请求级。

如果接受了没有中断请求级的中断请求，IPL 被设定为如**表 5.5** 所示的值。

表 5.5 没有中断请求级的中断和 IPL 的关系

没有中断请求级的中断源	IPL 的设定值
NMI、看门狗定时器、低电压检测、振荡停止检测	7
复位	0
其他	不变

5.3.3 寄存器保存

在中断响应顺序中，只将标志寄存器 （FLG）和程序计数器 （PC）的内容压栈。压栈的顺序是：标志寄存器 - 程序计数器。接受中断请求前后的堆栈状态如**图 5.4** 所示。

在高速中断的中断响应顺序中，将标志寄存器 （FLG）保存到标志保存寄存器 （SVF）、程序计数器（PC）保存到 PC 保存寄存器 （SVP）。

必须在中断程序的开始通过软件将其他需要的寄存器保存。如果使用 PUSHM 指令，就能用 1 条指令将堆栈指针 （SP）以外的全部寄存器保存。

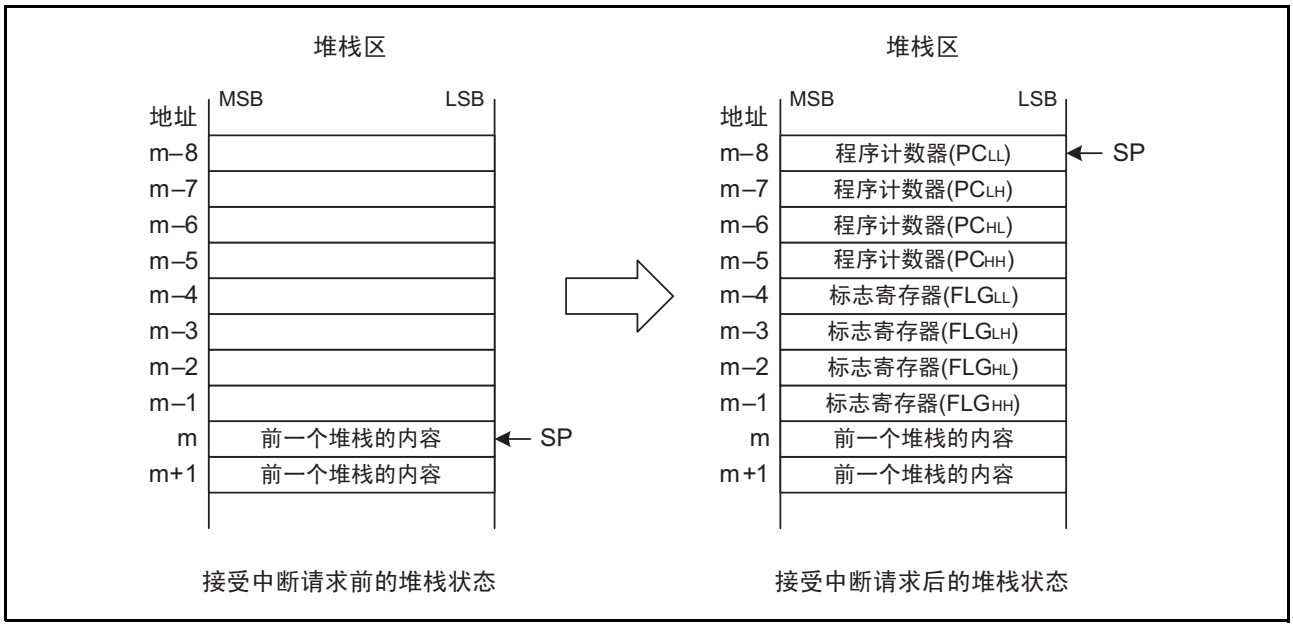


图 5.4 接受中断请求前后的堆栈状态

5.4 从中断程序的返回

如果在中断程序的最后执行 REIT 指令，就恢复到中断响应顺序前被压栈的标志寄存器 （FLG）和程序计数器 （PC）的内容。在高速中断的情况下，如果在中断程序的最后执行 FREIT 指令，就恢复到中断响应顺序前被保存到保存寄存器的标志寄存器 （FLG）和程序计数器 （PC）的内容。然后，返回到接受中断请求前正在执行的程序，继续执行被中断的处理。

