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。

お客様各位

資料中の「三菱電機」、「三菱XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って株式会社日立製作所及び三菱電機株式会社のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。
従いまして、本資料中には「三菱電機」、「三菱電機株式会社」、「三菱半導体」、「三菱XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

注:「高周波・光素子事業、パワーデバイス事業については三菱電機にて引き続き事業運営を行います。」

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

User's Manual

16

M16C/80、M16C/70 グループ

ソフトウエアマニュアル

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
M16Cファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2002.02

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三菱半導体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三菱電機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、三菱電機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、三菱電機は、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。三菱半導体製品のご購入に当たりましては、事前に三菱電機または特約店へ最新の情報をご確認頂きますとともに、三菱電機半導体情報ホームページ（http://www.semicon.melco.co.jp/）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、三菱電機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。三菱電機は、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、三菱電機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、文書による三菱電機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三菱電機または特約店までご照会ください。

本書の使い方

本書はM16C/80, M16C/70シリーズのソフトウエアマニュアルです。M16C/80, M16C/70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。

○M16C/80, M16C/70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**

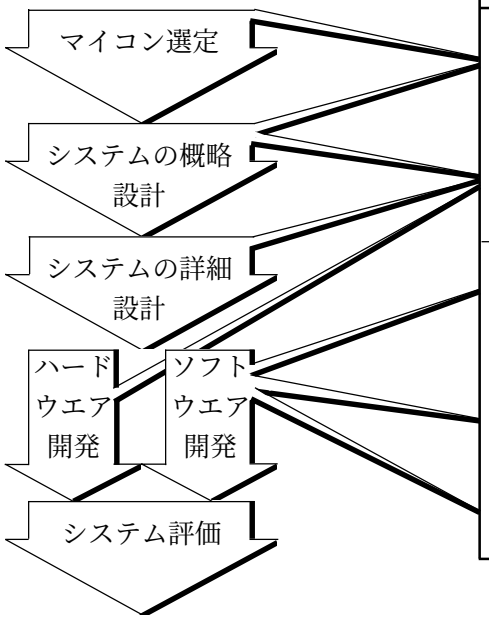
○機能からニーモニックと記載ページを確認する ..... **機能別ページ早見表**

○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、巻末には記号集、用語集、および索引を記載しています。

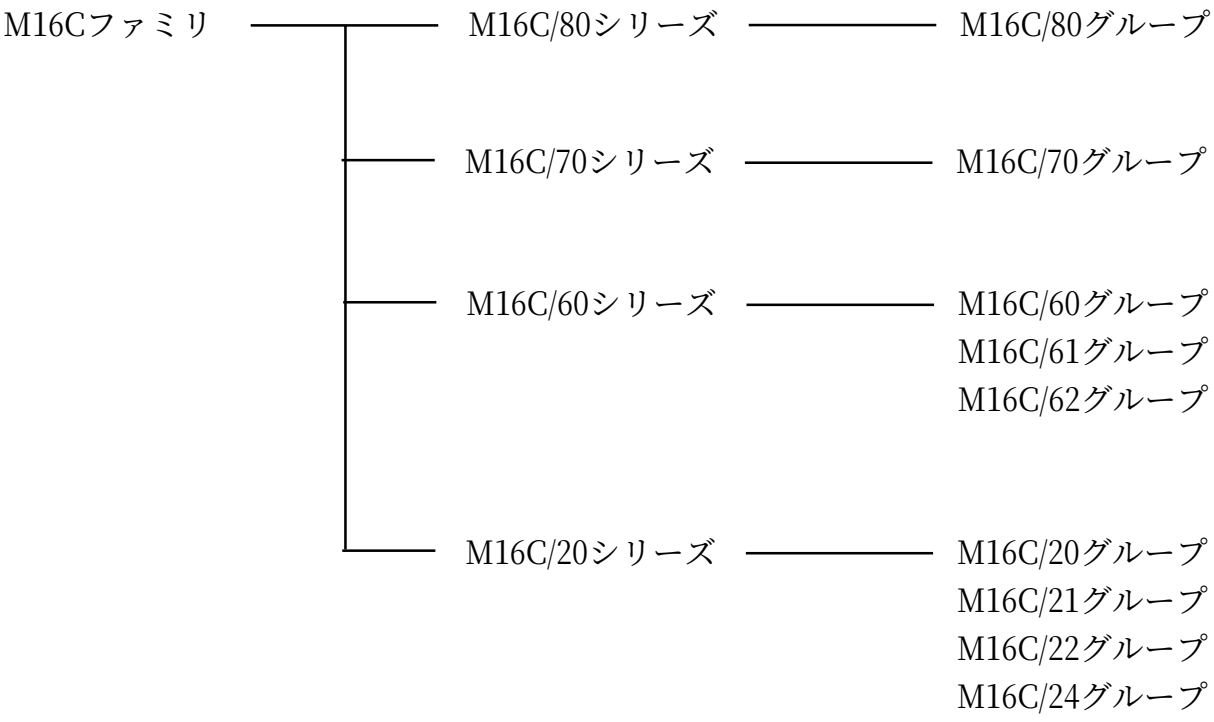
M16Cファミリ関連ドキュメント一覧

用途
（マイコン開発の流れ）



ドキュメントの種類		記載内容
ハードウエア	データシート／データブック	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）
ハードウエア	ユーザーズマニュアル	ハードウエアの仕様、動作、応用例（周辺との接続、ソフトウエアとの関わり）の詳細
ソフトウエア	プログラミングマニュアル	アセンブリ言語、C言語によるプログラムの作成方法
ソフトウエア	ソフトウエアマニュアル	各命令（アセンブリ言語）の動作の詳細
ソフトウエア	参考プログラム集	アセンブリ言語の参考プログラム集

M16Cファミリ体系



目　　次

アルファベット別ページ早見表

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
JMP	93	229	RTS	129	272
JMPI	94	231	SBB	130	273
JMPS	95	232	SBJNZ	131	275
JSR	96	233	SC*cnd*	132	276
JSRI	97	234	SCEQ/Z	132	276
JSRS	98	235	SCGE	132	276
LDC	99	235	SCGEU/C	132	276
LDCTX	100	238	SCGT	132	276
LDIPL	101	239	SCGTU	132	276
MAX	102	239	SCLE	132	276
MIN	103	241	SCLEU	132	276
MOV	104	243	SCLT	132	276
MOVA	106	252	SCLTU/NC	132	276
MOV*Dir*	107	253	SCN	132	276
MOVHH	107	253	SCNE/NZ	132	276
MOVHL	107	253	SCNO	132	276
MOVLH	107	253	SCPZ	132	276
MOVLL	107	253	SCMPU	133	277
MOVX	108	255	SHA	134	278
MUL	109	255	SHL	136	281
MULEX	110	257	SIN	138	283
MULU	111	257	SMOVB	139	284
NEG	112	259	SMOVF	140	284
NOP	113	259	SMOVU	141	285
NOT	114	260	SOUT	142	285
OR	115	260	SSTR	143	286
POP	117	263	STC	144	286
POPC	118	263	STCTX	145	288
POPM	119	264	STNZ	146	288
PUSH	120	265	STZ	147	289
PUSHA	121	267	STZX	148	289
PUSHC	122	267	SUB	149	290
PUSHM	123	268	SUBX	151	294
REIT	124	269	TST	152	296
RMPA	125	269	UND	154	298
ROLC	126	270	WAIT	155	298
RORC	127	270	XCHG	156	299
ROT	128	271	XOR	157	299

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	104	243
	MOVA	実効アドレスの転送	106	252
	MOVDir	４ビットデータ転送	107	253
	MOVX	符号拡張転送	108	255
	POP	レジスタ／メモリの復帰	117	263
	POPM	複数レジスタの復帰	119	264
	PUSH	レジスタ／メモリ／即値の退避	120	265
	PUSHA	実効アドレスの退避	121	267
	PUSHM	複数レジスタの退避	123	268
	STNZ	条件付き転送	146	288
	STZ	条件付き転送	147	289
	STZX	条件付き転送	148	289
	XCHG	交換	156	299
ビット処理	BAND	ビット論理積	52	188
	BCLR	ビットクリア	53	188
	BITINDEX	ビットインデックス	54	189
	BM*Cnd*	条件ビット転送	55	190
	BNAND	反転ビット論理積	56	192
	BNOR	反転ビット論理和	57	192
	BNOT	ビット反転	58	193
	BNTST	反転ビットテスト	59	193
	BNXOR	反転ビットの排他的論理和	60	194
	BOR	ビット論理和	61	194
	BSET	ビットセット	64	196
	BTST	ビットテスト	65	196
	BTSTC	ビットテスト＆クリア	66	197
	BTSTS	ビットテスト＆セット	67	198
	BXOR	ビット排他的論理和	68	198
シフト	ROLC	キャリー付き左回転	126	270
	RORC	キャリー付き右回転	127	270
	ROT	回転	128	271
	SHA	算術シフト	134	278
	SHL	論理シフト	136	281
算術	ABS	絶対値	43	174
	ADC	キャリー付き加算	44	174
	ADCF	キャリーフラグの加算	45	176
	ADD	キャリーなし加算	46	176
	ADDX	符号拡張キャリーなし加算	48	183
	CLIP	クリップ命令	69	199
	CMP	比較	70	200

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	CMPX	符号拡張比較	72	206
	DADC	キャリー付き10進加算	73	206
	DADD	キャリーなし10進加算	74	208
	DEC	デクリメント	75	210
	DIV	符号付き除算	76	210
	DIVU	符号なし除算	77	211
	DIVX	符号付き除算	78	212
	DSBB	ボロー付き10進減算	79	213
	DSUB	ボローなし10進減算	80	215
	EXTS	符号拡張	83	218
	EXTZ	ゼロ拡張	84	220
	INC	インクリメント	88	223
	MAX	最大値選択	102	239
	MIN	最小値選択	103	241
	MUL	符号付き乗算	109	255
	MULEX	拡張符号付き乗算	110	257
	MULU	符号なし乗算	111	257
	NEG	2の補数	112	259
	RMPA	積和演算	125	269
	SBB	ボロー付き減算	130	273
	SUB	ボローなし減算	149	290
	SUBX	符号拡張ボローなし減算	151	294
論理	AND	論理積	50	186
	NOT	全ビット反転	114	260
	OR	論理和	115	260
	TST	テスト	152	296
	XOR	排他的論理和	157	299
ジャンプ	ADJNZ	加算＆条件分岐	49	185
	SBJNZ	減算＆条件分岐	131	275
	J*Cnd*	条件分岐	92	229
	JMP	無条件分岐	93	229
	JMPI	間接分岐	94	231
	JMPS	スペシャルページ分岐	95	232
	JSR	サブルーチン呼び出し	96	233
	JSRI	間接サブルーチン呼び出し	97	234
	JSRS	スペシャルページサブルーチン呼び出し	98	235
	RTS	サブルーチンからの復帰	129	272
ストリング	SCMPU	ストリング比較	133	277
	SIN	ストリング入力	138	283
	SMOVB	逆方向のストリング転送	139	284
	SMOVF	順方向のストリング転送	140	284

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
ストリング	SMOVU	ストリング転送比較	141	285
	SOUT	ストリング出力	142	285
	SSTR	ストリングストア	143	286
その他	BRK	デバッグ割り込み	62	195
	BRK2	デバッグ割り込み2	63	195
	ENTER	スタックフレームの構築	81	217
	EXITD	スタックフレームの解放	82	217
	FCLR	フラグレジスタのビットクリア	85	221
	FREIT	高速割り込みからの復帰	86	221
	FSET	フラグレジスタのビットセット	87	222
	INDEX Type	インデックス	89	223
	INT	ソフトウエア割り込み	90	228
	INTO	オーバフロー割り込み	91	228
	LDC	専用レジスタへの転送	99	235
	LDCTX	コンテキストの復帰	100	238
	LDIPL	割り込み許可レベルの設定	101	239
	NOP	ノーオペレーション	113	259
	POPC	専用レジスタの復帰	118	263
	PUSHC	専用レジスタの退避	122	267
	REIT	割り込みからの復帰	124	269
	STC	専用レジスタからの転送	144	286
	STCTX	コンテキストの退避	145	288
	SC*cnd*	条件設定	132	276
	UND	未定義命令割り込み	154	298
	WAIT	ウエイト	155	298

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/R2R0	R0H/R2/-	R1L/R1/R3R1	R1H/R3/-	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24	#IMM8	#IMM16	#IMM24	#IMM32	#IMM	[[An]]	[dsp:8[An]]	[dsp:8[SB/FB]]	[dsp:16[An]]	[dsp:16[SB/FB]]	[dsp:24[An]]	[abs16]	[abs24]		
ABS	○*2	○	○*3	○	○	○	○	○	○	○	○								○	○	○	○	○	○	○	○	43	174
ADC	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○												44	174
ADCF	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	45	176
ADD*1	○	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○		○	○	○	○	○	○	○	○	○	○	46	176
ADDX	○*2	○*4	○*3	○*5	○	○	○	○	○	○	○	○	○	○					○	○	○	○	○	○	○	○	48	183
ADJNZ*1	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○					○									49	185
AND	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	50	186
BITINDEX	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○														54	189
CLIP	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○												69	199
CMP	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○	○	○	○	○	○	○	○	○	○	70	200
CMPX	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○	○					○	○	○	○	○	○	○	○	72	206
DADC	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○												73	206
DADD	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○												74	208
DEC	○	○	○	○	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	75	210
DIV	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	76	210
DIVU	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	77	211
DIVX	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	78	212
DSBB	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	79	213
DSUB	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	80	215
ENTER														○													81	217
EXTS	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○														83	218
EXTZ	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○														84	220
INC	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	88	223
INDEXType	○*2	○	○*3	○	○	○	○	○	○	○	○	○	○														89	223

*1　特定命令アドレッシングをもちます。
*2　R0L/R0を選択できます。
*3　R1L/R1を選択できます。
*4　R0Lだけ選択できます。
*5　R0Hだけ選択できます。
*6　R1Lだけ選択できます。
*7　R1H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/R2R0	R0H/R2/-	R1L/R1/R3R1	R1H/R3/-	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24	#IMM8	#IMM16	#IMM24	#IMM32	#IMM	[[An]]	[dsp:8[An]]	[dsp:8[SB/FB]]	[dsp:16[An]]	[dsp:16[SB/FB]]	[dsp:24[An]]	[abs16]	[abs24]		
INT																		○									90	228
JMP*1													○														93	229
JMPI*1	○*2	○*3	○*4	○*5	○	○	○	○	○	○	○	○	○														94	231
JMPS														○													95	232
JSRI	○*2	○*3	○*4	○*5	○	○	○	○	○	○	○	○	○														97	234
JSRS														○													98	235
LDC*1	○*2	○*3	○*4	○*5	○	○	○	○	○	○	○	○	○		○	○											99	235
LDIPL																		○									101	239
MAX	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○	○	○												102	239
MIN	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○	○	○												103	241
MOV*1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	104	243
MOVA	○*8		○*9		○		○	○	○	○	○	○	○														106	252
MOVDir	○*10	○*11	○*12	○*13		○	○	○	○	○	○	○	○														107	253
MOVX	○*8		○*9		○	○	○	○	○	○	○	○	○	○					○	○	○	○	○	○	○	○	108	255
MUL	○*6	○	○*7		○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	109	255
MULEX				○*5	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	110	257
MULU	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	111	257
NEG	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	112	259
NOT	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	114	260
OR	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	115	260
POP	○*6	○	○*7	○	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	117	263
POPM*1	○	○	○	○																							119	264
PUSH	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○		○	○	○	○	○	○	○	○	120	265
PUSHA							○	○	○	○	○	○	○														121	267

*1　特定命令アドレッシングをもちます。
*2　R0/R2R0を選択できます。
*3　R2だけ選択できます。
*4　R1/R3R1を選択できます。
*5　R3だけ選択できます。
*6　R0L/R0を選択できます。
*7　R1L/R1を選択できます。
*8　R2R0だけ選択できます。
*9　R3R1だけ選択できます。
*10　R0Lだけ選択できます。
*11　R0Hだけ選択できます。
*12　R1Lだけ選択できます。
*13　R1H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/R2R0	R0H/R2/-	R1L/R1/R3R1	R1H/R3/-	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24	#IMM8	#IMM16	#IMM24	#IMM32	#IMM	[[An]]	[dsp:8[An]]	[dsp:8[SB/FB]]	[dsp:16[An]]	[dsp:16[SB/FB]]	[dsp:24[An]]	[abs16]	[abs24]		
PUSHM*1	○	○	○	○																							123	268
ROLC	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	126	270
RORC	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	127	270
ROT	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○					○	○	○	○	○	○	○	○	○	128	271
SBB	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○	○	○												130	273
SBJNZ*1	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○					○									131	275
SCCnd	○*4	○*5	○*6	○*7	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	132	276
SHA	○	○	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	○	134	278
SHL	○	○	○	○	○	○	○	○	○	○	○	○	○					○	○	○	○	○	○	○	○	○	136	281
STC*1	○*4	○*5	○*6	○*7	○	○	○	○	○	○	○	○	○														144	286
STCTX*1	○	○	○	○																							145	288
STNZ	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	146	288
STZ	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	147	289
STZX	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	148	289
SUB	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○		○	○	○	○	○	○	○	○	149	290
SUBX	○*8	○*9	○*10	○*11	○	○	○	○	○	○	○	○	○	○					○	○	○	○	○	○	○	○	151	294
TST	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○	○	○												152	296
XCHG	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○						○	○	○	○	○	○	○	○	156	299
XOR	○*2	○	○	○*3	○	○	○	○	○	○	○	○	○	○	○				○	○	○	○	○	○	○	○	157	299

*1　特定命令アドレッシングをもちます。
*2　R0L/R0を選択できます。
*3　R1L/R1を選択できます。
*4　R0だけを選択できます。
*5　R2だけ選択できます。
*6　R1だけ選択できます。
*7　R3だけ選択できます。
*8　R0L/R2R0を選択できます。
*9　R0Hだけ選択できます。
*10　R1L/R3R1を選択できます。
*11　R1Hだけ選択できます。

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
	label	SB/FB	ISP/USP	FLG	INTB	SVP/VCT	SVF	DMD0/DMD1	DCT0/DCT1	DRC0/DRC1	DMA0/DMA1	DRA0/DRA1	DSA0/DSA1		
ADD[*1]			○											**46**	**176**
ADJNZ[*1]	○													**49**	**185**
J*Cnd*	○													**92**	**229**
JMP[*1]	○													**93**	**229**
JSR[*1]	○													**96**	**233**
LDC[*1]		○	○	○	○	○	○	○	○	○	○	○	○	**99**	**235**
POPC		○	○	○	○	○	○	○	○					**118**	**263**
POPM[*1]		○												**119**	**264**
PUSHC		○	○	○	○	○	○	○	○					**122**	**267**
PUSHM[*1]		○												**123**	**268**
SBJNZ[*1]	○													**131**	**275**
STC[*1]		○	○	○	○	○	○	○	○	○	○	○	○	**144**	**286**

*1　一般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（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,R0L/R0H	bit,R1L/R1H	bit,An	bit,[An]	bit,base:11[An]	bit,base:11[SB/FB]	bit,base:19[An]	bit,base:19[SB/FB]	bit,base:27[An]	bit,base:27	bit,base:19	U/I/O/B/S/Z/D/C		
BAND	○	○	○	○	○	○	○	○	○	○	○		52	188
BCLR	○	○	○	○	○	○	○	○	○	○	○		53	188
BM*Cnd*	○	○	○	○	○	○	○	○	○	○	○	○	55	190
BNAND	○	○	○	○	○	○	○	○	○	○	○		56	192
BNOR	○	○	○	○	○	○	○	○	○	○	○		57	192
BNOT	○	○	○	○	○	○	○	○	○	○	○		58	193
BNTST	○	○	○	○	○	○	○	○	○	○	○		59	193
BNXOR	○	○	○	○	○	○	○	○	○	○	○		60	194
BOR	○	○	○	○	○	○	○	○	○	○	○		61	194
BSET	○	○	○	○	○	○	○	○	○	○	○		64	196
BTST	○	○	○	○	○	○	○	○	○	○	○		65	196
BTSTC	○	○	○	○	○	○	○	○	○	○	○		66	197
BTSTS	○	○	○	○	○	○	○	○	○	○	○		67	198
BXOR	○	○	○	○	○	○	○	○	○	○	○		68	198
FCLR												○	85	221
FSET												○	87	222

第１章

概要

1.1　M16C/80, M16C/70 シリーズの特長

M16C/80, M16C/70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106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。

乗算器を内蔵してますので､高速な乗算ができます。

■ M16C/80, M16C/70 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、または2本を組み合わせて32ビットレジスタとして使用可能）
アドレスレジスタ　24ビット×2本
ベースレジスタ　24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

● 16M バイトのリニアなアドレス空間

分岐距離に応じた相対ジャンプ命令

●高速な命令実行時間

最短1サイクル命令：106命令中39命令が1サイクル命令をもつ

■スピード性能

スピード性能を表1.1.1に示します。

表 1.1.1　スピード性能

	M16C/80 シリーズ 動作周波数:20MHz	M16C/70 シリーズ 動作周波数:34MHz
レジスタ間転送	50ns	29ns
レジスターメモリ間転送	100ns	58ns
レジスタ間加減算	50ns	29ns
8ビット×8ビットレジスタ間演算	150ns	88.3ns
16ビット×16ビットレジスタ間演算	150ns	88.3ns
16ビット÷8ビットレジスタ間演算	0.9 μs	0.53 μs
32ビット÷16ビットレジスタ間演算	1.2 μs	0.71 μs

1.2 アドレス空間

アドレス空間を図 1.2.1 に示します。

000000$_{16}$ 番地～ 0003FF$_{16}$ 番地は SFR（スペシャルファンクションレジスタ）領域です。

000400$_{16}$ 番地以降はメモリ領域です。内部 RAM、ROM の番地は品種によって異なりますので、詳細は各グループのデータシートをご覧ください。

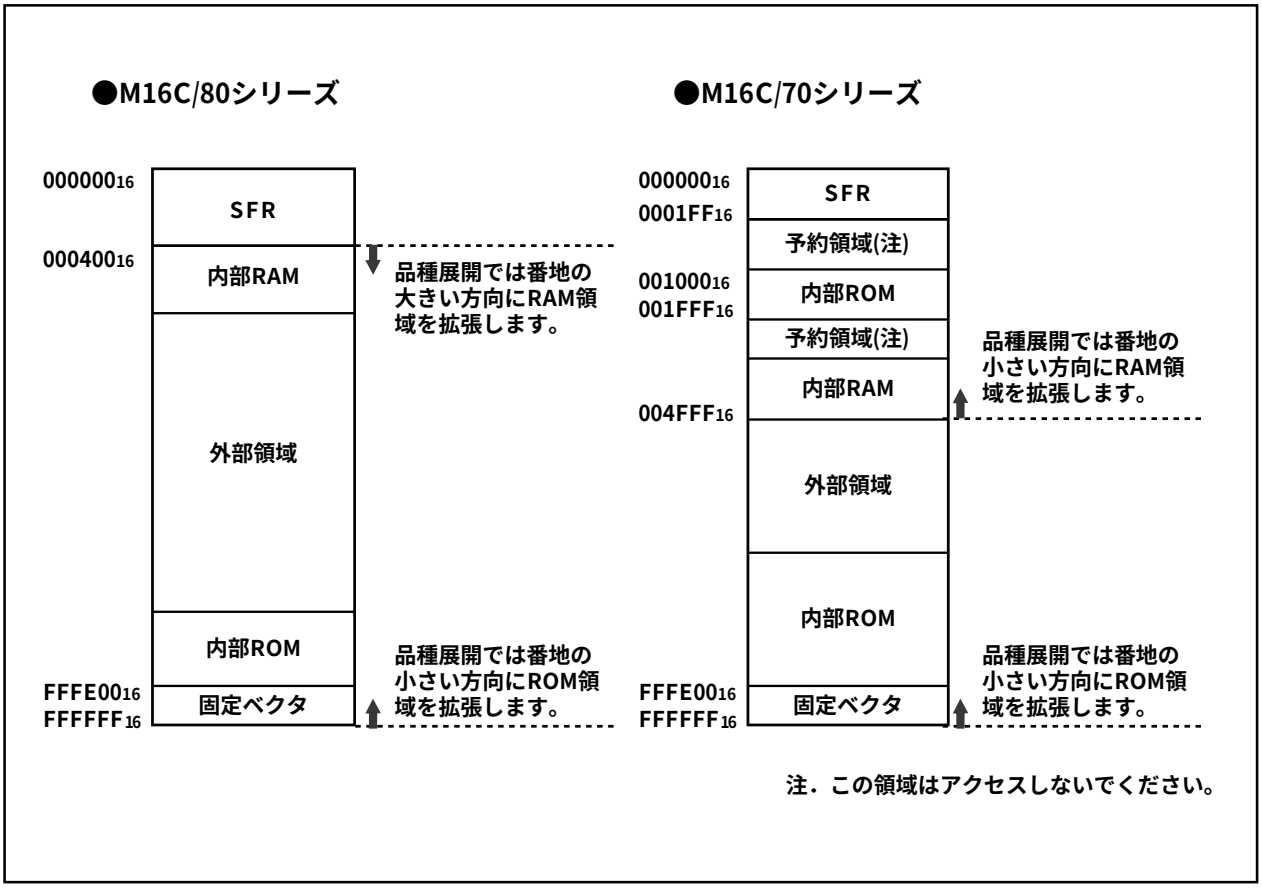


図 1.2.1 アドレス空間

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28個のレジスタがあります。これらのうち、R0、R1、R2、R3、A0、A1、SB、FBの8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。

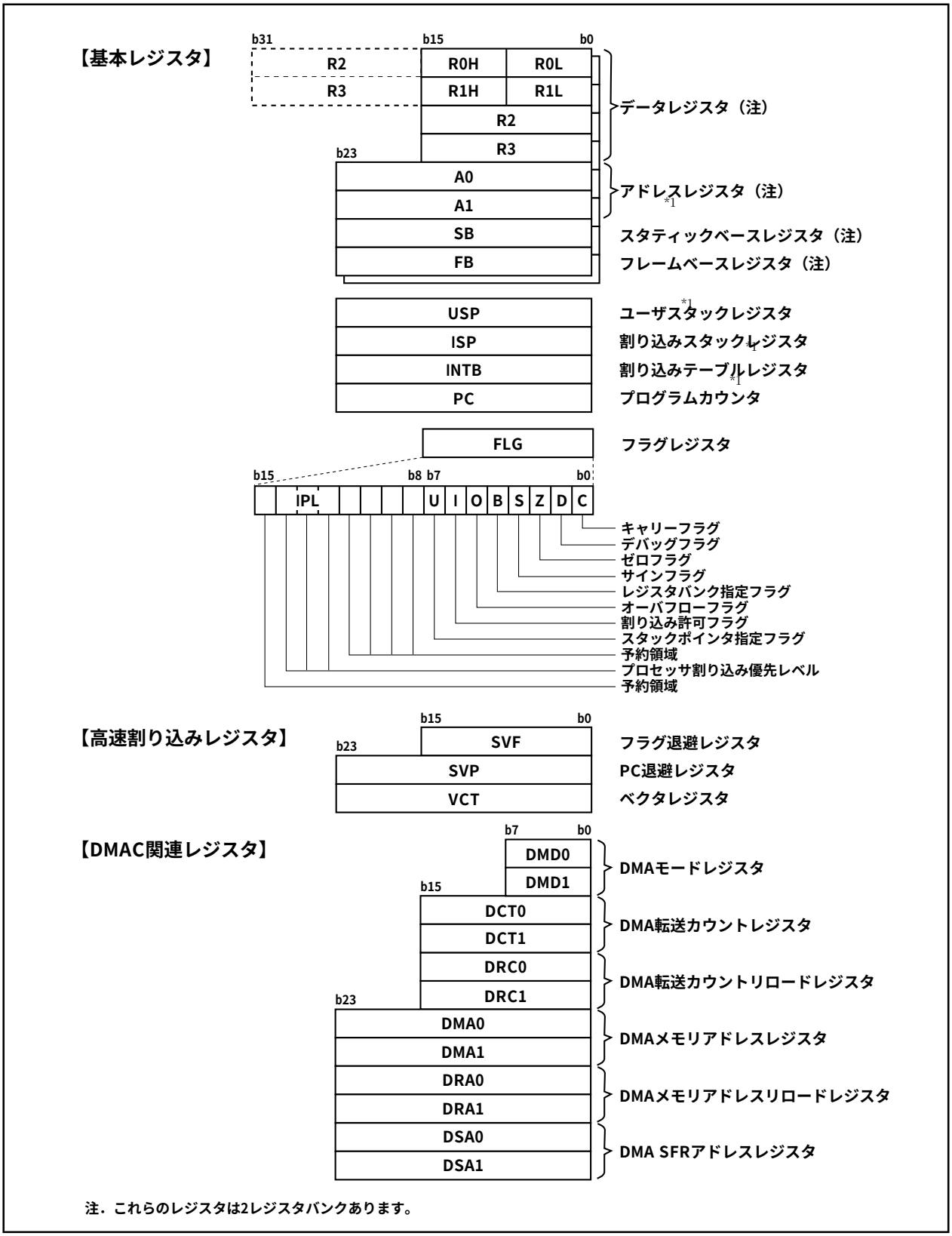


図 1.3.1　中央演算処理装置のレジスタ構成

(1) データレジスタ(R0/R0H/R0L/R1/R1H/R1L/R2/R3/R2R0/R3R1)

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できます。また、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。

(2) アドレスレジスタ(A0/A1)

アドレスレジスタ(A0/A1)は24ビットで構成されており、データレジスタと同等の機能をもちます。また、アドレスレジスタ間接アドレッシングおよびアドレスレジスタ相対アドレッシングに使用します。

(3) スタティックベースレジスタ(SB)

スタティックベースレジスタ(SB)は24ビットで構成されており、SB相対アドレッシングに使用します。

(4) フレームベースレジスタ(FB)

フレームベースレジスタ(FB)は24ビットで構成されており、FB相対アドレッシングに使用します。

(5) プログラムカウンタ(PC)

プログラムカウンタ(PC)は24ビットで構成されており、次に実行する命令の番地を示します。

(6) 割り込みテーブルレジスタ(INTB)

割り込みテーブルレジスタ(INTB)は24ビットで構成されており、割り込みベクタテーブルの先頭番地を示します。

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

スタックポインタは、ユーザスタックポインタ(USP)と割り込みスタックポインタ(ISP)の2種類があり、共に24ビットで構成されています。

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

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

USP, ISPには偶数を設定してください。偶数を設定した方が実行効率が良くなります。

(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。

(9) フラグ退避レジスタ(SVF)

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

(10) PC 退避レジスタ(SVP)

PC退避レジスタ(SVP)は24ビットで構成されており、高速割り込み発生時プログラムカウンタを退避させるのに使用します。

(11) ベクタレジスタ(VCT)

ベクタレジスタ(VCT)は 24 ビットで構成されており、高速割り込み発生時飛び先番地を示します。

(12) DMA モードレジスタ(DMD0/DMD1)

DMA モードレジスタ(DMD0/DMD1)は 8 ビットで構成されており、DMA の転送モードなどを設定するレジスタです。

(13) DMA 転送カウントレジスタ(DCT0/DCT1)

DMA 転送カウントレジスタ(DCT0/DCT1)は 16 ビットで構成されており、DMA の転送回数を設定するレジスタです

(14) DMA 転送カウントリロードレジスタ(DRC0/DRC1)

DMA 転送カウントリロードレジスタ(DRC0/DRC1)は 16 ビットで構成されており、DMA 転送カウントレジスタのリロードレジスタです。

(15) DMA メモリアドレスレジスタ(DMA0/DMA1)

DMA メモリアドレスレジスタ(DMA0/DMA1)は24ビットで構成されており、DMA の転送元あるいは転送先のメモリアドレスを設定するレジスタです。

(16) DMA SFR アドレスレジスタ(DSA0/DSA1)

DMA SFR アドレスレジスタ(DSA0/DSA1)は 24 ビットで構成されており、DMA 転送の転送元あるいは転送先の固定アドレスを設定するレジスタです。

(17) DMA メモリアドレスリロードレジスタ(DRA0/DRA1)

DMA メモリアドレスリロードレジスタ(DRA0/DRA1)は 24 ビットで構成されており、DMA メモリアドレスレジスタへのリロードレジスタです。

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)　ビット０：キャリーフラグ（C フラグ）

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

(2)　ビット１：デバッグフラグ（D フラグ）

シングルステップ割り込みを許可するフラグです。

このフラグが“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。

(3)　ビット２：ゼロフラグ（Z フラグ）

演算の結果が０のとき“１”になり、それ以外のとき“０”になります。

(4)　ビット３：サインフラグ（S フラグ）

演算の結果が負のとき“１”になり、それ以外のとき“０”になります。

(5)　ビット４：レジスタバンク指定フラグ（B フラグ）

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

(6)　ビット５：オーバフローフラグ（O フラグ）

演算の結果がオーバフローしたときに“１”になります。

(7)　ビット６：割り込み許可フラグ（I フラグ）

マスカブル割り込みを許可するフラグです。

このフラグが“０”のとき割り込みは禁止され、“１”のとき許可されます。

割り込みを受け付けると、このフラグは“０”になります。

(8)　ビット７：スタックポインタ指定フラグ（U フラグ）

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

(9)　ビット８～ビット１１：予約領域

（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。

（11） ビット１５：予約領域

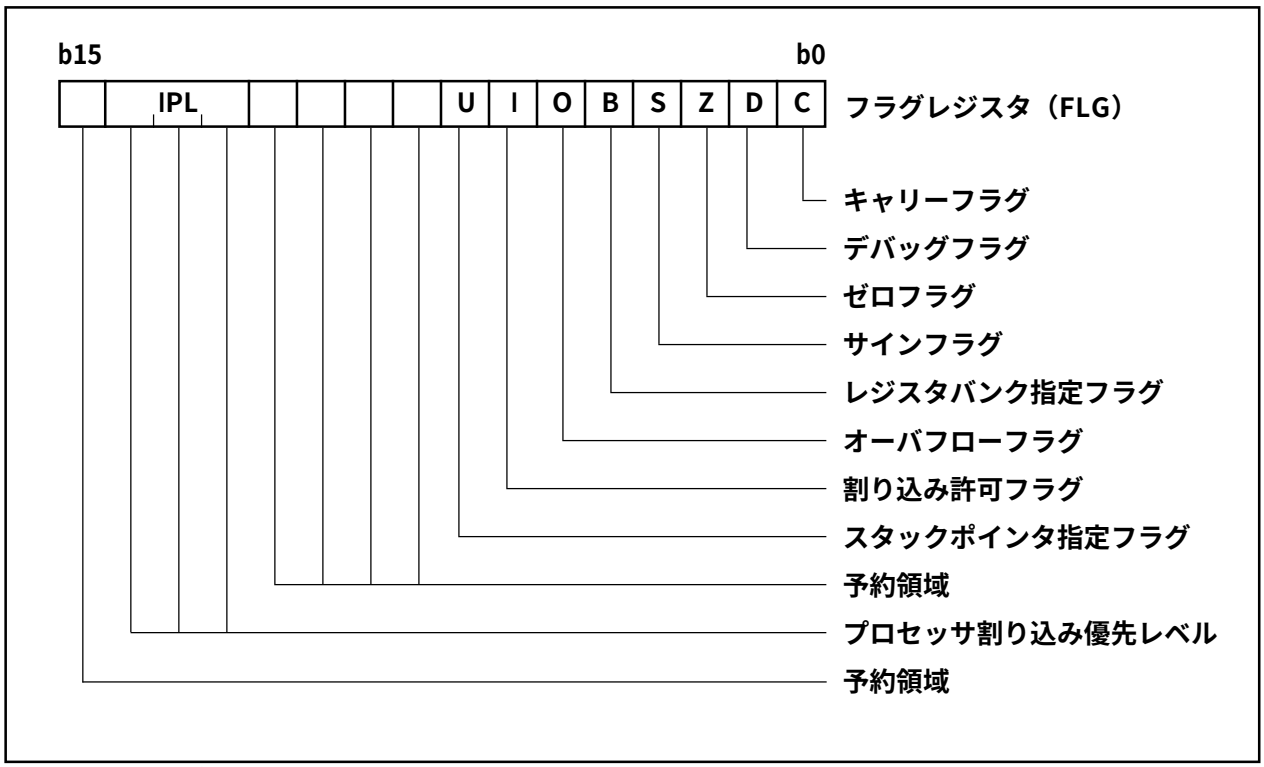
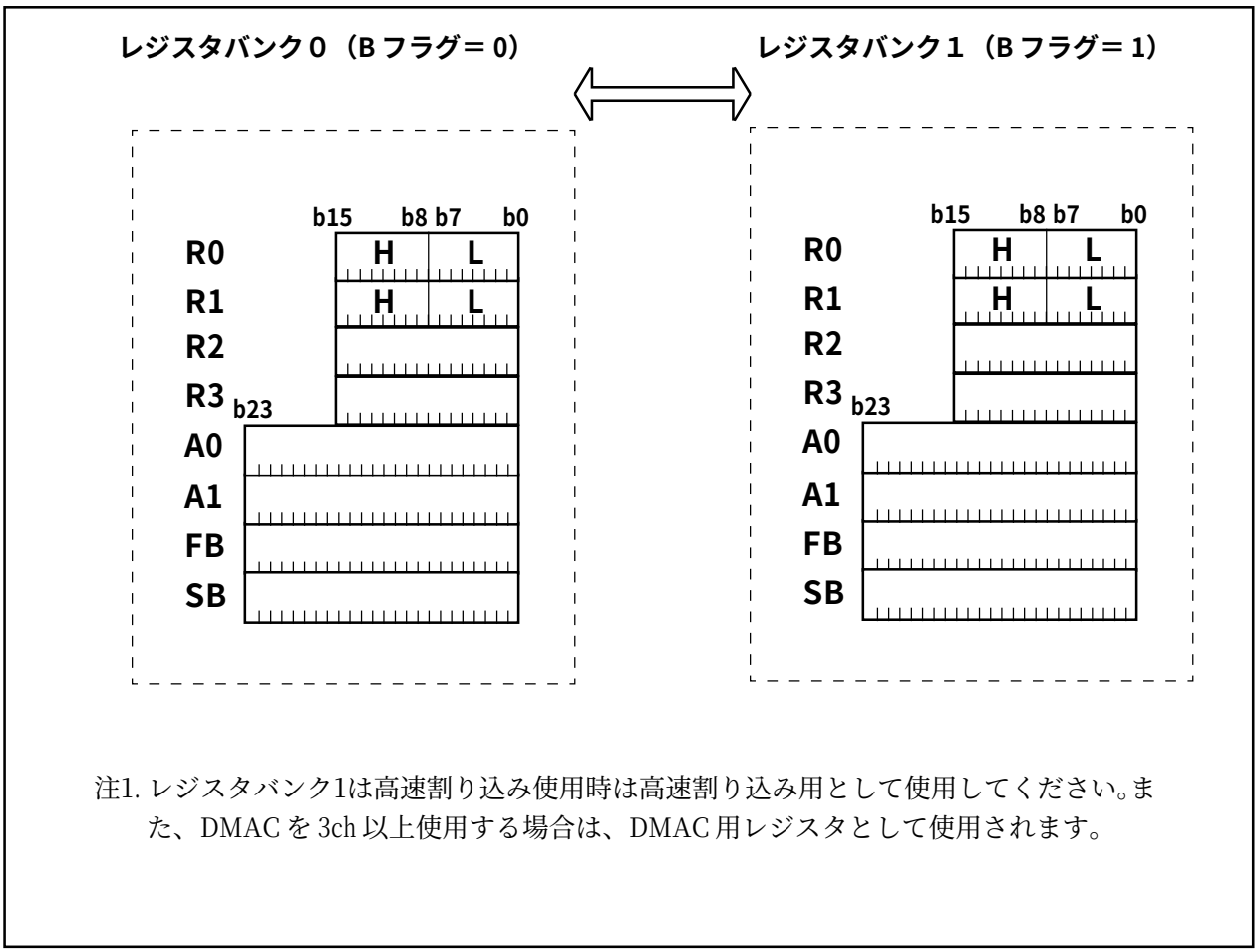


図 1.4.1　フラグレジスタ（FLG）の構成

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）、フレームベースレジスタ（FB）、およびスタティックベースレジスタ(SB)で構成されたレジスタバンクが2セットあります。レジスタバンクは、フラグレジスタ（FLG）のレジスタバンク指定フラグ（B フラグ）で切り替わります。

レジスタバンクの構成を図 1.5.1 に示します。



注1. レジスタバンク1は高速割り込み使用時は高速割り込み用として使用してください。また、DMAC を 3ch 以上使用する場合は、DMAC 用レジスタとして使用されます。

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）　：00000_{16}
・スタティックベースレジスタ（SB）　：00000_{16}
・フレームベースレジスタ（FB）　：00000_{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）　：00000_{16}
・割り込みスタックポインタ（ISP）　：00000_{16}
・フラグレジスタ（FLG）　：0000_{16}
・DMA モードレジスタ（DMD0/DMD1）　：00_{16}
・DMA 転送カウントレジスタ(DCT0/DCT1)　：不定
・DMA 転送カウントリロードレジスタ(DRC0/DRC1)　：不定
・DMA メモリアドレスレジスタ(DMA0/DMA1)　：不定
・DMA SFR アドレスレジスタ(DSA0/DSA1)　：不定
・DMA メモリアドレスリロードレジスタ(DRA0/DRA1)　：不定
・フラグ退避レジスタ(SVF)　：不定
・PC 退避レジスタ(SVP)　：不定
・ベクタレジスタ(VCT)　：不定

1.7 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。

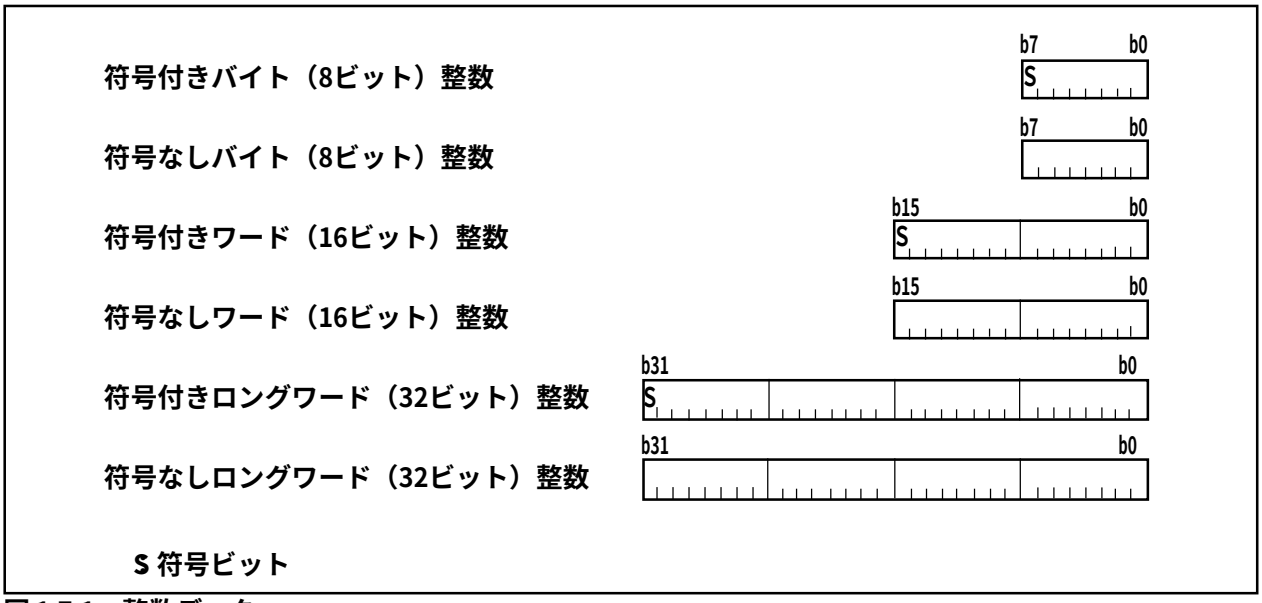


図 1.7.1 整数データ

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。

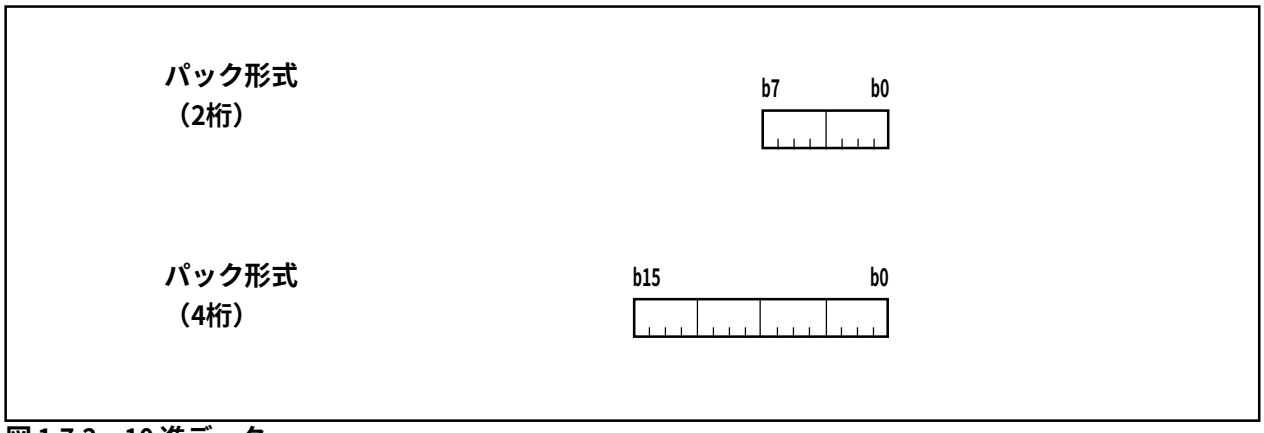


図 1.7.2 10進データ

1.7.3 ビット

(1) 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,RnH/bit,RnL/bit,An**）によって指定できます。**bit,RnH/RnL**は、データレジスタ（**RnH/RnL**）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。

bit,RnH/**bit,RnL/bit,An** の **bit** は 0 ～ 7 の範囲で指定できます。

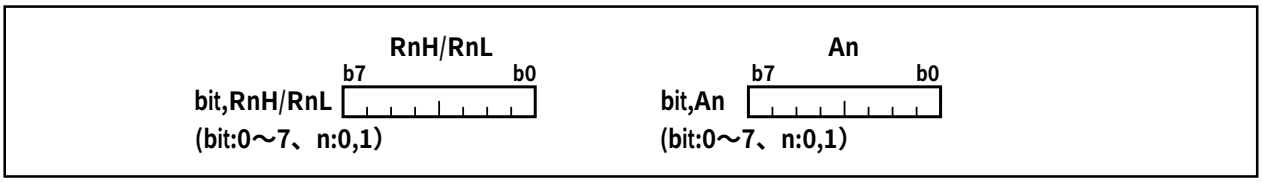


図 1.7.3 レジスタのビット指定

(2) 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。

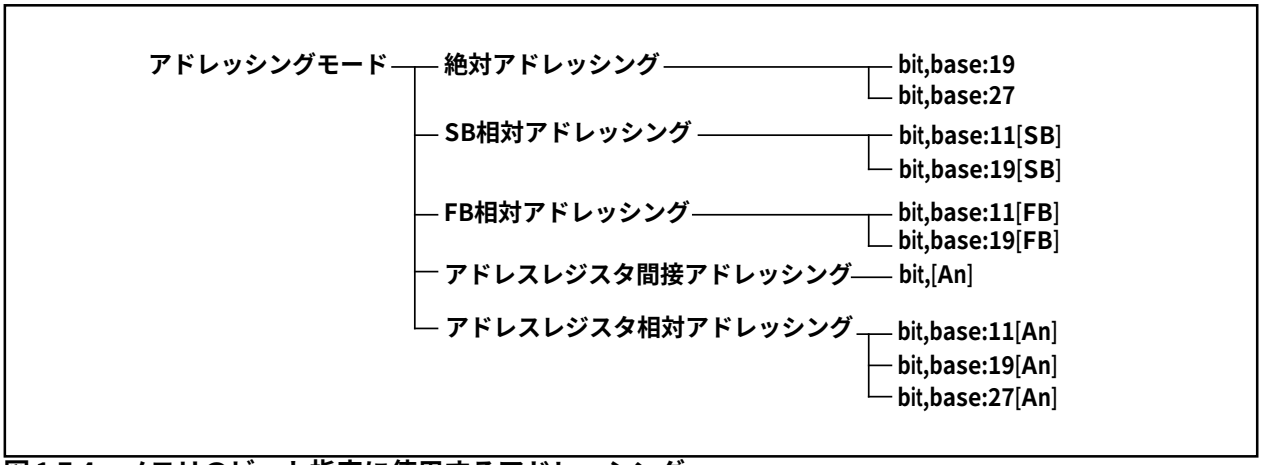


図 1.7.4 メモリのビット指定に使用するアドレッシング

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		アクセス範囲
	下限（番地）	上限（番地）	
bit,base:19	000000$_{16}$	00FFFF$_{16}$	
bit,base:27	000000$_{16}$	FFFFFF$_{16}$	
bit,base:11[SB]	[SB]	[SB]+000FF$_{16}$	000000$_{16}$ ～ FFFFFF$_{16}$
bit,base:19[SB]	[SB]	[SB]+0FFFF$_{16}$	000000$_{16}$ ～ FFFFFF$_{16}$
bit,base:11[FB]	[FB]－ 00008$0_{16}$	[FB]+00007F$_{16}$	000000$_{16}$ ～ FFFFFF$_{16}$
bit,base:19[FB]	[FB]－ 00800$0_{16}$	[FB]+007FFF$_{16}$	000000$_{16}$ ～ FFFFFF$_{16}$
bit,[An]	000000$_{16}$	FFFFFF$_{16}$	
bit,base:11[An]	[An]	[An]+0000FF$_{16}$	000000$_{16}$ ～ FFFFFF$_{16}$
bit,base:19[An]	[An]	[An]+00FFFF$_{16}$	000000$_{16}$ ～ FFFFFF$_{16}$
bit,base:27[An]	[An]	[An]+FFFFFF$_{16}$	000000$_{16}$ ～ FFFFFF$_{16}$

