お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

User's Manual

32

R32C/100シリーズ

ソフトウェアマニュアル

ルネサスマイクロコンピュータ
M16Cファミリ／ R32C/100シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2009.01

本資料ご利用に際しての留意事項

１．本資料は、お客様に用途に応じた適切な弊社製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
２．本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
３．本資料に記載の製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替および外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
４．本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認いただきますとともに、弊社ホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
５．本資料に記載した情報は、正確を期すため慎重に制作したものですが、万一本資料の記述の誤りに起因する損害がお客様に生じた場合においても、弊社はその責任を負いません。
６．本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。弊社は、適用可否に対する責任は負いません。
７．本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会ください。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
８．第７項にかかわらず、本資料に記載された製品は、下記の用途には使用しないでください。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
　１）生命維持装置。
　２）人体に埋め込み使用するもの。
　３）治療行為（患部切り出し、薬剤投与等）を行うもの。
　４）その他、直接人命に影響を与えるもの。
９．本資料に記載された製品のご使用につき、特に最大定格、動作電源電圧範囲、放熱特性、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて製品をご使用された場合の故障および事故につきましては、弊社はその責任を負いません。
１０．弊社は製品の品質および信頼性の向上に努めておりますが、特に半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。弊社製品の故障または誤動作が生じた場合も人身事故、火災事故、社会的損害などを生じさせないよう、お客様の責任において冗長設計、延焼対策設計、誤動作防止設計などの安全設計（含むハードウェアおよびソフトウェア）およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特にマイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
１１．本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
１２．本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断りいたします。
１３．本資料に関する詳細についてのお問い合わせ、その他お気付きの点等がございましたら弊社営業窓口までご照会ください。

本書の表記について

本書では、マイクロコンピュータの動作を詳細に説明するために、一定のルールに基づいて記載しています。

本書では、以下のような表記を用いています。

分類	記述	内容
記号	IMM	即値(Immediate)を表します。
	IMMEX	演算時に符号拡張(Extension)される即値を表します。
	abs	絶対アドレッシングの絶対(Absolute)値を表します。
	dsp	相対アドレッシングの変位(Displacement)を表します。
	[ ]	間接アドレッシングを表します。
	label	プログラムカウンタ相対で使用されるラベル(Label)を表します。
	bit	ビット番号を表します。
	R0LB、R0B、R2R0Bなど	下線部のように“B”を付加したレジスタシンボルは、バンク1レジスタを表します。
数値	000b	最後に“b”を付加した数値は2進数を表します。
	0000h	最後に“h”を付加した数値は16進数を表します。
	100	数字のみの場合は10進数(BCD)を表します。
ビット長指定	#IMM:8など	下線部はオペランド記号の有効ビット数を表します。
	:3	有効ビット長が3ビットであることを表します。
	:4	有効ビット長が4ビットであることを表します。
	:8	有効ビット長が8ビットであることを表します。
	:16	有効ビット長が16ビットであることを表します。
	:24	有効ビット長が24ビットであることを表します。
	:32	有効ビット長が32ビットであることを表します。
命令フォーマット	MOV:Sなど	下線部は命令フォーマットを指定する記号です。
	:G	ジェネリック形式を表します。
	:Q	クイック形式を表します。
	:S	ショート形式を表します。
	:Z	ゼロ形式を表します。
演算サイズ	MOV.Wなど	下線部は命令の演算サイズを指定する記号です。
	.B	バイト(8bit)演算を指定します。
	.W	ワード(16bit)演算を指定します。
	.L	ロングワード(32bit)演算を指定します。
	.BW	バイトからワードへの変換を指定します。
	.BL	バイトからロングワードへの変換を指定します。
	.WL	ワードからロングワードへの変換を指定します。
分岐距離指定	JMP.Aなど	下線部は分岐の相対距離の有効ビット数を指定する記号です。
	.S	3ビットのPC前方相対を表します。有効値は1 ~ 8です。
	.B	8ビットのPC相対を表します。有効値は-128 ~ +127です。
	.W	16ビットのPC相対を表します。 有効値は-32768 ~ +32767です。
	.A	24ビットのPC相対を表します。 有効値は-8388608 ~ +8388607です。

分類	記述	内容
オペレーション		原則としてC言語の文法規則に則っています。以下、本書で使用している記述について説明します。
	=	代入演算子です。右辺の値を左辺に代入します。
	-	単項演算子の負号、または二項演算子の「差」を表します。
	+	二項演算子の「和」を表します。
	*	ポインタ演算子、または二項演算子の「積」を表します。
	/	二項演算子の「商」を表します。
	%	二項演算子の「剰余」を表します。
	~	単項ビット演算子の「NOT」を表します。
	&	二項ビット演算子の「AND」を表します。
	|	二項ビット演算子の「OR」を表します。
	^	二項ビット演算子の「Exclusive OR」を表します。
	(float)	キャスト演算子です。
	;	文の終了を表します。
	{ }	複文の開始と終了を表します。{}内には複数の文が記述できます。
	if (式) 文1 else 文2	if文を表します。式を評価して、真であれば文1を、偽であれば文2を実行します。
	for (文1;式;文2) 文3	for文を表します。文1を実行した後、式を評価して、真であれば文3を実行します。文3の実行後は、文2を実行した後、式を評価します。
	do 文 while (式);	do文を表します。式が真の間、文を実行します。式の真偽にかかわらず、文は最低1回実行されます。
	while (式) 文	while文を表します。式が真の間、文を実行します。
	==、!=	比較演算子です。順に「等しい」、「等しくない」を表します。
	>、<	比較演算子です。順に「大なり」、「小なり」を表します。
	>=、<=	比較演算子です。'>'、'<' に '==' の条件が加わります。
	&&、||	論理演算子です。左側の条件と、右側の条件のそれぞれ「AND」と「OR」を表します。
	true	C言語にはありません。*Cnd* で指定した条件が成立したときに "1" になります。
選択可能なsrc/dest	#IMMEX	#IMMEX:8または#IMMEX:16が指定できます。
	#IMM	演算サイズに応じて、#IMM:8、#IMM:16、または#IMM:32が指定できます。 #IMM:3や#IMM:4は含みません。

目次

R32C/100シリーズ命令一覧

表 C.1　**アルファベット別ページ早見表**　(1 / 5)

ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
ABS	**絶対値**	44	180
ADC	**キャリー付き加算**	45	180
ADCF	**キャリーフラグの加算**	46	181
ADD	**キャリーなし加算**	47	182
ADDF	**浮動小数点加算**	49	184
ADSF	**サインフラグの加算**	50	185
AND	**論理積**	51	186
BCLR	**ビットクリア**	53	187
BITINDEX	**ビットインデックス**	54	188
BM*Cnd*	**条件ビット転送**	55	188
BMC	C**フラグが** "1" **の場合**	55	188
BMEQ	**等しい場合**	55	188
BMGE	**等しいか符号付きで大きい場合**	55	188
BMGEU	**等しいか大きい場合**	55	188
BMGT	**符号付きで大きい場合**	55	188
BMGTU	**大きい**	55	188
BMLE	**等しいか符号付きで小さい場合**	55	188
BMLEU	**等しいか小さい場合**	55	188
BMLT	**符号付きで小さい場合**	55	188
BMLTU	**小さい場合**	55	188
BMN	**負の場合**	55	188
BMNC	C**フラグが** "0" **の場合**	55	188
BMNE	**等しくない場合**	55	188
BMNO	O**フラグが** "0" **の場合**	55	188
BMNZ	Z**フラグが** "0" **の場合**	55	188
BMO	O**フラグが** "1" **の場合**	55	188
BMPZ	**正またはゼロの場合**	55	188
BMZ	Z**フラグが** "1" **の場合**	55	188
BNOT	**ビット反転**	56	189
BRK	**デバッグ割り込み**	57	189
BRK2	**デバッグ割り込み**2	58	189
BSET	**ビットセット**	59	190
BTST	**ビットテスト**	60	190
BTSTC	**ビットテスト**&**クリア**	61	191
BTSTS	**ビットテスト**&**セット**	62	191
CLIP	**クリップ**	63	192
CMP	**比較**	64	192
CMPF	**浮動小数点比較**	66	194

表 C.1　アルファベット別ページ早見表 (続き) (2 / 5)

ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
CNVIF	整数→浮動小数点数変換	67	195
DADC	キャリー付き10進加算	68	196
DADD	10進加算	69	197
DEC	デクリメント	70	198
DIV	符号付き除算	71	199
DIVF	浮動小数点除算	72	200
DIVU	符号なし除算	73	201
DIVX	符号付き除算	74	202
DSBB	ボロー付き10進減算	75	203
DSUB	10進減算	76	204
EDIV	符号付き除算	77	205
EDIVU	符号なし除算	78	206
EDIVX	符号付き除算	79	207
EMUL	符号付き乗算	80	208
EMULU	符号なし乗算	81	209
ENTER	スタックフレーム生成	82	209
EXITD	スタックフレーム解放	83	210
EXITI	割り込みスタックフレーム解放	84	210
EXTS	符号拡張	85	210
EXTZ	ゼロ拡張	86	212
FCLR	フラグのクリア	88	214
FREIT	高速割り込みからの復帰	89	214
FSET	フラグのセット	90	214
INC	インクリメント	91	215
INDEX*Type*	インデックス	92	216
INDEXB	第1、第2オペランドに加算	92	216
INDEX1	第1オペランドに加算	92	215
INDEX2	第2オペランドに加算	92	216
INT	INT命令割り込み	93	217
INTO	オーバフロー割り込み	94	217
J*Cnd*	条件分岐	95	217
JC	フラグが “1” の場合	95	217
JEQ	等しい場合	95	217
JGE	等しいか符号付きで大きい場合	95	217
JGEU	等しいか大きい場合	95	217
JGT	符号付きで大きい場合	95	217
JGTU	大きい	95	217
JLE	等しいか符号付きで小さい場合	95	217
JLEU	等しいか小さい場合	95	217
JLT	符号付きで小さい場合	95	217

表 C.1　アルファベット別ページ早見表 (続き) (3 / 5)

ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
JLTU	小さい場合	95	217
JN	負の場合	95	217
JNC	Cフラグが “0” の場合	95	217
JNE	等しくない場合	95	217
JNO	Oフラグが “0” の場合	95	217
JNZ	Zフラグが “0” の場合	95	217
JO	Oフラグが “1” の場合	95	217
JPZ	正またはゼロの場合	95	217
JZ	Zフラグが “1” の場合	95	217
JMP	無条件分岐	96	218
JMPI	間接分岐	97	219
JSR	サブルーチン分岐	98	220
JSRI	間接サブルーチン分岐	99	220
LDC	専用レジスタへの転送	100	221
LDCTX	コンテキストの復帰	101	222
LDIPL	割り込み優先レベル設定	103	223
MAX	最大値選択	104	223
MIN	最小値選択	105	225
MOV	転送	106	226
MOVA	実効アドレス転送	108	231
MOV*Dir*	4ビットデータ転送	109	231
MOVHH	src上位からdest上位へ転送	109	231
MOVHL	src上位からdest下位へ転送	109	231
MOVLH	src下位からdest上位へ転送	109	231
MOVLL	src下位からdest下位へ転送	109	231
MUL	符号付き乗算	110	232
MULF	浮動小数点乗算	111	233
MULU	符号なし乗算	112	234
MULX	丸め付き乗算	113	236
NEG	符号反転	115	236
NOP	ノーオペレーション	116	237
NOT	論理反転	117	237
OR	論理和	118	237
POP	レジスタ/メモリの復帰	119	238
POPC	専用レジスタの復帰	120	239
POPM	複数レジスタの復帰	121	239
PUSH	レジスタ/メモリ/即値の退避	122	239
PUSHA	実効アドレスの退避	124	241
PUSHC	専用レジスタの退避	125	242

表 C.1　アルファベット別ページ早見表 (続き) (4 / 5)

ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
PUSHM	複数レジスタの退避	126	242
REIT	割り込みからの復帰	127	242
RMPA	積和演算	128	243
ROLC	キャリー付き左回転	129	243
RORC	キャリー付き右回転	130	243
ROT	回転	131	244
ROUND	浮動小数点数→整数変換	132	245
RTS	サブルーチンからの復帰	133	246
SBB	ボロー付き減算	134	246
SC*Cnd*	条件設定	135	247
SCC	Cフラグが “1” の場合	135	247
SCEQ	等しい場合	135	247
SCGE	等しいか符号付きで大きい場合	135	247
SCGEU	等しいか大きい場合	135	247
SCGT	符号付きで大きい場合	135	247
SCGTU	大きい	135	247
SCLE	等しいか符号付きで小さい場合	135	247
SCLEU	等しいか小さい場合	135	247
SCLT	符号付きで小さい場合	135	247
SCLTU	小さい場合	135	247
SCN	負の場合	135	247
SCNC	Cフラグが “0” の場合	135	247
SCNE	等しくない場合	135	247
SCNO	Oフラグが “0” の場合	135	247
SCNZ	Zフラグが “0” の場合	135	247
SCO	Oフラグが “1” の場合	135	247
SCPZ	正またはゼロの場合	135	247
SCZ	Zフラグが “1” の場合	135	247
SCMPU	ストリング比較	136	248
SHA	算術シフト	137	248
SHL	論理シフト	138	249
SIN	ストリング入力	140	251
SMOVB	逆方向ストリング転送	141	251
SMOVF	順方向ストリング転送	142	251
SMOVU	ストリング転送	143	252
SOUT	ストリング出力	144	252
SSTR	ストリングストア	145	252
STC	専用レジスタから転送	146	253
STCTX	コンテキストの退避	147	254
STNZ	条件付き転送	149	254

表 C.1　**アルファベット別ページ早見表 (続き)** (5 / 5)

ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
STOP	**ストップ**	150	254
STZ	**条件付き転送**	151	255
STZX	**条件付き転送**	152	255
SUB	**ボローなし減算**	153	256
SUBF	**浮動小数点減算**	154	257
SUNTIL	**ストリングサーチ**	155	258
SWHILE	**ストリングサーチ**	156	258
TST	**テスト**	157	258
UND	**未定義命令割り込み**	158	259
WAIT	**ウェイト**	159	259
XCHG	**交換**	160	260
XOR	**排他的論理和**	161	260

表 C.2　機能別ページ早見表 (1 / 3)

機能	ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
転送	MOV	**転送**	106	226
	MOVA	**実効アドレス転送**	108	231
	MOVDir	4**ビットデータ転送**	109	231
	POP	**レジスタ/メモリの復帰**	119	238
	POPM	**複数レジスタの復帰**	121	239
	PUSH	**レジスタ/メモリ/即値の退避**	122	239
	PUSHA	**実効アドレスの退避**	124	241
	PUSHM	**複数レジスタの退避**	126	242
	STNZ	**条件付き転送**	149	254
	STZ	**条件付き転送**	151	255
	STZX	**条件付き転送**	152	255
	XCHG	**交換**	160	260
ビット処理	BCLR	**ビットクリア**	53	187
	BITINDEX	**ビットインデックス**	54	188
	BMCnd	**条件ビット転送**	55	188
	BNOT	**ビット反転**	56	189
	BSET	**ビットセット**	59	190
	BTST	**ビットテスト**	60	190
	BTSTC	**ビットテスト**&**クリア**	61	191
	BTSTS	**ビットテスト**&**セット**	62	191
シフト	ROLC	**キャリー付き左回転**	129	243
	RORC	**キャリー付き右回転**	130	243
	ROT	**回転**	131	244
	SHA	**算術シフト**	137	248
	SHL	**論理シフト**	138	249
算術演算	ABS	**絶対値**	44	180
	ADC	**キャリー付き加算**	45	180
	ADCF	**キャリーフラグの加算**	46	181
	ADD	**キャリーなし加算**	47	182
	ADSF	**サインフラグの加算**	50	185
	CLIP	**クリップ**	63	192
	CMP	**比較**	64	192
	DEC	**デクリメント**	70	198
	DIV	**符号付き除算**	71	199
	DIVU	**符号なし除算**	73	201
	DIVX	**符号付き除算**	74	202
	EDIV	**符号付き除算**	77	205
	EDIVU	**符号なし除算**	78	206
	EDIVX	**符号付き除算**	79	207

表 C.2　機能別ページ早見表 (続き) (2 / 3)

機能	ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
算術演算	EMUL	符号付き乗算	80	208
	EMULU	符号なし乗算	81	209
	EXTS	符号拡張	85	210
	EXTZ	ゼロ拡張	86	212
	INC	インクリメント	91	215
	MAX	最大値選択	104	223
	MIN	最小値選択	105	225
	MUL	符号付き乗算	110	232
	MULU	符号なし乗算	112	234
	MULX	丸め付き乗算	113	236
	NEG	符号反転	115	236
	SBB	ボロー付き減算	134	246
	SUB	ボローなし減算	153	256
10進演算	DADC	キャリー付き10進加算	68	196
	DADD	10進加算	69	197
	DSBB	ボロー付き10進減算	75	203
	DSUB	10進減算	76	204
浮動小数点演算	ADDF	浮動小数点加算	49	184
	CMPF	浮動小数点比較	66	194
	CNVIF	整数→浮動小数点数変換	67	195
	DIVF	浮動小数点除算	72	200
	MULF	浮動小数点乗算	111	233
	ROUND	浮動小数点数→整数変換	132	245
	SUBF	浮動小数点減算	154	257
積和演算	RMPA	積和演算	128	243
論理演算	AND	論理積	51	186
	NOT	論理反転	117	237
	OR	論理和	118	237
	TST	テスト	157	258
	XOR	排他的論理和	161	260
分岐	JCnd	条件分岐	95	217
	JMP	無条件分岐	96	218
	JMPI	間接分岐	97	219
	JSR	サブルーチン分岐	98	220
	JSRI	間接サブルーチン分岐	99	220
	RTS	サブルーチンからの復帰	133	246

表 C.2　機能別ページ早見表 (続き) (3 / 3)

機能	ニーモニック	内容	機能記載ページ	命令コード/サイクル数記載ページ
ストリング	SCMPU	ストリング比較	136	248
	SIN	ストリング入力	140	251
	SMOVB	逆方向ストリング転送	141	251
	SMOVF	順方向ストリング転送	142	251
	SMOVU	ストリング転送	143	252
	SOUT	ストリング出力	144	252
	SSTR	ストリングストア	145	252
	SUNTIL	ストリングサーチ	155	258
	SWHILE	ストリングサーチ	156	258
専用レジスタ操作	FCLR	フラグのクリア	88	214
	FSET	フラグのセット	90	214
	LDC	専用レジスタへの転送	100	221
	POPC	専用レジスタの復帰	120	239
	PUSHC	専用レジスタの退避	125	242
	STC	専用レジスタから転送	146	253
割り込み	BRK	デバッグ割り込み	57	189
	BRK2	デバッグ割り込み2	58	189
	FREIT	高速割り込みからの復帰	89	214
	INT	INT命令割り込み	93	217
	INTO	オーバフロー割り込み	94	217
	LDIPL	割り込み優先レベル設定	103	223
	REIT	割り込みからの復帰	127	242
	UND	未定義命令割り込み	158	259
高級言語サポート	ENTER	スタックフレーム生成	82	209
	EXITD	スタックフレーム解放	83	210
	EXITI	割り込みスタックフレーム解放	84	210
OSサポート	LDCTX	コンテキストの復帰	101	222
	STCTX	コンテキストの退避	147	254
その他	INDEXType	インデックス	92	216
	NOP	ノーオペレーション	116	237
	SCCnd	条件設定	135	247
	STOP	ストップ	150	254
	WAIT	ウェイト	159	259

1. 概要

1.1 R32C/100シリーズの特長

R32C/100シリーズは、組み込み機器を対象として開発されたシングルチップマイクロコンピュータです。

C言語に適した命令を持ち、使用頻度の高い命令を1バイトオペコードに配置していますので、アセンブリ言語を使用しても、C言語を使用しても、より少ないメモリ容量で効率の良いプログラムを開発できます。また、1クロックで実行する命令を多数持たせ、より高速な演算処理を実現しました。

豊富なアドレッシングモードに対応した108種類の命令セットを持ち、レジスタ–レジスタ、レジスタ–メモリ、メモリ–メモリ間の演算やビットおよび4ビットデータを対象とする演算ができます。

また、乗算器、浮動小数点演算器を内蔵していますので、高速な演算ができます。

(1) 特長

(a) レジスタ構成

データレジスタ	32ビット×4本(16ビットレジスタとしても使用可能。また、内2本は8ビットレジスタとしても使用可能)
アドレスレジスタ	32ビット×4本
ベースレジスタ	32ビット×2本

(b) 特長ある命令セット

C言語に適した命令(スタックフレーム操作)	: ENTER、EXITD、など
レジスタ、メモリを区別しない命令	: MOV、ADD、SUB、など
強力なビット処理命令	: BCLR、BTST、BSET、など
4ビット転送命令	: MOVLL、MOVHL、など
使用頻度の高い1バイト命令	: MOV、ADD、SUB、JMP、など
高速な1サイクル命令	: MOV、ADD、SUB、など
固定小数点乗算命令	: MULX
浮動小数点演算命令	: ADDF、SUBF、MULF、DIVF、など

(c) 4Gバイトのリニアなアドレス空間

分岐距離に応じた相対ジャンプ命令

(d) 高速な命令実行時間

最短1サイクル命令　: 108命令中36命令が1サイクル命令

(2) 性能 (動作周波数 100MHz時)

レジスタ間転送	10ns
レジスタ–メモリ間転送	20ns
レジスタ間加減算	10ns
8ビット×8ビットレジスタ間演算	20ns
16ビット×16ビットレジスタ間演算	20ns
32ビット×32ビットレジスタ間演算	20ns
16ビット÷16ビットレジスタ間演算	150ns
32ビット÷32ビットレジスタ間演算	220ns

1.2 アドレス空間

アドレス空間を図 1.1に示します。

00000000h番地~000003FFh番地と00040000h番地~0004FFFFh番地は、SFR(スペシャルファンクションレジスタ)領域です。

00000400h番地~0003FFFFh番地、00060000h番地~0007FFFFh番地、およびFFE00000h番地~FFFFFFFFh番地はメモリ領域です。品種展開では00000400h番地から番地の大きい方向に対してRAM領域を、FFFFFFFFh番地から番地の小さい方向にROM領域を拡張します。ただし、FFFFFFDCh番地~FFFFFFFFh番地は固定ベクタ領域です。

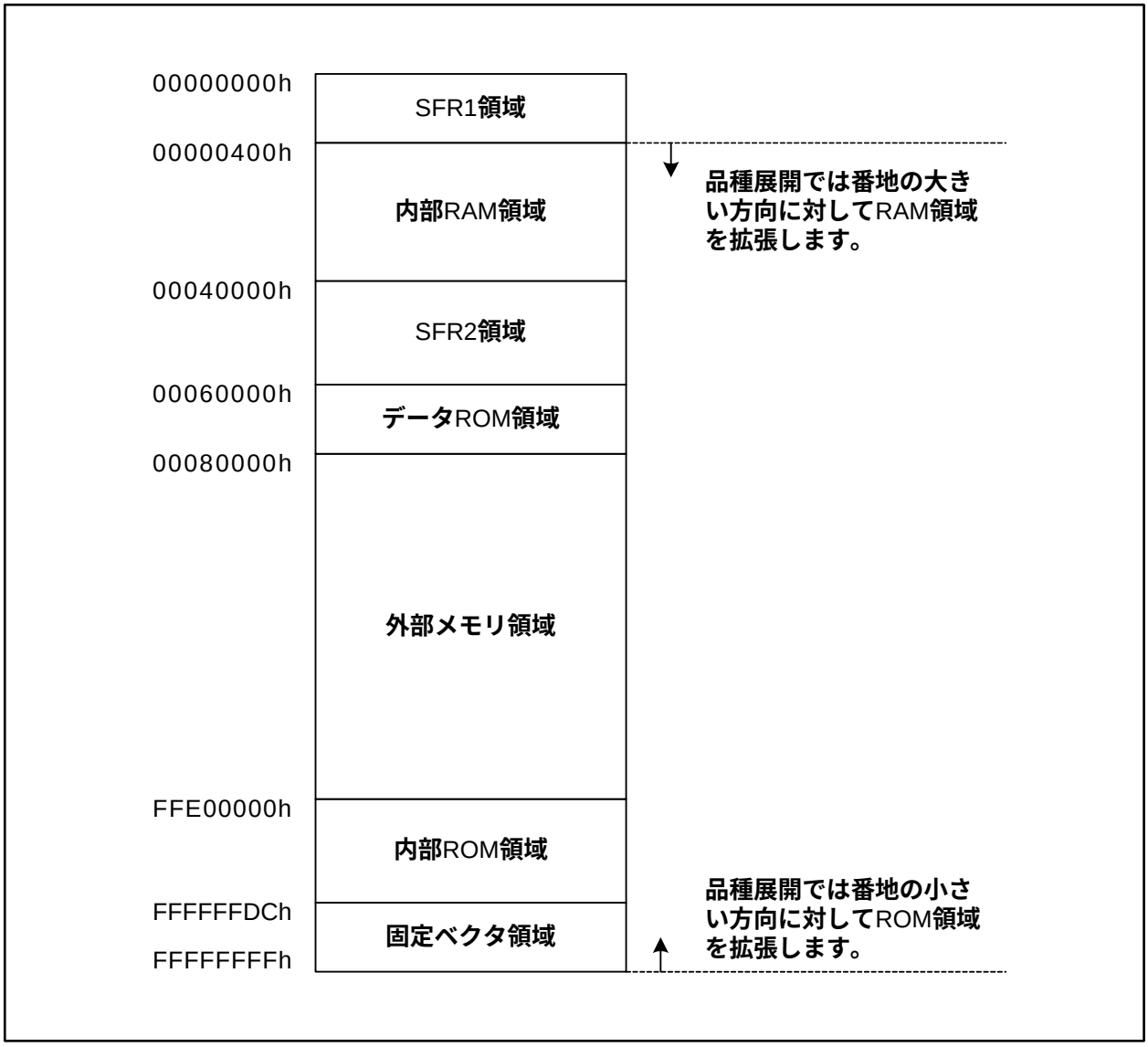


図 1.1 アドレス空間

1.3 レジスタ構成

中央演算処理装置には図 1.2 に示すレジスタがあります。これらのうち、R2R0、R3R1、R6R4、R7R5、A0、A1、A2、A3、SB、FBの10個のレジスタは2バンクあります。レジスタバンクの切り替えはレジスタバンク指定フラグ(フラグレジスタのBフラグ)で行います。

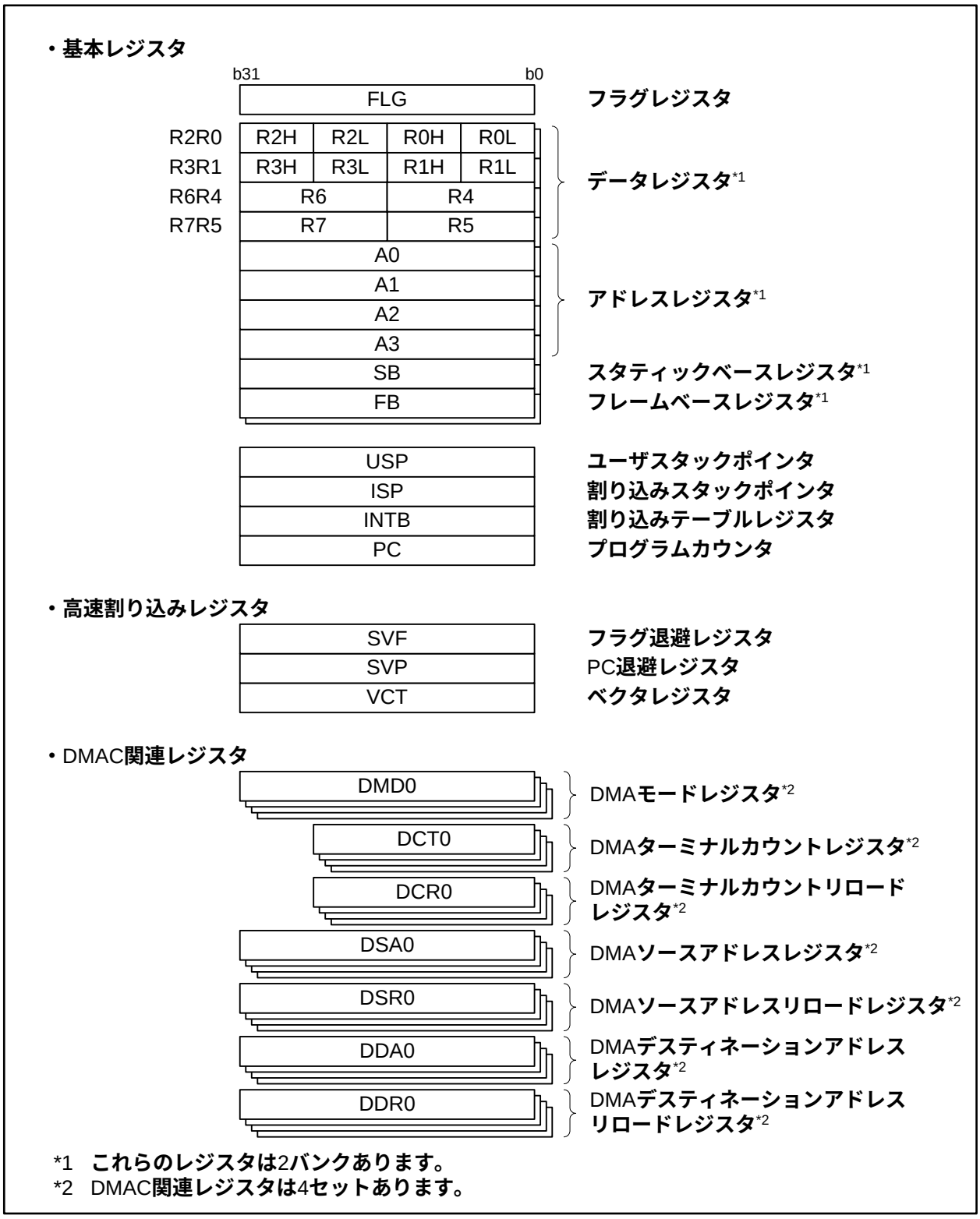


図 1.2 中央演算処理装置のレジスタ構成

1.3.1 基本レジスタ

基本レジスタには以下の9種類があります。

(1) **データレジスタ** (R2R0/R3R1/R6R4/R7R5/R0/R0H/R0L/R1/R1H/R1L/R2/R2H/R2L/R3/R3H/R3L/R4/ R5/R6/R7)

データレジスタ (R2R0/R3R1/R6R4/R7R5) は32ビットで構成されており、主に転送や算術、論理演算に使用します。

R2R0/R3R1/R6R4/R7R5は、上位 (R2/R3/R6/R7) と下位 (R0/R1/R4/R5) を別々に16ビットのデータレジスタとして使用できます。

また、R2R0/R3R1は、上位 (R2H/R3H)、中上位 (R2L/R3L)、中下位 (R0H/R1H)、下位 (R0L/R1L) を別々に8ビットのデータレジスタとしても使用できます。

(2) **アドレスレジスタ** (A0/A1/A2/A3)

アドレスレジスタ (A0/A1/A2/A3) は32ビットで構成されており、データレジスタと同等の機能をもちます。また、アドレスレジスタ間接アドレッシングおよびアドレスレジスタ相対アドレッシングに使用します。

(3) **スタティックベースレジスタ** (SB)

スタティックベースレジスタ (SB) は32ビットで構成されており、SB相対アドレッシングに使用します。

(4) **フレームベースレジスタ** (FB)

フレームベースレジスタ (FB) は32ビットで構成されており、FB相対アドレッシングに使用します。

(5) **プログラムカウンタ** (PC)

プログラムカウンタ (PC) は32ビットで構成されており、次に実行する命令の番地を示します。

(6) **割り込みテーブルレジスタ** (INTB)

割り込みテーブルレジスタ (INTB) は32ビットで構成されており、割り込みベクタテーブルの先頭番地を示します。

(7) **ユーザスタックポインタ** (USP) / **割り込みスタックポインタ** (ISP)

スタックポインタは、ユーザスタックポインタ (USP) と割り込みスタックポインタ (ISP) の2種類があり、共に32ビットで構成されています。

使用するスタックポインタ (USP/ISP) は、スタックポインタ指定フラグ (Uフラグ) によって切り替えられます。

スタックポインタ指定フラグ (Uフラグ) は、フラグレジスタ (FLG) のビット7です。

USP, ISPには4の倍数を設定してください。4の倍数を設定したほうが実行効率がよくなります。

(8) **フラグレジスタ** (FLG)

フラグレジスタ (FLG) は16ビットの予約ビットを含む32ビットのレジスタで構成されており、1ビット単位でフラグとして使用します。各フラグの機能は、「1.3.4 フラグレジスタ (FLG)」を参照してください。

1.3.2 高速割り込みレジスタ

高速割り込みレジスタは割り込み応答を高速化するために設けられたレジスタで、以下の3個のレジスタがあります。割り込みシーケンスでのレジスタ退避に、スタック領域の代わりにこれらのレジスタを使用することで割り込み応答の高速化を図っています。

(1) **フラグ退避レジスタ** (SVF)

フラグ退避レジスタ (SVF) は 16 ビットで構成されており、高速割り込み発生時にフラグレジスタ (FLG) を退避させるために使用します。

(2) PC**退避レジスタ** (SVP)

PC 退避レジスタ (SVP) は 32 ビットで構成されており、高速割り込み発生時プログラムカウンタを退避させるために使用します。

(3) **ベクタレジスタ** (VCT)

ベクタレジスタ (VCT) は 32 ビットで構成されており、高速割り込み発生時の分岐先番地を示します。

1.3.3 DMAC**関連レジスタ**

DMAC 関連のレジスタには以下の 7 種類あります。

(1) DMA**モードレジスタ** (DMD0/DMD1/DMD2/DMD3)

DMA モードレジスタ (DMD0/DMD1/DMD2/DMD3) は 32 ビットで構成されており、DMA の転送モードなどを設定するレジスタです。

(2) DMA**ターミナルカウントレジスタ** (DCT0/DCT1/DCT2/DCT3)

DMA ターミナルカウントレジスタ (DCT0/DCT1/DCT2/DCT3) は 24 ビットで構成されており、DMA の転送回数をカウントするレジスタです。

(3) DMA**ターミナルカウントリロードレジスタ** (DCR0/DCR1/DCR2/DCR3)

DMA ターミナルカウントリロードレジスタ (DCR0/DCR1/DCR2/DCR3) は 24 ビットで構成されており、DMA ターミナルカウントレジスタの初期値を設定するレジスタです。

(4) DMA**ソースアドレスレジスタ** (DSA0/DSA1/DSA2/DSA3)

DMA ソースアドレスレジスタ (DSA0/DSA1/DSA2/DSA3) は 32 ビットで構成されており、DMA の転送元のアドレスを保持するレジスタです。

(5) DMA**ソースアドレスリロードレジスタ** (DSR0/DSR1/DSR2/DSR3)

DMA ソースアドレスリロードレジスタ (DSR0/DSR1/DSR2/DSR3) は 32 ビットで構成されており、DMA ソースアドレスレジスタの初期値を設定するレジスタです。

(6) DMA **デスティネーションアドレスレジスタ** (DDA0/DDA1/DDA2/DDA3)

DMA デスティネーションアドレスレジスタ (DDA0/DDA1/DDA2/DDA3) は 32 ビットで構成されており、DMA の転送先のアドレスを保持するレジスタです。

(7) DMA**デスティネーションアドレスリロードレジスタ** (DDR0/DDR1/DDR2/DDR3)

DMAデスティネーションアドレスリロードレジスタ(DDR0/DDR1/DDR2/DDR3)は32ビットで構成されており、DMA デスティネーションアドレスレジスタの初期値を設定するレジスタです。

1.3.4 フラグレジスタ (FLG)

フラグレジスタ (FLG) の構成を図 1.3 に示します。

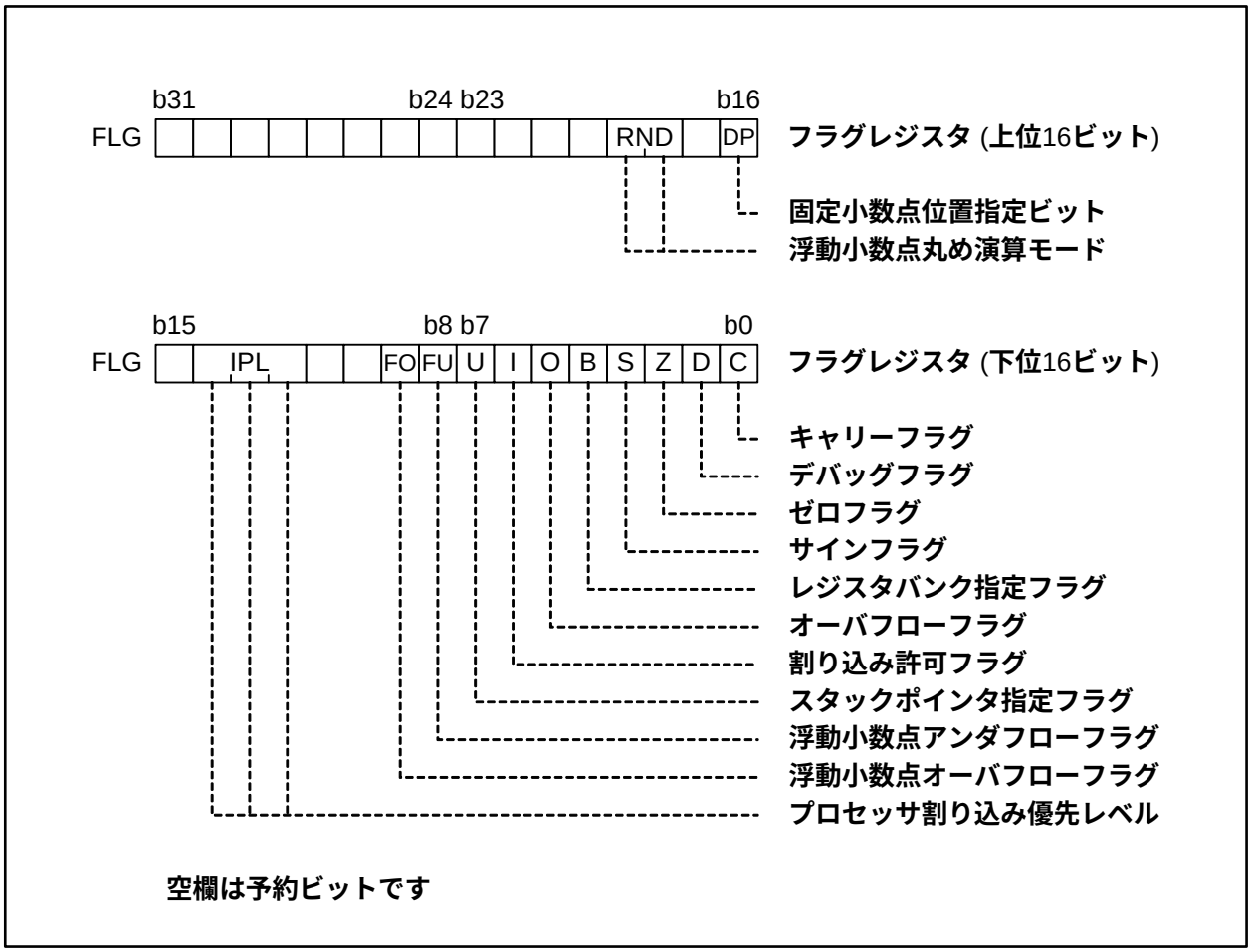


図 1.3 フラグレジスタ (FLG) の構成

各フラグの機能を以下に示します。

(1) **ビット** 0 : **キャリーフラグ** (C **フラグ**)

算術論理ユニットで発生したキャリー、ボロー、シフトアウトしたビット等を保持します。

(2) **ビット1 : デバッグフラグ (Dフラグ)**

シングルステップ割り込みを許可するフラグです。

このフラグを “1” にすると、命令実行後シングルステップ割り込みが発生します。割り込みを受け付けると、このフラグは “0” になります。

(3) **ビット2 : ゼロフラグ (Zフラグ)**

演算の結果が0のとき “1” になり、それ以外のとき “0” になります。

(4) **ビット3 : サインフラグ (Sフラグ)**

演算の結果が負のとき “1” になり、それ以外のとき “0” になります。

(5) **ビット4 : レジスタバンク指定フラグ (Bフラグ)**

レジスタバンクの選択を行います。このフラグが “0” のときレジスタバンク0が指定され、“1” のときレジスタバンク1が指定されます。

(6) **ビット5 : オーバフローフラグ (Oフラグ)**

演算の結果がオーバフローしたとき “1” になり、それ以外のとき “0” になります。

(7) **ビット6 : 割り込み許可フラグ (Iフラグ)**

マスカブル割り込みを許可するフラグです。

このフラグが “0” のとき割り込みは禁止され、“1” のとき許可されます。

割り込みを受け付けると、このフラグは “0” になります。

(8) **ビット7 : スタックポインタ指定フラグ (Uフラグ)**

このフラグが “0” のとき割り込みスタックポインタ(ISP)が指定され、“1” のときユーザスタックポインタ(USP)が指定されます。

ハードウェア割り込みを受け付けたとき、またはソフトウェア割り込み番号0~127のINT命令を実行したとき、このフラグは “0” になります。

(9) **ビット8 : 浮動小数点アンダフローフラグ(FUフラグ)**

浮動小数点演算の結果が、最小の正規化数を下回った場合(アンダフロー)、“1” になり、それ以外のとき “0” になります。

また、オペランドのデータが正規化数でも0でもない(不正入力値)場合にも、“1” になります。

(10)**ビット9 : 浮動小数点オーバフローフラグ(FOフラグ)**

浮動小数点演算の結果が、最大の正規化数を上回った場合(オーバフロー)、“1” になり、それ以外のとき “0” になります。

また、オペランドのデータが正規化数でも0でもない(不正入力値)場合にも、“1” になります。

(11)**ビット**10~**ビット**11 : **予約領域**

(12)**ビット**12~**ビット**14 : **プロセッサ割り込み優先レベル** (IPL)

プロセッサ割り込み優先レベル(IPL)は3ビットで構成されており、レベル0からレベル7までの8段階のプロセッサ割り込み優先レベルを指定します。

要求があった割り込みの割り込み要求レベルが、プロセッサ割り込み優先レベル(IPL)より大きい場合、その割り込みが許可されます。

プロセッサ割り込み優先レベル(IPL)をレベル7(111b)に設定した場合、すべての割り込みが禁止されます。

(13)**ビット**15 : **予約領域**

(14)**ビット**16 : **固定小数点位置指定ビット**(DP**ビット**)

固定小数点数の小数点位置を指定するビットです。また、固定小数点乗算の結果から、どの部分を最終演算結果として抜き出すかを指定するビットでもあります。

MULX命令で使用します。

(15)**ビット**17 : **予約領域**

(16)**ビット**18~**ビット**19 : **浮動小数点丸め演算モード**(RND)

浮動小数点丸め演算モード(RND)は2ビットで構成されており、浮動小数点演算の結果を丸める方式を指定します。

RND	**丸めモード**	**説明**
00b	**最近値**	**表現可能な最も近い値に丸めます。元の値が表現可能な2値の中間値の場合は、ビット0が "0" になるように丸めます。**
01b	**予約**	**設定しないでください。**
10b	**-∞方向**	**演算結果が-∞に近づくように丸めます。**
11b	**ゼロ方向**	**演算結果が0に近づくように丸めます。**

(17)**ビット**20~**ビット**31 : **予約領域**

1.3.5 レジスタバンク

データレジスタ(R2R0/R3R1/R6R4/R7R5)、アドレスレジスタ(A0/A1/A2/A3)、フレームベースレジスタ(FB)、およびスタティックベースレジスタ(SB)で構成されたレジスタバンクが2セットあります。レジスタバンクは、フラグレジスタ(FLG)のレジスタバンク指定フラグ(Bフラグ)で切り替えます。

レジスタバンクの構成を図 1.4に示します。

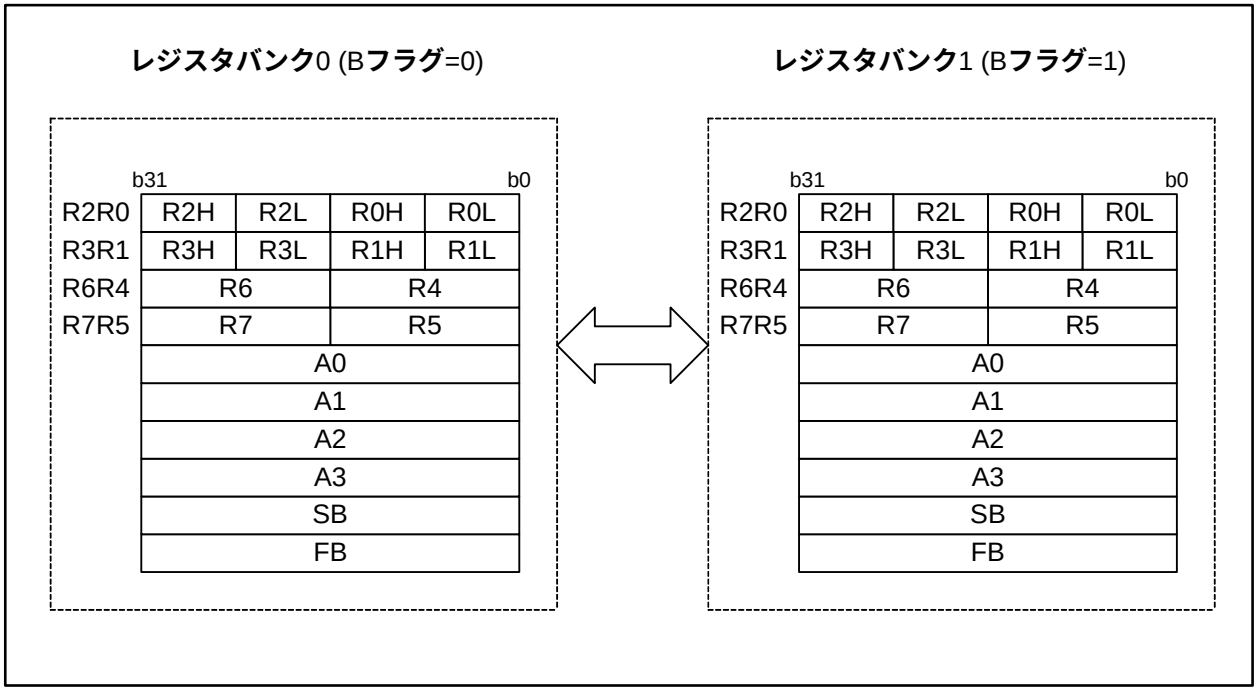


図 1.4 レジスタバンクの構成

R32C/100シリーズでは、レジスタバンク指定フラグを0にしたままで、レジスタバンク1をアクセスできるアドレッシングモードを用意しています。

1.3.6 リセット解除後の内部レジスタ値

リセット解除後の各レジスタの値を以下に示します。

- データレジスタ(R2R0/R3R1/R6R4/R7R5) : 00000000h
- アドレスレジスタ(A0/A1/A2/A3) : 00000000h
- スタティックベースレジスタ(SB) : 00000000h
- フレームベースレジスタ(FB) : 00000000h
- 割り込みテーブルレジスタ(INTB) : 00000000h
- ユーザスタックポインタ(USP) : 00000000h
- 割り込みスタックポインタ(ISP) : 00000000h
- フラグレジスタ(FLG) : 00000000h
- DMAモードレジスタ(DMD0/DMD1/DMD2/DMD3) : 00000000h
- DMAターミナルカウントレジスタ(DCT0/DCT1/DCT2/DCT3) : 不定
- DMAターミナルカウントリロードレジスタ(DCR0/DCR1/DCR2/DCR3) : 不定
- DMAソースアドレスレジスタ(DSA0/DSA1/DSA2/DSA3) : 不定
- DMAソースアドレスリロードレジスタ(DSR0/DSR1/DSR2/DSR3) : 不定
- DMAデスティネーションアドレスレジスタ(DDA0/DDA1/DDA2/DDA3) : 不定
- DMAデスティネーションアドレスリロードレジスタ(DDR0/DDR1/DDR2/DDR3) : 不定
- フラグ退避レジスタ(SVF) : 不定
- PC退避レジスタ(SVP) : 不定
- ベクタレジスタ(VCT) : 不定

