

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ひ申し上げます。

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情報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。お客様にかかる法令を遵守しないことにより生じた損害に関し、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。

H8/300Hシリーズ

プログラミングマニュアル

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

H8 ファミリ

安全設計に関するお願い

1. 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

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (<http://www.renesas.com>)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したのですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任を負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

はじめに

H8/300H シリーズは、内部 32 ビット構成の H8/300H CPU をコアとしています。H8/300H CPU は、16 ビット×16 本の汎用レジスタと高速動作を指向した簡潔で最適化された命令セットを備え、16M バイトのリニアなアドレス空間を扱うことができます。命令は、H8/300 シリーズとオブジェクトレベルで上位互換を保っており、容易に H8/300 シリーズから移行できます。また、高級言語 C で書かれたプログラムも効率的に実行できます。

本マニュアルは、H8/300H CPU の命令の詳細について記載しており、H8/300H シリーズ共通に使用することができます。

なお、ハードウェアの詳細については、当該 LSI のハードウェアマニュアルをご覧ください。

目次

第1章 CPU

1.1	概要	1
1.1.1	特長	1
1.1.2	H8/300CPU との相違点	2
1.2	CPU 動作モード	2
1.3	アドレス空間	5
1.4	レジスタ構成	6
1.4.1	概要	6
1.4.2	汎用レジスタ	6
1.4.3	コントロールレジスタ	7
1.4.4	CPU 内部レジスタの初期値	8
1.5	データ構成	9
1.5.1	汎用レジスタのデータ構成	9
1.5.2	メモリ上でのデータ構成	10
1.6	命令セット	10
1.6.1	概要	10
1.6.2	命令とアドレッシングモードの組合せ	11
1.6.3	命令の機能別一覧	12
1.6.4	命令の基本フォーマット	17
1.7	アドレッシングモードと実効アドレスの計算方法	18
1.7.1	アドレッシングモード	18
1.7.2	実効アドレスの計算方法	21

第2章 各命令の説明

2.1	表と記号の説明	25
2.1.1	アセンブラフォーマット	25
2.1.2	オペレーション	26
2.1.3	コンディションコード	27
2.1.4	インストラクションフォーマット	27
2.1.5	レジスタの指定方法	27
2.1.6	ビット操作命令におけるビットデータのアクセス方法	28
2.2	各命令の説明	29
2.3	命令セット一覧	164
2.3.1	命令とアドレッシングモードの組合せ	164
2.3.2	命令セット一覧	165

2.4	命令コード一覧	177
2.5	オペレーションコードマップ	185
2.6	命令実行ステート数	188
2.7	コンディションコードの変化	193
2.8	命令実行中のバス状態	197

第3章 処理状態

3.1	概要	207
3.2	プログラム実行状態	208
3.3	例外処理状態	208
3.3.1	例外処理の種類と優先度	208
3.3.2	例外処理の動作	209
3.4	バス権解放状態	211
3.5	リセット状態	211
3.6	低消費電力状態	211
3.6.1	スリープモード	211
3.6.2	ソフトウェアスタンバイモード	211
3.6.3	ハードウェアスタンバイモード	211

第4章 基本動作タイミング

4.1	概要	213
4.2	内蔵メモリ (RAM、ROM)	213
4.3	内蔵周辺モジュールアクセスタイミング	214
4.4	外部アドレス空間アクセスタイミング	216

1. CPU

1.1 概要

H8/300H CPU は、H8/300CPU の上位互換のアーキテクチャを持つ内部 32 ビット構成の高速 CPU です。H8/300H CPU は、16 ビット×16 本の汎用レジスタを持ち、16M バイトのリニアなアドレス空間を扱うことができ、リアルタイム制御に最適です。

1.1.1 特長

H8/300H CPU には、次の特長があります。

- H8/300CPU の上位互換
H8/300オブジェクトプログラムを実行可能
- 汎用レジスタ方式
16ビット×16本（8ビット×16本、32ビット×8本としても使用可能）
- 62 種類の基本命令
8/16/32ビット演算命令
乗除算命令
強力なビット操作命令
- 8 種類のアドレッシングモード
レジスタ直接（Rn）
レジスタ間接（@ERn）
ディスプレイメント付レジスタ間接（@（d:16,ERn）／@（d:24,ERn））
ポストインクリメント／プリデクリメントレジスタ間接（@ERn+／@-ERn）
絶対アドレス（@aa:8／@aa:16／@aa:24）
イミディエイト（#xx:8／#xx:16／#xx:32）
プログラムカウンタ相対（@（d:8,PC）／@（d:16,PC））
メモリ間接（@@aa:8）
- 16Mバイトのアドレス空間
- 高速動作
頻出命令をすべて2～4ステートで実行
最高動作周波数:20MHzの場合
8/16/32ビットレジスタ間加減算 100ns
8×8ビットレジスタ間乗算 700ns
16÷8ビットレジスタ間除算 700ns
16×16ビットレジスタ間乗算 1100ns
32÷16ビットレジスタ間除算 1100ns
- 2 種類の CPU 動作モード
ノーマルモード／アドバンスモード
- 低消費電力状態
SLEEP命令により低消費電力状態に遷移

1. CPU

1.1.2 H8/300CPU との相違点

H8/300H CPU は、H8/300CPU に対して、次の点が追加、拡張されています。

- 汎用レジスタを拡張
16ビット×8本の拡張レジスタを追加
- アドレス空間を拡張
ノーマルモードのとき、H8/300CPUと同一の64kバイトのアドレス空間を使用可能
アドバンスモードのとき、最大16Mバイトのアドレス空間を使用可能
- アドレッシングモードを強化
16Mバイトのアドレス空間を有効に使用可能
- 命令強化
符号付き乗除算命令などを追加
32ビット転送、演算命令を追加

1.2 CPU 動作モード

H8/300H CPU は、ノーマルモードおよびアドバンスモードの2つのCPU動作モードを持っています。サポートするアドレス空間は、ノーマルモードの場合最大64kバイト、アドバンスモードの場合最大16Mバイトとなります。

各モードはLSIのモード端子によって選択されます。詳細は当該LSIのハードウェアマニュアルを参照してください。

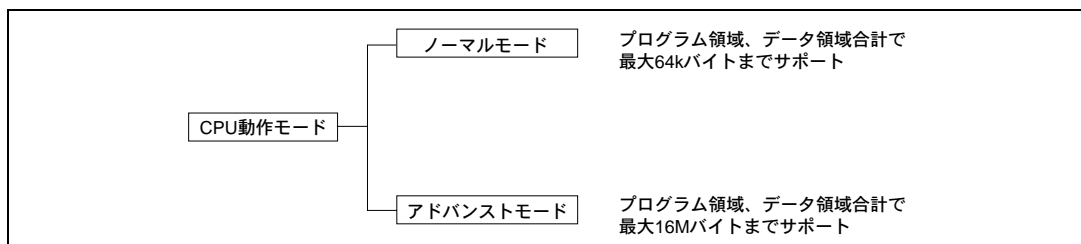


図 1.1 CPU 動作モード

(1) ノーマルモード

ノーマルモードでは例外処理ベクタ、スタックの構造がH8/300CPUと同一になります。

(a) アドレス空間

H8/300CPUと同様、最大64kバイトをアクセス可能です。

(b) 拡張レジスタ (En)

拡張レジスタ (E0～E7) は、16ビットレジスタとして、または32ビットレジスタの上位16ビットとして使用できます。

拡張レジスタ En は、対応する汎用レジスタ Rn をアドレスレジスタとして使用している場合でも、16ビットレジスタとして任意の値を設定することができます (ただし、プリデクリメントレジスタ間接 (@-Rn)、ポストインクリメントレジスタ間接 (@Rn+) により汎用レジスタ Rn が参照された場合、キャリ/ボローが発生すると、対応する拡張レジスタ En の内容に伝播しますので注意してください)。

(c) 命令セット

H8/300CPU に対して追加された命令およびアドレッシングモードはすべて使用できます。実効アドレス (EA) の下位16ビットのみが有効となります。

(d) 例外処理ベクタテーブルおよびメモリ間接の分岐アドレス

ノーマルモードでは、H'0000 から始まる先頭領域に例外処理ベクタテーブル領域が割り当てられており、各 16 ビットの分岐先アドレスを格納します。ノーマルモードの例外処理ベクタテーブルの構造を図 1.2 に示します。例外処理ベクタテーブルは各製品ごとに異なりますので、詳細は当該 LSI のハードウェアマニュアルを参照してください。

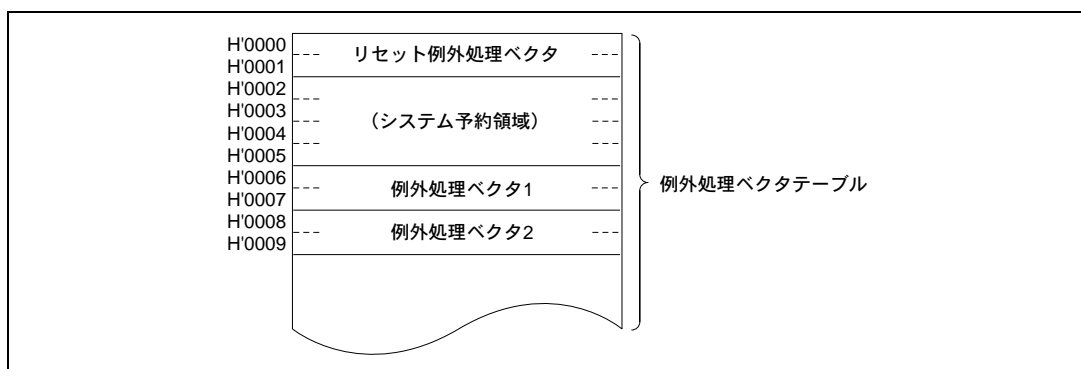


図 1.2 例外処理ベクタテーブル（ノーマルモード）

メモリ間接 (@@aa:8) は、JMP および JSR 命令で使用されます。命令コードに含まれる 8 ビット絶対アドレスによりメモリ上のオペランドを指定し、この内容が分岐先アドレスとなります。

ノーマルモードでは、オペランドは 16 ビット（ワード）となり、この 16 ビットが分岐先アドレスとなります。なお、分岐先アドレスを格納できるのは、H'0000～H'00FF の領域であり、この範囲の先頭領域は例外処理ベクタテーブルと共通となっていますので注意してください。

(e) スタック構造

サブルーチン分岐時の PC スタック構造と、例外処理時の PC と CCR のスタックの構造を図 1.3 に示します。

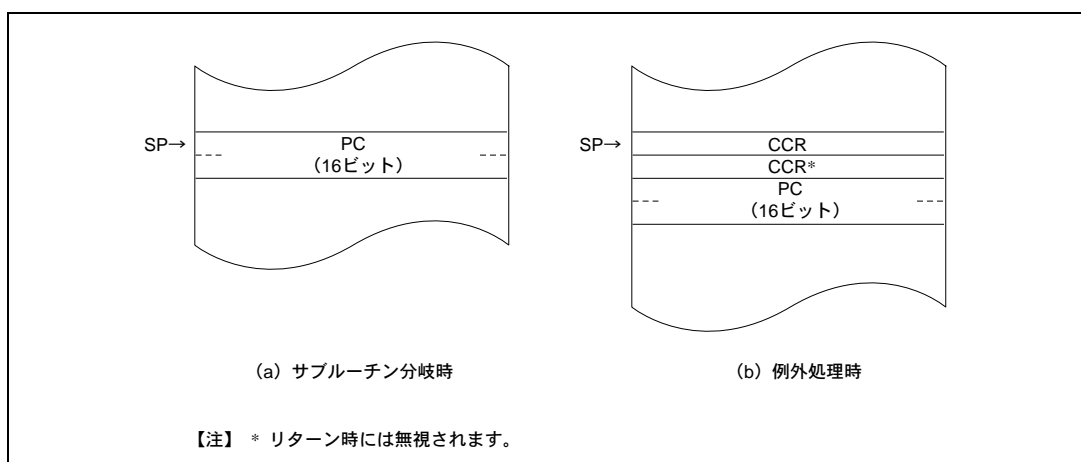


図 1.3 ノーマルモードのスタック構造

1. CPU

(2) アドバンストモード

(a) アドレス空間

最大 16M バイトをリニアにアクセス可能です。

(b) 拡張レジスタ (En)

拡張レジスタ (E0~E7) は、16 ビットレジスタとして、または 32 ビットレジスタ・アドレスレジスタの上位 16 ビットとして使用できます。

(c) 命令セット

命令およびアドレッシングモードはすべて使用できます。

(d) 例外処理ベクタテーブル、メモリ間接の分岐アドレス

アドバンストモードでは、H'000000 から始まる先頭領域に 32 ビット単位で例外処理ベクタテーブル領域が割り当てられており、上位 8 ビットは無視され 24 ビットの分岐先アドレスを格納します (図 1.4 参照)。例外処理ベクタテーブルは各製品ごとに異なりますので、詳細は当該 LSI のハードウェアマニュアルを参照してください。

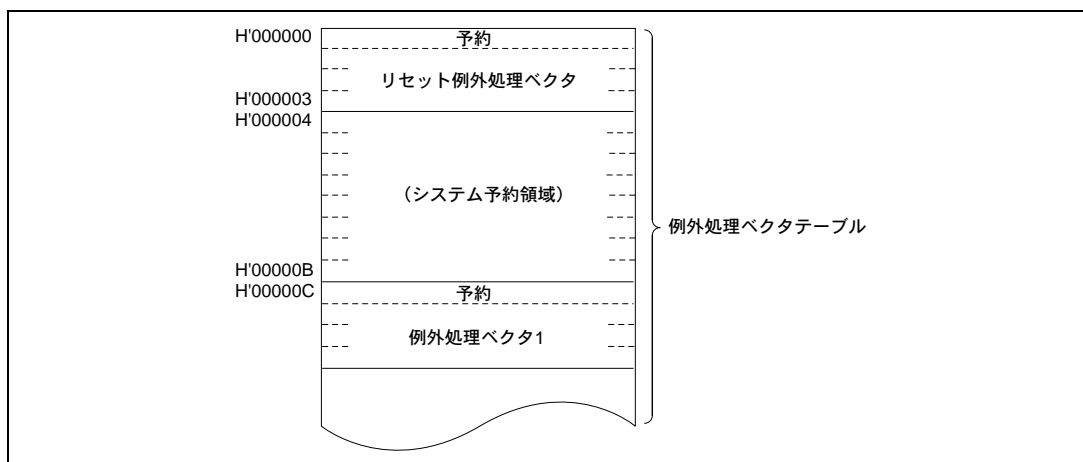


図 1.4 例外処理ベクタテーブル (アドバンストモード)

メモリ間接 (@@aa:8) は、JMP および JSR 命令で使用されます。命令コードに含まれる 8 ビット絶対アドレスによるメモリ上のオペランドを指定し、この内容が分岐先アドレスとなります。

アドバンストモードでは、オペランドは 32 ビット (ロングワード) となり、この 32 ビットの下位 24 ビットが分岐先アドレスとなります。なお、分岐先アドレスを格納できるのは、H'000000~H'0000FF の領域であり、この範囲の先頭領域は例外処理ベクタテーブルと共通となっていますので注意してください。

(e) スタック構造

アドバンストモード時のサブルーチン分岐時の PC のスタック構造と、例外処理時の PC と CCR のスタックの構造を図 1.5 に示します。

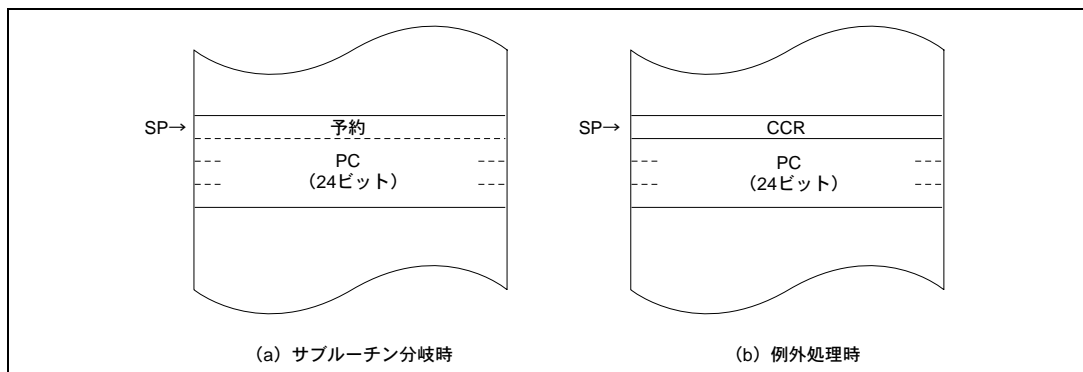


図 1.5 アドバンストモードのスタック構造

1.3 アドレス空間

H8/300H CPU のメモリマップを図 1.6 に示します。H8/300H CPU は、ノーマルモードのとき最大 64k バイト、またはアドバンストモードのとき最大 16M バイトのアドレス空間をリニアに使用することができます。

アドレス空間は動作モードなどによって異なります。詳細は当該 LSI のハードウェアマニュアルを参照してください。

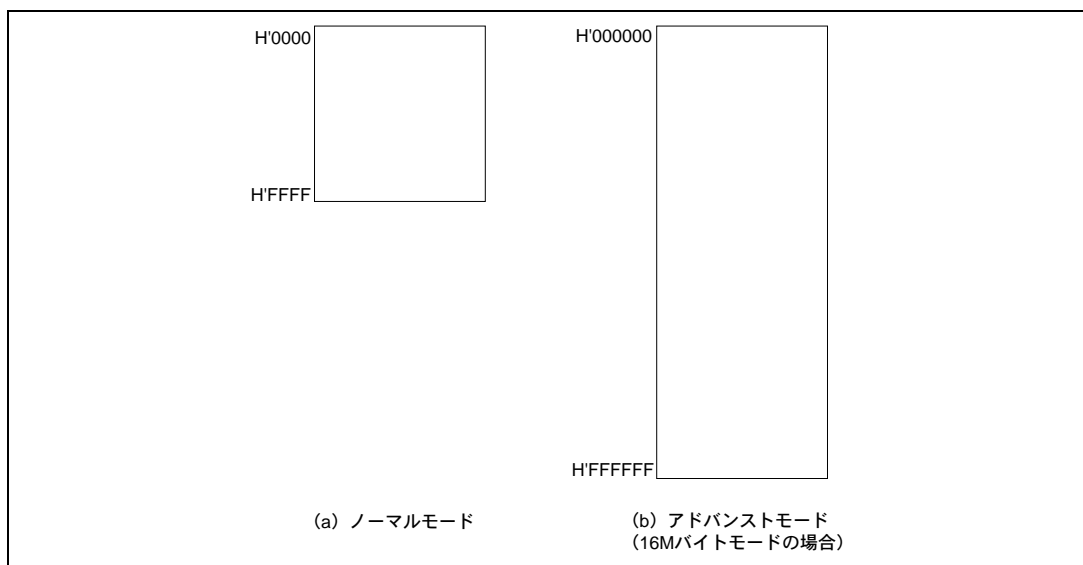


図 1.6 メモリマップ

1.4 レジスタ構成

1.4.1 概要

H8/300H CPU の内部レジスタ構成を図 1.7 に示します。これらのレジスタは、汎用レジスタとコントロールレジスタの 2 つに分類することができます。

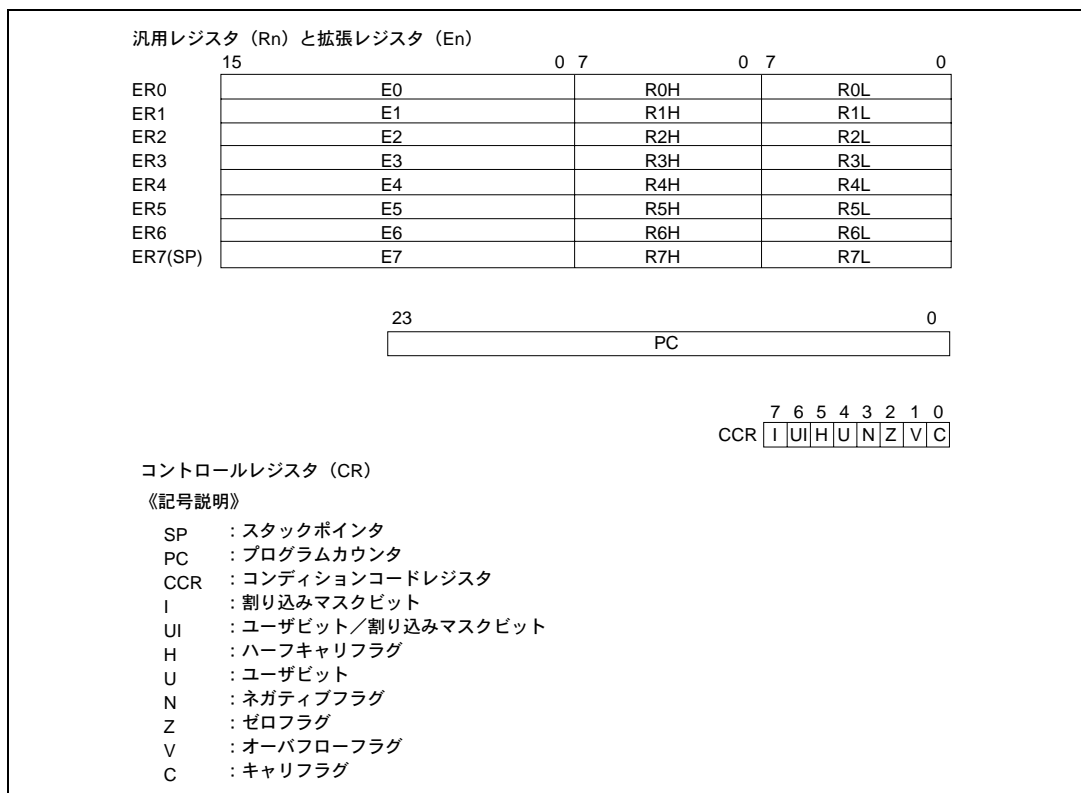


図 1.7 CPU 内部レジスタ構成

1.4.2 汎用レジスタ

H8/300H CPU は、32 ビット長の汎用レジスタを 8 本持っています。汎用レジスタは、すべて同じ機能を持っており、アドレスレジスタとしてもデータレジスタとしても使用することができます。データレジスタとしては 32 ビット、16 ビットおよび 8 ビットレジスタとして使用できます。

アドレスレジスタおよび 32 ビットレジスタとしては、一括して汎用レジスタ ER (ER0~ER7) として使用します。

16 ビットレジスタとしては、汎用レジスタ ER を分割して汎用レジスタ E (E0~E7)、汎用レジスタ R (R0~R7) として使用します。これらは同等の機能を持っており、16 ビットレジスタを最大 16 本まで使用することができます。なお、汎用レジスタ E (E0~E7) を、特に拡張レジスタと呼ぶ場合があります。

8 ビットレジスタとしては、汎用レジスタ R を分割して汎用レジスタ RH (R0H~R7H)、汎用レジスタ RL (R0L~R7L) として使用します。これらは同等の機能を持っており、8 ビットレジスタを最大 16 本まで使用することができます。

汎用レジスタの使用方法を図 1.8 に示します。各レジスタ独立に使用方法を選択することができます。

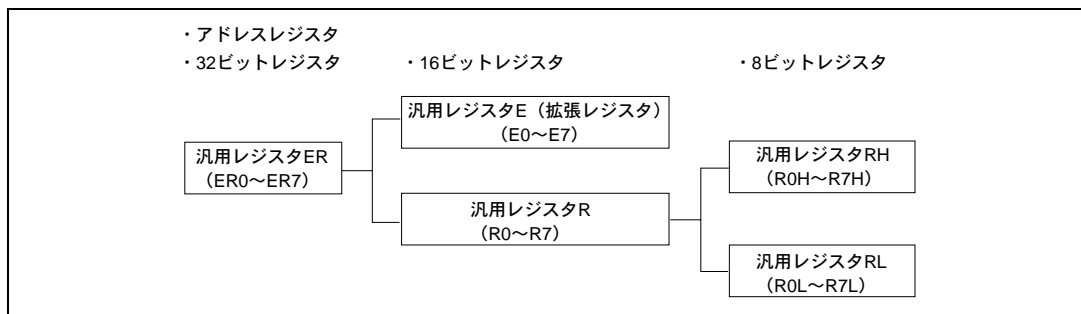


図 1.8 汎用レジスタの使用方法

汎用レジスタ ER7 には、汎用レジスタとしての機能に加えて、スタックポインタ (SP) としての機能が割り当てられており、例外処理やサブルーチン分岐などで暗黙的に使用されます。スタックの状態を図 1.9 に示します。

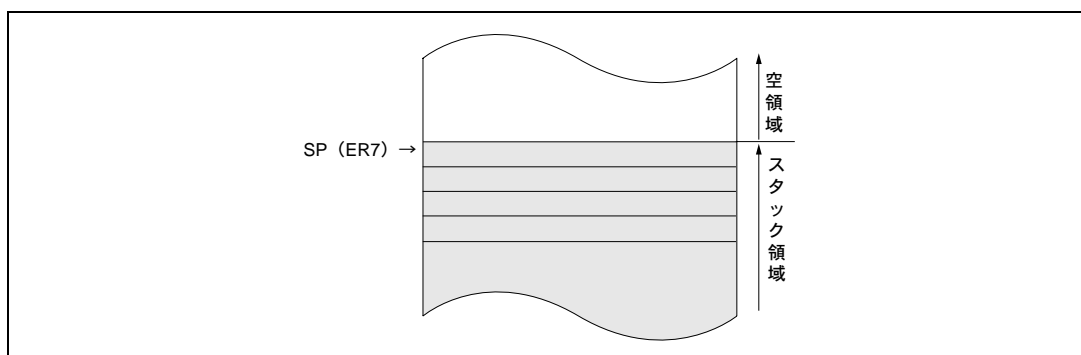


図 1.9 スタックの状態

1.4.3 コントロールレジスタ

コントロールレジスタには、24 ビットのプログラムカウンタ (PC) と 8 ビットのコンディションコードレジスタ (CCR) があります。

(1) プログラムカウンタ (PC)

24 ビットのカウンタで、CPU が次に実行する命令のアドレスを示しています。CPU の命令は、すべて 2 バイト (ワード) を単位としているため、最下位ビットは無効です (命令コードのリード時には最下位ビットは 0 とみなされます)。

(2) コンディションコードレジスタ (CCR)

8 ビットのレジスタで、CPU の内部状態を示しています。割り込みマスクビット (I) とハーフキャリ (H)、ネガティブ (N)、ゼロ (Z)、オーバフロー (V)、キャリ (C) の各フラグを含む 8 ビットで構成されています。

ビット 7 : 割り込みマスクビット (I)

本ビットが 1 にセットされると、割り込みがマスクされます。ただし、NMI は I ビットに関係なく受け付けられます。例外処理の実行が開始されたときに 1 にセットされます。

ビット 6 : ユーザビット / 割り込みマスクビット (UI)

ソフトウェア (LDC、STC、ANDC、ORC、XORC 命令) でリード / ライトできます。割り込みマスクビットとしても使用可能です。詳細は当該 LSI のハードウェアマニュアルを参照してください。

ビット5：ハーフキャリフラグ (H)

ADD.B、ADDX.B、SUB.B、SUBX.B、CMP.B、NEG.B 命令の実行により、ビット3にキャリまたはボローが生じたとき1にセットされ、生じなかったとき0にクリアされます。また、ADD.W、SUB.W、CMP.W、NEG.W 命令の実行により、ビット11にキャリまたはボローが生じたとき、ADD.L、SUB.L、CMP.L、NEG.L 命令の実行により、ビット27にキャリまたはボローが生じたとき1にセットされ、生じなかったとき0にクリアされます。

ビット4：ユーザビット (U)

ソフトウェア (LDC、STC、ANDC、ORC、XORC 命令) でリード/ライトできます。

ビット3：ネガティブフラグ (N)

データの最上位ビットを符号ビットとみなし、最上位ビットの値を格納します。

ビット2：ゼロフラグ (Z)

データがゼロのとき1にセットされ、ゼロ以外のとき0にクリアされます。

ビット1：オーバフローフラグ (V)

算術演算命令の実行により、オーバフローが生じたとき1にセットされます。それ以外のとき0にクリアされます。

ビット0：キャリフラグ (C)

演算の実行により、キャリが生じたとき1にセットされ、生じなかったとき0にクリアされます。キャリには次の種類があります。

- (a) 加算結果のキャリ
- (b) 減算結果のボロー
- (c) シフト/ローテートのキャリ

また、キャリフラグには、ビットアキュムレータ機能があり、ビット操作命令で使用されます。

なお、命令によってはフラグが変化しない場合があります。

各命令ごとのフラグの変化については、2.2.1以降の各命令の説明を参照してください。

CCR は、LDC、STC、ANDC、ORC、XORC 命令で操作することができます。また、N、Z、V、Cの各フラグは、条件分岐命令 (Bcc) で使用されます。

1.4.4 CPU 内部レジスタの初期値

リセット例外処理によって、CPU 内部レジスタのうち、PC はベクタからロードすることにより初期化され、CCR の I ビットは 1 にセットされますが、汎用レジスタと CCR の他のビットは初期化されません。SP (ER7) の初期値も不定です。したがって、リセット直後に、MOV.L 命令を使用して SP を初期化してください。

1.5 データ構成

H8/300H CPUは、1ビット、4ビットBCD、8ビット（バイト）、16ビット（ワード）、および32ビット（ロングワード）のデータを扱うことができます。

1ビットデータはビット操作命令で扱われ、オペランドデータ（バイト）の第nビット（n=0、1、2、…、7）という形式でアクセスされます。

なお、DAA および DAS の10進補正命令では、バイトデータは2桁の4ビットBCDデータとなります。

1.5.1 汎用レジスタのデータ構成

汎用レジスタのデータ構成を図 1.10 に示します。

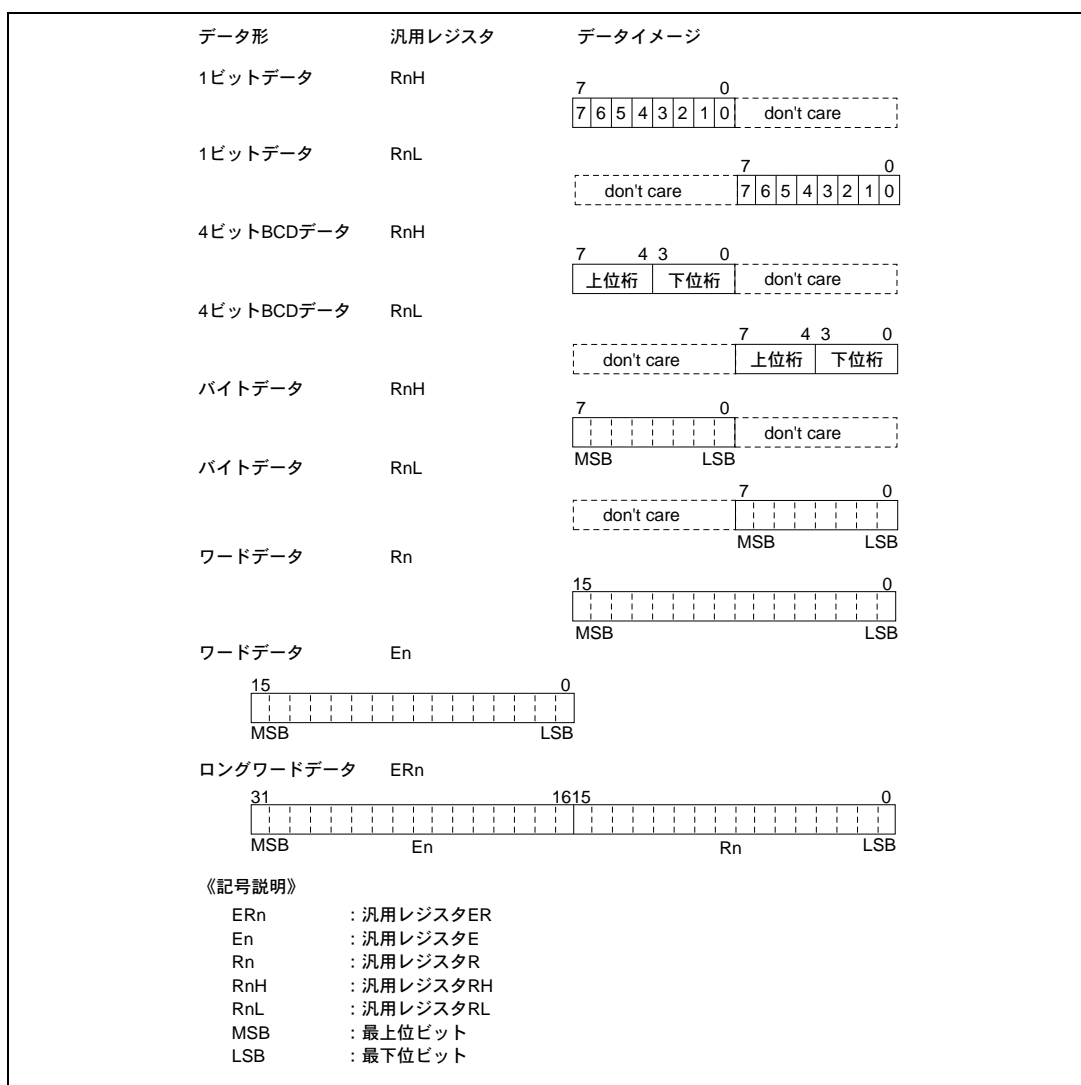


図 1.10 汎用レジスタのデータ構成

1.5.2 メモリ上でのデータ構成

メモリ上でのデータ構成を図 1.11 に示します。

H8/300H CPU は、メモリ上のワードデータ／ロングワードデータをアクセスすることができます。これらは、偶数番地から始まるデータに限定されます。奇数番地から始まるワードデータ／ロングワードデータをアクセスした場合、アドレスの最下位ビットは 0 とみなされ、1 番地前から始まるデータをアクセスします。この場合、アドレスエラーは発生しません。命令コードについても同様です。

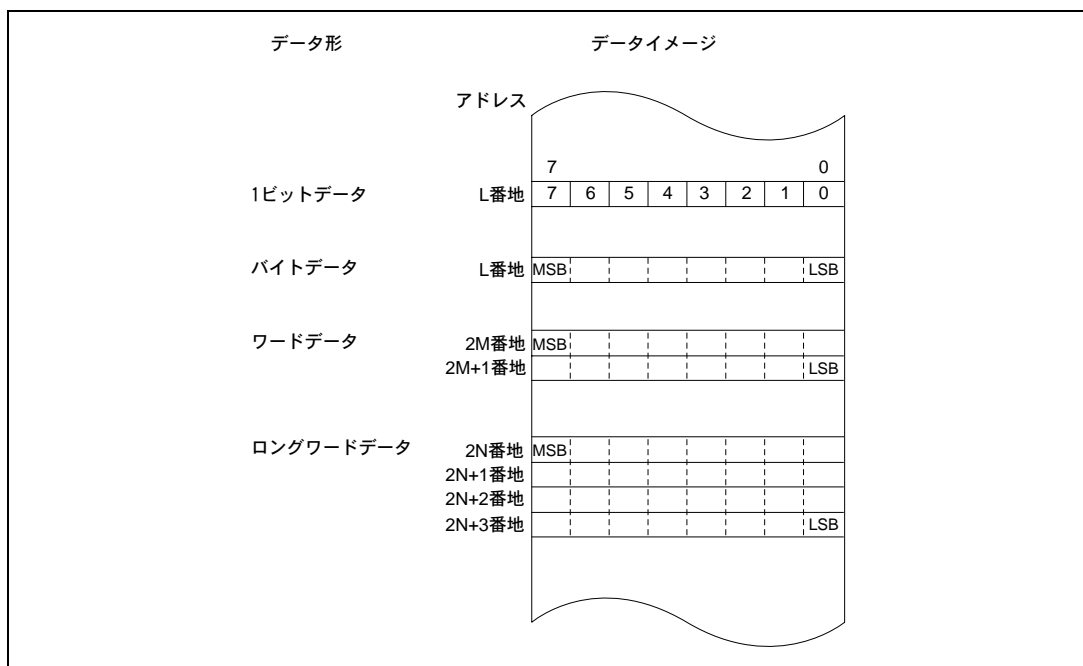


図 1.11 メモリ上でのデータ構成

なお、SP (ER7) をアドレスレジスタとしてスタックをアクセスするときは、必ずワードサイズまたはロングワードサイズでアクセスしてください。

1.6 命令セット

1.6.1 概要

H8/300H CPU の命令は合計 62 種類あり、各命令のもつ機能によって表 1.1 に示すように分類されます。各命令についての詳細は「2.2 各命令の説明」を参照してください。

表 1.1 命令の分類

機能	命令	種類
データ転送命令	MOV、PUSH* ¹ 、POP* ¹ 、MOVTPE、MOVFPPE	3
算術演算命令	ADD、SUB、ADDX、SUBX、INC、DEC、ADDS、SUBS、DAA、DAS、MULXU、MULXS、DIVXU、DIVXS、CMP、NEG、EXTS、EXTU	18
論理演算命令	AND、OR、XOR、NOT	4
シフト命令	SHAL、SHAR、SHLL、SHLR、ROTL、ROTR、ROTXL、ROTXR	8
ビット操作命令	BSET、BCLR、BNOT、BTST、BAND、BIAND、BOR、BIOR、BXOR、BIXOR、BLD、BILD、BST、BIST	14
分岐命令	Bcc* ² 、JMP、BSR、JSR、RTS	5
システム制御命令	TRAPA、RTE、SLEEP、LDC、STC、ANDC、ORC、XORC、NOP	9
ブロック転送命令	EEPMOV	1

合計 62 種類

【注】 ■ : H8/300H CPU で追加された命令

- *1 POP.W Rn、PUSH.W Rn は、それぞれ MOV.W @SP+,Rn、MOV.W Rn,@-SP と同一です。
また、POP.L ERn、PUSH.L ERn は、それぞれ MOV.L @SP+,Rn、MOV.L Rn,@-SP と同一です。
- *2 Bcc は条件分岐命令の総称です。

1.6.2 命令とアドレッシングモードの組合せ

H8/300H CPU で使用できる命令とアドレッシングモードの組合せを表 1.2 に示します。

1. CPU

表 1.2 命令とアドレッシングモードの組合せ

機能	命令	アドレッシングモード												
		# xx	Rn	@ERn	@(d:16,ERn)	@(d:24,ERn)	@-ERn/@ERn+	@aa:8	@aa:16	@aa:24	@(d:8,PC)	@(d:16,PC)	@@aa:8	-
データ転送命令	MOV	BWL	BWL	BWL	BWL	BWL	BWL	B	BWL	BWL	-	-	-	-
	POP, PUSH	-	-	-	-	-	-	-	-	-	-	-	-	WL
	MOVEPE、 MOVTPE	-	-	-	-	-	-	-	B	-	-	-	-	-
算術演算命令	ADD, CMP	BWL	BWL	-	-	-	-	-	-	-	-	-	-	-
	SUB	WL	BWL	-	-	-	-	-	-	-	-	-	-	-
	ADDX, SUBX	B	B	-	-	-	-	-	-	-	-	-	-	-
	ADDS, SUBS	-	L*1	-	-	-	-	-	-	-	-	-	-	-
	INC, DEC	-	BWL	-	-	-	-	-	-	-	-	-	-	-
	DAA, DAS	-	B	-	-	-	-	-	-	-	-	-	-	-
	MULXU、 DIVXU	-	BW	-	-	-	-	-	-	-	-	-	-	-
	MULXS、 DIVXS	-	BW	-	-	-	-	-	-	-	-	-	-	-
	NEG	-	BWL	-	-	-	-	-	-	-	-	-	-	-
	EXTU, EXTS	-	WL	-	-	-	-	-	-	-	-	-	-	-
論理演算命令	AND, OR、 XOR	BWL	BWL	-	-	-	-	-	-	-	-	-	-	-
	NOT	-	BWL	-	-	-	-	-	-	-	-	-	-	-
	シフト命令	-	BWL	-	-	-	-	-	-	-	-	-	-	-
ビット操作命令	-	B	B	-	-	-	-	B	-	-	-	-	-	
分岐命令	Bcc, BSR	-	-	-	-	-	-	-	-	-	○	○	-	-
	JMP, JSR	-	-	○	-	-	-	-	-	○*2	-	-	○	-
	RTS	-	-	-	-	-	-	-	-	-	-	-	-	○
システム制御命令	TRAPA	-	-	-	-	-	-	-	-	-	-	-	-	○
	RTE	-	-	-	-	-	-	-	-	-	-	-	-	○
	SLEEP	-	-	-	-	-	-	-	-	-	-	-	-	○
	LDC	B	B	W	W	W	W	-	W	W	-	-	-	-
	STC	-	B	W	W	W	W	-	W	W	-	-	-	-
	ANDC、 ORC、 XORC	B	-	-	-	-	-	-	-	-	-	-	-	-
	NOP	-	-	-	-	-	-	-	-	-	-	-	-	○
ブロック転送命令	-	-	-	-	-	-	-	-	-	-	-	-	-	BW

《記号説明》

B:バイト

W:ワード

L:ロングワード

■ :H8/300H CPUで追加された命令

【注】 *1 ADDS, SUBS命令のオペランドサイズは、H8/300H CPUではロングワード、H8/300CPUではワードサイズです。

*2 JMP, JSR命令の絶対アドレス (@aa) のビット長は、H8/300H CPUでは24ビット、H8/300H CPUでは16ビットです。

1.6.3 命令の機能別一覧

表 1.3 に命令の機能別一覧を示します。また、以下に表 1.3 で使用される記号の意味を示します。

オペレーションの記号

Rd	汎用レジスタ (デスティネーション) *
Rs	汎用レジスタ (ソース側) *
Rn	汎用レジスタ*
ERn	汎用レジスタ (32 ビットレジスタ)
(EAd)	デスティネーションオペランド
(EAs)	ソースオペランド
CCR	コンディションコードレジスタ
N	CCR の N (ネガティブ) フラグ
Z	CCR の Z (ゼロ) フラグ
V	CCR の V (オーバフロー) フラグ
C	CCR の C (キャリ) フラグ
PC	プログラムカウンタ
SP	スタックポインタ
#IMM	イミディエイトデータ
disp	ディスプレースメント
+	加算
-	減算
×	乗算
÷	除算
^	論理積
∨	論理和
⊕	排他的論理和
→	転送
~	反転論理 (論理的補数)
:3/:8/:16/:24	3/8/16/24 ビット長

【注】*汎用レジスタは、8 ビット (R0H~R7H、R0L~R7L)、16 ビット (R0~R7、E0~E7)、または 32 ビットレジスタ (ER0~ER7) です。

表 1.3 命令の機能別一覧

分類	命令	サイズ*	機能
データ 転送命令	MOV	B/W/L	(EAs)→Rd、Rs→(EAd) 汎用レジスタと汎用レジスタ、または汎用レジスタとメモリ間でデータ転送します。また、イミディエイトデータを汎用レジスタに転送します。
	MOVFPPE	B	(EAs)→Rd 外部メモリの内容 (@aa:16 で指定) を E クロックに同期したタイミングで汎用レジスタに転送します。
	MOVTPPE	B	Rs→(EAs) 汎用レジスタの内容を E クロックに同期したタイミングで外部メモリ (@aa:16 で指定) に転送します。
	POP	W/L	@SP+→Rn スタックから汎用レジスタへデータを復帰します。 POP.W Rn は MOV.W @SP+,Rn と、また POP.L ERn は MOV.L @SP+,ERn と同一です。

1. CPU

分類	命令	サイズ*	機能
データ 転送命令	PUSH	W/L	Rn→@-SP 汎用レジスタの内容をスタックに退避します。 PUSH.W RnはMOV.W Rn,@-SPと、またPUSH.L ERnはMOV.L ERn, @-SPと同一です。
算術演算 命令	ADD SUB	B/W/L	Rd±Rs→Rd、Rd±#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデ ータ間の加減算を行います（バイトサイズでの汎用レジスタとイミディ エイトデータ間の減算はできません。SUBX命令またはADD命令を使用 してください）。
	ADDX SUBX	B	Rd±Rs±C→Rd、Rd±#IMM±C→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデ ータ間のキャリ付きの加減算を行います。
	INC DEC	B/W/L	Rd±1→Rd、Rd±2→Rd 汎用レジスタに1または2を加減算します（バイトサイズでは1の加 減算のみ可能です）。
	ADDS SUBS	L	Rd±1→Rd、Rd±2→Rd、Rd±4→Rd 32ビットレジスタに1、2、または4を加減算します。
	DAA DAS	B	Rd(10進補正)→Rd 汎用レジスタ上の加減算結果をCCRを参照して4ビットBCDデータ に補正します。
	MULXU	B/W	Rd×Rs→Rd 汎用レジスタと汎用レジスタ間の符号なし乗算を行います。 8ビット×8ビット→16ビット、16ビット×16ビット→32ビットの乗 算が可能です。
	MULXS	B/W	Rd×Rs→Rd 汎用レジスタと汎用レジスタ間の符号付き乗算を行います。 8ビット×8ビット→16ビット、16ビット×16ビット→32ビットの乗 算が可能です。
	DIVXU	B/W	Rd÷Rs→Rd 汎用レジスタと汎用レジスタ間の符号なし除算を行います。 16ビット÷8ビット→商8ビット余り8ビット、 32ビット÷16ビット→商16ビット余り16ビットの除算が可能です。
	DIVXS	B/W	Rd÷Rs→Rd 汎用レジスタと汎用レジスタ間の符号付き除算を行います。 16ビット÷8ビット→商8ビット余り8ビット、 32ビット÷16ビット→商16ビット余り16ビットの除算が可能です。
	CMP	B/W/L	Rd-Rs、Rd-#IMM 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデ ータ間の比較を行い、その結果をCCRに反映します。
	NEG	B/W/L	0-Rd→Rd 汎用レジスタの内容の2の補数（算術的補数）をとります。
EXTU	W/L	Rd（ゼロ拡張）→Rd 16ビットレジスタの下位8ビットをワードサイズにゼロ拡張します。 または、32ビットレジスタの下位16ビットをロングワードサイズにゼ ロ拡張します。	

分類	命令	サイズ*	機 能
算術演算命令	EXTS	W/L	Rd (符号拡張) →Rd 16 ビットレジスタの下位 8 ビットをワードサイズに符号拡張します。または、32 ビットレジスタの下位 16 ビットをロングワードサイズに符号拡張します。
論理演算命令	AND	B/W/L	Rd∧Rs→Rd、Rd∧#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理積をとります。
	OR	B/W/L	Rd∨Rs→Rd、Rd∨#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理和をとります。
	XOR	B/W/L	Rd⊕Rs→Rd、Rd⊕#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の排他的論理和をとります。
	NOT	B/W/L	~Rd→Rd 汎用レジスタの内容の 1 の補数（論理的補数）をとります。
シフト命令	SHAL	B/W/L	Rd (シフト処理) →Rd 汎用レジスタの内容を算術的にシフトします。
	SHAR	B/W/L	Rd (シフト処理) →Rd 汎用レジスタの内容を算術的にシフトします。
	SHLL	B/W/L	Rd (シフト処理) →Rd 汎用レジスタの内容を論理的にシフトします。
	SHLR	B/W/L	Rd (シフト処理) →Rd 汎用レジスタの内容を論理的にシフトします。
	ROTL	B/W/L	Rd (ローテート処理) →Rd 汎用レジスタの内容をローテートします。
	ROTR	B/W/L	Rd (ローテート処理) →Rd 汎用レジスタの内容をローテートします。
ビット操作命令	ROTXL	B/W/L	Rd (ローテート処理) →Rd 汎用レジスタの内容をキャリフラグを含めてローテートします。
	ROTXR	B/W/L	Rd (ローテート処理) →Rd 汎用レジスタの内容をキャリフラグを含めてローテートします。
	BSET	B	1→ (<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを 1 にセットします。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定します。
	BCLR	B	0→ (<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを 0 にクリアします。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定します。
	BNOT	B	~ (<ビット番号>of<EAd>) → (<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを反転します。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定します。
	BTST	B	~ (<ビット番号>of<EAd>) →Z 汎用レジスタまたはメモリのオペランドの指定された 1 ビットをテストし、ゼロフラグに反映します。ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容下位 3 ビットで指定します。
BAND	B	C∧ (<ビット番号>of<EAd>) →C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。	
BIAND	B	C∧[~ (<ビット番号>of<EAd>)] →C 汎用レジスタまたはメモリのオペランドの指定された 1 ビットを反転し、キャリフラグとの論理積をとり、結果をキャリフラグに格納します。ビット番号は、3 ビットのイミディエイトデータで指定されます。	

1. CPU

分類	命令	サイズ*	機能																																																			
ビット 操作命令	BOR	B	$C \vee (<\text{ビット番号}> \text{of} <\text{EAd}>) \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定された1ビットとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。																																																			
	BIOR	B	$C \vee [\sim (<\text{ビット番号}> \text{of} <\text{EAd}>)] \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。																																																			
	BXOR	B	$C \oplus (<\text{ビット番号}> \text{of} <\text{EAd}>) \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定された1ビットとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。																																																			
	BIXOR	B	$C \oplus [\sim (<\text{ビット番号}> \text{of} <\text{EAd}>)] \rightarrow C$ 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。																																																			
	BLD	B	$(<\text{ビット番号}> \text{of} <\text{EAd}>) \rightarrow C$ 汎用レジスタレジスタまたはメモリのオペランドの指定された1ビットをキャリフラグに転送します。																																																			
	BILD	B	$\sim (<\text{ビット番号}> \text{of} <\text{EAd}>) \rightarrow C$ 汎用レジスタレジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグに転送します。 ビット番号は、3ビットのイミディエイトデータで指定されます。																																																			
	BST	B	$C \rightarrow (<\text{ビット番号}> \text{of} <\text{EAd}>)$ 汎用レジスタまたはメモリのオペランドの指定された1ビットに、キャリフラグの内容を転送します。																																																			
	BIST	B	$C \rightarrow \sim (<\text{ビット番号}> \text{of} <\text{EAd}>)$ 汎用レジスタまたはメモリのオペランドの指定された1ビットに、キャリフラグを反転して転送します。 ビット番号は、3ビットのイミディエイトデータで指定されます。																																																			
分岐命令	Bcc	—	指定した条件が成立しているとき、指定されたアドレスへ分岐します。 分岐条件を下表に示します。 <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>ニーモニック</th> <th>説明</th> <th>分岐条件</th> </tr> </thead> <tbody> <tr> <td>BRA(BT)</td> <td>Always(True)</td> <td>Always</td> </tr> <tr> <td>BRN(BF)</td> <td>Never(False)</td> <td>Never</td> </tr> <tr> <td>BHI</td> <td>High</td> <td>$C \vee Z = 0$</td> </tr> <tr> <td>BLS</td> <td>Low or Same</td> <td>$C \vee Z = 1$</td> </tr> <tr> <td>BCC(BHS)</td> <td>Carry Clear(High or Same)</td> <td>$C = 0$</td> </tr> <tr> <td>BCS(BLO)</td> <td>Carry Set(Low)</td> <td>$C = 1$</td> </tr> <tr> <td>BNE</td> <td>Not Equal</td> <td>$Z = 0$</td> </tr> <tr> <td>BEQ</td> <td>Equal</td> <td>$Z = 1$</td> </tr> <tr> <td>BVC</td> <td>oVerflow Clear</td> <td>$V = 0$</td> </tr> <tr> <td>BVS</td> <td>oVerflow Set</td> <td>$V = 1$</td> </tr> <tr> <td>BPL</td> <td>PLus</td> <td>$N = 0$</td> </tr> <tr> <td>BMI</td> <td>MInus</td> <td>$N = 1$</td> </tr> <tr> <td>BGE</td> <td>Greater or Equal</td> <td>$N \oplus V = 0$</td> </tr> <tr> <td>BLT</td> <td>Less Than</td> <td>$N \oplus V = 1$</td> </tr> <tr> <td>BGT</td> <td>Greater Than</td> <td>$Z \vee (N \oplus V) = 0$</td> </tr> <tr> <td>BLE</td> <td>Less or Equal</td> <td>$Z \vee (N \oplus V) = 1$</td> </tr> </tbody> </table>	ニーモニック	説明	分岐条件	BRA(BT)	Always(True)	Always	BRN(BF)	Never(False)	Never	BHI	High	$C \vee Z = 0$	BLS	Low or Same	$C \vee Z = 1$	BCC(BHS)	Carry Clear(High or Same)	$C = 0$	BCS(BLO)	Carry Set(Low)	$C = 1$	BNE	Not Equal	$Z = 0$	BEQ	Equal	$Z = 1$	BVC	oVerflow Clear	$V = 0$	BVS	oVerflow Set	$V = 1$	BPL	PLus	$N = 0$	BMI	MInus	$N = 1$	BGE	Greater or Equal	$N \oplus V = 0$	BLT	Less Than	$N \oplus V = 1$	BGT	Greater Than	$Z \vee (N \oplus V) = 0$	BLE	Less or Equal	$Z \vee (N \oplus V) = 1$
	ニーモニック	説明	分岐条件																																																			
	BRA(BT)	Always(True)	Always																																																			
BRN(BF)	Never(False)	Never																																																				
BHI	High	$C \vee Z = 0$																																																				
BLS	Low or Same	$C \vee Z = 1$																																																				
BCC(BHS)	Carry Clear(High or Same)	$C = 0$																																																				
BCS(BLO)	Carry Set(Low)	$C = 1$																																																				
BNE	Not Equal	$Z = 0$																																																				
BEQ	Equal	$Z = 1$																																																				
BVC	oVerflow Clear	$V = 0$																																																				
BVS	oVerflow Set	$V = 1$																																																				
BPL	PLus	$N = 0$																																																				
BMI	MInus	$N = 1$																																																				
BGE	Greater or Equal	$N \oplus V = 0$																																																				
BLT	Less Than	$N \oplus V = 1$																																																				
BGT	Greater Than	$Z \vee (N \oplus V) = 0$																																																				
BLE	Less or Equal	$Z \vee (N \oplus V) = 1$																																																				
JMP	—	指定されたアドレスへ無条件に分岐します。																																																				
BSR	—	指定されたアドレスへサブルーチン分岐します。																																																				