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。

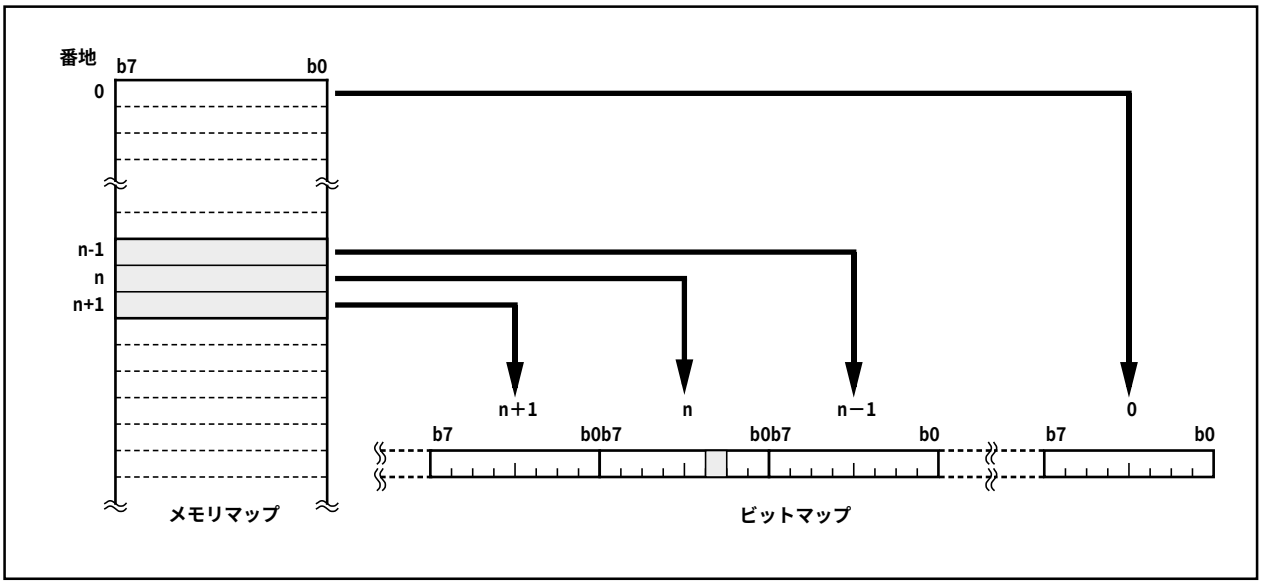


図 1.7.5　メモリマップとビットマップの関係

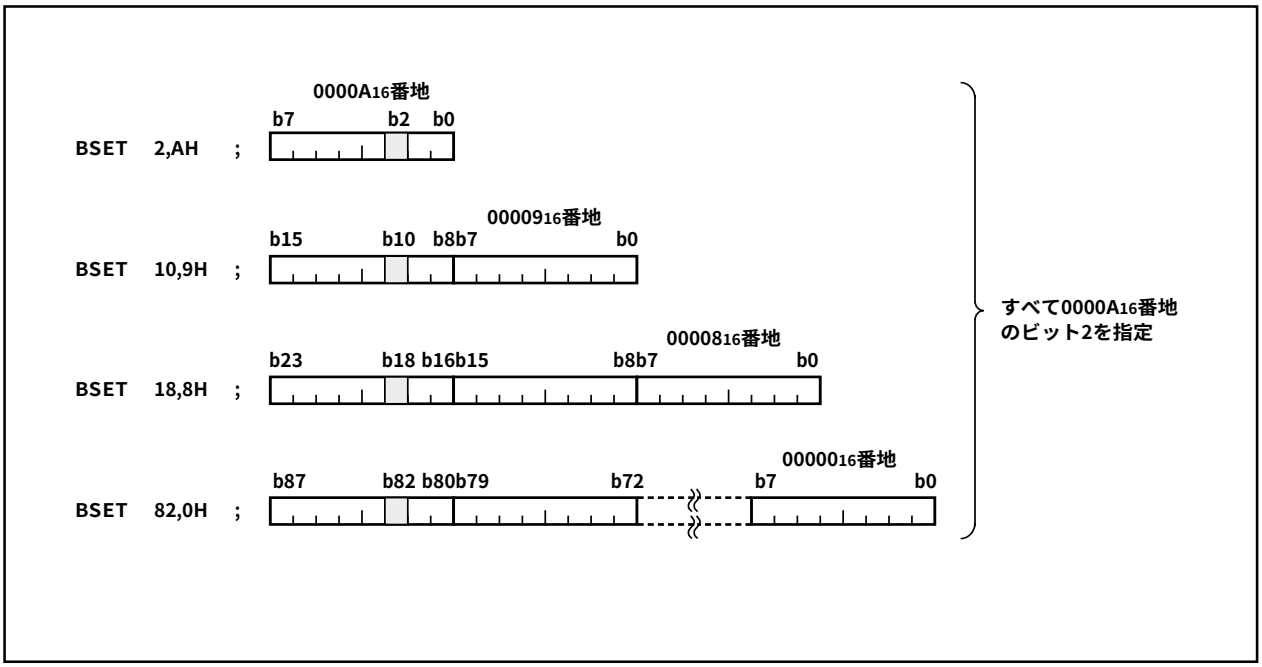


図 1.7.6　0000A16 番地のビット 2 の指定例

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、アドレスレジスタ（**An**）に設定したアドレスを基準とし、対象となるビット位置を **bit** に設定します。

アドレスレジスタ間接アドレッシングの場合、指定できるビット範囲は 0 ～ 7 です。

アドレスレジスタ相対アドレッシングの場合、アドレスレジスタ（**An**）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bit基準（0）とし、対象となるビット位置を **bit** 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)、文字列転送比較(SCMPU命令)、文字列転送(SMOVU命令)、文字列入力(SIN 命令)、文字列出力(SOUT 命令)の 7 種類で使用できます。

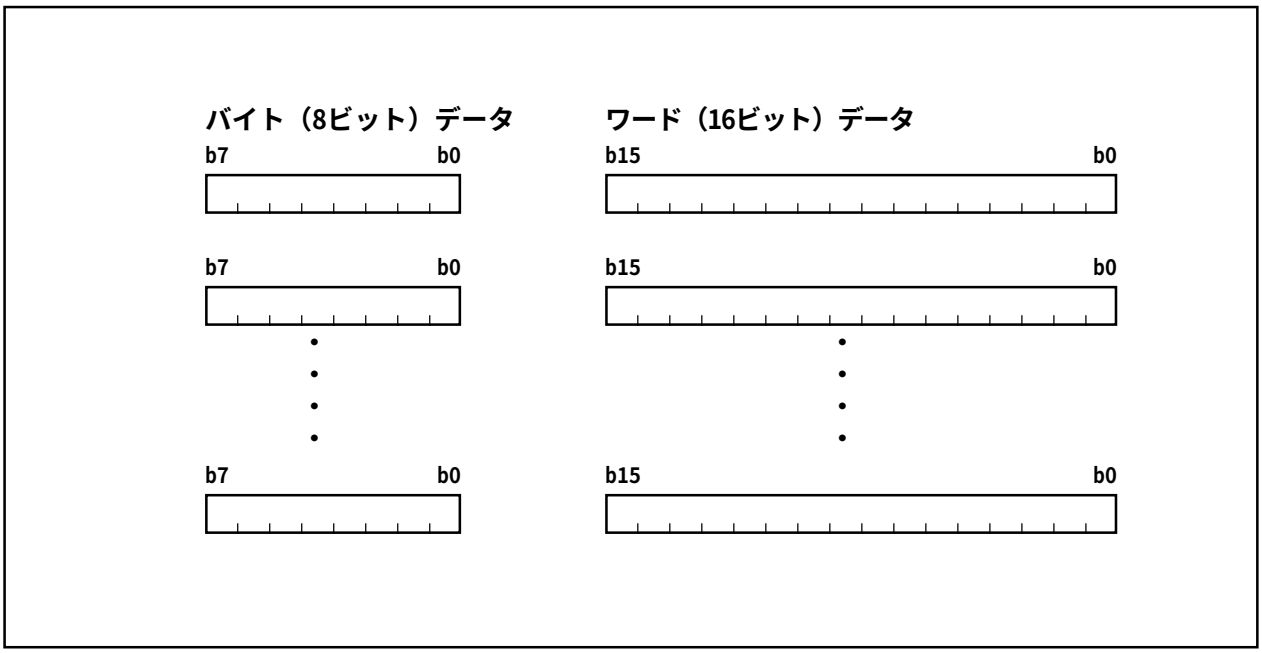


図 1.7.7 ストリングデータ

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。

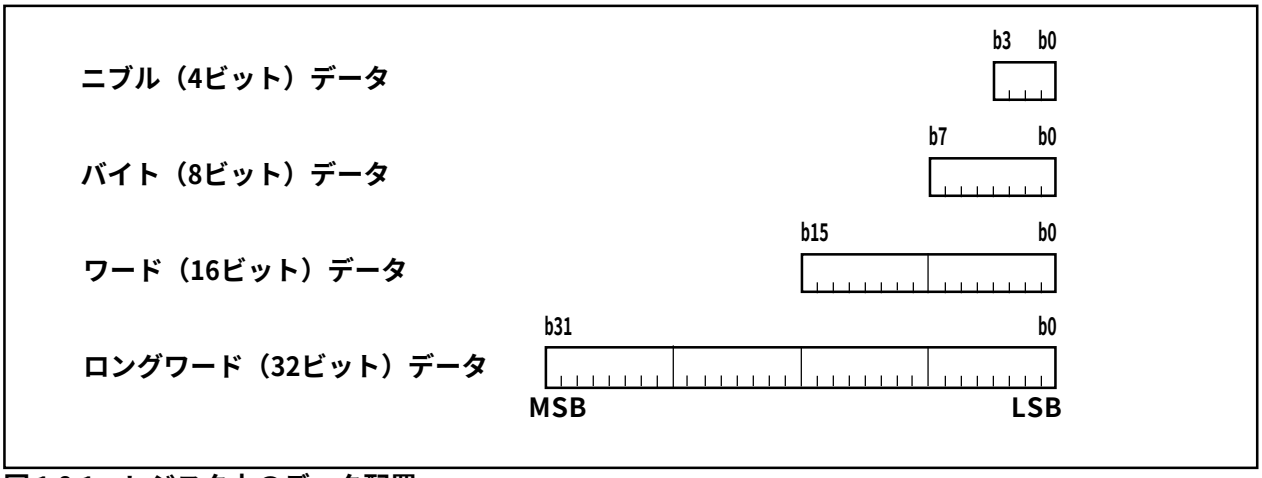


図 1.8.1 レジスタ上のデータ配置

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。

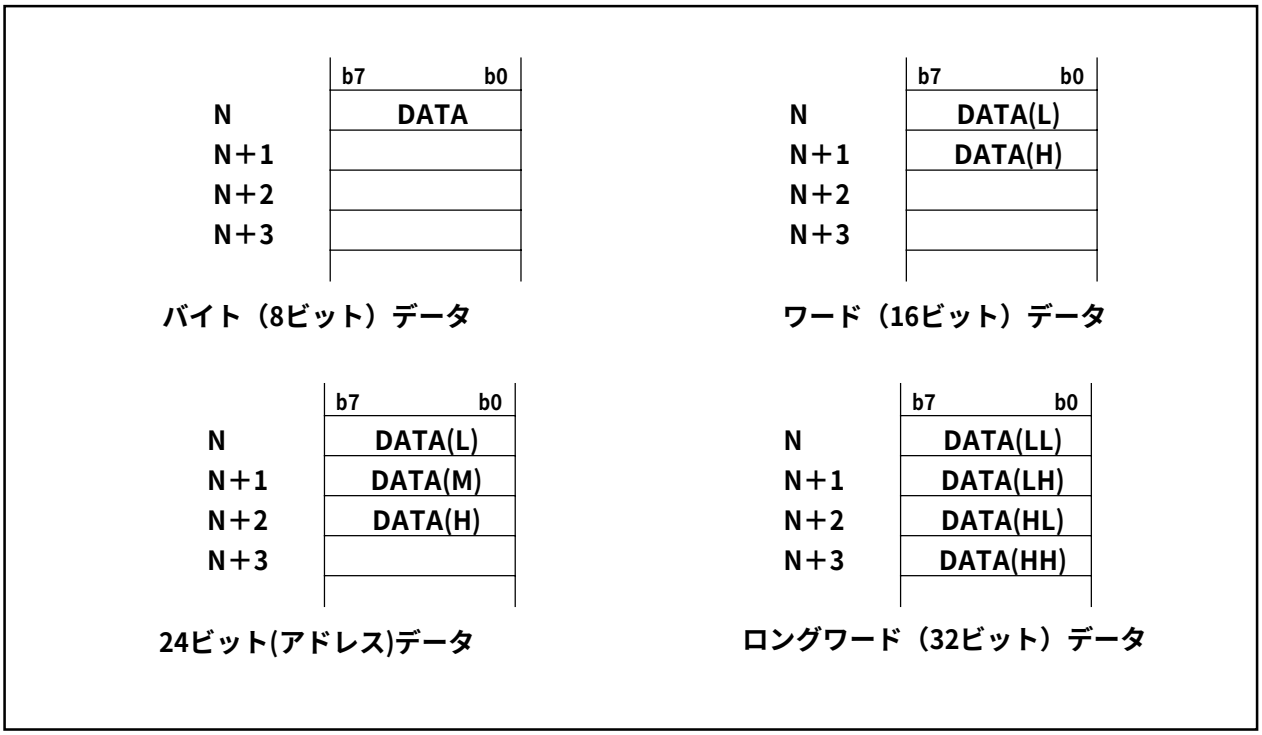


図 1.8.2 メモリ上のデータ配置

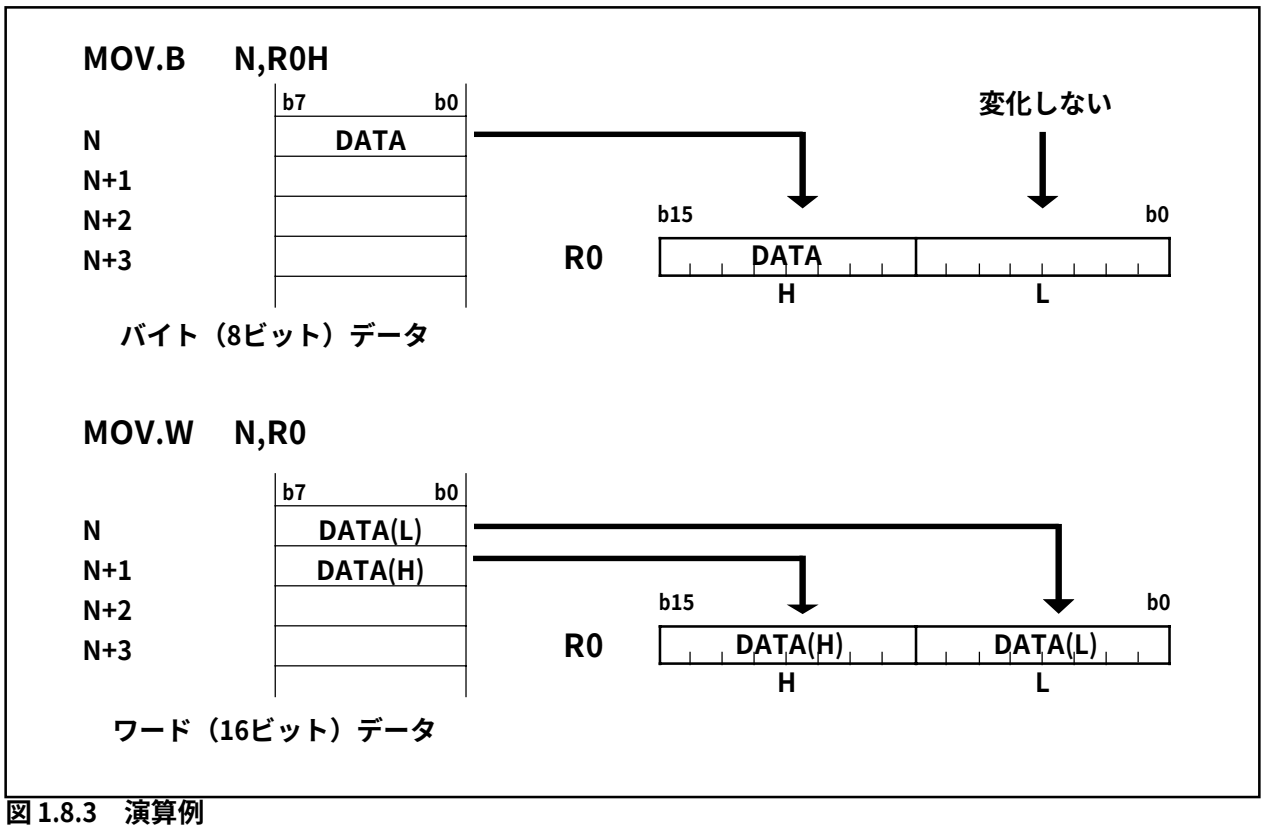


図 1.8.3 演算例

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) ジェネリック形式(:G)

ジェネリック形式のオペコードは2～3バイトです。このオペコードには、動作および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(2～3バイト)とsrcコード(0～4バイト)およびdestコード(0～3バイト)で構成されます。

(2) クイック形式(:Q)

クイック形式のオペコードは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(0～3バイト)で構成されます。

(3) ショート形式(:S)

ショート形式のオペコードは1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。

(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

ベクタテーブルには、スペシャルページベクタテーブルと割り込みベクタテーブルがあります。スペシャルページベクタテーブルは固定ベクタテーブルです。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。FFFE00$_{16}$番地からFFFFDB$_{16}$番地にスペシャルページベクタテーブル、FFFFDC$_{16}$番地からFFFFFF$_{16}$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。

スペシャルページベクタテーブルは、1ベクタテーブルに対して2バイトで構成されています。各ベクタテーブルには、サブルーチンの先頭アドレスの下位16ビットを設定します。また、1ベクタテーブルごとに、スペシャルページ番号（18～255）があり、JSRSおよびJMPS命令では、このスペシャルページ番号を使用します。

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。

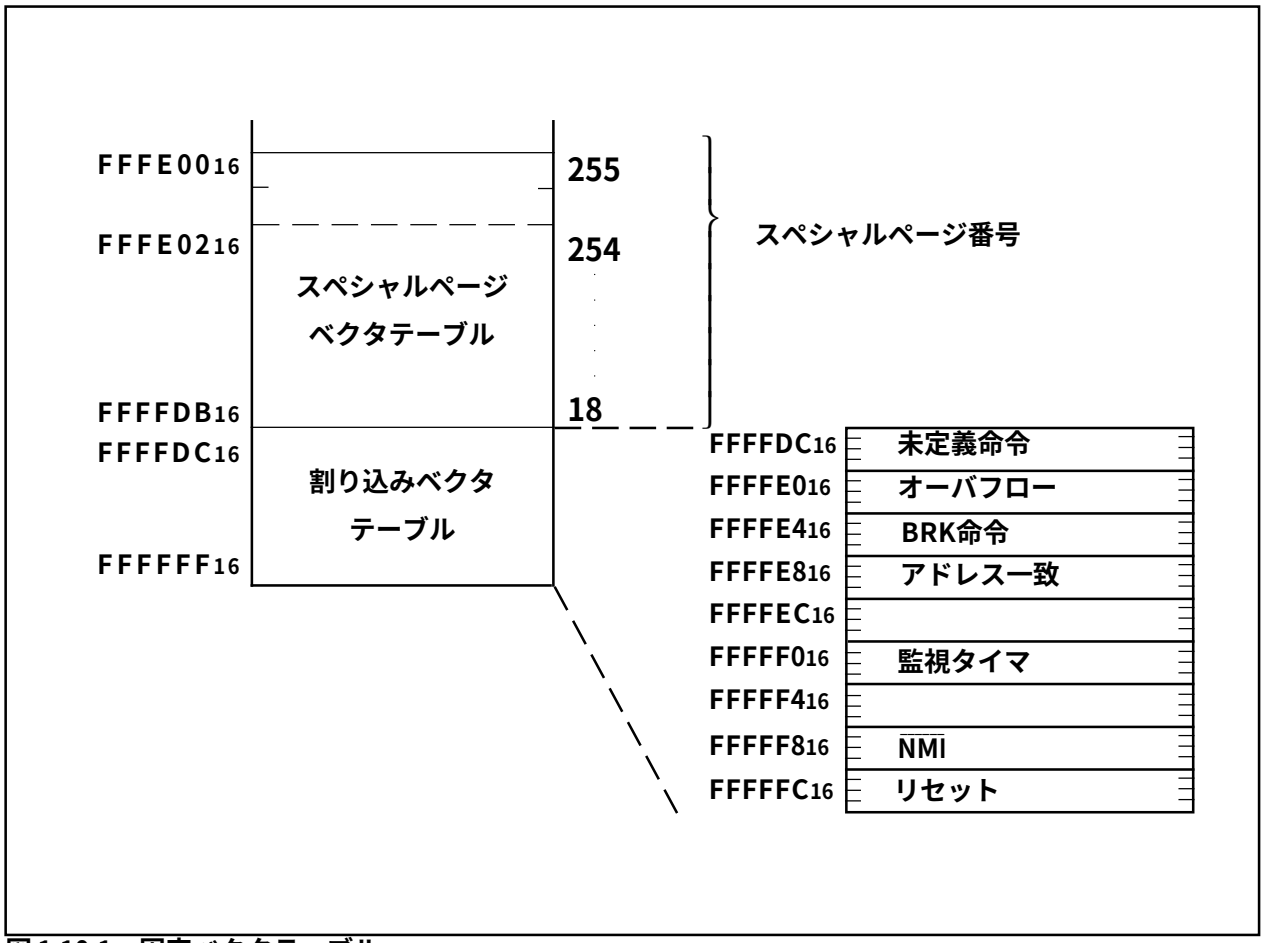


図1.10.1 固定ベクタテーブル

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も可変ベクタテーブルに割り当てられます。その割り当てについては、ソフトウエア割り込み番号の0側から割り当てられますが、品種によって数が異なることがあります。よって、INT命令割り込みを使用する場合、ソフトウエア割り込み番号63、62、61・・・の順番で使用することを推奨します。

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

ソフトウエア割り込み番号0～31では、割り込み要求受け付け時にスタックポインタ指定フラグ(Uフラグ)を退避し、Uフラグを“0”にして割り込みスタックポインタ(ISP)を選択した後、割り込みシーケンスを実行します。割り込みルーチンから復帰するときに割り込み要求受け付け前のUフラグが復帰されます。

ソフトウエア割り込み番号32～63では、スタックポインタは切り替わりません。

周辺I/O割り込みでは、ソフトウエア割り込み番号に関係なく、ソフトウエア割り込み番号0～31と同様に、割り込み要求受け付け時に割り込みスタックポインタ(ISP)を選択します。

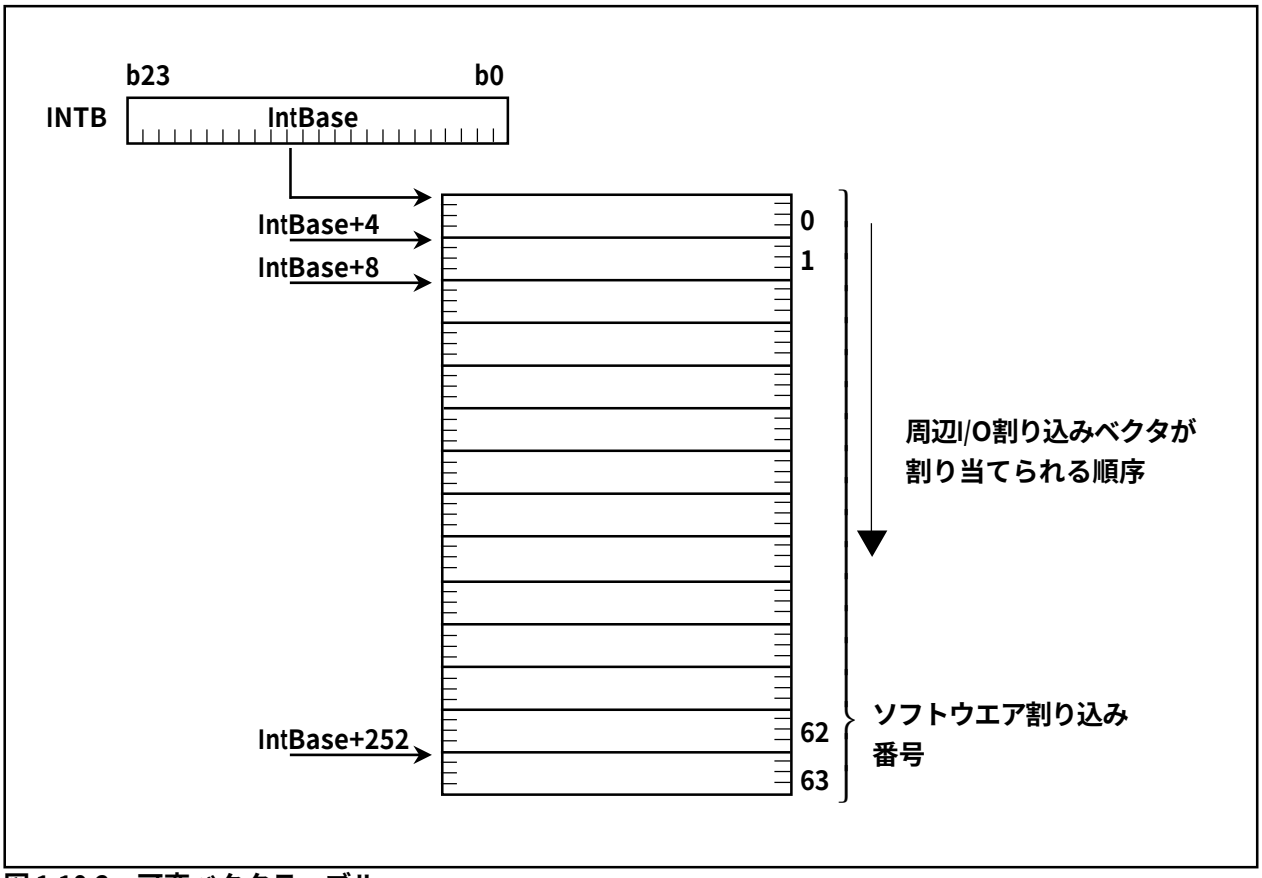


図1.10.2　可変ベクタテーブル

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4つのタイプがあります。

(1) 一般命令アドレッシング

00000016番地からFFFFFF16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) 間接命令アドレッシング

00000016番地からFFFFFF16番地までの領域をアクセスするアドレッシングです。
以下に間接命令アドレッシングの各名称を示します。

・絶対間接
・アドレスレジスタ二段間接
・アドレスレジスタ相対間接
・SB相対間接
・FB相対間接

(3) 特定命令アドレッシング

00000016番地からFFFFFF16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。

・専用レジスタ直接
・プログラムカウンタ相対

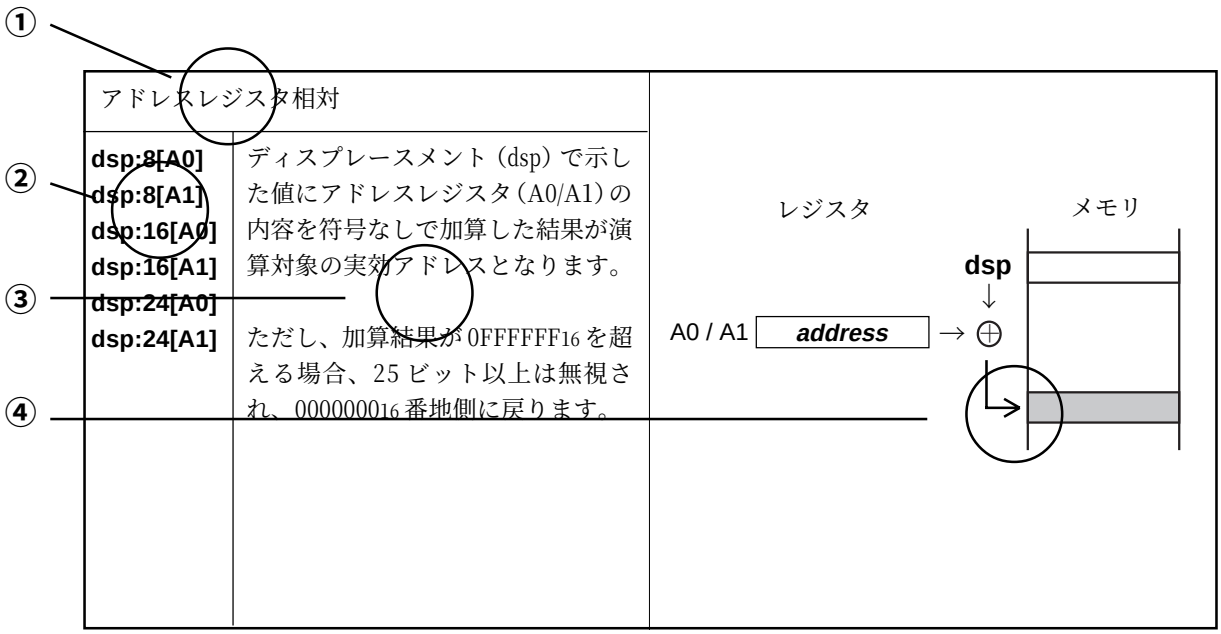
(4) ビット命令アドレッシング

00000016番地からFFFFFF16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。

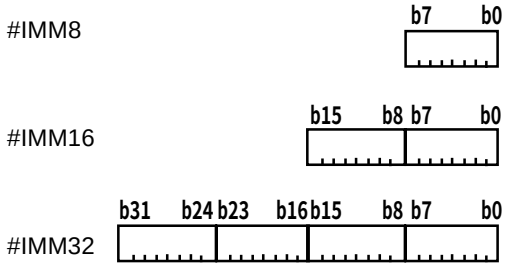
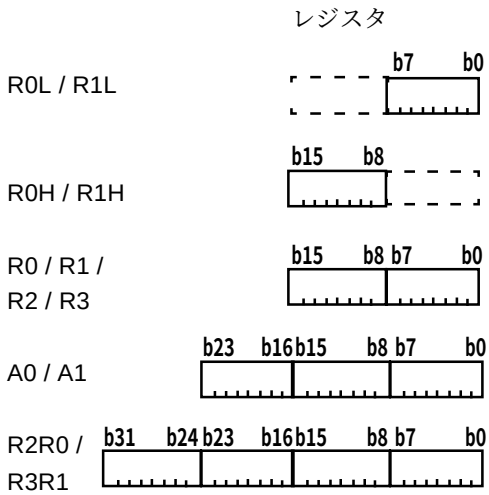
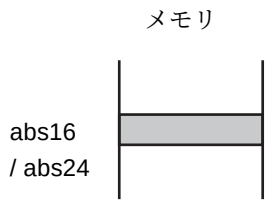
③解説

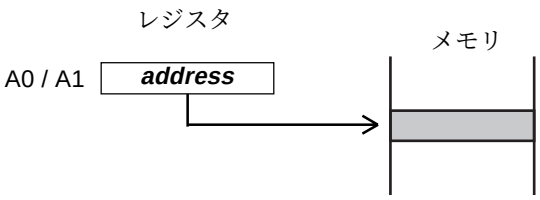
動作、実効アドレスの範囲を説明します。

④動作図

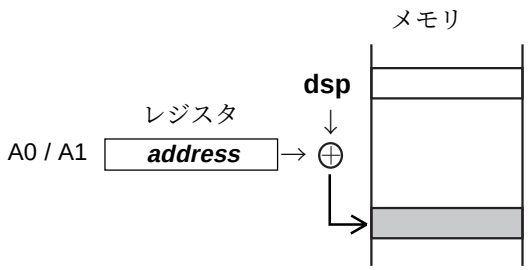
動作を図で説明します。

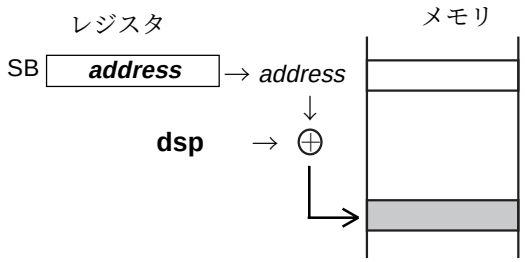
2.3 一般命令アドレッシング

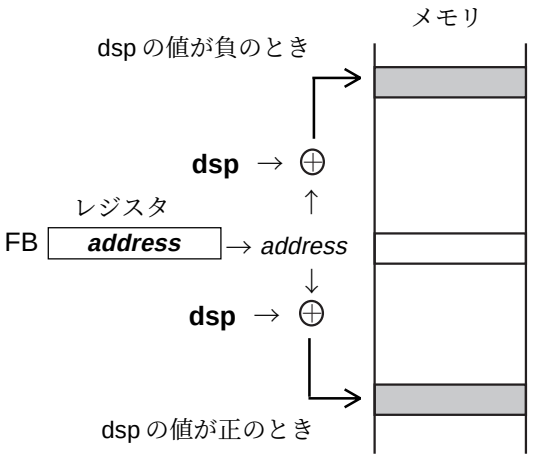
<table>
<tr><td colspan="2">即値</td><td rowspan="2"> 
</td></tr>
<tr><td>#IMM
#IMM8
#IMM16
#IMM32</td><td>#IMM で示した即値が演算の対象となります。</td></tr>
<tr><td colspan="2">レジスタ直接</td><td rowspan="2"> 
</td></tr>
<tr><td>R0L
R0H
R1L
R1H
R0
R1
R2
R3
A0
A1
R2R0
R3R1</td><td>指定したレジスタが演算の対象となります。</td></tr>
<tr><td colspan="2">絶対</td><td rowspan="2"> 
</td></tr>
<tr><td>abs16
abs24</td><td>abs で示した値が演算対象の実効アドレスとなります。

実効アドレスの範囲は、abs16 では 000000₁₆～000FFFF₁₆で、abs24 では 000000₁₆～0FFFFFF₁₆ 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。

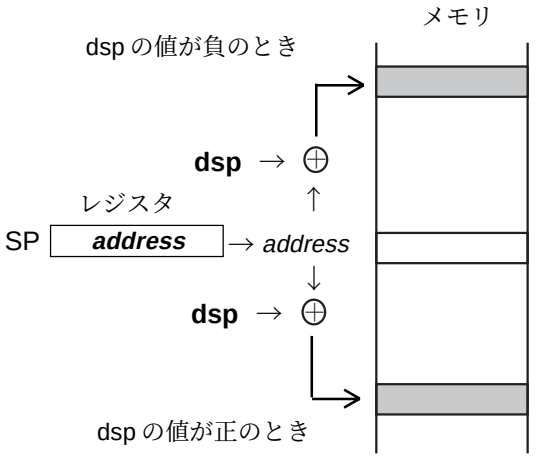
実効アドレスの範囲は、000000₁₆～0FFFFFF₁₆ です。</td></tr>
</table>

<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]
dsp:24[A0]
dsp:24[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。

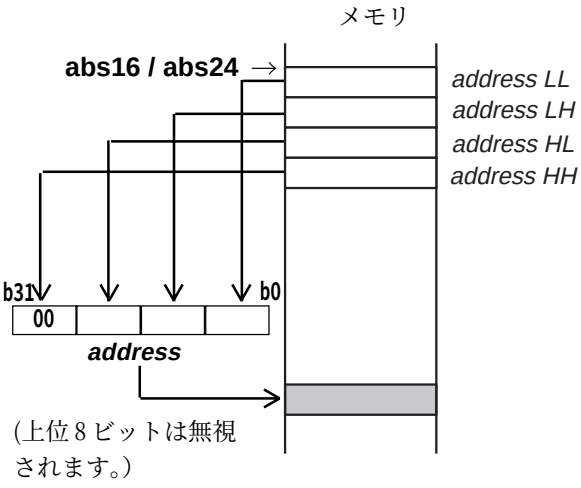
ただし、加算結果が0FFFFFF16を超える場合、25 ビット以上は無視され、0000000₁₆ 番地側に戻ります。</td></tr>
<tr><td colspan="2">SB 相対</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:8[SB]
dsp:16[SB]</td><td>スタティックベースレジスタ（SB）の内容で示したアドレスにディスプレースメント（dsp）で示した値を符号なしで加算した結果が演算対象の実効アドレスとなります。

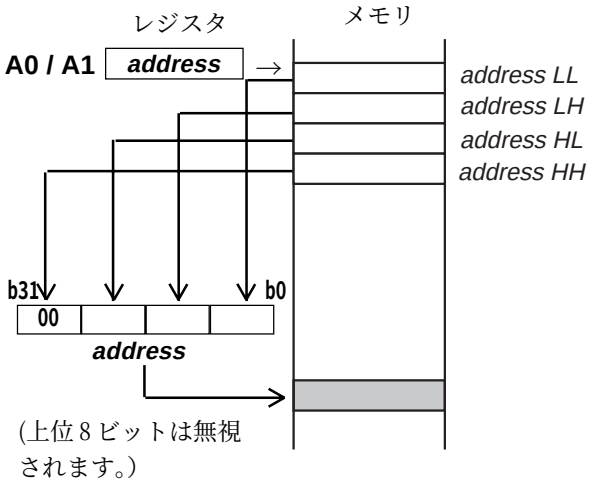
ただし、加算結果が0FFFFFF16を超える場合、25 ビット以上は無視され、0000000₁₆ 番地側に戻ります。</td></tr>
<tr><td colspan="2">FB 相対</td><td rowspan="2"> 
</td></tr>
<tr><td>dsp:8[FB]
dsp:16[FB]</td><td>フレームベースレジスタ（FB）の内容で示したアドレスにディスプレースメント（dsp）で示した値を符号付きで加算した結果が演算対象の実効アドレスとなります。

ただし、加算結果が 0000000₁₆ ～ 0FFFFFF16を超える場合、25ビット以上は無視され、000000₁₆ 番地側、または 0FFFFFF16 番地側に戻ります。</td></tr>
</table>

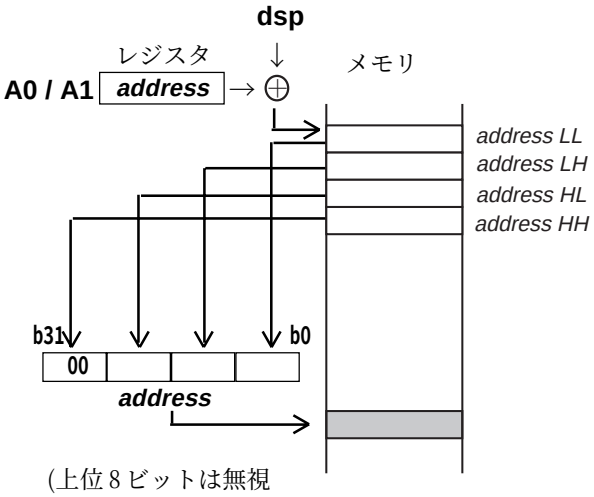
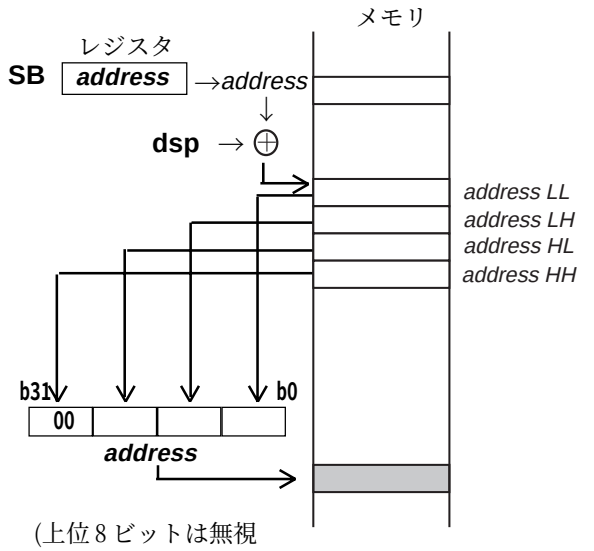
スタックポインタ相対		
dsp:8[SP]	スタックポインタ（SP）の内容で示したアドレスにディスプレースメント（dsp）で示した値を符号付きで加算した結果が演算対象の実効アドレスとなります。スタックポインタ（SP）は、Uフラグで示すスタックポインタが対象となります。 ただし、加算結果が 000000016 ～ 0FFFFFF16 を超える場合、25 ビット以上は無視され、000000016 番地側または 0FFFFFF16 番地側に戻ります。 このアドレッシングは MOV 命令で使用できます。	

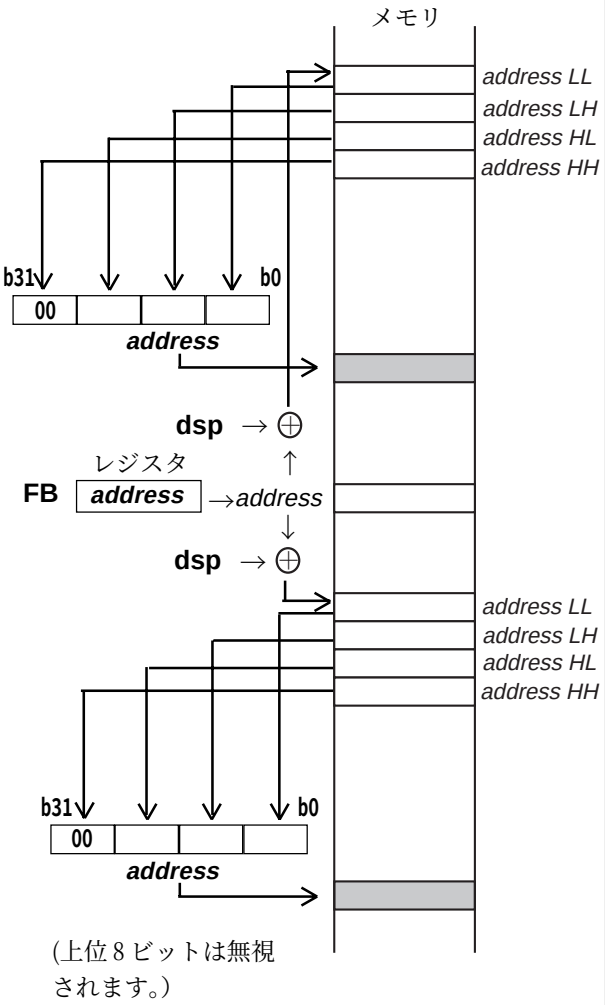
2.4 間接命令アドレッシング

<table>
<tr><td colspan="2">絶対間接</td><td rowspan="2"> 
</td></tr>
<tr><td>[abs16]
[abs24]</td><td>絶対アドレッシングの内容で示した4バイトの値が演算対象の実行アドレスとなります。

実行アドレス範囲は000000₁₆～0FFFFFF₁₆です。</td></tr>
<tr><td colspan="2">アドレスレジスタ二段間接</td><td rowspan="2"> 
</td></tr>
<tr><td>[[A0]]
[[A1]]</td><td>アドレスレジスタ間接の内容で示した4バイトの値が演算対象の実行アドレスとなります。

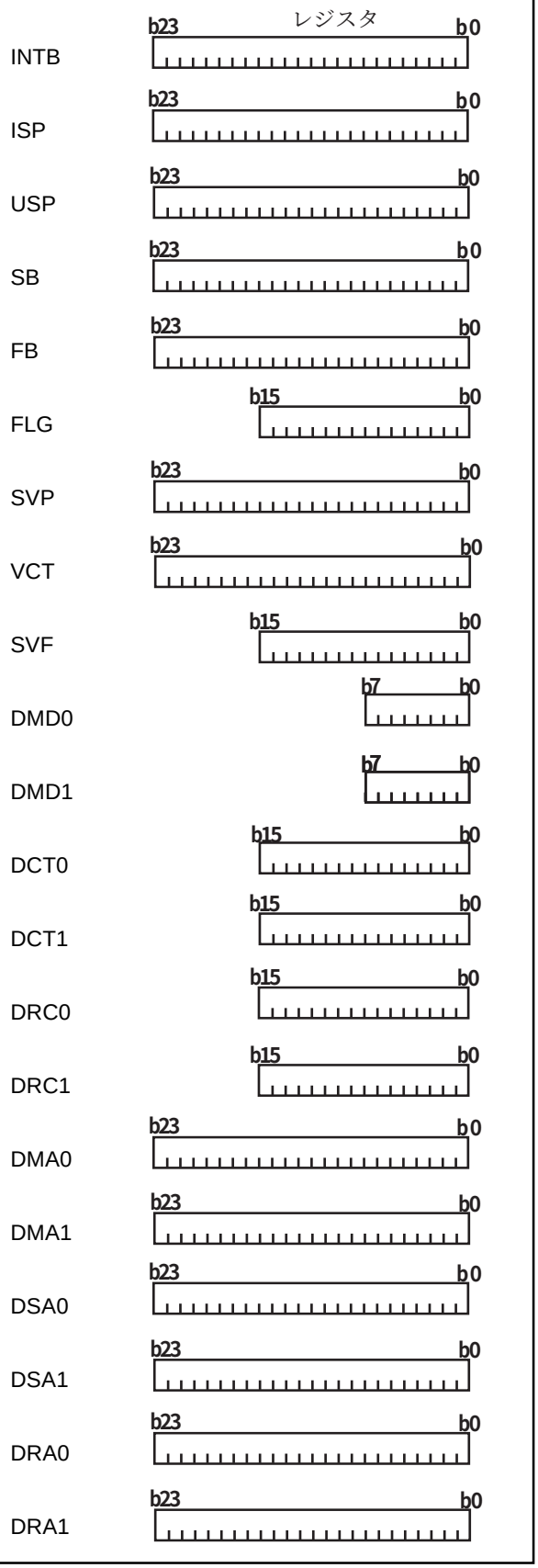
実行アドレス範囲は000000₁₆～0FFFFFF₁₆です。</td></tr>
</table>

アドレスレジスタ相対間接		 (上位 8 ビットは無視されます。)
[dsp:8[A0]] **[dsp:8[A1]]** **[dsp:16[A0]]** **[dsp:16[A1]]** **[dsp:24[A0]]** **[dsp:24[A1]]**	アドレスレジスタ相対の内容で示した4バイトの値が実行アドレスとなります。 実行アドレスの範囲は、000000$_{16}$ ～ 0FFFFFF$_{16}$ です。	
SB 相対間接		 (上位 8 ビットは無視されます。)
[dsp:8[SB]] **[dsp:16[SB]]**	SB 相対の内容で示した 4 バイトの値が演算対象の実行アドレスとなります。 実行アドレス範囲は 000000$_{16}$ ～ 0FFFFFF$_{16}$ です。	

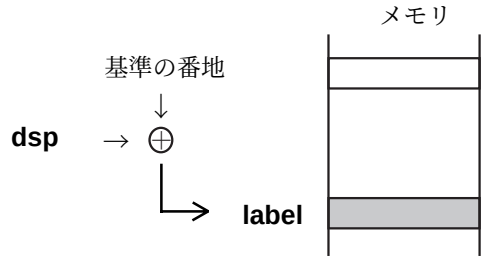
FB 相対間接		
[dsp:8[FB]] **[dsp:16[FB]]**	FB 相対の内容で示した 4 バイトの値が演算対象の実行アドレスとなります。 実行アドレス範囲は 000000$_{16}$ ～ 0FFFFFF$_{16}$ です。	

2.5 特定命令アドレッシング

専用レジスタ直接	
INTB **ISP** **SP** **SB** **FB** **FLG** **SVP** **VCT** **SVF** **DMD0** **DMD1** **DCT0** **DCT1** **DRC0** **DRC1** **DMA0** **DMA1** **DSA0** **DSA1** **DRA0** **DRA1**	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 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。

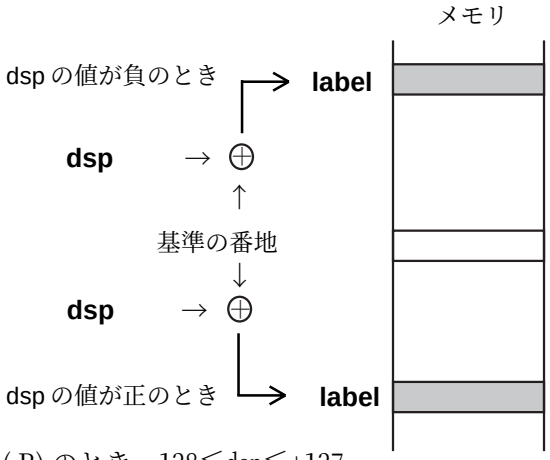


<table>
<tr><td colspan="2">プログラムカウンタ相対</td><td rowspan="2"></td></tr>
<tr><td>label</td><td>○分岐距離指定子（.length）が（.S）の場合
基準の番地にディスプレースメント（dsp）で示した値を符号なしで加算した結果が実効アドレスとなります。

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 000000₁₆ ～ 0FFFFFF₁₆を超える場合、25ビット以上は無視され、000000₁₆ 番地または 0FFFFFF₁₆ 番地側に戻ります。

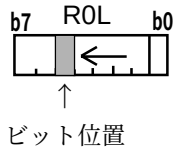
このアドレッシングは、JMP、JSR 命令で使用できます。</td><td> 

(.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>

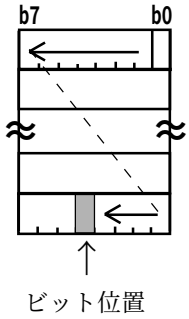
2.6 ビット命令アドレッシング

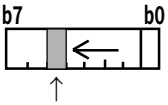
このアドレッシングは以下の命令で使用できます。

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 , R0L
 
</td></tr>
<tr><td>bit,R0L
bit,R0H
bit,R1L
bit,R1H
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。

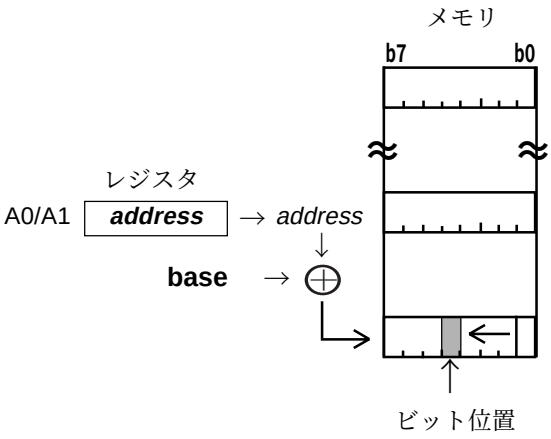
ビット位置（bit）は 0～7 が指定できます。

A0、A1 では下位 8 ビットが指定できます。</td></tr>
<tr><td colspan="2">絶対</td><td rowspan="2">base
 
</td></tr>
<tr><td>bit,base:19
bit,base:27</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。

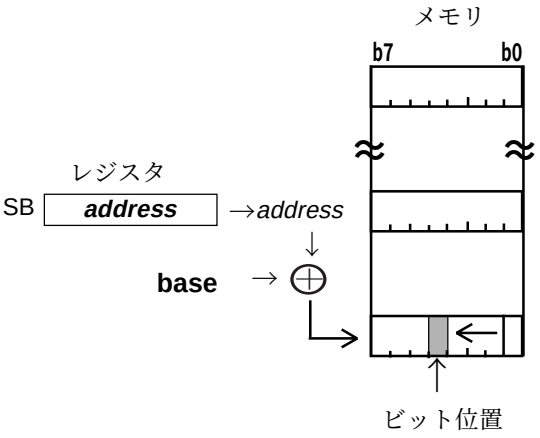
bit,base:19、bit,base:27で指定できるアドレスの範囲は 000000₁₆ 番地～000FFFF₁₆ 番地、000000₁₆ 番地～0FFFFFF₁₆ 番地です。</td></tr>
<tr><td colspan="2">アドレスレジスタ間接</td><td rowspan="2">レジスタ
A0/A1 address →
 
</td></tr>
<tr><td>bit,[A0]
bit,[A1]</td><td>アドレスレジスタ（A0/A1）の内容で示したアドレスのビット 0 からビット数だけ離れたビットが演算の対象となります。

