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

16

R8C/Tinyシリーズ

ソフトウェアマニュアル

ルネサス16ビットシングルチップマイクロコンピュータ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2003.06

安全設計に関するお願い

・弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

・本資料は、お客様が用途に応じた適切なルネサス テクノロジ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ルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。

・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の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。

・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全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ（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。

・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誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。

・本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位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。ルネサス テクノロジは、適用可否に対する責任は負いません。

・本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。

・本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。

・本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

本書の使い方

本書はR8C/Tinyシリーズのソフトウエアマニュアルです。R8C/TinyシリーズのCPUコアをもつ全品種で共通に使用することができます。

本書を使用する上で、電気回路、論理回路、およびマイクロコンピュータの基本的な知識が必要です。

本書は6つの章で構成されています。以下に、目的に応じた参照先(章、節)を示します。

○R8C/Tiny シリーズの概要、特長を理解する ..... **第1章「概要」**

○各アドレッシングモードの動作を理解する ..... **第2章「アドレッシングモード」**

○命令機能を理解する
(構文、オペレーション、機能、選択可能なsrc / dest(label)、フラグ変化、記述例、関連命令) ..... **第3章「機能」**

○命令コード、サイクル数を理解する ..... **第4章「命令コード／サイクル数」**

○割り込みを理解する ..... **第5章「割り込み」**

○サイクル数の計算方法を理解する ..... **第6章「サイクル数の計算」**

また、目次の直後には各種ページ早見表を記載しており、目的に応じて機能と命令コード／サイクル数の記載ページを調べることができます。

○ニーモニックから記載ページを確認する ..... **アルファベット別ページ早見表**

○機能からニーモニックと記載ページを確認する ..... **機能別ページ早見表**

○ニーモニックからアドレッシングと記載ページを確認する ..... **アドレッシング別ページ早見表**

さらに、巻末にはQ&A、用語集、記号集、および索引を記載しています。

Ｍ１６Ｃファミリ関連ドキュメント

M16Cファミリでは次のドキュメントを用意しています。

ドキュメントの種類	記載内容
ショートシート	ハードウエアの概要
データシート	ハードウエアの概要と電気的特性
ハードウエアマニュアル	ハードウエアの仕様(ピン配置、メモリマップ、周辺機能の仕様、電気的特性、タイミング)
ソフトウエアマニュアル	命令 (アセンブリ言語) の動作の詳細
アプリケーションノート	周辺機能の応用例 参考プログラム M16Cファミリ入門用基本機能説明 アセンブリ言語、C言語によるプログラムの作成方法

目次

アルファベット別ページ早見表

ニーモニック	機能記載ページ	命令コード/サイクル数記載ページ	ニーモニック	機能記載ページ	命令コード/サイクル数記載ページ
ABS	39	138	DIVU	68	171
ADC	40	138	DIVX	69	172
ADCF	41	140	DSBB	70	173
ADD	42	140	DSUB	71	175
ADJNZ	44	146	ENTER	72	177
AND	45	147	EXITD	73	178
BAND	47	150	EXTS	74	178
BCLR	48	150	FCLR	75	179
BM*Cnd*	49	152	FSET	76	180
BMEQ/Z	49	152	INC	77	180
BMGE	49	152	INT	78	181
BMGEU/C	49	152	INTO	79	182
BMGT	49	152	J*Cnd*	80	182
BMGTU	49	152	JEQ/Z	80	182
BMLE	49	152	JGE	80	183
BMLEU	49	152	JGEU/C	80	182
BMLT	49	152	JGT	80	183
BMLTU/NC	49	152	JGTU	80	182
BMN	49	152	JLE	80	183
BMNE/NZ	49	152	JLEU	80	182
BMNO	49	152	JLT	80	183
BMO	49	152	JLTU/NC	80	182
BMPZ	49	152	JN	80	182
BNAND	50	153	JNE/NZ	80	182
BNOR	51	154	JNO	80	183
BNOT	52	154	JO	80	183
BNTST	53	155	JPZ	80	182
BNXOR	54	156	JMP	81	183
BOR	55	156	JMPI	82	185
BRK	56	157	JSR	83	187
BSET	57	157	JSRI	84	188
BTST	58	158	LDC	85	189
BTSTC	59	159	LDCTX	86	190
BTSTS	60	160	LDE	87	191
BXOR	61	160	LDINTB	88	192
CMP	62	161	LDIPL	89	193
DADC	64	165	MOV	90	193
DADD	65	167	MOVA	92	200
DEC	66	169			
DIV	67	170			

アルファベット別ページ早見表

ニーモニック	機能記載ページ	命令コード/サイクル数記載ページ	ニーモニック	機能記載ページ	命令コード/サイクル数記載ページ
MOV*Dir*	93	201	ROT	112	220
MOVHH	93	201	RTS	113	221
MOVHL	93	201	SBB	114	222
MOVLH	93	201	SBJNZ	115	224
MOVLL	93	201	SHA	116	225
MUL	94	203	SHL	117	228
MULU	95	205	SMOVB	118	230
NEG	96	207	SMOVF	119	231
NOP	97	207	SSTR	120	231
NOT	98	208	STC	121	232
OR	99	209	STCTX	122	233
POP	101	211	STE	123	233
POPC	102	213	STNZ	124	235
POPM	103	213	STZ	125	235
PUSH	104	214	STZX	126	236
PUSHA	105	216	SUB	127	236
PUSHC	106	216	TST	129	239
PUSHM	107	217	UND	130	241
REIT	108	217	WAIT	131	241
RMPA	109	218	XCHG	132	242
ROLC	110	218	XOR	133	243
RORC	111	219			

機能別ページ早見表

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	90	193
	MOVA	実効アドレスの転送	92	200
	MOVDir	４ビットデータ転送	93	201
	POP	レジスタ／メモリの復帰	101	211
	POPM	複数レジスタの復帰	103	213
	PUSH	レジスタ／メモリ／即値の退避	104	214
	PUSHA	実効アドレスの退避	105	216
	PUSHM	複数レジスタの退避	107	217
	LDE	拡張データ領域からの転送	87	191
	STE	拡張データ領域への転送	123	233
	STNZ	条件付き転送	124	235
	STZ	条件付き転送	125	235
	STZX	条件付き転送	126	236
	XCHG	交換	132	242
ビット処理	BAND	ビット論理積	47	150
	BCLR	ビットクリア	48	150
	BM*Cnd*	条件ビット転送	49	152
	BNAND	反転ビット論理積	50	153
	BNOR	反転ビット論理和	51	154
	BNOT	ビット反転	52	154
	BNTST	反転ビットテスト	53	155
	BNXOR	反転ビットの論理和	54	156
	BOR	ビット論理和	55	156
	BSET	ビットセット	57	157
	BTST	ビットテスト	58	158
	BTSTC	ビットテスト＆クリア	59	159
	BTSTS	ビットテスト＆セット	60	160
	BXOR	ビット排他的論理和	61	160
シフト	ROLC	キャリー付き左回転	110	218
	RORC	キャリー付き右回転	111	219
	ROT	回転	112	220
	SHA	算術シフト	116	225
	SHL	論理シフト	117	228
算術	ABS	絶対値	39	138
	ADC	キャリー付き加算	40	138
	ADCF	キャリーフラグの加算	41	140
	ADD	キャリーなし加算	42	140
	CMP	比較	62	161
	DADC	キャリー付き10進加算	64	165

機能別ページ早見表

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	DADD	キャリーなし10進加算	65	167
	DEC	デクリメント	66	169
	DIV	符号付き除算	67	170
	DIVU	符号なし除算	68	171
	DIVX	符号付き除算	69	172
	DSBB	ボロー付き10進減算	70	173
	DSUB	ボローなし10進減算	71	175
	EXTS	符号拡張	74	178
	INC	インクリメント	77	180
	MUL	符号付き乗算	94	203
	MULU	符号なし乗算	95	205
	NEG	2の補数	96	207
	RMPA	積和演算	109	218
	SBB	ボロー付き減算	114	222
	SUB	ボローなし減算	127	236
論理	AND	論理積	45	147
	NOT	全ビット反転	98	208
	OR	論理和	99	209
	TST	テスト	129	239
	XOR	排他的論理和	133	243
ジャンプ	ADJNZ	加算＆条件分岐	44	146
	SBJNZ	減算＆条件分岐	115	224
	J*Cnd*	条件分岐	80	182
	JMP	無条件分岐	81	183
	JMPI	間接分岐	82	185
	JSR	サブルーチン呼び出し	83	187
	JSRI	間接サブルーチン呼び出し	84	188
	RTS	サブルーチンからの復帰	113	221
ストリング	SMOVB	逆方向のストリング転送	118	230
	SMOVF	順方向のストリング転送	119	231
	SSTR	ストリングストア	120	231
その他	BRK	デバッグ割り込み	56	157
	ENTER	スタックフレームの構築	72	177
	EXITD	スタックフレームの解放	73	178
	FCLR	フラグレジスタのビットクリア	75	179
	FSET	フラグレジスタのビットセット	76	180
	INT	ソフトウエア割り込み	78	181
	INTO	オーバフロー割り込み	79	182
	LDC	専用レジスタへの転送	85	189

機能別ページ早見表

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
その他	LDCTX	コンテキストの復帰	86	190
	LDINTB	INTBレジスタへの転送	88	192
	LDIPL	割り込み許可レベルの設定	89	193
	NOP	ノーオペレーション	97	207
	POPC	専用レジスタの復帰	102	213
	PUSHC	専用レジスタの退避	106	216
	REIT	割り込みからの復帰	108	217
	STC	専用レジスタからの転送	121	232
	STCTX	コンテキストの退避	122	233
	UND	未定義命令割り込み	130	241
	WAIT	ウエイト	131	241

アドレッシング別ページ早見表（一般命令アドレッシング）

ニーモニック	アドレッシング															機能記載ページ	命令コード／サイクル数記載ページ
	R0L/R0	R0H/R1	R1L/R2	R1H/R3	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16	#IMM8	#IMM16	#IMM20	#IMM		
ABS	○	○	○	○	○	○	○	○	○	○	○					39	138
ADC	○	○	○	○	○	○	○	○	○	○	○	○	○			40	138
ADCF	○	○	○	○	○	○	○	○	○	○	○					41	140
ADD*1	○	○	○	○	○	○	○	○	○	○	○	○	○			42	140
ADJNZ*1	○	○	○	○	○	○	○	○	○	○	○				○	44	146
AND	○	○	○	○	○	○	○	○	○	○	○	○	○			45	147
CMP	○	○	○	○	○	○	○	○	○	○	○	○	○			62	161
DADC	○	○										○	○			64	165
DADD	○	○										○	○			65	167
DEC	○	○			○			○			○					66	169
DIV	○	○	○	○	○	○	○	○	○	○	○	○	○			67	170
DIVU	○	○	○	○	○	○	○	○	○	○	○	○	○			68	171
DIVX	○	○	○	○	○	○	○	○	○	○	○	○	○			69	172
DSBB	○	○										○	○			70	173
DSUB	○	○										○	○			71	175
ENTER												○				72	177
EXTS	○		○*2			○	○	○	○	○	○					74	178
INC	○*3	○*4			○			○			○					77	180
INT															○	78	181
JMPI*1	○	○	○	○	○	○	○	○		○	○					82	185
JSRI*1	○	○	○	○	○	○	○	○		○	○					84	188
LDC*1	○	○	○	○	○	○	○	○	○	○	○		○			85	189
LDE*1	○	○	○	○	○	○	○	○	○	○	○					87	191
LDINTB														○		88	192
LDIPL															○	89	193

*1　特定命令アドレッシングをもちます。

*2　R1Lだけ選択できます。

*3　R0Lだけ選択できます。

*4　R0Hだけ選択できます。

アドレッシング別ページ早見表（一般命令アドレッシング）

ニーモニック	アドレッシング															機能記載ページ	命令コード／サイクル数記載ページ
	R0L/R0	R0H/R1	R1L/R2	R1H/R3	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16	#IMM8	#IMM16	#IMM20	#IMM		
MOV[*1]	○	○	○	○	○	○	○	○	○	○	○	○	○			90	193
MOVA	○	○	○	○	○		○	○	○	○	○					92	200
MOV*Dir*	○	○	○	○		○	○	○	○	○	○					93	201
MUL	○	○	○	○	○	○	○	○	○	○	○	○	○			94	203
MULU	○	○	○	○	○	○	○	○	○	○	○	○	○			95	205
NEG	○	○	○	○	○	○	○	○	○	○	○					96	207
NOT	○	○	○	○	○	○	○	○	○	○	○					98	208
OR	○	○	○	○	○	○	○	○	○	○	○	○	○			99	209
POP	○	○	○	○	○	○	○	○	○	○	○					101	211
POPM[*1]	○	○	○	○	○											103	213
PUSH	○	○	○	○	○	○	○	○	○	○	○					104	214
PUSHA							○	○	○	○	○					105	216
PUSHM[*1]	○	○	○	○	○											107	217
ROLC	○	○	○	○	○	○	○	○	○	○	○					110	218
RORC	○	○	○	○	○	○	○	○	○	○	○					111	219
ROT	○	○	○	○	○	○	○	○	○	○	○				○	112	220
SBB	○	○	○	○	○	○	○	○	○	○	○	○	○			114	222
SBJNZ[*1]	○	○	○	○	○	○	○	○	○	○	○				○	115	224
SHA[*1]	○	○	○	○	○	○	○	○	○	○	○				○	116	225
SHL[*1]	○	○	○	○	○	○	○	○	○	○	○				○	117	228
STC[*1]	○	○	○	○	○	○	○	○	○	○	○					121	232
STCTX[*1]											○					122	233
STE[*1]	○	○	○	○	○	○	○	○	○	○	○					123	233
STNZ	○	○						○			○	○				124	235
STZ	○	○						○			○	○				125	235

*1　特定命令アドレッシングをもちます。

アドレッシング別ページ早見表（一般命令アドレッシング）

ニーモニック	アドレッシング															機能記載ページ	命令コード／サイクル数記載ページ
	R0L/R0	R0H/R1	R1L/R2	R1H/R3	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16	#IMM8	#IMM16	#IMM20			
STZX	○	○						○			○	○				126	236
SUB	○	○	○	○	○	○	○	○	○	○	○	○	○			127	236
TST	○	○	○	○	○	○	○	○	○	○	○	○	○			129	239
XCHG	○	○	○	○	○	○	○	○	○	○	○					132	242
XOR	○	○	○	○	○	○	○	○	○	○	○	○	○			133	243

*1　特定命令アドレッシングをもちます。

アドレッシング別ページ早見表（特定命令アドレッシング）

ニーモニック	アドレッシング													機能記載ページ	命令コード／サイクル数記載ページ
	dsp:20[A0]	dsp:20[A1]	abs20	R2R0/R3R1	A1A0	[A1A0]	dsp:8[SP]	label	SB/FB	ISP/USP	FLG	INTBL/INTBH	PC		
ADD*1										○				42	140
ADJNZ*1								○						44	146
J*Cnd*								○						80	182
JMP			○					○						81	183
JMPI*1	○	○		○	○									82	185
JSR			○					○						83	187
JSRI*1	○	○		○	○									84	188
LDC*1									○	○	○	○		85	189
LDCTX			○											86	190
LDE*1	○		○			○								87	191
LDINTB												○*2		88	192
MOV*1							○							90	193
POPC									○	○	○	○		102	213
POPM*1									○					103	213
PUSHC									○	○	○	○		106	216
PUSHM*1									○					107	217
SBJNZ*1								○						115	224
SHA*1				○										116	225
SHL*1				○										117	228
STC*1				○	○				○	○	○	○	○	121	232
STCTX*1			○											122	233
STE*1	○		○			○								123	233

*1　一般命令アドレッシングをもちます。

*2　LDINTB命令を使用すると、INTBL、INTBHを同時に設定することができます。

アドレッシング別ページ早見表（ビット命令アドレッシング）

ニーモニック	アドレッシング										機能記載ページ	命令コード／サイクル数記載ページ
	bit,Rn	bit,An	[An]	base:8[An]	bit,base:8[SB/FB]	base:16[An]	bit,base:16[SB]	bit,base:16	bit,base:11	U/I/O/B/S/Z/D/C		
BAND	○	○	○	○	○	○	○	○			47	150
BCLR	○	○	○	○	○	○	○	○	○		48	150
BM*Cnd*	○	○	○	○	○	○	○	○		○	49	152
BNAND	○	○	○	○	○	○	○	○			50	153
BNOR	○	○	○	○	○	○	○	○			51	154
BNOT	○	○	○	○	○	○	○	○	○		52	154
BNTST	○	○	○	○	○	○	○	○			53	155
BNXOR	○	○	○	○	○	○	○	○			54	156
BOR	○	○	○	○	○	○	○	○			55	156
BSET	○	○	○	○	○	○	○	○	○		57	157
BTST	○	○	○	○	○	○	○	○	○		58	158
BTSTC	○	○	○	○	○	○	○	○			59	159
BTSTS	○	○	○	○	○	○	○	○			60	160
BXOR	○	○	○	○	○	○	○	○			61	160
FCLR										○	75	179
FSET										○	76	180

第１章

概要

1.1 R8C/Tiny シリーズの特長

R8C/Tiny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もち､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もたせ､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89種類の命令セットをもち､レジスターレジスタ､レジスターメモリ､メモリー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。

乗算器を内蔵している品種では､高速な乗算ができます。

1.1.1 R8C/Tiny シリーズの特長

●レジスタ構成

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

●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	：BNOT、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

●高速な命令実行時間

最短1サイクル命令：89命令中20命令が1サイクル命令をもつ
（約75%の命令が5サイクル以下）

1.1.2 スピード性能

レジスタ間転送	0.1 μs
レジスターメモリ間転送	0.1 μs
レジスタ間加減算	0.1 μs
8ビット×8ビットレジスタ間演算	0.2 μs
16ビット×16ビットレジスタ間演算	0.250 μs
16ビット÷8ビットレジスタ間演算	0.904 μs
32ビット÷16ビットレジスタ間演算	1.248 μs

[条件]
・乗算器内蔵品種
・クロック周波数　20MHz

1.2 アドレス空間

アドレス空間を図 1.2.1 に示します。

00000$_{16}$番地～002FF$_{16}$番地はSFR（スペシャルファンクションレジスタ）領域です。品種展開では002FF$_{16}$番地から番地の小さい方向に対して SFR 領域を拡張します。

00400$_{16}$ 番地以降はメモリ領域です。品種展開では 00400$_{16}$ 番地から番地の大きい方向に対して RAM 領域を、0FFFF$_{16}$ 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、0FFDC$_{16}$ 番地～ 0FFFF$_{16}$ 番地は固定ベクタ領域です。

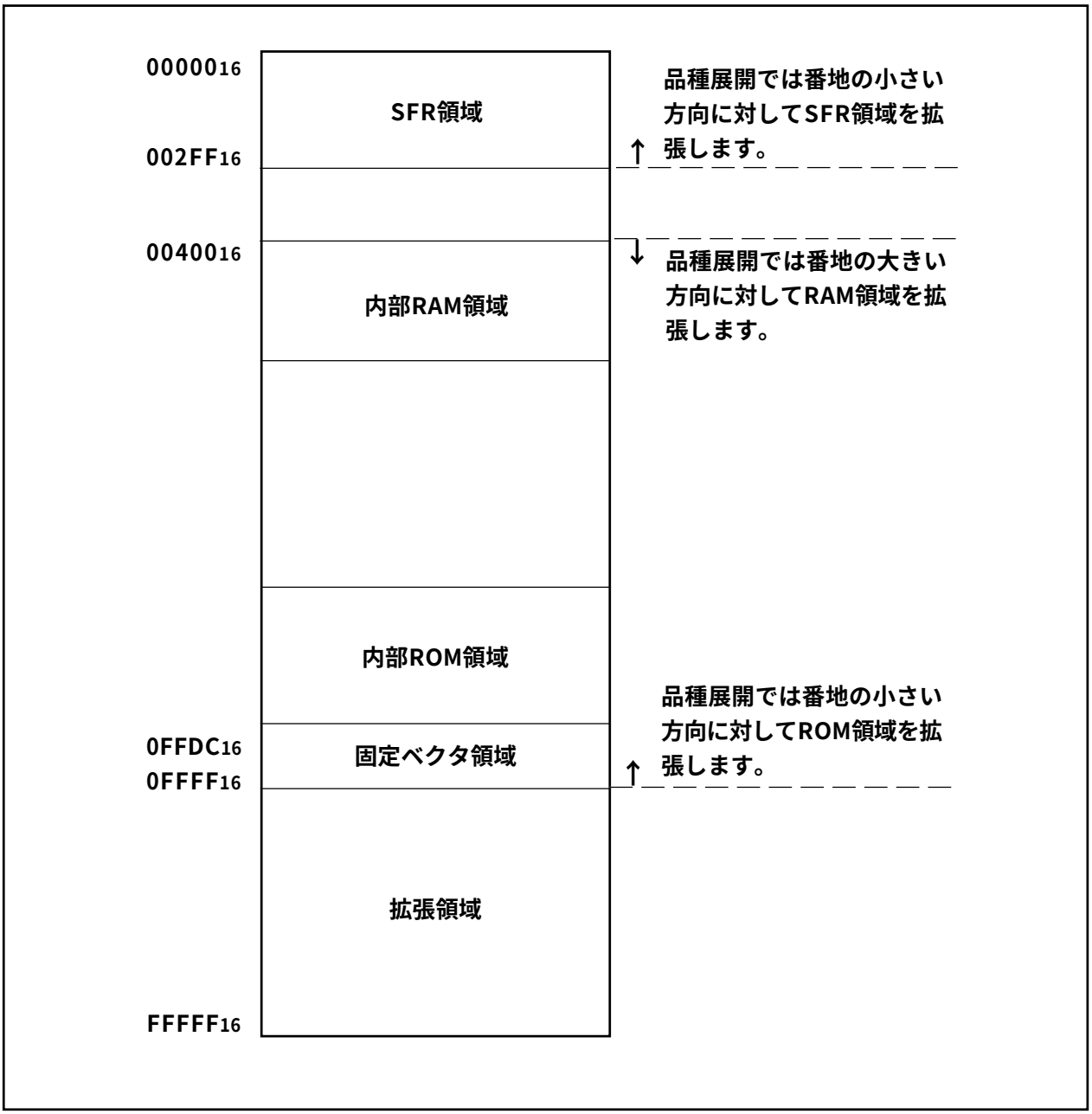


図 1.2.1 アドレス空間

1.3 レジスタ構成

中央演算処理装置には図1.3.1に示す13個のレジスタがあります。これらのうち、R0、R1、R2、R3、A0、A1、FBの7個は2セットあり、2つのレジスタバンクを構成しています。

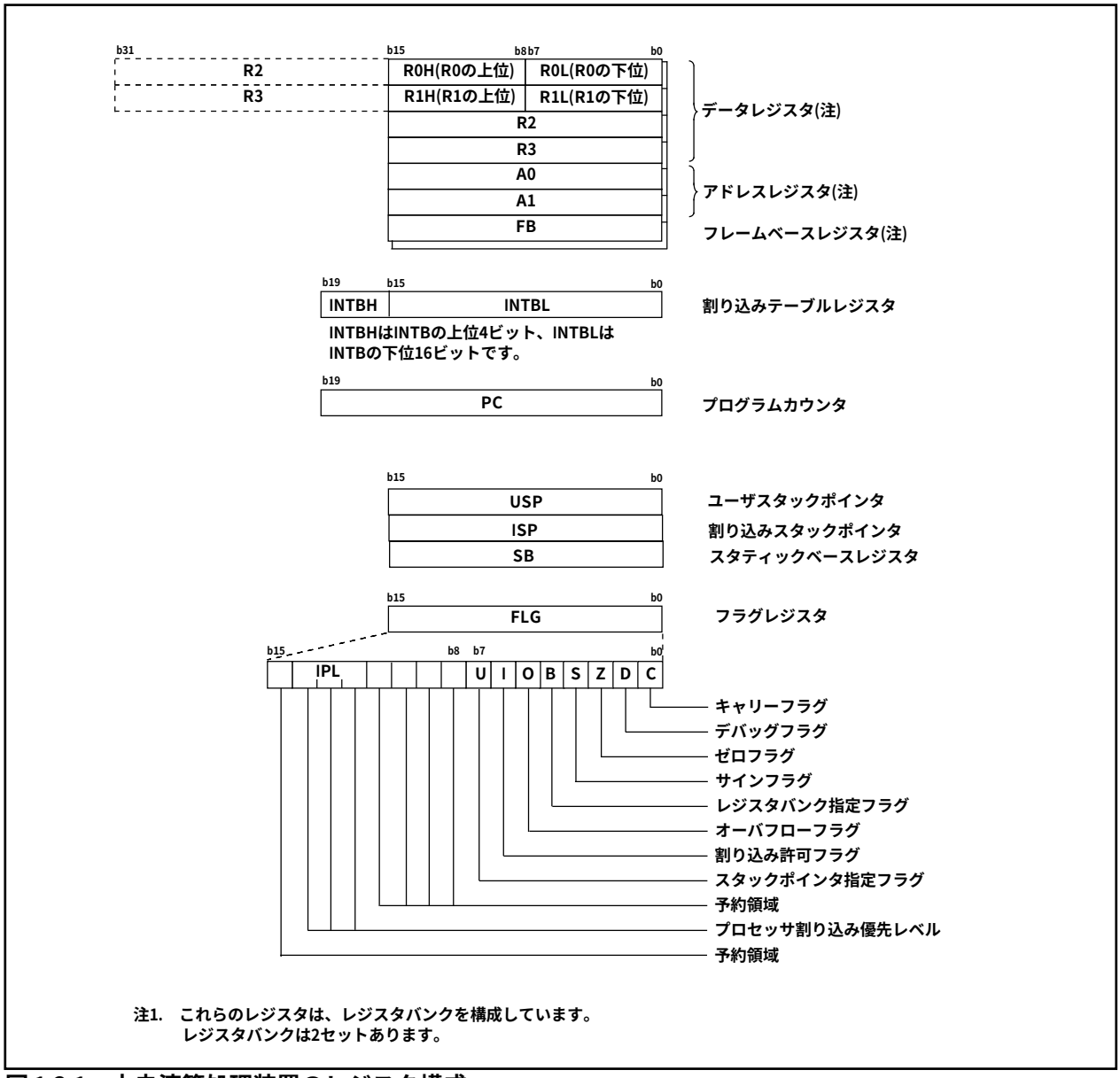


図 1.3.1 中央演算処理装置のレジスタ構成

1.3.1 データレジスタ（R0/R0H/R0L/R1/R1H/R1L/R2/R3）

データレジスタ（R0/R1/R2/R3）は16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。

R0/R1は、上位（R0H/R1H）と下位（R0L/R1L）を別々に8ビットのデータレジスタとして使用することもできます。また、一部の命令ではR2とR0、R3とR1を組合せて32ビットのデータレジスタ（R2R0/R3R1）としても使用できます。

1.3.2　アドレスレジスタ（A0/A1）

アドレスレジスタ（A0/A1）は16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。

一部の命令ではA1とA0とを組合せて32ビットのアドレスレジスタ（A1A0）としても使用できます。

1.3.3　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16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。

1.3.4　プログラムカウンタ（PC）

プログラムカウンタ（PC）は20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。

1.3.5　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20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。

1.3.6　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16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。

1.3.7　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16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。

1.3.8　フラグレジスタ（FLG）

フラグレジスタ（FLG）は11ビットで構成されており、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、「1.4　フラグレジスタ（FLG）」を参照してください。

1.4　フラグレジスタ（FLG）

フラグレジスタ（FLG）の構成を図1.4.1に示します。また、各フラグの機能を以下に示します。

1.4.1　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。

1.4.2　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。

このフラグが“１”のとき、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。

1.4.3　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。

1.4.4　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。

1.4.5　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。

1.4.6　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に“１”になります。

1.4.7　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。

1.4.8　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”のときユーザースタックポインタ（USP）が指定されます。

ハードウエア割り込みを受け付けたとき、またはソフトウエア割り込み番号0～31のINT命令を実行したとき、このフラグは“０”になります。

1.4.9　ビット８～ビット１１：予約領域

1.4.10　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～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優先レベルが、プロセッサ割り込み優先レベル（IPL）より大きい場合、その割り込みは許可されます。

1.4.11　ビット１５：予約領域

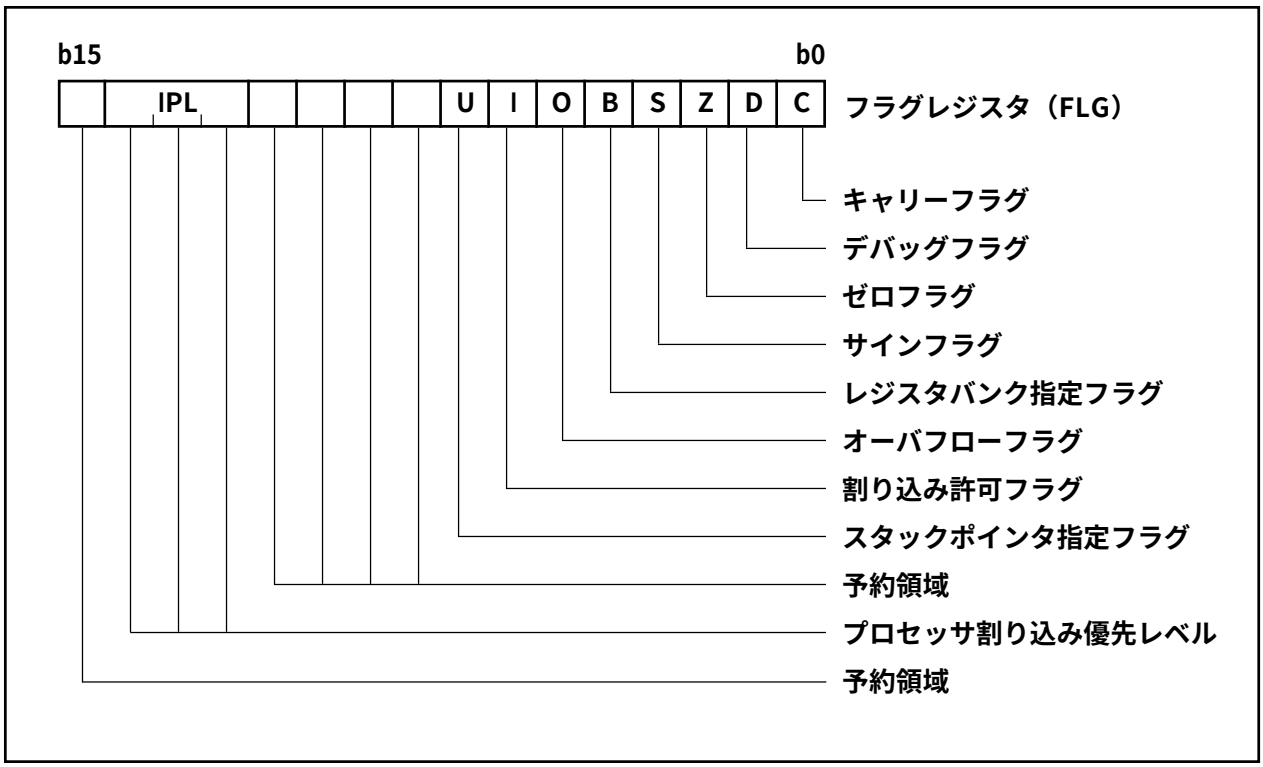


図 1.4.1　フラグレジスタ（FLG）の構成

1.5　レジスタバンク

データレジスタ（R0/R1/R2/R3）、アドレスレジスタ（A0/A1）、およびフレームベースレジスタ（FB）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）で切り替わります。

レジスタバンクの構成を図 1.5.1 に示します。

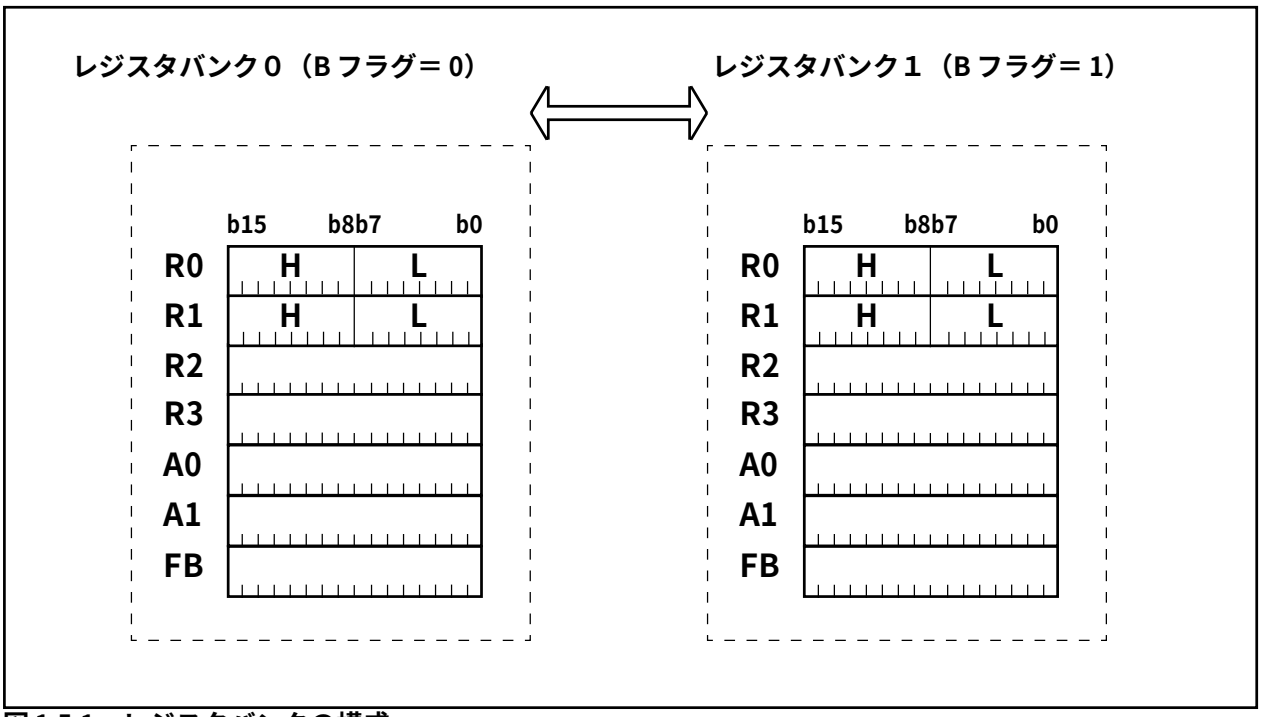


図 1.5.1　レジスタバンクの構成

1.6　リセット解除後の内部状態

リセット解除後の各レジスタの内容を以下に示します。

・データレジスタ（R0/R1/R2/R3）　：0000_{16}
・アドレスレジスタ（A0/A1）　：0000_{16}
・フレームベースレジスタ（FB）　：0000_{16}
・割り込みテーブルレジスタ（INTB）　：00000_{16}
・ユーザスタックポインタ（USP）　：0000_{16}
・割り込みスタックポインタ（ISP）　：0000_{16}
・スタティックベースレジスタ（SB）　：0000_{16}
・フラグレジスタ（FLG）　：0000_{16}

1.7 データタイプ

データタイプは整数、10 進、ビット、ストリングの 4 種類です。

1.7.1 整数

整数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。

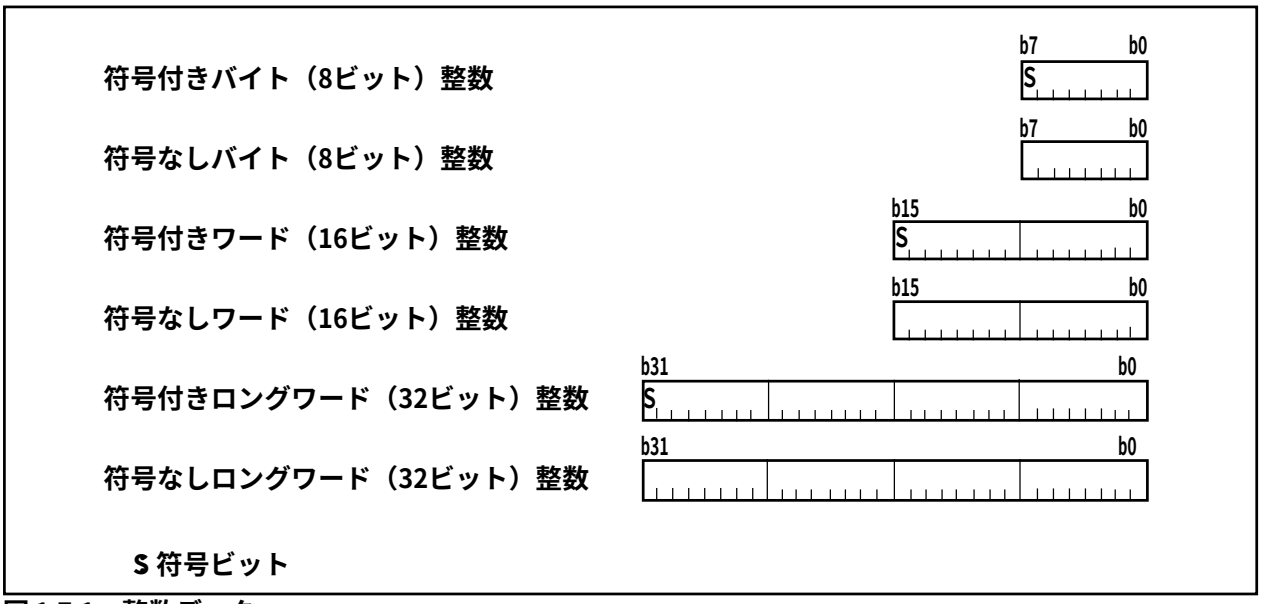


図 1.7.1 整数データ

1.7.2　10 進

10 進のデータタイプは、DADC、DADD、DSBB、DSUB の 4 種類で使用できます。

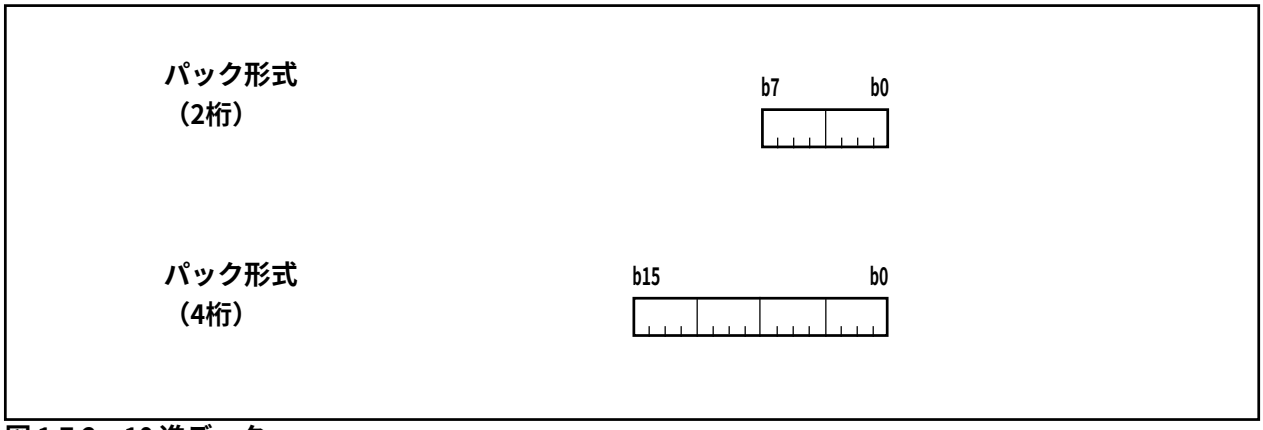


図 1.7.2　10 進データ

1.7.3 ビット

●レジスタのビット

レジスタのビット指定を図 1.7.3 に示します。

レジスタのビットはレジスタ直接（**bit,Rn**/**bit,An**）によって指定できます。**bit,Rn** は、データレジスタ（**Rn**）のビット指定に使用し、**bit,An** はアドレスレジスタ（**An**）のビット指定に使用します。

各レジスタのビットは、LSBからMSBにそれぞれ0～15のビット番号をもちます。したがって、**bit,Rn**/**bit,An** の **bit** は 0 ～ 15 の範囲で指定できます。

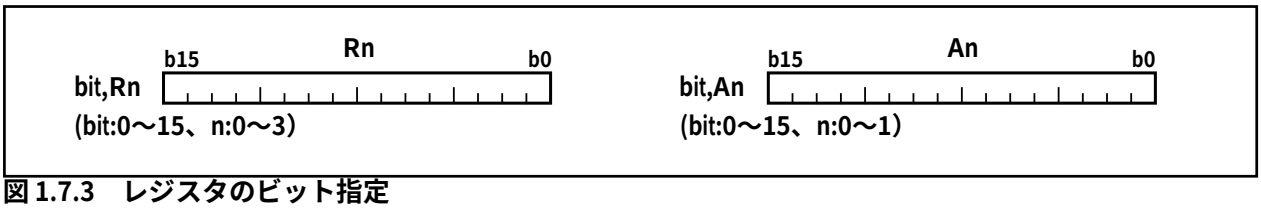


図 1.7.3 レジスタのビット指定

●メモリのビット

図1.7.4にメモリのビット指定に使用するアドレッシングを、表1.7.1に各アドレッシングでビットを指定できるアドレスを示します。メモリのビットは表 1.7.1 に示す範囲で指定してください。