分類	命令	サイズ*	機能
分岐命令	JSR	—	指定されたアドレスへサブルーチン分岐します。
	RTS	—	サブルーチンから復帰します。
システム 制御命令	TRAPA	—	命令トラップ例外処理を行います。
	RTE	—	例外処理ルーチンから復帰します。
	SLEEP	—	低消費電力状態に遷移します。
	LDC	B/W	(EAs) →CCR 汎用レジスタまたはメモリの内容を CCR に転送します。また、イミディエイトデータを CCR に転送します。CCR は 8 ビットですが、メモリと CCR 間の転送はワードサイズで行われ、上位 8 ビットが有効になります。
	STC	B/W	CCR → (EAd) CCR の内容を汎用レジスタまたはメモリに転送します。CCR は 8 ビットですが、CCR とメモリ間の転送はワードサイズで行われ、上位 8 ビットが有効になります。
	ANDC	B	CCR ∧ #IMM → CCR CCR とイミディエイトデータの論理積をとります。
	ORC	B	CCR ∨ #IMM → CCR CCR とイミディエイトデータの論理和をとります。
	XORC	B	CCR ⊕ #IMM → CCR CCR とイミディエイトデータの排他的論理和をとります。
ブロック 転送命令	EEPMOV.B	—	if R4L≠0 then Repeat@ER5+ → @ER6+ R4L-1 → R4L Until R4L=0 else next;
	EEPMOV.W	—	if R4≠0 then Repeat@ER5+ → @ER6+ R4-1 → R4 Until R4=0 else next; ブロック転送命令です。ER5 で示されるアドレスから始まり、R4L または R4 で指定されるバイト数のデータを、ER6 で示されるアドレスのロケーションへ転送します。転送終了後、次の命令を実行します。

【注】 *サイズはオペランドサイズを示します。

B:バイト

W:ワード

L:ロングワード

1.6.4 命令の基本フォーマット

H8/300H CPU の命令は、2 バイト（ワード）を単位にしています。各命令はオペレーションフィールド (op)、レジスタフィールド (r)、EA 拡張部 (EA)、およびコンディションフィールド (cc) から構成されています。

1. CPU

(1) オペレーションフィールド

命令の機能を表し、アドレッシングモードの指定、オペランドの処理内容を指定します。命令の先頭4ビットを必ず含みます。2つのオペレーションフィールドを持つ場合もあります。

(2) レジスタフィールド

汎用レジスタを指定します。アドレスレジスタのとき3ビット、データレジスタのとき3ビットまたは4ビットです。2つのレジスタフィールドを持つ場合、またはレジスタフィールドを持たない場合もあります。

(3) EA 拡張部

イミディエイトデータ、絶対アドレスまたはディスプレースメントを指定します。8ビット、16ビット、または32ビットです。24ビットアドレスおよびディスプレースメントは、上位8ビットをすべて0 (H'00) とした32ビットデータとして扱われます。

(4) コンディションフィールド

Bcc 命令の分岐条件を指定します。

図 1.12 に命令フォーマットの例を示します。

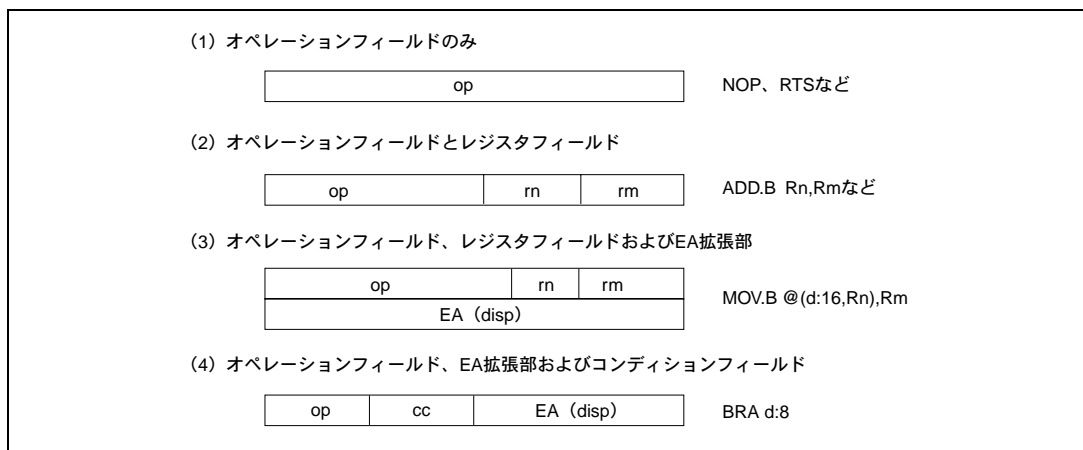


図 1.12 命令フォーマットの例

1.7 アドレッシングモードと実効アドレスの計算方法

1.7.1 アドレッシングモード

H8/300H CPU は表 1.4 に示すように、8種類のアドレッシングモードをサポートしています。命令ごとに、使用できるアドレッシングモードは異なります。

演算命令では、レジスタ直接、およびイミディエイトが使用できます。

転送命令では、プログラムカウンタ相対とメモリ間接を除くすべてのアドレッシングモードが使用できます。

また、ビット操作命令では、オペランドの指定にレジスタ直接、レジスタ間接、および絶対アドレス (@aa:8) が使用できます。さらに、オペランド中のビット番号を指定するためにレジスタ直接 (BSET、BCLR、BNOT、BTST の各命令)、およびイミディエイト (3ビット) が独立して使用できます。

表 1.4 アドレッシングモード一覧表

No.	アドレッシングモード	記号
1	レジスタ直接	Rn
2	レジスタ間接	@ERn
3	ディスプレースメント付きレジスタ間接	@(d:16,ERn)/@(d:24,ERn)
4	ポストインクリメントレジスタ間接 プリデクリメントレジスタ間接	@ERn+ @-ERn
5	絶対アドレス	@aa:8/@aa:16/@aa:24
6	イミディエイト	#xx:8/#xx:16/#xx:32
7	プログラムカウンタ相対	@(d:8,PC)/@(d:16,PC)
8	メモリ間接	@@aa:8

(1) レジスタ直接 Rn

命令コードのレジスタフィールドで指定されるレジスタ（8ビット、16ビットまたは32ビット）がオペランドとなります。

8ビットレジスタとしてはR0H～R7H、R0L～R7Lを指定可能です。

16ビットレジスタとしてはR0～R7、E0～E7を指定可能です。

32ビットレジスタとしてはER0～ER7を指定可能です。

(2) レジスタ間接 @ERn

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容の下位24ビットをアドレスとしてメモリ上のオペランドを指定します。

(3) ディスプレースメント付きレジスタ間接 @(d:16,ERn) / @(d:24,ERn)

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容に命令コード中に含まれる16ビットディスプレースメントまたは24ビットディスプレースメントを加算した内容の下位24ビットをアドレスとしてメモリ上のオペランドを指定します。加算に際して、ディスプレースメントは符号拡張されます。

(4) ポストインクリメントレジスタ間接 @ERn+ / プリデクリメントレジスタ間接 @-ERn**(a) ポストインクリメントレジスタ間接 @ERn+**

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容の下位24ビットをアドレスとしてメモリ上のオペランドを指定します。その後、アドレスレジスタの内容（32ビット）に1、2または4が加算され、加算結果がアドレスレジスタに格納されます。バイトサイズでは1、ワードサイズでは2、ロングワードサイズでは4がそれぞれ加算されます。ワードサイズまたはロングワードサイズのとき、アドレスレジスタの内容が偶数となるようにしてください。

(b) プリデクリメントレジスタ間接 @-ERn

命令コードのレジスタフィールドで指定されるアドレスレジスタ（ERn）の内容から1、2または4を減算した内容の下位24ビットをアドレスとしてメモリ上のオペランドを指定します。その後、減算結果がアドレスレジスタに格納されます。バイトサイズでは1、ワードサイズでは2、ロングワードサイズでは4がそれぞれ減算されます。ワードサイズまたはロングワードサイズのとき、アドレスレジスタの内容が偶数になるようにしてください。

(5) 絶対アドレス @aa:8 / @aa:16 / @aa:24

命令コード中に含まれる絶対アドレスで、メモリ上のオペランドを指定します。

絶対アドレスは8ビット（@aa:8）、16ビット（@aa:16）、または24ビット（@aa:24）です。

8ビット絶対アドレスの場合、上位16ビットはすべて1（H'FFFF）となります。16ビット絶対ア

1. CPU

ドレスの場合、上位 8 ビットは符号拡張されます。24 ビット絶対アドレスの場合、全アドレス空間をアクセスできます。

絶対アドレスのアクセス範囲を表 1.5 に示します。

表 1.5 絶対アドレスのアクセス範囲

	ノーマルモード	アドバンスモード (16M バイトモードの場合)
8 ビット (@aa:8)	H'FFF00~H'FFFF (65,280~65,535)	H'FFFF00~H'FFFFFF (16,776,960~16,777,215)
16 ビット (@aa:16)	H'0000~H'FFFF (0~65,535)	H'000000~H'007FFF,H'FF8000~H'FFFFFF (0~32,767、16,744,448~16,777,215)
24 ビット (@aa:24)	H'0000~H'FFFF (0~65,535)	H'000000~H'FFFFFF (0~16,777,215)

アクセス範囲の詳細については当該 LSI のハードウェアマニュアルを参照してください。

(6) イミディエイト #xx:8/#xx:16/#xx:32

命令コード中に含まれる 8 ビット (#xx:8)、16 ビット (#xx:16)、または 32 ビット (#xx:32) のデータを直接オペランドとして使用します。

なお、ADDS、SUBS、INC、DEC 命令では、イミディエイトデータが命令コード中に暗黙的に含まれます。ビット操作命令では、ビット番号を指定するための 3 ビットのイミディエイトデータが、命令コード中に含まれる場合があります。また、TRAPA 命令ではベクタアドレスを指定するための 2 ビットのイミディエイトデータが、命令コードの中に含まれます。

(7) プログラムカウンタ相対 @(d:8,PC)/@(d:16,PC)

Bcc、BSR 命令で使用されます。PC の内容で指定される 24 ビットのアドレスに、命令コード中に含まれる 8 ビット、または 16 ビットディスプレースメントを加算して 24 ビットの分岐アドレスを生成します。加算に際して、ディスプレースメントは 24 ビットに符号拡張されます。また加算される PC の内容は次の命令の先頭アドレスとなっていますので、分岐可能範囲は分岐命令に対して -126~+128 バイト (-63~+64 ワード) または -32766~+32768 バイト (-16383~+16384 ワード) です。このとき、加算結果が偶数となるようにしてください。

(8) メモリ間接 @@aa:8

JMP、JSR 命令で使用されます。命令コード中に含まれる 8 ビット絶対アドレスでメモリ上のオペランドを指定し、この内容を分岐アドレスとして分岐します。

8 ビット絶対アドレスの上位のビットはすべて 0 となりますので、分岐アドレスを格納できるのは 0~255 (ノーマルモードのとき H'0000~H'00FF、アドバンスモードのとき H'000000~H'0000FF) 番地です。

ノーマルモードの場合は、メモリ上のオペランドはワードサイズで指定し、16 ビットの分岐アドレスを生成します。

また、アドバンスモードの場合は、メモリ上のオペランドはロングワードサイズで指定します。このうち先頭の 1 バイトは無視され、24 ビットの分岐アドレスを生成します。

ただし、分岐アドレスを格納可能なアドレスの先頭領域は、例外処理ベクタ領域と共通になっていますから注意してください。詳細は当該 LSI のハードウェアマニュアルを参照してください。

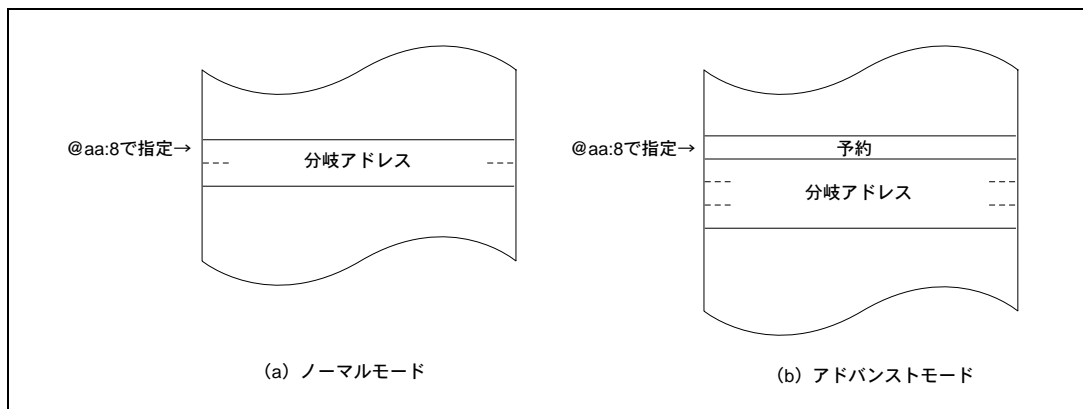


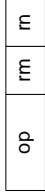
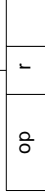

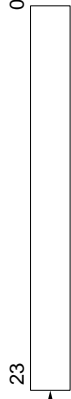

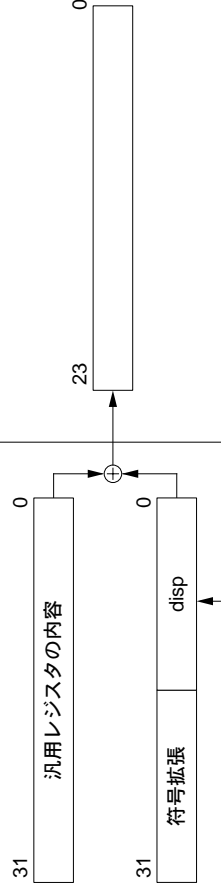
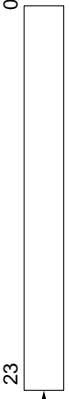

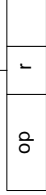
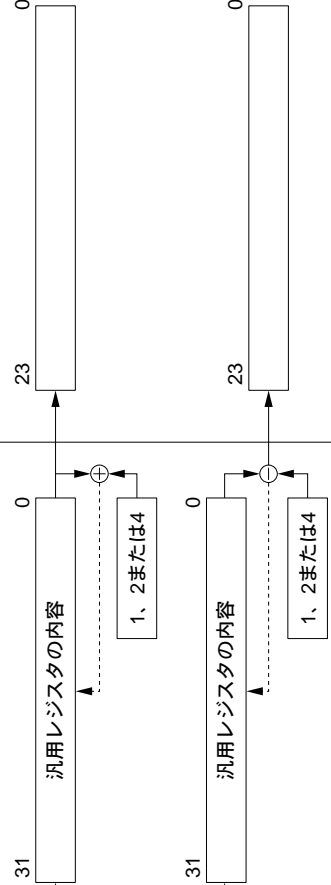

図 1.13 メモリ間接による分岐アドレスの指定

ワードサイズ、ロングワードサイズでメモリを指定する場合、および分岐アドレスを指定する場合に奇数アドレスを指定すると、最下位ビットは0とみなされ、1番地前から始まるデータまたは命令コードをアクセスします（「1.5.2 メモリ上でのデータ構成」を参照してください）。

1.7.2 実効アドレスの計算方法

各アドレッシングモードにおける実効アドレス（EA:Effective Address）の計算法を表 1.6 に示します。ノーマルモードの場合、実効アドレスの上位8ビットは無視され、16ビットのアドレスとなります。

表1.6 実効アドレスの計算方法

No	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス (EA)								
1	レジスタ直接 (Rn) 		オペランドは汎用レジスタの内容です。								
2	レジスタ間接 (@ERn) 										
3	ディスプレースメント付レジスタ間接 @(d:16,ERn) / @(d:24,ERn) 										
4	ポストインクリメントレジスタ間接/プリデクリメントレジスタ間接 ・ポストインクリメントレジスタ間接 @ERn+  ・プリデクリメントレジスタ間接 @-ERn 	 <table border="1" data-bbox="1077 898 1186 1226"> <thead> <tr> <th>オペランドサイズ</th> <th>加減算される値</th> </tr> </thead> <tbody> <tr> <td>バイト</td> <td>1</td> </tr> <tr> <td>ワード</td> <td>2</td> </tr> <tr> <td>ロングワード</td> <td>4</td> </tr> </tbody> </table>	オペランドサイズ	加減算される値	バイト	1	ワード	2	ロングワード	4	
オペランドサイズ	加減算される値										
バイト	1										
ワード	2										
ロングワード	4										

No	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス (EA)
5	<p>絶対アドレス</p> <p>@aa:8</p> <p>@aa:16</p> <p>@aa:24</p>		
6	<p>イミディエイト#xx:8/#xx:16/#xx:32</p>		オペランドはイミディエイトデータです。

No	アドレッシングモード・命令フォーマット	実効アドレス計算方法	実効アドレス (EA)
7	<p>アドレッシングモード・命令フォーマット</p> <p>プログラムカウンタ相対 @(d:8,PC)/@(d:16,PC)</p>		
8	<p>メモリ間接 @@aa:8</p> <ul style="list-style-type: none"> ・ノーマルモード ・アドバンストモード 		

2. 各命令の説明

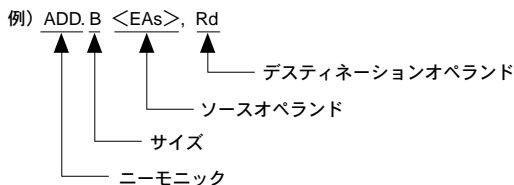
2.1 表と記号の説明

「2.2 各命令の説明」の表の見方について説明します。なお、同一の命令についての説明でも、複数ページにわたっているものがありますから注意してください。

(1)ニーモニック	(2)分類
(3)オペレーション	(6)コンディションコード
(4)アセンブラフォーマット	
(5)オペランドサイズ	
(7)説明	
(8)使用可能なレジスタ	
(9)オペランド形式と実行ステート数	
(10)注意事項	

- (1) ニーモニック (フルネーム)
命令のニーモニックとフルネームを示します。
- (2) 分類
命令の機能を示します。
- (3) オペレーション
命令の操作を簡潔に示します。(2.1.2を参照)
- (4) アセンブラフォーマット
命令のアセンブラフォーマットを示します。(2.1.1を参照)
- (5) オペランドサイズ
使用できるオペランドのサイズを示します。
- (6) コンディションコード
命令実行後のコンディションコードレジスタ (CCR) の各ビットの変化を示します。(2.1.3を参照)
- (7) 説明
命令の動作について詳細に説明します。
- (8) 使用可能なレジスタ
命令コードのレジスタフィールドで指定できるレジスタを示します。
- (9) オペランド形式と実行ステート数
命令のアドレッシングモード、インストラクションフォーマット、ならびに実行ステート数を示します。
- (10) 注意事項
命令を実行するうえでの注意事項などを示します。

2.1.1 アセンブラフォーマット



オペランドサイズは、バイト (B)、ワード (W)、ロングワード (L) があります。命令によって、使用できるオペランドサイズは異なります。

<EA>は、複数のアドレッシングモードが使用できることを示します。H8/300H CPU がサポートするアドレッシングモードは、次の 8 種類です。実効アドレスの計算方法については「1.7 アドレッシングモードと実効アドレスの計算方法」を参照してください。

2. 各命令の説明

記号	アドレッシングモード
Rn	レジスタ直接
@ERn	レジスタ間接
@(d:16,ERn)/(d:24,ERn)	ディスプレイースメント (16/24 ビット) 付レジスタ間接
@ERn+/@-ERn	ポストインクリメントレジスタ間接/プリデクリメントレジスタ間接
@aa:8/@aa:16/@aa:24	絶対アドレス (8/16/24 ビット)
#xx:8/#xx:16/#xx:32	イミディエイト (8/16/32 ビット)
@(d:8,PC)/(d:16, PC)	プログラムカウンタ相対 (8/16 ビット)
@@aa:8	メモリ間接

なお、:8/:16/:24/:32 は省略することができます。特に絶対アドレス、およびディスプレイースメントについては:8/:16/:24 を省略すると、値の範囲に応じてアセンブラが最適化を行います。

詳細は「H8/300 シリーズクロスアセンブラユーザーズマニュアル」を参照してください。

2.1.2 オペレーション

オペレーションの欄で使用されている記号と動作記号を以下に示します。

Rd	デスティネーション側の汎用レジスタ
Rs	ソース側の汎用レジスタ
Rn	汎用レジスタ
ERd	デスティネーション側の汎用レジスタ (アドレスレジスタまたは 32 ビットレジスタ)
ERs	ソース側の汎用レジスタ (アドレスレジスタまたは 32 ビットレジスタ)
ERn	汎用レジスタ (32 ビットレジスタ)
(EAd)	デスティネーションオペランド
(EAs)	ソースオペランド
PC	プログラムカウンタ
SP	スタックポインタ
CCR	コンディションコードレジスタ
N	CCR の N (ネガティブ) フラグ
Z	CCR の Z (ゼロ) フラグ
V	CCR の V (オーバーフロー) フラグ
C	CCR の C (キャリ) フラグ
disp	ディスプレイースメント
→	左辺のオペランドから右辺のオペランドへの転送、 または左辺の状態から右辺の状態への遷移
+	両辺のオペランドを加算
-	左辺のオペランドから右辺のオペランドを減算
×	両辺のオペランドを乗算
÷	左辺のオペランドから右辺のオペランドで除算
^	両辺のオペランドの論理積
∨	両辺のオペランドの論理和
⊕	両辺のオペランドの排他的論理和
~	反転論理 (論理的補数)
()<>	オペランドの内容

【注】 *汎用レジスタは、8 ビット (R0H~R7H, R0L~R7L)、16 ビット (R0~R7, E0~E7) または 32 ビット (ER0~ER7) です。

2.1.3 コンディションコード

コンディションコードの欄で使用されている記号を以下に示します。

記号	内容
↕	実行結果にしたがって変化することを表します。
*	不確定であることを表します（値を保証しません）。
0	常に0にクリアされることを表します。
1	常に1にセットされることを表します。
—	実行結果に影響を受けないことを表します。
△	条件によって異なります。注意事項を参照してください。

コンディションコードの変化の詳細については「2.7 コンディションコードの変化」を参照してください。

2.1.4 インストラクションフォーマット

インストラクションフォーマットの欄で使用されている記号を以下に示します。

記号	内容
IMM	イミディエイトデータ (2、3、8、16、32 ビット)
abs	絶対アドレス (8、16、24 ビット)
disp	ディスプレースメント (8、16、24 ビット)
rs、rd、rn	レジスタフィールド (4 ビット) rs、rd、rn はそれぞれオペランドの形式の Rs、Rd、Rn に対応
ers、erd、ern	レジスタフィールド (3 ビット) ers、erd、ern はオペランドの形式の ERs、ERd、ERn に対応

2.1.5 レジスタの指定方法

(1) アドレスレジスタの指定

汎用レジスタをアドレスレジスタとして使用するとき (@ERn、@(d:16,ERn)、@(d:24,ERn)、@ERn+、@-ERn) は 32 ビットのレジスタフィールド (ers、erd) で指定されます。

(2) データレジスタの指定

汎用レジスタは、データレジスタとして使用するとき、32 ビット、16 ビットまたは 8 ビットレジスタです。

32 ビットレジスタとして使用するとき、3 ビットのレジスタフィールド (ers、erd、ern) で指定されます。

16 ビットレジスタとして使用するとき、4 ビットのレジスタフィールド (rs、rd、rn) で指定されます。このときレジスタフィールドの下位 3 ビットがレジスタ番号を示し、上位 1 ビットが 1 のとき汎用レジスタ En が指定され、0 のとき汎用レジスタ Rn が指定されます。

また、8 ビットレジスタとして使用するとき、4 ビットのレジスタフィールド (rs、rd、rn) で指定されます。また、このときレジスタフィールドの下位 3 ビットがレジスタ番号を示し、上位 1 ビットが 1 のとき汎用レジスタ RnL が指定され、0 のとき汎用レジスタ RnH が指定されます。

この対応を以下に示します。

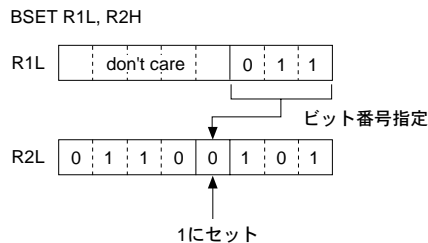
2. 各命令の説明

アドレスレジスタ 32 ビットレジスタ		16 ビットレジスタ		8 ビットレジスタ	
レジスタ フィールド	汎用レジスタ	レジスタ フィールド	汎用レジスタ	レジスタ フィールド	汎用レジスタ
000	ER0	0000	R0	0000	R0H
001	ER1	0001	R1	0001	R1H
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
111	ER7	0111	R7	0111	R7H
		1000	E0	1000	R0L
		1001	E1	1001	R1L
		⋮	⋮	⋮	⋮
		⋮	⋮	⋮	⋮
		1111	E7	1111	R7L

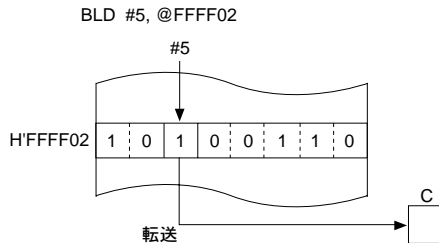
2.1.6 ビット操作命令におけるビットデータのアクセス方法

ビットデータは、レジスタまたはメモリ上のオペランドデータ（バイト）の第 n ビット（ $n=0, 1, 2, 3, \dots, 7$ ）という形でアクセスされます。このとき、ビット番号は、3 ビットのイミディエイトデータまたは汎用レジスタの内容（下位 3 ビットのみ有効）によって指定されます。

(例1) R2H のビット 3 を 1 にセットする場合



(例2) H'FFFF02 番地のビット 5 をビットアキュムレータに転送する場合



なお、ビット操作命令のオペランドサイズおよびアドレス形式はレジスタまたはメモリ上のオペランドデータについて示しています。

2.2 各命令の説明

2.2.1 以降に各命令について説明します。

2.2.1(1) ADD(B)

ADD (ADD binary)		2進加算																													
<ul style="list-style-type: none"> ●オペレーション Rd+(EAs)→Rd 	<ul style="list-style-type: none"> ●コンディションコード 																														
<ul style="list-style-type: none"> ●アセンブラフォーマット ADD.B <EAs>, Rd 	<div style="text-align: center;"> <table border="1"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>↑</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table> </div> <p>H: ビット3にキャリが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>C: ビット7にキャリが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p>			I	UI	H	U	N	Z	V	C	—	—	↑	—	↑	↑	↑	↑												
I	UI	H	U	N	Z	V	C																								
—	—	↑	—	↑	↑	↑	↑																								
<ul style="list-style-type: none"> ●オペランドサイズ バイト 																															
<ul style="list-style-type: none"> ●説明 <p>8ビットレジスタRdの内容（デスティネーションオペランド）とソースオペランドを加算し、結果を8ビットレジスタRdに格納します。</p>																															
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ <p>Rd : R0L~R7L, R0H~R7H Rs : R0L~R7L, R0H~R7H</p>																															
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>ADD.B</td> <td>#xx:8,Rd</td> <td>8</td> <td>rd</td> <td>IMM</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>ADD.B</td> <td>Rs,Rd</td> <td>0</td> <td>8</td> <td>rs</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	イミディエイト	ADD.B	#xx:8,Rd	8	rd	IMM		2	レジスタ直接	ADD.B	Rs,Rd	0	8	rs	rd	2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																								
			第1バイト	第2バイト	第3バイト	第4バイト																									
イミディエイト	ADD.B	#xx:8,Rd	8	rd	IMM		2																								
レジスタ直接	ADD.B	Rs,Rd	0	8	rs	rd	2																								
<ul style="list-style-type: none"> ●注意事項 																															

2. 各命令の説明

2.2.1(2) ADD(W)

ADD (ADD binary)		2進加算																															
<p>●オペレーション Rd+(EAs)→Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>↓</td><td>—</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td> </tr> </table>			I	UI	H	U	N	Z	V	C	—	—	↓	—	↓	↓	↓	↓														
I	UI	H	U	N	Z	V	C																										
—	—	↓	—	↓	↓	↓	↓																										
<p>●アセンブラフォーマット ADD.W <EAs>, Rd</p>	<p>H: ビット11にキャリが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>C: ビット15にキャリが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p>																																
<p>●オペランドサイズ ワード</p>																																	
<p>●説明</p> <p>16ビットレジスタRdの内容（デスティネーションオペランド）とソースオペランドを加算し、結果を16ビットレジスタRdに格納します。</p>																																	
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0~R7, E0~E7 Rs : R0~R7, E0~E7</p>																																	
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>ADD.W</td> <td>#xx:16,Rd</td> <td>7</td><td>9</td><td>1</td><td>rd</td> <td>IMM</td> <td>4</td> </tr> <tr> <td>レジスタ直接</td> <td>ADD.W</td> <td>Rs,Rd</td> <td>0</td><td>9</td><td>rs</td><td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	イミディエイト	ADD.W	#xx:16,Rd	7	9	1	rd	IMM	4	レジスタ直接	ADD.W	Rs,Rd	0	9	rs	rd		2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																										
			第1バイト	第2バイト	第3バイト	第4バイト																											
イミディエイト	ADD.W	#xx:16,Rd	7	9	1	rd	IMM	4																									
レジスタ直接	ADD.W	Rs,Rd	0	9	rs	rd		2																									
<p>●注意事項</p>																																	

2.2.1(3) ADD(L)

ADD (ADD binary)		2進加算																																							
<p>●オペレーション ERd+(EAs)→ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>↑</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table> <p>H: ビット27にキャリが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>C: ビット31にキャリが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p>			I	UI	H	U	N	Z	V	C	—	—	↑	—	↑	↑	↑	↑																						
I	UI	H	U	N	Z	V	C																																		
—	—	↑	—	↑	↑	↑	↑																																		
<p>●アセンブラフォーマット ADD.L <EAs>, ERd</p>																																									
<p>●オペランドサイズ ロングワード</p>																																									
<p>●説明 32ビットレジスタERdの内容（デスティネーションオペランド）とソースオペランドを加算し、結果を32ビットレジスタERdに格納します。</p>																																									
<p>●使用可能な汎用レジスタ ERd : ER0~ER7 ERs : ER0~ER7</p>																																									
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="6">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> <th>第5バイト</th> <th>第6バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>ADD.L</td> <td>#xx:32,Rd</td> <td>7</td> <td>A</td> <td>1</td> <td>0</td> <td>erd</td> <td>IMM</td> <td></td> <td>6</td> </tr> <tr> <td>レジスタ直接</td> <td>ADD.L</td> <td>ERs,ERd</td> <td>0</td> <td>A</td> <td>1</td> <td>ers</td> <td>0</td> <td>erd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	イミディエイト	ADD.L	#xx:32,Rd	7	A	1	0	erd	IMM		6	レジスタ直接	ADD.L	ERs,ERd	0	A	1	ers	0	erd		2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット						実行ステート数																																
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト																																	
イミディエイト	ADD.L	#xx:32,Rd	7	A	1	0	erd	IMM		6																															
レジスタ直接	ADD.L	ERs,ERd	0	A	1	ers	0	erd		2																															
<p>●注意事項</p>																																									

2. 各命令の説明

2.2.2 ADDS

ADDS (ADD with Sign extention)		アドレスデータ2進加算																																					
<p>●オペレーション</p> <p>Rd+1→ERd Rd+2→ERd Rd+4→ERd</p>	<p>●コンディションコード</p> <p>I U I H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H : 実行前の値が保持されます。 N : 実行前の値が保持されます。 Z : 実行前の値が保持されます。 V : 実行前の値が保持されます。 C : 実行前の値が保持されます。</p>			—	—	—	—	—	—	—	—																												
—	—	—	—	—	—	—	—																																
<p>●アセンブラフォーマット</p> <p>ADD.S #1, ERd ADD.S #2, ERd ADD.S #4, ERd</p>																																							
<p>●オペランドサイズ</p> <p>ロングワード</p>																																							
<p>●説明</p> <p>32ビットレジスタERdの内容（デスティネーションオペランド）に1、2または4を加算します。ADD命令とは異なり、コンディションコードは実行前の値を保持します。</p>																																							
<p>●使用可能な汎用レジスタ</p> <p>ERd : ER0~ER7</p>																																							
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ADDS</td> <td>#1,ERd</td> <td>0</td> <td>B</td> <td>0</td> <td>0:erd</td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>ADDS</td> <td>#2,ERd</td> <td>0</td> <td>B</td> <td>8</td> <td>0:erd</td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>ADDS</td> <td>#4,ERd</td> <td>0</td> <td>B</td> <td>9</td> <td>0:erd</td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ADDS	#1,ERd	0	B	0	0:erd	2	レジスタ直接	ADDS	#2,ERd	0	B	8	0:erd	2	レジスタ直接	ADDS	#4,ERd	0	B	9	0:erd	2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																																
			第1バイト	第2バイト	第3バイト	第4バイト																																	
レジスタ直接	ADDS	#1,ERd	0	B	0	0:erd	2																																
レジスタ直接	ADDS	#2,ERd	0	B	8	0:erd	2																																
レジスタ直接	ADDS	#4,ERd	0	B	9	0:erd	2																																
<p>●注意事項</p>																																							

2.2.3 ADDX

ADDX (ADD with eXtend carry)		キャリ付加算																													
<p>●オペレーション Rd+(EAs)+C→Rd</p>	<p>●コンディションコード</p> <p style="text-align: center;">I UI H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↑</td> </tr> </table> <p>H: ビット3にキャリが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき実行前の値が保持され、それ以外の場合は0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>C: ビット7にキャリが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p>			—	—	↑	—	↑	↑	↑	↑																				
—	—	↑	—	↑	↑	↑	↑																								
<p>●アセンブラフォーマット ADDX <EAs>, Rd</p>																															
<p>●オペランドサイズ バイト</p>																															
<p>●説明</p> <p>8ビットレジスタRdの内容（デスティネーションオペランド）とソースオペランドとキャリフラグの値を加算し、結果を8ビットレジスタRdに格納します。</p>																															
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H Rs : R0L~R7L, R0H~R7H</p>																															
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>ADDX</td> <td>#xx:8,Rd</td> <td>9</td> <td>rd</td> <td>IMM</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>ADDX</td> <td>Rs,Rd</td> <td>0</td> <td>E</td> <td>rs</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	イミディエイト	ADDX	#xx:8,Rd	9	rd	IMM		2	レジスタ直接	ADDX	Rs,Rd	0	E	rs	rd	2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																								
			第1バイト	第2バイト	第3バイト	第4バイト																									
イミディエイト	ADDX	#xx:8,Rd	9	rd	IMM		2																								
レジスタ直接	ADDX	Rs,Rd	0	E	rs	rd	2																								
<p>●注意事項</p>																															

2. 各命令の説明

2.2.4(1) AND(B)

AND (AND logical)		論理積								
●オペレーション Rd^(EAs) → Rd	●コンディションコード I UI H U N Z V C <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↓</td><td>0</td><td>—</td> </tr> </table>		—	—	—	—	↑	↓	0	—
—	—	—	—	↑	↓	0	—			
●アセンブラフォーマット AND.B <EAs>, Rd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。									
●オペランドサイズ バイト										
●説明 8ビットレジスタRdの内容（デスティネーションオペランド）とソースオペランドの論理積をとり、結果を8ビットレジスタRdに格納します。										
●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H Rs : R0L~R7L, R0H~R7H										
●オペランド形式と実行ステート数										
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数			
			第1バイト	第2バイト	第3バイト	第4バイト				
イミディエイト	AND.B	#xx:8,Rd	E	rd	IMM		2			
レジスタ直接	AND.B	Rs,Rd	1	6	rs	rd	2			
●注意事項										

2.2.4(2) AND(W)

AND (AND logical)		論理積						
●オペレーション Rd \wedge (EAs) \rightarrow Rd	●コンディションコード I UI H U N Z V C — — — — † † 0 —							
●アセンブラフォーマット AND.W <EAs>, Rd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき 1 にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。							
●オペランドサイズ ワード								
●説明 16ビットレジスタRdの内容 (デスティネーションオペランド) とソースオペランドの論理積をとり、結果を16ビットレジスタRdに格納します。								
●使用可能な汎用レジスタ Rd : R0~R7, E0~E7 Rs : R0~R7, E0~E7								
●オペランド形式と実行ステート数								
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	
			第1バイト	第2バイト	第3バイト	第4バイト		
イミディエイト	AND.W	#xx:16,Rd	7	9	6	rd	IMM	4
レジスタ直接	AND.W	Rs,Rd	6	6	rs	rd		2
●注意事項								

2. 各命令の説明

2.2.4(3) AND(L)

AND (AND logical)		論理積																																						
<ul style="list-style-type: none"> ●オペレーション ERd\wedge (EAs) \rightarrow ERd 	<ul style="list-style-type: none"> ●コンディションコード <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>U</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↓</td><td>0</td><td>—</td> </tr> </table> H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。 		I	U	H	U	N	Z	V	C	—	—	—	—	↑	↓	0	—																						
I	U	H	U	N	Z	V	C																																	
—	—	—	—	↑	↓	0	—																																	
<ul style="list-style-type: none"> ●アセンブラフォーマット AND.L <EAs>, ERd 																																								
<ul style="list-style-type: none"> ●オペランドサイズ ロングワード 																																								
<ul style="list-style-type: none"> ●説明 32ビットレジスタERdの内容 (デスティネーションオペランド) とソースオペランドの論理積をとり、結果を32ビットレジスタERdに格納します。 																																								
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ ERd : ER0~ER7 ERs : ER0~ER7 																																								
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="6">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> <th>第5バイト</th> <th>第6バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>AND.L</td> <td>#xx:32,ERd</td> <td>7</td> <td>A</td> <td>6</td> <td>0</td> <td>erd</td> <td colspan="2">IMM</td> <td>6</td> </tr> <tr> <td>レジスタ直接</td> <td>AND.L</td> <td>ERs,ERd</td> <td>0</td> <td>1</td> <td>F</td> <td>0</td> <td>6</td> <td>6</td> <td>0'ers;0'erd</td> <td>4</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット						実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	イミディエイト	AND.L	#xx:32,ERd	7	A	6	0	erd	IMM		6	レジスタ直接	AND.L	ERs,ERd	0	1	F	0	6	6	0'ers;0'erd	4
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット							実行 ステート 数																											
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト																																
イミディエイト	AND.L	#xx:32,ERd	7	A	6	0	erd	IMM		6																														
レジスタ直接	AND.L	ERs,ERd	0	1	F	0	6	6	0'ers;0'erd	4																														
<ul style="list-style-type: none"> ●注意事項 																																								

2.2.5 ANDC

ANDC (AND Control register)		CCRとの論理積					
●オペレーション CCR^ #IMM→CCR	●コンディションコード I UI H U N Z V C ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑						
●アセンブラフォーマット ANDC #xx : 8, CCR	I : 実行結果の対応するビットの値が格納されます。 UI : 実行結果の対応するビットの値が格納されます。 H : 実行結果の対応するビットの値が格納されます。 U : 実行結果の対応するビットの値が格納されます。 N : 実行結果の対応するビットの値が格納されます。 Z : 実行結果の対応するビットの値が格納されます。 V : 実行結果の対応するビットの値が格納されます。 C : 実行結果の対応するビットの値が格納されます。						
●オペランドサイズ バイト							
●説明 CCRの内容とイミディエイトデータの論理積をとり、結果をCCRに格納します。 なお、本命令の実行終了時点では、NMIを含めてすべての割り込みは受け付けられません。							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
イミディエイト	ANDC	#xx:8,CCR	0	6	IMM		2
●注意事項							

2. 各命令の説明

2.2.6 BAND

BAND (Bit AND)		ビット論理積																																					
<ul style="list-style-type: none"> ●オペレーション C^ (<ビット番号>of<EAd>) →C 	<ul style="list-style-type: none"> ●コンディションコード <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行結果が保持されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	↑																				
I	UI	H	U	N	Z	V	C																																
—	—	—	—	—	—	—	↑																																
<ul style="list-style-type: none"> ●アセンブラフォーマット BAND #xx:3, <EAd> 																																							
<ul style="list-style-type: none"> ●オペランドサイズ バイト 																																							
<ul style="list-style-type: none"> ●説明 <p>デスティネーションオペランドの指定された1ビットとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。</p> <div style="text-align: center;"> <p>ビット番号 7 6 5 4 3 2 1 0</p> <p><EAd> → [7-bit register]</p> <p>#xx:3で指定</p> <p>c [] ^ [] → [] c</p> </div>																																							
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																							
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BAND</td> <td>#xx:3,Rd</td> <td>7</td> <td>6</td> <td>0:IMM:rd</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BAND</td> <td>#xx:3,@ERd</td> <td>7</td> <td>C</td> <td>0:erd:0</td> <td>7 6 0:IMM:0</td> <td>6</td> </tr> <tr> <td>絶対アドレス</td> <td>BAND</td> <td>#xx:3,@aa:8</td> <td>7</td> <td>E</td> <td>abs</td> <td>7 6 0:IMM:0</td> <td>6</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	BAND	#xx:3,Rd	7	6	0:IMM:rd		2	レジスタ間接	BAND	#xx:3,@ERd	7	C	0:erd:0	7 6 0:IMM:0	6	絶対アドレス	BAND	#xx:3,@aa:8	7	E	abs	7 6 0:IMM:0	6
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																																
			第1バイト	第2バイト	第3バイト	第4バイト																																	
レジスタ直接	BAND	#xx:3,Rd	7	6	0:IMM:rd		2																																
レジスタ間接	BAND	#xx:3,@ERd	7	C	0:erd:0	7 6 0:IMM:0	6																																
絶対アドレス	BAND	#xx:3,@aa:8	7	E	abs	7 6 0:IMM:0	6																																
<ul style="list-style-type: none"> ●注意事項 <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																							

2.2.7 Bcc

Bcc (Branch conditionally)		条件付分岐										
<p>●オペレーション</p> <p>If condition is true, then PC+disp→PC else next;</p>	<p>●コンディションコード</p> <p>I U I H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H: 演算前の値が保持されます。 N: 演算前の値が保持されます。 Z: 演算前の値が保持されます。 V: 演算前の値が保持されます。 C: 演算前の値が保持されます。</p>				—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—					
<p>●アセンブラフォーマット</p> <p>Bcc disp</p> <p style="margin-left: 20px;">└─ コンディションフィールド</p>												
<p>●オペランドサイズ</p> <p>—</p>												
<p>●説明</p> <p>コンディションフィールド(cc)で指定された条件が成立していると、PCにディスプレイメントを加えたアドレスに分岐し、条件が不成立の場合は次の命令を実行します。アドレス計算に用いられるPCの値は本命令の直後の命令の先頭アドレスです。ディスプレイメントは符号付き8ビットまたは16ビットデータで、分岐できる範囲は本命令に対して-126~+128、-32766~+32768バイトです。</p>												
ニーモニック	説明	cc	条件	符号と条件の対応*								
BRA(BT)	Always(True)	0000	True									
BRN(BF)	Never(False)	0001	False									
BHI	High	0010	C<Z=0	X>Y 符号なし								
BLS	Low or Same	0011	C<Z=1	X≤Y 符号なし								
BCC(BHS)	Carry Clear(High or Same)	0100	C=0	X≥Y 符号なし								
BCS(BLO)	Carry Set(LOW)	0101	C=1	X<Y 符号なし								
BNE	Not Equal	0110	Z=0	X≠Y 符号なし/あり								
BEQ	Equal	0111	Z=1	X=Y 符号なし/あり								
BVC	oVerflow Clear	1000	V=0									
BVS	oVerflow Set	1001	V=1									
BPL	PLus	1010	N=0									
BMI	MInus	1011	N=1									
BGE	Greater or Equal	1100	N⊕V=0	X≥Y 符号あり								
BLT	Less Than	1101	N⊕V=1	X<Y 符号あり								
BGT	Greater Than	1110	Z∨(N⊕V)=0	X>Y 符号あり								
BLE	Less or Equal	1111	Z∨(N⊕V)=1	X≤Y 符号あり								
<p>【注】 * 直前の命令がCMP命令のとき、Xは汎用レジスタの内容(デスティネーションオペランド)、Yはソースオペランドです。</p>												

2. 各命令の説明

2.2.7 Bcc

Bcc (Branch conditionally)			条件付分岐				
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
プログラム カウンタ相対	BRA(BT)	d:8	4	0	disp		4
		d:16	5	8	0	0	disp
プログラム カウンタ相対	BRN(BF)	d:8	4	1	disp		4
		d:16	5	8	1	0	disp
プログラム カウンタ相対	BHI	d:8	4	2	disp		4
		d:16	5	8	2	0	disp
プログラム カウンタ相対	BLS	d:8	4	3	disp		4
		d:16	5	8	3	0	disp
プログラム カウンタ相対	BCC(BHS)	d:8	4	4	disp		4
		d:16	5	8	4	0	disp
プログラム カウンタ相対	BCS(BLO)	d:8	4	5	disp		4
		d:16	5	8	5	0	disp
プログラム カウンタ相対	BNE	d:8	4	6	disp		4
		d:16	5	8	6	0	disp
プログラム カウンタ相対	BEQ	d:8	4	7	disp		4
		d:16	5	8	7	0	disp
プログラム カウンタ相対	BVC	d:8	4	8	disp		4
		d:16	5	8	8	0	disp
プログラム カウンタ相対	BVS	d:8	4	9	disp		4
		d:16	5	8	9	0	disp
プログラム カウンタ相対	BPL	d:8	4	A	disp		4
		d:16	5	8	A	0	disp
プログラム カウンタ相対	BMI	d:8	4	B	disp		4
		d:16	5	8	B	0	disp
プログラム カウンタ相対	BGE	d:8	4	C	disp		4
		d:16	5	8	C	0	disp
プログラム カウンタ相対	BLT	d:8	4	D	disp		4
		d:16	5	8	D	0	disp
プログラム カウンタ相対	BGT	d:8	4	E	disp		4
		d:16	5	8	E	0	disp
プログラム カウンタ相対	BLE	d:8	4	F	disp		4
		d:16	5	8	F	0	disp

●注意事項

1. 分岐先アドレスは、必ず偶数になるようにしてください。
2. BRA、BRN、BCC、BCSの機械語はそれぞれBT、BF、BHS、BLOと同一です。

2.2.8 BCLR

BCLR (Bit CLear)		ビットクリア																																																																																																								
<ul style="list-style-type: none"> ●オペレーション 0 → (<ビット番号>of<EAd>) 	<ul style="list-style-type: none"> ●コンディションコード I UI H U N Z V C <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。 			—	—	—	—	—	—	—	—																																																																																															
—	—	—	—	—	—	—	—																																																																																																			
<ul style="list-style-type: none"> ●アセンブラフォーマット BCLR #xx:3, <EAd> BCLR Rn, <EAd> 																																																																																																										
<ul style="list-style-type: none"> ●オペランドサイズ バイト 																																																																																																										
<ul style="list-style-type: none"> ●説明 デスティネーションオペランドの指定された1ビットを0にクリアします。ビット番号は、3ビットのイミディエイトデータまたは8ビットレジスタRnの内容の低位3ビットで指定されます。指定された1ビットのテストは行いません（コンディションコードは変化しません）。 <div style="text-align: center;"> </div>																																																																																																										
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7 Rn : R0L~R7L, R0H~R7H 																																																																																																										
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BCLR</td> <td>#xx:3,Rd</td> <td>7</td><td>2</td> <td>0</td><td>IMM</td> <td>rd</td> <td></td><td></td> <td></td><td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BCLR</td> <td>#xx:3,@ERd</td> <td>7</td><td>D</td> <td>0</td><td>erd</td> <td>0</td> <td>7</td><td>2</td> <td>0</td><td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BCLR</td> <td>#xx:3,@aa:8</td> <td>7</td><td>F</td> <td></td><td>abs</td> <td></td> <td>7</td><td>2</td> <td>0</td><td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>レジスタ直接</td> <td>BCLR</td> <td>Rn,Rd</td> <td>6</td><td>2</td> <td></td><td>rn</td> <td>rd</td> <td></td><td></td> <td></td><td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BCLR</td> <td>Rn,@ERd</td> <td>7</td><td>D</td> <td>0</td><td>erd</td> <td>0</td> <td>6</td><td>2</td> <td></td><td>rn</td> <td>0</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BCLR</td> <td>Rn,@aa:8</td> <td>7</td><td>F</td> <td></td><td>abs</td> <td></td> <td>6</td><td>2</td> <td></td><td>rn</td> <td>0</td> <td>8</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BCLR	#xx:3,Rd	7	2	0	IMM	rd					2	レジスタ間接	BCLR	#xx:3,@ERd	7	D	0	erd	0	7	2	0	IMM	0	8	絶対アドレス	BCLR	#xx:3,@aa:8	7	F		abs		7	2	0	IMM	0	8	レジスタ直接	BCLR	Rn,Rd	6	2		rn	rd						2	レジスタ間接	BCLR	Rn,@ERd	7	D	0	erd	0	6	2		rn	0	8	絶対アドレス	BCLR	Rn,@aa:8	7	F		abs		6	2		rn	0	8
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数																																																																																															
			第1バイト		第2バイト		第3バイト		第4バイト																																																																																																	
レジスタ直接	BCLR	#xx:3,Rd	7	2	0	IMM	rd					2																																																																																														
レジスタ間接	BCLR	#xx:3,@ERd	7	D	0	erd	0	7	2	0	IMM	0	8																																																																																													
絶対アドレス	BCLR	#xx:3,@aa:8	7	F		abs		7	2	0	IMM	0	8																																																																																													
レジスタ直接	BCLR	Rn,Rd	6	2		rn	rd						2																																																																																													
レジスタ間接	BCLR	Rn,@ERd	7	D	0	erd	0	6	2		rn	0	8																																																																																													
絶対アドレス	BCLR	Rn,@aa:8	7	F		abs		6	2		rn	0	8																																																																																													
<ul style="list-style-type: none"> ●注意事項 @aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。 																																																																																																										