000000₁₆ 番地～ 0FFFFFF₁₆ 番地のビットが対象となります。

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

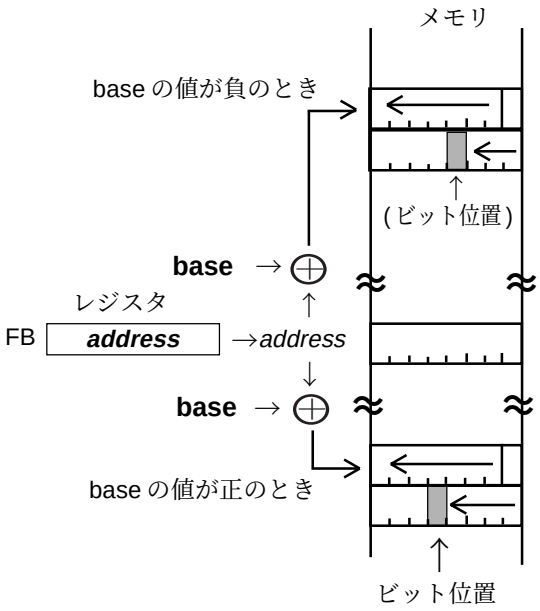
<table>
<tr><td colspan="2">アドレスレジスタ相対</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,base:11[A0]
bit,base:11[A1]
bit,base:19[A0]
bit,base:19[A1]
bit,base:27[A0]
bit,base:27[A1]</td><td>アドレスレジスタ(A0/A1)の内容で示したアドレスにbaseで示した値を符号なしで加算したアドレスのビット0から、bitで示したビット数だけ離れたビットが演算の対象になります。

ただし、対象となるビットのアドレスが0FFFFFF16を超える場合、25ビット以上は無視され、0000000016番地側に戻ります。

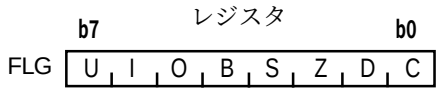
bit,base:11、bit,base:19、bit,base:27で指定できるアドレスの範囲はアドレスレジスタの値からそれぞれ256バイト、65536バイト、16777216バイトです。</td></tr>
<tr><td colspan="2">SB相対</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,base:11[SB]
bit,base:19[SB]</td><td>スタティックベースレジスタ(SB)の内容で示したアドレスにbaseで示した値を符号なしで加算したアドレスのビット0から、bitで示したビット数だけ離れたビットが演算の対象となります。

ただし、対象となるビットのアドレスが0FFFFFF16を超える場合、25ビット以上は無視され、0000000016番地側に戻ります。

bit,base:11、bit,base:19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256バイト、65536バイトです。</td></tr>
</table>

<table>
<tr><td colspan="2">FB 相対</td><td rowspan="2"> 
</td></tr>
<tr><td>bit,base:11[FB]
bit,base:19[FB]</td><td>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。

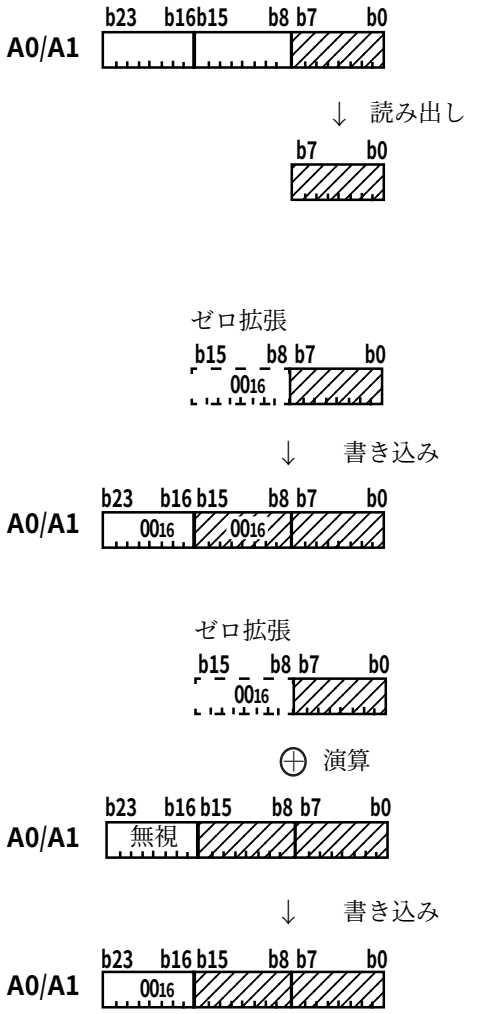
ただし、対象となるビットのアドレスが 000000₁₆ ～ 0FFFFFF₁₆ を超える場合、25 ビット以上は無視され、000000₁₆ 番地側、または 0FFFFFF₁₆ 番地側に戻ります。

bit,base:11、 bit,base:19 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256バイト、65536バイトです。</td></tr>
<tr><td colspan="2">FLG 直接</td><td rowspan="2"> 
</td></tr>
<tr><td>U
I
O
B
S
Z
D
C</td><td>指定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。</td></tr>
</table>

2.7　24 ビットレジスタでの読み出し・書き込みの動作説明

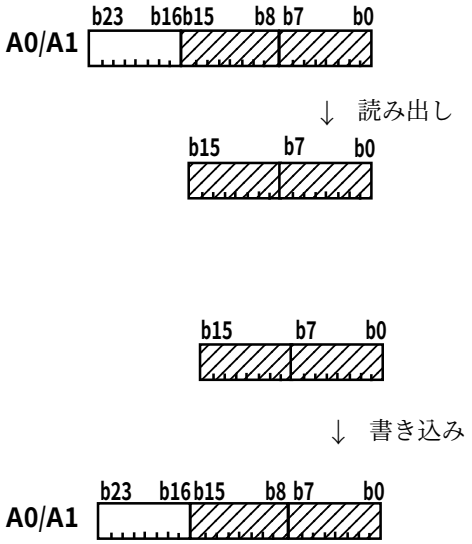
24 ビットレジスタ(A0、A1)を src または dest にしたときの動作を、サイズ指定子(.size/.B .W .L)ごとに以下に説明します。

サイズ指定子(.size)に(.B)を指定した場合	
・読み出し 下位 8 ビットを読み出します。フラグは 8 ビット演算の結果で変化します。 ・書き込み [転送命令] src を 16 ビットにゼロ拡張し、下位 16 ビットに格納します。そのとき、上位 8 ビットは "0" になります。フラグは16ビット転送データにより変化します。 [演算命令] src を 16 ビットにゼロ拡張し、16 ビットで演算します。そのとき、上位 8 ビットは "0" になります。フラグは16ビット演算の結果により変化します。	

サイズ指定子(.size)に(.W)を指定した場合

・読み出し
　下位16ビットを読み出します。フラグは16ビット演算の結果により変化します。

・書き込み
　下位16ビットに書き込みます。そのとき、上位8ビットは"0"になります。フラグは16ビット演算の結果により変化します。



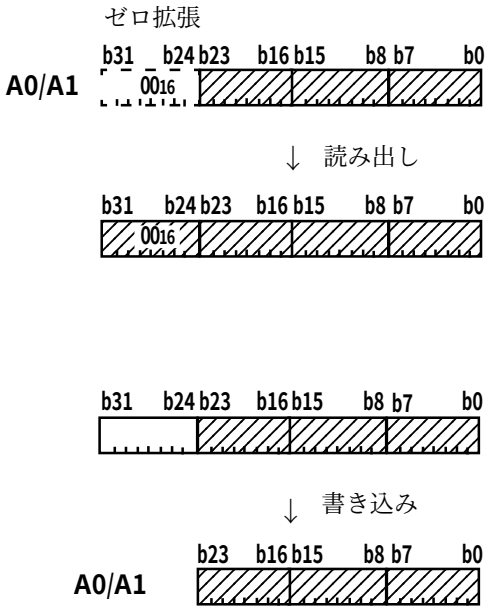
サイズ指定子(.size)に(.L)を指定した場合

・読み出し
　ゼロ拡張を行い、32ビットを読み出します。フラグは32ビット演算の結果により変化します。

・書き込み
　上位8ビットを無視し、下位24ビットを書き込みます。フラグは32ビット演算の結果により変化します。(24ビットレジスタの値ではありません。)

　　例：MOV.L　#80000000h, A0
　　実行後のフラグ変化
　　Sフラグ＝1(MSBはb31になります。)
　　Zフラグ＝0(32ビットすべて0で1になります。)

上記命令実行後A0の値は00000000_{16}となりますがフラグ変化は32ビットデータで行うためSフラグは1、Zフラグは0となります。



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、記述例について命令ごとに説明しています。

本章の見方について以下に実例をあげて示します。

3
機能
3.2　機能

① OR　　論理和　　OR

② 【命令コード／サイクル数】　　OR

③ 【構文】　　Page=260

OR.size (:format)　src,dest

G , S（指定可能）
B , W

④ 【オペレーション】

dest ← src ∨ dest　　[dest] ← src ∨ [dest]
dest ← [src] ∨ dest　　[dest] ← [src] ∨ [dest]

⑤ 【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がアドレスレジスタ(A0、A1)のとき、srcをゼロ拡張し16ビットで演算し、上位8ビットは0になります。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ビットを演算の対象とします。
・サイズ指定子(.size)に(.W)を指定した場合、destがアドレスレジスタのとき、上位8ビットは0になります。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16ビットを演算の対象とします。

⑥ 【選択可能なsrc / dest】 *1　（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/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1　間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、#IMM以外で使用できます。
*2　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にアドレスレジスタを選択できません。

⑦ 【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。

⑧ 【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。
```

115

①ニーモニック

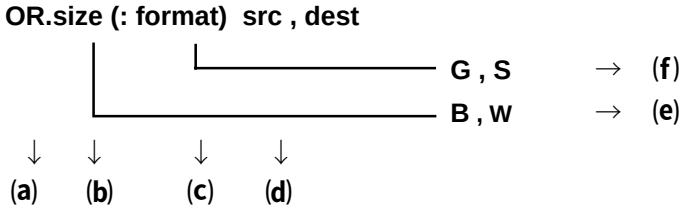
本ページで説明するニーモニックを示しています。

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　OR

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。

❶ **OR**　　*論理和*　　**OR**

❷ OR　　【命令コード／サイクル数】

❸ 【構文】　　Page=260

OR.size (:format)　src,dest

G , S（指定可能）

B , W

④ 【オペレーション】

dest ← src ∨ dest　　[dest] ← src ∨ [dest]

dest ← [src] ∨ dest　　[dest] ← [src] ∨ [dest]

⑤ 【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がアドレスレジスタ(A0、A1)のとき、srcをゼロ拡張し16ビットで演算し、上位8ビットは0になります。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ビットを演算の対象とします。

・サイズ指定子(.size)に(.W)を指定した場合、destがアドレスレジスタのとき、上位8ビットは0になります。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16ビットを演算の対象とします。

⑥ 【選択可能なsrc / dest】*1　（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/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1　間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、#IMM以外で使用できます。

*2　サイズ指定子(.size)に(.B)を指定する場合、srcとdestに同時にアドレスレジスタを選択できません。

⑦ 【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。

⑧ 【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。
```

115

④オペレーション

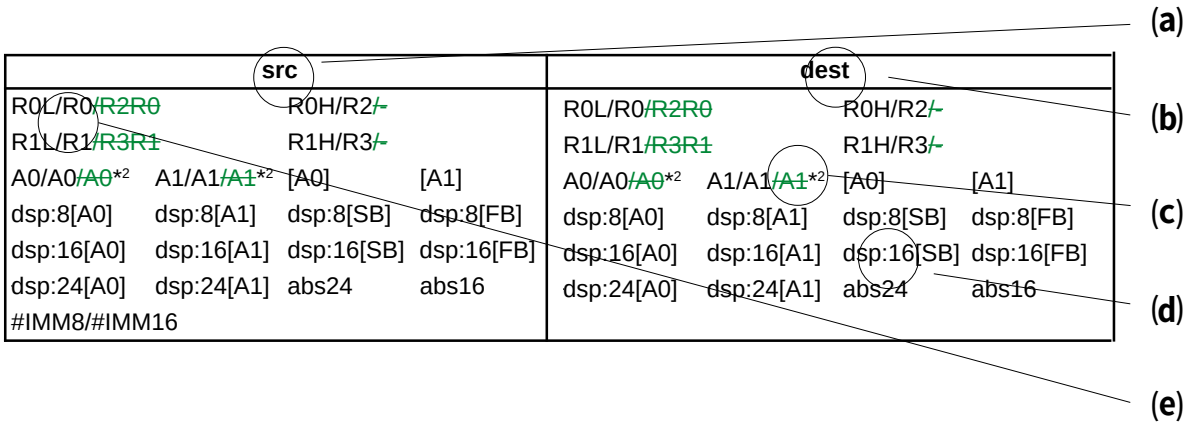
命令のオペレーションを記号で説明しています。

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/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

(a) src(source)として選択できる項目

(b) dest(destination)として選択できる項目

(c) 選択できないアドレッシング

(d) 選択できるアドレッシング

(e) スラッシュの左側（R0L）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
スラッシュの真ん中（R0）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
スラッシュの右側（R2R0）は取り扱うデータサイズがロングワード（32 ビット）の場合のアドレッシング

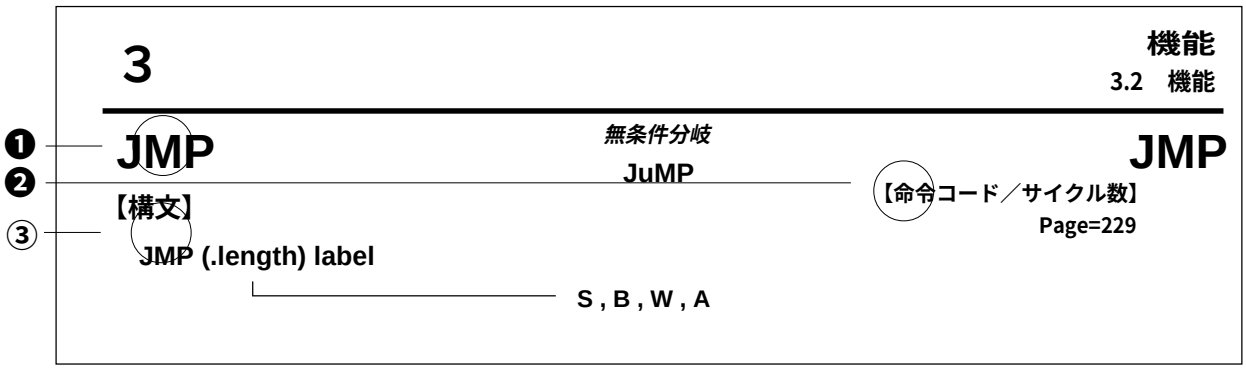
⑦フラグ変化

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。

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。

JMP (.length) label
S , B , W , A →　(d)
↓　↓　↓
(a)　(b)　(c)

(a)　ニーモニック　JMP

ニーモニックを記述します。

(b)　分岐距離指定子　.length

分岐する距離を記述します。JMP、JSR 命令については(.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　24 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=174

【オペレーション】

dest ← |dest|

[dest] ← |[dest]|

【機能】

・dest の絶対値をとり、dest に格納します。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

条件

O ： 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]
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=174

【オペレーション】

dest ← dest + src + C

【機能】

・dest とsrc とCフラグを加算し、dest に格納します。
・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(A0 、A1) のとき、src をゼロ拡張し16 ビットで演算します。そのとき上位8 ビットは0 になります。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 ビットを演算の対象とします。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/-		R0L/R0/~~R2R0~~		R0H/R2/-	
R1L/R1/~~R3R1~~		R1H/R3/-		R1L/R1/~~R3R1~~		R1H/R3/-	
A0/A0/~~A0~~*1	A1/A1/~~A1~~*1	[A0]	[A1]	A0/A0/~~A0~~*1	A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*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。
ADC.W   R1,[A1]
```

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=176

【オペレーション】

dest ← dest + C
[dest] ← [dest] + C

【機能】

・dest とCフラグを加算し、dest に格納します。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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】

```
ADCF.B    R0L
ADCF.W    Ram:16[A0]
```

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 , L**

【命令コード／サイクル数】

Page=176

【オペレーション】

dest ← dest + src　　[dest] ← [dest] + src

dest ← dest + [src]　　[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がアドレスレジスタ(A0、A1)のとき、srcをゼロ拡張し16ビットで演算します。そのとき、上位8ビットは0になります。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ビットを演算の対象とします。
- サイズ指定子(.size)に(.W)を指定した場合、destがアドレスレジスタのとき、上位8ビットは0になります。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16ビットを演算の対象とします。
- サイズ指定子(.size)に(.L)を指定した場合、destがアドレスレジスタのとき、destをゼロ拡張し32ビットで演算を行い、演算結果の下位24ビットをdestに格納します。また、srcがアドレスレジスタのときsrcをゼロ拡張し、32ビットで演算を行います。32ビットの演算結果でフラグも変化します。
- サイズ指定子(.size)に(.L)を指定した場合、destがSPのとき、destを32ビットにゼロ拡張し、srcを32ビットに符号拡張し演算を行います。演算結果の下位24ビットをdestに格納します。32ビットの演算結果でフラグも変化します。

【選択可能なsrc / dest】 *1　（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/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*3	A1/A1/A1*3	[A0]	[A1]	A0/A0/A0*3	A1/A1/A1*3	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM				~~SP/SP/~~SP*2			

*1　間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

*2　演算の対象はUフラグで示すスタックポインタです。

*3　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、+2147483647(.L)　または−2147483648(.L)　、+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、+4294967295(.L)　、+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]],abs16
```

【フォーマット別src / dest】

G フォーマット*1

src				dest			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16/#IMM32				~~SP/SP/~~SP*3			

*1 間接アドレッシング[src]、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2 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。

*3 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#IMM16 のみ選択できます。サイズ指定子(.size) は(.L) のみ指定できます。この場合、間接アドレッシングは使用できません。

Q フォーマット*4

src				dest			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0/R2R0		R0H/R2/-	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L/R1/R3R1		R1H/R3/-	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	A0/A0/A0	A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM3*6/#IMM4*7				~~SP/SP/~~SP*5			

*4 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

*5 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#IMM3のみ選択できます。

*6 dest がSPのとき#IMM3が選択できます。取りうる範囲は+1 ≦#IMM3≦+8 です。

*7 dest がSP以外のとき#IMM4が選択できます。取りうる範囲は−8 ≦#IMM4≦+7 です。

S フォーマット*8

src				dest			
~~R0L/R0~~	~~dsp:8[SB]~~	~~dsp:8[FB]~~	~~abs16~~	R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16
#IMM8/#IMM16*9							
#1*10	#2*10			A0*10	A1*10		
#IMM8*10				SP*10			

*8 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

*9 サイズ指定子(.size) には(.B) と(.W) のみ指定できます。

*10 サイズ指定子(.size) には(.L) のみ指定できます。この場合、間接アドレッシングは使用できません。

ADDX

符号拡張キャリーなし加算
ADDition eXtend sign

ADDX

【構文】

ADDX src,dest

【命令コード／サイクル数】

Page=183

【オペレーション】

dest ← dest + EXTS(src)　　[dest] ← [dest] + EXTS(src)
dest ← dest + EXTS([src])　　[dest] ← [dest] + EXTS([src])

【機能】

・8 ビットのsrc を32 ビット符号拡張し、32 ビットのdest を加算し、dest に格納します。
・destがアドレスレジスタ(A0、A1)のとき、destをゼロ拡張し32ビットで演算を行い、演算結果の下位24ビットをdestに格納します。32ビットの演算結果でフラグも変化します。srcがアドレスレジスタのとき、下位8ビットを演算対象とします。

【選択可能なsrc / dest】*1

src				dest			
R0L/~~R0/R2R0~~		R0H/~~R2/-~~		~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
R1L/~~R1/R3R1~~		R1H/~~R3/-~~		~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
A0/~~A0/A0~~	A1/~~A1/A1~~	[A0]	[A1]	~~A0/A0/~~A0	~~A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8							

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

【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、+2147483647 または−2147483648 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、+4294967295 を超えると“１”、それ以外のとき“０”になります。

【記述例】

```
ADDX    R0L,A0
ADDX    RAM:8[SB],R2R0
ADDX    [A0],A1
```

ADJNZ

加算＆条件分岐

ADdition then Jump on Not Zero

ADJNZ

【構文】

ADJNZ.size src,dest,label

size: B , W

【命令コード／サイクル数】

Page=185

【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。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なsrc / dest / label】

src	dest				label
#IMM4*1	R0L/R0/~~R2R0~~		R0H/R2/~~-~~		**PC*2 -126≦label≦PC*2+129**
	R1L/R1/~~R3R1~~		R1H/R3/~~-~~		
	~~A0~~/A0/~~A0~~	~~A1~~/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]	dsp:16[FB]	
	dsp:24[A0]	dsp:24[A1]	abs24	abs16	

*1 取りうる範囲は−8 ≦#IMM4≦+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
```

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=186

【オペレーション】

dest ← src ∧ dest　　[dest] ← src ∧ [dest]

dest ← [src] ∧ dest　　[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 がアドレスレジスタのとき、src をゼロ拡張し16 ビットで演算し、上位8 ビットは0 になります。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 ビットを演算の対象とします。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】 *1　（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/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2　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
AND.W:G     [A0],[[A1]]
```

【フォーマット別src / dest】

G フォーマット*1

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2 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*3

src				dest			
~~R0L/R0~~	~~dsp:8[SB]~~	~~dsp:8[FB]~~	~~abs16~~	R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16
#IMM8/#IMM16							

*3 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

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=188

【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。

・src がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なsrc】

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BAND    5,[A0]
```

BCLR

ビットクリア
Bit CLeaR

BCLR

【構文】

BCLR dest

【命令コード／サイクル数】

Page=188

【オペレーション】

dest ← 0

【機能】

・dest に“ 0 ”を格納します。
・dest がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なdest】

dest			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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
BCLR    16,Ram:19[SB]
BCLR    5,[A0]
```

BITINDEX

ビットインデックス
BIT INDEX

BITINDEX

【構文】

BITINDEX.size src

B , W

【命令コード／サイクル数】

Page=189

【オペレーション】

【機能】

・次のビット命令のアドレッシングをモディファイします。
・この命令の直後には、割り込み要求を受け付けません。
・src で指定したオペランドが次のビット命令のsrc またはdest のインデックス値となります。
・詳細は3.3　インデックス命令を参照してください。

【選択可能なsrc】

src			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~	A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs:24	abs:16

【フラグ変化】

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

【記述例】

```
BITINDEX　R0
BITINDEX　[A0]
```

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=190

【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。
・dest がアドレスレジスタのとき、下位8 ビットが指定できます。

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,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19
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:11[SB]
BMZ     C
```

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=192

【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。

・src がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なsrc】

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BNAND    5,[A0]
```

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=192

【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。
・src がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なsrc】

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BNOR    5,[A0]
```

BNOT

ビット反転
Bit NOT

BNOT

【構文】
BNOT dest

【命令コード／サイクル数】
Page=193

【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。
・dest がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なdest】

dest			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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
BNOT    16,Ram:19[SB]
BNOT    5,[A0]
```

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=193

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

【機能】
・src の反転をZフラグとCフラグに転送します。
・src がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なsrc】

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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
BNTST    16,Ram:19[SB]
BNTST    5,[A0]
```

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=194

【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。

・src がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なsrc】

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BNXOR    5,[A0]
```

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=194

【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。
・src がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なsrc】

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BOR     5,[A0]
```

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= 195

【オペレーション】

FFFFE7$_{16}$番地がFF$_{16}$以外の場合

SP ← SP − 2
M(SP) ← FLG
SP ← SP − 2
M(SP)*1 ← (PC + 1)$_H$
SP ← SP − 2
M(SP) ← (PC + 1)$_{ML}$
PC ← M(FFFFE4$_{16}$)

*1 上位8ビットは不定になります。

FFFFE7$_{16}$番地がFF$_{16}$の場合

SP ← SP − 2
M(SP) ← FLG
SP ← SP − 2
M(SP)*2 ← (PC + 1)$_H$
SP ← SP − 2
M(SP) ← (PC + 1)$_{ML}$
PC ← M(IntBase)

*2 上位8ビットは不定になります。

【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
```

BRK2

デバッグ割り込み2

BReaK2

BRK2

【構文】

BRK2

【命令コード／サイクル数】

Page=195

【オペレーション】

SP ← SP − 2
M(SP) ← FLG
SP ← SP − 2
M(SP)*1 ← (PC + 1)H
SP ← SP − 2
M(SP) ← (PC + 1)ML
PC ← M(0020 16)

*1 上位8ビットは不定になります。

【機能】

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

【フラグ変化】*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 BRK2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】

```
BRK2
```

BSET

ビットセット
Bit SET

BSET

【構文】

BSET **dest**

【命令コード／サイクル数】

Page=196

【オペレーション】

dest ← 1

【機能】

・dest に“1”を格納します。
・dest がアドレスレジスタのとき、下位8 ビットが指定できます。

【選択可能なdest】

dest			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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
BSET    16,Ram:19[SB]
BSET    5,[A0]
```

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=196

【オペレーション】

Z ← $\overline{\text{src}}$
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。
・src がアドレスレジスタのとき、下位8 ビットが指定できます。

【選択可能なsrc】

G フォーマット

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

S フォーマット

src
bit,base:19

【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
BTST    16,Ram:19[SB]
BTST    5,[A0]
```

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=197

【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。
・dest がアドレスレジスタのとき、下位8 ビットが指定できます。
・本命令はSFR領域に対して使用しないでください。

【選択可能なdest】

dest			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BTSTC   5,[A0]
```

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=198

【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。
・dest がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。
・本命令はSFR領域に対して使用しないでください。

【選択可能なdest】

dest			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BTSTS    5,[A0]
```

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=198

【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。
・src がアドレスレジスタ(A0 、A1) のとき、下位8 ビットが指定できます。

【選択可能なsrc】

src			
bit,R0L	bit,R0H	bit,R1L	bit,R1H
bit,A0	bit,A1	bit,[A0]	bit,[A1]
bit,base:11[A0]	bit,base:11[A1]	bit,base:11[SB]	bit,base:11[FB]
bit,base:19[A0]	bit,base:19[A1]	bit,base:19[SB]	bit,base:19[FB]
bit,base:27[A0]	bit,base:27[A1]	bit,base:27	bit,base:19

【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:19[SB]
BXOR    5,[A0]
```

CLIP

クリップ命令
CLIP

CLIP

【構文】

CLIP.size src1, src2, dest

size: B , W

【命令コード／サイクル数】

Page= 199

【オペレーション】

if src1>dest
then dest ← src1
if src2<dest
then dest ← src2

【機能】

・src1 とdest を符号付きで比較して、src1 がdest より大きければ、dest にsrc1 の内容を格納し、次にsrc2 とdest を符号付きで比較して、src2 がdest より小さければ、dest にsrc2 の内容を格納します。src1 ≦dest ≦ src2 ならば、dest は変化しません。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタでdest に書き込みが生じたとき、上位8 ビットは0 になります。
・src1 とsrc2 には、src1＜src2 となるよう設定してください。

【選択可能なsrc】

src1, src2				dest			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0~~/R2R0~~		R0H/R2/-	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L/R1~~/R3R1~~		R1H/R3/-	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/~~A0~~/A0~~	~~A1/~~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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

【フラグ変化】

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

条件

【記述例】

```
CLIP.W    #5,#10,R1
CLIP.W    #-5,#5,[A0]
```

CMP

比較
CoMPare

CMP

【構文】

CMP.size (:format) src,dest

G , Q , S（指定可能）
B , W , L

【命令コード／サイクル数】

Page= 200

【オペレーション】

dest ← src　　[dest] ← src
dest ← [src]　　[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 がアドレスレジスタ(A0 、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 ビットを演算の対象とします。
・サイズ指定子(.size)に(.L)を指定した場合、srcまたはdestがアドレスレジスタのとき、アドレスレジスタをゼロ拡張し32ビットで演算を行います。32ビットの演算結果でフラグも変化します。

【選択可能なsrc / dest】*1　　（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/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM4/#IMM8/#IMM16/#IMM32							

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。
*2　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、+2147483647(.L) または2147483648(.L) 、+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*1

src				dest			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
~~#IMM4~~/#IMM8/#IMM16/#IMM32							

*1 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2 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*3*4

src				dest			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0~~/R2R0~~		R0H/R2~~/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L/R1~~/R3R1~~		R1H/R3~~/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	A0/A0~~/A0~~	A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM4*5~~/#IMM8/#IMM16/#IMM32~~							

*3 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*4 サイズ指定子(.size) には(.B) または(.W) のみ指定できます。

*5 取りうる範囲は−8 ≦#IMM4≦+7 です。

S フォーマット*6*7

src				dest			
~~R0L/R0~~	~~dsp:8[SB]~~	~~dsp:8[FB]~~	~~abs16~~	R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16
#IMM8/#IMM16							
~~R0L/R0~~	dsp:8[SB]	dsp:8[FB]	abs16	R0L/R0	~~dsp:8[SB]~~	~~dsp:8[FB]~~	~~abs16~~
~~#IMM~~							

*6 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*7 サイズ指定子(.size) には(.B) 、(.W) のみ指定できます。

CMPX

符号拡張比較
CoMPare eXtend sign

CMPX

【構文】
CMP src,dest

【命令コード／サイクル数】
Page= 206

【オペレーション】
dest/[dest] ←EXTS(src)

【機能】
- 32 ビットのdest から32 ビット符号拡張した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。
- dest がアドレスレジスタ(A0 、A1) のときゼロ拡張し、32 ビットで演算を行い、その結果でフラグも変化します。

【選択可能なsrc / dest】*1

src				dest			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/A0/~~A0	~~A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8							

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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、+2147483647(.L) または-2147483648(.L) を超えると“１”、それ以外のとき“０”になります。
- S ： 演算の結果、MSBが“１”になると“１”、それ以外のとき“０”になります。
- Z ： 演算の結果が0 のとき“１”、それ以外のとき“０”になります。
- C ： 符号なし演算の結果、0 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】
```
CMPX    #10,R2R0
CMPX    #5,A0
```

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= 206

【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。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0~~/A0~~	~~A1/~~A1~~/A1~~	[A0]	[A1]	~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

【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
DADC.W    [A0],R2
```

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= 208

【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。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0	A1/A1/A1	[A0]	[A1]	A0/A0/A0	A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

【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
DADD.W    [A0],[A1]
```

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= 210

【オペレーション】

dest ← dest − 1
[dest] ← [dest] − 1

【機能】

・dest から１を減算し、dest に格納します。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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
DEC.W   R0
```

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= 210

【オペレーション】

サイズ指定子(.size) が(.W) のとき

R0(商)、R2(剰余)←R2R0 ÷src　または　R2R0 ÷[src]

サイズ指定子(.size) が(.B) のとき

R0L(商)、R0H(剰余)←R0 ÷src　または　R0 ÷[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 がアドレスレジスタ(A0 、A1) のとき、アドレスレジスタの下位8 ビットを演算の対象とします。

・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フラグは“ 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、src がアドレスレジスタのとき、アドレスレジスタの下位16 ビットを演算の対象とします。

・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フラグは“ 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*2			
R0L/R0/~~R2R0~~		R0H/R2/-	
R1L/R1/~~R3R1~~		R1H/R3/-	
A0/A0/~~A0~~	A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16			

*2　間接アドレッシング[src]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

条件

O ： 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
DIV.W   [[A1]]
```

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= 211

【オペレーション】

サイズ指定子(.size) が(.W) のとき

R0(商)、R2(剰余)←R2R0 ÷src　または　R2R0 ÷[src]

サイズ指定子(.size) が(.B) のとき

R0L(商)、R0H(剰余)←R0 ÷src　または　R0 ÷[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 がアドレスレジスタ(A0 、A1) のとき、アドレスレジスタの下位8 ビットを演算の対象とします。

・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フラグは“ 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、src がアドレスレジスタのとき、アドレスレジスタの下位16 ビットを演算の対象とします。

・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フラグは“ 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*2			
R0L/R0/~~R2R0~~		R0H/R2/-	
R1L/R1/~~R3R1~~		R1H/R3/-	
A0/A0/~~A0~~	A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16			

*2　間接アドレッシング[src]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

条件

O ： 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
DIVU.W    [[A0]]
```

DIVX

符号付き除算
DIVide eXtension

DIVX

【構文】

DIVX.size src

size: B , W

【命令コード／サイクル数】

Page= 212

【オペレーション】

サイズ指定子(.size) が(.W) のとき

R0(商)、R2(剰余)←R2R0 ÷src　または　R2R0 ÷[src]

サイズ指定子(.size) が(.B) のとき

R0L(商)、R0H(剰余)←R0 ÷src　または　R0 ÷[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 がアドレスレジスタ(A0 、A1) のとき、アドレスレジスタの下位8 ビットを演算の対象とします。

・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フラグは“ 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、src がアドレスレジスタのとき、アドレスレジスタの下位16 ビットを演算の対象とします。

・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フラグは“ 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*2			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~	A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16			

*2　間接アドレッシング[src]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

条件

O ： 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
```

DSBB

ボロー付き10進減算

Decimal SuBtract with Borrow

DSBB

【構文】

【命令コード／サイクル数】

Page= 213

```
DSBB.size   src,dest
        └──────────── B , W
```

【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。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0~~/R2R0~~		R0H/R2~~/-~~		R0L/R0~~/R2R0~~		R0H/R2~~/-~~	
R1L/R1~~/R3R1~~		R1H/R3~~/-~~		R1L/R1~~/R3R1~~		R1H/R3~~/-~~	
~~A0~~/A0~~/A0~~	~~A1~~/A1~~/A1~~	[A0]	[A1]	~~A0~~/A0~~/A0~~	~~A1~~/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

【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
DSBB.W    [A0],[A1]
```

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= 215

【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。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0~~/A0/~~A0~~	~~A1~~/A1/~~A1~~	[A0]	[A1]	~~A0~~/A0/~~A0~~	~~A1~~/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

【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
DSUB.W   [A0],[A1]
```

ENTER

スタックフレームの構築
ENTER function

ENTER

【構文】
ENTER　　src

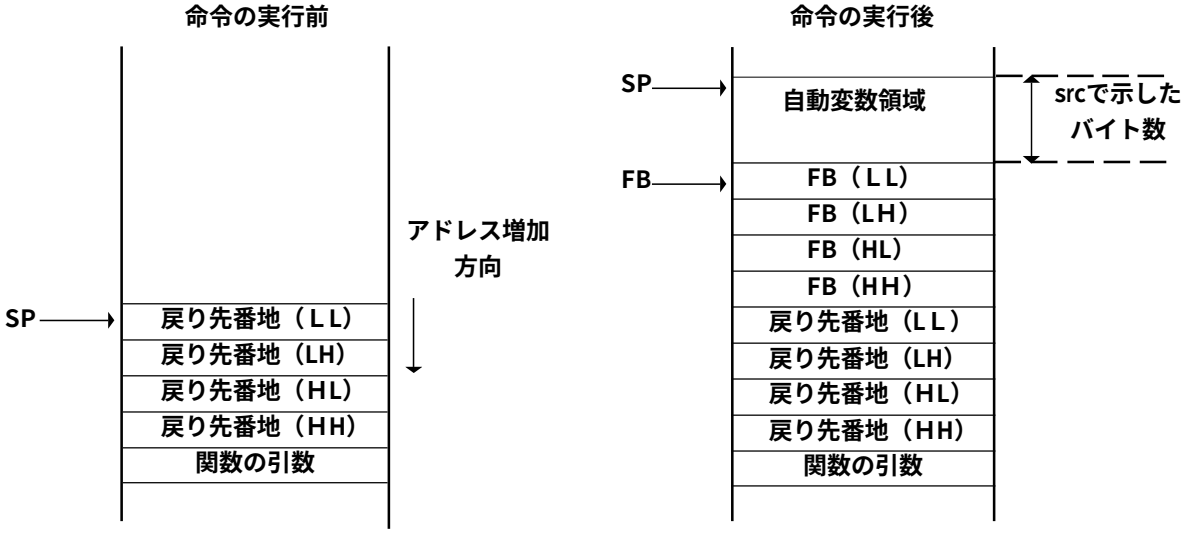
【命令コード／サイクル数】
Page= 217

【オペレーション】

SP	←	SP － 2	
M(SP)*1	←	FBH	
SP	←	SP － 2	
M(SP)	←	FBML	
FB	←	SP	
SP	←	SP － src	*1　上位8ビットは不定になります。

【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。src は偶数を設定してください。(奇数も設定できますが、偶数を設定した方が実行効率が良くなります。)
・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    #4
```

EXITD

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

EXITD

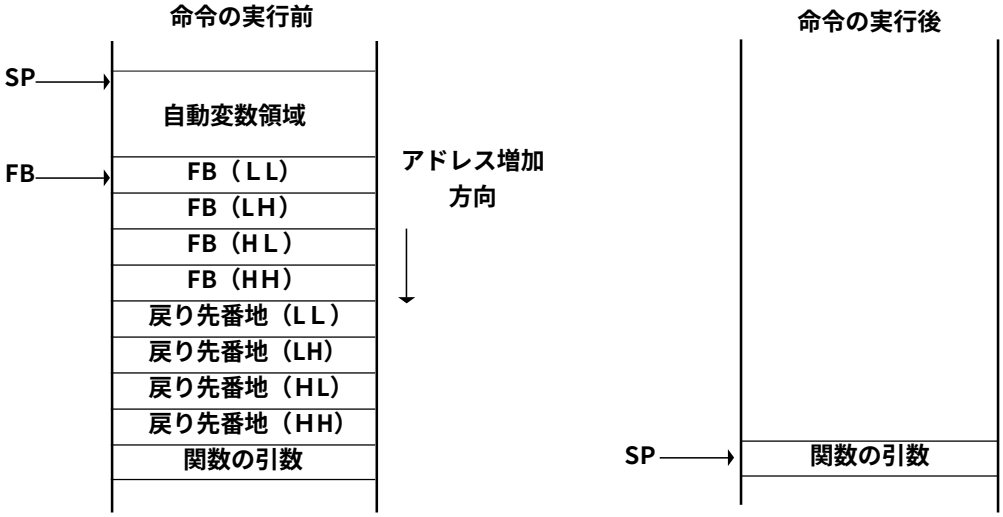
【構文】
EXITD

【命令コード／サイクル数】
Page= 217

【オペレーション】

SP	←	FB	
FBL	←	M(SP)	
SP	←	SP + 2	
FBH	←	M(SP)	
SP	←	SP + 2	
PCML	←	M(SP)	
SP	←	SP + 2	
PCH	←	M(SP)*1	
SP	←	SP + 2	*1 下位8ビットが保存されます。

【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

EXTS

符号拡張
EXTend Sign

EXTS

【構文】

```
EXTS.size   dest
            └──────────── B , W
EXTS.size   src,dest
            └──────────── B
```

【命令コード／サイクル数】

Page= 218

【オペレーション】

dest ← EXTS(dest)
dest ← EXTS(src)

【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、srcまたはdestを16ビットに符号拡張します。destがアドレスレジスタ(A0、A1)のとき、上位8ビットは0になります。
・サイズ指定子(.size)に(.W)を指定した場合、destを32ビットに符号拡張します。destにR0を選択したとき上位バイトにはR2を、R1を選択したとき上位バイトにはR3を使用します。destがアドレスレジスタのとき、演算結果の下位24ビットをdestに格納します。

【選択可能なsrc / dest】

dest*1			
R0L/R0~~/R2R0~~		~~R0H/R2/-~~	
R1L/R1~~/R3R1~~		~~R1H/R3/-~~	
~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 サイズ指定子(.size) には(.B) または(.W) のみ指定できます。

src*2				dest*2			
R0L~~/R0/R2R0~~		R0H~~/R2/-~~		~~R0L/~~R0~~/R2R0~~		~~R0H/~~R2~~/-~~	
R1L~~/R1/R3R1~~		R1H~~/R3/-~~		~~R1L/~~R1~~/R3R1~~		~~R1H/~~R3~~/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	[A0]	[A1]	~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16

*2 サイズ指定子(.size) には(.B) のみ指定できます。

【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】

```
EXTS.B   R0L
EXTS.W   R0
EXTS.W   [A0]
```

EXTZ

ゼロ拡張
EXTend Zero

EXTZ

【構文】
EXTZ src,dest

【命令コード／サイクル数】
Page= 220

【オペレーション】
dest ← EXTZ(src)

【機能】
・src を16 ビットにゼロ拡張し、dest に格納します。dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なsrc / dest】

src				dest			
R0L/~~R0/R2R0~~		R0H/~~R2/-~~		~~R0L/~~R0/~~R2R0~~		~~R0H/~~R2/~~-~~	
R1L/~~R1/R3R1~~		R1H/~~R3/-~~		~~R1L/~~R1/~~R3R1~~		~~R1H/~~R3/~~-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	[A0]	[A1]	~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
~~#IMM~~							

【フラグ変化】

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

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

【記述例】

```
EXTZ    R0L,R2
EXTZ    [A1],[A0]
```

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= 221

【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
```

FREIT

高速割り込みからの復帰

Fast REturn from InTerrupt

FREIT

【構文】

FREIT

【命令コード／サイクル数】

Page= 221

【オペレーション】

FLG ← SVF
PC ← SVP

【機能】

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

【フラグ変化】

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

*1 SVFの内容になります。

【記述例】

FREIT

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= 222

【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
```

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= 223

【オペレーション】

dest ← dest + 1

[dest] ← [dest] + 1

【機能】

・dest に1を加算し、dest に格納します。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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
INC.B   [[A1]]
```

INDEX*Type*　インデックス　INDEX Type　INDEX*Type*

【構文】

【命令コード／サイクル数】

Page= 223

INDEX*Type*.size　src

B , W

【オペレーション】

【機能】

・次の命令のアドレッシングをモディファイします。
・この命令の直後には割り込み要求を受け付けません。
・配列をアクセスするのに用います。
・詳細は3.3　インデックス命令を参照してください。
・*Type*には次の種類があります。

Type	機　　能
B BD BS	次の命令のアドレッシングをバイト単位でモディファイします。
W WD WS	次の命令のアドレッシングをワード単位でモディファイします。
L LD LS	次の命令のアドレッシングをロングワード単位でモディファイします。

【選択可能なsrc】

src			
R0L/R0/R2R0		R0H/R2	
R1L/R1/R3R1		R1H/R3	
A0/A0/A0	A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

【フラグ変化】

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

【記述例】

```
INDEXB.W    R0
INDEXLS.B   [A0]
```

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= 228

【オペレーション】

SP ← SP − 2
M(SP) ← FLG
SP ← SP − 2
M(SP)*1 ← (PC + 2) H
SP ← SP − 2
M(SP) ← (PC + 2) ML
PC ← M(IntBase + src × 4)

*1 上位8ビットは不定になります。

【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。
・src の割り込み番号は、0 ≦src ≦63 です。

【選択可能なsrc】

src
#IMM6*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 ≦#IMM6≦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
```

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= 228

【オペレーション】

SP ← SP − 2
M(SP) ← FLG
SP ← SP − 2
M(SP)*1 ← (PC + 1)H
SP ← SP − 2
M(SP) ← (PC + 1)ML
PC ← M(FFFFE016)

*1 上位8ビットは不定になります。

【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
```

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=229

【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 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
```

JMP

無条件分岐

JuMP

JMP

【構文】

JMP(.length) label

S , B , W , A

【命令コード／サイクル数】

Page=229

【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	abs24

*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

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=231

【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~~/R2R0~~		~~R0H/~~R2~~/-~~	
~~R1L/~~R1~~/R3R1~~		~~R1H/~~R3~~/-~~	
~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

分岐距離指定子(.length)　に(.A) を指定した場合

src			
~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
~~A0/A0/~~A0	~~A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

【フラグ変化】

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

【記述例】

```
JMPI.A      A1
JMPI.W      R0
```

JMPS

スペシャルページ分岐
JuMP Special page

JMPS

【構文】
JMPS src

【命令コード／サイクル数】
Page=232

【オペレーション】

PCH ← FF16
PCML ← M(FFFE16 − src × 2)

【機能】

・スペシャルページベクタテーブルの各テーブルに設定している番地にFF0000 16を加算した番地へ分岐します。分岐できる領域は、FF0000 16番地からFFFFFF16番地です。
・スペシャルページベクタテーブルはFFFE00 16番地からFFFFDB16番地の領域に配置されています。
・src は、スペシャルページ番号です。スペシャルページ番号は、FFFE00 16番地が255 でFFFFDA16番地が18 です。

【選択可能なsrc】

src
#IMM8*1*2

*1 #IMMは、スペシャルページ番号です。
*2 取りうる範囲は18 ≦#IMM8≦255 です。

【フラグ変化】

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

【記述例】

```
JMPS    #20
```

JSR

サブルーチン呼び出し
Jump SubRoutine

JSR

【構文】

JSR(.length) label
W , A

【命令コード／サイクル数】
Page=233

【オペレーション】

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

*1 上位8ビットは0になります。
*2 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	abs24

*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
```

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=234

【オペレーション】

分岐距離指定子(.length) が(.W) のとき

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

分岐距離指定子(.length) が(.A) のとき

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

*1 上位8ビットは0になります。
*2 nは命令のバイト数です。

【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/~~R2R0~~		~~R0H~~/R2/~~-~~	
~~R1L~~/R1/~~R3R1~~		~~R1H~~/R3/~~-~~	
~~A0~~/A0/~~A0~~	~~A1~~/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

分岐距離指定子(.length) に(.A) を指定した場合

src			
~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
~~A0/A0/~~A0	~~A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

【フラグ変化】

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

【記述例】

```
JSRI.A    A1
JSRI.W    R0
```

JSRS

スペシャルページサブルーチン呼び出し
Jump SubRoutine Special page

JSRS

【構文】
JSRS src

【命令コード／サイクル数】
Page=235

【オペレーション】

SP ← SP − 2
M(SP) *1 ← (PC + 2) H
SP ← SP − 2
M(SP) ← (PC + 2) ML
PCH ← FF16
PCML ← M (FFFE16 − src × 2)

*1 上位8ビットは0になります。

【機能】

・スペシャルページベクタテーブルの各テーブルに設定している番地に、FF0000 16を加算した番地へサブルーチン分岐します。サブルーチン分岐できる領域は、FF0000 16番地からFFFFFF16番地です。
・スペシャルページベクタテーブルはFFFE00 16番地からFFFFDB16番地の領域に配置されています。
・src は、スペシャルページ番号です。 スペシャルページ番号は、FFFE00 16番地が255 でFFFFDA16番地が18 です。

【選択可能なsrc】

src
#IMM8*1*2

*1 #IMMは、スペシャルページ番号です。
*2 取りうる範囲は18 ≦#IMM8≦255 です。

【フラグ変化】

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

【記述例】

```
JSRS    #18
```

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=235

【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 にメモリを指定したとき、以下のバイト数分メモリが必要です。

2 バイト：DMD0*1、 DMD1*1、FLG、 DCT0 、 DCT1 、DRC0、DRC1、SVF

4 バイト*2：FB、SB、SP*3、ISP*3、INTB*3、VCT、SVP、DMA0、DMA1、DRA0、DRA1、DSA0 、DSA1

*1 src の下位8 ビットが書き込まれます。

*2 src の下位24 ビットが書き込まれます。

*3 奇数も設定できますが、SP、ISP、INTBには偶数を設定してください。偶数を設定した方が実行効率が良くなります。

【選択可能なsrc / dest】

src				dest			
~~R0L~~/R0/~~R2R0~~		~~R0H~~/R2/~~-~~		DMD0	DMD1	DCT0	DCT1
~~R1L~~/R1/~~R3R1~~		~~R1H~~/R3/~~-~~		DRC0	DRC1	FLG	SVF
~~A0~~/A0/~~A0~~	~~A1~~/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]	dsp:16[FB]				
dsp:24[A0]	dsp:24[A1]	abs24	abs16				
#IMM16							
~~R0L/R0~~/R2R0		~~R0H/R2~~/-		FB	SB	SP*4	ISP
~~R1L/R1~~/R3R1		~~R1H/R3~~/-		INTB	VCT	SVP	
~~A0/A0~~/A0	~~A1/A1~~/A1	[A0]	[A1]	DMA0	DMA1	DRA0	DRA1
dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]	DSA0	DSA1		
dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]				
dsp:24[A0]	dsp:24[A1]	abs24	abs16				
#IMM24							

*4 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	*5	*5	*5	*5	*5	*5	*5	*5

*5 dest がFLGのときのみ変化します。

【記述例】

```
LDC     A0,FB
```

LDCTX

コンテキストの復帰
LoaD ConTeXt

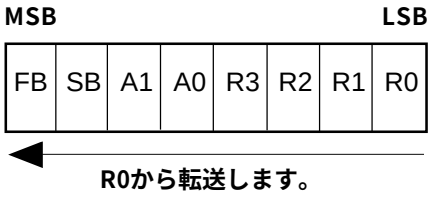
LDCTX

【構文】
LDCTX abs16,abs24

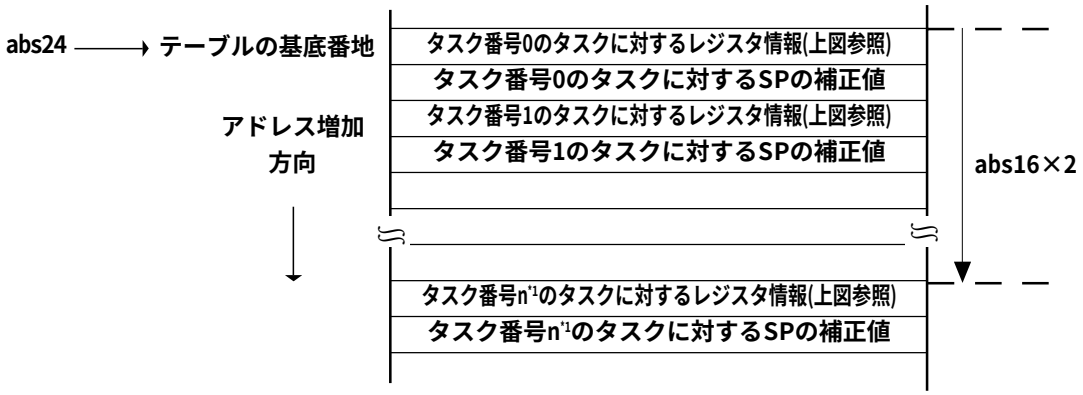
【命令コード／サイクル数】
Page=238

【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、abs24 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の合計を設定してください。このとき、R0、R1、R2、R3レジスタは2バイトと、A0、A1、SB、FBレジスタは4バイトと数えます。
・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。abs24 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
```

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=239

【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
#IMM3[*1]

*1　取りうる範囲は0 ≦#IMM3≦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
```

MAX

最大値選択
Max select

MAX

【構文】

MAX.size src,dest
　　└──── **B , W**

【命令コード／サイクル数】

Page= 239

【オペレーション】

if(src > dest)
thendest ← src

【機能】

・src とdest を符号付きで比較し、src がdest より大きければdest にsrc を転送し、src がdest より小さいか等しければ変化ありません。。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタでdest に書き込みが生じたとき、上位8 ビットは0 になります。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16 ビットを転送します。

【選択可能なsrc / dest】

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0~~/A0~~	~~A1/~~A1~~/A1~~	[A0]	[A1]	~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

【フラグ変化】

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

【記述例】

```
MAX.B    #0ABH,R0L
MAX.W    #–1,R2
```

MIN

最小値選択
Min select

MIN

【構文】

MIN.size src,dest
└──── **B , W**

【命令コード／サイクル数】

Page= 241

【オペレーション】

if(src < dest)
thendest ← src

【機能】

・src とdest を符号付きで比較し、src がdest より小さければdest にsrc を転送し、src がdest より大きいか等しければ変化ありません。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタでdest に書き込みが生じたとき、上位8 ビットは0 になります。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16 ビットを転送します。

【選択可能なsrc / dest】

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0~~/A0/~~A0~~	~~A1~~/A1/~~A1~~	[A0]	[A1]	~~A0~~/A0/~~A0~~	~~A1~~/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

【フラグ変化】

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

【記述例】