1.4 データタイプ

データタイプは整数型、10進数型、固定小数点型、浮動小数点型、ビット型、ストリング型の6種類です。

1.4.1 整数型

整数には符号付きと符号なしがあります。符号付き整数の負の値は2の補数で表現します。

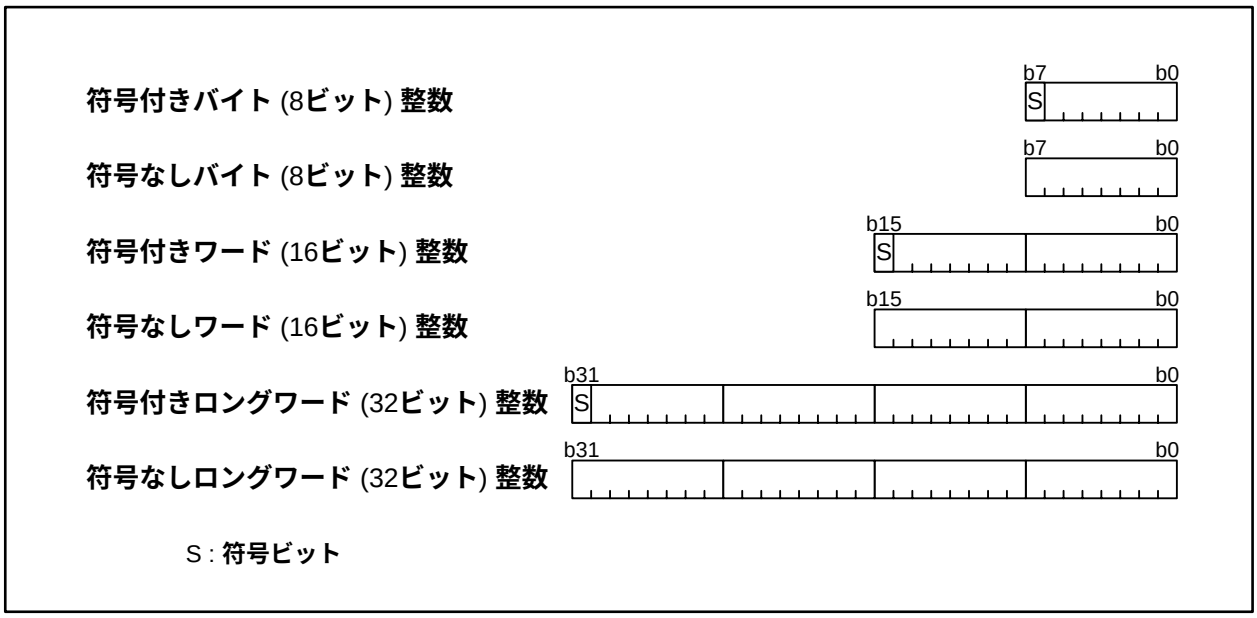


図 1.5 整数型

1.4.2 10進数型

2進化10進数(BCD)を扱うことができます。10進数は、DADC、DADD、DSBB、DSUBの4種類の命令で使用できます。

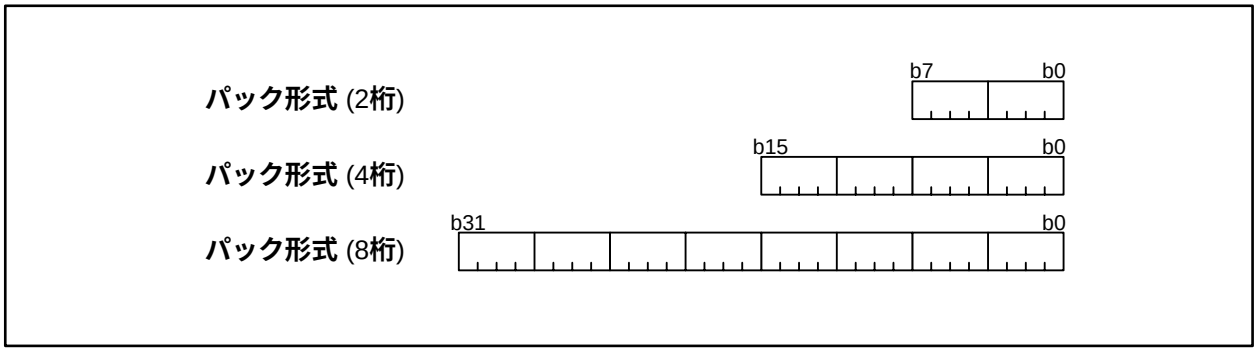


図 1.6 10進数型

1.4.3 固定小数点型

一般に固定小数点数の乗算は整数乗算命令とシフト命令で行いますが、R32C/100 シリーズでは、固定少数点専用の乗算命令MULX を搭載しています。MULX 命令でサポートする固定小数点数のフォーマットは図 1.7 のとおりです。なお、DP=1 のときは8 ビット固定小数点数はありません。

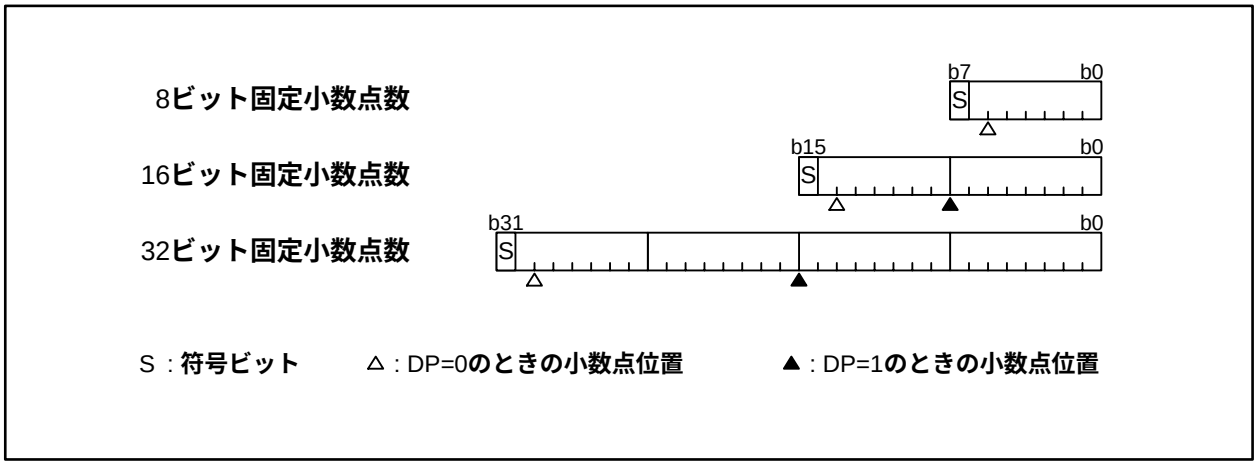


図 1.7 固定小数点型

1.4.4 浮動小数点型

R32C/100 シリーズでは、IEEE754 で規定されている単精度の浮動小数点型をサポートしています。

浮動小数点型は浮動小数点演算命令ADDF、CMPF、CNVIF、DIVF、MULF、ROUND、SUBF の7 種類の命令で使用できます。

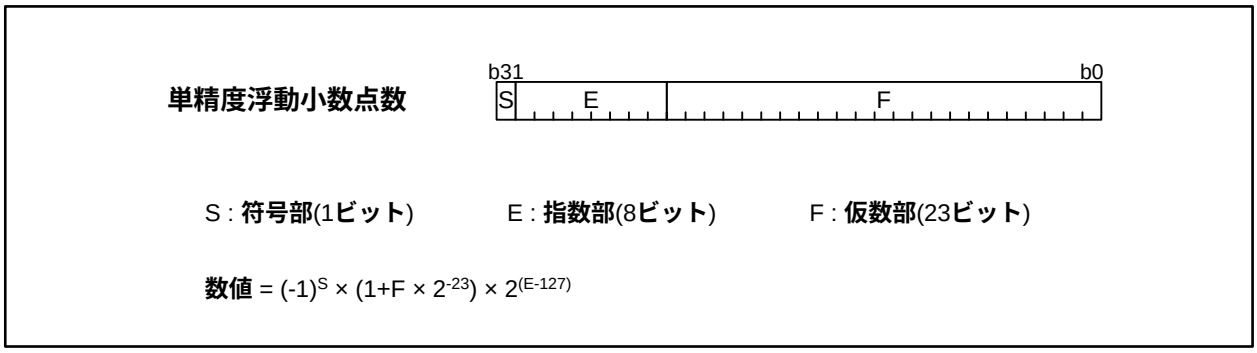


図 1.8 浮動小数点型

浮動小数点型は以下の数値をサポートしています。

- 0<E<255 (正規化数—Normal Numbers)
- E=0 かつ F=0 (ゼロ—Signed Zero)

以下の数値はサポートしていません。

- E=0 かつ F>0 (非正規化数—Subnormal Numbers)
- E=255 かつ F=0 (無限大—Infinity)
- E=255 かつ F>0 (非数—NaN: Not-a-Number)

1.4.5 ビット型

ビット型はBCLR、BSET、BNOT、BTST、BM*Cnd*、BTSTS、BTSTCの7種類の命令で使用できます。

レジスタのビットは、レジスタ名と0~7のビット番号で指定します。指定できるレジスタは8ビット長のレジスタ(R0L/R0H/R1L/R1H/R2L/R2H/R3L/R3H)だけです。

メモリのビットも同様に対象とするアドレスと0~7のビット番号で指定します。アドレス指定に使用できるアドレッシングモードは、絶対(BTST:Sのみ)、0番地相対、アドレスレジスタ間接、アドレスレジスタ相対、SB相対、FB相対の6種類です。

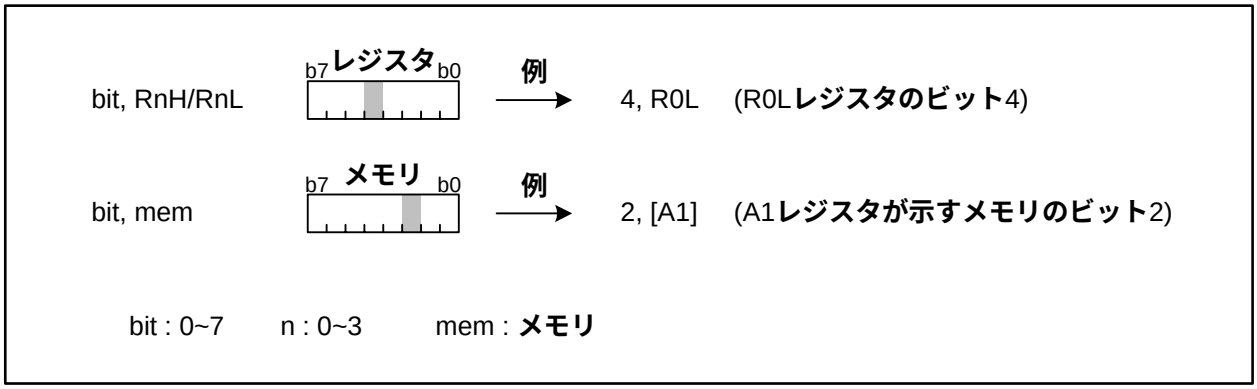


図 1.9 ビット型

1.4.6 ストリング型

ストリング型はバイト(8ビット)、ワード(16ビット)、ロングワード(32ビット)のうち、同じ種類のデータを任意の数だけ連続して並べたデータタイプです。

ストリング命令SMOVB、SMOVF、SMOVU、SCMPU、SIN、SOUT、SSTR、SUNTIL、SWHILEの9種類の命令で使用できます。

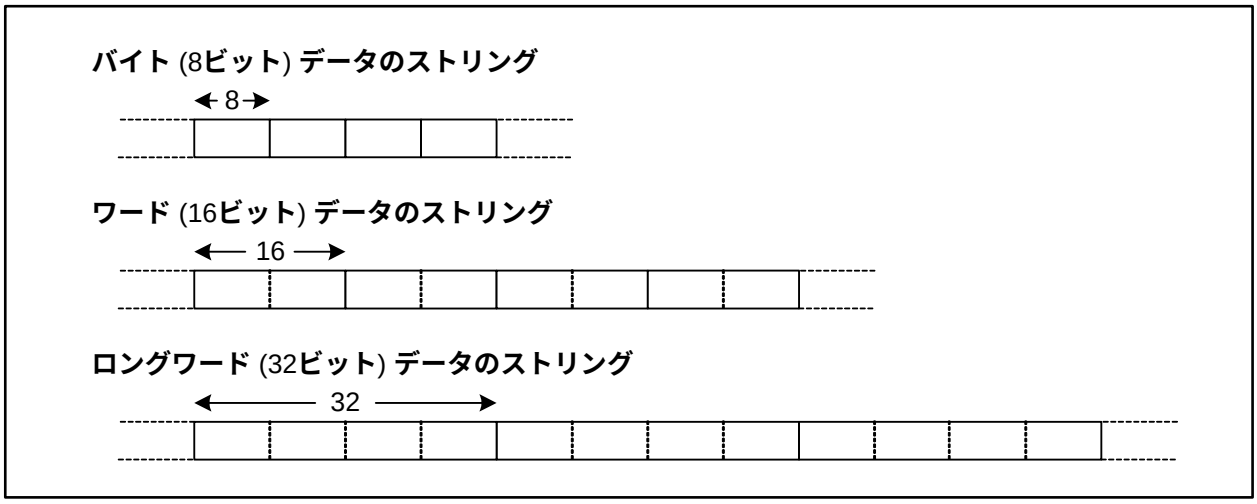


図 1.10 ストリング型

1.5 データ配置

1.5.1 レジスタ上のデータ配置

レジスタのデータサイズとビット番号の関係を図 1.11 に示します。

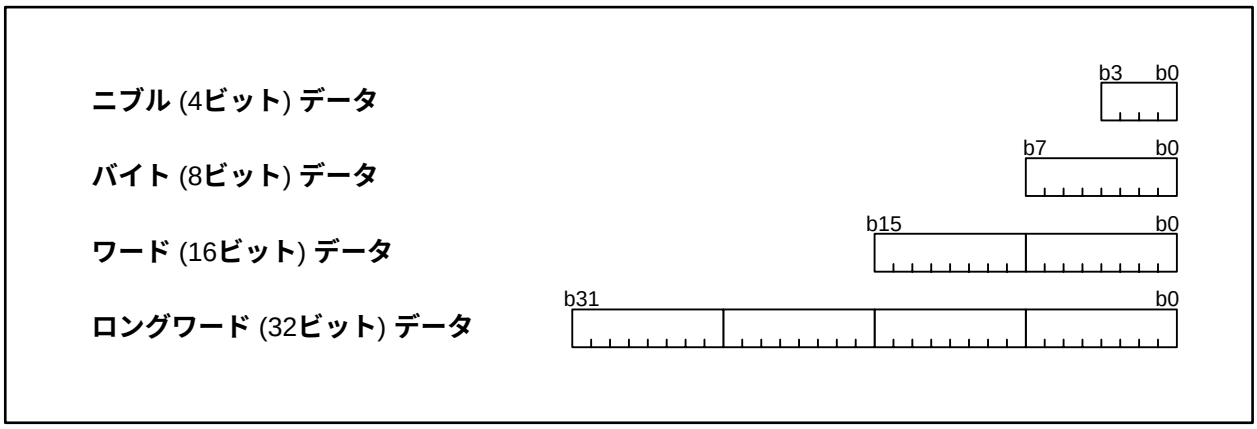


図 1.11 レジスタ上のデータ配置

1.5.2 メモリ上のデータ配置

メモリ上のデータ配置を図 1.12 に示します。

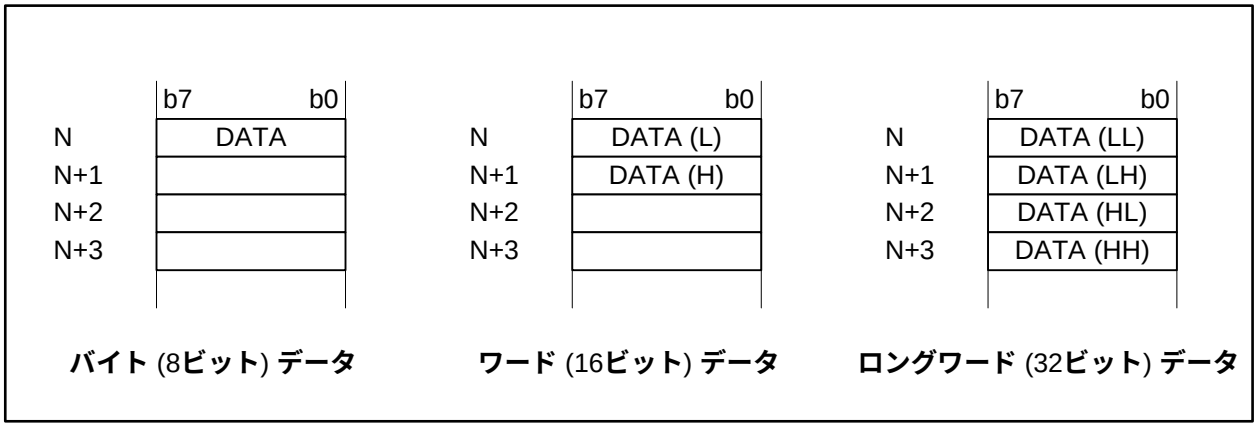


図 1.12 メモリ上のデータ配置

1.6 命令フォーマット

R32C/100シリーズの命令は図1.13のように構成され、1命令あたり最短で1バイト、最長で14バイトになります。命令の動作、オペランドの種類と数、命令の長さはオペコードの部分で決定されます。

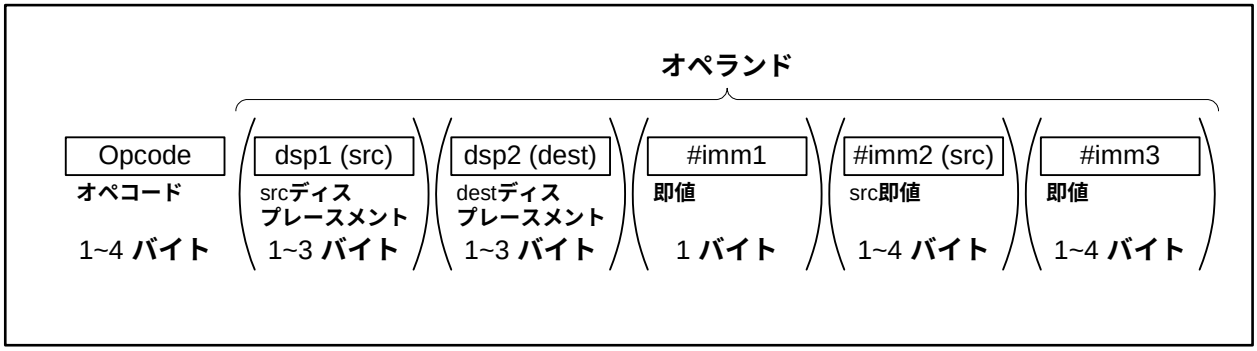


図 1.13 命令フォーマット

R32C/100 シリーズでは、使用頻度が高いと思われる命令がより少ないバイト数になるように命令コードを定義しています。また一部の命令では、使用できるアドレッシングモードを限定することでさらに命令バイト数を少なくしたフォーマットが選択できます。

命令のフォーマットには、ジェネリック、クイック、ショート、ゼロの4つの形式があります。

各フォーマットの特徴を以下に示します。

(1) ジェネリック形式(:G)

ジェネリック形式のオペコードは2~3バイトです。このオペコードには、動作およびsource (以下srcと省略) とdestination (以下destと省略)のアドレッシングモードの情報が含まれます。

命令コードはオペコード(2~3バイト)とsrcコード(0~4バイト)およびdestコード(0~3バイト)で構成されます。

(2) クイック形式(:Q)

クイック形式のオペコードは1~2バイトです。このオペコードには、動作および即値データとdestのアドレッシングモードの情報が含まれます。ただしオペコードに含まれる即値データは3ビットもしくは4ビットで表現できる数値に制限されます。

命令コードは即値データを含むオペコード(1~2バイト)とdestコード(0~3バイト)で構成されます。

(3) ショート形式(:S)

ショート形式のオペコードは1~2バイトです。このオペコードには、動作およびsrcとdestのアドレッシングモードの情報が含まれます。ただし、使用できるアドレッシングモードは制限されます。

命令コードはオペコード(1~2バイト)とsrcコード(0~4バイト)およびdestコード(0~2バイト)で構成されます。

(4) ゼロ形式(:Z)

ゼロ形式のオペコードは1バイトです。このオペコードには、動作(および即値データ)とdestのアドレッシングモードの情報が含まれます。ただし、即値データは0固定です。また、使用できるアドレッシングモードも制限されます。

命令コードはオペコード(1バイト)とdestコード(0~2バイト)で構成されます。

1.6.1 オペコード

オペコードは1~3バイト長のビット列で構成されます。オペランドのアドレッシングモードが「間接命令アドレッシング」または「バンク1レジスタ直接」のときは、オペコードの先頭に8ビットのビット列が付き、2~4バイトの長さになります。

1.6.2 オペランド

オペランドはdsp1、dsp2、#imm1、#imm2、#imm3の各フィールドで構成されます。各フィールドは命令や選択するアドレッシングモードによって省略されます。

srcコードはdsp1フィールドまたは#imm2フィールドに配置され、destコードはdsp2フィールドに配置されます。#imm1フィールドはオペコードを拡張するときに用いられます。ただし、srcコードを2つ持つ命令(CLIP、STZX)では、src1を#imm2フィールドに、src2を#imm3フィールドに配置します。

(1) dsp1フィールド

dsp1フィールドは1~3バイトで構成されます。srcのアドレッシングモードが絶対もしくは相対(スタックポインタ相対を除く)のときのみこのフィールドが存在します。

(2) dsp2フィールド

dsp2フィールドは1~3バイトで構成されます。destのアドレッシングモードが相対(プログラムカウンタ相対を除く)のときのみこのフィールドが存在します。

(3) #imm1フィールド

#imm1フィールドは1バイトで構成されます。一部の命令にのみこのフィールドが存在します。

BM*Cnd*命令	#imm1フィールドに8ビットの即値を配置し、下位4ビットで条件コードを指定します。
LDC/STC命令	#imm1フィールドに配置された8ビットの即値でDMAC関連レジスタおよびVCTレジスタを指定します。SB、FB、FLG、SP、ISP、SVF、SVP、INTBをアクセスするときは、このフィールドは存在しません。
MOV dsp:8[SP]	#imm1フィールドに配置された8ビットの即値がスタックポインタに対するディスプレースメントdsp:8を表します。
PUSHM/POPM命令	#imm1フィールドに8ビットの即値を配置し、“1” になっているビットの位置で退避/復帰するレジスタを指定します。
ROT/SHA/SHL命令	#imm1フィールドに配置された8ビットの即値がシフト数を表します。

(4) #imm2フィールド

#imm2フィールドは1/2/4バイトで構成されます。srcのアドレッシングモードが即値、符号拡張即値のときのみこのフィールドが存在します。

(5) #imm3フィールド

#imm3フィールドは1/2/4バイトで構成されます。一部の命令にのみこのフィールドが存在します。

CLIP/STZX命令	srcオペランドに2つの即値を指定します。第1の即値を#imm2フィールドに、第2の即値を#imm3フィールドに設定します。

1.7 ベクタテーブル

ベクタテーブルには、割り込みベクタテーブルがあり、割り込みベクタテーブルには、固定ベクタテーブルと可変ベクタテーブルがあります。

1.7.1 固定ベクタテーブル

固定ベクタテーブルは、固定アドレス(FFFFFFDCh~FFFFFFFFh番地)に配置された割り込みベクタテーブルです。図 1.14に固定ベクタテーブルを示します。

割り込みベクタテーブルは、1本あたり4バイトからなる9本の割り込みベクタで構成されています。各ベクタには、割り込みルーチンの先頭アドレスを設定します。

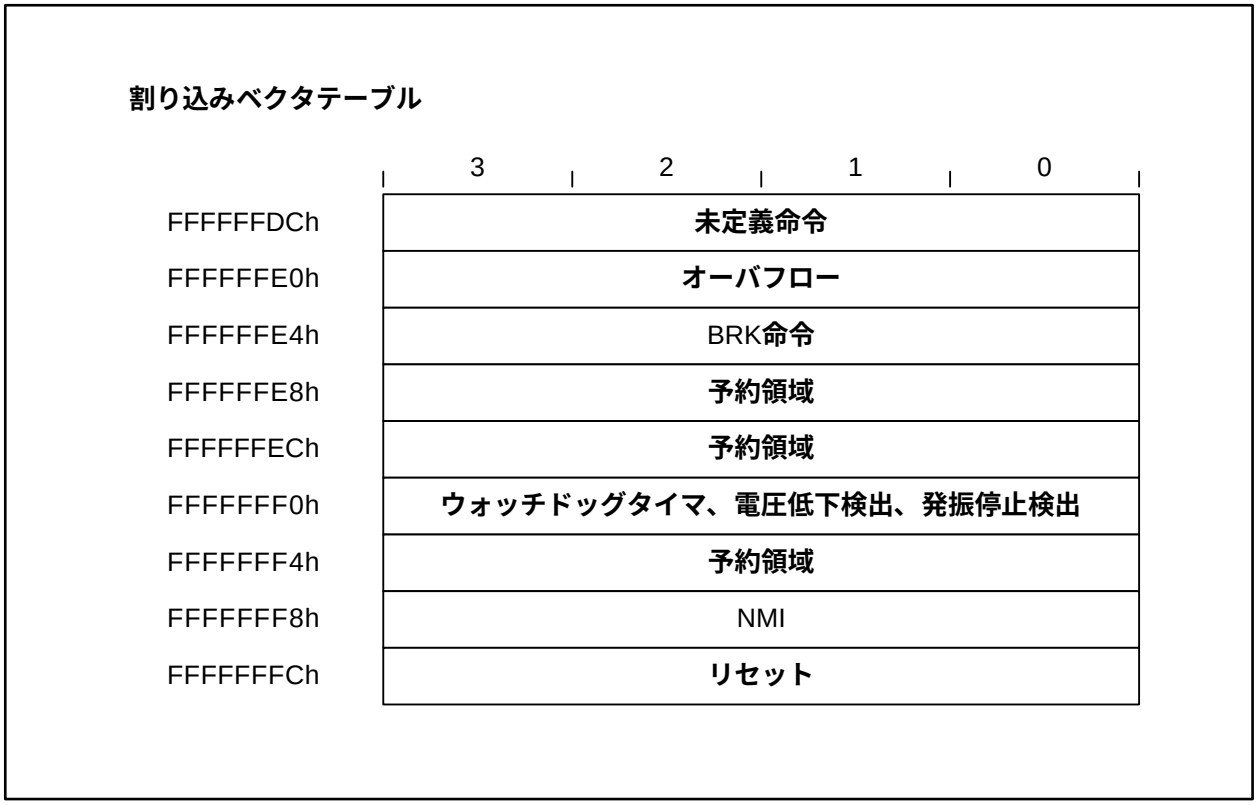


図 1.14 固定ベクタテーブル

1.7.2 可変ベクタテーブル

可変ベクタテーブルは、テーブルの配置アドレスを変えることができるベクタテーブルです。可変ベクタテーブルは、割り込みテーブルレジスタ (INTB) の内容で示された値を先頭アドレス (IntBase) とする 1K バイトの割り込みベクタテーブルです。図 1.15 に可変ベクタテーブルを示します。

可変ベクタテーブルは、1 本あたり 4 バイトからなる 256 本の割り込みベクタで構成されています。各ベクタ領域には、割り込み処理ルーチンの先頭アドレスを設定します。また、ベクタごとにソフトウェア割り込み番号 (0~255) があり、INT 命令では、このソフトウェア割り込み番号を使用します。

内蔵周辺機能の割り込み (周辺機能割り込み) も可変ベクタテーブルに割り当てられます。周辺機能割り込みの割り込みベクタは、ソフトウェア割り込み番号の 0 番側から割り当てられますが、品種によって数が異なることがあります。よって、INT 命令割り込みを使用する場合、ソフトウェア番号割り込み番号 255 から使用することを推奨します。

INT 命令割り込みに使用するスタックポインタ (SP) は、ソフトウェア割り込み番号によって異なります。

ソフトウェア割り込み番号 0~127 では、割り込み要求受け付け時にスタックポインタ指定フラグ (U フラグ) を退避し、U フラグを “0” にして割り込みスタックポインタ (ISP) を選択した後、割り込みシーケンスを実行します。割り込み処理ルーチンから復帰するときに割り込み要求受け付け前の U フラグが復帰されます。

ソフトウェア割り込み番号 128~255 では、スタックポインタは切り替わりません。

周辺機能割り込みでは、ソフトウェア割り込み番号に関係なく、割り込み要求受け付け時に割り込みスタックポインタ (ISP) を選択します。

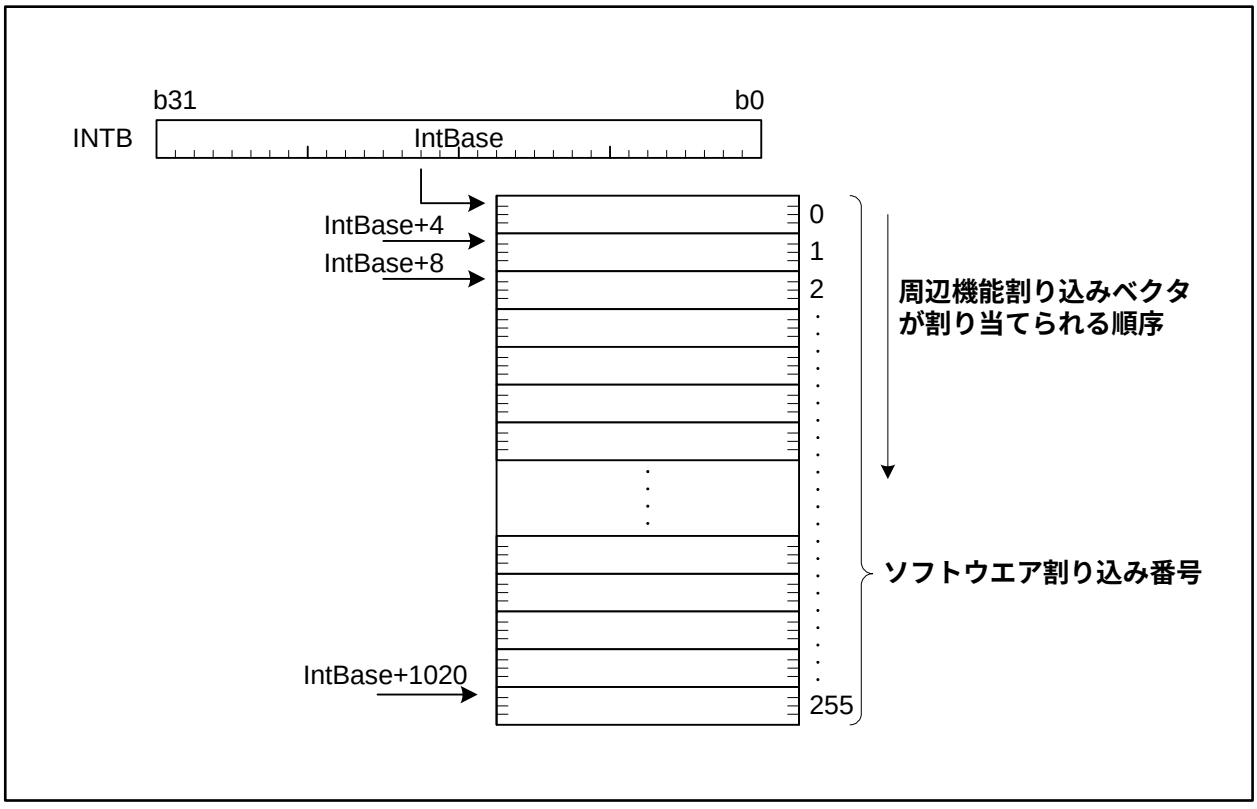


図 1.15 可変ベクタテーブル

2. アドレッシングモード

2.1 アドレッシングモード

本章ではアドレッシングモードを示す記号、動作についてアドレッシングモードごとに説明しています。アドレッシングモードは、以下に示す5つのタイプがあります。

(1) 一般命令アドレッシング

汎用レジスタやメモリをアクセスする最も一般的なアドレッシングです。

- レジスタ直接
- 即値
- 符号拡張即値
- 0番地相対
- アドレスレジスタ間接
- アドレスレジスタ相対
- SB相対
- FB相対

(2) 間接命令アドレッシング

メモリに書かれた情報を元にメモリをアクセスするアドレッシングです。一般命令アドレッシングをサポートするほぼすべての命令で使用できます。

- 0番地相対間接
- アドレスレジスタ二段間接
- アドレスレジスタ相対間接
- SB相対間接
- FB相対間接

(3) 拡張命令アドレッシング

一般命令アドレッシングでは直接アクセスできないレジスタやメモリをアクセスしたり、コードサイズを削減するために拡張されたアドレッシングです。

- バンク1レジスタ直接
- 短縮即値
- スタックポインタ相対

(4) 特定命令アドレッシング

専用レジスタやフラグ、メモリをアクセスするアドレッシング、および分岐に使用するアドレッシングです。一部の命令でのみサポートしています。

- 専用レジスタ直接
- FLG直接
- 絶対
- プログラムカウンタ相対

(5) ビット命令アドレッシング

汎用レジスタやメモリをビット単位でアクセスするアドレッシングです。

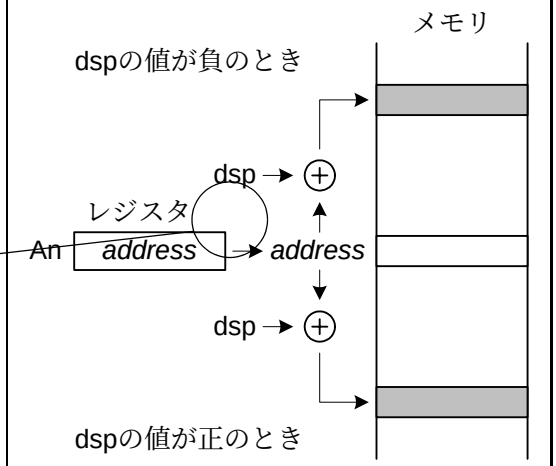
- レジスタ直接
- 絶対
- 0番地相対
- アドレスレジスタ間接
- アドレスレジスタ相対
- SB相対
- FB相対

2.2 本章の見方

本章の見方を以下に実例をあげて示します。

(1)
(2)
(3)
(4)

アドレスレジスタ相対	
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]	**アドレスレジスタ**(A0/A1/A2/A3)**の値とディスプレースメント**(dsp)**の値を符号付きで加算した結果の下位**32**ビットが演算対象の実効アドレスとなります。** **実効アドレスの範囲は、**00000000h~ FFFFFFFFh**です。**



(1) 名称

アドレッシングの名称です。

(2) 記号

アドレッシングモードを示す記号です。

“:3”、“:4”、“:8”、“:16”、“:24”、“:32” は、直前の値の有効ビット数を示します。マニュアルの記載上、有効ビット数を明記する必要があるために記載していますが、通常は記述する必要はありません。“ dsp ”などの記号は数値やシンボルに置き換えて記述してください。

(3) 解説

動作、実効アドレスの範囲を説明します。

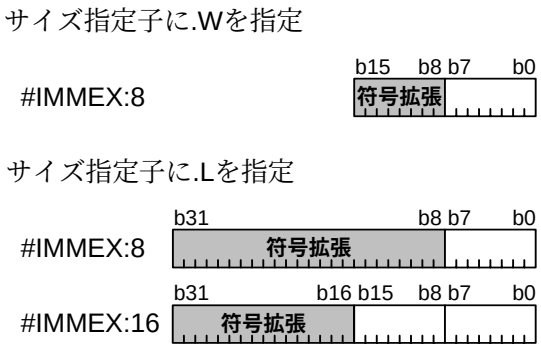
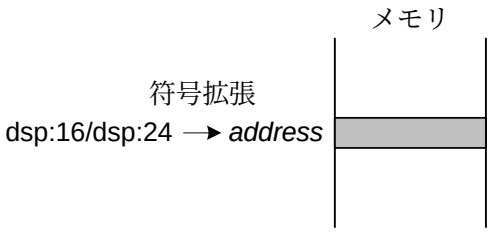
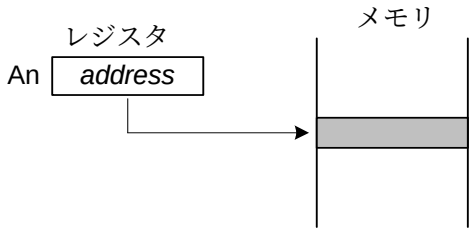
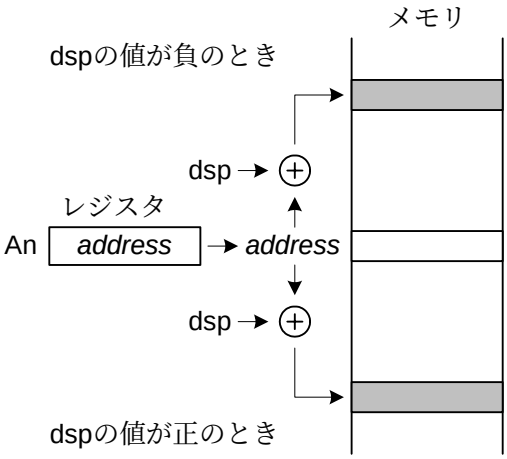
(4) 動作図

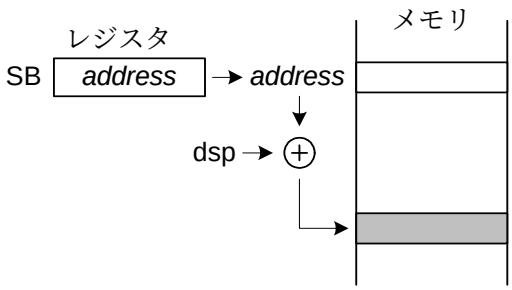
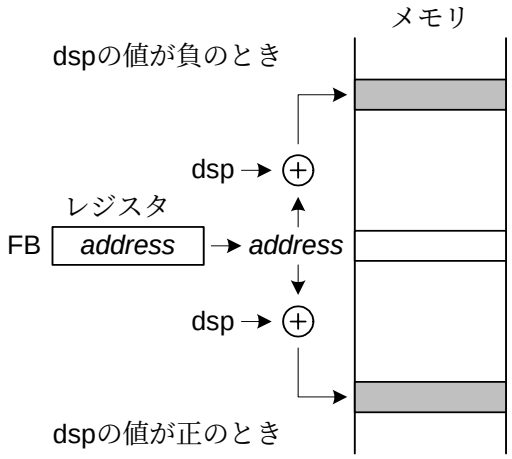
動作を図で説明します。

2.3 一般命令アドレッシング

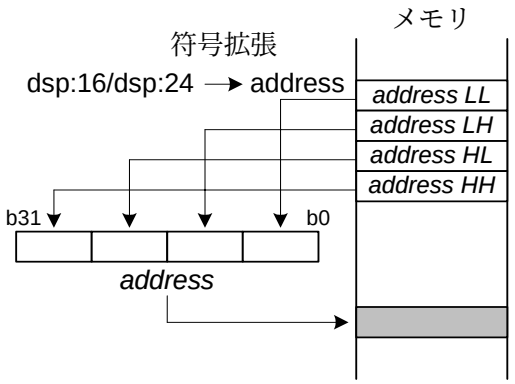
汎用レジスタやメモリをアクセスする最も一般的なアドレッシングです。ほとんどの命令で使用できます。

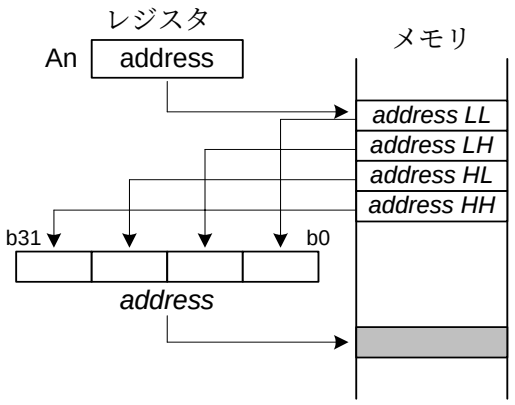
レジスタ直接		
R0L R0H R1L R1H R2L R2H R3L R3H R0 R1 R2 R3 R4 R5 R6 R7 R2R0 R3R1 R6R4 R7R5 A0 A1 A2 A3 R3R1R2R0 R7R5R6R4 A1A0 A3A2	**指定したレジスタが演算の対象となります。**	**レジスタ** R0L/R1L R0H/R1H R2L/R3L R2H/R3H R0/R1/ R4/R5 R2/R3/ R6/R7 R2R0/R3R1/ R6R4/R7R5/ A0/A1/A2/A3 R3R1R2R0/ R7R5R6R4/ A1A0/A3A2
即値		
#IMM #IMM:8 #IMM:16 #IMM:32	#IMM **で示した即値が演算の対象となります。**	#IMM:8 #IMM:16 #IMM:32

符号拡張即値		
#IMMEX #IMMEX:8 #IMMEX:16	#IMMEXで示した即値をサイズ指定子で指定されたビット長に符号拡張した結果が演算の対象となります。	
0番地相対		
dsp:16 dsp:24	ディスプレースメント(dsp)で示した値を符号拡張した結果が演算対象の実効アドレスとなります。 実効アドレスの範囲は、dsp:16では00000000h~00007FFFhと、FFFF8000h~FFFFFFFFhで、dsp:24では00000000h~007FFFFFhと、FF800000h~FFFFFFFFhです。	
アドレスレジスタ間接		
[A0] [A1] [A2] [A3]	アドレスレジスタ(A0/A1/A2/A3)の値が演算対象の実効アドレスとなります。 実効アドレスの範囲は、 00000000h~ FFFFFFFFhです。	
アドレスレジスタ相対		
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]	アドレスレジスタ(A0/A1/A2/A3)の値とディスプレースメント(dsp)の値を符号付きで加算した結果の下位32ビットが演算対象の実効アドレスとなります。 実効アドレスの範囲は、 00000000h~ FFFFFFFFhです。	

SB相対		
dsp:8[SB] dsp:16[SB] dsp:24[SB]	スタティックベースレジスタ(SB)の値とディスプレースメント(dsp)の値を符号なしで加算した結果の下位32ビットが演算対象の実効アドレスとなります。 実効アドレスの範囲は、 00000000h~ FFFFFFFFhです。	
FB相対		
dsp:8[FB] dsp:16[FB]	フレームベースレジスタ(FB)の値とディスプレースメント(dsp)の値を符号付きで加算した結果の下位32ビットが演算対象の実効アドレスとなります。 実効アドレスの範囲は、 00000000h~ FFFFFFFFhです。	

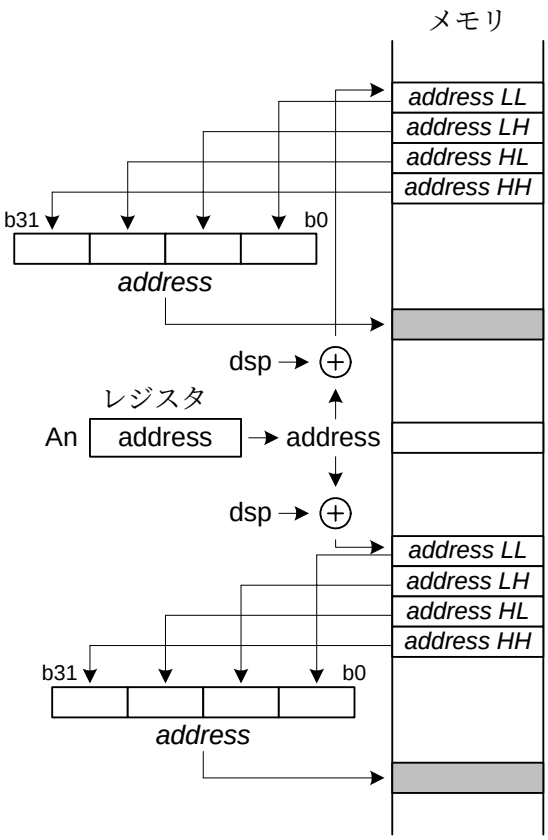
2.4 間接命令アドレッシング

<table>
<tr><td colspan="2">0番地相対間接</td><td rowspan="2"> 
</td></tr>
<tr><td>[dsp:16]
[dsp:24]</td><td>0番地相対アドレッシングで指定されたアドレスにある4バイトの値が演算対象の実効アドレスとなります。

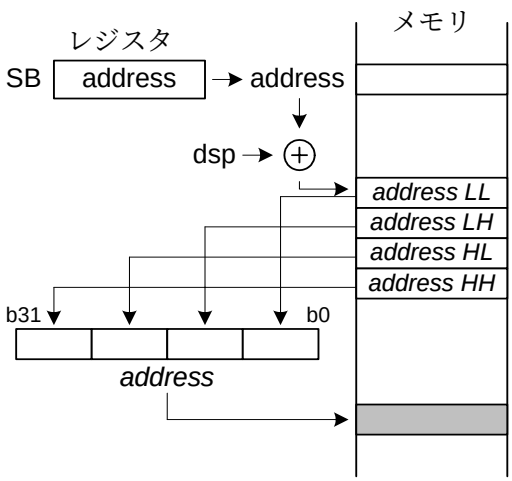
実効アドレスの範囲は、
00000000h~ FFFFFFFFhです。</td></tr>
<tr><td colspan="2">アドレスレジスタ二段間接</td><td rowspan="2"> 
</td></tr>
<tr><td>[[A0]]
[[A1]]
[[A2]]
[[A3]]</td><td>アドレスレジスタ間接で指定されたアドレスにある4バイトの値が演算対象の実効アドレスとなります。