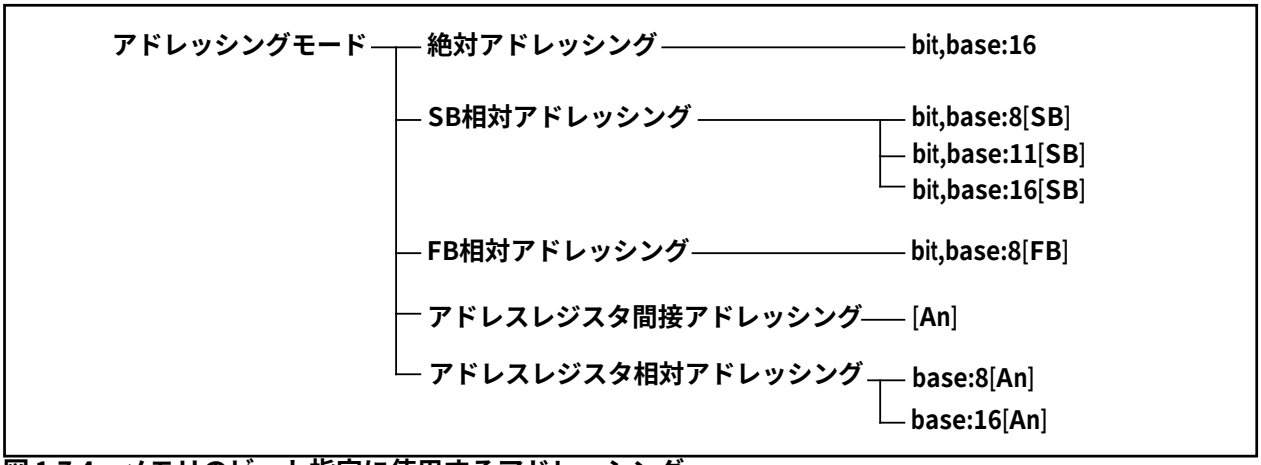


図 1.7.4 メモリのビット指定に使用するアドレッシング

表 1.7.1 各アドレッシングでビットを指定できるアドレス

アドレッシング	指定範囲		備考
	下限（番地）	上限（番地）	
bit,base:16	00000_{16}	$01FFF_{16}$	
bit,base:8[SB]	[SB]	[SB]+$0001F_{16}$	アクセス範囲は 00000_{16} ～ $0FFFF_{16}$ です。
bit,base:11[SB]	[SB]	[SB]+$000FF_{16}$	アクセス範囲は 00000_{16} ～ $0FFFF_{16}$ です。
bit,base:16[SB]	[SB]	[SB]+$01FFF_{16}$	アクセス範囲は 00000_{16} ～ $0FFFF_{16}$ です。
bit,base:8[FB]	[FB]－ 00010_{16}	[FB]+$0000F_{16}$	アクセス範囲は 00000_{16} ～ $0FFFF_{16}$ です。
[An]	00000_{16}	$01FFF_{16}$	
base:8[An]	base:8	base:8+$01FFF_{16}$	アクセス範囲は 00000_{16} ～ $020FE_{16}$ です。
base:16[An]	base:16	base:16+$01FFF_{16}$	アクセス範囲は 00000_{16} ～ $0FFFF_{16}$ です。

① bit,base によるビット指定

メモリマップとビットマップの関係を図 1.7.5 に示します。

メモリのビットは、連続したビットの配列として扱うことができます。ビットの指定は、任意の **bit** と **base** の組合せによって指定できます。**base** に設定したアドレスのビット 0 を基準（0）とし、対象となるビット位置を **bit** に設定します。0000A16 番地のビット 2 の指定例を図 1.7.6 に示します。

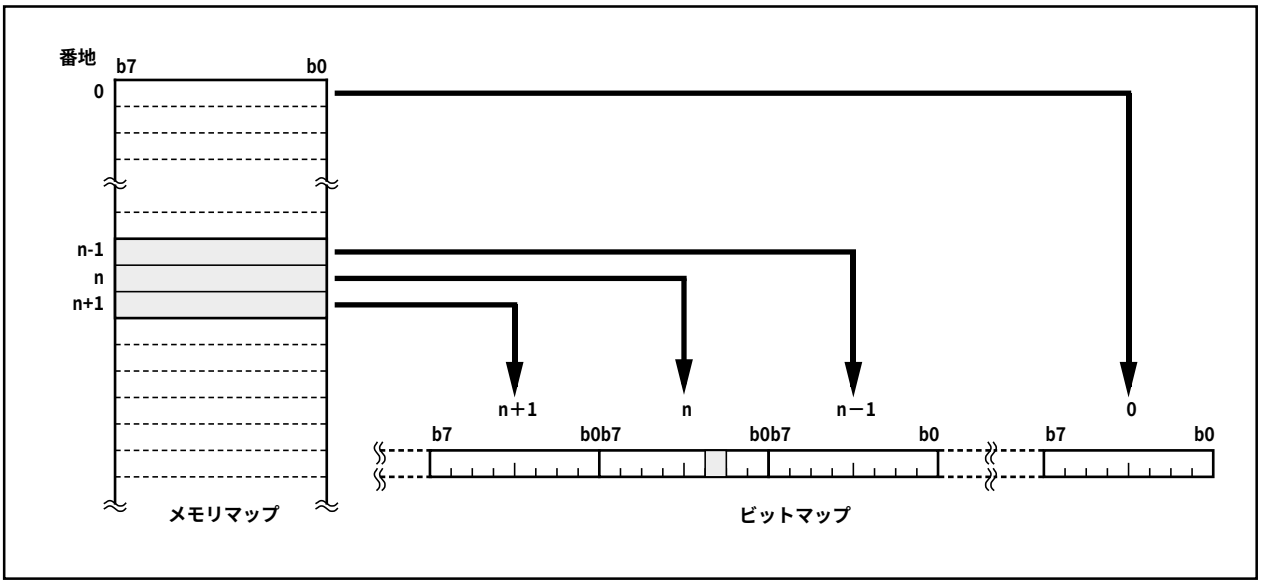


図 1.7.5　メモリマップとビットマップの関係

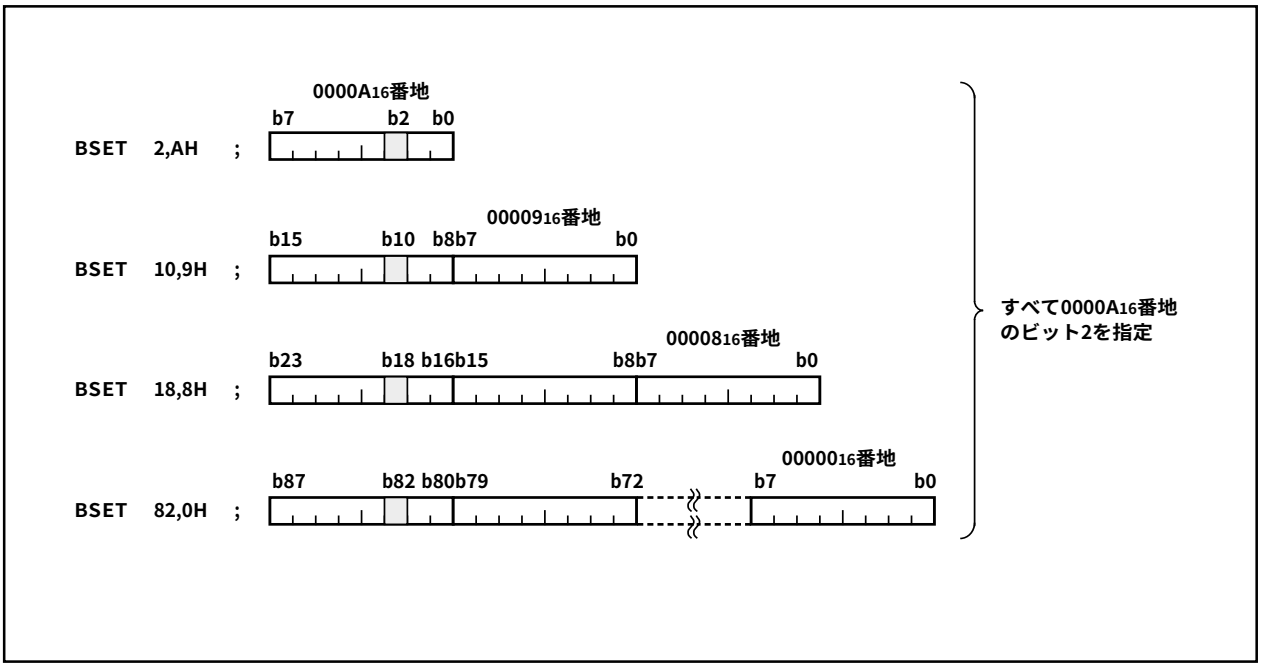


図 1.7.6　0000A16 番地のビット 2 の指定例

② SB/FB 相対によるビット指定

SB/**FB**相対アドレッシングの場合、スタティックベースレジスタ（**SB**）/フレームベースレジスタ（**FB**）に設定したアドレスと **base** に設定したアドレスを加算したアドレスのビット 0 を基準（0）とし、対象となるビット位置を **bit** に設定します。

③アドレスレジスタ間接 / アドレスレジスタ相対によるビット指定

アドレスレジスタ間接アドレッシングの場合、0000_{16} 番地のビット 0 を基準（0）とし、対象となるビット位置をアドレスレジスタ（**An**）に設定します。

アドレスレジスタ相対アドレッシングの場合、**base** に設定した番地のビット 0 を基準（0）とし、対象となるビット位置をアドレスレジスタ（**An**）に設定します。

1.7.4 ストリング

ストリングとはバイト(8ビット)、またはワード(16ビット)のデータを任意の長さだけ連続して並べたデータタイプです。

このデータタイプは、ストリング命令の文字列後方転送(SMOVB 命令)、文字列前方転送(SMOVF 命令)、指定した領域のイニシャライズ(SSTR 命令)の 3 種類で使用できます。

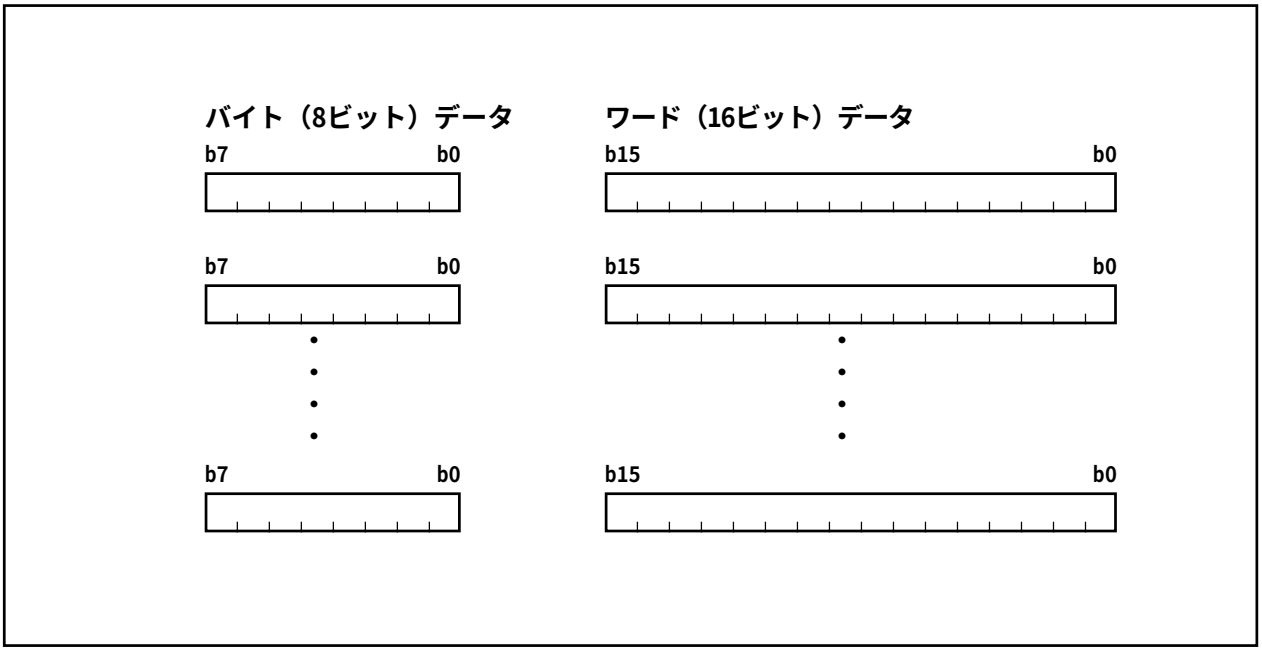


図 1.7.7 ストリングデータ

1.8 データ配置

1.8.1 レジスタのデータ配置

図 1.8.1 にレジスタのデータサイズとビット番号の関係を示します。

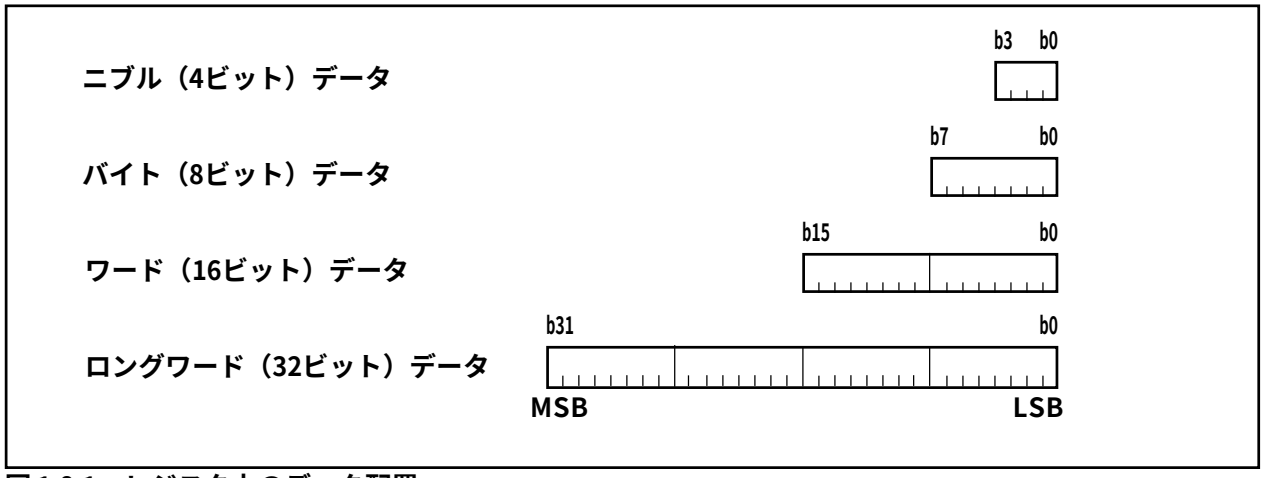


図 1.8.1 レジスタ上のデータ配置

1.8.2 メモリ上のデータ配置

図 1.8.2 にメモリ上のデータ配置を、図 1.8.3 に演算例を示します。

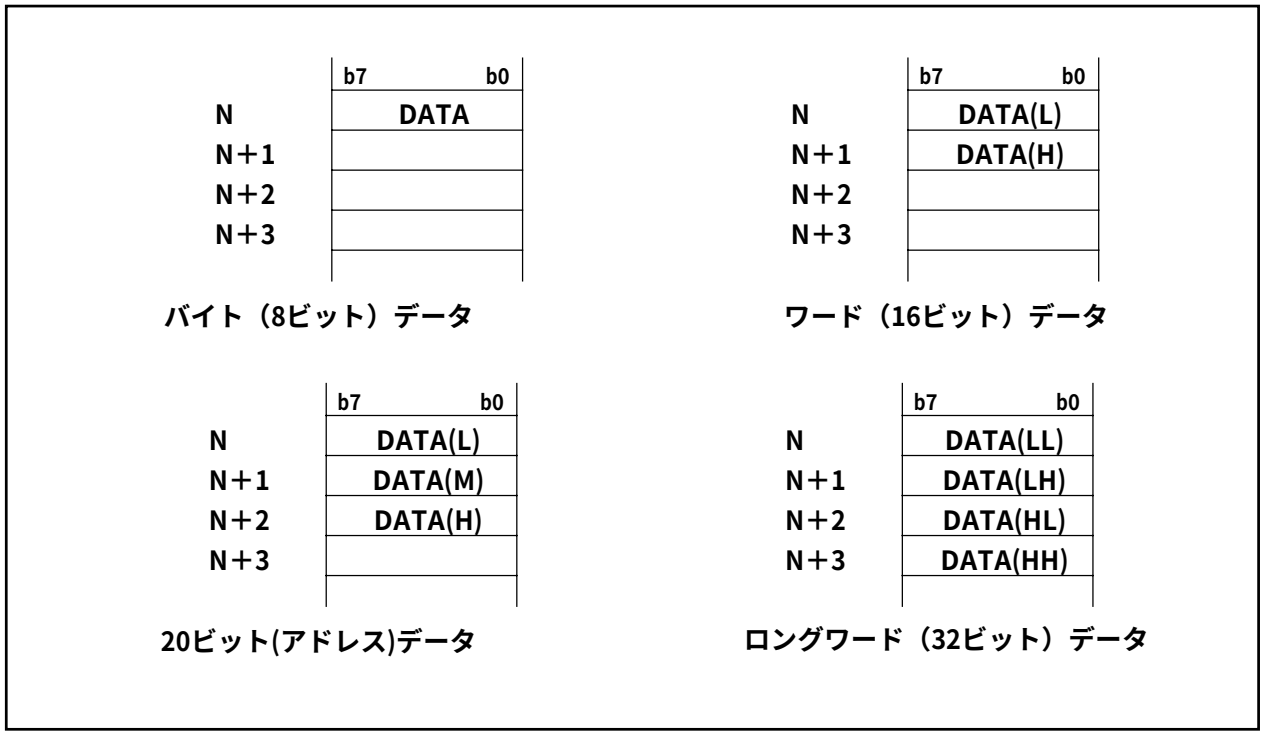


図 1.8.2 メモリ上のデータ配置

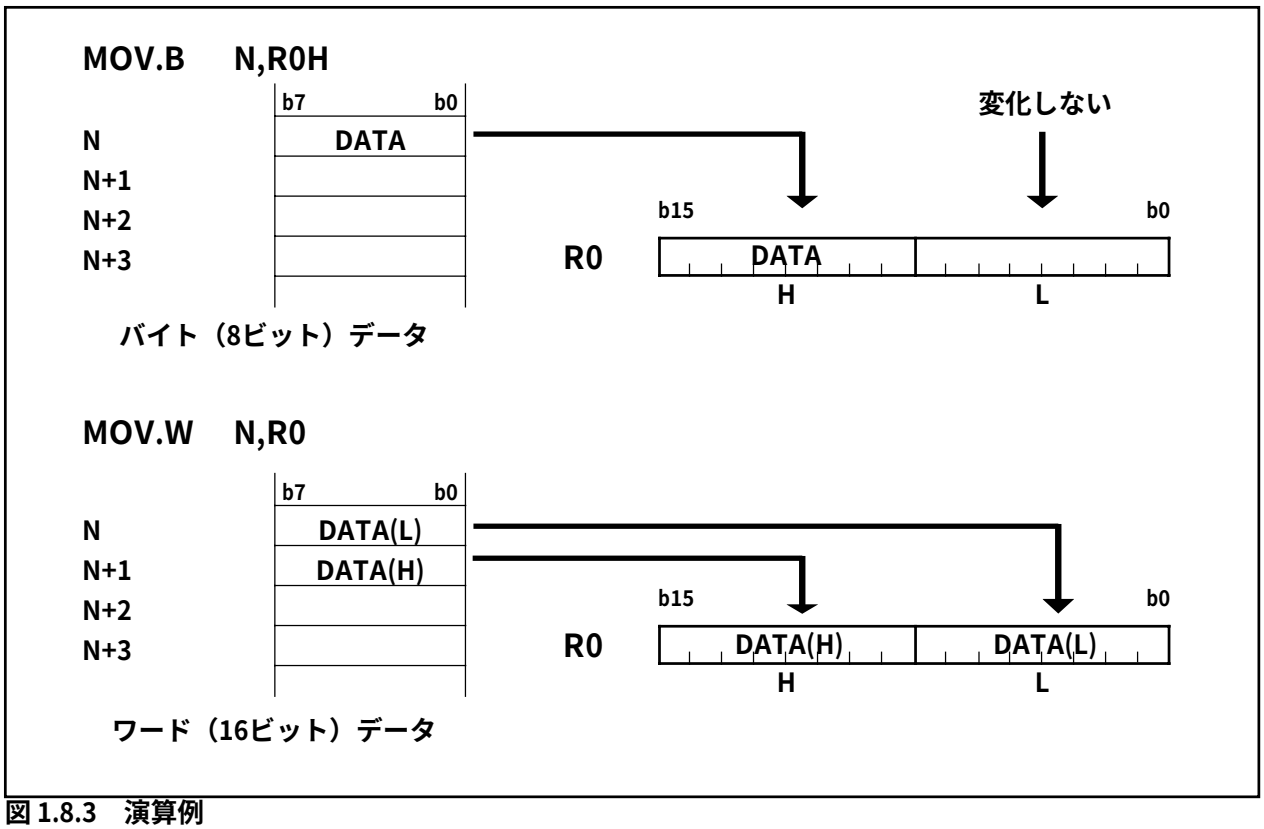


図 1.8.3 演算例

1.9 命令フォーマット

命令のフォーマットは、ジェネリック、クイック、ショート、ゼロの4つに分類できます。フォーマットが選択できる命令のバイト数は、ゼロ形式が最も少なく、ショート、クイック、ジェネリックの順に多くなります。

フォーマットの特長を以下に示します。

1.9.1 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バイトです。このオペコードには、動作および src＊1と dest＊2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バイト)と src コード(0～3バイト)および dest コード(0～3バイト)で構成されます。

1.9.2 クイック形式(:Q)

クイック形式のオペコードは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－7～+8または－8～+7（命令によって異なります）で表現できる数値です。

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

1.9.3 ショート形式(:S)

ショート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 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バイト)と src コード(0～2バイト)および dest コード(0～2バイト)で構成されます。

1.9.4 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 sourceの略称
*2 destinationの略称

1.10　ベクタテーブル

ベクタテーブルとして、割り込みベクタテーブルがあります。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。

1.10.1　固定ベクタテーブル

固定ベクタテーブルは、アドレスが固定のベクタテーブルです。0FFDC16 番地から 0FFFF16 番地に割り込みベクタテーブルの一部を配置しています。図 1.10.1 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 ベクタテーブルに対して 4 バイトで構成されています。各ベクタテーブルには、割り込みルーチンの先頭アドレスを設定します。

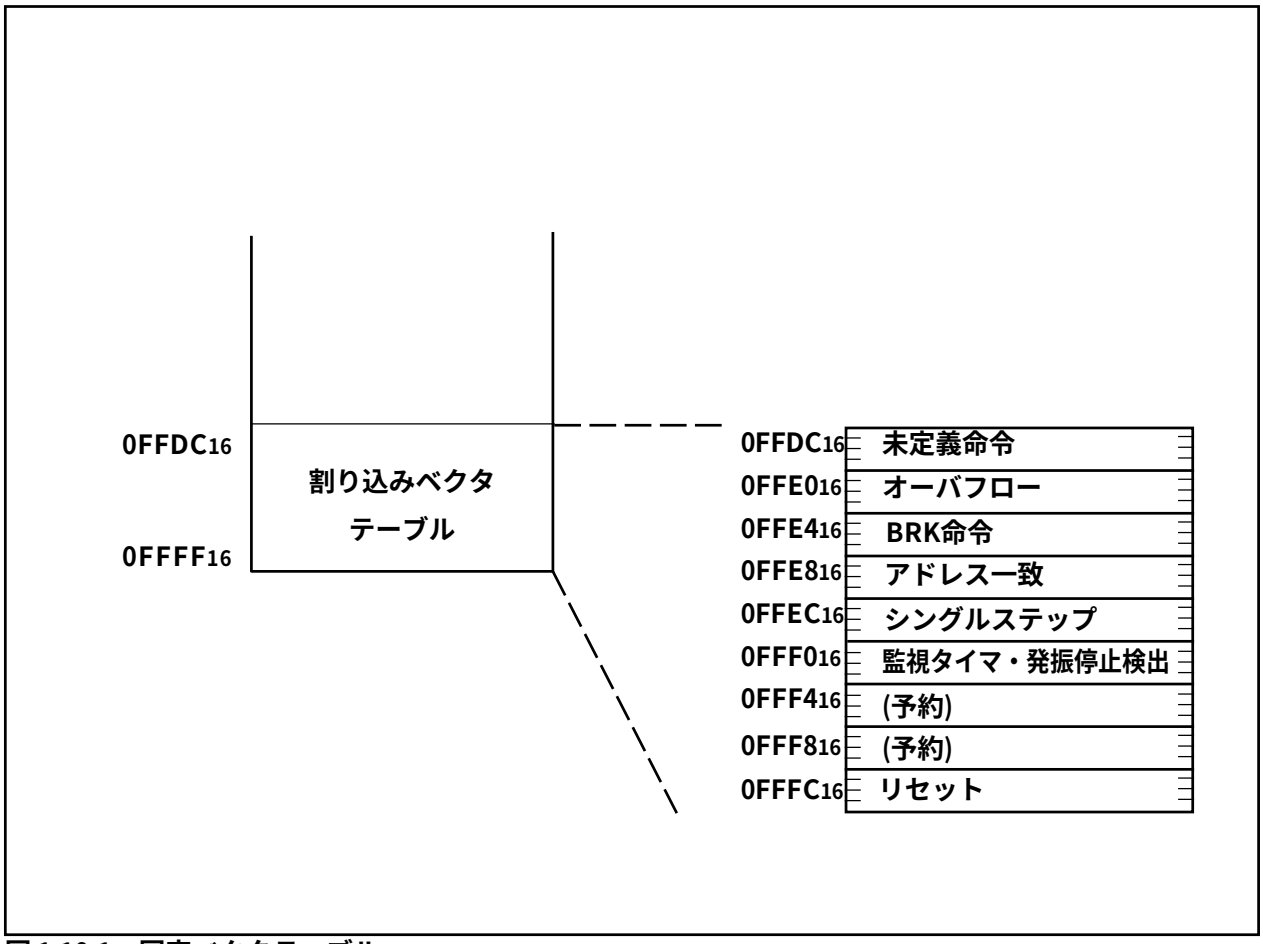


図 1.10.1　固定ベクタテーブル

1.10.2　可変ベクタテーブル

可変ベクタテーブルは、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256バイトの割り込みベクタテーブルです。図1.10.2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ベクタテーブルに対して4バイトで構成されています。各ベクタテーブルには、割り込みルーチンの先頭アドレスを設定します。

また、1ベクタテーブルごとに、ソフトウエア割り込み番号（0～63）があり、INT命令では、このソフトウエア割り込み番号を使用します。

品種展開によって内蔵される周辺機能の割り込みは、ソフトウエア割り込み番号0～31に割り当てられます。

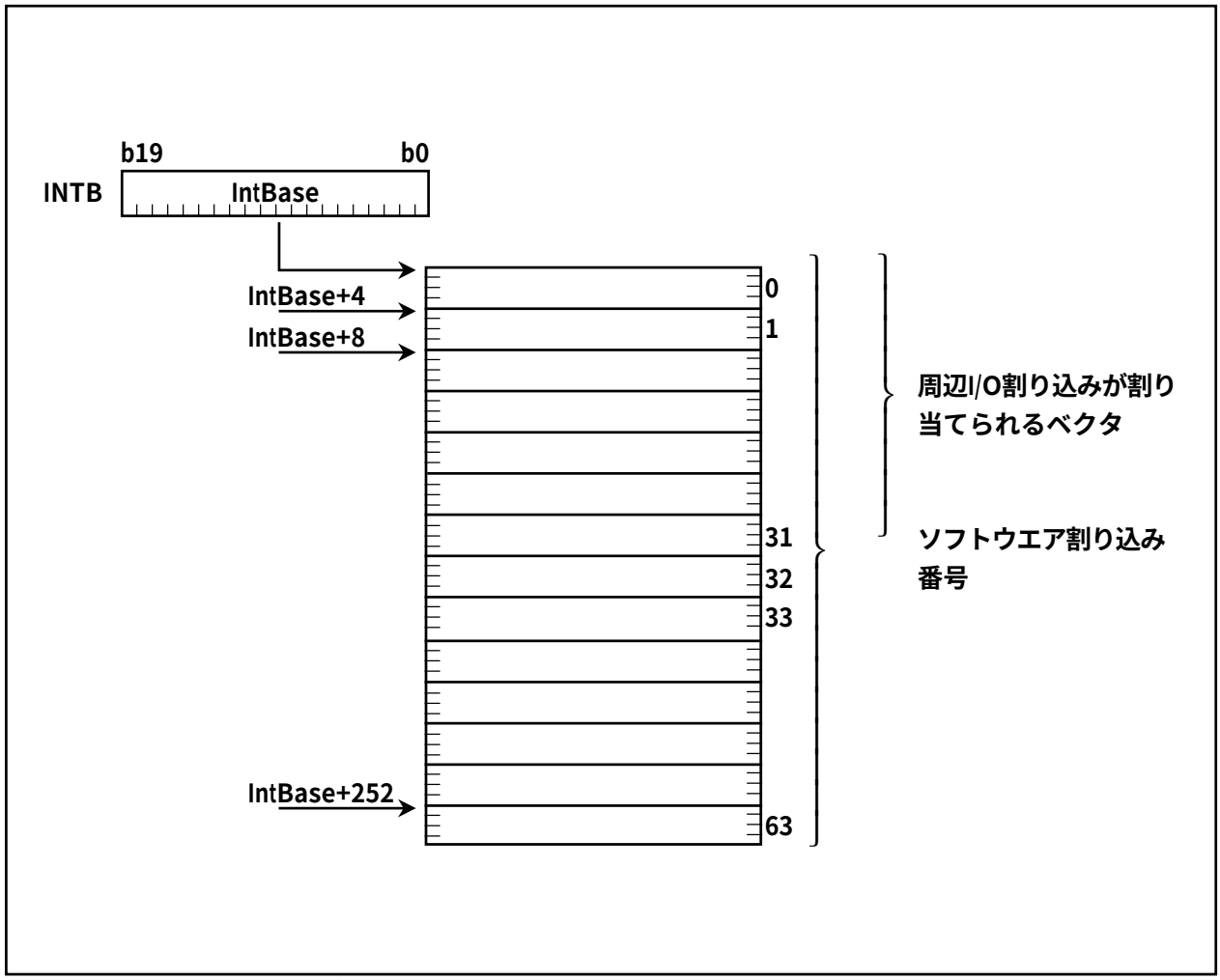


図1.10.2　可変ベクタテーブル

第2章

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３つのタイプがあります。

2.1.1 一般命令アドレッシング

0000016 番地から 0FFFF16 番地までの領域をアクセスするアドレッシングです。
以下に一般命令アドレッシングの各名称を示します。

・即値
・レジスタ直接
・絶対
・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
・スタックポインタ相対

2.1.2 特定命令アドレッシング

0000016 番地から FFFFF16 番地までの領域をアクセスするアドレッシング、および専用レジスタをアクセスするアドレッシングです。
以下に特定命令アドレッシングの各名称を示します。

・20 ビット絶対
・20 ビットディスプレースメント付きアドレスレジスタ相対
・32 ビットアドレスレジスタ間接
・32 ビットレジスタ直接
・専用レジスタ直接
・プログラムカウンタ相対

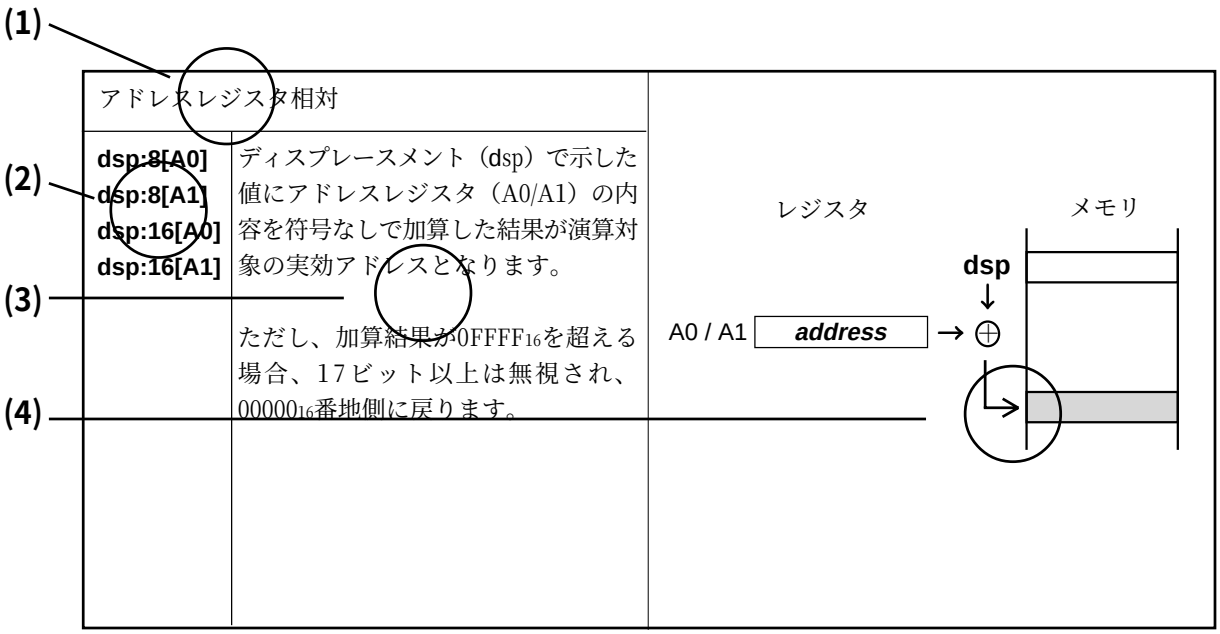
2.1.3 ビット命令アドレッシング

0000016 番地から 0FFFF16 番地までの領域をビット単位でアクセスするアドレッシングです。
以下にビット命令アドレッシングの各名称を示します。

・レジスタ直接
・絶対
・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
・FLG 直接

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)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。

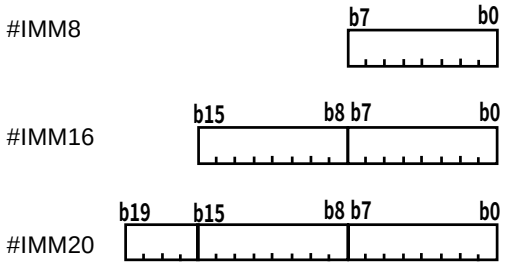
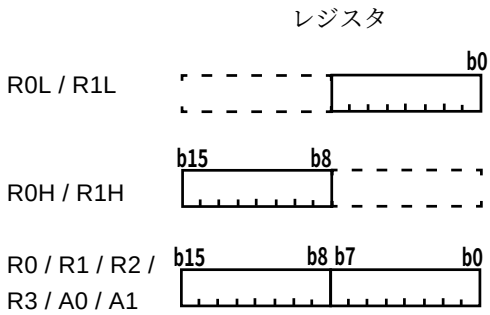
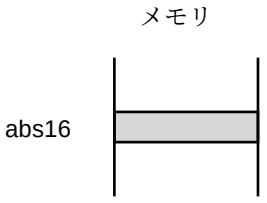
(3)解説

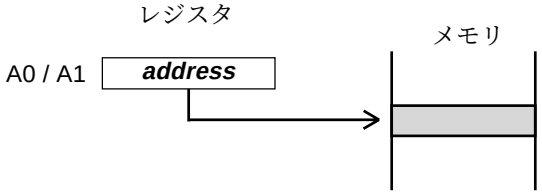
動作、実効アドレスの範囲を説明します。

(4)動作図

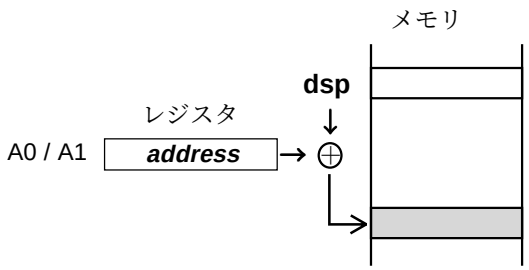
動作を図で説明します。

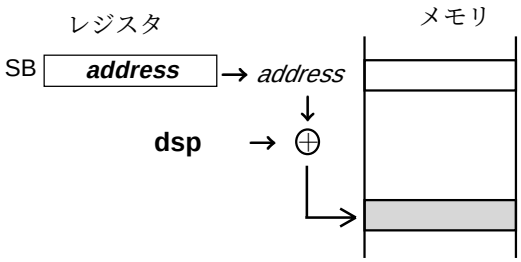
2.3 一般命令アドレッシング

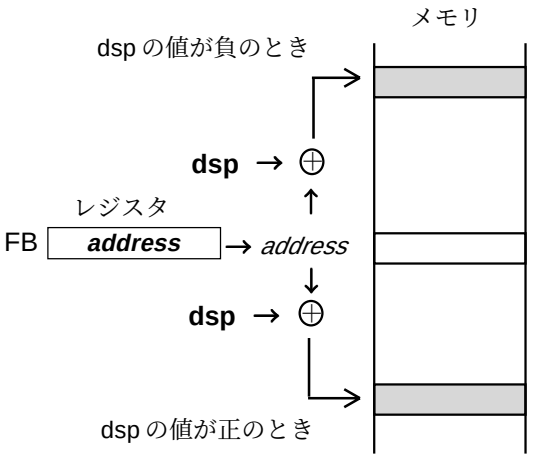
<table>
<tr><td colspan="2">即値</td><td rowspan="2"> 
</td></tr>
<tr><td>#IMM
#IMM8
#IMM16
#IMM20</td><td>#IMMで示した即値が演算の対象となります。</td></tr>
<tr><td colspan="2">レジスタ直接</td><td rowspan="2"> 
</td></tr>
<tr><td>R0L
R0H
R1L
R1H
R0
R1
R2
R3
A0
A1</td><td>指定したレジスタが演算の対象となります。</td></tr>
<tr><td colspan="2">絶対</td><td rowspan="2"> 
</td></tr>
<tr><td>abs16</td><td>abs16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、0000016～0FFFF16です。</td></tr>
<tr><td colspan="2">アドレスレジスタ間接</td><td rowspan="2"> 
</td></tr>
<tr><td>[A0]
[A1]</td><td>アドレスレジスタ（A0/A1）の内容で示した値が演算対象の実効アドレスとなります。

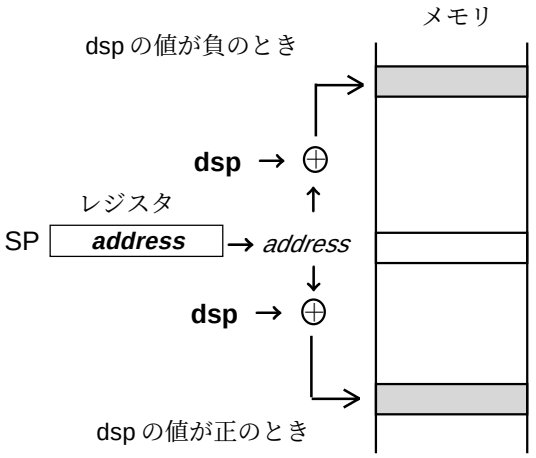
実効アドレスの範囲は、0000016～0FFFF16です。</td></tr>
</table>

<table>
<tr><td colspan="2">アドレスレジスタ相対</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:8[A0]
dsp:8[A1]
dsp:16[A0]
dsp:16[A1]</td><td>ディスプレースメント（dsp）で示した値にアドレスレジスタ（A0/A1）の内容を符号なしで加算した結果が演算対象の実効アドレスとなります。

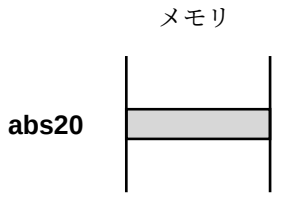
ただし、加算結果が 0FFFF16 を超える場合、17 ビット以上は無視され、0000016 番地側に戻ります。</td></tr>
<tr><td colspan="2">SB 相対</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:8[SB]
dsp:16[SB]</td><td>スタティックベースレジスタ（SB）の内容で示したアドレスにディスプレースメント（dsp）で示した値を符号なしで加算した結果が演算対象の実効アドレスとなります。

ただし、加算結果が 0FFFF16 を超える場合、17 ビット以上は無視され、0000016 番地側に戻ります。</td></tr>
<tr><td colspan="2">FB 相対</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:8[FB]</td><td>フレームベースレジスタ（FB）の内容で示したアドレスにディスプレースメント（dsp）で示した値を符号付きで加算した結果が演算対象の実効アドレスとなります。

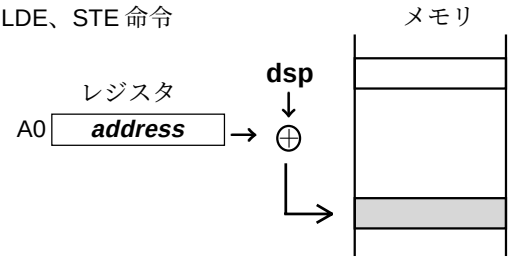
ただし、加算結果が 0000016 ～ 0FFFF16 を超える場合、17 ビット以上は無視され、0000016 番地側、または 0FFFF16 番地側に戻ります。</td></tr>
</table>

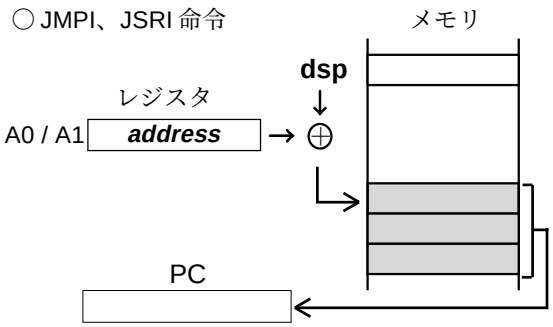
スタックポインタ相対		
dsp:8[SP]	スタックポインタ（SP）の内容で示したアドレスにディスプレースメント（dsp）で示した値を符号付きで加算した結果が演算対象の実効アドレスとなります。スタックポインタ（SP）は、Uフラグで示すスタックポインタが対象となります。 ただし、加算結果が 00000$_{16}$ ～ 0FFFF$_{16}$ を超える場合、17 ビット以上は無視され、00000$_{16}$ 番地側または 0FFFF$_{16}$ 番地側に戻ります。 このアドレッシングはMOV命令で使用できます。	

2.4 特定命令アドレッシング

<table>
<tr><td colspan="2">20 ビット絶対</td><td rowspan="2"> 
</td></tr>
<tr><td>abs20</td><td>abs20 で示した値が演算対象の実効アドレスとなります。

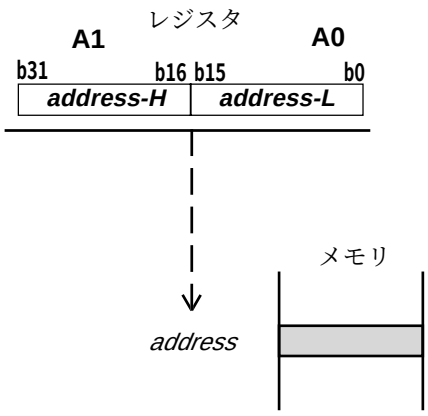
実効アドレスの範囲は、0000016 ～ FFFFF16 です。

このアドレッシングは LDE、STE、JSR、JMP 命令で使用できます。</td></tr>
<tr><td colspan="2">20 ビットディスプレースメント付き
アドレスレジスタ相対</td><td rowspan="2">○ LDE、STE 命令
 