```
MIN.B    #0ABH,R0L
MIN.W    #–1,R2
```

MOV

転送

MOVe

MOV

【構文】

MOV.size (:format) src,dest

G , Q , Z , S （指定可能）

B , W , L

【命令コード／サイクル数】

Page= 243

【オペレーション】

dest ← src　　[dest] ← src

dest ← [src]　　[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 がアドレスレジスタ(A0 、A1) のとき、src をゼロ拡張し16ビットで転送します。そのとき上位8 ビットは0 になります。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。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを転送します。

・サイズ指定子(.size)に(.L)を指定した場合、destがアドレスレジスタのとき、srcの上位8ビットを無視し、srcの下位24ビットをdestに格納します。srcがアドレスレジスタのとき、srcをゼロ拡張し32ビットで転送します。32ビットの転送結果でフラグも変化します。

【選択可能なsrc / dest】 *1　（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/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM	dsp:8[SP]*3			dsp:8[SP]*3			

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2　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。

*3　src またはdest がdsp:8[SP] のとき、src 、dest ともに間接アドレッシング[src] 、[dest] は使用できません。

【フラグ変化】

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

条件

S ： 転送データのMSBが“１”のとき“１”、それ以外のとき“０”になります。

Z ： 転送データが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
MOV.W      [A1],[[A0]]
```

【フォーマット別src / dest】

G フォーマット*1

src				dest			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16/#IMM32		dsp:8[SP]*3*5		dsp:8[SP]*3*4*5			

*1 間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、dsp:8[SP]、#IMM以外で使用できます。
*2 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。
*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。
*4 srcが#IMM以外のときdsp8:[SP]は選択できます。また、dsp8:[SP]はサイズ指定子(.size)に(.B)または(.W)を指定する場合選択できます。
*5 srcまたはdestがdsp:8[SP]のとき、src、destともに間接アドレッシング[src]、[dest]は使用できません。

Q フォーマット*6*7

src				dest			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0~~/R2R0~~		R0H/R2~~/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L/R1~~/R3R1~~		R1H/R3~~/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	A0/A0~~/A0~~	A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM4*8							

*6 間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、dsp:8[SP]、#IMM以外で使用できます。
*7 サイズ指定子(.size)には(.B)および(.W)を指定できます。
*8 取りうる範囲は−8≦#IMM4≦+7です。

S フォーマット*9

src				dest			
R0L/R0*10*11	dsp:8[SB]*11	dsp:8[FB]*11	abs16*11	R0L/R0*10*11	R1L/R1*11*12	dsp:8[SB]*11	dsp:8[FB]*11
#IMM8/#IMM16*11				abs16*11	~~A0~~	~~A1~~	
~~R0L/R0~~	dsp:8[SB]*14	dsp:8[FB]*14	abs16*14	~~R0L~~	~~R0H~~	~~dsp:8[SB]~~	~~dsp:8[FB]~~
#IMM16*13/#IMM24*14				~~abs16~~	~~A0~~/A0*13/A0*14		~~A1~~/A1*13/A1*14

*9 間接アドレッシング[src]、[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、dsp:8[SP]、#IMM以外で使用できます。
*10 srcとdestに同じレジスタを選択できません。
*11 サイズ指定子(.size)には(.B)および(.W)を指定できます。
*12 srcが#IMM8/IMM16以外のとき、destにR1L/R1は指定できます。
*13 サイズ指定子(.size)には(.W)を指定できます。この場合destに間接アドレッシングモードは選択できません。
*14 サイズ指定子(.size)には(.L)を指定できます。この場合destに間接アドレッシングモードは選択できません。

Z フォーマット*15

src				dest			
~~R0L~~	~~R0H~~	~~dsp:8[SB]~~	~~dsp:8[FB]~~	R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16
~~abs16~~	#0			~~A0~~	~~A1~~		

*15 サイズ指定子(.size) には(.B) および(.W) を指定できます。

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= 252

【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/R2R0~~		~~R0H/R2/-~~		~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/A0/~~A0	~~A1/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]	dsp:16[FB]	~~dsp:16[A0]~~	~~dsp:16[A1]~~	~~dsp:16[SB]~~	~~dsp:16[FB]~~
dsp:24[A0]	dsp:24[A1]	abs24	abs16	~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~
~~#IMM~~							

【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
```

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= 253

【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(8ビット)の上位4ビットをdest(8ビット)の上位4ビットに転送します
HL	src(8ビット)の上位4ビットをdest(8ビット)の下位4ビットに転送します
LH	src(8ビット)の下位4ビットをdest(8ビット)の上位4ビットに転送します
LL	src(8ビット)の下位4ビットをdest(8ビット)の下位4ビットに転送します

【選択可能なsrc / dest】

src				dest			
R0L~~/R0/R2R0~~		R0H~~/R2/-~~		R0L~~/R0/R2R0~~		~~R0H/R2/-~~	
R1L~~/R1/R3R1~~		R1H~~/R3/-~~		~~R1L/R1/R3R1~~		~~R1H/R3/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	[A0]	[A1]	~~A0/A0/A0~~	~~A1/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]	dsp:16[FB]	~~dsp:16[A0]~~	~~dsp:16[A1]~~	~~dsp:16[SB]~~	~~dsp:16[FB]~~
dsp:24[A0]	dsp:24[A1]	abs24	abs16	~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~
~~#IMM~~							
R0L~~/R0/R2R0~~		~~R0H/R2/-~~		R0L~~/R0/R2R0~~		R0H~~/R2/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L~~/R1/R3R1~~		R1H~~/R3/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/A0/A0~~	~~A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
~~#IMM~~							

【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]
```

MOVX

符号拡張転送
MOVe eXtend sign

MOVX

【構文】

MOVX **src,dest**

【命令コード／サイクル数】

Page= 255

【オペレーション】

dest ← EXTS(src)　　[dest] ← EXTS(src)

【機能】

・8 ビットの即値を32 ビットに符号拡張しdest に転送します。
・dest がアドレスレジスタ(A0 、A1) のとき、下位24 ビットを転送します。32 ビットの転送でフラグも変化します。

【選択可能なsrc / dest】

src				**dest**[*1]			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/A0/~~A0	~~A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8*[2]							

*1　間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。
*2　取りうる範囲は－128≦#IMM8≦+127です。

【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”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】

```
MOVX    #10,A0
MOVX    #5,[[A1]]
```

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= 255

【オペレーション】

dest ← dest × src　　[dest] ← [dest] × src
dest ← dest × [src]　　[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 のどちらか一方にアドレスレジスタ(A0 、A1) を指定したときはアドレスレジスタの下位8 ビットの内容を演算します。dest にアドレスレジスタを指定したとき、上位8 ビットは0 になります。
・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 またはR1 を選択した場合、結果はそれぞれR2R0 またはR3R1 へ格納します。dest にアドレスレジスタを選択した場合、32 ビット演算結果の下位24 ビットを格納します。src にアドレスレジスタを選択した場合、下位16 ビットの内容で演算します。

【選択可能なsrc / dest】*1

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/-		R0L/R0/~~R2R0~~		~~R0H/R2/-~~	
R1L/R1/~~R3R1~~		R1H/R3/-		R1L/R1/~~R3R1~~		~~R1H/R3/-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。
*2　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
MUL.W    [A0],[[A1]]
```

MULEX

拡張符号付き乗算
MULtiple EXtend

MULEX

【構文】
MULEX **src**

【命令コード／サイクル数】
Page= 257

【オペレーション】
R1R2R0 ← R2R0 × src
R1R2R0 ← R2R0 × [src]

【機能】
・src(16 ビットデータ)とR2R0 を符号付きで乗算し、結果をR1R2R0 に格納します。

【選択可能なsrc】

src*1			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~	
~~R1L/R1/R3R1~~		~~R1H/~~R3~~/-~~	
~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[src]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

【記述例】

```
MULEX   A0
MULEX   R3
MULEX   Ram
MULEX   [[A0]]
```

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= 257

【オペレーション】

dest ← dest × src　　[dest] ← [dest] × src
dest ← dest × [src]　　[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のどちらか一方にアドレスレジスタ(A0、A1)を選択したときはアドレスレジスタの下位8ビットの内容を演算します。destにアドレスレジスタを選択したとき、上位8ビットは0になります。
・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 またはR1 を選択した場合、結果はそれぞれR2R0 またはR3R1 へ格納します。dest にアドレスレジスタを選択した場合、32 ビット演算結果の下位24 ビットを格納します。src にアドレスレジスタを選択した場合、下位16 ビットの内容で演算します。

【選択可能なsrc / dest】*1

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		~~R0H/R2/-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		~~R1H/R3/-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。
*2　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
MULU.W    [R1],[[A0]]
```

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= 259

【オペレーション】

dest ← 0 − dest

[dest] ← 0 − [dest]

【機能】

・dest の2の補数をとり、dest に格納します。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

条件

O ： 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
NEG.W   [[A0]]
```

NOP

ノーオペレーション
No OPeration

NOP

【構文】
NOP

【命令コード／サイクル数】
Page= 259

【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 dest

B , W

【命令コード／サイクル数】

Page= 260

【オペレーション】

dest ← $\overline{dest}$

[dest] ←[$\overline{dest}$]

【機能】

・dest を反転し、dest に格納します。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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
```

OR

論理和

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= 260

【オペレーション】

dest ← src ∨ dest　　[dest] ← src ∨ [dest]

dest ← [src] ∨ dest　　[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 がアドレスレジスタ(A0、A1)のとき、src をゼロ拡張し16ビットで演算し、上位8 ビットは0 になります。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 ビットを演算の対象とします。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】 *1　（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/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2　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
OR.W:G    [R1],[[A0]]
```

【フォーマット別src / dest】

G フォーマット*1

src				dest			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1 間接アドレッシング[src]、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2 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/R0	dsp:8[SB]	dsp:8[FB]	abs16	R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16
#IMM8/#IMM16							

*2 間接アドレッシング[src]、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

POP

レジスタ／メモリの復帰

POP

POP

【構文】

POP.size dest

B , W

【命令コード／サイクル数】

Page= 263

【オペレーション】

dest ← M(SP)
SP ← SP + 2 *1

[dest] ← M(SP)
SP ← SP + 2 *1

*1 サイズ指定子(.size)が(.B)でもSPは＋2されます。

【機能】

・dest をスタック領域から復帰します。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*2			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*2 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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
```

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= 263

【オペレーション】

destがDCT0、DCT1、DMD0、DMD1、DRC0、DRC1、SVF、FLGの場合

dest *1 ← M(SP)

SP ← SP + 2

*1 dest がDMD0、DMD1のとき、下位8 ビットが保存されます。

destがFB、SB、SP、ISP、INTBの場合

dest*2 ← M(SP)

SP*3 ← SP + 4

*2 下位3 バイトが保存されます。

*3 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4 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。

・退避するスタック領域は、Uフラグで示すスタックポインタになります。

【選択可能なdest】

dest			
FB	SB	SP*1	ISP
INTB			
DCT0	DCT1	DMD0	DMD1
DRC0	DRC1	SVF	FLG

*1 Uフラグで示すスタックポインタが対象となります。

【フラグ変化】

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

*2 dest がFLGのときのみ変化します。

【記述例】

```
POPC    SB
```

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= 264

【オペレーション】

dest*3 ← M(SP)
SP ← SP + n1*1 × 2
SP ← SP + n2*2 × 4
*1 n1はR0、R1、R2、R3の復帰するレジスタ数。
*2 n2はA0、A1、SB、FBの復帰するレジスタ数。
*3 dest がA0、A1、SB、FBのとき、下位3バイトが保存されます。

【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。

【選択可能なdest】

dest*3							
R0	R1	R2	R3	A0	A1	SB	FB

*3 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
```

PUSH

レジスタ／メモリ／即値の退避

PUSH

PUSH

【構文】

PUSH.size src

B , W , L

【命令コード／サイクル数】

Page= 265

【オペレーション】

サイズ指定子(.size)が(.B)のとき

SP ← SP − 2

M(SP)*1 ← src/[src]

サイズ指定子(.size)が(.W)のとき

SP ← SP − 2

M(SP) ← src/[src]

*1 上位8 ビットは不定となります。
サイズ指定子(.size) が(.B) でもSPは−2 されます。

サイズ指定子(.size)が(.L)のとき

SP ← SP − 4

M(SP)*2 ← src/[src]

*2 srcがアドレスレジスタ(A0、A1)のとき、
上位8ビットは0となります。

【機能】

・src をスタック領域へ退避します。
・サイズ指定子(.size) に(.W) を指定した場合、src がアドレスレジスタのとき、下位16 ビットを演算対象にします。

【選択可能なsrc】

src*3			
R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-	
~~A0/~~A0/A0	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16/#IMM32			

*3 間接アドレッシング[src]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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.L  R2R0
```

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= 267

【オペレーション】

SP ← SP − 4

M(SP)*1 ← EVA(src)

*1 上位8 ビットは不定になります。

【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/R2R0~~		~~R0H/R2/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~	
~~A0/A0/A0~~	~~A1/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

【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]
```

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= 267

【オペレーション】

srcがDCT0、DCT1、DMD0、DMD1、DRC0、DRC1、SVF、FLGの場合

SP ← SP － 2

M(SP)*1 ← src

*1 src がDMD0、DMD1のとき上位8 ビットは不定となります。

srcがFB、SB、SP、ISP、INTBの場合

SP ← SP － 4

M(SP)*2 ← src*3

*2 上位8 ビットは0 になります。

*3 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、4 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*3	ISP
INTB			
DCT0	DCT1	DMD0	DMD1
DRC0	DRC1	SVF	FLG

*3 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
```

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】

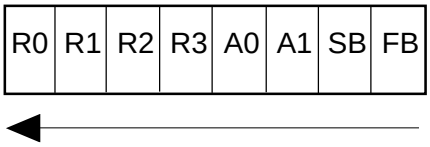
Page= 268

【オペレーション】

SP　　←　SP　－　n1*1　×　2
SP　　←　SP　－　n2*2　×　4
M(SP)*3　　←　src
*1　n1はR0、R1、R2、R3の退避するレジスタ数。
*2　n2はA0、A1、SB、FBの退避するレジスタ数。
*3　src がA0 、A1 、SB、FBのとき上位8 ビットは0 になります。

【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*4							
R0	R1	R2	R3	A0	A1	SB	FB

*4　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
```

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= 269

【オペレーション】

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

*1 下位8 ビットを保存します。

【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 スタック上の値になります。

【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= 269

【オペレーション】 *1

Repeat

	R1R2R0	←	R1R2R0 + 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R1R2R0 に格納します。
・命令終了時のアドレスレジスタの内容は、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。
・R1R2R0 には初期値を設定してください。

【フラグ変化】

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

条件
O ： 演算結果が+2 31-1 または−2 31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】

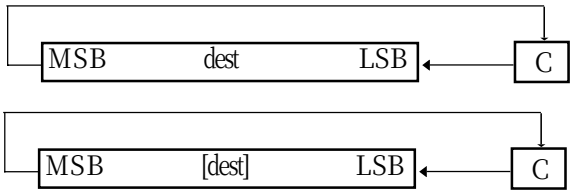
ROLC.size dest

B , W

【命令コード／サイクル数】

Page= 270

【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。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0~~/R2R0~~		R0H/R2~~/-~~	
R1L/R1~~/R3R1~~		R1H/R3~~/-~~	
~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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”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 ： 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
ROLC.W   [[A0]]
```

RORC

キャリー付き右回転
ROtate to Right with Carry

RORC

【構文】

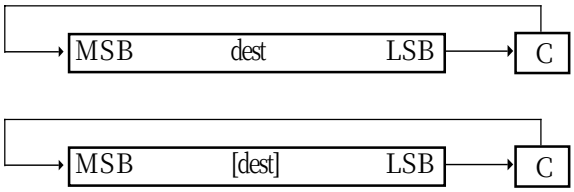
RORC.size dest

B , W

【命令コード／サイクル数】

Page= 270

【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。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なdest】

dest*1			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/~~A0/~~A0~~	~~A1/~~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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【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”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 ： 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
RORC.W   [[A0]]
```

ROT

回転
ROTate

ROT

【構文】

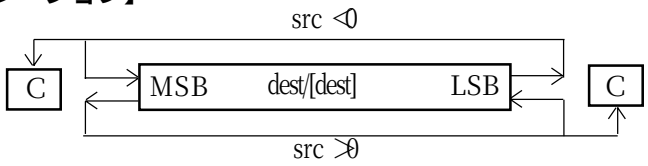
ROT.size src,dest

B , W

【命令コード／サイクル数】

Page= 271

【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〜+8(≠0)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、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。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、上位8 ビットは0 になります。

【選択可能なsrc / dest】

src				dest*1			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0~~/R2R0~~		R0H/R2~~/-~~	
~~R1L/R1/R3R1~~		R1H~~/R3/-~~		R1L/R1~~/R3R1~~*2		R1H/R3~~/-~~*2	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/~~A0~~/A0~~	~~A1/~~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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM4*3							

*1　間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。
*2　srcがR1Hの場合、destにR1またはR1Hを選択できません。
*3　取りうる範囲は−8≦#IMM4≦+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	−	−	−	−	○	○	−	○

*4　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
```

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= 272

【オペレーション】

PCML ← M(SP)
SP ← SP + 2
PCH ← M(SP) *1
SP ← SP + 2
*1 下位8 ビットを保存します。

【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= 273

【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 がアドレスレジスタ(A0 、A1) のとき、src をゼロ拡張して16 ビットで演算し、上位8 ビットは0 になります。また、src がアドレスレジスタ(A0 、A1) のとき、アドレスレジスタの下位8 ビットを演算の対象とします。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~*1	A1/A1/~~A1~~*1	[A0]	[A1]	A0/A0/~~A0~~*1	A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*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。
```

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= 275

【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。
・サイズ指定子(.size)に(.W)を指定した場合、destがアドレスレジスタ(A0、A1)のとき、上位8ビットは0になります。

【選択可能なsrc / dest / label】

src	dest				label
#IMM4[*1]	R0L/R0~~/R2R0~~		R0H/R2~~/-~~		**PC[*2] -126≦label≦PC[*2]+129**
	R1L/R1~~/R3R1~~		R1H/R3~~/-~~		
	~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]	
	dsp:24[A0]	dsp:24[A1]	abs24	abs16	

*1　取りうる範囲は−7≦#IMM4≦+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
SBJNZ.W    #2,[A1],label
```

SCCnd

条件設定

Store Condition on condition

SCCnd

【構文】

SCCnd dest

【命令コード／サイクル数】

Page= 276

【オペレーション】

if true then	dest	←	1	if true then	[dest]	←	1
else	dest	←	0	else	[dest]	←	0

【機能】

・*Cnd*で指定された条件が真のときにdest(1 ワード)に1を格納し、偽のときに0を格納します。

・dest にアドレスレジスタ(A0 、A1) を選択したとき、アドレスレジスタの上位8 ビットは0 になります。

・*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*1			
~~R0L~~/R0/~~R2R0~~		~~R0H~~/R2	
~~R1L~~/R1/~~R3R1~~		~~R1H~~/R3	
~~A0~~/A0/~~A0~~	~~A1~~/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]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

【記述例】

```
SCC     R0
SCC     [dsp:8[A0]]
```

SCMPU

ストリング比較
String CoMPare Unequal

SCMPU

【構文】

SCMPU.size
　　└──── **B , W**

【命令コード／サイクル数】

Page= 277

【オペレーション】

サイズ指定子(.size) が(.B)のとき

Repeat
　M(A0) －M(A1) (バイトで比較)
　tmp0 ←M(A0)
　tmp2 ←M(A1)
　A0 ←A0 ＋1
　A1 ←A1 ＋1
Until(tmp0 ＝0) ‖(tmp0 ≠tmp2)

tmp0,tmp2: 一時レジスタ

サイズ指定子(.size) が(.W)のとき

Repeat
　M(A0) －M(A1) (バイトで比較)
　If M(A0) ＝M(A1) **and** M(A0) ≠0 **then** M(A0 ＋1) －M(A1 ＋1)
　　(バイトで比較)
　tmp0 ←M(A0)
　tmp1 ←M(A0 ＋1)
　tmp2 ←M(A1)
　tmp3 ←M(A1 ＋1)
　A0 ←A0 ＋2
　A1 ←A1 ＋2
Until(tmp0 ＝0) ‖(tmp1 ＝0) ‖(tmp0 ≠tmp2) ‖(tmp1 ≠tmp3)

tmp0,tmp1,tmp2,tmp3:一時レジスタ

【機能】

・比較元番地(A0) から比較先番地(A1) にアドレスの加算方向へ比較の結果、不一致になるまで、またはM(A0)=0 、またはM(A0 ＋1) ＝0 (サイズ指定子(.size) が(.W) のとき) になるまでストリング比較を行います。
・命令終了時のアドレスレジスタ(A0 、A1) は、不定となります。
・命令実行中に割り込み要求があった場合は、１データ比較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	－	－	○	－	○	○	－	○

条件

Ｏ ： M(A0) －M(A1) の符号付き演算の結果、＋127 または－128 を超えると“１”、それ以外のとき“０”になります。
Ｓ ： M(A0) －M(A1) の演算の結果、MSBが“１”になると“１”、それ以外のとき“０”になります。
Ｚ ： M(A0) に0 を検出して終了したとき、M(A0) －M(A1) ＝0(比較結果が一致) のとき“１”、M(A0) －M(A1) ≠0(比較結果が不一致) のとき“０”になります。
Ｃ ： M(A0) －M(A1) の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】

```
SCMPU.W
```

SHA

算術シフト
SHift Arithmetic

SHA

【構文】

SHA.size src,dest

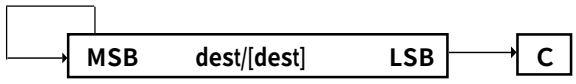
B , W , L

【命令コード／サイクル数】

Page= 278

【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、サイズ指定子(.size) に(.B) または(.W) を指定したとき、シフト回数は−8 〜+8(≠0) です。−9 以下、0 および+9 以上は設定できません。サイズ指定子(.size) に(.L) を指定したとき、シフト回数は−16 〜+16(≠0) です。−17 以下、0 および+17 以上は設定できません。
・src がレジスタの場合、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。
・サイズ指定子(.size) に(.L) を指定した場合、dest がアドレスレジスタ(A0 、A1) のとき、dest をゼロ拡張し32 ビットで演算を行い、演算結果の下位24 ビットをdest に格納します。

【選択可能なsrc / dest】

src				dest*1			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0/R2R0		R0H/R2/-	
~~R1L/R1/R3R1~~		R1H*2~~/R3/-~~		R1L/R1/R3R1*2		R1H/R3/-*2	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/~~A0/A0	~~A1/~~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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM4/#IMM8* 3							

*1 間接アドレッシング[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2 src がR1Hの場合、dest にR1、R1H またはR3R1 を選択できません。

*3 サイズ指定子(.size) に(.B) または(.W) を指定したとき、取りうる範囲は−8 ≦#IMM4≦+8(≠0) 、(.L) を指定したときは−16 ≦#IMM8≦+16(≠0) です。

【フラグ変化】*1

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*2 ：シフト結果のMSBとシフトアウトしたビットが全て同じ値のとき“ 0 ”、1つでも異なるとき“ 1 ”になります。

S*2 ：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*2 ：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*2 ：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。

*2　サイズ指定子(.size) に(.L) が指定されており、dest がアドレスレジスタ(A0/A1) の場合は不定。

【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,Ram:8[A1]
SHA.W    R1H,[[A1]]
```

SHL

論理シフト
SHift Logical

SHL

【構文】

SHL.size src,dest

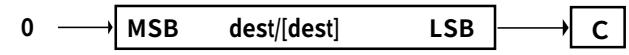
B , W , L

【命令コード／サイクル数】

Page= 281

【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、サイズ指定子(.size)に(.B)または(.W)を指定したとき、シフト回数は－8～+8(≠0)です。－9以下、0および+9以上は設定できません。サイズ指定子(.size)に(.L)を指定したとき、シフト回数は－16～+16(≠0)です。－17以下、0および+17以上は設定できません。
・srcがレジスタの場合、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*1			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0/R2R0		R0H/R2/-	
~~R1L/R1/R3R1~~		R1H*2~~/R3/-~~		R1L/R1/R3R1*2		R1H/R3/-*2	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	~~A0/~~A0/A0	~~A1/~~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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM4/IMM8*3							

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。
*2 srcがR1Hの場合、destにR1、R1HまたはR3R1を選択できません。
*3 サイズ指定子(.size)に(.B)または(.W)を指定したとき、取りうる範囲は－8≦#IMM4≦+8(≠0)、(.L)を指定したときは－16≦#IMM8≦+16(≠0)です。

【フラグ変化】*1

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*2：　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*2：　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*2：　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。

*2　サイズ指定子(.size) に(.L) が指定されており、dest がアドレスレジスタ(A0/A1) の場合は不定。

【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,Ram:8[A1]
SHL.W    R1H,[[A0]]
```

SIN

ストリング入力
String INput

SIN

【構文】

SIN.size

B , W

【命令コード／サイクル数】

Page= 283

【オペレーション】 *1

サイズ指定子(.size) が(.B)のとき

While R3 ≠0 **Do**

M(A1) ← M(A0)
A1 ← A1 + 1
R3 ← R3 − 1

end

サイズ指定子(.size) が(.W)のとき

While R3 ≠0 **Do**

M(A1) ← M(A0)
A1 ← A1 + 2
R3 ← R3 − 1

end

*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】

・A0 で示される固定の転送元番地からR3 で指定される回数、A1 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A1 は、最後に転送したデータの次の番地を示します。
・命令実行中に割り込み要求があった場合は、１データ転送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】

SIN.W

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= 284

【オペレーション】 *1

サイズ指定子(.size) が(.B)のとき

While R3 ≠0 **Do**

M(A1) ← M(A0)
A0 ← A0 － 1
A1 ← A1 － 1
R3 ← R3 － 1

end

サイズ指定子(.size) が(.W)のとき

While R3 ≠0 **Do**

M(A1) ← M(A0)
A0 ← A0 － 2
A1 ← A1 － 2
R3 ← R3 － 1

end

*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】

・A0 で示される転送元番地からA1 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。
・転送回数はR3 に設定します。
・命令終了時のアドレスレジスタ(A0 、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
```

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= 284

【オペレーション】 *1

サイズ指定子(.size) が(.B)のとき

While R3 ≠0 **Do**

M(A1) ← M(A0)
A0 ← A0 + 1
A1 ← A1 + 1
R3 ← R3 − 1

end

サイズ指定子(.size) が(.W)のとき

While R3 ≠0 **Do**

M(A1) ← M(A0)
A0 ← A0 + 2
A1 ← A1 + 2
R3 ← R3 − 1

end

*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】

・A0 で示される転送元番地からA1 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。
・転送回数はR3 に設定します。
・命令終了時のアドレスレジスタ(A0 、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
```

SMOVU

ストリング転送比較
String MOVe Unequal

SMOVU

【構文】

SMOVU.size
　　└──── **B , W**

【命令コード／サイクル数】

Page= 285

【オペレーション】

サイズ指定子(.size)が(.B)のとき

Repeat
　M(A1) ← M(A0) (バイトで転送)
　tmp0 ← M(A0)
　A0 ← A0 + 1
　A1 ← A1 + 1
Until tmp0＝0

tmp0: 一時レジスタ

サイズ指定子(.size)が(.W)のとき

Repeat
　M(A1) ← M(A0)(ワードで転送)
　tmp0 ← M(A0)
　tmp1 ← M(A0＋1)
　A0 ← A0 + 2
　A1 ← A1 + 2
Until (tmp0＝0) ‖ (tmp1＝0)

tmp0, tmp1: 一時レジスタ

【機能】

・A0で示される転送元番地からA1で示される転送先番地にアドレスの加算方向へ0が検出されるまでストリング転送を行います。
・命令終了時のアドレスレジスタ(A0、A1)は、不定となります。
・命令実行中に割り込み要求があった場合は、１データ転送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】

```
SMOVU.B
```

SOUT

ストリング出力
String OUTput

SOUT

【構文】

SOUT.size
　　└──── **B , W**

【命令コード／サイクル数】

Page= 285

【オペレーション】 *1

サイズ指定子(.size) が(.B)のとき

While R3 ≠0 **Do**
　M(A1) ← M(A0)
　A0 ← A0 + 1
　R3 ← R3 − 1
end

サイズ指定子(.size) が(.W)のとき

While R3 ≠0 **Do**
　M(A1) ← M(A0)
　A0 ← A0 + 2
　R3 ← R3 − 1
end

*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】

・A0で示される転送元番地からR3で指定される回数、A1で示される固定の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は、最後に転送したデータの次の番地を示します。
・命令実行中に割り込み要求があった場合は、１データ転送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】

```
SOUT.W
```

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= 286

【オペレーション】[*1]

サイズ指定子(.size) が(.B)のとき

While R3 ≠0 **Do**
　　M(A1) ← R0L
　　A1 ← A1 + 1
　　R3 ← R3 − 1
end

サイズ指定子(.size) が(.W)のとき

While R3 ≠0 **Do**
　　M(A1) ← R0
　　A1 ← A1 + 2
　　R3 ← R3 − 1
end

*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】

・R0L/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
```

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= 286

【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 にメモリを指定したとき、以下のバイト数分メモリが必要となります。

2バイト：DMD0*1、 DMD1*1、FLG、 DCT0 、 DCT1 、DRC0、DRC1、SVF

4バイト：FB*1、SB*1、SP*1、ISP*1、INTB*1、VCT*1、SVP*1、DMA0*1、DMA1*1、DRA0*1、DRA1*1、DSA0*1、DSA1*1

*1 dest の上位1バイトは不定になります。

【選択可能なsrc / dest】

src				dest			
DMD0	DMD1	DCT0	DCT1	~~R0L/~~R0~~/R2R0~~		~~R0H/~~R2~~/-~~	
DRC0	DRC1	FLG	SVF	~~R1L/~~R1~~/R3R1~~		~~R1H/~~R3~~/-~~	
				~~A0/~~A0~~/A0~~	~~A1/~~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]	dsp:16[FB]
				dsp:24[A0]	dsp:24[A1]	abs24	abs16
FB	SB	SP*2	ISP	~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
INTB	VCT	SVP		~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
DMA0	DMA1	DRA0	DRA1	~~A0/A0/~~A0	~~A1/A1/~~A1	[A0]	[A1]
DSA0	DSA1			dsp:8[A0]	dsp:8[A1]	dsp:8[SB]	dsp:8[FB]
				dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
				dsp:24[A0]	dsp:24[A1]	abs24	abs16

*2 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     FLG,R0
STC     FB,A0
```

STCTX

コンテキストの退避
STore ConTeXt

STCTX

【構文】

STCTX　　abs16,abs24

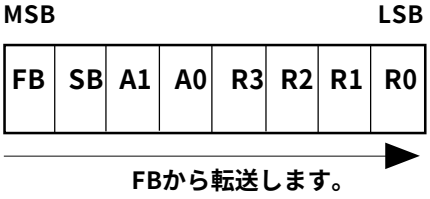
【命令コード／サイクル数】

Page= 288

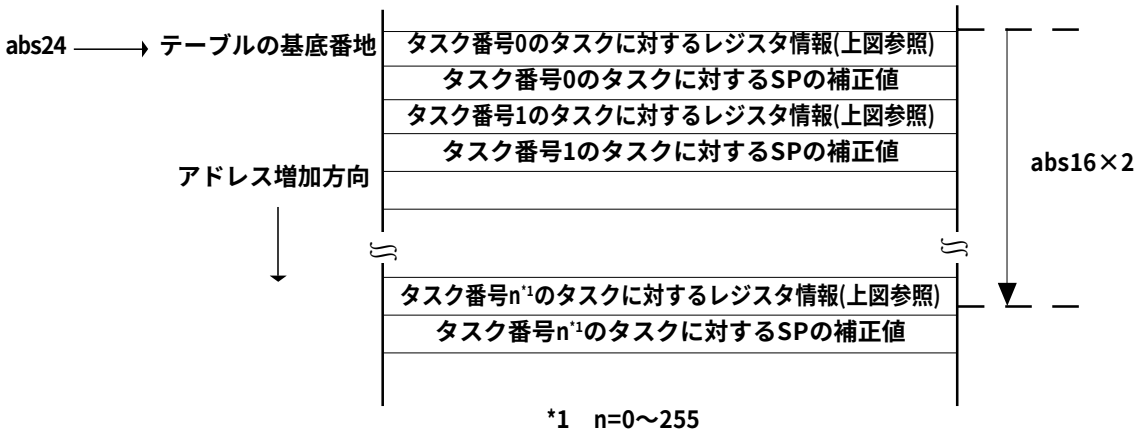
【オペレーション】

【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、abs24 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。R0、R1、R2、R3レジスタを転送するとき2バイト、A0、A1、SB、FBは4バイトとして計算されます。
・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。abs24 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。



【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
```

STNZ

条件付き転送

STore on Not Zero

STNZ

【構文】

STNZ.size src,dest

B , W

【命令コード／サイクル数】

Page= 288

【オペレーション】

if Z= 0 **then** dest ← src

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。“１”のときdest は変化しません。
・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(A0 、A1) のとき、src をゼロ拡張し16 ビットで演算します。そのとき、上位8 ビットは0 になります。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。

【選択可能なsrc / dest】

src				dest*1			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0~~/R2R0~~		R0H/R2~~/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L/R1~~/R3R1~~		R1H/R3~~/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	A0/A0~~/A0~~	A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

【記述例】

```
STNZ.B    #5,Ram:8[SB]
STNZ.W    #15,[[A1]]
```

STZ

条件付き転送

STore on Zero

STZ

【構文】

STZ.size src,dest

B , W

【命令コード／サイクル数】

Page= 289

【オペレーション】

if Z= 1 **then** dest ← src

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。“０”のときdest は変化しません。

・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(A0 、A1) のとき、src をゼロ拡張し16 ビットで演算します。そのとき、上位8 ビットは0 になります。

・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。

【選択可能なsrc / dest】

src				dest*1			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	A0/A0/~~A0~~	A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

【記述例】

```
STZ.B    #5,Ram:8[SB]
STZ.W    #10,[[A0]]
```

STZX

条件付き転送

STore on Zero eXtention

STZX

【構文】

STZX.size src1,src2,dest

└──── **B , W**

【命令コード／サイクル数】

Page= 289

【オペレーション】

If Z = 1 **then** dest ← src1
else dest ← src2

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。
・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(A0 、A1) のとき、src をゼロ拡張し16ビットで演算します。そのとき、上位8 ビットは0 になります。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。

【選択可能なsrc / dest】

src				dest*1			
~~R0L/R0/R2R0~~		~~R0H/R2/-~~		R0L/R0~~/R2R0~~		R0H/R2~~/-~~	
~~R1L/R1/R3R1~~		~~R1H/R3/-~~		R1L/R1~~/R3R1~~		R1H/R3~~/-~~	
~~A0/A0/A0~~	~~A1/A1/A1~~	~~[A0]~~	~~[A1]~~	A0/A0~~/A0~~	A1/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]~~	~~dsp:16[FB]~~	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
~~dsp:24[A0]~~	~~dsp:24[A1]~~	~~abs24~~	~~abs16~~	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1 間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

【記述例】

```
STZX.B  #1,#2,Ram:8[SB]
STZX.W  #5,#10,[R0]
```

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 , L

【命令コード／サイクル数】

Page= 290

【オペレーション】

dest ← dest ― src　　[dest] ← [dest] ― src
dest ← dest ― [src]　　[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 がアドレスレジスタ(A0 、A1) のとき、src をゼロ拡張し16 ビットで演算します。そのとき、上位8 ビットは0 になります。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 ビットを演算の対象とします。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。
・サイズ指定子(.size)に(.L)を指定した場合、destがアドレスレジスタのとき、destをゼロ拡張し32ビットで演算を行い、演算結果の下位24ビットをdestに格納します。またsrcがアドレスレジスタのときsrcをゼロ拡張し、32ビットで演算を行います。32ビットの演算結果でフラグも変化します。

【選択可能なsrc / dest】 *1　（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/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16/#IMM32							

*1　間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。
*2　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、+2147483647(.L) または2147483648(.L) 、+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]
```

【フォーマット別src / dest】

G フォーマット*1

src				dest			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*2	A1/A1/A1*2	[A0]	[A1]	A0/A0/A0*2	A1/A1/A1*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16/#IMM32							

*1 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*2 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

src				dest*3			
~~R0L/R0~~	~~dsp:8[SB]~~	~~dsp:8[FB]~~	~~abs16~~	R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16
#IMM8/#IMM16*4							

*3 間接アドレッシング[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

*4 サイズ指定子(.size) には(.B) と(.W) のみ指定できます。

SUBX

符号拡張ボローなし減算
SUBtract eXtend

SUBX

【構文】

SUBX **src,dest**

【命令コード／サイクル数】

Page= 294

【オペレーション】

dest ← dest − EXT(src)　　[dest] ← [dest] − EXT(src)

dest ← dest − EXT([src])　　[dest] ← [dest] − EXT([src])

【機能】

・32 ビットのdest から8 ビットのsrc を32 ビット符号拡張したものを減算し、dest に格納します。

・destがアドレスレジスタ(A0 、A1) のとき、destをゼロ拡張し32ビットで演算を行い、演算結果の下位24ビットをdestに格納します。32ビットの演算結果でフラグも変化します。

【選択可能なsrc / dest】 *1

src				dest			
R0L/~~R0/R2R0~~		R0H/~~R2/-~~		~~R0L/R0/~~R2R0		~~R0H/R2/-~~	
R1L/~~R1/R3R1~~		R1H/~~R3/-~~		~~R1L/R1/~~R3R1		~~R1H/R3/-~~	
A0/~~A0/A0~~	A1/~~A1/A1~~	[A0]	[A1]	~~A0/A0/~~A0	~~A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8							

*1 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp:8[SP] 、#IMM以外で使用できます。

【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、+2147483647 または−2147483648 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】

```
SUBX    R0L,A0
SUBX    Ram:8[SB],R2R0
SUBX    #2,[A0]
```

TST

テスト
TeST

TST

【構文】

TST.size (:format)　　src,dest

G , S（指定可能）
B , W

【命令コード／サイクル数】

Page= 296

【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 がアドレスレジスタ(A0 、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 ビットを演算の対象とします。
・サイズ指定子(.size) に(.W) を指定した場合、src がアドレスレジスタのとき、アドレスレジスタの下位16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/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*1	A1/A1/A1*1	[A0]	[A1]	A0/A0/A0*1	A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*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。
```

【フォーマット別src / dest】

G フォーマット

src				dest			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0*1	A1/A1/A1*1	[A0]	[A1]	A0/A0/A0*1	A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*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

src				dest			
R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16	R0L/R0	dsp:8[SB]	dsp:8[FB]	abs16
#IMM8/#IMM16							

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= 298

【オペレーション】

SP ← SP − 2
M(SP) ← FLG
SP ← SP − 2
M(SP)*1 ← (PC + 1)H
SP ← SP − 2
M(SP) ← (PC + 1)ML
PC ← M(FFFFDC16)
*1 上位8ビットは不定になります。

【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= 298

【オペレーション】

【機能】

・プログラムの実行を停止します。ストップ/ ウエイト復帰用割り込み優先順位設定ビットよりも高い優先順位の割り込みを受け付けるか、リセットが発生するとプログラムの実行を開始します。

【フラグ変化】

ﾌﾗｸﾞ	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= 299

【オペレーション】

dest ⟷ src　　[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 がアドレスレジスタ(A0 、A1) のとき、src をゼロ拡張した24 ビットのデータがアドレスレジスタに入り、アドレスレジスタの下位8 ビットがsrc に入ります。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、src をゼロ拡張した24 ビットのデータがアドレスレジスタに入り、アドレスレジスタの下位16 ビットがsrc に入ります。
src がアドレスレジスタのとき、dest をゼロ拡張した24 ビットのデータがアドレスレジスタに入り、アドレスレジスタの下位16 ビットがdest に入ります。

【選択可能なsrc / dest】

src				dest*1			
R0L/R0/R2R0		R0H/R2/-		R0L/R0/R2R0		R0H/R2/-	
R1L/R1/R3R1		R1H/R3/-		R1L/R1/R3R1		R1H/R3/-	
A0/A0/A0	A1/A1/A1	[A0]	[A1]	A0/A0/A0	A1/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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM							

*1　間接アドレッシング[dest]は、R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp: 8[SP]、#IMM以外で使用できます。

【フラグ変化】

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

【記述例】

```
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]
```

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= 299

【オペレーション】

dest ← dest ∀ src　　[dest] ← [dest] ∀ src
dest ← dest ∀ [src]　　[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 がアドレスレジスタ(A0 、A1) のとき、src をゼロ拡張し16 ビットで演算し、上位8 ビットは0 になります。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 ビットを演算の対象とします。
・サイズ指定子(.size) に(.W) を指定した場合、dest がアドレスレジスタのとき、上位8 ビットは0 になります。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16 ビットを演算の対象とします。

【選択可能なsrc / dest】 *1

src				dest			
R0L/R0/~~R2R0~~		R0H/R2/~~-~~		R0L/R0/~~R2R0~~		R0H/R2/~~-~~	
R1L/R1/~~R3R1~~		R1H/R3/~~-~~		R1L/R1/~~R3R1~~		R1H/R3/~~-~~	
A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[A0]	[A1]	A0/A0/~~A0~~*2	A1/A1/~~A1~~*2	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16
#IMM8/#IMM16							

*1 間接アドレッシング[src] 、[dest] は R0L/R0/R2R0 、R0H/R2/- 、R1L/R1/R3R1 、R1H/R3/- 、SP/SP/SP 、dsp: 8[SP] 、#IMM以外で使用できます。
*2 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
XOR.W    [A0],[[A1]]
```

3.3 インデックス命令

本項ではインデックス命令について命令ごとに説明しています。

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

モディファイできる領域は 0 ～ 65535(64KB)です。

インデックス命令の直後には、割り込み要求を受け付けません。

インデックス命令は、以下に示す 10 種類があります。

(1) INDEXB.size　src

INDEXB 命令(INDEX Byte)はバイト単位の配列に対応します。

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

INDEXB 命令の次に実行する命令では src、dest ともにメモリを選択してください。また、サイズ指定子は .B を指定してください。

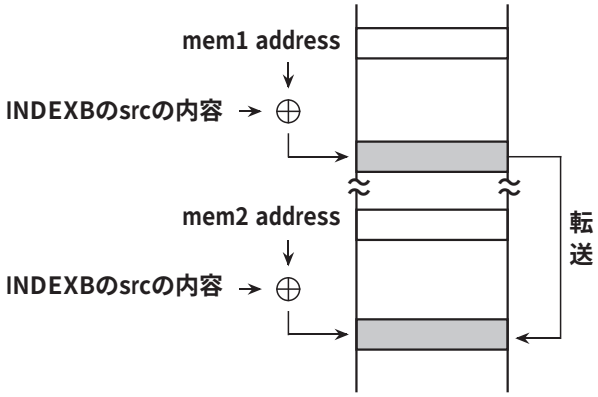
例：

```
INDEXB.B      src
MOV.B:G       mem1, mem2
       .Bを指定          メモリ
```

C 言語での動作説明

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

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



INDEXB 命令がモディファイする命令

ADC、ADD:G[*1*2]、AND、CMP:G[*1]、MAX、MIN、MOV:G[*1*3]、MUL、MULU、OR、SBB、SUB、TST、XOR 命令の src と dest

*1　Gフォーマットのみ指定できます。

*2　ADD 命令の dest に SP は使用できません。

*3　MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXB 命令の次に使用しないでください。

(2) INDEXBD.size src

INDEXBD 命令(INDEX Byte Dest)はバイト単位の配列に対応します。

INDEXBD 命令は次に実行する命令の dest(一部命令では src)が示すアドレスに INDEXBD 命令の src の内容を符号なしで加算し実行アドレスとします。

INDEXBD 命令の次に実行する命令では dest(一部命令では src)にメモリを選択してください。また、サイズ指定子は .B を指定してください。

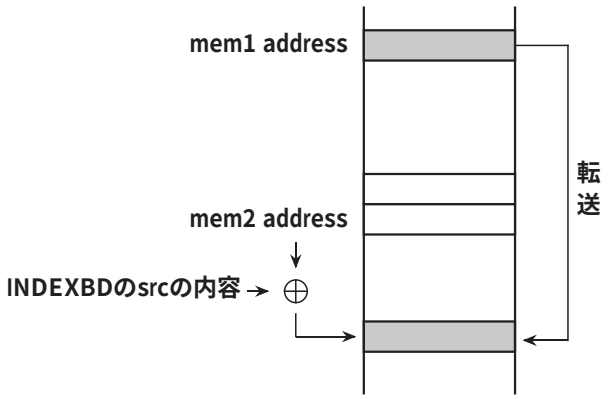
例：

```
INDEXBD.B    src
MOV.B:G              mem1,mem2
      .Bを指定                 メモリ
```

C 言語での動作説明

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

mem2[src] = mem1;
```



INDEXBD 命令がモディファイする命令

ABS、ADC、ADCF、ADD:G*1*2、AND、CLIP、CMP:G*1、DEC、INC、MAX、MIN、MOV:G*1*3、MUL、MULU、NEG、NOT、OR、POP、ROLC、RORC、ROT、SBB、SHA、SHL、STNZ、STZ、STZX、SUB、TST、XCHG、XOR 命令の dest

DIV、DIVU、DIVX、PUSH 命令の src

*1 Gフォーマットのみ指定できます。

*2 ADD 命令の dest に SP は使用できません。

*3 MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXBD 命令の次に使用しないでください。

(3) INDEXBS.size src

INDEXBS 命令(INDEX Byte Src)はバイト単位の配列に対応します。

INDEXBS 命令は次に実行する命令の src が示すアドレスに INDEXBS 命令の src の内容を符号なしで加算し実行アドレスとします。

INDEXBS 命令の次に実行する命令では src にメモリを選択してください。また、サイズ指定子は .B を指定してください。

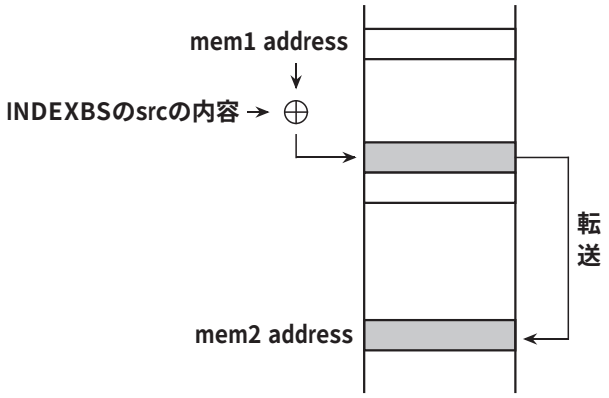
例：

```
INDEXBS.B     src
MOV.B:G       mem1, mem2
       .Bを指定          メモリ
```

C 言語での動作説明

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

mem2= mem1[src];
```



INDEXBS 命令がモディファイする命令

ADC、ADD:G[*1*2]、AND、CMP:G[*1]、MAX、MIN、MOV:G[*1*3]、MUL、MULU、OR、SBB、SUB、TST、XOR 命令の src

*1　Gフォーマットのみ指定できます。

*2　ADD 命令の dest に SP は使用できません。

*3　MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXBS 命令の次に使用しないでください。

(4) INDEXW.size src

INDEXW 命令(INDEX Word)はワード単位の配列に対応します。

INDEXW 命令は次に実行する命令の src、dest が示すアドレスに INDEXW 命令の src の内容を 2 倍した値を符号なしで加算し実行アドレスとします。INDEXW 命令の src の取りうる範囲は 0 ～ 32767 です。それ以上を設定しないでください。

INDEXW 命令の次に実行する命令では src、dest ともにメモリを選択してください。また、サイズ指定子は .W を指定してください。

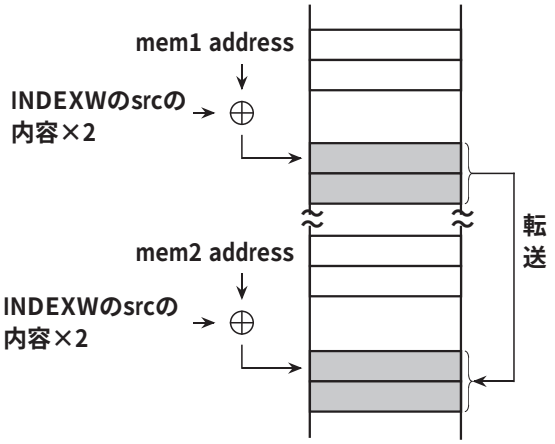
例：

```
INDEXW.B    src
MOV.W:G             mem1, mem2
      .Wを指定                  メモリ
```

C 言語での動作説明

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

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



INDEXW 命令がモディファイする命令

ADC、ADD:G[*1*2]、AND、CMP:G[*1]、MAX、MIN、MOV:G[*1*3]、MUL、MULU、OR、SBB、SUB、TST、XOR 命令の src と dest

*1 Gフォーマットのみ指定できます。

*2 ADD 命令の dest に SP は使用できません。

*3 MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXW 命令の次に使用しないでください。

(5) INDEXWD.size src

INDEXWD 命令(INDEX Word Dest)はワード単位の配列に対応します。

INDEXWD 命令は次に実行する命令の dest(一部命令では src)が示すアドレスに INDEXWD 命令の src の内容を 2 倍した値を符号なしで加算し実行アドレスとします。INDEXWD 命令の src の取りうる範囲は 0 ～ 32767 です。それ以上を設定しないでください。

INDEXWD 命令の次に実行する命令では dest(一部命令では src)にメモリを選択してください。また、サイズ指定子は .W を指定してください。

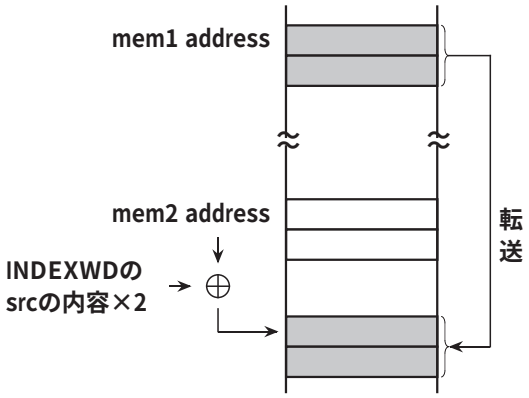
例：

```
INDEXWD.B   src
MOV.W:G     mem1, mem2
       .Wを指定          メモリ
```

C 言語での動作説明

```
char        src;
int         mem1;
int         mem2[];

mem2[src] = mem1;
```



INDEXWD 命令がモディファイする命令

ABS、ADC、ADCF、ADD:G[*1][*2]、AND、CLIP、CMP:G[*1]、DEC、INC、MAX、MIN、MOV:G[*1][*3]、MUL、MULU、NEG、NOT、OR、POP、ROLC、RORC、ROT、SBB、SCcnd、SHA、SHL、STNZ、STZ、STZX、SUB、TST、XCHG、XOR 命令の dest

DIV、DIVU、DIVX、PUSH、JMPI、JSRI 命令の src

*1　Gフォーマットのみ指定できます。

*2　ADD 命令の dest に SP は使用できません。

*3　MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXWD 命令の次に使用しないでください。

(6) INDEXWS.size src

INDEXWS 命令(INDEX Word Src)はワード単位の配列に対応します。

INDEXWS 命令は次に実行する命令の src が示すアドレスに INDEXWS 命令の src の内容を 2 倍した値を符号なしで加算し実行アドレスとします。INDEXWS 命令の src の取りうる範囲は 0 ～ 32767 です。それ以上を設定しないでください。