2. 各命令の説明

2.2.9 BIAND

BIAND (Bit Invert AND)		ビット論理積																																																																	
<p>●オペレーション C[^][~ (<ビット番号>of<EAd>)]→C</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↓</td> </tr> </table> <p>H： 実行前の値が保持されます。 N： 実行前の値が保持されます。 Z： 実行前の値が保持されます。 V： 実行前の値が保持されます。 C： 実行結果が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	↓																																																	
I	UI	H	U	N	Z	V	C																																																												
—	—	—	—	—	—	—	↓																																																												
<p>●アセンブラフォーマット BIAND #xx:3, <EAd></p>																																																																			
<p>●オペランドサイズ バイト</p>																																																																			
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットを反転し、これとキャリフラグの論理積をとり、結果をキャリフラグに格納します。ビット番号は、3ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。</p> <div style="text-align: center;"> </div>																																																																			
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																																																			
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> <th>第5バイト</th> <th>第6バイト</th> <th>第7バイト</th> <th>第8バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BIAND</td> <td>#xx:3,Rd</td> <td>7</td> <td>6</td> <td>1</td> <td>IMM</td> <td>rd</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BIAND</td> <td>#xx:3,@ERd</td> <td>7</td> <td>C</td> <td>0</td> <td>erd</td> <td>0</td> <td>7</td> <td>6</td> <td>1</td> <td>IMM</td> <td>0</td> <td></td> <td>6</td> </tr> <tr> <td>絶対アドレス</td> <td>BIAND</td> <td>#xx:3,@aa:8</td> <td>7</td> <td>E</td> <td></td> <td>abs</td> <td></td> <td>7</td> <td>6</td> <td>1</td> <td>IMM</td> <td>0</td> <td></td> <td>6</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>			アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	レジスタ直接	BIAND	#xx:3,Rd	7	6	1	IMM	rd							2	レジスタ間接	BIAND	#xx:3,@ERd	7	C	0	erd	0	7	6	1	IMM	0		6	絶対アドレス	BIAND	#xx:3,@aa:8	7	E		abs		7	6	1	IMM	0		6
アドレッシングモード*	ニーモニック	オペランド形式				インストラクションフォーマット									実行ステート数																																																				
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト																																																									
レジスタ直接	BIAND	#xx:3,Rd	7	6	1	IMM	rd							2																																																					
レジスタ間接	BIAND	#xx:3,@ERd	7	C	0	erd	0	7	6	1	IMM	0		6																																																					
絶対アドレス	BIAND	#xx:3,@aa:8	7	E		abs		7	6	1	IMM	0		6																																																					
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																			

2.2.10 BILD

BILD (Bit Invert LoaD)		ビット転送									
●オペレーション ~ (<ビット番号>of<EAd>) →C	●コンディションコード I UI H U N Z V C — — — — — — — ↓										
●アセンブラフォーマット BILD #xx:3, <EAd>	H : 実行前の値が保持されます。 N : 実行前の値が保持されます。 Z : 実行前の値が保持されます。 V : 実行前の値が保持されます。 C : 指定ビットの内容が反転されて格納されます。										
●オペランドサイズ バイト											
●説明 デスティネーションオペランドの指定された1ビットを反転し、これをキャリフラグに転送します。ビット番号は、3ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。											
●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7											
●オペランド形式と実行ステート数											
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数				
			第1バイト	第2バイト	第3バイト	第4バイト					
レジスタ直接	BILD	#xx:3,Rd	7	7	1 IMM	rd		2			
レジスタ間接	BILD	#xx:3,@ERd	7	C	0 erd	0	7	7	1 IMM	0	6
絶対アドレス	BILD	#xx:3,@aa:8	7	E		abs	7	7	1 IMM	0	6
【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。											
●注意事項 @aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。											

2. 各命令の説明

2.2.11 BIOR

BIOR (Bit Invert inclusive OR)		ビット論理和																																																																		
<p>●オペレーション Cv[~ (<ビット番号>of<EAd>)]→C</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↓</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行結果が格納されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	↓																																																	
I	UI	H	U	N	Z	V	C																																																													
—	—	—	—	—	—	—	↓																																																													
<p>●アセンブラフォーマット BIOR #xx:3, <EAd></p>																																																																				
<p>●オペランドサイズ バイト</p>																																																																				
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットを反転し、これとキャリフラグの論理和をとり、結果をキャリフラグに格納します。ビット番号は、3ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。</p> <div style="text-align: center;"> <p>ビット番号 7 0</p> <p><EAd></p> <p>#xx:3で指定</p> <p>反転</p> <p>C v C</p> </div>																																																																				
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																																																				
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BIOR</td> <td>#xx:3,Rd</td> <td>7</td> <td>4</td> <td>1</td> <td>IMM</td> <td>rd</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BIOR</td> <td>#xx:3,@ERd</td> <td>7</td> <td>C</td> <td>0</td> <td>erd</td> <td>0</td> <td>7</td> <td>4</td> <td>1</td> <td>IMM</td> <td>0</td> <td></td> <td>6</td> </tr> <tr> <td>絶対アドレス</td> <td>BIOR</td> <td>#xx:3,@aa:8</td> <td>7</td> <td>E</td> <td></td> <td>abs</td> <td></td> <td>7</td> <td>4</td> <td>1</td> <td>IMM</td> <td>0</td> <td></td> <td>6</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BIOR	#xx:3,Rd	7	4	1	IMM	rd							2	レジスタ間接	BIOR	#xx:3,@ERd	7	C	0	erd	0	7	4	1	IMM	0		6	絶対アドレス	BIOR	#xx:3,@aa:8	7	E		abs		7	4	1	IMM	0		6
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数																																																									
			第1バイト		第2バイト		第3バイト		第4バイト																																																											
レジスタ直接	BIOR	#xx:3,Rd	7	4	1	IMM	rd							2																																																						
レジスタ間接	BIOR	#xx:3,@ERd	7	C	0	erd	0	7	4	1	IMM	0		6																																																						
絶対アドレス	BIOR	#xx:3,@aa:8	7	E		abs		7	4	1	IMM	0		6																																																						
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																				

2.2.12 BIST

BIST (Bit Invert STore)		ビット転送																																																															
<p>●オペレーション ~C→ (<ビット番号>of<EAd>)</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	—																																														
I	UI	H	U	N	Z	V	C																																																										
—	—	—	—	—	—	—	—																																																										
<p>●アセンブラフォーマット BIST #xx:3, <EAd></p>																																																																	
<p>●オペランドサイズ バイト</p>																																																																	
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットのロケーションに、キャリフラグの内容を反転して転送します。ビット番号は、3ビットのイミディエイトデータで指定されます。なお、デスティネーションオペランドの指定されない他のビットの内容は変化しません。</p> <div style="text-align: center;"> <p>ビット番号 7 0</p> <p><EAd> →</p> <p>#xx:3で指定</p> <p>C → 反転</p> </div>																																																																	
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																																																	
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BIST</td> <td>#xx:3,Rd</td> <td>6</td><td>7</td> <td>1</td><td>IMM</td> <td>rd</td> <td></td><td></td> <td></td><td></td> <td></td><td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BIST</td> <td>#xx:3,@ERd</td> <td>7</td><td>D</td> <td>0</td><td>erd</td> <td>0</td> <td>6</td><td>7</td> <td>1</td><td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BIST</td> <td>#xx:3,@aa:8</td> <td>7</td><td>F</td> <td></td><td>abs</td> <td></td> <td>6</td><td>7</td> <td>1</td><td>IMM</td> <td>0</td> <td>8</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BIST	#xx:3,Rd	6	7	1	IMM	rd						2	レジスタ間接	BIST	#xx:3,@ERd	7	D	0	erd	0	6	7	1	IMM	0	8	絶対アドレス	BIST	#xx:3,@aa:8	7	F		abs		6	7	1	IMM	0	8
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数																																																						
			第1バイト		第2バイト		第3バイト		第4バイト																																																								
レジスタ直接	BIST	#xx:3,Rd	6	7	1	IMM	rd						2																																																				
レジスタ間接	BIST	#xx:3,@ERd	7	D	0	erd	0	6	7	1	IMM	0	8																																																				
絶対アドレス	BIST	#xx:3,@aa:8	7	F		abs		6	7	1	IMM	0	8																																																				
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																	

2. 各命令の説明

2.2.13 BIXOR

BIXOR (Bit Invert eXclusive OR)		ビット排他的論理和																																				
<p>●オペレーション C⊕ [~ (<ビット番号>of<EAd>)]→C</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↑</td> </tr> </table> <p>H： 実行前の値が保持されます。 N： 実行前の値が保持されます。 Z： 実行前の値が保持されます。 V： 実行前の値が保持されます。 C： 実行結果が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	↑																				
I	UI	H	U	N	Z	V	C																															
—	—	—	—	—	—	—	↑																															
<p>●アセンブラフォーマット BIXOR #xx:3, <EAd></p>																																						
<p>●オペランドサイズ バイト</p>																																						
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットを反転し、これとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。ビット番号は、3ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。</p> <div style="text-align: center;"> <p>ビット番号 7 0</p> <p><EAd></p> <p>#xx:3で指定</p> <p>反転</p> <p>C ⊕ [] → C</p> </div>																																						
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																						
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BIXOR</td> <td>#xx:3,Rd</td> <td>7</td> <td>5</td> <td>1 IMM rd</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BIXOR</td> <td>#xx:3,@ERd</td> <td>7</td> <td>C</td> <td>0: erd 0</td> <td>7 5 1 IMM 0</td> <td>6</td> </tr> <tr> <td>絶対アドレス</td> <td>BIXOR</td> <td>#xx:3,@aa:8</td> <td>7</td> <td>E</td> <td>abs</td> <td>7 5 1 IMM 0</td> <td>6</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>			アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	BIXOR	#xx:3,Rd	7	5	1 IMM rd		2	レジスタ間接	BIXOR	#xx:3,@ERd	7	C	0: erd 0	7 5 1 IMM 0	6	絶対アドレス	BIXOR	#xx:3,@aa:8	7	E	abs	7 5 1 IMM 0	6
アドレッシングモード*	ニーモニック	オペランド形式				インストラクションフォーマット					実行ステート数																											
			第1バイト	第2バイト	第3バイト	第4バイト																																
レジスタ直接	BIXOR	#xx:3,Rd	7	5	1 IMM rd		2																															
レジスタ間接	BIXOR	#xx:3,@ERd	7	C	0: erd 0	7 5 1 IMM 0	6																															
絶対アドレス	BIXOR	#xx:3,@aa:8	7	E	abs	7 5 1 IMM 0	6																															
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																						

2.2.14 BLD

BLD (Bit LoaD)		ビット転送																																																															
<p>●オペレーション (<ビット番号>of<EAd>) →C</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↑</td> </tr> </table> <p>H : 実行前の値が保持されます。 N : 実行前の値が保持されます。 Z : 実行前の値が保持されます。 V : 実行前の値が保持されます。 C : 指定ビットの内容が格納されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	↑																																														
I	UI	H	U	N	Z	V	C																																																										
—	—	—	—	—	—	—	↑																																																										
<p>●アセンブラフォーマット BLD #xx:3, <EAd></p>																																																																	
<p>●オペランドサイズ バイト</p>																																																																	
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットをキャリフラグに転送します。ビット番号は、3ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。</p> <div style="text-align: center;"> </div>																																																																	
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																																																	
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BLD</td> <td>#xx:3,Rd</td> <td>7</td> <td>7</td> <td>0</td> <td>IMM</td> <td>rd</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BLD</td> <td>#xx:3,@ERd</td> <td>7</td> <td>C</td> <td>0</td> <td>erd</td> <td>0</td> <td>7</td> <td>7</td> <td>0</td> <td>IMM</td> <td>0</td> <td>6</td> </tr> <tr> <td>絶対アドレス</td> <td>BLD</td> <td>#xx:3,@aa:8</td> <td>7</td> <td>E</td> <td></td> <td>abs</td> <td></td> <td>7</td> <td>7</td> <td>0</td> <td>IMM</td> <td>0</td> <td>6</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BLD	#xx:3,Rd	7	7	0	IMM	rd						2	レジスタ間接	BLD	#xx:3,@ERd	7	C	0	erd	0	7	7	0	IMM	0	6	絶対アドレス	BLD	#xx:3,@aa:8	7	E		abs		7	7	0	IMM	0	6
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数																																																						
			第1バイト		第2バイト		第3バイト		第4バイト																																																								
レジスタ直接	BLD	#xx:3,Rd	7	7	0	IMM	rd						2																																																				
レジスタ間接	BLD	#xx:3,@ERd	7	C	0	erd	0	7	7	0	IMM	0	6																																																				
絶対アドレス	BLD	#xx:3,@aa:8	7	E		abs		7	7	0	IMM	0	6																																																				
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																	

2. 各命令の説明

2.2.15 BNOT

BNOT (Bit NOT)		ビット転送																																																																																																							
<p>●オペレーション ~ (<ビット番号>of<EAd>) → (<ビット番号>of<EAd>)</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	—																																																																																							
I	UI	H	U	N	Z	V	C																																																																																																		
—	—	—	—	—	—	—	—																																																																																																		
<p>●アセンブラフォーマット BNOT #xx:3, <EAd> BNOT Rn, <EAd></p>																																																																																																									
<p>●オペランドサイズ バイト</p>																																																																																																									
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットを反転します。ビット番号は、3ビットのイミディエイトデータまたは8ビットレジスタの内容の下位3ビットで指定されます。指定された1ビットのテストは行いません。(コンディションコードは変化しません。)</p>																																																																																																									
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7 Rn : R0L~R7L, R0H~R7H</p>																																																																																																									
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BNOT</td> <td>#xx:3,Rd</td> <td>7</td><td>1</td> <td>0</td><td>IMM</td> <td>rd</td> <td></td><td></td> <td></td><td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BNOT</td> <td>#xx:3,@ERd</td> <td>7</td><td>D</td> <td>0</td><td>erd</td> <td>0</td> <td>7</td><td>1</td> <td>0</td><td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BNOT</td> <td>#xx:3,@aa:8</td> <td>7</td><td>F</td> <td></td><td>abs</td> <td></td> <td>7</td><td>1</td> <td>0</td><td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>レジスタ直接</td> <td>BNOT</td> <td>Rn,Rd</td> <td>6</td><td>1</td> <td>rn</td><td>rd</td> <td></td><td></td><td></td> <td></td><td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BNOT</td> <td>Rn,@ERd</td> <td>7</td><td>D</td> <td>0</td><td>erd</td> <td>0</td> <td>6</td><td>1</td> <td>rn</td><td>0</td> <td></td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BNOT</td> <td>Rn,@aa:8</td> <td>7</td><td>F</td> <td></td><td>abs</td> <td></td> <td>6</td><td>1</td> <td>rn</td><td>0</td> <td></td> <td>8</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>			アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BNOT	#xx:3,Rd	7	1	0	IMM	rd					2	レジスタ間接	BNOT	#xx:3,@ERd	7	D	0	erd	0	7	1	0	IMM	0	8	絶対アドレス	BNOT	#xx:3,@aa:8	7	F		abs		7	1	0	IMM	0	8	レジスタ直接	BNOT	Rn,Rd	6	1	rn	rd							2	レジスタ間接	BNOT	Rn,@ERd	7	D	0	erd	0	6	1	rn	0		8	絶対アドレス	BNOT	Rn,@aa:8	7	F		abs		6	1	rn	0		8
アドレッシングモード*	ニーモニック	オペランド形式				インストラクションフォーマット									実行ステート数																																																																																										
			第1バイト		第2バイト		第3バイト		第4バイト																																																																																																
レジスタ直接	BNOT	#xx:3,Rd	7	1	0	IMM	rd					2																																																																																													
レジスタ間接	BNOT	#xx:3,@ERd	7	D	0	erd	0	7	1	0	IMM	0	8																																																																																												
絶対アドレス	BNOT	#xx:3,@aa:8	7	F		abs		7	1	0	IMM	0	8																																																																																												
レジスタ直接	BNOT	Rn,Rd	6	1	rn	rd							2																																																																																												
レジスタ間接	BNOT	Rn,@ERd	7	D	0	erd	0	6	1	rn	0		8																																																																																												
絶対アドレス	BNOT	Rn,@aa:8	7	F		abs		6	1	rn	0		8																																																																																												
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																																																									

2.2.16 BOR

BOR (Bit inclusive OR)		ビット論理和					
●オペレーション Cv (<ビット番号>of<EAd>) →C	●コンディションコード I UI H U N Z V C — — — — — — — †						
●アセンブラフォーマット BOR #xx:3, <EAd>	H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行結果が格納されます。						
●オペランドサイズ バイト							
●説明 デスティネーションオペランドの指定された1ビットとキャリフラグとの論理和を取り、結果をキャリフラグに格納します。ビット番号は、3ビットのイミディエイトデータで指定されます。デスティネーションの内容は変化しません。							
<p>ビット番号 7 0 <EAd> → [7 6 5 4 3 2 1 0] #xx:3で指定 ↓ C [] V [] → [] C</p>							
●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7							
●オペランド形式と実行ステート数							
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数
			第1バイト	第2バイト	第3バイト	第4バイト	
レジスタ直接	BOR	#xx:3,Rd	7 4	0 IMM rd			2
レジスタ間接	BOR	#xx:3,@ERd	7 C	0 erd 0	7 4	0 IMM 0	6
絶対アドレス	BOR	#xx:3,@aa:8	7 E	abs	7 4	0 IMM 0	6
【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。							
●注意事項 @aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。							

2. 各命令の説明

2.2.17 BSET

BSET (Bit SET)		ビットセット																																																																																																								
<p>●オペレーション 1→ (<ビット番号>of<EAd>)</p>	<p>●コンディションコード</p> <p>I U I H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H : 実行前の値が保持されます。 N : 実行前の値が保持されます。 Z : 実行前の値が保持されます。 V : 実行前の値が保持されます。 C : 実行前の値が保持されます。</p>			—	—	—	—	—	—	—	—																																																																																															
—	—	—	—	—	—	—	—																																																																																																			
<p>●アセンブラフォーマット BSET #xx:3, <EAd> BSET Rn, <EAd></p>																																																																																																										
<p>●オペランドサイズ バイト</p>																																																																																																										
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットを1にセットします。ビット番号は、3ビットのイミディエイトデータまたは8ビットレジスタの内容の下位3ビットで指定されます。指定された1ビットのテストは行いません。(コンディションコードは変化しません。)</p> <div style="text-align: center;"> </div>																																																																																																										
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7 Rn : R0L~R7L, R0H~R7H</p>																																																																																																										
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BSET</td> <td>#xx:3,Rd</td> <td>7</td><td>0</td> <td>0</td><td>IMM</td> <td>rd</td> <td></td><td></td> <td></td><td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BSET</td> <td>#xx:3,@ERd</td> <td>7</td><td>D</td> <td>0</td><td>erd</td> <td>0</td> <td>7</td><td>0</td> <td>0</td><td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BSET</td> <td>#xx:3,@aa:8</td> <td>7</td><td>F</td> <td></td><td>abs</td> <td></td> <td>7</td><td>0</td> <td>0</td><td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>レジスタ直接</td> <td>BSET</td> <td>Rn,Rd</td> <td>6</td><td>0</td> <td>rn</td><td>rd</td> <td></td><td></td><td></td> <td></td><td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BSET</td> <td>Rn,@ERd</td> <td>7</td><td>D</td> <td>0</td><td>erd</td> <td>0</td> <td>6</td><td>0</td> <td>rn</td><td>0</td> <td>0</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BSET</td> <td>Rn,@aa:8</td> <td>7</td><td>F</td> <td></td><td>abs</td> <td></td> <td>6</td><td>0</td> <td>rn</td><td>0</td> <td>0</td> <td>8</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BSET	#xx:3,Rd	7	0	0	IMM	rd					2	レジスタ間接	BSET	#xx:3,@ERd	7	D	0	erd	0	7	0	0	IMM	0	8	絶対アドレス	BSET	#xx:3,@aa:8	7	F		abs		7	0	0	IMM	0	8	レジスタ直接	BSET	Rn,Rd	6	0	rn	rd							2	レジスタ間接	BSET	Rn,@ERd	7	D	0	erd	0	6	0	rn	0	0	8	絶対アドレス	BSET	Rn,@aa:8	7	F		abs		6	0	rn	0	0	8
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数																																																																																															
			第1バイト		第2バイト		第3バイト		第4バイト																																																																																																	
レジスタ直接	BSET	#xx:3,Rd	7	0	0	IMM	rd					2																																																																																														
レジスタ間接	BSET	#xx:3,@ERd	7	D	0	erd	0	7	0	0	IMM	0	8																																																																																													
絶対アドレス	BSET	#xx:3,@aa:8	7	F		abs		7	0	0	IMM	0	8																																																																																													
レジスタ直接	BSET	Rn,Rd	6	0	rn	rd							2																																																																																													
レジスタ間接	BSET	Rn,@ERd	7	D	0	erd	0	6	0	rn	0	0	8																																																																																													
絶対アドレス	BSET	Rn,@aa:8	7	F		abs		6	0	rn	0	0	8																																																																																													
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																																																										

2.2.18 BSR

BSR (Branch to SubRoutine)		サブルーチン分岐						
●オペレーション PC→@-SP PC+disp→SP	●コンディションコード I U I H U N Z V C — — — — — — — —							
●アセンブラフォーマット BSR disp	H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。							
●オペランドサイズ —								
●説明 指定されたアドレスにサブルーチン分岐します。PCの内容をリスタートアドレスとしてスタックに退避し、PCにディスプレースメントを加えたアドレスに分岐します。スタックに退避されるPCの内容は本命令の直後の命令の先頭アドレスです。ディスプレースメントは符号付き8ビットまたは16ビットで、分岐できる範囲は本命令に対して-126~+128、-32766~+32768バイトです。								
●オペランド形式と実行ステート数								
アドレッシング モード*	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート数				
			第1バイト	第2バイト	第3バイト	第4バイト	7ビット ノーマル	8ビット アドバンス
プログラム カウンタ相対	BSR	d:8	5	5	disp		6	8
		d:16	5	C	0	0	8	10
●注意事項 ノーマルモードとアドバンスモードではスタックの構造が異なりますので、注意してください。ノーマルモードのとき退避されるPCの内容は、下位16ビットのみです。								
ノーマルモード		アドバンスモード						
分岐先アドレスは、必ず偶数になるようにしてください。								

2. 各命令の説明

2.2.19 BST

BST (Bit STore)		サブルーチン分岐																																																														
<p>●オペレーション C→ (<ビット番号>of<EAd>)</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	—																																													
I	UI	H	U	N	Z	V	C																																																									
—	—	—	—	—	—	—	—																																																									
<p>●アセンブラフォーマット BST #xx:3, <EAd></p>																																																																
<p>●オペランドサイズ バイト</p>																																																																
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットのロケーションに、キャリフラグの内容を転送します。ビット番号は、3ビットのイミディエイトデータで指定されます。</p>																																																																
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																																																
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BST</td> <td>#xx:3,Rd</td> <td>6</td> <td>7</td> <td>0</td> <td>IMM</td> <td>rd</td> <td></td> <td></td> <td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BST</td> <td>#xx:3,@ERd</td> <td>7</td> <td>D</td> <td>0</td> <td>erd</td> <td>0</td> <td>6</td> <td>7</td> <td>0</td> <td>IMM</td> <td>0</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>BST</td> <td>#xx:3,@aa:8</td> <td>7</td> <td>F</td> <td></td> <td>abs</td> <td></td> <td>6</td> <td>7</td> <td>0</td> <td>IMM</td> <td>0</td> <td>8</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BST	#xx:3,Rd	6	7	0	IMM	rd					2	レジスタ間接	BST	#xx:3,@ERd	7	D	0	erd	0	6	7	0	IMM	0	8	絶対アドレス	BST	#xx:3,@aa:8	7	F		abs		6	7	0	IMM	0	8
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数																																																					
			第1バイト		第2バイト		第3バイト		第4バイト																																																							
レジスタ直接	BST	#xx:3,Rd	6	7	0	IMM	rd					2																																																				
レジスタ間接	BST	#xx:3,@ERd	7	D	0	erd	0	6	7	0	IMM	0	8																																																			
絶対アドレス	BST	#xx:3,@aa:8	7	F		abs		6	7	0	IMM	0	8																																																			
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																

2.2.20 BTST

BTST (Bit TeST)		ビットテスト																																																																													
<p>●オペレーション ~ (<ビット番号>of<EAd>) →Z</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>↓</td><td>—</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 指定したビットが0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	—	↓	—	—																																																												
I	UI	H	U	N	Z	V	C																																																																								
—	—	—	—	—	↓	—	—																																																																								
<p>●アセンブラフォーマット BTST #xx:3, <EAd> BTST Rn, <EAd></p>																																																																															
<p>●オペランドサイズ バイト</p>																																																																															
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットの状態を調べて、その結果をゼロフラグに反映します。ビット番号は、3ビットのイミディエイトデータまたは8ビットレジスタの内容の下位3ビットで指定されます。デスティネーションの内容は変化しません。</p> <div style="text-align: center;"> </div>																																																																															
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7 Rn : R0L~R7L, R0H~R7H</p>																																																																															
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BTST</td> <td>#xx:3,Rd</td> <td>7</td><td>3</td> <td>0: IMM</td><td>rd</td> <td></td><td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BTST</td> <td>#xx:3,@ERd</td> <td>7</td><td>C</td> <td>0: erd</td><td>0</td> <td>7</td><td>3</td> <td>0: IMM</td><td>0</td> </tr> <tr> <td>絶対アドレス</td> <td>BTST</td> <td>#xx:3,@aa:8</td> <td>7</td><td>E</td> <td colspan="2">abs</td> <td>7</td><td>3</td> <td>0: IMM</td><td>0</td> </tr> <tr> <td>レジスタ直接</td> <td>BTST</td> <td>Rn,Rd</td> <td>6</td><td>3</td> <td>rn</td><td>rd</td> <td></td><td></td><td></td><td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BTST</td> <td>Rn,@ERd</td> <td>7</td><td>C</td> <td>0: erd</td><td>0</td> <td>6</td><td>3</td> <td>rn</td><td>0</td> </tr> <tr> <td>絶対アドレス</td> <td>BTST</td> <td>Rn,@aa:8</td> <td>7</td><td>E</td> <td colspan="2">abs</td> <td>6</td><td>3</td> <td>rn</td><td>0</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>				アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	BTST	#xx:3,Rd	7	3	0: IMM	rd		2	レジスタ間接	BTST	#xx:3,@ERd	7	C	0: erd	0	7	3	0: IMM	0	絶対アドレス	BTST	#xx:3,@aa:8	7	E	abs		7	3	0: IMM	0	レジスタ直接	BTST	Rn,Rd	6	3	rn	rd				2	レジスタ間接	BTST	Rn,@ERd	7	C	0: erd	0	6	3	rn	0	絶対アドレス	BTST	Rn,@aa:8	7	E	abs		6	3	rn	0
アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																																																																								
			第1バイト	第2バイト	第3バイト	第4バイト																																																																									
レジスタ直接	BTST	#xx:3,Rd	7	3	0: IMM	rd		2																																																																							
レジスタ間接	BTST	#xx:3,@ERd	7	C	0: erd	0	7	3	0: IMM	0																																																																					
絶対アドレス	BTST	#xx:3,@aa:8	7	E	abs		7	3	0: IMM	0																																																																					
レジスタ直接	BTST	Rn,Rd	6	3	rn	rd				2																																																																					
レジスタ間接	BTST	Rn,@ERd	7	C	0: erd	0	6	3	rn	0																																																																					
絶対アドレス	BTST	Rn,@aa:8	7	E	abs		6	3	rn	0																																																																					
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																																															

2. 各命令の説明

2.2.21 BXOR

BXOR (Bit eXclusive OR)		ビット排他的論理和																																																													
<p>●オペレーション C⊕ (<ビット番号>of<EAd>) →C</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↓</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行結果が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	↓																																													
I	UI	H	U	N	Z	V	C																																																								
—	—	—	—	—	—	—	↓																																																								
<p>●アセンブラフォーマット BXOR #xx:3, <EAd></p>																																																															
<p>●オペランドサイズ バイト</p>																																																															
<p>●説明</p> <p>デスティネーションオペランドの指定された1ビットとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。ビット番号は、3ビットのイミディエイトデータで指定されません。デスティネーションの内容は変化しません。</p> <div style="text-align: center;"> <p>ビット番号 7 6 5 4 3 2 1 0</p> <p>#xx:3で指定</p> <p><EAd> → [7-bit register]</p> <p>C [] ⊕ [] → [] C</p> </div>																																																															
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0L~R7L, R0H~R7H ERd : ER0~ER7</p>																																																															
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード*</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th colspan="2">第3バイト</th> <th colspan="2">第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>BXOR</td> <td>#xx:3,Rd</td> <td>7</td> <td>5</td> <td>0</td> <td>IMM</td> <td>rd</td> <td></td> <td></td> <td></td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ間接</td> <td>BXOR</td> <td>#xx:3,@ERd</td> <td>7</td> <td>C</td> <td>0</td> <td>erd</td> <td>0</td> <td>7</td> <td>5</td> <td>0</td> <td>IMM</td> <td>0</td> <td>6</td> </tr> <tr> <td>絶対アドレス</td> <td>BXOR</td> <td>#xx:3,@aa:8</td> <td>7</td> <td>E</td> <td></td> <td>abs</td> <td></td> <td>7</td> <td>5</td> <td>0</td> <td>IMM</td> <td>0</td> <td>6</td> </tr> </tbody> </table> <p>【注】* アドレッシングモードはデスティネーションオペランドの指定<EAd>です。</p>			アドレッシングモード*	ニーモニック	オペランド形式	インストラクションフォーマット								実行ステート数	第1バイト		第2バイト		第3バイト		第4バイト		レジスタ直接	BXOR	#xx:3,Rd	7	5	0	IMM	rd					2	レジスタ間接	BXOR	#xx:3,@ERd	7	C	0	erd	0	7	5	0	IMM	0	6	絶対アドレス	BXOR	#xx:3,@aa:8	7	E		abs		7	5	0	IMM	0	6
アドレッシングモード*	ニーモニック	オペランド形式				インストラクションフォーマット									実行ステート数																																																
			第1バイト		第2バイト		第3バイト		第4バイト																																																						
レジスタ直接	BXOR	#xx:3,Rd	7	5	0	IMM	rd					2																																																			
レジスタ間接	BXOR	#xx:3,@ERd	7	C	0	erd	0	7	5	0	IMM	0	6																																																		
絶対アドレス	BXOR	#xx:3,@aa:8	7	E		abs		7	5	0	IMM	0	6																																																		
<p>●注意事項</p> <p>@aa:8のアクセス範囲については、当該LSIのハードウェアマニュアルを参照してください。</p>																																																															

2.2.22(1) CMP(B)

CMP (CoMPare)		比較		
●オペレーション Rd-(EAs), CCRセット/クリア	●コンディションコード I UI H U N Z V C — — † — † † † † †			
●アセンブラフォーマット CMP.B <EAs>, Rd	H: ビット3にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: ビット7にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。			
●オペランドサイズ バイト				
●説明 8ビットレジスタRdの内容 (デスティネーションオペランド) からソースオペランドを減算し、その結果にしたがってコンディションコードをセットまたはクリアします。8ビットレジスタRdの内容は変化しません。				
●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H Rs : R0L~R7L, R0H~R7H				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット 第1バイト 第2バイト 第3バイト 第4バイト	実行 ステート 数
イミディエイト	CMP.B	#xx:8,Rd	A rd IMM	2
レジスタ直接	CMP.B	Rs,Rd	1 C rs rd	2
●注意事項				

2. 各命令の説明

2.2.22(2) CMP(W)

CMP (CoMPare)		比較																														
<p>●オペレーション Rd-(EAs), CCRセット/クリア</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>↑</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table> <p>H: ビット11にボローが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>C: ビット15にボローが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p>		I	UI	H	U	N	Z	V	C	—	—	↑	—	↑	↑	↑	↑														
I	UI	H	U	N	Z	V	C																									
—	—	↑	—	↑	↑	↑	↑																									
<p>●アセンブラフォーマット CMP.W <EAs>, Rd</p>																																
<p>●オペランドサイズ ワード</p>																																
<p>●説明</p> <p>16ビットレジスタRdの内容 (デスティネーションオペランド) からソースオペランドを減算し、その結果にしたがってコンディションコードをセットまたはクリアします。16ビットレジスタRdの内容は変化しません。</p>																																
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0~R7, E0~E7</p> <p>Rs : R0~R7, E0~E7</p>																																
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>CMP.W</td> <td>#xx:16,Rd</td> <td>7</td> <td>9</td> <td>2</td> <td>rd</td> <td>IMM</td> <td>4</td> </tr> <tr> <td>レジスタ直接</td> <td>CMP.W</td> <td>Rs,Rd</td> <td>1</td> <td>D</td> <td>rs</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>			アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	イミディエイト	CMP.W	#xx:16,Rd	7	9	2	rd	IMM	4	レジスタ直接	CMP.W	Rs,Rd	1	D	rs	rd		2
アドレッシングモード	ニーモニック	オペランド形式				インストラクションフォーマット					実行ステート数																					
			第1バイト	第2バイト	第3バイト	第4バイト																										
イミディエイト	CMP.W	#xx:16,Rd	7	9	2	rd	IMM	4																								
レジスタ直接	CMP.W	Rs,Rd	1	D	rs	rd		2																								
<p>●注意事項</p>																																

2.2.22(3) CMP(L)

CMP (CoMPare)		比較																	
●オペレーション ERd-(EAs), CCRセット/クリア	●コンディションコード																		
	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>↑</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table>			I	UI	H	U	N	Z	V	C	—	—	↑	—	↑	↑	↑	↑
I	UI	H	U	N	Z	V	C												
—	—	↑	—	↑	↑	↑	↑												
●アセンブラフォーマット CMPL <EAs>, ERd	<p>H: ビット27にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>C: ビット31にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p>																		
●オペランドサイズ ロングワード																			
●説明 32ビットレジスタERdの内容（デスティネーションオペランド）からソースオペランドを減算し、その結果にしたがってCCRの各ビットをセットまたはクリアします。32ビットレジスタERdの内容は変化しません。																			
●使用可能な汎用レジスタ ERd : ER0~ER7 ERs : ER0~ER7																			
●オペランド形式と実行ステート数																			
アドレッシング モード	ニーモ ニック	オペランド 形式	<table border="1" style="width: 100%; text-align: center;"> <tr> <th colspan="6">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> <th>第5バイト</th> <th>第6バイト</th> </tr> </table>	インストラクションフォーマット						実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト			
インストラクションフォーマット						実行 ステート 数													
第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト														
イミディエイト	CMPL	#xx:32,ERd	7 A 2 0 erd	IMM			6												
レジスタ直接	CMPL	ERs,ERd	1 F 1 ers 0 erd				2												
●注意事項																			

2. 各命令の説明

2.2.23 DAA

DAA (Decimal Adjust Add)		10進補正																																																													
●オペレーション Rd (10進補正) →Rd	●コンディションコード																																																														
	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>*</td><td>—</td><td>↑</td><td>↑</td><td>*</td><td>↑</td> </tr> </table>			I	UI	H	U	N	Z	V	C	—	—	*	—	↑	↑	*	↑																																												
I	UI	H	U	N	Z	V	C																																																								
—	—	*	—	↑	↑	*	↑																																																								
●アセンブラフォーマット DAA Rd	H: 値を保証しません。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 値を保証しません。 C: ビット7にキャリが発生したとき1にセットされ、それ以外の場合は実行前の値が保持されます。																																																														
●オペランドサイズ バイト																																																															
●説明																																																															
ADD.B、ADDX.B命令で、4ビットBCDデータを加算した結果が8ビットレジスタRdおよびキャリフラグおよびハーフキャリフラグにあるとき、下表にしたがって8ビットレジスタRdの内容 (デスティネーションオペランド) を補正 (00、06、60、66を加算) します。																																																															
<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>補正前の Cフラグ</th> <th>補正前の 上位4ビット</th> <th>補正前の Hフラグ</th> <th>補正前の 下位4ビット</th> <th>加算される数 (16進数)</th> <th>補正後の Cフラグ</th> </tr> </thead> <tbody> <tr><td>0</td><td>0~9</td><td>0</td><td>0~9</td><td>00</td><td>0</td></tr> <tr><td>0</td><td>0~8</td><td>0</td><td>A~F</td><td>06</td><td>0</td></tr> <tr><td>0</td><td>0~9</td><td>1</td><td>0~3</td><td>06</td><td>0</td></tr> <tr><td>0</td><td>A~F</td><td>0</td><td>0~9</td><td>60</td><td>1</td></tr> <tr><td>0</td><td>9~F</td><td>0</td><td>A~F</td><td>66</td><td>1</td></tr> <tr><td>0</td><td>A~F</td><td>1</td><td>0~3</td><td>66</td><td>1</td></tr> <tr><td>1</td><td>1~2</td><td>0</td><td>0~9</td><td>60</td><td>1</td></tr> <tr><td>1</td><td>1~2</td><td>0</td><td>A~F</td><td>66</td><td>1</td></tr> <tr><td>1</td><td>1~3</td><td>1</td><td>0~3</td><td>66</td><td>1</td></tr> </tbody> </table>				補正前の Cフラグ	補正前の 上位4ビット	補正前の Hフラグ	補正前の 下位4ビット	加算される数 (16進数)	補正後の Cフラグ	0	0~9	0	0~9	00	0	0	0~8	0	A~F	06	0	0	0~9	1	0~3	06	0	0	A~F	0	0~9	60	1	0	9~F	0	A~F	66	1	0	A~F	1	0~3	66	1	1	1~2	0	0~9	60	1	1	1~2	0	A~F	66	1	1	1~3	1	0~3	66	1
補正前の Cフラグ	補正前の 上位4ビット	補正前の Hフラグ	補正前の 下位4ビット	加算される数 (16進数)	補正後の Cフラグ																																																										
0	0~9	0	0~9	00	0																																																										
0	0~8	0	A~F	06	0																																																										
0	0~9	1	0~3	06	0																																																										
0	A~F	0	0~9	60	1																																																										
0	9~F	0	A~F	66	1																																																										
0	A~F	1	0~3	66	1																																																										
1	1~2	0	0~9	60	1																																																										
1	1~2	0	A~F	66	1																																																										
1	1~3	1	0~3	66	1																																																										
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H、R7H																																																															
●オペランド形式と実行ステート数																																																															
<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>DAA</td> <td>Rd</td> <td>0</td><td>F</td> <td>0</td><td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	DAA	Rd	0	F	0	rd		2																																							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																																																								
			第1バイト	第2バイト	第3バイト	第4バイト																																																									
レジスタ直接	DAA	Rd	0	F	0	rd		2																																																							
●注意事項																																																															
上記以外の場合について本命令を実行したときの結果 (8ビットレジスタRdの内容、およびC、V、Z、N、Hの各フラグ) は保証しません。																																																															

2.2.24 DAS

DAS (Decimal Adjust Subtract)		10進補正																															
●オペレーション Rd (10進補正) →Rd	●コンディションコード I UI H U N Z V C — — * — † † * —																																
●アセンブラフォーマット DAS Rd	H: 値を保証しません。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 値を保証しません。 C: 実行前の値が保持されます。																																
●オペランドサイズ バイト																																	
●説明 SUB.B、SUBX.BおよびNEG.B命令で、4ビットBCDデータを減算した結果が8ビットレジスタRd、キャリフラグおよびハーフキャリフラグにあるとき、下表にしたがって8ビットレジスタRd (デスティネーションオペランド) の内容を補正 (00、FA、A0、9Aを加算) します。																																	
<table border="1"> <thead> <tr> <th>補正前のCフラグ</th> <th>補正前の上位4ビット</th> <th>補正前のHフラグ</th> <th>補正前の下位4ビット</th> <th>加算される数 (16進数)</th> <th>補正後のCフラグ</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0~9</td> <td>0</td> <td>0~9</td> <td>00</td> <td>0</td> </tr> <tr> <td>0</td> <td>0~8</td> <td>1</td> <td>6~F</td> <td>FA</td> <td>0</td> </tr> <tr> <td>1</td> <td>7~F</td> <td>0</td> <td>0~9</td> <td>A0</td> <td>1</td> </tr> <tr> <td>1</td> <td>6~F</td> <td>1</td> <td>6~F</td> <td>9A</td> <td>1</td> </tr> </tbody> </table>				補正前のCフラグ	補正前の上位4ビット	補正前のHフラグ	補正前の下位4ビット	加算される数 (16進数)	補正後のCフラグ	0	0~9	0	0~9	00	0	0	0~8	1	6~F	FA	0	1	7~F	0	0~9	A0	1	1	6~F	1	6~F	9A	1
補正前のCフラグ	補正前の上位4ビット	補正前のHフラグ	補正前の下位4ビット	加算される数 (16進数)	補正後のCフラグ																												
0	0~9	0	0~9	00	0																												
0	0~8	1	6~F	FA	0																												
1	7~F	0	0~9	A0	1																												
1	6~F	1	6~F	9A	1																												
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H																																	
●オペランド形式と実行ステート数																																	
アドレッシングモード	ニーモニック	オペランド形式	<table border="1"> <thead> <tr> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>DAS</td> <td>Rd</td> <td>1 F 0 rd</td> <td>2</td> </tr> </tbody> </table>	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	DAS	Rd	1 F 0 rd	2																
インストラクションフォーマット				実行ステート数																													
第1バイト	第2バイト	第3バイト	第4バイト																														
レジスタ直接	DAS	Rd	1 F 0 rd	2																													
●注意事項 上記以外の場合について本命令を実行したときの結果 (8ビットレジスタRdの内容、およびC、V、Z、N、Hの各フラグ) は保証しません。																																	

2. 各命令の説明

2.2.25(1) DEC(B)

DEC (DECrement)		デクリメント															
●オペレーション Rd-1→Rd	●コンディションコード																
●アセンブラフォーマット DEC.B Rd	I UI H U N Z V C ─ ─ ─ ─ † † † ─																
●オペランドサイズ バイト	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: 実行前の値が保持されます。																
●説明 8ビットレジスタRdの内容（デスティネーションオペランド）から1を減算し、結果を8ビットレジスタRdに格納します。																	
●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H																	
●オペランド形式と実行ステート数																	
アドレッシング モード	ニーモ ニック	オペランド 形式	<table border="1"> <thead> <tr> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>DEC.B</td> <td>Rd</td> <td>1 : A 0 : rd</td> <td>2</td> </tr> </tbody> </table>	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	DEC.B	Rd	1 : A 0 : rd	2
インストラクションフォーマット				実行 ステート 数													
第1バイト	第2バイト	第3バイト	第4バイト														
レジスタ直接	DEC.B	Rd	1 : A 0 : rd	2													
●注意事項 オーバフローは、H'80-1→H'7Fのとき発生します。																	

2.2.25(2) DEC(W)

DEC (DECrement)		デクリメント																															
<p>●オペレーション</p> <p>Rd-1→Rd Rd-2→Rd</p>	<p>●コンディションコード</p> <p>I U I H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: 実行前の値が保持されます。</p>			—	—	—	—	↑	↑	↑	—																						
—	—	—	—	↑	↑	↑	—																										
<p>●アセンブラフォーマット</p> <p>DEC.W #1, Rd DEC.W #2, Rd</p>																																	
<p>●オペランドサイズ</p> <p>ワード</p>																																	
<p>●説明</p> <p>16ビットレジスタRdの内容（デスティネーションオペランド）から1または2を減算し、結果を16ビットレジスタRdに格納します。</p>																																	
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0~R7, E0~E7</p>																																	
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>DEC.W</td> <td>#1,Rd</td> <td>1</td> <td>B</td> <td>5</td> <td>rd</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>DEC.W</td> <td>#2,Rd</td> <td>1</td> <td>B</td> <td>D</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	DEC.W	#1,Rd	1	B	5	rd		2	レジスタ直接	DEC.W	#2,Rd	1	B	D	rd		2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																										
			第1バイト	第2バイト	第3バイト	第4バイト																											
レジスタ直接	DEC.W	#1,Rd	1	B	5	rd		2																									
レジスタ直接	DEC.W	#2,Rd	1	B	D	rd		2																									
<p>●注意事項</p> <p>オーバフローは、H'8000-1→H'7FFF,H'8000-2→H'7FFE,H'8001-2→H'7FFFのとき発生します。</p>																																	

2. 各命令の説明

2.2.25(3) DEC(L)

DEC (DECrement)		デクリメント																													
<p>●オペレーション ERd-1→ERd ERd-2→ERd</p>	<p>●コンディションコード</p> <p>I U I H U N Z V C</p> <table border="1"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: 実行前の値が保持されます。</p>			—	—	—	—	↑	↑	↑	—																				
—	—	—	—	↑	↑	↑	—																								
<p>●アセンブラフォーマット DEC.L #1, ERd DEC.L #2, ERd</p>																															
<p>●オペランドサイズ ロングワード</p>																															
<p>●説明</p> <p>32ビットレジスタERdの内容（デスティネーションオペランド）から1または2を減算し、結果を32ビットレジスタERdに格納します。</p>																															
<p>●使用可能な汎用レジスタ ERd : ER0~ER7</p>																															
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>DEC.L</td> <td>#1,ERd</td> <td>1</td> <td>B</td> <td>7 0</td> <td>erd</td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>DEC.L</td> <td>#2,ERd</td> <td>1</td> <td>B</td> <td>F 0</td> <td>erd</td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	DEC.L	#1,ERd	1	B	7 0	erd	2	レジスタ直接	DEC.L	#2,ERd	1	B	F 0	erd	2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																								
			第1バイト	第2バイト	第3バイト	第4バイト																									
レジスタ直接	DEC.L	#1,ERd	1	B	7 0	erd	2																								
レジスタ直接	DEC.L	#2,ERd	1	B	F 0	erd	2																								
<p>●注意事項</p> <p>オーバフローは、H'80000000-1→H'7FFFFFFF,H'80000000-2→H'7FFFFFFE,H'80000001-2→H'7FFFFFFFのとき発生します。</p>																															

2.2.26(1) DIVXS(B)

DIVXS (DIVide eXtend as Signed)		符号付き除算																			
●オペレーション Rd ÷ Rs → Rd	●コンディションコード I UI H U N Z V C — — — — † † — —																				
●アセンブラフォーマット DIVXS.B Rs, Rd	H: 実行前の値が保持されます。 N: 商が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 除数が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。																				
●オペランドサイズ バイト																					
●説明 16ビットレジスタRdの内容（デスティネーションオペランド）を8ビットレジスタRsの内容（ソースオペランド）で符号付き除算し、結果を16ビットレジスタRdに格納します。演算は、16ビット ÷ 8ビット → 商8ビット、余り8ビットとして行われます。商はRdの下位8ビットに、余りは上位8ビットに格納されます。余りの符号は、被除数の符号に一致しています。																					
<table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">Rd</td> <td style="padding: 0 10px;">÷</td> <td style="border: 1px solid black; padding: 2px;">Rs</td> <td style="padding: 0 10px;">→</td> <td colspan="2" style="border: 1px solid black; padding: 2px;">Rd</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">被除数</td> <td></td> <td style="border: 1px solid black; padding: 2px;">除数</td> <td></td> <td style="border: 1px solid black; padding: 2px;">余り</td> <td style="border: 1px solid black; padding: 2px;">商</td> </tr> <tr> <td style="text-align: center;">16ビット</td> <td></td> <td style="text-align: center;">8ビット</td> <td></td> <td style="text-align: center;">8ビット</td> <td style="text-align: center;">8ビット</td> </tr> </table> <p>なお、ゼロ除算またはオーバフローが発生した場合の結果は保証されません。 ●DIVXS命令とゼロ除算およびオーバフローを参照してください。</p>				Rd	÷	Rs	→	Rd		被除数		除数		余り	商	16ビット		8ビット		8ビット	8ビット
Rd	÷	Rs	→	Rd																	
被除数		除数		余り	商																
16ビット		8ビット		8ビット	8ビット																
●使用可能な汎用レジスタ Rd : R0~R7、E0~E7 Rd : R0L~R7L、R0H~R7H																					
●オペランド形式と実行ステート数																					
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数														
レジスタ直接	DIVXS.B	Rs,Rd	第1バイト	第2バイト	第3バイト	第4バイト	16														
			0 1	D 0	5 1	rs rd															
●注意事項 Nフラグは、被除数と除数の符号が異なるとき1にセットされ、符号が同じとき0にクリアされます。したがって、商が0（ゼロ）でNフラグが1にセットされる場合があります。																					

2. 各命令の説明

2.2.26(2) DIVXS(W)

DIVXS (DIVide eXtend as Signed)		符号付き除算																									
<p>●オペレーション ERd ÷ Rs → ERd</p>	<p>●コンディションコード</p> <p style="text-align: center;">I UI H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↓</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 商が負のとき1にセットされ、それ以外 のときは0にクリアされます。 Z: 除数が0（ゼロ）のとき1にセットされ、 それ以外の場合は0にクリアされます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>		—	—	—	—	↑	↓	—	—																	
—	—	—	—	↑	↓	—	—																				
<p>●アセンブラフォーマット DIVXS.W Rs, ERd</p>																											
<p>●オペランドサイズ ワード</p>																											
<p>●説明</p> <p>32ビットレジスタERdの内容（デスティネーションオペランド）を16ビットレジスタRsの内容（ソースオペランド）で符号付き除算し、結果を32ビットレジスタERdに格納します。演算は、32ビット÷16ビット→商16ビット、余り16ビットとして行われます。商は32ビットレジスタERdの下位16ビット（Rd）に、余りは上位16ビット（Ed）に格納します。余りの符号は、被除数の符号に一致しています。</p> <div style="text-align: center; margin: 10px 0;"> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="text-align: center;">ERd</td></tr> <tr><td style="text-align: center;">被除数</td></tr> <tr><td style="text-align: center;">32ビット</td></tr> </table> <div style="display: inline-block; vertical-align: middle; text-align: center;">÷</div> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="text-align: center;">Rs</td></tr> <tr><td style="text-align: center;">除数</td></tr> <tr><td style="text-align: center;">16ビット</td></tr> </table> <div style="display: inline-block; vertical-align: middle; text-align: center;">→</div> <table border="1" style="display: inline-table; margin-left: 10px;"> <tr><td colspan="2" style="text-align: center;">ERd</td></tr> <tr><td style="text-align: center;">余り</td><td style="text-align: center;">商</td></tr> <tr><td style="text-align: center;">16ビット</td><td style="text-align: center;">16ビット</td></tr> </table> </div> <p>なお、ゼロ除算またはオーバーフローが発生した場合の結果は保証されません。 ●DIVXS命令とゼロ除算およびオーバーフローを参照してください。</p>			ERd	被除数	32ビット	Rs	除数	16ビット	ERd		余り	商	16ビット	16ビット													
ERd																											
被除数																											
32ビット																											
Rs																											
除数																											
16ビット																											
ERd																											
余り	商																										
16ビット	16ビット																										
<p>●使用可能な汎用レジスタ</p> <p>ERd : ER0~ER7 Rs : R0~R7, E0~E7</p>																											
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>DIVXS.W</td> <td>Rs, ERd</td> <td>0</td> <td>1</td> <td>D</td> <td>0</td> <td>5</td> <td>3</td> <td>rs</td> <td>0</td> <td>erd</td> <td>24</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	DIVXS.W	Rs, ERd	0	1	D	0	5	3	rs	0	erd	24
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数																
			第1バイト	第2バイト	第3バイト	第4バイト																					
レジスタ直接	DIVXS.W	Rs, ERd	0	1	D	0	5	3	rs	0	erd	24															
<p>●注意事項</p> <p>Nフラグは、被除数と除数の符号が異なるとき1にセットされ、符号が同じとき0にクリアされます。したがって、商が0（ゼロ）でNフラグが1にセットされる場合があります。</p>																											