○ JMPI、JSRI 命令
 
</td></tr>
<tr><td>dsp:20[A0]
dsp:20[A1]</td><td>ディスプレースメント(dsp)で示したアドレスにアドレスレジスタ（A0/A1）の内容を符号なしで加算した結果が演算対象の実効アドレスとなります。

ただし、加算結果がFFFFF16を超える場合、21 ビット以上は無視され、0000016 番地側に戻ります。

このアドレッシングは LDE、STE、JMPI、JSRI 命令で使用できます。

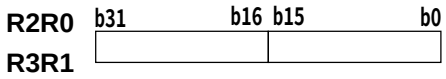
使用できるアドレッシングモードと命令の組み合わせを以下に示します。
dsp:20[A0]
→ LDE、STE、JMPI、JSRI 命令
dsp:20[A1]
→ JMPI、JSRI 命令</td></tr>
<tr><td colspan="2">32 ビットアドレスレジスタ間接</td><td rowspan="2"> 
</td></tr>
<tr><td>[A1A0]</td><td>アドレスレジスタ（A0/A1）を連結した 32 ビットで示したアドレスが演算対象の実効アドレスとなります。

ただし、連結したレジスタの値が FFFFF16を超える場合、21 ビット以上は無視されます。

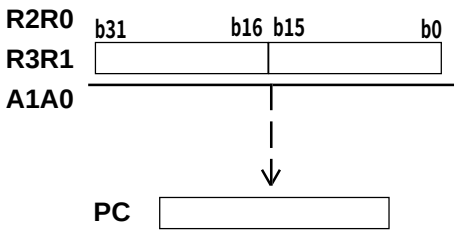
このアドレッシングは LDE、STE 命令で使用できます。</td></tr>
</table>

32ビットレジスタ直接	
R2R0 **R3R1** **A1A0**	指定した2つのレジスタを連結した、32ビットのレジスタが演算の対象となります。 このアドレッシングはSHL、SHA、JMPI、JSRI命令で使用できます。 使用できるレジスタと命令の組み合わせを以下に示します。 R2R0、R3R1 →SHL、SHA、JMPI、JSRI命令 A1A0 →JMPI、JSRI命令

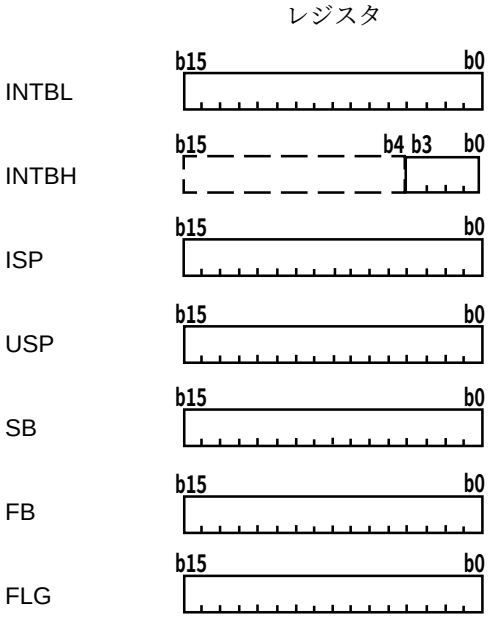
○SHL、SHA命令



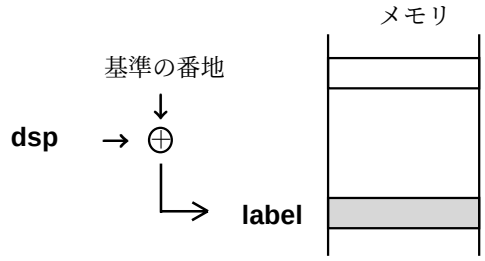
○JMPI、JSRI命令



専用レジスタ直接	
INTBL **INTBH** **ISP** **SP** **SB** **FB** **FLG**	指定した専用レジスタが演算の対象となります。 このアドレッシングは LDC、STC、PUSHC、POPC命令で使用できます。 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。

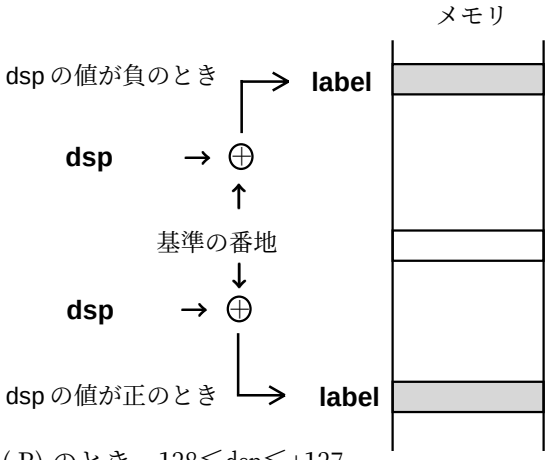


<table>
<tr><td colspan="2">プログラムカウンタ相対</td><td rowspan="2"></td></tr>
<tr><td>label</td><td>○分岐距離指定子（.length）が（.S）の場合
基準の番地にディスプレースメント（dsp）で示した値を符号なしで加算した結果が実効アドレスとなります。

このアドレッシングは、JMP 命令で使用できます。</td><td> 

+0≦dsp≦+7
*1　基準の番地は(命令の先頭番地 +2)です。</td></tr>
<tr><td></td><td>○分岐距離指定子（.length）が（.B）または（.W）の場合
基準の番地にディスプレースメント（dsp）で示した値を符号付きで加算した結果が実効アドレスとなります。

ただし、加算結果が 00000₁₆ ～ FFFFF₁₆ を超える場合、21 ビット以上は無視され、00000₁₆ 番地または FFFFF₁₆ 番地側に戻ります。

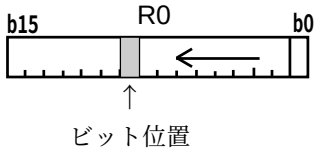
このアドレッシングは、JMP、JSR 命令で使用できます。</td><td> 

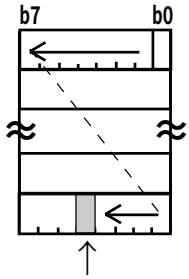
(.B) のとき－128≦dsp≦+127
(.W) のとき－32768≦dsp≦+32767
*2　基準の番地は命令によって異なります。</td></tr>
</table>

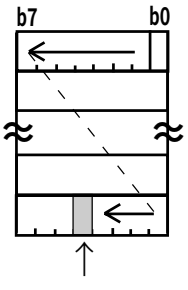
2.5 ビット命令アドレッシング

このアドレッシングは以下の命令で使用できます。

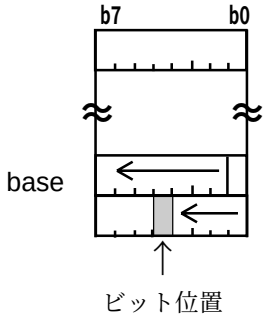
BCLR、BSET、BNOT、BTST、BNTST、BAND、BNAND、BOR、BNOR、BXOR、BNXOR、BM*Cnd*、BTSTS、BTSTC

<table>
<tr><td colspan="2">レジスタ直接</td><td rowspan="2">bit , R0
 
</td></tr>
<tr><td>bit,R0
bit,R1
bit,R2
bit,R3
bit,A0
bit,A1</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。

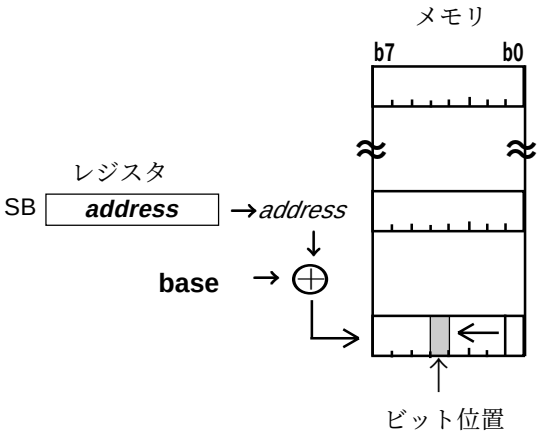
ビット位置（bit）は 0～15 が指定できます。</td></tr>
<tr><td colspan="2">絶対</td><td rowspan="2">base
 
</td></tr>
<tr><td>bit,base:16</td><td>base で示したアドレスのビット 0 から、bit で示したビット数だけ離れたビットが演算の対象となります。

00000₁₆番地～01FFF₁₆番地のビットが対象となります。</td></tr>
<tr><td colspan="2">アドレスレジスタ間接</td><td rowspan="2">00000₁₆
 
</td></tr>
<tr><td>[A0]
[A1]</td><td>00000₁₆番地のビット0から、アドレスレジスタ（A0/A1）で示したビット数だけ離れたビットが演算の対象となります。

00000₁₆番地～01FFF₁₆番地のビットが対象となります。</td></tr>
</table>

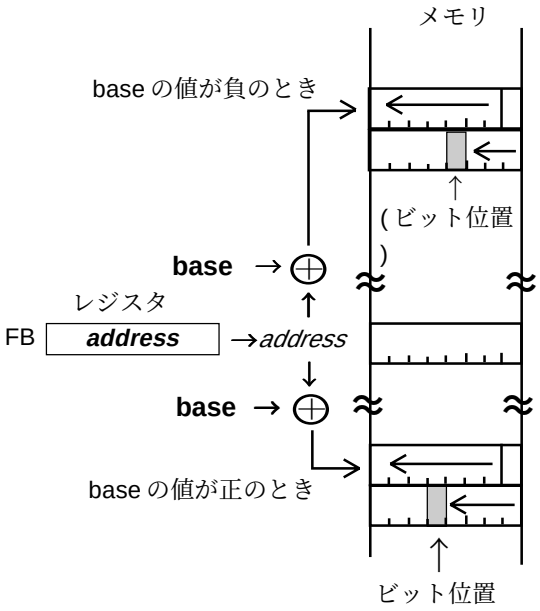
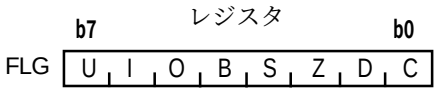
<table>
<tr><td colspan="2">アドレスレジスタ相対</td><td rowspan="2"> 
</td></tr>
<tr><td>base:8[A0]
base:8[A1]
base:16[A0]
base:16[A1]</td><td>baseで示したアドレスのビット0から、アドレスレジスタ（A0/A1）で示したビット数だけ離れたビットが演算の対象になります。

ただし、対象となるビットのアドレスが0FFFF16を超える場合、17ビット以上は無視され、000016番地側に戻ります。

アドレスレジスタ（A0/A1）で指定できるアドレスの範囲はbaseから8192バイトです。</td></tr>
<tr><td colspan="2">SB相対</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,base:8[SB]
bit,base:11[SB]
bit,base:16[SB]</td><td>スタティックベースレジスタ（SB）の内容で示したアドレスにbaseで示した値を符号なしで加算したアドレスのビット0から、bitで示したビット数だけ離れたビットが演算の対象となります。

ただし、対象となるビットのアドレスが0FFFF16を超える場合、17ビット以上は無視され、000016番地側に戻ります。

bit,base:8、bit,base:11、bit,base:16で指定できるアドレスの範囲はスタティックベースレジスタ（SB）の値からそれぞれ32バイト、256バイト、8192バイトです。</td></tr>
</table>

FB 相対		
bit,base:8[FB]	フレームベースレジスタ（FB）の内容で示した値にbaseで示した値を符号付きで加算したアドレスのビット0から、bitで示したビット数だけ離れたビットが演算の対象となります。 ただし、対象となるビットのアドレスが0000016～0FFFF16を超える場合、17ビット以上は無視され、0000016番地側、または0FFFF16番地側に戻ります。 bit,base:8で指定できるアドレスの範囲はフレームベースレジスタ(FB)の値を中心に32バイトです。	
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。	

第３章

機能

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、関連命令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。

(1)ニーモニック

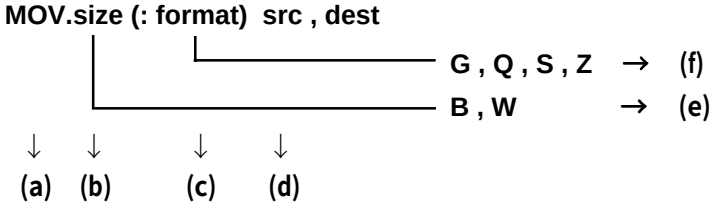
本ページで説明するニーモニックを示しています。

(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。(:format) を省略した場合、アセンブラが最適な指定子を選択します。



(a)　ニーモニック　MOV

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。(:format) を省略した場合、アセンブラが最適な指定子を選択します。(:format) を記述した場合、その内容が優先されます。

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)　(b)で指定できるデータサイズを示しています。

(f)　(c)で指定できる命令フォーマットを示しています。

(1) **MOV**　　*転送*　　**MOV**

(2) **MOVe**

【構文】　　【命令コード／サイクル数】

Page=193

(3) **MOV.size (:format) src,dest**

G , Q , Z , S （指定可能）

B , W

(4) **【オペレーション】**

dest ← src

(5) **【機能】**

・srcをdestに転送します。

・サイズ指定子(.size)に(.B)を指定した場合、destがアドレスレジスタのとき、srcをゼロ拡張し16ビットで転送します。また、srcがアドレスレジスタのとき、アドレスレジスタの下位8ビットを転送します。

(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				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	dsp:8[SP]	R2R0	R3R1	A1A0	dsp:8[SP]

(7) **【フラグ変化】**

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

条件

S ： 転送の結果、destのMSBが“1”のとき“1”それ以外のとき“0”になります。

Z ： 転送の結果が0のとき“1”それ以外のとき“0”になります。

(8) **【記述例】**

```
MOV.B:S     #0ABH,R0L
MOV.W       #-1,R2
```

(9) **【関連命令】**　LDE,STE,XCHG

(4)オペレーション

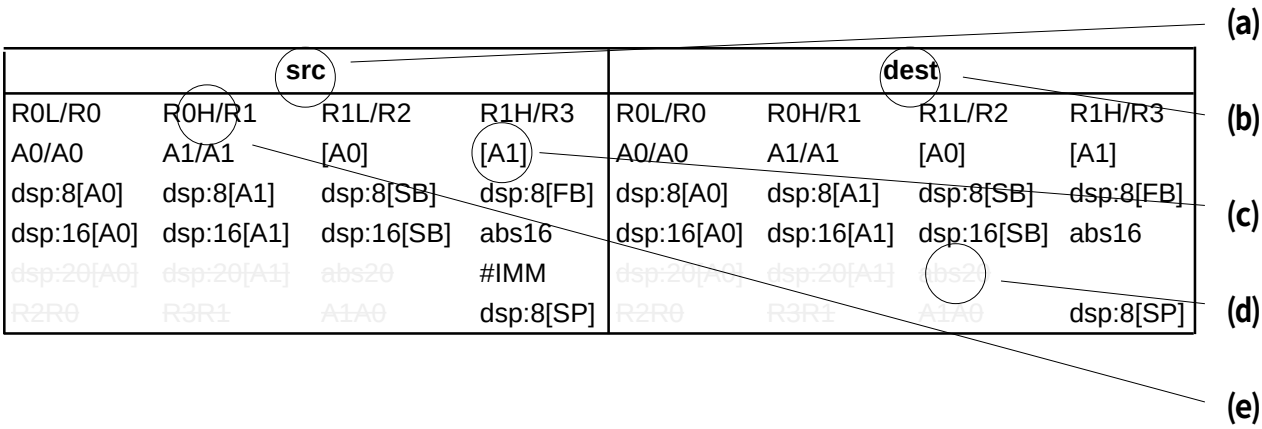
命令のオペレーションを記号で説明しています。

(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形式を示しています。



src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	dsp:8[SP]	R2R0	R3R1	A1A0	dsp:8[SP]

(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) 選択できるアドレッシング

(d) 選択できないアドレッシング

(e) スラッシュの左側（R0H）は取り扱うデータサイズがバイト（8ビット）の場合のアドレッシング
スラッシュの右側（R1）は取り扱うデータサイズがワード（16ビット）の場合のアドレッシング

(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。

“○”条件に従って変化します。

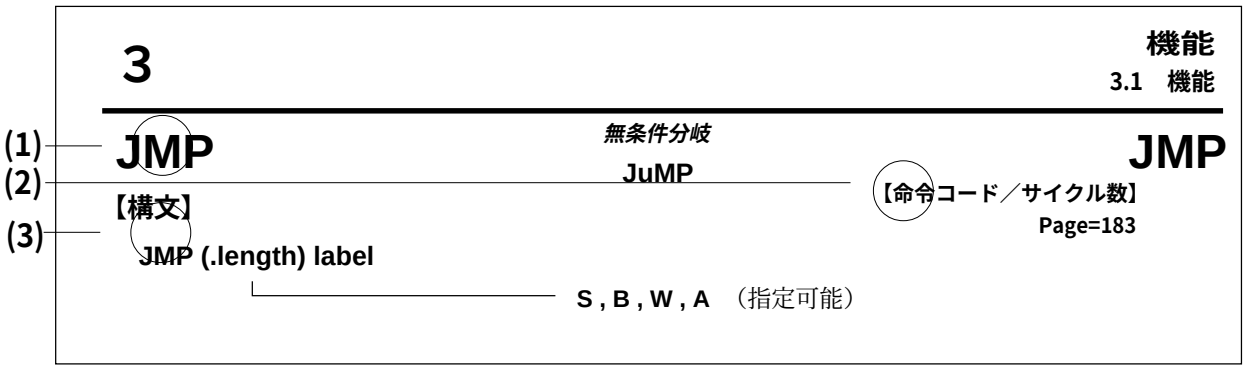
(8)記述例

命令の記述例を示しています。

(9)関連命令

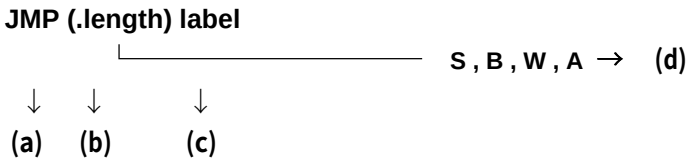
この命令に類似の動作や反対の動作をする命令など、関連する命令を示しています。

JMP、JPMI、JSR、JSRI 各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。



(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(.length)を省略した場合、アセンブラが最適な指定子を選択します。(.length)を記述した場合、その内容が優先されます。

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（+2 ～ +9）
.B　8 ビット PC 相対
.W　16 ビット PC 相対
.A　20 ビット絶対

(c)　オペランド　label

オペランドを記述します。

(d)　(b)で指定できる分岐距離を示しています。

ABS

絶対値
ABSolute

ABS

【構文】

ABS.size dest

B , W

【命令コード／サイクル数】

Page= 138

【オペレーション】

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			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	○	－	○	○	－	○

条件

O ： 演算前のdestが－128(.B)または－32768(.W)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   A0
```

ADC

キャリー付き加算
ADdition with Carry

ADC

【構文】

ADC.size src,dest

B , W

【命令コード／サイクル数】

Page= 138

【オペレーション】

dest ← src + dest + C

【機能】

・destとsrcとＣフラグを加算し、destに格納します。
・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	○	－	○	○	－	○

条件

O ： 符号付き演算の結果、+32767(.W)または－32768(.W)、+127(.B)または－128(.B)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、+65535(.W)、+255(.B)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   A0,R0
ADC.B   A0,R0L          ;A0の下位8ビットとR0Lを演算します。
ADC.B   R0L,A0          ;R0Lをゼロ拡張してA0と演算します。
```

【関連命令】 ADCF,ADD,SBB,SUB

ADCF

キャリーフラグの加算
ADdition Carry Flag

ADCF

【構文】

ADCF.size dest

B , W

【命令コード／サイクル数】

Page= 140

【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			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	○	－	○	○	－	○

条件

O ： 符号付き演算の結果、+32767(.W)または－32768(.W)、+127(.B)または－128(.B)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、+65535(.W)、+255(.B)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    Ram:16[A0]
```

【関連命令】 ADC,ADD,SBB,SUB

ADD

キャリーなし加算
ADDition

ADD

【構文】

ADD.size (:format)　　src,dest

G , Q , S（指定可能）
B , W

【命令コード／サイクル数】

Page= 140

【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。
・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。
・サイズ指定子(.size)に(.B)を指定した場合、destがスタックポインタのとき、srcを符号拡張し16ビットで演算します。

【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				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP*2
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。
*2　演算の対象はUフラグで示すスタックポインタです。srcには#IMMだけ選択できます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	○	－	○	○	－	○

条件

O ： 符号付き演算の結果、+32767(.W)または－32768(.W)、+127(.B)または－128(.B)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、+65535(.W)、+255(.B)を超えると“1”、それ以外のとき“0”になります。

【記述例】

```
ADD.B    A0,R0L          ;A0の下位8ビットとR0Lを演算します。
ADD.B    R0L,A0          ;R0Lをゼロ拡張してA0と演算します。
ADD.B    Ram:8[SB],R0L
ADD.W    #2,[A0]
```

【関連命令】　ADC,ADCF,SBB,SUB

【フォーマット別src / dest】

G フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP*2
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1 サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。
*2 演算の対象はUフラグで示すスタックポインタです。srcには#IMMだけ選択できます。

Q フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM*3	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP*2
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*2 演算の対象はUフラグで示すスタックポインタです。srcには#IMMだけ選択できます。
*3 取りうる範囲は−8≦#IMM≦+7です。

S フォーマット*4

src				dest			
R0L	R0H	dsp:8[SB]	dsp:8[FB]	R0L	R0H	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	
R0L*5	R0H*5	dsp:8[SB]	dsp:8[FB]	R0L*5	R0H*5	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	

*4 サイズ指定子(.size)には(.B)だけ指定できます。
*5 srcとdestに同じレジスタを選択できません。

ADJNZ

加算＆条件分岐
ADdition then Jump on Not Zero

ADJNZ

【構文】

ADJNZ.size src,dest,label
　　└──── **B , W**

【命令コード／サイクル数】

Page= 146

【オペレーション】

dest ← dest + src
if dest≠0 **then** jump label

【機能】

・destとsrcを加算し、destに格納します。
・加算した結果、0以外のときlabelへ分岐します。0のとき次の命令を実行します。
・本命令のオペコードは、SBJNZと同じです。

【選択可能なsrc / dest / label】

src	dest			label
#IMM[*1]	R0L/R0 R1H/R3 [A0] dsp:8[A1] dsp:16[A0] abs16	R0H/R1 A0/A0 [A1] dsp:8[SB] dsp:16[A1]	R1L/R2 A1/A1 dsp:8[A0] dsp:8[FB] dsp:16[SB]	PC[*2]–126≦label≦PC[*2]+129

*1　取りうる範囲は−8≦#IMM≦+7です。
*2　PCは命令の先頭番地を示します。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
ADJNZ.W　#–1,R0,label
```

【関連命令】　SBJNZ

AND

論理積

AND

AND

【構文】

AND.size (:format) src,dest

G , S（指定可能）

B , W

【命令コード／サイクル数】

Page= 147

【オペレーション】

dest ← src ∧ dest

【機能】

・destとsrcの論理積をとり、destに格納します。

・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【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				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

ﾌﾗｸﾞ	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     Ram:8[SB],R0L
AND.B:G   A0,R0L          ;A0の下位8ビットとR0Lを演算します。
AND.B:G   R0L,A0          ;R0Lをゼロ拡張してA0と演算します。
AND.B:S   #3,R0L
```

【関連命令】　OR,XOR,TST

【フォーマット別src / dest】

G フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

S フォーマット*2

src				dest			
R0L	R0H	dsp:8[SB]	dsp:8[FB]	R0L	R0H	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	
R0L*3	R0H*3	dsp:8[SB]	dsp:8[FB]	R0L*3	R0H*3	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	

*2　サイズ指定子(.size)には(.B)だけ指定できます。

*3　srcとdestに同じレジスタを選択できません。

BAND

ビット論理積
Bit AND carry flag

BAND

【構文】
BAND src

【命令コード／サイクル数】
Page= 150

【オペレーション】
C ← src ∧ C

【機能】
・Cフラグとsrcの論理積をとり、Cフラグに格納します。

【選択可能なsrc】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	○

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

【記述例】

```
BAND    flag
BAND    4,Ram
BAND    16,Ram:16[SB]
BAND    [A0]
```

【関連命令】 BOR,BXOR,BNAND,BNOR,BNXOR

BCLR

ビットクリア
Bit CLeaR

BCLR

【構文】

BCLR (:format) dest

G , S（指定可能）

【命令コード／サイクル数】

Page= 150

【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			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
e	bit,base:11[SB]*1		

*1 Sフォーマット時だけ選択できるdestです。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
BCLR    flag
BCLR    4,Ram:8[SB]
BCLR    16,Ram:16[SB]
BCLR    [A0]
```

【関連命令】 BSET,BNOT,BNTST,BTST,BTSTC,BTSTS

BM*Cnd*

条件ビット転送
Bit Move Condition

BM*Cnd*

【構文】
BM*Cnd*　　dest

【命令コード／サイクル数】
Page= 152

【オペレーション】
if true then　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∧$\overline{Z}$=1	大きい	＜	LEU	C∧$\overline{Z}$=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			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
C			

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	*1

*1　destにCフラグを指定したとき、変化します。

【記述例】
```
BMN     3,Ram:8[SB]
BMZ     C
```

【関連命令】　J*Cnd*

BNAND

反転ビット論理積
Bit Not AND carry flag

BNAND

【構文】
BNAND src

【命令コード／サイクル数】
Page= 153

【オペレーション】
C ← $\overline{\text{src}}$ ∧ C

【機能】
・Cフラグとsrcの反転の論理積をとり、Cフラグに格納します。

【選択可能なsrc】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	○

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

【記述例】

```
BNAND   flag
BNAND   4,Ram
BNAND   16,Ram:16[SB]
BNAND   [A0]
```

【関連命令】 BAND,BOR,BXOR,BNOR,BNXOR

BNOR

反転ビット論理和
Bit Not OR carry flag

BNOR

【構文】
BNOR src

【命令コード／サイクル数】
Page=154

【オペレーション】
C ← $\overline{\text{src}}$ ∨ C

【機能】
・Cフラグとsrcの反転の論理和をとり、Cフラグに格納します。

【選択可能なsrc】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	○

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

【記述例】

```
BNOR    flag
BNOR    4,Ram
BNOR    16,Ram:16[SB]
BNOR    [A0]
```

【関連命令】 BAND,BOR,BXOR,BNAND,BNXOR

BNOT

ビット反転
Bit NOT

BNOT

【構文】

BNOT(:format) dest

└──── **G , S** （指定可能）

【命令コード／サイクル数】

Page=154

【オペレーション】

dest ← $\overline{\text{dest}}$

【機能】

・destを反転し、destに格納します。

【選択可能なdest】

dest			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
e	bit,base:11[SB]*1		

*1 Sフォーマット時だけ選択できるdestです。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
BNOT    flag
BNOT    4,Ram:8[SB]
BNOT    16,Ram:16[SB]
BNOT    [A0]
```

【関連命令】 BCLR,BSET,BNTST,BTST,BTSTC,BTSTS

BNTST

反転ビットテスト
Bit Not TeST

BNTST

【構文】

BNTST **src**

【命令コード／サイクル数】

Page=155

【オペレーション】

Z ← $\overline{\text{src}}$

C ← $\overline{\text{src}}$

【機能】

・srcの反転をZフラグとCフラグに転送します。

【選択可能なsrc】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	○	－	○

条件

Z ： srcが“0”のとき“1”、それ以外のとき“0”になります。

C ： srcが“0”のとき“1”、それ以外のとき“0”になります。

【記述例】

```
BNTST   flag
BNTST   4,Ram:8[SB]
BNTST   16,Ram:16[SB]
BNTST   [A0]
```

【関連命令】 BCLR,BSET,BNOT,BTST,BTSTC,BTSTS

BNXOR

反転ビット排他的論理和

Bit Not eXclusive OR carry flag

BNXOR

【構文】

BNXOR　　src

【命令コード／サイクル数】

Page=156

【オペレーション】

C ← $\overline{\text{src}}$ ∀ C

【機能】

・Cフラグとsrcの反転の排他的論理和をとり、Cフラグに格納します。

【選択可能なsrc】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	○

条件

C ： 演算の結果が“1”のとき“1”、それ以外のとき“0”になります。

【記述例】

```
BNXOR    flag
BNXOR    4,Ram
BNXOR    16,Ram:16[SB]
BNXOR    [A0]
```

【関連命令】　BAND,BOR,BXOR,BNAND,BNOR

BOR

ビット論理和
Bit OR carry flag

BOR

【構文】
BOR src

【命令コード／サイクル数】
Page=156

【オペレーション】
C ← src ∨ C

【機能】
・Cフラグとsrcの論理和をとり、Cフラグに格納します。

【選択可能なsrc】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	○

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

【記述例】

```
BOR     flag
BOR     4,Ram
BOR     16,Ram:16[SB]
BOR     [A0]
```

【関連命令】 BAND,BXOR,BNAND,BNOR,BNXOR

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=157

【オペレーション】

SP ← SP − 2
M(SP) ← (PC + 1)H, FLG
SP ← SP − 2
M(SP) ← (PC + 1)ML
PC ← M(FFFE416)

【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】 *1

ﾌﾗｸﾞ	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。

*1 BRK命令実行前のフラグはスタック領域に退避され、割り込み後は左のとおりになります。

【記述例】

```
BRK
```

【関連命令】 INT,INTO

BSET

ビットセット
Bit SET

BSET

【構文】

BSET (:format)　dest

G , S（指定可能）

【命令コード／サイクル数】

Page=157

【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			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]*1		

*1　Sフォーマット時だけ選択できるdestです。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
BSET    flag
BSET    4,Ram:8[SB]
BSET    16,Ram:16[SB]
BSET    [A0]
```

【関連命令】　BCLR,BNOT,BNTST,BTST,BTSTC,BTSTS

BTST

ビットテスト
Bit TeST

BTST

【構文】

BTST (:format) src

G , S（指定可能）

【命令コード／サイクル数】

Page=158

【オペレーション】

Z ← $\overline{src}$

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】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
e	bit,base:11[SB]*1		

*1　Ｓフォーマット時だけ選択できるsrcです。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	○	－	○

条件

Ｚ ： srcが“０”のとき“１”、それ以外のとき“０”になります。

Ｃ ： srcが“１”のとき“１”、それ以外のとき“０”になります。

【記述例】

```
BTST    flag
BTST    4,Ram:8[SB]
BTST    16,Ram:16[SB]
BTST    [A0]
```

【関連命令】　BCLR,BSET,BNOT,BNTST,BTSTC,BTSTS

BTSTC

ビットテスト＆クリア
Bit TeST & Clear

BTSTC

【構文】

BTSTC dest

【命令コード／サイクル数】

Page=159

【オペレーション】

Z ← $\overline{\text{dest}}$
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。

【選択可能なdest】

dest			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	○	－	○

条件

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

【記述例】

```
BTSTC   flag
BTSTC   4,Ram
BTSTC   16,Ram:16[SB]
BTSTC   [A0]
```

【関連命令】 BCLR,BSET,BNOT,BNTST,BTST,BTSTS

BTSTS

ビットテスト＆セット
Bit TeST & Set

BTSTS

【構文】

BTSTS　　dest

【命令コード／サイクル数】

Page=160

【オペレーション】

Z　←　$\overline{\text{dest}}$
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。

【選択可能なdest】

dest			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	○	－	○

条件

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

【記述例】

```
BTSTS    flag
BTSTS    4,Ram
BTSTS    16,Ram:16[SB]
BTSTS    [A0]
```

【関連命令】　BCLR,BSET,BNOT,BNTST,BTST,BTSTC

BXOR

ビット排他的論理和
Bit eXclusive OR carry flag

BXOR

【構文】
BXOR src

【命令コード／サイクル数】
Page=160

【オペレーション】
C ← src ∀ C

【機能】
・Cフラグとsrcの排他的論理和をとり、Cフラグに格納します。

【選択可能なsrc】

src			
bit,R0	bit,R1	bit,R2	bit,R3
bit,A0	bit,A1	[A0]	[A1]
base:8[A0]	base:8[A1]	bit,base:8[SB]	bit,base:8[FB]
base:16[A0]	base:16[A1]	bit,base:16[SB]	bit,base:16
∈	bit,base:11[SB]		

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	○

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

【記述例】

```
BXOR    flag
BXOR    4,Ram
BXOR    16,Ram:16[SB]
BXOR    [A0]
```

【関連命令】 BAND,BOR,BNAND,BNOR,BNXOR

CMP

比較
CoMPare

CMP

【構文】

CMP.size (:format) src,dest

G , Q , S（指定可能）
B , W

【命令コード／サイクル数】

Page=161

【オペレーション】

dest ← src

【機能】

・destからsrcを減算した結果に従って、フラグレジスタの各フラグが変化します。
・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【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				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	○	－	○	○	－	○

条件

O ： 符号付き演算の結果、+32767(.W)または－32768(.W)、+127(.B)または－128(.B)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に等しいかまたは0より大きいとき“1”、それ以外のとき“0”になります。

【記述例】

```
CMP.B:S    #10,R0L
CMP.W:G    R0,A0
CMP.W      #–3,R0
CMP.B      #5,Ram:8[FB]
CMP.B      A0,R0L                    ;A0の下位8ビットとR0Lを比較します。
```

【フォーマット別src / dest】

G フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

Q フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM*2	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*2　取りうる範囲は−8≦#IMM≦+7です。

S フォーマット*3

src				dest			
R0L	R0H	dsp:8[SB]	dsp:8[FB]	R0L	R0H	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	
R0L*4	R0H*4	dsp:8[SB]	dsp:8[FB]	R0L*4	R0H*4	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	

*3　サイズ指定子(.size)には(.B)だけ指定できます。

*4　srcとdestに同じレジスタを選択できません。

DADC

キャリー付き10進加算
Decimal ADdition with Carry

DADC

【構文】

DADC.size src,dest
B , W

【命令コード／サイクル数】
Page=165

【オペレーション】

dest ← src + dest + C

【機能】

・destとsrcとCフラグを10進で加算し、destに格納します。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	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、+9999(.W)、+99(.B)を超えると“1”、それ以外のとき“0”になります。

【記述例】

```
DADC.B   #3,R0L
DADC.W   R1,R0
```

【関連命令】 DADD,DSUB,DSBB

DADD

キャリーなし10進加算
Decimal ADDition

DADD

【構文】

DADD.size src,dest

B , W

【命令コード／サイクル数】

Page=167

【オペレーション】

dest ← src + dest

【機能】

・destとsrcを10進で加算し、destに格納します。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	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、+9999(.W)、+99(.B)を超えると“1”、それ以外のとき“0”になります。

【記述例】

```
DADD.B    #3,R0L
DADD.W    R1,R0
```

【関連命令】 DADC,DSUB,DSBB

DEC

デクリメント
DECrement

DEC

【構文】

DEC.size dest

B , W

【命令コード／サイクル数】

Page=169

【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			
R0L*1	R0H*1	dsp:8[SB]*1	dsp:8[FB]*1
abs16*1	A0*2	A1*2	

*1　サイズ指定子(.size)には(.B)だけ指定できます。
*2　サイズ指定子(.size)には(.W)だけ指定できます。

【フラグ変化】

ﾌﾗｸﾞ	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.W   A0
DEC.B   R0L
```

【関連命令】 INC

DIV

符号付き除算
DIVide

DIV

【構文】

DIV.size src

B , W

【命令コード／サイクル数】

Page=170

【オペレーション】

サイズ指定子(.size)が(.B)のとき
R0L(商)、R0H(剰余)←R0÷src
サイズ指定子(.size)が(.W)のとき
R0(商)、R2(剰余)←R2R0÷src

【機能】

・R2R0(R0)[*1]を符号付きのsrcで除算します。商をR0(R0L)[*1]に、剰余をR2(R0H)[*1]に格納します。剰余の符号は被除数の符号と同一になります。( )[*1]内はサイズ指定子(.size)に(.B)を指定した場合です。
・サイズ指定子(.size)に(.B)を指定した場合、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。
・サイズ指定子(.size)に(.B)を指定した場合、演算の結果、商が8ビットを超えるか、または除数が0のとき、Oフラグが“1”になります。このとき、R0L、R0Hは不定になります。
・サイズ指定子(.size)に(.W)を指定した場合、演算の結果、商が16ビットを超えるか、または除数が0のとき、Oフラグが“1”になります。このとき、R0、R2は不定になります。

【選択可能なsrc】

src			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	○	－	－	－	－	－

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

【記述例】

```
DIV.B   A0          ;A0の下位8ビットが除数となります。
DIV.B   #4
DIV.W   R0
```

【関連命令】 DIVU,DIVX,MUL,MULU

DIVU

符号なし除算
DIVide Unsigned

DIVU

【構文】

DIVU.size src

B , W

【命令コード／サイクル数】

Page=171

【オペレーション】

サイズ指定子(.size)が(.B)のとき
R0L(商)、R0H(剰余)←R0÷src
サイズ指定子(.size)が(.W)のとき
R0(商)、R2(剰余)←R2R0÷src

【機能】

・R2R0(R0)[*1]を符号なしのsrcで除算します。商をR0(R0L)[*1]に、剰余をR2(R0H)[*1]に格納します。( )[*1]内はサイズ指定子(.size)に(.B)を指定した場合です。
・サイズ指定子(.size)に(.B)を指定した場合、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。
・サイズ指定子(.size)に(.B)を指定した場合、演算の結果、商が8ビットを超えるか、または除数が0のとき、Oフラグが“1”になります。このとき、R0L、R0Hは不定になります。
・サイズ指定子(.size)に(.W)を指定した場合、演算の結果、商が16ビットを超えるか、または除数が0のとき、Oフラグが“1”になります。このとき、R0、R2は不定になります。

【選択可能なsrc】

src			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM
R2R0	R3R1	A1A0	

【フラグ変化】

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

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

【記述例】

```
DIVU.B    A0          ;A0の下位8ビットが除数となります。
DIVU.B    #4
DIVU.W    R0
```

【関連命令】 DIV,DIVX,MUL,MULU

DIVX

符号付き除算
DIVide eXtension

DIVX

【構文】

DIVX.size src

B , W

【命令コード／サイクル数】

Page=172

【オペレーション】

サイズ指定子(.size)が(.B)のとき
R0L(商)、R0H(剰余)←R0÷src
サイズ指定子(.size)が(.W)のとき
R0(商)、R2(剰余)←R2R0÷src

【機能】

・R2R0(R0)[*1]を符号付きのsrcで除算します。商をR0(R0L)[*1]に、剰余をR2(R0H)[*1]に格納します。剰余の符号は除数の符号と同一になります。( )[*1]内はサイズ指定子(.size)に(.B)を指定した場合です。
・サイズ指定子(.size)に(.B)を指定した場合、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。
・サイズ指定子(.size)に(.B)を指定した場合、演算の結果、商が8ビットを超えるか、または除数が0のとき、Oフラグが“1”になります。このとき、R0L、R0Hは不定になります。
・サイズ指定子(.size)に(.W)を指定した場合、演算の結果、商が16ビットを超えるか、または除数が0のとき、Oフラグが“1”になります。このとき、R0、R2は不定になります。

【選択可能なsrc】

src			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM
R2R0	R3R1	A1A0	

【フラグ変化】

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

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

【記述例】

```
DIVX.B    A0          ;A0の下位8ビットが除数となります。
DIVX.B    #4
DIVX.W    R0
```

【関連命令】 DIV,DIVU,MUL,MULU

DSBB

ボロー付き10進減算

Decimal SuBtract with Borrow

DSBB

【構文】

DSBB.size src,dest

B , W

【命令コード／サイクル数】

Page=173

【オペレーション】

dest ← dest − src − $\overline{C}$

【機能】

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

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	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、0に等しいかまたは0より大きいとき“１”、それ以外のとき“０”になります。

【記述例】

```
DSBB.B    #3,R0L
DSBB.W    R1,R0
```

【関連命令】 DADC,DADD,DSUB

DSUB

ボローなし10進減算
Decimal SUBtract

DSUB

【構文】

DSUB.size src,dest

B , W

【命令コード／サイクル数】

Page=175

【オペレーション】

dest ← dest − src

【機能】

・destからsrcを10進で減算し、destに格納します。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	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、0に等しいかまたは0より大きいとき“1”、それ以外のとき“0”になります。

【記述例】

```
DSUB.B    #3,R0L
DSUB.W    R1,R0
```

【関連命令】 DADC,DADD,DSBB

ENTER

スタックフレームの構築
ENTER function

ENTER

【構文】
ENTER　　src

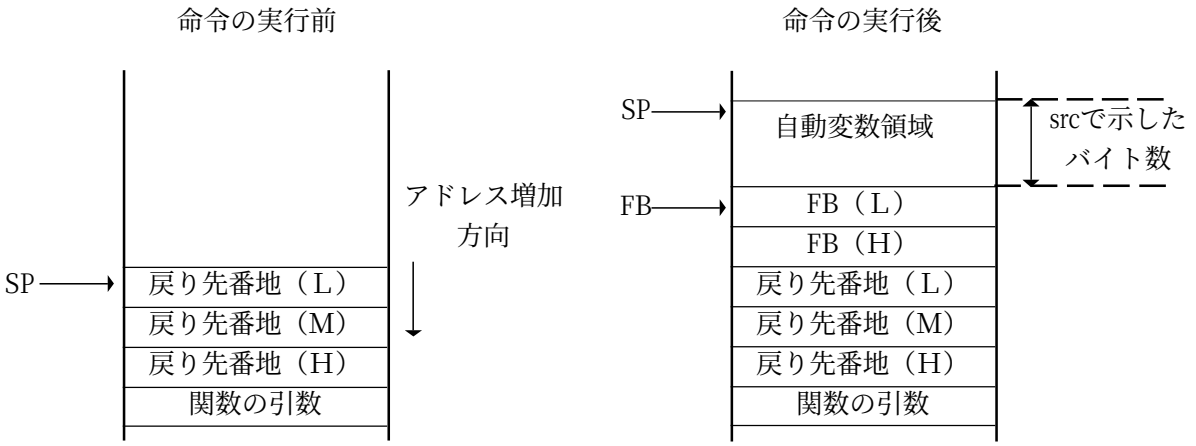
【命令コード／サイクル数】
Page=177

【オペレーション】

SP ← SP − 2
M(SP) ← FB
FB ← SP
SP ← SP − src

【機能】

・スタックフレームの生成を行います。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コールを行った後、呼び出された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
#IMM8

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
ENTER　　#3
```