INDEXWS 命令の次に実行する命令では src にメモリを選択してください。また、サイズ指定子は .W を指定してください。

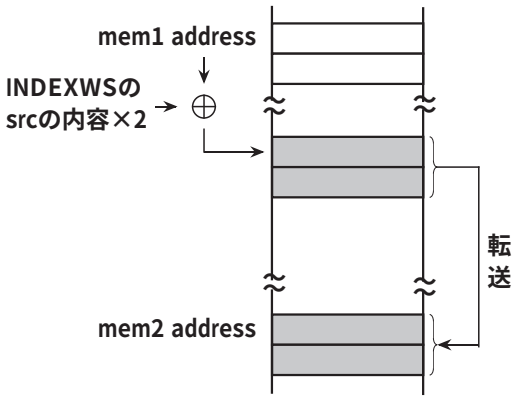
例：

```
INDEXWS.B    src
MOV.W:G      mem1, mem2
       .Wを指定          メモリ
```

C 言語での動作説明

```
char          src;
int           mem1[];
int           mem2;

mem2 = mem1[src];
```



INDEXWS 命令がモディファイする命令

ADC、ADD:G[*1*2]、AND、CMP:G[*1]、MAX、MIN、MOV:G[*1*3]、MUL、MULU、OR、SBB、SUB、TST、XOR 命令の src

*1　Gフォーマットのみ指定できます。

*2　ADD 命令の dest に SP は使用できません。

*3　MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXWS 命令の次に使用しないでください。

(7) INDEXL.size src

INDEXL 命令(INDEX Long word)はロングワード単位の配列に対応します。

INDEXL 命令は次に実行する命令の src、dest が示すアドレスに INDEXL 命令の src の内容を 4 倍した値を符号なしで加算し実行アドレスとします。INDEXL 命令の src の取りうる範囲は 0 ～ 16383 です。それ以上を設定しないでください。

INDEXL 命令の次に実行する命令では src、dest ともにメモリを選択してください。また、サイズ指定子は .L を指定してください。

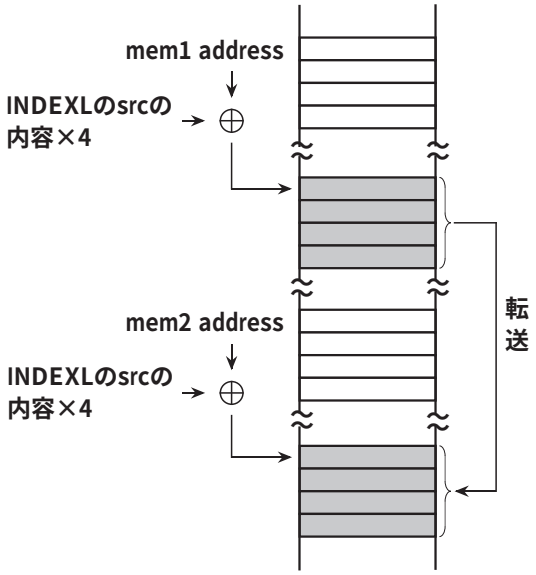
例：

```
INDEXL.B      src
MOV.L:G       mem1, mem2
      .Lを指定          メモリ
```

C 言語での動作説明

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

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



INDEXL 命令がモディファイする命令

ADD:G*1*2、CMP:G*1、MOV:G*1*3、SUB 命令の src と dest

*1　G フォーマットのみ指定できます。

*2　ADD 命令の dest に SP は使用できません。

*3　MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXL命令の次に使用しないでください。

(8) INDEXLD.size src

INDEXLD 命令(INDEX Long word Dest)はロングワード単位の配列に対応します。

INDEXLD 命令は次に実行する命令の dest(一部命令では src)が示すアドレスに INDEXLD 命令の src の内容を 4 倍した値を符号なしで加算し実行アドレスとします。INDEXLD 命令の src の取りうる範囲は 0 ～ 16383 です。それ以上を設定しないでください。

INDEXLD 命令の次に実行する命令では dest(一部命令では src)にメモリを選択してください。また、サイズ指定子は .L を指定してください。

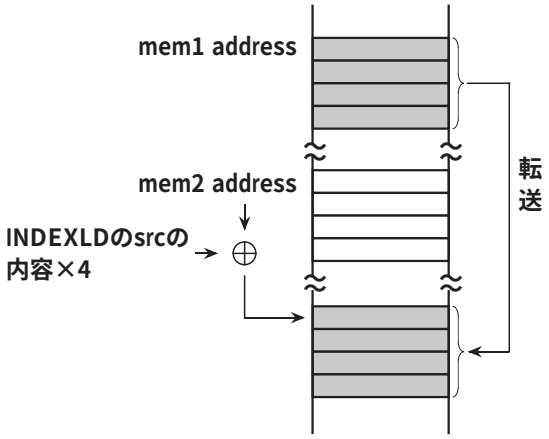
例：

```
INDEXLD.B    src
MOV.L:G      mem1, mem2
       .Lを指定          メモリ
```

C 言語での動作説明

```
char         src;
long         mem1;
long         mem2[];

mem2[src] = mem1;
```



INDEXLD 命令がモディファイする命令

ADD:G*1*2、CMP:G*1、MOV:G*1*3、SUB、SHA、SHL 命令の dest

JMPI、JSRI 命令の src

*1　G フォーマットのみ指定できます。

*2　ADD 命令の dest に SP は使用できません。

*3　MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXLD命令の次に使用しないでください。

(9) INDEXLS.size src

INDEXLS 命令(INDEX Long word Src)はロングワード単位の配列に対応します。

INDEXLS 命令は次に実行する命令の src が示すアドレスに INDEXLS 命令の src の内容を 4 倍した値を符号なしで加算し実行アドレスとします。INDEXLS 命令の src の取りうる範囲は 0 ～ 16383 です。それ以上を設定しないでください。

INDEXLS 命令の次に実行する命令では src にメモリを選択してください。また、サイズ指定子は .L を指定してください。

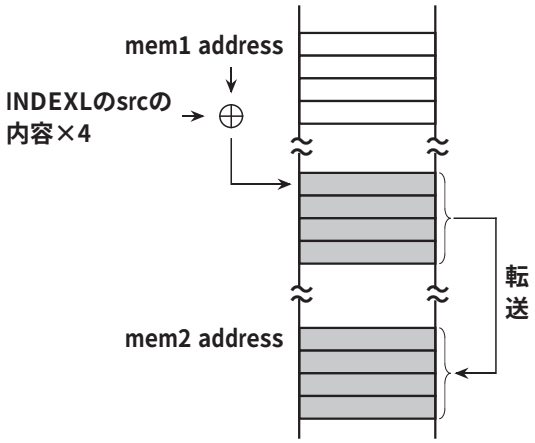
例：

```
INDEXLS.B     src
MOV.L:G       mem1, mem2
       .Lを指定          メモリ
```

C 言語での動作説明

```
char          src;
long          mem1[];
long          mem2;

mem2 = mem1[src];
```



INDEXLS 命令がモディファイする命令

ADD:G*1*2、CMP:G*1、MOV:G*1*3、SUB 命令の src

*1　G フォーマットのみ指定できます。

*2　ADD 命令の dest に SP は使用できません。

*3　MOV 命令の src または dest に dsp:8[SP]は使用できません。

上記以外の命令を INDEXLS 命令の次に使用しないでください。

(10) BITINDEX.size src

BITINDEX 命令は次に実行する命令の src または dest が示すアドレスのビット 0 から BITINDEX の src で示したビット数だけ離れたビットを対象とします。

BITINDEX 命令の次に実行する命令はビット命令としてください。また、src または dest にはメモリを指定してください。

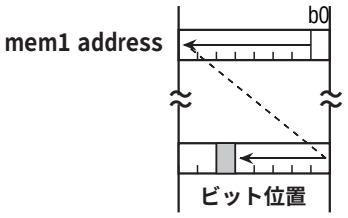
例：

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

ビット命令

無効となります。

メモリ



BITINDEX 命令がモディファイする命令

BAND、BNAND、BNOR、BNTST、BNXOR、BOR、BTST:G*1、BXOR 命令の src
BCLR、BMcnd、BNOT、BSET、BTSTC、BTSTS 命令の dest

*1 Gフォーマットのみ指定できます。

(11)インデックス命令の次に実行できる命令

それぞれのインデックス命令の次に実行できる命令を下表に示します。

	有効命令	
INDEXB.B/.W*2	ADC, ADD:G*4, AND, CMP:G, MAX, MIN, MOV:G*3, MUL, MULU, OR, SBB, SUB, TST, XOR 上記命令のsrcとdest	
INDEXBD.B/.W*2	ABS, ADC, ADCF, ADD:G*4, AND, CLIP, CMP:G, DEC, INC, MAX, MIN, MOV:G*3, MUL, MULU, NEG, NOT, OR, POP, ROLC, RORC, ROT, SBB, SHA, SHL, STNZ, STZ, STZX, SUB, TST, XCHG, XOR 上記命令のdest	DIV, DIVU, DIVX, PUSH 上記命令のsrc
INDEXBS.B/.W*2	ADC, ADD:G*4, AND, CMP:G, MAX, MIN, MOV:G*3, MUL, MULU, OR, SBB, SUB, TST, XOR 上記命令のsrc	
INDEXW.B/.W*2	ADC, ADD:G*4, AND, CMP:G, MAX, MIN, MOV:G*3, MUL, MULU, OR, SBB, SUB, TST, XOR 上記命令のsrcとdest	
INDEXWD.B/.W*2	ABS, ADC, ADCF, ADD:G*4, AND, CLIP, CMP:G, DEC, INC, MAX, MIN, MOV:G*3, MUL, MULU, NEG, NOT, OR, POP, ROLC, RORC, ROT, SBB, SHA, SHL, STNZ, STZ, STZX, SUB, TST, XCHG, XOR 上記命令のdest	DIV, DIVU, DIVX, PUSH, JMPI, JSRI 上記命令のsrc
INDEXWS.B/.W*2	ADC, ADD:G*4, AND, CMP:G, MAX, MIN, MOV:G*3, MUL, MULU, OR, SBB, SUB, TST, XOR 上記命令のsrc	
INDEXL.B/.W*2	ADD:G*4, CMP:G, MOV:G*3, SUB 上記命令のsrcとdest	
INDEXLD.B/.W*2	ADD:G*4, CMP:G, MOV:G*3, SHA, SHL, SUB 上記命令のdest	JMPI*1, JSRI*1 上記命令のsrc
INDEXLS.B/.W*2	ADD:G*4, CMP:G, MOV:G*3, SUB 上記命令のsrc	
BITINDEX.B/.W	BAND, BNAND, BNOR, BNTST, BNXOR, BOR, BTST:G, BXOR 上記命令のsrc	BCLR, BMcnd, BNOT, BSET, BTSTC, BTSTS 上記命令のdest

*1 サイズが.A(3バイト)を.L(4バイト)で指定するため、データテーブルの取り方に注意が必要です。

*2 ADD, CMP, MOV命令はGフォーマットのみ有効です。

*3 MOV命令のsrc またはdest に、dsp:8[SP] は使用できません。

*4 ADD命令のdest にSP は使用できません。

(12)アドレッシングモード

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

src				dest			
[A0]	[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]	dsp:16[FB]	dsp:16[A0]	dsp:16[A1]	dsp:16[SB]	dsp:16[FB]
dsp:24[A0]	dsp:24[A1]	abs24	abs16	dsp:24[A0]	dsp:24[A1]	abs24	abs16

*1 MOV命令はdsp8:[SP] は使用できません。

*2 ADD命令のSP は使用できません。

*3 R0L/R0/R2R0、R0H/R2/-、R1L/R1/R3R1、R1H/R3/-、SP/SP/SP、dsp:8[SP]、#IMMは使用できません。

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

b7				b0	b7				b0
1	1	0	1	0 1 0 1	1	1	1	0 1	#IMM

【バイト数／サイクル数】

④

バイト数/サイクル数	2/2

① MAX

② (1) MAX.size　　#IMM,dest

③

b7		b0	b7						b0	b7							b0
0000	0001		1	0	0	0	d4	d3	d2	SIZE	d1	d0	1	1	1	1	1

dest コード：dsp8 / dsp16/abs16 / dsp24/abs24 → #IMM8 / #IMM16

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/3	4/3	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5

①ニーモニック

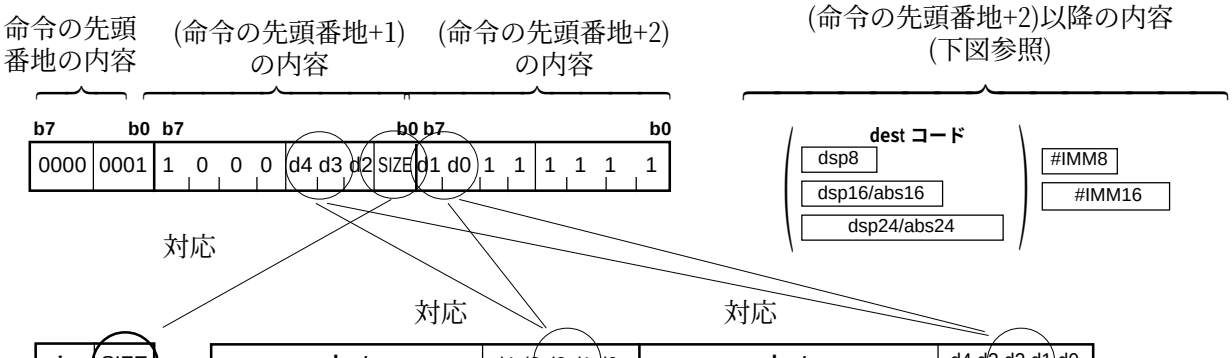
本ページで説明するニーモニックを示しています。

②構文

命令の構文を記号で示しています。

③命令コード

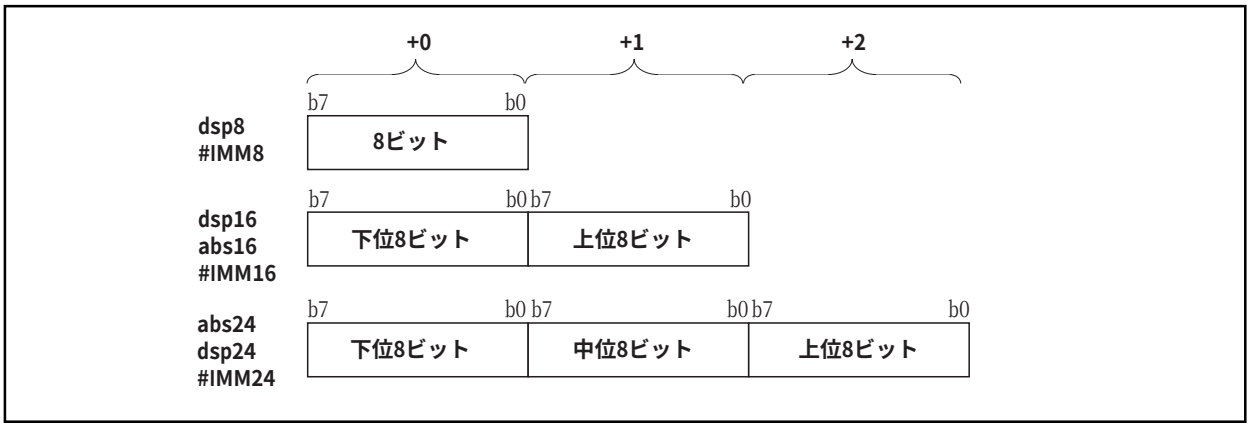
命令コードを示しています。() 内は選択する src/dest によって省略されます。



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

(命令の先頭番地 +2)以降は下記のとおり配置されます。

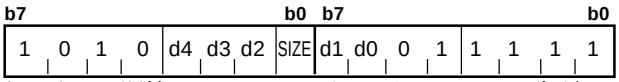


④バイト数／サイクル数表

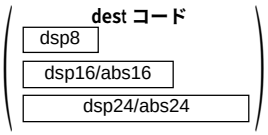
本命令の実行に必要なサイクル数と命令のバイト数を示しています。
サイクル数は最短時のものを記載しているため、命令キューバッファに取り込まれてるバイト数、外部バスを8ビットとして使用し外部メモリ領域をアクセスした場合や、バスサイクルに対するウエイト挿入の有無等の影響により増える可能性があります。
スラッシュの左側がバイト数、右側がサイクル数です。

ABS

(1) ABS.size dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

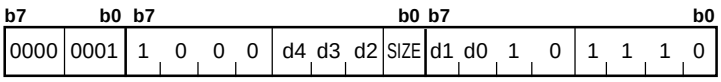
【バイト数／サイクル数】

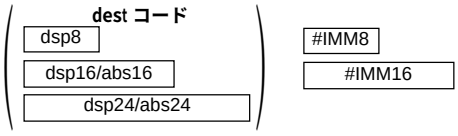
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*1 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

ADC

(1) ADC.size #IMM, dest





.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

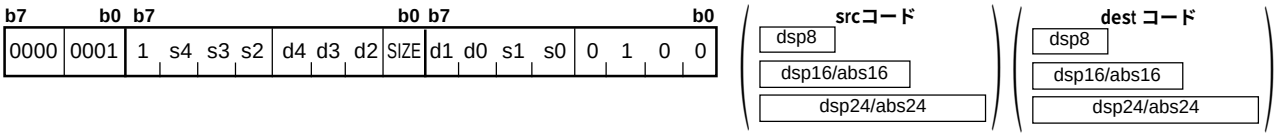
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/1	4/1	4/3	5/3	5/3	6/3	6/3	7/3	6/3	7/3

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

ADC

(2) ADC.size src, dest



.size	SIZE
.B	0
.W	1

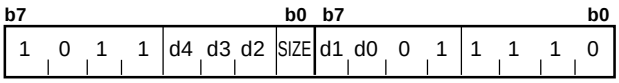
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

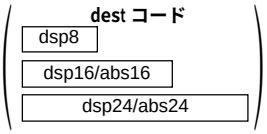
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
An	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:8[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:16[SB/FB]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:24[An]	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4
abs16	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs24	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4

ADCF

(1) ADCF.size dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

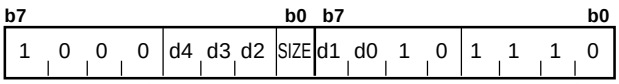
【バイト数／サイクル数】

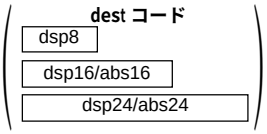
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

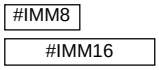
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

ADD

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







*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

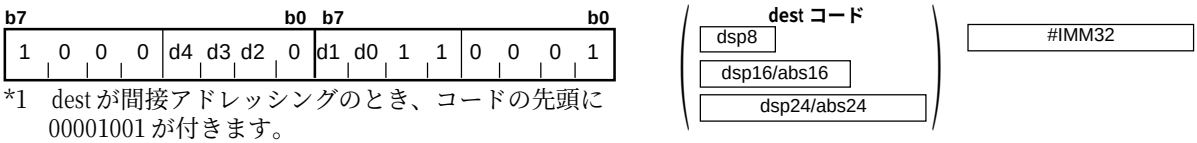
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

ADD

(2) ADD.L:G #IMM,dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

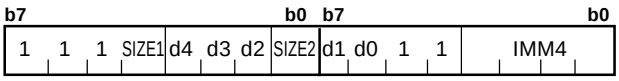
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	6/2	6/2	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

ADD

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



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE1	SIZE2
.B	0	0
.W	0	1
.L	1	0

#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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【バイト数／サイクル数】

・サイズ指定子(.size)に(.B)、(.W)を指定したとき

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

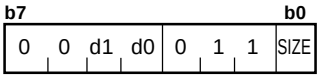
・サイズ指定子(.size)に(.L)を指定したとき

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

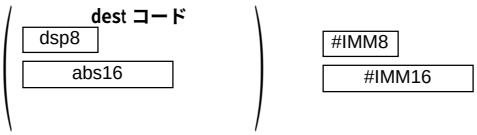
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

ADD

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



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

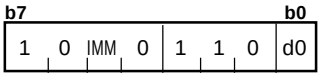
【バイト数／サイクル数】

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

*1 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*2 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

ADD

(5) ADD.L:S #IMM, A0/A1



#IMM	IMM
#1	0
#2	1

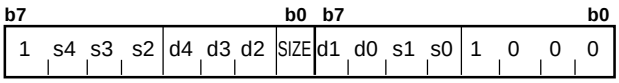
A0/A1	d0
A0	0
A1	1

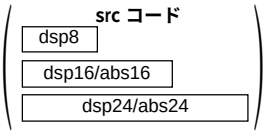
【バイト数／サイクル数】

バイト数/サイクル数	1/2

ADD

(6) ADD.size:G src, dest





*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

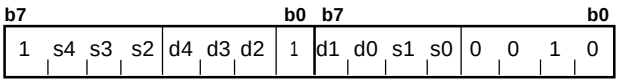
【バイト数／サイクル数】

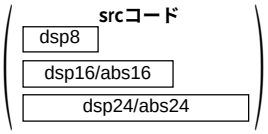
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
An	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:24[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
abs24	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4

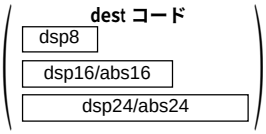
*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

ADD

(7) ADD.L:G src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

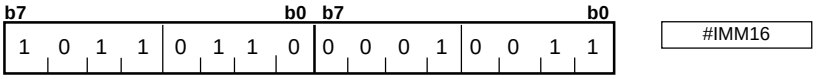
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
An	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
[An]	2/5	2/5	2/8	3/8	3/8	4/8	4/8	5/8	4/8	5/8
dsp:8[An]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:8[SB/FB]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:16[An]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:16[SB/FB]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:24[An]	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8
abs16	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
abs24	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

ADD

(8) ADD.L:G　　#IMM16, SP

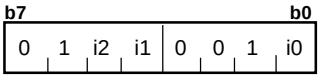


【バイト数／サイクル数】

バイト数/サイクル数	4/2

ADD

(9) ADD.L:Q　　#IMM3, SP

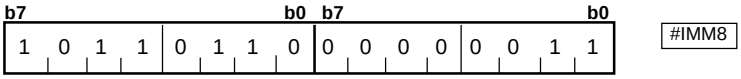


#IMM3	i2 i1 i0	#IMM3	i2 i1 i0
+1	0 0 0	+5	1 0 0
+2	0 0 1	+6	1 0 1
+3	0 1 0	+7	1 1 0
+4	0 1 1	+8	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	1/1

(10) ADD.L:S #IMM8, SP

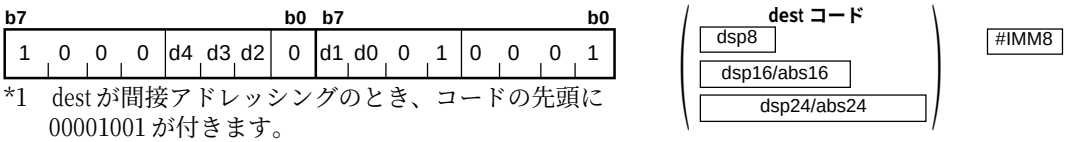


【バイト数／サイクル数】

バイト数/サイクル数	3/2

ADDX

(1) ADDX #IMM, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

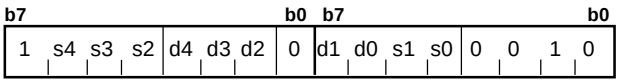
【バイト数／サイクル数】

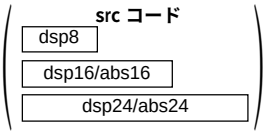
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5

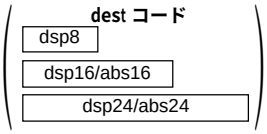
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

ADDX

(2) ADDX src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

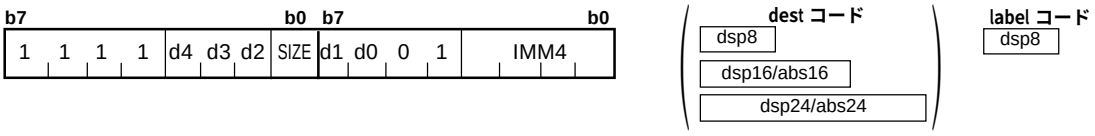
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
An	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
[An]	2/5	2/5	2/8	3/8	3/8	4/8	4/8	5/8	4/8	5/8
dsp:8[An]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:8[SB/FB]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:16[An]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:16[SB/FB]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:24[An]	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8
abs16	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
abs24	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

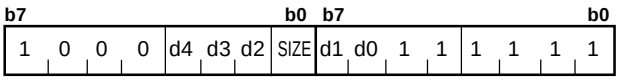
【バイト数／サイクル数】

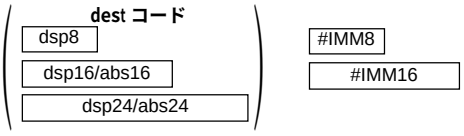
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

*1　label に分岐したとき、表中のサイクル数は +2 されます。

AND

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





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

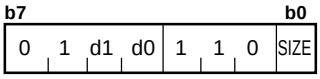
【バイト数／サイクル数】

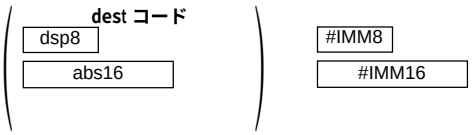
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

AND

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





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

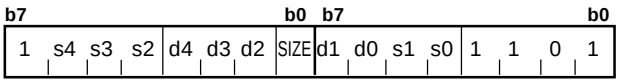
【バイト数／サイクル数】

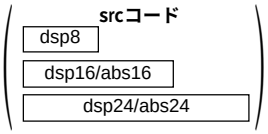
dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/3	4/3

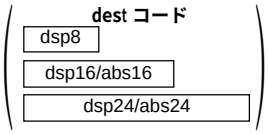
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

AND

(3) AND.size:G src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

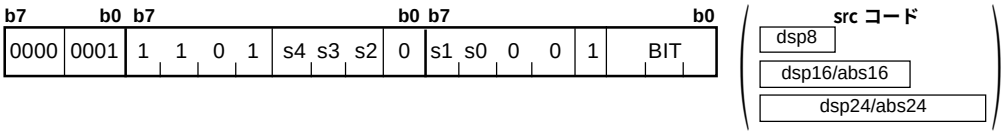
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
An	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:24[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
abs24	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

BAND

(1) BAND src



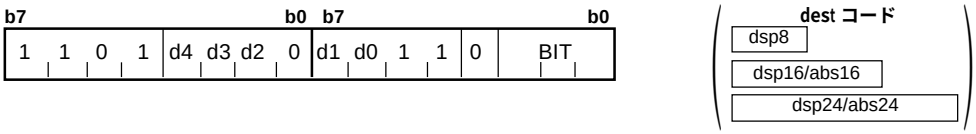
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11[An]	bit,base:11[SB/FB]	bit,base:19[An]	bit,base:19[SB/FB]	bit,base:27	bit,base:19	bit,base:27
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

BCLR

(1) BCLR dest



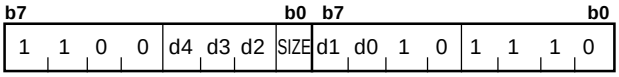
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:27[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

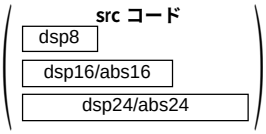
【バイト数／サイクル数】

dest	bit,Rn	bit,An	bit,[An]	bit,base:11[An]	bit,base:11[SB/FB]	bit,base:19[An]	bit,base:19[SB/FB]	bit,base:27[An]	bit,base:19	bit,base:27
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

BITINDEX

(1) BITINDEX.size　　src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

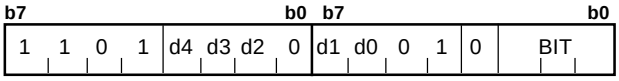
【バイト数／サイクル数】

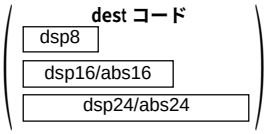
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/4	2/4	2/6	3/3	3/6	4/6	4/6	5/6	4/6	5/6

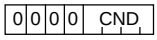
*1　次に実行する命令のサイクル数を +1 します。

BM*cnd*

(1) BM*cnd*　dest







dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

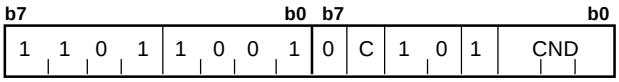
Cnd	CND	***Cnd***	CND
LTU/NC	0 0 0 0	GEU/C	1 0 0 0
LEU	0 0 0 1	GTU	1 0 0 1
NE/NZ	0 0 1 0	EQ/Z	1 0 1 0
PZ	0 0 1 1	N	1 0 1 1
NO	0 1 0 0	O	1 1 0 0
GT	0 1 0 1	LE	1 1 0 1
GE	0 1 1 0	LT	1 1 1 0

【バイト数／サイクル数】

dest	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

BM*cnd*

(2) BM*cnd*　C



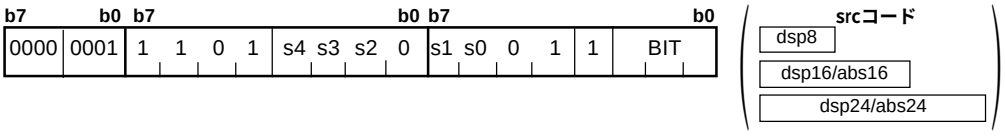
Cnd	C	CND	*Cnd*	C	CND
LTU/NC	0	0 0 0	GEU/C	1	0 0 0
LEU	0	0 0 1	GTU	1	0 0 1
NE/NZ	0	0 1 0	EQ/Z	1	0 1 0
PZ	0	0 1 1	N	1	0 1 1
NO	0	1 0 0	O	1	1 0 0
GT	0	1 0 1	LE	1	1 0 1
GE	0	1 1 0	LT	1	1 1 0

【バイト数／サイクル数】

バイト数/サイクル数	2/2

BNAND

(1) BNAND src



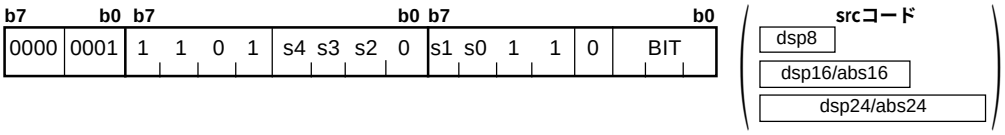
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

BNOR

(1) BNOR src



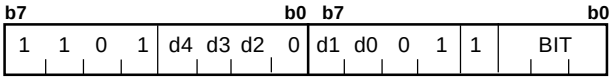
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

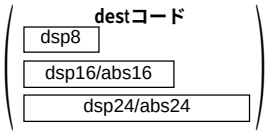
【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

BNOT

(1) BNOT　dest





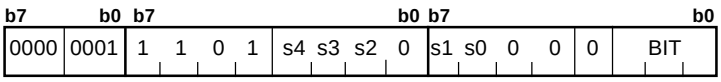
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

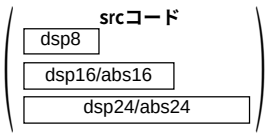
【バイト数／サイクル数】

dest	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

BNTST

(1) BNTST　src





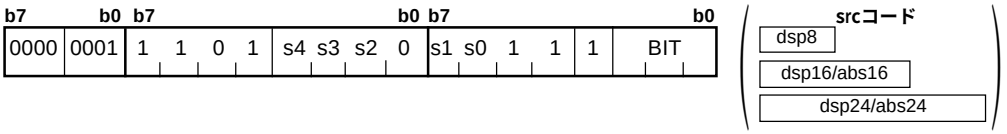
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

BNXOR

(1) BNXOR src



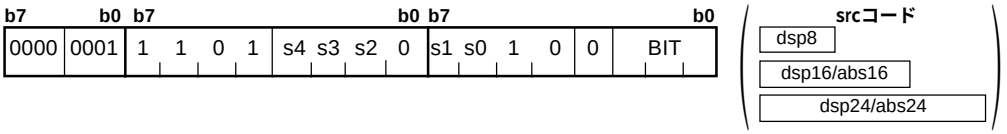
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

BOR

(1) BOR src



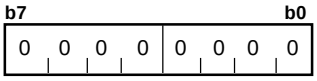
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

BRK

(1) BRK



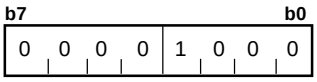
【バイト数／サイクル数】

バイト数/サイクル数	1/17

*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 されます。このとき、FFFFE7$_{16}$ 番地には FF$_{16}$ を設定してください。

BRK2

(1) BRK2

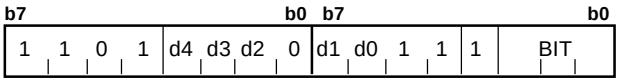


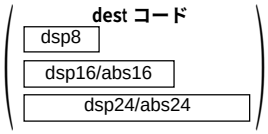
【バイト数／サイクル数】

バイト数/サイクル数	1/19

BSET

(1) BSET dest





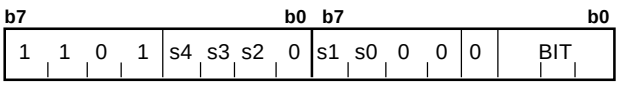
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

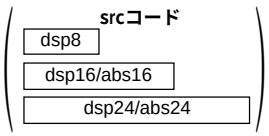
【バイト数／サイクル数】

dest	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

BTST

(1) BTST:G src





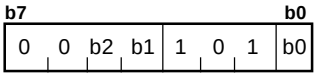
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

BTST

(2) BTST:S src



srcコード

abs16

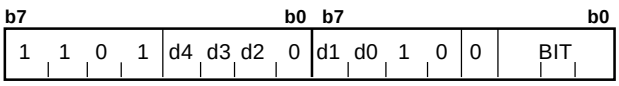
src	bit,base:19

【バイト数／サイクル数】

バイト数/サイクル数	3/3

BTSTC

(1) BTSTC dest



dest コード

dsp8
dsp16/abs16
dsp24/abs24

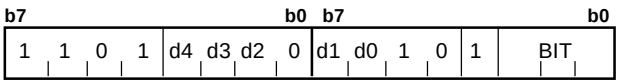
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

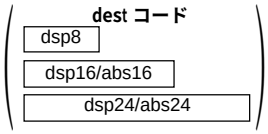
【バイト数／サイクル数】

dest	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

BTSTS

(1) BTSTS　dest





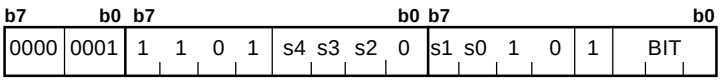
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

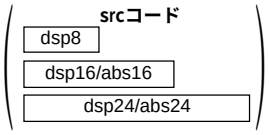
【バイト数／サイクル数】

dest	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

BXOR

(1) BXOR　　src





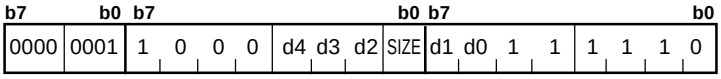
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	bit,R0L	1 0 0 1 0	bit,base:11[SB/FB]	bit,base:11[SB]	0 0 1 1 0
	bit,R0H	1 0 0 0 0		bit,base:11[FB]	0 0 1 1 1
	bit,R1L	1 0 0 1 1	bit,base:19[An]	bit,base:19[A0]	0 1 0 0 0
	bit,R1H	1 0 0 0 1		bit,base:19[A1]	0 1 0 0 1
An	bit,A0	0 0 0 1 0	bit,base:19[SB/FB]	bit,base:19[SB]	0 1 0 1 0
	bit,A1	0 0 0 1 1		bit,base:19[FB]	0 1 0 1 1
[An]	bit,[A0]	0 0 0 0 0	bit,base:27[An]	bit,base:27[A0]	0 1 1 0 0
	bit,[A1]	0 0 0 0 1		bit,base:27[A1]	0 1 1 0 1
bit,base:11[An]	bit,base:11[A0]	0 0 1 0 0	bit,base:19	bit,base:19	0 1 1 1 1
	bit,base:11[A1]	0 0 1 0 1	bit,base:27	bit,base:27	0 1 1 1 0

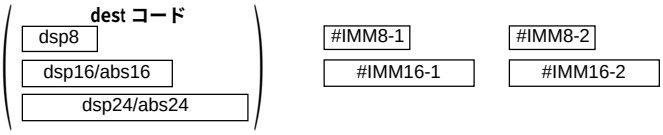
【バイト数／サイクル数】

src	bit,Rn	bit,An	bit,[An]	bit,base:11 [An]	bit,base:11 [SB/FB]	bit,base:19 [An]	bit,base:19 [SB/FB]	bit,base:27 [An]	bit,base:19	bit,base:27
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

CLIP

(1) CLIP.size　　#IMM1, #IMM2, dest





.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

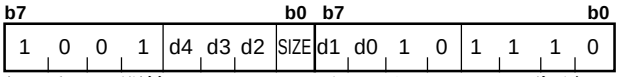
【バイト数／サイクル数】

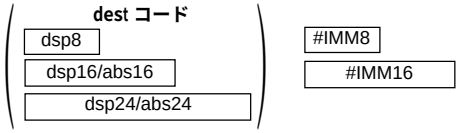
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	5/6	5/6	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は +2 されます。

CMP

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





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

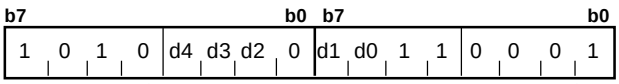
【バイト数／サイクル数】

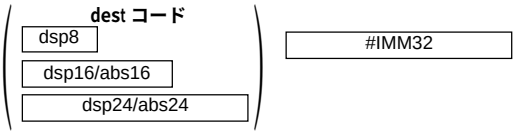
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

CMP

(2) CMP.L:G #IMM32, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

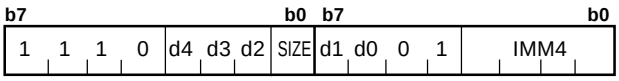
【バイト数／サイクル数】

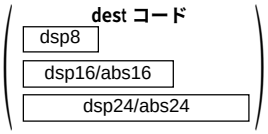
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	6/2	6/2	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

CMP

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





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

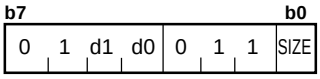
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

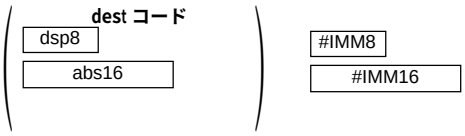
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

CMP

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



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

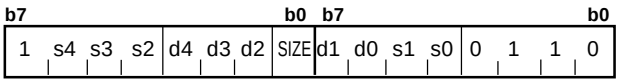
【バイト数／サイクル数】

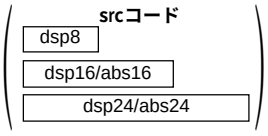
dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/3	4/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

CMP

(5) CMP.size:G src, dest





*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

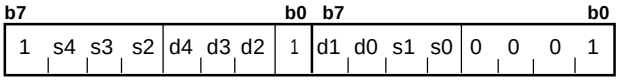
【バイト数／サイクル数】

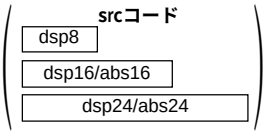
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
An	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:24[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
abs24	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4

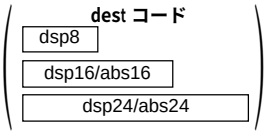
*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

CMP

(6) CMP.L:G src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

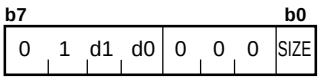
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
An	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
[An]	2/5	2/5	2/8	3/8	3/8	4/8	4/8	5/8	4/8	5/8
dsp:8[An]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:8[SB/FB]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:16[An]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:16[SB/FB]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:24[An]	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8
abs16	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
abs24	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8

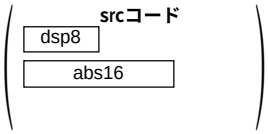
*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

CMP

(7) CMP.size:S　　src, R0/R0L



*1　src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

src		d1	d0
dsp:8[SB/FB]	dsp:8[SB]	1	0
	dsp:8[FB]	1	1
abs16	abs16	0	1

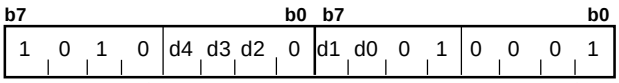
【バイト数／サイクル数】

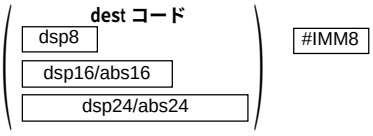
src	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/3	3/3

*2　src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

CMPX

(1) CMPX #IMM, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	--- / --- /R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	--- / --- /R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	--- / --- /-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	--- / --- /-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

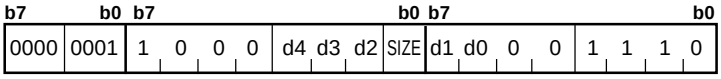
【バイト数／サイクル数】

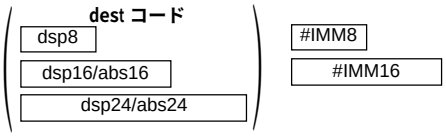
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

DADC

(1) DADC.size #IMM, dest





.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

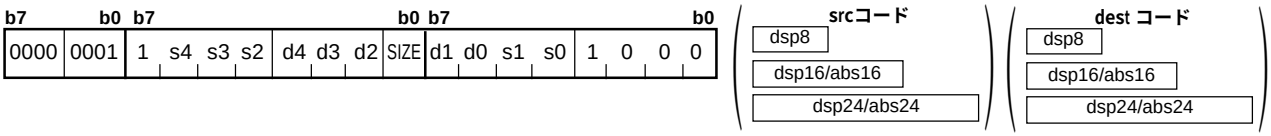
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/4	4/4	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

DADC

(2) DADC.size src, dest



.size	SIZE
.B	0
.W	1

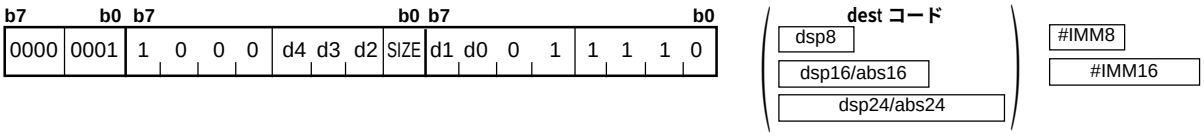
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/4	3/4	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
An	3/4	3/4	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
[An]	3/6	3/6	3/7	4/7	4/7	5/7	5/7	6/7	5/7	6/7
dsp:8[An]	4/6	4/6	4/7	5/7	5/7	6/7	6/7	7/7	6/7	7/7
dsp:8[SB/FB]	4/6	4/6	4/7	5/7	5/7	6/7	6/7	7/7	6/7	7/7
dsp:16[An]	5/6	5/6	5/7	6/7	6/7	7/7	7/7	8/7	7/7	8/7
dsp:16[SB/FB]	5/6	5/6	5/7	6/7	6/7	7/7	7/7	8/7	7/7	8/7
dsp:24[An]	6/6	6/6	6/7	7/7	7/7	8/7	8/7	9/7	8/7	9/7
abs16	5/6	5/6	5/7	6/7	6/7	7/7	7/7	8/7	7/7	8/7
abs24	6/6	6/6	6/7	7/7	7/7	8/7	8/7	9/7	8/7	9/7

DADD

(1) DADD.size　　#IMM, dest



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

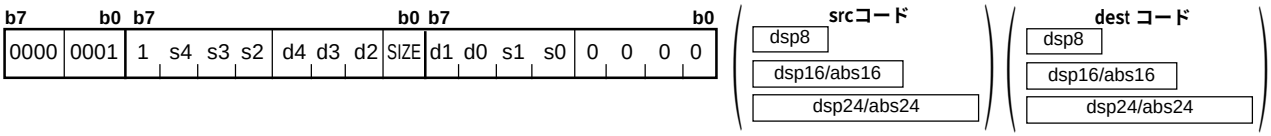
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/4	4/4	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

DADD

(2) DADD.size　　src, dest



.size	SIZE
.B	0
.W	1

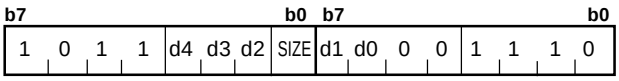
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/4	3/4	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
An	3/4	3/4	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
[An]	3/6	3/6	3/7	4/7	4/7	5/7	5/7	6/7	5/7	6/7
dsp:8[An]	4/6	4/6	4/7	5/7	5/7	6/7	6/7	7/7	6/7	7/7
dsp:8[SB/FB]	4/6	4/6	4/7	5/7	5/7	6/7	6/7	7/7	6/7	7/7
dsp:16[An]	5/6	5/6	5/7	6/7	6/7	7/7	7/7	8/7	7/7	8/7
dsp:16[SB/FB]	5/6	5/6	5/7	6/7	6/7	7/7	7/7	8/7	7/7	8/7
dsp:24[An]	6/6	6/6	6/7	7/7	7/7	8/7	8/7	9/7	8/7	9/7
abs16	5/6	5/6	5/7	6/7	6/7	7/7	7/7	8/7	7/7	8/7
abs24	6/6	6/6	6/7	7/7	7/7	8/7	8/7	9/7	8/7	9/7

DEC

(1) DEC.size dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

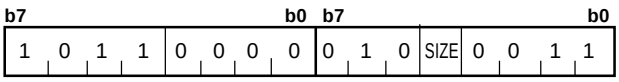
【バイト数／サイクル数】

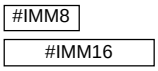
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

DIV

(1) DIV.size #IMM





.size	SIZE
.B	0
.W	1

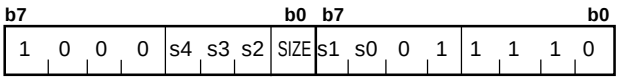
【バイト数／サイクル数】

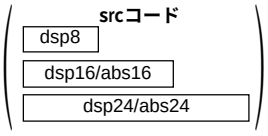
バイト数/サイクル数	3/18

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1、サイクル数は +6 されます。

DIV

(2) DIV.size src





*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

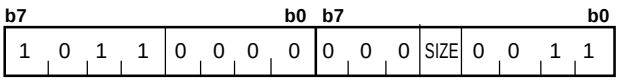
【バイト数／サイクル数】

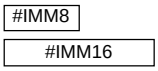
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/18	2/18	2/20	3/20	3/20	4/20	4/20	5/20	4/20	5/20

*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +6 されます。

DIVU

(1) DIVU.size #IMM





.size	SIZE
.B	0
.W	1

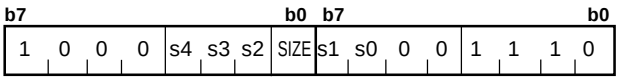
【バイト数／サイクル数】

バイト数/サイクル数	3/18

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1、サイクル数は +5 されます。

DIVU

(2) DIVU.size src



*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

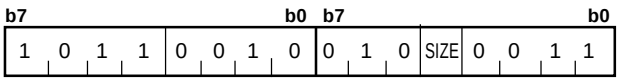
【バイト数／サイクル数】

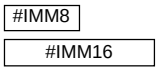
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/18	2/18	2/20	3/20	3/20	4/20	4/20	5/20	4/20	5/20

*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +5 されます。

DIVX

(1) DIVX.size #IMM





.size	SIZE
.B	0
.W	1

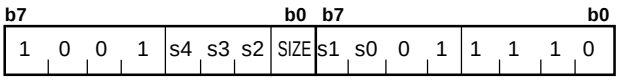
【バイト数／サイクル数】

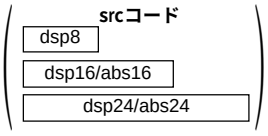
バイト数/サイクル数	3/18

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1、サイクル数は +6 されます。

DIVX

(2) DIVX.size src





*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

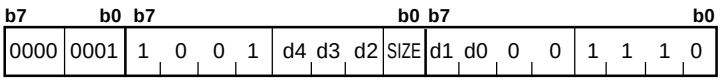
【バイト数／サイクル数】

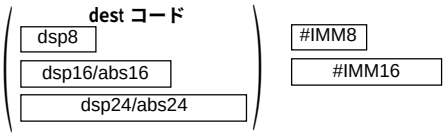
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/18	2/18	2/20	3/20	3/20	4/20	4/20	5/20	4/20	5/20

*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +6 されます。

DSBB

(1) DSBB.size #IMM, dest





.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

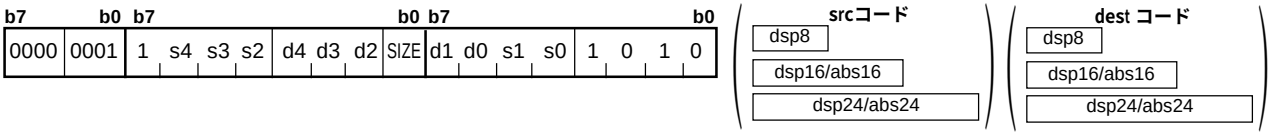
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/2	4/2	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

DSBB

(2) DSBB.size src, dest



.size	SIZE
.B	0
.W	1

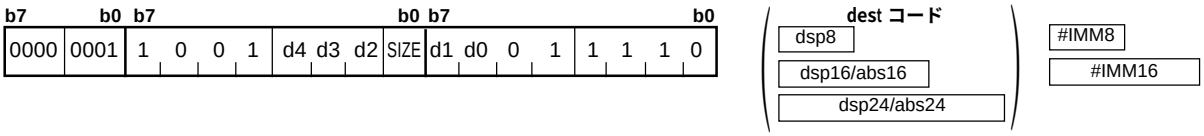
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
An	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
[An]	3/4	3/4	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5
dsp:8[An]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:8[SB/FB]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:16[An]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:16[SB/FB]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:24[An]	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5
abs16	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
abs24	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5

DSUB

(1) DSUB.size　　#IMM, dest



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

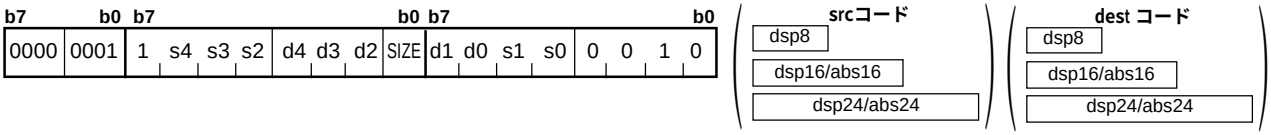
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/2	4/2	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

DSUB

(2) DSUB.size src, dest



.size	SIZE
.B	0
.W	1

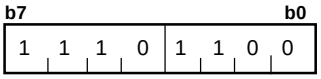
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
An	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
[An]	3/4	3/4	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5
dsp:8[An]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:8[SB/FB]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:16[An]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:16[SB/FB]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:24[An]	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5
abs16	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
abs24	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5