2.2.26(3) DIVXS

DIVXS (DIVide eXtend as Signed)	符号付き除算										
<p>●DIVXS命令とゼロ除算およびオーバフロー</p>											
<p>DIVXS命令は、ゼロ除算およびオーバフローの検出を行っていません。したがって、以下に示すようなプログラムを参考にして、ゼロ除算の検出とオーバフローの対策を行ってください。</p>											
<p>1. DIVXS.B R0L,R1を行う場合の対策</p>											
<p>(例1) 除数、被除数を正数にして演算を行い、DIVXU命令のゼロ除算およびオーバフロー対策に帰着させる対策</p>											
<pre>MOV.B R0L, R0L BEQ ZERODIV ANDC #AF, CCR BPL L1 NEG.B R0L ORC #10, CCR</pre>	<pre>; 除数の判定 ; ゼロ除算ならば、ZERODIVに分岐 ; CCRのUI、Uビットを0にクリア ; 除数が正数ならばL1に分岐 ; 除数の符号を反転する ; CCRのUビットを1にセット</pre>										
<pre>L1: MOV.W R1, R1 BPL L2 NEG.W R1 XORC #50, CCR</pre>	<pre>; 被除数の判定 ; 被除数が正数ならばL2に分岐 ; 被除数の符号を反転する ; CCRのUI、Uビットを反転</pre>										
<pre>L2: MOV.B R1H, R2L EXTU.W R2 DIVXU.B R0L, R2 MOV.B R2H, R1H DIVXU.B R0L, R1 MOV.B R2L, R2H MOV.B R1L, R2L</pre>	<pre>; ; ; 正数に変換した除数と被除数を用いDIVXU.B命令で ; 16ビット÷8ビット→商 (16ビット)、余り (8ビット) ; の演算を行います。 ; (●DIVXU命令とゼロ除算およびオーバフローを参照し ; てください)</pre>										
<pre>STC CCR, R1L BTST #6, R1L BEQ L3 NEG.B R1H</pre>	<pre>; CCRの内容をR1Lに転送 ; CCRのUIビットの判定 ; UI=1ならばL3に分岐 ; 余りの符号を反転する</pre>										
<pre>L3: BTST #4, R1L BEQ L4 NEG.W R2</pre>	<pre>; CCRのUビットの判定 ; U=1ならばL4に分岐 ; 商の符号を反転する</pre>										
<pre>L4: RTS</pre>											
<pre>ZERODIV:</pre>	<pre>; ゼロ除算処理ルーチン</pre>										
<p>この結果、商 (16ビット) はR2に、余り (8ビット) はR1Hに格納されています。</p>											
<div style="text-align: center;"> <table style="margin: auto;"> <tr> <td style="padding-right: 10px;">R0L</td> <td style="border: 1px solid black; padding: 2px 10px;">除数</td> </tr> <tr> <td style="padding-right: 10px;">R1</td> <td style="border: 1px solid black; padding: 2px 10px;">被除数</td> </tr> <tr> <td></td> <td style="text-align: center;">↓</td> </tr> <tr> <td style="padding-right: 10px;">R1H</td> <td style="border: 1px solid black; padding: 2px 10px;">余り</td> </tr> <tr> <td style="padding-right: 10px;">R2</td> <td style="border: 1px solid black; padding: 2px 10px;">商</td> </tr> </table> </div>		R0L	除数	R1	被除数		↓	R1H	余り	R2	商
R0L	除数										
R1	被除数										
	↓										
R1H	余り										
R2	商										

2.2.26(3) DIVXS

DIVXS (DIVide eXtend as Signed)	符号付き除算
<p>(例2) 除数 (8ビット) を16ビットに、被除数 (16ビット) を32ビットに符号拡張して除算する対策</p>	
<pre> EXTS.W R0 BEQ ZERODIV EXTS.L ER1 DIVXS.L R0,ER1 RTS ZERODIV: </pre>	
<p>この結果、商 (16ビット) はR1に、余り (8ビット) はE1 (16ビットに符号拡張) に格納されます。</p>	<pre> ER1 符号拡張 被除数 ER1 余り 商 </pre>
<p>2. DIVXS.W R0,ER1 を行う場合の対策</p>	
<p>(例) 除数、被除数を正数にして演算を行い、DIVXU命令のゼロ除算およびオーバフロー対策に帰着させる対策</p>	
<pre> MOV.W R0, R0 ; ゼロ除算か BEQ ZERODIV ; ゼロ除算ならば、ZERODIVに分岐 ANDC #AF, CCR ; CCRのUI、Uビットを0にクリア BPL L1 ; 除数が正数ならばL1に分岐 NEG.W R0 ; 除数の符号を反転する ORC #10, CCR ; CCRのUビットを1にセット L1: MOV.L ER1, ER1 ; 被除数の判定 BPL L2 ; 被除数が正数ならばL2に分岐 NEG.L ER1 ; 被除数の符号を反転する XORC #50, CCR ; CCRのUI、Uビットを反転 L2: MOV.W E1, E2 ; EXTU.L ER2 ; DIVXU.W R0, ER2 ; MOV.W E2, R1 ; DIVXU.B R0, ER1 ; MOV.W R2, E2 ; MOV.W R1, R2 ; STC CCR, R1L ; CCRの内容をR1Lに転送 BTST #6, R1L ; CCRのUIビットの判定 BEQ L3 ; UI=1ならばL3に分岐 NEG.W E1 ; 余りの符号を反転する L3: BTST #4, R1L ; CCRのUビットの判定 BEQ L4 ; U=1ならばL4に分岐 NEG.L ER2 ; 商の符号を反転する L4: RTS ZERODIV: ; ゼロ除算処理ルーチン </pre>	
<p>この結果、商 (32ビット) はER2に、余り (16ビット) はE1に格納されています。</p>	<pre> R0 偶数 ER1 被除数 E1 余り ER2 商 </pre>

2.2.26(3) DIVXS

DIVXS (DIVide eXtend as Signed)						符号付き除算
1. (例1)および2.では、CCRのUI、Uビットに除数、被除数の符号を反映しています。これを用いてDIVXU命令による符号なし除算の結果の商、余りの符号を、以下のように修正しています。						
UI	U	除数	被除数	余り	商	符号修正
0	0	正	正	正	正	符号の修正はありません。
0	1	負	正	正	負	商の符号を反転します。
1	0	負	負	負	正	余りの符号を反転します。
1	1	正	負	負	負	商、余りのいずれも符号を反転します。

2. 各命令の説明

2.2.27(1) DIVXU(B)

DIVXU (DIVide eXtend as Unsigned)		除算																				
<p>●オペレーション Rd ÷ Rs → Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↑</td> <td>↑</td> <td>—</td> <td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 除数が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 除数が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	—	—				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	—	—															
<p>●アセンブラフォーマット DIVXU.B Rs, Rd</p>																						
<p>●オペランドサイズ バイト</p>																						
<p>●説明</p> <p>16ビットレジスタRdの内容（デスティネーションオペランド）を8ビットレジスタRsの内容（ソースオペランド）で符号なし除算し、結果を16ビットレジスタRdに格納します。演算は、16ビット ÷ 8ビット → 商8ビット、余り8ビットとして行われます。商はRdの下位8ビットに、余りは上位8ビットに格納されます。</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">Rd</td> <td></td> <td style="text-align: center;">Rs</td> <td></td> <td style="text-align: center;">Rd</td> </tr> <tr> <td style="text-align: center;">被除数</td> <td style="text-align: center;">÷</td> <td style="text-align: center;">除数</td> <td style="text-align: center;">→</td> <td style="text-align: center;">余り 商</td> </tr> <tr> <td style="text-align: center;">16ビット</td> <td></td> <td style="text-align: center;">8ビット</td> <td></td> <td style="text-align: center;">8ビット 8ビット</td> </tr> </table> <p>なお、ゼロ除算またはオーバフローが発生した場合の結果は保証されません。オーバフローについては、●DIVXU命令とゼロ除算およびオーバフローを参照してください。</p>			Rd		Rs		Rd	被除数	÷	除数	→	余り 商	16ビット		8ビット		8ビット 8ビット					
Rd		Rs		Rd																		
被除数	÷	除数	→	余り 商																		
16ビット		8ビット		8ビット 8ビット																		
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0~R7, E0~E7 Rs : R0L~R7L, R0H~R7H</p>																						
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>DIVXU.B</td> <td>Rs,Rd</td> <td style="text-align: center;">5</td> <td style="text-align: center;">1</td> <td style="text-align: center;">rs</td> <td style="text-align: center;">rd</td> <td style="text-align: center;">14</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	DIVXU.B	Rs,Rd	5	1	rs	rd	14
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	DIVXU.B	Rs,Rd	5	1	rs	rd	14															
<p>●注意事項</p>																						

2.2.27(2) DIVXU(W)

DIVXU (DIVide eXtend as Unsigned)		除算																			
●オペレーション ERd ÷ Rs → ERd	●コンディションコード																				
●アセンブラフォーマット DIVXU.W Rs, ERd	I UI H U N Z V C — — — — † † — —																				
●オペランドサイズ ワード	H: 実行前の値が保持されます。 N: 除数が負のとき1にセットされ、それ以外 のときは0にクリアされます。 Z: 除数が0（ゼロ）のとき1にセットされ、 それ以外の場合は0にクリアされます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。																				
●説明																					
32ビットレジスタERdの内容（デスティネーションオペランド）を16ビットレジスタRsの内容（ソースオペランド）で符号なし除算し、結果を32ビットレジスタERdに格納します。演算は、32ビット ÷ 16ビット → 商16ビット、余り16ビットとして行われます。商は32ビットレジスタERdの下位16ビットに、余りは上位16ビットに格納します。																					
<table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="text-align: center; padding: 5px;">ERd</td> <td style="padding: 0 10px;"></td> <td style="text-align: center; padding: 5px;">Rs</td> <td style="padding: 0 10px;">→</td> <td colspan="2" style="text-align: center; padding: 5px;">ERd</td> </tr> <tr> <td style="text-align: center; border: 1px solid black; padding: 2px;">被除数</td> <td style="text-align: center; padding: 0 5px;">÷</td> <td style="text-align: center; border: 1px solid black; padding: 2px;">除数</td> <td></td> <td style="text-align: center; border: 1px solid black; padding: 2px;">余り</td> <td style="text-align: center; border: 1px solid black; padding: 2px;">商</td> </tr> <tr> <td style="text-align: center; font-size: small;">32ビット</td> <td></td> <td style="text-align: center; font-size: small;">16ビット</td> <td></td> <td style="text-align: center; font-size: small;">16ビット</td> <td style="text-align: center; font-size: small;">16ビット</td> </tr> </table>				ERd		Rs	→	ERd		被除数	÷	除数		余り	商	32ビット		16ビット		16ビット	16ビット
ERd		Rs	→	ERd																	
被除数	÷	除数		余り	商																
32ビット		16ビット		16ビット	16ビット																
なお、ゼロ除算またはオーバフローが発生した場合の結果は保証されません。オーバフローについては、●DIVXU命令とゼロ除算およびオーバフローを参照してください。																					
●使用可能な汎用レジスタ																					
ERd : ER0~ER7 Rs : R0~R7、E0~E7																					
●オペランド形式と実行ステート数																					
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数														
			第1バイト	第2バイト	第3バイト	第4バイト															
レジスタ直接	DIVXU.W	Rs, ERd	5	3	rs 0 erd		22														
●注意事項																					

2.2.27(3) DIVXU

DIVXU (DIVide eXtend as Unsigned)	除算
<p>●DIVXU命令とゼロ除算およびオーバーフロー</p> <p>DIVXU命令は、ゼロ除算およびオーバーフローの検出を行っていません。したがって、以下に示すようなプログラムを参考にして、ゼロ除算の検出とオーバーフローの対策を行ってください。</p> <p>1. DIVXU.B R0L,R1を行う場合の対策</p> <p>(例1) 除算を2回行い商を16ビットにする対策</p> <pre> CMP.B #0,R0L ; ゼロ除算か BEQ ZERODIV ; ゼロ除算ならば、ZERODIVに分岐 MOV.B R1H,R2L ; 被除数が上位8ビットをR2Lに転送し EXTU.W R2 ; 16ビットにゼロ拡張 DIVXU.B R0L,R2 (*2) ; 被除数上位8ビットを除算 MOV.B R2H,R1H (*3) ; 余り(部分)をR1Hに転送 DIVXU.B R0L,R1 (*4) ; 余り(部分)と被除数下位8ビットを除算 MOV.B R2L,R2H ; R2Hに商上位を格納 MOV.B R1L,R2L (*5) ; R2Lに商下位を格納 RTS ZERODIV: ; ゼロ除算処理ルーチン </pre> <p>この結果、16ビット÷8ビット→商(16ビット)、余り(8ビット)の演算を行ったことになりオーバーフローは起こりません。演算結果の商(16ビット)はR2に、余り(8ビット)はR1Hに格納されます。</p> <div style="text-align: center;"> </div>	

2.2.27(3) DIVXU

DIVXU (DIVide eXtend as Unsigned)	除算														
(例2) ワードサイズの除算を行う対策															
<pre> EXTU.W R0 ; 除数 (8ビット) を16ビットにゼロ拡張 BEQ ZERODIV ; ゼロ除算ならば、ZERODIVに分岐 EXTU.L ER1 ; 被除数 (16ビット) を32ビットにゼロ拡張 EXTU.W R0,ER1 ; DIVXU.Wにより演算 RTS ZERODIV: ; ゼロ除算処理ルーチン </pre>															
<p>この結果、16ビット÷8ビットの演算を32ビット÷16ビット→商 (16ビット)、余り (16ビット) で行ったことになり、オーバーフローは起こりません。演算結果の商 (16ビット) はR1に、余り (8ビット) はE1の下位8ビットに格納されます (E1の上位8ビットは、すべて0となります)。</p>															
<table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="padding: 5px;">R0L</td> <td style="border: 1px solid black; padding: 2px 10px;">除数</td> </tr> <tr> <td style="padding: 5px;">R1</td> <td style="border: 1px solid black; padding: 2px 10px;">被除数</td> </tr> <tr> <td colspan="2" style="text-align: center; padding: 5px;">↓</td> </tr> <tr> <td style="padding: 5px;">R0L</td> <td style="border: 1px solid black; padding: 2px 10px;">ゼロ拡張 除数</td> </tr> <tr> <td style="padding: 5px;">ER1</td> <td style="border: 1px solid black; padding: 2px 10px;">ゼロ拡張 被除数</td> </tr> <tr> <td colspan="2" style="text-align: center; padding: 5px;">↓</td> </tr> <tr> <td style="padding: 5px;">ER1</td> <td style="border: 1px solid black; padding: 2px 10px;">余り 商</td> </tr> </table>		R0L	除数	R1	被除数	↓		R0L	ゼロ拡張 除数	ER1	ゼロ拡張 被除数	↓		ER1	余り 商
R0L	除数														
R1	被除数														
↓															
R0L	ゼロ拡張 除数														
ER1	ゼロ拡張 被除数														
↓															
ER1	余り 商														

2.2.27(3) DIVXU

DIVXU (DIVide eXtend as Unsigned)	除算
<p>2. DIVXU.W R0,ER1を行う場合の対策</p> <p>(例) 除算を2回行い商を32ビットにする対策</p> <pre style="font-family: monospace; padding-left: 20px;"> MOV.W R0,R0 ;ゼロ除算か BEQ ZERODIV ;ゼロ除算ならば、ZERODIVに分岐 MOV.W E1,E2 ;被除数が上位16ビットをR2に転送し、 EXTU.L ER2 (*1) ;32ビットにゼロ拡張 DIVXU.W R0,ER2 (*2) ;被除数上位16ビットを除算 MOV.W E2,E1 (*3) ;余り(部分)をE1に転送 DIVXU.W R0,ER1 (*4) ;余り(部分)と被除数下位16ビットを除算 MOV.W R2,E2 ;E2に商上位を格納 MOV.W R1,R2 (*5) ;R2に商下位を格納 RTS ZERODIV: ;ゼロ除算処理ルーチン </pre> <p>この結果、32ビット÷16ビット→商(32ビット)、余り(16ビット)の演算を行ったことになりオーバーフローは起こりません。演算結果の商(32ビット)はER2に、余り(16ビット)はE1に格納されます。</p> <div style="text-align: center; margin-top: 20px;"> <pre style="font-family: monospace; margin: 0 auto;"> R0 [除数] ER1 [被除数] ↓ ER2 [ゼロ拡張 被除数上位] (*1) ↓ ER2 [余り(部分) 商上位] (*2) ↓ ER1 [余り(部分) 被除数下位] (*3) ↓ ER1 [余り 商下位] (*4) ↓ ER1 [余り 商下位] (*5) ER2 [商] </pre> </div>	

2.2.28(1) EEPMOV(B)

EEPMOV (MOVE data to EEPROM)		ブロック転送																									
<p>●オペレーション</p> <pre>if R4L≠0 then Repeat @ER5+→@ER6+ R4L-1→R4L Until R4L=0 else next;</pre>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>	I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	—	<p>●アセンブラフォーマット</p> <p>EEPMOV.B</p>									
I	UI	H	U	N	Z	V	C																				
—	—	—	—	—	—	—	—																				
<p>●オペランドサイズ</p> <p>—</p>																											
<p>●説明</p> <p>ブロック転送命令です。ER5で示されるメモリ上のデータをER6で示されるメモリへ転送し、ER5、ER6の値をインクリメント、R4Lの値をデクリメントします。R4Lの内容が0（ゼロ）となるまで上記動作を繰り返します。その後、次の命令を実行します。本命令でのデータ転送は、バイトサイズデータの連続転送となります。転送バイト数はR4Lで示されます。アセンブラフォーマットのバイト表示は、8ビットレジスタR4Lで示します（最大転送バイト数は255バイトとなります）。データ転送中は割り込みの検出を行いません。</p> <p>本命令の実行終了時には、R4Lは0（ゼロ）を、またER5、ER6はそれぞれ（最終アドレス+1）の内容を保持しています。</p>																											
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>EEPMOV.B</td> <td></td> <td>7</td><td>B</td> <td>5</td><td>C</td> <td>5</td><td>9</td> <td>8</td><td>F</td> <td>8+4n*</td> </tr> </tbody> </table> <p>【注】* R4Lの初期設定値がnの場合です。このとき転送データはnバイトですが、データアクセスは2(n+1)回行われ、このデータアクセスに必要なステート数は4(n+1)です。(n=0、1、2・・・255)</p>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	—	EEPMOV.B		7	B	5	C	5	9	8	F	8+4n*
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																				
			第1バイト	第2バイト	第3バイト	第4バイト																					
—	EEPMOV.B		7	B	5	C	5	9	8	F	8+4n*																
<p>●注意事項</p> <p>本命令ではまず、ER5、ER6で示されるメモリのリードを行い、その後、データのブロック転送を行います。</p> <p>本命令の実行ステート数はH8/300CPUと異なります。</p>																											

2. 各命令の説明

2.2.28(2) EEPMOV(W)

EEPMOV (MOVE data to EEPROM)		ブロック転送																																	
<p>●オペレーション</p> <pre>if R4≠0 then Repeat @ER5+→@ER6+ R4-1→R4 Until R4=0 else next;</pre>	<p>●コンディションコード</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>	I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	—																		
I	UI	H	U	N	Z	V	C																												
—	—	—	—	—	—	—	—																												
<p>●アセンブラフォーマット</p> <p>EEPMOV.W</p>																																			
<p>●オペランドサイズ</p> <p>—</p>																																			
<p>●説明</p> <p>ブロック転送命令です。ER5で示されるメモリ上のデータをER6で示されるメモリへ転送し、ER5、ER6の値をインクリメント、R4の値をデクリメントします。R4の内容が0（ゼロ）となるまで上記動作を繰り返します。その後、次の命令を実行します。本命令でのデータ転送は、バイトサイズデータの連続転送となります。転送バイト数はR4で示されます。アセンブラフォーマットのワード表示は、16ビットレジスタR4で示します（最大転送バイト数は65535バイトとなります）。データ転送中はNMI以外の割り込みの検出を行いません。</p> <p>NMI割り込みが発生しない状態での本命令の実行終了時には、R4は0（ゼロ）を、またER5、ER6はそれぞれ（最終アドレス+1）の内容を保持しています。</p> <p>NMI割り込みが発生すると、転送中の1バイトの転送終了後NMI割り込み例外処理を行います。このときR4は残りの転送バイト数を、またER5、ER6はそれぞれ次の転送アドレスを示します。NMI割り込み例外処理で退避されるPCは直後の命令の先頭アドレスです。</p> <p>●EEPMOV.W命令とNMI割り込みを参照してください。</p>																																			
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="8">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> <th>第5バイト</th> <th>第6バイト</th> <th>第7バイト</th> <th>第8バイト</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>EEPMOV.W</td> <td></td> <td>7</td> <td>B</td> <td>D</td> <td>4</td> <td>5</td> <td>9</td> <td>8</td> <td>F</td> <td>8+4n*</td> </tr> </tbody> </table> <p>【注】* R4の初期設定値がnの場合です。このとき転送データはnバイトですが、データアクセスは2 (n+1) 回行われ、このデータアクセスに必要なステート数は4 (n+1) です。(n=0、1、2・・・65535)</p>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット								実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	—	EEPMOV.W		7	B	D	4	5	9	8	F	8+4n*
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット								実行 ステート 数																								
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト																									
—	EEPMOV.W		7	B	D	4	5	9	8	F	8+4n*																								
<p>●注意事項</p> <p>本命令ではまず、ER5、ER6で示されるメモリのリードを行い、その後データのブロック転送を行います。</p>																																			

2.2.28(2) EEPMOV(W)

EEPMOV (MOVE data to EEPROM)	ブロック転送
<p>●EEPMOV.W命令とNMI割り込み</p> <p>EEPMOV.W命令実行中にNMI割り込みが発生すると、転送中の1バイト転送終了後、NMI割り込み例外処理を実行します。このときのレジスタの内容は次のようになっています。</p> <p>ER5 : 残りの転送元アドレスの先頭 ER6 : 残りの転送先アドレスの先頭 R4 : 残りの転送バイト数</p> <p>また、このNMI割り込み例外処理時にスタックされるPCの値は本命令の直後の命令の先頭アドレスになっています。したがって、EEPMOV.W命令実行中にNMI割り込みが発生する場合には以下のようなプログラムで対策を行ってください。</p> <p>(例)</p> <pre>L1: EEPMOV.W MOV.W R4,R4 BNE L1</pre> <p>なお、EEPMOV.B命令ではNMI割り込みを含めてすべての割り込みを受け付けません。</p>	

2. 各命令の説明

2.2.29(1) EXTS(W)

EXST (EXTend as Signed)		符号拡張																					
<p>●オペレーション ($\langle \text{ビット7} \rangle$ of Rd) \rightarrow ($\langle \text{ビット15} \sim 8 \rangle$ of Rd)</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↓</td><td>↓</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外のときは0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↓	↓	0	—					
I	UI	H	U	N	Z	V	C																
—	—	—	—	↓	↓	0	—																
<p>●アセンブラフォーマット EXTS.W Rd</p>																							
<p>●オペランドサイズ ワード</p>																							
<p>●説明 16ビットレジスタRdの低位8ビットの符号を上位方向にコピーし、ワードサイズに符号拡張します (Rdのビット7をビット15~8にコピーします)。</p> <div style="text-align: center;"> </div>																							
<p>●使用可能な汎用レジスタ Rd : R0~R7, E0~E7</p>																							
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>EXTS.W</td> <td>Rd</td> <td>1</td> <td>7</td> <td>D</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	EXTS.W	Rd	1	7	D	rd		2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数												
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	EXTS.W	Rd	1	7	D	rd		2															
<p>●注意事項</p>																							

2.2.29(2) EXTS(L)

EXTS (EXTend as Signed)		符号拡張					
●オペレーション (\langle ビット15 \rangle of ERd) → (\langle ビット31~16 \rangle of ERd)	●コンディションコード						
●アセンブラフォーマット EXTS.L ERd	I UI H U N Z V C — — — — † † 0 —						
●オペランドサイズ ロングワード	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外のときは0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。						
●説明							
32ビットレジスタERdの下位16ビットの符号ビットを上位方向にコピーし、ロングワードサイズに符号拡張します (ERdのビット15をビット31~16にコピーします)。							
●使用可能な汎用レジスタ ERd : ER0~ER7							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
レジスタ直接	EXTS.L	ERd	1	7	F 0: erd		2
●注意事項							

2. 各命令の説明

2.2.30(1) EXTU(W)

EXTU (EXTend as Unsigned)		ゼロ拡張																					
<ul style="list-style-type: none"> ●オペレーション 0 → (<ビット15~8>of Rd) 	<ul style="list-style-type: none"> ●コンディションコード <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>0</td><td>↓</td><td>0</td><td>—</td> </tr> </table> H: 実行前の値が保持されます。 N: 常に0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。 		I	UI	H	U	N	Z	V	C	—	—	—	—	0	↓	0	—					
I	UI	H	U	N	Z	V	C																
—	—	—	—	0	↓	0	—																
<ul style="list-style-type: none"> ●アセンブラフォーマット EXTU.W Rd 																							
<ul style="list-style-type: none"> ●オペランドサイズ ワード 																							
<ul style="list-style-type: none"> ●説明 16ビットレジスタRdの下位8ビットワードサイズにゼロ拡張します。Rdの上位8ビット（ビット15~8）に0（ゼロ）が入ります。 <div style="text-align: center;"> <table border="1" style="margin: auto;"> <tr> <td colspan="2" style="text-align: center;">Rd</td> <td colspan="2" style="text-align: center;">Rd</td> </tr> <tr> <td style="text-align: center;">ビット15</td> <td style="text-align: center;">7</td> <td style="text-align: center;">ビット15</td> <td style="text-align: center;">7</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">Don't care</td> <td style="text-align: center;">8ビット</td> <td style="text-align: center;">ゼロ拡張</td> <td style="text-align: center;">8ビット</td> </tr> <tr> <td style="text-align: center;">8ビット</td> <td style="text-align: center;">8ビット</td> <td style="text-align: center;">8ビット</td> <td style="text-align: center;">8ビット</td> </tr> </table> </div>			Rd		Rd		ビット15	7	ビット15	7	0	0	0	0	Don't care	8ビット	ゼロ拡張	8ビット	8ビット	8ビット	8ビット	8ビット	
Rd		Rd																					
ビット15	7	ビット15	7																				
0	0	0	0																				
Don't care	8ビット	ゼロ拡張	8ビット																				
8ビット	8ビット	8ビット	8ビット																				
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ Rd : R0~R7、E0~E7 																							
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>EXTU.W</td> <td>Rd</td> <td style="text-align: center;">1</td> <td style="text-align: center;">7</td> <td style="text-align: center;">5</td> <td style="text-align: center;">rd</td> <td></td> <td style="text-align: center;">2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	EXTU.W	Rd	1	7	5	rd		2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数												
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	EXTU.W	Rd	1	7	5	rd		2															
<ul style="list-style-type: none"> ●注意事項 																							

2.2.30(2) EXTU(L)

EXTU (EXTend as Unsigned)		ゼロ拡張					
●オペレーション 0 → (<ビット31~16>of ERd)	●コンディションコード I UI H U N Z V C — — — — 0 † 0 —						
●アセンブラフォーマット EXTU.L ERd	H: 実行前の値が保持されます。 N: 常に0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外のときは0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。						
●オペランドサイズ ロングワード							
●説明 32ビットレジスタERdの下位16ビット (汎用レジスタRd) をゼロ拡張してロングワードサイズにします。ERdの上位16ビット (ビット31~16) に0 (ゼロ) が入ります。							
●使用可能な汎用レジスタ ERd : ER0~ER7							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
レジスタ直接	EXTU.L	ERd	第1バイト 1	第2バイト 7	第3バイト 7 0	第4バイト erd	2
●注意事項							

2. 各命令の説明

2.2.31(1) INC(B)

INC (INCRement)		インクリメント													
●オペレーション Rd+1 →Rd	●コンディションコード														
	I UI H U N Z V C ┌─┬─┬─┬─┬─┬─┬─┬─┐ └─┴─┴─┴─┴─┴─┴─┴─┘														
●アセンブラフォーマット INC.B Rd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外のときは0にクリアされます。 C: 実行前の値が保持されます。														
●オペランドサイズ バイト															
●説明 8ビットレジスタRdの内容（デスティネーションオペランド）に1を加算し、結果を8ビットレジスタRdに格納します。															
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H															
●オペランド形式と実行ステート数															
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数											
レジスタ直接	INC.B	Rd	<table border="1"> <thead> <tr> <th colspan="2">第1バイト</th> <th colspan="2">第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A</td> <td>0</td> <td>rd</td> <td></td> <td></td> </tr> </tbody> </table>		第1バイト		第2バイト		第3バイト	第4バイト	0	A	0	rd	
第1バイト		第2バイト		第3バイト	第4バイト										
0	A	0	rd												
●注意事項 オーバーフローはH'7F+1→H'80のとき発生します。															

2.2.31(2) INC(W)

INC (INCrement)		インクリメント																															
<ul style="list-style-type: none"> ●オペレーション Rd+1 →Rd Rd+2 →Rd 	<ul style="list-style-type: none"> ●コンディションコード 																																
<ul style="list-style-type: none"> ●アセンブラフォーマット INC.W #1, Rd INC.W #2, Rd 	<p style="text-align: center;">I U I H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↓</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↓</td> <td style="width: 15px; height: 15px;">—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外のときは0にクリアされます。 C: 実行前の値が保持されます。</p>			—	—	—	—	↑	↓	↑	↓	—																					
—	—	—	—	↑	↓	↑	↓	—																									
<ul style="list-style-type: none"> ●オペランドサイズ ワード 																																	
<ul style="list-style-type: none"> ●説明 16ビットレジスタRdの内容（デスティネーションオペランド）に1または2を加算し、結果を16ビットレジスタRdに格納します。 																																	
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ Rd : R0~R7、E0~E7 																																	
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>INC.W</td> <td>#1,Rd</td> <td>0</td> <td>B</td> <td>5</td> <td>rd</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>INC.W</td> <td>#2,Rd</td> <td>0</td> <td>B</td> <td>D</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	INC.W	#1,Rd	0	B	5	rd		2	レジスタ直接	INC.W	#2,Rd	0	B	D	rd		2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																										
			第1バイト	第2バイト	第3バイト	第4バイト																											
レジスタ直接	INC.W	#1,Rd	0	B	5	rd		2																									
レジスタ直接	INC.W	#2,Rd	0	B	D	rd		2																									
<ul style="list-style-type: none"> ●注意事項 オーバフローはH'7FFF + 1→H'8000,H'7FFF + 2→H'8001,H'7FFE + 2→H'8000のとき発生します。 																																	

2. 各命令の説明

2.2.31(3) INC(L)

INC (INCrement)		インクリメント																													
<p>●オペレーション</p> <p>ERd+1 →ERd ERd+2 →ERd</p>	<p>●コンディションコード</p> <p>I U I H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↓</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外のときは0にクリアされます。 C: 実行前の値が保持されます。</p>			—	—	—	—	↑	↓	↑	—																				
—	—	—	—	↑	↓	↑	—																								
<p>●アセンブラフォーマット</p> <p>INC.L #1, ERd INC.L #2, ERd</p>																															
<p>●オペランドサイズ</p> <p>ロングワード</p>																															
<p>●説明</p> <p>32ビットレジスタERdの内容（デスティネーションオペランド）に1または2を加算し、結果を32ビットレジスタERdに格納します。</p>																															
<p>●使用可能な汎用レジスタ</p> <p>ERd : ER0~ER7</p>																															
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>INC.L</td> <td>#1,ERd</td> <td>0</td> <td>B</td> <td>7</td> <td>0:erd</td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>INC.L</td> <td>#2,ERd</td> <td>0</td> <td>B</td> <td>F</td> <td>0:erd</td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	INC.L	#1,ERd	0	B	7	0:erd	2	レジスタ直接	INC.L	#2,ERd	0	B	F	0:erd	2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																								
			第1バイト	第2バイト	第3バイト	第4バイト																									
レジスタ直接	INC.L	#1,ERd	0	B	7	0:erd	2																								
レジスタ直接	INC.L	#2,ERd	0	B	F	0:erd	2																								
<p>●注意事項</p> <p>オーバフローはH'7FFFFFFF+1→H'80000000,H'7FFFFFFF+2→H'80000001,H'7FFFFFFE+2→H'80000000のとき発生します。</p>																															

2.2.32 JMP

JMP (JuMP)		無条件ジャンプ						
●オペレーション 実効アドレス → PC	●コンディションコード I U I H U N Z V C — — — — — — — —							
●アセンブラフォーマット JMP <EA>	H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。							
●オペランドサイズ —								
●説明 指定された実効アドレスに無条件分岐します。								
●使用可能な汎用レジスタ ERn : ER0~ER7								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート数	
			第1バイト	第2バイト	第3バイト	第4バイト	ノーマル	アドバンス
レジスタ間接	JMP	@ERn	5	9	0 ern 0			4
絶対アドレス	JMP	@aa:24	5	A	abs			6
メモリ間接	JMP	@@aa:8	5	B	abs			8 10
●注意事項 ノーマルモードとアドバンスモードでは、分岐アドレスの構造および実行ステート数が異なりますので注意してください。 分岐先アドレスは、必ず偶数になるようにしてください。								

2. 各命令の説明

2.2.33 JSR

JSR (Jump to SubRoutine)		サブルーチンジャンプ																																													
<ul style="list-style-type: none"> ●オペレーション PC→@-SP 実効アドレス →PC 	<ul style="list-style-type: none"> ●コンディションコード I U I H U N Z V C <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> 	—	—	—	—	—	—	—	—																																						
—	—	—	—	—	—	—	—																																								
<ul style="list-style-type: none"> ●アセンブラフォーマット JSR <EA> 	<ul style="list-style-type: none"> H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。 																																														
<ul style="list-style-type: none"> ●オペランドサイズ — 																																															
<ul style="list-style-type: none"> ●説明 PCの内容をリスタートアドレスとしてスタックに退避し、指定された実効アドレスに分岐します。退避されるPC値は本命令の直後の命令の先頭アドレスになります。 																																															
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ ERn : ER0~ER7 																																															
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th colspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> <th>ノーマル</th> <th>アドバンスド</th> </tr> </thead> <tbody> <tr> <td>レジスタ間接</td> <td>JSR</td> <td>@ERn</td> <td>5</td> <td>D</td> <td>0 ern 0</td> <td></td> <td></td> <td>6</td> <td>8</td> </tr> <tr> <td>絶対アドレス</td> <td>JSR</td> <td>@aa:24</td> <td>5</td> <td>E</td> <td colspan="3" style="text-align: center;">abs</td> <td>8</td> <td>10</td> </tr> <tr> <td>メモリ間接</td> <td>JSR</td> <td>@@aa:8</td> <td>5</td> <td>F</td> <td>abs</td> <td></td> <td></td> <td>8</td> <td>12</td> </tr> </tbody> </table>			アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数		第1バイト	第2バイト	第3バイト	第4バイト	ノーマル	アドバンスド	レジスタ間接	JSR	@ERn	5	D	0 ern 0			6	8	絶対アドレス	JSR	@aa:24	5	E	abs			8	10	メモリ間接	JSR	@@aa:8	5	F	abs			8	12
アドレッシングモード	ニーモニック	オペランド形式				インストラクションフォーマット				実行ステート数																																					
			第1バイト	第2バイト	第3バイト	第4バイト	ノーマル	アドバンスド																																							
レジスタ間接	JSR	@ERn	5	D	0 ern 0			6	8																																						
絶対アドレス	JSR	@aa:24	5	E	abs			8	10																																						
メモリ間接	JSR	@@aa:8	5	F	abs			8	12																																						
<ul style="list-style-type: none"> ●注意事項 ノーマルモードとアドバンスドモードでは、スタックおよび分岐アドレスの構想が異なりますので注意してください。ノーマルモードのとき退避されるPCの内容は、下位16ビットのみです。分岐先アドレスは、必ず偶数になるようにしてください。 <div style="display: flex; justify-content: space-around; align-items: flex-end;"> <div style="text-align: center;"> <p>ノーマルモード</p> </div> <div style="text-align: center;"> <p>アドバンスドモード</p> </div> </div>																																															

2.2.34(1) LDC(B)

LDC (Load to Control register)		CCR転送					
●オペレーション (EAs) →CCR	●コンディションコード I UI H U N Z V C ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑						
●アセンブラフォーマット LDC.B <EAs>, CCR	I : ソースオペランドの対応するビットの値が格納されます。 H : ソースオペランドの対応するビットの値が格納されます。 N : ソースオペランドの対応するビットの値が格納されます。 Z : ソースオペランドの対応するビットの値が格納されます。 V : ソースオペランドの対応するビットの値が格納されます。 C : ソースオペランドの対応するビットの値が格納されます。						
●オペランドサイズ バイト							
●説明 ソースオペランドをCCRに転送します。 なお、本命令の実行終了時点では、NMIを含めてすべての割り込みは受け付けられません。							
●使用可能な汎用レジスタ Rs : R0L~R7L, R0H~R7H							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
イミディエイト	LDC.B	#xx:8,CCR	0 7	IMM			2
レジスタ直接	LDC.B	Rs,CCR	0 3	0 rs			2
●注意事項							

2. 各命令の説明

2.2.34(2) LDC(W)

LDC (Load to Control register)		CCR転送									
<p>●オペレーション (EAs) →CCR</p>	<p>●コンディションコード</p> <p style="text-align: center;">I U I H U N Z V C</p> <table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> <td style="border: 1px solid black; padding: 2px;">↑</td> </tr> </table> <p>I : ソースオペランドの対応するビットの値が格納されます。 H : ソースオペランドの対応するビットの値が格納されます。 N : ソースオペランドの対応するビットの値が格納されます。 Z : ソースオペランドの対応するビットの値が格納されます。 V : ソースオペランドの対応するビットの値が格納されます。 C : ソースオペランドの対応するビットの値が格納されます。</p>		↑	↑	↑	↑	↑	↑	↑	↑	↑
↑	↑	↑	↑	↑	↑	↑	↑	↑			
<p>●アセンブラフォーマット LDC.W <EAs>, CCR</p>											
<p>●オペランドサイズ ワード</p>											
<p>●説明</p> <p>ソースオペランドをCCRに転送します。CCRはバイトサイズですが転送はワードサイズで行われ、偶数アドレスの内容がCCRに格納されます。 本命令の実行終了時点では、NMIを含めてすべての割り込みは受け付けられません。</p>											
<p>●使用可能な汎用レジスタ ERs : ER0~ER7</p>											

2.2.34(2) LDC(W)

LDC (Load to Control register)		CCR転送																
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット										実行 ステート 数					
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト						
レジスタ間接	LDC.W	@ERs,CCR	0	1	4	0	6	9	0:ers	0							6	
ディスプレイ レジスタ付	LDC.W	@(d*16)ERs,CCR	0	1	4	0	6	F	0:ers	0	disp						8	
レジスタ間接	LDC.W	@(d*24)ERs,CCR	0	1	4	0	7	8	0:ers	0	6	B	2	0	0	0	disp	12
ポストアイン レジスタ間接	LDC.W	@ERs+,CCR	0	1	4	0	6	D	0:ers	0							8	
絶対アドレス	LDC.W	@aa:16,CCR	0	1	4	0	6	B	0	0	abs						8	
	LDC.W	@aa:24,CCR	0	1	4	0	6	B	2	0	0	0	abs			10		

●注意事項

2. 各命令の説明

2.2.35(1) MOV(B)

MOV (MOVE data)		転送					
●オペレーション Rs→Rd	●コンディションコード I UI H U N Z V C — — — — † † 0 —						
●アセンブラフォーマット MOV.B Rs, Rd	H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。						
●オペランドサイズ バイト							
●説明 8ビットレジスタRsの内容を8ビットレジスタRdへ転送します。このとき転送するデータを 検査し、その結果をCCRに反映します。							
●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H Rs : R0L~R7L, R0H~R7H							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
レジスタ直接	MOV.B	Rs,Rd	第1バイト 0 C	第2バイト rs rd	第3バイト	第4バイト	2
●注意事項							

2.2.35(2) MOV(W)

MOV (MOVE data)		転送										
●オペレーション Rs→Rd	●コンディションコード I UI H U N Z V C — — — — † † 0 —											
●アセンブラフォーマット MOV.W Rs, Rd	H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。											
●オペランドサイズ ワード												
●説明 16ビットレジスタRsの内容を16ビットレジスタRdへ転送します。このとき転送するデータを 検査し、その結果をCCRに反映します。												
●使用可能な汎用レジスタ Rd : R0~R7, E0~E7 Rs : R0~R7, E0~E7												
●オペランド形式と実行ステート数												
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数								
レジスタ直接	MOV.W	Rs,Rd	<table border="1"> <thead> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>0 D</td> <td>rs rd</td> <td></td> <td></td> </tr> </tbody> </table>	第1バイト	第2バイト	第3バイト	第4バイト	0 D	rs rd			2
第1バイト	第2バイト	第3バイト	第4バイト									
0 D	rs rd											
●注意事項												

2. 各命令の説明

2.2.35(3) MOV(L)

MOV (MOVE data)		転送																								
<p>●オペレーション ERs→ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	—								
I	UI	H	U	N	Z	V	C																			
—	—	—	—	↑	↑	0	—																			
<p>●アセンブラフォーマット MOV.L ERs, ERd</p>																										
<p>●オペランドサイズ ロングワード</p>																										
<p>●説明 32ビットレジスタERsの内容を32ビットレジスタERdへ転送します。このとき転送するデータを 検査し、その結果をCCRに反映します。</p>																										
<p>●使用可能な汎用レジスタ ERd : ER0~ER7 ERs : ER0~ER7</p>																										
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>MOV.L</td> <td>ERs,ERd</td> <td>0</td> <td>F</td> <td>1</td> <td>ers</td> <td>0</td> <td>erd</td> <td></td> <td></td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	MOV.L	ERs,ERd	0	F	1	ers	0	erd			2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数															
			第1バイト	第2バイト	第3バイト	第4バイト																				
レジスタ直接	MOV.L	ERs,ERd	0	F	1	ers	0	erd			2															
<p>●注意事項</p>																										

2.2.35(4) MOV(B)

MOV (MOVE data)	転送																
<p>●オペレーション (EAs) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>	I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	—
I		UI	H	U	N	Z	V	C									
—		—	—	—	↑	↑	0	—									
<p>●アセンブラフォーマット MOV.B <EAs>, Rd</p>																	
<p>●オペランドサイズ バイト</p>																	
<p>●説明 ソースオペランドの内容を8ビットレジスタRdに転送します。このとき、転送するデータを検査し、その結果をCCRに反映します。</p>																	
<p>●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H Rs : ER0~ER7</p>																	

2.2.35(4) MOV(B)

MOV (MOVE data)		インストラクションフォーマット										転送		
アドレッシングモード	ニーモニック	オペランド形式	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	実行ステップ数			
イミディエイト	MOV.B	#xx:8,Rd	F	rd							2			
レジスタ間接	MOV.B	@ERs,Rd	6	0:ers	rd						4			
ディスプレイレジスタ間接	MOV.B	@(d:16,ERs),Rd	6	0:ers	rd	disp					6			
ポストアインクリメントレジスタ間接	MOV.B	@ERs+,Rd	7	8	0:ers	0	6	A	2	rd	0	0	disp	10
絶対アドレス	MOV.B	@ERs+,Rd	6	C	0:ers	rd								6
	MOV.B	@aa:8,Rd	2	rd	abs									4
	MOV.B	@aa:16,Rd	6	A	0	rd	abs							6
	MOV.B	@aa:24,Rd	6	A	2	rd	0	0		abs				8

●注意事項

「MOV.B @ER7 + Rd」は、SP (ER7) の内容が奇数値となるため使用しないでください。詳細は「3.3.2 例外処理の動作」またはハードウェアマニュアルを参照してください。

@aa:8のアクセス範囲については、各製品のハードウェアマニュアルを参照してください。

2.2.35(5) MOV(W)

MOV (MOVE data)	転送																
<p>●オペレーション (EAs) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↓</td><td>↓</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>	I	UI	H	U	N	Z	V	C	—	—	—	—	↓	↓	0	—
I		UI	H	U	N	Z	V	C									
—		—	—	—	↓	↓	0	—									
<p>●アセンブラフォーマット MOV.W <EAs>, Rd</p>																	
<p>●オペランドサイズ ワード</p>																	
<p>●説明 ソースオペランドの内容を16ビットレジスタRdへ転送します。このとき転送するデータを検査し、その結果をCCRに反映します。</p>																	
<p>●使用可能な汎用レジスタ Rd : R0~R7, E0~E7 ERs : ER0~ER7</p>																	

2.2.35(5) MOV(W)

MOV (Move data)		転送												
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット								実行 ステート 数			
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト				
イミディエイト	MOV.W	#xx:16,Rd	7	9	0	rd	IMM					4		
レジスタ間接	MOV.W	@ERs,Rd	6	9	0:ers	rd						4		
ディスプレ- スメント付	MOV.W	@(d:16,ERs),Rd	6	F	0:ers	rd	disp					6		
レジスタ間接	MOV.W	@(d:24,ERs),Rd	7	8	0:ers	0	6	B	2	rd	0	0	disp	10
ポストアイン- デックス間接	MOV.W	@ERs+,Rd	6	D	0:ers	rd								6
絶対アドレス	MOV.W	@aa:16,Rd	6	B	0	rd	abs							6
	MOV.W	@aa:24,Rd	6	B	2	rd	0	0	0	abs				8

●注意事項

1. アドレス<EAs>は必ず偶数になるようにしてください。
2. 「MOV.W @R7+,Rd」の機械語はPOP.W Rdと同一です。

2.2.35(6) MOV(L)

MOV (MOVE data)		転送
<ul style="list-style-type: none"> ●オペレーション (EAs) →ERd 	<ul style="list-style-type: none"> ●コンディションコード 	
<ul style="list-style-type: none"> ●アセンブラフォーマット MOV.L <EAs>, ERd 	<div style="text-align: center;"> I UI H U N Z V C ┌─┴─┴─┴─┴─┴─┴─┴─┐ ┌─┴─┴─┴─┴─┴─┴─┴─┘ </div> <ul style="list-style-type: none"> H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。 	
<ul style="list-style-type: none"> ●オペランドサイズ ロングワード 		
<ul style="list-style-type: none"> ●説明 <p>ソースオペランドの内容を32ビットレジスタERdへ転送します。このとき転送するデータを検査し、その結果をCCRに反映します。</p> <p>実効アドレスが示す先頭の1ワードのメモリの内容が拡張レジスタEdに格納され、次の1ワードのメモリの内容が汎用レジスタRdに格納されます。</p>		
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ <p>ERd : ER0~ER7 ERs : ER0~ER7</p>		

2.2.35(6) MOV(L)

MOV (MOVE data)		転送																
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット										実行 ステート メント 数					
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト						
イミディエイト	MOV.L	#xx:32,Rd	7	A	0	0	0	IMM							6			
レジスタ間接	MOV.L	@ERs,ERd	0	1	0	0	6	9	0	0	0					8		
ディスプレー スメント付 レジスタ間接	MOV.L	@(d'16)ERs,ERd	0	1	0	0	6	F	0	0	0	disp				10		
ポストアイン クリメント レジスタ間接	MOV.L	@(d'24)ERs,ERd	0	1	0	0	7	8	0	0	0	6	2	0	0	0	disp	14
絶対アドレス	MOV.L	@ERs+,ERd	0	1	0	0	6	D	0	0	0					10		
	MOV.L	@aa:16,ERd	0	1	0	0	6	B	0	0	0	abs				10		
	MOV.L	@aa:24,ERd	0	1	0	0	6	B	2	0	0	0	0	abs			12	

●注意事項

1. アドレス<EAs>は必ず偶数になるようにしてください。
2. 「MOV.L @ER7+,ERd」の機械語はPOPL ERdと同一です。

2.2.35(7) MOV(B)

MOV (MOVE data)	転送																
<p>●オペレーション Rs → (EAd)</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↓</td><td>↓</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>	I	UI	H	U	N	Z	V	C	—	—	—	—	↓	↓	0	—
I		UI	H	U	N	Z	V	C									
—		—	—	—	↓	↓	0	—									
<p>●アセンブラフォーマット MOV.B Rs, <EAd></p>																	
<p>●オペランドサイズ バイト</p>																	
<p>●説明</p> <p>8ビットレジスタRsの内容（ソースオペランド）をデスティネーションのロケーションに転送します。このとき、転送するデータを検査し、その結果をCCRに反映します。</p>																	
<p>●使用可能な汎用レジスタ</p> <p>Rs : R0L~R7L, R0H~R7H ERs : ER0~ER7</p>																	

2.2.35(7) MOV(B)

MOV (MOVE data)		転送												
●オペランド形式と実行ステート数														
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット								実行 ステート 数			
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト				
レジスタ間接	MOV.B	Rs,@ERd	6	8	1	erd	rs						4	
ディスプレー メント付 レジスタ間接	MOV.B	Rs,@(d:16,ERd)	6	E	1	erd	rs	disp					6	
	MOV.B	Rs,@(d:24,ERd)	7	8	0	erd	0	A	A	rs	0	0	disp	10
アドレス レジスタ間接	MOV.B	Rs,@-ERd	6	C	1	erd	rs							6
	MOV.B	Rs,@aa:8	3	rs		abs								4
絶対アドレス	MOV.B	Rs,@aa:16	6	A	8	rs	abs							6
	MOV.B	Rs,@aa:24	6	A	A	rs	0	0	0	abs				8
●注意事項														
<ol style="list-style-type: none"> 1. 「MOV.B Rs,@-ER7」は、SP (ER7) の内容が奇数値となるため使用しないでください。詳細は「3.3.2 例外処理の動作」またはハードウェアマニュアルを参照してください。 2. MOV.B RnL,@-ERnまたはMOV.B RnH,@-ERnを実行すると（実行前のERnの内容-1）の下位RnLまたは上位RnHが転送されます。 														

2.2.35(8) MOV(W)