【関連命令】　EXITD

EXITD

スタックフレームの解放
EXIT and Deallocate stack frame

EXITD

【構文】
EXITD

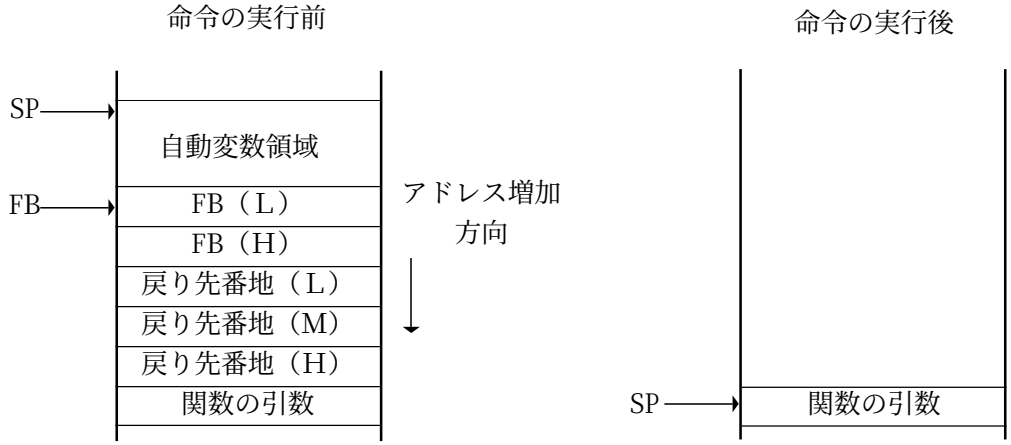
【命令コード／サイクル数】
Page=178

【オペレーション】

SP	←	FB
FB	←	M(SP)
SP	←	SP + 2
PCML	←	M(SP)
SP	←	SP + 2
PCH	←	M(SP)
SP	←	SP + 1

【機能】

・スタックフレームの解放とサブルーチンからの復帰を行います。
・本命令はENTER命令と対で使用してください。
・下記にENTER命令を実行した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】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
EXITD
```

【関連命令】 ENTER

EXTS

符号拡張
EXTend Sign

EXTS

【構文】

EXTS.size dest

B , W

【命令コード／サイクル数】

Page=178

【オペレーション】

dest ← EXT(dest)

【機能】

・destを符号拡張し、destに格納します。
・サイズ指定子(.size)に(.B)を指定した場合、destを16ビットに符号拡張します。
・サイズ指定子(.size)に(.W)を指定した場合、R0を32ビットに符号拡張します。このとき、上位バイトにはR2を使用します。

【選択可能なdest】

dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	○	○	－	－

条件

S ： サイズ指定子(.size)に(.B)を指定した場合、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。サイズ指定子(.size)に(.W)を指定した場合、変化しません。

Z ： サイズ指定子(.size)に(.B)を指定した場合、演算の結果が0のとき“1”、それ以外のとき“0”になります。サイズ指定子に(.W)を指定したときは、変化しません。

【記述例】

```
EXTS.B    R0L
EXTS.W    R0
```

FCLR

フラグレジスタのビットクリア
Flag register CLeaR

FCLR

【構文】
FCLR dest

【命令コード／サイクル数】
Page=179

【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							
C	D	Z	S	B	O	I	U

【フラグ変化】

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

*1 選択したフラグが“0”になります。

【記述例】

```
FCLR    I
FCLR    S
```

【関連命令】 FSET

FSET

フラグレジスタのビットセット
Flag register SET

FSET

【構文】
FSET dest

【命令コード／サイクル数】
Page=180

【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							
C	D	Z	S	B	O	I	U

【フラグ変化】

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

*1 選択したフラグが“1”になります。

【記述例】

```
FSET    I
FSET    S
```

【関連命令】 FCLR

INC

インクリメント
INCrement

INC

【構文】

INC.size dest

B , W

【命令コード／サイクル数】

Page=180

【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			
R0L*1	R0H*1	dsp:8[SB]*1	dsp:8[FB]*1
abs16*1	A0*2	A1*2	

*1 サイズ指定子(.size)には(.B)だけ指定できます。
*2 サイズ指定子(.size)には(.W)だけ指定できます。

【フラグ変化】

ﾌﾗｸﾞ	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.W   A0
INC.B   R0L
```

【関連命令】 DEC

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=181

【オペレーション】

SP ← SP − 2
M(SP) ← (PC + 2)H, FLG
SP ← SP − 2
M(SP) ← (PC + 2)ML
PC ← M(IntBase + src × 4)

【機能】

・srcで指定したソフトウエア割り込みが発生します。srcは、ソフトウエア割り込み番号です。
・srcが31以下の場合、Uフラグが“0”になりISPを使用します。
・srcが32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*1*2

*1 #IMMは、ソフトウエア割り込み番号です。
*2 取りうる範囲は**0 ≦#IMM≦63**です。

【フラグ変化】

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

*3 INT命令実行前のフラグはスタック領域に退避され、割り込み後は左のとおりになります。

条件
U ： ソフトウエア割り込み番号が31以下のとき“0”になります。ソフトウエア割り込み番号が32以上のとき変化しません。
I ： “0”になります。
D ： “0”になります。

【記述例】

```
INT     #0
```

【関連命令】 BRK,INTO

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=182

【オペレーション】

SP ← SP − 2
M(SP) ← (PC + 1)H, FLG
SP ← SP − 2
M(SP) ← (PC + 1)ML
PC ← M(FFFE016)

【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、“0”のとき次の命令を実行します。
・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】

ﾌﾗｸﾞ	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。

*1 INTO命令実行前のフラグはスタック領域に退避され、割り込み後は左のとおりになります。

【記述例】

```
INTO
```

【関連命令】 BRK,INT

J*Cnd*

条件分岐
Jump on Condition

J*Cnd*

【構文】

J*Cnd*　label

【命令コード／サイクル数】

Page=182

【オペレーション】

if true then jump label

【機能】

・前命令の実行結果を下記条件で判断し分岐します。*Cnd*で示した条件が真であればlabelへ分岐します。偽であれば次の命令を実行します。

・*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∧$\overline{Z}$=1	大きい	＜	LEU	C∧$\overline{Z}$=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	***Cnd***
PC[*1]–127 ≦ label ≦ PC[*1]+128	GEU/C,GTU,EQ/Z,N,LTU/NC,LEU,NE/NZ,PZ
PC[*1]–126 ≦ label ≦ PC[*1]+129	LE,O,GE,GT,NO,LT

*1　PCは命令の先頭番地を示します。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
JEQ     label
JNE     label
```

【関連命令】　BM*Cnd*

JMP

無条件分岐

JuMP

JMP

【構文】

JMP(.length) label

S , B , W , A（指定可能）

【命令コード／サイクル数】

Page=183

【オペレーション】

PC ← label

【機能】

・labelへ分岐します。

【選択可能なlabel】

.length	label
.S	PC*1+2≦ label ≦ PC*1+9
.B	PC*1–127 ≦ label ≦ PC*1+128
.W	PC*1–32767 ≦ label ≦ PC*1+32768
.A	abs20

*1 PCは命令の先頭番地を示します。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	–	–	–	–	–	–	–	–

【記述例】

```
JMP     label
```

【関連命令】 JMPI

JMPI

間接分岐
JuMP Indirect

JMPI

【構文】

JMPI.length src
W , A

【命令コード／サイクル数】
Page=185

【オペレーション】

分岐距離指定子(.length)が(.W)のとき
PC ← PC ± src

分岐距離指定子(.length)が(.A)のとき
PC ← src

【機能】

・srcが示す番地に分岐します。srcがメモリのとき、下位番地が格納されている番地を指定してください。
・分岐距離指定子(.length)に(.W)を指定した場合、命令の先頭番地とsrcを符号付き加算した番地に分岐します。また、srcがメモリのとき、必要なメモリ容量は2バイトです。
・分岐距離指定子(.length)に(.A)を指定した場合、srcがメモリのとき、必要なメモリ容量は3バイトです。

【選択可能なsrc】

分岐距離指定子(.length)に(.W)を指定した場合

src			
~~R0L~~ R0	~~R0H~~ R1	~~R1L~~ R2	~~R1H~~ R3
~~A0~~ A0	~~A1~~ A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
~~dsp:16[A0]~~	~~dsp:16[A1]~~	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	~~abs20~~	
~~R2R0~~	~~R3R1~~	~~A1A0~~	

分岐距離指定子(.length)に(.A)を指定した場合

src			
~~R0L/R0~~	~~R0H/R1~~	~~R1L/R2~~	~~R1H/R3~~
~~A0/A0~~	~~A1/A1~~	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
~~dsp:16[A0]~~	~~dsp:16[A1]~~	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	~~abs20~~	
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
JMPI.A    A1A0
JMPI.W    R0
```

【関連命令】 JMP

JSR

サブルーチン呼び出し
Jump SubRoutine

JSR

【構文】

JSR(.length) label

W , A（指定可能）

【命令コード／サイクル数】

Page=187

【オペレーション】

SP ← SP − 1
M(SP) ← (PC + n)H
SP ← SP − 2
M(SP) ← (PC + n)ML
PC ← label
*1 nは命令のバイト数です。

【機能】

・labelへサブルーチン分岐します。

【選択可能なlabel】

.length	label
.W	PC*1–32767≦ label ≦ PC*1+32768
.A	abs20

*1 PCは命令の先頭番地を示します。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
JSR.W   func
JSR.A   func
```

【関連命令】 JSRI

JSRI

間接サブルーチン呼び出し
Jump SubRoutine Indirect

JSRI

【構文】

JSRI.length src

W , A

【命令コード／サイクル数】

Page=188

【オペレーション】

分岐距離指定子(.length)が(.W)のとき

SP ← SP − 1
M(SP) ← (PC + n)H
SP ← SP − 2
M(SP) ← (PC + n)ML
PC ← PC ± src

*1 nは命令のバイト数です。

分岐距離指定子(.length)が(.A)のとき

SP ← SP − 1
M(SP) ← (PC + n)H
SP ← SP − 2
M(SP) ← (PC + n)ML
PC ← src

【機能】

・srcが示す番地にサブルーチン分岐します。srcがメモリのとき、下位番地が格納されている番地を指定してください。
・分岐距離指定子(.length)に(.W)を指定した場合、命令の先頭番地とsrcを符号付き加算した番地にサブルーチン分岐します。また、srcがメモリのとき、必要なメモリ容量は2バイトです。
・分岐距離指定子(.length)に(.A)を指定した場合、srcがメモリのとき、必要なメモリ容量は3バイトです。

【選択可能なsrc】

分岐距離指定子(.length)に(.W)を指定した場合

src			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

分岐距離指定子(.length)に(.A)を指定した場合

src			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
JSRI.A   A1A0
JSRI.W   R0
```

【関連命令】 JSR

LDC

専用レジスタへの転送

LoaD Control register

LDC

【構文】

LDC src,dest

【命令コード／サイクル数】

Page=188

【オペレーション】

dest ← src

【機能】

・srcをdestが示す専用レジスタに転送します。srcがメモリのとき、必要なメモリ容量は2バイトです。
・INTBL、またはINTBHに転送する場合は、連続して転送するようにしてください。
・この命令の直後には、割り込み要求を受け付けません。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	FB	SB	SP*1	ISP
A0/A0	A1/A1	[A0]	[A1]	FLG	INTBH	INTBL	
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]				
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16				
dsp:20[A0]	dsp:20[A1]	abs20	#IMM				
R2R0	R3R1	A1A0					

*1 Uフラグで示すスタックポインタが対象になります。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	*2	*2	*2	*2	*2	*2	*2	*2

*2 destがFLGのときだけ変化します。

【記述例】

```
LDC     R0,SB
LDC     A0,FB
```

【関連命令】 POPC,PUSHC,STC,LDINTB

LDCTX

コンテキストの復帰
LoaD ConTeXt

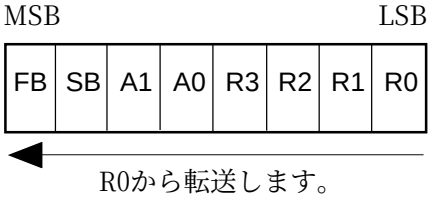
LDCTX

【構文】
LDCTX abs16,abs20

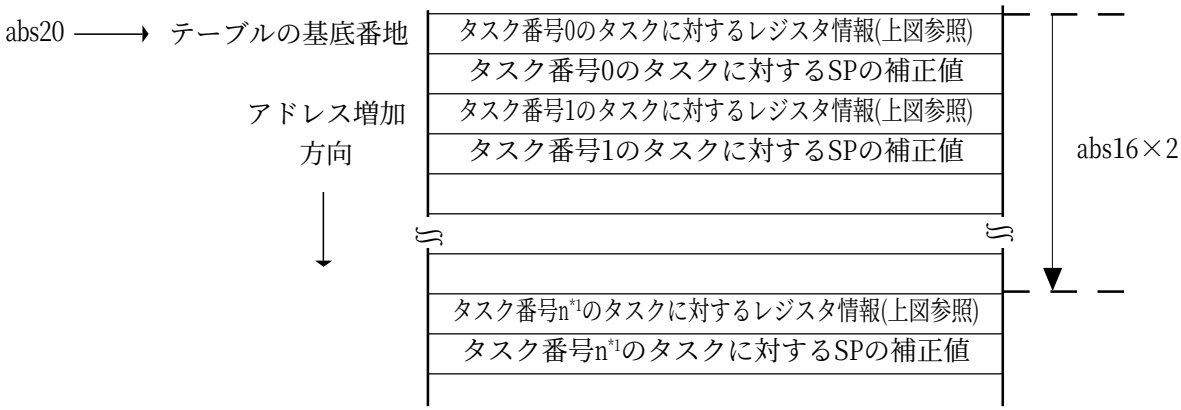
【命令コード／サイクル数】
Page=190

【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。
・abs16にはタスク番号が格納されているRAMの番地を、abs20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バイト数を設定してください。
・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。“１”で転送するレジスタ、“０”で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。abs20で示した番地がテーブルの基底番地となり、基底番地からabs16の内容の２倍離れた番地に格納されたデータがレジスタの情報、次の番地がスタックポインタの補正値を示します。



*1　n=0～255

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
LDCTX    Ram,Rom_TBL
```

【関連命令】 STCTX

LDE

拡張データ領域からの転送

LoaD from EXtra far data area

LDE

【構文】

LDE.size src,dest

B, W

【命令コード／サイクル数】

Page=191

【オペレーション】

dest ← src

【機能】

・拡張領域にあるsrcをdestに転送します。

・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで転送します。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	[A1A0]	R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	○	○	－	－

条件

S ： 転送の結果、destのMSBが“1”のとき“1”、それ以外のとき“0”になります。

Z ： 転送の結果、destが0のとき“1”、それ以外のとき“0”になります。

【記述例】

```
LDE.W   [A1A0],R0
LDE.B   Rom_TBL,A0
```

【関連命令】 STE,MOV,XCHG

LDINTB

INTBレジスタへの転送

LoaD INTB register

LDINTB

【構文】

LDINTB　　src

【命令コード／サイクル数】

Page=192

【オペレーション】

INTBHL ← src

【機能】

・srcをINTBに転送します。

・LDINTB命令は、以下の命令で構成されるマクロ命令です。

LDC　#IMM, INTBH

LDC　#IMM, INTBL

【選択可能なsrc】

src
#IMM20

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
LDINTB　　#0F0000H
```

【関連命令】　LDC,STC,PUSHC,POPC

LDIPL

割り込み許可レベルの設定

LoaD Interrupt Permission Level

LDIPL

【構文】

LDIPL src

【命令コード／サイクル数】

Page=193

【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*1

*1 取りうる範囲は0≦#IMM≦7です。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
LDIPL       #2
```

MOV

転送

MOVe

MOV

【構文】

MOV.size (:format) src,dest

G , Q , Z , S （指定可能）

B , W

【命令コード／サイクル数】

Page= 193

【オペレーション】

dest ← src

【機能】

・srcをdestに転送します。

・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで転送します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを転送します。

【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				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM*2	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	dsp:8[SP]*3	R2R0	R3R1	A1A0	dsp:8[SP]*2*3

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

*2　srcが#IMMの場合、destにdsp:8[SP]を選択できません。

*3　演算の対象はUフラグで示すスタックポインタです。また、srcとdestに同時にdsp:8［SP]を選択できません。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	○	○	－	－

条件

S ： 転送の結果、destのMSBが“1”のとき“1”、それ以外のとき“0”になります。

Z ： 転送の結果が0のとき“1”、それ以外のとき“0”になります。

【記述例】

```
MOV.B:S   #0ABH,R0L
MOV.W     #–1,R2
```

【関連命令】　LDE,STE,XCHG

【フォーマット別src / dest】

G フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM*2	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0	dsp:8[SP]*3	R2R0	R3R1	A1A0	dsp:8[SP]*2*3

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。
*2　srcが#IMMの場合、destにdsp:8[SP]を選択できません。
*3　演算の対象はUフラグで示すスタックポインタです。また、srcとdestに同時にdsp:8［SP]を選択できません。

Q フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM*4	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*4　取りうる範囲は−8≦#IMM≦+7です。

S フォーマット

src				dest			
R0L*5*6*7	R0H*5*6*8	dsp:8[SB]*5	dsp:8[FB]*5	R0L*5*6	R0H*5*6		
abs16*5	#IMM			abs16	A0*5*8	A1*5*7	
R0L*5*6	R0H*5*6	dsp:8[SB]	dsp:8[FB]	R0L*5*6	R0H*5*6	dsp:8[SB]*5	dsp:8[FB]*5
abs16	#IMM			abs16*5	A0	A1	
R0L	R0H	dsp:8[SB]	dsp:8[FB]	R0L*5	R0H*5	dsp:8[SB]*5	dsp:8[FB]*5
abs16	#IMM*9			abs16*5	A0*9	A1*9	

*5　サイズ指定子(.size)には(.B)だけ指定できます。
*6　srcとdestに同じレジスタを選択できません。
*7　srcがR0Lの場合、destにはアドレスレジスタとしてA1だけ選択できます。
*8　srcがR0Hの場合、destにはアドレスレジスタとしてA0だけ選択できます。
*9　サイズ指定子(.size)には(.B)および(.W)を指定できます。

Z フォーマット

src				dest			
R0L	R0H	dsp:8[SB]	dsp:8[FB]	R0L	R0H	dsp:8[SB]	dsp:8[FB]
abs16	#0			abs16	A0	A1	

MOVA

実効アドレスの転送
MOVe effective Address

MOVA

【構文】
MOVA src,dest

【命令コード／サイクル数】
Page= 200

【オペレーション】
dest ← EVA(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			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L R0	R0H R1	R1L R2	R1H R3
A0/A0	A1/A1	[A0]	[A1]	A0 A0	A1 A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
MOVA    Ram:16[SB],A0
```

【関連命令】 PUSHA

MOV*Dir*

4ビットデータ転送
MOVe nibble

MOV*Dir*

【構文】

MOV*Dir*　　src,dest

【命令コード／サイクル数】

Page= 201

【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】

・srcまたはdestのどちらか一方にR0Lを選択してください。

Dir	機能
HH	srcの上位4ビットをdestの上位4ビットに転送します
HL	srcの上位4ビットをdestの下位4ビットに転送します
LH	srcの下位4ビットをdestの上位4ビットに転送します
LL	srcの下位4ビットをdestの下位4ビットに転送します

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

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

MUL

符号付き乗算
MULtiple

MUL

【構文】

MUL.size src,dest

B , W

【命令コード／サイクル数】

Page= 203

【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)に(.B)を指定した場合、src、destともに8ビットで演算し、結果を16ビットで格納します。srcまたはdestのどちらか一方にA0またはA1を指定したときはA0またはA1の下位8ビットの内容を演算します。
・サイズ指定子(.size)に(.W)を指定した場合、src、destともに16ビットで演算し、結果を32ビットで格納します。destにR0、R1、A0を選択した場合、結果はそれぞれR2R0、R3R1、A1A0へ格納します。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
MUL.B    A0,R0L          ;R0LとA0の下位8ビットを演算します。
MUL.W    #3,R0
MUL.B    R0L,R1L
MUL.W    A0,Ram
```

【関連命令】　DIV,DIVU,DIVX,MULU

MULU

符号なし乗算
MULtiple Unsigned

MULU

【構文】

MULU.size src,dest
B , W

【命令コード／サイクル数】
Page= 205

【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)に(.B)を指定した場合、src、destともに8ビットで演算し、結果を16ビットで格納します。srcまたはdestのどちらか一方にA0またはA1を選択したときはA0またはA1の下位8ビットの内容を演算します。
・サイズ指定子(.size)に(.W)を指定した場合、src、destともに16ビットで演算し、結果を32ビットで格納します。destにR0、R1、A0を選択した場合、結果はそれぞれR2R0、R3R1、A1A0へ格納します。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
MULU.B    A0,R0L          ;R0LとA0の下位8ビットを演算します。
MULU.W    #3,R0
MULU.B    R0L,R1L
MULU.W    A0,Ram
```

【関連命令】　DIV,DIVU,DIVX,MUL

NEG

2の補数
NEGate

NEG

【構文】

NEG.size dest

B , W

【命令コード／サイクル数】

Page= 207

【オペレーション】

dest ← 0 − dest

【機能】

・destの2の補数をとり、destに格納します。

【選択可能なdest】

dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

【フラグ変化】

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

条件

O ： 演算前のdestが−128(.B)または−32768(.W)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   A1
```

【関連命令】 NOT

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= 207

【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】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】
NOP

NOT

全ビット反転

NOT

NOT

【構文】

NOT.size (:format) dest

G , S（指定可能）

B , W

【命令コード／サイクル数】

Page= 208

【オペレーション】

dest ← $\overline{\text{dest}}$

【機能】

・destを反転し、destに格納します。

【選択可能なdest】

dest			
R0L*1/R0	R0H*1/R1	R1L/R2	R1H/R3
A0	A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]*1	dsp:8[FB]*1
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16*1
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

*1 Gフォーマット、Sフォーマットで選択できます。
その他のdestはGフォーマットで選択できます。

【フラグ変化】

ﾌﾗｸﾞ	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   A1
```

【関連命令】 NEG

OR

論理和

OR

OR

【構文】

OR.size (:format) src,dest

G , S（指定可能）

B , W

【命令コード／サイクル数】

Page= 209

【オペレーション】

dest ← src ∨ dest

【機能】

・destとsrcの論理和をとり、destに格納します。

・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【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				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

ﾌﾗｸﾞ	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      Ram:8[SB],R0L
OR.B:G    A0,R0L          ;A0の下位8ビットとR0Lを演算します。
OR.B:G    R0L,A0          ;R0Lをゼロ拡張してA0と演算します。
OR.B:S    #3,R0L
```

【関連命令】　AND,XOR,TST

【フォーマット別src / dest】

G フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

S フォーマット*2

src				dest			
R0L	R0H	dsp:8[SB]	dsp:8[FB]	R0L	R0H	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	
R0L*3	R0H*3	dsp:8[SB]	dsp:8[FB]	R0L*3	R0H*3	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	

*2　サイズ指定子(.size)には(.B)だけ指定できます。

*3　srcとdestに同じレジスタを選択できません。

POP

レジスタ／メモリの復帰

POP

POP

【構文】

POP.size (:format) dest

G , S（指定可能）

B , W

【命令コード／サイクル数】

Page= 211

【オペレーション】

サイズ指定子(.size)が(.B)のとき			サイズ指定子(.size)が(.W)のとき		
dest	←	M(SP)	dest	←	M(SP)
SP	←	SP + 1	SP	←	SP + 2

【機能】

・destをスタック領域から復帰します。

【選択可能なdest】

dest			
R0L[*1]/R0	R0H[*1]/R1	R1L/R2	R1H/R3
~~A0/~~A0[*1]	~~A1/~~A1[*1]	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
~~dsp:20[A0]~~	~~dsp:20[A1]~~	~~abs20~~	
~~R2R0~~	~~R3R1~~	~~A1A0~~	

*1 Gフォーマット、Sフォーマットで選択できます。
その他のdestはGフォーマットで選択できます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
POP.B   R0L
POP.W   A0
```

【関連命令】 PUSH,POPM,PUSHM

POPC

専用レジスタの復帰

POP Control register

POPC

【構文】

POPC dest

【命令コード／サイクル数】

Page= 213

【オペレーション】

dest ← M(SP)

SP*1 ← SP + 2

*1 destがSPの場合、またはUフラグが“0”の状態でdestがISPの場合、SPは2を加算されません。

【機能】

・スタック領域からdestで示す専用レジスタに復帰します。

・割り込みテーブルレジスタを復帰する場合は、必ずINTBH、INTBLを連続して復帰してください。

・この命令の直後には、割り込み要求を受け付けません。

【選択可能なdest】

dest
FB SB SP*2 ISP FLG INTBH INTBL

*2 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】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	*3	*3	*3	*3	*3	*3	*3	*3

*3 destがFLGのときだけ変化します。

【記述例】

```
POPC    SB
```

【関連命令】 PUSHC,LDC,STC,LDINTB

POPM

複数レジスタの復帰
POP Multiple

POPM

【構文】

POPM **dest**

【命令コード／サイクル数】

Page= 213

【オペレーション】

dest ← M(SP)

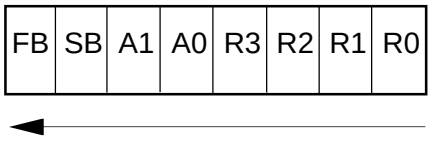
SP ← SP + N*1 × 2

*1 復帰するレジスタ数。

【機能】

・destで選択したレジスタを一括してスタック領域から復帰します。

・スタック領域から以下の優先順位で復帰します。



R0から復帰します

【選択可能なdest】

dest*2
R0 R1 R2 R3 A0 A1 SB FB

*2 複数のdestを選択することができます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
POPM    R0,R1,A0,SB,FB
```

【関連命令】 POP,PUSH,PUSHM

PUSH

レジスタ／メモリ／即値の退避
PUSH

PUSH

【構文】

PUSH.size (:format)　　src

- **G , S**（指定可能）
- **B , W**

【命令コード／サイクル数】

Page= 214

【オペレーション】

サイズ指定子(.size)が(.B)のとき			サイズ指定子(.size)が(.W)のとき		
SP	←	SP － 1	SP	←	SP － 2
M(SP)	←	src	M(SP)	←	src

【機能】

・srcをスタック領域へ退避します。

【選択可能なsrc】

src			
R0L[*1]/R0	R0H[*1]/R1	R1L/R2	R1H/R3
A0[*1]	A1[*1]	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM
R2R0	R3R1	A1A0	

*1　Gフォーマット、Sフォーマットで選択できます。
　　その他のsrcはGフォーマットで選択できます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
PUSH.B    #5
PUSH.W    #100H
PUSH.B    R0L
PUSH.W    A0
```

【関連命令】　POP,POPM,PUSHM

PUSHA

実効アドレスの退避
PUSH effective Address

PUSHA

【構文】

PUSHA **src**

【命令コード／サイクル数】
Page= 216

【オペレーション】

SP ← SP − 2
M(SP) ← EVA(src)

【機能】

・srcの実効アドレスをスタック領域へ退避します。

【選択可能なsrc】

src			
~~R0L/R0~~	~~R0H/R1~~	~~R1L/R2~~	~~R1H/R3~~
~~A0/A0~~	~~A1/A1~~	~~[A0]~~	~~[A1]~~
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
~~dsp:20[A0]~~	~~dsp:20[A1]~~	~~abs20~~	
~~R2R0~~	~~R3R1~~	~~A1A0~~	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
PUSHA   Ram:8[FB]
PUSHA   Ram:16[SB]
```

【関連命令】 MOVA

PUSHC

専用レジスタの退避

PUSH Control register

PUSHC

【構文】

PUSHC　　src

【命令コード／サイクル数】

Page= 216

【オペレーション】

SP　　←　　SP　－　2

M(SP)　　←　　src*1

*1　srcがSPの場合、またはUフラグが“0”の状態でsrcがISPの場合、2を減算される前のSPが退避されます。

【機能】

・srcで示す専用レジスタをスタック領域へ退避します。

【選択可能なsrc】

src
FB　SB　SP*2　ISP　FLG　INTBH　INTBL

*2　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】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
PUSHC　　SB
```

【関連命令】　　POPC,LDC,STC,LDINTB

PUSHM

複数レジスタの退避
PUSH Multiple

PUSHM

【構文】

PUSHM **src**

【命令コード／サイクル数】

Page= 217

【オペレーション】

SP ← SP − N*1 × 2

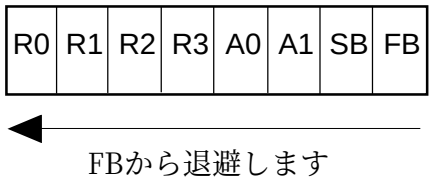
M(SP) ← src

*1 退避するレジスタ数。

【機能】

・srcで選択したレジスタを一括してスタック領域へ退避します。

・スタック領域へは以下の優先順位で退避します。



FBから退避します

【選択可能なsrc】

src*2
R0 R1 R2 R3 A0 A1 SB FB

*2 複数のsrcを選択することができます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
PUSHM   R0,R1,A0,SB,FB
```

【関連命令】 POP,PUSH,POPM

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= 217

【オペレーション】

PCML ← M(SP)
SP ← SP + 2
PCH,FLG ← M(SP)
SP ← SP + 2

【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】

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

*1 割り込み要求が受け付けられる前のFLGの状態に戻ります。

【記述例】

REIT

RMPA

積和演算

Repeat MultiPle & Addition

RMPA

【構文】

RMPA.size

B , W

【命令コード／サイクル数】

Page= 218

【オペレーション】[*1]

Repeat

	R2R0(R0)[*2]	←	R2R0(R0)[*2]	+	M(A0)	×	M(A1)
	A0	←	A0	+	2 (1)[*2]		
	A1	←	A1	+	2 (1)[*2]		
	R3	←	R3	−	1		
Until	R3 = 0						

*1　R3に0を設定して実行したとき、本命令は無視されます。

*2　()[*2]内は、サイズ指定子(.size)に(.B)を指定した場合です。

【機能】

・A0を被乗数番地、A1を乗数番地、R3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、結果はR2R0(R0)[*1]に格納します。

・演算中にオーバフローするとOフラグが“1”になり、演算を終了します。R2R0(R0)[*1]には、最後の加算結果を格納しています。A0、A1、およびR3は不定です。

・命令終了時のA0またはA1の内容は、最後に読み出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、演算の加算終了後(R3の内容が1減算された後)に割り込みを受け付けます。

・R2R0(R0)[*1]には初期値を設定してください。

*1　()[*1]内は、サイズ指定子(.size)に(.B)を指定した場合です。

【フラグ変化】

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

条件

O ： 演算中に+2147483647(.W)または−2147483648(.W)、+32767(.B)または−32768(.B)を超えると“1”、それ以外のとき“0”になります。

【記述例】

```
RMPA.B
```

ROLC

キャリー付き左回転
ROtate to Left with Carry

ROLC

【構文】

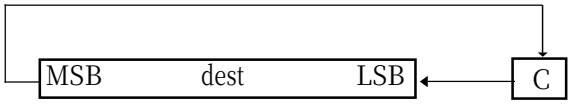
ROLC.size dest

B , W

【命令コード／サイクル数】

Page= 218

【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			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0	A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	○	○	－	○

条件

Ｓ ： 演算の結果、MSBが“１”のとき“１”、それ以外のとき“０”になります。
Ｚ ： 演算の結果、dest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    R0
```

【関連命令】 RORC,ROT,SHA,SHL

RORC

キャリー付き右回転
ROtate to Right with Carry

RORC

【構文】

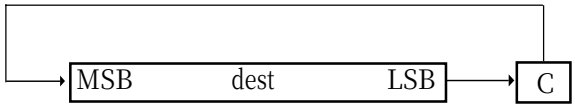
RORC.size dest

B , W

【命令コード／サイクル数】

Page= 219

【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			
R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0	A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	○	○	－	○

条件

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

Z ： 演算の結果、dest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    R0
```

【関連命令】 ROLC,ROT,SHA,SHL

ROT

回転
ROTate

ROT

【構文】

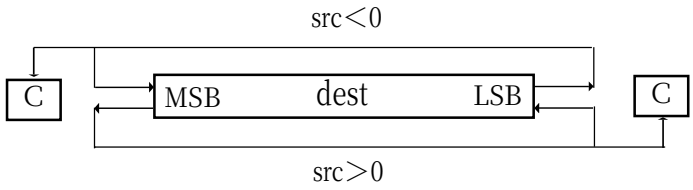
ROT.size src,dest

B , W

【命令コード／サイクル数】

Page= 220

【オペレーション】



【機能】

- destをsrcで示すビット数分回転します。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。
- srcが即値の場合、回転回数は−8〜−1および+1〜+8です。−9以下、0、および+9以上は設定できません。
- srcがレジスタの場合、サイズ指定子(.size)に(.B)を指定したとき、回転回数は−8〜+8です。0は設定できますが、回転しません。また、フラグレジスタの各フラグも変化しません。−9以下および+9以上を設定すると回転した結果は不定になります。
- srcがレジスタの場合、サイズ指定子(.size)に(.W)を指定したとき、回転回数は−16〜+16です。0は設定できますが、回転しません。また、フラグレジスタの各フラグも変化しません。−17以下および+17以上を設定すると回転した結果は不定になります。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H[*1]	R0L/R0	R0H/R1[*1]	R1L/R2	R1H/R3[*1]
A0/A0	A1/A1	[A0]	[A1]	A0 A0	A1 A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM[*2]	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　srcがR1Hの場合、destにR1またはR1Hを選択できません。

*2　取りうる範囲は−8≦#IMM≦+8です。ただし、0は設定できません。

【フラグ変化】

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

*1　回転回数が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”のとき“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 ： 最後に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   R1H,R2
```

【関連命令】　ROLC,RORC,SHA,SHL

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= 221

【オペレーション】

PCML	←	M(SP)
SP	←	SP + 2
PCH	←	M(SP)
SP	←	SP + 1

【機能】
・サブルーチンから復帰します。

【フラグ変化】

ﾌﾗｸﾞ	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

【命令コード／サイクル数】

Page= 222

【オペレーション】

dest ← dest − src − $\overline{C}$

【機能】

・destからsrcとＣフラグの反転を減算し、destに格納します。

・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

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

条件

O ： 符号付き演算の結果、+32767(.W)または−32768(.W)、+127(.B)または−128(.B)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に等しいかまたは0より大きいとき“１”、それ以外のとき“０”になります。

【記述例】

```
SBB.B    #2,R0L
SBB.W    A0,R0
SBB.B    A0,R0L          ;A0の下位8ビットとR0Lを演算します。
SBB.B    R0L,A0          ;R0Lをゼロ拡張してA0と演算します。
```

【関連命令】 ADC,ADCF,ADD,SUB

SBJNZ

減算＆条件分岐
SuBtract then Jump on Not Zero

SBJNZ

【構文】

SBJNZ.size src,dest,label
B , W

【命令コード／サイクル数】
Page= 224

【オペレーション】

dest ← dest − src
if dest≠0 **then** jump label

【機能】

・destからsrcを減算し、destに格納します。
・減算した結果、0 以外のときlabelへ分岐します。0のとき次の命令を実行します。
・本命令のオペコードは、ADJNZと同じです。

【選択可能なsrc / dest / label】

src	dest			label
#IMM[*1]	R0L/R0 R1H/R3 [A0] dsp:8[A1] dsp:16[A0] abs16	R0H/R1 A0/A0 [A1] dsp:8[SB] dsp:16[A1]	R1L/R2 A1/A1 dsp:8[A0] dsp:8[FB] dsp:16[SB]	PC[*2]–126 ≦ label ≦ PC[*2]+129

*1　取りうる範囲は−7≦#IMM≦+8です。
*2　PCは命令の先頭番地を示します。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SBJNZ.W    #1,R0,label
```

【関連命令】　ADJNZ

SHA

算術シフト
SHift Arithmetic

SHA

【構文】

SHA.size src,dest

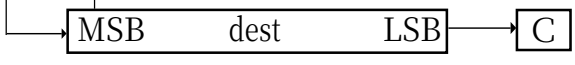
B , W , L

【命令コード／サイクル数】

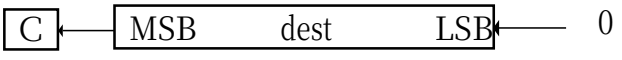
Page= 225

【オペレーション】

src＜0 のとき



src＞0 のとき



【機能】

・destをsrcで示すビット数分算術シフトします。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。
・srcが即値の場合、シフト回数は－8～－1および+1～+8です。－9以下、0、および+9以上は設定できません。
・srcがレジスタの場合、サイズ指定子(.size)に(.B)を指定したとき、シフト回数は－8～+8です。0は設定できますが、シフトしません。また、フラグレジスタの各フラグも変化しません。－9以下および+9以上を設定するとシフトした結果は不定になります。
・srcがレジスタの場合、サイズ指定子(.size)に(.W)または(.L)を指定したとき、シフト回数は－16～+16です。0は設定できますが、シフトしません。また、フラグレジスタの各フラグも変化しません。－17以下および+17以上を設定するとシフトした結果は不定になります。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H[*1]/R3	R0L/R0	R0H/R1[*1]	R1L/R2	R1H/R3[*1]
A0/A0	A1/A1	[A0]	[A1]	A0 A0	A1 A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM[*2]	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0[*3]	R3R1[*3]	A1A0	

*1　srcがR1Hの場合、destにR1またはR1Hを選択できません。
*2　取りうる範囲は－8≦#IMM≦+8です。ただし、0は設定できません。
*3　サイズ指定子(.size)には(.L)だけ指定できます。その他のdestは(.B)または(.W)を指定できます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	○	－	○	○	－	○

*1　シフト回数が0のとき、フラグは変化しません。

条件

O ： 演算の結果、MSBが“1”から“0”へ、または“0”から“1”へ変化したとき“1”、それ以外のとき“0”になります。ただし、サイズ指定子(.size)に(.L)を指定したときは変化しません。
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。ただし、サイズ指定子(.size)に(.L)を指定したときは不定になります。
C ： 最後に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。ただし、サイズ指定子(.size)に(.L)を指定したときは不定になります。

【記述例】

```
SHA.B   #3,R0L          ;左算術シフト
SHA.B   #–3,R0L         ;右算術シフト
SHA.L   R1H,R2R0
```

【関連命令】 ROLC,RORC,ROT,SHL

SHL

論理シフト
SHift Logical

SHL

【構文】

SHL.size src,dest

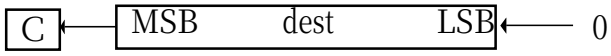
B , W , L

【命令コード／サイクル数】
Page= 228

【オペレーション】

src＜0 のとき

src＞0 のとき



【機能】

・destをsrcで示すビット数分論理シフトします。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。
・srcが即値の場合、シフト回数は－8～－1および+1～+8です。－9以下、0、および+9以上は設定できません。
・srcがレジスタの場合、サイズ指定子(.size)に(.B)を指定したとき、シフト回数は－8～+8です。0は設定できますが、シフトしません。また、フラグレジスタの各フラグも変化しません。－9以下および+9以上を設定するとシフトした結果は不定になります。
・srcがレジスタの場合、サイズ指定子(.size)に(.W)または(.L)を指定したとき、シフト回数は－16～+16です。0は設定できますが、シフトしません。また、フラグレジスタの各フラグも変化しません。－17以下および+17以上を設定するとシフトした結果は不定になります。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H[*1]	R0L/R0	R0H/R1[*1]	R1L/R2	R1H/R3[*1]
A0/A0	A1/A1	[A0]	[A1]	A0 A0	A1 A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM[*2]	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0[*3]	R3R1[*3]	A1A0	

*1　srcがR1Hの場合、destにR1またはR1Hを選択できません。
*2　取りうる範囲は－8≦#IMM≦+8です。ただし、0は設定できません。
*3　サイズ指定子(.size)には(.L)だけ指定できます。その他のdestは(.B)または(.W)を指定できます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	○	○	－	○

*1　シフト回数が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。ただし、サイズ指定子(.size)に(.L)を指定したときは不定になります。
C ： 最後に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。ただし、サイズ指定子(.size)に(.L)を指定したときは不定になります。

【記述例】

```
SHL.B   #3,R0L          ;左論理シフト
SHL.B   #–3,R0L         ;右論理シフト
SHL.L   R1H,R2R0
```

【関連命令】 ROLC,RORC,ROT,SHA

SMOVB

逆方向のストリング転送
String MOVe Backward

SMOVB

【構文】

SMOVB.size
　　└──── **B , W**

【命令コード／サイクル数】

Page= 230

【オペレーション】 *1

サイズ指定子(.size)が(.B)のとき

Repeat
　M(A1) ← M(2^{16} ×R1H + A0)
　A0*2 ← A0 − 1
　A1 ← A1 − 1
　R3 ← R3 − 1
Until R3 = 0

サイズ指定子(.size)が(.W)のとき

Repeat
　M(A1) ← M(2^{16} ×R1H + A0)
　A0*2 ← A0 − 2
　A1 ← A1 − 2
　R3 ← R3 − 1
Until R3 = 0

*1　R3に0を設定して実行したとき、本命令は無視されます。
*2　A0がアンダフローした場合、R1Hの内容は1減算されます。

【機能】

・20ビットで示される転送元番地から16ビットで示される転送先番地にアドレスの減算方向へストリング転送を行います。
・転送元番地の上位4ビットはR1H、転送元番地の下位16ビットはA0、転送先番地はA1、転送回数はR3に設定します。
・命令終了時のA0またはA1は、最後に読み出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データ転送終了後に割り込みを受け付けます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SMOVB.B
```

【関連命令】　SMOVF,SSTR

SMOVF

順方向のストリング転送
String MOVe Forward

SMOVF

【構文】

SMOVF.size
└ **B , W**

【命令コード／サイクル数】

Page= 231

【オペレーション】[*1]