ENTER

(1) ENTER　#IMM



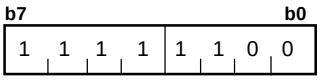
#IMM8

【バイト数／サイクル数】

バイト数/サイクル数	2/4

EXITD

(1) EXITD

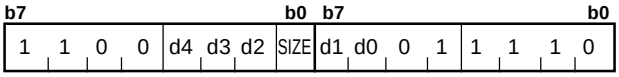


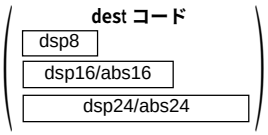
【バイト数／サイクル数】

バイト数/サイクル数	1/8

EXTS

(1) EXTS.size　　dest





.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

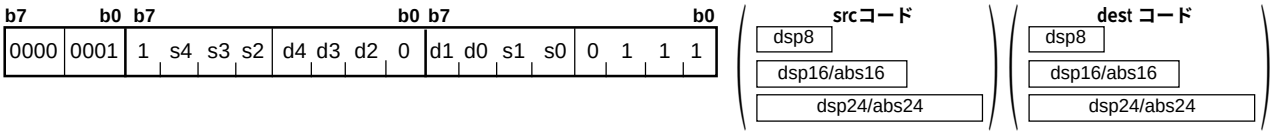
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*1　サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +1 されます。

EXTS

(2) EXTS.B src,dest



src		s4	s3	s2	s1	s0	src		s4	s3	s2	s1	s0
Rn	R0L/---/---	1	0	0	1	0	dsp:8[SB/FB]	dsp:8[SB]	0	0	1	1	0
	R1L/---/---	1	0	0	1	1		dsp:8[FB]	0	0	1	1	1
	R0H/---/-	1	0	0	0	0	dsp:16[An]	dsp:16[A0]	0	1	0	0	0
	R1H/---/-	1	0	0	0	1		dsp:16[A1]	0	1	0	0	1
An	---	0	0	0	1	0	dsp:16[SB/FB]	dsp:16[SB]	0	1	0	1	0
	---	0	0	0	1	1		dsp:16[FB]	0	1	0	1	1
[An]	[A0]	0	0	0	0	0	dsp:24[An]	dsp:24[A0]	0	1	1	0	0
	[A1]	0	0	0	0	1		dsp:24[A1]	0	1	1	0	1
dsp:8[An]	dsp:8[A0]	0	0	1	0	0	abs16	abs16	0	1	1	1	1
	dsp:8[A1]	0	0	1	0	1	abs24	abs24	0	1	1	1	0

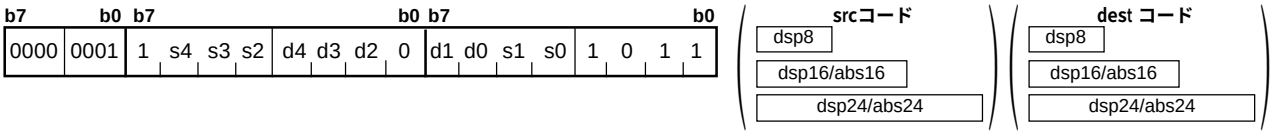
dest		d4	d3	d2	d1	d0	dest		d4	d3	d2	d1	d0
Rn	---/R0/---	1	0	0	1	0	dsp:8[SB/FB]	dsp:8[SB]	0	0	1	1	0
	---/R1/---	1	0	0	1	1		dsp:8[FB]	0	0	1	1	1
	---/R2/-	1	0	0	0	0	dsp:16[An]	dsp:16[A0]	0	1	0	0	0
	---/R3/-	1	0	0	0	1		dsp:16[A1]	0	1	0	0	1
An	A0	0	0	0	1	0	dsp:16[SB/FB]	dsp:16[SB]	0	1	0	1	0
	A1	0	0	0	1	1		dsp:16[FB]	0	1	0	1	1
[An]	[A0]	0	0	0	0	0	dsp:24[An]	dsp:24[A0]	0	1	1	0	0
	[A1]	0	0	0	0	1		dsp:24[A1]	0	1	1	0	1
dsp:8[An]	dsp:8[A0]	0	0	1	0	0	abs16	abs16	0	1	1	1	1
	dsp:8[A1]	0	0	1	0	1	abs24	abs24	0	1	1	1	0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:8[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:16[SB/FB]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:24[An]	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4
abs16	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs24	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4

EXTZ

(1) EXTZ src,dest



src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1s0
Rn	R0L/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	---	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	---	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

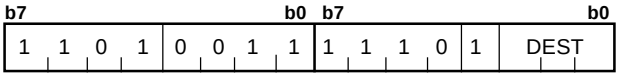
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:8[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:16[SB/FB]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:24[An]	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4
abs16	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs24	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4

FCLR

(1) FCLR　　dest



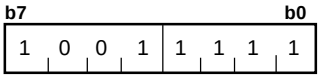
dest	DEST
C	0 0 0
D	0 0 1
Z	0 1 0
S	0 1 1
B	1 0 0
O	1 0 1
I	1 1 0
U	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	2/1

FREIT

(1) FREIT

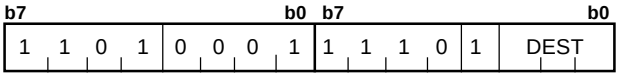


【バイト数／サイクル数】

バイト数/サイクル数	1/3

FSET

(1) FSET　　dest



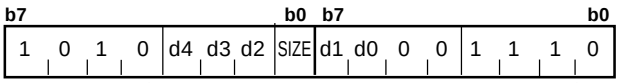
dest	DEST
C	0 0 0
D	0 0 1
Z	0 1 0
S	0 1 1
B	1 0 0
O	1 0 1
I	1 1 0
U	1 1 1

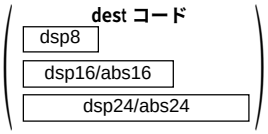
【バイト数／サイクル数】

バイト数/サイクル数	2/1

INC

(1) INC.size dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

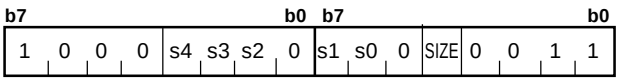
【バイト数／サイクル数】

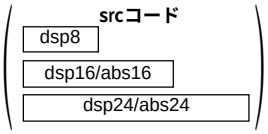
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

INDEXB

(1) INDEXB.size src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

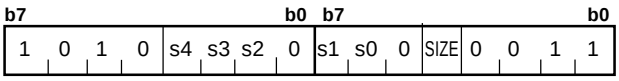
【バイト数／サイクル数】

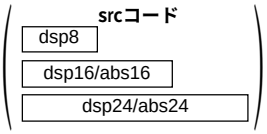
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +2 されます。

INDEXBD

(1) INDEXBD.size src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

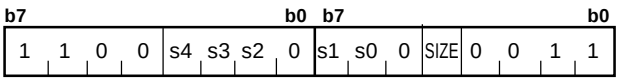
【バイト数／サイクル数】

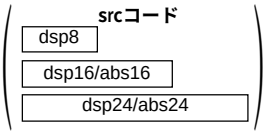
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +1 されます。

INDEXBS

(1) INDEXBS.size src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

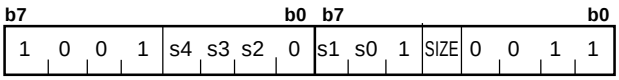
【バイト数／サイクル数】

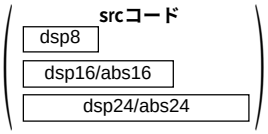
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +1 されます。

INDEXL

(1) INDEXL.size src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

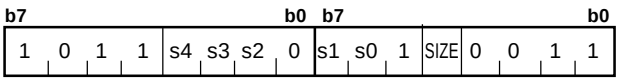
【バイト数／サイクル数】

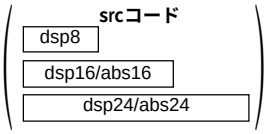
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/4	2/4	2/6	3/6	3/6	4/6	4/6	5/6	4/6	5/6

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +2 されます。

INDEXLD

(1) INDEXLD.size src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

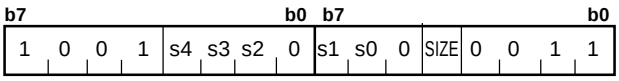
【バイト数／サイクル数】

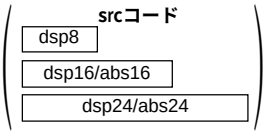
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +1 されます。

INDEXLS

(1) INDEXLS.size src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

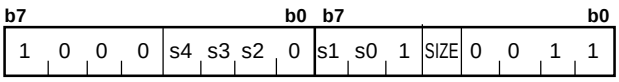
【バイト数／サイクル数】

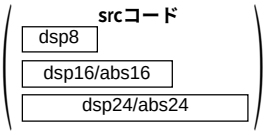
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +1 されます。

INDEXW

(1) INDEXW.size src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

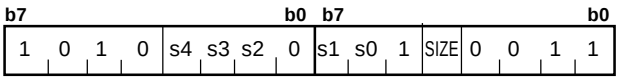
【バイト数／サイクル数】

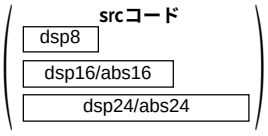
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2	2/2	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

*1 サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +2 されます。

INDEXWD

(1) INDEXWD.size　　src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

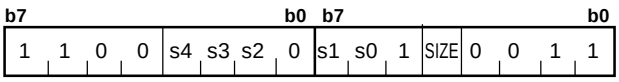
【バイト数／サイクル数】

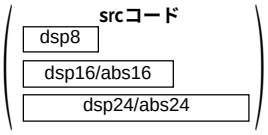
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*1　サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +1 されます。

INDEXWS

(1) INDEXWS.size　　src





.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

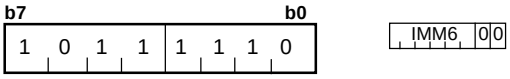
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*1　サイズ指定子(.size)が(.W)のとき、表中のサイクル数は +1 されます。

INT

(1) INT　　#IMM

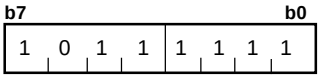


【バイト数／サイクル数】

バイト数/サイクル数	2/ 12

INTO

(1) INTO



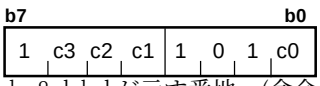
【バイト数／サイクル数】

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 +13 されます。

J*cnd*

(1) J*cnd*　　label



dsp8=label が示す番地－(命令の先頭番地＋ 1)

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

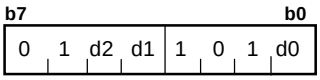
【バイト数／サイクル数】

バイト数/サイクル数	2/1

*1　label に分岐したとき、表中のサイクル数は +2 されます。

JMP

(1) JMP.S　　label



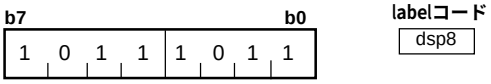
label	d2 d1 d0	label	d2 d1 d0
PC + 2	0 0 0	PC + 6	1 0 0
PC + 3	0 0 1	PC + 7	1 0 1
PC + 4	0 1 0	PC + 8	1 1 0
PC + 5	0 1 1	PC + 9	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	1/3

JMP

(2) JMP.B　label



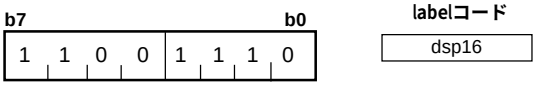
dsp8=label が示す番地－(命令の先頭番地＋ 1)

【バイト数／サイクル数】

バイト数/サイクル数	2/3

JMP

(3) JMP.W　label



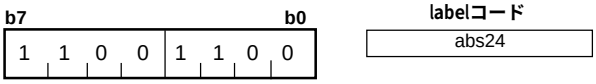
dsp16=label が示す番地－(命令の先頭番地＋ 1)

【バイト数／サイクル数】

バイト数/サイクル数	3/3

JMP

(4) JMP.A　label

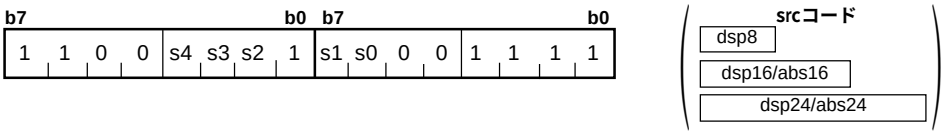


【バイト数／サイクル数】

バイト数/サイクル数	4/3

JMPI

(1) JMPI.W　src



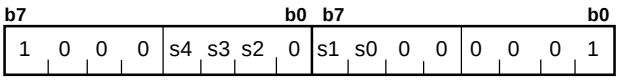
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

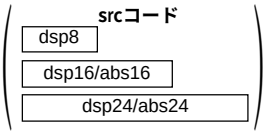
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/7	2/7	2/8	3/8	3/8	4/8	4/8	5/8	4/8	5/8

JMPI

(2) JMPI.A　src





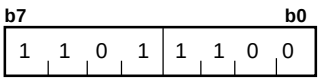
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/5	2/5	2/7	3/7	3/7	4/7	4/7	5/7	4/7	5/7

JMPS

(1) JMPS　#IMM8

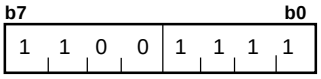


【バイト数／サイクル数】

バイト数/サイクル数	2/8

JSR

(1) JSR.W　label



labelコード
dsp16

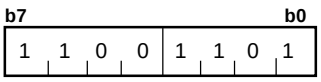
dsp16=label が示す番地－(命令の先頭番地＋ 1)

【バイト数／サイクル数】

バイト数/サイクル数	3/3

JSR

(2) JSR.A　label



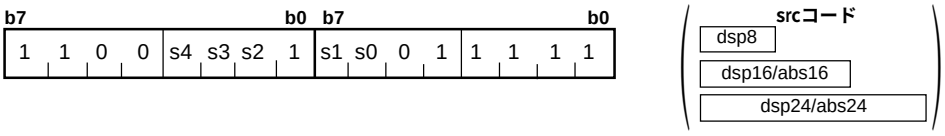
labelコード
abs24

【バイト数／サイクル数】

バイト数/サイクル数	4/3

JSRI

(1) JSRI.W　src



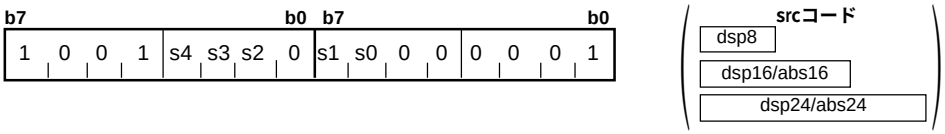
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/7	2/7	2/8	3/8	3/8	4/8	4/8	5/8	4/8	5/8

JSRI

(2) JSRI.A　src



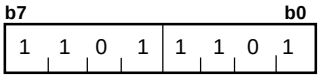
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/5	2/5	2/7	3/7	3/7	4/7	4/7	5/7	4/7	5/7

JSRS

(1) JSRS　　#IMM8

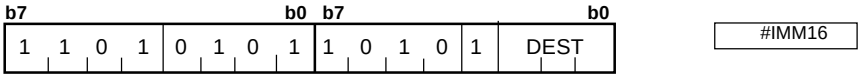


【バイト数／サイクル数】

バイト数/サイクル数	2/8

LDC

(1) LDC　　#IMM16, dest



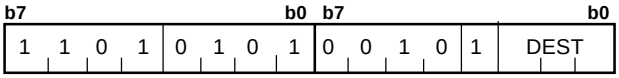
dest	DEST
DCT0	0 0 0
DCT1	0 0 1
FLG	0 1 0
SVF	0 1 1
DRC0	1 0 0
DRC1	1 0 1
DMD0	1 1 0
DMD1	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	4/1

LDC

(2) LDC　　#IMM24, dest



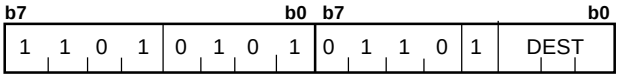
dest	DEST
INTB	0 0 0
SP	0 0 1
SB	0 1 0
FB	0 1 1
SVP	1 0 0
VCT	1 0 1
---	1 1 0
ISP	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	5/2

LDC

(3) LDC　　#IMM24, dest



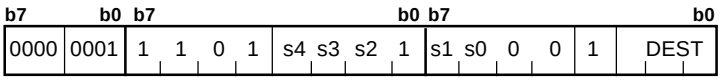
dest	DEST
---	0 0 0
---	0 0 1
DMA0	0 1 0
DMA1	0 1 1
DRA0	1 0 0
DRA1	1 0 1
DSA0	1 1 0
DSA1	1 1 1

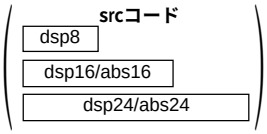
【バイト数／サイクル数】

バイト数/サイクル数	5/2

LDC

(4) LDC src, dest





src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

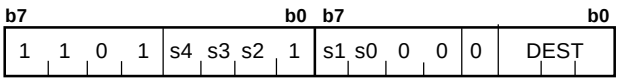
dest	DEST
DCT0	0 0 0
DCT1	0 0 1
FLG	0 1 0
SVF	0 1 1
DRC0	1 0 0
DRC1	1 0 1
DMD0	1 1 0
DMD1	1 1 1

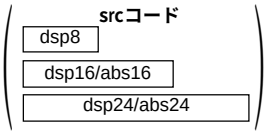
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

LDC

(5) LDC src, dest





src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

dest	DEST
INTB	0 0 0
SP	0 0 1
SB	0 1 0
FB	0 1 1
SVP	1 0 0
VCT	1 0 1
---	1 1 0
ISP	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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2	2/2	2/6	3/6	3/6	4/6	4/6	5/6	4/6	5/6

LDC

(6) LDC　　src, dest

b7	b0	b7				b0	b7				b0
0000	0001	1	1	0	1	s4 s3 s2 1	s1	s0	0	0	0 DEST

srcコード
dsp8
dsp16/abs16
dsp24/abs24

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

dest	DEST
---	0 0 0
---	0 0 1
DMA0	0 1 0
DMA1	0 1 1
DRA0	1 0 0
DRA1	1 0 1
DSA0	1 1 0
DSA1	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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3	3/3	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6

LDCTX

(1) LDCTX　　abs16,abs24

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

abs16　　abs24

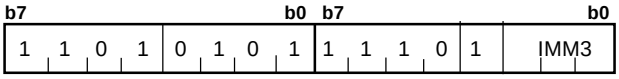
【バイト数／サイクル数】

バイト数/サイクル数	7/10 + m

*1　m は転送回数です。 m=(R0、R1、R2、R3 の数)＋2×(A0、A1、FB、SB の数)

LDIPL

(1) LDIPL #IMM

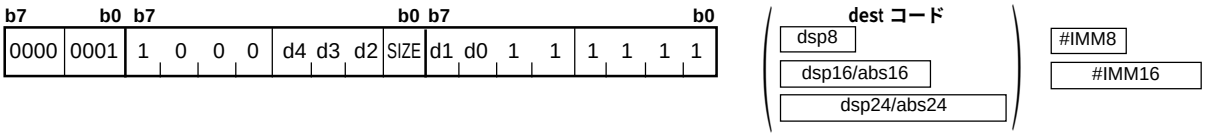


【バイト数／サイクル数】

バイト数/サイクル数	2/2

MAX

(1) MAX.size #IMM,dest



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

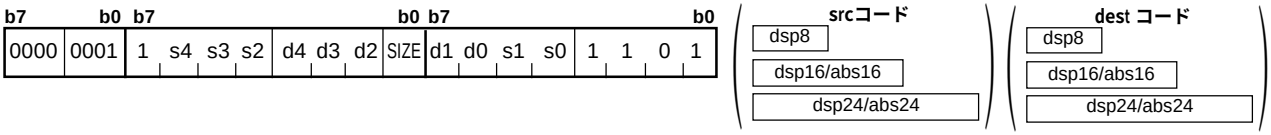
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/3	4/3	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

MAX

(2) MAX.size src, dest



.size	SIZE
.B	0
.W	1

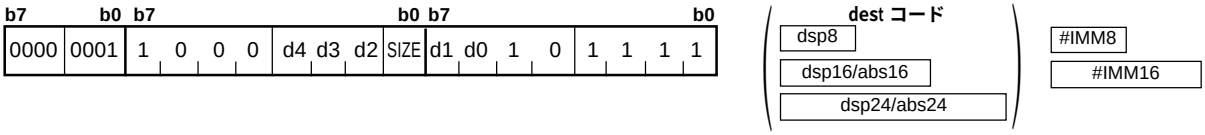
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
An	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
[An]	3/4	3/4	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5
dsp:8[An]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:8[SB/FB]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:16[An]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:16[SB/FB]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:24[An]	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5
abs16	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
abs24	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5

MIN

(1) MIN.size　　#IMM,dest



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

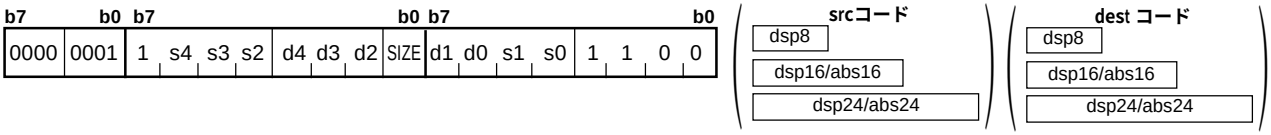
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/3	4/3	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

MIN

(2) MIN.size src, dest



.size	SIZE
.B	0
.W	1

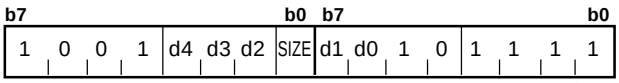
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

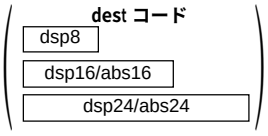
【バイト数／サイクル数】

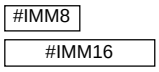
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
An	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
[An]	3/4	3/4	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5
dsp:8[An]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:8[SB/FB]	4/4	4/4	4/5	5/5	5/5	6/5	6/5	7/5	6/5	7/5
dsp:16[An]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:16[SB/FB]	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
dsp:24[An]	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5
abs16	5/4	5/4	5/5	6/5	6/5	7/5	7/5	8/5	7/5	8/5
abs24	6/4	6/4	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5

MOV

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







*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

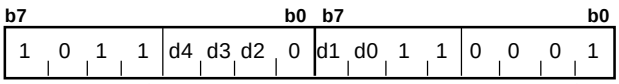
【バイト数／サイクル数】

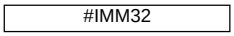
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

MOV

(2) MOV.L:G #IMM,dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

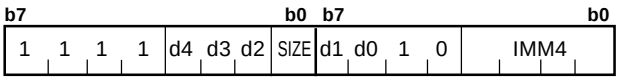
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	6/2	6/2	6/2	7/2	7/2	8/2	8/2	9/2	8/2	9/2

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MOV

(3) MOV.size:Q #IMM4, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

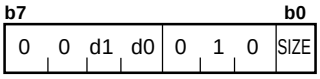
【バイト数／サイクル数】

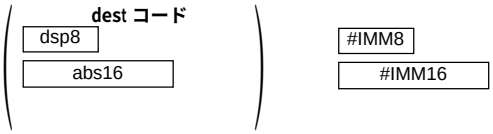
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/1	3/1	3/1	4/1	4/1	5/1	4/1	5/1

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MOV

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





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

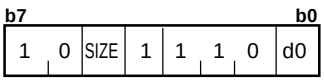
【バイト数／サイクル数】

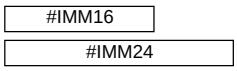
dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/2	4/2

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

MOV

(5) MOV.size:S #IMM, A0/A1





.size	SIZE
.W	0
.L	1

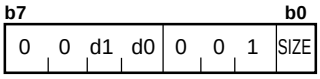
A0/A1	d0
A0	0
A1	1

【バイト数／サイクル数】

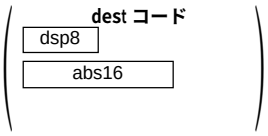
#IMM	An
#IMM16	3/1
#IMM24	4/2

MOV

(6) MOV.size:Z #0, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

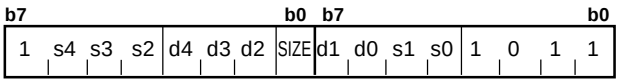
【バイト数／サイクル数】

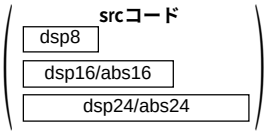
dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	1/1	2/1	3/1

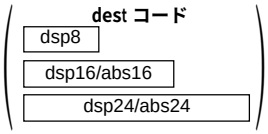
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MOV

(7) MOV.size:G src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

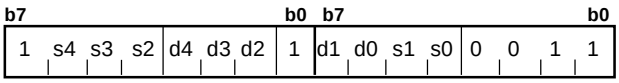
【バイト数／サイクル数】

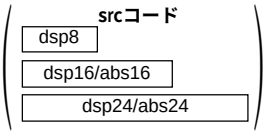
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/1	2/1	2/1	3/1	3/1	4/1	4/1	5/1	4/1	5/1
An	2/1	2/1	2/1	3/1	3/1	4/1	4/1	5/1	4/1	5/1
[An]	2/3	2/3	2/3	3/2	3/2	4/2	4/2	5/2	4/2	5/2
dsp:8[An]	3/3	3/3	3/3	4/2	4/2	5/2	5/2	6/2	5/2	6/2
dsp:8[SB/FB]	3/3	3/3	3/3	4/2	4/2	5/2	5/2	6/2	5/2	6/2
dsp:16[An]	4/3	4/3	4/3	5/2	5/2	6/2	6/2	7/2	6/2	7/2
dsp:16[SB/FB]	4/3	4/3	4/3	5/2	5/2	6/2	6/2	7/2	6/2	7/2
dsp:24[An]	5/3	5/3	5/3	6/2	6/2	7/2	7/2	8/2	7/2	8/2
abs16	4/3	4/3	4/3	5/2	5/2	6/2	6/2	7/2	6/2	7/2
abs24	5/3	5/3	5/3	6/2	6/2	7/2	7/2	8/2	7/2	8/2

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

MOV

(8) MOV.L:G src, dest





*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

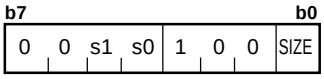
【バイト数／サイクル数】

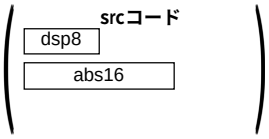
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/2	2/2	2/2	3/2	3/2	4/2	4/2	5/2	4/2	5/2
An	2/2	2/2	2/2	3/2	3/2	4/2	4/2	5/2	4/2	5/2
[An]	2/4	2/4	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4
dsp:8[An]	3/4	3/4	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[SB/FB]	3/4	3/4	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:16[An]	4/4	4/4	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[SB/FB]	4/4	4/4	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:24[An]	5/4	5/4	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs16	4/4	4/4	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
abs24	5/4	5/4	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

MOV

(9) MOV.size:S src, R0L/R0





*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

src		s1 s0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

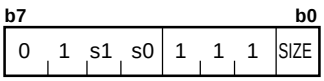
【バイト数／サイクル数】

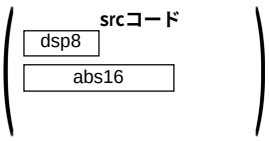
src	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/2	3/2

*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MOV

(10) MOV.size:S src, R1L/R1





*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

src		s1 s0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

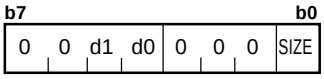
【バイト数／サイクル数】

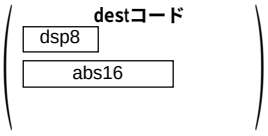
src	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	1/2	2/2	3/2

*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MOV

(11) MOV.size:S R0L/R0, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d1 d0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

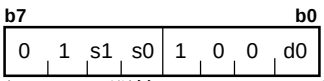
【バイト数／サイクル数】

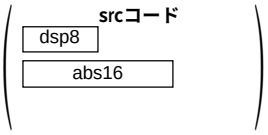
dest	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/1

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MOV

(12) MOV.L:S src, A0/A1





*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

src		s1 s0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

A0/A1	d0
A0	0
A1	1

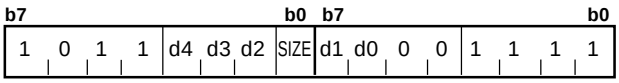
【バイト数／サイクル数】

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

*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MOV

(13) MOV.size:G　　dsp:8[SP], dest



srcコード: dsp8

dest コード: dsp8 / dsp16/abs16 / dsp24/abs24

.size	SIZE
.B	0
.W	1

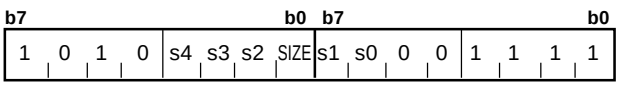
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3	3/3	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

MOV

(14) MOV.size:G　　src, dsp:8[SP]



srcコード: dsp8 / dsp16/abs16 / dsp24/abs24

dest コード: dsp8

.size	SIZE
.B	0
.W	1

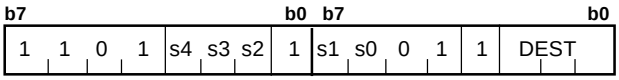
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

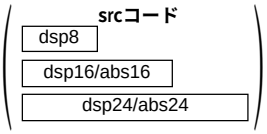
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3	3/3	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

MOVA

(1) MOVA src, dest





dest	DEST
R2R0	0 0 0
R3R1	0 0 1
A0	0 1 0
A1	0 1 1

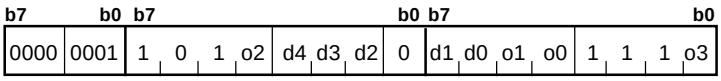
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	dsp:8[A1]	0 0 1 0 1		dsp:16[FB]	0 1 0 1 1
dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	dsp:8[FB]	0 0 1 1 1		dsp:24[A1]	0 1 1 0 1
dsp:16[An]	dsp:16[A0]	0 1 0 0 0	abs16	abs16	0 1 1 1 1
	dsp:16[A1]	0 1 0 0 1	abs24	abs24	0 1 1 1 0

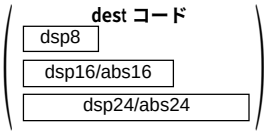
【バイト数／サイクル数】

src	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	4/2	4/2	5/2	4/2	5/2

MOV*Dir*

(1) MOV*Dir* R0L, dest





Dir	o3 o2 o1 o0
LL	0 1 0 0
HL	0 1 0 1
LH	0 1 1 0
HH	0 1 1 1

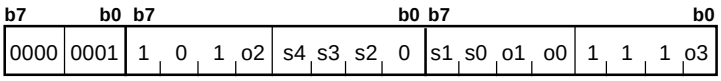
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	---	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	---	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

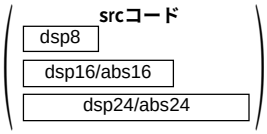
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
MOVHH, MOVLL	3/3	3/3	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5
MOVHL, MOVLH	3/6	3/6	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8

MOV*Dir*

(2) MOV*Dir* src, R0L





Dir	o3 o2 o1 o0
LL	0 0 0 0
HL	0 0 0 1
LH	0 0 1 0
HH	0 0 1 1

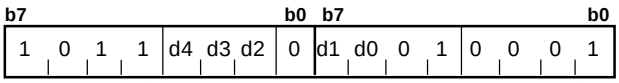
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	---	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	---	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

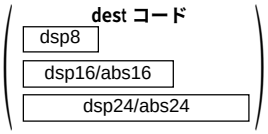
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
MOVHH, MOVLL	3/3	3/3	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5
MOVHL, MOVLH	3/6	3/6	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8

MOVX

(1) MOVX #IMM, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

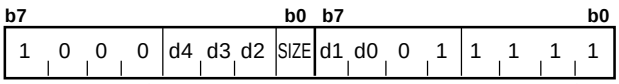
【バイト数／サイクル数】

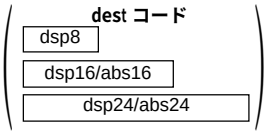
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/2	4/2	4/2	5/2	5/2	6/2	5/2	6/2

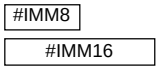
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MUL

(1) MUL.size #IMM, dest







*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

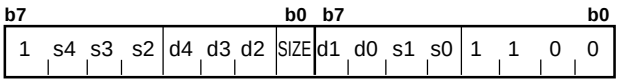
【バイト数／サイクル数】

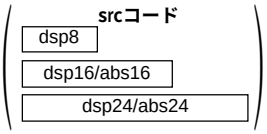
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3	3/3	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5

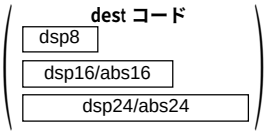
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

MUL

(2) MUL.size src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

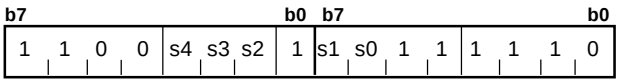
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/3	2/3	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
An	2/3	2/3	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
[An]	2/5	2/5	2/6	3/6	3/6	4/6	4/6	5/6	4/6	5/6
dsp:8[An]	3/5	3/5	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
dsp:8[SB/FB]	3/5	3/5	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
dsp:16[An]	4/5	4/5	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6
dsp:16[SB/FB]	4/5	4/5	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6
dsp:24[An]	5/5	5/5	5/6	6/6	6/6	7/6	7/6	8/6	7/6	8/6
abs16	4/5	4/5	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6
abs24	5/5	5/5	5/6	6/6	6/6	7/6	7/6	8/6	7/6	8/6

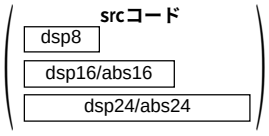
*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

MULEX

(1) MULEX src



*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

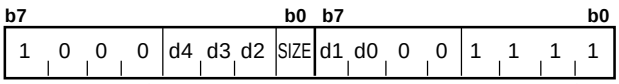
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/8	2/8	2/10	3/10	3/10	4/10	4/10	5/10	4/10	5/10

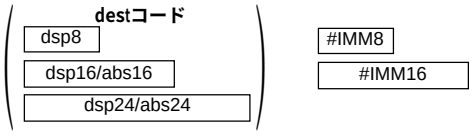
*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

MULU

(1) MULU.size #IMM,dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

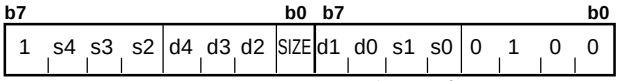
【バイト数／サイクル数】

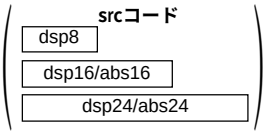
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3	3/3	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5

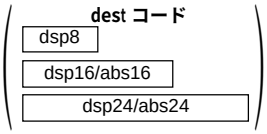
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

MULU

(2) MULU.size src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

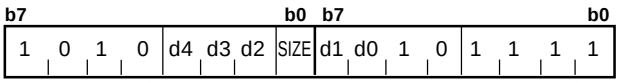
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/3	2/3	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
An	2/3	2/3	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
[An]	2/5	2/5	2/6	3/6	3/6	4/6	4/6	5/6	4/6	5/6
dsp:8[An]	3/5	3/5	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
dsp:8[SB/FB]	3/5	3/5	3/6	4/6	4/6	5/6	5/6	6/6	5/6	6/6
dsp:16[An]	4/5	4/5	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6
dsp:16[SB/FB]	4/5	4/5	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6
dsp:24[An]	5/5	5/5	5/6	6/6	6/6	7/6	7/6	8/6	7/6	8/6
abs16	4/5	4/5	4/6	5/6	5/6	6/6	6/6	7/6	6/6	7/6
abs24	5/5	5/5	5/6	6/6	6/6	7/6	7/6	8/6	7/6	8/6

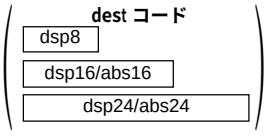
*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

NEG

(1) NEG.size　　dest



*1　dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

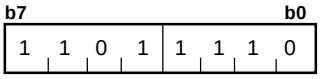
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2　dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

NOP

(1) NOP

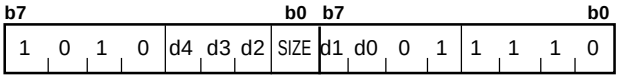


【バイト数／サイクル数】

バイト数/サイクル数	1/1

NOT

(1)NOT .size dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

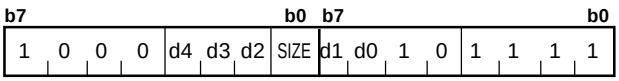
【バイト数／サイクル数】

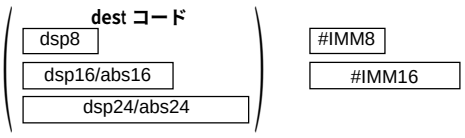
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

OR

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





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

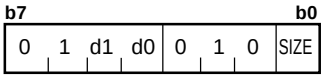
【バイト数／サイクル数】

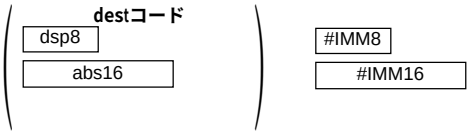
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

OR

(2) OR.size:S #IMM,dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

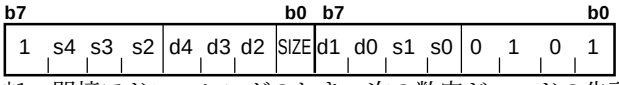
【バイト数／サイクル数】

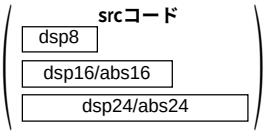
dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/3	4/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

OR

(3) OR.size:G src, dest





*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

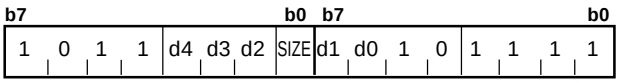
【バイト数／サイクル数】

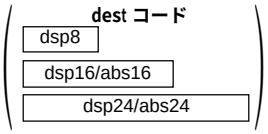
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
An	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:24[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
abs24	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

POP

(1) POP.size dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

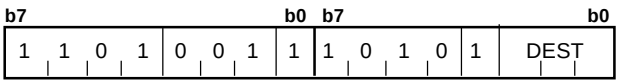
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/3	2/3	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

POPC

(1) POPC dest



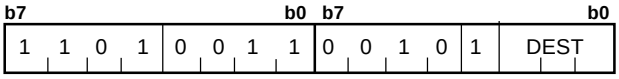
dest	DEST	dest	DEST
DCT0	0 0 0	DRC0	1 0 0
DCT1	0 0 1	DRC1	1 0 1
FLG	0 1 0	DMD0	1 1 0
SVF	0 1 1	DMD1	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	2/3

POPC

(2) POPC　　dest



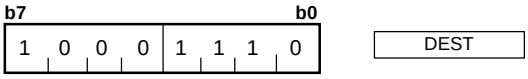
dest	DEST	dest	DEST
INTB	0 0 0	---	1 0 0
SP	0 0 1	---	1 0 1
SB	0 1 0	---	1 1 0
FB	0 1 1	ISP	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	2/4

POPM

(1)POPM　　dest



DEST

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

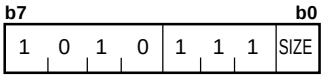
【バイト数／サイクル数】

バイト数/サイクル数	2/1+m

*2　mは復帰するレジスタ数です。 m=(R0、R1、R2、R3の数)＋2×(A0、A1、FB、SBの数)

PUSH

(1) PUSH.size #IMM



#IMM8
#IMM16

.size	SIZE
.B	0
.W	1

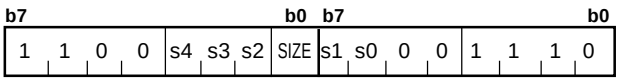
【バイト数／サイクル数】

バイト数/サイクル数	2/1

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

PUSH

(2) PUSH.size src



*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

srcコード
dsp8
dsp16/abs16
dsp24/abs24

.size	SIZE
.B	0
.W	1

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

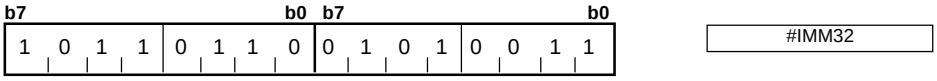
【バイト数／サイクル数】

src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2 src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

PUSH

(3) PUSH.L　#IMM32

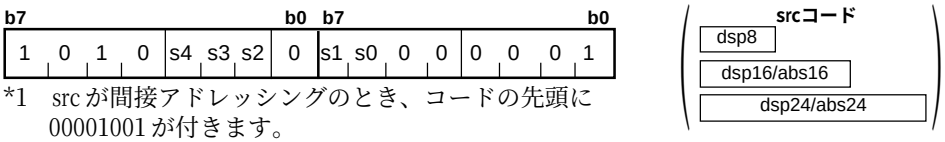


【バイト数／サイクル数】

バイト数/サイクル数	6/3

PUSH

(4) PUSH.L　　src



*1　src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1 s0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

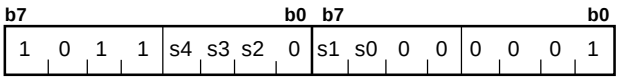
【バイト数／サイクル数】

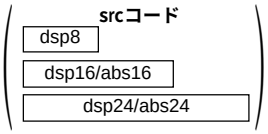
src	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5

*2　src が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

PUSHA

(1) PUSHA　　src





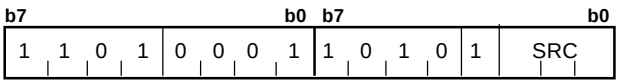
src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1s0
Rn	---/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	---	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	---	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	---	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	---	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【バイト数／サイクル数】

src	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3	3/3	4/3	4/3	5/3	4/3	5/3

PUSHC

(1) PUSHC　　src



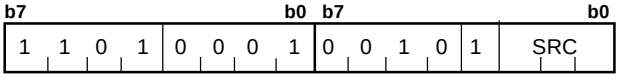
src	SRC	src	SRC
DCT0	0 0 0	DRC0	1 0 0
DCT1	0 0 1	DRC1	1 0 1
FLG	0 1 0	DMD0	1 1 0
SVF	0 1 1	DMD1	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	2/1

PUSHC

(2) PUSHC　　src



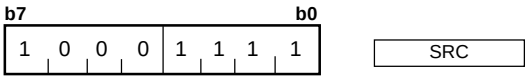
src	SRC	src	SRC
INTB	0 0 0	---	1 0 0
SP	0 0 1	---	1 0 1
SB	0 1 0	---	1 1 0
FB	0 1 1	ISP	1 1 1

【バイト数／サイクル数】

バイト数/サイクル数	2/4

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。

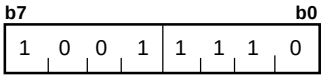
【バイト数／サイクル数】

バイト数/サイクル数	2/m

*2　mは退避するレジスタ数です。 m=(R0、R1、R2、R3の数)＋2×(A0、A1、FB、SBの数)

REIT

(1) REIT

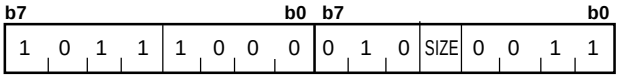


【バイト数／サイクル数】

バイト数/サイクル数	1/6

RMPA

(1) RMPA.size



.size	SIZE
.B	0
.W	1

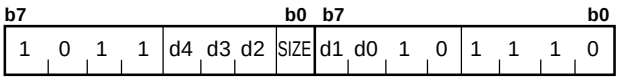
【バイト数／サイクル数】

バイト数/サイクル数	2/7+2m

*1　mは演算回数です。

ROLC

(1) ROLC.size dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

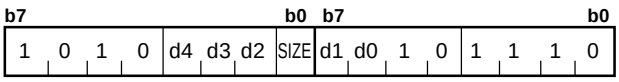
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

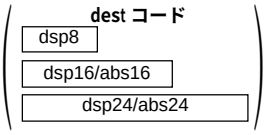
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

RORC

(1) RORC.size dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

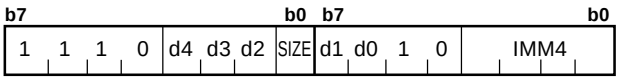
【バイト数／サイクル数】

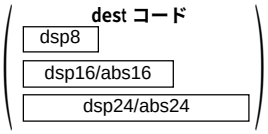
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

ROT

(1) ROT.size #IMM,dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

#IMM	IMM4	dest	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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

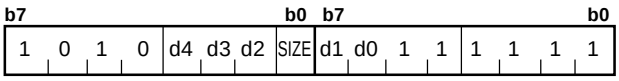
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/m	2/m	2/2+m	3/2+m	3/2+m	4/2+m	4/2+m	5/2+m	4/2+m	5/2+m

*2 mは回転回数です。
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

ROT

(2) ROT.size R1H,dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

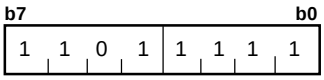
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
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	5/3+m	4/3+m	5/3+m

*2 mは回転回数です。
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

RTS

(1) RTS

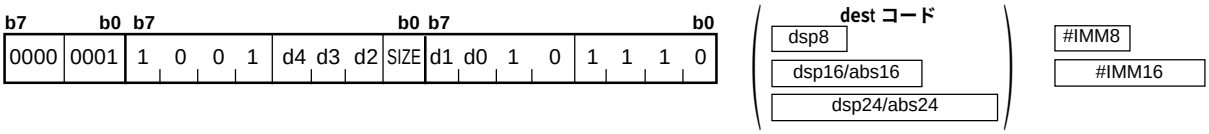


【バイト数／サイクル数】

バイト数/サイクル数	1/6

SBB

(1) SBB.size　　#IMM, dest



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

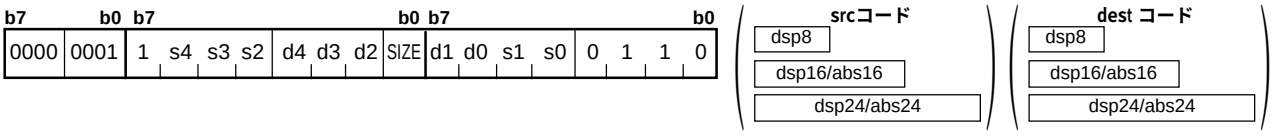
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/1	4/1	4/3	5/3	5/3	6/3	6/3	7/3	6/3	7/3

*1　サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

SBB

(2) SBB.size src, dest



.size	SIZE
.B	0
.W	1

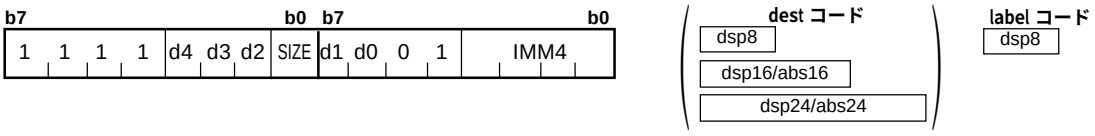
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
An	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:8[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:16[SB/FB]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:24[An]	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4
abs16	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs24	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4

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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

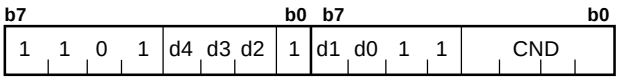
【バイト数／サイクル数】

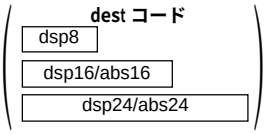
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4

*1　label に分岐したとき、表中のサイクル数は +2 されます。

SC*Cnd*

(1) SC*Cnd* dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

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

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R2/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R3/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	---/A0/---	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	---/A1/---	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

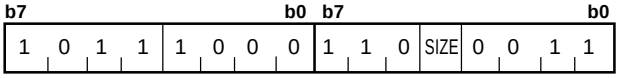
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/1	2/1	2/1	3/1	3/1	4/1	4/1	5/1	4/1	5/1

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SCMPU

(1) SCMPU.size



.size	SIZE
.B	0
.W	1

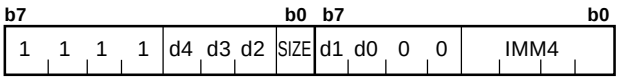
【バイト数／サイクル数】

サイズ指定子	バイト数/サイクル数		備　考
	一致で終了	不一致で終了	
.B	2/6+3m	2/6+3m	最後の0(null)がワードの上位8ビット
.W	2/6+1.5m	2/9+1.5m	
.W	2/8+1.5m	2/10+1.5m	最後の0(null)がワードの下位8ビット

*1　mは比較回数です。

SHA

(1) SHA.size #IMM, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

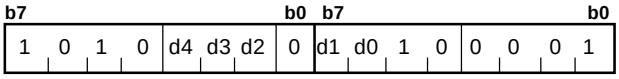
【バイト数／サイクル数】

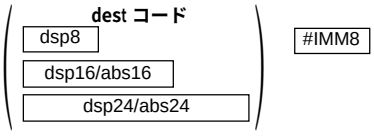
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/m	2/m	2/2+m	3/2+m	3/2+m	4/2+m	4/2+m	5/2+m	4/2+m	5/2+m

*2 mはシフト回数です。
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SHA

(2) SHA.L #IMM, dest





*1 destが間接アドレッシングのとき、コードの先頭に00001001が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

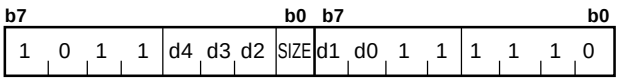
【バイト数／サイクル数】

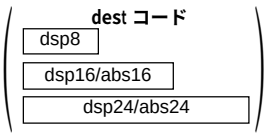
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3+m	3/3+m	3/2+m	4/3+m	4/3+m	5/3+m	5/3+m	6/3+m	5/3+m	6/3+m

*2 mはシフト回数です。
*3 destが間接アドレッシングのとき、表中のバイト数は+1、サイクル数は+3されます。

SHA

(3) SHA.size R1H, dest





*1 destが間接アドレッシングのとき、コードの先頭に00001001が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

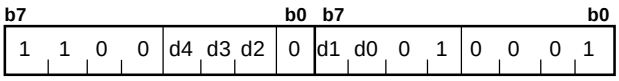
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
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	5/3+m	4/3+m	5/3+m

*2 mはシフト回数です。
*3 destが間接アドレッシングのとき、表中のバイト数は+1、サイクル数は+3されます。

SHA

(4) SHA.L R1H, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

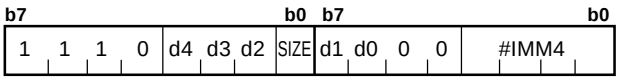
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/4+m	2/4+m	2/4+m	3/4+m	3/4+m	4/4+m	4/4+m	5/4+m	4/4+m	5/4+m

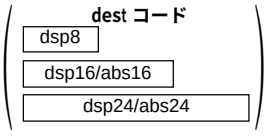
*2 mはシフト回数です。
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SHL

(1) SHL.size #IMM, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

#IMM	IMM4	dest	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		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【バイト数／サイクル数】

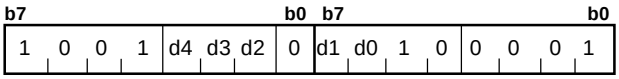
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/m	2/m	2/2+m	3/2+m	3/2+m	4/2+m	4/2+m	5/2+m	4/2+m	5/2+m

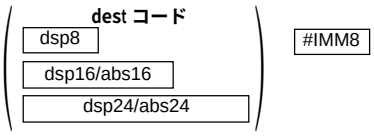
*2 mはシフト回数です。

*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SHL

(2) SHL.L #IMM, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

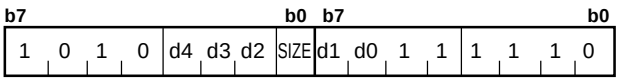
【バイト数／サイクル数】

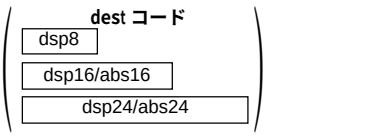
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3+m	3/3+m	3/3+m	4/3+m	4/3+m	5/3+m	5/3+m	6/3+m	5/3+m	6/3+m

*2 mはシフト回数です。
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SHL

(3) SHL.size R1H, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

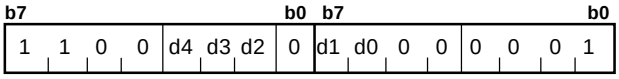
【バイト数／サイクル数】

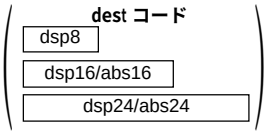
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/2+m	2/2+m	3/3+m	3/3+m	3/3+m	4/3+m	4/3+m	5/3+m	4/3+m	5/3+m