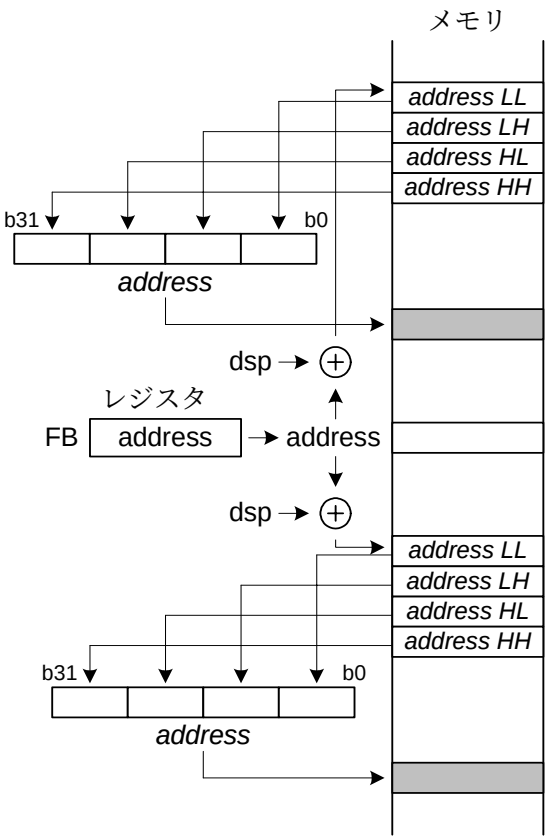
実効アドレスの範囲は、
00000000h~ FFFFFFFFhです。</td></tr>
</table>

アドレスレジスタ相対間接	
[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]]	**アドレスレジスタ相対で指定されたアドレスにある4バイトの値が演算対象の実効アドレスとなります。** **実効アドレスの範囲は、** 00000000h~ FFFFFFFFh**です。**

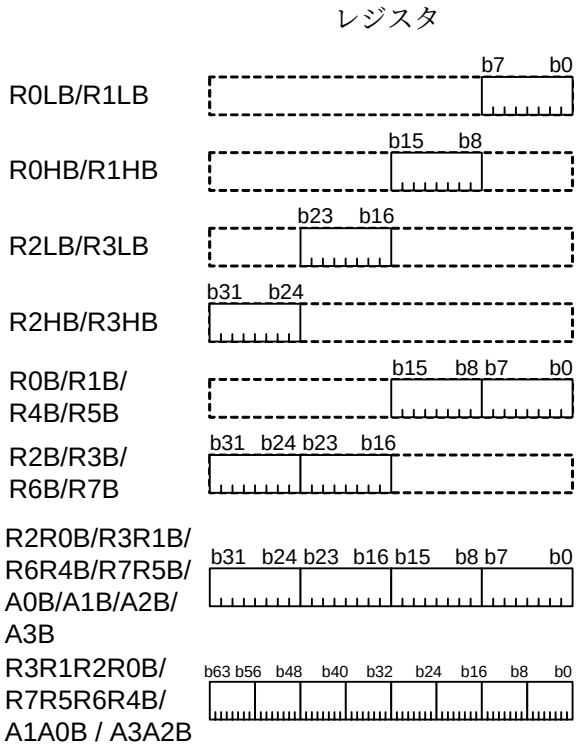
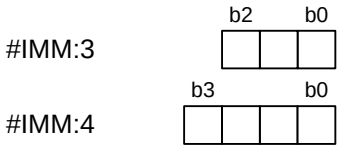


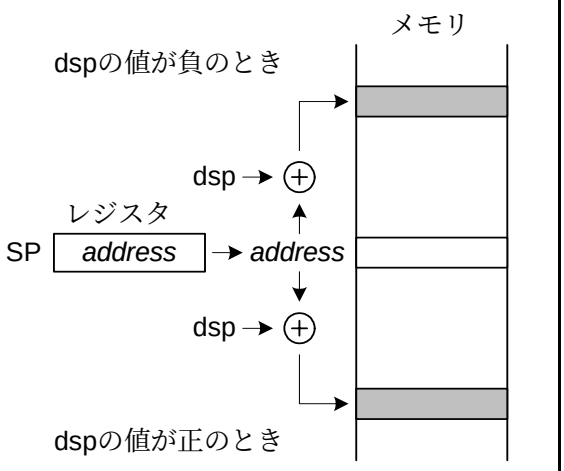
SB**相対間接**	
[dsp:8[SB]] [dsp:16[SB]] [dsp:24[SB]]	SB**相対で指定されたアドレスにある4バイトの値が演算対象の実効アドレスとなります。** **実効アドレスの範囲は、** 00000000h~ FFFFFFFFh**です。**



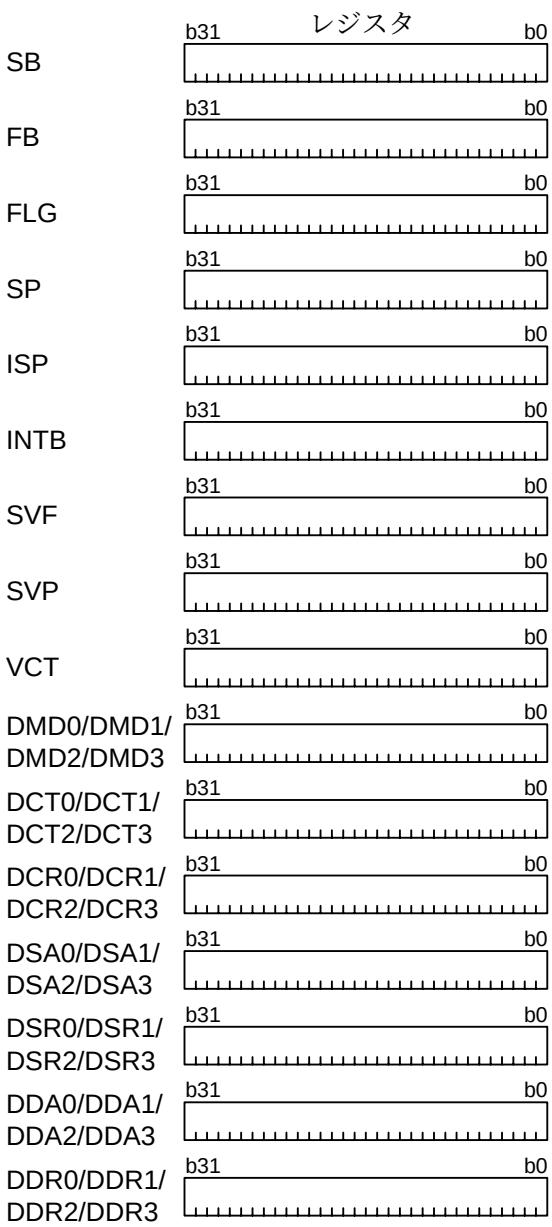
FB相対間接		
[dsp:8[FB]] [dsp:16[FB]]	FB相対で指定されたアドレスにある4バイトの値が演算対象の実効アドレスとなります。 実効アドレスの範囲は、 00000000h~ FFFFFFFFhです。	

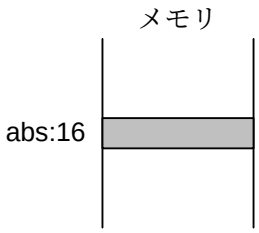
2.5 拡張命令アドレッシング

バンク1レジスタ直接		
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	指定したバンク1レジスタが演算の対象となります。 Bフラグの値に関わらず、常にバンク1レジスタ直接が指定できます。	
短縮即値		
#0 #IMM:3 #IMM:4	ゼロ、または#IMMで示した即値が演算の対象となります。 ゼロフォーマットや、クイックフォーマットで使用されます。	

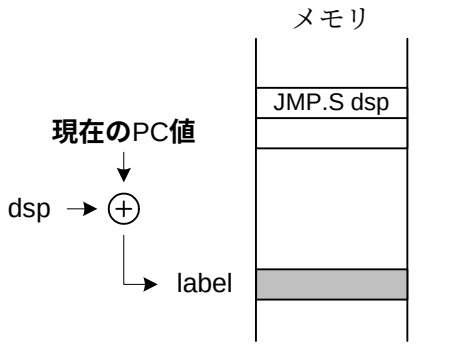
スタックポインタ相対		
dsp:8[SP]	スタックポインタ(SP)の値とディスプレースメント(dsp)の値を符号付きで加算した結果の下位32ビットが演算対象の実効アドレスとなります。 実効アドレスの範囲は、 00000000h~ FFFFFFFFhです。 このアドレッシングはMOV命令で使用できます。	

2.6 特定命令アドレッシング

専用レジスタ直接		
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	**指定した専用レジスタが演算の対象となります。** **SPを指定した場合、Uフラグで示すスタックポインタが対象となります。** **このアドレッシングは**ADD:Q、LDC、POPC、PUSHC、STC**命令で使用できます。**	
FLG**直接**		
U I O B S Z D C	**指定したフラグが演算の対象となります。** **このアドレッシングは**FCLR、FSET**命令で使用できます。**	

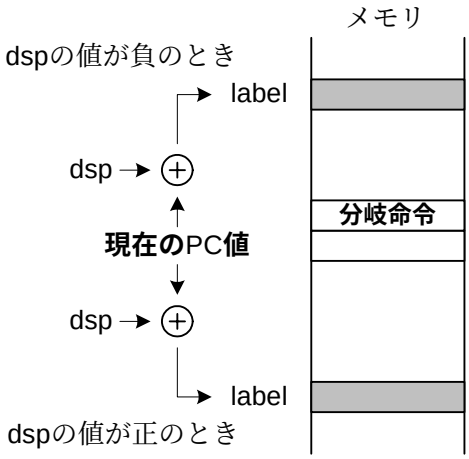
<table>
<tr><td colspan="2">絶対</td><td rowspan="2"> 
</td></tr>
<tr><td>abs:16</td><td>absで示した値が演算対象の実効アドレスとなります。

実効アドレスの範囲は、
00000000h~0000FFFFhです。

このアドレッシングはLDCTX、
STCTX命令で使用できます。</td></tr>
<tr><td colspan="2">プログラムカウンタ相対</td><td rowspan="2"> 

+1≦dsp≦+8
現在のPC値は「分岐命令の番地+1」です。</td></tr>
<tr><td>label
(dsp:3)</td><td>分岐距離指定子(.length)が “.S” の場合
現在のプログラムカウンタ(PC)の値とディスプレースメント(dsp)の値を符号なしで加算した結果の下位32ビットが実効アドレスとなります。

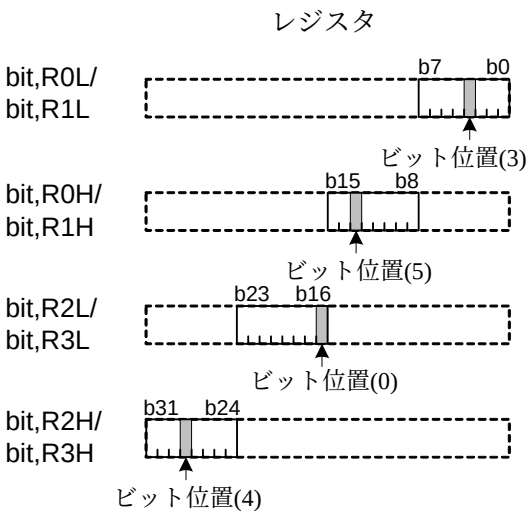
このアドレッシングは、JMP命令で使用できます。</td></tr>
<tr><td>label
(dsp:8)
(dsp:16)
(dsp:24)</td><td>分岐距離指定子(.length)が “.B” または “.W”、“.A” の場合
現在のプログラムカウンタ(PC)の値とディスプレースメント(dsp)の値を符号付きで加算した結果の下位32ビットが実効アドレスとなります。

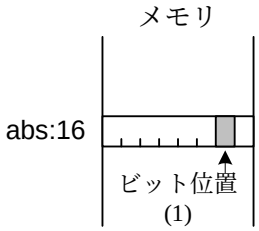
このアドレッシングは、JCnd、
JMP、JSR命令で使用できます。</td><td> 

“.B” のとき、-128≦dsp≦+127
“.W” のとき、-32768≦dsp≦+32767
“.A” のとき、-8388608≦dsp≦+8388607
現在のPC値は「分岐命令の番地+1」です。</td></tr>
</table>

2.7 ビット命令アドレッシング

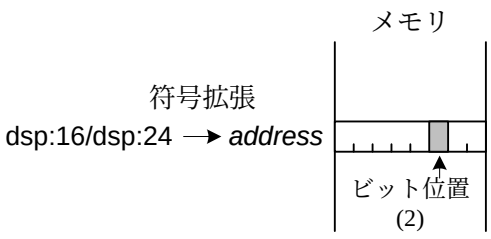
このアドレッシングは以下の命令で使用できます。
BCLR、BSET、BNOT、BTST、BM*Cnd*、BTSTC、BTSTS

<table>
<tr><td colspan="2">レジスタ直接</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>指定したレジスタのビットが演算の対象となります。

ビット位置(bit)には0~7が指定できます。</td></tr>
<tr><td colspan="2">絶対</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,abs:16</td><td>absで示した値が演算対象の実効アドレスとなります。

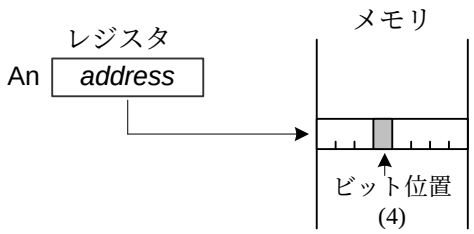
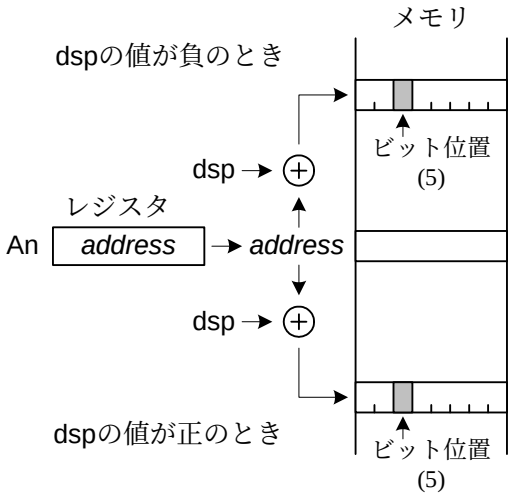
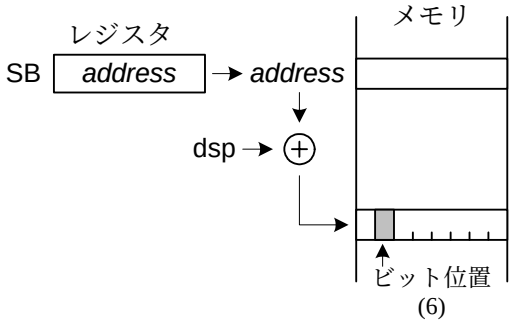
実効アドレスの範囲は、
00000000h~0000FFFFhです。

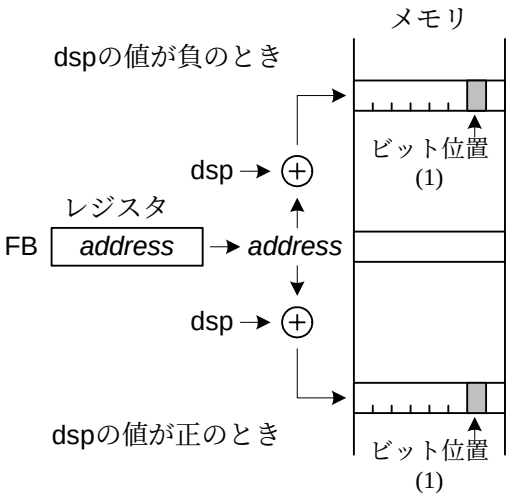
ビット位置(bit)には0~7が指定できます。

このアドレッシングはBTST命令でのみ使用できます。</td></tr>
<tr><td colspan="2">0番地相対</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,dsp:16
bit,dsp:24</td><td>ディスプレースメント(dsp)で示した値を符号拡張した結果が示すアドレスのビットが演算対象となります。

指定できるアドレスの範囲は、
dsp:16では00000000h~00007FFFhと、FFFF8000h~FFFFFFFFhで、
dsp:24では00000000h~007FFFFFhと、FF800000h~FFFFFFFFhです。

ビット位置(bit)には0~7が指定できます。</td></tr>
</table>

アドレスレジスタ間接		
bit,[A0] bit,[A1] bit,[A2] bit,[A3]	アドレスレジスタ(A0/A1/A2/A3)の値が示すアドレスのビットが演算対象となります。 指定できるアドレスの範囲は00000000h~FFFFFFFFh番地です。 ビット位置(bit)には0~7が指定できます。	
アドレスレジスタ相対		
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]	アドレスレジスタ(A0/A1/A2/A3)の値とディスプレースメント(dsp)の値を符号付きで加算した結果の下位32ビットが示すアドレスのビットが演算対象となります。 指定できるアドレスの範囲は、00000000h~FFFFFFFFhです。 ビット位置(bit)には0~7が指定できます。	
SB相対		
bit,dsp:8[SB] bit,dsp:16[SB] bit,dsp:24[SB]	スタティックベースレジスタ(SB)の値とディスプレースメント(dsp)の値を符号なしで加算した結果の下位32ビットが示すアドレスのビットが演算対象となります。 指定できるアドレスの範囲は、00000000h~ FFFFFFFFhです。 ビット位置(bit)には0~7が指定できます。	

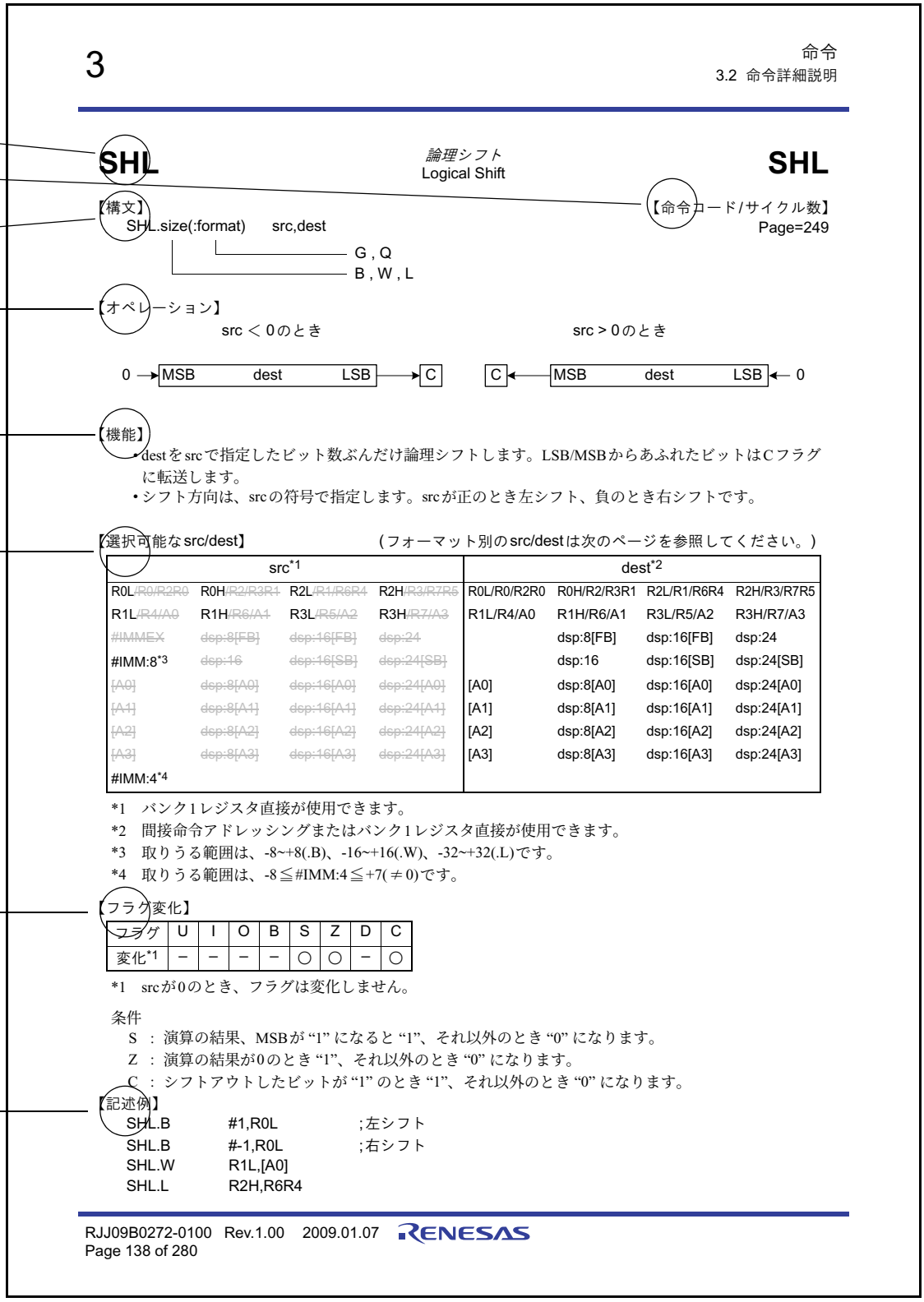
FB 相対		
bit,dsp:8[FB] bit,dsp:16[FB]	**フレームベースレジスタ** (FB) **の値とディスプレースメント** (dsp) **の値を符号付きで加算した結果の下位** 32 **ビットが示すアドレスのビットが演算対象となります。** **指定できるアドレスの範囲は、** 00000000h~ FFFFFFFFh **です。** **ビット位置** (bit) **には** 0~7 **が指定できます。**	

3. 命令

3.1 本章の見方

本章では、構文、オペレーション、機能、選択可能な src/dest、フラグ変化、記述例について命令ごとに説明しています。

本章の見方について以下に実例をあげて示します。



3

命令
3.2 命令詳細説明

(1) **SHL** *論理シフト* Logical Shift **SHL**

(2) 【命令コード/サイクル数】 Page=249

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

(4) 【オペレーション】

src ＜ 0のとき

src > 0のとき

(5) 【機能】
- destをsrcで指定したビット数ぶんだけ論理シフトします。LSB/MSBからあふれたビットはCフラグに転送します。
- シフト方向は、srcの符号で指定します。srcが正のとき左シフト、負のとき右シフトです。

(6) 【選択可能なsrc/dest】 （フォーマット別のsrc/destは次のページを参照してください。）

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 バンク1レジスタ直接が使用できます。
*2 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*3 取りうる範囲は、-8~+8(.B)、-16~+16(.W)、-32~+32(.L)です。
*4 取りうる範囲は、-8≦#IMM:4≦+7(≠0)です。

(7) 【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	−	−	−	−	○	○	−	○

*1 srcが0のとき、フラグは変化しません。

条件
S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： シフトアウトしたビットが “1” のとき “1”、それ以外のとき “0” になります。

(8) 【記述例】

```
SHL.B   #1,R0L      ;左シフト
SHL.B   #-1,R0L     ;右シフト
SHL.W   R1L,[A0]
SHL.L   R2H,R6R4
```

RJJ09B0272-0100 Rev.1.00 2009.01.07
Page 138 of 280

(1) **ニーモニック**

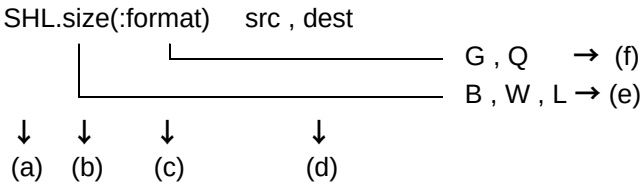
本ページで説明するニーモニックを示しています。

(2) **命令コード/サイクル数**

命令コードとサイクル数の記載ページを示しています。
命令コードとサイクル数については、このページを参照してください。

(3) **構文**

命令の構文を記号で示しています。



(a) **ニーモニック** SHL

ニーモニックを記述します。

(b) **サイズ指定子** .size

取り扱うデータサイズを記述します。指定できるサイズを以下に示します。

.B バイト(8ビット)
.W ワード(16ビット)
.L ロングワード(32ビット)

サイズ指定子をもたない命令もあります。

(c) **命令フォーマット指定子** :format

命令のフォーマットを記述します。省略することもできます。
指定できる命令フォーマットを以下に示します。

:G ジェネリック形式
:Q クイック形式
:S ショート形式
:Z ゼロ形式

命令フォーマット指定子をもたない命令もあります。

(d) **オペランド** src, dest

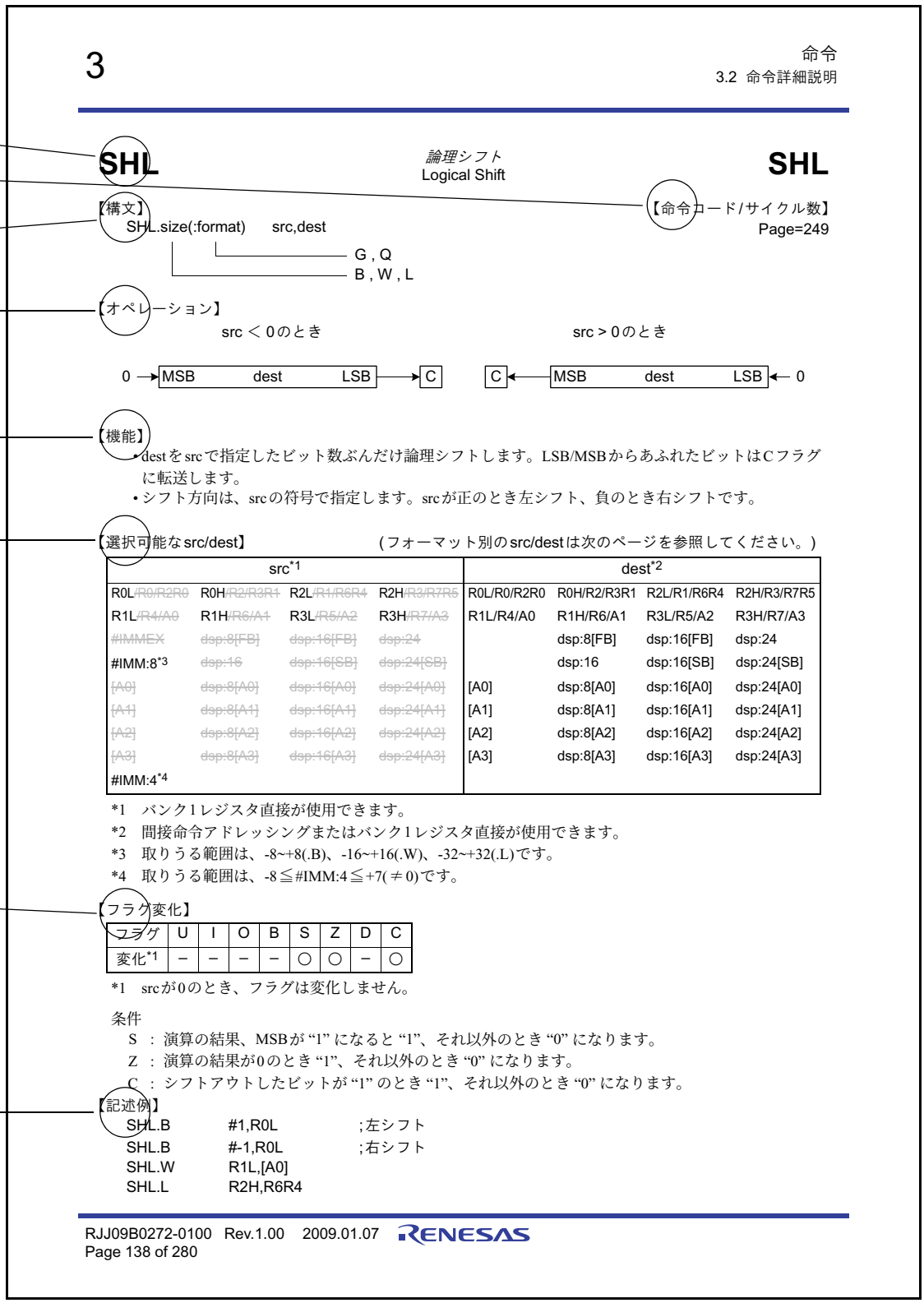
オペランドを記述します。

(e) **指定できるサイズ指定子**

(b)で指定できるデータサイズを示しています。

(f) **指定できる命令フォーマット指定子**

(c)で指定できる命令フォーマットを示しています。



(1) **SHL** *論理シフト* Logical Shift **SHL**

(2) 【命令コード/サイクル数】 Page=249

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

(4) 【オペレーション】

src ＜ 0のとき

src > 0のとき

(5) 【機能】
- destをsrcで指定したビット数ぶんだけ論理シフトします。LSB/MSBからあふれたビットはCフラグに転送します。
- シフト方向は、srcの符号で指定します。srcが正のとき左シフト、負のとき右シフトです。

(6) 【選択可能なsrc/dest】 (フォーマット別のsrc/destは次のページを参照してください。)

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 バンク1レジスタ直接が使用できます。
*2 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*3 取りうる範囲は、-8~+8(.B)、-16~+16(.W)、-32~+32(.L)です。
*4 取りうる範囲は、-8≦#IMM:4≦+7(≠0)です。

(7) 【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	−	−	−	−	○	○	−	○

*1 srcが0のとき、フラグは変化しません。

条件
S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： シフトアウトしたビットが “1” のとき “1”、それ以外のとき “0” になります。

(8) 【記述例】

```
SHL.B   #1,R0L      ;左シフト
SHL.B   #-1,R0L     ;右シフト
SHL.W   R1L,[A0]
SHL.L   R2H,R6R4
```

RJJ09B0272-0100 Rev.1.00 2009.01.07
Page 138 of 280

(4) オペレーション

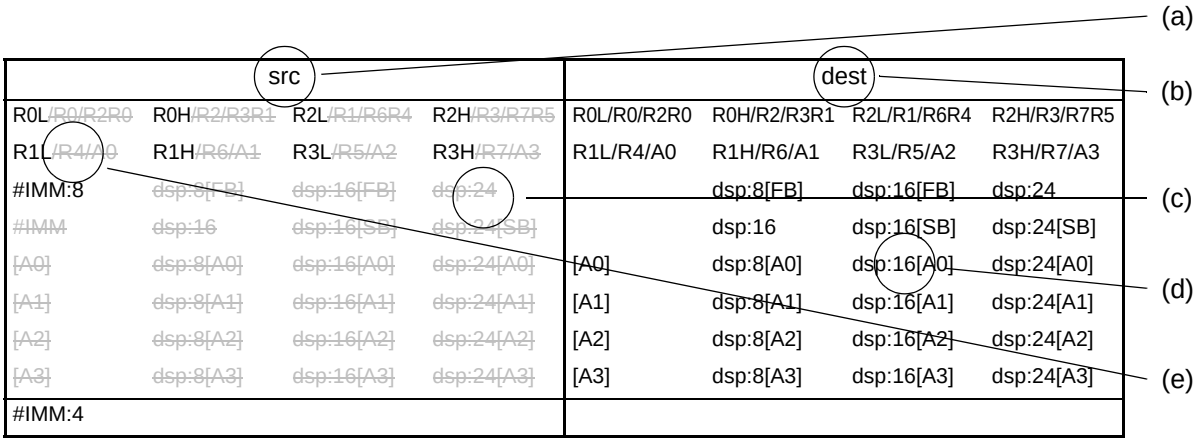
命令の動作をC言語風の表記を用いて説明しています。

(5) 機能

命令の機能、注意事項を説明しています。

(6) 選択可能なsrc/dest (label)

命令がオペランドをもつとき、オペランドとして選択できるアドレッシングモードを示しています。



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) src (source)**として選択できる項目**

(b) dest (destination)**として選択できる項目**

(c) **選択できないアドレッシング**

(d) **選択できるアドレッシング**

(e) **演算サイズによって変化するアドレッシング**

左 (R1L)　演算サイズがバイト (8ビット) のときに使われるアドレッシング
中央 (R4)　演算サイズがワード (16ビット) のときに使われるアドレッシング
右 (A0)　演算サイズがロングワード (32ビット) のときに使われるアドレッシング

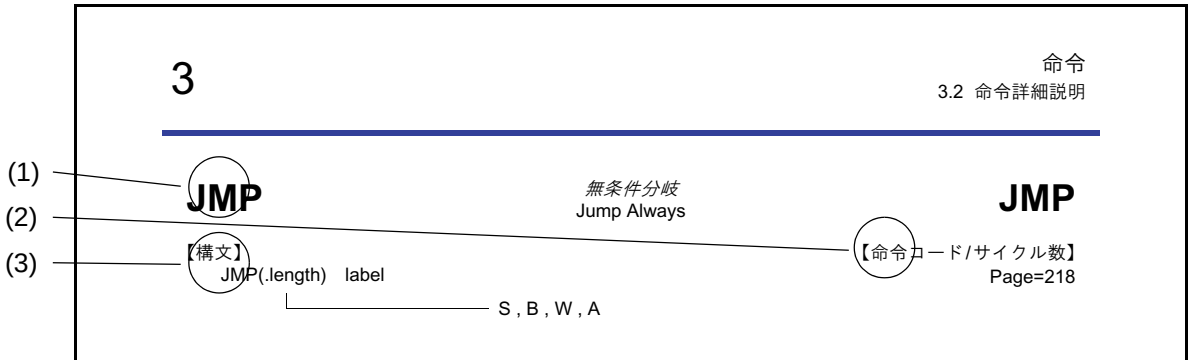
(7) フラグ変化

命令実行後のフラグの変化を示します。表中に示す記号の意味は次のとおりです。
“–”　変化しません。
“○”　条件によって変化します。

(8) 記述例

命令の記述例を示しています。

JMP、JMPI、JSR、JSRI命令の構文について以下に実例をあげて示します。



(3) 構文

命令の構文を記号で示しています。

JMP (.length) label

S , B , W , A → (d)

↓ (a) ↓ (b) ↓ (c)

(a) ニーモニック JMP

ニーモニックを記述します。

(b) 分岐距離指定子 .length

分岐する距離を記述します。JMP、JSR命令については省略できます。
指定できる分岐距離を以下に示します。

.S	3ビットPC前方相対
.B	8ビットPC相対
.W	16ビットPC相対
.A	24ビットPC相対
.L	32ビット絶対

(c) オペランド label

オペランドを記述します。

(d) 指定できる分岐距離指定子

(b)で指定できる分岐距離を示しています。

3.2 命令詳細説明

次ページよりR32C/100シリーズの各命令の詳細説明を示します。

ABS

絶対値
Absolute

ABS

【構文】

ABS.size dest

B , W , L

【命令コード/サイクル数】

Page=180

【オペレーション】

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

【機能】

• destの絶対値をとり、destに格納します。

【選択可能な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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O ： 演算前のdestが-128(.B)、-32768(.W)、-2147483648(.L)のとき “1”、それ以外のとき “0” になります。
S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： 不定になります。

【記述例】

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

ADC

キャリー付き加算
Add with Carry

ADC

【構文】

ADC.size src,dest

B , W , L

【命令コード/サイクル数】

Page=180

【オペレーション】

dest = dest + src + C;

【機能】

• dest と src と C フラグを加算し、dest に格納します。

【選択可能な 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 間接命令アドレッシングまたはバンク 1 レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O : 符号付き演算のオーバフローを示します。演算の結果、-128(.B) または +127(.B)、-32768(.W) または +32767(.W)、-2147483648(.L) または +2147483647(.L) を超えると “1”、それ以外のとき “0” になります。

S : 演算の結果、MSB が “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が 0 のとき “1”、それ以外のとき “0” になります。

C : 符号なし演算のオーバフローを示します。演算の結果、+255(.B)、+65535(.W)、+4294967295(.L) を超えると “1”、それ以外のとき “0” になります。

【記述例】

```
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

キャリーフラグの加算
Add Carry Flag

ADCF

【構文】

ADCF.size dest

B , W , L

【命令コード/サイクル数】

Page=181

【オペレーション】

dest = dest + C;

【機能】

• destとCフラグを加算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O : 符号付き演算のオーバフローを示します。演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

C : 符号なし演算のオーバフローを示します。演算の結果、+255(.B)、+65535(.W)、+4294967295(.L)を超えると “1”、それ以外のとき “0” になります。

【記述例】

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

ADD

キャリーなし加算
Add

ADD

【構文】

ADD.size(:format)　src,dest

G , Q , S
B , W , L

【命令コード/サイクル数】

Page=182

【オペレーション】

dest = dest + src;

【機能】

• dest と src を加算し、dest に格納します。

【選択可能な src/dest**】**　　**(フォーマット別の** src/dest**は次のページを参照してください。)**

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2　演算の対象はUフラグで示されるスタックポインタです。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O ： 符号付き演算のオーバフローを示します。演算の結果、-128(.B) または +127(.B)、-32768(.W) または +32767(.W)、-2147483648(.L) または +2147483647(.L) を超えると “1”、それ以外のとき “0” になります。
S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： 符号なし演算のオーバフローを示します。演算の結果、+255(.B)、+65535(.W)、+4294967295(.L) を超えると “1”、それ以外のとき “0” になります。

【記述例】

```
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]]
```

【フォーマット別src/dest】

Gフォーマット

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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 演算の対象はUフラグで示されるスタックポインタです。
*3 サイズ指定子(.size)には “.L” のみ指定できます。

Qフォーマット

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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 取りうる範囲は-8≦#IMM:4≦+7です。
*3 取りうる範囲は#IMM:3=4,8,12,16,20です。
*4 演算の対象はUフラグで示されるスタックポインタです。
*5 サイズ指定子(.size)には “.L” のみ指定できます。

Sフォーマット

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

*1 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

ADDF

浮動小数点加算
Add Floating Point

ADDF

【構文】
ADDF src,dest

【命令コード/サイクル数】
Page=184

【オペレーション】
dest = dest + src ;

【機能】
- srcとdestを符号付きで浮動小数点加算し、destに格納します。
- 演算結果は、フラグレジスタ(FLG)に指定された丸めモードに従って丸められます。
- 演算結果が最大の正規化数を超えたとき、結果は符号に応じて正の最大値(7F7FFFFFh)、または負の最大値(FF7FFFFFh)になります。
- 演算結果が最小の正規化数を下回ったとき、結果は丸めモードに従ってゼロ(00000000h)、または正の最小値(00800000h)、または負の最小値(80800000h)になります。
- 不正入力に対する演算結果は不定です。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	○	○	−	−	○	−	○	○	−	○

条件
FO : 不正入力と、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
FU : 不正入力と、演算の結果が最小の正規化数を下回ったとき “1”、それ以外のとき “0” になります。
O : 不正入力と、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果、すべてのビットが “0” のとき “1”、それ以外のとき “0” になります。
C : 不定になります。

【記述例】
```
ADDF    R2R0,R3R1
ADDF    [A1],R2R0
ADDF    mem[FB],R3R1
```

ADSF

サインフラグの加算
Add Sign Flag

ADSF

【構文】

ADSF.size dest

B , W , L

【命令コード/サイクル数】

Page=185

【オペレーション】

dest = dest + S;

【機能】

• dest と S フラグを加算し、dest に格納します。

【選択可能な 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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O : 符号付き演算のオーバフローを示します。演算の結果、-128(.B) または +127(.B)、-32768(.W) または +32767(.W)、-2147483648(.L) または +2147483647(.L) を超えると “1”、それ以外のとき “0” になります。

S : 演算の結果、MSB が “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が 0 のとき “1”、それ以外のとき “0” になります。

C : 符号なし演算のオーバフローを示します。演算の結果、+255(.B)、+65535(.W)、+4294967295(.L) を超えると “1”、それ以外のとき “0” になります。

【記述例】

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

AND

論理積
And Logical

AND

【構文】

AND.size(:format)　src,dest

G , S
B , W , L

【命令コード/サイクル数】

Page=186

【オペレーション】

dest = dest & src;

【機能】

• dest と src の論理積をとり、dest に格納します。

【選択可能な src/dest**】**　　(**フォーマット別の** src/dest**は次のページを参照してください。**)

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	−

条件

S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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]]
```

【フォーマット別src/dest】

Gフォーマット

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

Sフォーマット

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

*1　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

BCLR

ビットクリア
Clear a Bit

BCLR

【構文】

BCLR dest

【命令コード/サイクル数】

Page=187

【オペレーション】

dest = 0;

【機能】

• dest に “0” を格納します。

【選択可能な 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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

BITINDEX

ビットインデックス
Index next Bit Instruction

BITINDEX

【構文】

BITINDEX.size src

B , W , L

【命令コード/サイクル数】

Page=188

【オペレーション】

【機能】

- 次のビット命令のアドレッシングモードをモディファイします。
- この命令の直後には、割り込み要求を受け付けません。
- srcで指定したオペランドが次の命令のsrcまたはdestのインデックス値(bit)となります

【選択可能な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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

BM*Cnd*

条件ビット転送
Move a Bit Conditionally

BM*Cnd*

【構文】

BM*Cnd* dest

【命令コード/サイクル数】

Page=188

【オペレーション】

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

【機能】

- *Cnd*で示す条件の真偽値をdestに転送します。真の場合 “1”、偽の場合 “0” が転送されます。
- *Cnd*には次の種類があります。

Cnd	条件		式	*Cnd*	条件		式
GEU/C	C == 1	**等しいまたは大きい/Cフラグが** “1”	≦	LTU/NC	C == 0	**小さい/Cフラグが** “0”	>
EQ/Z	Z == 1	**等しい/Zフラグが** “1”	=	NE/NZ	Z == 0	**等しくない/Zフラグが** “0”	≠
GTU	C & ~Z == 1	**大きい**	<	LEU	C & ~Z == 0	**等しいまたは小さい**	≧
PZ	S == 0	**正またはゼロ**	0≦	N	S == 1	**負**	0>
GE	S ^ O == 0	**等しい、または符号付きで大きい**	≦	LE	(S ^ O) | Z == 1	**等しい、または符号付きで小さい**	≧
GT	(S ^ O) | Z == 0	**符号付きで大きい**	<	LT	S ^ O == 1	**符号付きで小さい**	>
O	O == 1	**Oフラグが** “1”		NO	O == 0	**Oフラグが** “0”	

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

BNOT

ビット反転
Not a Bit

BNOT

【構文】
BNOT dest

【命令コード/サイクル数】
Page=189

【オペレーション】
dest = ~dest;

【機能】
• destを反転し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

BRK

デバッグ割り込み
Break

BRK

【構文】
BRK

【命令コード/サイクル数】
Page=189

【オペレーション】

FFFFFFE7h**番地が“** FFh **”以外の場合**

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

FFFFFFE7h**番地が“** FFh **”の場合**

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

【機能】
- BRK割り込みが発生します。
- BRK割り込みはノンマスカブル割り込みです。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	○	○	−	−	−	−	○	−

条件
U ：“0” になります。
I ：“0” になります。
D ：“0” になります。

BRK命令実行前のフラグはスタックに退避されます。

【記述例】
BRK

BRK2

デバッグ割り込み2
Break 2

BRK2

【構文】
BRK2

【命令コード/サイクル数】
Page=189

【オペレーション】

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

【機能】

- この命令はデバッガで使用する専用命令です。ユーザプログラムでは使用しないでください。
- BRK2割り込みが発生します。
- BRK2割り込みはノンマスカブル割り込みです。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	○	○	−	−	−	−	○	−

条件
U ： “0” になります。
I ： “0” になります。
D ： “0” になります。

BRK2命令実行前のフラグはスタックに退避されます。

【記述例】

```
BRK2
```

BSET

ビットセット
Set a Bit

BSET

【構文】

BSET dest

【命令コード/サイクル数】

Page=190

【オペレーション】

dest = 1;

【機能】

- dest に “1” を格納します。

【選択可能な 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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

BTST

ビットテスト
Test a Bit

BTST

【構文】

BTST (:format)　　src

G , S

【命令コード/サイクル数】

Page=190

【オペレーション】

Z　=　~src;
C　=　src;

【機能】

• srcの反転をZフラグに、srcをCフラグに転送します。

【選択可能なsrc】

Gフォーマット

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

Sフォーマット

src
bit,abs:16

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	○	−	○

条件

Z　：srcが “0” のとき “1”、srcが “1” のとき “0” になります。
C　：srcが “1” のとき “1”、srcが “0” のとき “0” になります。

【記述例】

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

BTSTC

ビットテスト& クリア
Test a Bit and Clear

BTSTC

【構文】

BTSTC dest

【命令コード/サイクル数】

Page=191

【オペレーション】

```
Z    =   ~dest;
C    =   dest;
dest =   0;
```

【機能】

- destの反転をZフラグに、destをCフラグに転送します。その後、destに “0” を格納します。
- 本命令をSFR領域に対して使用しないでください。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	○	−	○

条件

Z ： destが “0” のとき “1”、destが “1” のとき “0” になります。
C ： destが “1” のとき “1”、destが “0” のとき “0” になります。

【記述例】

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

BTSTS

ビットテスト& セット
Test a Bit and Set

BTSTS

【構文】
BTSTS dest

【命令コード/サイクル数】
Page=191

【オペレーション】
```
Z    =   ~dest;
C    =   dest;
dest =   1;
```

【機能】
- destの反転をZフラグに、destをCフラグに転送します。その後、destに “1” を格納します。
- 本命令をSFR領域に対して使用しないでください。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	○	−	○

条件
Z ： destが “0” のとき “1”、destが “1” のとき “0” になります。
C ： destが “1” のとき “1”、destが “0” のとき “0” になります。

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

CLIP

クリップ
Clip

CLIP

【構文】

CLIP.size src1,src2,dest

B , W , L

【命令コード/サイクル数】
Page=192

【オペレーション】

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

【機能】

- src1≦dest≦src2となるようにdestの値をクリップします。
- 比較は符号付きで行われます。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

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

CMP

比較
Compare

CMP

【構文】

CMP.size(:format)　src,dest

G , Q , S
B , W , L

【命令コード/サイクル数】

Page=192

【オペレーション】

dest - src;

【機能】

• destからsrcを減算した結果にしたがって、フラグレジスタ(FLG)の各フラグを変化させます。

【選択可能なsrc/dest】　　**(フォーマット別のsrc/destは次のページを参照してください。)**

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O ： 符号付き演算のオーバフローを示します。演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。

S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。

C ： 符号なし演算の結果、0以上のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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]]
```

【フォーマット別src/dest】

Gフォーマット

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

Qフォーマット

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2　取りうる範囲は-8≦#IMM:4≦+7です。

Sフォーマット

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

*1　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

CMPF

浮動小数点比較
Compare Floating Point

CMPF

【構文】

CMPF src,dest

【命令コード/サイクル数】

Page=194

【オペレーション】

dest - src ;

【機能】

- destからsrcを減算した結果にしたがって、フラグレジスタ(FLG)の各フラグを変化させます。
- 不正入力に対する演算結果は不定です。

【選択可能な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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	○	○	−	−	○	−	○	○	−	○

条件

FO ： 不正入力のとき “1”、それ以外のとき “0” になります。
FU ： 不正入力のとき “1”、それ以外のとき “0” になります。
O ： 不正入力のとき “1”、それ以外のとき “0” になります。
S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果、すべてのビットが “0” のとき “1”、それ以外のとき “0” になります。
C ： 不定になります。

【記述例】

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

CNVIF

整数→浮動小数点数変換
Convert Integer to Floating Point

CNVIF

【構文】

CNVIF src,dest

【命令コード/サイクル数】

Page=195

【オペレーション】

dest = (float) src;

【機能】

- srcを単精度浮動小数点数に変換します。
- 演算結果は、フラグレジスタ(FLG)に指定された丸めモードに従って丸められます。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	○	○	−	−	○	−	○	○	−	○

条件

FO : “0” になります。
FU : “0” になります。
O : “0” になります。
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果、すべてのビットが “0” のとき “1”、それ以外のとき “0” になります。
C : 不定になります。

【記述例】

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

DADC

キャリー付き10進加算
Add Decimal with Carry

DADC

【構文】

DADC.size src,dest

B , W , L

【命令コード/サイクル数】

Page=196

【オペレーション】

dest = dest + src + C;

【機能】

• destとsrcとCフラグを10進加算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	○

条件

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

C : 演算の結果、+99(.B)、+9999(.W)、+99999999(.L)を超えると “1”、それ以外のとき “0” になります。

【記述例】