サイズ指定子(.size)が(.B)のとき

Repeat
M(A1) ← M(2^{16} ×R1H + A0)
A0[*2] ← A0 + 1
A1 ← A1 + 1
R3 ← R3 − 1
Until R3 = 0

サイズ指定子(.size)が(.W)のとき

Repeat
M(A1) ← M(2^{16} ×R1H + A0)
A0[*2] ← A0 + 2
A1 ← A1 + 2
R3 ← R3 − 1
Until R3 = 0

*1 R3に0を設定して実行したとき、本命令は無視されます。
*2 A0がオーバフローした場合、R1Hの内容は1加算されます。

【機能】

・20ビットで示される転送元番地から16ビットで示される転送先番地にアドレスの加算方向へストリング転送を行います。
・転送元番地の上位4ビットはR1H、転送元番地の下位16ビットはA0、転送先番地はA1、転送回数はR3に設定します。
・命令終了時のA0またはA1は、最後に読み出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データ転送終了後に割り込みを受け付けます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SMOVF.W
```

【関連命令】 SMOVB,SSTR

SSTR

ストリングストア
String SToRe

SSTR

【構文】

SSTR.size
　　└──── **B , W**

【命令コード／サイクル数】
Page= 231

【オペレーション】[*1]

サイズ指定子(.size)が(.B)のとき

```
Repeat
    M(A1)  ←  R0L
    A1     ←  A1  +  1
    R3     ←  R3  −  1
Until          R3  =  0
```

サイズ指定子(.size)が(.W)のとき

```
Repeat
    M(A1)  ←  R0
    A1     ←  A1  +  2
    R3     ←  R3  −  1
Until          R3  =  0
```

*1　R3に0を設定して実行したとき、本命令は無視されます。

【機能】

・R0をストアするデータ、A1を転送するアドレス、R3を転送回数とし、ストリングストアを行います。
・命令終了時のA0またはA1の内容は、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データ転送終了後に割り込みを受け付けます。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	−	−	−	−	−	−	−	−

【記述例】

```
SSTR.B
```

【関連命令】　SMOVB,SMOVF

STC

専用レジスタからの転送

STore from Control register

STC

【構文】

STC **src,dest**

【命令コード／サイクル数】

Page= 232

【オペレーション】

dest ← src

【機能】

・destにsrcで示す専用レジスタを転送します。destがメモリのとき、下位番地の格納番地を指定してください。

・destがメモリの場合、srcがPCのとき、必要なメモリ容量は3バイトです。srcがPC以外のとき、必要なメモリ容量は2バイトです。

【選択可能なsrc / dest】

src				dest			
FB	SB	SP*1	ISP	R0L R0	R0H R1	R1L R2	R1H R3
FLG	INTBH	INTBL		A0 A0	A1 A1	[A0]	[A1]
				dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
				dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
				dsp:20[A0]	dsp:20[A1]	abs20	
				R2R0	R3R1	A1A0	
PC				R0L/R0	R0H/R1	R1L/R2	R1H/R3
				A0/A0	A1/A1	[A0]	[A1]
				dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
				dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
				dsp:20[A0]	dsp:20[A1]	abs20	
				R2R0	R3R1	A1A0	

*1　Uフラグで示すスタックポインタが対象になります。

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
STC     SB,R0
STC     FB,A0
```

【関連命令】 POPC,PUSHC,LDC,LDINTB

STCTX

コンテキストの退避
STore ConTeXt

STCTX

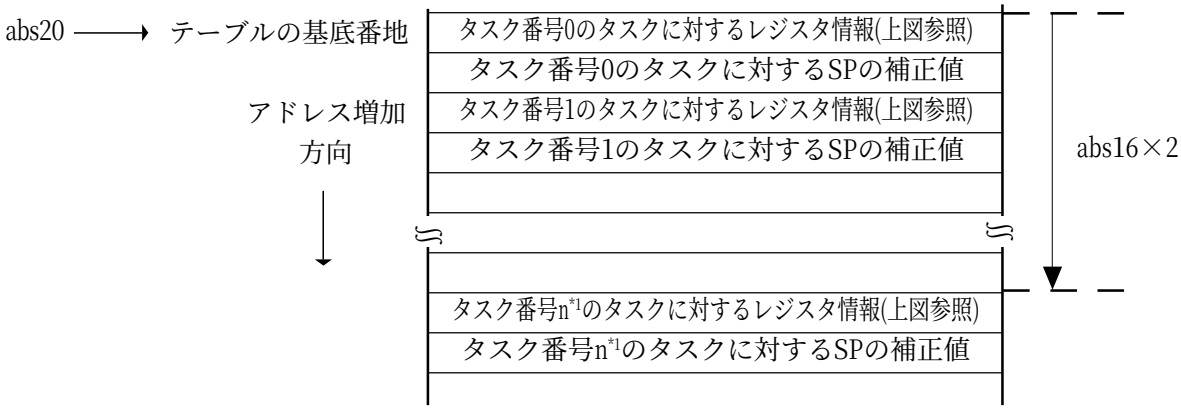
【構文】
STCTX abs16,abs20

【命令コード／サイクル数】
Page= 233

【オペレーション】

【機能】
- タスクのコンテキストをスタック領域へ退避します。
- abs16にはタスク番号が格納されているRAMの番地を、abs20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）から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バイト数を設定してください。
- 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。“１”で転送するレジスタ、“０”で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。abs20で示した番地がテーブルの基底番地となり、基底番地からabs16の内容の２倍離れた番地に格納されたデータがレジスタの情報、次の番地がスタックポインタの補正値を示します。



*1　n=0～255

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】
```
STCTX   Ram,Rom_TBL
```
【関連命令】 LDCTX

STE

拡張データ領域への転送

STore to EXtra far data area

STE

【構文】

【命令コード／サイクル数】

Page= 233

STE.size src,dest

B , W

【オペレーション】

dest ← src

【機能】

・srcを拡張領域にあるdestに転送します。

・サイズ指定子(.size)に(.B)を指定した場合、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。ただし、フラグは演算前のA0またはA1の状態(16ビット)で変化します。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	[A1A0]	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	○	○	－	－

条件

S：演算の結果、MSBが“1”のとき“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】

```
STE.B   R0L,[A1A0]
STE.W   R0,10000H[A0]
```

【関連命令】 MOV,LDE,XCHG

STNZ

条件付き転送

STore on Not Zero

STNZ

【構文】

STNZ **src,dest**

【命令コード／サイクル数】

Page= 235

【オペレーション】

if Z = 0 **then** dest ← src

【機能】

・Ｚフラグが“０”のとき、srcをdestに転送します。

【選択可能なsrc / dest】

src	dest			
#IMM8	R0L	R0H	dsp:8[SB]	dsp:8[FB]
	abs16	A0	A1	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
STNZ    #5,Ram:8[SB]
```

【関連命令】 STZ,STZX

STZ

条件付き転送

STore on Zero

STZ

【構文】

STZ **src,dest**

【命令コード／サイクル数】

Page= 235

【オペレーション】

if Z = 1 **then** dest ← src

【機能】

・Ｚフラグが“１”のとき、srcをdestに転送します。

【選択可能なsrc / dest】

src	dest
#IMM8	R0L R0H dsp:8[SB] dsp:8[FB] abs16 A0 A1

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
STZ     #5,Ram:8[SB]
```

【関連命令】 STNZ,STZX

STZX

条件付き転送

STore on Zero eXtention

STZX

【構文】

STZX **src1,src2,dest**

【命令コード／サイクル数】

Page= 236

【オペレーション】

If Z = 1 **then**

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】

src	dest			
#IMM8	R0L	R0H	dsp:8[SB]	dsp:8[FB]
	abs16	A0	A1	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
STZX    #1,#2,Ram:8[SB]
```

【関連命令】 STZ,STNZ

SUB

ボローなし減算
SUBtract

SUB

【構文】

SUB.size (:format) src,dest

- **G , S**（指定可能）
- **B , W**

【命令コード／サイクル数】

Page= 236

【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。
・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【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				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

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

条件

O ： 符号付き演算の結果、+32767(.W)または−32768(.W)、+127(.B)または−128(.B)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に等しいかまたは0より大きいとき“1”、それ以外のとき“0”になります。

【記述例】

```
SUB.B    A0,R0L              ;A0の下位8ビットとR0Lを演算します。
SUB.B    R0L,A0              ;R0Lをゼロ拡張してA0と演算します。
SUB.B    Ram:8[SB],R0L
SUB.W    #2,[A0]
```

【関連命令】　ADC,ADCF,ADD,SBB

【フォーマット別src / dest】

G フォーマット

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	SP/SP
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

S フォーマット*2

src				dest			
R0L	R0H	dsp:8[SB]	dsp:8[FB]	R0L	R0H	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	
R0L*3	R0H*3	dsp:8[SB]	dsp:8[FB]	R0L*3	R0H*3	dsp:8[SB]	dsp:8[FB]
abs16	#IMM			abs16	A0	A1	

*2　サイズ指定子(.size)には(.B)だけ指定できます。

*3　srcとdestに同じレジスタを選択できません。

TST

テスト
TeST

TST

【構文】

TST.size src,dest

B , W

【命令コード／サイクル数】

Page= 239

【オペレーション】

dest ∧ src

【機能】

・srcとdestの論理積をとった結果でフラグレジスタの各フラグが変化します。
・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1 サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

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

条件

S ： 演算の結果、MSBが“1”のとき“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   #3,R0L
TST.B   A0,R0L          ;A0の下位8ビットとR0Lを演算します。
TST.B   R0L,A0          ;R0Lをゼロ拡張してA0と演算します。
```

【関連命令】 AND,OR,XOR

UND

未定義命令割り込み
UNDefined instruction

UND

【構文】
UND

【命令コード／サイクル数】
Page= 241

【オペレーション】

SP ← SP − 2
M(SP) ← (PC + 1)H, FLG
SP ← SP − 2
M(SP) ← (PC + 1)ML
PC ← M(FFFDC16)

【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】

ﾌﾗｸﾞ	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。

*1 UND命令実行前のフラグはスタック領域に退避され、割り込み後は左のとおりになります。

【記述例】

```
UND
```

WAIT

ウエイト
WAIT

WAIT

【構文】
WAIT

【命令コード／サイクル数】
Page= 241

【オペレーション】

【機能】

・プログラムの実行を停止します。IPL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】

ﾌﾗｸﾞ	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

【命令コード／サイクル数】

Page= 242

【オペレーション】

dest ←→ src

【機能】

・srcとdestの内容を交換します。
・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張した16ビットのデータがA0またはA1に入り、A0またはA1の下位8ビットがsrcに入ります。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0	A1/A1	[A0]	[A1]	A0/A0	A1/A1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0	[A1A0]	R2R0	R3R1	A1A0	

【フラグ変化】

ﾌﾗｸﾞ	U	I	O	B	S	Z	D	C
変化	－	－	－	－	－	－	－	－

【記述例】

```
XCHG.B    R0L,A0          ;A0の下位８ビットとR0Lをゼロ拡張した値を交換します。
XCHG.W    R0,A1
XCHG.B    R0L,[A0]
```

【関連命令】 MOV,LDE,STE

XOR

排他的論理和
eXclusive OR

XOR

【構文】

XOR.size src,dest

B , W

【命令コード／サイクル数】

Page= 243

【オペレーション】

dest ← dest ∀ src

【機能】

・srcとdestの排他的論理和をとり、destに格納します。

・サイズ指定子(.size)に(.B)を指定した場合、destがA0またはA1のとき、srcをゼロ拡張し16ビットで演算します。また、srcがA0またはA1のとき、A0またはA1の下位8ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0	R0H/R1	R1L/R2	R1H/R3	R0L/R0	R0H/R1	R1L/R2	R1H/R3
A0/A0*1	A1/A1*1	[A0]	[A1]	A0/A0*1	A1/A1*1	[A0]	[A1]
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	abs16
dsp:20[A0]	dsp:20[A1]	abs20	#IMM	dsp:20[A0]	dsp:20[A1]	abs20	
R2R0	R3R1	A1A0		R2R0	R3R1	A1A0	