MOV (MOVE data)	転送																
<p>●オペレーション Rs→ (EAd)</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↓</td><td>↓</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>	I	UI	H	U	N	Z	V	C	—	—	—	—	↓	↓	0	—
I		UI	H	U	N	Z	V	C									
—		—	—	—	↓	↓	0	—									
<p>●アセンブラフォーマット MOV.W Rs, <EAd></p>																	
<p>●オペランドサイズ ワード</p>																	
<p>●説明 16ビットレジスタRsの内容（ソースオペランド）をデスティネーションのロケーションに転送します。このとき転送するデータを検査し、その結果をCCRに反映します。</p>																	
<p>●使用可能な汎用レジスタ Rs : R0~R7, E0~E7 ERd : ER0~ER7</p>																	

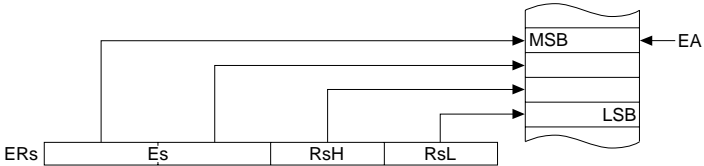
2.2.35(8) MOV(W)

MOV (MOVE data)		インストラクションフォーマット										転送
アドレッシング モード	ニーモ ニック	オペランド 形式	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	実行 ステート 数	
レジスタ間接	MOV.W	Rs,@ERd	6	9 1:erd rs							4	
ディスプレイ メント付 レジスタ間接	MOV.W	Rs,@(d:16;ERd)	6	F 1:erd rs	disp						6	
バイト メント レジスタ間接	MOV.W	Rs,@(d:24;ERd)	7	8 0:erd 0	6 B A rs	0 0	disp				10	
絶対アドレス	MOV.W	Rs,@-ERd	6	D 1:erd rs							6	
	MOV.W	Rs,@aa:16	6	B 8 rs	abs						6	
	MOV.W	Rs,@aa:24	6	B A rs	0 0	abs					8	

●注意事項

1. アドレス<EAd>は必ず偶数になるようにしてください。
2. 「MOV.W Rs,@-ER7」の機械語はPUSH.W Rsと同一です。
3. MOV.W Rn,@-ERnを実行すると（実行前のERnの内容-2）が転送されます。

2.2.35(9) MOV(L)

MOV (MOVe data)	転送																
<ul style="list-style-type: none"> ●オペレーション ERs → (EAd) 	<ul style="list-style-type: none"> ●コンディションコード <table border="1" data-bbox="786 397 1064 459"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↑</td> <td>↑</td> <td>0</td> <td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>	I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	—
I		UI	H	U	N	Z	V	C									
—		—	—	—	↑	↑	0	—									
<ul style="list-style-type: none"> ●アセンブラフォーマット MOV.L ERs, <EAd> 																	
<ul style="list-style-type: none"> ●オペランドサイズ ロングワード 																	
<ul style="list-style-type: none"> ●説明 <p>32ビットレジスタERsの内容（ソースオペランド）をデスティネーションのロケーションに転送します。このとき、転送するデータを検査し、その結果をCCRに反映します。</p> <p>実効アドレスが示す先頭の1ワードに拡張レジスタの内容が、次の1ワードに汎用レジスタRdの内容が格納されます。</p> 																	
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ <p>ERs : ER0~ER7 ERd : ER0~ER7</p>																	

2.2.35(9) MOV(L)

MOV (MOVE data)		インストラクションフォーマット										転送	
アドレッシング モード	ニーモ ニック	オペランド 形式	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト	実行 スラット 数
レジスタ間接	MOV.L	ERs, @ERd	0	1	0	0	6	9	1	erd	0	ers	8
ディスプレイ アメント付 レジスタ間接	MOV.L	ERs, @({16,ERd})	0	1	0	0	6	F	1	erd	0	ers	10
プリ レジスタ間接	MOV.L	ERs, @({24,ERd})	0	1	0	0	7	8	1	erd	0	ers	14
絶対アドレス	MOV.L	ERs, @ERd	0	1	0	0	6	D	1	erd	0	ers	10
	MOV.L	ERs, @aa:16	0	1	0	0	6	B	8	0	ers	abs	10
	MOV.L	ERs, @aa:24	0	1	0	0	6	B	A	0	ers	abs	12

●オペランド形式と実行スラット数

●注意事項

1. アドレス<EAd>は必ず偶数になるようにしてください。
2. 「MOV.L ERs, @-ER7」の機械語はPUSH.L ERsと同一です。
3. MOV.L ERn, @-ERnを実行すると（実行前のERnの内容-4）が転送されます。

2.2.36 MOVFPE

MOVFPE (MOVE From Peripheral with E clock)		E同期データ転送		
●オペレーション (EAs) → Rd E同期	●コンディションコード I UI H U N Z V C — — — — † † 0 —			
●アセンブラフォーマット MOVFPE @aa : 16, Rd	H : 実行前の値が保持されます。 N : 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z : 転送データが0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V : 常に0にクリアされます。 C : 実行前の値が保持されます。			
●オペランドサイズ バイト				
●説明 16ビット絶対アドレスで指定されるメモリの内容を、Eクロックに同期したタイミングで汎用レジスタRdに転送します。このとき転送するデータを検査し、結果をCCRに反映します。 【注】Eクロック出力端子を備えていない製品およびシングルチップモードでは、本命令を使用しないでください。				
●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数
絶対アドレス	MOVFPE	@aa:16,Rd	第1バイト : 6 A 第2バイト : 4 rd 第3バイト : 第4バイト : abs	*
●注意事項 1. 本命令では、上記以外のアドレッシングモードおよびワードサイズ/ロングワードサイズのデータは扱えません。 2. 本命令のデータ転送には、9~16ステートを必要とします。ただし、一定ではありません。詳細は、当該LSIのハードウェアマニュアルを参照してください。				

2. 各命令の説明

2.2.37 MOVTPPE

MOVTPPE (MOVE To Peripheral with E clock)		E同期データ転送																						
<p>●オペレーション Rs → (EAd) E同期</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↓</td> <td>↓</td> <td>0</td> <td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↓	↓	0	—					
I	UI	H	U	N	Z	V	C																	
—	—	—	—	↓	↓	0	—																	
<p>●アセンブラフォーマット MOVTPPE Rs,@aa : 16</p>																								
<p>●オペランドサイズ バイト</p>																								
<p>●説明</p> <p>汎用レジスタRsの内容（ソースオペランド）を、Eクロックに同期したタイミングで、16ビット絶対アドレスで指定されるデスティネーションのロケーションに転送します。このとき転送するデータを検査し、結果をCCRに反映します。</p> <p>【注】Eクロック出力端子を備えていない製品およびシングルチップモードでは、本命令を使用しないでください。</p>																								
<p>●使用可能な汎用レジスタ Rs : R0L~R7L、R0H~R7H</p>																								
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>絶対アドレス</td> <td>MOVTPPE</td> <td>Rs,@aa:16</td> <td>6</td> <td>A</td> <td>C</td> <td>rs</td> <td>abs</td> <td>*</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	絶対アドレス	MOVTPPE	Rs,@aa:16	6	A	C	rs	abs	*
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																	
			第1バイト	第2バイト	第3バイト	第4バイト																		
絶対アドレス	MOVTPPE	Rs,@aa:16	6	A	C	rs	abs	*																
<p>●注意事項</p> <ol style="list-style-type: none"> 1. 本命令では、上記以外のアドレッシングモードおよびワードサイズ/ロングワードサイズのデータは扱えません。 2. 本命令のデータ転送には、9~16ステートを必要とします。ただし、一定ではありません。詳細は、当該LSIのハードウェアマニュアルを参照してください。 																								

2.2.38(1) MULXS(B)

MULXS (MULTiPLY eXtend as Signed)		符号付き乗算																									
<p>●オペレーション Rd×Rs→Rd</p>		<p>●コンディションコード</p> <p>I UI H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↓</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>		—	—	—	—	↑	↓	—	—																
—	—	—	—	↑	↓	—	—																				
<p>●アセンブラフォーマット MULXS.B Rs, Rd</p>																											
<p>●オペランドサイズ バイト</p>																											
<p>●説明</p> <p>16ビットレジスタRdの内容の下位8ビット（デスティネーションオペランド）を8ビットレジスタRsの内容（ソースオペランド）で符号付き乗算し、結果を16ビットレジスタRdに格納します。Rdを汎用レジスタRとしたとき、RsはRdHまたはRdLを指定することも可能です。演算は、8ビット×8ビット→16ビットで行われます。</p> <div style="text-align: center; margin: 10px 0;"> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Don't care</td> <td style="padding: 2px;">Rd</td> <td style="padding: 2px;">被乗数</td> <td style="padding: 2px;">8ビット</td> </tr> </table> × <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Rs</td> <td style="padding: 2px;">乗数</td> <td style="padding: 2px;">8ビット</td> </tr> </table> → <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Rd</td> <td style="padding: 2px;">積</td> <td style="padding: 2px;">16ビット</td> </tr> </table> </div>				Don't care	Rd	被乗数	8ビット	Rs	乗数	8ビット	Rd	積	16ビット														
Don't care	Rd	被乗数	8ビット																								
Rs	乗数	8ビット																									
Rd	積	16ビット																									
<p>●使用可能な汎用レジスタ</p> <p>Rd : R0~R7、E0~E7 Rd : R0L~R7L、R0H~R7H</p>																											
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>MULXS.B</td> <td>Rs,Rd</td> <td>0</td> <td>1</td> <td>C</td> <td>0</td> <td>5</td> <td>0</td> <td>rs</td> <td>rd</td> <td>16</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	MULXS.B	Rs,Rd	0	1	C	0	5	0	rs	rd	16
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																				
			第1バイト	第2バイト	第3バイト	第4バイト																					
レジスタ直接	MULXS.B	Rs,Rd	0	1	C	0	5	0	rs	rd	16																
<p>●注意事項</p>																											

2. 各命令の説明

2.2.38(2) MULXS(W)

MULXS (MULTiPLY eXtend as Signed)		符号付き乗算																									
<p>●オペレーション ERd×Rs→ERd</p>	<p>●コンディションコード</p> <p style="text-align: center;">I U I H U N Z V C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">↑</td> <td style="width: 15px; height: 15px;">↓</td> <td style="width: 15px; height: 15px;">—</td> <td style="width: 15px; height: 15px;">—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>		—	—	—	—	↑	↓	—	—																	
—	—	—	—	↑	↓	—	—																				
<p>●アセンブラフォーマット MULXS.W Rs, ERd</p>																											
<p>●オペランドサイズ ワード</p>																											
<p>●説明</p> <p>32ビットレジスタERdの内容の下位16ビット（デスティネーションオペランド）と16ビットレジスタRsの内容（ソースオペランド）も符号付き乗算し、結果を32ビットレジスタERdに格納します。</p> <p>RsはEdまたはRdを指定することも可能です。 演算は、16ビット×16ビット→32ビットで行われます。</p> <div style="text-align: center; margin: 10px 0;"> <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">ERd</td> <td colspan="2"></td> <td style="padding: 2px;">Rs</td> <td colspan="2"></td> <td style="padding: 2px;">ERd</td> </tr> <tr> <td style="padding: 2px;">Don't care</td> <td style="padding: 2px;">被乗数</td> <td style="padding: 2px;">×</td> <td style="padding: 2px;">乗数</td> <td style="padding: 2px;">→</td> <td style="padding: 2px;">積</td> <td></td> </tr> <tr> <td style="padding: 2px;"></td> <td style="padding: 2px;">16ビット</td> <td style="padding: 2px;"></td> <td style="padding: 2px;">16ビット</td> <td style="padding: 2px;"></td> <td style="padding: 2px;">32ビット</td> <td style="padding: 2px;"></td> </tr> </table> </div>			ERd			Rs			ERd	Don't care	被乗数	×	乗数	→	積			16ビット		16ビット		32ビット					
ERd			Rs			ERd																					
Don't care	被乗数	×	乗数	→	積																						
	16ビット		16ビット		32ビット																						
<p>●使用可能な汎用レジスタ</p> <p>ERd : ER0~ER7 Rs : R0~R7、E0~E7</p>																											
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>MULXS.W</td> <td>Rs,ERd</td> <td>0</td> <td>1</td> <td>C</td> <td>0</td> <td>5</td> <td>2</td> <td>rs</td> <td>0</td> <td>erd</td> <td>24</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	MULXS.W	Rs,ERd	0	1	C	0	5	2	rs	0	erd	24
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数																
			第1バイト	第2バイト	第3バイト	第4バイト																					
レジスタ直接	MULXS.W	Rs,ERd	0	1	C	0	5	2	rs	0	erd	24															
<p>●注意事項</p>																											

2.2.39(1) MULXU(B)

MULXU (MULTiPLY eXtend as Unsigned)		乗算															
●オペレーション Rd×Rs→Rd	●コンディションコード I UI H U N Z V C — — — — — — — — H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。																
●アセンブラフォーマット MULXU.B Rs, Rd																	
●オペランドサイズ バイト																	
●説明 16ビットレジスタRdの内容の下位8ビット（デスティネーションオペランド）と8ビットレジスタRsの内容（ソースオペランド）を符号なし乗算し、結果を16ビットレジスタRdに格納します。Rdを汎用レジスタRとしたとき、RsはRdHまたはRdLを指定することも可能です。演算は、8ビット×8ビット→16ビットで行われます。																	
<table style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">Rd</td> <td></td> <td style="text-align: center;">Rs</td> <td></td> <td style="text-align: center;">Rd</td> </tr> <tr> <td style="text-align: center;">Don't care</td> <td style="text-align: center;">被乗数</td> <td style="text-align: center;">×</td> <td style="text-align: center;">乗数</td> <td style="text-align: center;">→ 積</td> </tr> <tr> <td></td> <td style="text-align: center;">8ビット</td> <td></td> <td style="text-align: center;">8ビット</td> <td style="text-align: center;">16ビット</td> </tr> </table>			Rd		Rs		Rd	Don't care	被乗数	×	乗数	→ 積		8ビット		8ビット	16ビット
Rd		Rs		Rd													
Don't care	被乗数	×	乗数	→ 積													
	8ビット		8ビット	16ビット													
●使用可能な汎用レジスタ Rd : R0~R7、E0~E7 Rd : R0L~R7L、R0H~R7H																	
●オペランド形式と実行ステート数																	
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数										
レジスタ直接	MULXU.B	Rs,Rd	第1バイト	第2バイト	第3バイト	第4バイト	14										
			5	0	rs	rd											
●注意事項																	

2. 各命令の説明

2.2.39(2) MULXU(W)

MULXU (MULTIPLY eXtend as Unsigned)		乗算																					
<p>●オペレーション ERd×Rs→ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	—	—	—	—					
I	UI	H	U	N	Z	V	C																
—	—	—	—	—	—	—	—																
<p>●アセンブラフォーマット MULXU.W Rs, ERd</p>																							
<p>●オペランドサイズ ワード</p>																							
<p>●説明</p> <p>32ビットレジスタERdの内容の下位16ビット（デスティネーションオペランド）と16ビットレジスタRsの内容（ソースオペランド）を符号なし乗算し、結果を32ビットレジスタERdに格納します。RsはEdを指定することも可能です。 演算は、16ビット×16ビット→32ビットで行われます。</p> <div style="text-align: center; margin: 10px 0;"> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="text-align: center;">ERd</td></tr> <tr><td style="text-align: center;">Don't care</td></tr> <tr><td style="text-align: center;">被乗数</td></tr> <tr><td style="text-align: center;">16ビット</td></tr> </table> × <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="text-align: center;">Rs</td></tr> <tr><td style="text-align: center;">乗数</td></tr> <tr><td style="text-align: center;">16ビット</td></tr> </table> → <table border="1" style="display: inline-table; margin-left: 10px;"> <tr><td style="text-align: center;">ERd</td></tr> <tr><td style="text-align: center;">積</td></tr> <tr><td style="text-align: center;">32ビット</td></tr> </table> </div>			ERd	Don't care	被乗数	16ビット	Rs	乗数	16ビット	ERd	積	32ビット											
ERd																							
Don't care																							
被乗数																							
16ビット																							
Rs																							
乗数																							
16ビット																							
ERd																							
積																							
32ビット																							
<p>●使用可能な汎用レジスタ</p> <p>ERd : ER0~ER7 Rs : R0~R7、E0~E7</p>																							
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>MULXU.W</td> <td>Rs,ERd</td> <td style="text-align: center;">5</td> <td style="text-align: center;">2</td> <td style="text-align: center;">rs</td> <td style="text-align: center;">0</td> <td style="text-align: center;">erd</td> <td style="text-align: center;">22</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	MULXU.W	Rs,ERd	5	2	rs	0	erd	22
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数												
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	MULXU.W	Rs,ERd	5	2	rs	0	erd	22															
<p>●注意事項</p>																							

2. 各命令の説明

2.2.40(2) NEG(W)

NEG(NEGate)		2進符号反転																	
●オペレーション 0-Rd→Rd	●コンディションコード																		
	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td style="text-align: center;">—</td><td style="text-align: center;">—</td><td style="text-align: center;">↑</td><td style="text-align: center;">—</td><td style="text-align: center;">↑</td><td style="text-align: center;">↑</td><td style="text-align: center;">↑</td><td style="text-align: center;">↑</td> </tr> </table>			I	UI	H	U	N	Z	V	C	—	—	↑	—	↑	↑	↑	↑
I	UI	H	U	N	Z	V	C												
—	—	↑	—	↑	↑	↑	↑												
●アセンブラフォーマット NEG.W Rd	<p>H: ビット11にボローが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p> <p>C: ビット15にボローが発生したとき1にセットされ、それ以外のときは0にクリアされます。</p>																		
●オペランドサイズ ワード																			
●説明 16ビットレジスタRdの内容（デスティネーションオペランド）の2の補数を取り（H'0000から減算し）、結果を16ビットレジスタRdに格納します。ただし、実行前のRdの内容がH'8000の場合の結果はH'8000となります。																			
●使用可能な汎用レジスタ Rd : R0~R7、E0~E7																			
●オペランド形式と実行ステート数																			
アドレッシング モード	ニーモ ニック	オペランド 形式	<table border="1" style="width: 100%; text-align: center;"> <tr> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> <tr> <td>レジスタ直接</td> <td>NEG.W</td> <td>Rd</td> <td>1 7 9 rd</td> <td>2</td> </tr> </table>	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	NEG.W	Rd	1 7 9 rd	2		
インストラクションフォーマット				実行 ステート 数															
第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	NEG.W	Rd	1 7 9 rd	2															
●注意事項 オーバフローは、実行前のRdの内容がH'8000のとき発生します。																			

2.2.40(3) NEG(L)

NEG(NEGate)		2進符号反転										
●オペレーション 0-ERd→ERd	●コンディションコード											
●アセンブラフォーマット NEG.L ERd	I UI H U N Z V C ─ ─ † ─ † † † †											
●オペランドサイズ ロングワード	H: ビット27にボローが発生したとき1にセットされ、それ以外のときは0にクリアされます。 N: 実行結果が負のとき1にセットされ、それ以外のときは0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外のときは0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外のときは0にクリアされます。 C: ビット31にボローが発生したとき1にセットされ、それ以外のときは0にクリアされます。											
●説明 32ビットレジスタERdの内容（デスティネーションオペランド）の2の補数を取り（H'00000000から減算し）、結果を32ビットレジスタERdに格納します。ただし、実行前のERdの内容がH'80000000の場合の結果はH'80000000となります。												
●使用可能な汎用レジスタ ERd : ER0~ER7												
●オペランド形式と実行ステート数												
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数								
レジスタ直接	NEG.L	ERd	<table border="1"> <thead> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>7</td> <td>B 0</td> <td>erd</td> </tr> </tbody> </table>	第1バイト	第2バイト	第3バイト	第4バイト	1	7	B 0	erd	2
第1バイト	第2バイト	第3バイト	第4バイト									
1	7	B 0	erd									
●注意事項 オーバフローは、実行前のERdの内容がH'80000000のとき発生します。												

2. 各命令の説明

2.2.41 NOP

NOP(No OPeration)		無操作					
●オペレーション PC+2→PC	●コンディションコード I U I H U N Z V C — — — — — — — —						
●アセンブラフォーマット NOP	H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。						
●オペランドサイズ —							
●説明 PCのインクリメントのみを行い、次の命令に実行が移ります。CPUの内部状態には影響を与えません。							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
—	NOP		0	0	0	0	2
●注意事項							

2.2.42(1) NOT(B)

NOT(NOT=logical complement)		論理反転																				
●オペレーション ~Rd→Rd	●コンディションコード I UI H U N Z V C ┌─┬─┬─┬─┬─┬─┬─┬─┐ └─┴─┴─┴─┴─┴─┴─┴─┘																					
●アセンブラフォーマット NOT.B Rd																						
●オペランドサイズ バイト																						
H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。																						
●説明 8ビットレジスタRdの内容（デスティネーションオペランド）の1の補数を取り、結果を8ビットレジスタRdに格納します。																						
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H																						
●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>NOT.B</td> <td>Rd</td> <td>1</td> <td>7</td> <td>0</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	NOT.B	Rd	1	7	0	rd	2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	NOT.B	Rd	1	7	0	rd	2															
●注意事項																						

2. 各命令の説明

2.2.42(2) NOT(W)

NOT(NOT=logical complement)		論理反転					
●オペレーション ~Rd→Rd	●コンディションコード I U I H U N Z V C ┌─┬─┬─┬─┬─┬─┬─┬─┐ └─┴─┴─┴─┴─┴─┴─┴─┘						
●アセンブラフォーマット NOT.W Rd							
●オペランドサイズ ワード							
●説明 16ビットレジスタRdの内容（デスティネーションオペランド）の1の補数を取り、結果を16ビットレジスタRdに格納します。		H： 実行前の値が保持されます。 N： 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z： 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V： 常に0にクリアされます。 C： 実行前の値が保持されます。					
●使用可能な汎用レジスタ Rd : R0~R7、E0~E7							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
レジスタ直接	NOT.W	Rd	第1バイト	第2バイト	第3バイト	第4バイト	
			1	7	1	rd	
							2
●注意事項							

2.2.42(3) NOT(L)

NOT(NOT=logical complement)		論理反転																				
●オペレーション ~ERd→ERd	●コンディションコード I U I H U N Z V C ┌─┬─┬─┬─┬─┬─┬─┬─┐ └─┴─┴─┴─┴─┴─┴─┴─┘																					
●アセンブラフォーマット NOT.L ERd																						
●オペランドサイズ ロングワード																						
H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。																						
●説明 32ビットレジスタERdの内容（デスティネーションオペランド）の1の補数を取り、結果を32ビットレジスタERdに格納します。																						
●使用可能な汎用レジスタ ERd : ER0~ER7																						
●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>NOT.L</td> <td>ERd</td> <td>1</td> <td>7</td> <td>3</td> <td>0 erd</td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	NOT.L	ERd	1	7	3	0 erd	2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	NOT.L	ERd	1	7	3	0 erd	2															
●注意事項																						

2. 各命令の説明

2.2.43(1) OR(B)

OR(inclusive OR logical)		論理和					
●オペレーション Rd _v (EAs) →Rd	●コンディションコード I UI H U N Z V C — — — — † † 0 —						
●アセンブラフォーマット OR.B <EAs>, Rd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。						
●オペランドサイズ バイト							
●説明 8ビットレジスタRdの内容（デスティネーションオペランド）と、ソースオペランドの論理和をとり、結果を8ビットレジスタRdに格納します。							
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H Rs : R0L~R7L、R0H~R7H							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
イミディエイト	OR.B	#xx:8,Rd	C	rd	IMM		2
レジスタ直接	OR.B	Rs,Rd	1	4	rs	rd	2
●注意事項							

2.2.43(2) OR(W)

OR(inclusive OR logical)		論理和						
●オペレーション Rd _v (EAs) → Rd	●コンディションコード I UI H U N Z V C — — — — † † 0 —							
●アセンブラフォーマット OR.W <EAs>, Rd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。							
●オペランドサイズ ワード								
●説明 16ビットレジスタRdの内容 (デスティネーションオペランド) と、ソースオペランドの論理和をとり、結果を16ビットレジスタRdに格納します。								
●使用可能な汎用レジスタ Rd: R0~R7, E0~E7 Rs: R0~R7, E0~E7								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	
			第1バイト	第2バイト	第3バイト	第4バイト		
イミディエイト	OR.W	#xx:16,Rd	7	9	4	rd	IMM	4
レジスタ直接	OR.W	Rs,Rd	6	4	rs	rd		2
●注意事項								

2. 各命令の説明

2.2.43(3) OR(L)

OR(inclusive OR logical)		論理和								
●オペレーション ERd _v (EAs) →ERd	●コンディションコード I UI H U N Z V C — — — — † † 0 —									
●アセンブラフォーマット OR.L <EAs>, ERd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。									
●オペランドサイズ ロングワード										
●説明 32ビットレジスタERdの内容 (デスティネーションオペランド) と、ソースオペランドの論理和をとり、結果を32ビットレジスタERdに格納します。										
●使用可能な汎用レジスタ ERd : ER0~ER7 ERs : ER0~ER7										
●オペランド形式と実行ステート数										
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット						実行 ステート 数	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト		
イミディエイト	OR.L	#xx:32,ERd	7	A	4	0:erd	IMM			6
レジスタ直接	OR.L	ERs,ERd	0	1	F	0	6	4	0:ers0:erd	4
●注意事項										

2.2.44 ORC

ORC(inclusive OR Control register)		CCRとの論理和		
●オペレーション CCR \vee #IMM \rightarrow CCR	●コンディションコード I UI H U N Z V C ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑			
●アセンブラフォーマット ORC #xx : 8 CCR	I : 実行結果の対応するビットの値が格納されます。 UI : 実行結果の対応するビットの値が格納されます。 H : 実行結果の対応するビットの値が格納されます。 U : 実行結果の対応するビットの値が格納されます。 N : 実行結果の対応するビットの値が格納されます。 Z : 実行結果の対応するビットの値が格納されます。 V : 実行結果の対応するビットの値が格納されます。 C : 実行結果の対応するビットの値が格納されます。			
●オペランドサイズ バイト				
●説明 CCRの内容とイミディエイトデータの論理和をとり、結果をCCRに格納します。 本命令の実行終了時点では、NMIを含めてすべての割り込みは受け付けられません。				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット 第1バイト 第2バイト 第3バイト 第4バイト	実行 ステート 数
イミディエイト	ORC	#xx:8,CCR	0 4 IMM	2
●注意事項				

2. 各命令の説明

2.2.45(1) POP(W)

POP(POP data)			スタックよりデータ復帰				
●オペレーション @SP+→Rn	●コンディションコード I U I H U N Z V C — — — — † † 0 —						
●アセンブラフォーマット POP.W Rn	H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。						
●オペランドサイズ ワード							
●説明 スタックから16ビットレジスタRnへデータを復帰します。このとき復帰するデータを調査し、その結果をCCRに反映します。							
●使用可能な汎用レジスタ Rn : R0~R7、E0~E7							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
—	POP.W	Rn	6	D	7	m	6
●注意事項 本命令は、MOV.W @SP+,Rnと同一です。							

2.2.45(2) POP(L)

POP(POP data)			スタックよりデータ復帰									
●オペレーション @SP+→ERn	●コンディションコード I UI H U N Z V C — — — — † † 0 —											
●アセンブラフォーマット POP.L ERn	H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。											
●オペランドサイズ ロングワード												
●説明 スタックから32ビットレジスタERnへデータを復帰します。このとき復帰するデータを調査し、その結果をCCRに反映します。												
●使用可能な汎用レジスタ ERn : ER0~ER7												
●オペランド形式と実行ステート数												
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数					
			第1バイト	第2バイト	第3バイト	第4バイト						
—	POP.L	ERn	0	1	0	0	6	D	7	0	ern	10
●注意事項 本命令は、MOV.L @SP+,ERnと同一です。												

2. 各命令の説明

2.2.46(1) PUSH(W)

PUSH(PUSH date)		スタックヘデータ退避		
●オペレーション Rn→@-SP	●コンディションコード I U I H U N Z V C — — — — † † 0 —			
●アセンブラフォーマット PUSH.W Rn	H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。			
●オペランドサイズ ワード				
●説明 16ビットレジスタRnの内容をスタックに退避します。このとき退避するデータを検査し、その結果をCCRに反映します。				
●使用可能な汎用レジスタ Rn : R0~R7、E0~E7				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数
—	PUSH.W	Rn	第1バイト: 6 D 第2バイト: F m 第3バイト: 第4バイト:	6
●注意事項				
1. 本命令は、MOV.W Rn,@-SPと同一です。 2. PUSH.W R7または、PUSH.W E7を実行するとアドレス計算（ER7-2→ER7実行）後のR7またはE7がスタックに退避されます。				

2.2.46(2) PUSH(L)

PUSH(PUSH date)		スタックへデータ退避		
●オペレーション ERn→@-SP	●コンディションコード I UI H U N Z V C — — — — † † 0 —			
●アセンブラフォーマット PUSH.L ERn	H: 実行前の値が保持されます。 N: 転送データが負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 転送データが0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。			
●オペランドサイズ ロングワード				
●説明 32ビットレジスタERnの内容をスタックに退避します。このとき退避するデータを検査し、その結果をCCRに反映します。				
●使用可能な汎用レジスタ ERn : ER0~ER7				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット 第1バイト 第2バイト 第3バイト 第4バイト	実行 ステート 数
—	PUSH.L	ERn	0 1 0 0 6 D F 0 ern	10
●注意事項 1. 本命令は、MOV.L ERn,@-SPと同一です。 2. PUSH.L ER7を実行すると実効アドレス計算（ER7-4→ER7実行）後のER7がスタックに退避されます。				

2. 各命令の説明

2.2.47(1) ROTL(B)

ROTL(ROTate Left)		ローテート																				
<p>●オペレーション Rd (左ローテート) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット7値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	↑															
<p>●アセンブラフォーマット ROTL.B Rd</p>																						
<p>●オペランドサイズ バイト</p>																						
<p>●説明</p> <p>8ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、左方向に1ビットローテート (回転) します。ローテートしてシフトアウトしたビットは、ビット0に戻り、かつキャリフラグに反映されます。</p> <div style="text-align: center;"> </div>																						
<p>●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H</p>																						
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTL.B</td> <td>Rd</td> <td>1</td> <td>2</td> <td>8</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>			アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTL.B	Rd	1	2	8	rd	2
アドレッシングモード	ニーモニック	オペランド形式				インストラクションフォーマット					実行ステート数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	ROTL.B	Rd	1	2	8	rd	2															
<p>●注意事項</p>																						

2.2.47(2) ROTL(W)

ROTL(ROTate Left)		ローテート																				
<p>●オペレーション Rd (左ローテート) →Rd</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット15値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	↑															
<p>●アセンブラフォーマット ROTL.W Rd</p>																						
<p>●オペランドサイズ ワード</p>																						
<p>●説明</p> <p>16ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、左方向に1ビットローテート (回転) します。ローテートしてシフトアウトしたビットは、ビット0に戻り、かつキャリフラグに反映されます。</p>																						
<p>●使用可能な汎用レジスタ Rd : R0~R7、E0~E7</p>																						
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTL.W</td> <td>Rd</td> <td>1</td> <td>2</td> <td>9</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>			アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTL.W	Rd	1	2	9	rd	2
アドレッシングモード	ニーモニック	オペランド形式				インストラクションフォーマット					実行ステート数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	ROTL.W	Rd	1	2	9	rd	2															
<p>●注意事項</p>																						

2. 各命令の説明

2.2.47(3) ROTL(L)

ROTL(ROTate Left)		ローテート																				
<p>●オペレーション ERd (左ローテート) →ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット31の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	—				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	—															
<p>●アセンブラフォーマット ROTL.L ERd</p>																						
<p>●オペランドサイズ ロングワード</p>																						
<p>●説明</p> <p>32ビットレジスタERdの内容（テストオペランド）のビット群を、左方向に1ビットローテート（回転）します。ローテートしてシフトアウトしたビットは、ビット0に戻り、かつキャリフラグに反映されます。</p> <div style="text-align: center;"> </div>																						
<p>●使用可能な汎用レジスタ ERd : ER0~ER7</p>																						
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTL.L</td> <td>ERd</td> <td>1</td> <td>2</td> <td>B 0</td> <td>erd</td> <td>2</td> </tr> </tbody> </table>			アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTL.L	ERd	1	2	B 0	erd	2
アドレッシングモード	ニーモニック	オペランド形式				インストラクションフォーマット					実行ステート数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	ROTL.L	ERd	1	2	B 0	erd	2															
<p>●注意事項</p>																						

2.2.48(1) ROTR(B)

ROTR(ROTate Right)		ローテート					
●オペレーション Rd (右ローテート) →Rd	●コンディションコード I UI H U N Z V C — — — — † † 0 †						
●アセンブラフォーマット ROTR.B Rd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。						
●オペランドサイズ バイト							
●説明 8ビットレジスタRdの内容（デスティネーションオペランド）のビット群を、右方向に1ビットローテート（回転）します。ローテートしてシフトアウトしたビットは、ビット7に戻り、かつキャリフラグに反映されます。							
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
レジスタ直接	ROTR.B	Rd	第1バイト 1	第2バイト 3	第3バイト 8	第4バイト rd	2
●注意事項							

2. 各命令の説明

2.2.48(2) ROTR(W)

ROTR(ROTate Right)		ローテート																				
<ul style="list-style-type: none"> ●オペレーション Rd (右ローテート) →Rd 	<ul style="list-style-type: none"> ●コンディションコード 																					
<ul style="list-style-type: none"> ●アセンブラフォーマット ROTR.W Rd 	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	↑															
<ul style="list-style-type: none"> ●オペランドサイズ ワード 																						
<ul style="list-style-type: none"> ●説明 <p>16ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、右方向に1ビットローテート (回転) します。ローテートしてシフトアウトしたビットは、ビット15に戻り、かつキャリフラグに反映されます。</p>																						
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ Rd : R0~R7、E0~E7 																						
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTR.W</td> <td>Rd</td> <td>1</td> <td>3</td> <td>9</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTR.W	Rd	1	3	9	rd	2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	ROTR.W	Rd	1	3	9	rd	2															
<ul style="list-style-type: none"> ●注意事項 																						

2.2.48(3) ROTR(L)

ROTR(ROtate Right)		ローテート						
●オペレーション ERd (右ローテート) →ERd	●コンディションコード I UI H U N Z V C — — — — † † 0 †							
●アセンブラフォーマット ROTR.L ERd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。							
●オペランドサイズ ロングワード								
●説明 32ビットレジスタERdの内容 (テストオペランド) のビット群を、右方向に1ビットローテート (回転) します。ローテートしてシフトアウトしたビットは、ビット31に戻り、かつキャリフラグに反映されます。								
●使用可能な汎用レジスタ ERd : ER0~ER7								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	
レジスタ直接	ROTR.L	ERd	第1バイト	第2バイト	第3バイト	第4バイト		
			1	3	B	0	erd	2
●注意事項								

2. 各命令の説明

2.2.49(1) ROTXL(B)

ROTXL(ROtate with eXtend carry Left)		キャリ付ローテート																						
<p>●オペレーション Rd (キャリ付左ローテート) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット7の値が格納されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑					
I	UI	H	U	N	Z	V	C																	
—	—	—	—	↑	↑	0	↑																	
<p>●アセンブラフォーマット ROTXL.B Rd</p>																								
<p>●オペランドサイズ バイト</p>																								
<p>●説明</p> <p>8ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、キャリフラグを含めて左方向に1ビットローテート (回転) します。ビット0にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに反映されます。</p> <div style="text-align: center;"> </div>																								
<p>●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H</p>																								
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTXL.B</td> <td>Rd</td> <td>1</td> <td>2</td> <td>0</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTXL.B	Rd	1	2	0	rd		2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																	
			第1バイト	第2バイト	第3バイト	第4バイト																		
レジスタ直接	ROTXL.B	Rd	1	2	0	rd		2																
<p>●注意事項</p>																								

2.2.49(2) ROTXL(W)

ROTXL(ROtate with eXtend carry Left)		キャリ付ローテート																						
<p>●オペレーション Rd (キャリ付左ローテート) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット15の値が格納されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑					
I	UI	H	U	N	Z	V	C																	
—	—	—	—	↑	↑	0	↑																	
<p>●アセンブラフォーマット ROTXL.W Rd</p>																								
<p>●オペランドサイズ ワード</p>																								
<p>●説明</p> <p>16ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、キャリフラグを含めて左方向に1ビットローテート (回転) します。ビット0にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。</p> <div style="text-align: center;"> </div>																								
<p>●使用可能な汎用レジスタ Rd : R0~R7、E0~E7</p>																								
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTXL.W</td> <td>Rd</td> <td>1</td> <td>2</td> <td>1</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTXL.W	Rd	1	2	1	rd		2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																	
			第1バイト	第2バイト	第3バイト	第4バイト																		
レジスタ直接	ROTXL.W	Rd	1	2	1	rd		2																
<p>●注意事項</p>																								

2. 各命令の説明

2.2.49(3) ROTXL(L)

ROTXL(ROtate with eXtend carry Left)		キャリ付ローテート																				
<p>●オペレーション ERd (キャリ付左ローテート) →ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット31の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	↑															
<p>●アセンブラフォーマット ROTXL.L ERd</p>																						
<p>●オペランドサイズ ロングワード</p>																						
<p>●説明</p> <p>32ビットレジスタERdの内容 (デスティネーションオペランド) のビット群を、キャリフラグを含めて左方向に1ビットローテート (回転) します。ビット0にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに反映されます。</p> <div style="text-align: center;"> </div>																						
<p>●使用可能な汎用レジスタ ERd : ER0~ER7</p>																						
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTXL.L</td> <td>ERd</td> <td>1</td> <td>2</td> <td>3</td> <td>0 erd</td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTXL.L	ERd	1	2	3	0 erd	2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	ROTXL.L	ERd	1	2	3	0 erd	2															
<p>●注意事項</p>																						

2.2.50(1) ROTXR(B)

ROTXR(ROTate with eXtend carry Right)		キャリ付ローテート																						
<p>●オペレーション Rd (キャリ付右ローテート) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑					
I	UI	H	U	N	Z	V	C																	
—	—	—	—	↑	↑	0	↑																	
<p>●アセンブラフォーマット ROTXR.B Rd</p>																								
<p>●オペランドサイズ バイト</p>																								
<p>●説明</p> <p>8ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、キャリフラグを含めて右方向に1ビットローテート (回転) します。ビット7にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。</p> <div style="text-align: center;"> </div>																								
<p>●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H</p>																								
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTXR.B</td> <td>Rd</td> <td>1</td> <td>3</td> <td>0</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTXR.B	Rd	1	3	0	rd		2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																	
			第1バイト	第2バイト	第3バイト	第4バイト																		
レジスタ直接	ROTXR.B	Rd	1	3	0	rd		2																
<p>●注意事項</p>																								

2. 各命令の説明

2.2.50(2) ROTXR(W)

ROTXR(ROTate with eXtend carry Right)		キャリ付ローテート																						
<p>●オペレーション Rd (キャリ付右ローテート) → Rd</p>		<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑					
I	UI	H	U	N	Z	V	C																	
—	—	—	—	↑	↑	0	↑																	
<p>●アセンブラフォーマット ROTXR.W Rd</p>																								
<p>●オペランドサイズ ワード</p>																								
<p>●説明</p> <p>16ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、キャリフラグを含めて右方向に1ビットローテート (回転) します。ビット15にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに格納されます。</p> <div style="text-align: center;"> </div>																								
<p>●使用可能な汎用レジスタ Rd : R0~R7、E0~E7</p>																								
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>ROTXR.W</td> <td>Rd</td> <td>1</td> <td>3</td> <td>1</td> <td>rd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	ROTXR.W	Rd	1	3	1	rd		2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																	
			第1バイト	第2バイト	第3バイト	第4バイト																		
レジスタ直接	ROTXR.W	Rd	1	3	1	rd		2																
<p>●注意事項</p>																								

2.2.50(3) ROTXR(L)

ROTXR(ROTate with eXtend carry Right)		キャリ付ローテート		
●オペレーション ERd (キャリ付右ローテート) →ERd	●コンディションコード I UI H U N Z V C — — — — † † 0 †			
●アセンブラフォーマット ROTXR.L ERd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。			
●オペランドサイズ ロングワード				
●説明 32ビットレジスタERdの内容 (デスティネーションオペランド) のビット群を、キャリフラグを含めて右方向に1ビットローテート (回転) します。ビット31にはキャリフラグの値が入り、ローテートしてシフトアウトしたビットはキャリフラグに反映されます。				
●使用可能な汎用レジスタ ERd : ER0~ER7				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数
レジスタ直接	ROTXR.L	ERd	第1バイト: 1 3 第2バイト: 3 0 第3バイト: erd	2
●注意事項				

2. 各命令の説明

2.2.51 RTE

RTE (ReTurn from Exception)		例外処理からのリターン																					
<ul style="list-style-type: none"> ●オペレーション @SP+→CCR @SP+→PC 	<ul style="list-style-type: none"> ●コンディションコード I UI H U N Z V C ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ 																						
<ul style="list-style-type: none"> ●アセンブラフォーマット RTE 	<ul style="list-style-type: none"> I : スタックの内容の対応するビットの値が格納されます。 UI : スタックの内容の対応するビットの値が格納されます。 H : スタックの内容の対応するビットの値が格納されます。 U : スタックの内容の対応するビットの値が格納されます。 N : スタックの内容の対応するビットの値が格納されます。 Z : スタックの内容の対応するビットの値が格納されます。 V : スタックの内容の対応するビットの値が格納されます。 C : スタックの内容の対応するビットの値が格納されます。 																						
<ul style="list-style-type: none"> ●オペランドサイズ - 																							
<ul style="list-style-type: none"> ●説明 例外処理ルーチンから復帰します。スタックからCCRとPCを復帰し、復帰したPCが示すアドレスから処理を行います。本命令を実行する直前のCCRおよびPCの内容は失われます。 																							
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>-</td> <td>RTE</td> <td></td> <td>5</td> <td>6</td> <td>7</td> <td>0</td> <td>10</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	-	RTE		5	6	7	0	10
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																
			第1バイト	第2バイト	第3バイト	第4バイト																	
-	RTE		5	6	7	0	10																
<ul style="list-style-type: none"> ●注意事項 ノーマルモードとアドバンスモードでは、スタックの構造が異なりますので注意してください。 <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>ノーマルモード</p> </div> <div style="text-align: center;"> <p>アドバンスモード</p> </div> </div>																							

2.2.52 RTS

RTS (ReTurn from Subroutine)		サブルーチンリターン						
●オペレーション @SP+→PC	●コンディションコード I U I H U N Z V C — — — — — — — —							
●アセンブラフォーマット RTS	H : 実行前の値が保持されます。 N : 実行前の値が保持されます。 Z : 実行前の値が保持されます。 V : 実行前の値が保持されます。 C : 実行前の値が保持されます。							
●オペランドサイズ —								
●説明 サブルーチンから復帰します。スタックからPCを復帰し、復帰したPCが示すアドレスから処理を行います。本命令を実行する直前のPCの内容は失われます。								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート数	
			第1バイト	第2バイト	第3バイト	第4バイト	ノーマル	アドバンスト
—	RTS		5	4	7	0	8	10
●注意事項 ノーマルモードとアドバンストモードでは、スタックの構造および実行ステート数が異なりますので注意してください。 ノーマルモードのとき復帰されるPCの内容は下位16ビットのみです。								
<p>ノーマルモード</p>		<p>アドバンストモード</p>						

2. 各命令の説明

2.2.53(1) SHAL(B)

SHAL(SHift Arithmetic Left)		算術シフト																					
<p>●オペレーション Rd (左算術シフト) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: 実行前のビット7の値が格納されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	↑	↑				
I	UI	H	U	N	Z	V	C																
—	—	—	—	↑	↑	↑	↑																
<p>●アセンブラフォーマット SHAL.B Rd</p>																							
<p>●オペランドサイズ バイト</p>																							
<p>●説明</p> <p>8ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、左方向に1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット0には0が格納されます。</p> <div style="text-align: center;"> </div>																							
<p>●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H</p>																							
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHAL.B</td> <td>Rd</td> <td>1</td> <td>0</td> <td>8</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHAL.B	Rd	1	0	8	rd	2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	SHAL.B	Rd	1	0	8	rd	2																
<p>●注意事項</p> <p>本命令とSHLL命令とは、オーバフローフラグの動作が異なります。</p>																							

2.2.53(2) SHAL(W)

SHAL(SHift Arithmetic Left)		算術シフト					
●オペレーション Rd (左算術シフト) →Rd	●コンディションコード						
●アセンブラフォーマット SHAL.W Rd	I UI H U N Z V C ── ── ── ── † † † †						
●オペランドサイズ ワード	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: 実行前のビット15の値が格納されます。						
●説明							
16ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、左方向へ算術的に1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット0には0が格納されます。							
●使用可能な汎用レジスタ Rd : R0~R7、E0~E7							
●オペランド形式と実行ステート数							
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数
			第1バイト	第2バイト	第3バイト	第4バイト	
レジスタ直接	SHAL.W	Rd	1	0	9	rd	2
●注意事項							
本命令とSHLL命令とでは、オーバフローフラグの動作が異なります。							

2. 各命令の説明

2.2.53(3) SHAL(L)

SHAL(SHift Arithmetic Left)		算術シフト																					
<p>●オペレーション ERd (左算術シフト) →ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: 実行前のビット31の値が格納されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	↑	↑				
I	UI	H	U	N	Z	V	C																
—	—	—	—	↑	↑	↑	↑																
<p>●アセンブラフォーマット SHAL.L ERd</p>																							
<p>●オペランドサイズ ロングワード</p>																							
<p>●説明</p> <p>32ビットレジスタERdの内容 (デスティネーションオペランド) のビット群を、左方向へ算術的に1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット0には0が格納されます。</p> <div style="text-align: center;"> </div>																							
<p>●使用可能な汎用レジスタ ERd : ER0~ER7</p>																							
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHAL.L</td> <td>ERd</td> <td>1</td> <td>0</td> <td>B</td> <td>0 erd</td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHAL.L	ERd	1	0	B	0 erd	2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	SHAL.L	ERd	1	0	B	0 erd	2																
<p>●注意事項</p> <p>本命令とSHLL命令とは、オーバフローフラグの動作が異なります。</p>																							

2.2.54(1) SHAR(B)

SHAR(SHift Arithmetic Right)		算術シフト																					
<p>●オペレーション Rd (右算術シフト) →Rd</p>		<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C																
—	—	—	—	↑	↑	0	↑																
<p>●アセンブラフォーマット SHAR.B Rd</p>																							
<p>●オペランドサイズ バイト</p>																							
<p>●説明</p> <p>8ビットレジスタRdの内容（デスティネーションオペランド）のビット群を、右方向へ算術的に1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット7にはシフト処理前のビット7がセットされます。ビット7は変化しないので、符号変化は起こりません。</p> <div style="text-align: center;"> </div>																							
<p>●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H</p>																							
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHAR.B</td> <td>Rd</td> <td>1</td> <td>1</td> <td>8</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHAR.B	Rd	1	1	8	rd	2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	SHAR.B	Rd	1	1	8	rd	2																
<p>●注意事項</p>																							

2. 各命令の説明

2.2.54(2) SHAR(W)

SHAR(SHift Arithmetic Right)		算術シフト																					
<p>●オペレーション Rd (右算術シフト) →Rd</p>		<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C																
—	—	—	—	↑	↑	0	↑																
<p>●アセンブラフォーマット SHAR.W Rd</p>																							
<p>●オペランドサイズ ワード</p>																							
<p>●説明</p> <p>16ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、右方向へ算術的に1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット15にはシフト処理前のビット15が格納されます。ビット15は変化しないので、符号変化は起こりません。</p> <div style="text-align: center;"> </div>																							
<p>●使用可能な汎用レジスタ Rd : R0~R7、E0~E7</p>																							
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHAR.W</td> <td>Rd</td> <td>1</td> <td>1</td> <td>9</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHAR.W	Rd	1	1	9	rd	2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	SHAR.W	Rd	1	1	9	rd	2																
<p>●注意事項</p>																							

2.2.54(3) SHAR(L)

SHAR(SHift Arithmetic Right)		算術シフト		
●オペレーション ERd (右算術シフト) →ERd	●コンディションコード I UI H U N Z V C — — — — † † 0 †			
●アセンブラフォーマット SHAR.L ERd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。			
●オペランドサイズ ロングワード				
●説明 32ビットレジスタERdの内容 (デスティネーションオペランド) のビット群を、右方向へ算術的に1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット31にはシフト処理前のビット31が格納されます。ビット31は変化しないので、符号変化は起こりません。				
●使用可能な汎用レジスタ ERd : ER0~ER7				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数
レジスタ直接	SHAR.L	ERd	第1バイト: 1 1 第2バイト: B 0 第3バイト: erd 第4バイト:	
●注意事項				

2. 各命令の説明

2.2.55(1) SHLL(B)

SHLL(SHift Logical Left)		論理シフト																				
<ul style="list-style-type: none"> ●オペレーション Rd (左論理シフト) →Rd 	<ul style="list-style-type: none"> ●コンディションコード <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット7の値が格納されます。 		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	↑															
<ul style="list-style-type: none"> ●アセンブラフォーマット SHLL.B Rd 																						
<ul style="list-style-type: none"> ●オペランドサイズ バイト 																						
<ul style="list-style-type: none"> ●説明 8ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、左方向へ1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット0には0が格納されます。 <div style="text-align: center;"> </div>																						
<ul style="list-style-type: none"> ●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H 																						
<ul style="list-style-type: none"> ●オペランド形式と実行ステート数 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHLL.B</td> <td>Rd</td> <td>1</td> <td>0</td> <td>0</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHLL.B	Rd	1	0	0	rd	2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	SHLL.B	Rd	1	0	0	rd	2															
<ul style="list-style-type: none"> ●注意事項 本命令とSHAL命令とでは、オーバーフローフラグの動作が異なります。 																						

2.2.55(2) SHLL(W)

SHLL(SHift Logical Left)		論理シフト																				
<p>●オペレーション Rd (左論理シフト) →Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット15の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	↑															
<p>●アセンブラフォーマット SHLL.W Rd</p>																						
<p>●オペランドサイズ ワード</p>																						
<p>●説明</p> <p>16ビットレジスタRdの内容（デスティネーションオペランド）のビット群を、左方向へ1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット0には0が格納されます。</p> <div style="text-align: center;"> </div>																						
<p>●使用可能な汎用レジスタ Rd : R0~R7、E0~E7</p>																						
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHLL.W</td> <td>Rd</td> <td>1</td> <td>0</td> <td>1</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>			アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHLL.W	Rd	1	0	1	rd	2
アドレッシング モード	ニーモ ニック	オペランド 形式				インストラクションフォーマット					実行 ステート 数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	SHLL.W	Rd	1	0	1	rd	2															
<p>●注意事項</p> <p>本命令とSHAL命令とは、オーバーフローフラグの動作が異なります。</p>																						

2. 各命令の説明

2.2.55(3) SHLL(L)

SHLL(SHift Logical Left)		論理シフト																				
<p>●オペレーション ERd (左論理シフト) →ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>↑</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット31の値が格納されます。</p>		I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	↑				
I	UI	H	U	N	Z	V	C															
—	—	—	—	↑	↑	0	↑															
<p>●アセンブラフォーマット SHLL.L ERd</p>																						
<p>●オペランドサイズ ロングワード</p>																						
<p>●説明</p> <p>32ビットレジスタERdの内容 (デスティネーションオペランド) のビット群を、左方向へ1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット0には0が格納されます。</p> <div style="text-align: center;"> </div>																						
<p>●使用可能な汎用レジスタ ERd : ER0~ER7</p>																						
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHLL.L</td> <td>ERd</td> <td>1</td> <td>0</td> <td>3</td> <td>0:erd</td> <td>2</td> </tr> </tbody> </table>			アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHLL.L	ERd	1	0	3	0:erd	2
アドレッシングモード	ニーモニック	オペランド形式				インストラクションフォーマット					実行ステート数											
			第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	SHLL.L	ERd	1	0	3	0:erd	2															
<p>●注意事項</p> <p>本命令とSHAL命令とでは、オーバーフローフラグの動作が異なります。</p>																						