```
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進加算
Add Decimal

DADD

【構文】
DADD.size src,dest
B , W , L

【命令コード/サイクル数】
Page=197

【オペレーション】
dest = dest + src;

【機能】
• destとsrcを10進加算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	○

条件
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C : 演算の結果、+99(.B)、+9999(.W)、+99999999(.L)を超えると “1”、それ以外のとき “0” になります。

【記述例】

```
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

デクリメント
Decrement

DEC

【構文】
DEC.size dest
B , W , L

【命令コード/サイクル数】
Page=198

【オペレーション】
dest = dest - 1;

【機能】
• destから1を減算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	○	○	–	–

条件
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

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

DIV

符号付き除算
Signed Divide

DIV

【構文】

DIV.size src,dest

B , W , L

【命令コード/サイクル数】

Page=199

【オペレーション】

dest = dest / src;

【機能】

- destをsrcで符号付き除算し、商をdestに格納します。商は0方向に丸められます。
- サイズ指定子(.size)に ".B" を指定した場合、src/destとも8ビットで演算し、結果を8ビットで格納します。
- サイズ指定子(.size)に ".W" を指定した場合、src/destとも16ビットで演算し、結果を16ビットで格納します。
- サイズ指定子(.size)に ".L" を指定した場合、src/destとも32ビットで演算し、結果を32ビットで格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 除数(src)が0のとき、または演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると "1"、それ以外のとき "0" になります。

【記述例】

```
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

浮動小数点除算
Divide Floating Piont

DIVF

【構文】
DIVF src,dest

【命令コード/サイクル数】
Page=200

【オペレーション】
dest = dest / src ;

【機能】
- destをsrcで符号付き除算し、商をdestに格納します。
- 商は、フラグレジスタ(FLG)に指定された丸めモードに従って丸められます。
- 演算結果が最大の正規化数を超えたとき、結果は符号に応じて正の最大値(7F7FFFFFh)、または負の最大値(FF7FFFFFh)になります。
- 演算結果が最小の正規化数を下回ったとき、結果は丸めモードに従ってゼロ(00000000h)、または正の最小値(00800000h)、または負の最小値(80800000h)になります。
- 不正入力に対する演算結果は不定です。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	○	○	−	−	○	−	○	○	−	○

条件
FO : 不正入力と、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
FU : 不正入力と、演算の結果が最小の正規化数を下回ったとき “1”、それ以外のとき “0” になります。
O : 除数(src)が0のとき、または、不正入力や、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果、すべてのビットが “0” のとき “1”、それ以外のとき “0” になります。
C : 不定になります。

【記述例】
```
DIVF    R2R0,R3R1
DIVF    [A1],R2R0
DIVF    mem[FB],R3R1
```

DIVU

符号なし除算
Unsigned Divide

DIVU

【構文】

DIVU.size src,dest

B , W , L

【命令コード/サイクル数】

Page=201

【オペレーション】

dest = dest / src;

【機能】

- destをsrcで符号なし除算し、商をdestに格納します。商は0方向に丸められます。
- サイズ指定子(.size)に ".B" を指定した場合、src/destとも8ビットで演算し、結果を8ビットで格納します。
- サイズ指定子(.size)に ".W" を指定した場合、src/destとも16ビットで演算し、結果を16ビットで格納します。
- サイズ指定子(.size)に ".L" を指定した場合、src/destとも32ビットで演算し、結果を32ビットで格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 除数(src)が0のとき "1"、それ以外のとき "0" になります。

【記述例】

```
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

符号付き除算
Signed Divide extra

DIVX

【構文】

DIVX.size src,dest

B , W , L

【命令コード/サイクル数】

Page=202

【オペレーション】

dest = dest / src;

【機能】

- destをsrcで符号付き除算し、商をdestに格納します。商は-∞方向に丸められます。
- サイズ指定子(.size)に ".B" を指定した場合、src/destとも8ビットで演算し、結果を8ビットで格納します。
- サイズ指定子(.size)に ".W" を指定した場合、src/destとも16ビットで演算し、結果を16ビットで格納します。
- サイズ指定子(.size)に ".L" を指定した場合、src/destとも32ビットで演算し、結果を32ビットで格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 除数(src)が0のとき、または演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると "1"、それ以外のとき "0" になります。

【記述例】

```
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

ボロー付き10進減算
Subtract Decimal with Borrow

DSBB

【構文】

DSBB.size src,dest

B , W , L

【命令コード/サイクル数】

Page=203

【オペレーション】

dest = dest - src - ~C;

【機能】

• destからsrcとCフラグの反転を10進減算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	○

条件

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C : 演算の結果が0以上のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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進減算
Subtract Decimal

DSUB

【構文】

DSUB.size src,dest

B , W , L

【命令コード/サイクル数】

Page=204

【オペレーション】

dest = dest - src;

【機能】

• destからsrcを10進減算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	○

条件

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C : 演算の結果が0以上のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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

符号付き除算
Extended Signed Divide with Remainder

EDIV

【構文】

EDIV.size src,dest

B , W , L

【命令コード/サイクル数】

Page=205

【オペレーション】

destL(**商**) = dest / src;
destH(**剰余**)= dest % src;

【機能】

- destをsrcで符号付き除算し、商をdestの下位に、剰余をdestの上位に格納します。商は0方向に丸められ剰余の符号は被除数(dest)の符号と同一になります。
- サイズ指定子(.size)に ".B" を指定した場合、srcを8ビット、destを16ビットで演算し、商と剰余をそれぞれ8ビットで格納します。destにはR0(R0H:R0L)、R1(R1H:R1L)、R2(R2H:R2L)、R3(R3H:R3L)の4種類が指定できます。
- サイズ指定子(.size)に ".W" を指定した場合、srcを16ビット、destを32ビットで演算し、商と剰余をそれぞれ16ビットで格納します。destにはR2R0(R2:R0)、R3R1(R3:R1)、R6R4(R6:R4)、R7R5(R7:R5)の4種類が指定できます。
- サイズ指定子(.size)に ".L" を指定した場合、srcを32ビット、destを64ビットで演算し、商と剰余をそれぞれ32ビットで格納します。destにはR3R1R2R0(R3R1:R2R0)、R7R5R6R4(R7R5:R6R4)、A1A0(A1:A0)、A3A2(A3:A2)の4種類が指定できます。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 除数(src)が0のとき、または演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると "1"、それ以外のとき "0" になります。

【記述例】

```
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

符号なし除算
Extended Unsigned Divide with Remainder

EDIVU

【構文】

EDIVU.size src,dest

B , W , L

【命令コード/サイクル数】

Page=206

【オペレーション】

destL(**商**) = dest / src;
destH(**剰余**)= dest % src;

【機能】

- destをsrcで符号なし除算し、商をdestの下位に、剰余をdestの上位に格納します。
- サイズ指定子(.size)に “.B” を指定した場合、srcを8ビット、destを16ビットで演算し、商と剰余をそれぞれ8ビットで格納します。destにはR0(R0H:R0L)、R1(R1H:R1L)、R2(R2H:R2L)、R3(R3H:R3L)の4種類が指定できます。
- サイズ指定子(.size)に “.W” を指定した場合、srcを16ビット、destを32ビットで演算し、商と剰余をそれぞれ16ビットで格納します。destにはR2R0(R2:R0)、R3R1(R3:R1)、R6R4(R6:R4)、R7R5(R7:R5)の4種類が指定できます。
- サイズ指定子(.size)に “.L” を指定した場合、srcを32ビット、destを64ビットで演算し、商と剰余をそれぞれ32ビットで格納します。destにはR3R1R2R0(R3R1:R2R0)、R7R5R6R4(R7R5:R6R4)、A1A0(A1:A0)、A3A2(A3:A2)の4種類が指定できます。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 演算の結果、商が8ビット(.B)、16ビット(.W)、32ビット(.L)を超えるか、または除数(src)が0のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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

符号付き除算
Extended Signed Divide extra with Remainder

EDIVX

【構文】

EDIVX.size src,dest

B , W , L

【命令コード/サイクル数】

Page=207

【オペレーション】

destL(**商**) = dest / src;
destH(**剰余**)= dest % src;

【機能】

- destをsrcで符号付き除算し、商をdestの下位に、剰余をdestの上位に格納します。商は-∞方向に丸められ剰余の符号は除数(src)の符号と同一になります。
- サイズ指定子(.size)に “.B” を指定した場合、srcを8ビット、destを16ビットで演算し、商と剰余をそれぞれ8ビットで格納します。destにはR0(R0H:R0L)、R1(R1H:R1L)、R2(R2H:R2L)、R3(R3H:R3L)の4種類が指定できます。
- サイズ指定子(.size)に “.W” を指定した場合、srcを16ビット、destを32ビットで演算し、商と剰余をそれぞれ16ビットで格納します。destにはR2R0(R2:R0)、R3R1(R3:R1)、R6R4(R6:R4)、R7R5(R7:R5)の4種類が指定できます。
- サイズ指定子(.size)に “.L” を指定した場合、srcを32ビット、destを64ビットで演算し、商と剰余をそれぞれ32ビットで格納します。destにはR3R1R2R0(R3R1:R2R0)、R7R5R6R4(R7R5:R6R4)、A1A0(A1:A0)、A3A2(A3:A2)の4種類が指定できます。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 除数(src)が0のとき、または演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。

【記述例】

```
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

符号付き乗算
Extended Signed Multiply

EMUL

【構文】

EMUL.size src,dest

B , W , L

【命令コード/サイクル数】

Page=208

【オペレーション】

destH:dest = dest * src;

【機能】

- srcとdestを符号付き乗算し、結果をdestH:destに格納します。
- サイズ指定子(.size)に“.B”を指定した場合、src、destとも8ビットで演算し、結果を16ビットで格納します。destにはR0L(R0H:R0L)、R1L(R1H:R1L)、R2L(R2H:R2L)、R3L(R3H:R3L)の4種類が指定できます。
- サイズ指定子(.size)に“.W”を指定した場合、src、destとも16ビットで演算し、結果を32ビットで格納します。destにはR0(R2:R0)、R1(R3:R1)、R4(R6:R4)、R5(R7:R5)の4種類が指定できます。
- サイズ指定子(.size)に“.L”を指定した場合、src、destとも32ビットで演算し、結果を64ビットで格納します。destにはR2R0(R3R1:R2R0)、R6R4(R7R5:R6R4)、A0(A1:A0)、A2(A3:A2)の4種類が指定できます。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
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

符号なし乗算
Extended Unsigned Multiply

EMULU

【構文】

EMULU.size src,dest

B , W , L

【命令コード/サイクル数】

Page=209

【オペレーション】

dest = dest * src;

【機能】

- srcとdestを符号なし乗算し、結果をdestに格納します。
- サイズ指定子(.size)に“.B”を指定した場合、src、destとも8ビットで演算し、結果を16ビットで格納します。destにはR0L(R0H:R0L)、R1L(R1H:R1L)、R2L(R2H:R2L)、R3L(R3H:R3L)の4種類が指定できます。
- サイズ指定子(.size)に“.W”を指定した場合、src、destとも16ビットで演算し、結果を32ビットで格納します。destにはR0(R2:R0)、R1(R3:R1)、R4(R6:R4)、R5(R7:R5)の4種類が指定できます。
- サイズ指定子(.size)に“.L”を指定した場合、src、destとも32ビットで演算し、結果を64ビットで格納します。destにはR2R0(R3R1:R2R0)、R6R4(R7R5:R6R4)、A0(A1:A0)、A2(A3:A2)の4種類が指定できます。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
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

スタックフレーム生成
Enter and Create Stack Frame

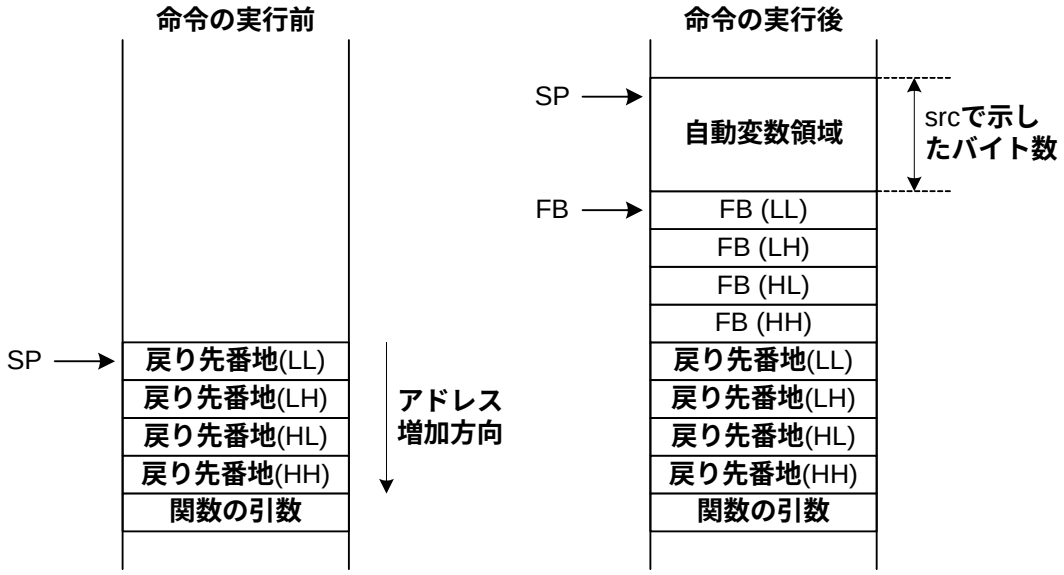
ENTER

【構文】
ENTER src

【命令コード/サイクル数】
Page=209

【オペレーション】
```
SP    =   SP - 4;
*SP   =   FB;
FB    =   SP;
SP    =   SP - src;
```

【機能】
- スタックフレームを生成します。srcはスタックフレームのサイズです。
- 下記にサブルーチンの先頭でENTER命令を実行する前後のスタック領域の状態を示します。



【選択可能なsrc】

src
#IMM:8*1　　#IMM:16*1

*1　#IMMには4の倍数を設定してください。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】
```
ENTER     #12
```

EXITD

スタックフレーム解放
Exit and Deallocate Stack Frame

EXITD

【構文】
EXITD

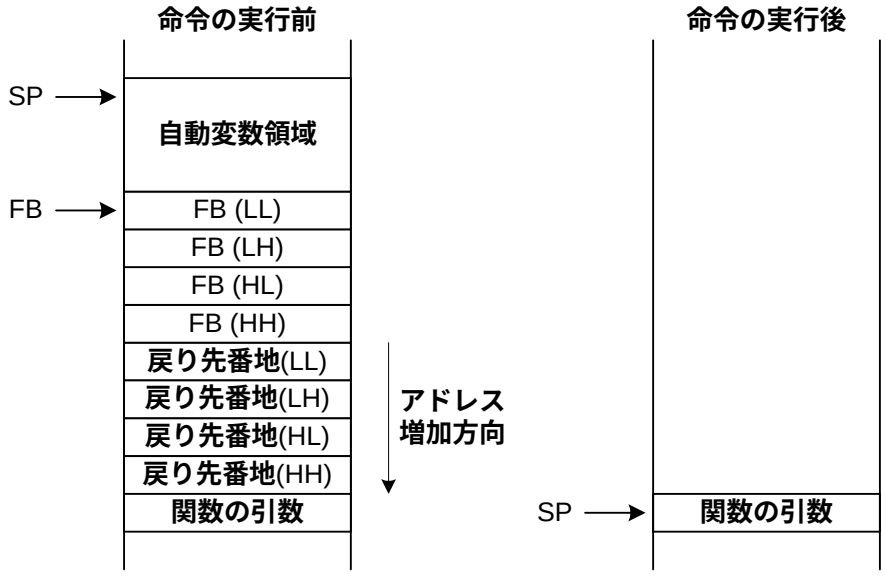
【命令コード/サイクル数】
Page=210

【オペレーション】

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

【機能】

- スタックフレームを解放し、サブルーチンから復帰します。
- 本命令はENTER命令と対で使用してください。
- 下記にサブルーチンの最後でEXITD命令を実行する前後のスタック領域の状態を示します。



【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
EXITD
```

EXITI

割り込みスタックフレーム解放
Exit Interrupt and Deallocate Stack Frame

EXITI

【構文】
EXITI

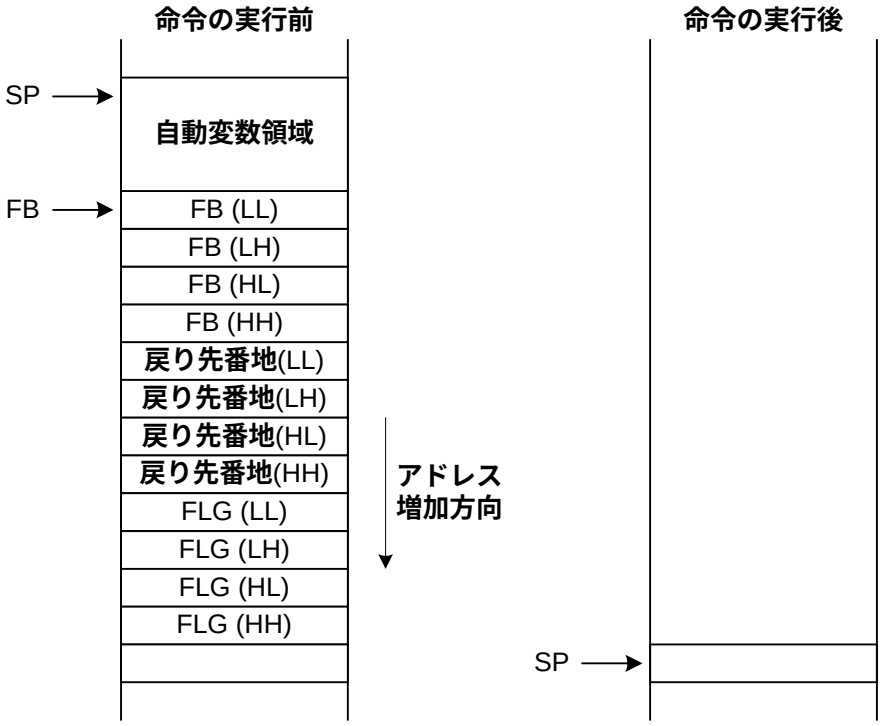
【命令コード/サイクル数】
Page=210

【オペレーション】

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

【機能】

- スタックフレームを解放し、割り込みルーチンから復帰します。
- 本命令はENTER命令と対で使用してください。
- 下記に割り込みルーチンの最後でEXITI命令を実行する前後のスタック領域の状態を示します。



【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	○	○	○	○	○	○	○	○

*1　すべて割り込み発生前の値(スタックに退避された値)になります。

【記述例】
EXITI

EXTS

符号拡張
Sign Extend

EXTS

【構文】

EXTS.size src,dest

BW , BL , WL

【命令コード/サイクル数】

Page=210

【オペレーション】

dest = (short | long) src;

【機能】

- srcを符号拡張し、destに格納します。
- サイズ指定子(.size)が ".BW" の場合、8ビットのsrcを16ビットに符号拡張しdestに格納します。
- サイズ指定子(.size)が ".BL" の場合、8ビットのsrcを32ビットに符号拡張しdestに格納します。
- サイズ指定子(.size)が ".WL" の場合、16ビットのsrcを32ビットに符号拡張しdestに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 取りうる範囲は-128≦IMM≦+127(.BW, .BL)、-32768≦IMM≦+32767(.WL)です。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	−

条件

S : 演算の結果、MSBが "1" になると "1"、それ以外のとき "0" になります。
Z : 演算の結果が0のとき "1"、それ以外のとき "0" になります。

【記述例】

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

EXTZ

ゼロ拡張
Zero Extend

EXTZ

【構文】

EXTZ.size(:format)　src,dest

G , S
BW , BL , WL

【命令コード/サイクル数】

Page=212

【オペレーション】

dest　=　(unsigned short | unsigned long) src;

【機能】

- srcをゼロ拡張し、destに格納します。
- サイズ指定子(.size)が ".BW" の場合、8ビットのsrcを16ビットにゼロ拡張しdestに格納します。
- サイズ指定子(.size)が ".BL" の場合、8ビットのsrcを32ビットにゼロ拡張しdestに格納します。
- サイズ指定子(.size)が ".WL" の場合、16ビットのsrcを32ビットにゼロ拡張しdestに格納します。

【選択可能なsrc/dest】　　**(フォーマット別のsrc/destは次のページを参照してください。**

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2　取りうる範囲は0≦IMM≦+255(.BW, .BL)、0≦IMM≦+65535(.WL)です。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	−

条件
S　：常に "0" になります。
Z　：演算の結果が0のとき "1"、それ以外のとき "0" になります。

【記述例】

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

【フォーマット別src/dest】

Gフォーマット

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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

Sフォーマット

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 サイズ指定子(.size)には ".WL" のみ指定できます。従いまして、srcは16ビット、destは32ビット固定です。

*2 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

FCLR

フラグのクリア
Clear a Flag

FCLR

【構文】
FCLR dest

【命令コード/サイクル数】
Page=214

【オペレーション】
dest = 0;

【機能】
• destに “0” を格納します。

【選択可能なdest】

dest							
U	I	O	B	S	Z	D	C

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	*1	*1	*1	*1	*1	*1	*1	*1

*1 指定されたフラグが “0” になります。

【記述例】
```
FCLR I
FCLR S
FCLR O
```

FREIT

高速割り込みからの復帰
Return from Fast Interrupt

FREIT

【構文】
FREIT

【命令コード/サイクル数】
Page=214

【オペレーション】
```
FLG  =  SVF;
PC   =  SVP;
```

【機能】
- 高速割り込み要求を受け付けた時に退避したPCおよびFLGを高速割り込みレジスタから復帰し、割り込みルーチンから戻ります。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	○	○	○	○	○	○	○	○

*1　すべて割り込み発生前の値(SVFに退避された値)になります。

【記述例】
FREIT

FSET

フラグのセット
Set a Flag

FSET

【構文】
FSET dest

【命令コード/サイクル数】
Page=214

【オペレーション】
dest = 1;

【機能】
• destに “1” を格納します。

【選択可能なdest】

dest							
U	I	O	B	S	Z	D	C

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	*1	*1	*1	*1	*1	*1	*1	*1

*1 指定されたフラグが “1” になります。

【記述例】
```
FSET    I
FSET    S
FSET    O
```

INC

インクリメント
Increment

INC

【構文】

INC.size dest

B , W , L

【命令コード/サイクル数】

Page=215

【オペレーション】

dest = dest + 1;

【機能】

• destに1を加算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	○	○	–	–

条件

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

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

INDEX*Type*

インデックス
Index

INDEX*Type*

【構文】

INDEX*Type*.size src

B , W , L

【命令コード/サイクル数】

Page=216

【オペレーション】

【機能】

- 次の命令のアドレッシングモードをモディファイします。
- この命令の直後には割り込み要求を受け付けません。
- 配列をアクセスするために使用します。
- *Type*には次の種類があります。
- 詳細は「3.3 インデックス命令」を参照してください。

Type	機能
B	**次の命令がジェネリック形式のオペランドを2つ持つ命令のとき、**srcと**dest**の**両方をモディファイします。**
1	**次の命令がジェネリック形式またはショート形式のオペランドを1つしか持たない命令のとき当該オペランドを、ジェネリック形式のオペランドを2つ持つ命令のとき、**src**をモディファイします。**
2	**次の命令がジェネリック形式のオペランドを2つ持つ命令のときは**dest**をモディファイします。**

【選択可能な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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

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

INT

INT 命令割り込み
Interrupt

INT

【構文】
INT src

【命令コード/サイクル数】
Page=217

【オペレーション】

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

【機能】

- srcで指定した番号のソフトウェア割り込みが発生します。
- 割り込み番号(src)の範囲は、0≦src≦255です。
- srcが127以下の場合、Uフラグが “0” になりISPを使用します。
- srcが128以上の場合、Uフラグが示すスタックポインタを使用します。
- INT命令によって発生する割り込みはノンマスカブル割り込みです。

【選択可能なsrc】

src
#IMM:8*1

*1　#IMM:8は、ソフトウェア割り込み番号です。取りうる範囲は0≦IMM:8≦255です。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	○	○	−	−	−	−	○	−

*1　命令実行前のフラグはスタック領域に退避されます。

条件
U ： ソフトウェア割り込み番号が127以下のとき “0” になります。それ以外のときは変化しません。
I ： “0” になります。
D ： “0” になります。

【記述例】

```
INT     #0
```

INTO

オーバフロー割り込み
Interrupt on Overflow

INTO

【構文】
INTO

【命令コード/サイクル数】
Page=217

【オペレーション】

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

【機能】

- O フラグが “1” のときオーバフロー割り込みが発生します。
- O フラグが “0” のとき、割り込みは発生しません。
- オーバフロー割り込みはノンマスカブル割り込みです。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	○	○	−	−	−	−	○	−

*1　命令実行前のフラグはスタック領域に退避されます。

条件
U ： “0” になります。
I ： “0” になります。
D ： “0” になります。

【記述例】

```
INTO
```

JCnd

条件分岐
Jump Conditionally

JCnd

【構文】
JCnd label

【命令コード/サイクル数】
Page=217

【オペレーション】
if (true)
PC = label; (= PC + dsp)

【機能】
- *Cnd*で示す条件の真偽値を判断し分岐します。真の場合は分岐しますが、偽の場合は分岐しません。
- *Cnd*には次の種類があります。

Cnd	条件		式	*Cnd*	条件		式
GEU /C	C == 1	**等しいまたは大きい/ Cフラグが** “1”	≦	LTU/ NC	C == 0	**小さい/ Cフラグが** “0”	>
EQ/ Z	Z == 1	**等しい/ Zフラグが** “1”	=	NE/ NZ	Z == 0	**等しくない/ Zフラグが** “0”	≠
GTU	C & ~Z == 1	**大きい**	<	LEU	C & ~Z == 0	**等しいまたは小さい**	≧
PZ	S == 0	**正またはゼロ**	0≦	N	S == 1	**負**	0>
GE	S ^ O == 0	**等しい、または符号付きで大きい**	≦	LE	(S ^ O) | Z == 1	**等しい、または符号付きで小さい**	≧
GT	(S ^ O) | Z == 0	**符号付きで大きい**	<	LT	S ^ O == 1	**符号付きで小さい**	>
O	O == 1	**Oフラグが** “1”		NO	O == 0	**Oフラグが** “0”	

【選択可能なlabel】

label
PC*1 + dsp:8*2

*1 PCはJ*Cnd*命令のある番地+1です。
*2 dsp:8の範囲は-128≦dsp≦127です。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】
```
JC      label1
JLT     label2
```

JMP

無条件分岐
Jump Always

JMP

【構文】

JMP(.length)　label

S , B , W , A

【命令コード/サイクル数】

Page=218

【オペレーション】

PC　=　label;　(= PC + dsp)

【機能】

• labelへ分岐します。

【選択可能な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の値はJMP命令のある番地+1です。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
JMP.B　　label1
JMP.A　　label2
```

JMPI

間接分岐
Jump Indirectly

JMPI

【構文】

JMPI.length src

length: W , L

【命令コード/サイクル数】

Page=219

【オペレーション】

分岐距離指定子(.length)**が** ".W" **のとき**

PC = PC*1 + src;

分岐距離指定子(.length)**が** ".L" **のとき**

PC = src;

*1 PCの値はJMP命令のある番地です。

【機能】

- srcの内容に対し相対分岐または絶対分岐します。
- 分岐距離指定子(.length)に ".W" を指定した場合、JMPI命令のある番地とsrcの内容を符号付きで加算した番地に分岐します。srcがメモリの場合、必要なメモリ容量は2バイトです。
- 分岐距離指定子(.length)に ".L" を指定した場合、srcの内容が示す番地に分岐します。srcがメモリの場合、必要なメモリ容量は4バイトです。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

JSR

サブルーチン分岐
Jump to Subroutine

JSR

【構文】

JSR(.length)　label
　　W , A

【命令コード/サイクル数】
Page=220

【オペレーション】

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

*1　(PC+n) はJSR命令の次の命令の番地です。

【機能】

• labelが示すサブルーチンへ分岐します。

【選択可能なlabel】

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

*1　PCの値はJSR命令のある番地+1です。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
JSR.W　　subroutine1
JSR.A　　subroutine2
```

JSRI

間接サブルーチン分岐
Jump to Subroutine Indirectly

JSRI

【構文】

JSRI.length src

W , L

【命令コード/サイクル数】

Page=220

【オペレーション】

分岐距離指定子(.length)**が** ".W" **のとき**

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

分岐距離指定子(.length)**が** ".L" **のとき**

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

*1 (PC+n) はJSRI命令の次の命令の番地です。
*2 PCの値はJSRI命令のある番地です。

【機能】

- srcの内容に対し相対サブルーチン分岐または絶対サブルーチン分岐します。
- 分岐距離指定子(.length)に ".W" を指定した場合、JSRI命令のある番地とsrcの内容を符号付きで加算した番地にサブルーチン分岐します。srcがメモリの場合、必要なメモリ容量は2バイトです。
- 分岐距離指定子(.length)に ".L" を指定した場合、srcの内容が示す番地にサブルーチン分岐します。srcがメモリの場合、必要なメモリ容量は4バイトです。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

LDC

専用レジスタへの転送
Load into Control Register

LDC

【構文】
LDC src,dest

【命令コード/サイクル数】
Page=221

【オペレーション】
dest = src;

【機能】
• srcをdestに転送します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 #IMMEXはDMAC関連レジスタ、VCTのsrcとして使用できません。代わりに、#IMM:8、#IMM:16を指定すると、ゼロ拡張して転送されます。
*3 #IMM:8、#IMM:16はDMAC関連レジスタ、VCTのsrcとしてのみ使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	*1	*1	*1	*1	*1	*1	*1	*1	*1	*1

*1 destがFLGのときだけ変化します。

【記述例】

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

LDCTX

コンテキストの復帰
Load Context

LDCTX

【構文】

LDCTX src,dest

【命令コード/サイクル数】

Page=222

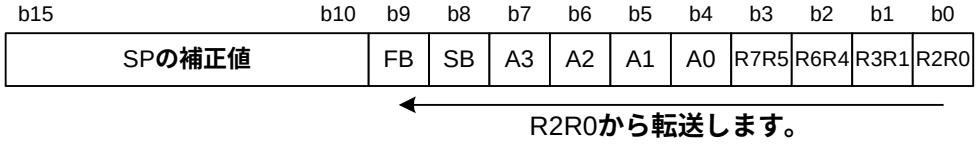
【オペレーション】

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

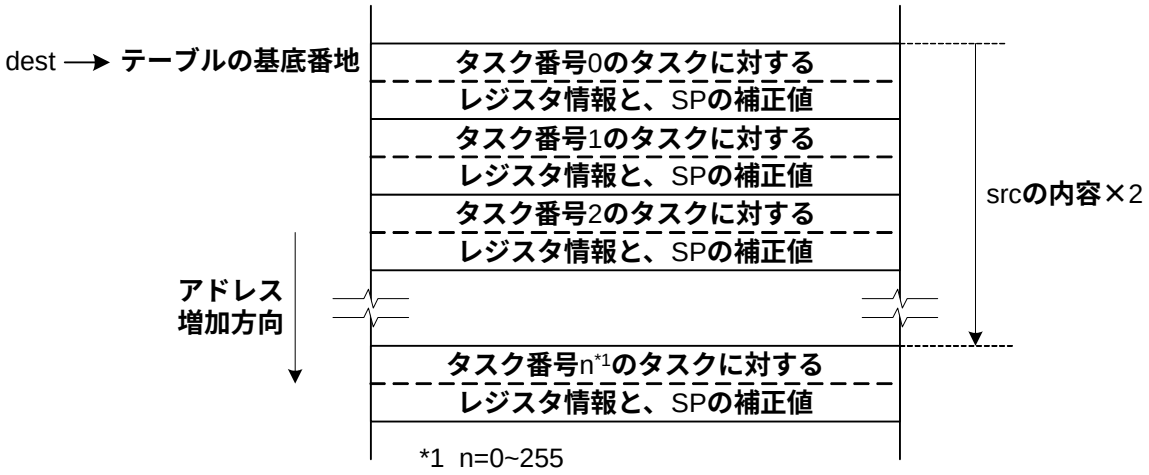
*1　nは復帰するレジスタの数です。

【機能】

- タスクのコンテキストをスタック領域から復帰します。
- srcにはタスク番号が格納されているRAMの番地を、destにはテーブルデータの先頭番地を設定してください。
- タスク番号によってテーブルデータの中から必要なレジスタ情報を指定し、そのレジスタ情報に従ってスタック領域のデータを各レジスタに転送します。その後、スタックポインタ(SP)にSPの補正値を加算します。SPの補正値には転送するレジスタの合計バイト数を設定してください。なお、テーブルデータに “0000h” は設定しないでください。
- 転送するレジスタの情報は次のとおり構成されています。ビットが “1” のとき転送するレジスタ、“0” のとき転送しないレジスタを示します。



- テーブルデータは次のとおり構成されています。destで示した番地がテーブルの基底番地となり、srcの内容の2倍をオフセットとする番地に格納されたワードデータのビット0~9にレジスタの情報、ビット10~15がスタックポインタの補正値を示します。



【選択可能なsrc/dest**】**

src	dest
abs:16	dsp:24

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
    LDCTX       ram,rom_tbl
```

LDIPL

割り込み優先レベル設定
Load Interrupt Priority Level

LDIPL

【構文】
LDIPL src

【命令コード/サイクル数】
Page=223

【オペレーション】
IPL = src;

【機能】
• src を IPL に転送します。

【選択可能な src】

src
#IMM:3

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】
LDIPL #2

MAX

最大値選択
Select Maximum Value

MAX

【構文】

MAX.size src,dest

B , W , L

【命令コード/サイクル数】
Page=223

【オペレーション】

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

【機能】

• src と dest を符号付きで比較し、大きい方の値を dest に格納します。

【選択可能な 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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

MIN

最小値選択
Select Minimum Value

MIN

【構文】

MIN.size src,dest

B , W , L

【命令コード/サイクル数】

Page=225

【オペレーション】

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

【機能】

• src と dest を符号付きで比較し、小さい方の値を dest に格納します。

【選択可能な 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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

MOV

転送
Move

MOV

【構文】

MOV.size(:format) src,dest

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

【命令コード/サイクル数】

Page=226

【オペレーション】

dest = src;

【機能】

• src を dest に転送します。

【選択可能な src/dest**】** (**フォーマット別の** src/dest**は次のページを参照してください。**)

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	−

条件

S ： src の MSB が “1” のとき “1”、それ以外のとき “0” になります。

Z ： src が 0 のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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]]
```

【フォーマット別src/dest】

Gフォーマット

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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 演算の対象はUフラグで示されるスタックポインタです。また、srcとdest同時にdsp:8[SP]を選択できません。
*3 srcにdsp:8[SP]を選択した場合、destに間接命令アドレッシングは選択できません。
*4 destにdsp:8[SP]を選択した場合、srcに#IMMEXまたは#IMMは選択できません。

Qフォーマット

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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 取りうる範囲は、-8≦#IMM:4≦7です。

Sフォーマット

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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 バンク1レジスタ直接は指定できません。
*3 サイズ指定子(.size)には “.L” のみ指定できます。

Zフォーマット

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

*1 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

MOVA

実効アドレス転送
Move Effective Address

MOVA

【構文】
MOVA src,dest

【命令コード/サイクル数】
Page=231

【オペレーション】
dest = &src;

【機能】
• srcの実効アドレスをdestに転送します。

【選択可能な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 バンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

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

MOV*Dir*

4ビットデータ転送
Move Nibble Data

MOV*Dir*

【構文】

MOV*Dir* src,dest

【命令コード/サイクル数】

Page=231

【オペレーション】

Dir	オペレーション
HH	H4:dest = H4:src
HL	L4:dest = H4:src
LH	H4:dest = L4:src
LL	L4:dest = L4:src

【機能】

- *Dir*がHHのときsrc (8ビット)の上位4ビットをdest (8ビット)の上位4ビットに転送します。
- *Dir*がHLのときsrc (8ビット)の上位4ビットをdest (8ビット)の下位4ビットに転送します。
- *Dir*がLHのときsrc (8ビット)の下位4ビットをdest (8ビット)の上位4ビットに転送します。
- *Dir*がLLのときsrc (8ビット)の下位4ビットをdest (8ビット)の下位4ビットに転送します。
- srcまたはdestのどちらか一方にR0Lを指定してください。

【選択可能なsrc/dest】

src*1				dest*1			
R0L/~~R0/R2R0~~	R0H/~~R2/R3R1~~	R2L/~~R1/R6R4~~	R2H/~~R3/R7R5~~	R0L/~~R0/R2R0~~*2			
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				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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2 バンク1レジスタ直接は指定できません。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

MUL

符号付き乗算
Signed Multiply

MUL

【構文】

MUL.size src,dest

B , W , L

【命令コード/サイクル数】

Page=232

【オペレーション】

dest = dest * src;

【機能】

- srcとdestを符号付きで乗算し、destに格納します。
- サイズ指定子(.size)が “.B” の場合、src/destとも8ビットで演算し、結果を8ビットで格納します。
- サイズ指定子(.size)が “.W” の場合、src/destとも16ビットで演算し、結果を16ビットで格納します。
- サイズ指定子(.size)が “.L” の場合、src/destとも32ビットで演算し、結果を32ビットで格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 符号付き演算のオーバフローを示します。演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。

【記述例】

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

MULF

浮動小数点乗算
Multiply Floating Point

MULF

【構文】

MULF src,dest

【命令コード/サイクル数】

Page=233

【オペレーション】

dest = dest * src;

【機能】

- srcとdestを符号付きで浮動小数点乗算し、destに格納します。
- 演算結果は、フラグレジスタ(FLG)に指定された丸めモードに従って丸められます。
- 演算結果が最大の正規化数を超えたとき、結果は符号に応じて正の最大値(7F7FFFFFh)、または負の最大値(FF7FFFFFh)になります。
- 演算結果が最小の正規化数を下回ったとき、結果は丸めモードに従ってゼロ(00000000h)、または正の最小値(00800000h)、または負の最小値(80800000h)になります。
- 不正入力に対する演算結果は不定です。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	○	○	−	−	○	−	○	○	−	○

条件

FO : 不正入力と、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
FU : 不正入力と、演算の結果が最小の正規化数を下回ったとき “1”、それ以外のとき “0” になります。
O : 不正入力と、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果、すべてのビットが “0” のとき “1”、それ以外のとき “0” になります。
C : 不定になります。

【記述例】

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

MULU

符号なし乗算
Unsigned Multiply

MULU

【構文】

MULU.size src,dest

B , W , L

【命令コード/サイクル数】

Page=234

【オペレーション】

dest = dest * src;

【機能】

- srcとdestを符号なしで乗算し、destに格納します。
- サイズ指定子(.size)が “.B” の場合、src/destとも8ビットで演算し、結果を8ビットで格納します。
- サイズ指定子(.size)が “.W” の場合、src/destとも16ビットで演算し、結果を16ビットで格納します。
- サイズ指定子(.size)が “.L” の場合、src/destとも32ビットで演算し、結果を32ビットで格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O : 演算の結果、8ビット(.B)、16ビット(.W)、32ビット(.L)を超えると “1”、それ以外のとき “0” になります。

【記述例】

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

MULX

丸め付き乗算
Signed Multiply and Round

MULX

【構文】

MULX.size src,dest

B , W , L

【命令コード/サイクル数】

Page=236

【オペレーション】

```
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: 一時レジスタ
*2 サイズ指定子(.size)が “.B” のとき6、“.W” のとき14、“.L” のとき30になります。
*3 サイズ指定子(.size)が “.W” のとき8、“.L” のとき16になります。

【機能】

- srcとdestを符号付き乗算し、フラグレジスタ(FLG)のビット16(DPビット)にしたがってシフト、四捨五入を行い結果をdestに格納します。
- サイズ指定子(.size)に “.B” を指定した場合、src/destとも8ビットで演算し、結果を8ビットで格納します。
- サイズ指定子(.size)に “.W” を指定した場合、src/destとも16ビットで演算し、結果を16ビットで格納します。
- サイズ指定子(.size)に “.L” を指定した場合、src/destとも32ビットで演算し、結果を32ビットで格納します。
- src、destを固定小数点数とみなした乗算に使用できます(次頁図参照)。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

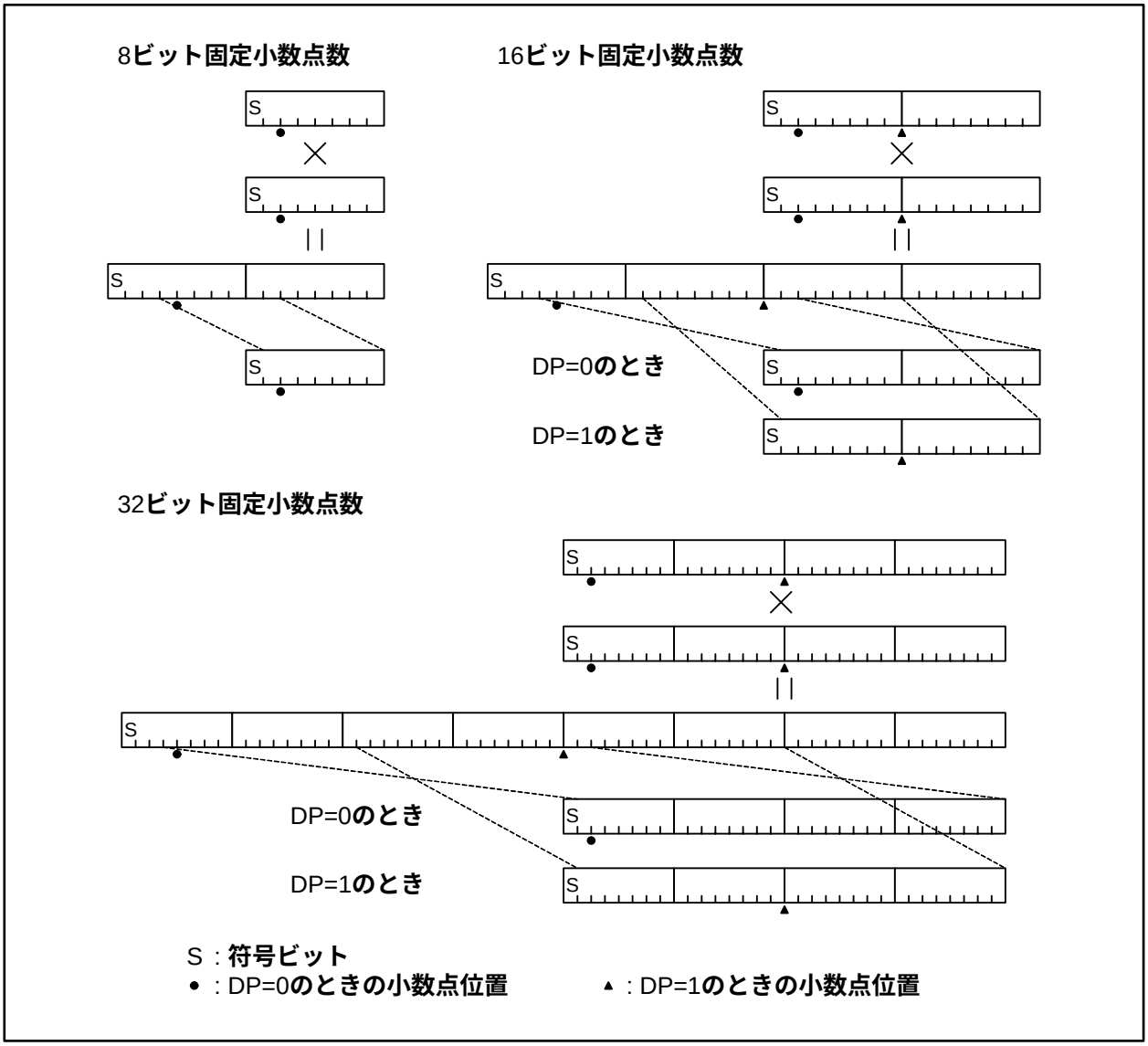


図 3.1　固定小数点乗算への応用例

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O ： 符号付き演算のオーバフローを示します。演算の結果、-128(.B) または +127(.B)、-32768(.W) または +32767(.W)、-2147483648(.L) または +2147483647(.L) を超えると “1”、それ以外のとき “0” になります。

【記述例】

```
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

符号反転
Negate

NEG

【構文】

NEG.size dest

B , W , L

【命令コード/サイクル数】

Page=236

【オペレーション】

dest = -dest;

【機能】

• destを符号反転し(2の補数をとり)、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O : 演算前のdestが-128(.B)、-32768(.W)、-2147483648(.L)のとき “1”、それ以外のとき “0” になります。
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

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

NOP

ノーオペレーション
No Operation

NOP

【構文】
NOP

【命令コード/サイクル数】
Page=237

【オペレーション】
PC = PC + 1;

【機能】
• PCに1を加算します。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】
NOP

NOT

論理反転
Logical Complement

NOT

【構文】

NOT.size dest

B , W , L

【命令コード/サイクル数】

Page=237

【オペレーション】

dest = ~dest;

【機能】

• dest を論理反転し、dest に格納します。

【選択可能な 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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	○	○	–	–

条件

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

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

OR

論理和
Or Logical

OR

【構文】

OR.size src,dest

B , W , L

【命令コード/サイクル数】

Page=237

【オペレーション】

dest = dest | src;

【機能】

• dest と src の論理和をとり、dest に格納します。

【選択可能な 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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	−

条件

S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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

レジスタ/メモリの復帰
Pop Data off the Stack

POP

【構文】

POP.size dest

B , W , L

【命令コード/サイクル数】

Page=238

【オペレーション】

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

*1 サイズ指定子(.size)が “.B”、“.W” でもSPには4が加算されます。

【機能】

- スタックから復帰したデータをdestに転送します。
- 使用されるスタックポインタは、Uフラグで示すスタックポインタになります。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

POPC

専用レジスタの復帰
Pop Contorl Register off the Stack

POPC

【構文】

POPC dest

【命令コード/サイクル数】

Page=239

【オペレーション】

```
dest  =  *SP;
SP    =  SP + 4;
```

【機能】

- スタックから復帰したデータをdestで示される専用レジスタに転送します。
- 使用されるスタックポインタは、Uフラグで示すスタックポインタになります。

【選択可能なdest】

dest			
SB	FB	FLG	SP*1
ISP	SVF	SVP	INTB

*1 Uフラグで示すスタックポインタが対象となります。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	*1	*1	*1	*1	*1	*1	*1	*1

*1 destがFLGのときだけ変化します。

【記述例】