必须在执行 REIT 指令、 FREIT 指令前使用 POPM 指令等，恢复在中断程序内通过软件保存的寄存器。

5.5 中断优先级

如果在同一采样点 （在调查是否有中断请求时）发生 2 个或者 2 个以上的中断请求，就接受优先级高的中断。

能通过中断请求级选择位设定任意的可屏蔽中断 （外围功能中断）的优先级。但是，如果设定相同的中断请求级，就接受由硬件设定的优先级 *1 高的中断。

复位 （复位为优先级最高的中断）和看门狗定时器中断等非屏蔽中断的优先级是由硬件设定的，由硬件设定的中断优先级如下所示：

看门狗定时器
复位＞低电压检测 ＞ NMI ＞外围功能＞单步
振荡停止检测

软件中断不受中断优先级的影响。如果执行指令，就一定转移到中断处理程序。

*1 因产品种类而不同，请参照硬件用户手册。

5.6 多重中断

转移到中断程序时的状态如下所示：

- 中断允许标志 （I标志）为 “0”（中断禁止状态）。
- 已接收中断的中断请求位为 “0”。
- 处理器中断优先级 （IPL）为已接受中断的中断请求级。

能通过在中断程序内将中断允许标志 （I 标志）置 “1”，接受请求级高于处理器中断优先级 （IPL）的中断请求，多重中断的处理例子如图 5.5 所示。

保持因优先级低而没有被接受的中断请求。在通过 REIT 指令、 FREIT 指令恢复 IPL 并且判断中断请求级时，如果被保持的中断请求的中断请求级大于被恢复的处理器中断优先级 （IPL），就接受被保持的中断请求。