2.2.56(1) SHLR(B)

SHLR(SHift Logical Right)		論理シフト																	
●オペレーション Rd (右論理シフト) →Rd	●コンディションコード																		
●アセンブラフォーマット SHLR.B Rd	<table border="1"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>0</td><td>↓</td><td>0</td><td>↓</td> </tr> </table>			I	UI	H	U	N	Z	V	C	—	—	—	—	0	↓	0	↓
I	UI	H	U	N	Z	V	C												
—	—	—	—	0	↓	0	↓												
●オペランドサイズ バイト	H: 実行前の値が保持されます。 N: 常に0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。																		
●説明																			
8ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、右方向へ1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット7には0が格納されます。																			
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H																			
●オペランド形式と実行ステート数																			
アドレッシング モード	ニーモ ニック	オペランド 形式	<table border="1"> <thead> <tr> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SHLR.B</td> <td>Rd</td> <td>1 1 0 rd</td> <td>2</td> </tr> </tbody> </table>	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SHLR.B	Rd	1 1 0 rd	2		
インストラクションフォーマット				実行 ステート 数															
第1バイト	第2バイト	第3バイト	第4バイト																
レジスタ直接	SHLR.B	Rd	1 1 0 rd	2															
●注意事項																			

2.2.56(2) SHLR(W)

SHLR(SHift Logical Right)		論理シフト						
●オペレーション Rd (右論理シフト) →Rd	●コンディションコード I UI H U N Z V C — — — — 0 † 0 †							
●アセンブラフォーマット SHLR.W Rd	H: 実行前の値が保持されます。 N: 常に0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。							
●オペランドサイズ ワード								
●説明 16ビットレジスタRdの内容 (デスティネーションオペランド) のビット群を、右方向へ1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット15には0が格納されず。								
●使用可能な汎用レジスタ Rd : R0~R7、E0~E7								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	
レジスタ直接	SHLR.W	Rd	1	1	1	rd		2
●注意事項								

2.2.56(3) SHLR(L)

SHLR(SHift Logical Right)		論理シフト		
●オペレーション ERd (右論理シフト) →ERd	●コンディションコード I UI H U N Z V C — — — — 0 † 0 †			
●アセンブラフォーマット SHLR.L ERd	H: 実行前の値が保持されます。 N: 常に0にクリアされます。 Z: 実行結果が0 (ゼロ) のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前のビット0の値が格納されます。			
●オペランドサイズ ロングワード				
●説明 32ビットレジスタERdの内容 (デスティネーションオペランド) のビット群を、右方向へ1ビットシフトします。シフトアウトしたビットはキャリフラグに格納され、ビット31には0が格納されます。				
●使用可能な汎用レジスタ ERd : ER0~ER7				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット	実行 ステート 数
レジスタ直接	SHLR.L	ERd	第1バイト: 1 1 第2バイト: 3 0 第3バイト: erd 第4バイト:	
●注意事項				

2. 各命令の説明

2.2.57 SLEEP

SLEEP(SLEEP)		低消費電力状態命令						
●オペレーション プログラム実行状態→低消費電力状態	●コンディションコード I U I H U N Z V C — — — — — — — —							
●アセンブラフォーマット SLEEP	H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。							
●オペランドサイズ —								
●説明 SLEEP命令を実行すると、CPUは低消費電力状態に入ります。低消費電力状態では、CPUの内部状態は保持され、命令の実行を停止し、例外処理要求の発生を待ち続けます。例外処理要求が発生すると、低消費電力状態は解除され、CPUは例外処理を開始します。このときNMI以外の割り込みでは、CPU側で割り込みがマスクされている場合、低消費電力状態は解除されません。								
●使用可能な汎用レジスタ —								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	
			第1バイト	第2バイト	第3バイト	第4バイト		
—	SLEEP		0	1	8	0		2
●注意事項 低消費電力状態については、当該LSIのハードウェアマニュアルを参照してください。								

2.2.58(1) STC(B)

STC(STore from Control register)		CCR転送		
●オペレーション CCR→Rd	●コンディションコード I UI H U N Z V C — — — — — — — —			
●アセンブラフォーマット STC.B CCR, Rd	H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。			
●オペランドサイズ バイト				
●説明 CCRの内容を8ビットレジスタRdに転送します。				
●使用可能な汎用レジスタ Rd:R0L~R7L, R0H~R7H				
●オペランド形式と実行ステート数				
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット 第1バイト 第2バイト 第3バイト 第4バイト	実行 ステート 数
レジスタ直接	STC.B	CCR,Rd	0 2 0 rd	2
●注意事項				

2. 各命令の説明

2.2.58(2) STC(W)

STC(STore from Control register)		CCR転送
●オペレーション CCR→EAd	●コンディションコード I UI H U N Z V C — — — — — — — —	
●アセンブラフォーマット STC.W CCR, EAd	H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。	
●オペランドサイズ ワード		
●説明 CCRの内容をデスティネーションのロケーションに転送します。CCRはバイトサイズですが転送はワードサイズで行われ、偶数アドレスにCCRの内容が格納されます。		
●使用可能な汎用レジスタ ERd: ER0~ER7		

2.2.58(2) STC(W)

STC (STore from Control register)		CCR転送															
アドレッシング モード	ニーモ ニック	オペランド 形式	インスタレーションフォーマット										実行 ステート 数				
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト					
レジスタ間接	STC.W	CCR,@ERd	0	1	4	0	6	9	1	erd	0						6
ディスプレ- ースメント付 レジスタ間接	STC.W	CCR,@(16,ERd)	0	1	4	0	6	F	1	erd	0	disp					8
ポストアイン クリメン ツァーレジスタ間接	STC.W	CCR,@(24,ERd)	0	1	4	0	7	8	0	erd	0	6	A	0	0	disp	12
絶対アドレス	STC.W	CCR,@ERd	0	1	4	0	6	D	1	erd	0						8
	STC.W	CCR,@aar.16	0	1	4	0	6	B	8	0	abs						8
	STC.W	CCR,@aar.24	0	1	4	0	6	B	A	0	0	0	abs				10

●注意事項

2.2.59(2) SUB(W)

SUB(SUBtract binary)		2進減算						
●オペレーション Rd ← (EAs) → Rd	●コンディションコード							
	I U I H U N Z V C ─ ─ † ─ † † † †							
●アセンブラフォーマット SUB.W <EAs>, Rd	H: ビット11にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。 C: ビット15にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。							
●オペランドサイズ ワード								
●説明 16ビットレジスタRdの内容（デスティネーションオペランド）からソースオペランドを減算し、結果を16ビットレジスタRdに格納します。								
●使用可能な汎用レジスタ Rd : R0~R7, E0~E7 Rs : R0~R7, E0~E7								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	
			第1バイト	第2バイト	第3バイト	第4バイト		
イミディエイト	SUB.W	#xx:16,Rd	7	9	3	rd	IMM	4
レジスタ直接	SUB.W	Rs,Rd	1	9	rs	rd		2
●注意事項								

2. 各命令の説明

2.2.59(3) SUB(L)

SUB(SUBtract binary)		2進減算																						
●オペレーション ERd←(EAs)→ERd	●コンディションコード																							
	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>↑</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table>			I	UI	H	U	N	Z	V	C	—	—	↑	—	↑	↑	↑	↑					
I	UI	H	U	N	Z	V	C																	
—	—	↑	—	↑	↑	↑	↑																	
●アセンブラフォーマット SUB.L <EAs>, ERd	<p>H: ビット27にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>C: ビット31にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p>																							
●オペランドサイズ ロングワード																								
●説明 32ビットレジスタERdの内容（デスティネーションオペランド）からソースオペランドを減算し、結果を32ビットレジスタERdに格納します。																								
●使用可能な汎用レジスタ ERd : ER0~ER7 ERs : ER0~ER7																								
●オペランド形式と実行ステート数																								
アドレッシングモード	ニーモニック	オペランド形式	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>SUB.L</td> <td>#xx:32,ERd</td> <td>7 A 3 0 erd</td> <td>IMM</td> <td>6</td> </tr> <tr> <td>レジスタ直接</td> <td>SUB.L</td> <td>ERs,ERd</td> <td>1 A 1 ers 0 erd</td> <td></td> <td>2</td> </tr> </tbody> </table>	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	イミディエイト	SUB.L	#xx:32,ERd	7 A 3 0 erd	IMM	6	レジスタ直接	SUB.L	ERs,ERd	1 A 1 ers 0 erd		2
インストラクションフォーマット				実行ステート数																				
第1バイト	第2バイト	第3バイト	第4バイト																					
イミディエイト	SUB.L	#xx:32,ERd	7 A 3 0 erd	IMM	6																			
レジスタ直接	SUB.L	ERs,ERd	1 A 1 ers 0 erd		2																			
●注意事項																								

2.2.60 SUBS

SUBS(SUBtract with Sign extention)		アドレスデータ2進減算																																								
<p>●オペレーション</p> <p>ERd-1→ERd ERd-2→ERd ERd-4→ERd</p>		<p>●コンディションコード</p> <p>I U I H U N Z V C</p> <table border="1"> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行前の値が保持されます。 Z: 実行前の値が保持されます。 V: 実行前の値が保持されます。 C: 実行前の値が保持されます。</p>		—	—	—	—	—	—	—	—																															
—	—	—	—	—	—	—	—																																			
<p>●アセンブラフォーマット</p> <p>SUBS #1, ERd SUBS #2, ERd SUBS #4, ERd</p>																																										
<p>●オペランドサイズ</p> <p>ロングワード</p>																																										
<p>●説明</p> <p>32ビットレジスタERdの内容（デスティネーションオペランド）から1、2または4を減算します。SUB命令とは異なり、コンディションコードは実行前の値を保持します。</p>																																										
<p>●使用可能な汎用レジスタ</p> <p>ERd : ER0~ER7</p>																																										
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシングモード</th> <th rowspan="2">ニーモニック</th> <th rowspan="2">オペランド形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行ステート数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>SUBS</td> <td>#1,ERd</td> <td>1</td> <td>B</td> <td>0</td> <td>0:erd</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>SUBS</td> <td>#2,ERd</td> <td>1</td> <td>B</td> <td>8</td> <td>0:erd</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>SUBS</td> <td>#4,ERd</td> <td>1</td> <td>B</td> <td>9</td> <td>0:erd</td> <td></td> <td>2</td> </tr> </tbody> </table>				アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	SUBS	#1,ERd	1	B	0	0:erd		2	レジスタ直接	SUBS	#2,ERd	1	B	8	0:erd		2	レジスタ直接	SUBS	#4,ERd	1	B	9	0:erd		2
アドレッシングモード	ニーモニック	オペランド形式	インストラクションフォーマット				実行ステート数																																			
			第1バイト	第2バイト	第3バイト	第4バイト																																				
レジスタ直接	SUBS	#1,ERd	1	B	0	0:erd		2																																		
レジスタ直接	SUBS	#2,ERd	1	B	8	0:erd		2																																		
レジスタ直接	SUBS	#4,ERd	1	B	9	0:erd		2																																		
<p>●注意事項</p>																																										

2. 各命令の説明

2.2.61 SUBX

SUBX(SUBtract with eXtend carry)		キャリ付減算																	
●オペレーション Rd-(EAs)-C→Rd	●コンディションコード																		
	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>↑</td><td>—</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table>			I	UI	H	U	N	Z	V	C	—	—	↑	—	↑	↑	↑	↑
I	UI	H	U	N	Z	V	C												
—	—	↑	—	↑	↑	↑	↑												
●アセンブラフォーマット SUBX <EAs>, Rd	<p>H: ビット3にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>Z: 実行結果が0（ゼロ）のとき実行前の値が保持され、それ以外の場合は0にクリアされます。</p> <p>V: オーバフローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p> <p>C: ビット7にボローが発生したとき1にセットされ、それ以外の場合は0にクリアされます。</p>																		
●オペランドサイズ バイト																			
●説明 8ビットレジスタRdの内容（デスティネーションオペランド）からソースオペランドとキャリフラグの値を減算し、結果を8ビットレジスタRdに格納します。																			
●使用可能な汎用レジスタ Rd : R0L~R7L、R0H~R7H																			
●オペランド形式と実行ステート数																			
アドレッシング モード	ニーモ ニック	オペランド 形式	<table border="1" style="width: 100%; text-align: center;"> <tr> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </table>	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト							
インストラクションフォーマット				実行 ステート 数															
第1バイト	第2バイト	第3バイト	第4バイト																
イミディエイト	SUBX	#xx:8,Rd	B rd IMM	2															
レジスタ直接	SUBX	Rs,Rd	1 E rs rd	2															
●注意事項																			

2.2.62 TRAPA

TRAPA(TRAPA Always)		無条件トラップ																					
<p>●オペレーション PC→@-SP CCR→@-SP <ベクタ>→PC</p>	<p>●コンディションコード</p> <table border="1"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>1</td> <td>△*</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> </tr> </table> <p>I : 常に1にセットされます。 UI : 注意事項を参照してください。 H : 実行前の値が保持されます。 N : 演算前の値が保持されます。 Z : 演算前の値が保持されます。 V : 演算前の値が保持されます。 C : 演算前の値が保持されます。</p>			I	UI	H	U	N	Z	V	C	1	△*	-	-	-	-	-	-				
I	UI	H	U	N	Z	V	C																
1	△*	-	-	-	-	-	-																
<p>●アセンブラフォーマット TRAPA #x:2</p>																							
<p>●オペランドサイズ</p>																							
<p>●説明</p> <p>プログラムカウンタ (PC) とコンディションコードレジスタ (CCR) をスタックに退避し、Iビットを1にセットします。次に指定した番号に対応するベクタアドレスの内容によって示されるアドレスへ分岐します。</p> <p>退避するPCの値は本命令の直後の命令の先頭アドレスになります。</p> <table border="1"> <thead> <tr> <th rowspan="2">#x</th> <th colspan="2">ベクタアドレス</th> </tr> <tr> <th>ノーマルモード</th> <th>アドバンスモード</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>H'0010~H'0011</td> <td>H'000020~H'000023</td> </tr> <tr> <td>1</td> <td>H'0012~H'0013</td> <td>H'000024~H'000027</td> </tr> <tr> <td>2</td> <td>H'0014~H'0015</td> <td>H'000028~H'00002B</td> </tr> <tr> <td>3</td> <td>H'0016~H'0017</td> <td>H'00002C~H'00002F</td> </tr> </tbody> </table>				#x	ベクタアドレス		ノーマルモード	アドバンスモード	0	H'0010~H'0011	H'000020~H'000023	1	H'0012~H'0013	H'000024~H'000027	2	H'0014~H'0015	H'000028~H'00002B	3	H'0016~H'0017	H'00002C~H'00002F			
#x	ベクタアドレス																						
	ノーマルモード	アドバンスモード																					
0	H'0010~H'0011	H'000020~H'000023																					
1	H'0012~H'0013	H'000024~H'000027																					
2	H'0014~H'0015	H'000028~H'00002B																					
3	H'0016~H'0017	H'00002C~H'00002F																					
<p>●オペランド形式と実行ステート数</p> <table border="1"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>レジスタ直接</td> <td>TRAPA</td> <td>#x:2</td> <td>5</td> <td>7</td> <td>00IMM</td> <td>0</td> <td>14</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	レジスタ直接	TRAPA	#x:2	5	7	00IMM	0	14
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																
			第1バイト	第2バイト	第3バイト	第4バイト																	
レジスタ直接	TRAPA	#x:2	5	7	00IMM	0	14																
<p>●注意事項</p> <p>*割り込みマスクビットとして使用しているとき1にセットされます。ユーザビットとして使用しているとき実行前の値が保持されます。詳細は、LSIのハードウェアマニュアルを参照してください。</p> <p>ノーマルモードとアドバンスモードではスタックおよびベクタの構造が異なりますので注意してください。</p>																							

2. 各命令の説明

2.2.63(1) XOR(B)

XOR(eXclusive OR logical)		排他的論理和																													
<p>●オペレーション Rd ⊕ (EAs) → Rd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td> <td>UI</td> <td>H</td> <td>U</td> <td>N</td> <td>Z</td> <td>V</td> <td>C</td> </tr> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>↑</td> <td>↑</td> <td>0</td> <td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	—												
I	UI	H	U	N	Z	V	C																								
—	—	—	—	↑	↑	0	—																								
<p>●アセンブラフォーマット XOR.B <EAs>, Rd</p>																															
<p>●オペランドサイズ バイト</p>																															
<p>●説明 8ビットレジスタRdの内容（デスティネーションオペランド）と、ソースオペランドの排他的論理和をとり、結果を8ビットレジスタRdに格納します。</p>																															
<p>●使用可能な汎用レジスタ Rd : R0L~R7L, R0H~R7H Rs : R0L~R7L, R0H~R7H</p>																															
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>XOR.B</td> <td>#xx:8,Rd</td> <td>D</td> <td>rd</td> <td>IMM</td> <td></td> <td>2</td> </tr> <tr> <td>レジスタ直接</td> <td>XOR.B</td> <td>Rs,Rd</td> <td>1</td> <td>5</td> <td>rs</td> <td>rd</td> <td>2</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	イミディエイト	XOR.B	#xx:8,Rd	D	rd	IMM		2	レジスタ直接	XOR.B	Rs,Rd	1	5	rs	rd	2
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数																								
			第1バイト	第2バイト	第3バイト	第4バイト																									
イミディエイト	XOR.B	#xx:8,Rd	D	rd	IMM		2																								
レジスタ直接	XOR.B	Rs,Rd	1	5	rs	rd	2																								
<p>●注意事項</p>																															

2.2.63(2) XOR(W)

XOR(eXclusive OR logical)		排他的論理和						
●オペレーション Rd ⊕ (EAs) → Rd	●コンディションコード I UI H U N Z V C — — — — † † 0 —							
●アセンブラフォーマット XOR.W <EAs>, Rd	H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。							
●オペランドサイズ ワード								
●説明 16ビットレジスタRdの内容（デスティネーションオペランド）と、ソースオペランドの排他的論理和をとり、結果を16ビットレジスタRdに格納します。								
●使用可能な汎用レジスタ Rd : R0~R7, E0~E7 Rs : R0~R7, E0~E7								
●オペランド形式と実行ステート数								
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット				実行 ステート 数	
			第1バイト	第2バイト	第3バイト	第4バイト		
イミディエイト	XOR.W	#xx:16,Rd	7	9	5	rd	IMM	4
レジスタ直接	XOR.W	Rs,Rd	6	5	rs	rd		2
●注意事項								

2. 各命令の説明

2.2.63(3) XOR(L)

XOR(eXclusive OR logical)		排他的論理和																																							
<p>●オペレーション ERd ⊕ (EAs) →ERd</p>	<p>●コンディションコード</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>—</td><td>—</td><td>—</td><td>—</td><td>↑</td><td>↑</td><td>0</td><td>—</td> </tr> </table> <p>H: 実行前の値が保持されます。 N: 実行結果が負のとき1にセットされ、それ以外の場合は0にクリアされます。 Z: 実行結果が0（ゼロ）のとき1にセットされ、それ以外の場合は0にクリアされます。 V: 常に0にクリアされます。 C: 実行前の値が保持されます。</p>			I	UI	H	U	N	Z	V	C	—	—	—	—	↑	↑	0	—																						
I	UI	H	U	N	Z	V	C																																		
—	—	—	—	↑	↑	0	—																																		
<p>●アセンブラフォーマット XOR.L <EAs>, ERd</p>																																									
<p>●オペランドサイズ ロングワード</p>																																									
<p>●説明</p> <p>32ビットレジスタERdの内容（デスティネーションオペランド）と、ソースオペランドの排他的論理和をとり、結果を32ビットレジスタERdに格納します。</p>																																									
<p>●使用可能な汎用レジスタ</p> <p>ERd : ER0~ER7 ERs : ER0~ER7</p>																																									
<p>●オペランド形式と実行ステート数</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th rowspan="2">アドレッシング モード</th> <th rowspan="2">ニーモ ニック</th> <th rowspan="2">オペランド 形式</th> <th colspan="6">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> <th>第5バイト</th> <th>第6バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>XOR.L</td> <td>#xx:32,ERd</td> <td>7</td> <td>A</td> <td>5</td> <td>0:erd</td> <td colspan="3">IMM</td> <td>6</td> </tr> <tr> <td>レジスタ直接</td> <td>XOR.L</td> <td>ERs,ERd</td> <td>0</td> <td>1</td> <td>F</td> <td>0</td> <td>6</td> <td>5</td> <td>0:ers 0:erd</td> <td>4</td> </tr> </tbody> </table>				アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット						実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	イミディエイト	XOR.L	#xx:32,ERd	7	A	5	0:erd	IMM			6	レジスタ直接	XOR.L	ERs,ERd	0	1	F	0	6	5	0:ers 0:erd	4
アドレッシング モード	ニーモ ニック	オペランド 形式	インストラクションフォーマット						実行 ステート 数																																
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト																																	
イミディエイト	XOR.L	#xx:32,ERd	7	A	5	0:erd	IMM			6																															
レジスタ直接	XOR.L	ERs,ERd	0	1	F	0	6	5	0:ers 0:erd	4																															
<p>●注意事項</p>																																									

2.2.64 XORC

XORC(eXclusive OR Control register)		CCRとの排他的論理和																			
●オペレーション CCR ⊕ #IMM → CCR	●コンディションコード																				
	<table style="margin-left: auto; margin-right: auto;"> <tr> <td>I</td><td>UI</td><td>H</td><td>U</td><td>N</td><td>Z</td><td>V</td><td>C</td> </tr> <tr> <td>↑</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td><td>↑</td> </tr> </table>			I	UI	H	U	N	Z	V	C	↑	↑	↑	↑	↑	↑	↑	↑		
I	UI	H	U	N	Z	V	C														
↑	↑	↑	↑	↑	↑	↑	↑														
●アセンブラフォーマット XOR.C #xx:8, CCR	<p>I : 実行結果の対応するビットの値が格納されます。</p> <p>UI : 実行結果の対応するビットの値が格納されます。</p> <p>H : 実行結果の対応するビットの値が格納されます。</p> <p>U : 実行結果の対応するビットの値が格納されます。</p> <p>N : 実行結果の対応するビットの値が格納されます。</p> <p>Z : 実行結果の対応するビットの値が格納されます。</p> <p>V : 実行結果の対応するビットの値が格納されます。</p> <p>C : 実行結果の対応するビットの値が格納されます。</p>																				
●オペランドサイズ バイト																					
●説明 CCRの内容とイミディエイトデータとの排他的論理和をとり、結果をCCRに格納します。 本命令の実行終了時点では、NMIを含めてすべての割り込みは受け付けられません。																					
●オペランド形式と実行ステート数																					
アドレッシング モード	ニーモ ニック	オペランド 形式	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th colspan="4">インストラクションフォーマット</th> <th rowspan="2">実行 ステート 数</th> </tr> <tr> <th>第1バイト</th> <th>第2バイト</th> <th>第3バイト</th> <th>第4バイト</th> </tr> </thead> <tbody> <tr> <td>イミディエイト</td> <td>XORC</td> <td>#xx:8,CCR</td> <td>0 5</td> <td>IMM</td> <td></td> <td></td> <td></td> <td>2</td> </tr> </tbody> </table>	インストラクションフォーマット				実行 ステート 数	第1バイト	第2バイト	第3バイト	第4バイト	イミディエイト	XORC	#xx:8,CCR	0 5	IMM				2
インストラクションフォーマット				実行 ステート 数																	
第1バイト	第2バイト	第3バイト	第4バイト																		
イミディエイト	XORC	#xx:8,CCR	0 5	IMM				2													
●注意事項																					

2. 各命令の説明

2.3 命令セット一覧

2.3.1 命令とアドレッシングモードの組合せ

表 2.1 命令セットの概要

機能	命令	アドレッシングモード												
		#xx	Rn	@ERn	@(d:16,ERn)	@(d:24,ERn)	@ERn+/-ERn	@aa:8	@aa:16	@aa:24	@(d:8,PC)	@(d:16,PC)	@@aa:8	—
データ転送命令	MOV	BWL	BWL	BWL	BWL	BWL	BWL	B	BWL	BWL	—	—	—	—
	POP, PUSH	—	—	—	—	—	—	—	—	—	—	—	—	WL
	MOVFP, MOVTP	—	—	—	—	—	—	—	B	—	—	—	—	—
算術演算命令	ADD, CMP	BWL	BWL	—	—	—	—	—	—	—	—	—	—	—
	SUB	WL	BWL	—	—	—	—	—	—	—	—	—	—	—
	ADDX, SUBX	B	B	—	—	—	—	—	—	—	—	—	—	—
	ADDS, SUBS	—	L	—	—	—	—	—	—	—	—	—	—	—
	INC, DEC	—	BWL	—	—	—	—	—	—	—	—	—	—	—
	DAA, DAS	—	B	—	—	—	—	—	—	—	—	—	—	—
	MULXU, MULXS, DIVXU, DIVXS	—	BW	—	—	—	—	—	—	—	—	—	—	—
	NEG	—	BWL	—	—	—	—	—	—	—	—	—	—	—
	EXTU, EXT8	—	WL	—	—	—	—	—	—	—	—	—	—	—
論理演算命令	AND, OR, XOR	BWL	BWL	—	—	—	—	—	—	—	—	—	—	—
	NOT	—	BWL	—	—	—	—	—	—	—	—	—	—	—
シフト命令	—	BWL	—	—	—	—	—	—	—	—	—	—	—	
ビット操作命令	—	B	B	—	—	—	—	B	—	—	—	—	—	
分岐命令	Bcc, BSR	—	—	—	—	—	—	—	—	—	○	○	—	—
	JMP, JSR	—	—	○	—	—	—	—	—	○	—	—	○	—
RTS	—	—	—	—	—	—	—	—	—	—	—	—	○	
システム制御命令	TRAPA, RTE	—	—	—	—	—	—	—	—	—	—	—	—	○
	SLEEP	—	—	—	—	—	—	—	—	—	—	—	—	○
	LDC	B	B	W	W	W	W	—	W	W	—	—	—	—
	STC	—	B	W	W	W	W	—	W	W	—	—	—	—
	ANDC, ORC, XORC	B	—	—	—	—	—	—	—	—	—	—	—	—
	NOP	—	—	—	—	—	—	—	—	—	—	—	—	○
ブロック転送命令	—	—	—	—	—	—	—	—	—	—	—	—	BW	

《記号説明》

B:バイト

W:ワード

L:ロングワード

2.3.2 命令セット一覧

表2.2 命令セット一覧

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)			オペレーション	コンディションコード						実行回数*1
		Rn	@(d, ERn)	@ERn/@ERn+ @aa		I	H	N	Z	V	C	
MOV	B	#xx	@ERn	@(d, ERn)	#xx8→Rd8	—	—	↑	↑	0	—	2
	B	2			Rs8→Rd8	—	—	↑	↑	0	—	2
	B				@ERs→Rd8	—	—	↑	↑	0	—	4
	B	2			@(d:16, ERs)→Rd8	—	—	↑	↑	0	—	6
	B	8			@(d:24, ERs)→Rd8	—	—	↑	↑	0	—	10
	B		2		@ERs→Rd8, ERs32+1→ERs32	—	—	↑	↑	0	—	6
	B			2	@aa:8→Rd8	—	—	↑	↑	0	—	4
	B			4	@aa:16→Rd8	—	—	↑	↑	0	—	6
	B			6	@aa:24→Rd8	—	—	↑	↑	0	—	8
	B	2			Rs8→@ERd	—	—	↑	↑	0	—	4
	B	4			Rs8→@(d:16, ERd)	—	—	↑	↑	0	—	6
	B	8			Rs8→@(d:24, ERd)	—	—	↑	↑	0	—	10
	B		2		ERd32-1→ERd32, Rs8→@ERd	—	—	↑	↑	0	—	6
	B			2	Rs8→@aa:8	—	—	↑	↑	0	—	4
	B			4	Rs8→@aa:16	—	—	↑	↑	0	—	6
	B			6	Rs8→@aa:24	—	—	↑	↑	0	—	8
	W	4			#xx:16→Rd16	—	—	↑	↑	0	—	4
	W				Rs16→Rd16	—	—	↑	↑	0	—	2
	W	2			@ERs→Rd16	—	—	↑	↑	0	—	4
	W		4		@(d:16, ERs)→Rd16	—	—	↑	↑	0	—	6
	W		8		@(d:24, ERs)→Rd16	—	—	↑	↑	0	—	10
	W		2		@ERs→Rd16, ERs32+2→@ERd32	—	—	↑	↑	0	—	6
	W			4	@aa:16→Rd16	—	—	↑	↑	0	—	6
	W			6	@aa:24→Rd16	—	—	↑	↑	0	—	8

2. 各命令の説明

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)				オペレーション	コンディションコード					実行回数*1				
		#xx	Rn	@(d, ERn)	@-ERn/@ERn+/@aa		@(d, PC)	@aa	I	H	N		Z	V	C	
MOV	MOV.W Rs, @ERd		2													4
	MOV.W Rs, @(d:16, ERd)			4												6
	MOV.W Rs, @(d:24, ERd)			8												10
	MOV.W Rs, @-ERd				2											6
	MOV.W Rs, @aa:16					4										6
	MOV.W Rs, @aa:24						6									8
	MOV.L #xx:32, Rd	L	6													6
	MOV.L @ERS, ERd	L	2	4												2
	MOV.L @ERS, ERd	L														8
	MOV.L @(d:16, ERs), ERd	L			6											10
	MOV.L @(d:24, ERs), ERd	L			10											14
	MOV.L @ERS+, ERd	L				4										10
	MOV.L @aa:16, ERd	L					6									10
	MOV.L @aa:24, ERd	L						8								12
MOV.L ERS, @ERd	L		4												8	
MOV.L ERS, @(d:16, ERd)	L			6											10	
MOV.L ERS, @(d:24, ERd)	L			10											14	
MOV.L ERS, @-ERd	L				4										10	
MOV.L ERS, @aa:16	L					6									10	
MOV.L ERS, @aa:24	L						8								12	
POP.W Rn	W														6	
POP.L ERn	L														10	
PUSH.W Rn	W														6	
PUSH.L ERn	L														10	
MOVFP E @aa:16, Rd	B					4									(6)	
MOVTF E @aa:16, Rd	B					4									(6)	

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)						オペレーション	コンディションコード							実行サイクル数*1	
		#xx	Rn	@ERn	@(d, ERn)	@-ERn/@ERn+	@aa		@(d, PC)	@aa	I	H	N	Z	V		C
ADD	ADD.B #xx:8, Rd	B	2												↑	↑	2
	ADD.B Rs, Rd	B	2												↑	↑	2
	ADD.W #xx:16, Rd	W	4												↑	↑	4
	ADD.W Rs, Rd	W	2												↑	↑	2
	ADD.L #xx:32, ERd	L	6												↑	↑	6
ADDX	ADD.L ERs, ERd	L	2												↑	↑	2
	ADDX.B #xx:8, Rd	B	2											↑	↑	2	
	ADDX.B Rs, Rd	B	2											(3)	↑	2	
	ADDS.L #1, ERd	L	2											(3)	↑	2	
	ADDS.L #2, ERd	L	2											—	—	2	
INC	ADDS.L #4, ERd	L	2											—	—	2	
	INC.B Rd	B	2											↑	↑	2	
	INC.W #1, Rd	W	2											↑	↑	2	
	INC.W #2, Rd	W	2											↑	↑	2	
	INCL #1, ERd	L	2											↑	↑	2	
DAA	INCL #2, ERd	L	2											↑	↑	2	
	DAA Rd	B	2										*	↑	*	2	
	SUB.B Rs, Rd	B	2											↑	↑	2	
	SUB.W #xx:16, Rd	W	4											↑	↑	4	
	SUB.W Rs, Rd	W	2											(1)	↑	2	
SUBX	SUB.L #xx:32, ERd	L	6											(2)	↑	6	
	SUB.L ERs, ERd	L	2											(2)	↑	2	
	SUBX #xx:8, Rd	B	2											↑	(3)	2	
	SUBX Rs, Rd	B	2											↑	(3)	2	
														↑	(3)	2	

2. 各命令の説明

二乗ノック	サイズ	アドレッシングモード/命令長 (バイト)						オペレーション		コンディションコード							実行回数 *1							
		#xx	Rn	@ERn	@(d, ERn)	@-ERn/@ERn+	@aa			@(d, PC)	@aa	I	H	N	Z	V		C	J	7	10	17	18	19
SUBS	SUBS #1, ERd	L	2																				2	
	SUBS #2, ERd	L	2																				2	
	SUBS #4, ERd	L	2																				2	
	DEC.B Rd	B	2																				2	
DEC	DEC.W #1, Rd	W	2																				2	
	DEC.W #2, Rd	W	2																				2	
	DEC.L #1, ERd	L	2																				2	
	DEC.L #2, ERd	L	2																				2	
DAS	DAS Rd	B	2																*	†	†	*	2	
	MULXU.B Rs, Rd	B	2																				14	
MULXU	MULXU.W Rs, ERd	W	2																				22	
	MULXS.B Rs, Rd	B	4																	†	†		16	
	MULXS.W Rs, ERd	W	4																	†	†		24	
	DIVXU.B Rs, Rd	B	2																	(6)	(7)		14	
DIVXU	DIVXU.W Rs, ERd	W	2																	(6)	(7)		22	
	DIVXS.B Rs, Rd	B	4																	(8)	(7)		16	
	DIVXS.W Rs, ERd	W	4																	(8)	(7)		24	
	DIVXS.L Rs, ERd	L	6																	(8)	(7)		24	
CMP	CMP.B #xx:8, Rd	B	2																	†	†	†	†	2
	CMP.B Rs, Rd	B	2																	†	†	†	†	2
	CMP.W #xx:16, Rd	W	4																	(1)	†	†	†	4
	CMP.W Rs, Rd	W	2																	(1)	†	†	†	2
	CMP.L #xx:32, ERd	L	6																	(2)	†	†	†	6
	CMP.L ERs, ERd	L	2																	(2)	†	†	†	2
NEG	NEG.B Rd	B	2																	†	†	†	†	2
	NEG.W Rd	W	2																	†	†	†	†	2
	NEG.L ERd	L	2																	†	†	†	†	2

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)				オペレーション	コンディションコード				実行サイクル数*1		
		#xx	Rn	@(d, ERn)	@-ERn/@ERn+ @aa @-(d, PC) @aa		I	H	Z	V		C	
EXTU	EXTU.W Rd	W	2			0→(<ビット15~8>of Rd16)	—	—	0	↑	0	—	2
	EXTU.L ERd	L	2			0→(<ビット31~16>of ERd32)	—	—	0	↑	0	—	2
EXTS	EXTS.W Rd	W	2			(<ビット7>of Rd16)	—	—	↑	↑	0	—	2
	EXTS.L ERd	L	2			(<ビット15>of ERd32) →(<ビット31~16>of ERd32)	—	—	↑	↑	0	—	2

(3) 論理演算命令

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)				オペレーション	コンディションコード				実行サイクル数*1		
		#xx	Rn	@(d, ERn)	@-ERn/@ERn+ @aa @-(d, PC) @aa		I	H	Z	V		C	
AND	AND.B #xx:8,Rd	B	2			Rd8 ∧ #xx:8 → Rd8	—	—	↑	↑	0	—	2
	AND.B Rs,Rd	B	2			Rd8 ∧ Rs8 → Rd8	—	—	↑	↑	0	—	2
	AND.W #xx:16,Rd	W	4			Rd16 ∧ #xx:16 → Rd16	—	—	↑	↑	0	—	4
	AND.W Rs,Rd	W	2			Rd16 ∧ Rs16 → Rd16	—	—	↑	↑	0	—	2
	AND.L #xx:32,ERd	L	6			ERd32 ∧ #xx:32 → ERd32	—	—	↑	↑	0	—	6
	AND.L ERs,ERd	L	4			ERd32 ∧ ERs32 → ERd32	—	—	↑	↑	0	—	4
OR	OR.B #xx:8,Rd	B	2			Rd8 ∨ #xx:8 → Rd8	—	—	↑	↑	0	—	2
	OR.B Rs,Rd	B	2			Rd8 ∨ Rs8 → Rd8	—	—	↑	↑	0	—	2
	OR.W #xx:16,Rd	W	4			Rd16 ∨ #xx:16 → Rd16	—	—	↑	↑	0	—	4
	OR.W Rs,Rd	W	2			Rd16 ∨ Rs16 → Rd16	—	—	↑	↑	0	—	2
	OR.L #xx:32,ERd	L	6			ERd32 ∨ #xx:32 → ERd32	—	—	↑	↑	0	—	6
	OR.L ERs,ERd	L	4			ERd32 ∨ ERs32 → ERd32	—	—	↑	↑	0	—	4
XOR	XOR.B #xx:8,Rd	B	2			Rd8 ⊕ #xx:8 → Rd8	—	—	↑	↑	0	—	2
	XOR.B Rs,Rd	B	2			Rd8 ⊕ Rs8 → Rd8	—	—	↑	↑	0	—	2
	XOR.W #xx:16,Rd	W	4			Rd16 ⊕ #xx:16 → Rd16	—	—	↑	↑	0	—	4
	XOR.W Rs,Rd	W	2			Rd16 ⊕ Rs16 → Rd16	—	—	↑	↑	0	—	2
	XOR.L #xx:32,ERd	L	6			ERd32 ∨ #xx:32 → ERd32	—	—	↑	↑	0	—	6
	XOR.L ERs,ERd	L	4			ERd32 ∨ ERs32 → ERd32	—	—	↑	↑	0	—	4
NOT	NOT.B Rd	B	2			~Rd8 → Rd8	—	—	↑	↑	0	—	2
	NOT.W Rd	W	2			~Rd16 → Rd16	—	—	↑	↑	0	—	2
	NOT.L ERd	L	2			~Rd32 → Rd32	—	—	↑	↑	0	—	2

2. 各命令の説明

ニーモニック		サイズ	アドレッシングモード/命令長 (バイト)					オペレーション	コンディションコード							実行回数 #1		
			#xx	Rn	@ERn	@(d, ERn)	@-ERn@ERn+		@aa	@(d, PC)	@aa	I	H	N	Z	V	C	ノード
SHAL	SHAL.B Rd	B	2															2
	SHAL.W Rd	W	2															2
	SHAL.L ERd	L	2															2
SHAR	SHAR.B Rd	B	2															2
	SHAR.W Rd	W	2															2
	SHAR.L ERd	L	2															2
SHLL	SHLL.B Rd	B	2															2
	SHLL.W Rd	W	2															2
	SHLL.L ERd	L	2															2
SHLR	SHLR.B Rd	B	2															2
	SHLR.W Rd	W	2															2
	SHLR.L ERd	L	2															2
ROTXL	ROTXL.B Rd	B	2															2
	ROTXL.W Rd	W	2															2
	ROTXL.L ERd	L	2															2
ROTXR	ROTXR.B Rd	B	2															2
	ROTXR.W Rd	W	2															2
	ROTXR.L ERd	L	2															2
ROTL	ROTL.B Rd	B	2															2
	ROTL.W Rd	W	2															2
	ROTL.L ERd	L	2															2
ROTR	ROTR.B Rd	B	2															2
	ROTR.W Rd	W	2															2
	ROTR.L ERd	L	2															2

2. 各命令の説明

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)						オペレーション	コンディションコード						実行回数 ^{※1}	
		#xx	Rn	@ERn	@(d, ERn)	@-ERn/@ERn+	@aa		@(d, PC)	@aa	I	H	N	Z		V
BST	BST #xx:3, Rd	B	2													2
	BST #xx:3, @ERd	B		4												8
	BST #xx:3, @aa:8	B					4									8
BIST	BIST #xx:3, Rd	B	2													2
	BIST #xx:3, @ERd	B		4												8
	BIST #xx:3, @aa:8	B					4									8
BAND	BAND #xx:3, Rd	B	2													2
	BAND #xx:3, @ERd	B		4												6
	BAND #xx:3, @aa:8	B					4									6
BIAND	BIAND #xx:3, Rd	B	2													2
	BIAND #xx:3, @ERd	B		4												6
	BIAND #xx:3, @aa:8	B					4									6
BOR	BOR #xx:3, Rd	B	2													2
	BOR #xx:3, @ERd	B		4												6
	BOR #xx:3, @aa:8	B					4									6
BIOR	BIOR #xx:3, Rd	B	2													2
	BIOR #xx:3, @ERd	B		4												6
	BIOR #xx:3, @aa:8	B					4									6
BXOR	BXOR #xx:3, Rd	B	2													2
	BXOR #xx:3, @ERd	B		4												6
	BXOR #xx:3, @aa:8	B					4									6
BIXOR	BIXOR #xx:3, Rd	B	2													2
	BIXOR #xx:3, @ERd	B		4												6
	BIXOR #xx:3, @aa:8	B					4									6

2. 各命令の説明

(7) システム制御命令

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)				オペレーション	コンディションコード				実行対ト数 ^{*1}					
		#xx	Rn	@(d, ERn)	@-ERn/@ERn+		@aa	@(d, PC)	@@aa	—		I	H	N	Z	V
Bcc	BGE d:8					2										4
	BGE d:16					4										6
	BLT d:8					2										4
	BLT d:16					4										6
	BGT d:8					2										4
	BGT d:16					4										6
JMP	BLE d:8					2										4
	BLE d:16					4										6
	JMP @ERn		2													4
	JMP @aa:24				4											6
	JMP @@aa:8						2									8
	JMP @aa:8							2								10
BSR	BSR d:8					2										6
	BSR d:16					4										8
JSR	JSR @ERn					2										6
	JSR @aa:24				4											8
	JSR @@aa:8						2									10
RTS	RTS									2						8
	RTS										2					10

(7) システム制御命令

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)						オペレーション	コンディションコード						実行回数*1	
		#xx	Rn	@ERn	@(d, ERn)	@-ERn@ERn+	@aa		@(d, PC)	@aa	I	H	N	Z		V
TRAPA	TRAPA #x:2	—							2	PC→@-SP, CCR→@-SP, <'/'>→PC	↑	—	—	—	—	14
RTE	RTE	—								CCR←@SP+, PC←@SP+	↑	↑	↑	↑	↑	10
SLEEP	SLEEP	—								低消費電力状態に遷移	—	—	—	—	—	2
LDC	LDC #xx:8, CCR	B	2							#xx:8→CCR	↑	↑	↑	↑	↑	2
	LDC Rs, CCR	B	2							Rs8→CCR	↑	↑	↑	↑	↑	2
	LDC @ERs, CCR	W		4						@ERs→CCR	↑	↑	↑	↑	↑	6
	LDC @(d:16, ERs), CCR	W			6					@(d:16, ERs)→CCR	↑	↑	↑	↑	↑	8
	LDC @(d:24, ERs), CCR	W			10					@(d:24, ERs)→CCR	↑	↑	↑	↑	↑	12
	LDC @ERs+, CCR	W				4				@ERs→CCR, ERs32+2→ERs32	↑	↑	↑	↑	↑	8
	LDC @aa:16, CCR	W					6			@aa:16→CCR	↑	↑	↑	↑	↑	8
	LDC @aa:24, CCR	W						8		@aa:24→CCR	↑	↑	↑	↑	↑	10
STC	STC CCR, Rd	B	2							CCR→Rd8	—	—	—	—	—	2
	STC CCR, @ERd	W		4						CCR→@ERd	—	—	—	—	—	6
	STC CCR, @(d:16, ERd)	W			6					CCR→@(d:16, ERd)	—	—	—	—	—	8
	STC CCR, @(d:24, ERd)	W			10					CCR→@(d:24, ERd)	—	—	—	—	—	12
	STC CCR, @-ERd	W				4				ERd32-2→ERd32, CCR→@ERd	—	—	—	—	—	8
	STC CCR, @aa:16	W					6			CCR→@aa:16	—	—	—	—	—	8
	STC CCR, @aa:24	W						8		CCR→@aa:24	—	—	—	—	—	10
ANDC	ANDC #xx:8, CCR	B	2							CCR^#xx:8→CCR	↑	↑	↑	↑	↑	2
ORC	ORC #xx:8, CCR	B	2							CCR V #xx:8→CCR	↑	↑	↑	↑	↑	2
XORC	XORC #xx:8, CCR	B	2							CCR@#xx:8→CCR	↑	↑	↑	↑	↑	2
NOP	NOP	—							2	PC←PC+2	—	—	—	—	—	2

(8) ブロック転送命令

ニーモニック	サイズ	アドレッシングモード/命令長 (バイト)				オペレーション	コンディションコード				実行ステータス数 *1						
		#xx	Rn	@ERn	@(d, ERn)		@ERnERn+	@aa	@(d, PC)	@aa		I	H	Z	V	C	ノーマル
EEMOV.B	—									4	if R4L ≠ 0 Repeat @ER5→@ER6 R5+1→R5 R6+1→R6 R4L-1→R4L Until R4L=0 else next;	—	—	—	—	—	8+4n *2
EEMOV.W	—									4	if R4 ≠ 0 Repeat @ER5→@ER6 R5+1→R5 R6+1→R6 R4-1→R4 Until R4=0 else next;	—	—	—	—	—	8+4n *2

【注】 *1 実行ステータス数は、オペコードおよびオペランドが内蔵メモリに存在する場合です。それ以外の場合は、「2.6 命令実行ステータス数」を参照してください。

*2 nはR4LまたはR4の設定値です。

- (1) ビット11から桁上がりまたはビット11へ桁下がりが発生したとき1にセットされ、それ以外のとき0にクリアされます。
- (2) ビット27から桁上がりまたはビット27へ桁下がりが発生したとき1にセットされ、それ以外のとき0にクリアされます。
- (3) 演算結果がゼロのとき、演算前の値を保持し、それ以外のとき0にクリアされます。
- (4) 補正結果に桁上がりが発生したとき、1にセットされ、それ以外のとき演算前の値を保持します。
- (5) エクロック同期転送命令の実行ステータス数は一定ではありません。
- (6) 除数が負のとき1にセットされ、それ以外のとき0にクリアされます。
- (7) 除数がゼロのとき1にセットされ、それ以外のとき0にクリアされます。
- (8) 商が負のとき1にセットされ、それ以外のとき0にクリアされます。

2.4 命令コード一覧

表2.3 命令コード一覧

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト								
ADD	ADD.B #xx:8,Rd	B	8	rd	IMM															
	ADD.B Rs,Rd	B	0	8	rs	rd														
	ADD.W #xx:16,Rd	W	7	9	1	rd		IMM												
	ADD.W Rs,Rd	W	0	9	rs	rd														
	ADD.L #xx:32,ERd	L	7	A	1	0	erd				IMM									
ADDS	ADD.L ERs,ERd	L	0	A	1	ers	0	erd												
	ADDS #1,ERd	L	0	B	0	0	erd													
	ADDS #2,ERd	L	0	B	8	0	erd													
	ADDS #4,ERd	L	0	B	9	0	erd													
	ADDS #x:8,Rd	B	9	rd	IMM															
AND	ADDX Rs,Rd	B	0	E	rs	rd														
	AND.B #xx:8,Rd	B	E	rd	IMM															
	AND.B Rs,Rd	B	1	6	rs	rd														
	AND.W #xx:16,Rd	W	7	9	6	rd			IMM											
	AND.W Rs,Rd	W	6	6	rs	rd														
ANDC	AND.L #xx:32,ERd	L	7	A	6	0	erd													
	AND.L ERs,ERd	L	0	1	F	0	ers	0	erd											
	ANDC #xx:8,CCR	B	0	6	IMM															
	BAND #x:3,@ERd	B	7	C	0	IMM	rd													
	BAND #x:3,@aa:8	B	7	E	abs	0	IMM	0												
Bcc	BRA d:8,(BT d:8)	-	4	0	disp															
	BRA d:16,(BT d:16)	-	5	8	0	0	disp													
	BRN d:8,(BF d:8)	-	4	1	disp															
	BRN d:16,(BF d:16)	-	5	8	1	0	disp													
	BHI d:8	-	4	2	disp															
	BHI d:16	-	5	8	2	0	disp													
	BLS d:8	-	4	3	disp															
	BLS d:16	-	5	8	3	0	disp													
	BCC d:8,(BHS d:8)	-	4	4	disp															
	BCC d:16,(BHS d:16)	-	5	8	4	0	disp													
	BCS d:8,(BLO d:8)	-	4	5	disp															
	BCS d:16,(BLO d:16)	-	5	8	5	0	disp													
	BNE d:8	-	4	6	disp															
	BNE d:16	-	5	8	6	0	disp													
	BEQ d:8	-	4	7	disp															
BEQ d:16	-	5	8	7	0	disp														
BVC d:8	-	4	8	disp																
BVC d:16	-	5	8	8	0	disp														
BVS d:8	-	4	9	disp																
BVS d:16	-	5	8	9	0	disp														
BPL d:8	-	4	A	disp																
BPL d:16	-	5	8	A	0	disp														

2. 各命令の説明

命令	二一モニツク	サイズ	インストラクシヨソフオーマツト																	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト								
Bcc (続き)	BMI d:8	—	4	B	disp															
	BMI d:16	—	5	B	B	0														
	BGE d:8	—	4	C	disp															
	BGE d:16	—	5	C	C	0														
	BLT d:8	—	4	D	disp															
	BLT d:16	—	5	D	D	0														
	BGT d:8	—	4	E	disp															
	BGT d:16	—	5	E	E	0														
	BLE d:8	—	4	F	disp															
	BLE d:16	—	5	F	F	0														
BCLR	BCLR #xx:3,Rd	B	7	2	0:IMM	rd														
	BCLR #xx:3,@ERd	B	7	D	0:erd	0	7	2	0:IMM	0										
	BCLR #xx:3,@aa:8	B	7	F	abs		7	2	0:IMM	0										
	BCLR Rn,Rd	B	6	2	rn	rd														
	BCLR Rn,@ERd	B	7	D	0:erd	0	6	2	rn	0										
	BCLR Rn,@aa:8	B	7	F	abs		6	2	rn	0										
BIAND	BIAND #xx:3,Rd	B	7	6	1:IMM	rd														
	BIAND #xx:3,@ERd	B	7	C	0:erd	0	7	6	1:IMM	0										
	BIAND #xx:3,@aa:8	B	7	E	abs		7	6	1:IMM	0										
	BIAND #xx:3,Rd	B	7	7	1:IMM	rd														
BILD	BILD #xx:3,Rd	B	7	7	1:IMM	rd														
	BILD #xx:3,@ERd	B	7	C	0:erd	0	7	7	1:IMM	0										
	BILD #xx:3,@aa:8	B	7	E	abs		7	7	1:IMM	0										
	BILD #xx:3,Rd	B	7	7	1:IMM	rd														
BIOR	BIOR #xx:3,Rd	B	7	4	1:IMM	rd														
	BIOR #xx:3,@ERd	B	7	C	0:erd	0	7	4	1:IMM	0										
	BIOR #xx:3,@aa:8	B	7	E	abs		7	4	1:IMM	0										
	BIOR #xx:3,Rd	B	6	7	1:IMM	rd														
BIST	BIST #xx:3,@ERd	B	7	D	0:erd	0	6	7	1:IMM	0										
	BIST #xx:3,@aa:8	B	7	F	abs		6	7	1:IMM	0										
	BIST #xx:3,Rd	B	7	5	1:IMM	rd														
	BIST #xx:3,@ERd	B	7	C	0:erd	0	7	5	1:IMM	0										
BIXOR	BIXOR #xx:3,@ERd	B	7	E	abs		7	5	1:IMM	0										
	BIXOR #xx:3,@aa:8	B	7	E	abs		7	5	1:IMM	0										
	BIXOR #xx:3,Rd	B	7	7	0:IMM	rd														
	BIXOR #xx:3,@ERd	B	7	C	0:erd	0	7	7	0:IMM	0										
BLD	BLD #xx:3,Rd	B	7	7	0:IMM	rd														
	BLD #xx:3,@ERd	B	7	C	0:erd	0	7	7	0:IMM	0										
	BLD #xx:3,@aa:8	B	7	E	abs		7	7	0:IMM	0										
	BLD #xx:3,Rd	B	7	7	0:IMM	rd														
BNOT	BNOT #xx:3,Rd	B	7	1	0:IMM	rd														
	BNOT #xx:3,@ERd	B	7	D	0:erd	0	7	1	0:IMM	0										
	BNOT #xx:3,@aa:8	B	7	F	abs		7	1	0:IMM	0										
	BNOT Rn,Rd	B	6	1	rn	rd														
BOR	BNOT Rn,@ERd	B	7	D	0:erd	0	6	1	rn	0										
	BNOT Rn,@aa:8	B	7	F	abs		6	1	rn	0										
	BOR #xx:3,Rd	B	7	4	0:IMM	rd														
	BOR #xx:3,@ERd	B	7	C	0:erd	0	7	4	0:IMM	0										
BOR #xx:3,@aa:8	B	7	E	abs		7	4	0:IMM	0											