*2 mはシフト回数です。
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SHL

(4) SHL.L R1H, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

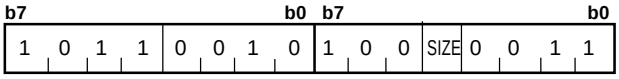
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/4+m	2/4+m	2/4+m	3/4+m	3/4+m	4/4+m	4/4+m	5/4+m	4/4+m	5/4+m

*2 mはシフト回数です。
*3 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SIN

(1) SIN.size



.size	SIZE
.B	0
.W	1

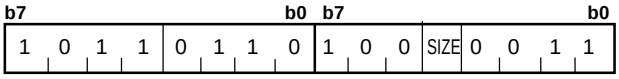
【バイト数／サイクル数】

バイト数/サイクル数	2/1+2m

*1 mは転送回数です。

SMOVB

(1) SMOVB.size



.size	SIZE
.B	0
.W	1

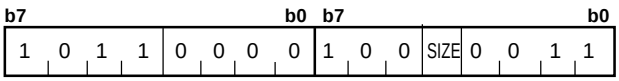
【バイト数／サイクル数】

バイト数/サイクル数	2/1+2m

*1　mは転送回数です。

SMOVF

(1) SMOVF.size



.size	SIZE
.B	0
.W	1

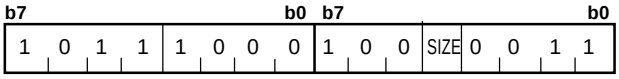
【バイト数／サイクル数】

バイト数/サイクル数	2/1+2m

*1　mは転送回数です。

SMOVU

(1) SMOVU.size



.size	SIZE
.B	0
.W	1

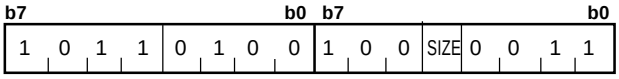
【バイト数／サイクル数】

バイト数/サイクル数	2/1+2m

*1　mは転送回数です。

SOUT

(1) SOUT.size



.size	SIZE
.B	0
.W	1

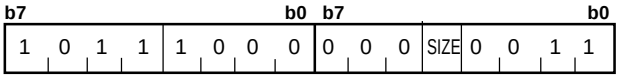
【バイト数／サイクル数】

バイト数/サイクル数	2/1+2m

*1　mは転送回数です。

SSTR

(1) SSTR.size



.size	SIZE
.B	0
.W	1

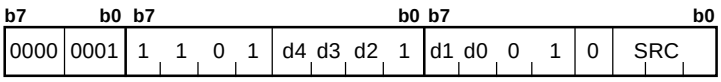
【バイト数／サイクル数】

バイト数/サイクル数	2/2+m

*1 mは転送回数です。

STC

(1) STC src, dest



src	SRC
-	000
-	001
DMA0	010
DMA1	011
DRA0	100
DRA1	101
DSA0	110
DSA1	111

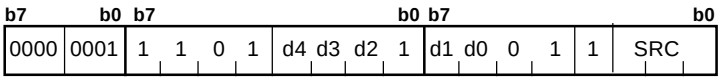
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

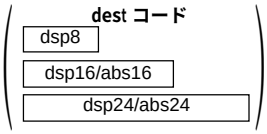
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/3	3/3	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

STC

(2) STC src, dest





src	SRC
DCT0	000
DCT1	001
FLG	010
SVF	011
DRC0	100
DRC1	101
DMD0	110
DMD1	111

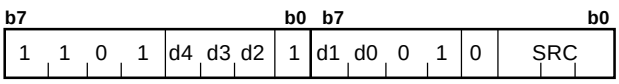
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

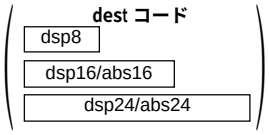
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/2	4/2	4/2	5/2	5/2	6/2	5/2	6/2

STC

(3) STC src, dest





src	SRC
INTB	000
SP	001
SB	010
FB	011
SVP	100
VCT	101
-	110
ISP	111

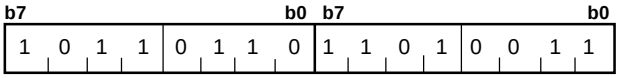
dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/3	2/3	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3

STCTX

(1) STCTX abs16, abs24



abs16　　abs24

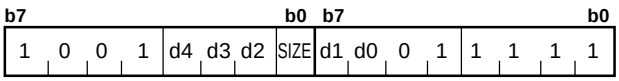
【バイト数／サイクル数】

バイト数/サイクル数	7/10+2m

*2　mは転送回数です。

STNZ

(1) STNZ.size　　#IMM, dest



*1　destが間接アドレッシングのとき、コードの先頭に00001001が付きます。

dest コード：dsp8 / dsp16/abs16 / dsp24/abs24

#IMM8 / #IMM16

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

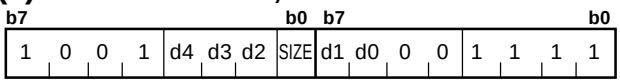
【バイト数／サイクル数】

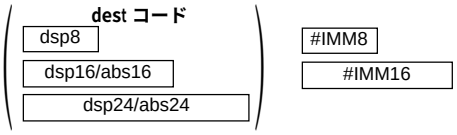
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/2	4/2	4/2	5/2	5/2	6/2	5/2	6/2

*1　destが間接アドレッシングのとき、表中のバイト数は+1、サイクル数は+3されます。
*2　サイズ指定子(.size)が(.W)のとき、表中のバイト数は+1されます。

STZ

(1) STZ.size #IMM, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

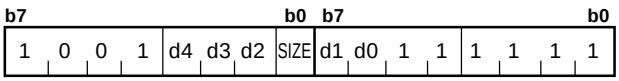
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/2	4/2	4/2	5/2	5/2	6/2	5/2	6/2

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 Z フラグが 0 のとき、表中のサイクル数は +1 されます。
*4 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

STZX

(1) STZX.size #IMM1, #IMM2, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【バイト数／サイクル数】

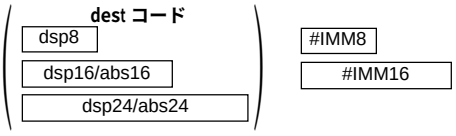
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	4/3	4/3	4/3	5/3	5/3	6/3	6/3	7/3	6/3	7/3

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +2 されます。

SUB

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

*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

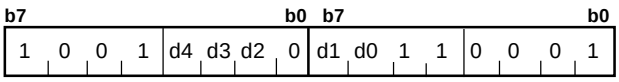
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

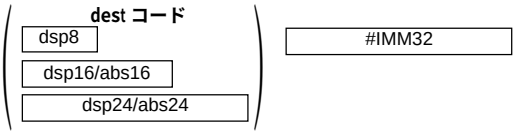
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

SUB

(2) SUB.L:G #IMM, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

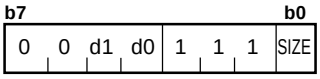
【バイト数／サイクル数】

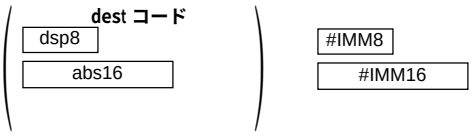
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	6/2	6/2	6/5	7/5	7/5	8/5	8/5	9/5	8/5	9/5

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SUB

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





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

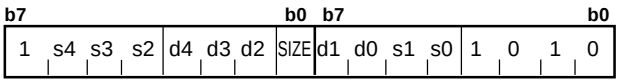
【バイト数／サイクル数】

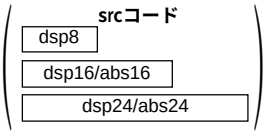
dest	Rn	dsp:8[SB/FB]	abs16
バイト数/サイクル数	2/1	3/3	4/3

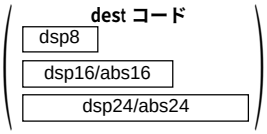
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。
*3 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

SUB

(4) SUB.size:G src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

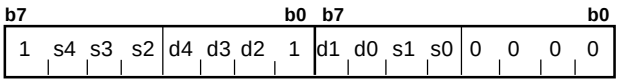
【バイト数／サイクル数】

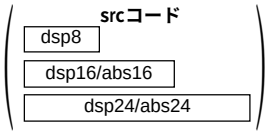
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
An	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:24[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
abs24	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

SUB

(5) SUB.L:G src, dest





*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

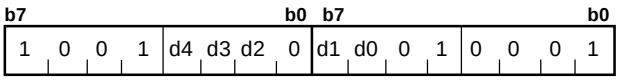
【バイト数／サイクル数】

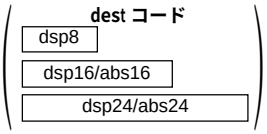
src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
An	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
[An]	2/5	2/5	2/8	3/8	3/8	4/8	4/8	5/8	4/8	5/8
dsp:8[An]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:8[SB/FB]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:16[An]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:16[SB/FB]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:24[An]	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8
abs16	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
abs24	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

SUBX

(1) SUBX #IMM, dest





*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

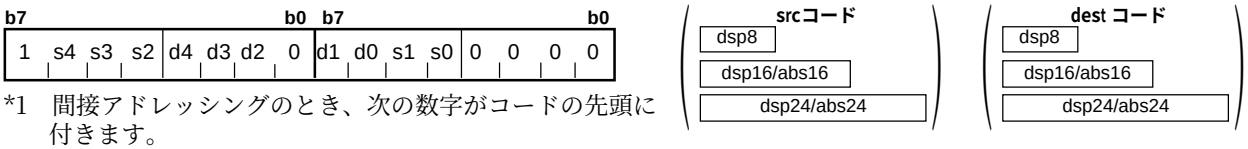
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/2	3/2	3/5	4/5	4/5	5/5	5/5	6/5	5/5	6/5

*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

SUBX

(2) SUBX src, dest



*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

src		s4 s3 s2 s1 s0	src		s4 s3 s2 s1s0
Rn	R0L/---/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/---/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	---/---/R2R0	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	---/---/R3R1	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	---/---/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	---/---/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

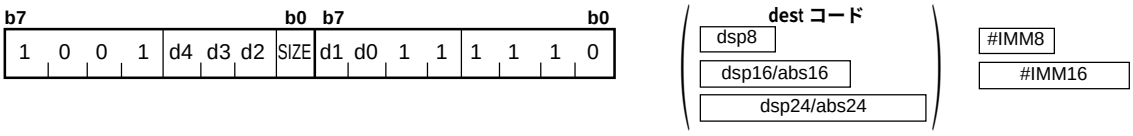
【バイト数／サイクル数】

src \ dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
Rn	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
An	2/2	2/2	2/5	3/5	3/5	4/5	4/5	5/5	4/5	5/5
[An]	2/5	2/5	2/8	3/8	3/8	4/8	4/8	5/8	4/8	5/8
dsp:8[An]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:8[SB/FB]	3/5	3/5	3/8	4/8	4/8	5/8	5/8	6/8	5/8	6/8
dsp:16[An]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:16[SB/FB]	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
dsp:24[An]	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8
abs16	4/5	4/5	4/8	5/8	5/8	6/8	6/8	7/8	6/8	7/8
abs24	5/5	5/5	5/8	6/8	6/8	7/8	7/8	8/8	7/8	8/8

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

TST

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



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

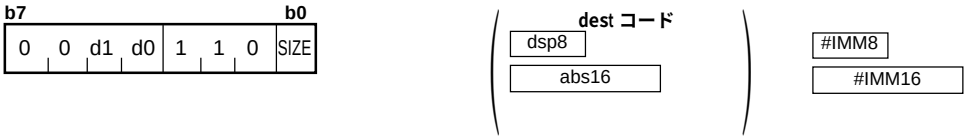
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

TST

(2) TST.size:S #IMM, dest



.size	SIZE
.B	0
.W	1

dest		d1 d0
Rn	R0L/R0	0 0
dsp:8[SB/FB]	dsp:8[SB]	1 0
	dsp:8[FB]	1 1
abs16	abs16	0 1

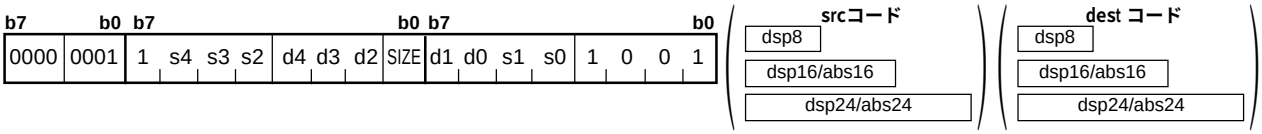
【バイト数／サイクル数】

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

*1 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

TST

(3) TST.size:G　　src, dest



.size	SIZE
.B	0
.W	1

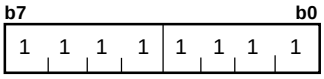
src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
An	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3
[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:8[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:16[SB/FB]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
dsp:24[An]	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4
abs16	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs24	6/3	6/3	6/4	7/4	7/4	8/4	8/4	9/4	8/4	9/4

UND

(1) UND

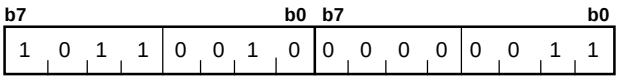


【バイト数／サイクル数】

バイト数/サイクル数	1/13

WAIT

(1) WAIT

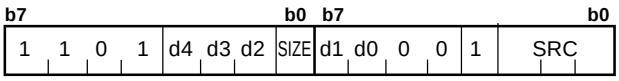


【バイト数】

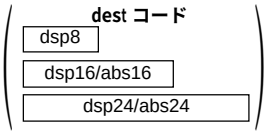
バイト数	2

XCHG

(1) XCHG.size src, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

src	SRC
R0L/R0/---	0 0 0
R1L/R1/---	0 0 1
R0H/R2/-	1 0 0
R1H/R3/-	1 0 1
A0	0 1 0
A1	0 1 1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

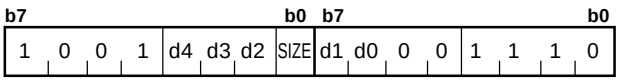
【バイト数／サイクル数】

dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	2/3	2/3	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4

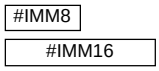
*2 dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。

XOR

(1) XOR.size #IMM, dest



*1 dest が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。



.size	SIZE
.B	0
.W	1

dest		d4 d3 d2 d1 d0	dest		d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

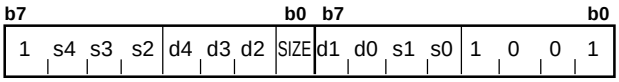
【バイト数／サイクル数】

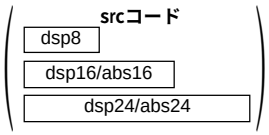
dest	Rn	An	[An]	dsp:8[An]	dsp:8[SB/FB]	dsp:16[An]	dsp:16[SB/FB]	dsp:24[An]	abs16	abs24
バイト数/サイクル数	3/1	3/1	3/3	4/3	4/3	5/3	5/3	6/3	5/3	6/3

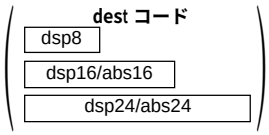
*2 サイズ指定子(.size)が(.W)のとき、表中のバイト数は +1 されます。

XOR

(2) XOR.size src, dest







*1 間接アドレッシングのとき、次の数字がコードの先頭に付きます。
src が間接アドレッシングのとき、01000001
dest が間接アドレッシングのとき、00001001
src と dest が間接アドレッシングのとき、01001001

.size	SIZE
.B	0
.W	1

src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0	src/dest		s4 s3 s2 s1 s0 d4 d3 d2 d1 d0
Rn	R0L/R0/---	1 0 0 1 0	dsp:8[SB/FB]	dsp:8[SB]	0 0 1 1 0
	R1L/R1/---	1 0 0 1 1		dsp:8[FB]	0 0 1 1 1
	R0H/R2/-	1 0 0 0 0	dsp:16[An]	dsp:16[A0]	0 1 0 0 0
	R1H/R3/-	1 0 0 0 1		dsp:16[A1]	0 1 0 0 1
An	A0	0 0 0 1 0	dsp:16[SB/FB]	dsp:16[SB]	0 1 0 1 0
	A1	0 0 0 1 1		dsp:16[FB]	0 1 0 1 1
[An]	[A0]	0 0 0 0 0	dsp:24[An]	dsp:24[A0]	0 1 1 0 0
	[A1]	0 0 0 0 1		dsp:24[A1]	0 1 1 0 1
dsp:8[An]	dsp:8[A0]	0 0 1 0 0	abs16	abs16	0 1 1 1 1
	dsp:8[A1]	0 0 1 0 1	abs24	abs24	0 1 1 1 0

【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/FB]	dsp:24[An]	abs16	abs24
Rn	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
An	2/1	2/1	2/3	3/3	3/3	4/3	4/3	5/3	4/3	5/3
[An]	2/3	2/3	2/4	3/4	3/4	4/4	4/4	5/4	4/4	5/4
dsp:8[An]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:8[SB/FB]	3/3	3/3	3/4	4/4	4/4	5/4	5/4	6/4	5/4	6/4
dsp:16[An]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:16[SB/FB]	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
dsp:24[An]	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4
abs16	4/3	4/3	4/4	5/4	5/4	6/4	6/4	7/4	6/4	7/4
abs24	5/3	5/3	5/4	6/4	6/4	7/4	7/4	8/4	7/4	8/4

*2 src または dest が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +3 されます。また src と dest の両方が間接アドレッシングのとき、表中のバイト数は +1、サイクル数は +6 されます。

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、表 5.1.2 にエミュレータ専用割り込み(ノンマスカブル)とベクタテーブルを示します。

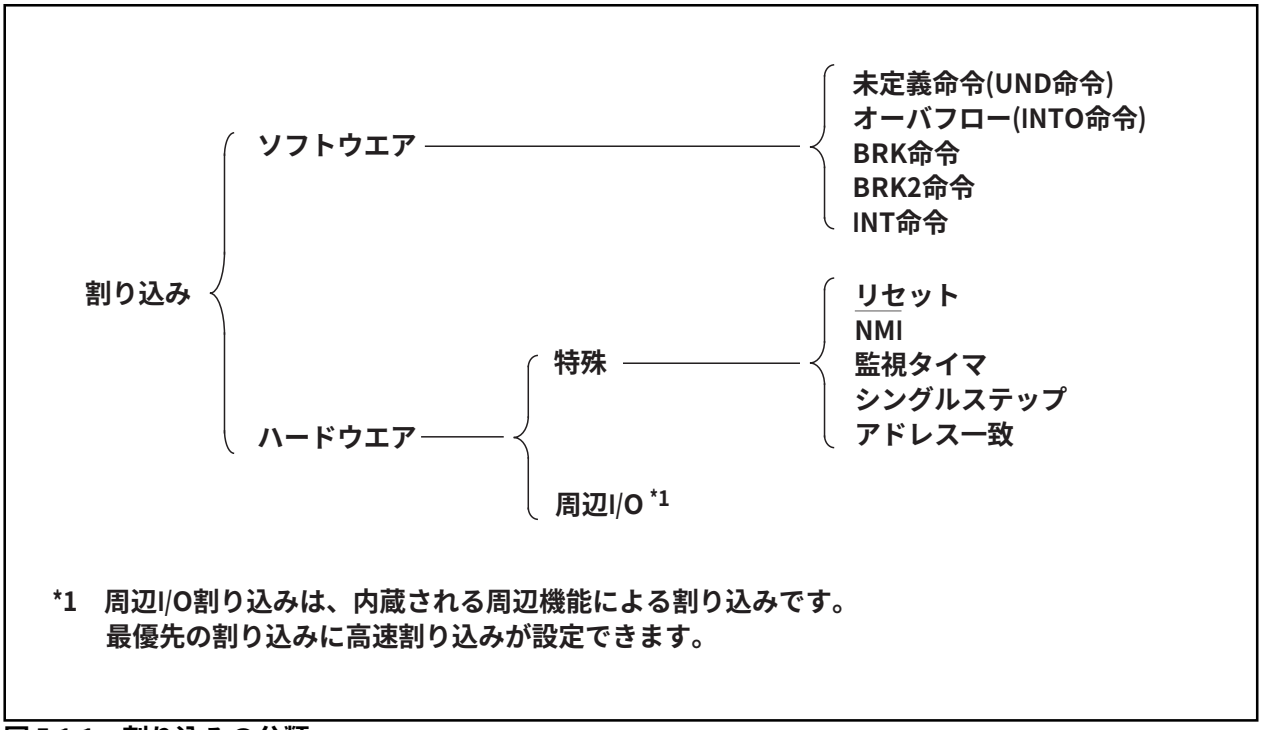


図 5.1.1　割り込みの分類

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	ベクタテーブル番地 アドレス(L)～アドレス(H)	備 考
未定義命令	FFFFDC16～FFFFDF16	UND命令で割り込み
オーバフロー	FFFFE016～FFFFE316	INTO命令で割り込み
BRK命令	FFFFE416～FFFFE716	FFFFE716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	FFFFE816～FFFFEB16	アドレス一致割り込み許可ビットあり
監視タイマ	FFFFF016～FFFFF316	
$\overline{\text{NMI}}$	FFFFF816～FFFFFB16	$\overline{\text{NMI}}$端子入力による外部割り込み
リセット	FFFFFC16～FFFFFF16	

表 5.1.2 エミュレータ専用割り込み(ノンマスカブル)とベクタテーブル

割り込み要因	ベクタテーブル番地 アドレス(L)～アドレス(H)	備 考
BRK2命令	エミュレータ専用割り込みベクタテーブルレジスタ 00002016～00002316	デバッガ用割り込み
シングルステップ	エミュレータ専用割り込みベクタテーブルレジスタ 00002016～00002316	デバッガ用割り込み

■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

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。

(1)未定義命令割り込み

未定義命令割り込みは、UND 命令を実行すると発生します。

(2) オーバフロー割り込み

オーバフロー割り込みは、オーバフローフラグ(O フラグ)が“1”のとき INTO 命令を実行すると発生します。演算によって O フラグが変化する命令を以下に示します。

ABS、ADC、ADCF、ADD、ADDX、CMP、CMPX、DIV、DIVU、DIVX、NEG、RMPA、SBB、SCMPU、SHA、SUB、SUBX

(3) BRK 割り込み

BRK 割り込みは、BRK 命令を実行すると発生します。

(4) BRK2 割り込み

BRK2 割り込みは BRK2 命令を実行すると発生します。

この割り込みはエミュレータ専用割り込みです。ユーザプログラムでは使用しないでください。

(5) INT 命令割り込み

INT 命令割り込みは、ソフトウエア割り込み番号 0～63 を指定し、INT 命令を実行すると発生します。なお、ソフトウエア割り込み番号 0～43 は周辺 I/O 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 I/O 割り込みと同じ割り込みルーチンを実行できます。

INT命令割り込みに使用するスタックポインタ(SP)は、ソフトウエア割り込み番号によって異なります。ソフトウエア割り込み番号0～31では、割り込み要求受け付け時にスタックポインタ指定フラグ(U フラグ)を退避し、U フラグを“0”にして割り込みスタックポインタ(ISP)を選択した後、割り込みシーケンスを実行します。割り込みルーチンから復帰するときに割り込み要求受け付け前の U フラグが復帰されます。ソフトウエア割り込み番号 32～63 では、スタックポインタは切り替わりません。

ただし、周辺I/O割り込みでは、割り込み要求受け付け時にスタックポインタ指定フラグ(U フラグ)を退避し、U フラグを“0”にして割り込みスタックポインタ(ISP)を選択します。そのため、ソフトウエア割り込み番号 32～43 では周辺 I/O 割り込みか INT 命令かで U フラグの動作は異なります。

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 I/O 割り込みがあります。

(1)特殊割り込み

特殊割り込みは、ノンマスカブル割り込みです。

●リセット

リセットは、$\overline{\text{RESET}}$ 端子に“ L ”を入力すると発生します。

● $\overline{\text{NMI}}$ 割り込み

$\overline{\text{NMI}}$ 割り込みは、$\overline{\text{NMI}}$ 端子に“ L ”を入力すると発生します。

●監視タイマ割り込み

監視タイマによる割り込みです。

●アドレス一致割り込み

アドレス一致割り込みは、アドレス一致割り込み許可ビットを“ 1 ”にしたとき、プログラムの実行アドレスとアドレス一致レジスタの内容が一致すると発生します。

アドレス一致レジスタに命令の先頭番地以外の番地を設定した場合は、アドレス一致割り込みは発生しません。

●シングルステップ割り込み

デバッガ専用割り込みですので、通常は使用しないでください。シングルステップ割り込みは、デバッグフラグ(D フラグ)を“ 1 ”にすると、命令を 1 つ実行した後に発生します。

(2) 周辺 I/O 割り込み

周辺I/O割り込みは、内蔵される周辺機能による割り込みです。内蔵される周辺機能は品種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。割り込みベクタテーブルは INT 命令で使用するソフトウエア割り込み番号 0 ～ 43 と同一です。周辺 I/O 割り込みは、マスカブル割り込みです。周辺 I/O 割り込みについての詳細はユーザーズマニュアルを参照してください。

周辺I/O割り込みでは、割り込み要求受け付け時にスタックポインタ指定フラグ(Uフラグ)を退避し、U フラグを“ 0 ”にして割り込みスタックポインタ(ISP)を選択した後、割り込みシーケンスを実行します。割り込みシーケンスから復帰するときに割り込み要求受け付け前の U フラグが復帰されます。

(3) 高速割り込み

高速割り込みは、割り込みの応答を高速に実行できる割り込みです。周辺I/O割り込みの中で最優先の割り込みに使用できます。高速割り込みからの復帰には FREIT 命令を使用してください。

高速割り込みについての詳細はユーザーズマニュアルを参照してください。

5.2　割り込み制御

マスカブル割り込みの許可/禁止、受け付ける優先順位の設定について説明します。ここで説明する内容は、ノンマスカブル割り込みには該当しません。

マスカブル割り込みの許可および禁止は、割り込み許可フラグ(I フラグ)、割り込み優先レベル選択ビット、およびプロセッサ割り込み優先レベル(IPL)によって行います。また、割り込み要求の有無は、割り込み要求ビットに示されます。割り込み要求ビットおよび割り込み優先レベル選択ビットは、各割り込みの割り込み制御レジスタに配置されています。また、割り込み許可フラグ(I フラグ)、およびプロセッサ割り込み優先レベル(IPL)は、フラグレジスタ(FLG)に配置されています。

割り込み制御レジスタのメモリ配置およびレジスタ構成は、ユーザーズマニュアルを参照してください。

5.2.1　割り込み許可フラグ(I フラグ)

割り込み許可フラグ(I フラグ)は、マスカブル割り込みの禁止／許可の制御を行います。このフラグを“ 1 ”にすると、すべてのマスカブル割り込みは許可され、“ 0 ”にすると禁止されます。このフラグはリセット解除後“ 0 ”になります。

Iフラグを変化させたとき、変化した内容が割り込み要求受付判定に反映されるのは次のタイミングです。

・REIT、FREIT 命令で変化させたとき、その REIT、FREIT 命令から反映される。

・FCLR、FSET、POPC、LDC 各命令で変化させたとき、次の命令から反映される。

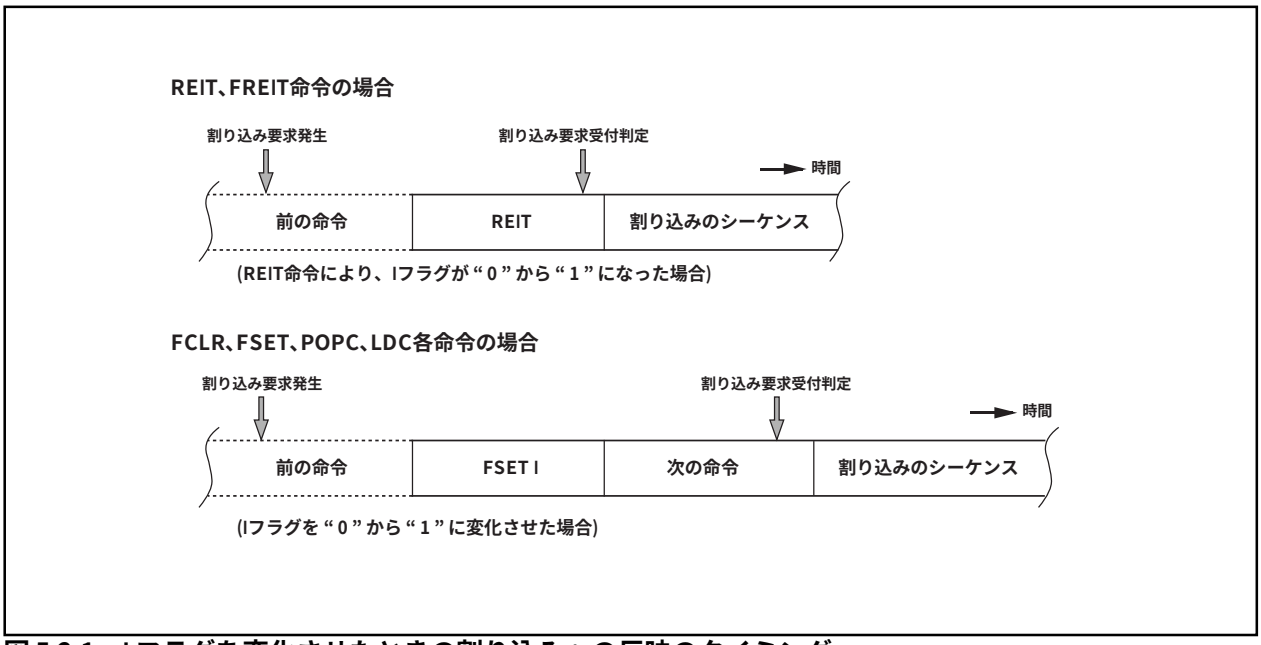


図 5.2.1　I フラグを変化させたときの割り込みへの反映のタイミング

5.2.2　割り込み要求ビット

割り込み要求ビットは割り込み要求が発生すると“ 1 ”になり、割り込み要求が受け付けられるまで保持されます。割り込み要求が受け付けられると“ 0 ”になります。

また、このビットはソフトウエアによって“ 0 ”にできます(“ 1 ”を書き込まないでください)。

5.2.3 割り込み優先レベル選択ビット、およびプロセッサ割り込み優先レベル(IPL)

割り込み優先レベルは、割り込み制御レジスタの中の割り込み優先レベル選択ビットで設定します。割り込み要求発生時、割り込み優先レベルは、プロセッサ割り込み優先レベル(IPL)と比較され、割り込みの優先レベルがプロセッサ割り込み優先レベル(IPL)より大きい場合だけ、その割り込みは許可されます。したがって、割り込み優先レベルにレベル0を設定すれば、その割り込みは禁止されます。

表5.2.1に割り込み優先レベルの設定を、表5.2.2にプロセッサ割り込み優先レベル(IPL)の内容による割り込み許可レベルを示します。

割り込み要求が受け付けられる条件を以下に示します。

●割り込み許可フラグ(I フラグ)　＝ “1”
●割り込み要求ビット　＝ “1”
●割り込み優先レベル　＞ プロセッサ割り込み優先レベル(IPL)

割り込み許可フラグ(Iフラグ)、割り込み要求ビット、割り込み優先レベル選択ビット、およびプロセッサ割り込み優先レベル(IPL)はそれぞれ独立しており、互いに影響を与えることはありません。

表 5.2.1 割り込み優先レベルの設定

割り込み優先レベル選択ビット	割り込み優先レベル	優先順位
b2 b1 b0 0 0 0	レベル0 (割り込み禁止)	―――
0 0 1	レベル1	低い
0 1 0	レベル2	↓
0 1 1	レベル3	↓
1 0 0	レベル4	↓
1 0 1	レベル5	↓
1 1 0	レベル6	↓
1 1 1	レベル7	高い

表 5.2.2 プロセッサ割り込み優先レベル(IPL)の内容による割り込み許可レベル

プロセッサ割り込み優先レベル(IPL)	許可される割り込み優先レベル
IPL_2 IPL_1 IPL_0 0 0 0	レベル1以上を許可
0 0 1	レベル2以上を許可
0 1 0	レベル3以上を許可
0 1 1	レベル4以上を許可
1 0 0	レベル5以上を許可
1 0 1	レベル6以上を許可
1 1 0	レベル7以上を許可
1 1 1	すべてのマスカブル割り込みを禁止

プロセッサ割り込み優先レベル(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、FREIT 命令でプロセッサ割り込み優先レベル(IPL)を変化させたとき、その REIT、FREIT 命令から反映される。
・POPC、LDC、LDIPL 各命令でプロセッサ割り込み優先レベル(IPL)を変化させたとき、次の命令から反映される。
・各割り込み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 2 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

割り込みが禁止状態で、割り込み制御レジスタを書き換える命令を実行しているときに、そのレジスタに対応する割り込み要求が発生した場合、命令によっては割り込み要求ビットがセットされないことがあります。このことが問題になる場合は、以下の命令を使用してレジスタを変更するようにしてください。

対象となる命令・・・AND、OR、BCLR、BSET

5.3 割り込みシーケンス

割り込み要求が受け付けられてから割り込みルーチンが実行されるまでの、割り込みシーケンスについて説明します。

命令実行中に割り込み要求が発生すると、その命令の実行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、SCMPU, SIN, SMOVB, SMOVF, SMOVU, SSTR, SOUT, 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、次の動作を順次行います。

(1)00000016 番地(高速割り込みの場合、00000216 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)を獲得する。その後、該当する割り込みの要求ビットが“0”になる。
(2)割り込みシーケンス直前のフラグレジスタ(FLG)の内容を CPU 内部の一時レジスタ*1 に退避する。
(3)割り込み許可フラグ(I フラグ)、デバッグフラグ(D フラグ)、およびスタックポインタ指定フラグ(U フラグ)を“0”にする(ただし U フラグは、ソフトウエア割り込み番号 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*1 の内容をスタック領域に退避する。高速割り込みの場合は、フラグ退避レジスタ(SVF)に退避する。
(5)プログラムカウンタ(PC)の内容をスタック領域に退避する。高速割り込みの場合は、PC退避レジスタ(SVP)に退避する。
(6)プロセッサ割り込み優先レベル(IPL)に、受け付けた割り込みの割り込み優先レベルを設定する。

割り込みシーケンス終了後は、割り込みルーチンの先頭番地から命令を実行します。

*1 ユーザは使用できません。

5.3.1　割り込み応答時間

割り込み応答時間とは、割り込み要求が発生してから割り込みルーチン内の最初の命令を実行するまでの時間を示します。この時間は、割り込み要求発生時点から、そのとき実行している命令が終了するまでの時間(a)と割り込みシーケンスを実行する時間(b)で構成されます。図5.3.1に割り込み応答時間を示します。

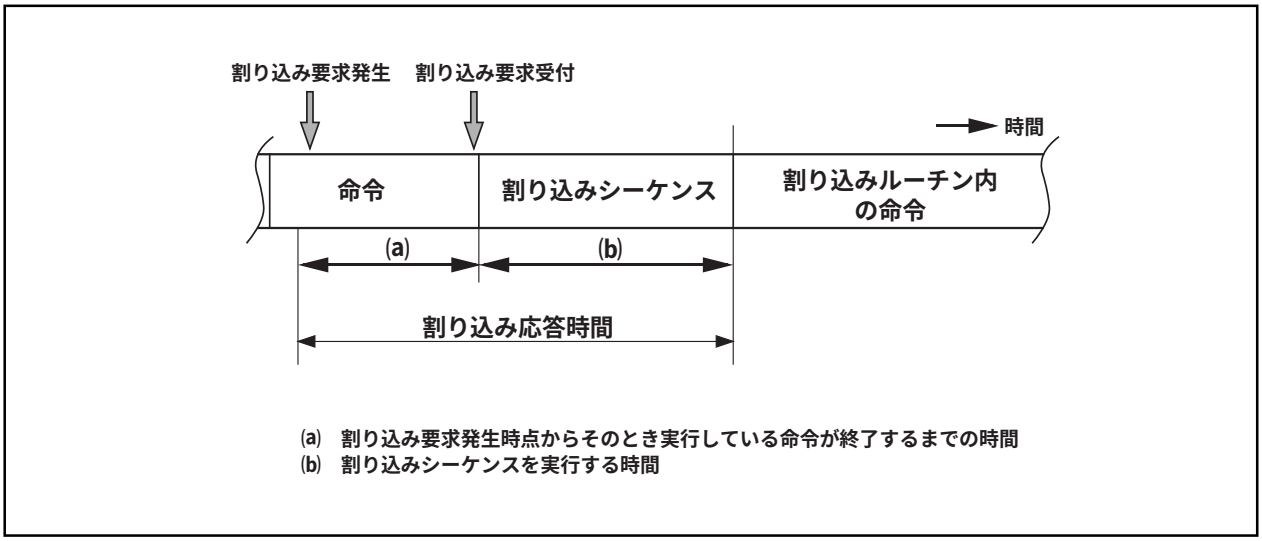


図 5.3.1　割り込み応答時間

(a)の時間は、実行している命令によって異なります。DIVX 命令が最大で 24*1 サイクルです。
(b)の時間は表 5.3.1 のとおりです。

*1　除数が即値かレジスタのときです。除数がメモリのときは、下記の値が加算されます。
- ・通常アドレッシング　　2 ＋ X
- ・インデクスアドレッシング　　3 ＋ X
- ・間接アドレッシング　　5 ＋ X ＋ 2Y
- ・間接インデクスアドレッシング　　6 ＋ X ＋ 2Y

Xは除数の領域のウエイト数です。Yは間接アドレスが格納されている領域のウエイト数です。もし、これらが奇数番地か、8 ビットバス領域にあるなら、その値を 2 倍してください。

表 5.3.1　割り込みシーケンス実行時間

割り込み	割り込みベクタの番地	16ビットバス	8ビットバス
周辺I/O	偶数	14サイクル	16サイクル
	奇数 *2	16サイクル	16サイクル
INT命令	偶数	12サイクル	14サイクル
	奇数 *2	14サイクル	14サイクル
$\overline{\text{NMI}}$ 監視タイマ 未定義命令 アドレス一致	偶数 *1	13サイクル	15サイクル
オーバフロー	偶数 *1	14サイクル	16サイクル
BRK命令 (可変ベクタテーブル)	偶数	17サイクル	19サイクル
	奇数 *2	19サイクル	19サイクル
シングルステップ BRK2命令 BRK命令 (固定ベクタテーブル)	偶数 *1	19サイクル	21サイクル
高速割り込み *3	ベクタテーブルは内部レジスタ	5サイクル	

*1　ベクタテーブルのアドレスは偶数固定です。
*2　割り込みベクタテーブルは、なるべく偶数番地に配置するようにしてください。
*3　高速割り込みは、これらの条件に影響されません。

5.3.2　割り込み要求受付時のプロセッサ割り込み優先レベル(IPL)の変化

割り込み要求が受け付けられると、プロセッサ割り込み優先レベル(IPL)には受け付けた割り込みの割り込み優先レベルが設定されます。

割り込み優先レベルをもたない割り込み要求が受け付けられたときは、表5.3.2に示す値がIPLに設定されます。

表 5.3.2　割り込み優先レベルをもたない割り込みと IPL の関係

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、$\overline{\text{NMI}}$	7
リセット	0
その他	変化しない

5.3.3 レジスタ退避

割り込みシーケンスでは、フラグレジスタ(FLG)とプログラムカウンタ(PC)の内容だけがスタック領域に退避されます。

スタック領域へ退避する順番は、FLG レジスタをまず退避し、次にプログラムカウンタを 32 ビットに拡張した上位16ビットと下位16ビットを退避します。図5.3.2に割り込み要求受付前のスタックの状態と、割り込み要求受付後のスタックの状態を示します。高速割り込みの割り込みシーケンスではフラグレジスタ(FLG)をフラグ退避レジスタ(SVF)に、プログラムカウンタ(PC)をPC退避レジスタ(SVP)に退避されます。

その他の必要なレジスタは、割り込みルーチンの最初でソフトウエアによって退避してください。PUSHM命令を用いると、１命令でスタックポインタ(SP)を除くすべてのレジスタを退避することができます。

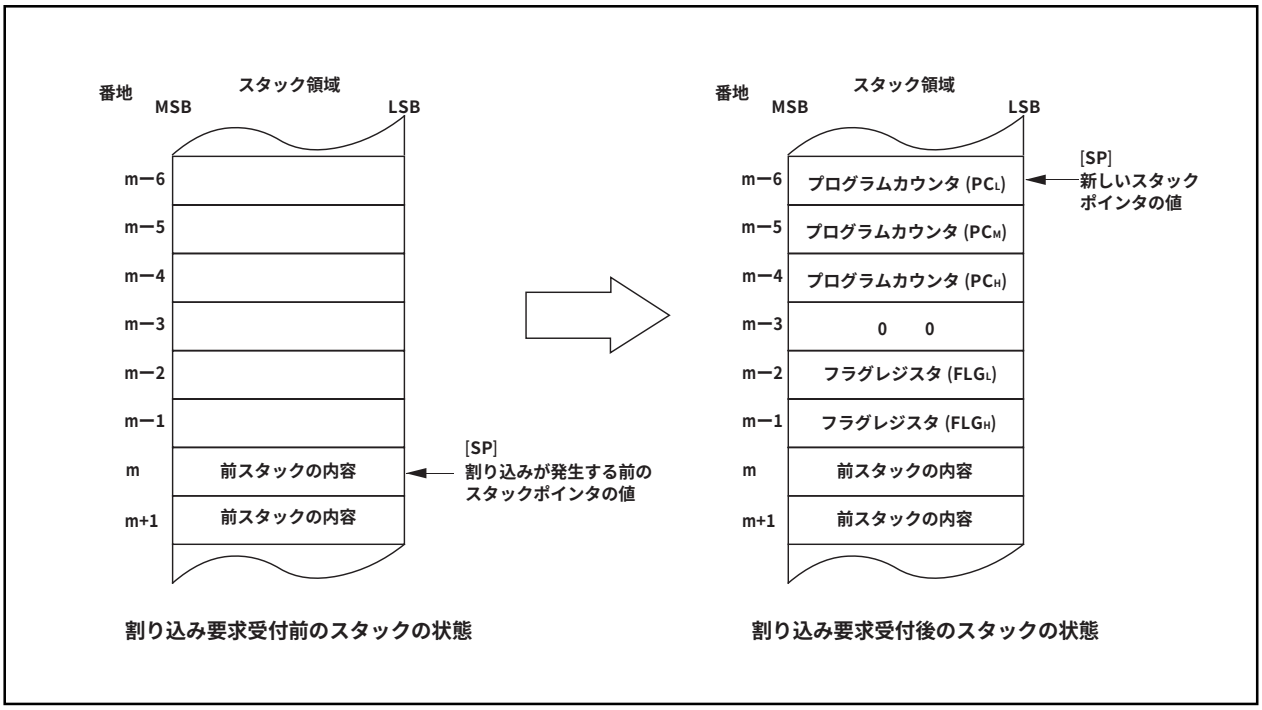


図 5.3.2 割り込み要求受付前／割り込み要求受付後のスタックの状態

5.4 割り込みルーチンからの復帰

割り込みルーチンの最後でREIT命令を実行すると、スタック領域に退避されていた割り込みシーケンス直前のフラグレジスタ(FLG)、およびプログラムカウンタ(PC)の内容が復帰されます。高速割り込みの場合は、割り込みルーチンの最後で FREIT 命令を実行すると、退避レジスタに退避されていた割り込みシーケンス直前のフラグレジスタ(FLG)、およびプログラムカウンタ(PC)の内容が復帰されます。その後、割り込み要求受付前に実行していたプログラムに戻り、中断されていた処理が継続して実行されます。

割り込みルーチン内でソフトウエアによって退避したレジスタは、REIT、FREIT 命令実行前に POPM 命令などを使用して復帰してください。

5.5　割り込み優先順位

同一サンプリング時点(割り込みの要求があるかどうかを調べるタイミング)で2つ以上の割り込み要求が存在した場合は、優先順位の高い割り込みが受け付けられます。

マスカブル割り込み(周辺I/O割り込み)の優先順位は、割り込み優先レベル選択ビットによって任意の優先順位を設定することができます。ただし、割り込み優先レベルが同じ設定値の場合は、一番最初に要求がきた割り込みを最初に受け付け、それ以降はハードウエアで設定されている優先度[*1]の高い割り込みが受け付けられます。

リセット(リセットは優先順位の一番高い割り込みとして扱われます)、監視タイマ割り込みなど、ノンマスカブル割り込みの優先順位はハードウエアで設定されています。図5.5.1にハードウエアで設定されている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。命令を実行すると必ず割り込みルーチンへ分岐します。

*1　品種により異なりますので、ユーザーズマニュアルを参照してください。

リセット＞$\overline{\text{NMI}}$＞監視タイマ＞周辺I/O＞シングルステップ＞アドレス一致

図 5.5.1　ハードウエアで設定されている割り込み優先順位

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、FREIT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)

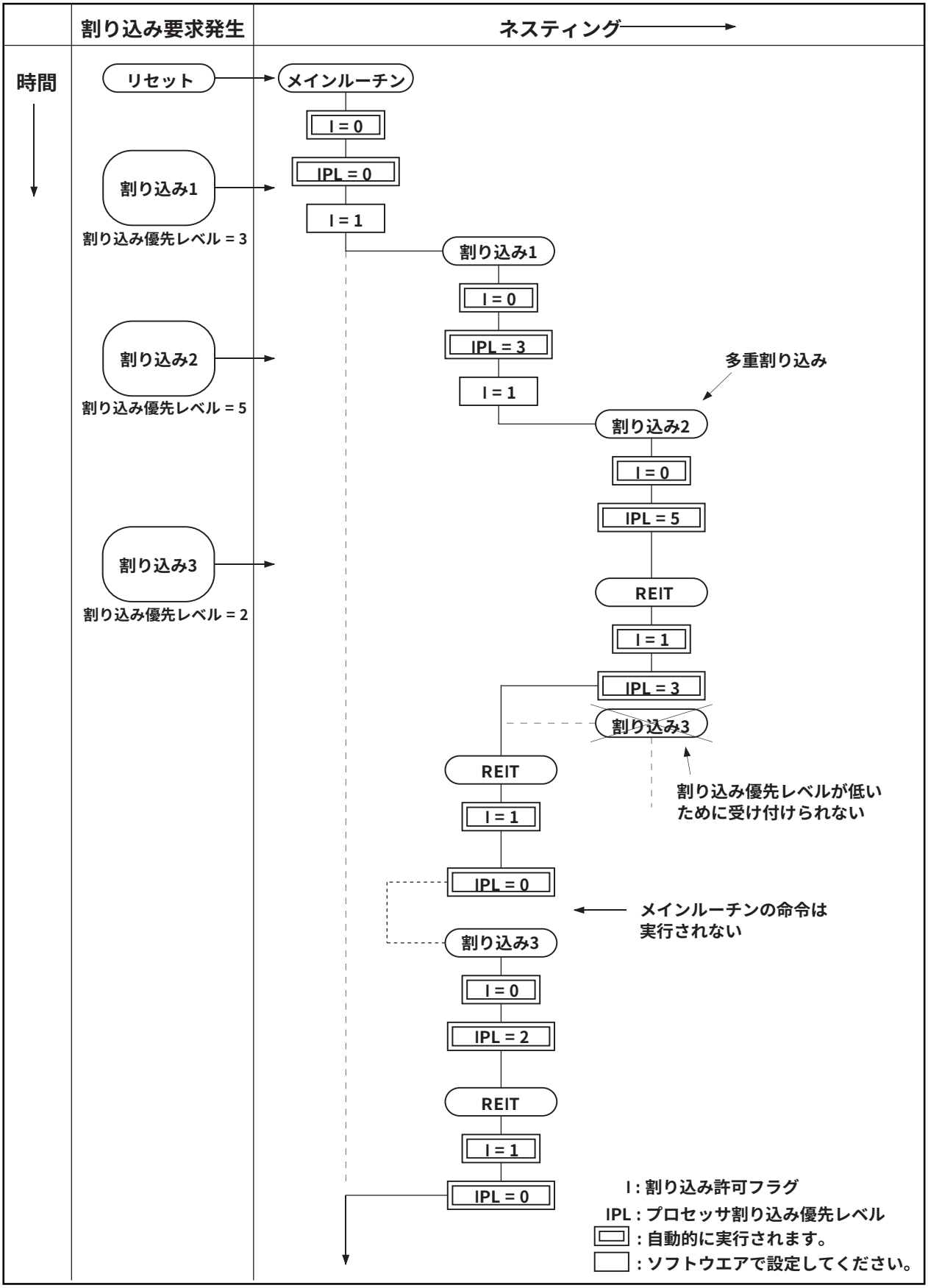

割り込み要求発生
ネスティング
時間
リセット
メインルーチン
I = 0
IPL = 0
I = 1
割り込み1
割り込み優先レベル = 3
割り込み1
I = 0
IPL = 3
I = 1
割り込み2
割り込み優先レベル = 5
多重割り込み
割り込み2
I = 0
IPL = 5
REIT
I = 1
IPL = 3
割り込み3
割り込み3
割り込み優先レベル = 2
REIT
I = 1
IPL = 0
メインルーチンの命令は
実行されない
割り込み優先レベルが低い
ために受け付けられない
割り込み3
I = 0
IPL = 2
REIT
I = 1
IPL = 0
I : 割り込み許可フラグ
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

(1)　000000$_{16}$番地、000002$_{16}$番地の読み出し

●マスカブル割り込みが発生した場合、割り込みシーケンスの中でCPUは、割り込み情報(割り込み番号と割り込み要求レベル)を00000016番地から読み出します。高速割り込みの場合、00000216番地から読み出します。
それを読み出すことでその割り込みが発生する割り込み要求ビットが“0”になります。
ただし、ソフトウエアにより00000016番地、00000216番地を読み出しても、この要求ビットは“0”になりません。

(2)　スタックポインタの設定

●リセット直後スタックポインタの値は、“00000016”に初期化されています。そのため、スタックポインタに値を設定する前に割り込みを受け付けると、暴走の要因となります。割り込みを受け付ける前に、必ずスタックポインタに値を設定してください。
特に、$\overline{\text{NMI}}$割り込みを使用する場合は、プログラムの先頭でスタックポインタを初期化してください。
$\overline{\text{NMI}}$割り込みを含むすべての割り込みは、リセット後、1命令を実行した直後から受け付けられます。
スタックポインタには偶数アドレスを設定してください。偶数を設定した方が効率が良くなります。

(3)　割り込み制御レジスタの変更

●割り込みが禁止状態で、割り込み制御レジスタを書き換える命令を実行しているときに、そのレジスタに対応する割り込み要求が発生した場合、命令によっては割り込み要求ビットがセットされないことがあります。このことが問題になる場合は、以下の命令を使用してレジスタを変更するようにしてください。
　対象となる命令・・・AND、OR、BCLR、BSET

5.8　ストップモード・ウエイトモードの解除

周辺I/O割り込みでストップモード・ウエイトモードを解除する場合、対象となる割り込みは、あらかじめ割り込み許可状態にし、割り込みの優先レベルをストップ/ウエイト復帰用割り込み優先順位設定ビットで指定したレベルより高いレベルに設定する必要があります。ストップ/ウエイト復帰用割り込み順位設定ビットはフラグレジスタ(FLG)のプロセッサ割り込みレベル(IPL)と同じ値を設定してください。