*1 サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にA0またはA1を選択できません。

【フラグ変化】

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

条件

S ： 演算の結果、MSBが“1”のとき“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   A0,R0L          ;A0の下位8ビットとR0Lを演算します。
XOR.B   R0L,A0          ;R0Lをゼロ拡張してA0と演算します。
XOR.B   #3,R0L
XOR.W   A0,A1
```

【関連命令】 AND,OR,TST

レイアウトの都合上、このページは白紙です。

第４章

命令コード／サイクル数

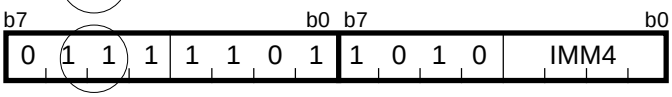
4.1　本章の見方

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
4.2　命令コード／サイクル数

① LDIPL

② (1) LDIPL #IMM



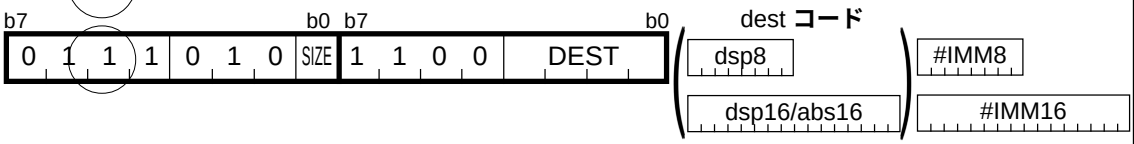
③

④ 【バイト数／サイクル数】

バイト数／サイクル数	2/2

① MOV

② (1) MOV.size:G #IMM, dest



③

.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0000	dsp:8[An]	dsp:8[A0]	1000
	R0H/R1	0001		dsp:8[A1]	1001
	R1L/R2	0010	dsp:8[SB/FB]	dsp:8[SB]	1010
	R1H/R3	0011		dsp:8[FB]	1011
An	A0	0100	dsp:16[An]	dsp:16[A0]	1100
	A1	0101		dsp:16[A1]	1101
[An]	[A0]	0110	dsp:16[SB]	dsp:16[SB]	1110
	[A1]	0111	abs16	abs16	1111

④ 【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数/サイクル数	3/2	3/2	3/3	4/3	4/3	5/3	5/3	5/3

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

①ニーモニック

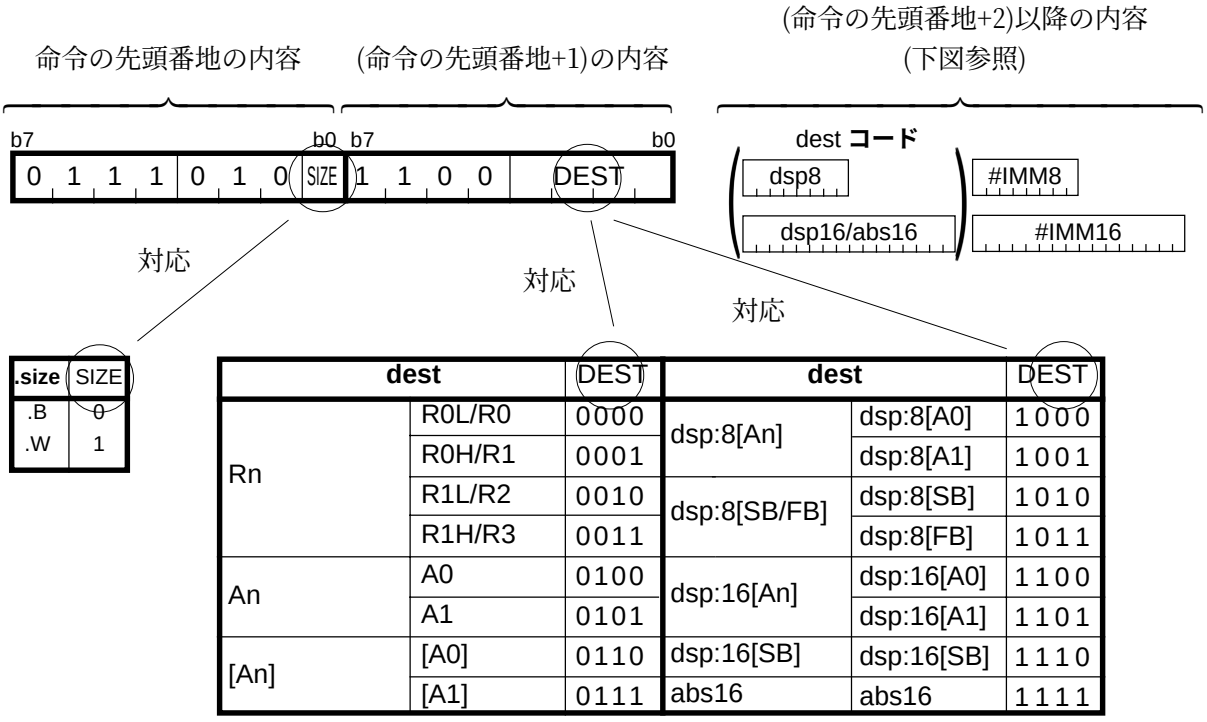
本ページで説明するニーモニックを示しています。

②構文

命令の構文を記号で示しています。

③命令コード

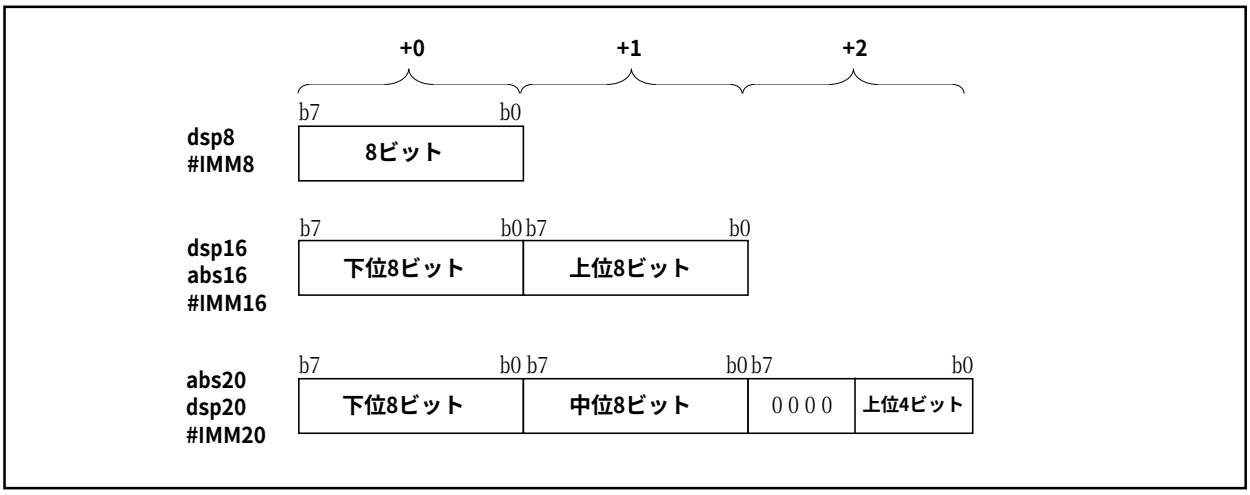
命令コードを示しています。（）内は選択する src/dest によって省略されます。



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0000	dsp:8[An]	dsp:8[A0]	1000
	R0H/R1	0001		dsp:8[A1]	1001
	R1L/R2	0010	dsp:8[SB/FB]	dsp:8[SB]	1010
	R1H/R3	0011		dsp:8[FB]	1011
An	A0	0100	dsp:16[An]	dsp:16[A0]	1100
	A1	0101		dsp:16[A1]	1101
[An]	[A0]	0110	dsp:16[SB]	dsp:16[SB]	1110
	[A1]	0111	abs16	abs16	1111

(命令の先頭番地 +2)以降は下記のとおり配置されます。

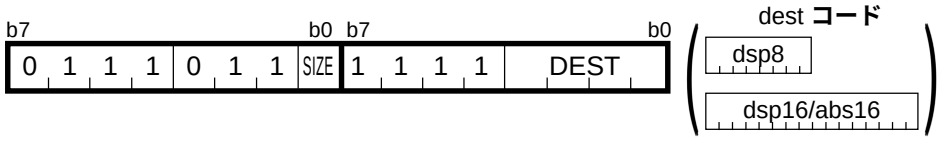


④バイト数／サイクル数表

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。
ただし、サイクル数は、ソフトウエアウエイト等の影響により増える可能性があります。
スラッシュの左側がバイト数、右側がサイクル数です。

ABS

(1) ABS.size dest



.size	SIZE
.B	0
.W	1

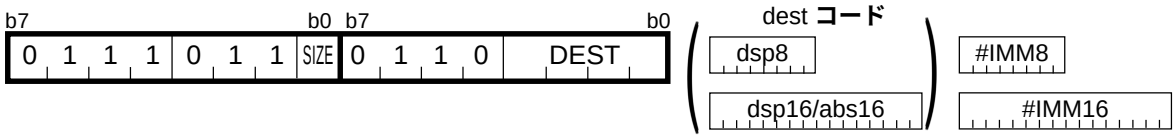
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/3	2/3	2/5	3/5	3/5	4/5	4/5	4/5

ADC

(1) ADC.size #IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

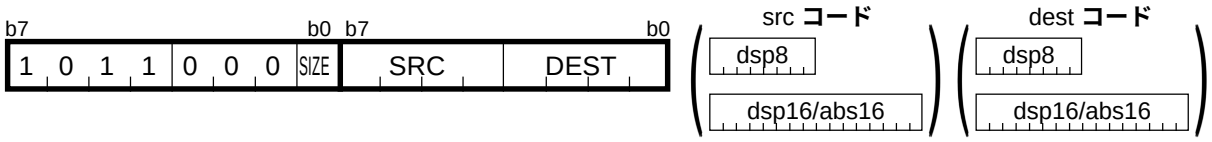
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

ADC

(2) ADC.size src, dest



.size	SIZE
.B	0
.W	1

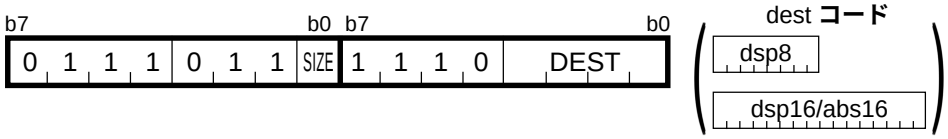
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

ADCF

(1) ADCF.size　　dest



.size	SIZE
.B	0
.W	1

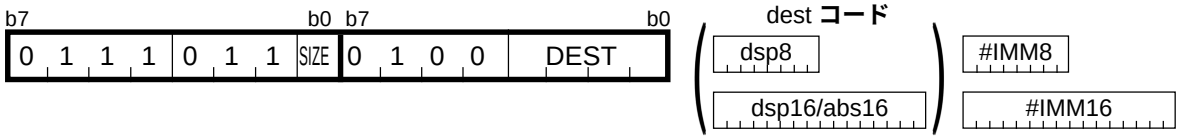
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

ADD

(1) ADD.size:G　　#IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

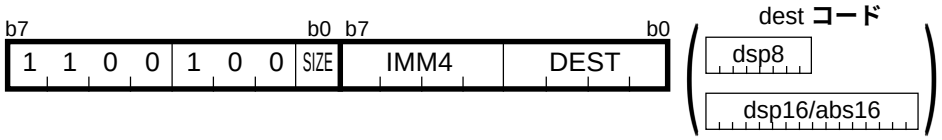
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は１バイト増加します。

ADD

(2) ADD.size:Q　#IMM, dest



.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
0	0 0 0 0	–8	1 0 0 0
+1	0 0 0 1	–7	1 0 0 1
+2	0 0 1 0	–6	1 0 1 0
+3	0 0 1 1	–5	1 0 1 1
+4	0 1 0 0	–4	1 1 0 0
+5	0 1 0 1	–3	1 1 0 1
+6	0 1 1 0	–2	1 1 1 0
+7	0 1 1 1	–1	1 1 1 1

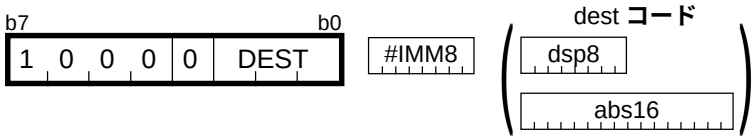
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

ADD

(3) ADD.B:S #IMM8, dest



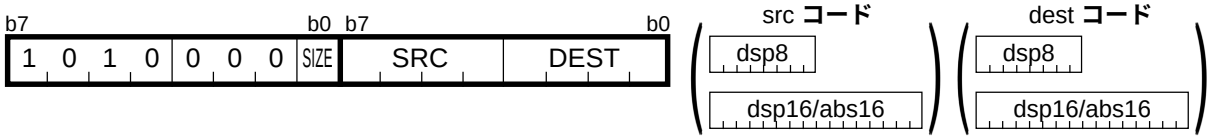
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	2/1	3/3	4/3

ADD

(4) ADD.size:G src, dest



.size	SIZE
.B	0
.W	1

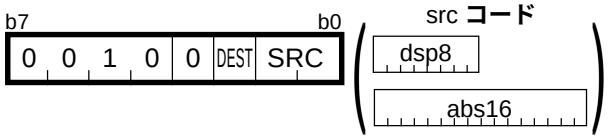
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

ADD

(5) ADD.B:S src, R0L/R0H



src		SRC
Rn	R0L/R0H	0　0
dsp:8[SB/FB]	dsp:8[SB]	0　1
	dsp:8[FB]	1　0
abs16	abs16	1　1

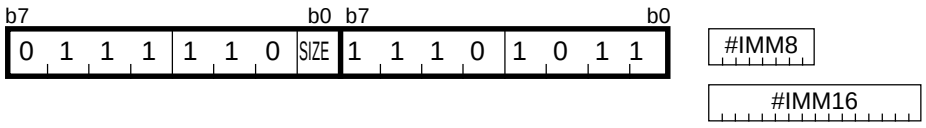
dest	DEST
R0L	0
R0H	1

【バイト数／サイクル数】

src	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/2	2/3	3/3

ADD

(6) ADD.size:G　#IMM, SP



.size	SIZE
.B	0
.W	1

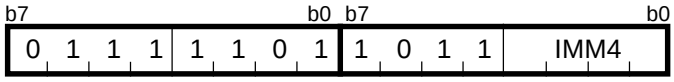
【バイト数／サイクル数】

バイト数／サイクル数	3/2

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

ADD

(7) ADD.size:Q　　#IMM, SP



*1　サイズ指定子(.size)に(.B)または(.W)のいずれを選択した場合でも、命令コードは同じです。

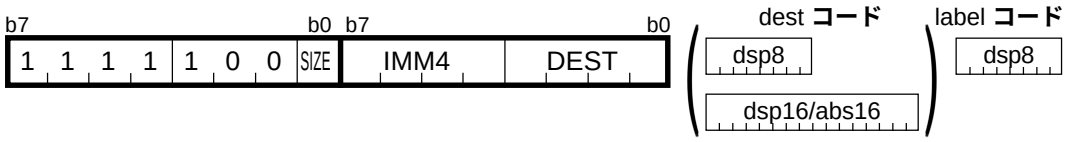
#IMM	IMM4	#IMM	IMM4
0	0 0 0 0	–8	1 0 0 0
+1	0 0 0 1	–7	1 0 0 1
+2	0 0 1 0	–6	1 0 1 0
+3	0 0 1 1	–5	1 0 1 1
+4	0 1 0 0	–4	1 1 0 0
+5	0 1 0 1	–3	1 1 0 1
+6	0 1 1 0	–2	1 1 1 0
+7	0 1 1 1	–1	1 1 1 1

【バイト数／サイクル数】

バイト数／サイクル数	2/1

ADJNZ

(1) ADJNZ.size #IMM, dest, label



dsp8(label コード) = label が示す番地－(命令の先頭番地 +2)

.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
0	0 0 0 0	–8	1 0 0 0
+1	0 0 0 1	–7	1 0 0 1
+2	0 0 1 0	–6	1 0 1 0
+3	0 0 1 1	–5	1 0 1 1
+4	0 1 0 0	–4	1 1 0 0
+5	0 1 0 1	–3	1 1 0 1
+6	0 1 1 0	–2	1 1 1 0
+7	0 1 1 1	–1	1 1 1 1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

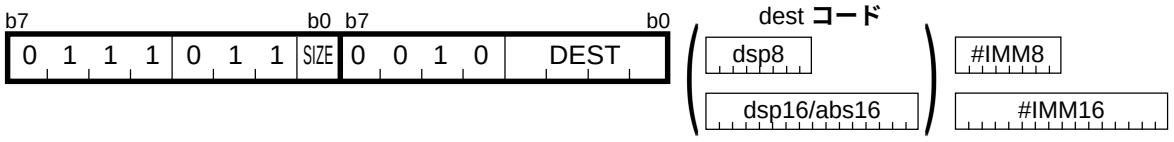
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/3	3/3	3/5	4/5	4/5	5/5	5/5	5/5

*1 label に分岐したとき、表中のサイクル数は 4 サイクル増加します。

AND

(1) AND.size:G　#IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

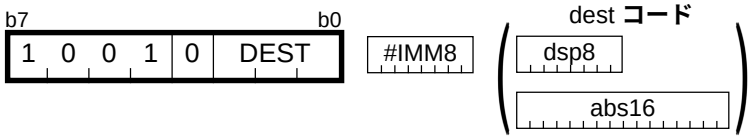
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

AND

(2) AND.B:S　#IMM8, dest



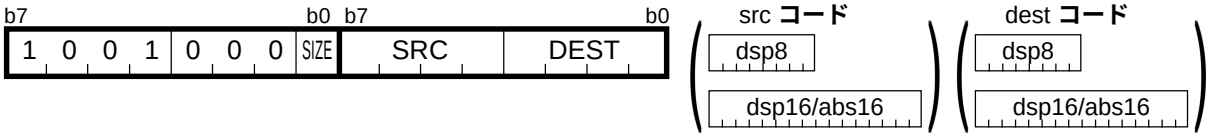
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	2/1	3/3	4/3

AND

(3) AND.size:G　src, dest



.size	SIZE
.B	0
.W	1

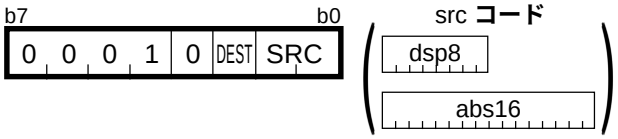
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

AND

(4) AND.B:S src, R0L/R0H



src		SRC
Rn	R0L/R0H	0　0
dsp:8[SB/FB]	dsp:8[SB]	0　1
	dsp:8[FB]	1　0
abs16	abs16	1　1

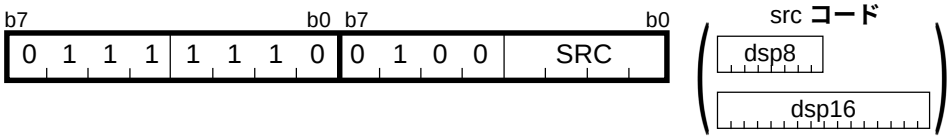
dest	DEST
R0L	0
R0H	1

【バイト数／サイクル数】

src	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/2	2/3	3/3

BAND

(1) BAND src



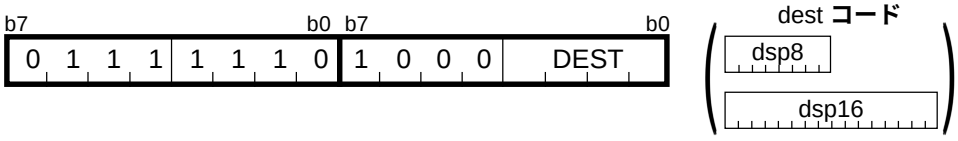
src		SRC	src		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BCLR

(1) BCLR:G dest



dest		DEST	dest		DEST
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

dest	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/2	3/2	2/6	3/6	3/3	4/6	4/3	4/3

BCLR

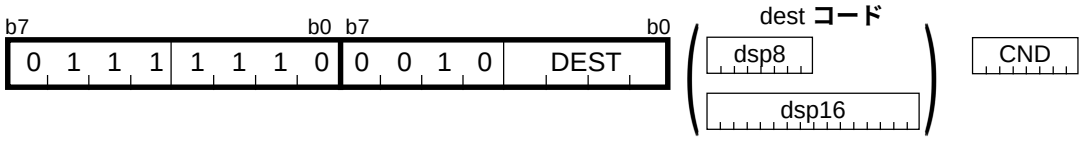
(2) BCLR:S　　bit, base:11[SB]

【バイト数／サイクル数】

バイト数／サイクル数	2/3

BM*Cnd*

(1) BM*Cnd* dest



dest		DEST	dest		DEST
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

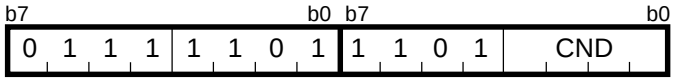
Cnd	CND	***Cnd***	CND
GEU/C	0 0 0 0 0 0 0 0	LTU/NC	1 1 1 1 1 0 0 0
GTU	0 0 0 0 0 0 0 1	LEU	1 1 1 1 1 0 0 1
EQ/Z	0 0 0 0 0 0 1 0	NE/NZ	1 1 1 1 1 0 1 0
N	0 0 0 0 0 0 1 1	PZ	1 1 1 1 1 0 1 1
LE	0 0 0 0 0 1 0 0	GT	1 1 1 1 1 1 0 0
O	0 0 0 0 0 1 0 1	NO	1 1 1 1 1 1 0 1
GE	0 0 0 0 0 1 1 0	LT	1 1 1 1 1 1 1 0

【バイト数／サイクル数】

dest	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	4/6	4/6	3/10	4/10	4/7	5/10	5/7	5/7

BM*Cnd*

(2) BM*Cnd*　　C



Cnd	CND	***Cnd***	CND
GEU/C	0 0 0 0	PZ	0 1 1 1
GTU	0 0 0 1	LE	1 0 0 0
EQ/Z	0 0 1 0	O	1 0 0 1
N	0 0 1 1	GE	1 0 1 0
LTU/NC	0 1 0 0	GT	1 1 0 0
LEU	0 1 0 1	NO	1 1 0 1
NE/NZ	0 1 1 0	LT	1 1 1 0

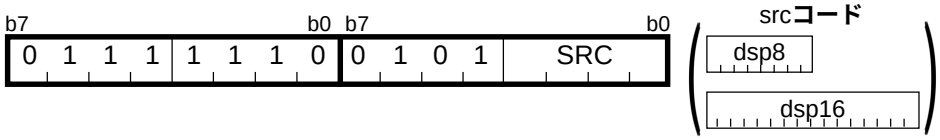
【バイト数／サイクル数】

バイト数／サイクル数	2/1

*1　条件が真のとき、表中のサイクル数は1サイクル増加します。

BNAND

(1) BNAND　　src



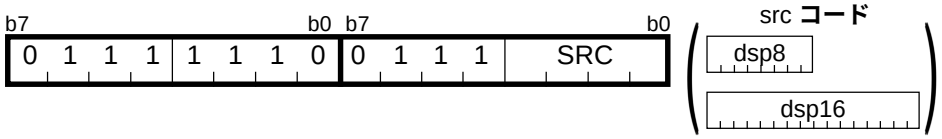
src		SRC	**src**		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BNOR

(1) BNOR src



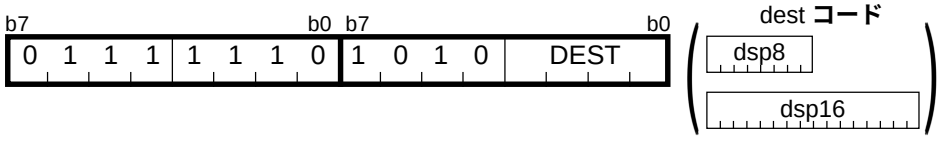
src		SRC	src		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BNOT

(1) BNOT:G dest



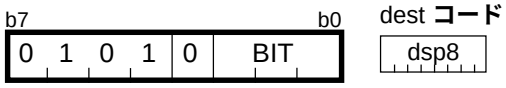
dest		DEST	dest		DEST
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

dest	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/2	3/2	2/6	3/6	3/3	4/6	4/3	4/3

BNOT

(2) BNOT:S　　bit, base:11[SB]

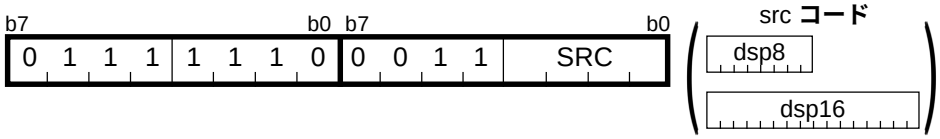


【バイト数／サイクル数】

バイト数／サイクル数	2/3

BNTST

(1) BNTST　　src



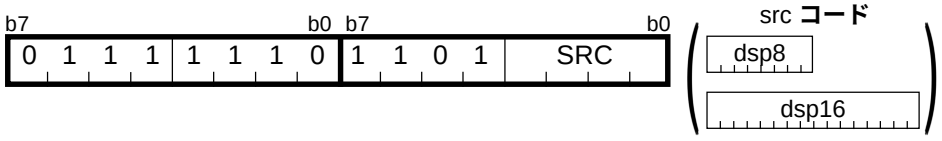
src		SRC	src		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BNXOR

(1) BNXOR src



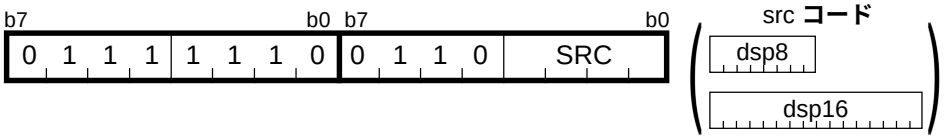
src		SRC	src		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BOR

(1) BOR src



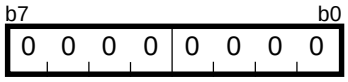
src		SRC	src		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BRK

(1) BRK



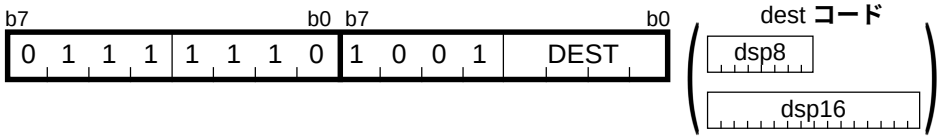
【バイト数／サイクル数】

バイト数／サイクル数	1/27

*1 BRK 割り込みの飛び先番地を割り込みテーブルレジスタ(INTB)によって指定する場合、表中のサイクル数は 2 サイクル増加します。このとき、FFFE4$_{16}$ 番地～ FFFE7$_{16}$ 番地には FF$_{16}$ を設定してください。

BSET

(1) BSET:G dest



dest		DEST	dest		DEST
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

dest	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/2	3/2	2/6	3/6	3/3	4/6	4/3	4/3

BSET

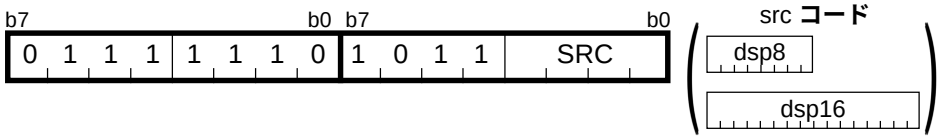
(2) BSET:S bit, base:11[SB]

【バイト数／サイクル数】

バイト数／サイクル数	2/3

BTST

(1) BTST:G src



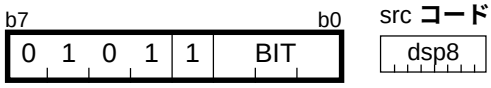
src		SRC	src		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/2	3/2	2/6	3/6	3/3	4/6	4/3	4/3

BTST

(2) BTST:S bit, base:11[SB]

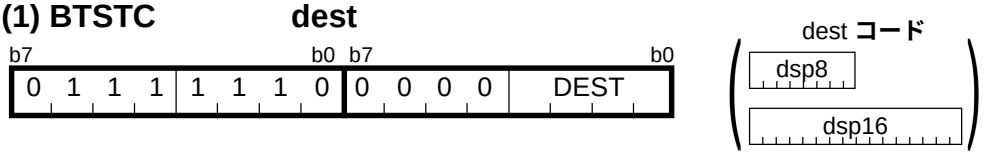


【バイト数／サイクル数】

バイト数／サイクル数	2/3

BTSTC

(1) BTSTC dest



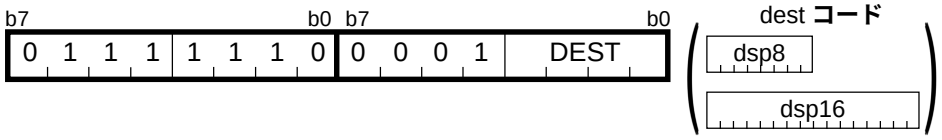
dest		DEST	dest		DEST
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

dest	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BTSTS

(1) BTSTS dest



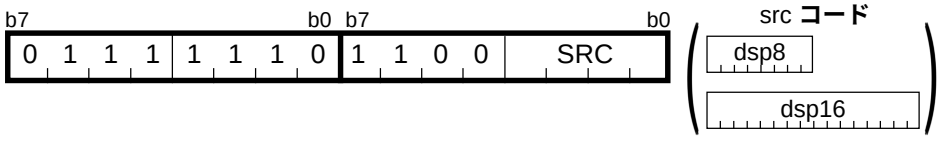
dest		DEST	dest		DEST
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

dest	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

BXOR

(1) BXOR src



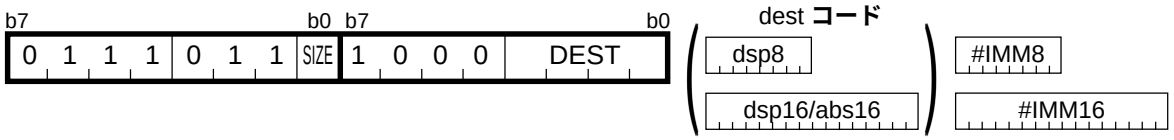
src		SRC	src		SRC
bit,Rn	bit,R0	0 0 0 0	base:8[An]	base:8[A0]	1 0 0 0
	bit,R1	0 0 0 1		base:8[A1]	1 0 0 1
	bit,R2	0 0 1 0	bit,base:8 [SB/FB]	bit,base:8[SB]	1 0 1 0
	bit,R3	0 0 1 1		bit,base:8[FB]	1 0 1 1
bit,An	bit,A0	0 1 0 0	base:16[An]	base:16[A0]	1 1 0 0
	bit,A1	0 1 0 1		base:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	bit,base:16[SB]	bit,base:16[SB]	1 1 1 0
	[A1]	0 1 1 1	bit,base:16	bit,base:16	1 1 1 1

【バイト数／サイクル数】

src	bit,Rn	bit,An	[An]	base:8 [An]	bit,base:8 [SB/FB]	base:16 [An]	bit,base:16 [SB]	bit,base:16
バイト数／サイクル数	3/3	3/3	2/7	3/7	3/4	4/7	4/4	4/4

CMP

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



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

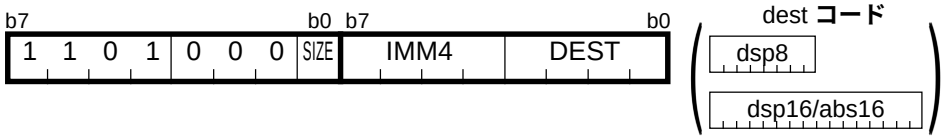
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

CMP

(2) CMP.size:Q　　#IMM, dest



.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
0	0 0 0 0	–8	1 0 0 0
+1	0 0 0 1	–7	1 0 0 1
+2	0 0 1 0	–6	1 0 1 0
+3	0 0 1 1	–5	1 0 1 1
+4	0 1 0 0	–4	1 1 0 0
+5	0 1 0 1	–3	1 1 0 1
+6	0 1 1 0	–2	1 1 1 0
+7	0 1 1 1	–1	1 1 1 1

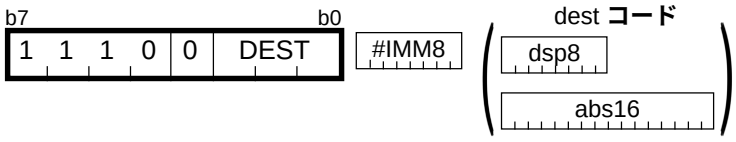
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

CMP

(3) CMP.B:S　　#IMM8, dest



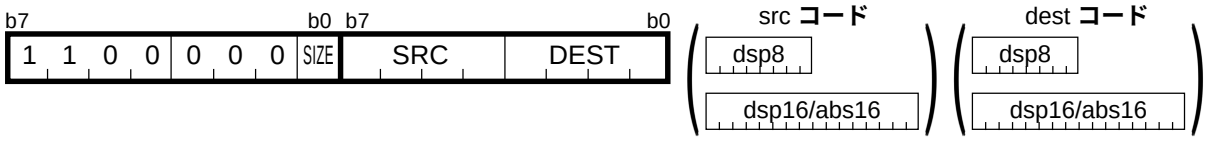
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	2/1	3/3	4/3

CMP

(4) CMP.size:G src, dest



.size	SIZE
.B	0
.W	1

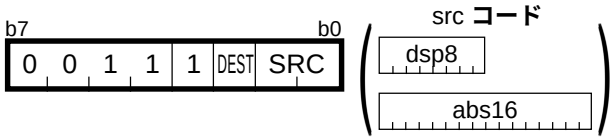
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

CMP

(5) CMP.B:S src, R0L/R0H



src		SRC
Rn	R0L/R0H	0 0
dsp:8[SB/FB]	dsp:8[SB]	0 1
	dsp:8[FB]	1 0
abs16	abs16	1 1

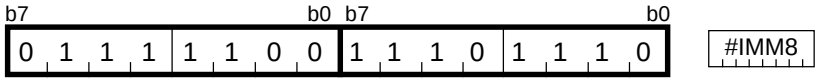
dest	DEST
R0L	0
R0H	1

【バイト数／サイクル数】

src	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/2	2/3	3/3

DADC

(1) DADC.B #IMM8, R0L

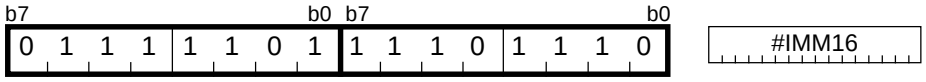


【バイト数／サイクル数】

バイト数／サイクル数	3/5

DADC

(2) DADC.W　　#IMM16, R0

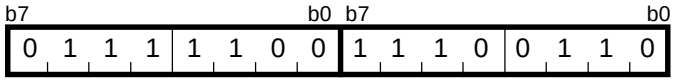


【バイト数／サイクル数】

バイト数／サイクル数	4/5

DADC

(3) DADC.B　　R0H, R0L

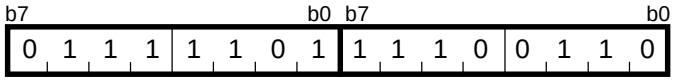


【バイト数／サイクル数】

バイト数／サイクル数	2/5

DADC

(4) DADC.W R1, R0

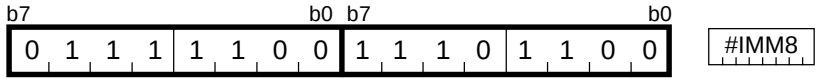


【バイト数／サイクル数】

バイト数／サイクル数	2/5

DADD

(1) DADD.B #IMM8, R0L



【バイト数／サイクル数】

バイト数／サイクル数	3/5

DADD

(2) DADD.W　　#IMM16, R0

b7							b0	b7							b0
0	1	1	1	1	1	0	1	1	1	1	0	1	1	0	0

#IMM16

【バイト数／サイクル数】

バイト数／サイクル数	4/5

DADD

(3) DADD.B　　R0H, R0L

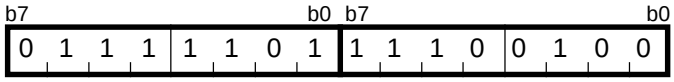
b7							b0	b7							b0
0	1	1	1	1	1	0	0	1	1	1	0	0	1	0	0

【バイト数／サイクル数】

バイト数／サイクル数	2/5

DADD

(4) DADD.W R1, R0

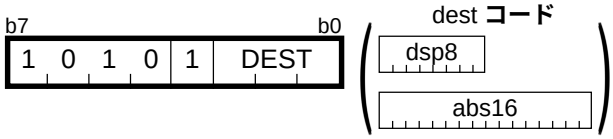


【バイト数／サイクル数】

バイト数／サイクル数	2/5

DEC

(1) DEC.B dest



dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/1	2/3	3/3

DEC

(2) DEC.W dest

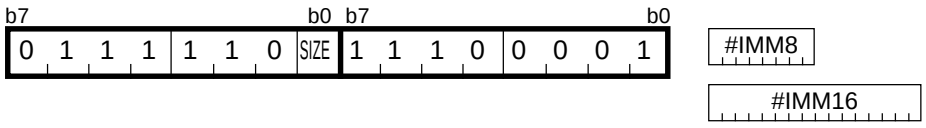
dest	DEST
A0	0
A1	1

【バイト数／サイクル数】

バイト数/サイクル数	1/1

DIV

(1) DIV.size #IMM



.size	SIZE
.B	0
.W	1

【バイト数／サイクル数】

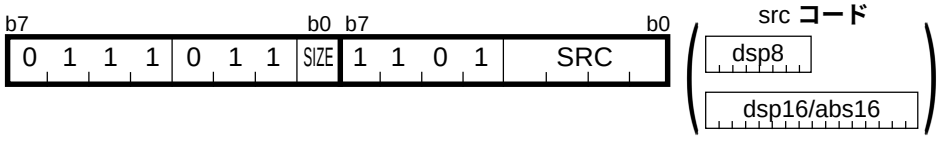
バイト数/サイクル数	3/22

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト、サイクル数は6サイクルそれぞれ増加します。

*2 オーバフロー発生時や除数、非除数の値によっては、サイクル数が減少する場合があります。

DIV

(2) DIV.size src



.size	SIZE
.B	0
.W	1

src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

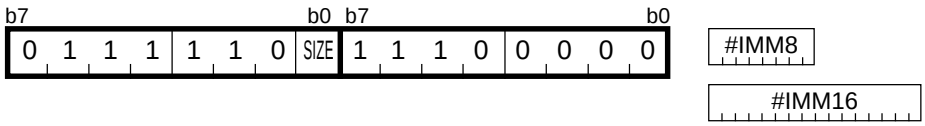
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/22	2/22	2/24	3/24	3/24	4/24	4/24	4/24

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は6サイクル増加します。
*2 オーバフロー発生時や除数、非除数の値によっては、サイクル数が減少する場合があります。

DIVU

(1) DIVU.size #IMM



.size	SIZE
.B	0
.W	1

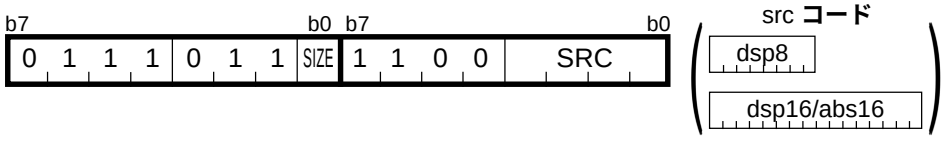
【バイト数／サイクル数】

バイト数／サイクル数	3/18

*2 オーバフロー発生時や除数、非除数の値によっては、サイクル数が減少する場合があります。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト、サイクル数は7サイクルそれぞれ増加します。

DIVU

(2) DIVU.size src



.size	SIZE
.B	0
.W	1

src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

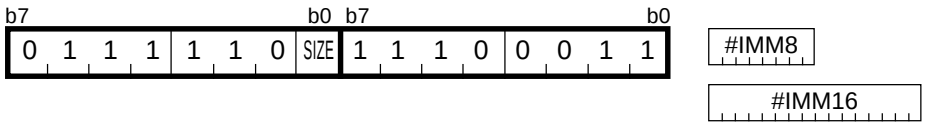
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/18	2/18	2/20	3/20	3/20	4/20	4/20	4/20

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は7サイクル増加します。
*2 オーバフロー発生時や除数、非除数の値によっては、サイクル数が減少する場合があります。

DIVX

(1) DIVX.size #IMM



.size	SIZE
.B	0
.W	1

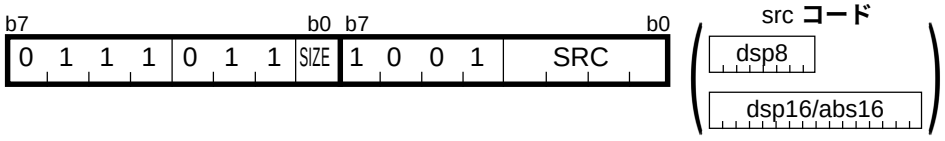
【バイト数／サイクル数】

バイト数／サイクル数	3/22

*2 オーバフロー発生時や除数、非除数の値によっては、サイクル数が減少する場合があります。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト、サイクル数は6サイクルそれぞれ増加します。

DIVX

(2) DIVX.size src



.size	SIZE
.B	0
.W	1

src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

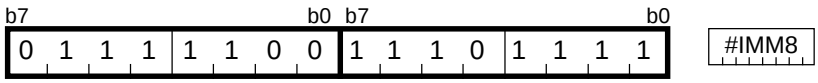
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/22	2/22	2/24	3/24	3/24	4/24	4/24	4/24

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は 6 サイクル増加します。

*2 オーバフロー発生時や除数、非除数の値によっては、サイクル数が減少する場合があります。

DSBB

(1) DSBB.B #IMM8, R0L

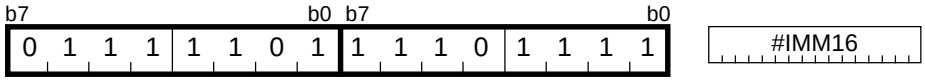


【バイト数／サイクル数】

バイト数／サイクル数	3/4

DSBB

(2) DSBB.W　　#IMM16, R0

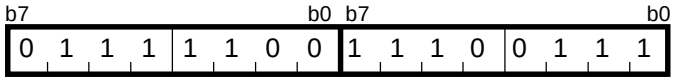


【バイト数／サイクル数】

バイト数／サイクル数	4/4

DSBB

(3) DSBB.B　　R0H, R0L

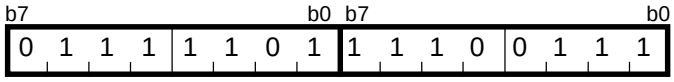


【バイト数／サイクル数】

バイト数／サイクル数	2/4

DSBB

(4) DSBB.W　　R1, R0

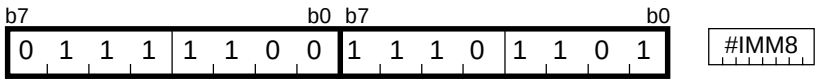


【バイト数／サイクル数】

バイト数／サイクル数	2/4

DSUB

(1) DSUB.B　　#IMM8, R0L

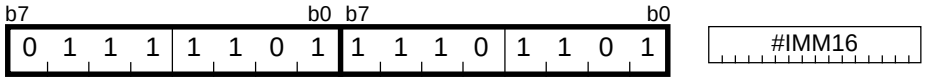


【バイト数／サイクル数】

バイト数／サイクル数	3/4

DSUB

(2) DSUB.W　　#IMM16, R0

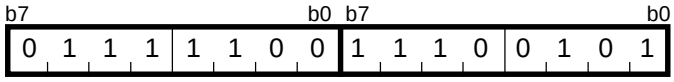


【バイト数／サイクル数】

バイト数／サイクル数	4/4

DSUB

(3) DSUB.B　　R0H, R0L

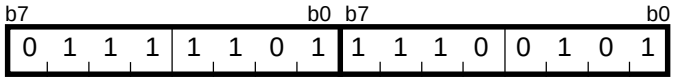


【バイト数／サイクル数】

バイト数／サイクル数	2/4

DSUB

(4) DSUB.W　　R1, R0

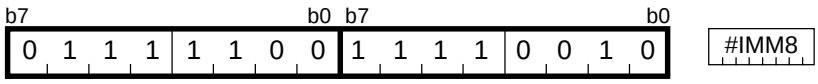


【バイト数／サイクル数】

バイト数／サイクル数	2/4

ENTER

(1) ENTER　　#IMM8

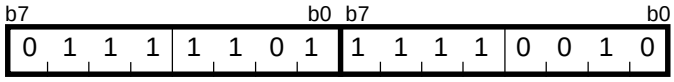


【バイト数／サイクル数】

バイト数／サイクル数	3/4

EXITD

(1) EXITD

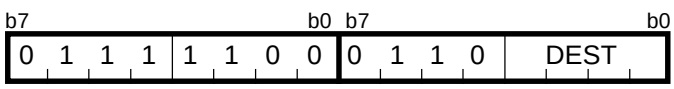


【バイト数／サイクル数】

バイト数／サイクル数	2/9

EXTS

(1) EXTS.B　　dest



dest		DEST	dest		DEST
Rn	R0L	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	---	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
---	---	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

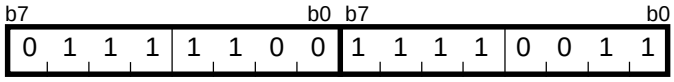
*1　--- 印は選択できません。

【バイト数／サイクル数】

dest	Rn	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/3	2/5	3/5	3/5	4/5	4/5	4/5

EXTS

(2) EXTS.W　　R0

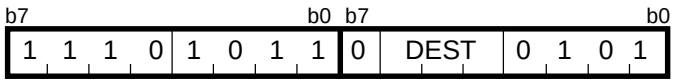


【バイト数／サイクル数】

バイト数／サイクル数	2/3

FCLR

(1) FCLR　　dest



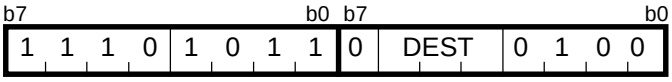
dest	DEST
C	0 0 0
D	0 0 1
Z	0 1 0
S	0 1 1
B	1 0 0
O	1 0 1
I	1 1 0
U	1 1 1

【バイト数／サイクル数】

バイト数／サイクル数	2/2

FSET

(1) FSET　　dest



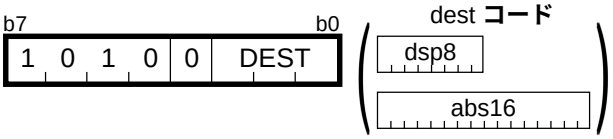
dest	DEST
C	0 0 0
D	0 0 1
Z	0 1 0
S	0 1 1
B	1 0 0
O	1 0 1
I	1 1 0
U	1 1 1

【バイト数／サイクル数】

バイト数／サイクル数	2/2

INC

(1) INC.B　　dest



dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/1	2/3	3/3

INC

(2) INC.W　　dest

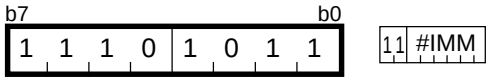
dest	DEST
A0	0
A1	1

【バイト数／サイクル数】

バイト数／サイクル数	1/1

INT

(1) INT　　#IMM

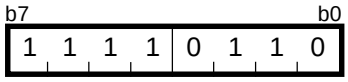


【バイト数／サイクル数】

バイト数／サイクル数	2/19

INTO

(1) INTO



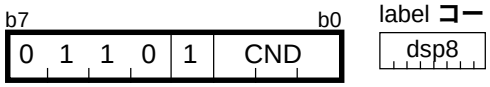
【バイト数／サイクル数】

バイト数／サイクル数	1/1

*1　O フラグが 1 のとき、表中のサイクル数は 19 サイクル増加します。

J*Cnd*

(1) J*Cnd*　　label



dsp8 = label が示す番地－(命令の先頭番地 +1)

Cnd	CND	*Cnd*	CND
GEU/C	0 0 0	LTU/NC	1 0 0
GTU	0 0 1	LEU	1 0 1
EQ/Z	0 1 0	NE/NZ	1 1 0
N	0 1 1	PZ	1 1 1

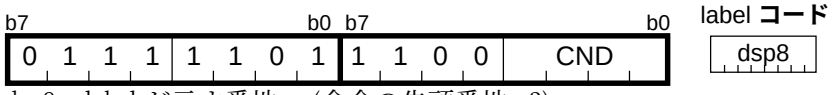
【バイト数／サイクル数】

バイト数／サイクル数	2/2

*2　label に分岐したとき、表中のサイクル数は 2 サイクル増加します。

J*Cnd*

(2) J*Cnd*　　label



label コード

dsp8 = label が示す番地－ (命令の先頭番地 +2)

Cnd	CND	*Cnd*	CND
LE	1 0 0 0	GT	1 1 0 0
O	1 0 0 1	NO	1 1 0 1
GE	1 0 1 0	LT	1 1 1 0

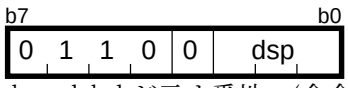
【バイト数／サイクル数】

バイト数／サイクル数	3/2

*1　label に分岐したとき、表中のサイクル数は 2 サイクル増加します。

JMP

(1) JMP.S　　label



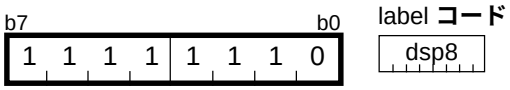
dsp = label が示す番地－(命令の先頭番地 + 2)

【バイト数／サイクル数】

バイト数／サイクル数	1/5

JMP

(2) JMP.B　　label



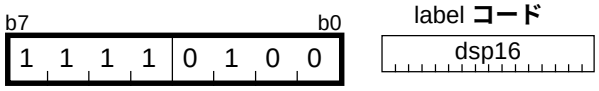
dsp8 = label が示す番地 －(命令の先頭番地 +1)

【バイト数／サイクル数】

バイト数／サイクル数	2/4

JMP

(3) JMP.W　　label



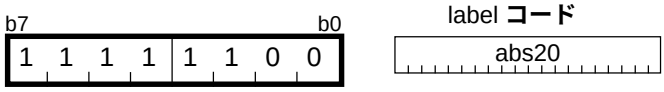
dsp16 = label が示す番地 －(命令の先頭番地 +1)

【バイト数／サイクル数】

バイト数／サイクル数	3/4

JMP

(4) JMP.A　　label

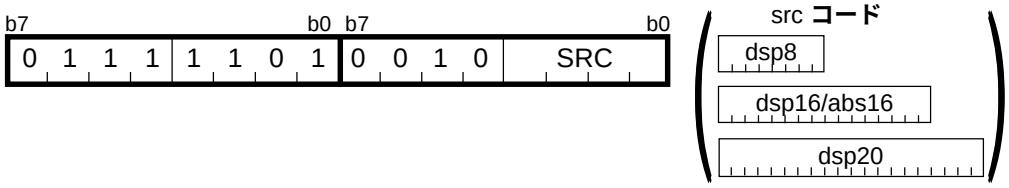


【バイト数／サイクル数】

バイト数／サイクル数	4/4

JMPI

(1) JMPI.W　　src



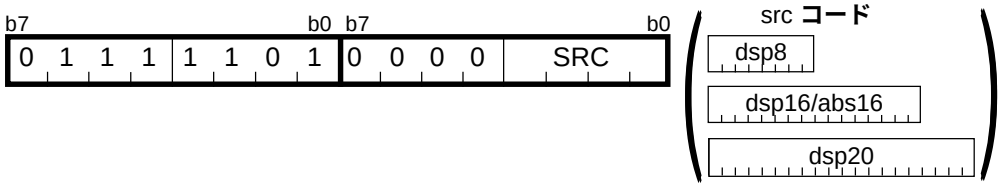
src		SRC	src		SRC
Rn	R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:20[An]	dsp:20[A0]	1 1 0 0
	A1	0 1 0 1		dsp:20[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:20[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/7	2/7	2/11	3/11	3/11	5/11	4/11	4/11

JMPI

(2) JMPI.A src



src		SRC	src		SRC
Rn	R2R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R3R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	---	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A1A0	0 1 0 0	dsp:20[An]	dsp:20[A0]	1 1 0 0
	---	0 1 0 1		dsp:20[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

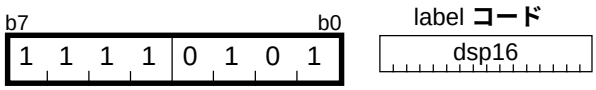
*1 --- 印は選択できません。

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:20[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/6	2/6	2/10	3/10	3/10	5/10	4/10	4/10

JSR

(1) JSR.W　　label



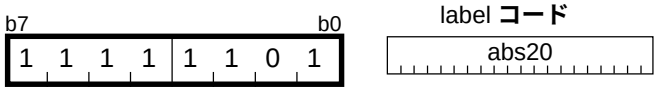
dsp16 = label が示す番地 －(命令の先頭番地 +1)

【バイト数／サイクル数】

バイト数／サイクル数	3/8

JSR

(2) JSR.A　　label

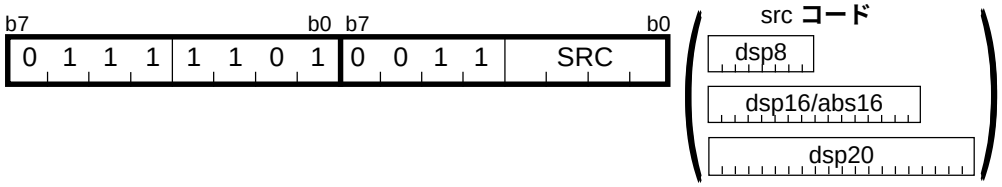


【バイト数／サイクル数】

バイト数／サイクル数	4/9

JSRI

(1) JSRI.W src



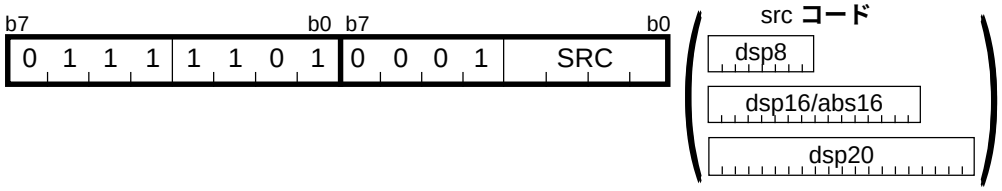
src		SRC	src		SRC
Rn	R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:20[An]	dsp:20[A0]	1 1 0 0
	A1	0 1 0 1		dsp:20[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:20[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/11	2/11	2/15	3/15	3/15	5/15	4/15	4/15

JSRI

(2) JSRI.A src



src		SRC	src		SRC
Rn	R2R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R3R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	---	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A1A0	0 1 0 0	dsp:20[An]	dsp:20[A0]	1 1 0 0
	---	0 1 0 1		dsp:20[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:20[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/11	2/11	2/15	3/15	3/15	5/15	4/15	4/15

LDC

(1) LDC #IMM16, dest

dest	DEST
---	0 0 0
INTBL	0 0 1
INTBH	0 1 0
FLG	0 1 1
ISP	1 0 0
SP	1 0 1
SB	1 1 0
FB	1 1 1

*1 --- 印は選択できません。

【バイト数／サイクル数】

バイト数／サイクル数	4/2

LDC

(2) LDC src, dest

src		SRC	src		SRC
Rn	R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

dest	DEST
---	0 0 0
INTBL	0 0 1
INTBH	0 1 0
FLG	0 1 1
ISP	1 0 0
SP	1 0 1
SB	1 1 0
FB	1 1 1

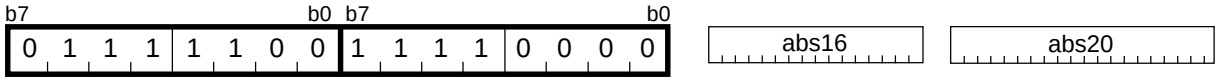
*1 --- 印は選択できません。

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

LDCTX

(1) LDCTX　　abs16, abs20



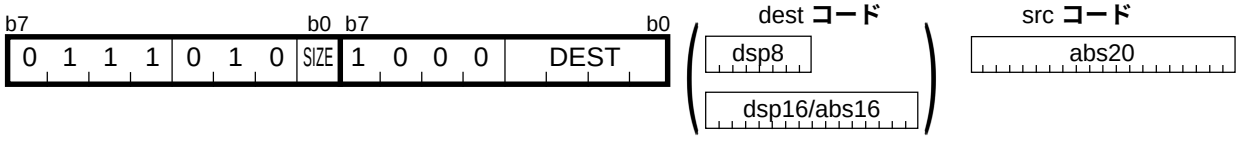
【バイト数／サイクル数】

バイト数／サイクル数	7/11+2×m

*2　m は転送回数です。

LDE

(1) LDE.size abs20, dest



.size	SIZE
.B	0
.W	1

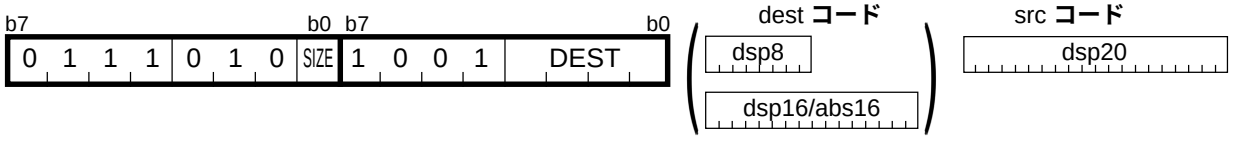
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	5/4	5/4	5/5	6/5	6/5	7/5	7/5	7/5

LDE

(2) LDE.size dsp:20[A0], dest



.size	SIZE
.B	0
.W	1

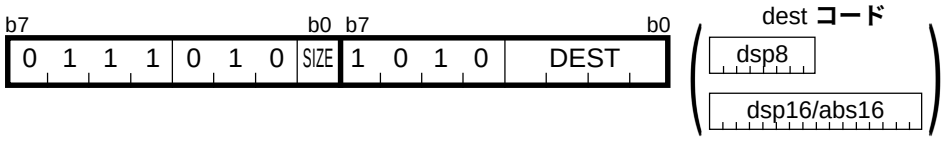
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	5/4	5/4	5/5	6/5	6/5	7/5	7/5	7/5

LDE

(3) LDE.size [A1A0], dest



.size	SIZE
.B	0
.W	1

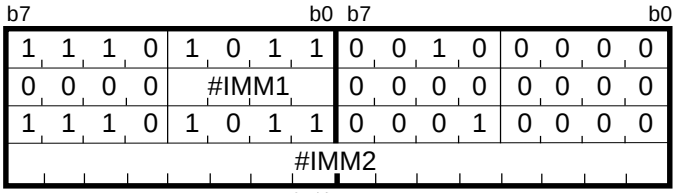
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/4	2/4	2/5	3/5	3/5	4/5	4/5	4/5

LDINTB

(1) LDINTB #IMM



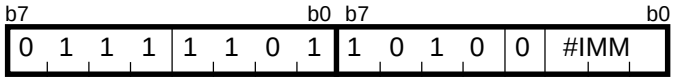
*1 #IMM1 は #IMM の上位 4 ビットです。
#IMM2 は #IMM の下位 16 ビットです。

【バイト数／サイクル数】

バイト数／サイクル数	8/4

LDIPL

(1) LDIPL　　#IMM

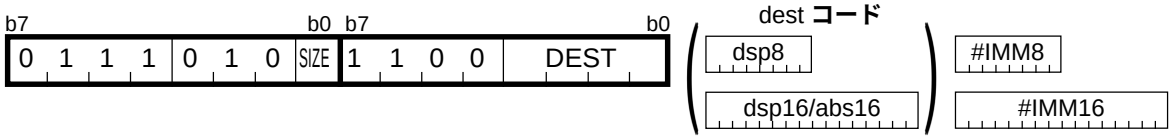


【バイト数／サイクル数】

バイト数／サイクル数	2/2

MOV

(1) MOV.size:G　　#IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

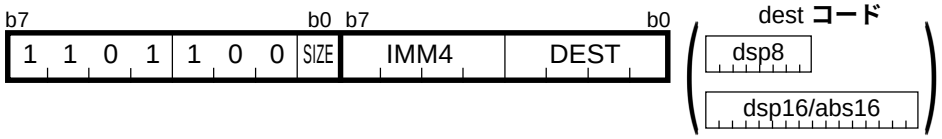
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/3	4/3	4/3	5/3	5/3	5/3

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

MOV

(2) MOV.size:Q #IMM, dest



.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
0	0 0 0 0	–8	1 0 0 0
+1	0 0 0 1	–7	1 0 0 1
+2	0 0 1 0	–6	1 0 1 0
+3	0 0 1 1	–5	1 0 1 1
+4	0 1 0 0	–4	1 1 0 0
+5	0 1 0 1	–3	1 1 0 1
+6	0 1 1 0	–2	1 1 1 0
+7	0 1 1 1	–1	1 1 1 1

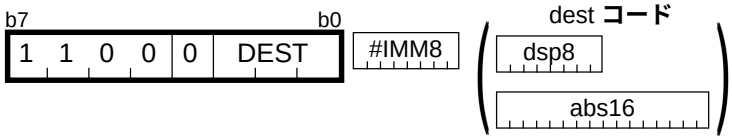
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/2	3/2	3/2	4/2	4/2	4/2

MOV

(3) MOV.B:S　　#IMM8, dest



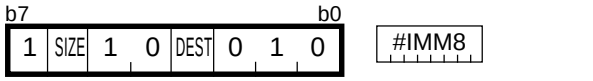
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	2/1	3/2	4/2

MOV

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



#IMM8

#IMM16

.size	SIZE
.B	1
.W	0

dest	DEST
A0	0
A1	1

【バイト数／サイクル数】

バイト数／サイクル数	2/1

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト、サイクル数は1サイクルそれぞれ増加します。

MOV

(5) MOV.B:Z #0, dest

dest コード

dsp8

abs16

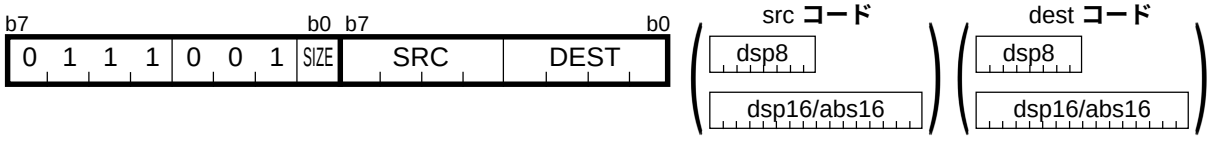
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/1	2/2	3/2

MOV

(6) MOV.size:G src, dest



.size	SIZE
.B	0
.W	1

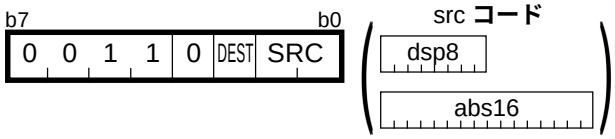
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/2	3/2	3/2	4/2	4/2	4/2
An	2/2	2/2	2/2	3/2	3/2	4/2	4/2	4/2
[An]	2/3	2/3	2/3	3/3	3/3	4/3	4/3	4/3
dsp:8[An]	3/3	3/3	3/3	4/3	4/3	5/3	5/3	5/3
dsp:8[SB/FB]	3/3	3/3	3/3	4/3	4/3	5/3	5/3	5/3
dsp:16[An]	4/3	4/3	4/3	5/3	5/3	6/3	6/3	6/3
dsp:16[SB]	4/3	4/3	4/3	5/3	5/3	6/3	6/3	6/3
abs16	4/3	4/3	4/3	5/3	5/3	6/3	6/3	6/3

MOV

(7) MOV.B:S src, dest



src		SRC
Rn	R0L/R0H	0 0
dsp:8[SB/FB]	dsp:8[SB]	0 1
	dsp:8[FB]	1 0
abs16	abs16	1 1

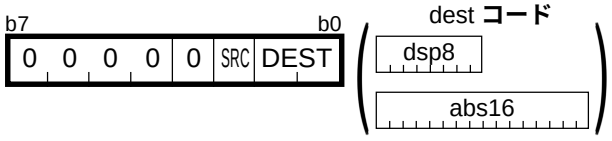
dest	DEST
A0	0
A1	1

【バイト数／サイクル数】

src	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/2	2/3	3/3

MOV

(8) MOV.B:S R0L/R0H, dest



src	SRC
R0L	0
R0H	1

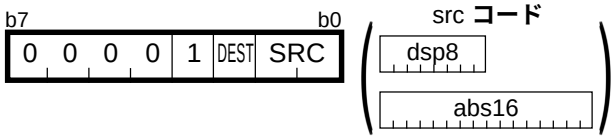
dest		DEST
dsp:8[SB/FB]	dsp:8[SB]	0 1
	dsp:8[FB]	1 0
abs16	abs16	1 1

【バイト数／サイクル数】

dest	dsp:8[SB/FB]	abs16
バイト数／サイクル数	2/2	3/2

MOV

(9) MOV.B:S　　src, R0L/R0H



src		SRC
Rn	R0L/R0H	0 0
dsp:8[SB/FB]	dsp:8[SB]	0 1
	dsp:8[FB]	1 0
abs16	abs16	1 1

dest	DEST
R0L	0
R0H	1

【バイト数／サイクル数】

src	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/2	2/3	3/3

MOV

(10) MOV.size:G　　dsp:8[SP], dest

.size	SIZE
.B	0
.W	1

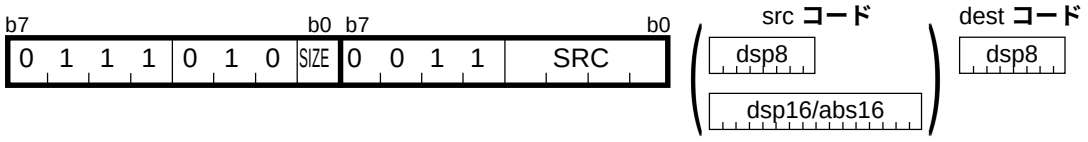
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/3	4/3	4/3	5/3	5/3	5/3