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト								
BSET	BSET #xx:3,Rd	B 7 0	0:IMM	rd																
	BSET #xx:3,@ERd	B 7 D	0:erd	0	7	0	0:IMM	0												
	BSET #xx:3,@aa:8	B 7 F	abs		7	0	0:IMM	0												
	BSET Rn,Rd	B 6 0	rn	rd																
	BSET Rn,@ERd	B 7 D	0:erd	0	6	0	rn	0												
	BSET Rn,@aa:8	B 7 F	abs		6	0	rn	0												
BSR	BSR d:8	- 5 5	disp																	
	BSR d:16	- 5 5	C 0	0																
BST	BST #xx:3,Rd	B 6 7	0:IMM	rd																
	BST #xx:3,@ERd	B 7 D	0:erd	0	6	7	0:IMM	0												
	BST #xx:3,@aa:8	B 7 F	abs		6	7	0:IMM	0												
	BST #xx:3,Rd	B 7 3	0:IMM	rd																
BTST	BTST #xx:3,Rd	B 7 C	0:erd	0	7	3	0:IMM	0												
	BTST #xx:3,@ERd	B 7 C	0:erd	0	7	3	0:IMM	0												
	BTST #xx:3,@aa:8	B 7 E	abs		7	3	0:IMM	0												
	BTST Rn,Rd	B 6 3	rn	rd																
	BTST Rn,@ERd	B 7 C	0:erd	0	6	3	rn	0												
	BTST Rn,@aa:8	B 7 E	abs		6	3	rn	0												
BXOR	BXOR #xx:3,Rd	B 7 5	0:IMM	rd																
	BXOR #xx:3,@ERd	B 7 C	0:erd	0	7	5	0:IMM	0												
	BXOR #xx:3,@aa:8	B 7 E	abs		7	5	0:IMM	0												
	BXOR #xx:3,Rd	B 7 A	rd	IMM																
CMP	CMP.B Rs,Rd	B 1 C	rs	rd																
	CMP.W #xx:16,Rd	W 7 9	2	rd			IMM													
	CMP.W Rs,Rd	W 1 D	rs	rd																
	CMP.L #xx:32,ERd	L 7 A	2	0:erd																
	CMP.L ERs,ERd	L 1 F	1:ers	0:erd																
	DAA Rd	B 0 F	0	rd																
DAS	DAS Rd	B 1 F	0	rd																
	DEC.B Rd	B 1 A	0	rd																
DEC	DEC.W #1,Rd	W 1 B	5	rd																
	DEC.W #2,Rd	W 1 B	D	rd																
	DECL.#1,ERd	L 1 B	7	0:erd																
	DECL.#2,ERd	L 1 B	F	0:erd																
DIVXS	DIVXS.B Rs,Rd	B 0 1	D	0	5	1	rs	rd												
	DIVXS.W Rs,ERd	W 0 1	D	0	5	3	rs	0:erd												
DIVXU	DIVXU.B Rs,Rd	B 5 1	rs	rd																
	DIVXU.W Rs,ERd	W 5 3	rs	0:erd																
EEMOV	EEMOV.B	- 7 B	5	C	5	9	8	F												
	EEMOV.W	- 7 B	D	4	5	9	8	F												
EXTS	EXTS.W Rd	W 1 7	D	rd																
	EXTS.L ERd	L 1 7	F	0:erd																
EXTU	EXTU.W Rd	W 1 7	5	rd																
	EXTU.L ERd	L 1 7	7	0:erd																

2. 各命令の説明

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト								
INC	INC.B Rd	B	0	A	0	rd														
	INC.W #1, Rd	W	0	B	5	rd														
	INC.W #2, Rd	W	0	B	D	rd														
	INCL.#1, ERd	L	0	B	7	0:erd														
	INCL.#2, ERd	L	0	B	F	0:erd														
	JMP @ERn	-	5	9	0:ern	0														
JMP	JMP @aa:24	-	5	A		abs														
	JMP @aa:8	-	5	B		abs														
	JSR @ERn	-	5	D	0:ern	0														
	JSR @aa:24	-	5	E		abs														
LDC	JSR @aa:8	-	5	F		abs														
	LDC #xx:8, CCR	B	0	7	IMM															
	LDC Rs, CCR	B	0	3	0	rs														
	LDC @ERS, CCR	W	0	1	4	0	6	9	0:ers	0										
	LDC @(d:16, ERs), CCR	W	0	1	4	0	6	F	0:ers	0										
	LDC @(d:24, ERs), CCR	W	0	1	4	0	7	8	0:ers	0										
	LDC @ER+, CCR	W	0	1	4	0	6	D	0:ers	0										
	LDC @aa:16, CCR	W	0	1	4	0	6	B	0	0										
	LDC @aa:24, CCR	W	0	1	4	0	6	B	2	0										
	MOV.B #xx:8, Rd	B	F	rd	IMM															
	MOV.B Rs, Rd	B	0	C	rs	rd					0	0								
	MOV	MOV.B @ERS, Rd	B	6	8	0:ers	rd													
MOV.B @(d:16, ERs), Rd		B	6	E	0:ers	rd														
MOV.B @(d:24, ERs), Rd		B	7	8	0:ers	0														
MOV.B @ER+, Rd		B	6	C	0:ers	rd														
MOV.B @aa:8, Rd		B	2	rd		abs														
MOV.B @aa:16, Rd		B	6	A	0	rd														
MOV.B @aa:24, Rd		B	6	A	2	rd														
MOV.B Rs, @ERd		B	6	8	1:erd	rs														
MOV.B Rs, @(d:16, ERd)		B	6	E	1:erd	rs														
MOV.B Rs, @(d:24, ERd)		B	7	8	0:erd	0														
MOV.B Rs, @+ERd		B	6	C	1:erd	rs														
MOV.B Rs, @aa:8		B	3	rs		abs														
MOV.B Rs, @aa:16		B	6	A	8	rs														
MOV.B Rs, @aa:24		B	6	A	A	rs	0	0												
MOV.W #xx:16, Rd		W	7	9	0	rd														
MOV.W Rs, Rd		W	0	D	rs	rd														
MOV.W @ERS, Rd	W	6	9	0:ers	rd															
MOV.W @(d:16, ERs), Rd	W	6	F	0:ers	rd															
MOV.W @(d:24, ERs), Rd	W	7	8	0:ers	0															
MOV.W @ER+, Rd	W	6	D	0:ers	rd															
MOV.W @aa:16, Rd	W	6	B	0	rd															
MOV.W @aa:24, Rd	W	6	B	2	rd															

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト								
MOV (続き)	MOV.W Rs,@ERd	W	6	9	1:erd	rs														
	MOV.W Rs,@(d:16,ERd)	W	6	F	1:erd	rs	disp													
	MOV.W Rs,@(d:24,ERd)	W	7	8	0:erd	0	6	B	A	rs	0	0	0	0	disp					
	MOV.W Rs,@-ERd	W	6	D	1:erd	rs														
	MOV.W Rs,@aa:16	W	6	B	8	rs	abs													
	MOV.W Rs,@aa:24	W	6	B	A	rs	0	0			abs									
	MOV.L #xx:32,Rd	L	7	A	0:0:erd						IMM									
	MOV.L ERs,ERd	L	0	F	1:ers:0:erd															
	MOV.L @ERs,ERd	L	0	1	0	0	6	9	0:ers:0:erd											
	MOV.L @(d:16,ERs),ERd	L	0	1	0	0	6	F	0:ers:0:erd			disp								
MOV.L @(d:24,ERs),ERd	L	0	1	0	0	7	8	0:ers:0:erd			0	6	B	2	0:erd	0	0		disp	
MOV.L @ERs+,ERd	L	0	1	0	0	6	D	0:ers:0:erd												
MOV.L @aa:16,ERd	L	0	1	0	0	6	B	0:0:erd			abs									
MOV.L @aa:24,ERd	L	0	1	0	0	6	B	2:0:erd			abs									
MOV.L ERs,@ERd	L	0	1	0	0	6	9	1:erd:0:ers												
MOV.L ERs,@(d:16,ERd)	L	0	1	0	0	6	F	1:erd:0:ers			disp									
MOV.L ERs,@(d:24,ERd)	L	0	1	0	0	7	8	1:erd:0:ers			0	6	B	A	0:ers	0	0		disp	
MOV.L ERs,@-ERd	L	0	1	0	0	6	D	1:erd:0:ers												
MOV.L ERs,@aa:16	L	0	1	0	0	6	B	8:0:ers			abs									
MOV.L ERs,@aa:24	L	0	1	0	0	6	B	A:0:ers			0	0	0						abs	
MOV.FPE @aa:16,Rd	B	6	A	4	rd			abs												
MOV.FPE @aa:16	B	6	A	C	rs			abs												
MULXS	MULXS.B Rs,Rd	B	0	1	C	0	5	0	rs	rd										
	MULXS.W Rs,ERd	W	0	1	C	0	5	2	rs	0:erd										
MULXU	MULXU.B Rs,Rd	B	5	0	rs	rd														
	MULXU.W Rs,ERd	W	5	2	rs	0:erd														
NEG	NEG.B Rd	B	1	7	8	rd														
	NEG.W Rd	W	1	7	9	rd														
NOP	NEG.L ERd	L	1	7	B	0:erd														
	NOP	-	0	0	0	0														
NOT	NOT.B Rd	B	1	7	0	rd														
	NOT.W Rd	W	1	7	1	rd														
OR	NOT.L ERd	L	1	7	3	0:erd														
	OR.B #xx:8,Rd	B	C	rd	IMM															
	OR.B Rs,Rd	B	1	4	rs	rd														
	OR.W #xx:16,Rd	W	7	9	4	rd				IMM										
ORC	OR.W Rs,Rd	W	6	4	rs	rd														
	OR.L #xx:32,ERd	L	7	A	4:0:erd					IMM										
POP	OR.L ERs,ERd	L	0	1	F	0	6	4	0:ers:0:erd											
	ORC #xx:8,CCR	B	0	4	IMM															
POP	POP.W Rn	W	6	D	7	rn														
	POP.L ERn	L	0	1	0	0	6	D	7	0:ern										

2. 各命令の説明

命令	ニーモニック	サ フ ス	インストラクションフォーマット																	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト								
PUSH	PUSH.W Rn	W	6	D	F	rn														
	PUSH.L ERn	L	0	1	0	0	6	D	F	0:ern										
ROTL	ROTL.B Rd	B	1	2	8	rd														
	ROTL.W Rd	W	1	2	9	rd														
ROTR	ROTL.ERd	L	1	2	B	0:erd														
	ROTR.B Rd	B	1	3	8	rd														
ROTXL	ROTR.W Rd	W	1	3	9	rd														
	ROTR.L ERd	L	1	3	B	0:erd														
ROTXR	ROTXL.B Rd	B	1	2	0	rd														
	ROTXL.W Rd	W	1	2	1	rd														
RTE	ROTXR.B Rd	B	1	3	0	rd														
	ROTXR.W Rd	W	1	3	1	rd														
SHAL	ROTXR.L ERd	L	1	3	3	0:erd														
	RTS	-	5	6	7	0														
SHAR	RTS	-	5	4	7	0														
	SHAL.B Rd	B	1	0	8	rd														
SHLL	SHAL.W Rd	W	1	0	9	rd														
	SHAL.L ERd	L	1	0	B	0:erd														
SHLR	SHAR.B Rd	B	1	1	8	rd														
	SHAR.W Rd	W	1	1	9	rd														
SLEEP	SHAR.L ERd	L	1	1	B	0:erd														
	SHLL.B Rd	B	1	0	0	rd														
STC	SHLL.W Rd	W	1	0	1	rd														
	SHLL.L ERd	L	1	0	3	0:erd														
SUB	SHL.R.B Rd	B	1	1	0	rd														
	SHL.R.W Rd	W	1	1	1	rd														
SUB	SHL.L ERd	L	1	1	3	0:erd														
	SLEEP	-	0	1	8	0														
SUB	STC.CCR.Rd	B	0	2	0	rd														
	STC.CCR.@ERd	W	0	1	4	0	6	9	1:erd	0										
SUB	STC.CCR.@(d:16,ERd)	W	0	1	4	0	6	F	1:erd	0										
	STC.CCR.@(d:24,ERd)	W	0	1	4	0	7	8	0:erd	0	6	B	A	0	0	0				disp
SUB	STC.CCR.@ERd	W	0	1	4	0	6	D	1:erd	0										
	STC.CCR.@aa:16	W	0	1	4	0	6	B	8	0										
SUB	STC.CCR.@aa:24R	W	0	1	4	0	6	B	A	0	0	0								abs
	SUB.B Rs,Rd	B	1	8	rs	rd														
SUB	SUB.W #xx:16,Rd	W	7	9	3	rd														
	SUB.W Rs,Rd	W	1	9	rs	rd														
SUB	SUB.L #xx:32,ERd	L	7	A	3	0:erd														
	SUB.L ERs,ERd	L	1	A	1:ets	0:erd														

命令	ニーモニック	サイズ	インストラクションフォーマット																	
			第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第7バイト	第8バイト	第9バイト	第10バイト								
SUBS	SUBS #1,ERd	L	1	B	0	0	erd													
	SUBS #2,ERd	L	1	B	8	0	erd													
	SUBS #4,ERd	L	1	B	9	0	erd													
	SUBS #xx:8,Rd	B	B	rd		IMM														
SUBX	SUBX Rs,Rd	B	1	E	rs	rd														
	TRAPA #x:2	—	5	7	00	IMM	0													
XOR	XOR.B #xx:8,Rd	B	D	rd		IMM														
	XOR.B Rs,Rd	B	1	5	rs	rd														
	XOR.W #xx:16,Rd	W	7	9	5	rd			IMM											
	XOR.W Rs,Rd	W	6	5	rs	rd														
	XOR.L #xx:32,ERd	L	7	A	5	0	erd													
XORC	XORC.L ERs,ERd	L	0	1	F	0				5	0	ers	0	erd						
	XORC #xx:8,CCR	B	0	5		IMM														

2. 各命令の説明

《記号説明》

- IMM : イミディエイトデータ (2、3、8、16、32 ビット)
 abs : 絶対アドレス (8、16、24 ビット)
 disp : ディスプレースメント (8、16、24 ビット)
 rs、rd、rn : レジスタフィールド (4 ビットで 8 ビットレジスタまたは 16 ビットレジスタを指定します。rs、rd、rn はそれぞれオペランド形式の Rs、Rd、Rn に対応します。)
 ers、erd、ern : レジスタフィールド (3 ビットでアドレスレジスタまたは 32 ビットレジスタを指定します。ers、erd、ern はそれぞれオペランド形式の ERs、ERd、ERn に対応します。)

レジスタフィールドと汎用レジスタの対応を下表に示します。

アドレスレジスタ 32 ビットレジスタ		16 ビットレジスタ		8 ビットレジスタ	
レジスタフィールド	汎用レジスタ	レジスタフィールド	汎用レジスタ	レジスタフィールド	汎用レジスタ
000	ER0	0000	R0	0000	R0H
001	ER1	0001	R1	0001	R1H
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
111	ER7	0111	R7	0111	R7H
		1000	E0	1000	R0L
		1001	E1	1001	R1L
		⋮	⋮	⋮	⋮
		⋮	⋮	⋮	⋮
		1111	E7	1111	R7L

2.5 オペレーションコードマップ

表 2.4～表 2.6 にオペレーションコードマップを示します。

表 2.4 オペレーションコードマップ (1)



命令コード：

第1バイト	第2バイト
AH AL	BH BL

AH	AL	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	表2.5	STC	LDC	ORC	XORC	ANDC	LDC	ADD	表2.5	ADD	表2.5	表2.5	MOV	ADDX	表2.5	表2.5
1	表2.5	表2.5	OR	表2.5	OR	XOR	AND	AND	表2.5	SUB	表2.5	表2.5	表2.5	CMP	表2.5	SUBX	表2.5
2																	
3																	
4	BRA	BRN	BHI	BLS	BCC	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE	
5	MULXU	DIVXU	MULXU	DIVXU	RTS	BSR	RTE	TRAPA	表2.5	JMP	BSR	JSR					
6	BSET	BNOT	BCLR	BTST	OR	XOR	AND	BST	BIS								
7					BOR	BXOR	BAND	BID	BAND	BOR	BXOR	BAND	BID	MOV	表2.5	EPMOV	表2.6
8																	
9																	
A																	
B																	
C																	
D																	
E																	
F																	

2. 各命令の説明

表2.5 オペレーションコードマップ (2)

第1バイト		第2バイト	
AH	AL	BH	BL

命令コード：

BH/AH	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01	MOV				LDC/STC				SLEEP				表2.6	表2.6		表2.6
0A	INC												ADD			
0B	ADDS					INC		INC	ADDS					INC		INC
0F	DAA												MOV			
10	SHLL			SHLL					SHAL			SHAL	SHAL			
11	SHLR			SHLR					SHAR			SHAR	SHAR			
12	ROTXL			ROTXL					ROTL			ROTL	ROTL			
13	ROTXR			ROTXR					ROTR			ROTR	ROTR			
17	NOT			NOT				EXTU	NEG			NEG	NEG	EXTS		EXTS
1A	DEC												SUB			
1B	SUBS					DEC		DEC	SUBS					DEC		DEC
1F	DAS												CMP			
58	BRA	BRN	BHI	BLS	BCC	BCS	BNE	BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE
79	MOV	ADD	CMP	SUB	OR	XOR	AND									
7A	MOV	ADD	CMP	SUB	OR	XOR	AND									

表2.6 オペレーションコンコードマップ (3)

命令コード	第1バイト		第2バイト		第3バイト		第4バイト	
	AH	AL	BH	BL	CH	CL	DH	DL



命令コード	CL	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
AHALHBLCH																	
01C05		MULXS		MULXS													
01D05			DIVXS		DIVXS												
01F06						OR	XOR	AND									
7C06 ^{*1}																	
7C07 ^{*1}					BTST	BOR	BIXOR	BAND	BAND	BAND	BAND	BAND	BAND	BAND	BAND	BAND	BAND
7D06 ^{*1}		BSET	BNOT	BCLR													
7D07 ^{*1}		BSET	BNOT	BCLR													
7Eaa6 ^{*2}					BTST												
7Eaa7 ^{*2}					BTST	BOR	BIXOR	BAND	BAND	BAND	BAND	BAND	BAND	BAND	BAND	BAND	BAND
7Faa6 ^{*2}		BSET	BNOT	BCLR													
7Faa7 ^{*2}		BSET	BNOT	BCLR													

【注】 *1 rはレジスタ指定部
*2 aaは絶対アドレス指定部

2.6 命令実行ステート数

H8/300H CPU の各命令についての実行状態と実行ステート数の計算方法を示します。

表 2.8 に各命令の実行状態として、命令実行中に行われる命令フェッチ、データリード/ライト等のサイクル数を示し、表 2.7 に各々のサイズに必要なステート数を示します。

命令の実行ステート数は次の計算式で計算されます。

$$\text{実行ステート数} = I \cdot SI + J \cdot SJ + K \cdot SK + L \cdot SL + M \cdot SM + N \cdot SN$$

実行ステート数計算例

(例)

アドバンスモード、プログラム領域およびスタック領域を外部空間に設定、内部周辺モジュールアクセス時 8 ビットバス幅、外部デバイスアクセス時 16 ビットバス幅で 3 ステートアクセス 1 ウェイト挿入とした場合。

1. BSET #0, @FFFFC7:8

表 2.8 より、

$$I=L=2, J=K=M=N=0$$

表 2.7 より

$$SI=4, SL=3$$

$$\text{実行ステート数} = 2 \times 4 + 2 \times 3 = 14$$

2. JSR @@30

表 2.8 より

$$I=J=K=2, L=M=N=0$$

表 2.7 より

$$SI=SJ=SK=4$$

$$\text{実行ステート数} = 2 \times 4 + 2 \times 4 + 2 \times 4 = 24$$

表 2.7 実行状態 (サイクル) に要するステート数

実行状態 (サイクル)	アクセス対象						
	内 臓 メモリ	内臓周辺モジュール		外部デバイス			
		8ビット バス	16ビット バス	8ビットバス		16ビットバス	
				2ステート アクセス	3ステート アクセス	2ステート アクセス	3ステート アクセス
命令フェッチ S _I	2		3	4	6+2m	2	3+m*
分岐アドレスリード S _J							
スタック操作 S _K							
バイトデータアクセス S _L							
ワードデータアクセス S _M							
内部動作 S _N	1						

【注】 * MOVFPE, MOVTPPEについては当該LSIのハードウェアマニュアルを参照してください。

《記号説明》

m : 外部デバイスアクセス時のウェイトステート数

表 2.8 命令実行状態（サイクル）数

命令	ニーモニック	命令フェッチ	分岐アドレス リード	スタック操作	バイトデータ アクセス	ワードデータ アクセス	内部動作
		I	J	K	L	M	N
ADD	ADD.B #xx:8,Rd	1					
	ADD.B Rs,Rd	1					
	ADD.W #xx:16,Rd	2					
	ADD.W Rs,Rd	1					
	ADD.L #xx:32,ERd	3					
ADD.L ERs,ERd	1						
ADDS	ADDS #1/2/4,ERd	1					
ADDX	ADDX #xx:8,Rd	1					
	ADDX Rs,Rd	1					
AND	AND.B #xx:8,Rd	1					
	AND.B Rs,Rd	1					
	AND.W #xx:16,Rd	2					
	AND.W Rs,Rd	1					
	AND.L #xx:32,ERd	3					
AND.L ERs,ERd	2						
ANDC	ANDC #xx:8,CCR	1					
BAND	BAND #xx:3,Rd	1					
	BAND #xx:3,@ERd	2			1		
	BAND #xx:3,@aa:8	2			1		
Bcc	BRA d:8 (BT d:8)	2					
	BRN d:8 (BF d:8)	2					
	BHI d:8	2					
	BLS d:8	2					
	BCC d:8 (BHS d:8)	2					
	BCS d:8 (BLO d:8)	2					
	BNE d:8	2					
	BEQ d:8	2					
	BVC d:8	2					
	BVS d:8	2					
	BPL d:8	2					
	BMI d:8	2					
	BGE d:8	2					
	BLT d:8	2					
	BGT d:8	2					
	BLE d:8	2					
	BRA d:16 (BT d:16)	2					2
	BRN d:16 (BF d:16)	2					2
	BHI d:16	2					2
	BLS d:16	2					2
	BCC d:16 (BHS d:16)	2					2
	BCS d:16 (BLO d:16)	2					2
	BNE d:16	2					2
	BEQ d:16	2					2
	BVC d:16	2					2
	BVS d:16	2					2
	BPL d:16	2					2
BMI d:16	2					2	
BGE d:16	2					2	
BLT d:16	2					2	
BGT d:16	2					2	
BLE d:16	2					2	
BCLR	BCLR #xx:3,Rd	1					
	BCLR #xx:3,@ERd	2			2		
	BCLR #xx:3,@aa:8	2			2		
	BCLR Rn,Rd	1					
	BCLR Rn,@ERd	2			2		
	BCLR Rn,@aa:8	2			2		
BIAND	BIAND #xx:3,Rd	1					
	BIAND #xx:3,@ERd	2			1		
	BIAND #xx:3,@aa:8	2			1		
BILD	BILD #xx:3,Rd	1					
	BILD #xx:3,@ERd	2			1		
	BILD #xx:3,@aa:8	2			1		
BIOR	BIOR #xx:8,Rd	1					
	BIOR #xx:8,@ERd	2			1		
	BIOR #xx:8,@aa:8	2			1		
BIST	BIST #xx:3,Rd	1					
	BIST #xx:3,@ERd	2			2		
	BIST #xx:3,@aa:8	2			2		

2. 各命令の説明

命令	ニーモニック		命令フェッチ	分岐アドレス リード	スタック操作	バイトデータ アクセス	ワードデータ アクセス	内部動作
			I	J	K	L	M	N
BIXOR	BIXOR #xx:3,Rd		1					
	BIXOR #xx:3,@ERd		2			1		
	BIXOR #xx:3,@aa:8		2			1		
BLD	BLD #xx:3,Rd		1					
	BLD #xx:3,@ERd		2			1		
	BLD #xx:3,@aa:8		2			1		
BNOT	BNOT #xx:3,Rd		1					
	BNOT #xx:3,@ERd		2			2		
	BNOT #xx:3,@aa:8		2			2		
	BNOT Rn,Rd		1					
	BNOT Rn,@ERd		2			2		
	BNOT Rn,@aa:8		2			2		
BOR	BOR #xx:3,Rd		1					
	BOR #xx:3,@ERd		2			1		
	BOR #xx:3,@aa:8		2			1		
BSET	BSET #xx:3,Rd		1					
	BSET #xx:3,@ERd		2			2		
	BSET #xx:3,@aa:8		2			2		
	BSET Rn,Rd		1					
	BSET Rn,@ERd		2			2		
	BSET Rn,@aa:8		2			2		
BSR	BSR d:8	ノーマル	2		1			
		アドバンスト	2		2			
	BSR d:16	ノーマル	2		1			2
		アドバンスト	2		2			2
BST	BST #xx:3,Rd		1					
	BST #xx:3,@ERd		2			2		
	BST #xx:3,@aa:8		2			2		
BTST	BTST #xx:3,Rd		1					
	BTST #xx:3,@ERd		2			1		
	BTST #xx:3,@aa:8		2			1		
	BTST Rn,Rd		1					
	BTST Rn,@ERd		2			1		
	BTST Rn,@aa:8		2			1		
BXOR	BXOR #xx:3,Rd		1					
	BXOR #xx:3,@ERd		2			1		
	BXOR #xx:3,@aa:8		2			1		
CMP	CMP.B #xx:8,Rd		1					
	CMP.B Rs,Rd		1					
	CMP.W #xx:16,Rd		2					
	CMP.W Rs,Rd		1					
	CMP.L #xx:32,ERd		3					
	CMP.L ERs,ERd		1					
DAA	DAA Rd		1					
DAS	DAS Rd		1					
DEC	DEC.B Rd		1					
	DEC.W #1/2,Rd		1					
	DEC.L #1/2,ERd		1					
DIVXS	DIVXS.B Rs,Rd		2					12
	DIVXS.W Rs,ERd		2					20
DIVXU	DIVXU.B Rs,Rd		1					12
	DIVXU.W Rs,ERd		1					20
EEPMOV	EEPMOV.B		2			2n+2 *1		
	EEPMOV.W		2			2n+2 *1		
EXTS	EXTS.W Rd		1					
	EXTS.L ERd		1					
EXTU	EXTU.W Rd		1					
	EXTU.L ERd		1					
INC	INC.B Rd		1					
	INV.W #1/2,Rd		1					
	INC.L #1/2,ERd		1					
JMP	JMP@ERn		2					
	JMP@aa:24		2					2
	JMP@aa:8	ノーマル	2	1				2
		アドバンスト	2	2				2
JSR	JSR@ERn	ノーマル	2		1			
		アドバンスト	2		2			
		アドバンスト	2		2			
	JSR@aa:24	ノーマル	2		1			2
		アドバンスト	2		2			2

2. 各命令の説明

命令	ニーモニック		命令フェッチ	分岐アドレス リード	スタック操作	バイトデータ アクセス	ワードデータ アクセス	内部動作		
			I	J	K	L	M	N		
JSR	JSR @:aa:8	ノーマル	2	1	1					
		アドバンスト	2	2	2					
LDC	LDC #xx:8,CCR LDC Rs,CCR LDC @ERs,CCR LDC @(d:16,ERs),CCR LDC @(d:24,ERs),CCR LDC @ERs+,CCR LDC @aa:16,CCR LDC @aa:24,CCR		1							
			1							
			2				1			
			3				1			
			5				1			
			2				1	2		
			3				1			
MOV	MOV.B #xx:8,Rd MOV.B Rs,Rd MOV.B @ERs,Rd MOV.B @(d:16,ERs),Rd MOV.B @(d:24,ERs),Rd MOV.B @ERs+,Rd MOV.B @aa:8,Rd MOV.B @aa:16,Rd MOV.B @aa:24,Rd MOV.B Rs,@ERd MOV.B Rs,@(d:16,ERd) MOV.B Rs,@(d:24,ERd) MOV.B Rs,@-ERd MOV.B Rs,@aa:8 MOV.B Rs,@aa:16 MOV.B Rs,@aa:24 MOV.W #xx:16,Rd MOV.W Rs,Rd MOV.W @ERs,Rd MOV.W @(d:16,ERs),Rd MOV.W @(d:24,ERs),Rd MOV.W @ERs+,Rd MOV.W @aa:16,Rd MOV.W @aa:24,Rd MOV.W Rs,@ERd MOV.W Rs,@(d:16,ERd) MOV.W Rs,@(d:24,ERd) MOV.W Rs,@-ERd MOV.W Rs,@aa:16 MOV.W Rs,@aa:24 MOV.L #xx:32,ERd MOV.L ERs,ERd MOV.L @ERs,ERd MOV.L @(d:16,ERs),ERd MOV.L @(d:24,ERs),ERd MOV.L @ERs+,ERd MOV.L @aa:16,ERd MOV.L @aa:24,ERd MOV.L ERs,@ERd MOV.L ERs,@(d:16,ERd) MOV.L ERs,@(d:24,ERd) MOV.L ERs,@-ERd MOV.L ERs,@aa:16 MOV.L ERs,@aa:24		1							
			1							
			1				1			
			2				1			
			4				1			
			1				1		2	
			1				1			
			2				1			
			3				1			
			1				1			
			2				1			
			4				1			
			1				1		2	
			1				1			
			2				1			
			3				1			
			2							
			1					1		
			2					1		
			4					1		
			1					1	2	
			2					1		
			3					1		
			3					1		
			1					2		
			2					2		
			3					2		
			5					2		
			2					2	2	
			3					2		
			4					2		
			2					2		
			3					2		
			5					2		
			2					2	2	
			3					2		
			4					2		
		MOVFPPE	MOVFPPE @:aa:16,Rd		2			1 *2		
		MOVTPE	MOVTPE Rs,@:aa:16		2			1 *2		
		MULXS	MULXS.B Rs,Rd		2					12
			MULXS.W Rs,ERd		2					20
		MULXU	MULXU.B Rs,Rd		1					12
			MULXU.W Rs,ERd		1					20
		NEG	NEG.B Rd		1					
NEG.W Rd			1							
NEG.L ERd			1							
NOP	NOP		1							
NOT	NOT.B Rd		1							
	NOT.W Rd		1							
	NOT.L ERd		1							
OR	OR.B #xx:8,Rd		1							
	OR.B Rs,Rd		1							

2. 各命令の説明

命令	ニーモニック		命令フェッチ	分岐アドレス リード	スタック操作	バイトデータ アクセス	ワードデータ アクセス	内部動作
			I	J	K	L	M	N
OR	OR.W #xx:16,Rd		2					
	OR.W Rs,Rd		1					
	OR.L #xx:32,ERd		3					
	OR.L ERs,ERd		2					
ORC	ORC #xx:8,CCR		1					
POP	POP.W Rn		1				1	2
	POP.L ERn		2				2	2
PUSH	PUSH.W Rn		1				1	2
	PUSH.L ERn		2				2	2
ROTL	ROTL.B Rd		1					
	ROTL.W Rd		1					
	ROTL.L ERd		1					
ROTR	ROTR.B Rd		1					
	ROTR.W Rd		1					
	ROTR.L ERd		1					
ROTXL	ROTXL.B Rd		1					
	ROTXL.W Rd		1					
	ROTXL.L ERd		1					
ROTXR	ROTXR.B Rd		1					
	ROTXR.W Rd		1					
	ROTXR.L ERd		1					
RTE	RTE		2		2		2	
RTS	RTS		2		1		2	
	ノーマル アドバンスト		2		2		2	
SHAL	SHAL.B Rd		1					
	SHAL.W Rd		1					
	SHAL.L ERd		1					
SHAR	SHAR.B Rd		1					
	SHAR.W Rd		1					
	SHAR.L ERd		1					
SHLL	SHLL.B Rd		1					
	SHLL.W Rd		1					
	SHLL.L ERd		1					
SHLR	SHLR.B Rd		1					
	SHLR.W Rd		1					
	SHLR.L ERd		1					
SLEEP	SLEEP		1					
STC	STC CCR,Rd		1					
	STC CCR,@ERd		2				1	
	STC CCR,@(d:16,ERd)		3				1	
	STC CCR,@(d:24,ERd)		5				1	
	STC CCR,@-ERd		2				1	2
	STC CCR,@aa:16		3				1	
	STC CCR,@aa:24		4				1	
SUB	SUB.B Rs,Rd		1					
	SUB.W #xx:16,Rd		2					
	SUB.W Rs,Rd		1					
	SUB.L #xx:32,ERd		3					
	SUB.L ERs,ERd		1					
SUBS	SUBS #1/2/4,ERd		1					
SUBX	SUBX #xx:8,Rd		1					
	SUBX Rs,Rd		1					
TRAPA	TRAPA #x:2		2	1	2		4	
	ノーマル アドバンスト		2	2	2		4	
XOR	XOR.B #xx:8,Rd		1					
	XOR.B Rs,Rd		1					
	XOR.W #xx:16,Rd		2					
	XOR.W Rs,Rd		1					
	XOR.L 3xx:32,ERd		3					
	XOR.L ERs,ERd		2					
XORC	XORC #xx:8,CCR		1					

【注】 *1 nはR4L,R4の設定値です。ソース側、デスティネーション側のアクセスが、それぞれ (n+1) 回行われます。

*2 データアクセスに必要なステート数は、当該LSIのハードウェアマニュアルを参照してください。

2.7 コンディションコードの変化

CPU の各命令について、命令実行後のコンディションコードの変化を示します。
以下に、表中で行われている記号を説明します。

$m=$	$\left\{ \begin{array}{l} 31 : \text{ロングワードサイズの時} \\ 15 : \text{ワードサイズの時} \\ 7 : \text{バイトサイズの時} \end{array} \right.$
S_i	: ソースオペランドのビット <i>i</i>
D_i	: デスティネーションオペランドのビット <i>i</i>
R_i	: 結果のビット <i>i</i>
D_n	: デスティネーションオペランドの指定されたビット
—	: 影響なし
†	: 実行結果に応じて変化 (定義参照)
0	: 常に0にクリア
1	: 常に1にセット
*	: 値を保証しません
Z'	: 実行前のZフラグ
C'	: 実行前のCフラグ

2. 各命令の説明

表 2.9 コンディションコードの変化

命令	H	N	Z	V	C	定義
ADD	↑	↑	↑	↑	↑	$H = S_{m-4} \cdot D_{m-4} + D_{m-4} \cdot \overline{R_{m-4}} + S_{m-4} \cdot \overline{R_{m-4}}$ $N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $V = S_m \cdot D_m \cdot \overline{R_m} + \overline{S_m} \cdot \overline{D_m} \cdot R_m$ $C = S_m \cdot D_m + D_m \cdot \overline{R_m} + S_m \cdot \overline{R_m}$
ADDS	—	—	—	—	—	
ADDX	↑	↑	↑	↑	↑	$H = S_{m-4} \cdot D_{m-4} + D_{m-4} \cdot \overline{R_{m-4}} + S_{m-4} \cdot \overline{R_{m-4}}$ $N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $V = S_m \cdot D_m \cdot \overline{R_m} + \overline{S_m} \cdot \overline{D_m} \cdot R_m$ $C = S_m \cdot D_m + D_m \cdot \overline{R_m} + S_m \cdot \overline{R_m}$
AND	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
ANDC	↑	↑	↑	↑	↑	実行結果の対応するビットの値が格納されます。
BAND	—	—	—	—	↑	$C = C' \cdot D_n$
Bcc	—	—	—	—	—	
BCLR	—	—	—	—	—	
BIAND	—	—	—	—	↑	$C = C' \cdot \overline{D_n}$
BILD	—	—	—	—	↑	$C = \overline{D_n}$
BIOR	—	—	—	—	↑	$C = C' + \overline{D_n}$
BIST	—	—	—	—	—	
BIXOR	—	—	—	—	↑	$C = C' \cdot D_n + \overline{C'} \cdot \overline{D_n}$
BLD	—	—	—	—	↑	$C = D_n$
BNOT	—	—	—	—	—	
BOR	—	—	—	—	↑	$C = C' + D_n$
BSET	—	—	—	—	—	
BSR	—	—	—	—	—	
BST	—	—	—	—	—	
BTST	—	—	↑	—	—	$Z = \overline{D_n}$
BXOR	—	—	—	—	↑	$C = C' \cdot \overline{D_n} + \overline{C'} \cdot D_n$
CMP	↑	↑	↑	↑	↑	$H = S_{m-4} \cdot \overline{D_{m-4}} + \overline{D_{m-4}} \cdot R_{m-4} + S_{m-4} \cdot R_{m-4}$ $N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $V = \overline{S_m} \cdot D_m \cdot \overline{R_m} + S_m \cdot \overline{D_m} \cdot R_m$ $C = S_m \cdot \overline{D_m} + \overline{D_m} \cdot R_m + S_m \cdot R_m$
DAA	*	↑	↑	*	↑	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ C : 10進加算のキャリ
DAS	*	↑	↑	*	↑	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ C : 10進減算のポロー
DEC	—	↑	↑	↑	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $V = D_m \cdot \overline{R_m}$

命令	H	N	Z	V	C	定義
DIVXS	—	↑	↑	—	—	$N = S_m \cdot \overline{D_m} + \overline{S_m} \cdot D_m$ $Z = \overline{S_m} \cdot \overline{S_{m-1}} \cdot \dots \cdot \overline{S_0}$
DIVXU	—	↑	↑	—	—	$N = S_m$ $Z = \overline{S_m} \cdot \overline{S_{m-1}} \cdot \dots \cdot \overline{S_0}$
EEPMOV	—	—	—	—	—	
EXTS	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
EXTU	—	0	↑	0	—	$Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
INC	—	↑	↑	↑	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $V = D_m \cdot R_m$
JMP	—	—	—	—	—	
JSR	—	—	—	—	—	
LDC	↑	↑	↑	↑	↑	実行結果の対応するビットの値が格納されます。
MOV	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
MOVFPPE	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
MOVTPPE	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
MULXS	—	↑	↑	—	—	$N = R_{2m}$ $Z = \overline{R_{2m}} \cdot \overline{R_{2m-1}} \cdot \dots \cdot \overline{R_0}$
MULXU	—	—	—	—	—	
NEG	↑	↑	↑	↑	↑	$H = D_m - 4 + R_m - 4$ $N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $V = D_m \cdot R_m$ $C = D_m + R_m$
NOP	—	—	—	—	—	
NOT	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
OR	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
ORC	↑	↑	↑	↑	↑	実行結果の対応するビットの値が格納されます。
POP	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
PUSH	—	↑	↑	0	—	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$
ROTL	—	↑	↑	0	↑	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $C = D_m$
ROTR	—	↑	↑	0	↑	$N = R_m$ $Z = \overline{R_m} \cdot \overline{R_{m-1}} \cdot \dots \cdot \overline{R_0}$ $C = D_0$

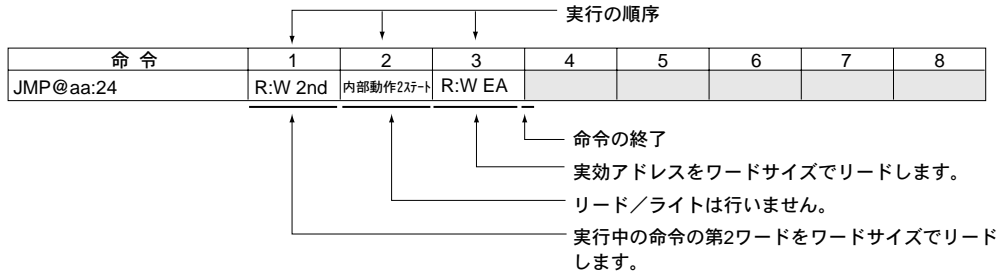
2. 各命令の説明

命令	H	N	Z	V	C	定義
ROTXL	—	↑	↑	0	↑	N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$ C=Dm
ROTXR	—	↑	↑	0	↑	N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$ C=D0
RTS	—	—	—	—	—	
RTE	↑	↑	↑	↑	↑	実行結果の対応するビットの値が格納されます。
SHAL	—	↑	↑	↑	↑	N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$ V= $Dm \cdot \overline{Dm-1} + \overline{Dm} \cdot Dm-1$ C=Dm
SHAR	—	↑	↑	0	↑	N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$ C=D0
SHLL	—	↑	↑	0	↑	N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$ C=Dm
SHLR	—	↑	↑	0	↑	N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$ C=D0
SLEEP	—	—	—	—	—	
STC	—	—	—	—	—	
SUB	↑	↑	↑	↑	↑	H= $Sm-4 \cdot \overline{Dm-4} + \overline{Dm-4} \cdot Rm-4 + Sm-4 \cdot Rm-4$ N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$ V= $\overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ C= $Sm \cdot \overline{Dm} + \overline{Dm} \cdot Rm + Sm \cdot Rm$
SUBS	—	—	—	—	—	
SUBX	↑	↑	↑	↑	↑	H= $Sm-4 \cdot \overline{Dm-4} + \overline{Dm-4} \cdot Rm-4 + Sm-4 \cdot Rm-4$ N=Rm Z= $Z' \cdot \overline{Rm} \cdot \dots \cdot \overline{R0}$ V= $\overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ C= $Sm \cdot \overline{Dm} + \overline{Dm} \cdot Rm + Sm \cdot Rm$
TRAPA	—	—	—	—	—	
XOR	—	↑	↑	0	—	N=Rm Z= $\overline{Rm} \cdot \overline{Rm-1} \cdot \dots \cdot \overline{R0}$
XORC	↑	↑	↑	↑	↑	実行結果の対応するビットの値が格納されます。

2.8 命令実行中のバス状態

H8/300H CPUの個々の命令についての実行状態を表2.10に示します。実行状態に必要なステート数に関しては、「表2.7 実行状態（サイクル）に要するステート数」を参照してください。

《表の見方》



《記号説明》

R:B	バイトサイズリードを行います。
R:W	ワードサイズリードを行います。
W:B	バイトサイズライトを行います。
W:W	ワードサイズライトを行います。
2nd	第2ワード（第3・第4バイト）のアドレスです。
3rd	第3ワード（第5・第6バイト）のアドレスです。
4th	第4ワード（第7・第8バイト）のアドレスです。
5th	第5ワード（第9・第10バイト）のアドレスです。
NEXT	実行中の命令の直後の命令の先頭アドレスです。
EA	実行アドレスです。
VEC	ベクタアドレスです。

8ビットバス・3ステートアクセス・ウェイトなしの場合、上記命令実行中のアドレスバス、 \overline{RD} 、 \overline{WR} （ \overline{HWR} または \overline{LWR} ）のタイミングを図2.1に示します。

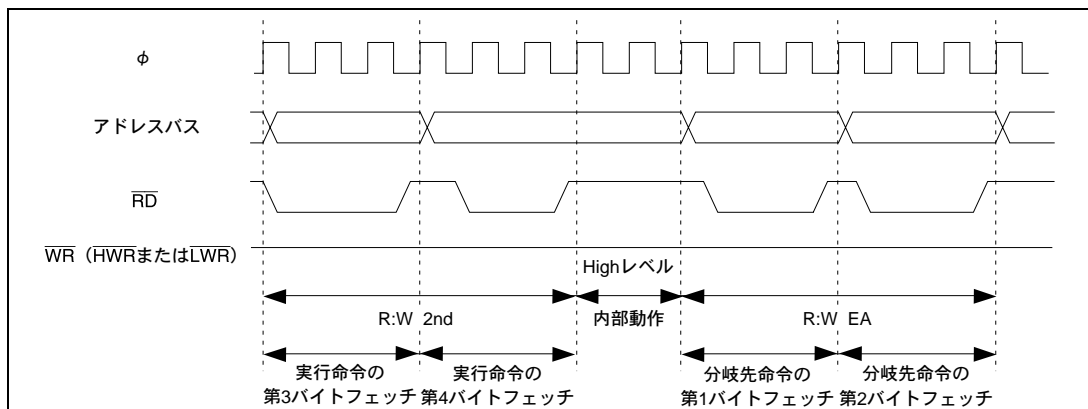


図2.1 アドレスバス、 \overline{RD} 、 \overline{WR} (\overline{HWR} または \overline{LWR})のタイミング
 (8ビットバス・3ステートアクセス・ウェイトなしの場合)

2. 各命令の説明

表 2.10 命令の実行状態

命令	1	2	3	4	5	6	7	8
ADD.B #xx:8,Rd	R:W NEXT							
ADD.B Rs,Rd	R:W NEXT							
ADD.W #xx:16, Rd	R:W 2nd	R:W NEXT						
ADD.W Rs,Rd	R:W NEXT							
ADD.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT					
ADD.L ERs,ERd	R:W NEXT							
ADDS#1/2/4,ERd	R:W NEXT							
ADDX #XX:8,Rd	R:W NEXT							
ADDX Rs,Rd	R:W NEXT							
AND.B #xx:8,Rd	R:W NEXT							
AND.B Rs,Rd	R:W NEXT							
AND.W #xx:16,Rd	R:W 2nd	R:W NEXT						
AND.W Rs,Rd	R:W NEXT							
AND.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT					
AND.L ERs,ERd	R:W 2nd	R:W NEXT						
ANDC #xx:8,CCR	R:W NEXT							
BAND #xx:3,Rd	R:W NEXT							
BAND #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT					
BAND #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT					
BRA d:8 (BT d:8)	R:W NEXT	R:W EA						
BRN d:8 (BF d:8)	R:W NEXT	R:W EA						
BHI d:8	R:W NEXT	R:W EA						
BLS d:8	R:W NEXT	R:W EA						
BCC d:8 (BHS d:8)	R:W NEXT	R:W EA						
BCS d:8 (BLO d:8)	R:W NEXT	R:W EA						
BNE d:8	R:W NEXT	R:W EA						
BEQ d:8	R:W NEXT	R:W EA						
BVC d:8	R:W NEXT	R:W EA						
BVS d:8	R:W NEXT	R:W EA						
BPL d:8	R:W NEXT	R:W EA						
BMI d:8	R:W NEXT	R:W EA						
BGE d:8	R:W NEXT	R:W EA						
BLT d:8	R:W NEXT	R:W EA						
BGT d:8	R:W NEXT	R:W EA						
BLE d:8	R:W NEXT	R:W EA						
BRA d:16 (BT d:16)	R:W 2nd	内部動作 2 ステート	R:W EA					
BRN d:16 (BF d:16)	R:W 2nd	内部動作 2 ステート	R:W EA					
BHI d:16	R:W 2nd	内部動作 2 ステート	R:W EA					
BLS d:16	R:W 2nd	内部動作 2 ステート	R:W EA					

2. 各命令の説明

命令	1	2	3	4	5	6	7	8
BCC d;16 (BHS d;16)	R:W 2nd	内部動作 2 ステート	R:W EA					
BCS d;16 (BLO d;16)	R:W 2nd	内部動作 2 ステート	R:W EA					
BNE d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BEQ d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BVC d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BVS d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BPL d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BMI d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BGE d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BLT d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BGT d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BLE d;16	R:W 2nd	内部動作 2 ステート	R:W EA					
BCLR #xx:3,Rd	R:W NEXT							
BCLR #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BCLR #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BCLR Rn,Rd	R:W NEXT							
BCLR Rn,@ERd	R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BCLR Rn,@aa:8	R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BIAND #xx:3,Rd	R:W NEXT							
BIAND #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT					
BIAND #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT					
BILD #xx:3,Rd	R:W NEXT							
BILD #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT					
BILD #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT					
BIOR #xx:8,Rd	R:W NEXT							
BIOR #xx:8,@ERd	R:W 2nd	R:B EA	R:W NEXT					
BIOR #xx:8,@aa:8	R:W 2nd	R:B EA	R:W NEXT					
BIST #xx:3,Rd	R:W NEXT							
BIST #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BIST #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BIXOR #xx:3,Rd	R:W NEXT							
BIXOR #xx:3,@ERd	R:W 2nd	R:B EA	R:W NEXT					
BIXOR #xx:3,@aa:8	R:W 2nd	R:B EA	R:W NEXT					
BLD #xx:3,Rd	R:W NEXT							

2. 各命令の説明

命令		1	2	3	4	5	6	7	8
BLD #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT					
BLD #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT					
BNOT #xx:3,Rd		R:W NEXT							
BNOT #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BNOT #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BNOT Rn,Rd		R:W NEXT							
BNOT Rn,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BNOT Rn,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BOR #xx:3,Rd		R:W NEXT							
BOR #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT					
BOR #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT					
BSET #xx:3,Rd		R:W NEXT							
BSET #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BSET #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BSET Rn,Rd		R:W NEXT							
BSET Rn,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BSET Rn,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BSR d:8	ノーマル	R:W NEXT	R:W EA	W:W スタック					
	アドバ ンスト	R:W NEXT	R:W EA	W:W スタック(H)	W:W スタック(L)				
BSR d:16	ノーマル	R:W 2nd	内部動作 2ステート	R:W EA	W:W スタック				
	アドバ ンスト	R:W 2nd	内部動作 2ステート	R:W EA	W:W スタック(H)	W:W スタック(L)			
BST #xx:3,Rd		R:W NEXT							
BST #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BST #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT	W:B EA				
BTST #xx:3,Rd		R:W NEXT							
BTST #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT					
BTST #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT					
BTST Rn,Rd		R:W NEXT							
BTST Rn,@ERd		R:W 2nd	R:B EA	R:W NEXT					
BTST Rn,@aa:8		R:W 2nd	R:B EA	R:W NEXT					
BXOR #xx:3,Rd		R:W NEXT							
BXOR #xx:3,@ERd		R:W 2nd	R:B EA	R:W NEXT					
BXOR #xx:3,@aa:8		R:W 2nd	R:B EA	R:W NEXT					
CMP.B #xx:8,Rd		R:W NEXT							
CMP.B Rs,Rd		R:W NEXT							
CMP.W #xx:16,Rd		R:W 2nd	R:W NEXT						
CMP.W Rs,Rd		R:W NEXT							
CMP.L #xx:32,ERd		R:W 2nd	R:W 3rd	R:W NEXT					
CMP.L ERs,ERd		R:W NEXT							
DAA Rd		R:W NEXT							