リセット、$\overline{\text{NMI}}$割り込みは、ストップ/ウエイト復帰用割り込み優先順位設定ビットの影響を受けず、ストップモード・ウエイトモードを解除します。

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

M16C/80、M16C/70シリーズは、8段(8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、16ビットバスで接続しているメモリ(内部メモリも含む)に対しデータを偶数番地からソフトウエアウエイト、$\overline{\text{RDY}}$などの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。
(1)ソフトウエアウエイト、$\overline{\text{RDY}}$などの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。
(2)8ビットバスに接続しているメモリから命令コードを読み込む
16ビットバスに比べサイクル数が増加します。

●ソフトウエアウエイト、$\overline{\text{RDY}}$などの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。

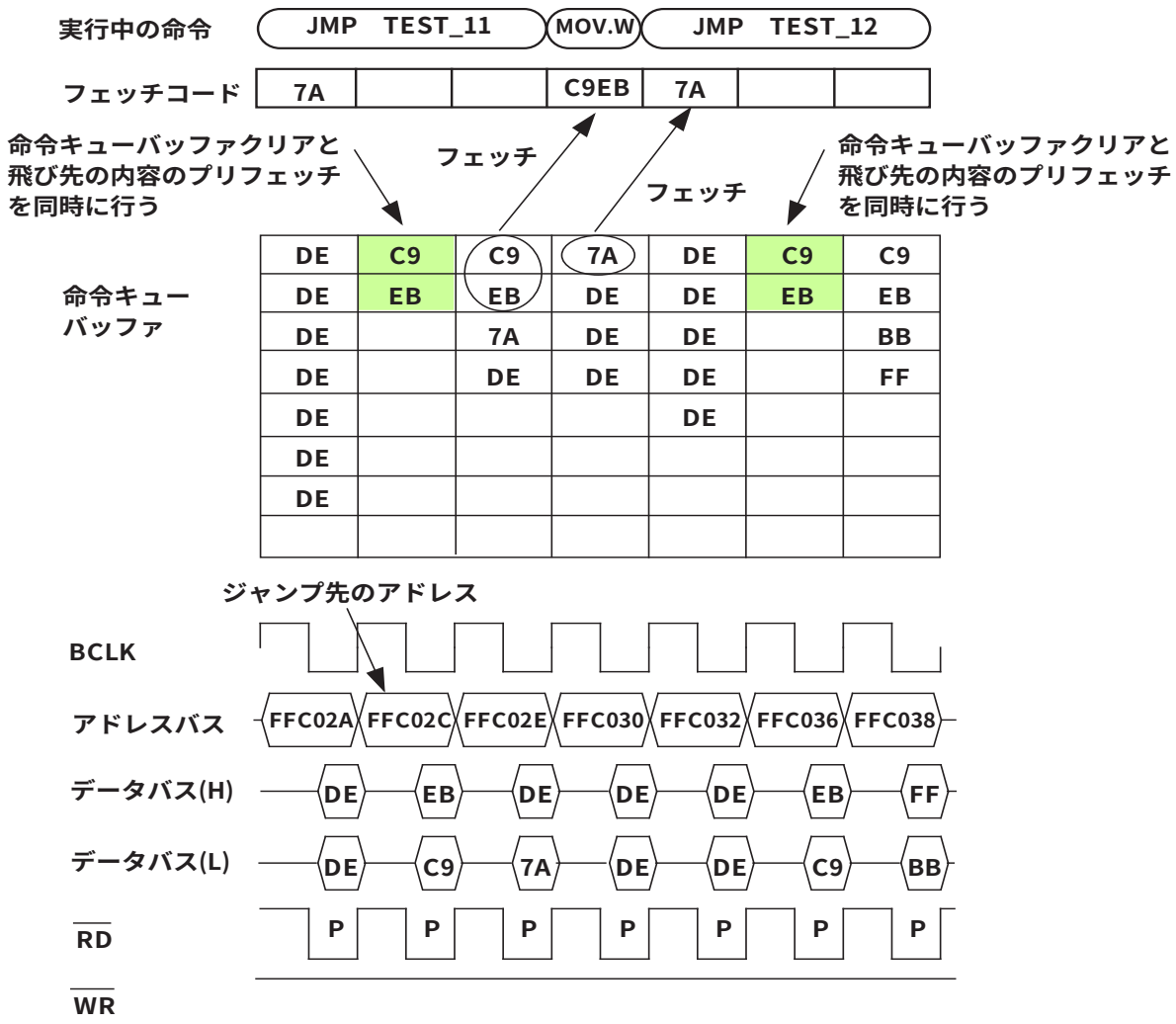
●8ビットバスに接続しているメモリに対して16ビットのデータを読み書きする。
1データの読み書きに対して2回読み書きします。そのため、1データの読み書きにつきサイクル数は、1サイクル増加します。

●16ビットバスに接続しているメモリに対して奇数番地から16ビットのデータを読み書きする。
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7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。

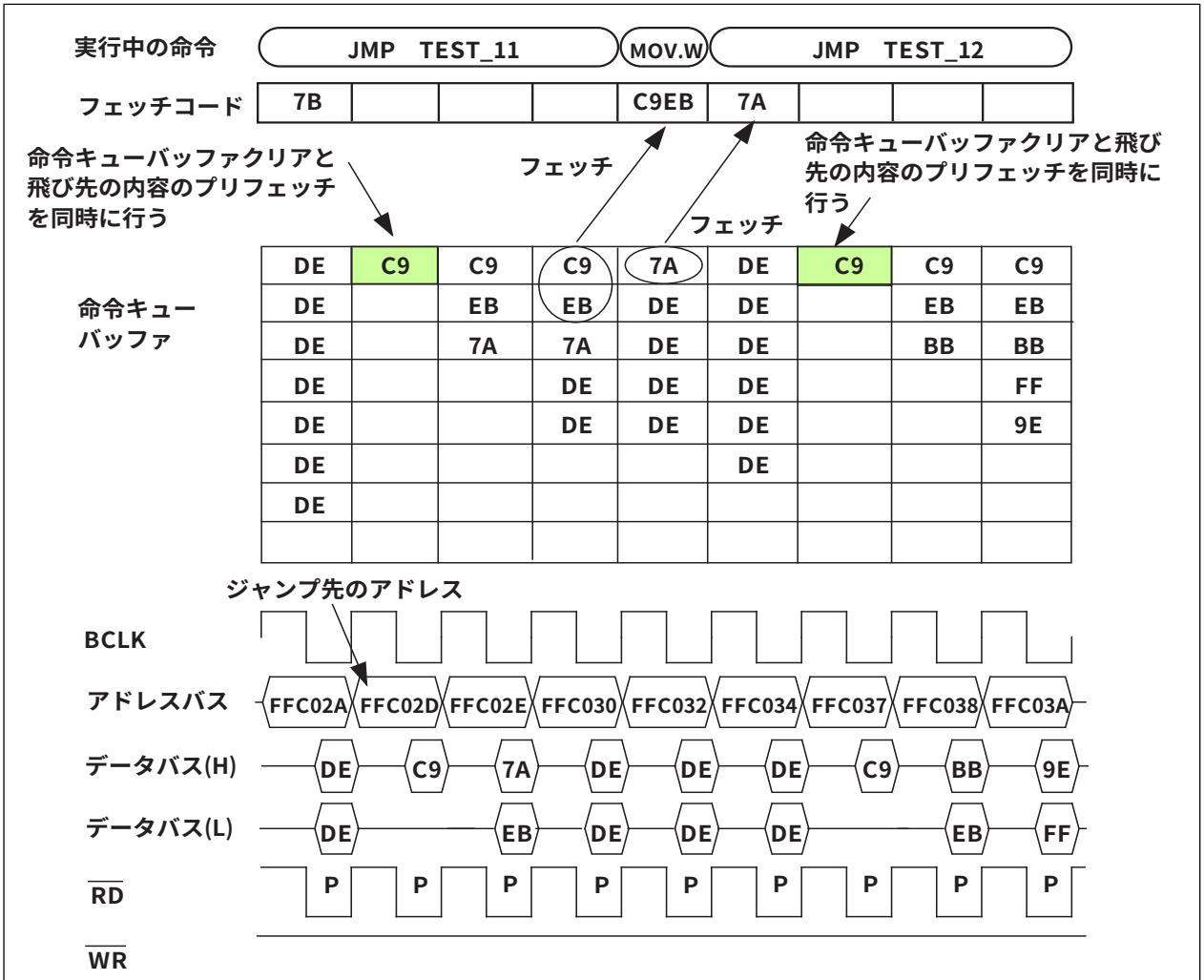
図6.1.1～図6.1.8に命令キューバッファの動作とCPUの実行サイクルの例を示します。



参考プログラム

```
アドレス　コード　　命令
FFC024　　TEST_10:
FFC024　7A　　JMP　　TEST_11
FFC025　DE　　NOP
FFC026　DE　　NOP
FFC027　DE　　NOP
FFC028　DE　　NOP
FFC029　DE　　NOP
FFC02A　DE　　NOP
FFC02B　DE　　NOP
FFC02C　　TEST_11:
FFC02C　C9EB　　MOV.W:G　　R0,R1
FFC02E　7A　　JMP　　TEST_12
FFC02F　DE　　NOP
FFC030　DE　　NOP
FFC031　DE　　NOP
FFC032　DE　　NOP
FFC033　DE　　NOP
FFC034　DE　　NOP
FFC035　DE　　NOP
FFC036　　TEST_12:
```

図6.1.1　レジスタ転送命令を偶数番地から開始する場合
(プログラム領域：16ビットバスウエイトなし、データ領域：16ビットバスウエイトなし)

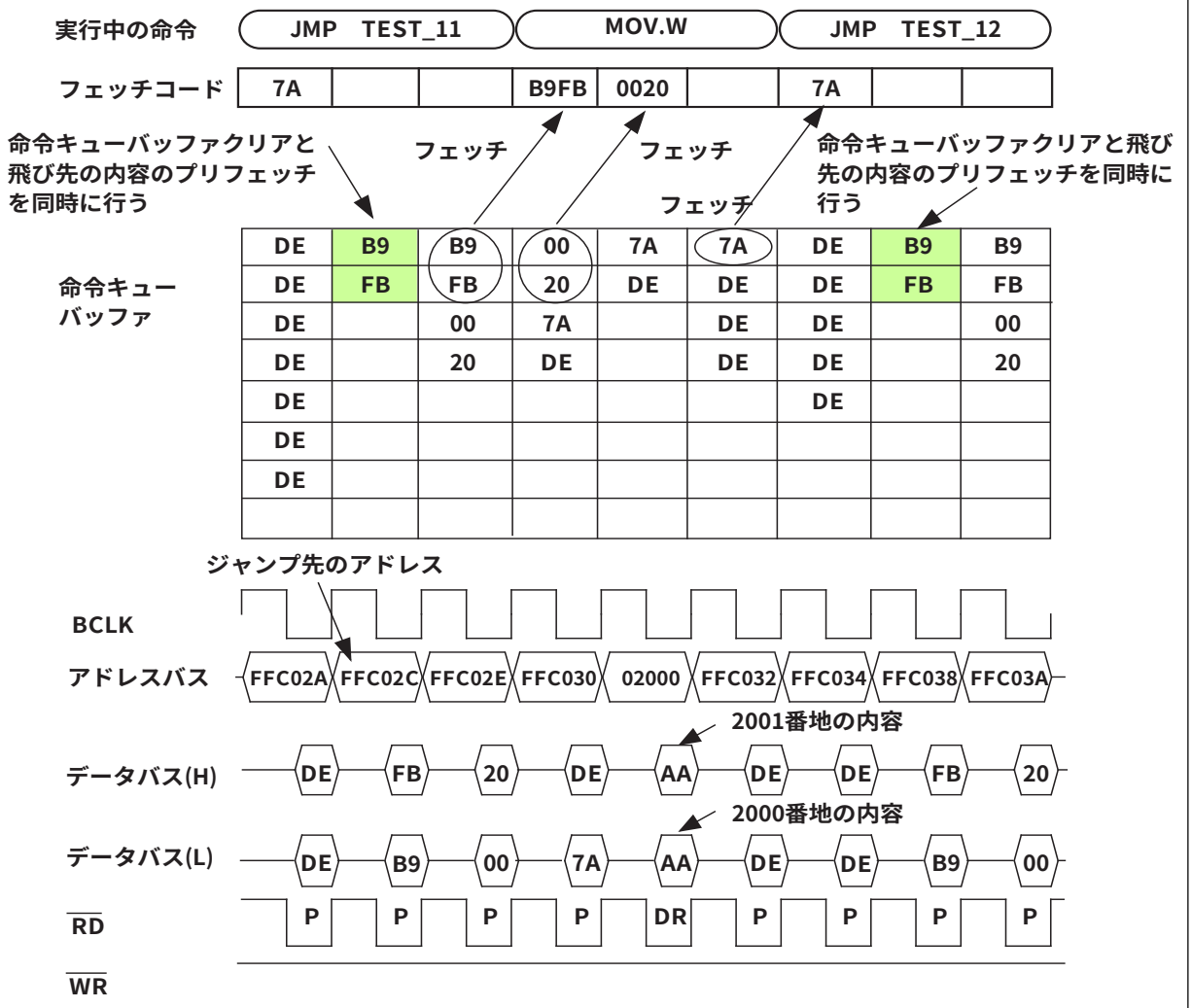


P : プリフェッチ(メモリから命令キューバッファへの取り込み)を示します。

:命令キューバッファクリアの箇所を示します。

```
参考プログラム
アドレス  コード        命令
FFC024          TEST_10:
FFC024    7B            JMP             TEST_11
FFC025    DE            NOP
FFC026    DE            NOP
FFC027    DE            NOP
FFC028    DE            NOP
FFC029    DE            NOP
FFC02A    DE            NOP
FFC02B    DE            NOP
FFC02C    DE            NOP
FFC02D          TEST_11:
FFC02D    C9EB          MOV.W:G         R0,R1
FFC02F    7A            JMP             TEST_12
FFC030    DE            NOP
FFC031    DE            NOP
FFC032    DE            NOP
FFC033    DE            NOP
FFC034    DE            NOP
FFC035    DE            NOP
FFC036    DE            NOP
FFC037          TEST_12:
```

図6.1.2　レジスタ転送命令を奇数番地から開始する場合
(プログラム領域：16ビットバスウエイトなし、データ領域：16ビットバスウエイトなし)

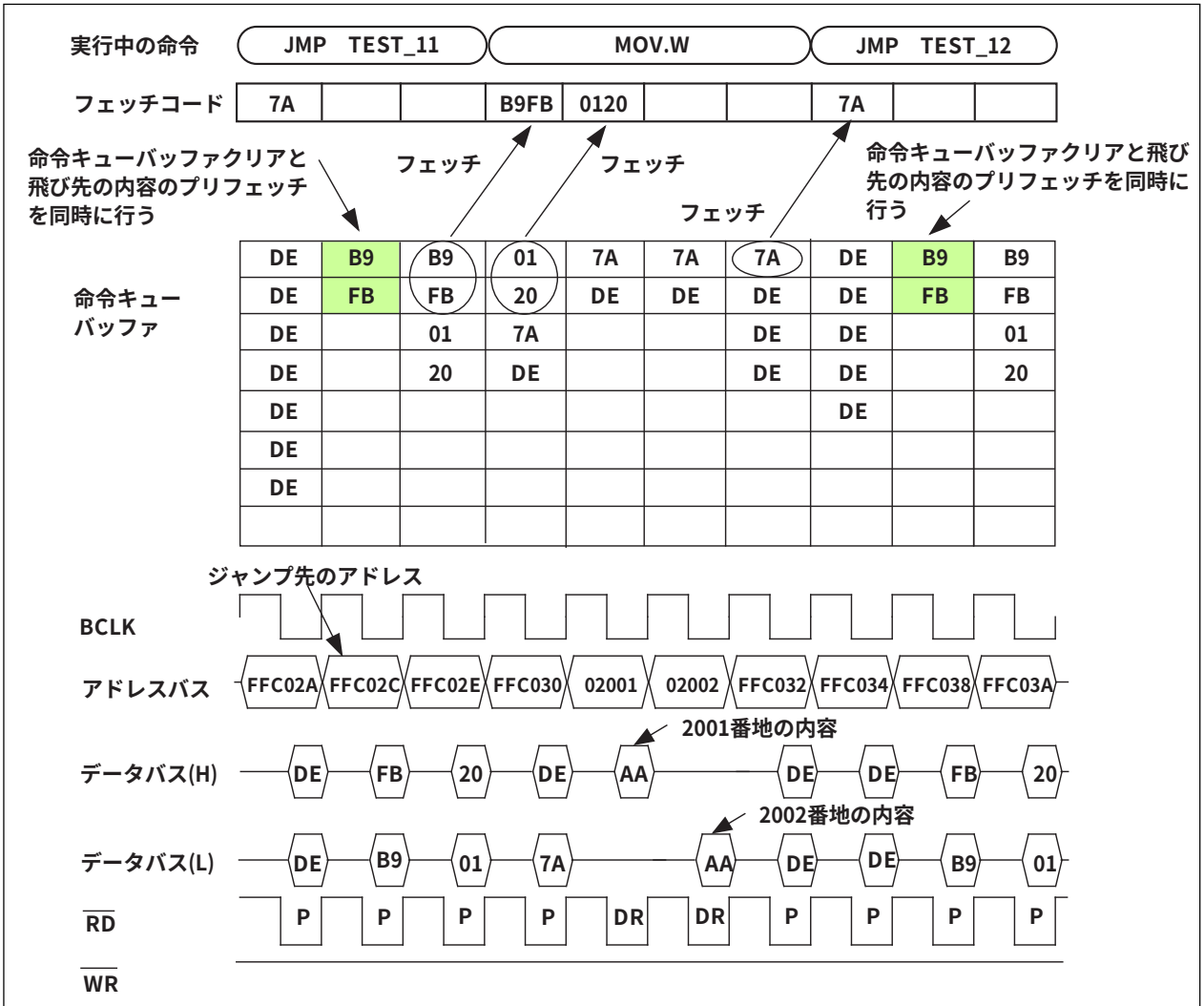


P : プリフェッチ(メモリから命令キューバッファへの取り込み)を示します。
DR : データのリードを示します。
:命令キューバッファクリアの箇所を示します。

参考プログラム

```
アドレス   コード        命令
FFC024          TEST_10:
FFC024   7A             JMP             TEST_11
FFC025   DE             NOP
FFC026   DE             NOP
FFC027   DE             NOP
FFC028   DE             NOP
FFC029   DE             NOP
FFC02A   DE             NOP
FFC02B   DE             NOP
FFC02C          TEST_11:
FFC02C   B9FB0020       MOV.W:G         02000h,R1
FFC030   7A             JMP             TEST_12
FFC031   DE             NOP
FFC032   DE             NOP
FFC033   DE             NOP
FFC034   DE             NOP
FFC035   DE             NOP
FFC036   DE             NOP
FFC037   DE             NOP
FFC038          TEST_12:
```

図6.1.3 偶数アドレスからの読み込み命令を偶数番地から開始する場合
(プログラム領域：16ビットバスウエイトなし、データ領域：16ビットバスウエイトなし)



P : プリフェッチ(メモリから命令キューバッファへの取り込み)を示します。
DR : データのリードを示します。
:命令キューバッファクリアの箇所を示します。

```
参考プログラム
アドレス   コード        命令
FFC024          TEST_10:
FFC024   7A             JMP             TEST_11
FFC025   DE             NOP
FFC026   DE             NOP
FFC027   DE             NOP
FFC028   DE             NOP
FFC029   DE             NOP
FFC02A   DE             NOP
FFC02B   DE             NOP
FFC02C          TEST_11:
FFC02C   B9FB0120       MOV.W:G         02001h,R1
FFC030   7A             JMP             TEST_12
FFC031   DE             NOP
FFC032   DE             NOP
FFC033   DE             NOP
FFC034   DE             NOP
FFC035   DE             NOP
FFC036   DE             NOP
FFC037   DE             NOP
FFC038          TEST_12:
```

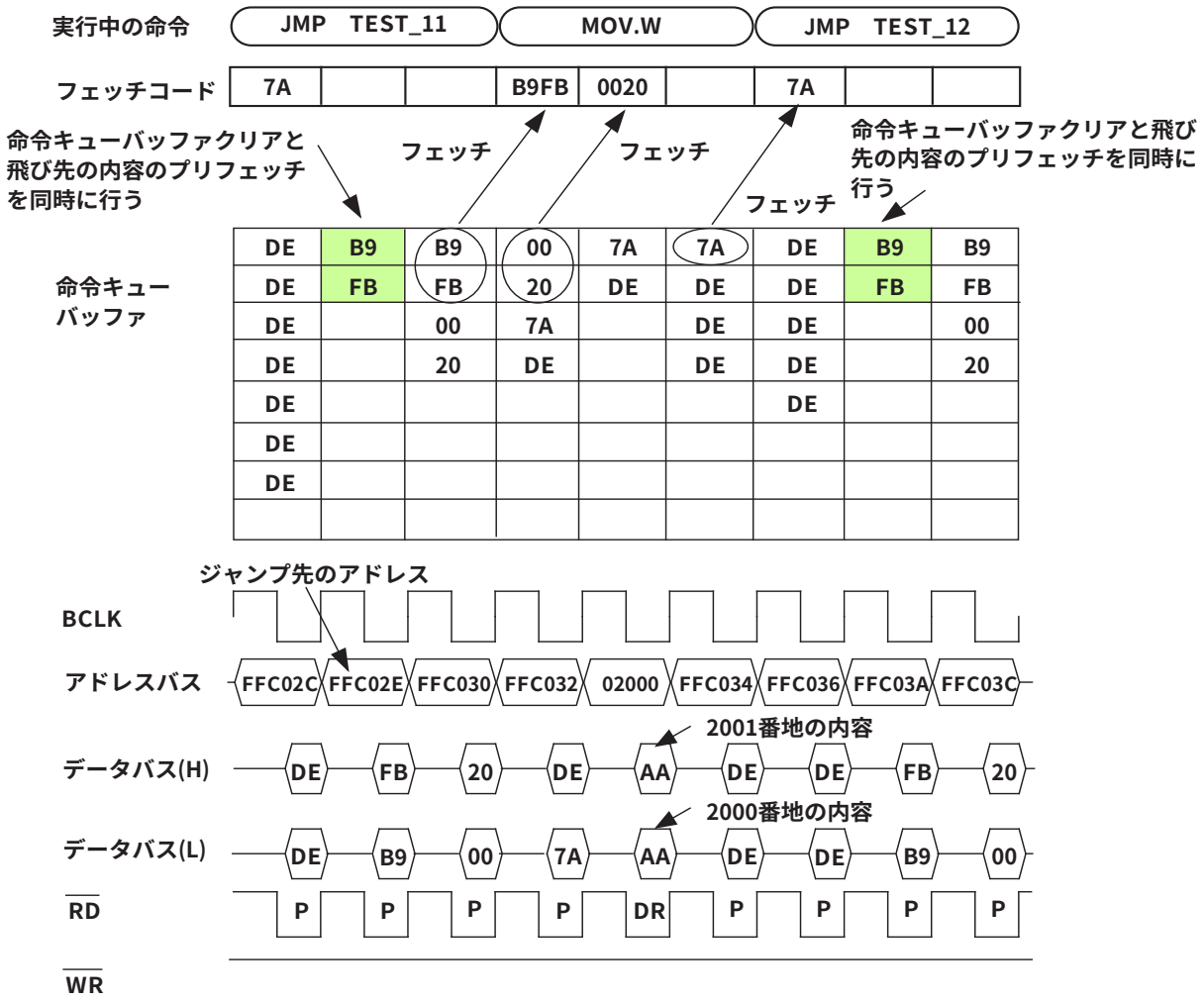
図6.1.4　奇数アドレスからの読み込み命令を偶数番地から開始する場合
(プログラム領域：16ビットバスウエイトなし、データ領域：16ビットバスウエイトなし)

P : プリフェッチ(メモリから命令キューバッファへの取り込み)を示します。
DR : データのリードを示します。
DW : データのライトを示します。
:命令キューバッファクリアの箇所を示します。

参考プログラム

```
アドレス  コード            命令
FFC024        TEST_10:
FFC024  7A                JMP         TEST_11
FFC025  DE                NOP
FFC026  DE                NOP
FFC027  DE                NOP
FFC028  DE                NOP
FFC029  DE                NOP
FFC02A  DE                NOP
FFC02B  DE                NOP
FFC02C        TEST_11:
FFC02C  B7FB00200220      MOV.W       02000h,02002h
FFC032  7A                JMP         TEST_12
FFC033  DE                NOP
FFC034  DE                NOP
FFC035  DE                NOP
FFC036  DE                NOP
FFC037  DE                NOP
FFC038  DE                NOP
FFC039  DE                NOP
FFC03A        TEST_12:
```

図6.1.5 偶数アドレスから偶数アドレスへのデータ転送命令を偶数番地から開始する場合
(プログラム領域：16ビットバスウエイトなし、データ領域：16ビットバスウエイトなし)

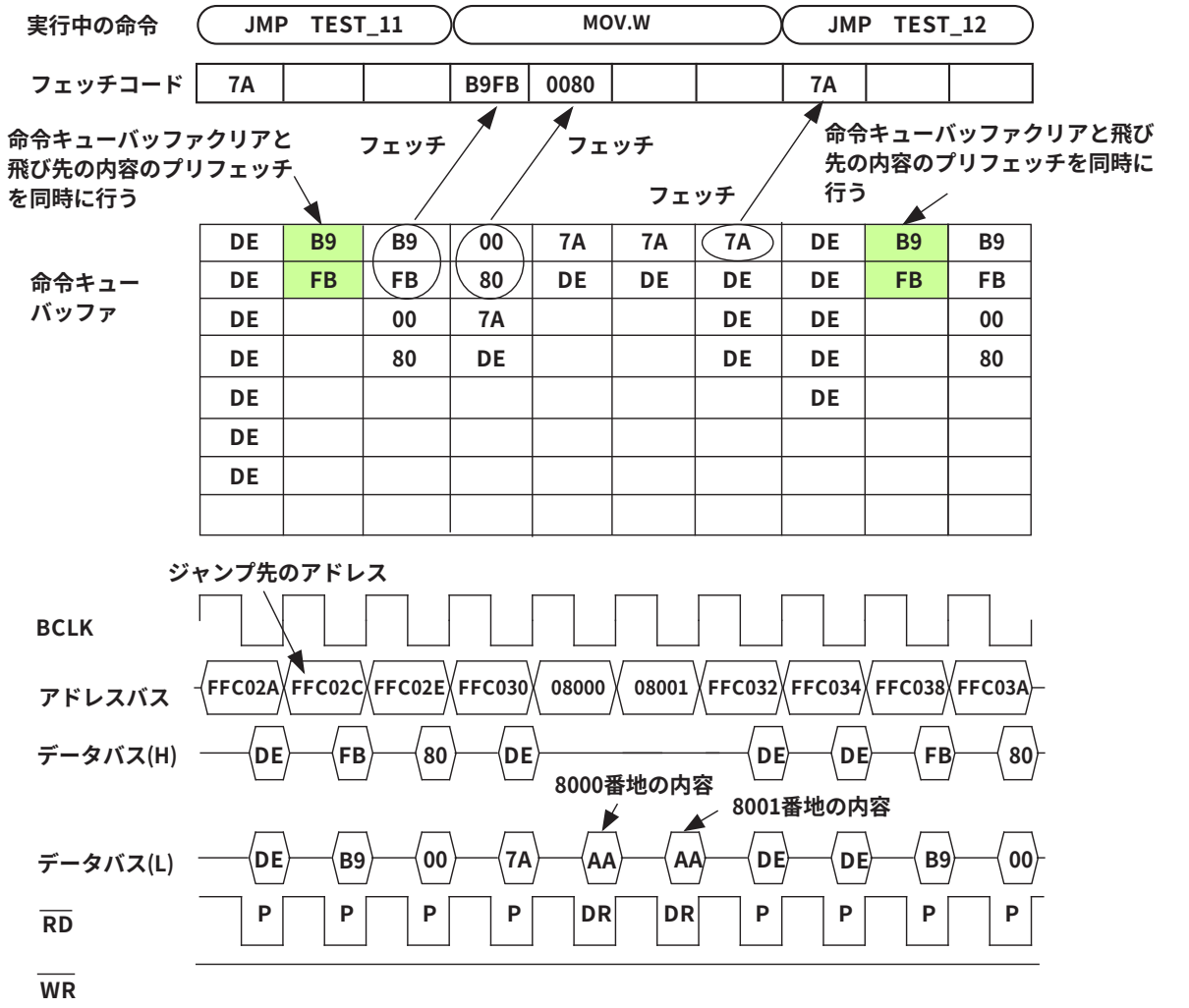


P : プリフェッチ(メモリから命令キューバッファへの取り込み)を示します。
DR : データのリードを示します。
　:命令キューバッファクリアの箇所を示します。

参考プログラム

```
アドレス   コード        命令
FFC026          TEST_10:
FFC026   7A             JMP             TEST_11
FFC027   DE             NOP
FFC028   DE             NOP
FFC029   DE             NOP
FFC02A   DE             NOP
FFC02B   DE             NOP
FFC02C   DE             NOP
FFC02D   DE             NOP
FFC02E          TEST_11:
FFC02E   B9FB0020       MOV.W:G         02000h,R1
FFC032   7A             JMP             TEST_12
FFC033   DE             NOP
FFC034   DE             NOP
FFC035   DE             NOP
FFC036   DE             NOP
FFC037   DE             NOP
FFC038   DE             NOP
FFC039   DE             NOP
FFC03A          TEST_12:
```

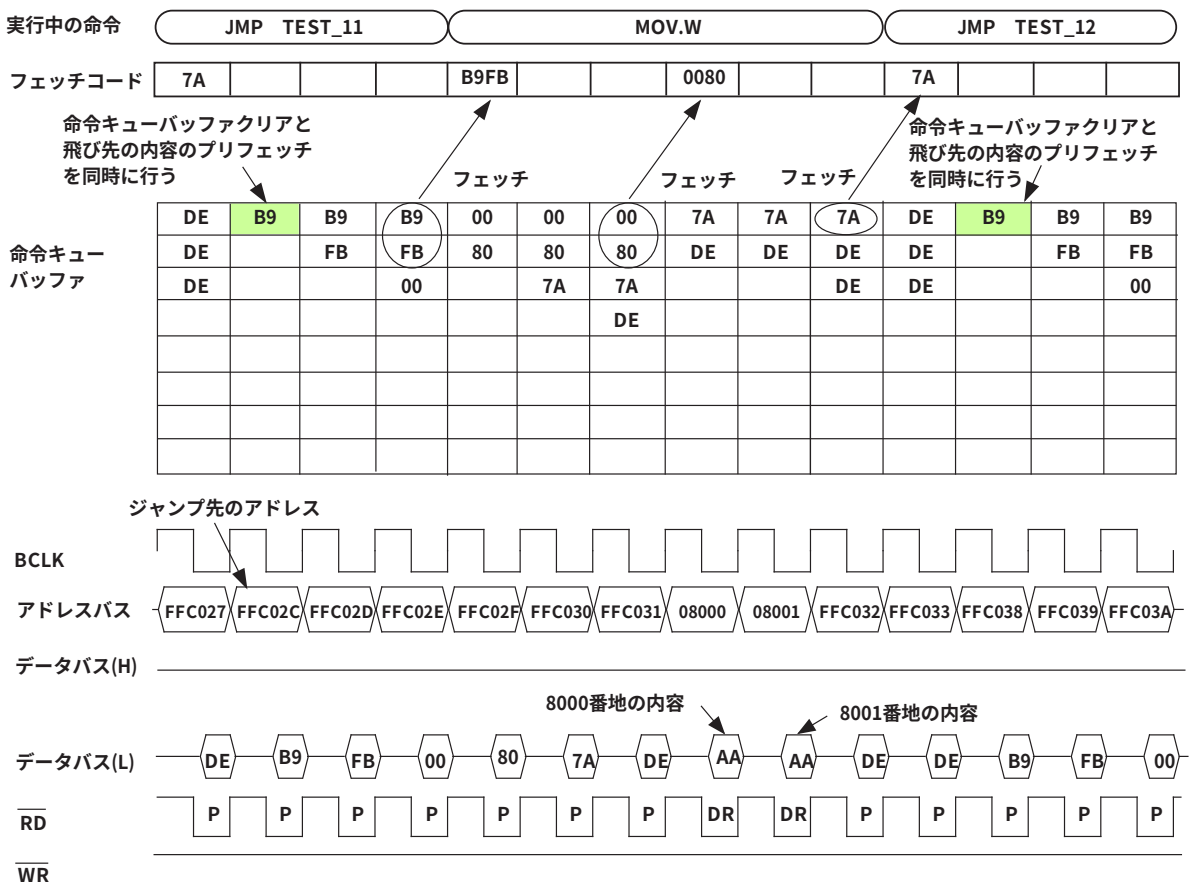
図6.1.6　偶数アドレスからの読み込み命令を偶数番地から開始する場合
(プログラム領域：16ビットバスウエイトなし、データ領域：16ビットバスウエイトあり)



P : プリフェッチ(メモリから命令キューバッファへの取り込み)を示します。
DR : データのリードを示します。
　　:命令キューバッファクリアの箇所を示します。

```
参考プログラム
アドレス　コード　　命令
FFC024　　TEST_10:
FFC024　7A　　JMP　　TEST_11
FFC025　DE　　NOP
FFC026　DE　　NOP
FFC027　DE　　NOP
FFC028　DE　　NOP
FFC029　DE　　NOP
FFC02A　DE　　NOP
FFC02B　DE　　NOP
FFC02C　　TEST_11:
FFC02C　B9FB0080　　MOV.W:G　　08000h,R1
FFC030　7A　　JMP　　TEST_12
FFC031　DE　　NOP
FFC032　DE　　NOP
FFC033　DE　　NOP
FFC034　DE　　NOP
FFC035　DE　　NOP
FFC036　DE　　NOP
FFC037　DE　　NOP
FFC038　　TEST_12:
```

図6.1.7　8ビットバスのメモリに対し読み込み命令を開始する場合
(プログラム領域：16ビットバスウエイトなし、データ領域：8ビットバスウエイトなし)



P : プリフェッチ(メモリから命令キューバッファへの取り込み)を示します。
DR : データのリードを示します。
:命令キューバッファクリアの箇所を示します。

参考プログラム

```
アドレス  コード       命令
FFC024          TEST_10:
FFC024  7A              JMP             TEST_11
FFC025  DE              NOP
FFC026  DE              NOP
FFC027  DE              NOP
FFC028  DE              NOP
FFC029  DE              NOP
FFC02A  DE              NOP
FFC02B  DE              NOP
FFC02C          TEST_11:
FFC02C  B9FB0080        MOV.W:G         08000h,R1
FFC030  7A              JMP             TEST_12
FFC031  DE              NOP
FFC032  DE              NOP
FFC033  DE              NOP
FFC034  DE              NOP
FFC035  DE              NOP
FFC036  DE              NOP
FFC037  DE              NOP
FFC038          TEST_12:
```

図6.1.8 8ビットバスのメモリに対し読み込み命令を開始する場合
(プログラム領域：8ビットバスウエイトなし、データ領域：8ビットバスウエイトなし)

Q & A

M16C ファミリ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 は正方向に相対アクセスできるのに対し、FB は正負両方向に相対アクセスできます。
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。
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。

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

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。

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。

用語集

このマニュアルソフトウエアで使用しいる用語について、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
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。	
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。	

用語	意味	関連語句
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

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
dsp24	24 ビットの変位
dsp16	16 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
EXTS(　)	(　)内の符号拡張
EXTZ(　)	(　)内のゼロ拡張
(HH)	レジスタまたはメモリの上位ワードの上位バイト（最上位バイト）
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
(HL)	レジスタまたはメモリの上位ワードの下位バイト
｜｜	絶対値
(LH)	レジスタまたはメモリの下位ワードの上位バイト
(LL)	レジスタまたはメモリの下位ワードの下位バイト（最下位バイト）
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
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 8 ビット
FLGL	フラグレジスタの下位 8 ビット
[　]	間接アドレス

索引

メ

ユ

リ

レ

ロ

ワ

vA0からvA1

p10　1.6　　割込テーブルレジスタ　00000000_{16} → 00000_{16} スタティックベースレジスタが２つあるので１つ削除
P20　図 1.10.2　周辺 I/O 割り込みが割り当てられるベクタの範囲 0-43 → 8-43
P36　下段　MOV.L　　#800000h, A03 → MOV.L　#80000000h, A0
p44　ADD【機能】追加 ・サイズ指定子(.size) に(.L)を指定した場合、dest がSP のとき、dest を32 ビットにゼロ拡張し、src を32 ビットに符号拡張し演算を行います。演算結果の下位24 ビットをdest に格納します。32 ビットの演算結果でフラグも変化します。
p49　ADJNZ, p131　SBJNZ 【選択可能な src / dest / label】 **PC^{*2} -128 ≦ label ≦ PC^{*2}+127 → PC^{*2} -126 ≦ label ≦ PC^{*2}+129**
P50 ～p66　BANDからBXOR ・Ram:16[SB]　→　Ram:19[SB]
p56、p57、p62、p63　BNOT、BNTST、BSET、BTST ・Ram:8[SB]　→　Ram
p67　CLIP【機能】追加 ・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(A0、A1)のとき、srcをゼロ拡張し16ビットで演算します。destに書き込みが生じた場合、上位8ビットは0になります。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ビットを演算の対象とします。 ・サイズ指定子(.size)に(.W)を指定した場合、destがアドレスレジスタでdestに書き込みが生じたとき、上位8ビットは0になります。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16ビットを演算の対象とします。
p70　CMPX 【記述例】 CMPX　[#5],A0　→　CMPX　#5,A0
p73　DEC 【記述例】 DEC.W　[R0]　→　DEC.W　R0
p70　DIVX 【記述例】 DIVX　[R0]　→　削除
p81　EXTS【機能】追加 ・サイズ指定子(.size)に(.B)を指定した場合、srcまたはdestを16ビットに符号拡張します。destがアドレスレジスタ(A0、A1)のとき、上位8ビットは0になります。 ・サイズ指定子(.size)に(.W)を指定した場合、destを32ビットに符号拡張します。destにR0を選択したとき上位バイトにはR2を、R1を選択したとき上位バイトにはR3を使用します。destがアドレスレジスタのとき、演算結果の下位24ビットをdestに格納します。
p81　EXTZ【機能】追加 ・destがアドレスレジスタ(A0、A1)のとき、上位8ビットは0になります。
p97　LDC【機能】追加 *1　src の下位8ビットが書き込まれます。
p100,101　MAX, MIN【機能】追加 ・src とdest ををを比較し、→・src とdest を符号付きで比較し、 ・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(A0、A1)のとき、src をゼロ拡張し16 ビットで転送します。dest に書き込みが生じた場合、上位8ビットは0になります。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。 ・サイズ指定子(.size) に(.W)を指定した場合、dest がアドレスレジスタでdest に書き込みが生じたとき、上位8ビットは0になります。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16ビットを転送します。

P102 MOV【選択可能な src / dest】 追加

*3 src または dest が dsp:8[SP]のとき、src、dest ともに間接アドレッシング[src]、[dest]は使用できません。

P103 MOV Gフォーマット

*5 src または dest が dsp:8[SP]のとき、src、dest ともに間接アドレッシング[src]、[dest]は使用できません。

追加

Sフォーマット dest 上段 R0H/R1 → R1L/R1

p129 SBJNZ 【記述例】

SBJNZ [A0],[A1],label → SBJNZ #2,[A1],label

p130 SCCnd【機能】追加

・cnd で指定された条件が真のときにdest →・*Cnd*で指定された条件が真のときにdest(1 バイト)

p131 SCMPU

ストリング転送比較→ストリング比較

【機能】変更

While
(M(A1) = M(A0) || M(A0) ≠ 0 **){**
A0 ← A0 + 1
A1 ← A1 + 1
}

While
(M(A1) = M(A0) || M(A0) ≠ 0 **){**
A0 ← A0 + 2
A1 ← A1 + 2
}

→

While (M(A1) = M(A0) || M(A0) ≠ 0 **) Do**
begin
A0 ← A0 + 1
A1 ← A1 + 1
end

While (M(A1) = M(A0) || M(A0) ≠ 0 **) Do**
begin
A0 ← A0 + 2
A1 ← A1 + 2
end

P132【選択可能な src / dest】

*3 -32 ≦ #IMM8 ≦ +32 → − 16 ≦ #IMM8 ≦ +16

p139 SMOVU

ストリング転送比較→ストリング比較

【機能】変更

While
(M(A0) ≠ 0 **){**
M(A1)← M(A0)
A0 ← A0 + 1
A1 ← A1 + 1
}

While
(M(A0) = 0 || M(A0+1) = 0 **){**
M(A1)← M(A0)
A0 ← A0 + 2
A1 ← A1 + 2
}

→

While (M(A0) ≠ 0 **) Do**
begin M(A1)← M(A0)
A0 ← A0 + 1
A1 ← A1 + 1
end

While (M(A0) = 0 || M(A0+1) = 0 **) Do**
begin M(A1)← M(A0)
A0 ← A0 + 2
A1 ← A1 + 2
end

p142 STC【機能】追加

*1 dest の上位1バイトは不定になります。

p161 インデックス命令のサイズ指定子を全て .B/.W に訂正。

４章 全面改訂

Rev.B → C
P4　2 行目 13 個のレジスタ　→　28 個のレジスタ
P5　(7)　ユーザスタックポインタ(USP)/ 割り込みスタックポインタ(ISP) USP, ISP には偶数を設定してください。偶数を設定した方が実行効率が良くなります。　→　追加
P9　図 1.5.1　注 1 レジスタバンク 1 は高速割り込み使用時は高速割り込み専用となります。　→　・・・高速割り込み用として使用してください。
P22　(2)　間接命令アドレッシング アドレスレジスタ間接間接　→　アドレスレジスタ二段間接
P44、P47、P70、P71、P104、P105、P103、P130、P149、P152、P153、P157 サイズ指定子・・・src と dest に同時にアドレスレジスタを選択できません。　→　サイズ指定子・・・src と dest に同時に A0 または A1 を選択できません。
P46、P50、P51、P115、P116 *　サイズ指定子(.size)に(.B)を指定する場合、src と dest に同時にアドレスレジスタを選択できません。 →　追加
p62　BRK, BRK2, ENTER, INT, INTO, PUSH, UND 【オペレーション】 *1　上位 8 ビットは不定になります。　追加
p82　EXITD, REIT, RTS 【オペレーション】 *1　下位 8 ビットが保存されます。　追加
p118　POPC 【オペレーション】 *1　dest が DMD0、DMD1 のとき、下位 8 ビットが保存されます。　追加 *2　下位 3 バイトが保存されます。　追加
p119　POPM 【オペレーション】 *3　dest が A0、A1、SB、FB のとき、下位 3 バイトが保存されます。
p70　CMP 【記述例】 CMP.W　　#-3,[R0]　→　CMP.W　　#-3,R0
p72　CMPX CoMPare → CoMPare eXtend sign 比較→符号拡張比較
P76 ～ P78 【機能】 ・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。　→　追加 ・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。　→　追加
p89　【記述例】 INDEXB　　→　INDEXB.W INDEXLS　　→　INDEXLS.B
p95　JMPS 【機能】

	・スペシャルページベクタテーブルは FFFE00$_{16}$ 番地から FFFFDA$_{16}$ → ・スペシャルページベクタテーブルは FFFE00$_{16}$ 番地から FFFFDB$_{16}$
p98	JSRS 【機能】 ・スペシャルページベクタテーブルは FFFE00$_{16}$ 番地から FFFFDA$_{16}$ → ・スペシャルページベクタテーブルは FFFE00$_{16}$ 番地から FFFFDB$_{16}$
p133	SCMPU 【オペレーション】 **If** M(A0)＝ M(A1) **then** M(A0 ＋ 1)－ M(A1 ＋ 1) → **If** M(A0)＝ M(A1) **and** M(A0)≠ 0 **then** M(A0 ＋ 1)－ M(A1 ＋ 1)
p135	SHA 条件 O → シフト結果の MSB とシフトアウトしたビットが全て同じ値のとき“ 0 ”、1 つでも異なるとき“ 1 ”になります。
p138	SIN, p139 SMOVB, p140 SMOVF, p142 SOUT, p143 SSTR 【オペレーション】 Repeat、Until ... の削除
p144	STC 【選択可能な src/dest】 #IMM16,#IMM24 削除
p154	UND 【オペレーション】 M(SP)← (PC + 1)ML　　M(SP)← (PC + 1)H SP ← SP － 2　→　SP ← SP － 2 M(SP)← (PC + 1)H　　M(SP)← (PC + 1)L
p144	STC 【選択可能な src/dest】 #IMM16,#IMM24 削除
p165	long mem1[]; → ling mem1;
p166	long mem2[]; → ling mem2;
p167	BITINDEX 命令がモディファイする命令 BTST → BTST:G*1 *1 Gフォーマットのみ指定できます。
p168	INDEXBD.B/.W*2 SCcnd 削除 BITINDEX.B/.W BTST → BTST:G BTST → BTSTC
4 章	4.2 命令コード / サイクル数 【バイト数 / サイクル数】 dsp:24[SB/FB]を全て削除
p229	Jcnd dsp8=label が示す番地－(命令の先頭番地＋ 1) 追加
p233	JSR dsp16=label が示す番地－(命令の先頭番地＋ 1) 追加

p245 (5) MOV.size:S

*1, *2 削除

p247 (7) MOV.size:G src, dest

【バイト数／サイクル数】

src \ dest	Rn	An	[An]
Rn	2/1	2/1	2/1
An	2/1	2/1	2/1
[An]	2/2	2/2	2/2
dsp:8[An]	3/2	3/2	3/2
dsp:8[SB/FB]	3/2	3/2	3/2
dsp:16[An]	4/2	4/2	4/2
dsp:16[SB/FB]	4/2	4/2	4/2
dsp:24[An]	5/2	5/2	5/2
dsp:24[SB/FB]	5/2	5/2	5/2
abs16	4/2	4/2	4/2
abs24	5/2	5/2	5/2

→

src \ dest	Rn	An	[An]
Rn	2/1	2/1	2/1
An	2/1	2/1	2/1
[An]	2/3	2/3	2/3
dsp:8[An]	3/3	3/3	3/3
dsp:8[SB/FB]	3/3	3/3	3/3
dsp:16[An]	4/3	4/3	4/3
dsp:16[SB/FB]	4/3	4/3	4/3
dsp:24[An]	5/3	5/3	5/3
dsp:24[SB/FB]	5/3	5/3	5/3
abs16	4/3	4/3	4/3
abs24	5/3	5/3	5/3

p249 (10) MOV.size:S src, R1L/R1

【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	2/2	3/2

→

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

p250 (12) MOV.L:S src, A0/A1

*1 src が間接アドレッシングのとき、コードの先頭に 00001001 が付きます。 追加

p268 (1) PUSHM src

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

→

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

【バイト数／サイクル数】

1/m → 2/m

P269 (1) RMPA.size

*1 mは回転回数です。 → *1 mは演算回数です。

p276 dest An

--- → ---/A0/---

--- → ---/A1/---

P277 (1) SCMPU.size

【バイト数／サイクル数】

サイズ指定子	バイト数/サイクル数	
	一致で終了	不一致で終了
.B	2/6+m	2/6+m
.W	2/6+m	2/9+m
.W	2/8+m	2/10+m

*1 mは転送回数です。

→

サイズ指定子	バイト数/サイクル数	
	一致で終了	不一致で終了
.B	2/6+3m	2/6+3m
.W	2/6+1.5m	2/9+1.5m
.W	2/8+1.5m	2/10+1.5m

*1 mは比較回数です。

５章 割り込み追加

Rev.C → D

P5　(9)　フラグ退避レジスタ
フラグ退避レジスタ(SVF)は 24 ビットで・・・　→　フラグ退避レジスタ(SVF)は 16 ビットで・・・

P10
・フラグ退避レジスタ(SVF)　：不定
・レジスタ退避レジスタ(SVP)　：不定
・ベクタレジスタ(VCT)　：不定　→　追加

P69　CLIP
【機能】
・src1 と src2 には、src1 < src2 となるよう設定してください。　→　追加

P99　LDC
【機能】
4 バイト*2：FB、SB、SP*3、ISP*3、INTB、・・・　→　4 バイト*2：FB、SB、SP*3、ISP*3、INTB*3、・・・
*3　奇数も設定できますが、SP、ISP には偶数を設定してください。　→　*3　奇数も設定できますが、SP、ISP、INTB には偶数を設定してください。

P118　POPC
【オペレーション】
SP　←　SP　＋　4　→　SP*3　←　SP　＋　4
*3　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 4 加算されません。　追加

P122　PUSH
【オペレーション】
M(SP)*2　←　src　→　M(SP)*2　←　src*3
*3　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、4 を減算される前の SP が退避されます。　追加

p149　SUB
【機能】
また src がアドレスレジスタのとき src をゼロ拡張し、32 ビットで演算を行います。　追加

p229　JMP
dsp=labal が示す番地－(命令の先頭番地＋ 2)　→　削除

【バイト数 / サイクル数】

バイト数/サイクル数	1/4

→

バイト数/サイクル数	1/3

6 章　サイクル数の計算　追加

1999-10-18

p120　PUSH
【オペレーション】
*2　src がアドレスレジスタ(A0、A1)のとき、上位 8 ビットは不定となります。　→　0 となります。

p234　(2)JSRI.A
d4 d3 d2 d1 d0　→　s4 s3 s2 s1 s0

Rev.D → D1　2000-2-9

p304　(2) オーバフロー割り込み
CMPX 命令追加

Rev.D1 → 1.0　2002-2-14

M16C/70 シリーズが追加

早見表 -1
　　JO →追加

早見表 -4
　　CPMX → CMPX

p4　図 1.3.1
　　スタティクベースレジスタ→スタティックベースレジスタ

p18　7 行目
　　ジェネリック形式のオペコードは 2 バイトです。→ジェネリック形式のオペコードは 2 ～ 3 バイトです。

p62　BRK
　　【オペレーション】
　　FFFFE4$_{16}$ 番地から FFFFE7$_{16}$ 番地に FF$_{16}$ 以外がある場合→ FFFFE7$_{16}$ 番地が FF$_{16}$ 以外の場合
　　FFFFE4$_{16}$ 番地から FFFFE7$_{16}$ 番地がすべて FF$_{16}$ の場合→ FFFFE7$_{16}$ 番地が FF$_{16}$ の場合

p131　SBJNZ
　　【機能】
　　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 16 ビットを演算の対象とします。→削除

p136　SHL
　　【選択可能な src / dest】
　　~~/#IMM8~~ → /IMM8

p141　SMOVU
　　【オペレーション】 8 行目
　　tmp → tmp0

p195　BRK
　　【バイト数／サイクル数】
　　このとき、FFFFE4$_{16}$ 番地～ FFFFE7$_{16}$ 番地には FF$_{16}$ を･･･→このとき、FFFFE7$_{16}$ 番地には FF$_{16}$ を･･･

p249　MOV (10)
　　【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	1/3	2/3	3/3

→

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

三菱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
M16C/80, M16C/70 シリーズ　Rev. 1.0

発行所　三菱電機株式会社半導体営業統括部
〒100-8310　東京都千代田区丸の内2-2-3
TEL　03-3218-9450

M16C/80、M16C/70 グループ
ソフトウエアマニュアル

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