```
POPC    SB
POPC    SVF
```

POPM

複数レジスタの復帰
Pop Registers off the Stack

POPM

【構文】
POPM dest

【命令コード/サイクル数】
Page=239

【オペレーション】

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

*1 nは復帰するレジスタの数です。

【機能】

- destで選択したレジスタを一括してスタックから復帰します。
- 使用されるスタックポインタは、Uフラグで示すスタックポインタになります。
- スタック領域から復帰する順序は以下のとおりです。

【選択可能なdest】

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

*1 destには複数のレジスタを指定できます。
*2 バンク1レジスタ直接が使用できます。ただし、レジスタ直接とバンク1レジスタ直接を混載して指定することはできません。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

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

PUSH

レジスタ/メモリ/即値の退避
Push Data on the Stack

PUSH

【構文】

PUSH.size(:format) src

G , S
B , W , L

【命令コード/サイクル数】

Page=239

【オペレーション】

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

*1 サイズ指定子(.size)が “.B”、“.W” でもSPは4減算されます。“.B” のときの上位24ビット、“.W” のときの上位16ビットは不定になります。

【機能】

- srcをスタックに退避します。
- 使用されるスタックポインタは、Uフラグで示すスタックポインタになります。

【選択可能なsrc】 **(フォーマット別のsrcは次のページを参照してください。)**

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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

【フォーマット別src】

Gフォーマット

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

Sフォーマット

src	
#IMMEX	#IMM

PUSHA

実効アドレスの退避
Push Effective Address on the Stack

PUSHA

【構文】

PUSHA src

【命令コード/サイクル数】

Page=241

【オペレーション】

```
SP    =   SP - 4;
*SP   =   &src;
```

【機能】

- srcの実効アドレスをスタックに退避します。
- 使用されるスタックポインタは、Uフラグで示すスタックポインタになります。

【選択可能な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]

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
PUSHA     mem[FB]
PUSHA     mem[A0]
```

PUSHC

専用レジスタの退避
Push Contorl Register on the Stack

PUSHC

【構文】

PUSHC src

【命令コード/サイクル数】

Page=242

【オペレーション】

```
SP    =   SP - 4;
*SP   =   src;
```

【機能】

- destで示される専用レジスタをスタックに退避します。
- 使用されるスタックポインタは、Uフラグで示すスタックポインタになります。

【選択可能なsrc】

src			
SB	FB	FLG	SP*1
ISP	SVF	SVP	INTB

*1 Uフラグで示すスタックポインタが対象となります。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
PUSHC   SB
PUSHC   SVF
```

PUSHM

複数レジスタの退避
Push Registers on the Stack

PUSHM

【構文】

PUSHM src

【命令コード/サイクル数】

Page=242

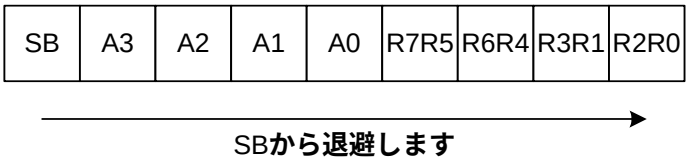
【オペレーション】

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

*1 nは退避するレジスタの数です。

【機能】

- srcで選択したレジスタを一括してスタックに退避します。
- 使用されるスタックポインタは、Uフラグで示すスタックポインタになります。
- スタック領域に退避する順序は以下のとおりです。



【選択可能なsrc】

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

*1 srcには複数のレジスタを指定できます。
*2 バンク1レジスタ直接が使用できます。ただし、レジスタ直接とバンク1レジスタ直接を混載して指定することはできません。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

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

REIT

割り込みからの復帰
Return from Interrupt

REIT

【構文】
REIT

【命令コード/サイクル数】
Page=242

【オペレーション】

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

【機能】

- 割り込み要求が受け付けられたときに退避したPCおよびFLGを復帰し、割り込みルーチンから戻ります。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	*1	*1	*1	*1	*1	*1	*1	*1

*1　スタック上の値になります。

【記述例】
REIT

RMPA

積和演算
Repeat Multiply and Accumulation

RMPA

【構文】

RMPA.size

B , W , L

【命令コード/サイクル数】

Page=243

【オペレーション】

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

*1 R7R5に0を設定して実行したとき、本命令は無視されます。

*2 nはサイズ指定子(.size)に “.B” を指定した場合1、“.W” を指定した場合2、“.L” を指定した場合4です。

【機能】

- A0を被乗数番地、A1を乗数番地、R7R5を回数とする積和演算を行います。演算は符号付きで行い、結果をR3R1:R2R0の64ビットに格納します。
- R7R5に設定可能な最大値は00020000h (131072)です。
- R4、R6は作業用レジスタとして使用され、積和演算中の途中結果はR4:R3R1:R2R0に積算されます。
- 命令終了時のアドレスレジスタの内容、およびR4、R6の内容は、不定となります。
- 命令実行前にR3R1:R2R0には初期値を設定してください。また、R4には、R3R1:R2R0が負のときは “FFFFh” を、正のときは “0000h” を設定してください。
- 命令実行中に割り込み要求があった場合は、演算を中断して割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	−	−	−	−

条件

O ： 演算結果が2^{63}-1または-2^{63}を超えると “1”、それ以外のとき “0” になります。

【記述例】

```
RMPA.W
```

ROLC

キャリー付き左回転
Rotate the bits to the Left with Carry

ROLC

【構文】

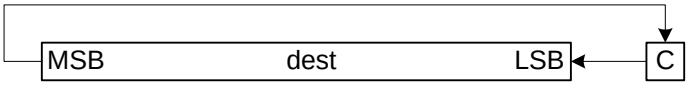
ROLC.size dest

B , W , L

【命令コード/サイクル数】

Page=243

【オペレーション】



【機能】

• destをCフラグを含めて1ビット左へ回転します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	○

条件

S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： シフトアウトしたビットが1のとき “1”、それ以外のとき “0” になります。

【記述例】

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

RORC

キャリー付き右回転
Rotate the bits to the Right with Carry

RORC

【構文】

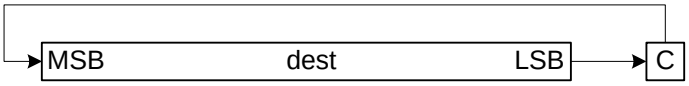
RORC.size dest

B , W , L

【命令コード/サイクル数】

Page=243

【オペレーション】



【機能】

• destをCフラグを含めて1ビット右へ回転します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	○

条件

S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： シフトアウトしたビットが1のとき “1”、それ以外のとき “0” になります。

【記述例】

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

ROT

回転
Rotate the bits

ROT

【構文】

ROT.size src,dest

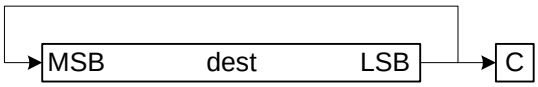
B , W , L

【命令コード/サイクル数】

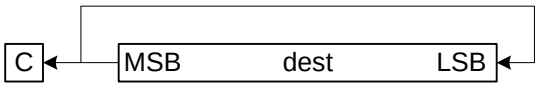
Page=244

【オペレーション】

src ＜ 0のとき



src > 0のとき



【機能】

- destをsrcで指定したビット数ぶんだけ回転します。LSB/MSBからあふれたビットはMSB/LSBとCフラグに転送します。
- 回転方向は、srcの符号で指定します。srcが正のとき左回転、負のとき右回転です。ただし、0のときは回転しません。

【選択可能な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　バンク1レジスタ直接が使用できます。
*2　取りうる範囲は、-8~+8(.B)、-16~+16(.W)、-32~+32(.L)です。
*3　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	−	−	−	−	○	○	−	○

*1　srcが0のとき、フラグは変化しません。

条件
S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： シフトアウトしたビットが1のとき “1”、それ以外のとき “0” になります。

【記述例】

```
ROT.B    #1,R0L      ;左回転
ROT.B    #-1,R0L     ;右回転
ROT.W    R1L,[A0]
ROT.L    R0H,R6R4
```

ROUND

浮動小数点数→整数変換
Round Floating Point to Integer

ROUND

【構文】
ROUND src,dest

【命令コード/サイクル数】
Page=245

【オペレーション】
dest = (long) src;

【機能】
- srcを整数に変換し、destに格納します。
- 演算結果は、フラグレジスタ(FLG)に指定された丸めモードに従って丸められ、また、-2147483648 ~ +2147483647の範囲に収まるようクリップされます。
- srcが不正入力値の場合、演算結果は不定になります。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	○	○	−	−	○	−	○	○	−	○

条件
FO : 不正入力と、演算の結果が -2147483648 または +2147483647 を超えると “1”、それ以外のとき “0” になります。
FU : 不正入力のとき “1”、それ以外のとき “0” になります。
O : 不正入力と、演算の結果が -2147483648 または +2147483647 を超えると “1”、それ以外のとき “0” になります。
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果、すべてのビットが “0” のとき “1”、それ以外のとき “0” になります。
C : 不定になります。

【記述例】
```
ROUND   R2R0,R3R1
ROUND   [A1],R2R0
ROUND   mem[FB],R3R1
```

RTS

サブルーチンからの復帰
Return from Subroutine

RTS

【構文】
RTS

【命令コード/サイクル数】
Page=246

【オペレーション】
PC = *SP;
SP = SP + 4;

【機能】
• サブルーチンから復帰します。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】
RTS

SBB

ボロー付き減算
Subtract with Borrow

SBB

【構文】

SBB.size src,dest

B , W , L

【命令コード/サイクル数】

Page=246

【オペレーション】

dest = dest - src - ~C;

【機能】

• destからsrcとCフラグの反転(ボロー)を減算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O : 符号付き演算のオーバフローを示します。演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

C : 符号なし演算の結果、0以上のとき “1”、それ以外のとき “0” になります。

【記述例】

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

SC*Cnd*　*条件設定* Store Condition Conditionally　SC*Cnd*

【構文】

SC*Cnd*.size　dest

B , W, L

【命令コード/サイクル数】

Page=247

【オペレーション】

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

【機能】

- *Cnd*で示す条件の真偽値をdestに設定します。真の場合は “1” を、偽の場合は “0” を設定します。
- *Cnd*には次の種類があります。

Cnd	条件		式	*Cnd*	条件		式
GEU/C	C == 1	**等しいまたは大きい/** C**フラグが** “1”	≦	LTU/NC	C == 0	**小さい/** C**フラグが** “0”	>
EQ/Z	Z == 1	**等しい/** Z**フラグが** “1”	=	NE/NZ	Z == 0	**等しくない/** Z**フラグが** “0”	≠
GTU	C & ~Z == 1	**大きい**	<	LEU	C & ~Z == 0	**等しいまたは小さい**	≧
PZ	S == 0	**正またはゼロ**	0≦	N	S == 1	**負**	0>
GE	S ^ O == 0	**等しい、または符号付きで大きい**	≦	LE	(S ^ O) | Z == 1	**等しい、または符号付きで小さい**	≧
GT	(S ^ O) | Z == 0	**符号付きで大きい**	<	LT	S ^ O == 1	**符号付きで小さい**	>
O	O == 1	O**フラグが** “1”		NO	O == 0	O**フラグが** “0”	

【選択可能な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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

SCMPU

ストリング比較
Compare Strings until not equal

SCMPU

【構文】

SCMPU.size
　　└──────── B , W

【命令コード/サイクル数】

Page=248

【オペレーション】

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

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

tmp0, tmp1 :**一時レジスタ**

【機能】

- A0で示される比較元番地とA1で示される比較先番地のデータを、比較の結果が不一致になるか、Nullキャラクタ'\0' (=00h)が検出されるまでアドレスの加算方向にストリング比較を行います。
- サイズ指定子(.size)が ".B" のときも ".W" のときも同じ動作になります。
- 命令終了時のアドレスレジスタ(A0,A1)は、不定になります。
- 命令実行中に割り込み要求があった場合は、演算を中断して割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O ： 不定 になります。
S ： 不定になります。
Z ： 双方のストリングが一致していたとき "1"、それ以外のとき "0" になります。
C ： (*A0 - *A1)を符号なしで演算した結果、0以上のとき "1"、それ以外のとき "0" になります。

【記述例】

```
SCMPU.W
```

SHA

算術シフト
Arithmetic Shift

SHA

【構文】

SHA.size src,dest

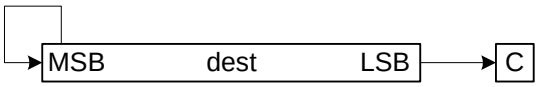
B , W , L

【命令コード/サイクル数】

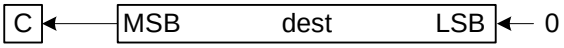
Page=248

【オペレーション】

src ＜ 0**のとき**



src > 0**のとき**



【機能】

- destをsrcで指定したビット数ぶんだけ算術シフトします。LSB/MSBからあふれたビットはCフラグに転送します。
- シフト方向は、srcの符号で指定します。srcが正のとき左シフト、負のとき右シフトです。ただし、0のときはシフトしません。

【選択可能な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　バンク1レジスタ直接が使用できます。
*2　取りうる範囲は、-8~+8(.B)、-16~+16(.W)、-32~+32(.L)です。
*3　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	−	−	○	−	○	○	−	○

*1　srcが0のとき、フラグは変化しません。

条件
O ： 左シフトの場合、演算結果のMSBとシフトアウトしたビットが全て同じ値のとき(シフト中に符号が変化しなかったとき)“0”、それ以外のとき“1”になります。右シフトの場合、“0”になります。
S ： 演算の結果、MSBが“1”になると“1”、それ以外のとき“0”になります。
Z ： 演算の結果が0のとき“1”、それ以外のとき“0”になります。
C ： シフトアウトしたビットが“1”のとき“1”、それ以外のとき“0”になります。

【記述例】

```
SHA.B    #3,R0L      ;左シフト
SHA.B    #-3,R0L     ;右シフト
SHA.W    R1L,[A0]
SHA.L    R2H,R6R4
```

SHL

論理シフト
Logical Shift

SHL

【構文】

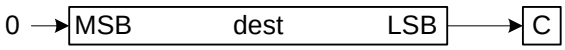
SHL.size(:format)　src,dest

G , Q
B , W , L

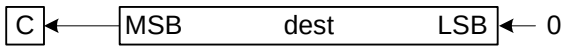
【命令コード/サイクル数】
Page=249

【オペレーション】

src ＜ 0**のとき**



src > 0**のとき**



【機能】

- destをsrcで指定したビット数ぶんだけ論理シフトします。LSB/MSBからあふれたビットはCフラグに転送します。
- シフト方向は、srcの符号で指定します。srcが正のとき左シフト、負のとき右シフトです。

【選択可能なsrc/dest**】**　　(**フォーマット別の**src/dest**は次のページを参照してください。**)

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　バンク1レジスタ直接が使用できます。
*2　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*3　取りうる範囲は、-8~+8(.B)、-16~+16(.W)、-32~+32(.L)です。
*4　取りうる範囲は、-8≦#IMM:4≦+7(≠0)です。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	−	−	−	−	○	○	−	○

*1　srcが0のとき、フラグは変化しません。

条件
S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。
C ： シフトアウトしたビットが “1” のとき “1”、それ以外のとき “0” になります。

【記述例】

```
SHL.B    #1,R0L      ;左シフト
SHL.B    #-1,R0L     ;右シフト
SHL.W    R1L,[A0]
SHL.L    R2H,R6R4
```

【フォーマット別src/dest】

Gフォーマット

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　バンク1レジスタ直接が使用できます。
*2　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

Qフォーマット

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　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。
*2　取りうる範囲は、-8≦#IMM:4≦+7(≠0)です。

SIN

ストリング入力
Input Strings

SIN

【構文】

SIN.size

B , W , L

【命令コード/サイクル数】

Page=251

【オペレーション】

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

*1　R7R5に0を設定して実行したとき、本命令は無視されます。
*2　サイズ指定子(.size)が “.B” のとき1、“.W” のとき2、“.L” のとき4になります。

【機能】

- A0で示される固定の転送元番地からA1で示される転送先番地へ、R7R5で指定される回数ぶんアドレスの加算方向にストリング転送を行います。
- 転送元番地はA0、転送先番地はA1、転送回数はR7R5に設定します。
- R7R5に設定可能な最大値は00FFFFFFhです。
- 命令終了時のA1は、最後に転送したデータの次の番地を示します。
- 命令実行中に割り込み要求があった場合は、1データ転送後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

SIN.W

SMOVB

逆方向ストリング転送
Move Strings Backward

SMOVB

【構文】

SMOVB.size
　　└──── B , W , L

【命令コード/サイクル数】

Page=251

【オペレーション】

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

*1　R7R5に0を設定して実行したとき、本命令は無視されます。
*2　サイズ指定子(.size)が “.B” のとき1、“.W” のとき2、“.L” のとき4になります。

【機能】

- A0で示される転送元番地からA1で示される転送先番地へ、R7R5で指定される回数、アドレスの減算方向にストリング転送を行います。
- 転送元番地はA0、転送先番地はA1、転送回数はR7R5に設定します。
- R7R5に設定可能な最大値は00FFFFFFhです。
- 命令終了時のアドレスレジスタ(A0,A1)は、最後に転送したデータの次の番地を示します。
- 命令実行中に割り込み要求があった場合は、1データ転送後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SMOVB.W
```

SMOVF

順方向ストリング転送
Move Strings Forward

SMOVF

【構文】

SMOVF.size

B , W , L , Q

【命令コード/サイクル数】

Page=251

【オペレーション】

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

*1 R7R5に0を設定して実行したとき、本命令は無視されます。
*2 サイズ指定子(.size)が “.B” のとき1、“.W” のとき2、“.L” のとき4、“.Q”のとき8になります。

【機能】

- A0で示される転送元番地からA1で示される転送先番地へ、R7R5で指定される回数、アドレスの加算方向にストリング転送を行います。
- 転送元番地はA0、転送先番地 はA1、転送回数はR7R5に設定します。
- R7R5に設定可能な最大値は00FFFFFFhです。
- 命令終了時のアドレスレジスタ(A0,A1)は、最後に転送したデータの次の番地を示します。
- 命令実行中に割り込み要求があった場合は、1データ転送後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SMOVF.W
```

SMOVU

ストリング転送
Move Strings While Unequal to Zero

SMOVU

【構文】

SMOVU.size
　　└──── B , W

【命令コード/サイクル数】

Page=252

【オペレーション】

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

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

tmp0　:**一時レジスタ**

【機能】

- A0で示される転送元番地からA1で示される転送先番地へ、Nullキャラクタ '\0' (=00h)が検出されるまでアドレスの加算方向にストリング転送を行います。転送はNullキャラクタ転送後に終了します。
- 転送元番地はA0、転送先番地はA1に設定します。
- サイズ指定子(.size)が ".B" のときも ".W" のときも同じ動作になります。
- 命令終了時のアドレスレジスタ(A0, A1)は、不定となります。
- 命令実行中に割り込み要求があった場合は、1データ転送後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SMOVU.W
```

SOUT

ストリング出力
Output Strings

SOUT

【構文】

SOUT.size
　　└──── B , W , L

【命令コード/サイクル数】

Page=252

【オペレーション】

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

*1　R7R5に0を設定して実行したとき、本命令は無視されます。
*2　サイズ指定子(.size)が “.B” のとき1、“.W” のとき2、“.L”のとき4になります。

【機能】

- A0で示される転送元番地からA1で示される固定の転送先番地へ、R7R5で指定される回数、アドレスの加算方向にストリング転送を行います。
- 転送元番地はA0、転送先番地はA1、転送回数はR7R5に設定します。
- R7R5に設定可能な最大値は00FFFFFFhです。
- 命令終了時のA0は、最後に転送したデータの次の番地を示します。
- 命令実行中に割り込み要求があった場合は、1データ転送後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SOUT.W
```

SSTR

ストリングストア
Store Strings

SSTR

【構文】

SSTR.size

B , W , L , Q

【命令コード/サイクル数】

Page=252

【オペレーション】

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

*1 R7R5に0を設定して実行したとき、本命令は無視されます。
*2 サイズ指定子(.size)が “.B” のときR0L、“.W” のときR0、“.L” のときR2R0、“.Q”のときR3R1R2R0になります。
*3 サイズ指定子(.size)が “.B” のとき1、“.W” のとき2、“.L” のとき4、“.Q”のとき8になります。

【機能】

- R0L/R0/R2R0/R3R1R2R0の内容をA1で示される転送先番地へ、R7R5で指定される回数、アドレスの加算方向にストリングストアを行います。
- 転送先番地はA1、転送回数はR7R5に設定します。
- R7R5に設定可能な最大値は00FFFFFFhです。
- 命令終了時のA1は、最後に書き込んだデータの次の番地を示します。
- 命令実行中に割り込み要求があった場合は、1データ転送後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

SSTR.W

STC

専用レジスタから転送
Store from Control Register

STC

【構文】

STC src,dest

【命令コード/サイクル数】

Page=253

【オペレーション】

dest = src;

【機能】

• src を dest に転送します。

【選択可能な 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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

STCTX

コンテキストの退避
Store Context

STCTX

【構文】
STCTX src,dest

【命令コード/サイクル数】
Page=254

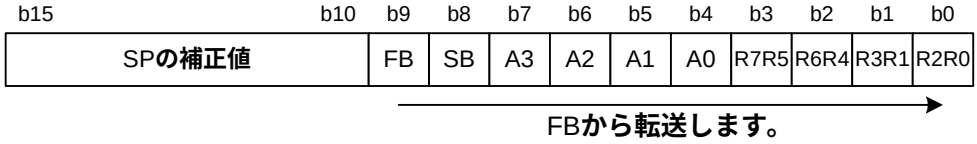
【オペレーション】

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

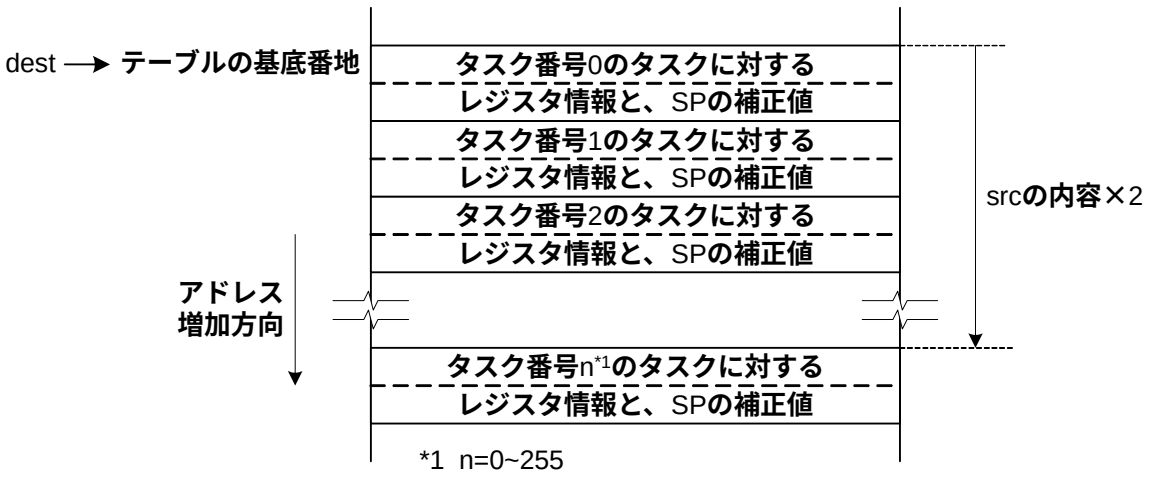
*1 nは退避するレジスタの数です。

【機能】

- タスクのコンテキストをスタック領域に退避します。
- srcにはタスク番号が格納されているRAMの番地を、destにはテーブルデータの先頭番地を設定してください。
- タスク番号によってテーブルデータの中から必要なレジスタ情報を指定し、そのレジスタ情報に従って各レジスタをスタック領域に転送します。その後、スタックポインタ(SP)にSPの補正値を減算します。SPの補正値には転送するレジスタの合計バイト数を設定してください。なお、テーブルデータに “0000h” は設定しないでください。
- 転送するレジスタの情報は次のとおり構成されています。ビットが “1” のとき転送するレジスタ、“0” のとき転送しないレジスタを示します。



- テーブルデータは次のとおり構成されています。destで示した番地がテーブルの基底番地となり、srcの内容の2倍をオフセットとする番地に格納されたワードデータのビット0~9にレジスタの情報、ビット10~15がスタックポインタの補正値を示します。



【選択可能なsrc/dest**】**

src	dest
abs:16	dsp:24

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
STCTX       ram,rom_tbl
```

STNZ

条件付き転送
Store on Not Zero

STNZ

【構文】

STNZ.size src,dest

B , W , L

【命令コード/サイクル数】

Page=254

【オペレーション】

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

【機能】

• Z フラグが “0” のとき、src を dest に転送します。“1” のとき dest は変化しません。

【選択可能な 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　間接命令アドレッシングまたはバンク 1 レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

STOP

ストップ
Stop

STOP

【構文】
STOP

【命令コード/サイクル数】
Page=254

【オペレーション】

【機能】
- プログラムの実行を停止します。ストップ/ウェイト復帰用割り込み優先レベル選択ビットよりも高い優先順位の割り込みを受け付けるか、リセットが発生するとプログラムの実行を開始します。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】
STOP

STZ

条件付き転送
Store on Zero

STZ

【構文】

STZ.size src,dest

B , W , L

【命令コード/サイクル数】

Page=255

【オペレーション】

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

【機能】

• Z フラグが “1” のとき、src を dest に転送します。“0” のとき dest は変化しません。

【選択可能な 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　間接命令アドレッシングまたはバンク 1 レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

STZX

条件付き転送
Store according to Zero Flag

STZX

【構文】

STZX.size src1,src2,dest

B , W , L

【命令コード/サイクル数】

Page=255

【オペレーション】

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

【機能】

• Z フラグが “1” のとき、src1 を dest に転送します。“0” のとき src2 を dest に転送します。

【選択可能な 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 間接命令アドレッシングまたはバンク 1 レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

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

SUB

ボローなし減算
Subtract

SUB

【構文】

SUB.size src,dest

B , W , L

【命令コード/サイクル数】

Page=256

【オペレーション】

dest = dest - src;

【機能】

• destからsrcを減算し、destに格納します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O : 符号付き演算のオーバフローを示します。演算の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

C : 符号なし演算の結果、0以上のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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

浮動小数点減算
Subtruct Floating Piont

SUBF

【構文】
SUBF src,dest

【命令コード/サイクル数】
Page=257

【オペレーション】
dest = dest - src ;

【機能】
- destからsrcを減算し、destに格納します。
- 演算結果は、フラグレジスタ(FLG)に指定された丸めモードに従って丸められます。
- 演算結果が最大の正規化数を超えたとき、結果は符号に応じて正の最大値(7F7FFFFFh)、または負の最大値(FF7FFFFFh)になります。
- 演算結果が最小の正規化数を下回ったとき、結果は丸めモードに従ってゼロ(00000000h)、または正の最小値(00800000h)、または負の最小値(80800000h)になります。
- 不正入力に対する演算結果は不定です。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	FO	FU	U	I	O	B	S	Z	D	C
変化	○	○	−	−	○	−	○	○	−	○

条件
FO : 不正入力と、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
FU : 不正入力と、演算の結果が最小の正規化数を下回ったとき “1”、それ以外のとき “0” になります。
O : 不正入力と、演算の結果が最大の正規化数を上回ったとき “1”、それ以外のとき “0” になります。
S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z : 演算の結果、すべてのビットが “0” のとき “1”、それ以外のとき “0” になります。
C : 不定になります。

【記述例】
```
SUBF    R2R0,R3R1
SUBF    [A1],R2R0
SUBF    mem[FB],R3R1
```

SUNTIL

ストリングサーチ
Search Equal String

SUNTIL

【構文】

SUNTIL.size

B , W , L

【命令コード/サイクル数】

Page=258

【オペレーション】

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

*1 サイズ指定子(.size)が “.B” のときR0L、“.W” のときR0、“.L” のときR2R0になります。
*2 R7R5に0を設定して実行したときも、比較は1度だけ実施されますが、命令終了時のA0とフラグは不定になります。
*3 サイズ指定子(.size)が “.B” のとき1、“.W” のとき2、“.L” のとき4加算されます。

【機能】

- R7R5で指定される回数、A0で示される比較先番地からアドレスの加算方向に、R0L/R0/R2R0の内容と一致するデータが現れるまで検索を行います。
- 比較先番地はA0、比較回数はR7R5に設定します。
- R7R5に設定可能な最大値は00FFFFFFhです。
- フラグは「*A0-R0L/R0/R2R0」の演算結果にしたがって変化します。
- 命令終了時のA0は、一致したデータの番地を示します。一致しなかったときは、次のデータの番地を示します。
- 命令終了後のR7R5は、「初期値-比較回数」となります。
- 命令実行中に割り込み要求があった場合は、1データ比較後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O ： 符号付き比較のオーバフローを示します。比較の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。
S ： 比較の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： 一致したとき “1”、それ以外のとき “0” になります。
C ： 符号なし比較の結果、0以上のとき “1”、それ以外のとき “0” になります。

【記述例】

SUNTIL.W

SWHILE

ストリングサーチ
Search Unequal String

SWHILE

【構文】

SWHILE.size
　　└──── B , W , L

【命令コード/サイクル数】

Page=258

【オペレーション】

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

*1　サイズ指定子(.size)が “.B” のときR0L、“.W” のときR0、“.L” のときR2R0になります。
*2　R7R5に0を設定して実行したときも、比較は1度だけ実施されますが、命令終了時のA0とフラグは不定になります。
*3　サイズ指定子(.size)が “.B” のとき1、“.W” のとき2、“.L” のとき4加算されます。

【機能】

- R7R5で指定される回数、A0で示される比較先番地からアドレスの加算方向に、R0L/R0/R2R0の内容と一致しないデータが現れるまで検索を行います。
- 比較先番地はA0、比較回数はR7R5に設定します。
- R7R5に設定可能な最大値は00FFFFFFhです。
- フラグは「*A0-R0L/R0/R2R0」の演算結果にしたがって変化します。
- 命令終了時のA0は、一致しなかったデータの番地を示します。すべて一致したときは、次のデータの番地を示します。
- 命令終了後のR7R5は、「初期値-比較回数」となります。
- 命令実行中に割り込み要求があった場合は、1データ比較後に割り込みを受け付けます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	○	−	○	○	−	○

条件

O ： 符号付き比較のオーバフローを示します。比較の結果、-128(.B)または+127(.B)、-32768(.W)または+32767(.W)、-2147483648(.L)または+2147483647(.L)を超えると “1”、それ以外のとき “0” になります。
S ： 比較の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。
Z ： すべて一致したとき “1”、それ以外のとき “0” になります。
C ： 符号なし比較の結果、0以上のとき “1”、それ以外のとき “0” になります。

【記述例】

```
SWHILE.W
```

TST

テスト
Test Logical

TST

【構文】

TST.size src,dest

B , W , L

【命令コード/サイクル数】

Page=258

【オペレーション】

dest & src;

【機能】

• destとsrcの論理積をとった結果に従って、フラグレジスタ(FLG)の各フラグが変化します。

【選択可能な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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	−

条件

S : 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z : 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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

未定義命令割り込み
Undefined Instruction Interrupt

UND

【構文】
UND

【命令コード/サイクル数】
Page=259

【オペレーション】

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

【機能】
- 未定義命令割り込みが発生します。
- 未定義命令割り込みはノンマスカブル割り込みです。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化*1	○	○	−	−	−	−	○	−

*1　命令実行前のフラグはスタック領域に退避されます。

条件
U ： “0” になります。
I ： “0” になります。
D ： “0” になります。

【記述例】
UND

WAIT

ウェイト
Wait

WAIT

【構文】
WAIT

【命令コード/サイクル数】
Page=259

【オペレーション】

【機能】
- プログラムの実行を停止します。ストップ/ウェイト復帰用割り込み優先レベル選択ビットよりも高い優先順位の割り込みを受け付けるか、リセットが発生するとプログラムの実行を開始します。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】
WAIT

XCHG

交換
Exchange

XCHG

【構文】

XCHG.size src,dest

B , W , L

【命令コード/サイクル数】

Page=260

【オペレーション】

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

tmp0: **一時レジスタ**

【機能】

• src と dest の内容を交換します。

【選択可能な 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　バンク1レジスタ直接が使用できます。
*2　間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
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

排他的論理和
Exclusive Or Logical

XOR

【構文】

XOR.size src,dest

B , W , L

【命令コード/サイクル数】

Page=260

【オペレーション】

dest = dest ^ src;

【機能】

• dest と src の排他的論理和をとり、dest に格納します。

【選択可能な 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 間接命令アドレッシングまたはバンク1レジスタ直接が使用できます。

【フラグ変化】

フラグ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	○	○	−	−

条件

S ： 演算の結果、MSBが “1” になると “1”、それ以外のとき “0” になります。

Z ： 演算の結果が0のとき “1”、それ以外のとき “0” になります。

【記述例】

```
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 インデックス命令

本節ではインデックス命令について命令ごとに説明しています。

インデックス命令は配列に対応した命令で、インデックス命令の次に実行する命令の src、dest が示すアドレスにインデックス命令の src の内容を符号なしで加算し実効アドレスとします。

モディファイできる範囲は 0~FFFFFFFFh の 4GB ぶんです。インデックス命令の src の範囲は次に実行する命令のサイズ指定子により以下のようになります。

次に実行する命令のサイズ指定子	インデックス命令のサイズ指定子	src の範囲	加算される値
.B	.B	0~FFh	src の内容
	.W	0~FFFFh	
	.L	0~FFFFFFFFh	
.W	.B	0~FFh	src の内容 × 2
	.W	0~FFFFh	
	.L	0~7FFFFFFFh	
.L	.B	0~FFh	src の内容 × 4
	.W	0~FFFFh	
	.L	0~3FFFFFFFh	

インデックス命令の直後には、割り込み要求を受け付けません。

インデックス命令には以下に示す 4 種類があります。

INDEXB
INDEX1
INDEX2
BITINDEX

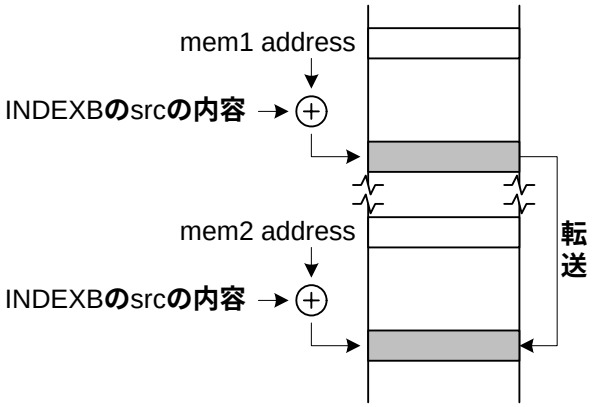
3.3.1 INDEXB.size src

INDEXB (Index Both Operands) 命令は次に実行する命令の src、dest が示すアドレスに INDEXB 命令の src の内容を符号なしで加算し実効アドレスとします。

INDEXB 命令の次に実行する命令では src、dest ともにメモリをアクセスするアドレッシングを選択してください。

例 1: 次命令のサイズ指定子が ".B" の場合

```
INDEXB.W    src
MOV.B:G     mem1, mem2
             \   /
            メモリ
```



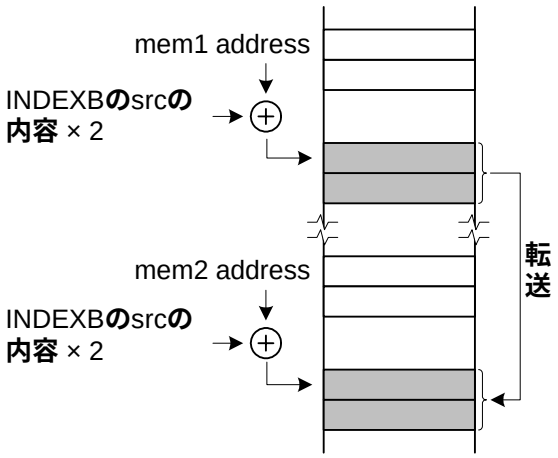
C 言語での動作説明

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

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

例2: 次命令のサイズ指定子が ".W" の場合

```
INDEXB.W    src
MOV.W:G     mem1, mem2
             ＼ ／
            メモリ
```



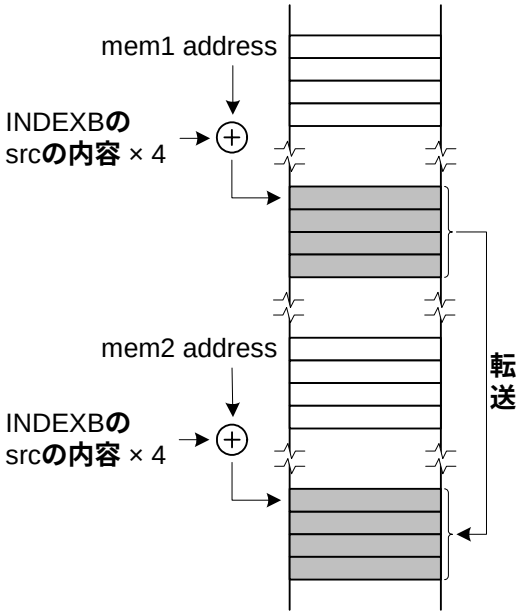
C言語での動作説明

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

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

例3: 次命令のサイズ指定子が ".L" の場合

```
INDEXB.W    src
MOV.L:G     mem1, mem2
             ＼ ／
            メモリ
```



C言語での動作説明

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

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

INDEXB命令がモディファイする命令

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命令のsrcとdest

*1 Gフォーマットのみ使用できます。
*2 ADD命令のdestにSPは使用できません。
*3 src (mem1)については符号拡張/ゼロ拡張前のサイズ、dest (mem2)については符号拡張/ゼロ拡張後のサイズに依存して、INDEXBのsrcを何倍するかが決まります。
*4 MOV命令のsrcまたはdestにdsp:8[SP]は使用できません。

上記以外の命令をINDEXB命令の次に使用しないでください。

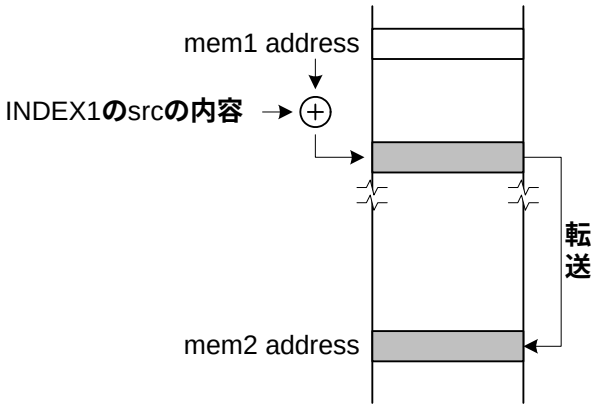
3.3.2 INDEX1.size src

INDEX1 (Index 1st Operand) 命令は次に実行する命令の第一オペランドが示すアドレスにINDEX1命令のsrcの内容を符号なしで加算し実効アドレスとします。

INDEX1 命令の次に実行する命令では第一オペランドにメモリをアクセスするアドレッシングを選択してください。

例1: 次命令のサイズ指定子が ".B" の場合

```
INDEX1.W    src
MOV.B:G     mem1, mem2
              |
            メモリ
```



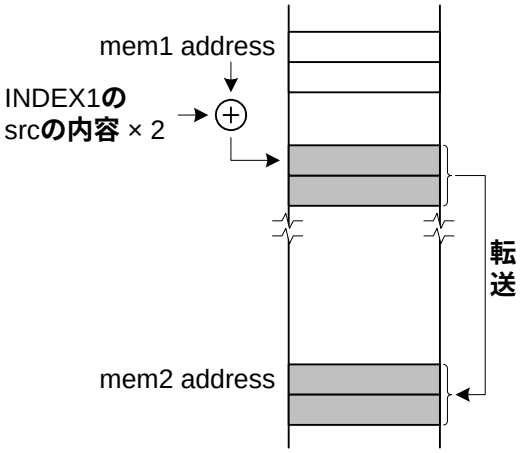
C言語での動作説明

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

mem2 = mem1[src];
```

例2: 次命令のサイズ指定子が ".W" の場合

```
INDEX1.W    src
MOV.W:G     mem1, mem2
              |
            メモリ
```



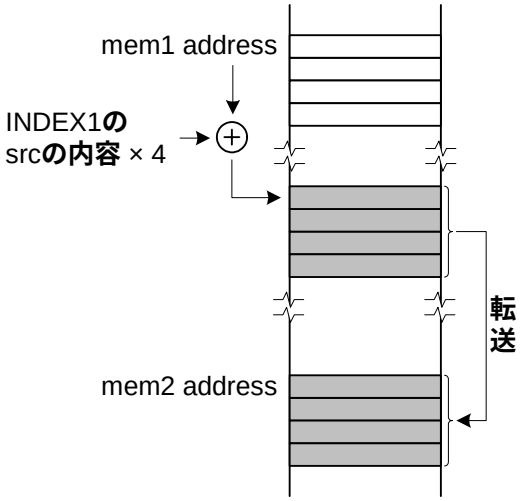
C言語での動作説明

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

mem2 = mem1[src];
```

例3: 次命令のサイズ指定子が ".L" の場合

```
INDEX1.W    src
MOV.L:G     mem1, mem2
            |
          メモリ
```



C言語での動作説明

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

mem2 = mem1[src];
```

INDEX1**命令がモディファイする命令**

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命令の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命令のdest

*1 Gフォーマットのみ指定できます。
*2 ADD命令のdestにSPは使用できません。
*3 符号拡張/ゼロ拡張前のサイズに依存して、INDEX1のsrcを何倍するかが決まります。
*4 MOV命令のsrcにdsp:8[SP]は使用できません。
*5 MOV:S src,R2R0/R0/R0L、MOV:S.L src,A0で使用できます。
*6 MOV*Dir* src,R0Lで使用できます。サイズ指定子は ".B" 扱いとなります。
*7 MOV:G dsp:8[SP],destで使用できます。
*8 MOV:S R2R0/R0/R0L,dest、MOV:S #IMM,destで使用できます。
*9 MOV*Dir* R0L,destで使用できます。サイズ指定子は ".B" 扱いとなります。

上記以外の命令をINDEX1命令の次に使用しないでください。

3.3.3 INDEX2.size src

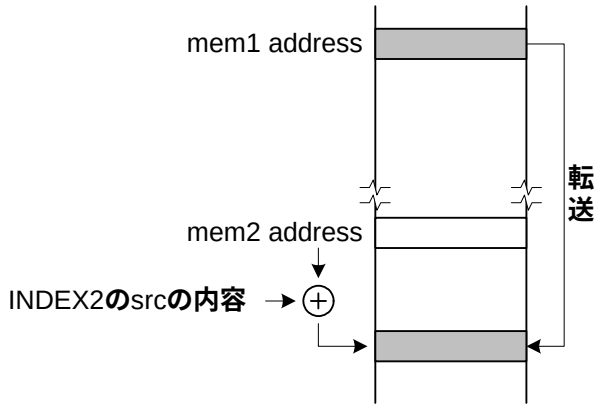
INDEX2 (Index 2nd Operand) 命令は次に実行する命令の第二オペランドが示すアドレスに INDEX2 命令の src の内容を符号なしで加算し実効アドレスとします。

INDEX2 命令の次に実行する命令では dest にメモリをアクセスするアドレッシングを選択してください。

例1: 次命令のサイズ指定子が ".B" の場合

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

メモリ



C言語での動作説明

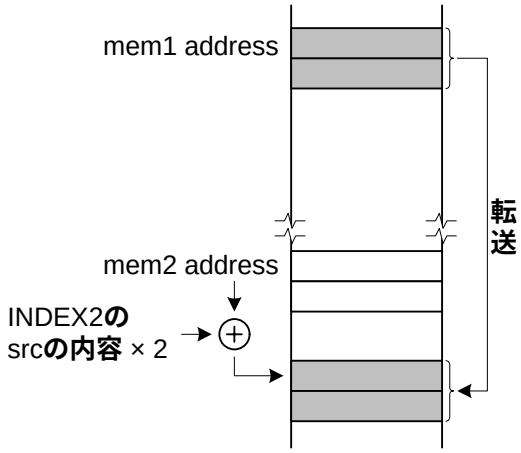
```
short   src;
char    mem1, mem2[];

mem2[src] = mem1;
```

例2: 次命令のサイズ指定子が ".W" の場合

```
INDEX2.W    src
MOV.W:G     mem1, mem2
```

メモリ



C言語での動作説明

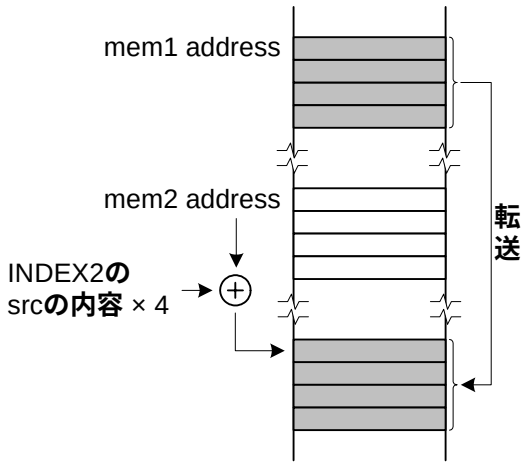
```
short   src;
short   mem1, mem2[];

mem2[src] = mem1;
```