MOV

(11) MOV.size:G src, dsp:8[SP]



.size	SIZE
.B	0
.W	1

src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4

MOVA

(1) MOVA src, dest

src		SRC
dsp:8[An]	dsp:8[A0]	1 0 0 0
	dsp:8[A1]	1 0 0 1
dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	dsp:8[FB]	1 0 1 1
dsp:16[An]	dsp:16[A0]	1 1 0 0
	dsp:16[A1]	1 1 0 1
dsp:16[SB]	dsp:16[SB]	1 1 1 0
abs16	abs16	1 1 1 1

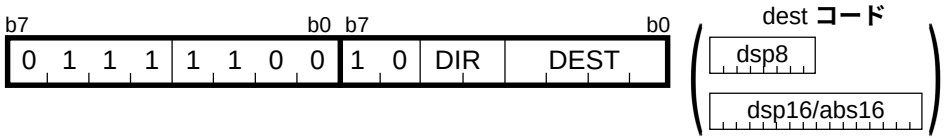
dest	DEST
R0	0 0 0
R1	0 0 1
R2	0 1 0
R3	0 1 1
A0	1 0 0
A1	1 0 1

【バイト数／サイクル数】

src	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	4/2	4/2	4/2

MOV*Dir*

(1) MOV*Dir*　　R0L, dest



Dir	DIR
LL	0 0
LH	1 0
HL	0 1
HH	1 1

dest		DEST	dest		DEST
Rn	---	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H	0 0 1 1		dsp:8[FB]	1 0 1 1
An	---	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

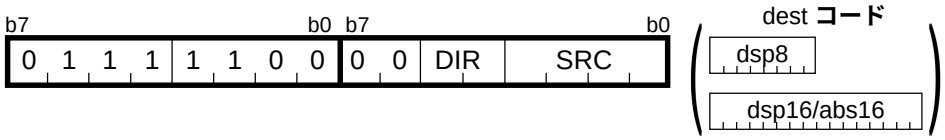
*1　--- 印は選択できません。

【バイト数／サイクル数】

dest	Rn	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
MOVHH, MOVLL	2/4	2/5	3/5	3/5	4/5	4/5	4/5
MOVHL, MOVLH	2/7	2/8	3/8	3/8	4/8	4/8	4/8

MOV*Dir*

(2) MOV*Dir*　　src, R0L



Dir	DIR
LL	0 0
LH	1 0
HL	0 1
HH	1 1

src		SRC	src		SRC
Rn	R0L	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H	0 0 1 1		dsp:8[FB]	1 0 1 1
An	---	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

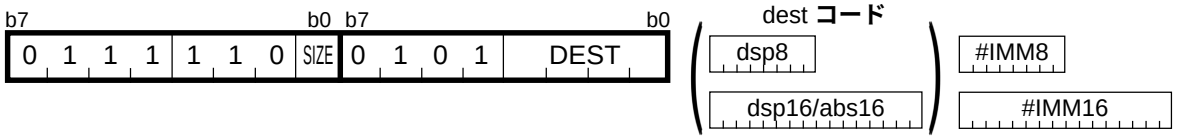
*1　--- 印は選択できません。

【バイト数／サイクル数】

src	Rn	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
MOVHH, MOVLL	2/3	2/5	3/5	3/5	4/5	4/5	4/5
MOVHL, MOVLH	2/6	2/8	3/8	3/8	4/8	4/8	4/8

MUL

(1) MUL.size #IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	--- /R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/---	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

【バイト数／サイクル数】

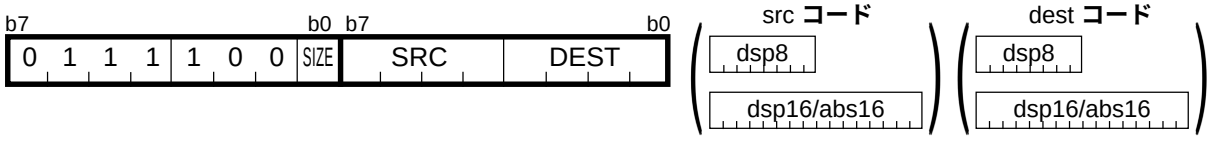
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/4	3/4	3/5	4/5	4/5	5/5	5/5	5/5

*2 サイズ指定子(.size)が(.W)の場合、dest が Rn、A n のとき、表中のバイト数は 1 バイト、サイクル数は 1 サイクルそれぞれ増加します。

*3 サイズ指定子(.size)が(.W)の場合、dest が Rn、A n 以外のとき、表中のバイト数は 1 バイト、サイクル数は 2 サイクルそれぞれ増加します。

MUL

(2) MUL.size src, dest



.size	SIZE
.B	0
.W	1

src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	--- /R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/---	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/4	2/4	2/5	3/5	3/5	4/5	4/5	4/5
An	2/4	2/5	2/5	3/5	3/5	4/5	4/5	4/5
[An]	2/6	2/6	2/6	3/6	3/6	4/6	4/6	4/6
dsp:8[An]	3/6	3/6	3/6	4/6	4/6	5/6	5/6	5/6
dsp:8[SB/FB]	3/6	3/6	3/6	4/6	4/6	5/6	5/6	5/6
dsp:16[An]	4/6	4/6	4/6	5/6	5/6	6/6	6/6	6/6
dsp:16[SB]	4/6	4/6	4/6	5/6	5/6	6/6	6/6	6/6
abs16	4/6	4/6	4/6	5/6	5/6	6/6	6/6	6/6

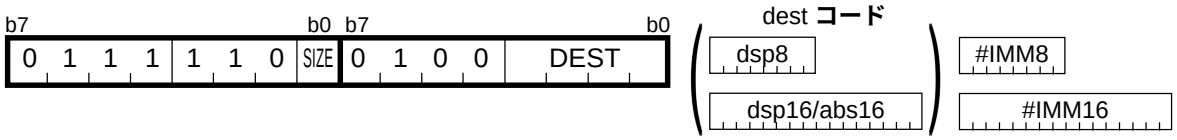
*2 サイズ指定子(.size)が(.W)の場合、src が An で dest が Rn のとき、表中のサイクル数は 1 サイクル増加します。

*3 サイズ指定子(.size)が(.W)の場合、src が An 以外で dest が Rn、An のとき、表中のサイクル数は 1 サイクル増加します。

*4 サイズ指定子(.size)が(.W)の場合、dest が Rn、An 以外のとき、表中のサイクル数は 2 サイクル増加します。

MULU

(1) MULU.size #IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	--- /R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/---	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

【バイト数／サイクル数】

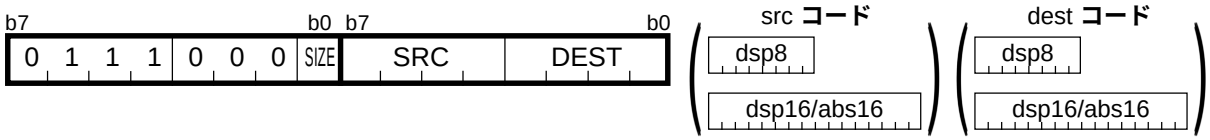
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/4	3/4	3/5	4/5	4/5	5/5	5/5	5/5

*2 サイズ指定子(.size)が(.W)の場合、dest が Rn、An のとき、表中のバイト数は 1 バイト、サイクル数は 1 サイクルそれぞれ増加します。

*3 サイズ指定子(.size)が(.W)の場合、dest が Rn、An 以外のとき、表中のバイト数は 1 バイト、サイクル数は 2 サイクルそれぞれ増加します。

MULU

(2) MULU.size src, dest



.size	SIZE
.B	0
.W	1

src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	--- /R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/---	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/4	2/4	2/5	3/5	3/5	4/5	4/5	4/5
An	2/4	2/5	2/5	3/5	3/5	4/5	4/5	4/5
[An]	2/6	2/6	2/6	3/6	3/6	4/6	4/6	4/6
dsp:8[An]	3/6	3/6	3/6	4/6	4/6	5/6	5/6	5/6
dsp:8[SB/FB]	3/6	3/6	3/6	4/6	4/6	5/6	5/6	5/6
dsp:16[An]	4/6	4/6	4/6	5/6	5/6	6/6	6/6	6/6
dsp:16[SB]	4/6	4/6	4/6	5/6	5/6	6/6	6/6	6/6
abs16	4/6	4/6	4/6	5/6	5/6	6/6	6/6	6/6

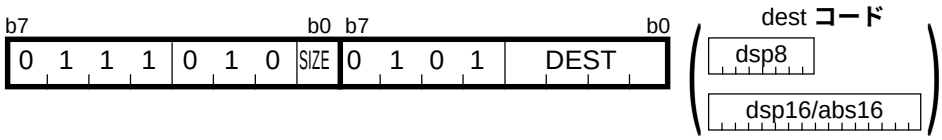
*2 サイズ指定子(.size)が(.W)の場合、src が An で dest が Rn のとき、表中のサイクル数は 1 サイクル増加します。

*3 サイズ指定子(.size)が(.W)の場合、src が An 以外で dest が Rn、An のとき、表中のサイクル数は 1 サイクル増加します。

*4 サイズ指定子(.size)が(.W)の場合、dest が Rn、An 以外のとき、表中のサイクル数は 2 サイクル増加します。

NEG

(1) NEG.size dest



.size	SIZE
.B	0
.W	1

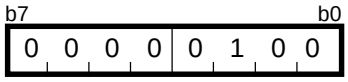
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

NOP

(1) NOP

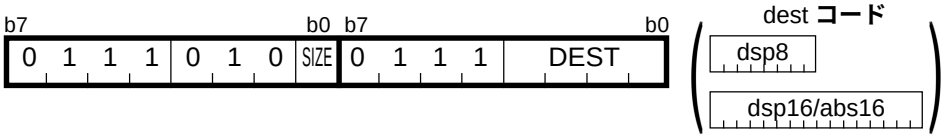


【バイト数／サイクル数】

バイト数／サイクル数	1/1

NOT

(1) NOT.size:G　　dest



.size	SIZE
.B	0
.W	1

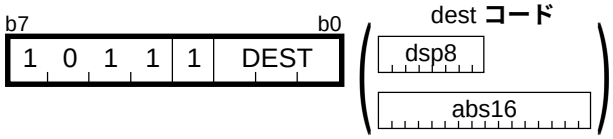
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

NOT

(2) NOT.B:S　　dest



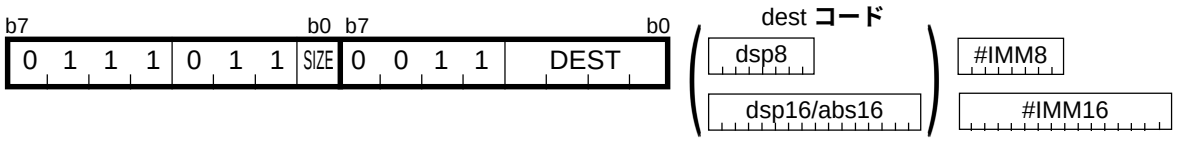
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/1	2/3	3/3

OR

(1) OR.size:G #IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

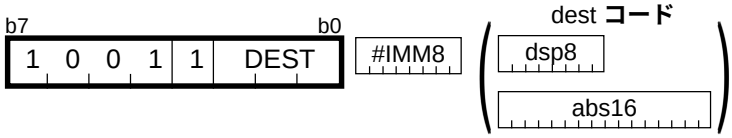
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

OR

(2) OR.B:S #IMM8, dest



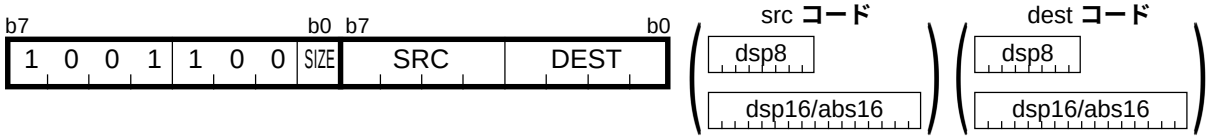
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	2/1	3/3	4/3

OR

(3) OR.size:G src, dest



.size	SIZE
.B	0
.W	1

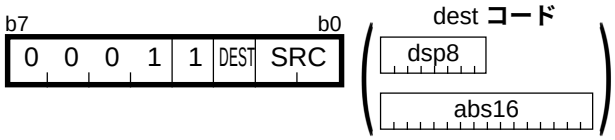
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

OR

(4) OR.B:S src, R0L/R0H



src		SRC
Rn	R0L/R0H	0 0
dsp:8[SB/FB]	dsp:8[SB]	0 1
	dsp:8[FB]	1 0
abs16	abs16	1 1

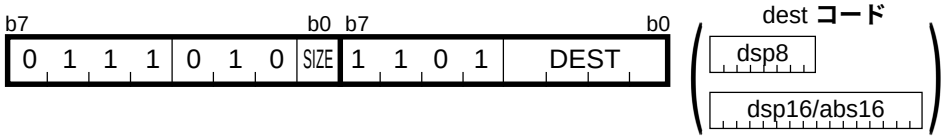
dest	DEST
R0L	0
R0H	1

【バイト数／サイクル数】

src	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/2	2/3	3/3

POP

(1) POP.size:G dest



.size	SIZE
.B	0
.W	1

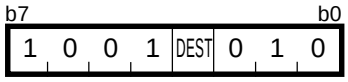
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4

POP

(2) POP.B:S　　dest



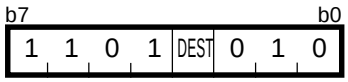
dest	DEST
R0L	0
R0H	1

【バイト数／サイクル数】

バイト数/サイクル数	1/3

POP

(3) POP.W:S　　dest



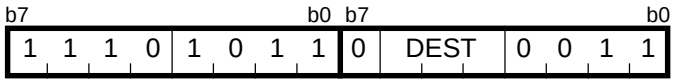
dest	DEST
A0	0
A1	1

【バイト数／サイクル数】

バイト数/サイクル数	1/3

POPC

(1) POPC　　dest



dest	DEST	dest	DEST
---	0 0 0	ISP	1 0 0
INTBL	0 0 1	SP	1 0 1
INTBH	0 1 0	SB	1 1 0
FLG	0 1 1	FB	1 1 1

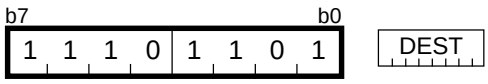
*1　--- 印は選択できません。

【バイト数／サイクル数】

バイト数／サイクル数	2/3

POPM

(1) POPM　　dest



dest							
FB	SB	A1	A0	R3	R2	R1	R0
DEST*2							

*2　選択されたレジスタに対応するビットは“1”です。
　　選択されていないレジスタに対応するビットは“0”です。

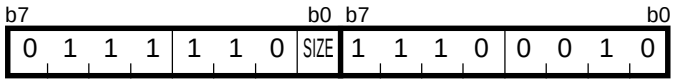
【バイト数／サイクル数】

バイト数／サイクル数	2/3

*3　復帰するレジスタが2つ以上の場合のサイクル数は2 × m(m:復帰するレジスタ数)となります。

PUSH

(1) PUSH.size:G #IMM



#IMM8

#IMM16

.size	SIZE
.B	0
.W	1

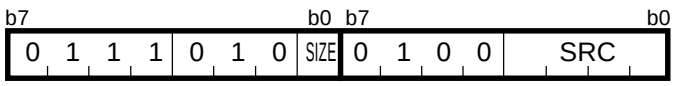
【バイト数／サイクル数】

バイト数／サイクル数	3/2

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

PUSH

(2) PUSH.size:G src



src コード

dsp8

dsp16/abs16

.size	SIZE
.B	0
.W	1

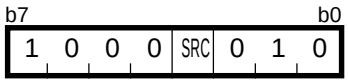
src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	4/4

PUSH

(3) PUSH.B:S　　src



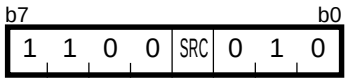
src	SRC
R0L	0
R0H	1

【バイト数／サイクル数】

バイト数／サイクル数	1/2

PUSH

(4) PUSH.W:S　　src



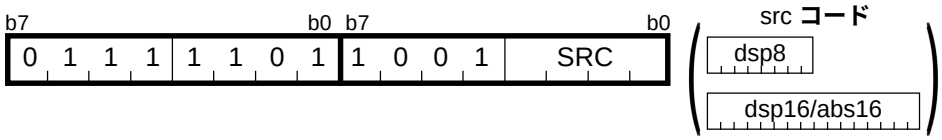
src	SRC
A0	0
A1	1

【バイト数／サイクル数】

バイト数／サイクル数	1/2

PUSHA

(1) PUSHA　　src



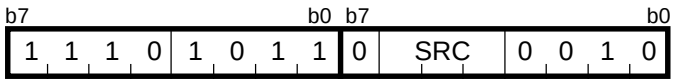
src		SRC
dsp:8[An]	dsp:8[A0]	1 0 0 0
	dsp:8[A1]	1 0 0 1
dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	dsp:8[FB]	1 0 1 1
dsp:16[An]	dsp:16[A0]	1 1 0 0
	dsp:16[A1]	1 1 0 1
dsp:16[SB]	dsp:16[SB]	1 1 1 0
abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs:16
バイト数／サイクル数	3/2	3/2	4/2	4/2	4/2

PUSHC

(1) PUSHC　　src



src	SRC	src	SRC
---	0 0 0	ISP	1 0 0
INTBL	0 0 1	SP	1 0 1
INTBH	0 1 0	SB	1 1 0
FLG	0 1 1	FB	1 1 1

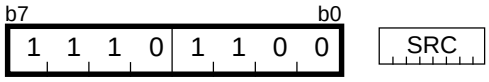
*1　--- 印は選択できません。

【バイト数／サイクル数】

バイト数／サイクル数	2/2

PUSHM

(1) PUSHM　　src



src							
R0	R1	R2	R3	A0	A1	SB	FB
SRC*1							

*1　選択されたレジスタに対応するビットは“1”です。
　　選択されていないレジスタに対応するビットは“0”です。

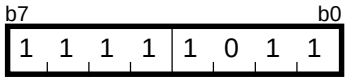
【バイト数／サイクル数】

バイト数／サイクル数	2/2×m

*2　mは退避するレジスタ数です。

REIT

(1) REIT

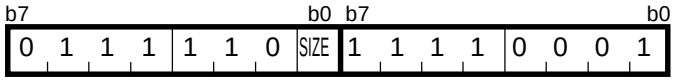


【バイト数／サイクル数】

バイト数／サイクル数	1/6

RMPA

(1) RMPA.size



.size	SIZE
.B	0
.W	1

【バイト数／サイクル数】

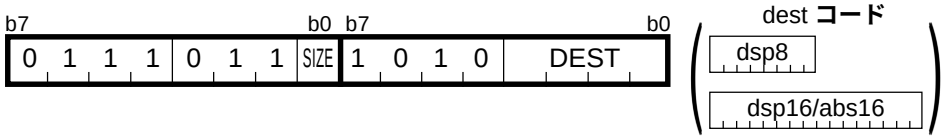
バイト数／サイクル数	2/4+7×m

*1 m は演算回数です。

*2 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は(6+9 × m)サイクルになります。

ROLC

(1) ROLC.size dest



.size	SIZE
.B	0
.W	1

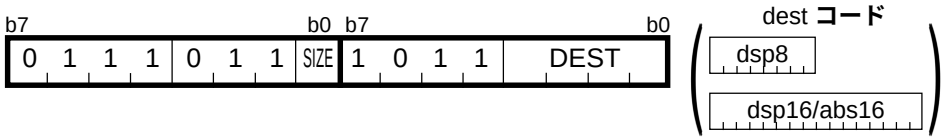
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

RORC

(1) RORC.size dest



.size	SIZE
.B	0
.W	1

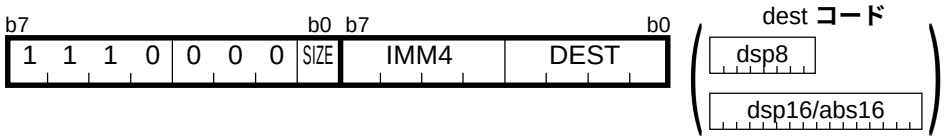
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	4/3

ROT

(1) ROT.size　　#IMM, dest



.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
+1	0 0 0 0	–1	1 0 0 0
+2	0 0 0 1	–2	1 0 0 1
+3	0 0 1 0	–3	1 0 1 0
+4	0 0 1 1	–4	1 0 1 1
+5	0 1 0 0	–5	1 1 0 0
+6	0 1 0 1	–6	1 1 0 1
+7	0 1 1 0	–7	1 1 1 0
+8	0 1 1 1	–8	1 1 1 1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

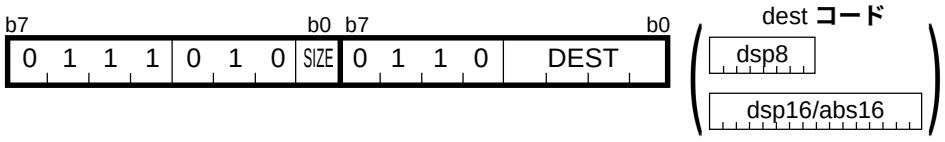
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1+m	2/1+m	2/2+m	3/2+m	3/2+m	4/2+m	4/2+m	4/2+m

*1　mは回転回数です。

ROT

(2) ROT.size R1H, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/---	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	--- /R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

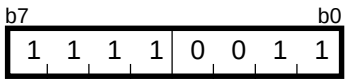
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/2+m	2/2+m	2/3+m	3/3+m	3/3+m	4/3+m	4/3+m	4/3+m

*2 mは回転回数です。

RTS

(1) RTS

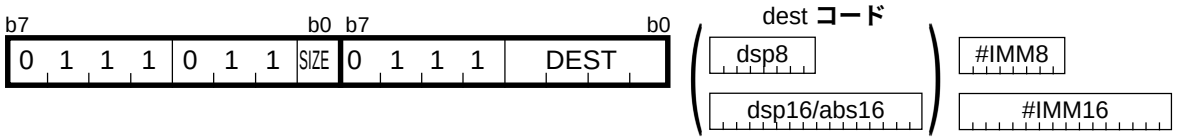


【バイト数／サイクル数】

バイト数／サイクル数	1/6

SBB

(1) SBB.size #IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

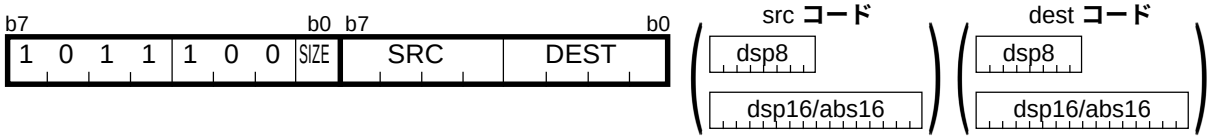
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

SBB

(2) SBB.size src, dest



.size	SIZE
.B	0
.W	1

src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

SBJNZ

(1) SBJNZ.size #IMM, dest, label

dsp8(label コード) = label が示す番地－(命令の先頭番地 +2)

.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
0	0 0 0 0	+8	1 0 0 0
–1	0 0 0 1	+7	1 0 0 1
–2	0 0 1 0	+6	1 0 1 0
–3	0 0 1 1	+5	1 0 1 1
–4	0 1 0 0	+4	1 1 0 0
–5	0 1 0 1	+3	1 1 0 1
–6	0 1 1 0	+2	1 1 1 0
–7	0 1 1 1	+1	1 1 1 1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

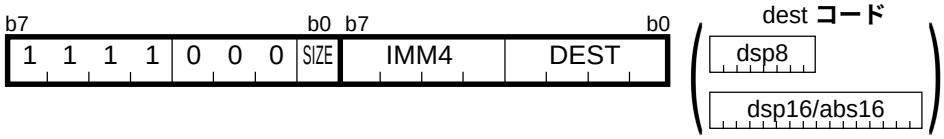
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/3	3/3	3/5	4/5	4/5	5/5	5/5	5/5

*1 label に分岐したとき、表中のサイクル数は 4 サイクル増加します。

SHA

(1) SHA.size　　#IMM, dest



.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
+1	0 0 0 0	–1	1 0 0 0
+2	0 0 0 1	–2	1 0 0 1
+3	0 0 1 0	–3	1 0 1 0
+4	0 0 1 1	–4	1 0 1 1
+5	0 1 0 0	–5	1 1 0 0
+6	0 1 0 1	–6	1 1 0 1
+7	0 1 1 0	–7	1 1 1 0
+8	0 1 1 1	–8	1 1 1 1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

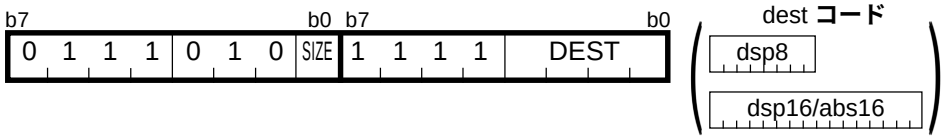
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1+m	2/1+m	2/2+m	3/2+m	3/2+m	4/2+m	4/2+m	4/2+m

*1　m はシフト回数です。

SHA

(2) SHA.size R1H, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/---	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	--- /R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

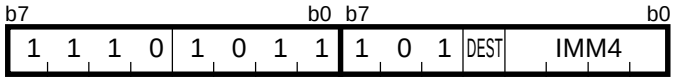
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/2+m	2/2+m	2/3+m	3/3+m	3/3+m	4/3+m	4/3+m	4/3+m

*2 m はシフト回数です。

SHA

(3) SHA.L #IMM, dest



#IMM	IMM4	#IMM	IMM4
+1	0 0 0 0	–1	1 0 0 0
+2	0 0 0 1	–2	1 0 0 1
+3	0 0 1 0	–3	1 0 1 0
+4	0 0 1 1	–4	1 0 1 1
+5	0 1 0 0	–5	1 1 0 0
+6	0 1 0 1	–6	1 1 0 1
+7	0 1 1 0	–7	1 1 1 0
+8	0 1 1 1	–8	1 1 1 1

dest	DEST
R2R0	0
R3R1	1

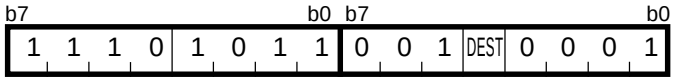
【バイト数／サイクル数】

バイト数／サイクル数	2/3+m

*2 m はシフト回数です。

SHA

(4) SHA.L　　R1H, dest



dest	DEST
R2R0	0
R3R1	1

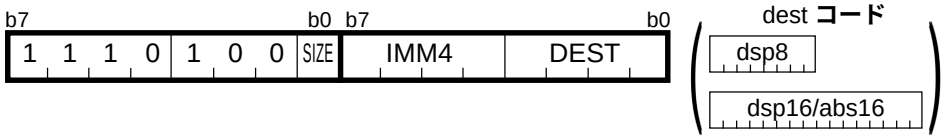
【バイト数／サイクル数】

バイト数／サイクル数	2/4+m

*1　m はシフト回数です。

SHL

(1) SHL.size #IMM, dest



.size	SIZE
.B	0
.W	1

#IMM	IMM4	#IMM	IMM4
+1	0 0 0 0	–1	1 0 0 0
+2	0 0 0 1	–2	1 0 0 1
+3	0 0 1 0	–3	1 0 1 0
+4	0 0 1 1	–4	1 0 1 1
+5	0 1 0 0	–5	1 1 0 0
+6	0 1 0 1	–6	1 1 0 1
+7	0 1 1 0	–7	1 1 1 0
+8	0 1 1 1	–8	1 1 1 1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

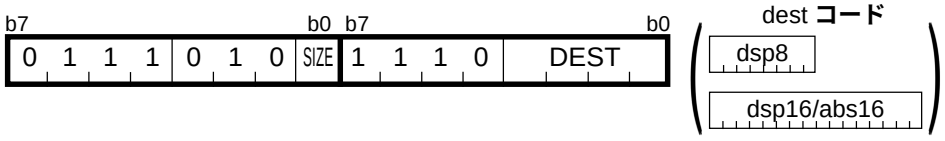
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1+m	2/1+m	2/2+m	3/2+m	3/2+m	4/2+m	4/2+m	4/2+m

*1 m はシフト回数です。

SHL

(2) SHL.size R1H, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/---	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	--- /R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

*1 --- 印は選択できません。

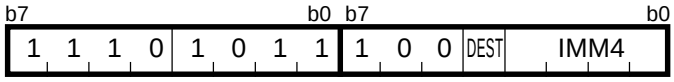
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/2+m	2/2+m	2/3+m	3/3+m	3/3+m	4/3+m	4/3+m	4/3+m

*2 m はシフト回数です。

SHL

(3) SHL.L #IMM, dest



#IMM	IMM4	#IMM	IMM4
+1	0 0 0 0	–1	1 0 0 0
+2	0 0 0 1	–2	1 0 0 1
+3	0 0 1 0	–3	1 0 1 0
+4	0 0 1 1	–4	1 0 1 1
+5	0 1 0 0	–5	1 1 0 0
+6	0 1 0 1	–6	1 1 0 1
+7	0 1 1 0	–7	1 1 1 0
+8	0 1 1 1	–8	1 1 1 1

dest	DEST
R2R0	0
R3R1	1

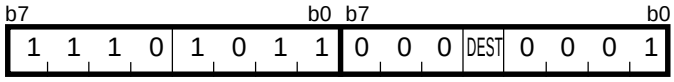
【バイト数／サイクル数】

バイト数／サイクル数	2/3+m

*2 m はシフト回数です。

SHL

(4) SHL.L　　R1H, dest



dest	DEST
R2R0	0
R3R1	1

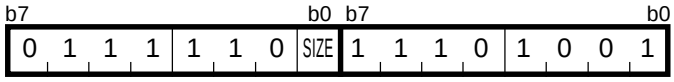
【バイト数／サイクル数】

バイト数/サイクル数	2/4+m

*1　m はシフト回数です。

SMOVB

(1) SMOVB.size



.size	SIZE
.B	0
.W	1

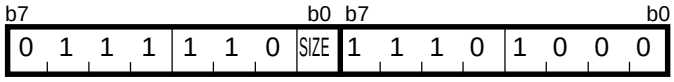
【バイト数／サイクル数】

バイト数/サイクル数	2/5+5×m

*2　m は転送回数です。

SMOVF

(1) SMOVF.size



.size	SIZE
.B	0
.W	1

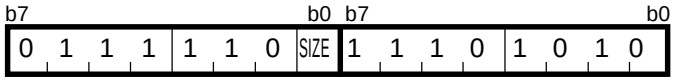
【バイト数／サイクル数】

バイト数/サイクル数	2/5+5×m

*1　m は転送回数です。

SSTR

(1) SSTR.size



.size	SIZE
.B	0
.W	1

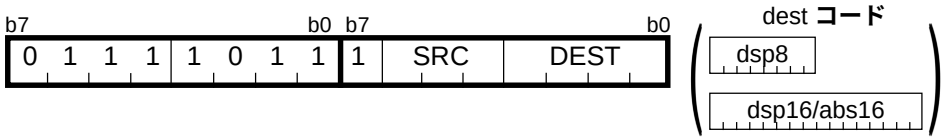
【バイト数／サイクル数】

バイト数/サイクル数	2/3+2×m

*1　m は転送回数です。

STC

(1) STC src, dest



src	SRC
---	0 0 0
INTBL	0 0 1
INTBH	0 1 0
FLG	0 1 1
ISP	1 0 0
SP	1 0 1
SB	1 1 0
FB	1 1 1

dest		DEST	dest		DEST
Rn	R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

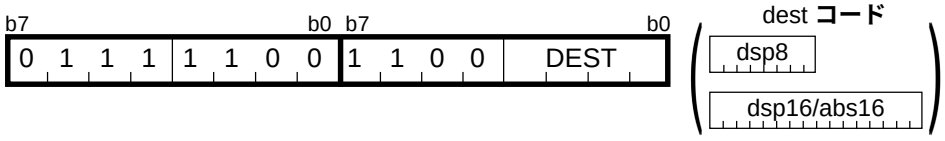
*1 --- 印は選択できません。

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/1	2/1	2/2	3/2	3/2	4/2	4/2	4/2

STC

(2) STC PC, dest



dest		DEST	dest		DEST
Rn	R2R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R3R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	---	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	---	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A1A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	---	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

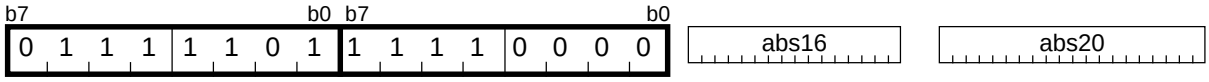
*1 --- 印は選択できません。

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3

STCTX

(1) STCTX　　abs16, abs20



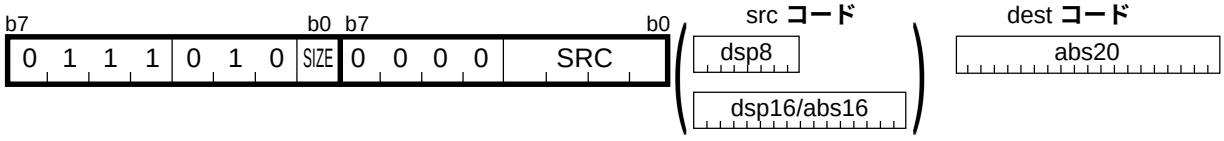
【バイト数／サイクル数】

バイト数/サイクル数	7/11+2×m

*1　mは転送回数です。

STE

(1) STE.size　　src, abs20



.size	SIZE
.B	0
.W	1

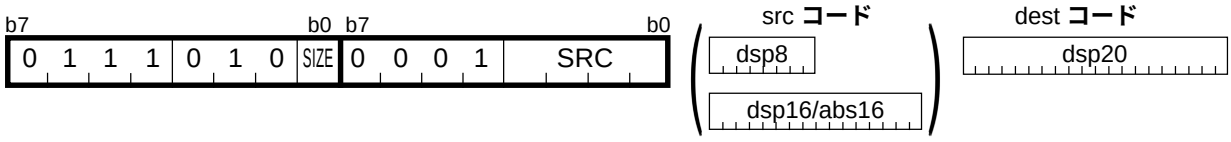
src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数/サイクル数	5/3	5/3	5/4	6/4	6/4	7/4	7/4	7/4

STE

(2) STE.size src, dsp:20[A0]



.size	SIZE
.B	0
.W	1

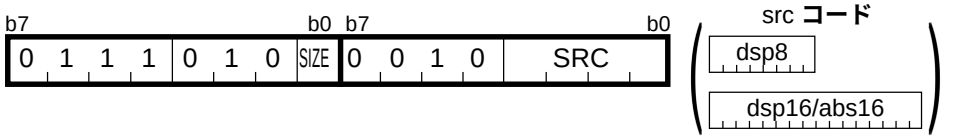
src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	5/3	5/3	5/4	6/4	6/4	7/4	7/4	7/4

STE

(3) STE.size src, [A1A0]



.size	SIZE
.B	0
.W	1

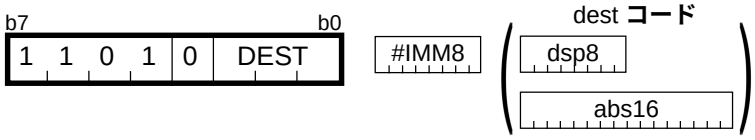
src		SRC	src		SRC
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4

STNZ

(1) STNZ　　#IMM8, dest



dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

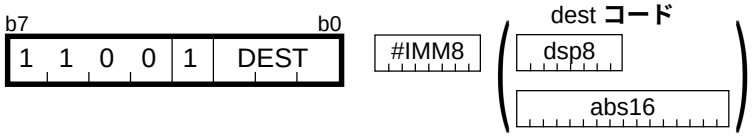
【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/2	4/2

*1　Ｚフラグが“０”のとき、表中のサイクル数は１サイクル増加します。

STZ

(1) STZ　　#IMM8, dest



dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/2	4/2

*2　Ｚフラグが“１”のとき、表中のサイクル数は１サイクル増加します。

STZX

(1) STZX #IMM81, #IMM82, dest

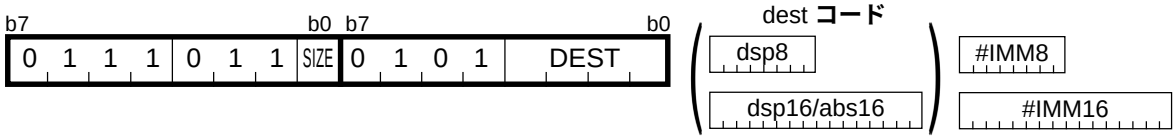
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	3/2	4/3	5/3

SUB

(1) SUB.size:G #IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

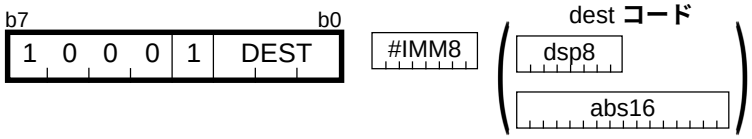
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

SUB

(2) SUB.B:S　　#IMM8, dest



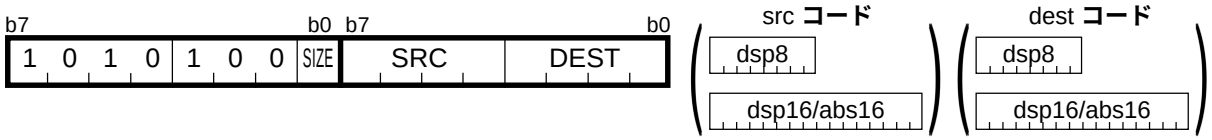
dest		DEST
Rn	R0H	0 1 1
	R0L	1 0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0 1
	dsp:8[FB]	1 1 0
abs16	abs16	1 1 1

【バイト数／サイクル数】

dest	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	2/1	3/3	4/3

SUB

(3) SUB.size:G src, dest



.size	SIZE
.B	0
.W	1

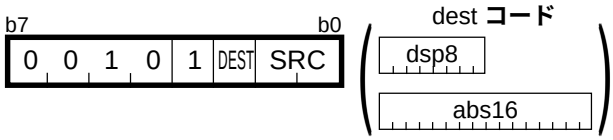
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

SUB

(4) SUB.B:S　　src, R0L/R0H



src		SRC
Rn	R0L/R0H	0 0
dsp:8[SB/FB]	dsp:8[SB]	0 1
	dsp:8[FB]	1 0
abs16	abs16	1 1

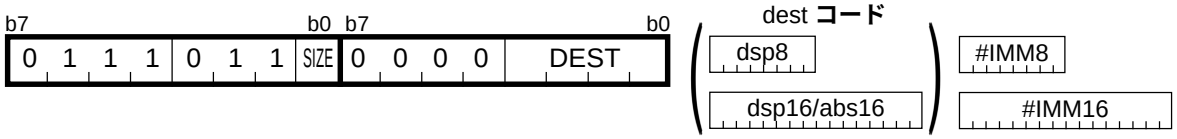
dest	DEST
R0L	0
R0H	1

【バイト数／サイクル数】

src	Rn	dsp:8[SB/FB]	abs16
バイト数／サイクル数	1/2	2/3	3/3

TST

(1) TST.size　　#IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST
Rn	R0L/R0	0 0 0 0
	R0H/R1	0 0 0 1
	R1L/R2	0 0 1 0
	R1H/R3	0 0 1 1
An	A0	0 1 0 0
	A1	0 1 0 1
[An]	[A0]	0 1 1 0
	[A1]	0 1 1 1

dest		DEST
dsp:8[An]	dsp:8[A0]	1 0 0 0
	dsp:8[A1]	1 0 0 1
dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	dsp:8[FB]	1 0 1 1
dsp:16[An]	dsp:16[A0]	1 1 0 0
	dsp:16[A1]	1 1 0 1
dsp:16[SB]	dsp:16[SB]	1 1 1 0
abs16	abs16	1 1 1 1

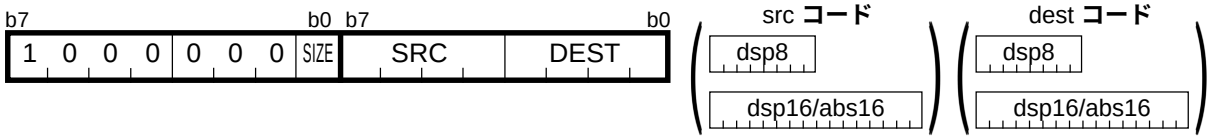
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

TST

(2) TST.size src, dest



.size	SIZE
.B	0
.W	1

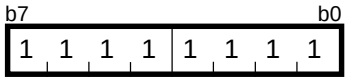
src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

UND

(1) UND

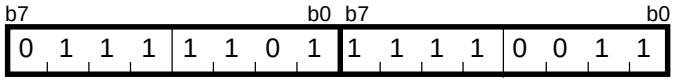


【バイト数／サイクル数】

バイト数／サイクル数	1/20

WAIT

(1) WAIT

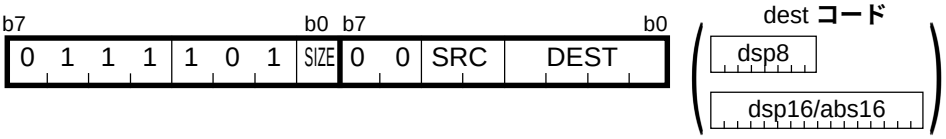


【バイト数／サイクル数】

バイト数／サイクル数	2/3

XCHG

(1) XCHG.size src, dest



.size	SIZE
.B	0
.W	1

src	SRC
R0L/R0	0 0
R0H/R1	0 1
R1L/R2	1 0
R1H/R3	1 1

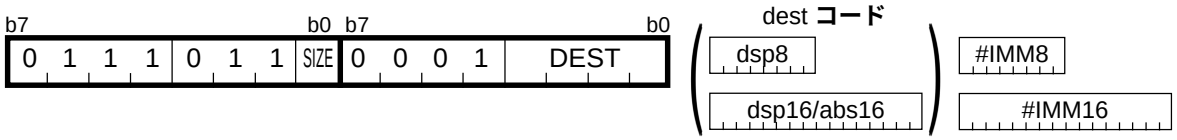
dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	2/4	2/4	2/5	3/5	3/5	4/5	4/5	4/5

XOR

(1) XOR.size #IMM, dest



.size	SIZE
.B	0
.W	1

dest		DEST	dest		DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

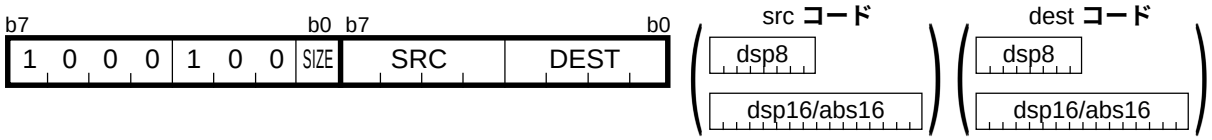
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
バイト数／サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は1バイト増加します。

XOR

(2) XOR.size src, dest



.size	SIZE
.B	0
.W	1

src/dest		SRC/DEST	src/dest		SRC/DEST
Rn	R0L/R0	0 0 0 0	dsp:8[An]	dsp:8[A0]	1 0 0 0
	R0H/R1	0 0 0 1		dsp:8[A1]	1 0 0 1
	R1L/R2	0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	1 0 1 0
	R1H/R3	0 0 1 1		dsp:8[FB]	1 0 1 1
An	A0	0 1 0 0	dsp:16[An]	dsp:16[A0]	1 1 0 0
	A1	0 1 0 1		dsp:16[A1]	1 1 0 1
[An]	[A0]	0 1 1 0	dsp:16[SB]	dsp:16[SB]	1 1 1 0
	[A1]	0 1 1 1	abs16	abs16	1 1 1 1

【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB]	abs16
Rn	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
An	2/2	2/2	2/3	3/3	3/3	4/3	4/3	4/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	4/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	5/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
dsp:16[SB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	6/4

第５章

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「1.10　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 5.1.1 に割り込みの分類を示します。表 5.1.1 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。

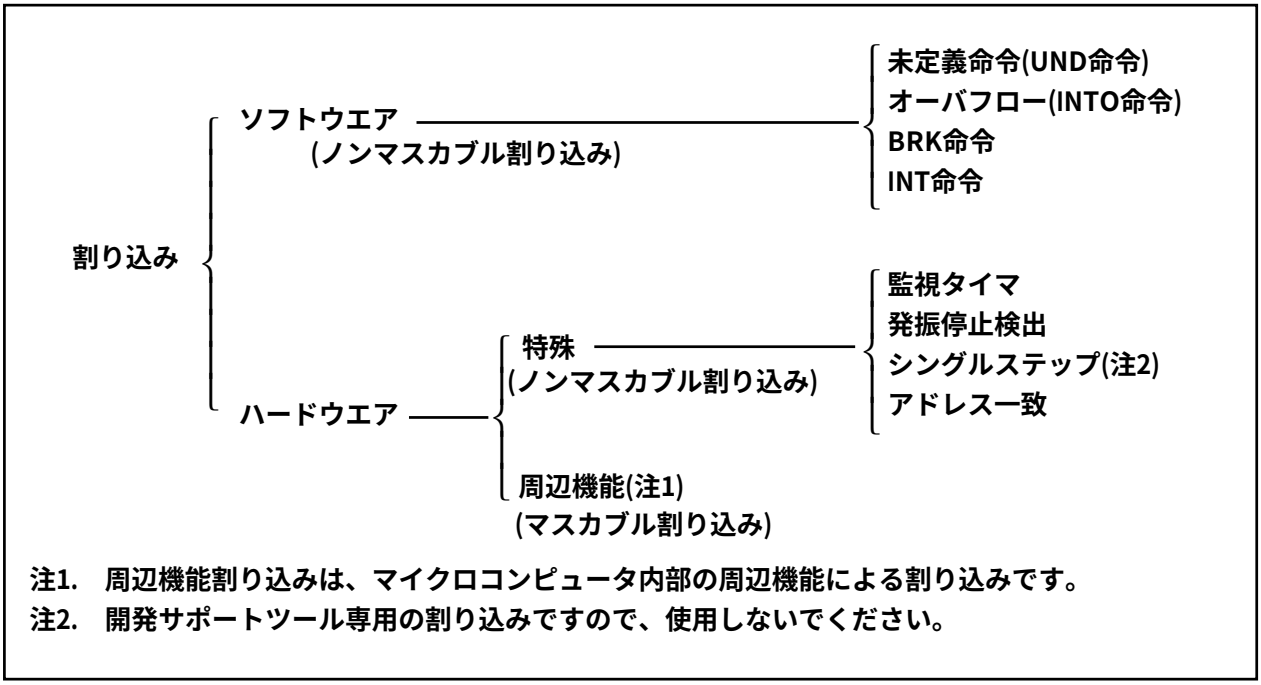


図5.1.1　割り込みの分類

・マスカブル割り込み　：　割り込み許可フラグ(Iフラグ)による割り込みの許可(禁止)や割り込み優先レベルによる割り込み優先順位の変更が**可能**

・ノンマスカブル割り込み　：　割り込み許可フラグ(Iフラグ)による割り込みの許可(禁止)や割り込み優先レベルによる割り込み優先順位の変更が**不可能**

表 5.1.1 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	ベクタ番地 番地(L)～番地(H)	備考
未定義命令	0FFDC16 ～ 0FFDF16	UND命令で割り込み
オーバフロー	0FFE016 ～ 0FFE316	INTO命令で割り込み
BRK命令	0FFE416 ～ 0FFE716	0FFE716番地の内容がFF16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
アドレス一致	0FFE816 ～ 0FFEB16	アドレス一致割り込み許可ビットあり
シングルステップ (注1)	0FFEC16 ～ 0FFEF16	使用禁止
監視タイマ、発振停止検出	0FFF016 ～ 0FFF316	
(予約)	0FFF416 ～ 0FFF716	
(予約)	0FFF816 ～ 0FFFB16	
リセット	0FFFC16 ～ 0FFFF16	

注1. 開発サポートツール専用の割り込みですので、使用しないでください。

5.1.2 ソフトウエア割り込み

ソフトウエア割り込みは、命令の実行によって発生します。ソフトウエア割り込みはノンマスカブル割り込みです。

●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。

●オーバフロー割り込み

オーバフロー割り込みは、O フラグが“1”(演算の結果がオーバフロー)の場合、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は次のとおりです。

ABS、ADC、ADCF、ADD、CMP、DIV、DIVU、DIVX、NEG、RMPA、SBB、SHA、SUB

● 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。

● INT 命令割り込み

INT 命令割り込みは、INT 命令を実行すると発生します。INT 命令で指定できるソフトウエア割り込み番号は0～63です。ソフトウエア割り込み番号4～31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。

ソフトウエア割り込み番号 0 ～ 31 では、命令実行時に U フラグを退避し、U フラグを“0”(ISP を選択)にした後、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退避しておいた U フラグを復帰します。ソフトウエア割り込み番号 32 ～ 63 では、命令実行時 U フラグは変化せず、そのとき選択されている SP を使用します。

5.1.3 ハードウエア割り込み

ハードウエア割り込みには、特殊割り込みと周辺機能割り込みがあります。

●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。

(1) 監視タイマ割り込み

監視タイマによる割り込みです。監視タイマ割り込み発生後は、監視タイマを初期化してください。監視タイマの詳細は、ハードウエアマニュアルを参照してください。

(2) 発振停止検出割り込み

発振停止検出機能による割り込みです。発振停止検出機能の詳細は、ハードウエアマニュアルを参照してください。

(3) シングルステップ割り込み

開発サポートツール専用の割り込みですので、使用しないでください。

(4) アドレス一致割り込み

アドレス一致割り込みは、AIER レジスタの AIER0 ビット、AIER1 ビットのうち、いずれか 1 つが“1”(アドレス一致割り込み許可)の場合、対応する RMAD0 ～ RMAD1 レジスタで示される番地の命令を実行する直前に発生します。

●周辺機能割り込み

周辺機能割り込みは、マイクロコンピュータ内部の周辺機能による割り込みです。周辺機能割り込みは、マスカブル割り込みです。

内蔵される周辺機能は品種展開によって異なりますので、それぞれの割り込み要因も品種展開によって異なります。周辺機能割り込みについての詳細は、ハードウエ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、禁止は、FLGレジスタのIフラグ、IPL、各割り込み制御レジスタのILVL2～ILVL0ビットで行います。また、割り込み要求の有無は、各割り込み制御レジスタのIRビットに示されます。

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、ハードウエアマニュアルを参照してください。

5.2.1 Iフラグ

Iフラグは、マスカブル割り込みを許可または禁止します。Iフラグを“1”(許可)にすると、マスカブル割り込みは許可され、“0”(禁止)にするとすべてのマスカブル割り込みは禁止されます。

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付判定に反映されるのは次のタイミングです。

・REIT命令で変化させたとき、そのREIT命令から反映される。

・FCLR、FSET、POPC、LDC各命令で変化させたとき、次の命令から反映される。

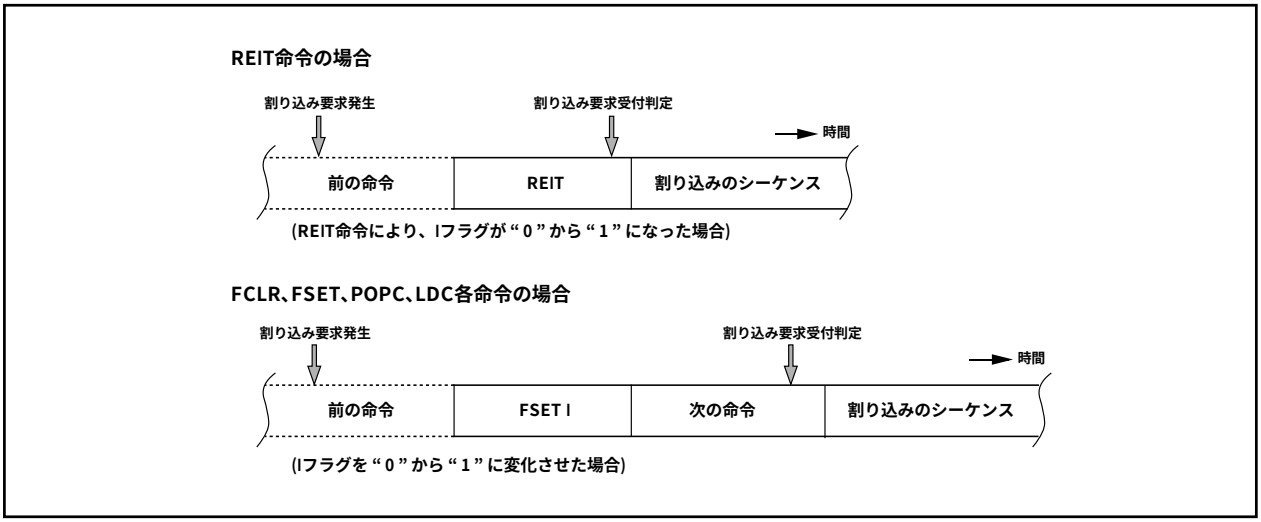


図5.2.1 Iフラグを変化させたときの割り込みへの反映のタイミング

5.2.2 IRビット

IRビットは割り込み要求が発生すると、“1”(割り込み要求あり)になります。割り込み要求が受け付けられ、対応する割り込みベクタに分岐した後、IRビットは“0”(割り込み要求なし)になります。

IRビットはプログラムによって“0”にできます。“1”を書かないでください。

5.2.3 ILVL2～ILVL0ビット、IPL

割り込み優先レベルは、ILVL2～ILVL0ビットで設定できます。

表5.2.1に割り込み優先レベルの設定、表5.2.2にIPLにより許可される割り込み優先レベルを示します。

割り込み要求が受け付けられる条件を次に示します。

・Iフラグ　＝　1
・IRビット　＝　1
・割り込み優先レベル　＞　IPL

Iフラグ、IRビット、ILVL2～ILVL0ビット、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。

表5.2.1 割り込み優先レベルの設定

ILVL2～ILVL0	割り込み優先レベル	優先順位
000_2	レベル0 (割り込み禁止)	―――
001_2	レベル1	低い
010_2	レベル2	↓
011_2	レベル3	↓
100_2	レベル4	↓
101_2	レベル5	↓
110_2	レベル6	↓
111_2	レベル7	高い

表5.2.2 IPLにより許可される割り込み優先レベル

IPL	許可される割り込み優先レベル
000_2	レベル1以上を許可
001_2	レベル2以上を許可
010_2	レベル3以上を許可
011_2	レベル4以上を許可
100_2	レベル5以上を許可
101_2	レベル6以上を許可
110_2	レベル7以上を許可
111_2	すべてのマスカブル割り込みを禁止

IPLまたは各割り込み優先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次のタイミングです。

・REIT命令でIPLを変化させたとき、REIT命令の最後のクロックから2クロック後に実行されている命令から反映される。
・POPC、LDC、LDIPL各命令でIPLを変化させたとき、使用した命令の最後のクロックから3クロック後に実行されている命令から反映される。
・各割り込みの割り込み優先レベルをMOV命令等で変化させたとき、使用した命令の最後のクロックから3クロック後に実行されている命令から反映される。

5.2.4 割り込み制御レジスタの変更

(1) 割り込み制御レジスタは、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変更してください。割り込み要求が発生する可能性がある場合は、割り込みを禁止した後、割り込み制御レジスタを変更してください。

(2) 割り込みを禁止して割り込み制御レジスタを変更する場合、使用する命令に注意してください。

IRビット以外のビットの変更

命令の実行中に、そのレジスタに対応する割り込み要求が発生した場合、IRビットが“1”(割り込み要求あり)にならず、割り込みが無視されることがあります。このことが問題になる場合は、次の命令を使用してレジスタを変更してください。

対象となる命令…AND、OR、BCLR、BSET

IRビットの変更

IRビットを“0”(割り込み要求なし)にする場合、使用する命令によってはIRビットが“0”にならないことがあります。IRビットはMOV命令を使用して“0”にしてください。

(3) Iフラグを使用して割り込みを禁止にする場合、次の参考プログラム例にしたがってIフラグの設定をしてください。(参考プログラム例の割り込み制御レジスタの変更は(2)を参照してください。)

例1～例3は内部バスと命令キューバッファの影響により割り込み制御レジスタが変更される前にIフラグが“1”(割り込み許可)になることを防ぐ方法です。

例1 : NOP命令で割り込み制御レジスタが変更されるまで待たせる例