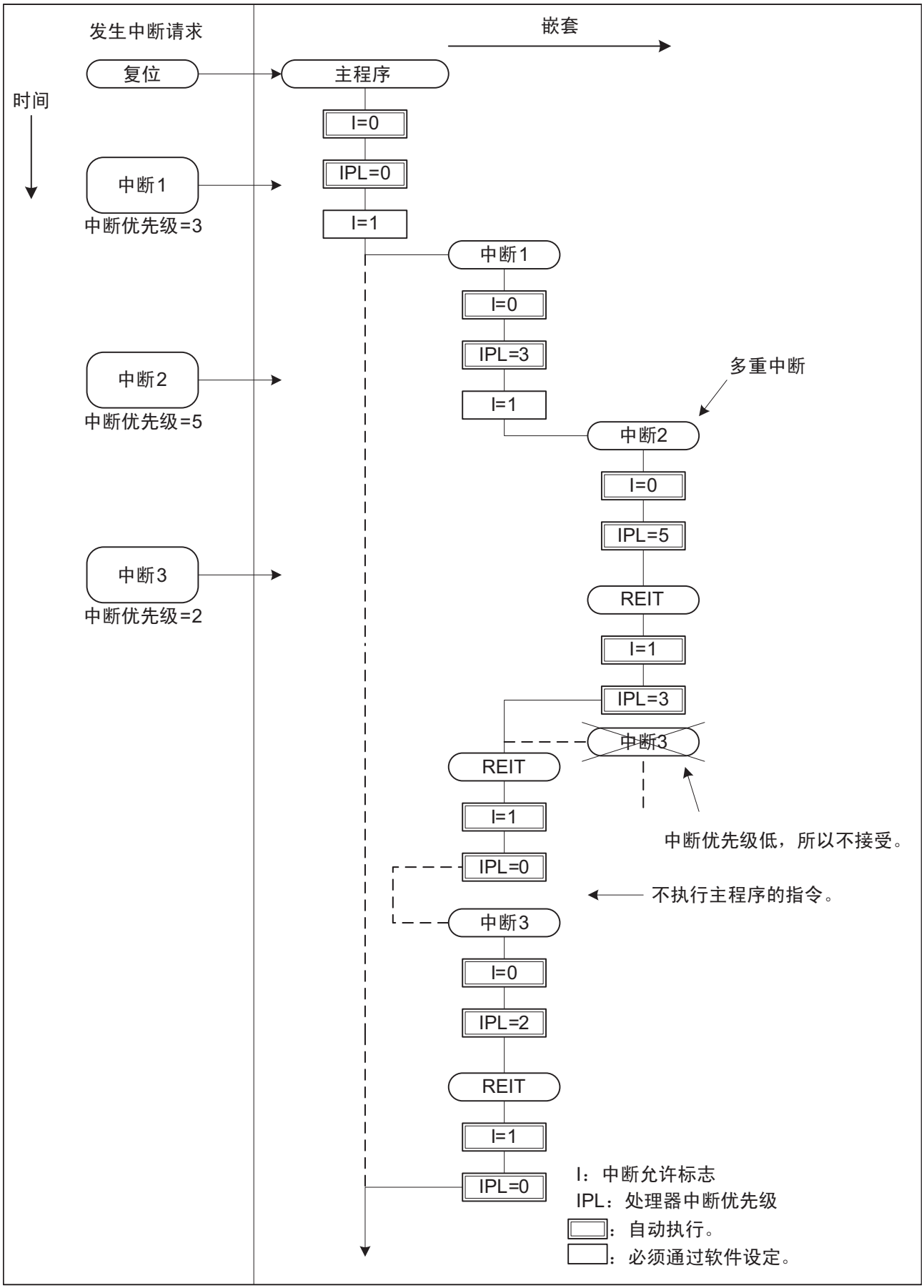

发生中断请求
嵌套
复位
主程序
时间
I=0
中断1
IPL=0
中断优先级=3
I=1
中断1
I=0
IPL=3
多重中断
中断2
I=1
中断优先级=5
中断2
I=0
IPL=5
中断3
REIT
中断优先级=2
I=1
IPL=3
中断3
REIT
I=1
中断优先级低，所以不接受。
IPL=0
不执行主程序的指令。
中断3
I=0
IPL=2
REIT
I=1
IPL=0
I：中断允许标志
IPL：处理器中断优先级
：自动执行。
：必须通过软件设定。

图 5.5 多重中断

5.7 中断的注意事项

以下举例说明使用中断时的注意事项：

(1) 堆栈指针的设定

在复位后，堆栈指针的值立即被初始化为 “00000000h”。如果在将值设定到堆栈指针前接受中断，就会导致程序失控，因此必须在允许中断前将值设定到堆栈指针。

尤其在使用 NMI 中断时，必须在程序的开始设定堆栈指针。在将 PM2 寄存器的 PM24 位置 “1” 后，就能立即接受 NMI 中断。

必须给堆栈指针设定 4 的倍数的地址，设定 4 的倍数能提高指令的执行效率。

(2) 中断控制寄存器的变更

必须在不发生与中断控制寄存器对应的中断请求处，更改中断控制寄存器。如果有可能发生中断请求，就必须在禁止中断后进行更改。

在更改中断控制寄存器后立即允许中断时，为了不受指令队列的影响而在写中断控制寄存器后使中断允许标志 （I 标志）置位，必须采取插入 NOP 等措施。

在禁止中断的状态下正在执行中断控制寄存器的改写指令时，如果发生与该寄存器对应的中断请求，根据指令，中断请求位就有可能不被置位。如果因此而引起问题，就必须使用以下的指令改写寄存器：

- AND
- OR
- BCLR
- BSET

(3) 停止模式 / 等待模式的解除

在通过外围功能中断解除停止模式或者等待模式时，需要事先将对象中断置为中断允许状态，并且将中断的请求级设定为高于用于从停止模式 / 等待模式返回的中断优先级选择位指定的优先级。必须给用于从停止模式 / 等待模式返回的中断优先级选择位设定与标志寄存器 （FLG）的处理器中断级 （IPL）相同的值。

复位和 NMI 中断不受用于从停止模式 / 等待模式返回的中断优先级选择位的影响，解除停止模式 / 等待模式。

索引

Symbols

数字

A

B

C

D

F

G

I

J

K

L

N

O

P

R

S

T

U

V

W

X

Y

Z

修订记录	R32C/100 系列 用户手册 软件篇

Rev.	发行日	修订内容	
		页	修订处
1.00	2010.08.20	—	初版发行

R32C/100 系列
用户手册 软件篇

Publication Date: Rev.1.00 Aug 20, 2010

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0

R32C/100 系列

RENESAS

瑞萨电子株式会社

RCJ09B0083-0100