例3: **次命令のサイズ指定子が** ".L" **の場合**

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

メモリ



C**言語での動作説明**

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

mem2[src] = mem1;
```

INDEX2**命令がモディファイする命令**

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命令のdest

*1 Gフォーマットのみ使用できます。
*2 ADD命令のdestにSPは使用できません。
*3 符号拡張/ゼロ拡張後のサイズに依存して、INDEX2のsrcを何倍するかが決まります。
*4 MOV命令のsrcおよびdestにdsp:8[SP]は使用できません。

上記以外の命令をINDEX2命令の次に使用しないでください。

3.3.4 BITINDEX.size src

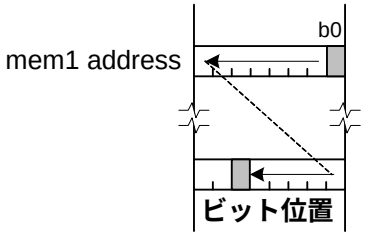
BITINDEX 命令 (Bit Index) は次に実行する命令のオペランドが示すアドレスのビット 0 から BITINDEX 命令の src で示したビット数だけ離れたビットを対象とします。

BITINDEX 命令の次に実行する命令はビット命令としてください。また、オペランドにはメモリをアクセスするアドレッシングを選択してください。

例:

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

ビット命令 **無効となります** **メモリ**



BITINDEX **命令がモディファイする命令**

BTST:G[*1] 命令の src

BCLR、BM*Cnd*、BNOT、BSET、BTSTC、BTSTS 命令の dest

*1 G フォーマットのみ使用できます。

上記以外の命令を BITINDEX 命令の次に使用しないでください。

3.3.5 インデックス命令の次に実行できる命令

それぞれのインデックス命令の次に実行できる命令を下表に示します。

インデックス命令	有効命令	
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 **上記命令の**src**と**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 **上記命令の**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 **上記命令の**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 **上記命令の**dest	
BITINDEX.B/W/L	BTST:G*1 **上記命令の**src	BCLR、BM*Cnd*、BNOT、BSET、BTSTC、BTSTS **上記命令の**dest

*1 Gフォーマットのみ使用できます。
*2 ADD命令のdestにSPは使用できません。
*3 MOV命令のsrcまたはdestにdsp:8[SP]は使用できません。
*4 MOV命令のsrcにdsp:8[SP]は使用できません。
*5 MOV:S src,R0L/R0/R2R0、MOV:S.L src,A0で使用できます。
*6 MOV*Dir* src,R0Lで使用できます。
*7 MOV:G dsp:8[SP],destで使用できます。
*8 MOV:S R0L/R0/R2R0,dest、MOV:S #IMM,destで使用できます。
*9 MOV*Dir* R0L,destで使用できます。

3.3.6 アドレッシングモード

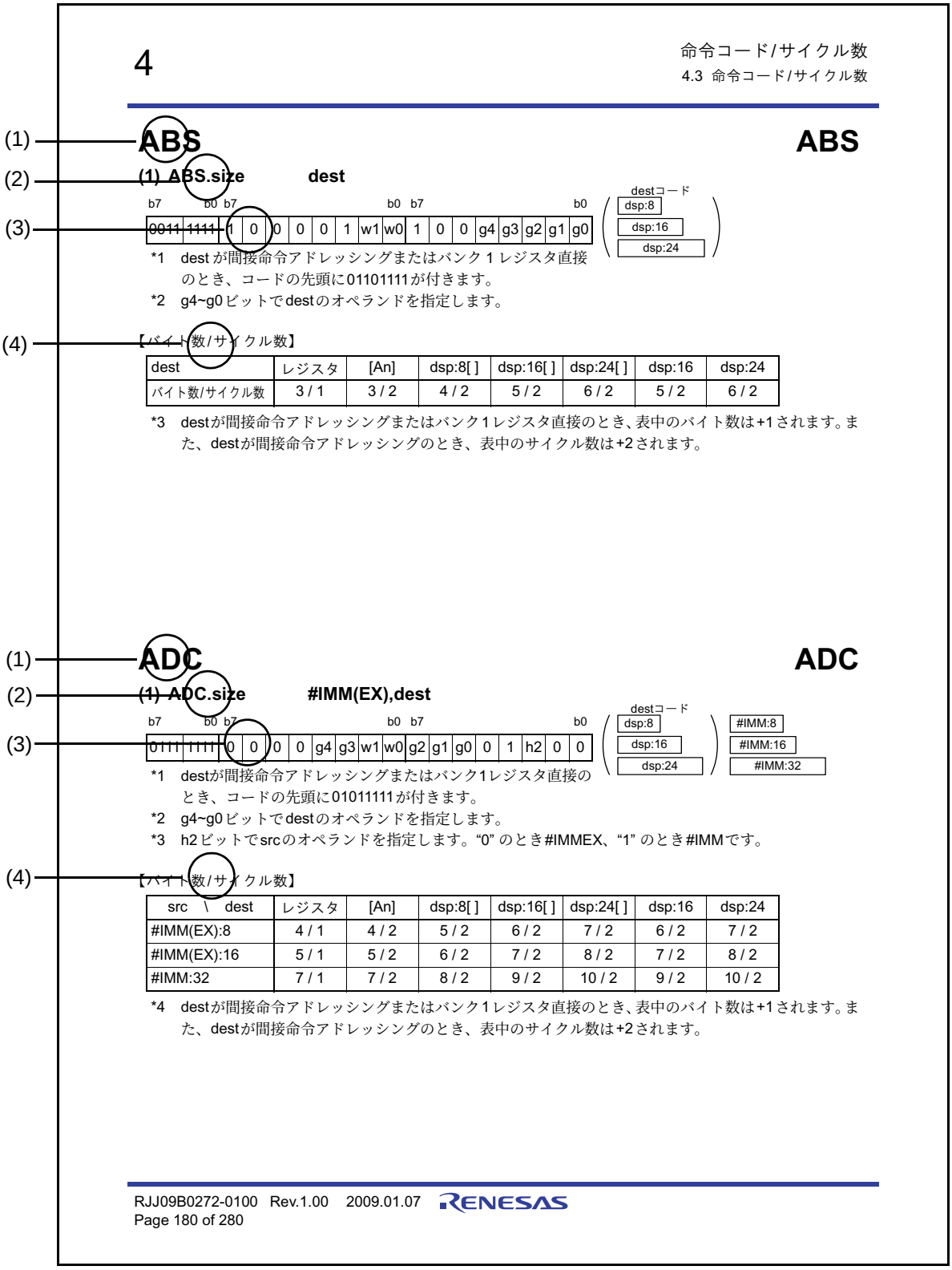
インデックス命令の次に実行できる命令で有効となるアドレッシングモードを下表に示します。それぞれの命令で間接アドレッシングモードが使用できます。

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. 命令コード/サイクル数

4.1 本章の見方

本章は命令コード、サイクル数をオペコードごとに説明しています。
本章の見方について以下に実例をあげて示します。



4

命令コード/サイクル数
4.3 命令コード/サイクル数

(1) **ABS** **ABS**

(2) **(1) ABS.size dest**

(3)

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コード: dsp:8 / dsp:16 / dsp:24

*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2 g4~g0 ビットで dest のオペランドを指定します。

(4) 【バイト数/サイクル数】

dest	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、表中のバイト数は+1されます。また、dest が間接命令アドレッシングのとき、表中のサイクル数は+2されます。

(1) **ADC** **ADC**

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

(3)

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コード: dsp:8 / dsp:16 / dsp:24 — #IMM:8 / #IMM:16 / #IMM:32

*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0 ビットで dest のオペランドを指定します。
*3 h2 ビットで src のオペランドを指定します。“0” のとき #IMMEX、“1” のとき #IMM です。

(4) 【バイト数/サイクル数】

src \ dest	レジスタ	[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 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、表中のバイト数は+1されます。また、dest が間接命令アドレッシングのとき、表中のサイクル数は+2されます。

RJJ09B0272-0100 Rev.1.00 2009.01.07
Page 180 of 280

(1) ニーモニック

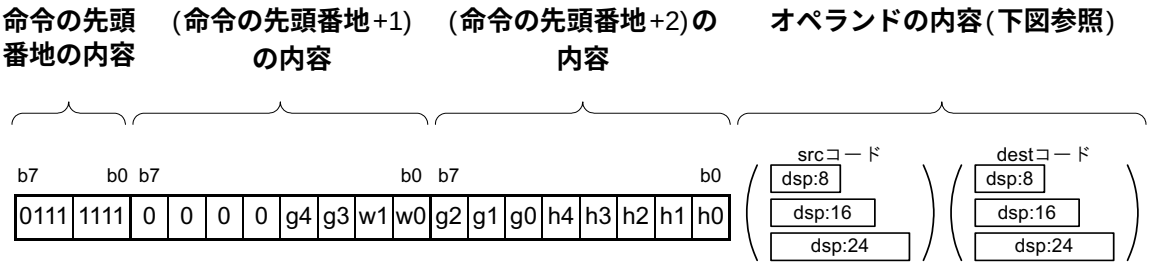
本ページで説明するニーモニックを示しています。

(2) 構文

命令の構文を記号で示しています。

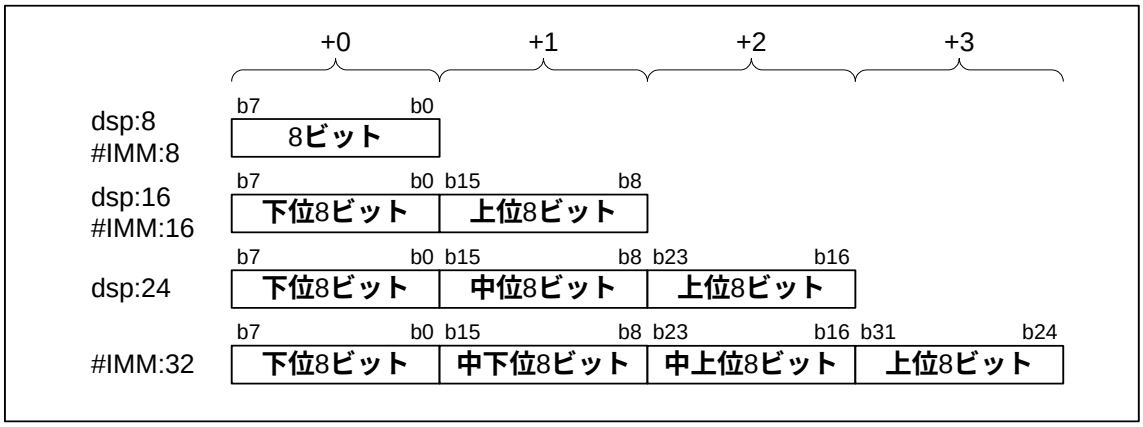
(3) 命令コード

命令コードを示しています。( )内は選択するsrc/destによって省略されます。



w1,w0ビット、g4~g0ビット、h4~h0ビットなどの内容については「4.2 アドレッシング」を参照してください。

オペランドの内容(上記例では(命令の先頭番地+3)以降)は下記のとおり配置されます。



(4) バイト数/サイクル数表

本命令の実行に必要なサイクル数と命令のバイト数を示しています。
サイクル数は最短時のものを記載しているため、次の場合などには増える可能性があります。

- 命令キューバッファに取り込まれているバイト数が少ない場合
- ライトサイクルとリードサイクルが重なりオペランドアクセスが待たされた場合
- SFR領域や外部メモリ領域をアクセスした場合
- バスサイクルに対してウェイトを挿入した場合

スラッシュの左側がバイト数、右側がサイクル数です。

4.2 アドレッシング

4.2.1 演算長の指定

演算長は命令コード中のw1,w0ビットで指定します。

表 4.1 演算長の指定

w1	w0	演算長*1	#IMMEXのオペランドと演算長
0	0	バイト	8ビット即値、ワード
0	1	ワード	8ビット即値、ロング
1	0	ロング	16ビット即値、ロング
1	1	予約	予約

*1 2オペランド命令で、srcが#IMMEXのときw1,w0ビットは#IMMEXのルールに従います。

4.2.2 オペランドの指定

オペランドの指定方法は命令、命令フォーマットにより異なります。
以下、アドレッシングごとにオペランドの指定方法を示します。

(1) 一般命令アドレッシング・ジェネリック形式

ジェネリック形式で使用されるオペランドgen1,gen2は、それぞれg4~g0ビット、h4~h0ビットで指定します。

表 4.2 ジェネリック形式でのオペランド指定

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 destに#IMMEX、#IMMは指定できません。

(2) 一般命令アドレッシング・クイック形式

クイック形式で使用されるオペランド#IMM:4は命令コード中のq3~q0ビットで指定します。

表 4.3 クイック形式でのオペランド指定(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

また、ADD命令のクイック形式で使用される#IMM:3は命令コード中のq2~q0ビットで指定します。

表 4.4 クイック形式でのオペランド指定(2)

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

(3) 一般命令アドレッシング・ショート形式

ショート形式で使用されるオペランドgen0は命令コード中のs1,s0ビットで指定します。

表 4.5 ショート形式でのオペランド指定

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 MOV:S src,R0L/R0/R2R0の場合に限りR0H/R2/R3R1**が使用されます。**

(4) レジスタ直接

レジスタ直接アドレッシングのみ使用可能なEDIV、EDIVU、EDIVX、EMUL、EMULU、EXTZ:S、MOVA命令のdest、ROT、SHA、SHL、XCHG命令のsrcで使用されるオペランドgregは、q2~q0ビットで指定します。命令ごとに下記のとおりのレジスタが使用されます。

表 4.6　EDIV、EDIVU、EDIVX命令でのオペランド指定

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、EMULU命令でのオペランド指定

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命令でのオペランド指定

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命令でのオペランド指定

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) 専用レジスタ直接(CPU)

専用レジスタ直接アドレッシングで使用されるオペランドcregは、q2~q0ビットで指定します。

表 4.10　専用レジスタ直接アドレッシングでのオペランド指定

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) 専用レジスタ直接(DMAC,VCT)

DMAC関連の専用レジスタとベクタレジスタ(VCT)をオペランドに指定するときは、命令コードの最後に#IMM:8を付加し、各レジスタは#IMM:8のビット列で指定します。

表 4.11　専用レジスタ直接アドレッシングでのレジスタ指定(1)

#IMM:8(16進)	レジスタ
03h	VCT
00h~07h (03hを除く)	チャネル0 DMAレジスタ
08h~0Fh (0Bhを除く)	チャネル1 DMAレジスタ
10h~17h (13hを除く)	チャネル2 DMAレジスタ
18h~1Fh (1Bhを除く)	チャネル3 DMAレジスタ

表 4.12　専用レジスタ直接アドレッシングでのレジスタ指定(2)

#IMM:8下位3ビット			レジスタ	
0	0	0	DSAn*1	DMAソースアドレスレジスタ
0	0	1	DDAn*1	DMAデスティネーションアドレスレジスタ
0	1	0	DCTn*1	DMAターミナルカウントレジスタ
0	1	1	予約	
1	0	0	DSRn*1	DMAソースアドレスリロードレジスタ
1	0	1	DDRn*1	DMAデスティネーションアドレスリロードレジスタ
1	1	0	DCRn*1	DMAターミナルカウントリロードレジスタ
1	1	1	DMDn*1	DMAモードレジスタ

*1　nはチャネル番号

(7) プログラムカウンタ相対・ショート形式

プログラムカウンタ相対・ショート形式で使用されるオペランドdsp3は、q2~q0ビットで指定します。

表 4.13　プログラムカウンタ相対・ショート形式でのオペランド指定

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) ビット命令アドレッシング

ビット命令アドレッシングで使用されるオペランドbitは、b2~b0ビットで指定します。

表 4.14　ビット命令アドレッシングでのビット位置指定

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直接アドレッシングで使用されるオペランドflgは、q2~q0ビットで指定します。

表 4.15 フラグ直接アドレッシングでのオペランド指定

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 条件の指定

条件は命令コード中のc3~c0ビットで指定します。

表 4.16 条件コードの指定

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	**予約**	1	1	1	1	**予約**

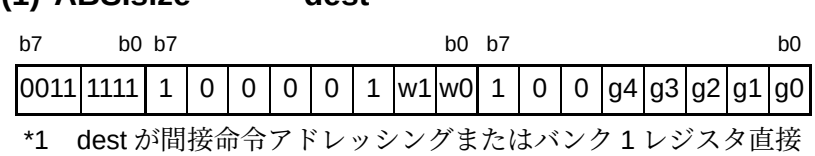
4.3 命令コード/サイクル数

次ページよりR32C/100シリーズの命令コード/サイクル数の詳細説明を示します。

ABS

ABS

(1) ABS.size dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

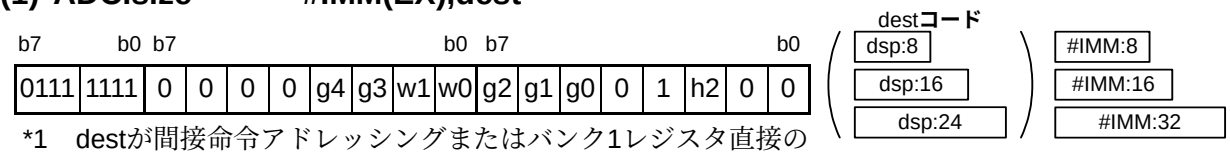
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ADC

ADC

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

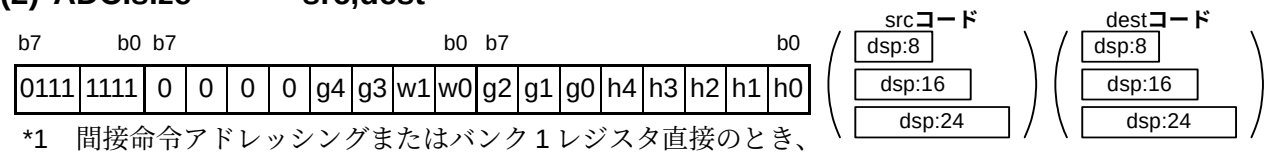
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ADC ADC

(2) ADC.size src,dest



*1 間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

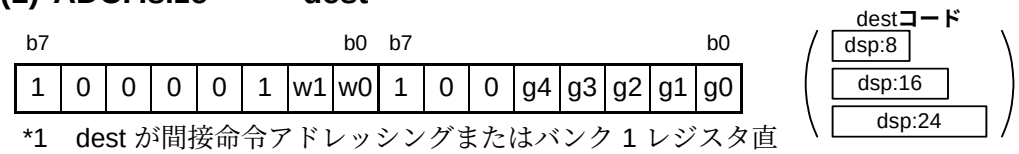
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

ADCF ADCF

(1) ADCF.size dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

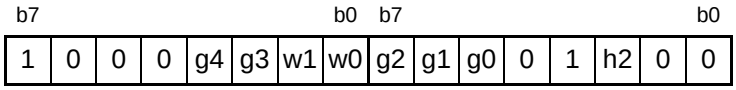
【バイト数/サイクル数】

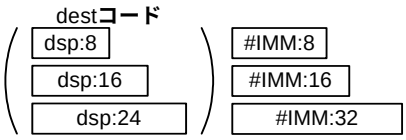
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ADD ADD

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





*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。
*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

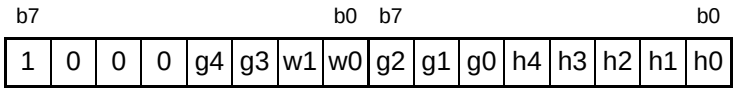
【バイト数/サイクル数】

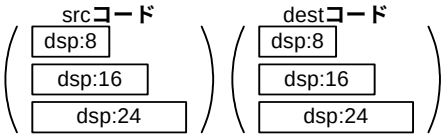
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ADD ADD

(2) ADD.size:G src,dest





*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111
*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

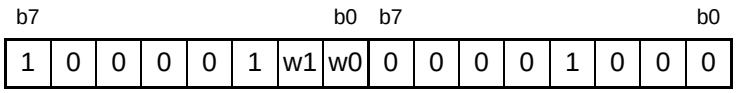
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

ADD　　ADD

(3) ADD.L:G　　#IMM,SP



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

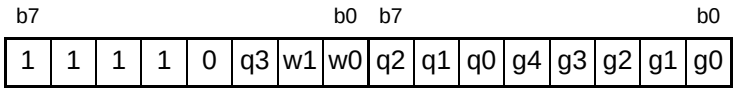
*1　w1,w0ビットで#IMMのオペランド長を指定します。
*2　#IMMはw1,w0にかかわらず32ビットに符号拡張されます。

【バイト数/サイクル数】

src	#IMM:8	#IMM:16	#IMM:32
バイト数/サイクル数	3 / 2	4 / 2	6 / 2

ADD　　ADD

(4) ADD.size:Q　　#IMM:4,dest



destコード
dsp:8
dsp:16
dsp:24

*1　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　q3~q0の4ビットで#IMM:4を、g4~g0ビットでdestのオペランドを指定します。

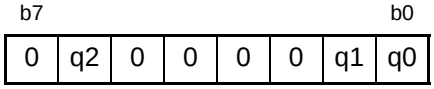
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ADD　　ADD

(5) ADD.L:Q　　#IMM:3,SP



*1　q2~q0ビットで#IMM:3を指定します。詳細は「4.2.2 オペランドの指定」を参照してください。

【バイト数/サイクル数】

バイト数/サイクル数	1 / 1

ADD　　ADD

(6) ADD.size:S　　#IMM,dest

b7							b0
0	1	1	1	s1	s0	w1	w0

destコード
- dsp:8
- dsp:16

→
- #IMM:8
- #IMM:16
- #IMM:32

*1　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　s1~s0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

src　\　dest	**レジスタ**	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　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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

destコード
- dsp:8
- dsp:16
- dsp:24

→
- #IMM:8
- #IMM:16
- #IMM:32

*1　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2　g4~g0ビットでdestのオペランドを指定します。
*3　h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

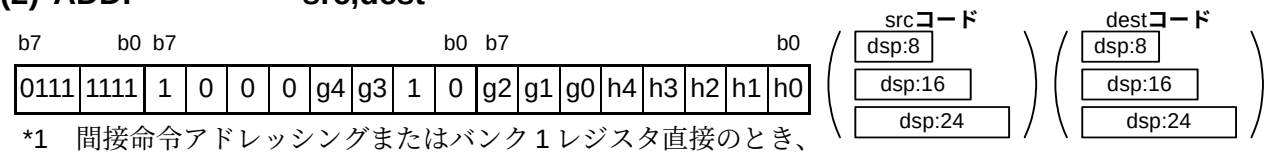
【バイト数/サイクル数】

src　\　dest	**レジスタ**	[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　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ADDF　　ADDF

(2) ADDF　　src,dest



*1　間接命令アドレッシングまたはバンク１レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2　g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

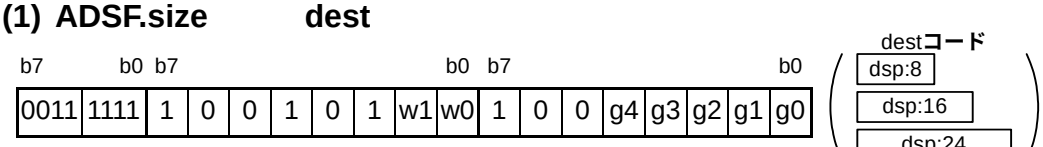
【バイト数/サイクル数】

src　\　dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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　srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク１レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

ADSF　　ADSF

(1) ADSF.size　　dest



*1　destが間接命令アドレッシングまたはバンク１レジスタ直接のとき、コードの先頭に01101111が付きます。

*2　g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

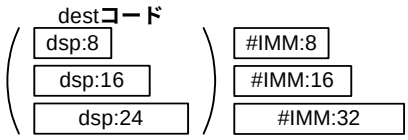
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。
*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

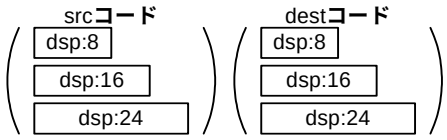
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111
*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

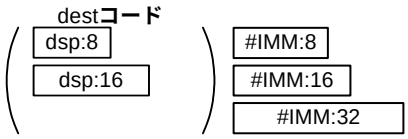
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2
[An]	2 / 2	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3
dsp:8[ ]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:16[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:24[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

AND　　AND

(3) AND.size:S　　#IMM,dest

b7							b0
0	1	1	0	s1	s0	w1	w0



*1　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　s1~s0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

src　\　dest	**レジスタ**	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　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　b2~b0の3ビットでビット位置を、g4~g0ビットでdestのオペランドを指定します。
*3　演算長はバイト固定です。

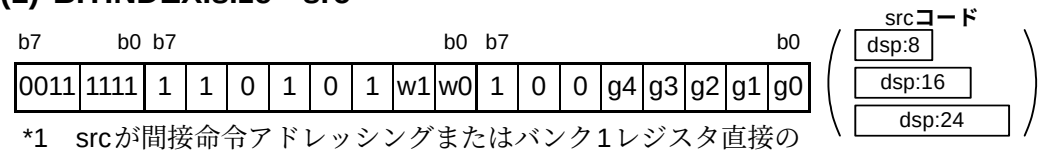
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*4　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

BITINDEX　　BITINDEX

(1) BITINDEX.size　src



*1　srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　g4~g0ビットでdestのオペランドを指定します。

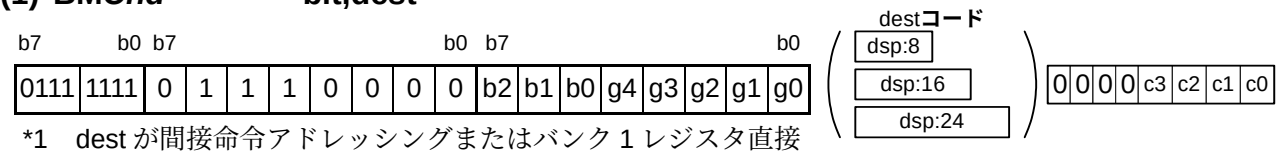
【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*3　srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。
*4　直後のビット命令のサイクル数は+2されます。

BM*Cnd*　　BM*Cnd*

(1) BM*Cnd*　　bit,dest



*1　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　b2~b0の3ビットでビット位置を、g4~g0ビットでdestのオペランドを指定します。
*3　c3~c0ビットで*Cnd*を指定します。詳細は「4.2.3 条件の指定」を参照してください。
*4　演算長はバイト固定です。

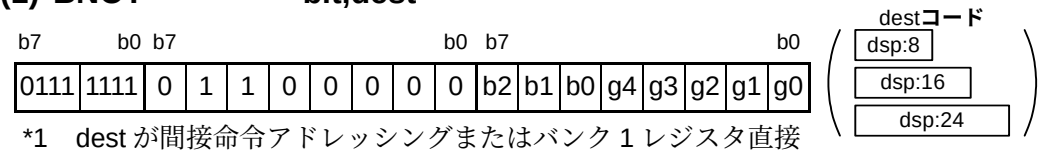
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 5	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5

*5　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

BNOT BNOT

(1) BNOT bit,dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 b2~b0の3ビットでビット位置を、g4~g0ビットでdestのオペランドを指定します。

*3 演算長はバイト固定です。

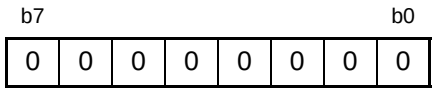
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

BRK BRK

(1) BRK



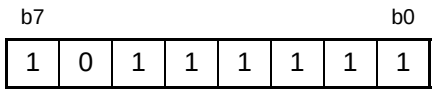
【バイト数/サイクル数】

バイト数/サイクル数	1 / 16

*1 可変ベクタの場合のサイクル数です。固定ベクタの場合は19サイクルになります。

BRK2 BRK2

(1) BRK2

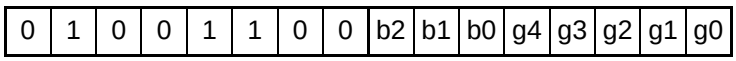


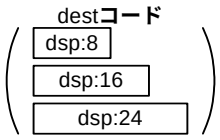
【バイト数/サイクル数】

バイト数/サイクル数	1 / 19

BSET BSET

(1) BSET bit,dest





*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2 b2~b0の3ビットでビット位置を、g4~g0ビットでdestのオペランドを指定します。
*3 演算長はバイト固定です。

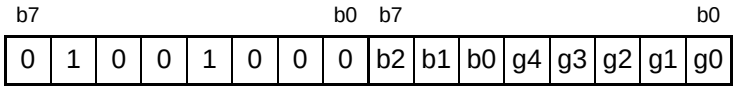
【バイト数/サイクル数】

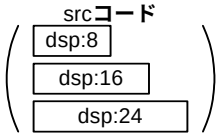
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

BTST BTST

(1) BTST:G bit,src





*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。
*2 b2~b0の3ビットでビット位置を、g4~g0ビットでsrcのオペランドを指定します。
*3 演算長はバイト固定です。

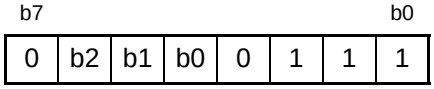
【バイト数/サイクル数】

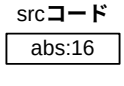
src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

BTST BTST

(2) BTST:S bit,abs:16





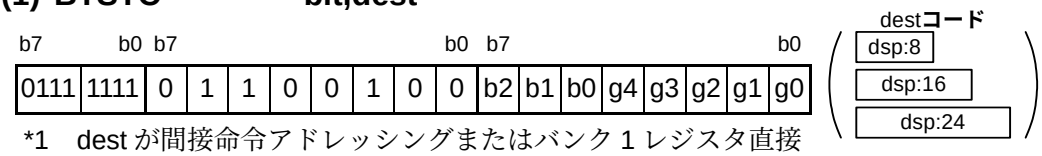
*1 演算長はバイト固定です。
*2 b2~b0の3ビットでビット位置を指定します。

【バイト数/サイクル数】

バイト数/サイクル数	3 / 2

BTSTC BTSTC

(1) BTSTC bit,dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 b2~b0の3ビットでビット位置を、g4~g0ビットでdestのオペランドを指定します。

*3 演算長はバイト固定です。

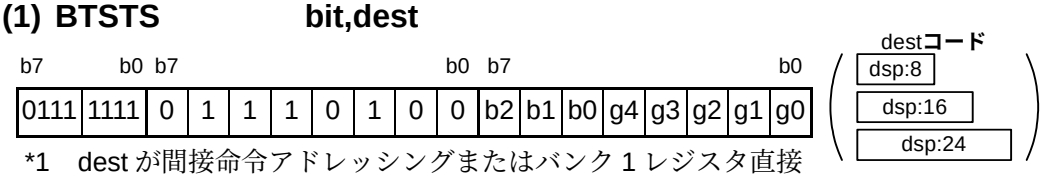
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

BTSTS BTSTS

(1) BTSTS bit,dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 b2~b0の3ビットでビット位置を、g4~g0ビットでdestのオペランドを指定します。

*3 演算長はバイト固定です。

【バイト数/サイクル数】

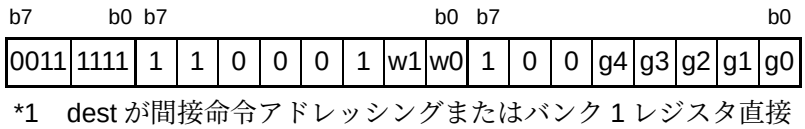
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

CLIP

CLIP

(1) CLIP　　#IMM1,#IMM2,dest



*1　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2　g4~g0 ビットで src のオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5

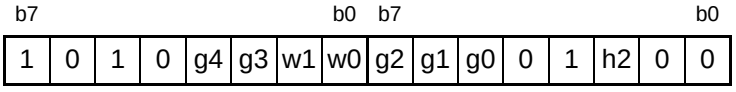
*3　サイズ指定子(.size)が “.W” のとき、表中のバイト数は+2、“.L” のとき+6されます。

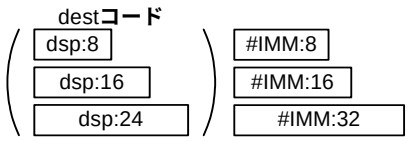
*4　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

CMP

CMP

(1) CMP.size:G　　#IMM(EX),dest





*1　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2　g4~g0 ビットで dest のオペランドを指定します。

*3　h2ビットで src のオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

src　\　dest	**レジスタ**	[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

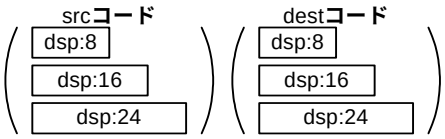
*4　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

CMP CMP

(2) CMP.size:G src,dest

b7 b0 b7 b0

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



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

CMP CMP

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

b7 b0 b7 b0

1	1	1	0	0	q3	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0

*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 q3~q0の4ビットで#IMM:4を、g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

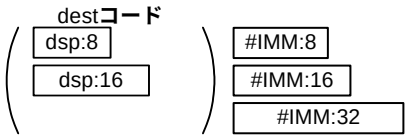
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

CMP　　CMP

(4) CMP.size:S　　#IMM,dest

b7							b0
0	0	1	1	s1	s0	w1	w0



*1　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　s1~s0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

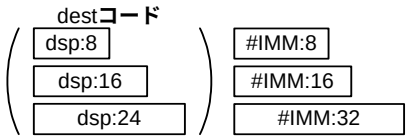
src　\　dest	レジスタ	dsp:8[ ]	dsp:16
#IMM:8	2 / 1	3 / 2	4 / 2
#IMM:16	3 / 1	4 / 2	5 / 2
#IMM:32	5 / 1	6 / 2	7 / 2

*3　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2　g4~g0ビットでdestのオペランドを指定します。
*3　h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

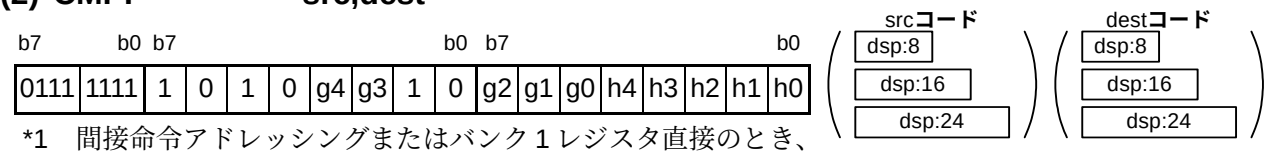
【バイト数/サイクル数】

src　\　dest	レジスタ	[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　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

CMPF CMPF

(2) CMPF src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

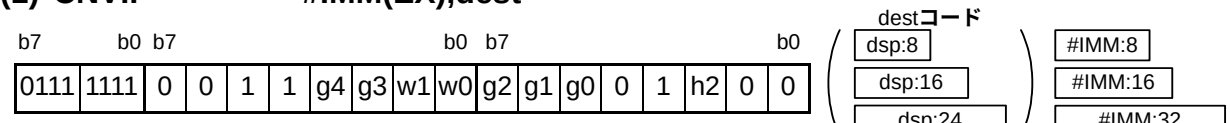
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

CNVIF CNVIF

(1) CNVIF #IMM(EX),dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

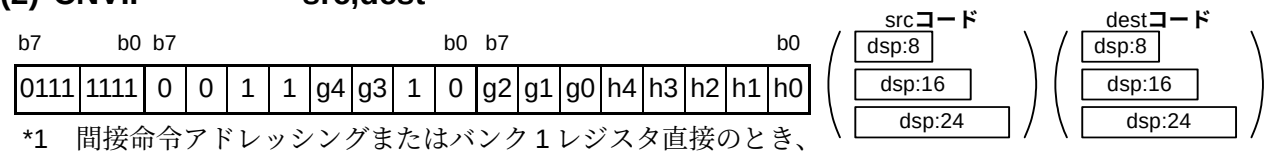
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

CNVIF CNVIF

(2) CNVIF src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

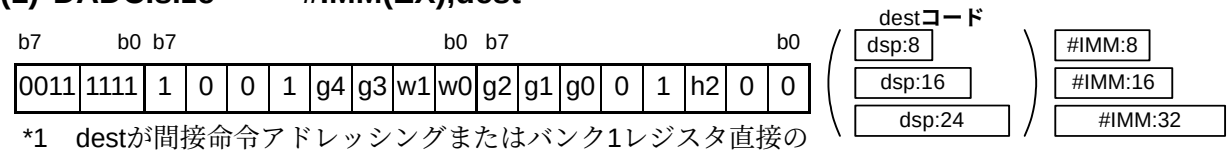
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DADC DADC

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

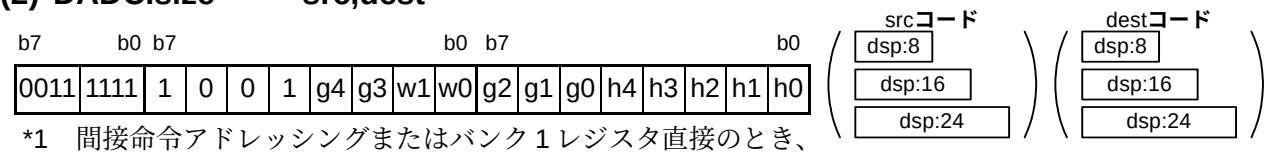
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 6	4 / 7	5 / 7	6 / 7	7 / 7	6 / 7	7 / 7
#IMM(EX):16	5 / 6	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
#IMM:32	7 / 6	7 / 7	8 / 7	9 / 7	10 / 7	9 / 7	10 / 7

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DADC

DADC

(2) DADC.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

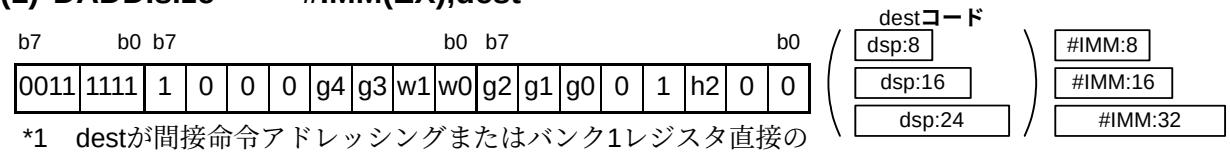
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	3 / 6	3 / 6	4 / 6	5 / 6	6 / 6	5 / 6	6 / 6
[An]	3 / 7	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7
dsp:8[ ]	4 / 7	4 / 7	5 / 7	6 / 7	7 / 7	6 / 7	7 / 7
dsp:16[ ]	5 / 7	5 / 7	6 / 7	7 / 7	8 / 7	7 / 7	8 / 7
dsp:24[ ]	6 / 7	6 / 7	7 / 7	8 / 7	9 / 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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DADD

DADD

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

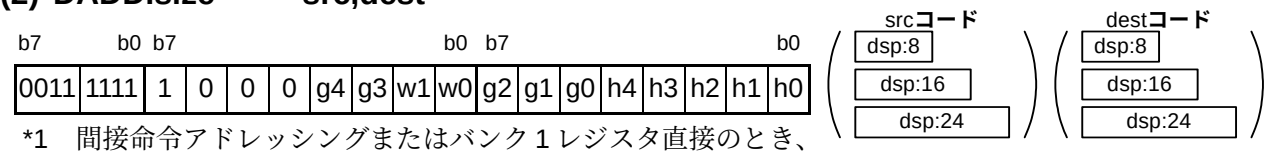
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DADD DADD

(2) DADD.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

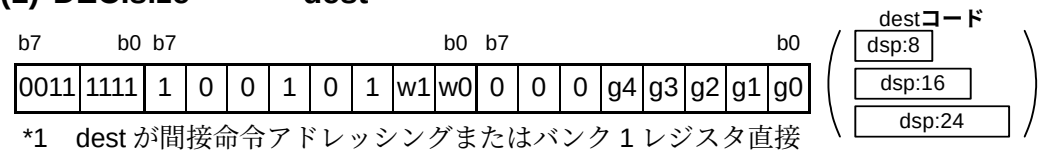
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DEC DEC

(1) DEC.size dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

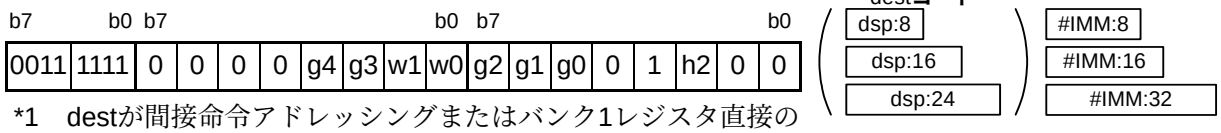
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DIV DIV

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。
*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

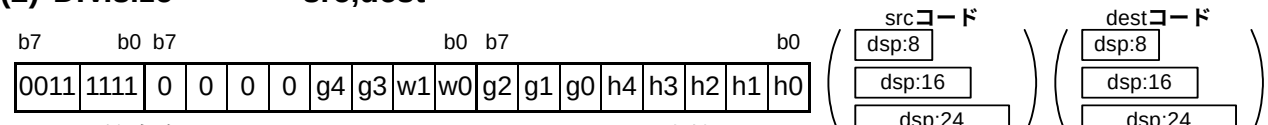
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。
*5 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DIV DIV

(2) DIV.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111
*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

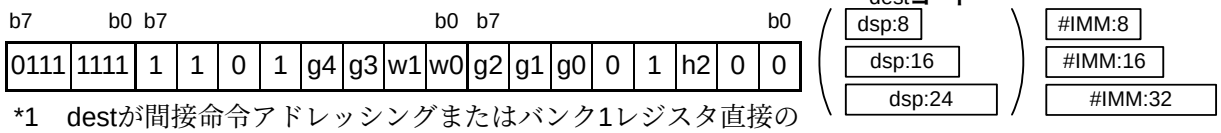
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。
*4 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DIVF DIVF

(1) DIVF #IMM(EX),dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

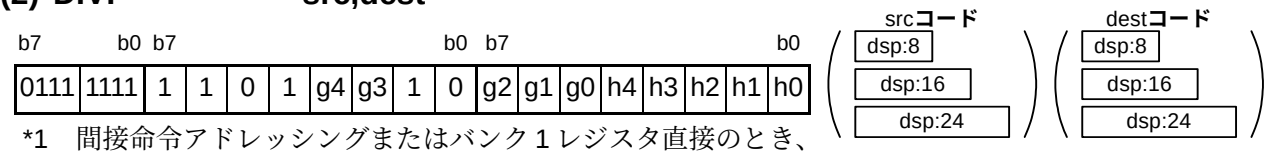
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DIVF DIVF

(2) DIVF src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

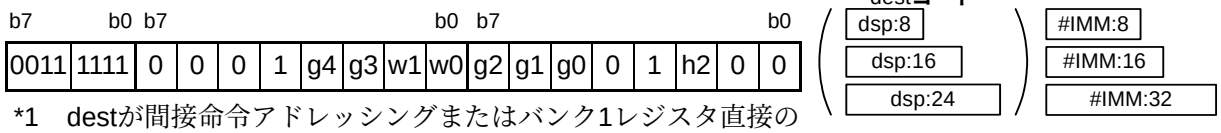
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DIVU DIVU

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。
*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

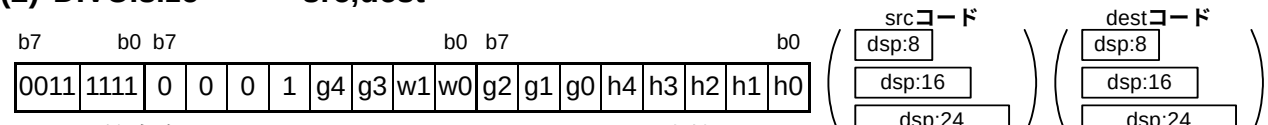
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。
*5 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DIVU DIVU

(2) DIVU.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111
*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

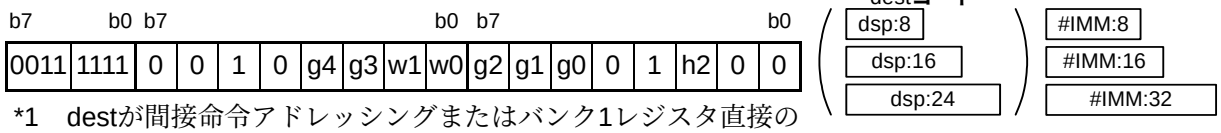
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。
*4 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DIVX DIVX

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。
*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

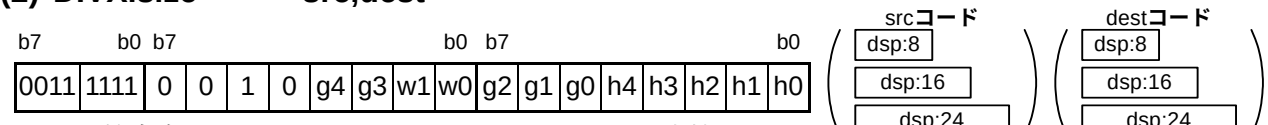
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。
*5 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DIVX DIVX

(2) DIVX.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111
*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

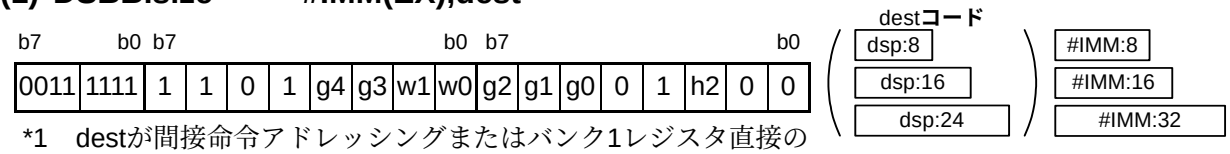
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。
*4 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DSBB DSBB

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

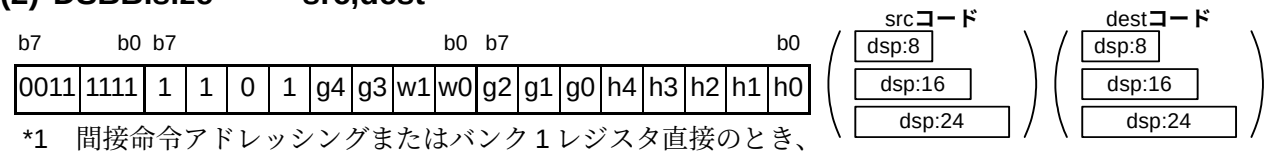
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DSBB DSBB

(2) DSBB.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

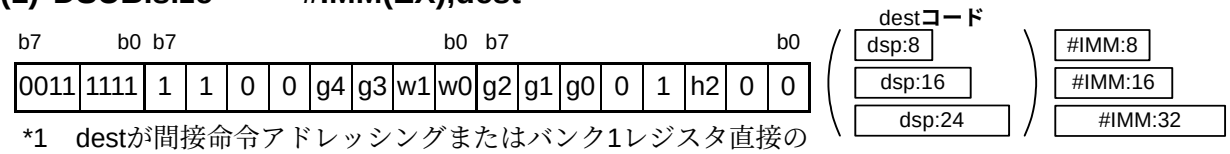
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

DSUB DSUB

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

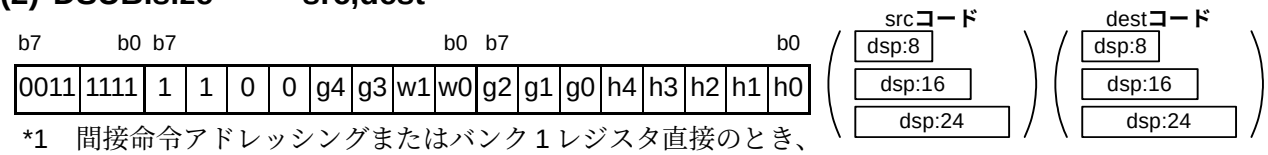
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

DSUB DSUB

(2) DSUB.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

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

srcコード
- #IMM:8
- #IMM:16
- #IMM:32
- dsp:8
- dsp:16
- dsp:24

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destがバンク1レジスタ直接のとき、01011111
srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのレジスタを指定します。

【バイト数/サイクル数】

src	**レジスタ**	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 16	4 / 16	5 / 16	7 / 16	3 / 16	4 / 16	5 / 16	6 / 16	5 / 16	6 / 16

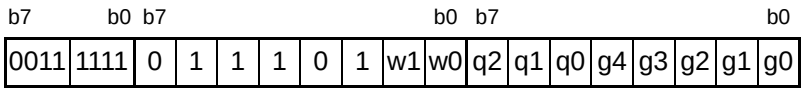
*3 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。

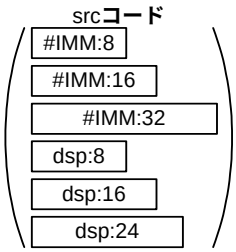
*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

EDIVU

EDIVU

(1) EDIVU.size src,dest





*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destがバンク1レジスタ直接のとき、01011111
srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのレジスタを指定します。

【バイト数/サイクル数】

src	**レジスタ**	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 14	4 / 14	5 / 14	7 / 14	3 / 14	4 / 14	5 / 14	6 / 14	5 / 14	6 / 14

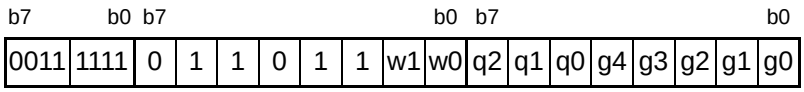