```
INT_SWITCH1：
  FCLR    I              ；割り込み禁止
  AND.B   #00H，0056H    ；TXICレジスタを“00₁₆”にする
  NOP                    ；
  NOP
  FSET    I              ；割り込み許可
```

例2 : ダミーリードでFSET命令を待たせる例

```
INT_SWITCH2：
  FCLR    I              ；割り込み禁止
  AND.B   #00H，0056H    ；TXICレジスタを“00₁₆”にする
  MOV.W   MEM，R0        ；ダミーリード
  FSET    I              ；割り込み許可
```

例3 : POPC命令でIフラグを変更する例

```
INT_SWITCH3：
  PUSHC   FLG
  FCLR    I              ；割り込み禁止
  AND.B   #00H，0056H    ；TXICレジスタを“00₁₆”にする
  POPC    FLG            ；割り込み許可
```

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、その命令の実行終了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、SMOVB、SMOVF、SSTR、RMPA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、次のように動作します。図5.3.1に割り込みシーケンスの実行時間を示します。

(1) 00000$_{16}$番地を読むことで、CPUは割り込み情報(割り込み番号、割り込み要求レベル)を獲得します。その後、該当する割り込みのIRビットが“0”(割り込み要求なし)になります。
(2) 割り込みシーケンス直前のFLGレジスタをCPU内部の一時レジスタ(注1)に退避します。
(3) FLGレジスタのうち、Iフラグ、Dフラグ、Uフラグは次のようになります。
Iフラグは“0”(割り込み禁止)
Dフラグは“0”(シングルステップ割り込みは割り込み禁止)
Uフラグは“0”(ISPを指定)
ただし、Uフラグは、ソフトウエア割り込み番号32～63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(注1)をスタックに退避します。
(5) PCをスタックに退避します。
(6) IPLに、受け付けた割り込みの割り込み優先レベルを設定します。
(7) 割り込みベクタに設定された割り込みルーチンの先頭番地がPCに入ります。

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。

注1. ユーザは使用できません。

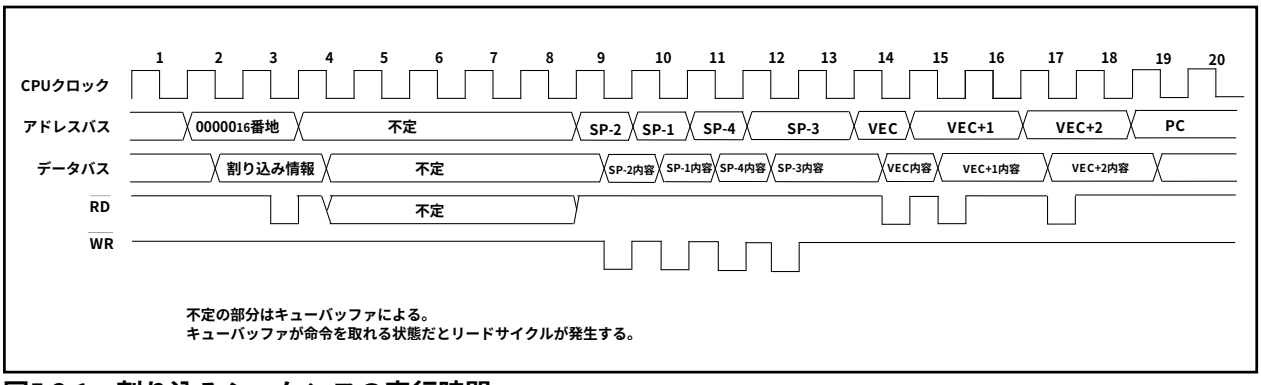


図5.3.1 割り込みシーケンスの実行時間

5.3.1 割り込み応答時間

図5.3.2に割り込み応答時間を示します。割り込み応答時間は、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です。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(図5.3.2の(a))と割り込みシーケンスを実行する時間(20サイクル(b))で構成されます。

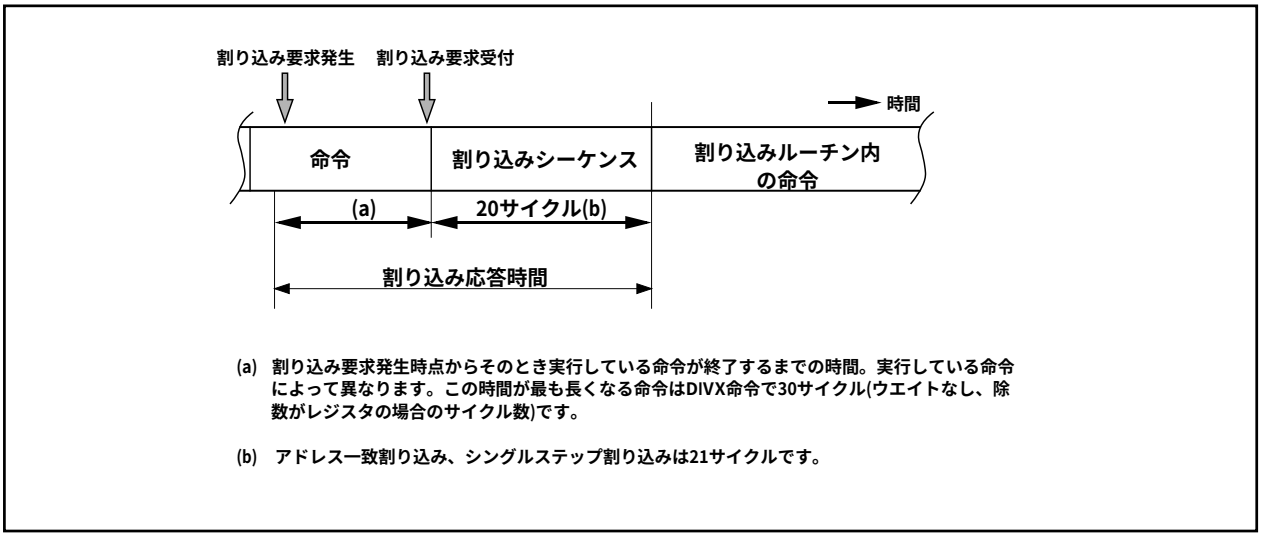


図5.3.2 割り込み応答時間

5.3.2 割り込み要求受付時の IPL の変化

マスカブル割り込みの割り込み要求が受け付けられると、IPLには受け付けた割り込みの割り込み優先レベルが設定されます。

ソフトウエア割り込みと特殊割り込み要求が受け付けられると表5.3.1に示す値がIPLに設定されます。表 5.3.1 にソフトウエア割り込み、特殊割り込み受け付け時の IPL の値を示します。

表5.3.1 ソフトウエア割り込み、特殊割り込み受け付け時のIPLの値

割り込み優先レベルをもたない割り込み要因	設定される IPL の値
監視タイマ、発振停止検出	7
ソフトウエア、アドレス一致、シングルステップ	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、FLG レジスタと PC をスタックに退避します。

スタックへは PC の上位 4 ビットと FLG レジスタの上位 4 ビット(IPL)、下位 8 ビットの合計 16 ビットをまず退避し、次に PC の下位 16 ビットを退避します。図 5.3.3 に割り込み要求受付前と後のスタックの状態を示します。

その他の必要なレジスタは、割り込みルーチンの最初でプログラムによって退避してください。PUSHM 命令を用いると、1 命令で SP を除くすべてのレジスタを退避できます。

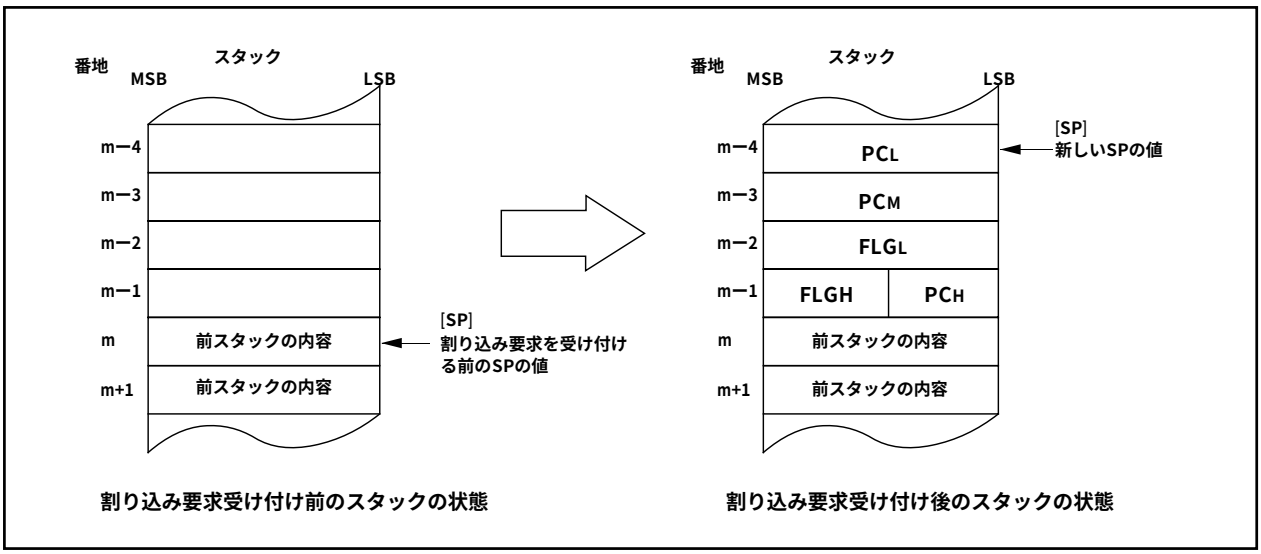


図5.3.3 割り込み要求受け付け前と後のスタックの状態

割り込みシーケンスで行われるレジスタ退避は、8 ビットずつ 4 回に分けて行われます。図 5.3.4 にレジスタ退避動作を示します。

注1. ソフトウエア番号32～63のINT命令を実行した場合は、U フラグが示す SP です。それ以外は、ISP です。

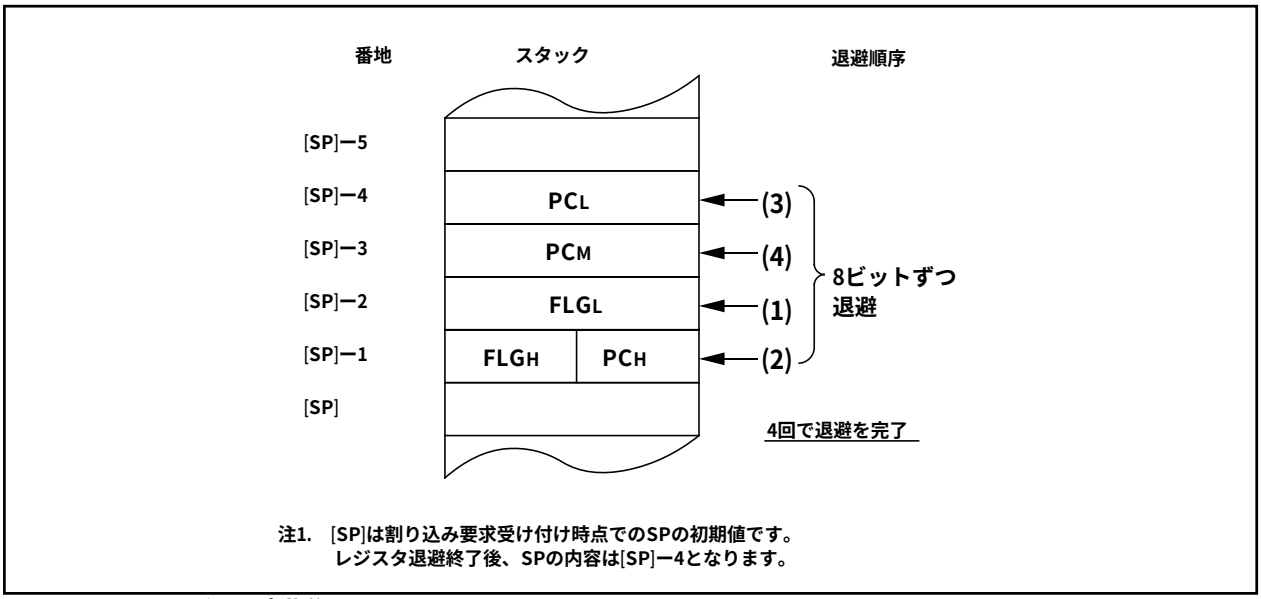


図5.3.4 レジスタ退避動作

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、スタックに退避していた割り込みシーケンス直前のFLGレジスタとPCが復帰します。その後、割り込み要求受け付け前に実行していたプログラムに戻ります。

割り込みルーチン内でプログラムによって退避したレジスタは、REIT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

1命令実行中に2つ以上の割り込み要求が発生した場合は、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)の優先レベルは、ILVL2～ILVL0ビットによって任意に選択できます。ただし、割り込み優先レベルが同じ設定値の場合は、ハードウエアで設定されている優先順位の高い割り込みが受け付けられます。

監視タイマ割り込みなど、特殊割り込みの優先順位はハードウエアで設定されています。図5.5.1にハードウエア割り込みの割り込み優先順位を示します。

ソフトウエア割り込みは割り込み優先順位の影響を受けません。命令を実行すると割り込みルーチンを実行します。

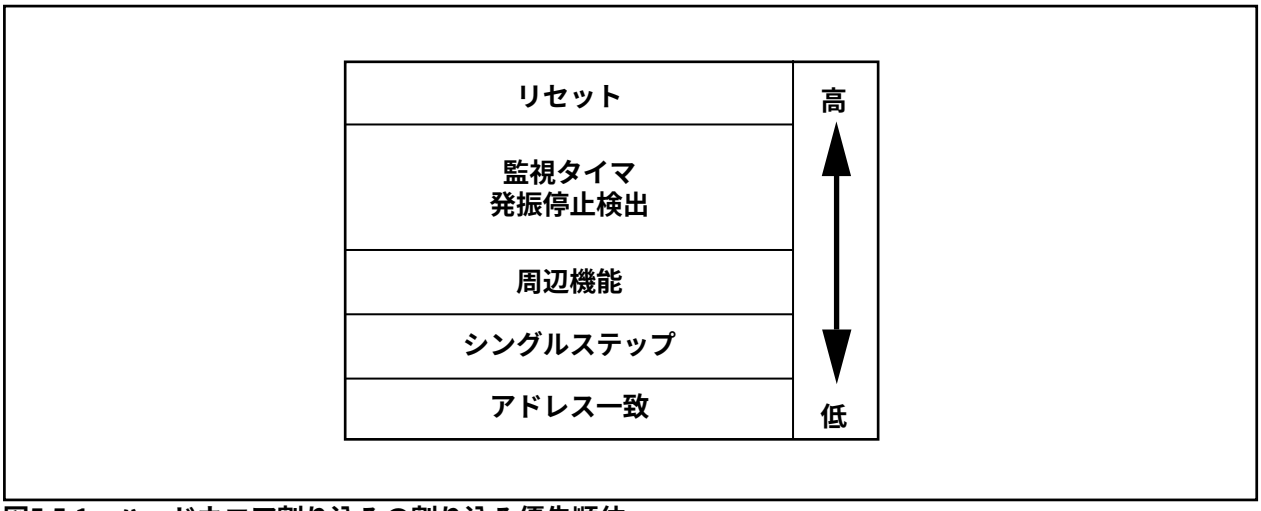


図5.5.1　ハードウエア割り込みの割り込み優先順位

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優先レベル

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)より高い優先順位をもつ割り込み要求を受け付けることができます。図5.6.1に多重割り込みについて示します。

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。そして、REIT 命令によって IPL が復帰され、割り込み優先順位の判定が行われたとき、以下の状態であれば保持されていた割り込み要求が受け付けられます。

保持されていた割り込み要求の割り込み優先レベル ＞ 復帰されたプロセッサ割り込み優先レベル(IPL)

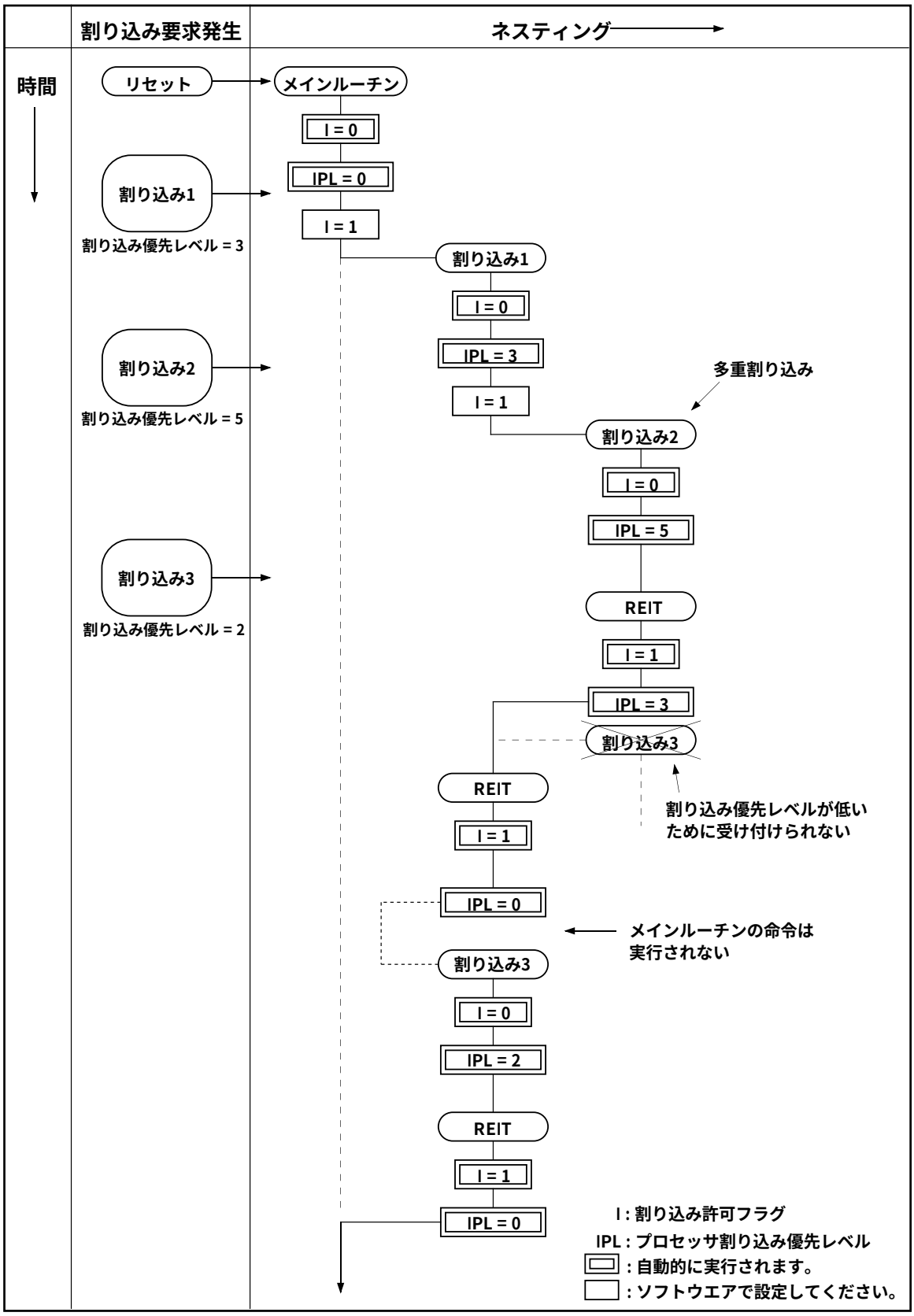

割り込み要求発生
ネスティング
時間
リセット
メインルーチン
I = 0
IPL = 0
I = 1
割り込み1
割り込み優先レベル = 3
割り込み1
I = 0
IPL = 3
I = 1
割り込み2
割り込み優先レベル = 5
多重割り込み
割り込み2
I = 0
IPL = 5
REIT
I = 1
IPL = 3
割り込み3
割り込み3
割り込み優先レベル = 2
割り込み優先レベルが低い
ために受け付けられない
REIT
I = 1
IPL = 0
メインルーチンの命令は
実行されない
割り込み3
I = 0
IPL = 2
REIT
I = 1
IPL = 0
I : 割り込み許可フラグ
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 5.6.1 多重割り込み

5.7 割り込みの注意事項

5.7.1 00000_{16} 番地の読み出し

プログラムで 00000_{16} 番地を読まないでください。マスカブル割り込みの割り込み要求を受け付けた場合、CPUは割り込みシーケンスの中で割り込み情報(割り込み番号と割り込み要求レベル)を 00000_{16} 番地から読みます。このとき、受け付けられた割り込みのIRビットが“0”になります。

プログラムで 00000_{16} 番地を読むと、許可されている割り込みのうち、最も優先順位の高い割り込みのIRビットが“0”になります。そのため、割り込みがキャンセルされたり、予期しない割り込みが発生することがあります。

5.7.2 SPの設定

割り込みを受け付ける前に、SPに値を設定してください。リセット後、SPは“0000_{16}”です。そのため、SP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。

5.7.3 割り込み制御レジスタの変更

(1) 割り込み制御レジスタは、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変更してください。割り込み要求が発生する可能性がある場合は、割り込みを禁止した後、割り込み制御レジスタを変更してください。

(2) 割り込みを禁止して割り込み制御レジスタを変更する場合、使用する命令に注意してください。

IRビット以外のビットの変更

命令の実行中に、そのレジスタに対応する割り込み要求が発生した場合、IRビットが“1”(割り込み要求あり)にならず、割り込みが無視されることがあります。このことが問題になる場合は、次の命令を使用してレジスタを変更してください。

対象となる命令…AND、OR、BCLR、BSET

IRビットの変更

IRビットを“0”(割り込み要求なし)にする場合、使用する命令によってはIRビットが“0”にならないことがあります。IRビットはMOV命令を使用して“0”にしてください。

(3) Iフラグを使用して割り込みを禁止にする場合、次の参考プログラム例にしたがってIフラグの設定をしてください。(参考プログラム例の割り込み制御レジスタの変更は(2)を参照してください。)

例1～例3は内部バスと命令キューバッファの影響により割り込み制御レジスタが変更される前にIフラグが“1”(割り込み許可)になることを防ぐ方法です。

例1 : NOP命令で割り込み制御レジスタが変更されるまで待たせる例

```
INT_SWITCH1：
  FCLR    I              ；割り込み禁止
  AND.B   #00H，0056H    ；TXICレジスタを“0016”にする
  NOP                    ；
  NOP
  FSET    I              ；割り込み許可
```

例2 : ダミーリードでFSET命令を待たせる例

```
INT_SWITCH2：
  FCLR    I              ；割り込み禁止
  AND.B   #00H，0056H    ；TXICレジスタを“0016”にする
  MOV.W   MEM，R0        ；ダミーリード
  FSET    I              ；割り込み許可
```

例3 : POPC命令でIフラグを変更する例

```
INT_SWITCH3：
  PUSHC   FLG
  FCLR    I              ；割り込み禁止
  AND.B   #00H，0056H    ；TXICレジスタを“0016”にする
  POPC    FLG            ；割り込み許可
```

第６章

サイクル数の計算

6.1　命令キューバッファ

6.1 命令キューバッファ

R8C/Tinyシリーズは、4段(4バイト)の命令キューバッファを持っています。CPUがバスを使用できる状態で命令キューバッファに空きがある場合、命令コードは命令キューバッファに取り込まれます。これをプリフェッチと言います。CPUは命令キューバッファに入っている命令コードを読み出しながら(フェッチ)プログラムを実行します。

第4章で説明しているサイクル数は、命令キューバッファに命令コードが揃っており、メモリに対し8ビットのデータをソフトウエアウエイトなしに読み書きする場合のサイクル数です。下記の場合、マニュアルに記述しているサイクル数より多くなります。

■命令キューバッファにCPUが必要とする命令コードが揃っていない。
実行に必要な命令コードが揃うまで命令コードを読み込みます。さらに次の場合、読み込みサイクル数が増加します。
・ソフトウエアウエイトサイクルが存在する領域から命令コードを読み込む
ウエイト数分だけ読み込むサイクル数が増加します。

■ソフトウエアウエイトサイクルが存在する領域にデータを読み書きする。
ウエイト数分だけサイクル数が増加します。

■16ビットのデータをSFR又は内部メモリに対して読み書きする。
1データの読み書きに対して2回読み書きします。そのため、1データの読み書きにつきサイクル数は、1サイクル増加します。

なお、同一タイミングでプリフェッチとデータアクセスが発生した場合、データアクセスが優先されます。

また、命令キューバッファ内に命令コードが3バイト以上存在するときは、命令キューバッファに空きがないと判断し、プリフェッチを行いません。

図6.1.1に読み込み命令を開始する場合（ソフトウエアウエイトなし）を示します。

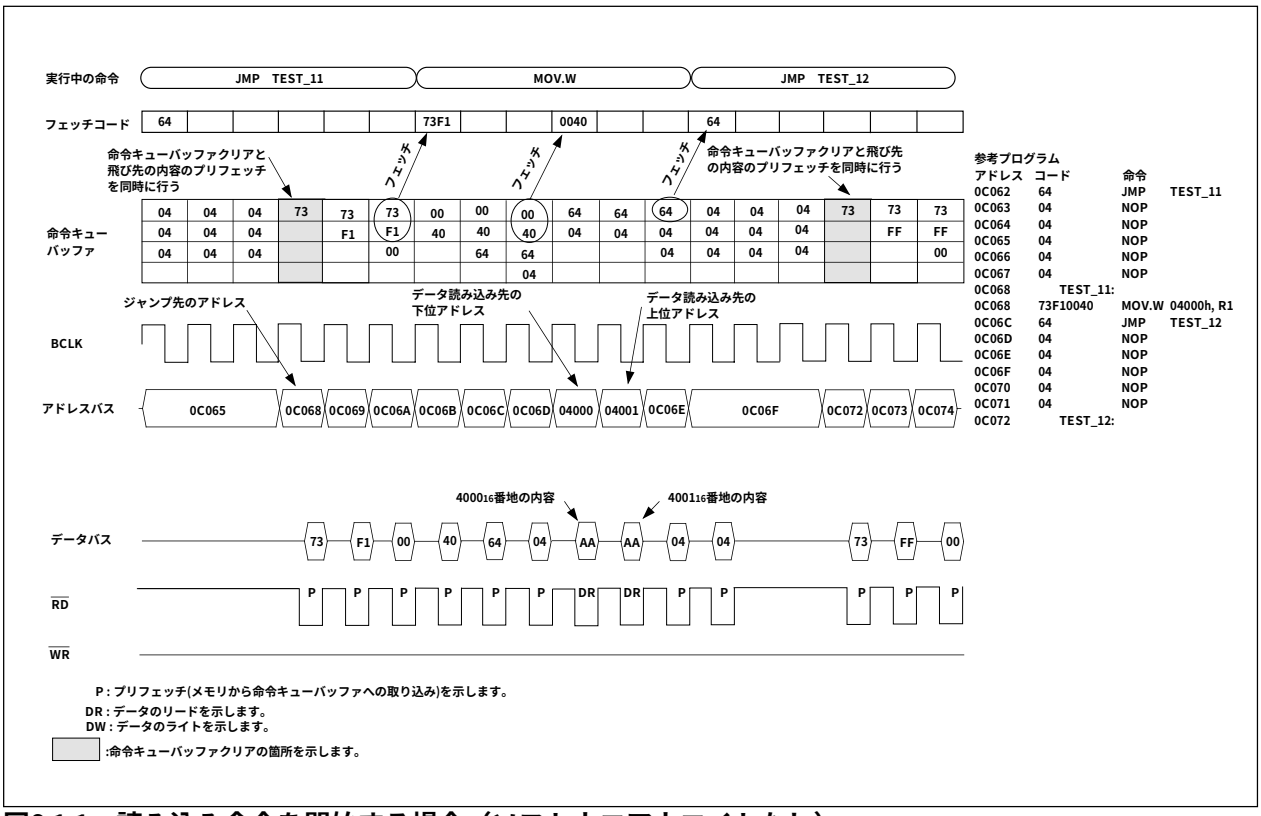


図6.1.1 読み込み命令を開始する場合（ソフトウエアウエイトなし）

レイアウトの都合上、このページは白紙です。

Q & A

R8C/Tiny シリーズを最大限にご活用いただくための情報を、Q&A 形式で以下に掲載します。

Q&Aは原則として1つの質問およびその回答を1ページ内に掲載しており、各ページの上段が質問事項、下段がその回答となっています（1つの質問・回答を2ページ以上に渡って掲載する場合は、右下にページ数を記載しています）。

また、各ページの右上にはそのページの内容に関係のある主な機能を示します。

Q

スタティックベースレジスタ(SB)とフレームベースレジスタ(FB)の使い分けは？

A

SBとFBは同じ機能をもちますので、アセンブラ言語でプログラムする場合は、自由に使用できます。C言語でプログラムする場合、FBはスタックフレームベースレジスタとして使用します。

Q

プログラム実行中に割り込みテーブルレジスタ(INTB)の値を変更することは可能か？

A

可能です。ただし、INTB の値を変更中に割り込み要求が発生すると、マイコンが暴走する可能性があります。したがって、プログラム実行中に頻繁に INTB の値を変更することは、推奨しません。

Q

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)の違い、役割は？

A

USPはOS使用時に使用します。いくつかのタスクが存在するとき、OSはタスクごとにレジスタ類を退避するスタック領域を確保します。また、そのタスクを実行中に発生する割り込みのために、タスクごとに割り込みで使用するスタック領域を確保する必要があります。ここで、USPとISPを使用すると、割り込み用のスタックは、各タスクで共用できるため、効率の良いスタック領域の使い方ができます。

Q

ビット命令を絶対アドレッシングで使用したときの命令コードはどのようになるのか？

A

BSET　bit,base:16 の場合を説明します。

この命令は、4 バイト命令です。命令コードの上位 2 バイトはオペコード部を示し、下位 2 バイトがアドレッシングモード部で、bit,base:16 を表現します。

下位 2 バイトと bit,base:16 との関係は、次のとおりです。

下位 2 バイト　=　base:16 × 8 + bit

例えば、BSET　2,0AH　　(000A16 番地のビット 2 を 1 にする)の場合、

下位 2 バイトは A × 8 + 2 ＝ 52H となります。

また、 BSET　18,8H　　(000816 番地のビット 0 から数えて 18 ビット目を 1 にする）の場合、

下位 2 バイトは 8 × 8 + 18 ＝ 52H となり、BSET　2,AH と同じ命令コードになります。

base:16 × 8 + bit の最大値 FFFFH は、1FFF16 番地のビット 7 を示します。これが、ビット命令を絶対アドレッシングで使用したときに指定できる最大のビットです。

Q

DIV 命令と DIVX 命令の違いは？

A

DIV 命令、DIVX 命令のいずれも符号付きの除算命令ですが、余りの符号が違います。

DIV 命令の余りの符号は、被除数の符号と同じになるのに対し、DIVX 命令の余りの符号は、除数の符号と同じになります。

また、一般に商、除数、被除数、と余りとの間には次の関係が成り立ちます。

被除数＝除数×商＋余り

余りの符号が違う結果、正の整数を負の整数で割ったとき、および負の整数を正の整数で割ったときの商も、両方の命令で異なってきます。

例えば、10 を－3 で割ったとき

DIV 命令では、－3 余り +1 になるのに対し、DIVX 命令では、－4 余り－2 になります。

また、－10 を +3 で割ったとき

DIV 命令では、－3 余り－1 になるのに対し、DIVX 命令では、－4 余り +2 になります。

用語集

このソフトウエアマニュアルで使用している用語について、以下に説明します。なお、この用語集は、このマニュアルにおいてだけ有効です。

用語	意味	関連語句
LSB	Least Significant Bit の略称。 データの最下位にあるビットを示す。	MSB
MSB	Most Significant Bit の略称。 データの最上位にあるビットを示す。	LSB
アンパック	結合している項目、またはパックされた情報を分離すること。8ビットの情報を下位4ビットと上位4ビットに分離したり、16ビットの情報を下位8ビットと上位8ビットに分離する意味でよく使われる。	パック
SFR領域	SFRは、Special Function Register の略称。マイコンに内蔵される周辺回路の制御ビットおよび制御レジスタが配置されている領域。	
演算	転送、比較、ビット処理、シフト、ローテート、算術、論理、分岐の総称。	
オーバフロー	演算の結果、表現可能な最大値を超えること。 (桁あふれ)	
オペコード	命令コードのうち、命令の動作を表すコード。	オペランド
オペランド	命令コードのうち、命令の動作の対象を表すコード。	オペコード

用語	意味	関連語句
拡張領域	R8C/Tinyシリーズでは、10000_{16}～$FFFFF_{16}$番地の領域を示します。	
キャリー	桁上がり。	ボロー
コンテキスト	プログラムで使用しているレジスタのこと。	
実行アドレス	修飾が行われた後の実際のアドレス。	
シフトアウト	レジスタ等の内容を右または左に動かすことにより、あふれること。	
10進加算	10進で加算すること。	
スタックフレーム	C言語の関数で使用する自動変換の領域。	
ストリング	文字列。	

用語	意味	関連語句
ゼロ拡張	データ長を拡張するとき、拡張される上位のビットを“0”にして拡張すること。 たとえば、FF_{16}を16ビットにゼロ拡張すると$00FF_{16}$になる。	
ディスプレースメント	変位。	
パック	データを結合すること。 2個の4ビットのデータを8ビットに結合したり、2個の8ビットのデータを16ビットに結合することでよく使われる。	アンパック
符号拡張	データ長を拡張するとき、拡張される上位のビットを符号ビットの状態にあわせて拡張すること。 たとえば、FF_{16}を符号拡張すると$FFFF_{16}$に、$0F_{16}$を符号拡張すると$000F_{16}$になる。	
符号ビット	正または負を表すビット(最上位ビット)。	
ボロー	桁借り(桁下がり)。	キャリー
マクロ命令	ソース・ランゲージ(source language)に書かれる1つの命令で、機械コードプログラムにコンパイルされたときには、幾つかの機械コード命令で表現される命令のこと。	

記号集

このソフトウエアマニュアルで使用している記号について、以下に説明します。なお、この記号集は、このマニュアルにおいてだけ有効です。

記　号	意　味
←	右辺から左辺への転送
←→	右辺と左辺との交換
＋	加算
－	減算
×	乗算
÷	除算
∧	論理積
∨	論理和
∀	排他的論理和
‾	否定
dsp16	16 ビットの変位
dsp20	20 ビットの変位
dsp8	8 ビットの変位
EVA(　)	(　)内で示す実効アドレス
EXT(　)	(　)内の符号拡張
(H)	レジスタまたはメモリの上位バイト
H4:	8 ビットレジスタまたは 8 ビットメモリの上位 4 ビット
｜｜	絶対値
(L)	レジスタまたはメモリの下位バイト
L4:	8 ビットレジスタまたは 8 ビットメモリの下位 4 ビット
LSB	Least Significant Bit　の略称
M(　)	(　)内で示すメモリの内容
(M)	レジスタまたはメモリの中位バイト
MSB	Most Significant Bit　の略称
PCH	プログラムカウンタの上位バイト
PCML	プログラムカウンタの中位バイトおよび下位バイト
FLGH	フラグレジスタの上位 4 ビット
FLGL	フラグレジスタの下位 8 ビット

索引

コ

サ

ス

セ

ソ

テ

ニ

ノ

ハ

フ

マ

メ

ユ

リ

レ

ロ

ワ

改訂履歴	R8C/Tiny シリーズソフトウエアマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003/6/19		初版発行

ルネサス16ビットシングルチップマイクロコンピュータ
ソフトウエアマニュアル
R8C/Tinyシリーズ

発行年月日　2003年6月19日　Rev. 1.00

発行　株式会社 ルネサス テクノロジ　営業企画統括部
〒100-0004　東京都千代田区大手町2-6-2

R8C/Tiny 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

RJJ09B0002-0100Z