2. 各命令の説明

命令	1	2	3	4	5	6	7	8
DAS Rd	R:W NEXT							
DEC.B Rd	R:W NEXT							
DEC.W #1/2, Rd	R:W NEXT							
DEC.L #1/2,ERd	R:W NEXT							
DIVXS.B Rs,Rd	R:W 2nd	R:W NEXT	内部動作 12 ステート					
DIVXS.W Rs,ERd	R:W 2nd	R:W NEXT	内部動作 20 ステート					
DIVXU.B Rs,Rd	R:W NEXT	内部動作 12 ステート						
DIVXU.W Rs,ERd	R:W NEXT	内部動作 20 ステート						
EPEMOV.B	R:W 2nd	R:B EAs *1	R:B EAd *1	R:B EAs *2	R:B EAd *2	R:W NEXT		
EPEMOV.W	R:W 2nd	R:B EAs *1	R:B EAd *1	R:B EAs *2	R:B EAd *2	R:W NEXT		
EXTS.W Rd	R:W NEXT			← n 回繰り返す *2→				
EXTS.L ERd	R:W NEXT							
EXTU.W Rd	R:W NEXT							
EXTU.L ERd	R:W NEXT							
INC.B Rd	R:W NEXT							
INC.W #1/2,Rd	R:W NEXT							
INC.L #1/2,ERd	R:W NEXT							
JMP @ERn	R:W NEXT	R:W EA						
JMP @aa:24	R:W 2nd	内部動作 2 ステート	R:W EA					
JMP@@ aa:8	ノーマル	R:W NEXT	R:W aa:8	内部動作 2 ステート	R:W EA			
	アドバ ンスト	R:W NEXT	R:W aa:8	R:W aa:8	内部動作 2 ステート	R:W EA		
JSR@ER n	ノーマル	R:W NEXT	R:W EA	W:W スタック				
	アドバ ンスト	R:W NEXT	R:W EA	W:W スタック(H)	W:W スタック(L)			
JSR@aa: 24	ノーマル	R:W 2nd	内部動作 2 ステート	R:W EA	W:W スタック			
	アドバ ンスト	R:W 2nd	内部動作 2 ステート	R:W EA	W:W スタック(H)	W:W スタック(L)		
JSR@@ aa:8	ノーマル	R:W NEXT	R:W aa:8	W:W スタック	R:W EA			
	アドバ ンスト	R:W NEXT	R:W aa:8	R:W aa:8	W:W スタック(H)	W:W スタック(L)	R:W EA	
LDC #xx8,CCR	R:W NEXT							
LDC Rs,CCR	R:W NEXT							
LDC@ERs,CCR	R:W 2nd	R:W NEXT	R:W EA					
LDC@(d:16,ERs),CC R	R:W 2nd	R:W 3rd	R:W NEXT	R:W EA				
LDC@(d:24,ERs),CC R	R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	R:W EA		
LDC@ERs+,CCR	R:W 2nd	R:W NEXT	内部動作 2 ステート	R:W EA				
LDC@aa:16,CCR	R:W 2nd	R:W 3rd	R:W NEXT	R:W EA				

2. 各命令の説明

命令	1	2	3	4	5	6	7	8
LDC@aa:24,CCR	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:W EA			
MOV.B #xx:8,Rd	R:W NEXT							
MOV.B Rs,Rd	R:W NEXT							
MOV.B @ERs,Rd	R:W NEXT	R:B EA						
MOV.B @(d:16,ERs),Rd	R:W 2nd	R:W NEXT	R:W EA					
MOV.B @(d:24,ERs),Rd	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:B EA			
MOV.B @ERs+,Rd	R:W NEXT	内部動作 2 ステート	R:B EA					
MOV.B @aa:8,Rd	R:W NEXT	R:B EA						
MOV.B @aa:16,Rd	R:W 2nd	R:W NEXT	R:B EA					
MOV.B @aa:24,Rd	R:W 2nd	R:W 3rd	R:W NEXT	R:B EA				
MOV.B Rs,@ERd	R:W NEXT	W:B EA						
MOV.B Rs,@(d:16,ERd)	R:W 2nd	R:W NEXT	w:B EA					
MOV.B Rs,@(d:24,ERd)	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:B EA			
MOV.B Rs,@-ERd	R:W NEXT	内部動作 2 ステート	W:B EA					
MOV.B Rs,@aa:8	R:W NEXT	W:B EA						
MOV.B Rs,@aa:16	R:W 2nd	R:W NEXT	W:B EA					
MOV.B Rs,@aa:24	R:W 2nd	R:W 3rd	R:W NEXT	W:B EA				
MOV.W #xx:16,Rd	R:W 2nd	R:W NEXT						
MOV.W Rs,Rd	R:W NEXT							
MOV.W @ERs,Rd	R:W NEXT	R:W EA						
MOV.W @(d:16,ERs),Rd	R:W 2nd	R:W NEXT	R:W EA					
MOV.W @(d:24,ERs),Rd	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:W EA			
MOV.W @ERs+,Rd	R:W NEXT	内部動作 2 ステート	R:W EA					
MOV.W @aa:16,Rd	R:W 2nd	R:W NEXT	R:W EA					
MOV.W @aa:24,Rd	R:W 2nd	R:W 3rd	R:W NEXT	R:B EA				
MOV.W Rs,@ERd	R:W NEXT	W:W EA						
MOV.W Rs,@(d:16,ERd)	R:W 2nd	R:W NEXT	W:W EA					
MOV.W Rs,@(d:24,ERd)	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:W EA			
MOV.W Rs,@-ERd	R:W NEXT	内部動作 2 ステート	W:W EA					
MOV.W Rs,@aa:16	R:W 2nd	R:W NEXT	W:W EA					
MOV.W Rs,@aa:24	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA				
MOV.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT					
MOV.L ERs,ERd	R:W NEXT							
MOV.L @ERs,ERd	R:W 2nd	R:W NEXT	R:W EA	R:W EA+2				

2. 各命令の説明

命令	1	2	3	4	5	6	7	8
MOV.L @(d:16,ERs),ERd	R:W 2nd	R:W 3rd	R:W NEXT	R:W EA	R:W EA+2			
MOV.L @(d:24,ERs),ERd	R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	R:W EA	R:W EA+2	
MOV.L @ERs+,ERd	R:W 2nd	R:W NEXT	内部動作 2 ステート	R:W EA	R:W EA+2			
MOV.L @aa:16,ERd	R:W 2nd	R:W 3rd	R:W NEXT	R:W EA	R:W EA+2			
MOV.L @aa:24,ERd	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	R:W EA	R:W EA+2		
MOV.L ERs,@ERd	R:W 2nd	R:W NEXT	W:W EA	W:W EA+2				
MOV.L ERs,@(d:16,ERd)	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA	W:W EA+2			
MOV.L ERs,@(d:24,ERd)	R:W 2nd	R:W 3rd	R:W 4th	R:W 5th	R:W NEXT	W:W EA	W:W EA+2	
MOV.L ERs,@-ERd	R:W 2nd	R:W NEXT	内部動作 2 ステート	W:W EA	W:W EA+2			
MOV.L ERs,@aa:16	R:W 2nd	R:W 3rd	R:W NEXT	W:W EA	W:W EA+2			
MOV.L ERs,@aa:24	R:W 2nd	R:W 3rd	R:W 4th	R:W NEXT	W:W EA	W:W EA+2		
MOVFEPP @aa:16,Rd	R:W 2nd	内部動作 2 ステート	R:W *3 EA					
MOVTPE Rs,@aa:16	R:W 2nd	内部動作 2 ステート	R:W *3 EA					
MULXS.B Rs,Rd	R:W 2nd	R:W NEXT	内部動作 12 ステート					
MULXS.W Rs,ERd	R:W 2nd	R:W NEXT	内部動作 20 ステート					
MULXU.B Rs,Rd	R:W NEXT	内部動作 12 ステート						
MULXU.W Rs,ERd	R:W NEXT	内部動作 20 ステート						
NEG.B Rd	R:W NEXT							
NEG.W Rd	R:W NEXT							
NEG.L ERd	R:W NEXT							
NOP	R:W NEXT							
NOT.B Rd	R:W NEXT							
NOT.W Rd	R:W NEXT							
NOT.L ERd	R:W NEXT							
OR.B #xx:8,Rd	R:W NEXT							
OR.B Rs,Rd	R:W NEXT							
OR.W #xx:16,Rd	R:W 2nd	R:W NEXT						
OR.W Rs,Rd	R:W NEXT							
OR.L #xx:32,ERd	R:W 2nd	R:W 3rd	R:W NEXT					
OR.L ERs,ERd	R:W 2nd	R:W NEXT						
ORC #xx:8,CCR	R:W NEXT							
POP.W Rn	R:W NEXT	内部動作 2 ステート	R:W スタック					
POP.L ERn	R:W 2nd	R:W NEXT	内部動作 2 ステート	R:W スタック(H)	R:W スタック(L)			
PUSH.W Rn	R:W NEXT	内部動作 2 ステート	W:W スタック					

2. 各命令の説明

命令	1	2	3	4	5	6	7	8
PUSH.L ERn	R:W 2 nd	R:W NEXT	内部動作 2 ステート	W:W スタック(L)	W:W スタック(H)			
ROTL.B Rd	R:W NEXT							
ROTL.W Rd	R:W NEXT							
ROTL.L ERd	R:W NEXT							
ROTR.B Rd	R:W NEXT							
ROTR.W Rd	R:W NEXT							
ROTR.L ERd	R:W NEXT							
ROTXL.B Rd	R:W NEXT							
ROTXL.W Rd	R:W NEXT							
ROTXL.L ERd	R:W NEXT							
ROTXR.B Rd	R:W NEXT							
ROTXR.W Rd	R:W NEXT							
ROTXR.L ERd	R:W NEXT							
RTE	R:W NEXT	R:W スタック(H)	R:W スタック(L)	内部動作 2 ステート	R:W * ⁴			
RTS	ノーマル	R:W NEXT	R:W スタック	内部動作 2 ステート	R:W * ⁴			
	アドバンス	R:W NEXT	R:W スタック(H)	R:W スタック(L)	内部動作 2 ステート	R:W * ⁴		
SHAL.B Rd	R:W NEXT							
SHAL.W Rd	R:W NEXT							
SHAL.L ERd	R:W NEXT							
SHAR.B Rd	R:W NEXT							
SHAR.W Rd	R:W NEXT							
SHAR.L ERd	R:W NEXT							
SHLL.B Rd	R:W NEXT							
SHLL.W Rd	R:W NEXT							
SHLL.L ERd	R:W NEXT							
SHLR.B Rd	R:W NEXT							
SHLR.W Rd	R:W NEXT							
SHLR.L ERd	R:W NEXT							
SLEEP	R:W NEXT							
STC CCR,Rd	R:W NEXT							
STC CCR,@ERd	R:W 2 nd	R:W NEXT	W:W EA					
STC CCR,@(d:16,ERd)	R:W 2 nd	R:W 3 rd	R:W NEXT	W:W EA				
STC CCR,@(d:24,ERd)	R:W 2 nd	R:W 3 rd	R:W 4 th	R:W 5 th	R:W NEXT	W:W EA		
STC CCR,@-ERd	R:W 2 nd	R:W NEXT	内部動作 2 ステート	W:W EA				
STC CCR,@aa:16	R:W 2 nd	R:W 3 rd	R:W NEXT	W:W EA				
STC CCR,@aa:24	R:W 2 nd	R:W 3 rd	R:W 4 th	R:W NEXT	W:W EA			
SUB.B Rs,Rd	R:W NEXT							
SUB.W #xx:16,Rd	R:W 2 nd	R:W NEXT						

2. 各命令の説明

命令		1	2	3	4	5	6	7	8
SUB.W Rs,Rd		R:W NEXT							
SUB.L #xx:32,ERd		R:W 2nd	R:W 3rd	R:W NEXT					
SUB.L ERs,ERd		R:W NEXT							
SUBS #1/2/4,ERd		R:W NEXT							
SUBX #xx:8,Rd		R:W NEXT							
SUBX Rs,Rd		R:W NEXT							
TRAPA #x:2	ノーマル	R:W NEXT	内部動作 2ステート	W:W スタック(L)	W:W スタック(H)	R:W VEC	内部動作 2ステート	R:W * ⁷	
	アドバンス	R:W NEXT	内部動作 2ステート	W:W スタック(L)	W:W スタック(H)	R:W VEC	R:W VEC+2	内部動作 2ステート	R:W * ⁷
XOR.B #xx8,Rd		R:W NEXT							
XOR.B Rs,Rd		R:W NEXT							
XOR.W #xx:16,Rd		R:W 2nd	R:W NEXT						
XOR.W Rs,Rd		R:W NEXT							
XOR.L #xx:32,ERd		R:W 2nd	R:W 3rd	R:W NEXT					
XOR.L ERs,ERd		R:W 2nd	R:W NEXT						
XORC #xx:8,CCR		R:W NEXT							
リセット 例外処理	ノーマル	R:W VEC	内部動作 2ステート	R:W * ⁵					
	アドバンス	R:W VEC	R:W VEC+2	内部動作 2ステート	R:W * ⁵				
割り込み 例外処理	ノーマル	R:W * ⁶	内部動作 2ステート	W:W スタック(L)	W:W スタック(H)	R:W VEC	内部動作 2ステート	R:W * ⁷	
	アドバンス	R:W * ⁶	内部動作 2ステート	W:W スタック(L)	W:W スタック(H)	R:W VEC	R:W VEC+2	内部動作 2ステート	R:W * ⁷

- 【注】 *1 EAs は ER5、EAd は ER6 の内容です。
- *2 EAs は ER5、EAd は ER6 の内容で、実行後それぞれ 1 が加算されます。
また、n は R4L または R4 の初期値であり、n=0 のときこれらは実行されません。
- *3 バイトサイズリード/ライトに必要なステート数は 9~16 です。
- *4 リターン後の先頭アドレスです。
- *5 プログラムのスタートアドレスです。
- *6 プリフェッチアドレスです。退避される PC に 2 を加算したアドレスです。
また、スリープモード、ソフトウェアスタンバイモードからの復帰時にはリード動作は行われず、内部動作となります。
- *7 割り込み処理ルーチンの先頭アドレスです。

3. 処理状態

3.1 概要

H8/300H CPU の処理状態には、プログラム実行状態、例外処理状態、低消費電力状態、リセット状態、およびバス権解放状態の 5 種類があります。さらに、低消費電力状態には、スリープモード、ソフトウェアスタンバイモード、およびハードウェアスタンバイモードがあります。処理状態の分類を図 3.1 に、各状態間の遷移を図 3.2 に示します。なお、詳細は、当該 LSI のハードウェアマニュアルを参照してください。

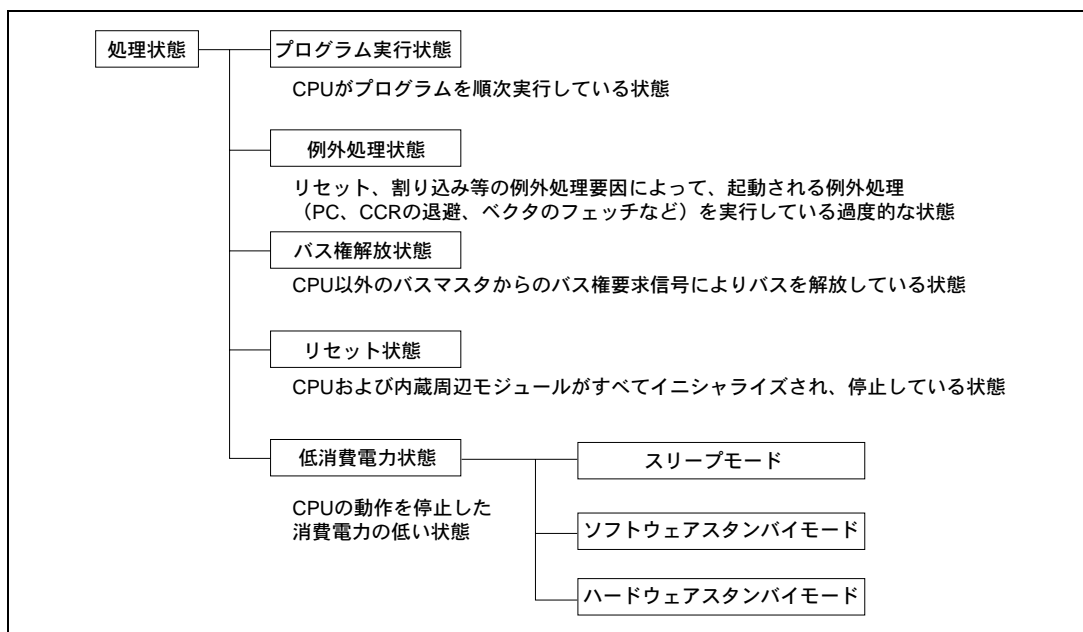


図 3.1 処理状態の分類

3. 処理状態

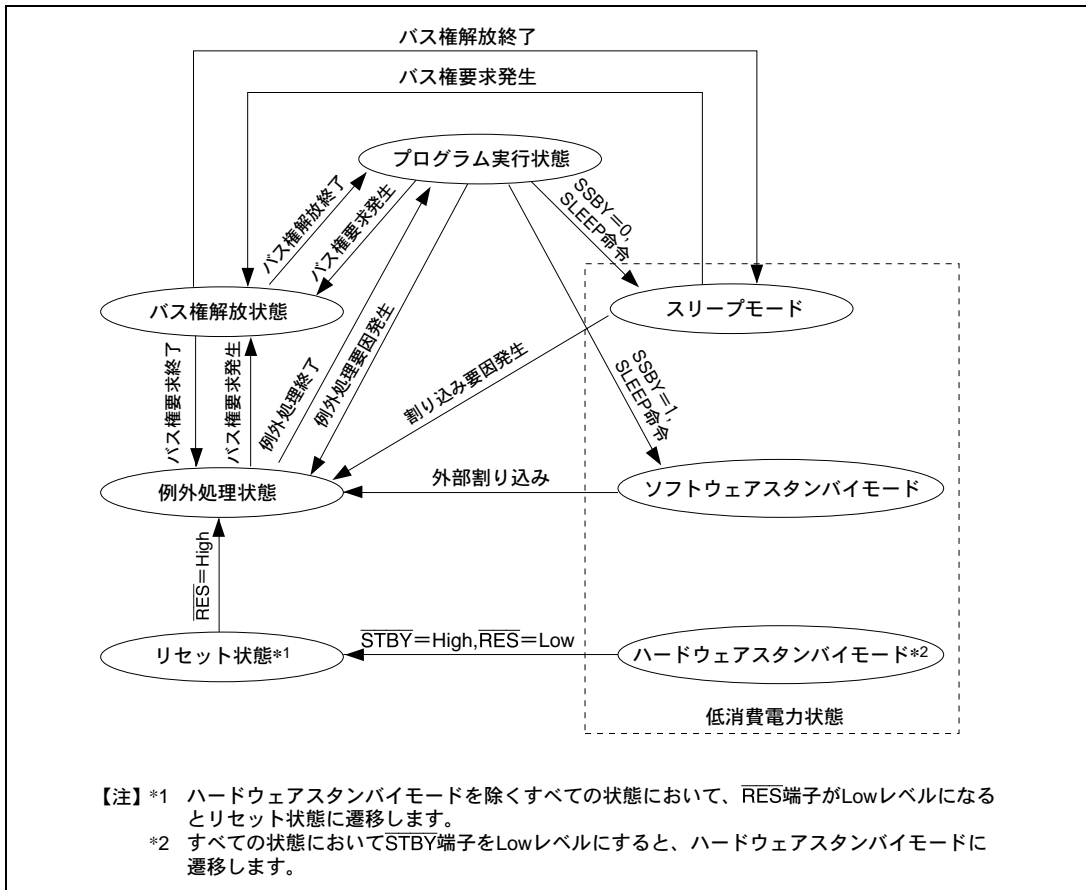


図 3.2 状態遷移図

3.2 プログラム実行状態

CPU がプログラムを順次実行している状態です。

3.3 例外処理状態

リセット、割り込み、またはトラップ命令の例外処理要因によって起動され、CPU が通常の処理状態の流れを変え、例外処理ベクタテーブルからスタートアドレスを取り出し、その番地に分岐する過度的な状態です。割り込みおよびトラップ命令例外処理では、SP (ER7) を参照して、PC および CCR の退避を行います。

3.3.1 例外処理の種類と優先度

例外処理には、リセット、割り込み、およびトラップ命令があります。表 3.1 に、例外処理の種類と優先度を示します。トラップ命令例外処理は、プログラム実行状態で常に受け付けられます。

表 3.1 例外処理の種類と優先度

優先度	例外処理要因	例外処理検出タイミング	例外処理開始タイミング
高 ↑ 低	リセット	クロック同期	RES 端子が Low レベルから High レベルに変化すると、ただちに例外処理を開始します。
	割り込み	命令の実行終了時*	割り込み要求が発生すると、命令の実行終了時または例外処理の終了時に例外処理を開始します。
	トラップ命令	TRAPA 命令実行時	トラップ (TRAPA) 命令を実行すると、例外処理を開始します。

【注】*ANDC、ORC、XORC、LDC 命令の実行終了時点、またはリセット例外処理の終了時点では、割り込み要因の検出を行いません。

例外処理要因は、図 3.3 に示すように分類されます。

例外処理要因とベクタ番号ならびにベクタアドレスの詳細は、当該 LSI のハードウェアマニュアルを参照してください。

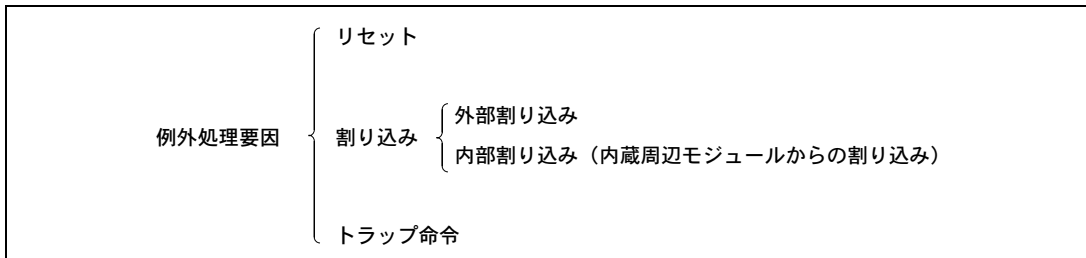


図 3.3 例外処理要因の分類

3.3.2 例外処理の動作

(1) リセット例外処理の動作

リセット例外処理は、最も優先度の高い例外処理です。RES 端子を Low レベルにしてリセット状態にした後、RES 端子を High レベルにすると、リセット例外処理が起動されます。リセット例外処理が起動されると、CPU は、例外処理ベクタテーブルからスタートアドレスを取り出し、その番地からプログラムの実行を開始します。リセット例外処理実行中、および終了後は、NMI を含めたすべての割り込みが禁止されます。

(2) 割り込み例外処理およびトラップ命令例外処理の動作

これらの例外処理が起動されると、CPU は SP (ER7) を参照して PC と CCR をスタックに退避します。次に、割り込みマスクビットを 1 にセットし、例外処理ベクタテーブルからスタートアドレスを取り出して分岐します。

退避される PC の値、ベクタテーブルより取り出されるスタートアドレスは、ノーマルモードでは 16 ビット、アドバンスドモードでは 24 ビットとなります。

例外処理終了後のスタックの構造を図 3.4 に示します。

3. 処理状態

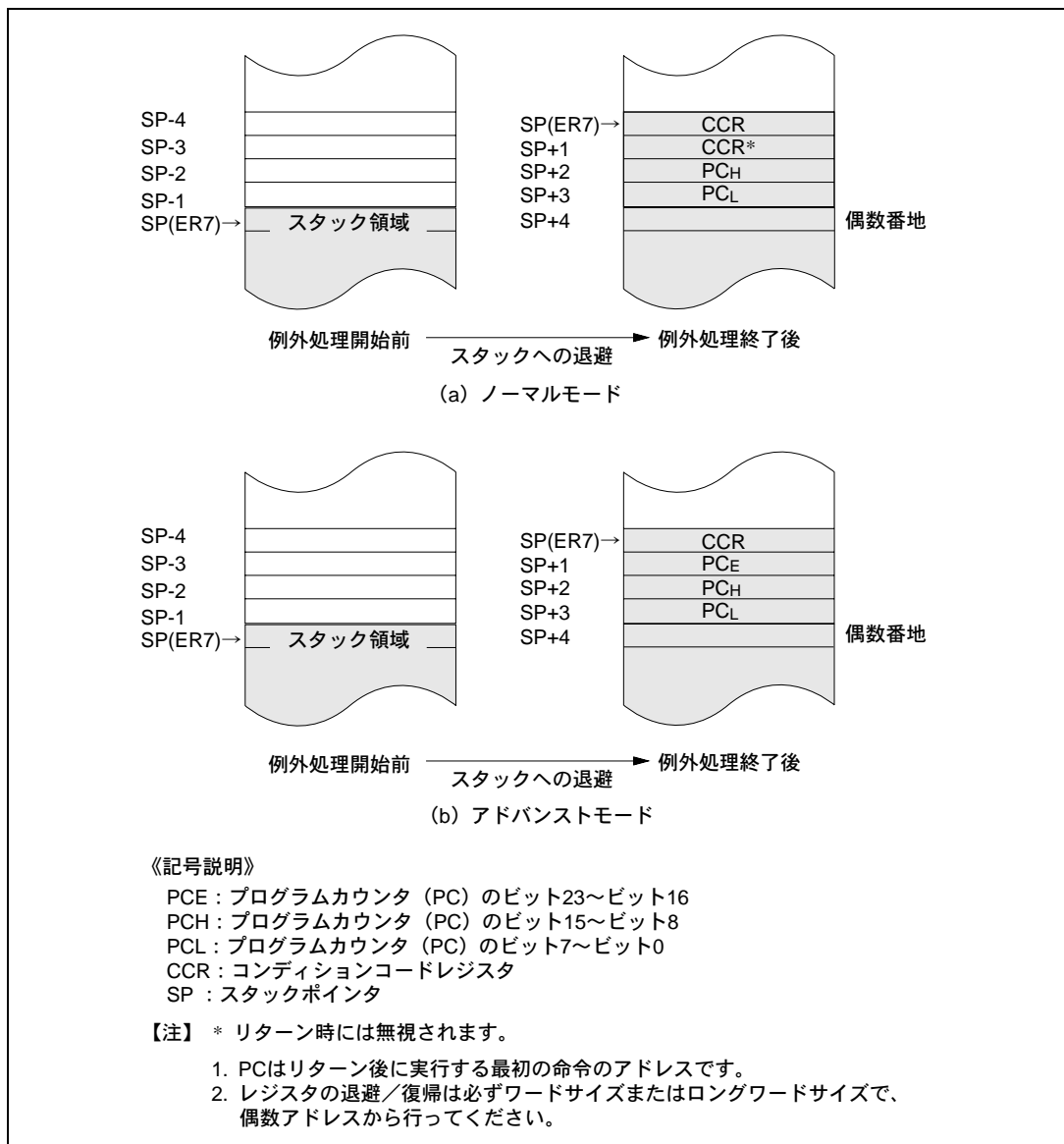


図 3.4 例外処理終了後のスタック状態

3.4 バス権解放状態

CPU 以外のバスマスタによるバス権要求に対して、バス権を解放した状態です。バス権解放状態では、CPU は内部動作を除き停止します。また、割り込みも受け付けられません。詳細は当該 LSI のハードウェアマニュアルを参照してください。

3.5 リセット状態

$\overline{\text{RES}}$ 端子が Low レベルになると、実行中の処理はすべて中止され、CPU はリセット状態になります。リセットによって CCR の I ビットが 1 にセットされます。リセット状態ではすべての割り込みが禁止されます。 $\overline{\text{RES}}$ 端子を Low レベルから High レベルにすると、リセット例外処理が開始されます。詳細は当該 LSI のハードウェアマニュアルを参照してください。

3.6 低消費電力状態

低消費電力状態は CPU の動作を停止して、消費電力を下げる状態です。スリープモード、ソフトウェアスタンバイモード、ハードウェアスタンバイモードがあります。詳細は当該 LSI のハードウェアマニュアルを参照してください。

3.6.1 スリープモード

スリープモードは、SSBY (ソフトウェアスタンバイ) ビットを 0 にクリアした状態で、SLEEP 命令を実行することによって遷移するモードです。

CPU の動作は SLEEP 命令実行直後で停止します。CPU の内部レジスタの内容は保持されます。

3.6.2 ソフトウェアスタンバイモード

ソフトウェアスタンバイモードは、SSBY ビットを 1 にセットした状態で、SLEEP 命令を実行することによって遷移するモードです。

CPU およびクロックをはじめ内蔵周辺モジュールのすべての動作が停止します。内蔵周辺モジュールはリセット状態になりますが、規定の電圧が与えられている限り CPU の内部レジスタの内容および内蔵 RAM の内容は保持されます。また、I/O ポートの状態も保持されます。

3.6.3 ハードウェアスタンバイモード

ハードウェアスタンバイモードは、 $\overline{\text{STBY}}$ 端子を Low レベルにすることによって遷移するモードです。

ソフトウェアスタンバイモードと同様に、CPU およびすべてのクロックは停止し、内蔵周辺モジュールはリセット状態になりますが、規定の電圧が与えられている限り CPU の内蔵 RAM の内容は保持されます。

3. 处理状态

4. 基本動作タイミング

4.1 概要

H8/300H CPU は、システムクロック (ϕ) を基準に動作しています。 ϕ の立ち上がりから次の立ち上がりまでの 1 単位をステートと呼びます。メモリサイクルまたはバスサイクルは、2 または 3 ステートで構成され、内蔵メモリ、内蔵周辺モジュール、または外部アドレス空間によってそれぞれ異なるアクセスを行います。詳細は当該 LSI のハードウェアマニュアルを参照してください。

4.2 内蔵メモリ (RAM、ROM)

内蔵メモリのアクセスは 2 ステートアクセスを行います。このとき、データバス幅は 16 ビットで、バイトおよびワードサイズアクセスが可能です。内蔵メモリアクセスサイクルを図 4.1 に、端子状態を図 4.2 に示します。

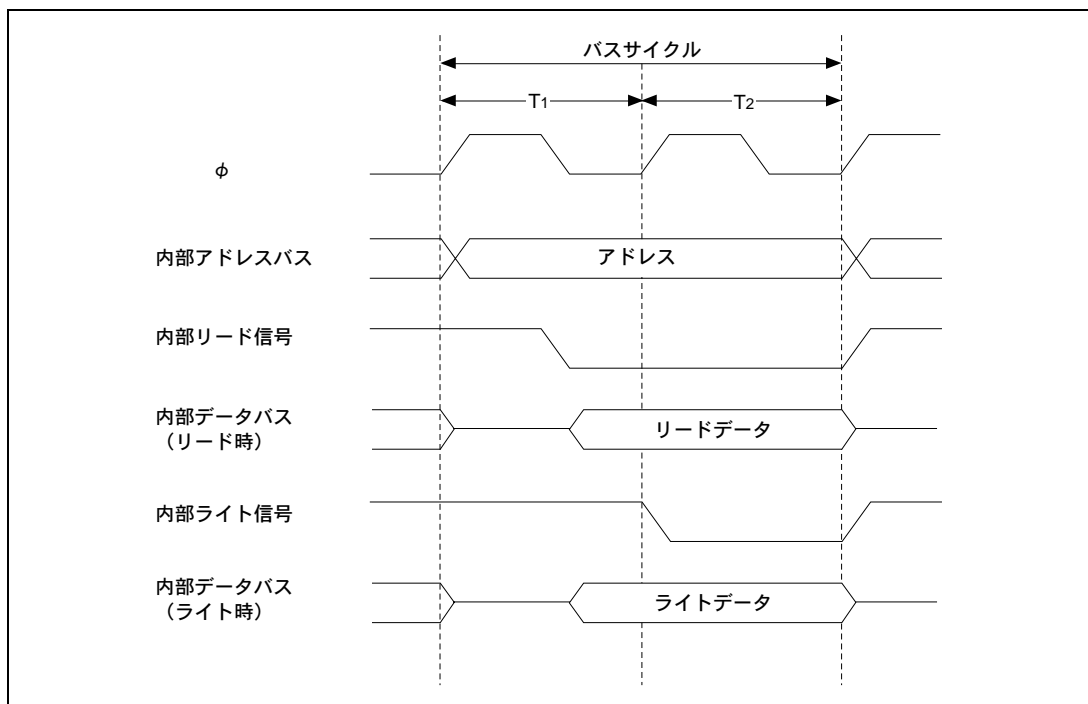


図 4.1 内蔵メモリアクセスサイクル

4. 基本動作タイミング

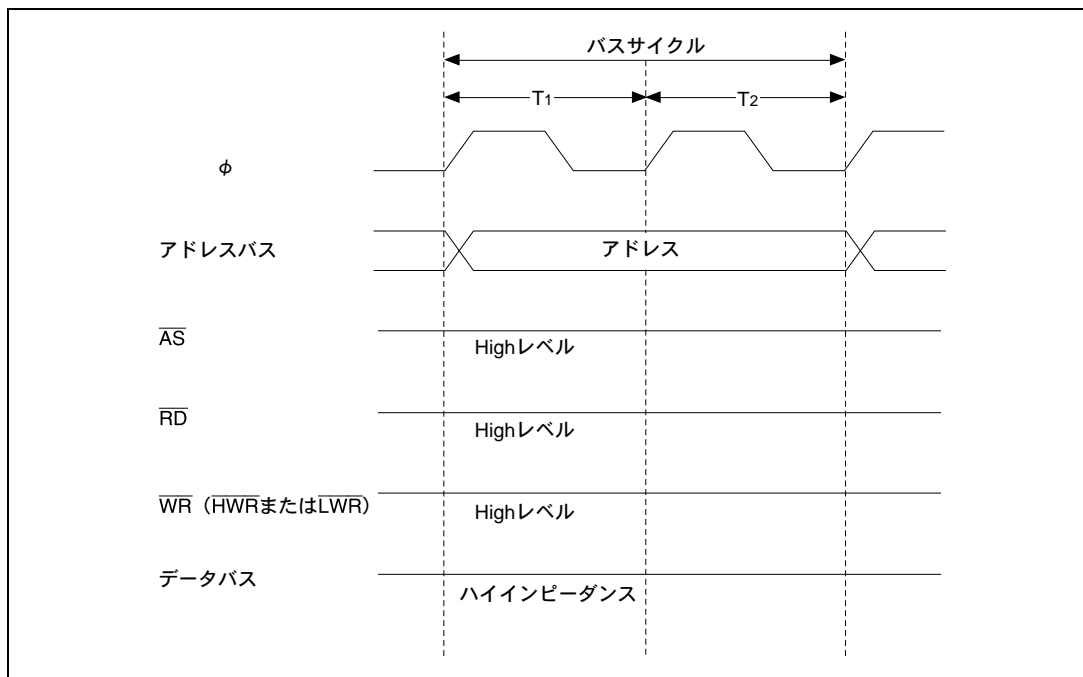


図 4.2 内蔵メモリアクセス時の端子状態

4.3 内蔵周辺モジュールアクセスタイミング

内蔵周辺モジュールのアクセスは3ステートで行われます。このとき、データバス幅は8ビットまたは16ビットで内部I/Oレジスタにより異なります。内蔵周辺モジュールアクセスタイミングを図4.3、端子状態を図4.4に示します。

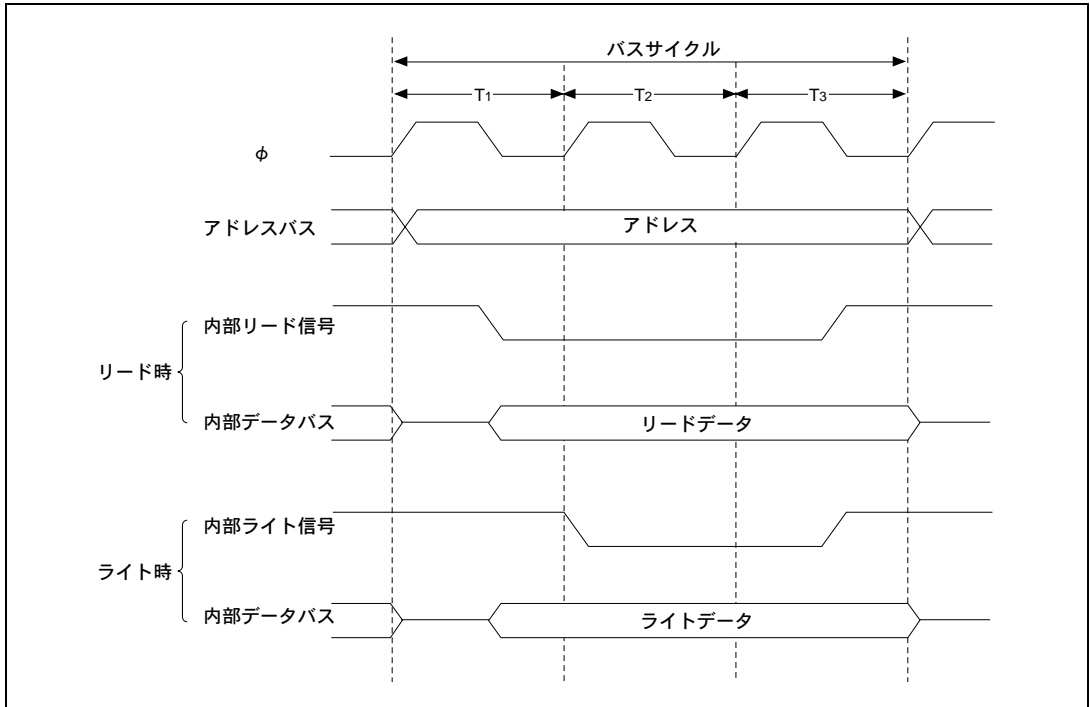


図 4.3 内蔵周辺モジュールアクセスサイクル

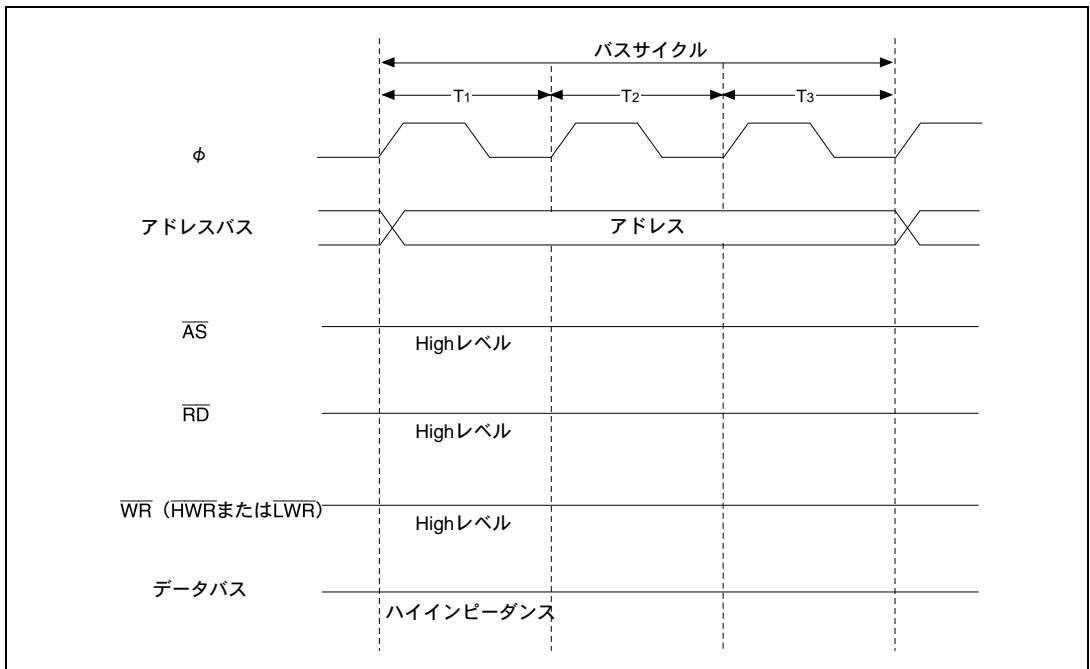


図 4.4 内蔵周辺モジュールアクセス時の端子状態

4.4 外部アドレス空間アクセスタイミング

外部アドレス空間のアクセスを行うときのデータバス幅は8ビットまたは16ビット、バスサイクルは2ステートまたは3ステートです。図4.5に2ステートアクセスおよび3ステートアクセスのリードタイミングを、図4.6に2ステートアクセスおよび3ステートアクセスのライトタイミングを示します。3ステートアクセスではウェイトステートを挿入することができます。詳細は当該LSIのハードウェアマニュアルを参照してください。

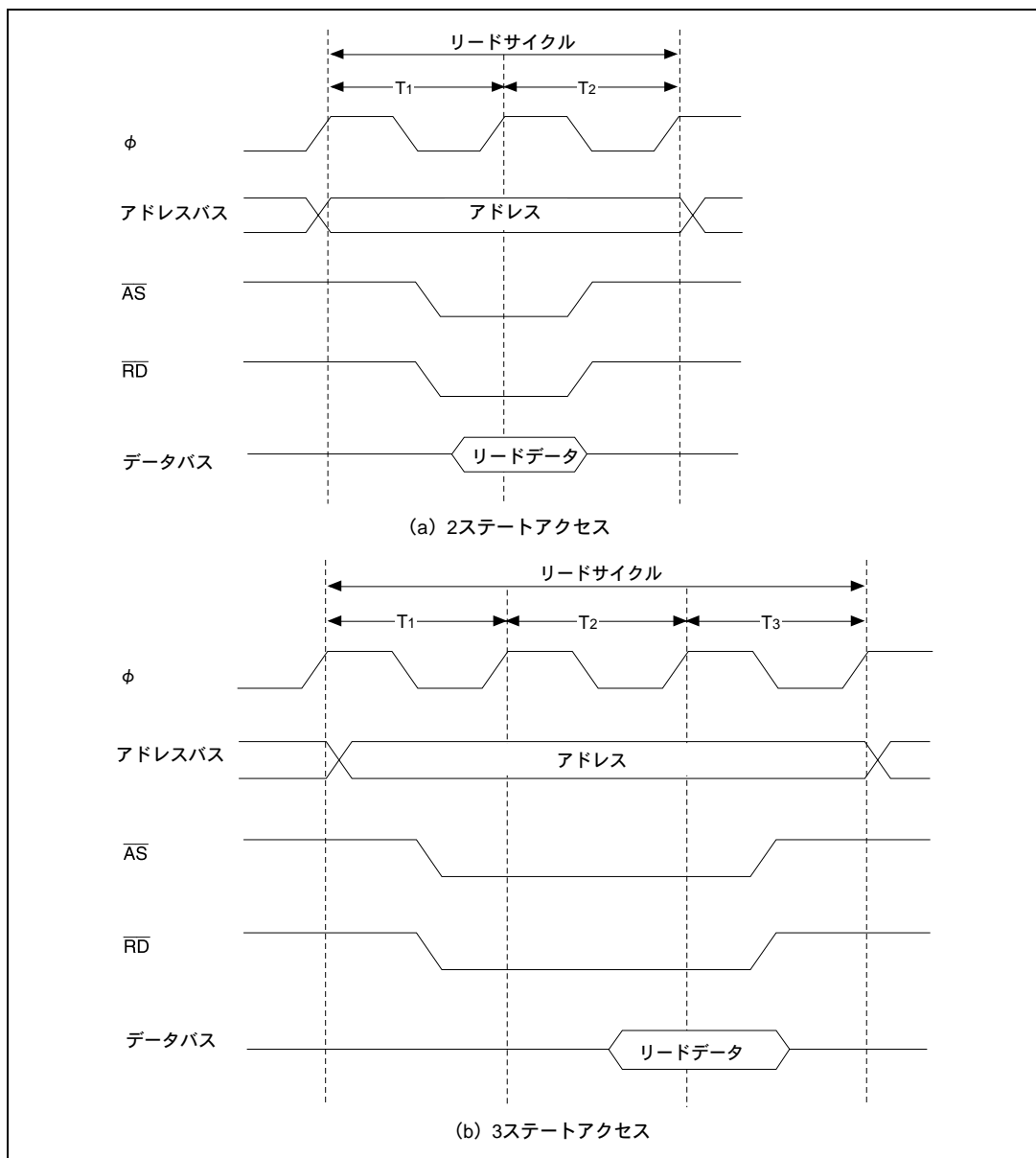


図 4.5 外部デバイスアクセスタイミング（リードタイミング）

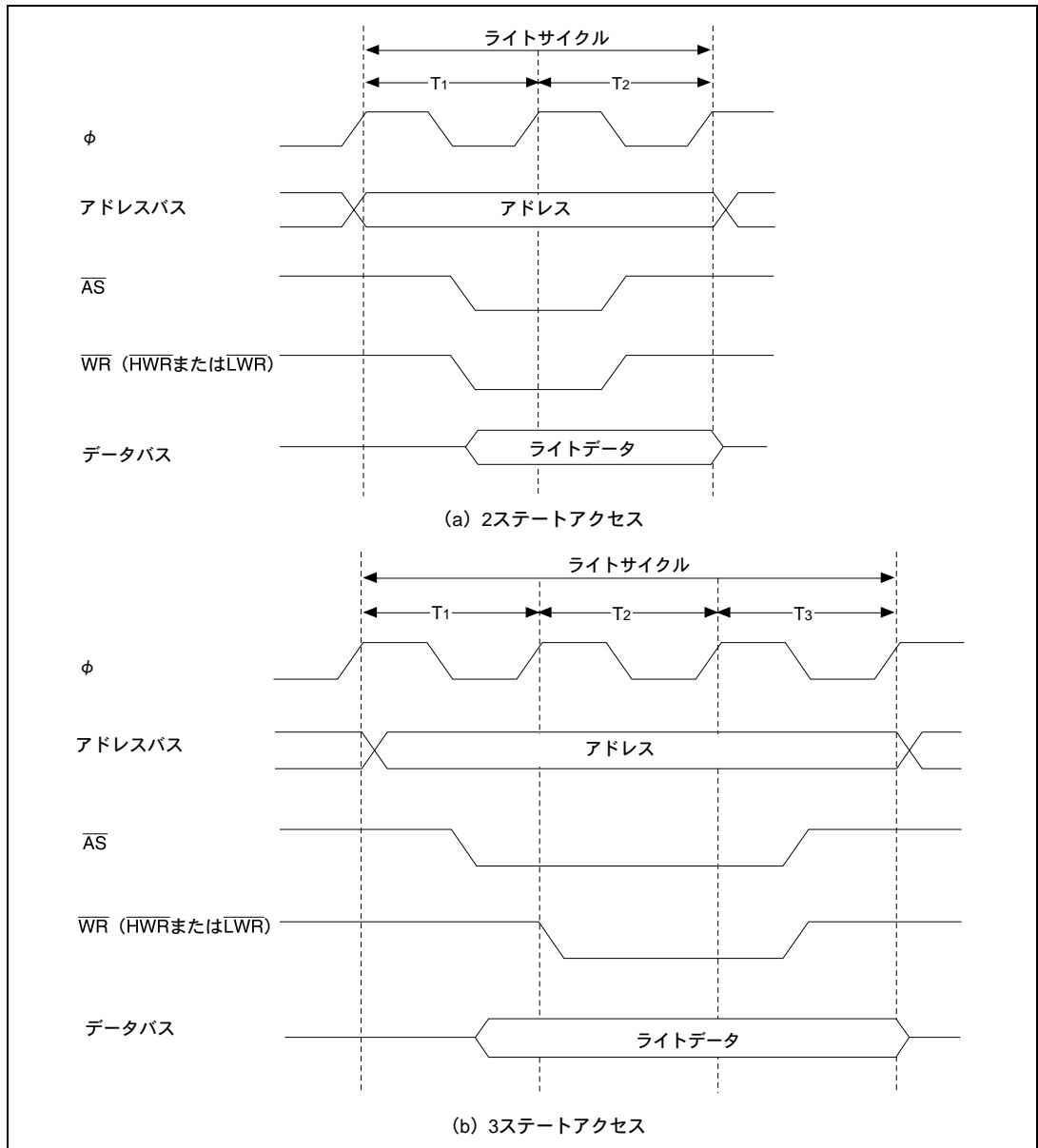


図 4.6 外部デバイスアクセスタイミング (ライトタイミング)

H8/300H シリーズ プログラミングマニュアル

発行年月 1993 年 6 月 第 1 版

1999 年 7 月 Rev. 4.00

発 行 株式会社ルネサス テクノロジ 営業企画統括部

〒100-0004 東京都千代田区大手町 2-6-2

編 集 株式会社ルネサス小平セミコン 技術ドキュメント部

©1993, 1999 Renesas Technology Corp. All rights reserved. Printed in Japan.



営業お問合せ窓口
株式会社ルネサス販売

<http://www.renesas.com>

本		支	社	〒100-0004	千代田区大手町2-6-2 (日本ビル)	(03) 5201-5350
京	浜	支	社	〒212-0058	川崎市幸区鹿島田890-12 (新川崎三井ビル)	(044) 549-1662
西	東	支	社	〒190-0023	立川市柴崎町2-2-23 (第二高島ビル2F)	(042) 524-8701
札	幌	支	店	〒060-0002	札幌市中央区北二条西4-1 (札幌三井ビル5F)	(011) 210-8717
東	北	支	社	〒980-0013	仙台市青葉区花京院1-1-20 (花京院スクエア13F)	(022) 221-1351
い	わ	支	店	〒970-8026	いわき市平小太郎町4-9 (損保ジャパンいわき第二ビル3F)	(0246) 22-3222
茨	城	支	社	〒312-0034	ひたちなか市堀口832-2 (日立システムプラザ勝田1F)	(029) 271-9411
新	潟	支	店	〒950-0087	新潟市東大通1-4-2 (新潟三井物産ビル3F)	(025) 241-4361
松	本	支	社	〒390-0815	松本市深志1-2-11 (昭和ビル7F)	(0263) 33-6622
中	部	業	本	〒460-0008	名古屋市中区栄3-13-20 (栄センタービル4F)	(052) 261-3000
浜	松	支	店	〒430-7710	浜松市板屋町111-2 (浜松アクタワー10F)	(053) 451-2131
西	部	業	本	〒541-0044	大阪市中央区伏見町4-1-1 (明治安田生命大阪御堂筋ビル)	(06) 6233-9500
北	陸	支	社	〒920-0031	金沢市広岡3-1-1 (金沢パークビル8F)	(076) 233-5980
中	国	支	社	〒730-0036	広島市中区袋町5-25 (広島袋町ビルディング8F)	(082) 244-2570
松	山	支	店	〒790-0003	松山市三番町4-4-6 (GEエジソンビル松山2号館3F)	(089) 933-9595
鳥	取	支	店	〒680-0822	鳥取市今町2-251 (日本生命鳥取駅前ビル)	(0857) 21-1915
九	州	支	社	〒812-0011	福岡市博多区博多駅前2-17-1 (ヒロカネビル本館5F)	(092) 481-7695
鹿	児	支	店	〒890-0053	鹿児島市中央町12-2 (明治安田生命鹿児島中央町ビル)	(099) 284-1748

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：カスタマサポートセンタ E-Mail: csc@renesas.com



H8/300H シリーズ プログラミングマニュアル



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

RJJ09B0141-0400O