*3 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。

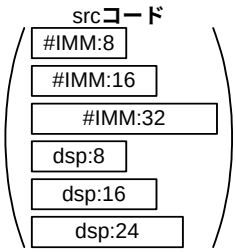
*4 ssrcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

EDIVX

EDIVX

(1) EDIVX.size src,dest





*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destがバンク1レジスタ直接のとき、01011111
srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのレジスタを指定します。

【バイト数/サイクル数】

src	**レジスタ**	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 16	4 / 16	5 / 16	7 / 16	3 / 16	4 / 16	5 / 16	6 / 16	5 / 16	6 / 16

*3 サイズ指定子(.size)が “.W” のとき、表中のサイクル数は+3、“.L” のとき+10されます。

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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コード
- #IMM:8
- #IMM:16
- #IMM:32
- dsp:8
- dsp:16
- dsp:24

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destがバンク1レジスタ直接のとき、01011111
srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのレジスタを指定します。

【バイト数/サイクル数】

src	**レジスタ**	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 2	4 / 2	5 / 2	7 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*3 サイズ指定子(.size)が “.B” または “.W” のときのサイクル数です。“.L” のとき+1されます。

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

EMULU

EMULU

(1) EMUL.size src,dest

b7							b0	b7							b0
0011	1111	0	1	1	1	0	0	w1	w0	q2	q1	q0	g4	g3	g2	g1	g0

srcコード
- #IMM:8
- #IMM:16
- #IMM:32
- dsp:8
- dsp:16
- dsp:24

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destがバンク1レジスタ直接のとき、01011111
srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのレジスタを指定します。

【バイト数/サイクル数】

src	レジスタ	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 2	4 / 2	5 / 2	7 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*3 サイズ指定子(.size)が “.B” または “.W” のときのサイクル数です。“.L” のとき+1されます。

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ENTER

ENTER

(1) ENTER #IMM:8

b7							b0
0	0	0	1	1	0	1	1

#IMM:8

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3

ENTER

ENTER

(2) ENTER #IMM:16

b7							b0
1	1	0	1	1	0	1	1

#IMM:16

【バイト数/サイクル数】

バイト数/サイクル数	3 / 3

EXITD

EXITD

(1) EXITD

b7							b0
1	1	1	0	1	1	1	1

【バイト数/サイクル数】

バイト数/サイクル数	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

【バイト数/サイクル数】

バイト数/サイクル数	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コード: dsp:8 / dsp:16 / dsp:24 → #IMM:8 / #IMM:16

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 w1,w0ビットでsrcとdestのオペランドサイズを指定します。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

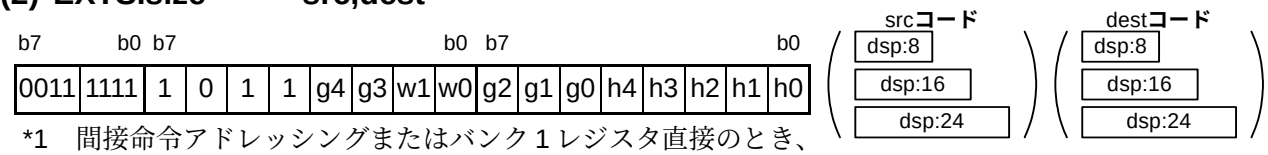
【バイト数/サイクル数】

src \ dest	レジスタ	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

EXTS EXTS

(2) EXTS.size src,dest



*1 間接命令アドレッシングまたはバンク１レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク１レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク１レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク１レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

*3 w1,w0ビットでsrcとdestのオペランドサイズを指定します。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

【バイト数/サイクル数】

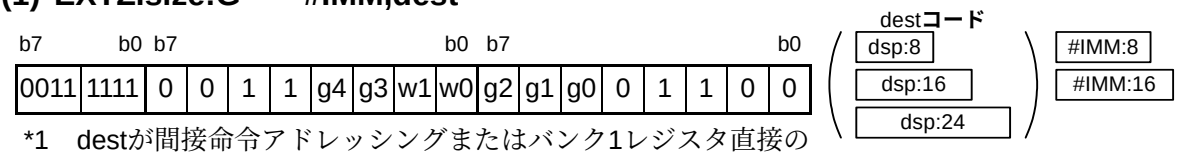
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	3 / 1	3 / 1	4 / 1	5 / 1	6 / 1	5 / 1	6 / 1
[An]	3 / 2	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
dsp:8[ ]	4 / 2	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
dsp:16[ ]	5 / 2	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
dsp:24[ ]	6 / 2	6 / 2	7 / 2	8 / 2	9 / 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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク１レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

EXTZ

EXTZ

(1) EXTZ.size:G #IMM,dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 w1,w0ビットでsrcとdestのオペランドサイズを指定します。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

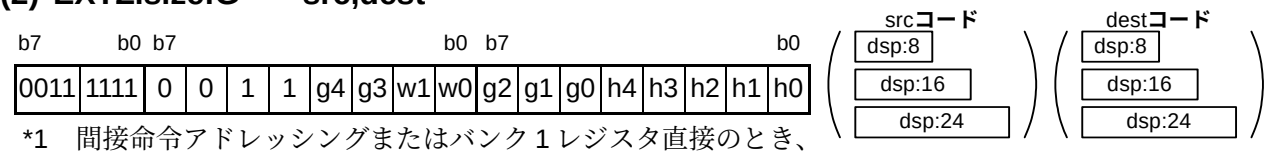
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

EXTZ EXTZ

(2) EXTZ.size:G src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

*3 w1,w0ビットでsrcとdestのオペランドサイズを指定します。

w1	w0	src	dest
0	0	B	W
0	1	B	L
1	0	W	L

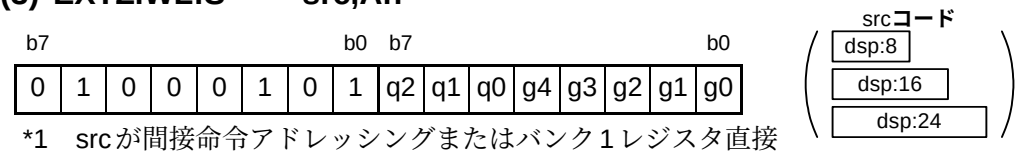
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

EXTZ EXTZ

(3) EXTZ.WL:S src,An



*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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 q2~q0の3ビットでdestのフラグを指定します。詳細は「4.2.2 オペランドの指定」の「(9) FLG直接」を参照してください。

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3

FREIT

FREIT

(1) FREIT

b7							b0
1	0	0	0	1	1	1	1

【バイト数/サイクル数】

バイト数/サイクル数	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 q2~q0の3ビットでdestのフラグを指定します。詳細は「4.2.2 オペランドの指定」の「(9) FLG直接」を参照してください。

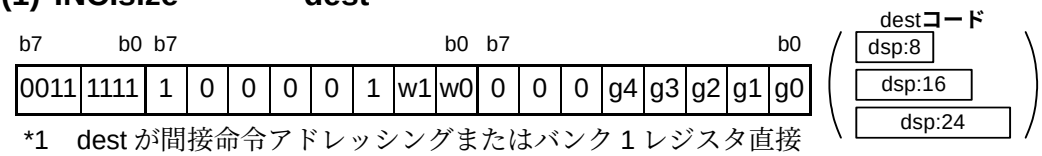
【バイト数/サイクル数】

バイト数/サイクル数	2 / 3

INC

INC

(1) INC.size dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0 ビットで dest のオペランドを指定します。

【バイト数/サイクル数】

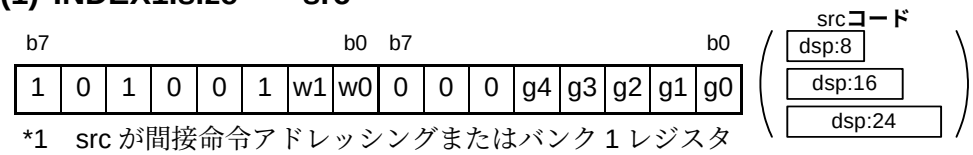
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

INDEX1

INDEX1

(1) INDEX1.size src



*1 src が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0 ビットで src のオペランドを指定します。

【バイト数/サイクル数】

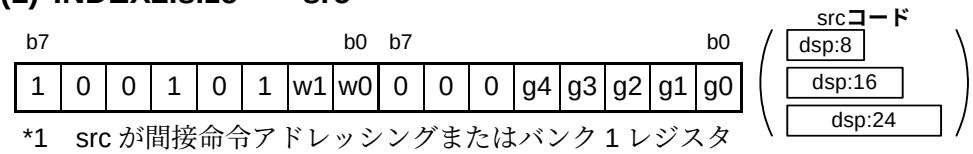
src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 直後の命令のサイクル数は+2されます。

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

INDEX2 INDEX2

(1) INDEX2.size src



*1 src が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

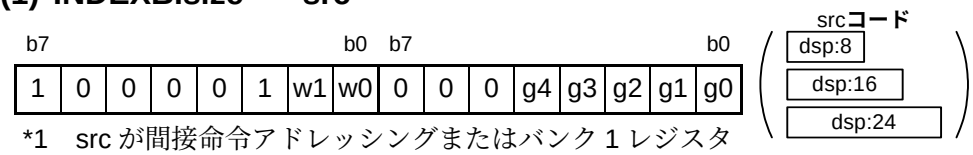
src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 直後の命令のサイクル数は+2されます。

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

INDEXB INDEXB

(1) INDEXB.size src



*1 src が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 直後の命令のサイクル数は+4されます。なお、直後の命令がEXTS、EXTZ命令の場合、サイクル数は+5されます。

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

INT

INT

(1) INT　　#IMM:8

b7							b0
1	1	1	1	0	0	1	1

#IMM:8

【バイト数/サイクル数】

バイト数/サイクル数	2 / 11

INTO

INTO

(1) INTO

b7							b0
1	0	1	0	1	1	1	1

【バイト数/サイクル数】

バイト数/サイクル数	1 / 1(12)*1

*1　Oフラグが “1” のとき12クロックになります。

JCnd

JCnd

(1) J*Cnd*　　label

b7							b0
c3	c2	c1	c0	0	0	1	1

labelコード
dsp:8

*1　c3~c0ビットで*Cnd*を指定します。詳細は「4.2.3 条件の指定 」を参照してください。

【バイト数/サイクル数】

バイト数/サイクル数	2 / 1(3)*2

*2　labelに分岐したとき、サイクル数は3になります。

JMP JMP

(1) JMP.S label

b7 b0

1	q2	q1	q0	0	1	1	1

*1 q2~q0 ビットで dsp3 を指定します。詳細は「4.2.2 オペランドの指定」の「(7) プログラムカウンタ相対・ショート形式」を参照してください。

【バイト数/サイクル数】

バイト数/サイクル数	1 / 1

JMP JMP

(2) JMP.B label

b7 b0

0	1	1	1	0	0	1	1

labelコード

dsp:8

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3

JMP JMP

(3) JMP.W label

b7 b0

1	0	1	0	1	0	1	1

labelコード

dsp:16

【バイト数/サイクル数】

バイト数/サイクル数	3 / 3

JMP JMP

(4) JMP.A label

b7 b0

1	1	1	0	1	0	1	1

labelコード

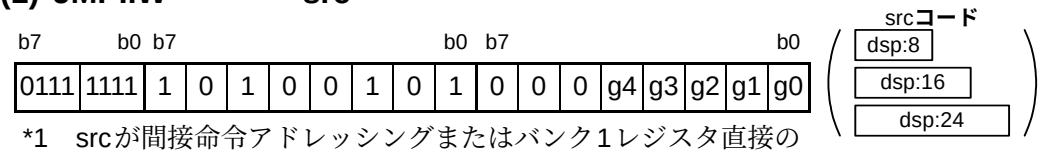
dsp:24

【バイト数/サイクル数】

バイト数/サイクル数	4 / 3

JMPI JMPI

(1) JMPI.W src



*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

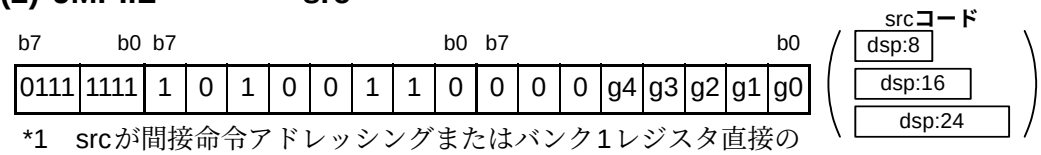
【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

JMPI JMPI

(2) JMPI.L src



*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

JSR

JSR

(1) JSR.W label

b7							b0
1	0	1	1	1	0	1	1

labelコード: dsp:16

【バイト数/サイクル数】

バイト数/サイクル数	3 / 3

JSR

JSR

(2) JSR.A label

b7							b0
1	1	1	1	1	0	1	1

labelコード: dsp:24

【バイト数/サイクル数】

バイト数/サイクル数	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コード: dsp:8 / dsp:16 / dsp:24

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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コード
- dsp:8
- dsp:16
- dsp:24

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 6	3 / 7	4 / 7	5 / 7	6 / 7	5 / 7	6 / 7

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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コード
- #IMM:8
- #IMM:16
- #IMM:32
- dsp:8
- dsp:16
- dsp:24

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

*3 q2~q0 ビットで dest の専用レジスタを指定します。詳細は「4.2.2 オペランドの指定」の「(5) 専用レジスタ直接(CPU)」を参照してください。

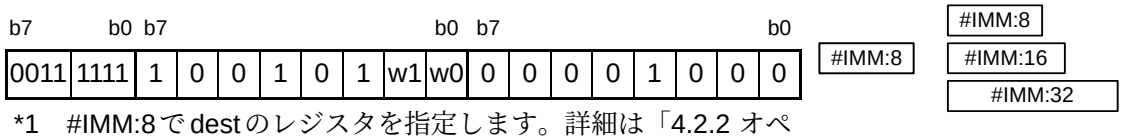
【バイト数/サイクル数】

src	レジスタ	#IMMEX:8	#IMMEX:16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 3	4 / 3	5 / 3	7 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

LDC　　LDC

(2) LDC　　#IMM,dest



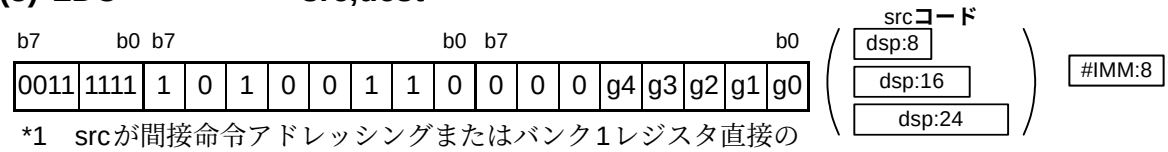
*1　#IMM:8でdestのレジスタを指定します。詳細は「4.2.2 オペランドの指定」の「(6) 専用レジスタ直接(DMAC,VCT)」を参照してください。

【バイト数/サイクル数】

src	#IMM:8	#IMM:16	#IMM:32
バイト数/サイクル数	5 / 3	6 / 3	8 / 3

LDC　　LDC

(3) LDC　　src,dest



*1　srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2　g4~g0ビットでsrcのオペランドを指定します。

*3　#IMM:8でdestのレジスタを指定します。詳細は「4.2.2 オペランドの指定」の「(6) 専用レジスタ直接(DMAC,VCT)」を参照してください。

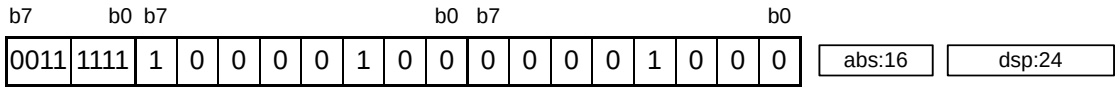
【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 3	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3

*4　srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

LDCTX　　LDCTX

(1) LDCTX　　abs:16,dsp:24



【バイト数/サイクル数】

バイト数/サイクル数	8 / 10+m*1

*1　mは転送するレジスタの数です。

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 q2~q0の3ビットで#IMM:3を指定します。

【バイト数/サイクル数】

バイト数/サイクル数	2 / 2

MAX

MAX

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

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

destコード

dsp:8	#IMM:8
dsp:16	#IMM:16
dsp:24	#IMM:32

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

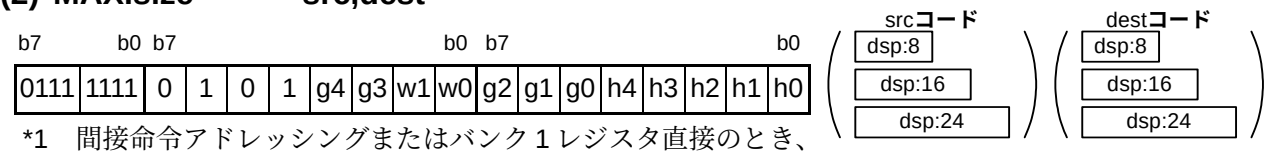
【バイト数/サイクル数】

src \ dest	レジスタ	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MAX MAX

(2) MAX.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

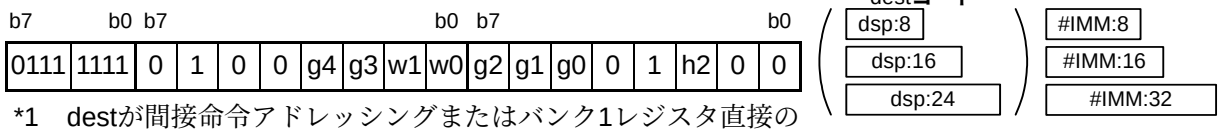
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

MIN　　MIN

(1) MIN.size　　#IMM(EX),dest



*1　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2　g4~g0ビットでdestのオペランドを指定します。

*3　h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

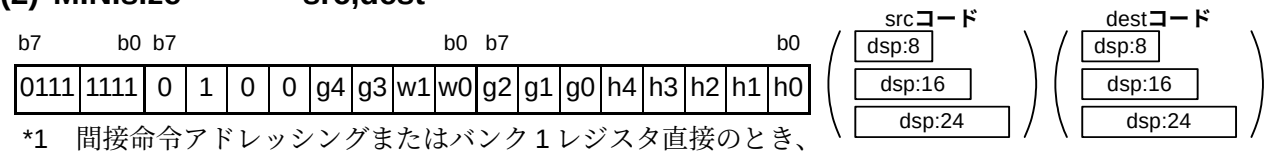
【バイト数/サイクル数】

src　\　dest	**レジスタ**	[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　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MIN　　MIN

(2) MIN.size　　src,dest



*1　間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2　g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

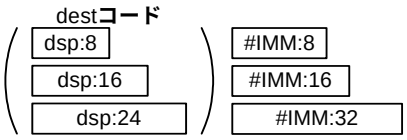
src　\　dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
[An]	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
dsp:8[ ]	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
dsp:16[ ]	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24[ ]	6 / 3	6 / 4	7 / 4	8 / 4	9 / 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　srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

MOV　　MOV

(1) MOV.size:G　　#IMM(EX),dest

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



*1　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2　g4~g0ビットでdestのオペランドを指定します。
*3　h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

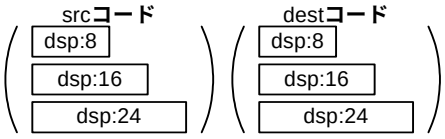
src　\　dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MOV　　MOV

(2) MOV.size:G　　src,dest

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



*1　間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
　srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
　srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111
*2　g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src　\　dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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　srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

MOV MOV

(3) MOV.size:G src,dsp:8[SP]

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでsrcのオペランドを指定します。

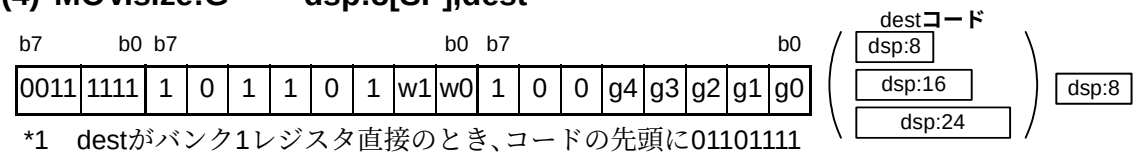
【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MOV MOV

(4) MOV.size:G dsp:8[SP],dest



*1 destがバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

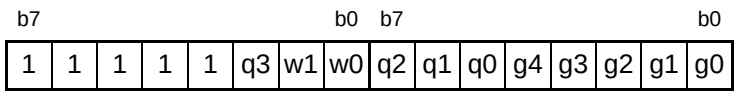
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 4	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4

*3 destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。

MOV MOV

(5) MOV.size:Q #IMM:4,dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 q3~q0の4ビットで#IMM:4を、g4~g0ビットでdestのオペランドを指定します。

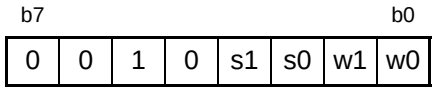
【バイト数/サイクル数】

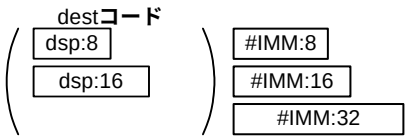
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 1	3 / 1	4 / 1	5 / 1	4 / 1	5 / 1

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MOV MOV

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





*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 s1~s0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

src \ dest	**レジスタ**	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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MOV MOV

(7) MOV.size:S R0L/R0/R2R0,dest

b7 b0

0	0	0	0	s1	s0	w1	w0

*1 destが間接命令アドレッシングのとき、コードの先頭に01101111が付きます。
*2 s1~s0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	dsp:8[ ]	dsp:16
バイト数/サイクル数	2 / 1	3 / 1

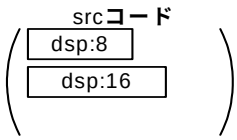
*3 destが間接命令アドレッシングのとき、表中のバイト数は+1、サイクル数は+2されます。

MOV MOV

(8) MOV.size:S src,R0L/R0/R2R0

b7 b0

0	1	0	1	s1	s0	w1	w0



*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。
*2 s1~s0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src	**レジスタ**	dsp:8[ ]	dsp:16
バイト数/サイクル数	1 / 1	2 / 1	3 / 1

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MOV　　MOV

(9) MOV.L:S　　src,A0

b7　　b0

0	1	0	0	s1	s0	1	0

srcコード
dsp:8
dsp:16

*1　src　　が間接命令アドレッシングのとき、コードの先頭に01101111が付きます。
*2　s1~s0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src	dsp:8[ ]	dsp:16
バイト数/サイクル数	2 / 1	3 / 1

*3　srcが間接命令アドレッシングのとき、表中のバイト数は+1、サイクル数は+2されます。

MOV　　MOV

(10)MOV.size:Z　　#0,dest

b7　　b0

0	0	0	1	s1	s0	w1	w0

destコード
dsp:8
dsp:16

*1　dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2　s1~s0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	dsp:8[ ]	dsp:16
バイト数/サイクル数	1 / 1	2 / 1	3 / 1

*3　destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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コード
- dsp:8
- dsp:16
- dsp:24

*1 dest がバンク 1 レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのレジスタを指定します。
*3 演算長はロングワード固定です。

【バイト数/サイクル数】

src	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	3 / 1	4 / 1	5 / 1	4 / 1	5 / 1

*4 destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。

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コード
- dsp:8
- dsp:16
- dsp:24

*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。
*2 g4~g0ビットでsrcのオペランドを指定します。
*3 d1~d0ビットで動作(*Dir*)を指定します。

d1	d0	*Dir*
0	0	LL
0	1	LH
1	0	HL
1	1	HH

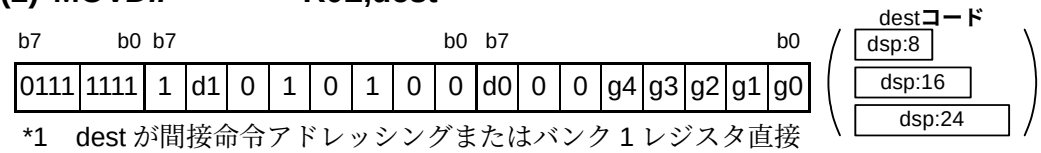
【バイト数/サイクル数】

src	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*4 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MOV*Dir* MOV*Dir*

(2) MOV*Dir* R0L,dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 d1~d0ビットで動作(*Dir*)を指定します。

d1	d0	*Dir*
0	0	LL
0	1	LH
1	0	HL
1	1	HH

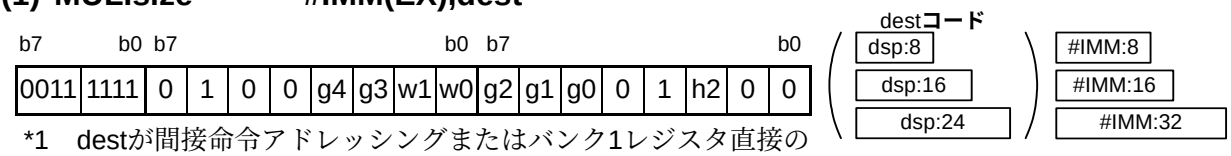
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 3	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MUL MUL

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

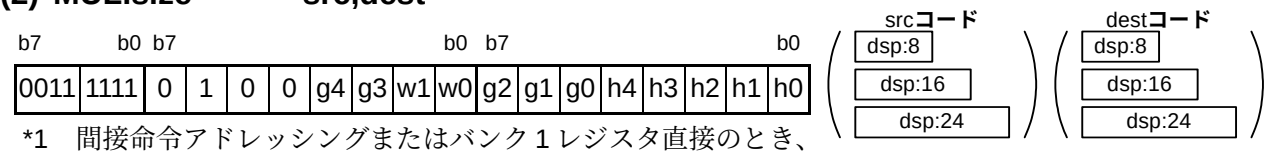
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MUL MUL

(2) MUL.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

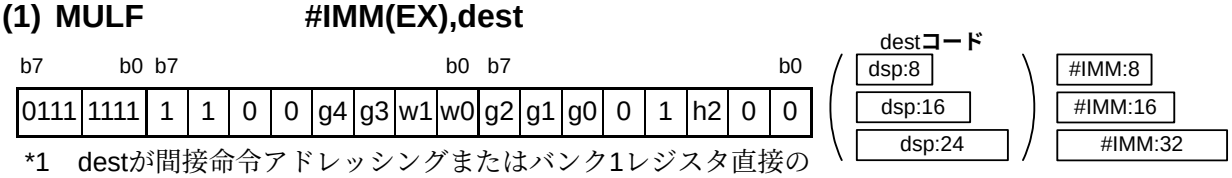
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

MULF MULF

(1) MULF #IMM(EX),dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

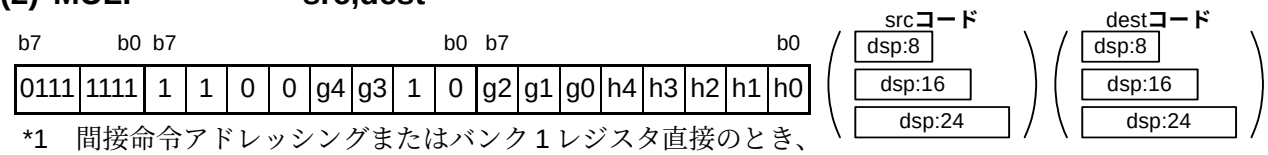
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMMEX:8	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
#IMMEX:16	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
#IMM:32	7 / 3	7 / 4	8 / 4	9 / 4	10 / 4	9 / 4	10 / 4

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MULF

MULF

(2) MULF src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

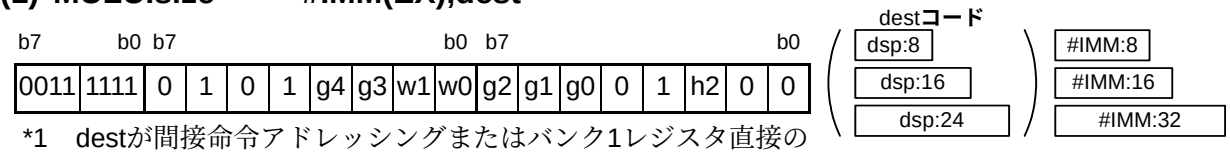
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
[An]	3 / 4	3 / 5	4 / 5	5 / 5	6 / 5	5 / 5	6 / 5
dsp:8[ ]	4 / 4	4 / 5	5 / 5	6 / 5	7 / 5	6 / 5	7 / 5
dsp:16[ ]	5 / 4	5 / 5	6 / 5	7 / 5	8 / 5	7 / 5	8 / 5
dsp:24[ ]	6 / 4	6 / 5	7 / 5	8 / 5	9 / 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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

MULU

MULU

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

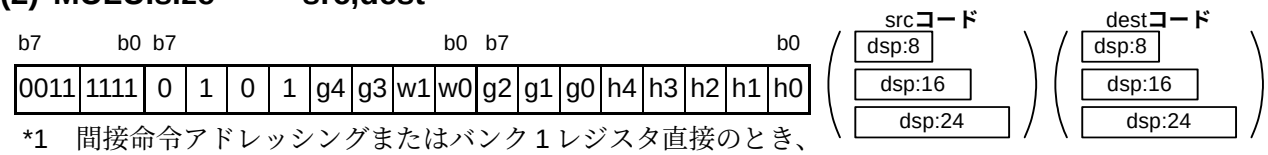
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
#IMM(EX):16	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
#IMM:32	7 / 2	7 / 3	8 / 3	9 / 3	10 / 3	9 / 3	10 / 3

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

MULU

MULU

(2) MULU.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

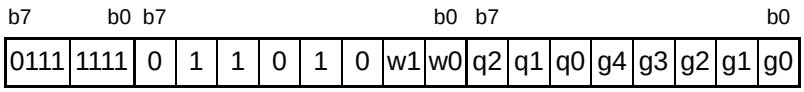
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
[An]	3 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4
dsp:8[ ]	4 / 3	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4
dsp:16[ ]	5 / 3	5 / 4	6 / 4	7 / 4	8 / 4	7 / 4	8 / 4
dsp:24[ ]	6 / 3	6 / 4	7 / 4	8 / 4	9 / 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

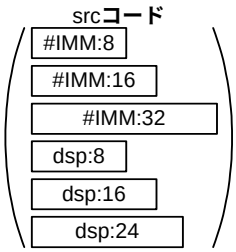
*3 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

MULX

MULX

(1) MULX.size src,dest





*1 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destがバンク1レジスタ直接のとき、01011111
srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでsrcのオペランドを、q2~q0ビットでdestのレジスタを指定します。

【バイト数/サイクル数】

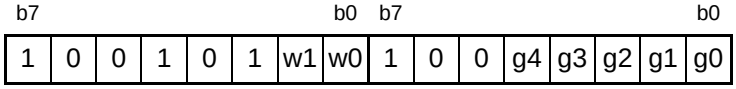
src	**レジスタ**	#IMM(EX):8	#IMM(EX):16	#IMM:32	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 3	4 / 3	5 / 3	7 / 3	3 / 4	4 / 4	5 / 4	6 / 4	5 / 4	6 / 4

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、またはdestがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングまたはバンク1レジスタ直接で、destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

NEG

NEG

(1) NEG.size dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

NOP NOP

(1) NOP

b7							b0
1	0	0	1	1	1	1	1

【バイト数/サイクル数】

バイト数/サイクル数	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コード
- dsp:8
- dsp:16
- dsp:24

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

OR OR

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

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

destコード
- dsp:8
- dsp:16
- dsp:24

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

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。
*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

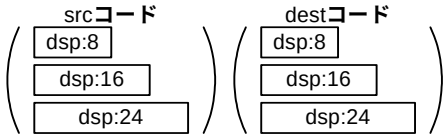
src \ dest	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

OR　　OR

(2) OR.size　　src,dest

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



*1　間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2　g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src　\　dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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　srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

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

*1　dest がバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2　g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 3	2 / 3	3 / 3	4 / 3	5 / 3	4 / 3	5 / 3

*3　destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。

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 q2~q0ビットでdestの専用レジスタを指定します。
*2 演算長はロングワード固定です。

【バイト数/サイクル数】

バイト数/サイクル数	2 / 4

POPM

POPM

(1) POPM dest

b7							b0
0	1	1	SB*1	1	0	1	1

#IMM:8

*1 SBレジスタを復帰するときは “1”、しないときは “0” になります。
*2 その他のレジスタは#IMM:8で指定します。

bit	7	6	5	4	3	2	1	0
レジスタ	A3	A2	A1	A0	R7R5	R6R4	R3R1	R2R0

*3 destがバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

【バイト数/サイクル数】

バイト数/サイクル数	2 / n*4

*4 nは復帰するレジスタの数です。
*5 destがバンク1レジスタ直接のとき、表中のバイト数は+1されます。

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コード
dsp:8
dsp:16
dsp:24

*1 src が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。
*2 g4~g0ビットでsrcのオペランドを指定します。

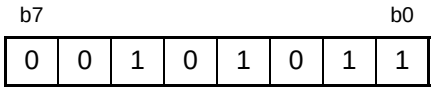
【バイト数/サイクル数】

src	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

PUSH PUSH

(2) PUSH.B:S #IMM:8

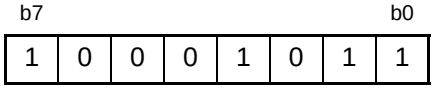


【バイト数/サイクル数】

バイト数/サイクル数	2 / 1

PUSH PUSH

(3) PUSH.W:S #IMM:16

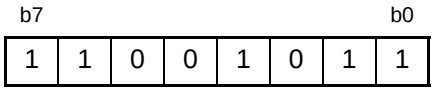


【バイト数/サイクル数】

バイト数/サイクル数	3 / 1

PUSH PUSH

(4) PUSH.L:S #IMM:32

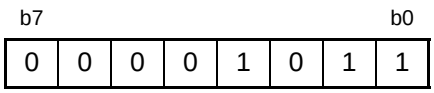


【バイト数/サイクル数】

バイト数/サイクル数	5 / 1

PUSH PUSH

(5) PUSH.W:S #IMMEX:8

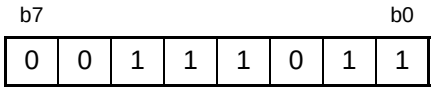


【バイト数/サイクル数】

バイト数/サイクル数	2 / 1

PUSH PUSH

(6) PUSH.L:S #IMMEX:8

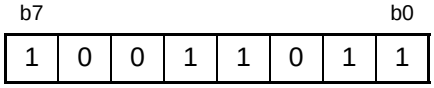


【バイト数/サイクル数】

バイト数/サイクル数	2 / 1

PUSH PUSH

(7) PUSH.L:S #IMMEX:16

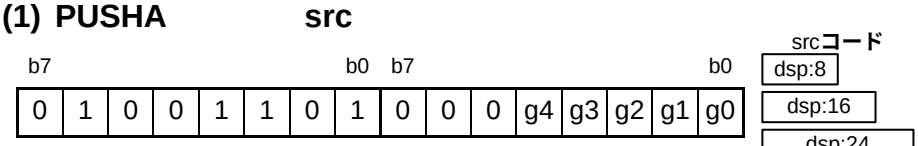


【バイト数/サイクル数】

バイト数/サイクル数	3 / 1

PUSHA PUSHA

(1) PUSHA src



*1 g4~g0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	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 q2~q0ビットでsrcの専用レジスタを指定します。
*2 演算長はロングワード固定です。

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3

PUSHM

PUSHM

(1) PUSHM src

b7							b0
0	1	0	SB*1	1	0	1	1

#IMM:8

*1 SBレジスタを退避するときは “1”、しないときは “0” になります。
*2 その他のレジスタは#IMM:8で指定します。

bit	7	6	5	4	3	2	1	0
レジスタ	R2R0	R3R1	R6R4	R7R5	A0	A1	A2	A3

*3 srcがバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

【バイト数/サイクル数】

バイト数/サイクル数	2 / n*4

*4 nは退避するレジスタの数です。
*5 srcがバンク1レジスタ直接のとき、表中のバイト数は+1されます。

REIT

REIT

(1) REIT

b7							b0
1	1	0	0	1	1	1	1

【バイト数/サイクル数】

バイト数/サイクル数	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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 11+1.5m*1

*1 mは演算回数です。

ROLC

ROLC

(1) ROLC.size dest

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

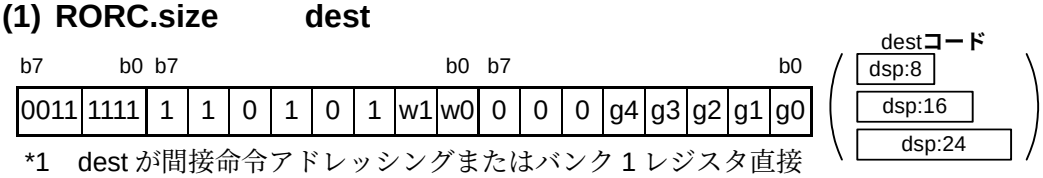
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

RORC

RORC

(1) RORC.size dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

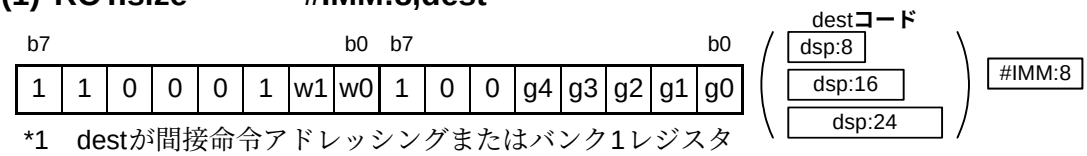
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ROT ROT

(1) ROT.size #IMM:8,dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

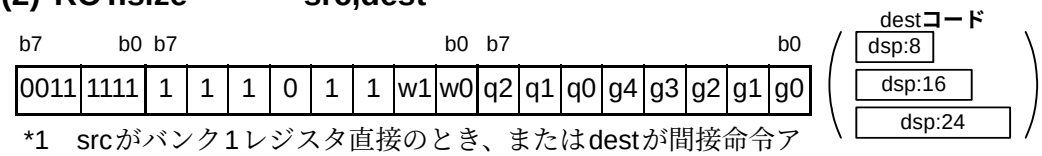
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ROT ROT

(2) ROT.size src,dest



*1 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。

srcがバンク1レジスタ直接のとき、01011111

destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111

srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 q2~q0ビットでsrcのレジスタを、g4~g0ビットでdestのオペランドを指定します。

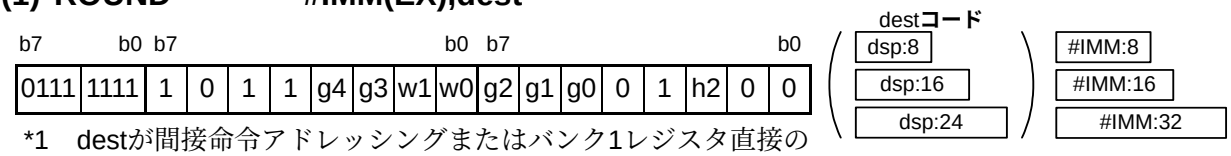
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ROUND ROUND

(1) ROUND #IMM(EX),dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

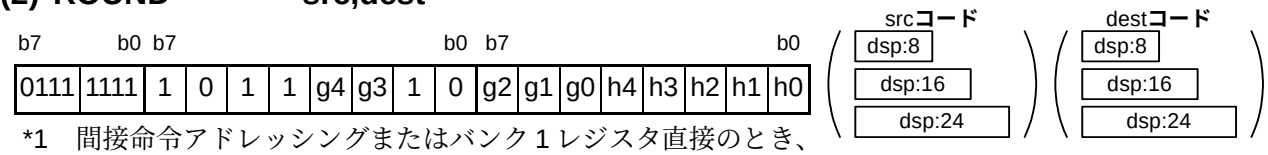
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

ROUND ROUND

(2) ROUND src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

RTS

RTS

(1) RTS

b7							b0
1	1	0	1	1	1	1	1

【バイト数/サイクル数】

バイト数/サイクル数	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コード	
dsp:8	#IMM:8
dsp:16	#IMM:16
dsp:24	#IMM:32

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

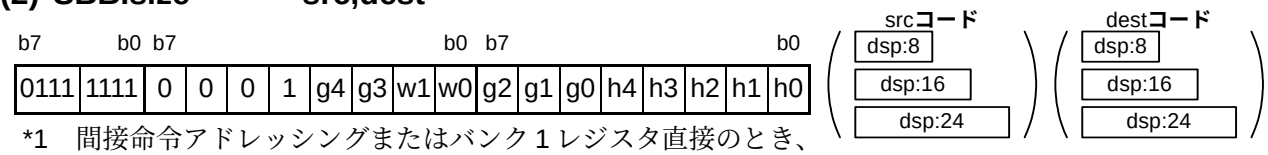
【バイト数/サイクル数】

src \ dest	レジスタ	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SBB SBB

(2) SBB.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

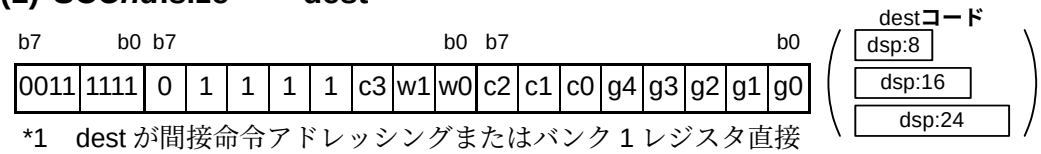
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

SCCnd *SCCnd*

(1) SC*Cnd*.size dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 c3~c0ビットで*Cnd*を指定します。詳細は「4.2.3 条件の指定」を参照してください。

*3 g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 2	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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 “.B”、“.W”をw0ビットで指定します。“0”のとき“.B”、“1”のとき“.W”です。

【バイト数/サイクル数】

バイト数/サイクル数	2 / 8+3m*2

*2 mは比較される8バイトワードの数です。

*3 比較元の開始番地が8バイト境界にない場合、サイクル数は+1されます。同様に比較先の開始番地が8バイト境界にない場合、サイクル数は+1されます。

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コード: dsp:8 / dsp:16 / dsp:24 → #IMM:8

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

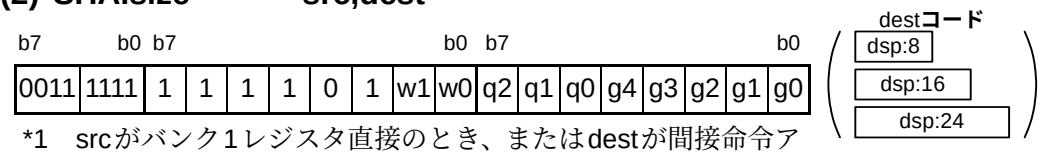
【バイト数/サイクル数】

dest	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SHA SHA

(2) SHA.size src,dest



*1 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcがバンク1レジスタ直接のとき、01011111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 q2~q0ビットでsrcのレジスタを、g4~g0ビットでdestのオペランドを指定します。

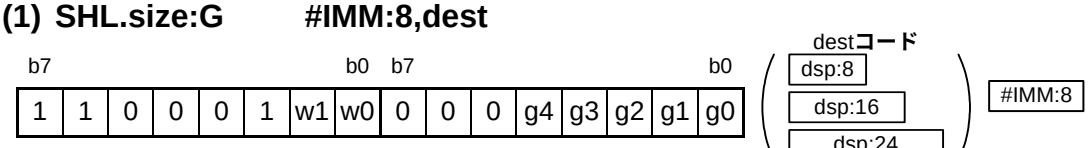
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SHL SHL

(1) SHL.size:G #IMM:8,dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

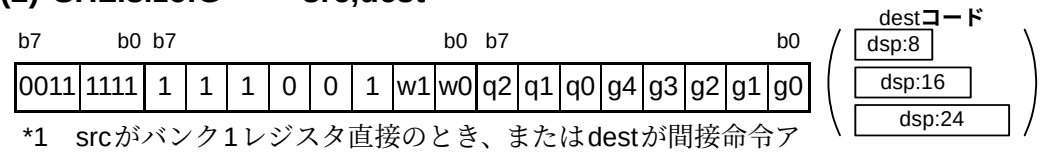
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SHL SHL

(2) SHL.size:G src,dest



*1 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcがバンク1レジスタ直接のとき、01011111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 q2~q0ビットでsrcのレジスタを、g4~g0ビットでdestのオペランドを指定します。

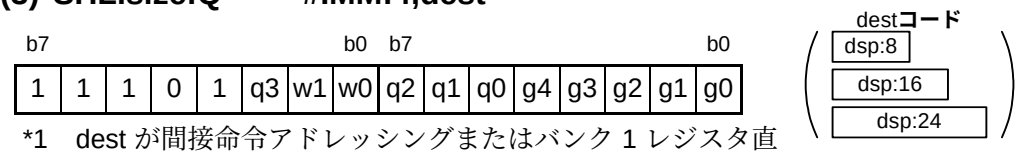
【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*3 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SHL SHL

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



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 q3~q0の4ビットで#IMM:4を、g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	2 / 1	2 / 2	3 / 2	4 / 2	5 / 2	4 / 2	5 / 2

*3 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3+2m*1

*1　mは演算回数です。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3+2m*1

*1　mは演算回数です。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 2+2m*1

*1　mは演算回数です。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 4+2m*1

*1　mは演算回数です。

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 “.B”、“.W”をw0ビットで指定します。“0”のとき“.B”、“1”のとき“.W”です。

【バイト数/サイクル数】

バイト数/サイクル数	2 / 5+3m*2

*2 mは比較される8バイトワードの数です。

*3 比較元の開始番地が8バイト境界にない場合、サイクル数は+1されます。同様に比較先の開始番地が8バイト境界にない場合、サイクル数は+1されます。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3+2m*1

*1 mは演算回数です。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 5+m*1

*1 mは演算回数です。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 8+m*1

*1 mは演算回数です。

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コード: dsp:8 / dsp:16 / dsp:24

*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 q2~q0ビットでsrcの専用レジスタを指定します。詳細は「4.2.2 オペランドの指定」の「(5) 専用レジスタ直接(CPU)」を参照してください。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 2	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

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コード: dsp:8 / dsp:16 / dsp:24 — #IMM:8

*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 #IMM:8 で src のレジスタを指定します。詳細は「4.2.2 オペランドの指定」の「(6) 専用レジスタ直接(DMAC,VCT)」を参照してください。

【バイト数/サイクル数】

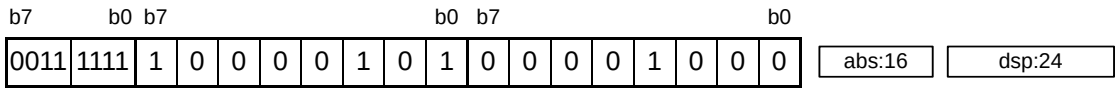
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 3	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

STCTX

STCTX

(1) STCTX abs:16,dsp:24



【バイト数/サイクル数】

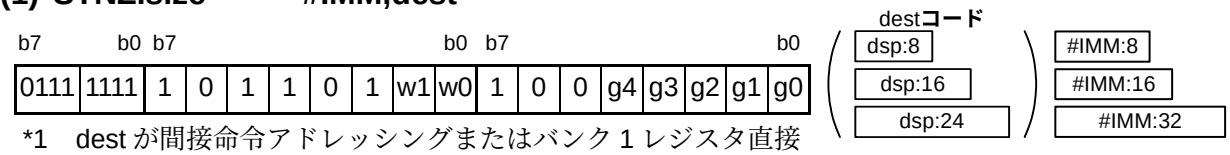
バイト数/サイクル数	8 / 10+m*1

*1 mは転送するレジスタの数です。

STNZ

STNZ

(1) STNZ.size #IMM,dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 2	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2

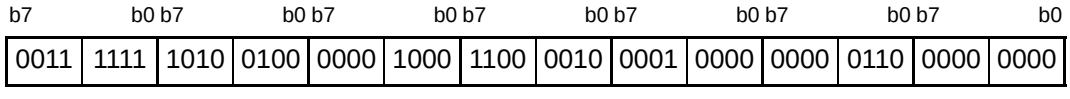
*3 サイズ指定子(.size)が ".W" のとき、表中のバイト数は+1、".L" のとき+3されます。

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

STOP

STOP

(1) STOP



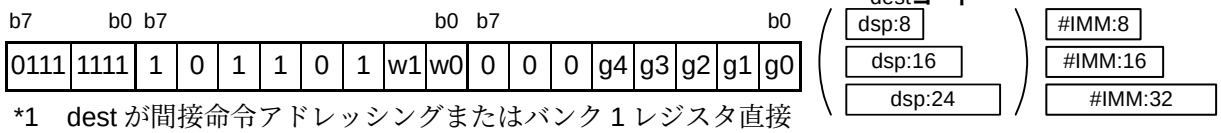
【バイト数/サイクル数】

バイト数/サイクル数	7 / 8

STZ

STZ

(1) STZ.size #IMM,dest



*1 dest が間接命令アドレッシングまたはバンク 1 レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0 ビットで dest のオペランドを指定します。

【バイト数/サイクル数】

dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 2	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2

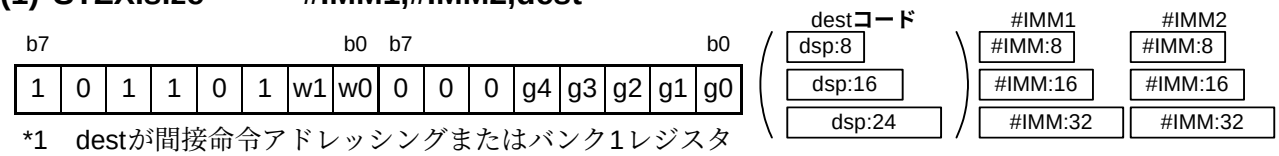
*3 サイズ指定子(.size)が “.W” のとき、表中のバイト数は+1、“.L” のとき+3されます。

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

STZX

STZX

(1) STZX.size #IMM1,#IMM2,dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01101111が付きます。

*2 g4~g0 ビットで dest のオペランドを指定します。

【バイト数/サイクル数】

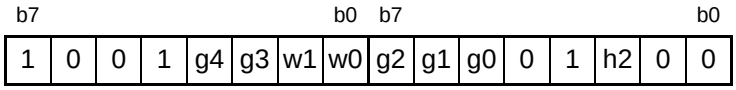
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	4 / 4	4 / 4	5 / 4	6 / 4	7 / 4	6 / 4	7 / 4

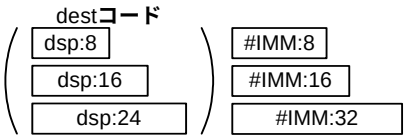
*3 サイズ指定子(.size)が “.W” のとき、表中のバイト数は+2、“.L” のとき+6されます。

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SUB SUB

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





*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。
*2 g4~g0ビットでdestのオペランドを指定します。
*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

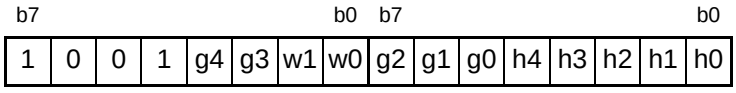
【バイト数/サイクル数】

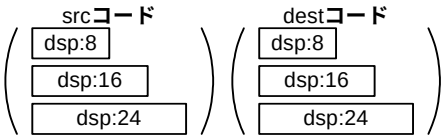
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
#IMM(EX):16	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM:32	6 / 1	6 / 2	7 / 2	8 / 2	9 / 2	8 / 2	9 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SUB SUB

(2) SUB.size src,dest





*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111
*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

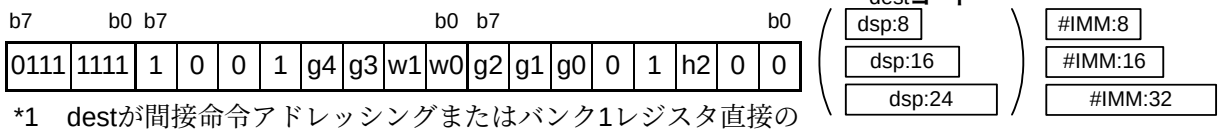
【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

SUBF SUBF

(1) SUBF #IMM(EX),dest



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

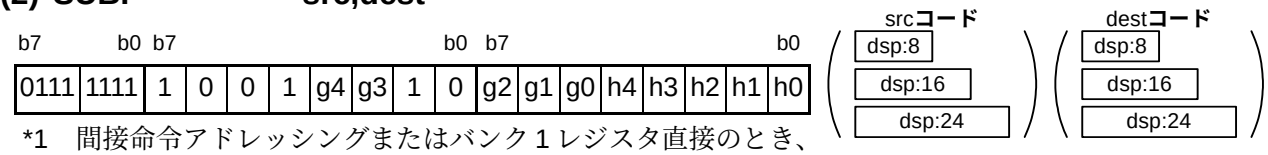
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

SUBF SUBF

(2) SUBF src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3+3m*1

*1 mは演算回数です。

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

【バイト数/サイクル数】

バイト数/サイクル数	2 / 3+3m*1

*1 mは演算回数です。

TST

TST

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

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

destコード

destコード	
dsp:8	#IMM:8
dsp:16	#IMM:16
dsp:24	#IMM:32

*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

【バイト数/サイクル数】

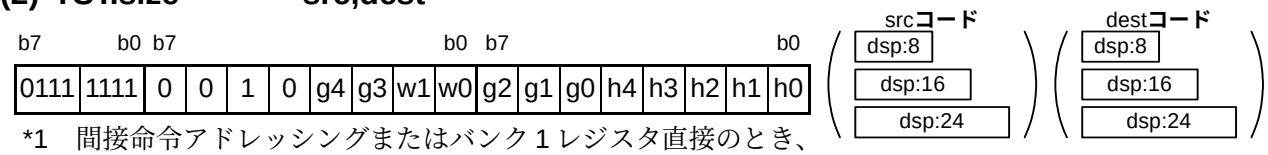
src \ dest	レジスタ	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
#IMM(EX):8	4 / 1	4 / 2	5 / 2	6 / 2	7 / 2	6 / 2	7 / 2
#IMM(EX):16	5 / 1	5 / 2	6 / 2	7 / 2	8 / 2	7 / 2	8 / 2
#IMM:32	7 / 1	7 / 2	8 / 2	9 / 2	10 / 2	9 / 2	10 / 2

*4 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

TST

TST

(2) TST.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

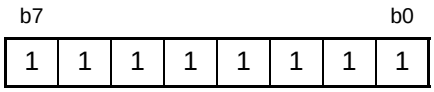
src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	3 / 1	3 / 2	4 / 2	5 / 2	6 / 2	5 / 2	6 / 2
[An]	3 / 2	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3
dsp:8[ ]	4 / 2	4 / 3	5 / 3	6 / 3	7 / 3	6 / 3	7 / 3
dsp:16[ ]	5 / 2	5 / 3	6 / 3	7 / 3	8 / 3	7 / 3	8 / 3
dsp:24[ ]	6 / 2	6 / 3	7 / 3	8 / 3	9 / 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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

UND

UND

(1) UND



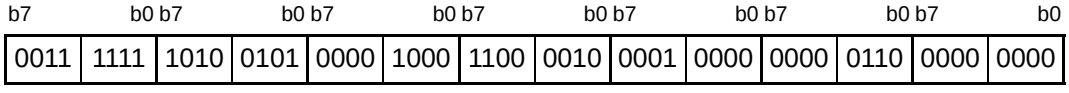
【バイト数/サイクル数】

バイト数/サイクル数	1 / 12

WAIT

WAIT

(1) WAIT



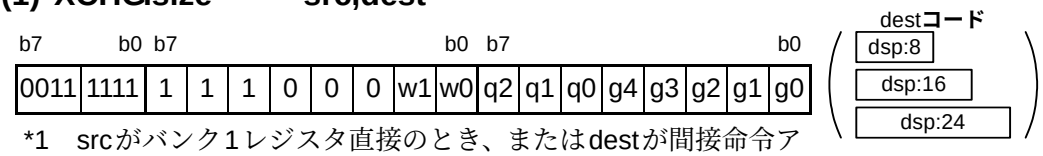
【バイト数/サイクル数】

バイト数/サイクル数	7 / 8

XCHG

XCHG

(1) XCHG.size src,dest



*1 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcがバンク1レジスタ直接のとき、01011111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 q2~q0ビットでsrcのレジスタを、g4~g0ビットでdestのオペランドを指定します。

【バイト数/サイクル数】

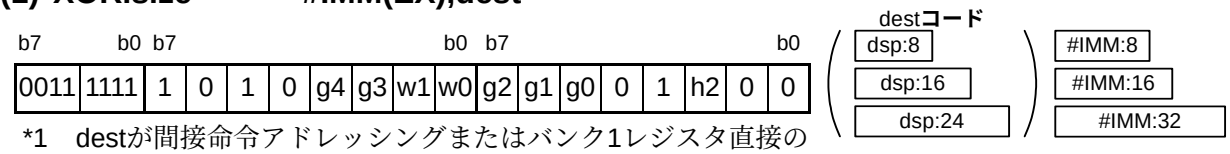
dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
バイト数/サイクル数	3 / 3	3 / 3	4 / 3	5 / 3	6 / 3	5 / 3	6 / 3

*3 srcがバンク1レジスタ直接のとき、またはdestが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcがバンク1レジスタ直接で、destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

XOR

XOR

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



*1 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、コードの先頭に01011111が付きます。

*2 g4~g0ビットでdestのオペランドを指定します。

*3 h2ビットでsrcのオペランドを指定します。“0” のとき#IMMEX、“1” のとき#IMMです。

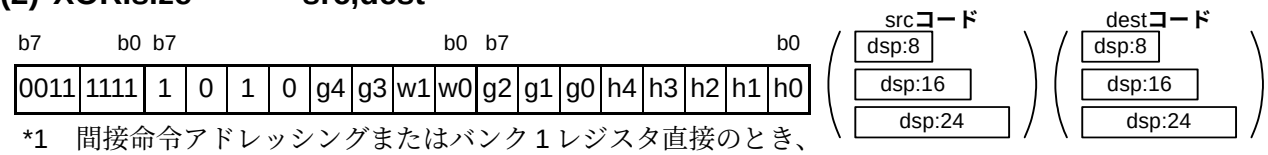
【バイト数/サイクル数】

src \ dest	**レジスタ**	[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 destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、destが間接命令アドレッシングのとき、表中のサイクル数は+2されます。

XOR XOR

(2) XOR.size src,dest



*1 間接命令アドレッシングまたはバンク1レジスタ直接のとき、次の数字がコードの先頭に付きます。
srcが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01101111
destが間接命令アドレッシングまたはバンク1レジスタ直接のとき、01011111
srcとdestの両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、01001111

*2 g4~g0ビットでdestのオペランドを、h4~h0ビットでsrcのオペランドを指定します。

【バイト数/サイクル数】

src \ dest	**レジスタ**	[An]	dsp:8[ ]	dsp:16[ ]	dsp:24[ ]	dsp:16	dsp:24
レジスタ	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 srcまたはdestの一方または両方が間接命令アドレッシングまたはバンク1レジスタ直接のとき、表中のバイト数は+1されます。また、srcまたはdestが間接命令アドレッシングのとき表中のサイクル数は+2され、両方が間接命令アドレッシングのとき表中のサイクル数は+4されます。

5. 割り込み

5.1 割り込みの概要

割り込み要求が受け付けられると、割り込みベクタテーブルに設定した割り込みルーチンへ分岐します。各割り込みベクタテーブルには、割り込みルーチンの先頭番地を設定してください。割り込みベクタテーブルの詳細は「1.7 ベクタテーブル」を参照してください。

5.1.1 割り込みの分類

図 5.1に割り込みの分類を示します。表 5.1に割り込み要因(ノンマスカブル)と固定ベクタテーブルを示します。

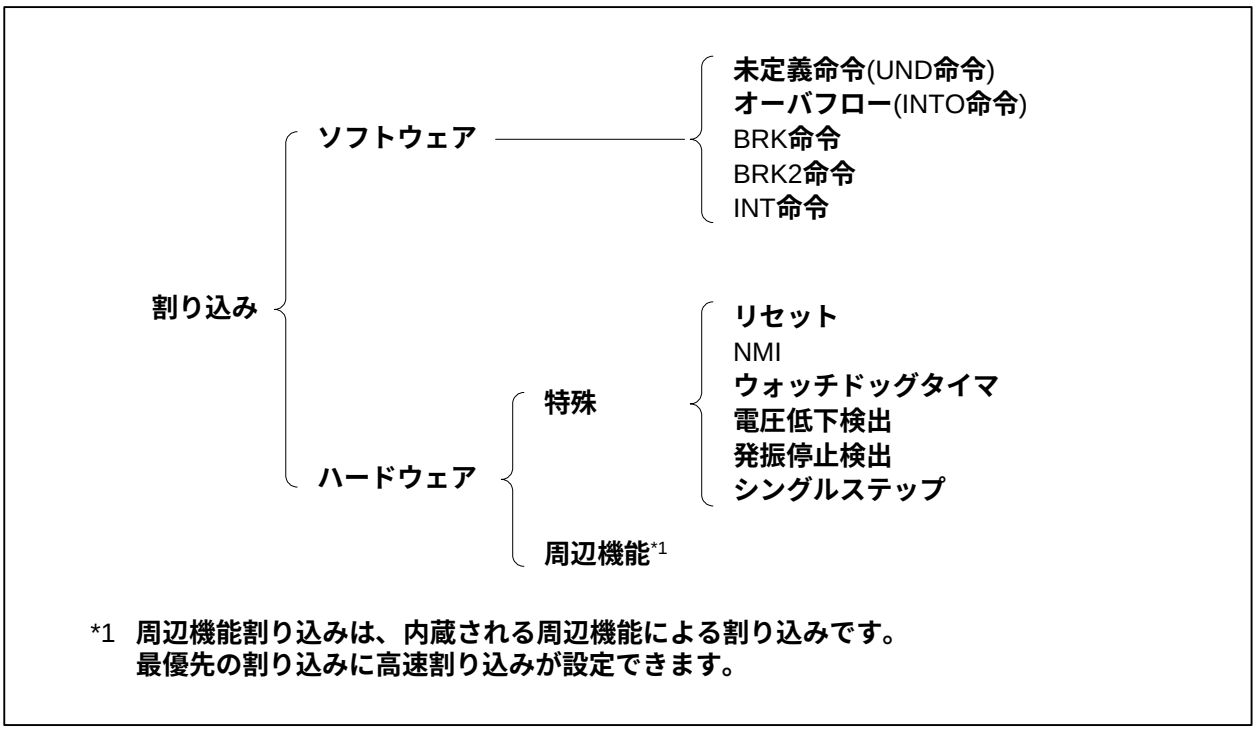


図 5.1 割り込みの分類

表 5.1 割り込み要因(ノンマスカブル)と固定ベクタテーブル

割り込み要因	ベクタテーブル番地 アドレス(L) ~ アドレス(H)	備 考
未定義命令	FFFFFFDCh ~ FFFFFFDFh	UND命令で割り込み
オーバフロー	FFFFFFE0h ~ FFFFFFE3h	INTO命令で割り込み
BRK命令	FFFFFFE4h ~ FFFFFFE7h	FFFFFFE7h番地が“FFh”の場合は可変ベクタテーブル内のベクタが示す番地から実行
ウォッチドッグタイマ 電圧低下検出 発振停止検出	FFFFFFF0h ~ FFFFFFF3h	
NMI	FFFFFFF8h ~ FFFFFFFBh	$\overline{\text{NMI}}$端子による外部割込み
リセット	FFFFFFFCh ~ FFFFFFFFh	

また、マスクの可、不可によりマスカブル割り込みとノンマスカブル割り込みに分類できます。

(1) **マスカブル割り込み**

割り込み許可フラグ(Iフラグ)による割り込み禁止(許可)や割り込み要求レベルによる割り込み優先順位の変更が可能な割り込みです。

(2) **ノンマスカブル割り込み**

割り込み許可フラグ(Iフラグ)による割り込み禁止(許可)や割り込み要求レベルによる割り込み優先順位の変更が不可能な割り込みです。

5.1.2 ソフトウェア割り込み

ソフトウェア割り込みは、命令の実行によって発生します。ソフトウェア割り込みはノンマスカブル割り込みです。

ソフトウェア割り込みには以下の4つがあります。

(1) **未定義命令割り込み**

未定義命令割り込みは、UND命令を実行すると発生します。

(2) **オーバフロー割り込み**

オーバフロー割り込みは、オーバフローフラグ(Oフラグ)が “1” のときINTO命令を実行すると発生します。演算によってOフラグが変化する命令を以下に示します。

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割り込み**

BRK割り込みはBRK命令を実行すると発生します。

(4) **BRK2割り込み**

BRK2割り込みはBRK2命令を実行すると発生します。

この割り込みはエミュレータ専用割り込みです。ユーザプログラムでは使用しないでください。

(5) **INT命令割り込み**

INT命令割り込みは、ソフトウェア割り込み番号0~255を指定し、INT命令を実行すると発生します。なお、ソフトウェア割り込み番号0~127は周辺機能割り込みに割り当てられますので、INT命令を実行することで周辺機能割り込みと同じ割り込みルーチンを実行できます。

INT命令割り込みに使用するスタックポインタ(SP)は、ソフトウェア割り込み番号によって異なります。ソフトウェア割り込み番号0~127では、割り込み要求受け付け時にスタックポインタ指定フラグ(Uフラグ)を退避し、Uフラグを “0” にして割り込みスタックポインタ(ISP)を選択した後、割り込みシーケンスを実行します。割り込みルーチンから復帰するときに割り込み要求受け付け前のUフラグが復帰されます。ソフトウェア割り込み番号128~255では、スタックポインタは切り替わりません。

5.1.3 ハードウェア割り込み

ハードウェア割り込みには、特殊割り込みと周辺機能割り込みがあります。
周辺機能割り込みでは最優先の割り込み1つだけに高速割り込みを設定することができます。

(1) 特殊割り込み

特殊割り込みはノンマスカブル割り込みです。

(a) リセット

リセットは、$\overline{\text{RESET}}$端子に “L” を入力すると発生します。

(b) NMI割り込み

NMI割り込みは、$\overline{\text{NMI}}$端子に “L” を入力すると発生します。

(c) ウォッチドッグタイマ割り込み

ウォッチドッグタイマによる割り込みです。

(d) 電圧低下検出割り込み

電圧低下検出回路による割り込みです。

(e) 発振停止検出割り込み

発振停止検出回路による割り込みです。

(f) シングルステップ割り込み

デバッガ専用割り込みですので、通常は使用しないでください。シングルステップ割り込みは、デバッグフラグ(Dフラグ)を “1” にすると、直後に発生します。

(2) 周辺機能割り込み

周辺機能割り込みは、内蔵される周辺機能による割り込みです。内蔵される周辺機能は品種ごとに異なりますので、それぞれの割り込み要因も品種ごとに異なります。割り込みベクタテーブルはINT命令で使用するソフトウェア割り込み番号0~127と同一です。周辺機能割り込みは、マスカブル割り込みです。周辺機能割り込みについての詳細はハードウェアマニュアルを参照してください。

周辺機能割り込みでは、割り込み要求受け付け時にスタックポインタ指定フラグ(Uフラグ)を退避し、Uフラグを “0” にして割り込みスタックポインタ(ISP)を選択した後、割り込みシーケンスを実行します。割り込みシーケンスから復帰するときに割り込み要求受け付け前のUフラグが復帰されます。

(3) 高速割り込み

高速割り込みは、割り込みの応答を高速に実行できる割り込みです。周辺機能割り込みの中で最優先の割り込み1つだけに使用できます。高速割り込みからの復帰にはFREIT命令を使用してください。

高速割り込みについての詳細はハードウェアマニュアルを参照してください。

5.2 割り込み制御

マスカブル割り込みの許可/禁止、受け付ける優先順位の設定について説明します。ここで説明する内容は、ノンマスカブル割り込みには該当しません。

マスカブル割り込みの許可および禁止は、割り込み許可フラグ(Iフラグ)、割り込み要求レベル選択ビット、およびプロセッサ割り込み優先レベル(IPL)によって行います。また、割り込み要求の有無は、割り込み要求ビットに示されます。割り込み要求ビットおよび割り込み要求レベル選択ビットは、各割り込みの割り込み制御レジスタに配置されています。また、割り込み許可フラグ(Iフラグ)、およびプロセッサ割り込み優先レベル(IPL)は、フラグレジスタ(FLG)に配置されています。

割り込み制御レジスタのメモリ配置、およびレジスタ構成は、ハードウェアマニュアルを参照してください。

5.2.1 割り込み許可フラグ(Iフラグ)

割り込み許可フラグ(Iフラグ)は、マスカブル割り込みの許可/禁止の制御を行います。このフラグを“1”にすると、すべてのマスカブル割り込みは許可され、“0”にすると禁止されます。このフラグはリセット解除後“0”になります。

Iフラグを変化させたとき、変化した内容が割り込み要求受け付け判定に反映されるのは次のタイミングです。

- REIT、FREIT命令で変化させたとき、そのREIT、FREIT命令から反映される。
- FCLR、FSET、POPC、LDC命令で変化させたとき、次の命令から反映される。

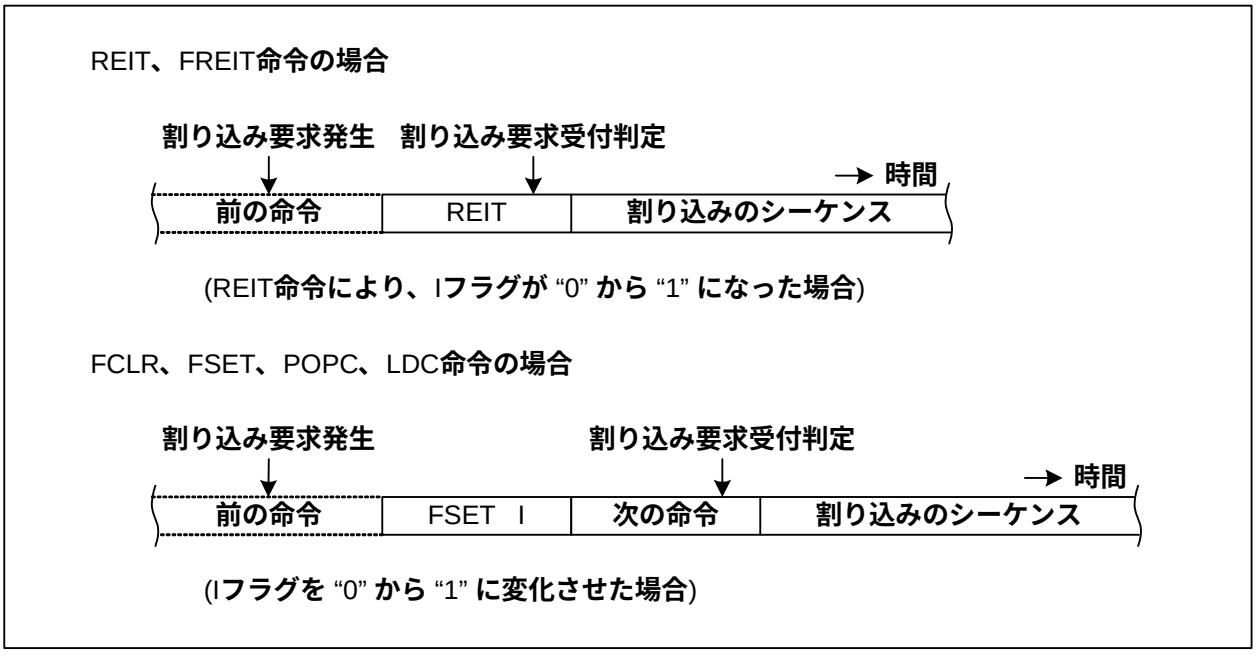


図 5.2 Iフラグを変化させたときの割り込みへの反映タイミング

5.2.2 割り込み要求ビット

割り込み要求ビットは割り込み要求が発生すると“1”になり、割り込み要求が受け付けられるまで保持されます。割り込み要求が受け付けられると“0”になります。

また、このビットはソフトウェアによって“0”にできます(“1”を書き込まないでください)。

5.2.3 割り込み要求レベルとプロセッサ割り込み優先レベル(IPL)

割り込み要求レベルは、割り込み制御レジスタの中の割り込み要求レベル選択ビットで設定します。割り込み要求発生時、割り込み要求レベルは、プロセッサ割り込み優先レベル(IPL)と比較され、割り込みの要求レベルがプロセッサ割り込み優先レベル(IPL)より大きい場合だけ、その割り込みが許可されます。したがって、割り込み要求レベルにレベル0を設定すれば、その割り込みは禁止されます。

表5.2に割り込み要求レベルの設定を、表5.3にプロセッサ割り込み優先レベル(IPL)の内容による割り込み許可レベルを示します。

表5.2 割り込み要求レベルの設定

割り込み要求レベル選択ビット			割り込み要求レベル	優先順位
b2	b1	b0		
1	1	1	レベル7	高い
1	1	0	レベル6	
1	0	1	レベル5	
1	0	0	レベル4	
0	1	1	レベル3	
0	1	0	レベル2	
0	0	1	レベル1	低い
0	0	0	レベル0(割り込み禁止)	—

表5.3 プロセッサ割り込み優先レベル(IPL)の内容による割り込み許可レベル

プロセッサ割り込み優先レベル(IPL)			許可される割り込み要求レベル
IPL_2	IPL_1	IPL_0	
1	1	1	すべてのマスカブル割り込みを禁止
1	1	0	レベル7のみを許可
1	0	1	レベル6以上を許可
1	0	0	レベル5以上を許可
0	1	1	レベル4以上を許可
0	1	0	レベル3以上を許可
0	0	1	レベル2以上を許可
0	0	0	レベル1以上を許可

割り込み要求が受け付けられる条件を以下に示します。

- 割り込み許可フラグ(Iフラグ) = “1”
- 割り込み要求ビット = “1”
- 割り込み要求レベル > プロセッサ割り込み優先レベル(IPL)

割り込み許可フラグ(Iフラグ)、割り込み要求ビット、割り込み要求レベル選択ビット、およびプロセッサ割り込み優先レベル(IPL)はそれぞれ独立しており、互いに影響を与えることはありません。

プロセッサ割り込み優先レベル(IPL)または、各割り込み要求レベルを変更させたとき、変化したレベルが割り込みに反映するのは以下のタイミングです。

- REIT、FREIT命令でプロセッサ割り込み優先レベル(IPL)を変化させたとき、そのREIT、FREIT命令から反映される。
- POPC、LDC、LDIPL命令でプロセッサ割り込み優先レベル(IPL)を変化させたとき、次の命令から反映される。
- 各割り込み制御レジスタの割り込み要求レベルをMOV命令等で変化させたとき、レジスタの値が書き換わって周辺バスクロックの1サイクル経過後から反映される。

5.2.4 割り込み制御レジスタの変更

割り込み制御レジスタの変更は、そのレジスタに対応する割り込み要求が発生しない箇所で行ってください。割り込み要求が発生する可能性がある場合は、割り込みを禁止状態にしてから変更してください。

また、割り込み制御レジスタを変更した直後に割り込みを許可状態にするときは、命令キューの影響により割り込み許可フラグ(Iフラグ)のセットが割り込み制御レジスタの書き込みより先に実行されないように、NOPを挿入するなどの対策を行ってください。

割り込みが禁止状態で、割り込み制御レジスタを書き換える命令を実行しているときに、そのレジスタに対応する割り込み要求が発生した場合、命令によっては割り込み要求ビットがセットされないことがあります。このことが問題になる場合は、以下の命令を使用してレジスタを変更するようにしてください。

- AND
- OR
- BCLR
- BSET

5.3 割り込みシーケンス

割り込み要求が受け付けられてから割り込みルーチンが実行されるまでの割り込みシーケンスについて説明します。

命令実行中に割り込み要求が発生すると、その命令の実行後に優先順位が判定され、次のサイクルから割り込みシーケンスに移ります。ただし、RMPA、SCMPU、SIN、SMOVB、SMOVF、SMOVU、SOUT、SSTR、SUNTIL、SWHILEの各命令は、命令実行中に割り込み要求が発生すると、命令の動作を一時中断し割り込みシーケンスに移ります。

割り込みシーケンスでは、次の動作を順次行います。

(1) CPUは割り込みアクノリッジを返すことで、割り込みコントローラから割り込み情報(割り込み番号、割り込み要求レベル)を取得する。その後、該当する割り込みの要求ビットが “0” になる。
(2) 割り込みシーケンス直前のフラグレジスタ(FLG)の内容をCPU内部の一時レジスタ*1に退避する。
(3) 割り込み許可フラグ(Iフラグ)、デバッグフラグ(Dフラグ)、およびスタックポインタ指定フラグ(Uフラグ)を “0” にする(ただしUフラグは、ソフトウェア割り込み番号128~255のINT命令を実行した場合は変化しません)。
(4) CPU内部の一時レジスタ*1の内容をスタック領域に退避する。高速割り込みの場合は、フラグ退避レジスタ(SVF)に退避する。
(5) プログラムカウンタ(PC)の内容をスタック領域に退避する。高速割り込みの場合は、PC退避レジスタ(SVP)に退避する。
(6) プロセッサ割り込み優先レベル(IPL)に、受け付けた割り込みの割り込み要求レベルを設定する。
(7) 割り込みベクタテーブルから受け付けた割り込み要因のベクタを取得する。
(8) 取得した割り込みベクタをプログラムカウンタ(PC)にセットする。

割り込みシーケンス終了後は、割り込みルーチンの先頭番地から命令を実行します。

*1 ユーザは使用できません。

5.3.1 割り込み応答時間

割り込み応答時間とは、割り込み要求が発生してから割り込みルーチン内の最初の命令を実行するまでの時間を示します。この時間は、割り込み要求発生時点から、そのとき実行している命令が終了するまでの時間(a)と割り込みシーケンスを実行する時間(b)で構成されます。図 5.3に割り込み応答時間を示します。

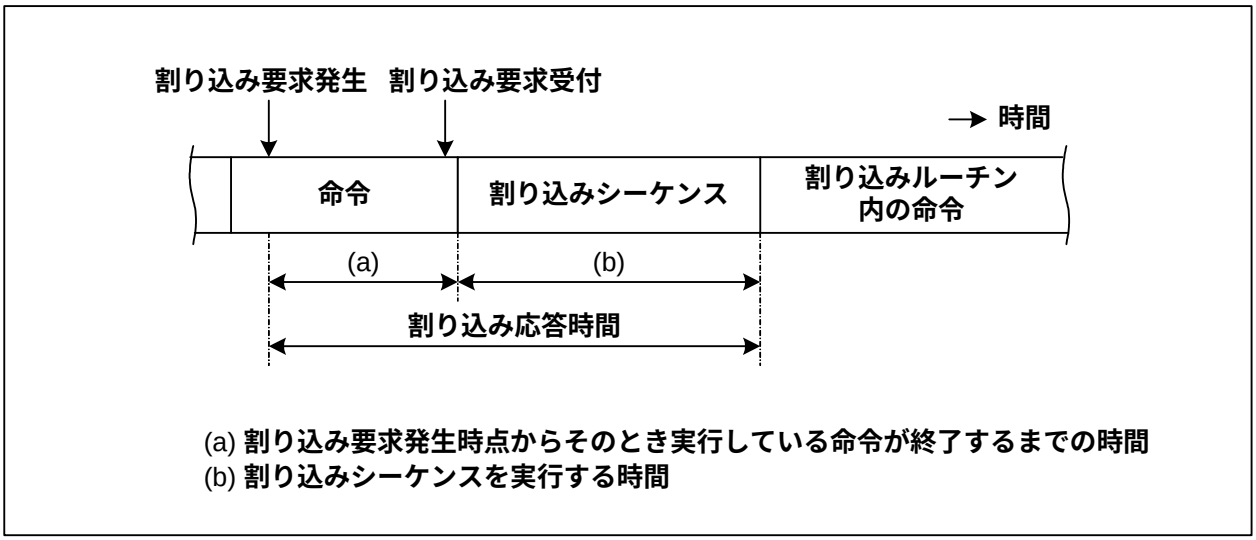


図 5.3 割り込み応答時間

(a)の時間は、実行している命令によって異なります。
(b)の時間は表 5.4のとおりです。

表 5.4 割り込みシーケンス実行時間

割り込み	割り込みベクタの番地	割り込みシーケンス実行時間(内部メモリ)
周辺機能	4**バイト境界***1	13**サイクル***2
INT**命令**	4**バイト境界***1	11**サイクル**
BRK**命令(可変ベクタテーブル)**	4**バイト境界***1	16**サイクル**
未定義命令	FFFFFFDCh*3	12**サイクル**
オーバフロー	FFFFFFE0h*3	12**サイクル**
BRK**命令(固定ベクタテーブル)**	FFFFFFE4h*3	19**サイクル**
ウォッチドッグタイマ **電圧低下検出** **発振停止検出**	FFFFFFF0h*3	11**サイクル**
NMI	FFFFFFF8h*3	10**サイクル**
シングルステップ BRK2**命令** DBC**割り込み**	4**バイト境界***3	19**サイクル**
高速割り込み*4	**ベクタテーブルは** **内部レジスタ**	11**サイクル**

*1 割り込みベクタテーブルは、なるべく4バイト境界に配置するようにしてください。4バイト境界以外に配置した場合、実行時間が延びます。
*2 周辺バスのウェイト数が3以上のときは、(ウェイト数-2)だけサイクル数が増加します。
*3 ベクタテーブルのアドレスは4バイト境界固定です。
*4 高速割り込みは、これらの条件に影響されません。

5.3.2 プロセッサ割り込み優先レベル(IPL)の変化

割り込み要求が受け付けられると、プロセッサ割り込み優先レベル(IPL)には受け付けた割り込みの割り込み要求レベルが設定されます。

割り込み要求レベルを持たない割り込み要求が受け付けられたときは、表 5.5に示す値がIPLに設定されます。

表 5.5 割り込み要求レベルを持たない割り込みとIPLの関係

割り込み要求レベルを持たない割り込み要因	設定されるIPLの値
NMI、ウォッチドッグタイマ、電圧低下検出、発振停止検出	7
リセット	0
その他	変化しない

5.3.3 レジスタ退避

割り込みシーケンスでは、フラグレジスタ(FLG)とプログラムカウンタ(PC)の内容だけがスタック領域に退避されます。スタック領域へ退避する順番は、フラグレジスタ–プログラムカウンタの順です。図 5.4に割り込み要求受付前のスタックの状態と、割り込み要求受付後のスタックの状態を示します。

高速割り込みの割り込みシーケンスでは、フラグレジスタ(FLG)はフラグ退避レジスタ(SVF)に、プログラムカウンタ(PC)はPC退避レジスタ(SVP)に退避されます。

その他の必要なレジスタは、割り込みルーチンの最初でソフトウェアによって退避してください。PUSHM命令を用いると、1命令でスタックポインタ(SP)を除くすべてのレジスタを退避することができます。

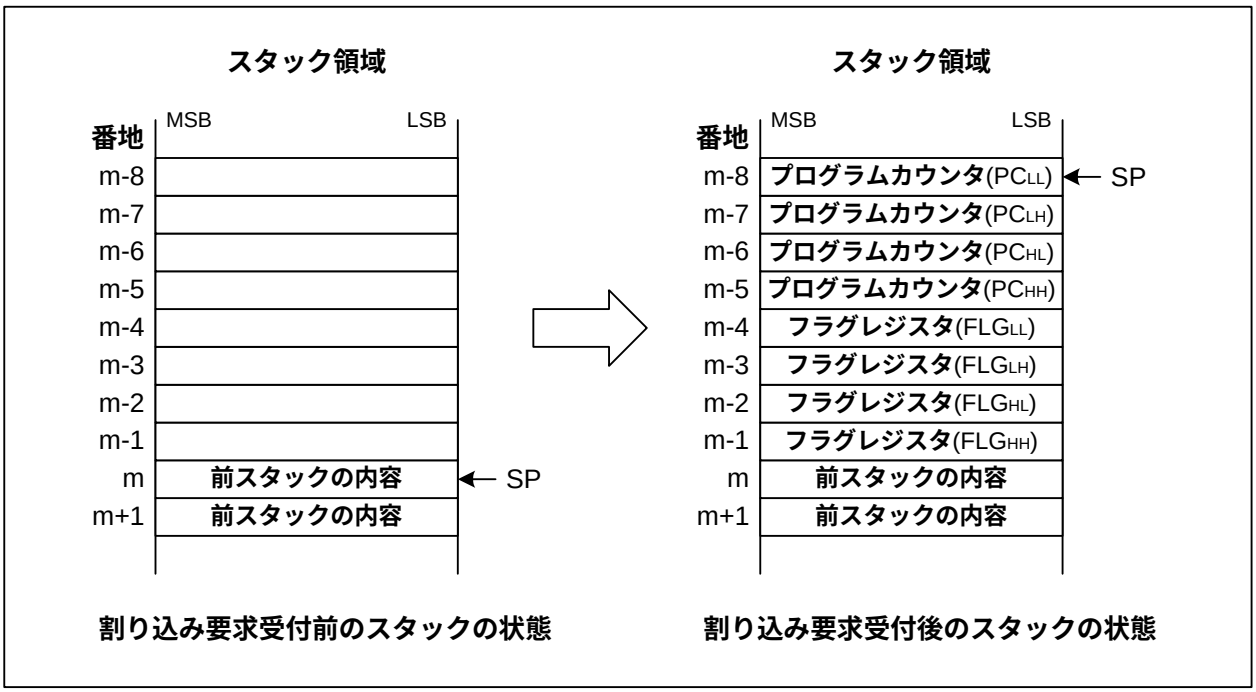


図 5.4 割り込み要求受付前/割り込み要求受付後のスタックの状態

5.4 割り込みルーチンからの復帰

割り込みルーチンの最後でREIT命令を実行すると、スタック領域に退避されていた割り込みシーケンス直前のフラグレジスタ(FLG)、およびプログラムカウンタ(PC)の内容が復帰されます。高速割り込みの場合は、割り込みルーチンの最後でFREIT命令を実行すると、退避レジスタに退避されていた割り込みシーケンス直前のフラグレジスタ(FLG)、およびプログラムカウンタ(PC)の内容が復帰されます。その後、割り込み要求受付前に実行していたプログラムに戻り、中断されていた処理が継続して実行されます。

割り込みルーチン内でソフトウェアによって退避したレジスタは、REIT、FREIT命令実行前にPOPM命令などを使用して復帰してください。

5.5 割り込み優先順位

同一サンプリング時点(割り込みの要求があるかどうかを調べるタイミング)で2つ以上の割り込み要求が存在した場合は、優先順位の高い割り込みが受け付けられます。

マスカブル割り込み(周辺機能割り込み)の優先順位は、割り込み要求レベル選択ビットによって任意の優先順位を設定することができます。ただし、割り込み要求レベルが同じ設定の場合は、ハードウェアで設定されている優先度*1の高い割り込みが受け付けられます。

リセット(リセットは優先順位が一番高い割り込みとして扱われます)、ウォッチドッグタイマ割り込みなど、ノンマスカブル割り込みの優先順位はハードウェアで設定されています。以下にハードウェアで設定されている割り込み優先順位を示します。

ウォッチドッグタイマ
リセット > 電圧低下検出 > NMI > 周辺機能 > シングルステップ
発振停止検出

ソフトウェア割り込みは割り込み優先順位の影響を受けません。命令を実行すると必ず割り込みルーチンへ分岐します。

*1 品種により異なります。ハードウェアマニュアルを参照してください。

5.6 多重割り込み

割り込みルーチンへ分岐したときの状態を以下に示します。

- 割り込み許可フラグ(Iフラグ)は“0”(割り込み禁止状態)
- 受け付けた割り込みの割り込み要求ビットは“0”
- プロセッサ割り込み優先レベル(IPL)は受け付けた割り込みの割り込み要求レベル

割り込みルーチン内で割り込み許可フラグ(Iフラグ)を“1”にすることによって、プロセッサ割り込み優先レベル(IPL)より高い要求レベルを持つ割り込み要求を受け付けることができます。図5.5に多重割り込みの処理例を示します。

なお、優先順位が低いために受け付けられなかった割り込み要求は保持されます。REIT、FREIT命令によってIPLが復帰され、割り込み要求レベルの判定が行われたとき、保持されていた割り込み要求の割り込み要求レベルが復帰されたプロセッサ割り込み優先レベル(IPL)より大きければ、保持されていた割り込み要求が受け付けられます。

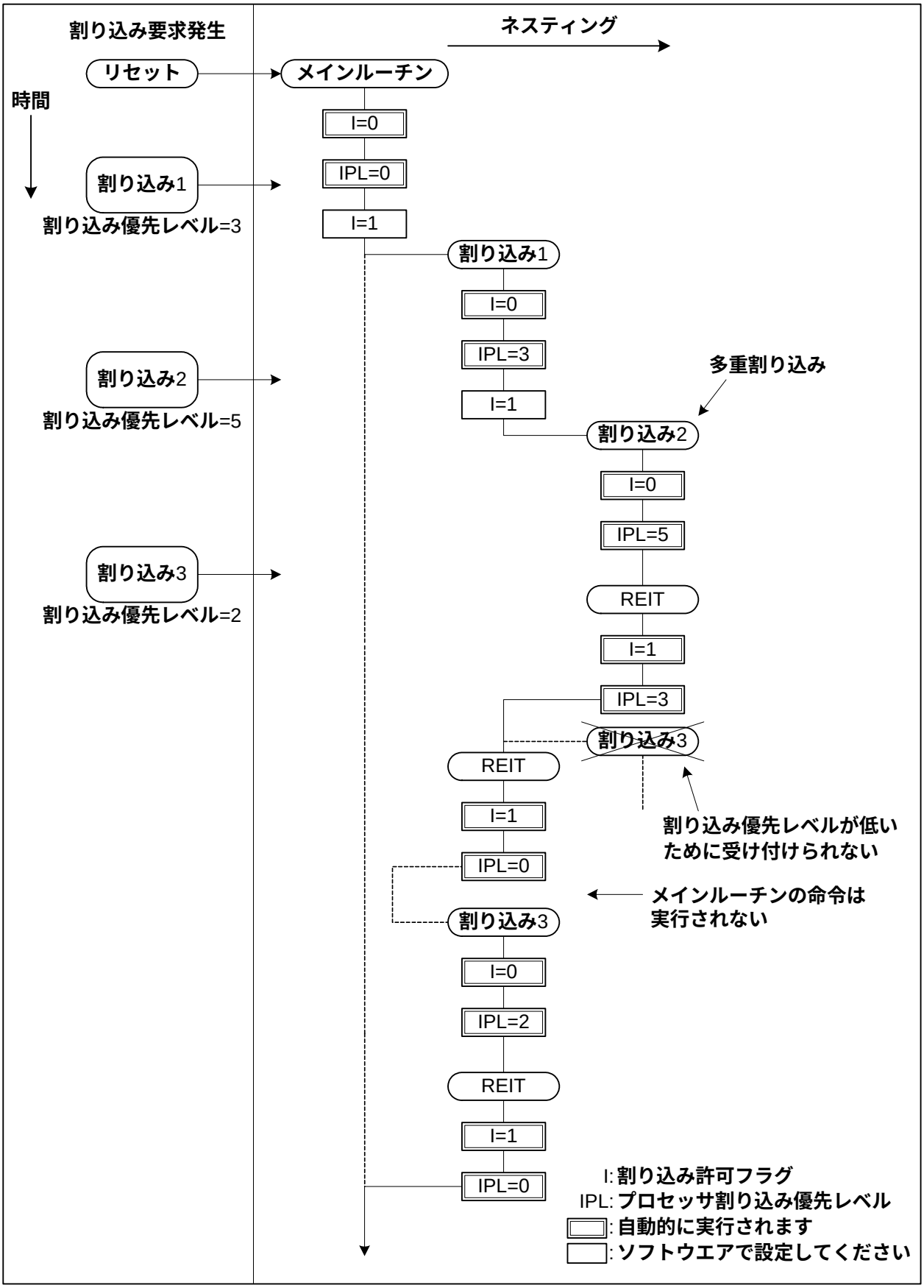

割り込み要求発生
ネスティング
リセット
メインルーチン
時間
I=0
IPL=0
割り込み1
I=1
割り込み優先レベル=3
割り込み1
I=0
IPL=3
I=1
多重割り込み
割り込み2
割り込み優先レベル=5
割り込み2
I=0
IPL=5
REIT
I=1
IPL=3
割り込み3
割り込み3
REIT
I=1
割り込み優先レベルが低い
ためにIPL=0
受け付けられない
メインルーチンの命令は
実行されない
割り込み3
I=0
IPL=2
REIT
I=1
IPL=0
I: 割り込み許可フラグ
IPL: プロセッサ割り込み優先レベル
: 自動的に実行されます
: ソフトウエアで設定してください

図 5.5 多重割り込み

5.7 割り込みの注意事項

以下に割り込みを使用する上での注意事項を列挙します。

(1) スタックポインタの設定

リセット直後、スタックポインタの値は00000000hに初期化されています。そのため、スタックポインタに値を設定する前に割り込みを受け付けると暴走の要因となります。割り込みを許可する前に、必ずスタックポインタに値を設定してください。

特に、NMI割り込みを使用する場合は、プログラムの先頭でスタックポインタを設定してください。NMI割り込みは、PM2レジスタのPM24ビットを“1”にした直後から受け付けられます。

スタックポインタには4の倍数のアドレスを設定してください。4の倍数を設定したほうが命令実行効率が良くなります。

(2) 割り込み制御レジスタの変更

割り込み制御レジスタの変更は、そのレジスタに対応する割り込み要求が発生しない箇所で行ってください。割り込み要求が発生する可能性がある場合は、割り込みを禁止状態にしてから変更してください。

また、割り込み制御レジスタを変更した直後に割り込みを許可状態にするときは、命令キューの影響により割り込み許可フラグ(Iフラグ)のセットが割り込み制御レジスタの書き込みより先に実行されないように、NOPを挿入するなどの対策を行ってください。

割り込みが禁止状態で、割り込み制御レジスタを書き換える命令を実行しているときに、そのレジスタに対応する割り込み要求が発生した場合、命令によっては割り込み要求ビットがセットされないことがあります。このことが問題になる場合は、以下の命令を使用してレジスタを変更するようにしてください。

- AND
- OR
- BCLR
- BSET

(3) ストップモード/ウェイトモードの解除

周辺機能割り込みでストップモード/ウェイトモードを解除する場合、対象となる割り込みはあらかじめ割り込み許可状態にし、割り込みの要求レベルをストップ/ウェイト復帰用割り込み優先レベル選択ビットで指定したレベルより高いレベルに設定する必要があります。ストップ/ウェイト復帰用割り込み優先レベル選択ビットにはフラグレジスタ(FLG)のプロセッサ割り込みレベル(IPL)と同じ値を設定してください。

リセット、NMI割り込みは、ストップ/ウェイト復帰用割り込み優先レベル選択ビットの影響を受けず、ストップモード/ウェイトモードを解除します。

索引

記号

数字

A

B

C

D

F

G

I

N

O

P

R

S

U

V

Z

あ

い

う

え

お

か

き

く

こ

め

ゆ

り

れ

わ

改訂記録	R32C/100シリーズ ソフトウェアマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2009.01.07	—	初版発行

R32C/100シリーズ ソフトウェアマニュアル

発行年月日　2009年1月7日　　Rev.1.00

発行　　株式会社 ルネサス テクノロジ　営業統括部
〒100-0004　東京都千代田区大手町2-6-2

R32C/100 シリーズ
ソフトウェアマニュアル

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ09B0272-0100